

[fixed]

**UM SERVIDOR DE GERENCIAMENTO DE
EVENTOS PARA O COMPARTILHAMENTO DE
DADOS DE PERFIL E CONTEXTO EM
APLICAÇÕES MÓVEIS**

WALDIR RIBEIRO PIRES JUNIOR

**UM SERVIDOR DE GERENCIAMENTO DE
EVENTOS PARA O COMPARTILHAMENTO DE
DADOS DE PERFIL E CONTEXTO EM
APLICAÇÕES MÓVEIS**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

**ORIENTADOR: ANTONIO ALFREDO FERREIRA LOUREIRO,
RICARDO AUGUSTO RABELO OLIVEIRA**

Belo Horizonte
Fevereiro de 2010

© 2010, Waldir Ribeiro Pires Junior
Todos os direitos reservados.

Pires Junior, Waldir Ribeiro.

P667s Um servidor de gerenciamento de eventos para o compartilhamento de dados de perfil e contexto em aplicações móveis. / Waldir Ribeiro Pires Junior. — Belo Horizonte, 2009.
xxi,180 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Prof. Antonio Alfredo Ferreira Loureiro.
Co-orientador: Ricardo Augusto Rabelo Oliveira.

1. Computação ubíqua - Tese. 2. Sistemas baseados em eventos - Tese. 3. Redes *Publish-Subscribe* - Tese. 4. Computação ciente de contexto – Tese.
I. Orientador. II Título.

CDU 519.6*75(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO


Um servidor de gerenciamento de eventos para o compartilhamento de dados de perfil e contexto em aplicações móveis

WALDIR RIBEIRO PIRES JUNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Orientador
Departamento de Ciência da Computação - UFMG


PROF. RICARDO AUGUSTO RABELO OLIVEIRA - Co-orientador
Departamento de Computação - UFOP


PROF. MARKUS ENDLER
Departamento de Informática - PUC - RJ


PROF. MARCO TULIO DE OLIVEIRA VALENTE
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 26 de fevereiro de 2010.

*(3) Confia no Senhor e faze o bem; habita na terra e alimenta-te da verdade.
(4) Agrada-te do Senhor, e Ele satisfará os desejos do teu coração. (5) Entrega o teu
caminho ao Senhor, confia Nele, e o mais Ele fará. Salmos 37.3-5*

Agradecimentos

Agradeço primeiramente a Deus, pela oportunidade, capacidade e sabedoria que Ele tem me dado. Agradeço a Ele pela eterna e infalível promessa escrita em Salmos 37.3-5.

Agradeço também aos meus familiares, principalmente minha esposa Claudia pela compreensão e apoio em todos os momentos. Dedico esta dissertação também ao mais novo membro da família - o meu filho Matheus. Que ele possa mirar mais alto que o pai e buscar sempre o perfeito discernimento que vem de Deus. Aos meus pais, irmãos, cunhadas e sogros pelas palavras de apoio durante todo este trajeto.

Agradeço aos orientadores Prof. Loureiro e Prof. Rabelo por terem acreditado em mim, pelos exemplos, conselhos e grande amizade. Agradeço também ao Prof. Endler pela experiência adquirida no projeto *Mobilis* que foi de grande utilidade em meu trabalho. Agradeço aos outros professores do DCC que me apoiaram e me auxiliaram desde a graduação: Prof. Robson, Prof. Virgílio, Prof. Antônio Otávio, e vários outros.

Agradeço aos colegas de trabalho do Synergia pelo apoio durante estes dois anos de muito trabalho, principalmente os gerentes pela compreensão. Agradeço também ao Prof. Robson pela oportunidade de retorno ao Synergia, que possibilitou todo este trabalho.

*“Delight yourself in the Lord,
and He will give you the desires of your heart.”*
(Psalms 37:4)

Resumo

A computação ubíqua define um novo modelo computacional de interação homem-máquina cujo processamento de informações se integra a objetos do dia a dia e atividades do usuário. A utilização de um dispositivo móvel neste paradigma permite que diversos elementos computacionais e sistemas auxiliem o usuário em suas tarefas e atividades. Em alguns casos, o usuário não estará ciente da presença e atividade destes elementos computacionais no ambiente em que se encontra.

Em ambientes móveis e/ou ubíquos, informações de perfil e contexto do usuário/aplicação mudam constantemente em um determinado período. Serviços presentes no dispositivo móvel e em servidores remotos também necessitam utilizar de uma forma transparente as informações de contexto local e remoto, a fim de proverem a adaptação em serviços e atividades do usuário móvel. Estes dois fatores apresentam em sistemas ubíquos a necessidade de gerenciamento e compartilhamento destas mudanças com outros componentes e sistemas.

Este trabalho propõe a utilização de um sistema baseado em eventos para o gerenciamento de informações de perfil e contexto em serviços e aplicações móveis/ubíquas. O servidor de eventos utilizado neste trabalho é responsável por criar eventos representando as mudanças detectadas no ambiente local e remoto, publicar estes eventos e notificar consumidores interessados nestes eventos. Desta forma, o sistema permite que informações relativas à estes eventos sejam compartilhadas com serviços e aplicações de interesse.

Nos testes realizados, utilizamos dois cenários: um guia turístico eletrônico e um serviço de contexto de emergências. Em ambos cenários desenvolvidos, o servidor de eventos mostrou-se útil em disseminar mudanças nas informações de perfil e contexto entre componentes locais e remotos presentes em um ambiente ubíquo emulado. O serviço coleta estas mudanças no lado do cliente e as envia para o servidor. O servidor então processa e compartilha as informações coletadas com outros consumidores. O serviço permite também o compartilhamento de mudanças ocorridas remotamente, vindas a partir de serviços Web oferecidos à aplicações e serviços móveis.

Palavras-chaves: Computação Ubíqua, Sistemas Baseados em Eventos, Redes *Publish-Subscribe*, Computação Ciente de Contexto

Abstract

Ubiquitous computing defines a new computational model of human-machine interaction in which information processing is integrated with daily objects and activities for the user. In this new paradigm, a user can activate and participate in several tasks and activities, in some cases not even being aware of the surrounding environment. This introduces the need of context-aware computing, which proposes the capability of devices to sense changes in mobile and/or ubiquitous environments and in the user's behavior.

In these environments, user/application profile and context information and state in mobile computing systems constantly change over a period of time. Services located at mobile devices as well as in remote servers also need to effortlessly access the information generated by these changes in order to provide the adaptation of activities from the mobile user. These issues present the need for ubiquitous systems to manage and share these changes amongst other components or systems.

This work proposes an event-based system for managing context information in ubiquitous services and applications. Based on the Publish/Subscribe model, the event service used in this work creates the events representing changes in the environment, publishes these events and notifies interested consumers in these events, allowing relevant information to be shared amongst interested applications and services.

In our work, we have implemented two application prototypes: an electronic tourist guide and an emergency context service. In both prototypes implemented, the event service proved useful in disseminating changes in profile and context information between peers in a emulated ubiquitous environment. The service collects the changes in profile and context information at the client side (e.g., mobile device or Web client) and sends them to the server for processing and sharing amongst other event consumers. The service also shares the changes in profile and context information occurring remotely, by the means of Web-based services that are offered to mobile applications and services.

Keywords: Ubiquitous Computing, Event-Based Systems, *Publish-Subscribe*

Networks, Context-Aware Computing

Lista de Figuras

2.1	Os principais termos relacionados com a computação ubíqua.	15
2.2	A visão em camadas de uma aplicação ciente de contexto [41].	21
2.3	As camadas de fluxo de um sistema baseado em eventos. As entidades definidas no modelo AMQP estão associadas com os elementos definidos no sistema proposto neste trabalho.	26
2.4	A pilha de <i>software</i> definida na plataforma <i>Android</i>	34
3.1	Principais fases de execução do modelo de gestão de eventos proposto. . . .	44
3.2	Representação das relações referentes a um sistema baseado em eventos sensível a informações de contexto.	48
3.3	Funções definidas no gestor de subscrições.	49
3.4	Ilustração das filas de subscrições e eventos em um sistema baseado em eventos.	50
3.5	Algoritmo de assinatura por elementos.	52
3.6	Algoritmo de cancelamento de assinatura de elementos.	53
3.7	Algoritmo de publicação de um evento.	54
3.8	Algoritmo de notificação de eventos para um determinado consumidor. . .	55
3.9	A visão geral do sistema de serviço de notificação de eventos para aplicações ubíquas.	56
3.10	Principais componentes do servidor de eventos.	58
3.11	Visão macro das interfaces de acesso.	59
3.12	As interfaces providas pelos componentes propostos no servidor de eventos.	60
3.13	Organização do serviço de eventos.	67
3.14	O fluxo de execução do processamento de eventos ocorrendo no dispositivo e no servidor de dados.	71
3.15	Subscrição por elementos de interesse pelo usuário móvel no <i>DroidGuide</i> . .	72

4.1	A taxonomia das informações de perfil e contexto do usuário definida para a aplicação turística.	78
4.2	O mapa georeferenciado da região da Pampulha utilizado no protótipo <i>DroidGuide</i>	79
4.3	Arquitetura do protótipo <i>DroidGuide</i>	81
4.4	Telas do módulo de aplicação de <i>login</i> , menu principal e telas de mapas. . .	82
4.5	Telas do módulo de perfil (modo estendido), contexto lógico e de configuração. 83	
4.6	Telas do módulo de servidor de eventos, apresentando as mensagens de comunicação, a barra de notificações e as mensagens de notificação recebidas a partir do servidor de dados.	84
4.7	Informações de contexto sendo requisitadas do servidor de eventos pela aplicação móvel.	85
4.8	Categorização e seleção de atividades turísticas realizadas pelo escalonador de atividades.	85
4.9	Acesso e subscrição de serviços Web sensíveis em informações pela aplicação móvel.	86
4.10	Funcionalidades de seleção de atividades turísticas.	87
4.11	Funcionalidades de sugestão de atividades turísticas.	88
4.12	Algoritmo de busca por atividades em função do perfil do usuário.	89
4.13	Algoritmo de verificação da assinatura do perfil e da atividade.	90
4.14	Tabela contendo o perfil de um turista e de atividades disponíveis.	90
4.15	Os resultados da seleção de atividades para cada uma das abordagens. . .	91
4.16	Funcionalidades de seleção de tópicos para notificações de eventos.	93
4.17	Diagrama de sequência de execução do mecanismo de envio e recepção de eventos pela aplicação móvel.	95
4.18	Mapeamento dos componentes no protótipo <i>DroidGuide</i> em função da arquitetura proposta.	98
5.1	Visualização de informações sobre rotas e estatísticas pela aplicação.	102
5.2	Visualização de detalhes de marcos apresentados sobre o mapa. Em (a), a visualização dos detalhes de um marco do tipo HOSPITAL . Em (b), a visualização de um evento de emergência criado sobre uma localização. Em (c), visualização da Unidade Móvel de Emergência (UnM) em atendimento a um evento de emergência. Em (d), a UnM após entregar a emergência a um marco fixo (i.e., hospital).	103
5.3	Diagrama de estados de um evento de emergência gerenciado pelo DECS. .	105

5.4	Execução do serviço DECS durante o tratamento de um evento de emergência. Em (a), o cliente é inicializado em um navegador Web; em (b) a UnM é despachada para o evento de emergência; em (c) um zoom da área onde a UnM é despachada para uma segunda emergência; e em (d) a apresentação de dados estatísticos coletados durante a execução do cliente.	107
5.5	Arquitetura do primeiro protótipo desenvolvido para o DECS.	109
5.6	Testes de renderização do cliente do serviço DECS no <i>Android</i> e no navegador <i>Firefox</i>	110
5.7	Execução do protótipo DECS em contexto coletivo.	111
5.8	Interação entre os arcabouços GWT e GAE no protótipo DECS.	112
5.9	Arquitetura do protótipo DECS 2.0.	115
5.10	Diagrama resumido de classes do Sistema DECS relativo ao tratamento de eventos.	118
5.11	Diagrama resumido de classes do Sistema DECS relativo às ações de emergência.	119
5.12	Diagrama resumido de classes do Sistema DECS relativo aos outros tipos de ações.	120
5.13	Visualização da área de cobertura do serviço de atendimento de emergências no modo automático de execução.	122
5.14	Publicação e visualização de eventos relacionados à localização.	123
5.15	Demonstração do envio de mensagens entre usuários. Em (a) e (b), a seleção dos usuários no contexto de execução. Em (c), o envio da mensagem para um usuário específico. Em (d), a listagem de eventos apresentando um evento relacionado à mensagem enviada para o usuário específico.	124
5.16	Execução do serviço DECS no modo coletivo com três clientes acessando um servidor.	126
5.17	Final da execução do serviço DECS no modo coletivo com três clientes acessando um servidor, após o atendimento a um evento de emergência. . .	126
5.18	As principais classes no lado do servidor no serviço DECS.	128
5.19	Visualização dos dados estatísticos de cada um dos clientes no modo coletivo.	129
5.20	Mapeamento dos componentes no protótipo DECS em função da arquitetura proposta.	131
6.1	Acesso e subscrição de serviços remotos pela aplicação móvel.	134
6.2	Notificação de eventos remotos detectados no lado do servidor:	135
6.3	Uma resposta do servidor remoto de dados contendo a listagem dos serviços disponíveis (SWBIs) no servidor remoto de dados para subscrita.	137

6.4	Execução da emulação da seleção de atividades turísticas em função do perfil do usuário móvel.	140
6.5	Resultado da execução da emulação do algoritmo 4.13.	141
6.6	Comportamento da variável do número de eventos disparados.	144
6.7	Histograma do número de eventos disparados por evento de emergência tratado no DECS.	145
6.8	Comportamento da transmissão de dados no cliente.	146
6.9	Transmissão de dados no cliente em função da distância percorrida.	147
6.10	Resumo dos resultados obtidos durante a execução do serviço DECS.	148
6.11	Detalhes relativos às instâncias destacadas nos resultados coletados.	148
6.12	Informações das rotas executadas pela instancia 179.	149
6.13	Informações das rotas executadas pela instancia 292.	150
6.14	Gráfico apresentando a relação entre o número de requisições por evento e a distância.	151
6.15	Gráfico apresentando a relação entre o número de requisições por evento de emergencia e a quantidade de dados transmitidos.	151
6.16	Gráfico apresentando a relação entre o número de movimentações realizadas pelas unidades móveis de emergencia e a distância.	152
6.17	Gráfico apresentando a distribuição da distância e a duração das instâncias coletadas.	152
6.18	Dados transmitidos do lado do servidor.	154
6.19	Cobertura do Serviço DECS na área geográfica pré-definida no trabalho.	155
6.20	Resultado da cobertura do Serviço DECS.	156
6.21	A configuração utilizada nos testes de desempenho e escalabilidade.	160
6.22	Teste 1: uso de CPU no servidor.	161
6.23	Teste 2: processamento de requisições.	163
6.24	Teste 2: número de erros e uso de CPU.	164
6.25	Teste 2: transmissão de dados.	165
6.26	Teste 3: processamento de requisições e uso de CPU.	167
6.27	Teste 3: dados transmitidos.	168
6.28	Teste 3: erros e negações de quota.	169
6.29	Erro de disponibilidade de quota no <i>DataStore</i> do servidor.	170

Lista de Tabelas

2.1	Plataformas de <i>software</i> para dispositivos móveis.	32
2.2	Padrões utilizados pelo modelo de computação nas nuvens.	36
3.1	O custo computacional dos algoritmos utilizados no serviço de eventos. . .	55
3.2	As interfaces do processador de eventos e do <i>container</i> de eventos.	61
3.3	As interfaces do processador de eventos e do <i>container</i> de eventos.	61
3.4	As interfaces presentes no gestor de subscrições.	61
3.5	As interfaces da camada de serviços disponíveis no servidor de eventos. . .	62
3.6	As interfaces da camada de aplicação disponíveis no servidor de eventos. .	64
3.7	As APIs de comunicação utilizadas no protótipo <i>DroidGuide</i> na linguagem Java.	65
3.8	Categorização do modelo e do serviço de eventos.	66
3.9	Modelo de propagação e tipos de eventos suportados.	67
3.10	Recursos funcionais do serviço de eventos.	68
3.11	Recursos não funcionais do serviço de eventos.	70
3.12	Recursos não funcionais do serviço de eventos.	70
3.13	Visão geral dos protótipos desenvolvidos no trabalho.	74
4.1	Módulos que compõem o guia turístico <i>DroidGuide</i>	80
4.2	Verificação de assinaturas de atividades turísticas com os interesses do usuário ($threshold = 2$).	88
4.3	Comparativo de recursos entre o <i>JavaME</i> e o <i>Android</i>	96
4.4	Componentes da arquitetura do servidor de eventos aplicada no protótipo <i>DroidGuide</i>	97
5.1	Coordenadas da área de cobertura do serviço DECS no modo automático de execução.	121
5.2	Componentes da arquitetura do servidor de eventos utilizada no protótipo DECS.	130

6.1	Um Exemplo de classificação de perfis para usuários e atividades turísticas.	138
6.2	Classificação do perfil de turistas usado na emulação.	138
6.3	Resumo das informações coletadas a partir do protótipo DECS.	142
6.4	Parâmetros de configuração do DECS.	143
6.5	Distribuição dos eventos de emergência por tipo.	156
6.6	Compatibilidade de navegadores Web para a execução do DECS.	157
6.7	As quotas providas pela infraestrutura do servidor em nuvem.	158
6.8	Resultados do teste 1.	160
6.9	Resultados do teste 2.	162
6.10	Resultados do teste 3.	166

Sumário

Agradecimentos	ix
Resumo	xiii
Abstract	xv
Lista de Figuras	xvii
Lista de Tabelas	xxi
1 Introdução	5
1.1 Motivação	6
1.2 Objetivos	8
1.3 Contribuições	9
1.4 Organização do Texto	10
2 Fundamentos Teóricos	13
2.1 Computação Móvel e Ubíqua	13
2.1.1 História	15
2.2 Computação Ciente de Contexto	16
2.3 Definição de Termos Relacionados	17
2.3.1 O Ambiente e o Dispositivo Móvel	17
2.3.2 Evento	18
2.3.3 Contexto	19
2.3.4 Coleta e Processamento de Informações de Contexto	20
2.3.5 Serviços remotos	22
2.4 Padrões de Comunicação e Transporte de Dados	23
2.5 Arquitetura Baseada em Eventos e Serviços	24
2.5.1 Sistemas Baseados em Eventos	25
2.5.2 Taxonomia em Sistemas Baseados em Eventos	26

2.5.3	Sistemas Baseados em Eventos para Aplicações Ubíquas	28
2.6	Tecnologias em Sistemas Distribuídos e Móveis	30
2.6.1	Plataformas de <i>Software</i> e Serviços	30
2.6.2	Aplicações Móveis	31
2.6.3	Sistemas Distribuídos	35
2.6.4	Tecnologias Web para Aplicações Móveis	37
2.6.5	Serviços Móveis e Ubíquos	38
2.7	Cenários de Uso	40
2.7.1	Turismo	40
2.7.2	Emergências e Desastres	40
3	O Serviço Proposto	43
3.1	Visão Geral	43
3.2	Definição Conceitual	47
3.2.1	Algoritmos Relacionados	50
3.3	Arquitetura	56
3.3.1	Interfaces de acesso	58
3.3.2	Os Serviços Remotos	63
3.4	Comunicação entre <i>Peers</i>	63
3.5	Classificação	66
3.6	Ciclo de Vida	70
3.7	Implementações de Referência	74
4	Guia Turístico <i>DroidGuide</i>	77
4.1	Visão Geral	77
4.2	Módulos do <i>DroidGuide</i>	80
4.3	Execução da Aplicação Móvel	84
4.3.1	Seleção de Serviços Web	85
4.3.2	Seleção de Atrações Turísticas	86
4.3.3	Seleção de Tópicos ou Canais de Interesse	92
4.4	Detalhes de Implementação	92
4.4.1	Sequência de Execução	94
4.4.2	Desafios e Limitações	95
4.5	Aplicação da Arquitetura Proposta	97
4.6	Resultados obtidos	99

5	Serviço de Contexto para Emergências	101
5.1	Visão Geral	101
5.2	O Fluxo de Gerenciamento de Emergências	102
5.2.1	Protótipos Desenvolvidos	108
5.3	Arquitetura	113
5.3.1	Componentes	113
5.4	Detalhes de Implementação	115
5.4.1	Localização de Marcos Fixos, Móveis e Rotas	115
5.4.2	Ações, Tratamento de Eventos e de Retorno	116
5.4.3	Processamento de Mensagens em XML	117
5.4.4	Modos de Execução	119
5.4.5	Publicação de Eventos e Mensagens	121
5.4.6	Execução em Contexto Coletivo	125
5.5	Comparativo entre Protótipos 1.0 e 2.0	129
5.6	Aplicação da Arquitetura Proposta	130
5.7	Resultados obtidos	132
6	Resultados Obtidos	133
6.1	Guia Turístico <i>DroidGuide</i>	133
6.1.1	Assinatura de Serviços Web	133
6.1.2	Notificação de Eventos	134
6.1.3	Seleção de Atividades Turísticas em Função do Perfil do usuário	136
6.2	Serviço de Contexto para Emergências DECS	139
6.2.1	Cenários de Execução	141
6.2.2	Resultados Obtidos	142
6.2.3	Servidor de Eventos	142
6.2.4	Cobertura do Serviço DECS	154
6.2.5	Testes Em Dispositivos Reais	156
6.3	Análise de Desempenho e Escalabilidade	157
6.3.1	Teste 1: Desempenho na Execução de Operações do Servidor de Eventos	159
6.3.2	Teste 2: Escalabilidade na Publicação de Eventos	161
6.3.3	Teste 3: Escalabilidade em Clientes Concorrentes	162
7	Considerações Finais	171
7.1	Resultados	171
7.2	Trabalhos Futuros	172

7.2.1	Modelagem de Dados e Plataformas de Software	172
7.2.2	Segurança e Transmissão de Dados	173
7.2.3	Processamento de Eventos em Aplicações Móveis	173
7.2.4	Localização e Informações Georeferenciadas	174
7.2.5	Composição de Serviços Web	174
Referências Bibliográficas		177

Siglas

SWBIs *Serviços Web Baseados em Informações de Perfil e Contexto*

PDAs *Personal Digital Assistants*

HTTP *HyperText Transfer Protocol*

XMPP *Extensible Messaging and Presence Protocol*

TCP *Transmission Control Protocol*

UDP *User Datagram Protocol*

SOAP *Simple Object Access Protocol*

IIOP *Internet Inter-Orb Protocol*

XML *Extensible Markup Language*

EDA *Event Driven Architecture*

SOA *Service Oriented Architecture*

MOM *Message Oriented Middleware*

JMS *Java Message Service*

AMQP *Advanced Message Queuing Protocol*

TCP/IP *Transmission Control Protocol over Internet Protocol*

MoCA *Mobile Collaboration Architecture*

ECI *Event Communication Interface*

P2P *Peer-2-Peer*

API *Application Programming Interface*

SW *Software*

SDK *Systems Development Kit*

GSM *Global System for Mobile Communications*

WLAN *Wireless Local Access Network*

GPRS *General Packet Radio Service*

EDGE *Enhanced Data rates for GSM Evolution*

3G *Third Generation Mobile Communications Standard*

RSSFs *Redes de Sensores Sem-Fio*

SSL *Secured Socket Layer*

TLS *Transport Layer Security*

AJAX *Asynchronous JavaScript and XML*

HTML *HyperText Markup Language*

GAE *Google Web AppEngine Framework*

JavaME *Java Mobile Edition*

JavaEE *Java Enterprise Edition*

JavaSE *Java Standard Edition*

QoS *Quality of Service*

UTI *Unidade de Tratamento Intensiva*

OSI *Open Systems Interconnection*

Pub/Sub *Publish/Subscribe*

OHA *Open Handset Alliance*

OES *OpenGL for Embedded Systems*

SaaS *Software as a Service*

JS *JavaScript*

GMaps API *Google Maps API*

Google Maps *Google Maps*

GWT *Google Web Toolkit*

IHC Interação Homem-Computador

IA Inteligência Artificial

CAC *Context-Aware Computing*

CTR *Context-Topic Relation*

CE/EC *Container de Eventos ou Event Container*

PCM *Profile and Context Manager*

DECS *Droid Emergency Context Service*

SM *Subscription Manager*

HSDPA *High-Speed Downlink Packet Access*

LAN *Local Area Network*

GPS *Global Positioning System*

MIDP *Mobile Information Device Profile*

PC *Personal Computer*

B2B *Business to Business*

B2C *Business to Client*

UnM Unidade Móvel de Emergência

PoV Paciente ou Vítima

LnM Marco Fixo ou *Landmark*

EvE Evento de Emergência

SC *Statistics Container*

OBML *Opera Binary Markup Language*

JSON *JavaScript Object Notation*

WS *Web Service*

SGBD Sistema de Gerenciamento de Banco de Dados

RFID *Radio-Frequency IDentification*

GUI *Graphical User Interface*

LCDUI *Liquid Crystal Display GUI*

Capítulo 1

Introdução

A computação ubíqua [63] define um novo modelo computacional de interação homem-máquina cujo processamento de informações se integra a objetos do dia-a-dia e atividades do usuário. A utilização de um dispositivo neste paradigma permite ao usuário ativar diversos elementos computacionais e sistemas de forma simultânea e transparente durante a execução de suas atividades comuns, em alguns casos não estando ciente da presença e atividade destes no ambiente. Esta característica introduz a necessidade da computação ubíqua ciente de contexto que define a capacidade de dispositivos embarcados em detectar mudanças no ambiente e no comportamento do usuário, tais como a localização, data e hora do dia, pessoas, dispositivos e serviços próximos e atividades do usuário. Esta ciência de contexto se aplica a diversos tipos de ambientes e cenários onde o usuário está presente, levando em consideração seu estado e condição no provimento de atividades e serviços a ele de forma transparente.

A computação ubíqua e a interação centrada no usuário definem o alicerce na área de ambientes inteligentes (*ambient intelligence*). Estes locais dizem respeito a ambientes que possuem elementos computacionais que respondem à presença de pessoas. Os dispositivos trabalham de forma orquestrada a fim de apoiar os seres humanos em suas atividades e tarefas diárias naturalmente através da informação e inteligência embutida e escondida na rede conectando dispositivos. Algumas das tecnologias e sistemas relacionados incluem: (i) elementos computacionais embutidos, (ii) dispositivos que são sensíveis às mudanças no ambiente e (iii) personalização de serviços. Os elementos computacionais embutidos estão conectados em rede e integrados ao ambiente, sendo assim capazes de perceberem mudanças, podendo assim adaptar e/ou antecipar tarefas para o usuário. Os dispositivos sensíveis às mudanças no ambiente são capazes de reconhecer o usuário e seu contexto atual, detectar mudanças e prever o seu comportamento. A personalização de serviços permite que dispositivos

e serviços remotos se adaptem em função das necessidades e desejos do usuário ou da aplicação. A união destes três elementos permite o provimento de diversos serviços capazes de auxiliar o usuário ou aplicação na execução de tarefas diárias, em alguns casos de forma transparente e intuitiva.

Apesar das vantagens apresentadas acima, as aplicações móveis existentes atualmente, pelo menos em sua maioria, não estão aptas a gerenciar toda a informação proveniente de ambientes, dispositivos e até mesmo do próprio usuário. A quantidade de dados a ser tratada e o seu compartilhamento entre os diversos componentes presentes no sistema são problemas não triviais. Sendo assim, seria interessante uma solução que ajude o gerenciamento das informações provenientes de elementos computacionais e disponibilizadas para serviços e aplicações móveis em um dado ambiente. Este gerenciamento deve envolver questões tais como coleta e processamento de informações de perfil e contexto e publicação e notificação de eventos para entidades interessadas e autorizadas a recebê-las. Alguns exemplos de informações de interesse observáveis incluem informações de perfil e contexto do usuário, informações sobre o ambiente, localização geográfica, dentre outros.

O gerenciamento de mudanças nas informações de perfil e contexto em sistemas ubíquos é essencial para aplicativos móveis e serviços nesse ambiente. Entidades produtoras de informações geram eventos representando mudanças ocorridas no sistema. Por sua vez, entidades consumidoras acessam e/ou são notificadas de eventos de interesse. Assim, o gerenciamento permite que eventos gerados por entidades produtoras sejam disponibilizados para entidades consumidoras. A geração e a notificação de eventos são feitas em função de regras e interesses previamente definidos para as entidades geradoras e consumidoras.

1.1 Motivação

Na computação ubíqua, informações de perfil e contexto referentes a usuários tipicamente sofrem mudanças ao longo do tempo. Por exemplo, mudanças meteorológicas e de tráfego, localização e estados de usuários e de dispositivos. Este fato introduz um desafio no controle de mudanças em um ambiente computacional móvel distribuído. As mudanças podem ocorrer tanto no ambiente em que o dispositivo se encontra quanto globalmente em serviços remotos de interesse do usuário e aplicação. Assim, a coleta e disponibilização de informações em ambientes de computação ubíqua são duas etapas importantes de um sistema de gerenciamento, que provê aos elementos interessados e autorizados a notificação de eventos de forma adequada.

A padronização no acesso a informações de perfil e contexto por aplicações móveis e serviços locais e remotos é também outra característica de grande importância. As aplicações móveis necessitam de acesso simplificado e padronizado às informações geradas pelas mudanças de perfil e contexto. A padronização e o acesso permitem a execução de tarefas de adaptação e o uso de serviços Web baseados nas informações coletadas. Os serviços remotos baseados em informações de perfil e contexto oferecem recursos ao usuário ou dispositivo móvel em função de seu perfil, estado ou condição, tais como o acesso a informações de contexto global e o provimento de atividades em função de seu perfil ou contexto atual. Por exemplo, em situações onde o usuário esteja com fome, serviços remotos podem oferecer informações sobre uma cafeteria para um rápido lanche ou um restaurante para uma refeição mais demorada. Estes serviços remotos se beneficiam do gerenciamento de mudanças de informações de perfil e contexto através do provimento de mensagens de notificação para clientes e serviços em função da coleta e notificação de informações no ambiente local pelo dispositivo móvel. Desta forma, os serviços remotos permitem criar uma extensão das informações de contexto local do usuário, oferecendo uma visão mais abrangente de seu contexto, disponibilizando assim novos serviços antes não possíveis ou viáveis. Alguns exemplos destes serviços incluem atividades baseadas em perfil e contexto (i.e., localização, interesses ou condições), monitoramento (i.e. médico, tráfego e clima), atividades (i.e. turismo, negócios, esportes), dentre outras.

A *World Wide Web* em geral pode oferecer diversos benefícios para dispositivos móveis, já que ela permite o uso de uma quantidade significativa de serviços, a maioria já disponível para computadores pessoais. Sendo assim, os serviços já existentes seriam ou não adaptados para uso em dispositivos contendo restrições de *hardware* e *software*. Podemos citar exemplos de serviços já existentes na Web que seriam úteis no âmbito móvel tais como o acesso à rotas e mapas, *e-mail*, notícias e informações, dentre outros. Utilizando a infra-estrutura existente da Web, sistemas remotos podem disponibilizar serviços para tal, incluindo assim o fornecimento de informações de perfil e contexto do usuário para serviços e vice-versa, permitindo o uso de notificações e adaptação de atividades do usuário e aplicação móvel.

Na computação móvel ciente de contexto, a capacidade de coleta e processamento de eventos a partir de mudanças de perfil e contexto tornam-se um importante requisito. A razão disto está na necessidade do compartilhamento destas informações com entidades residentes tanto no dispositivo móvel quanto remotamente interessadas, como em serviços presentes no dispositivo e serviços Web. Este compartilhamento possibilita o auxílio destes serviços ao usuário nas suas atividades diárias através do conhecimento do perfil e contexto envolvidos, podendo ocorrer tanto localmente em

cada elemento computacional presente no ambiente quanto remotamente em servidores e serviços disponíveis na Web. A partir da coleta destas informações, eventos são utilizados para representar as mudanças coletadas e processadas pelo sistema. Este gerenciamento possibilita também a assinatura e publicação de diferentes canais de eventos gerados pelo sistema para entidades consumidores de interesse. Quando disponíveis para consumo, os eventos podem prover informações úteis para aplicações móveis e serviços remotos, tais como o estado ou condição global de uma atividade ou atração de interesse do usuário móvel, a notificação de eventos críticos como desastres naturais e emergências, acesso à informações de transporte, turismo, dentre outros.

1.2 Objetivos

Os principais objetivos deste trabalho são:

- **Proposta de um servidor de eventos:** Prover uma especificação e protótipo de um servidor de eventos para aplicações e serviços móveis que permita o gerenciamento de informações de perfil e contexto local e remoto do usuário em um ambiente móvel;
- **Disseminação de informações de perfil e contexto:** Prover o acesso à informações de perfil e contexto de usuários móveis e aplicações por entidades consumidoras em geral de uma forma padronizada e simplificada. As entidades consumidoras incluem usuários móveis e/ou serviços Web, permitindo assim que aplicações móveis e serviços distribuam, acessem e compartilhem informações de uma forma distribuída utilizando o modelo arquitetural de sistemas baseado em eventos;

Um servidor de eventos fornece uma infra-estrutura independente da aplicação que provê suporte à construção de sistemas baseados em eventos, onde geradores de eventos (i.e., entidades produtoras) publicam eventos enquanto consumidores de eventos (i.e., entidades consumidoras) efetuam assinaturas a fim de receber mensagens de notificação baseadas nos eventos subscritos. A notificação de eventos realizada pelo sistema permite gerar uma ou mais mensagens para o consumidor, sendo que para um dado evento, o serviço de eventos poderá gerar uma ou mais mensagens de notificação de acordo com as assinaturas realizadas pelos consumidores. Carzaniga et al [9] apresenta a seleção e a entrega de notificações como sendo os dois principais serviços a serem providos para consumidores de eventos interessados através do uso de uma infra-estrutura baseada em eventos. Enquanto o processo de seleção de notificações

determina a maneira como notificações se associam com as assinaturas realizadas por entidades consumidoras, a entrega de notificações fornece o mecanismo de consumo destas por seus respectivos consumidores através do roteamento de eventos em sistemas distribuídos.

O servidor de eventos (SdE) proposto neste trabalho é responsável por gerenciar eventos em sistemas móveis e ubíquos. O serviço gerencia os eventos criados a partir de mudanças de informações de perfil e contexto do usuário, da aplicação e de serviços residentes no dispositivo e no servidor remoto. Este serviço disponibiliza estes eventos gerados para consumidores em forma de notificações por mensagens. Neste caso, a notificação informa ao consumidor a ocorrência de um evento coletado pelo dispositivo móvel ou pelo servidor remoto, permitindo que este (e.g., a aplicação ou o usuário) tome alguma iniciativa de reação e/ou adaptação.

Além do gerenciamento de eventos, o servidor também provê acesso a serviços remotos sensíveis às mudanças em informações de perfil e contexto coletadas a partir do usuário e da aplicação móvel. Boa parte destas informações origina de informações do próprio usuário captadas pelo perfil e contexto como localização, estado (movimento), interesses, dentre outros. Alguns exemplos de serviços Web disponibilizados pelo servidor de eventos incluem, por exemplo, provedores de informações de tráfego, clima, comércio, turismo, informações médicas e de emergências. No caso deste trabalho, entidades provedoras de serviços são responsáveis em disponibilizar estes serviços ao usuário móvel através de um único ponto de cadastro e acesso no servidor para uso em aplicações móveis. Assumimos neste trabalho que estes serviços já estão disponíveis para uso, sendo já acessíveis por aplicações móveis e provendo recursos para o usuário e para a aplicação.

1.3 Contribuições

As contribuições deste trabalho são:

- **Especificação e protótipos:** Provimento de uma especificação e protótipos do servidor de eventos (SdE) para a captação de mudanças de informações de perfil e contexto do dispositivo móvel (e.g., usuário e/ou aplicação) e também de serviços Web para atividades de adaptação sobre o ambiente e/ou nas tarefas do usuário;
- **Publicação de informações de perfil e contexto:** Provimento e utilização de informações de perfil e contexto do usuário móvel para serviços Web e para a notificação de eventos no cliente e no servidor;

- **Desenvolvimento e avaliação de protótipos:** Avaliação quantitativa e qualitativa de protótipos desenvolvidos quanto a viabilidade destes no processamento de eventos, notificação de mensagens, fornecimento e utilização de serviços Web baseados em informações de perfil e contexto no dispositivo (e.g., usuário e aplicação móvel) no que diz respeito às tecnologias existentes em sistemas móveis;
- **Utilização de tecnologias Web em Sistemas Ubíquos ou Pervasivos:** Avaliação qualitativa e quantitativa da viabilidade no provimento de serviços Web através do modelo computacional em nuvem para aplicações móveis e a utilização de tecnologias Web na construção de aplicações móveis.

1.4 Organização do Texto

Este documento está organizado da seguinte forma. O capítulo dois apresenta os principais fundamentos teóricos que se relacionam com este trabalho, que incluem as principais áreas da Ciência da Computação, a definição dos principais termos e tecnologias utilizadas em sistemas móveis e distribuídos.

O capítulo três apresenta a especificação do servidor de eventos (SdE) proposto, descrevendo uma definição conceitual, visão geral do serviço, algoritmos relacionados, componentes que o compõem e o processo de gerenciamento de eventos.

O capítulo quatro descreve o primeiro protótipo desenvolvido neste trabalho de uma aplicação móvel que utiliza os recursos do SdE. O guia Turístico *DroidGuide* é responsável por prover serviços Web para turistas sobre uma determinada região geográfica, tais como seleção de atividades turísticas, gestão de informações turísticas de perfil e contexto e serviços Web.

O capítulo cinco apresenta o segundo protótipo desenvolvido neste trabalho com o mesmo objetivo de avaliar quantitativamente o servidor de eventos. O protótipo *Droid Emergency Context Service* (DECS) tem como objetivo oferecer um serviço de gerenciamento de eventos de emergência sobre uma região geográfica, abrangindo a detecção do evento de emergência, o seu tratamento por uma UnM até a sua chegada ao seu destino (e.g., hospital, delegacia ou corpo de bombeiros).

O capítulo seis apresenta os resultados obtidos a partir dos dois protótipos desenvolvidos neste trabalho. O Guia Turístico *DroidGuide* demonstra a viabilidade na notificação de eventos, subscrição de serviços e a publicação de eventos em dispositivos móveis e no servidor remoto de dados. O DECS apresenta resultados quantitativos no que diz respeito à transmissão de dados, tais como o número de eventos publicados,

distância percorrida, tempo gasto em eventos de emergência, dentre outros. Na avaliação do DECS, realizamos a transmissão de dados entre clientes Web e o servidor de dados remoto, onde o servidor de eventos está localizado.

O capítulo sete apresenta as considerações finais do trabalho, que incluem a apresentação dos objetivos alcançados e possíveis trabalhos futuros em diversas áreas, tais como a segurança e a compressão de dados, a composição de serviços Web, dentre outros.

Capítulo 2

Fundamentos Teóricos

Este capítulo apresenta os fundamentos teóricos relacionados à este trabalho. Apresentamos neste capítulo os tópicos fundamentais a fim de esclarecer suposições do trabalho, as áreas da Ciência da Computação e como elas se relacionam com o trabalho, os trabalhos pesquisa relacionados já realizados e as tecnologias nas áreas de computação móvel e ubíqua aplicáveis na construção de sistemas distribuídos e móveis/ubíquos.

2.1 Computação Móvel e Ubíqua

A computação móvel, de uma forma geral, apresenta a habilidade de uso de tecnologias por usuários enquanto estes se movem, permitindo o acesso a informações e serviços de qualquer lugar e momento. Ela também define a capacidade de execução de aplicações em dispositivos de pequeno porte, tais como celulares, *Smartphones*, *Palms*, *Internet Tablets* e *Personal Digital Assistants* (PDAs). A comunicação nestes tipos de dispositivos é normalmente provida na forma sem fio, onde usuários utilizam serviços móveis de dados e voz. Além da capacidade de execução de aplicações móveis, limitações tais como a largura de banda na transmissão de dados, a intermitência no canal de comunicação, o consumo de energia e limitações na interface de usuário fazem parte das restrições da computação móvel.

A computação ubíqua, por sua vez, se beneficia de dispositivos móveis para uma melhor realização de atividades para o usuário, fazendo com que ele não necessite sempre estar explicitamente inserido no mundo computacional para utilizar os recursos presentes. Sendo assim, a computação ubíqua se integra ao mundo do próprio usuário, definindo um novo modelo computacional de interação homem-computador no qual o processamento de informações está integrado a objetos e atividades do dia a dia.

Ao contrário do paradigma da computação *desktop* onde um usuário conscientemente aciona um único dispositivo para a execução de uma ou mais atividades específicas, a computação ubíqua permite que o usuário acione diversos dispositivos computacionais e sistemas de forma simultânea e inconsciente durante a execução de suas atividades e tarefas, em alguns casos sem estar ciente da presença destes e de suas atividades no ambiente.

No meio acadêmico, pesquisadores e instituições de pesquisa utilizam termos diferenciados para representar o modelo computacional e alguns dos conceitos propostos pela computação ubíqua. Alguns destes termos incluem, como por exemplo, a "computação pervasiva" e a inteligência de ambientes ou *Ambient Technology*. A computação "pervasiva" propõe um modelo computacional onde dispositivos presentes em diversos locais em um ambiente são utilizados na execução de tarefas para o usuário de forma transparente e autônoma. Greenfield [23] apresenta o termo *everyware* como sendo relacionado ao modelo apresentado acima, onde toda a informação acessível através de telefones celulares ou navegadores Web se tornará acessível em qualquer local, a qualquer momento e entregue da forma mais apropriada em função da localização e contexto do usuário. A inteligência de ambientes se refere à ambientes eletrônicos que possuem a sensibilidade e responsividade sobre a presença de elementos (i.e., usuários, aplicação, dispositivos). Através de dispositivos com capacidade de trabalho colaborativo, um ambiente proverá apoio a seus usuários em suas atividades diárias de forma simples e natural. Através do uso de informações, inteligência embutida e conectividade de rede em dispositivos embarcados, o provimento de serviços para o usuário em um determinado ambiente torna-se viável e interessante.

Starner [50] apresenta o termo *Wearable Computing* como sendo um outro conceito relacionado à computação ubíqua, englobando pesquisas em diversas áreas relacionadas. Alguns destes incluem a Interação Homem-Computador (IHC) que apresenta novos projetos de interfaces de usuário e realidade aumentada, reconhecimento de padrões, uso de Redes de Sensores Sem-Fio (RSSFs) ou dispositivos como *Radio-Frequency IDentification* (RFID), uso de equipamentos *wearables* para aplicações ou habilidades específicas e tecidos eletrônicos. Outros temas são comuns a outras áreas da computação ubíqua como a computação móvel e pervasiva, inteligência de ambientes, gerenciamento de energia e de dissipação de calor, arquiteturas de *software* distribuídas e móveis e redes sem-fio pessoais (em inglês, *Personal Area Networks* (PAN)). No computador "vestível", temos uma interação constante entre o computador e o usuário, sem a necessidade de, por exemplo, ligar e desligar o(s) dispositivo(s) presentes. Estes dispositivos são "aumentados" ou inseridos fisicamente em todas as ações realizadas pelo usuário, tornando assim uma extensão da mente



Figura 2.1. Os principais termos relacionados com a computação ubíqua.

ou corpo do usuário. Outros termos relacionados incluem tecnologia que desaparece ou *disappearing technology/hardware* apresentado em Want et al. [61], computação centrada no usuário, adaptação pervasiva e dispositivos inteligentes. A relação dos principais termos relacionados com a computação ubíqua pode ser visualizada na Figura 2.1.

2.1.1 História

O cientista Mark Weiser apresentou o termo "computação ubíqua" [63] pela primeira vez por volta de 1998, quando trabalhava como pesquisador chefe no centro de pesquisas¹ da Xerox em Palo Alto, CA. Alguns dos primeiros artigos na área foram escritos por ele e outros pesquisadores do centro tais como o diretor do PARC e cientista chefe John Seely Brown. Weiser [63] propôs três formas básicas de dispositivos para sistemas ubíquos (i.e., dispositivos inteligentes ou *smart devices*): abas (*tabs*), tábuas (*pads*) e quadros (*boards*). Dispositivos utilizados atualmente, tais como os celulares, tocadores de

¹Xerox Palo Alto Research Center (PARC)

audio/vídeo portáteis, marcadores de RFIDs, navegadores *Global Positioning System* (GPS) e quadros interativos se baseiam nas três formas básicas propostas por Weiser.

Um dos primeiros sistemas ubíquos propostos foi o chamado "Fio Vivo" ou "*Live Wire*", também conhecido como o "fio pendurado" ou "*Dangling String*" [64]. Este dispositivo foi instalado no centro de pesquisas da XEROX (PARC), responsável por informar a atividade ou indicação de tráfego de rede através de um pequeno motor controlado por uma conexão de rede. Este exemplo foi denominado por Mark Weiser de tecnologia calma. Outros dispositivos foram concebidos, tais como o *Nabaztag* [59], cartão ativo ou *Active Badge* [62], o *Media Cup* [25] e a secretária eletrônica de bolinha de gude ou *Marble Answering Machine* [5].

2.2 Computação Ciente de Contexto

A computação ubíqua utiliza informações do ambiente para a definição do contexto e adaptação em sistemas em tempo real. Dey et. al [21] apresenta a computação ciente de contexto como sendo um paradigma no qual aplicações possuem a capacidade de descobrir e utilizar informações de contexto, tais como a localização, horário do dia, pessoas e dispositivos próximos, e atividades do usuário. Rossi et al. [45] apresenta a adaptação como sendo a capacidade de um sistema computacional ou um *middleware* em modificar seu comportamento em resposta às mudanças no contexto ambiental. Sendo assim, aplicações móveis e serviços remotos podem utilizar informações presentes no contexto para o provimento de serviços e conteúdo tanto para o usuário quanto para aplicações móveis presentes no dispositivo.

A ciência de contexto lida diretamente com as características e informações obtidas do ambiente, permitindo que sistemas computacionais reajam ou adaptem às mudanças ocorridas em função de usuários ou elementos externos. Estes sistemas [49] estão principalmente preocupados com a aquisição de contexto através, por exemplo, da utilização de sensores para a percepção de uma condição, a abstração e compreensão do contexto (i.e., associando um estímulo sensorial percebido a um contexto), e o comportamento da aplicação baseado no contexto reconhecido, como por exemplo, habilitar atividades específicas para o usuário em função das informações coletadas e inferidas.

Suponhamos um serviço de contexto para emergências com o objetivo de gerenciar o atendimento a chamadas de emergências em uma cidade ou município. No instante em que uma emergência ocorre, é possível coletarmos informações de forma quase instantânea tais como a localização e o tipo de evento a ser tratado (i.e., saúde, segurança ou incêndio). A partir da localização e o tipo informados, o serviço seleciona

uma unidade móvel de atendimento mais próxima que esteja disponível, levando em consideração fatores contextuais tais como a distância, o tempo estimado para a chegada ao local da emergência, condições de clima (i.e., chuva, neve, etc.) e tráfego. Na chegada ao local, a unidade móvel acrescenta informações de contexto ao evento sendo tratado pelo serviço, tais como informações adicionais sobre a condição da vítima, necessidades específicas que devem ser consideradas na chegada ao destino respectivo (i.e., hospital, delegacia ou corpo de bombeiros). Logo após o posicionamento da vítima à unidade móvel (quando aplicável), o serviço seleciona o destino apropriado mais próximo para um melhor tratamento do evento, podendo levar em consideração fatores como a distância, tempo de trajeto, condições de tráfego e da vítima (i.e., necessidades urgentes, vagas na Unidade de Tratamento Intensiva (UTI), etc.). Na chegada da unidade móvel ao seu destino, médicos presentes no pronto-atendimento terão acesso às informações coletadas da vítima durante o ciclo de atendimento, mais especificamente as condições do acidente, tempo ocorrido, dentre outras.

Aplicações móveis cientes de contexto e perfil de usuários têm se tornado um dos principais passos na evolução da computação móvel e ubíqua. A computação ciente de contexto possibilita o uso de informações de ambientes e do próprio usuário para o provimento da adaptação em aplicações residentes nestes dispositivos. Esta adaptação é exigida por sistemas sensíveis ao contexto em situações onde ocorrem mudanças no comportamento da aplicação e do usuário. Alguns exemplos de mudanças aplicáveis incluem o temperamento do usuário móvel, horário do dia, local, a conectividade e acesso a determinados tipos de redes de dados e voz, limitações na quantidade de energia disponível no dispositivo durante o acesso, visualização de conteúdo (i.e., áudio, vídeo, texto) pelo usuário, dentre outros. Através da utilização destas informações, diversos serviços e atividades relacionadas ao contexto do usuário se tornam possíveis e em alguns casos em tempo real.

2.3 Definição de Termos Relacionados

Esta seção apresenta a definição dos principais termos utilizados na definição do problema e como estes estão contextualizados no trabalho.

2.3.1 O Ambiente e o Dispositivo Móvel

O dicionário Merriam-Webster² define o ambiente como sendo as circunstâncias, objetos ou condições na qual uma entidade (e.g., uma pessoa e/ou objeto) está presente. O

²<http://www.merriam-webster.com/>

ambiente representa o local onde o usuário se encontra, podendo ser interno (i.e., um quarto, casa, prédio e aeroporto) ou externo (i.e., rua, bairro, distrito, cidade ou país). Nestes ambientes, dispositivos móveis podem usufruir de uma diversa quantidade de informações de contexto para o provimento de serviços, possibilitando assim a computação ciente de contexto e a adaptação em aplicações móveis residentes.

Na computação ubíqua, os ambientes possuem elementos computacionais visíveis ou não a fim de prover informações e serviços aos usuários presentes. Estes elementos possuem uma capacidade de processamento de dados e informações com dimensões de bolso, como por exemplo, os sensores e dispositivos móveis. Estes dispositivos podem possuir ou não uma tela e capacidade de entrada de dados através, por exemplo, de um teclado em miniatura. Alguns dispositivos móveis mais potentes fornecem a assistência e conveniência de um computador portátil em ambientes onde a utilização de computadores pessoais não é possível ou limitada. Dispositivos móveis, em geral, possuem a capacidade de se conectar a redes sem fio, obtendo desta forma acesso à Internet e a outros sistemas computacionais no escritório, na residência e em ambientes externos.

2.3.2 Evento

O evento representa o resultado de um acontecimento onde ocorre uma mudança de estado de um determinado elemento computacional. Semelhantemente, Mühl et al. [37] define o evento como sendo um acontecimento de interesse observável por um elemento computacional, podendo este ser um computador, um sensor, ou um dispositivo qualquer. Chandy [11] define um evento de uma forma mais ampla como sendo uma mudança de estado detectável por uma entidade no sistema (e.g., consumidores e/ou produtores). Eles são responsáveis por duas operações: (a) criar instâncias de eventos representando as mudanças nos quais outros elementos podem responder e (b) reagir ou adaptar a mudanças através de regras impostas no nível de aplicação. Estas regras definem intervalos de valores aceitáveis (e.g., *threshold*) para um determinado comportamento observado.

Consideramos neste trabalho dois tipos de eventos: a) os internos que ocorrem no âmbito do dispositivo móvel e b) os externos que ocorrem tanto no âmbito do ambiente em que o dispositivo se encontra quanto em serviços disponíveis na Web. Componentes de *hardware* e *software* que coletam informações de contexto do usuário, do dispositivo e/ou aplicação ou de serviços remotos produzem eventos em função de mudanças nestas informações. Isto permite que o *software* localizado no dispositivo móvel execute ações ou comandos de adaptação em decorrência de mudanças no meio. Desta forma, o

sistema determina seu fluxo de execução a partir de informações coletadas a partir de sensores, ações do usuário ou mensagens de outras aplicações e serviços. O evento permite que informações de contexto sejam correlacionadas às atividades e serviços locais e externos, permitindo assim a chamada e consumo de serviços por aplicações presentes no dispositivo móvel em função do contexto apresentado.

Sistemas baseados em eventos em geral processam eventos em forma de notificações através de uma filtragem ou correlação. A notificação tem como objetivo descrever os atributos associados a um evento, de tal forma que um determinado componente consumidor possa acessá-los e executar atividades para o usuário. O processo de filtragem permite que o sistema forneça determinados tipos de eventos para consumidores a partir de uma lista ou fila ou baseada em tópicos ou canais, permitindo o provimento de eventos de acordo com interesses de consumidores. Esta filtragem depende diretamente da correlação entre eventos e seus filtros, onde cada filtro determina os valores válidos de filtragem para os eventos existentes. O sistema seleciona e entrega aqueles que atendem às restrições de atributos definidas no filtro aos consumidores interessados. Em geral, a subscrição em eventos utiliza a seleção e entrega das notificações associadas através das diversas formas de filtragem, tais como a subscrição por tópicos ou temas, por tipos ou classificações, dentre outras.

2.3.3 Contexto

Dey et al. [21] formalizam a definição de contexto como sendo qualquer informação que possa ser utilizada para caracterizar uma situação de entidades (e.g., pessoa, lugar ou objeto) consideradas relevantes para interação entre um usuário e uma aplicação. O Dicionário Merriam-Webster define contexto de uma forma ainda mais genérica sendo as condições correlacionadas na qual algo existe ou acontece. Na computação, o contexto correlaciona as situações e mudanças no ambiente com aplicações e serviços em execução sobre este ambiente. Todo processo de gestão de contexto apresenta uma sensibilidade que procura lidar com estas mudanças através da coleta de informações a partir de sistemas computacionais acessíveis tais como sensores, atuadores, aplicativos e serviços. A partir da coleta de informações de contexto, dispositivos móveis podem reagir em função das mudanças detectadas, notificando assim o usuário móvel das mudanças.

O termo "ciente de contexto" apresentado por Schilit et al. [48] define as formas de representação das mudanças e ações em função do contexto para aplicações móveis. Isto demonstra a importância do conhecimento do contexto e monitoramento de sua mudança para que aplicações possam tanto usufruir de serviços sensíveis ao tal quanto adaptarem a estas mudanças. Chen et al. [13] acrescenta duas classificações de

aplicações cientes de contexto: ativa e passiva. Na computação ciente de contexto ativa, aplicações são capazes de se adaptar automaticamente às informações de contexto descobertas pelo seu comportamento. Diferentemente da ativa, as aplicações na passiva apresentam a informação recente de contexto coletada ao usuário para este possa acessá-la em um momento futuro e oportuno.

Chen et al. [13] apresenta o contexto como sendo uma divisão em quatro categorias: (a) o contexto computacional, (b) contexto do usuário, (c) contexto físico e (d) contexto do tempo. O contexto computacional apresenta informações de conectividade de rede, custo de transmissão, largura de banda e recursos próximos disponíveis tais como impressoras, telas e computadores. O contexto do usuário define o perfil, localização e pessoas próximas e em alguns casos a situação atual do usuário no âmbito psicológico e social. O contexto físico apresenta informações do ambiente em volta do usuário, tais como a iluminação, níveis de ruído, temperatura, informações de tráfego, objetos próximos e orientação. O contexto do tempo apresenta detalhes temporais do usuário tais como a hora do dia, semana, mês e estações do ano. Todas estas categorias unificadas definem o escopo de informações de contexto que sistemas móveis podem armazenar do usuário.

2.3.4 Coleta e Processamento de Informações de Contexto

No que diz respeito a sistemas de coleta e processamento de informações de perfil e contexto, podemos citar algumas propostas tais como em Chen et al. [14] para suporte em ambientes inteligentes. Dey [20] apresenta processos de suporte no desenvolvimento de aplicações móveis cientes de contexto. Patrick et al. [41] apresenta um estudo amplo de *middlewares* com suporte a informações de perfil e contexto para sistemas móveis. Uma composição em camadas de uma aplicação ciente de contexto é apresentada na Figura 2.2, onde nesta visão, é possível visualizar os tipos de informações gerenciáveis em sistemas cientes de contexto em diferentes níveis, tais como localização, recursos, pessoas próximas, distância entre o usuário e pessoas ou recursos, dentre outros.

Classificamos neste trabalho as atividades de coleta e processamento de informações de contexto em aplicações ubíquas em dois tipos: (a) informações de contexto local ou individual e (b) informações de contexto global. Em ambos os casos, atividades são associadas aos eventos gerados de tal forma que o dispositivo ou aplicação tenha a capacidade de efetuar a notificação e, quando aplicável, a adaptação apropriada em decorrência das mudanças no ambiente. O primeiro tipo de coleta utiliza dados de contexto coletados pela aplicação móvel para prover a geração de eventos quando mudanças neste âmbito forem detectadas. Algumas situações de mudança no contexto

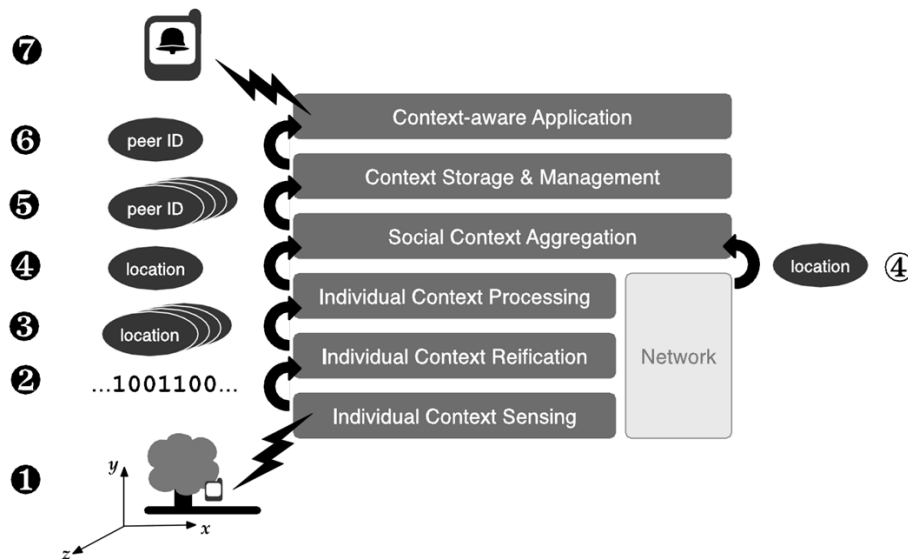


Figura 2.2. A visão em camadas de uma aplicação ciente de contexto [41].

do usuário móvel incluem a localização do dispositivo, nível de energia, temperatura local, movimentação (e.g., via acelerômetro), estado da conectividade, dentre outros.

O segundo tipo de informações de contexto propõe uma coleta e processamento de informações na detecção de mudanças do contexto no âmbito global ou externo. O sistema utiliza informações de serviços Web acessíveis pela aplicação a partir de mudanças decorrentes no dispositivo e também nos próprios serviços e/ou contexto global. Como exemplo, podemos citar uma aplicação turística que utiliza um servidor de eventos para o processamento de informações de localização do usuário. Baseada nestas informações, o servidor de eventos oferece serviços sensíveis à localização, tais como informações de tráfego e clima durante o trajeto e as condições presentes no destino definido pelo turista. Neste caso, o servidor remoto processa as informações de contexto e perfil remotos e envia eventos tanto para serviços remotos quanto para o dispositivo móvel.

O contexto externo utiliza informações de serviços na Web providos por entidades externas, porém concentradas em um único ponto de acesso no servidor por dispositivos neste trabalho. Estes serviços são responsáveis em prover acesso à informações e atividades para o usuário do dispositivo móvel a partir de informações de perfil e contexto coletadas tanto localmente pelo dispositivo móvel quanto no âmbito global por serviços de informações relacionados. Neste trabalho, assume-se que estes serviços estão disponíveis para uso pelo servidor de eventos e, conseqüentemente, por aplicações

e serviços móveis. Definimos também neste trabalho os tópicos ou canais de interesse a serem subscritos por estes serviços a fim de avaliarmos o funcionamento do servidor de eventos proposto neste trabalho.

2.3.5 Serviços remotos

Os serviços remotos tem como objetivo oferecer recursos ao usuário móvel tais como o acesso a informações de perfil e contexto remoto e o consumo de eventos oriundos de mudanças de dados de perfil e contexto do dispositivo móvel. Um *Web Service* (WS) pode ser definido como uma aplicação acessível onde outras aplicações e/ou seres humanos podem descobrir e invocar, como por exemplo, a partir de dispositivos móveis ou celulares (JSR 172 [52]). Maamar et al. [34] apresenta um serviço Web com as seguintes propriedades: (a) independência (a maior possível) de plataformas específicas e paradigmas computacionais; (b) interorganizacional e (c) facilmente construídos a partir de padrões como o *Extensible Markup Language* (XML) e o *Simple Object Access Protocol* (SOAP) [65]. Alguns trabalhos existentes já oferecem a possibilidade de integração destes serviços e recursos com aplicações residentes em dispositivos móveis, como em Christensen [18], em Chakraborty et al. [10] e em Heejung et al. [12]. Existem, inclusive, alguns trabalhos levando em consideração a ciência de contexto como em Debaty [19], onde é proposto uma aplicação que permite acesso à informações móveis e sensíveis ao contexto em uma variedade de ambientes computacionais ubíquos.

As tecnologias de Serviços Web têm sido consideradas como soluções promissoras em ambientes computacionais ubíquos. A principal razão disto está no fato destes serviços utilizarem padrões como o XML, o SOAP, provendo a interoperabilidade entre serviços e entre clientes, e sua alta capacidade de integração em processos de negócio (e.g., *Business to Business* (B2B) e *Business to Client* (B2C)). Diversos serviços Web já estão disponíveis para uso por aplicações, e serviços adicionais direcionados em informações de perfil e contexto podem ser facilmente criados com o objetivo de prover suporte a aplicações em execução nos diversos tipos de dispositivos móveis existentes. Podemos destacar algumas propostas de *workflows* de serviços Web para ambientes ubíquos que consideram mudanças em informações de perfil e contexto, tais como em Joohyun et al. [31] e em Cho et al. [17].

2.3.5.1 A Composição de Serviços Web

Os serviços disponíveis em ambientes ubíquos necessitam também prover de uma forma automática a adaptação para usuários móveis de acordo com as informações dinâmicas de perfil e contexto que podem ser obtidas tanto do usuário quanto do ambiente em que

ele está. Em alguns casos, a composição de serviços torna-se possível e útil, onde uma combinação de serviços Web semelhantes e/ou complementares seria disponibilizada ao usuário. O principal objetivo da composição está na satisfação das necessidades do usuário através da combinação de serviços existentes, em situações onde não há um serviço disponível no ambiente capaz de realizar uma determinada funcionalidade. Urbietta et al. [57] destaca a composição de serviços em ambientes inteligentes e dinâmicos com o foco nas seguintes características: especificação (qualidade, uso de recursos, etc.), execução (contingência e escalabilidade) e disponibilidade (topologia, infra-estrutura, etc.), usabilidade, adaptabilidade e eficiência no seu uso. Bronsted et al. [6] apresenta a composição de serviços como sendo uma composição de quatro principais áreas: ciência de contexto, gerenciamento de contingências, gerenciamento da heterogeneidade de dispositivos e prover recursos aos usuários móveis.

2.4 Padrões de Comunicação e Transporte de Dados

Os protocolos de comunicação e transporte de dados oferecem mecanismos para o envio de dados entre componentes distribuídos em um dado sistema. Os protocolos de comunicação de dados definem regras padronizadas na representação de dados, sinalização, autenticação e detecção de erros necessários no envio de informações sobre um canal de comunicações. Os protocolos no nível de aplicação dependem de outros protocolos em camadas inferiores para o transporte destes dados sobre a rede, como por exemplo, o *HyperText Transfer Protocol* (HTTP) e o *Extensible Messaging and Presence Protocol* (XMPP) [27]. As camadas inferiores do modelo *Open Systems Interconnection* (OSI) definem protocolos no nível de camada de transporte, Internet, enlace e física, provendo assim diversos serviços para camadas superiores.

No que diz respeito à utilização de protocolos de comunicação para sistemas distribuídos, Baldoni et al. [3] recomendam a adoção de protocolos padrões como o *Transmission Control Protocol* (TCP) ou *User Datagram Protocol* (UDP), ou protocolos de *middleware* baseados em TCP como o SOAP ou o *Internet Inter-Orb Protocol* (IIOP) na comunicação entre componentes distribuídos. A padronização na comunicação entre componentes presentes em sistemas distribuídos é de extrema importância devido à necessidade de interação entre clientes e serviços heterogêneos. Desta forma, a utilização de protocolos padronizados beneficia a integração entre os diversos tipos de componentes, sendo estes móveis ou fixos, locais ou remotos.

Em destaque na representação de dados na comunicação entre componentes, o XML define um importante papel na padronização das mensagens trocadas entre consumidores e produtores em um sistema baseado em eventos. Sua adoção na

representação de estruturas de dados arbitrárias na comunicação de dados pela Web (e.g., *HyperText Markup Language* (HTML) e *SOAP/Web Services*) tem sido largamente aplicada devido à sua simplicidade no envio e processamento das informações representadas. Outras vantagens incluem a flexibilidade no uso de marcadores ou *tags*, a padronização, independência de plataforma (e.g., imune a mudanças de tecnologia) e a capacidade de envio de documentos neste formato sobre o protocolo HTTP e através de mecanismos de segurança tais como *firewalls*.

2.5 Arquitetura Baseada em Eventos e Serviços

A arquitetura baseada em eventos ou *Event Driven Architecture* (EDA) [11] define um padrão de arquitetura de sistemas focado na produção, detecção, consumo e reação a eventos em decorrer de mudanças de estados conforme apresentado na seção 2.3.2. Esta operação de consumo e reação à eventos pode resultar em atividades executadas pelo usuário ou pelo próprio dispositivo. Podemos citar, por exemplo, cenários em uma aplicação móvel turística onde o usuário móvel inicia suas atividades de visitação a partir de uma lista de atrações de um local. Durante a sua locomoção e visita em atrações, seu estado poderá alterar continuamente, como por exemplo, "em movimento"(e.g. entre atrações), "na atração", "cansado", "com fome", "concluído". Neste caso, o guia turístico trata cada uma das mudanças de estado na forma de eventos produzidos e publicados por componentes e aplicações presentes no sistema. Podemos exemplificar também um serviço móvel de auxílio ao turista interessado em visitar atrações turísticas em uma determinada cidade. A partir do conhecimento de seu perfil (e.g., interesses, idade, estado civil), o serviço é capaz de sugerir atividades turísticas que melhor se encaixam no perfil definido pelo turista. Outras possibilidades incluem o agrupamento de turistas a partir de interesses mútuos, o provimento de serviços em tempo real em função de condições definidas no contexto do usuário (e.g., fome, sono, com pressa, dentre outros) e a resolução de conflitos entre grupos turísticos.

No que diz respeito à construção de sistemas baseados em eventos, o modelo arquitetônico utilizado tende para duas alternativas: o orientado a eventos e a serviços. Entretanto, estas alternativas possuem algumas características em comum, tais como modularização, baixo acoplamento e a alta adaptabilidade. Isto demonstra a integração fácil e natural de ambas alternativas na modelagem e desenvolvimento de sistemas distribuídos baseados em eventos. Enquanto o sistema eda fornece a publicação e notificação de eventos, o sistema baseado em *Service Oriented Architecture* (SOA) permite o fornecimento de acesso à serviços remotos para aplicações móveis. A arquitetura orientada a serviços permite a construção e integração de sistemas a partir

de uma composição de serviços interoperáveis. Um dos principais focos do modelo SOA está no baixo acoplamento entre serviços, sistemas operacionais, linguagens de programação e outras tecnologias que compõem as aplicações, permitindo assim uma computação distribuída e programação modular. Desta forma, toda a arquitetura criada é composta por um conjunto de serviços que comunicam entre si através da passagem de parâmetros entre consumidores e produtores de serviços.

2.5.1 Sistemas Baseados em Eventos

De acordo com Mühl et al. [37], os sistemas baseados em eventos utilizam padrões aplicáveis no desenho e implementação de aplicações e sistemas que necessitam transmitir eventos entre componentes e serviços com baixo acoplamento. De uma forma geral, sistemas baseados em eventos são compostos de emissores (ou agentes) e consumidores (e.g. *sink*) de eventos. Enquanto os agentes são responsáveis por detectar mudanças de estado e criar os eventos representando estas mudanças, os consumidores se responsabilizam por reagir ao receber notificações dos eventos gerados no sistema. Em alguns casos, um componente ou serviço externo ao consumidor fornece a reação a partir da filtragem, transformação e encaminhamento dos eventos para outro componente no sistema. O *middleware* orientado a mensagens ou *Message Oriented Middleware* (MOM) exemplifica o tratamento externo de eventos através da disseminação de mensagens e utilizando um sistema de enfileiramento destas. Caporuscio et al. [8] cita a grande aceitação deste paradigma como o padrão de comunicação para a disseminação de informações em sistemas, e neste contexto, a possibilidade de correlacionamento entre eventos que podem ser de interesse a aplicações e serviços móveis. Exemplos de MOMs utilizados em sistemas distribuídos incluem o *Java Message Service* (JMS) [51] e *middlewares* baseados em protocolos residentes no nível de aplicação como o XMPP e o *Advanced Message Queuing Protocol* (AMQP) [1].

No que diz respeito à composição de um sistema baseado em eventos, podemos destacar quatro componentes principais, conforme apresentado na Figura 2.3: 1) gerador de eventos, 2) canal de eventos, 3) processador de eventos e 4) atividades baseadas em eventos. O gerador de eventos é responsável por detectar mudanças de estado em objetos e representá-las em forma de eventos. Após a criação destes eventos, o canal de eventos permite a transmissão destes para os responsáveis pelo processamento (e.g. consumidores ou *sink*). Exemplos destes canais incluem mensagens de requisição e resposta sobre *Transmission Control Protocol over Internet Protocol* (TCP/IP) e arquivos de dados em formatos diversos, como o XML ou texto, dentre

outros. Após a transmissão dos eventos pelo canal, o processador fornece a identificação dos eventos e a seleção das atividades apropriadas. As atividades baseadas em eventos definem o momento em que as consequências dos eventos processados são visualizadas ou consumidas. Este consumo pode-se variar desde à notificação visual através de mensagens (e.g., tela de *popup* até uma execução de uma rotina interna da aplicação, como por exemplo, para gerenciar o consumo de energia do dispositivo.

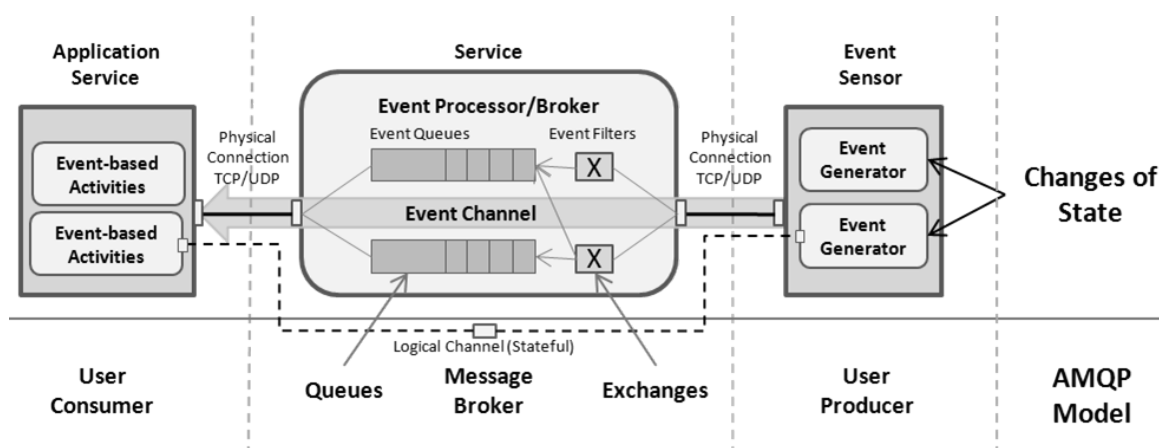


Figura 2.3. As camadas de fluxo de um sistema baseado em eventos. As entidades definidas no modelo AMQP estão associadas com os elementos definidos no sistema proposto neste trabalho.

2.5.2 Taxonomia em Sistemas Baseados em Eventos

Podemos citar diversos sistemas baseados em eventos com características similares e diferentes no que diz respeito ao modelo de eventos e o serviço de eventos utilizado. Meier et al. [36] apresentam uma classificação destes tipos de sistemas utilizando uma taxonomia a partir de uma pesquisa das principais implementações e levantando em consideração as principais características necessárias para estes sistemas. Neste trabalho, alguns sistemas de eventos foram avaliados, que incluem o CORBA *notification service model* [14], SIENA [9], SECO [24] e Hermes [42].

Nesta taxonomia, utilizaram-se três principais raízes: o sistema, o serviço e o modelo. Define-se o sistema como sendo a aplicação cliente que utiliza os recursos de um servidor de eventos para utilizar a comunicação baseada em eventos. O serviço de eventos implementa um modelo de eventos pré-definido, provendo assim a comunicação a um sistema baseado em eventos. Um modelo de eventos consiste de regras que descrevem um modelo de comunicação baseado em eventos. De forma similar, este

modelo define a visão que a aplicação possui do serviço de eventos ou a forma em que aplicações subscrevem por eventos no sistema.

Na dimensão do modelo de eventos, definem-se três categorias: *peer-to-peer*, mediador e implícito. Na categoria P2P, o sistema permite a subscrição direta por produtores e a publicação direta de eventos por produtores para seus consumidores. Na categoria "mediador", consumidores e produtores utilizam um mediador para a subscrição e publicação de eventos, respectivamente. Este mediador recebe as subscrições e repassa os eventos gerados por produtores em função dos consumidores subscritos. O mediador pode estar organizado na forma singular ou múltiplo. No singular, apenas um mediador interage com os elementos do sistema, enquanto o múltiplo utiliza vários mediadores com responsabilidades ou funcionalidades equivalentes ou diferentes. Utilizando mediadores múltiplos equivalentes, entidades podem subscrever ou enviar eventos para qualquer um destes. No caso de mediadores múltiplos diferentes, as entidades precisam utilizar os mediadores corretos para as principais operações tais como subscrição, publicação e notificação de eventos. Diferente dos modelos P2P e mediador, o implícito utiliza a subscrição por tipos de eventos, ao contrário da subscrição direta por entidades ou mediadores. Produtores geram eventos de determinados tipos que são entregues as suas respectivas entidades subscritas.

Na dimensão do serviço de eventos, é apresentada a classificação das propriedades deste serviço. Nesta dimensão, Meier et al. [36] define três categorias: organização, modelo de interação e recursos. A organização define a distribuição das entidades e o *middleware* de um sistema de eventos e a forma de colaboração entre estas entidades. Em relação às entidades, a distribuição pode ser centralizada (e.g., mesmo espaço de endereço ou local físico) ou distribuída (e.g., diferentes espaços de endereço ou locais físicos). O *middleware* que fornece o serviço de eventos pode estar acoplado no mesmo espaço de endereço que as entidades ou em espaços separados. Além de estar em espaços de endereço separados, o serviço de eventos pode ser particionado de forma única, isto é, um servidor de eventos no sistema, ou múltipla (e.g., distribuída em endereços diferentes). As principais características da organização estão na escalabilidade do sistema, tratamento de erros e falhas e o meio de comunicação entre as entidades e o *middleware*. Desta forma, abordagens mais centralizadas podem com mais frequência experimentar problemas de desempenho com o aumento da utilização, podendo assim sofrer uma quantidade maior de falhas do que abordagens distribuídas, que permitem a distribuição da comunicação em diferentes entidades. A interação apresenta o canal em que entidades consumidoras e produtoras se comunicam. Podemos classificar a interação em sistemas de eventos em duas categorias: intermediário e não intermediário.

No não intermediário, a comunicação entre produtores e consumidores é feita de forma direta através de componentes residentes em cada um destes. No intermediário, a comunicação é indireta, realizada por um ou mais componentes entre as duas partes. Na comunicação intermediária, componentes ou *middlewares* responsáveis pela comunicação podem estar organizados de forma centralizada ou distribuída. Enquanto na centralizada existe apenas um intermediador entre o produtor e consumidor, a distribuída envolve dois ou mais intermediadores na comunicação. Os recursos apresentam características funcionais e não-funcionais disponíveis em um serviço de eventos. Alguns dos recursos funcionais incluem o modelo de propagação, tipos, filtros, mobilidade, e composição de eventos. Para os recursos não funcionais, podemos citar a qualidade de serviço, ordenação e tolerância a falhas.

Na seção 4.1, apresentaremos a classificação do serviço de eventos proposto por este trabalho em função das categorias propostas em Meier et al. [36]. O nosso objetivo é demonstrar a caracterização definida no sistema proposto, a fim de apresentar os recursos presentes e não presentes no mesmo.

2.5.3 Sistemas Baseados em Eventos para Aplicações Ubíquas

O estudo de roteamento distribuído de eventos em sistemas *Publish/Subscribe* (*Pub/Sub*) apresenta vários casos que focam em sistemas distribuídos [3]. Entretanto, de uma forma geral, nenhum deles é aplicado ou avaliado diretamente sobre sistemas ubíquos. O modelo de comunicação baseado em eventos descreve um paradigma aplicável na interconexão de elementos que compõem aplicações em ambientes ubíquos de forma assíncrona. Estes ambientes em geral contem aplicações e elementos de rede que são heterogêneos e distribuídos. Eles são heterogêneos devido ao fato destes poderem executar diferentes tipos de tarefas diferentes para o usuário, requerendo conjuntos variados de componentes de *hardware* e *software*. Eles também são distribuídos de tal forma que um único componente poderá depender de diversos outros, como por exemplo, na obtenção de informações do ambiente através da utilização de componentes variados (e.g., medidores de luminosidade, temperatura, pressão atmosférica, umidade, qualidade do ar) e execução de tarefas pelo usuário a partir destas informações. O baixíssimo acoplamento entre componentes e a alta flexibilidade na adaptação e extensão de serviços para aplicações móveis servem como benefícios para a adoção desta arquitetura na construção de aplicações móveis e também de serviços ubíquos.

Os trabalhos relacionados envolvem pesquisas principalmente nas áreas de computação ciente de contexto e a modelagem de sistemas distribuídos baseados em

eventos. Caporuscio et al. [8] apresentam um projeto de desenho de um sistema baseado em eventos que permite as aplicações detectarem eventos em uma determinada região e avaliarem a relevância destes levando em consideração o contexto principal da aplicação. Isto permite que aplicações se adaptem às condições ambientais específicas, alcançando assim seus objetivos. A especificação utilizada inclui o desenho de um protótipo baseado no modelo *Pub/Sub*, similar ao utilizado neste trabalho. Entretanto, nosso trabalho enfoca na utilização deste modelo especificamente em ambientes ubíquos cientes de contexto através da utilização de mensagens de eventos gerados pelo sistema e publicados a clientes de interesse.

Sacramento et al. [46] propuseram uma arquitetura *middleware* que permite o desenvolvimento de serviços de processamento de informações de contexto e aplicações sensíveis ao contexto sobre redes sem fio. Na infraestrutura de comunicação do *Mobile Collaboration Architecture* (MoCA), a interface de comunicação de eventos ou *Event Communication Interface* (ECI) fornece serviços assíncronos de comunicação através do modelo de interação *Pub/Sub*. Dentro desta estrutura, o servidor de eventos publica os eventos ocorridos para consumidores interessados e subscritos. No nosso trabalho, estendemos o acesso a informações de perfil e contexto do usuário para serviços remotos existentes na Web através da publicação de eventos vindos do cliente móvel. A interação entre entidades usada pelo MoCA se assemelha muito à utilizada neste trabalho, com uma pequena diferença na forma como o serviço de eventos proposto foi concebido. Construímos o serviço utilizando recursos disponíveis em nuvem (i.e., processador de requisições, persistência de dados, dentre outros) com o objetivo de abstrair e simplificar a integração deste com outros serviços presentes na Web.

No que diz respeito à arquitetura (i.e., topologia e protocolo de comunicação), Carzaniga et al. [9] apresentam quatro soluções arquiteturais para sistemas distribuídos baseados em eventos que podem ser utilizadas na construção de aplicações ubíquas sensíveis de contexto e na disseminação de informações de contexto de diversos clientes. A primeira delas apresenta uma arquitetura cliente/servidor, onde clientes estão conectados a servidores e estes servidores podem ou não estar conectados entre si, sendo uma extensão de uma arquitetura centralizada. Servidores recebem eventos gerados em seus clientes e retornam notificações para os consumidores interessados, sendo estes usuários ou serviços remotos. A segunda proposta define uma arquitetura acíclica *Peer-2-Peer* (P2P) onde servidores se comunicam em forma de pares, garantindo assim um fluxo bidirecional de eventos, subscrições e notificações entre os clientes conectados. A terceira proposta apresenta uma arquitetura genérica P2P onde a conectividade entre servidores apresenta redundâncias, permitindo a comunicação bidirecional da mesma forma que a solução acíclica, mas permite múltiplos caminhos entre servidores e

indiretamente entre clientes. A quarta e última proposta apresenta uma solução híbrida onde diversos níveis de hierarquização e comunicação são definidos para servidores e clientes que compõem a arquitetura.

Diferente das propostas apresentadas acima, este trabalho utiliza uma solução de arquitetura cliente/servidor em nuvem para o sistema de gerenciamento de contexto e de eventos. Desta forma, clientes se conectam em um único servidor de dados remoto e acessam serviços remotos baseados em informações de eventos disponíveis a partir do servidor para a busca de informações relevantes de contexto remoto ou global. Além disto, o servidor é responsável por fornecer serviços de subscrição, filtragem e notificação de eventos que ocorram local e remotamente que são de interesse do usuário ou de serviços remotos baseados em informação.

2.6 Tecnologias em Sistemas Distribuídos e Móveis

Esta seção apresenta as tecnologias utilizadas na construção dos protótipos apresentados neste trabalho para a avaliação da viabilidade de um servidor de notificação de eventos para aplicações móveis.

2.6.1 Plataformas de Software e Serviços

As plataformas de *software* possuem um importante papel na construção de aplicações e serviços, independente destes estarem em dispositivos móveis, em computadores de mesa ou em servidores. Elas definem em geral uma arquitetura que serve de fundação para que aplicações e serviços possam utilizar e compartilhar funcionalidades presentes. O objetivo é criar uma abstração ou padronização de funcionalidades e serviços que podem ser usados por aplicações em execução sobre estas. Exemplos clássicos de plataformas de *software* incluem os sistemas e ambientes operacionais, base de dados, arcabouços de desenvolvimento para a Web e as plataformas de *software* Java e .NET [38]. Os arcabouços definidos por estas plataformas são similares a bibliotecas de *software* em geral, no sentido de que elas definem abstrações de código reusável através de uma interface de programação de aplicações ou API bem definida. Além disto, o arcabouço é também responsável pelo fluxo de controle dos programas sendo executados sobre o mesmo, oferecendo recursos de *hardware* e *software* do sistema e controlando os estados das aplicações e serviços em execução.

As plataformas também oferecem serviços genéricos nos quais diversos tipos de aplicações possuem acesso para prover funcionalidades mais específicas. Um exemplo deste cenário está na construção de aplicações móveis, onde estas utilizam serviços

providos pelas plataformas móveis disponíveis nos dispositivos. Estas plataformas apresentam um conjunto pré-definido de funcionalidades que permitem o reuso em que aplicações desenvolvidas sobre estas, apresentando atributos como a reusabilidade, a disponibilidade e confiabilidade.

Satyanarayanan [47] destaca dois dos principais desafios na computação ubíqua: (a) a união de várias tecnologias em um único sistema e (b) da integração transparente destes no contexto ubíquo de execução de aplicações e serviços para usuários. A integração envolve o estudo e análise das tecnologias viáveis e, em alguns casos tecnologicamente inviáveis no momento da construção de sistemas móveis e ubíquos. Muitas destas tecnologias utilizam mecanismos de comunicação proprietários ou diferentes entre si, sendo necessário um mecanismo padrão de comunicação e de transmissão de dados através de mensagens para o provimento da interação entre estes. O modelo *Pub/Sub* possibilita a união e a integração entre os diversos componentes no sistema (e.g. *hardware-hardware*, *hardware-software* e *software-software*), onde cada componente pode se comportar como um consumidor, como um produtor ou como ambos. Este modelo permite uma padronização na subscrição por eventos de interesse e também na comunicação de dados relevantes entre componentes do sistema, como por exemplo, durante mudanças de dados de perfil e contexto de sensores posicionados local ou remotamente em um ambiente.

Utilizando alguns cenários aplicáveis de tecnologias de sistemas móveis e ubíquos, é possível destacar tecnologias de *hardware* e *software* existentes hoje que podem ser utilizadas para tal construção da solução. Apesar da disponibilidade destas tecnologias no desenvolvimento de aplicações para o usuário móvel, a criação de componentes que possam fornecer os serviços necessários para aplicações ubíquas de forma unificada e totalmente integrada ainda é um desafio significativo. Por exemplo, para as tecnologias de *hardware*, podemos destacar os dispositivos móveis tais como os *notebooks/netbooks*, *handhelds*, PDAs *smartphones*, a comunicação de dados sem fio, redes de sensores sem fio, dentre outros. Em relação às tecnologias de *software*, podemos citar aplicativos e serviços processadores de informações para o usuário móvel tais como localização, interface gráfica de usuário, arcabouços de desenvolvimento, padrões de comunicação de dados como o HTTP e o XML, dentre outros.

2.6.2 Aplicações Móveis

As plataformas de *software* para dispositivos móveis definem o sistema operacional que controla um dispositivo, cujo conceito é similar aos sistemas operacionais presentes em máquinas *desktop* tais como o *MS Windows* ou *Linux*. Entretanto, devido às

Tabela 2.1. Plataformas de *software* para dispositivos móveis.

Plataformas de SW	Características
<i>Symbian OS</i> [55]	<i>Open-source</i> , C/C++, linguagem de programação <i>Python</i> [44]
<i>Java Mobile Edition</i> (JavaME) [54]	Linguagem de programação Java, Subconjunto do <i>Java Standard Edition</i> (JavaSE)
<i>Android</i>	Linux, Linguagem de programação Java
<i>IPhone SDK</i> [2]	C, <i>Objective C</i> , Proprietário da Apple
<i>.NET Compact Framework</i> [38]	Linguagem de programação <i>C Sharp</i> , Proprietário da Microsoft

restrições de *hardware* e *software* presentes, estes sistemas são mais simples, tratam a conectividade de dados de forma mais direta e abrangente (e.g., local e sem fio) e oferecem a manipulação de diferentes formatos de mídias (e.g., áudio, texto, imagem, vídeo) e métodos de entrada de dados, tais como telas sensíveis ao toque e teclados do tipo *querty*.

Uma das evoluções recentes na computação móvel está no provimento de serviços baseados em informações além dos serviços de voz e SMS já existentes. Esta mudança recente e progressiva tem se tornado um passo importante para a evolução da computação móvel, trazendo assim novos serviços anteriormente inviáveis. Apesar deste avanço, alguns desafios ainda existem, tais como: a) conectividade contínua entre diferentes meios de acesso a redes (e.g., casa, automóvel, *Wireless Local Access Network* (WLAN), *Third Generation Mobile Communications Standard* (3G)), b) interoperabilidade de *hardware* e *software* (plataformas e aplicações) e c) serviços baseados em informações de contexto (localização, hora, estado do dispositivo, conectividade, etc).

O mercado atual de dispositivos móveis oferece diversas plataformas de *software* para o desenvolvimento de aplicativos móveis e serviços. A listagem das principais plataformas pode ser visualizada na Tabela 2.1. Criada pela Google em Novembro de 2007, a plataforma de *software* e sistema operacional *Android*³ permite a criação de aplicativos na linguagem de programação Java de uma forma gerenciável através da utilização de uma máquina virtual, controlando assim o dispositivo via bibliotecas Java providas pela própria Google. Um consórcio denominado de *Open Handset Alliance* (OHA) composto de 48 empresas de *hardware*, *software* e telecomunicações foi criado com o objetivo de avançar a criação e uso de padrões abertos para dispositivos móveis. A maioria do código que compõe o *Android* é distribuída sob a licença Apache que define uma licença gratuita e de código aberto (*open-source*).

A plataforma *Android* define uma pilha de *software* para dispositivos móveis,

³<http://code.google.com/android/>

incluindo um sistema operacional, *middleware* e aplicações, conforme apresentado na Figura 2.4. O *Systems Development Kit* (SDK) fornece as ferramentas e as bibliotecas (e.g., APIs) necessárias para o desenvolvimento de aplicações sobre a plataforma usando a linguagem de programação Java. Os principais recursos oferecidos pela plataforma incluem:

- **Arcabouço de aplicações:** permite o reuso e substituição de componentes de SW;
- **Virtualização:** máquina virtual *Dalvik* otimizada para dispositivos móveis;
- **Navegação Web:** navegador Web integrado baseado no *WebKit Engine*;
- **Gráficos:** suporte a gráficos 2D e 3D baseado na especificação *OpenGL for Embedded Systems* (OES) 1.0;
- **Base de dados:** armazenamento de dados local através do *SQLite*;
- **Multimídia:** suporte a diferentes tipos de mídia (áudio, vídeo e imagens);
- **Rede:** conectividade de dados e voz através de tecnologias sem fio tais como *Global System for Mobile Communications* (GSM), *Bluetooth*, *Enhanced Data rates for GSM Evolution* (EDGE), 3G e WLAN;
- **Desenvolvimento:** ambiente de desenvolvimento composto de um emulador de dispositivo, ferramentas para depuração, análise de perfil de memória e desempenho e *plugins* para ambientes integrados de desenvolvimento tais como o Eclipse.

Na plataforma *Android*, utilizamos duas abordagens para a recepção e processamento de eventos vindos do servidor remoto de dados no protótipo *DroidGuide*. Na primeira delas, utilizamos a criação de processos concorrentes (e.g., *threads*) para o gerenciamento dos eventos na aplicação móvel. Na segunda abordagem, utilizamos uma extensão da classe *BroadcastReceiver* para o gerenciamento dos eventos. O receptor de *broadcast* ou *broadcast receiver* é um componente responsável por receber e reagir a eventos notificados durante a execução da aplicação. Exemplos de notificação incluem nível baixo de bateria, mudança de fuso horário, mudança de idioma, dentre outros. Além de receber as notificações, aplicações desenvolvidas no Android também podem submeter notificações a fim de notificar outros componentes a ocorrência de um evento de interesse. Aplicações podem possuir diversos receptores de *broadcast* para responderem a notificações importantes. Todos os receptores devem estender a classe

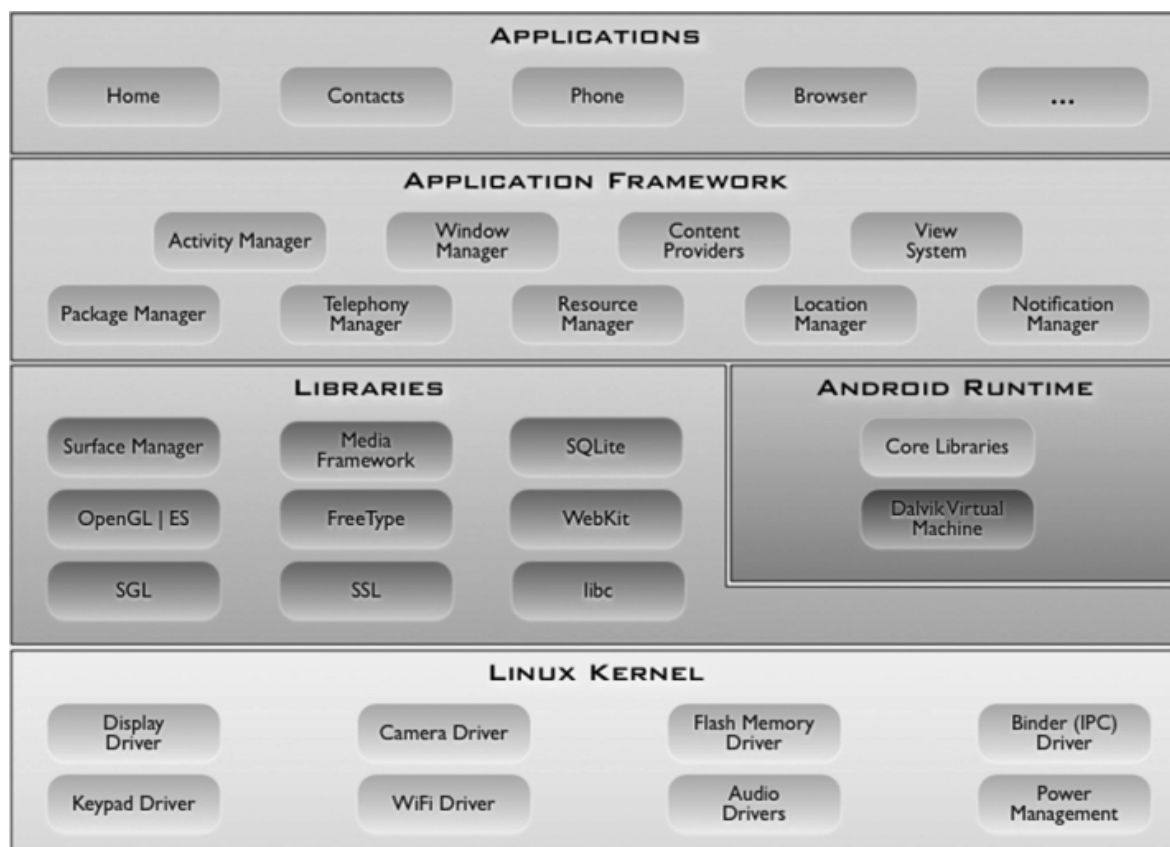


Figura 2.4. A pilha de *software* definida na plataforma *Android*.

base *BroadcastReceiver*. Os receptores não possuem interface gráfica, mas podem iniciar atividades em resposta a informações recebidas ou alertar o usuário móvel através do gerenciador de notificações ou *NotificationManager*.

Os receptores de *Broadcast* recebem intenções de execução a partir de atividades em execução no dispositivo. Neste conceito, componentes produtores publicam intenções de execução (e.g., *Intents*) para que receptores possam executar determinadas atividades em função da recepção da mesma. As intenções e o tratamento destas se assemelham à eventos produzidos e publicados em um sistema de eventos distribuído. No caso do *Android*, Cada receptor possui um filtro responsável por selecionar ou não intenções recebidas, definindo assim se devem ser consumidas ou não.

Além do *Android*, podemos destacar também outras plataformas de *software* para dispositivos móveis, tais como o JavaME. O JavaME [54] define uma especificação de um subconjunto da plataforma JavaSE voltado para a construção de aplicativos para dispositivos pequenos e com pouco poder computacional (e.g., energia, conectividade e processamento). Neste trabalho, buscamos por uma plataforma de *software* para

dispositivos móveis com as seguintes características: 1) código aberto (*open-source*), 2) biblioteca de programação rica com vários recursos de interface gráfica (componentes gráficos, interface com o usuário), 3) suporte a navegação Web com recursos utilizados tais como o serviço Web de mapas e *JavaScript* (JS), 4) possibilidade de construção e execução de protótipos com vários clientes simultâneos e 5) API de conectividade de dados rica com suporte a diferentes tipos de conexões de rede. De todas as características destacadas acima, o *Android* se mostrou o mais adequado na construção dos protótipos deste trabalho, seguido pelo JavaME e o *Symbian OS*.

2.6.3 Sistemas Distribuídos

Um sistema distribuído é composto de diversos elementos computacionais autônomos que se comunicam através de uma rede de computadores. Estes elementos computacionais (e.g., computadores, sensores, dispositivos móveis) interagem entre si com o objetivo de alcançar um determinado objetivo. A computação distribuída faz uso destes tipos de sistemas, onde os problemas ou responsabilidades são divididos entre os elementos, e cada um destes é responsável em solucioná-los. A computação distribuída também propõe a definição de algoritmos e modelos utilizando grafos com representações de redes com topologia dinâmica, tendo como benefício o foco no comportamento do sistema ao invés da topologia da mesma. Alguns exemplos de sistemas distribuídos incluem sistemas com arquitetura multicamada tais como cliente-servidor e multicamada, redes e serviços de telecomunicações tais como a Internet, RSSFs, computação em grade e em nuvem.

Vaquero et al. [58] define a computação em nuvem de uma forma genérica como sendo a utilização e desenvolvimento de tecnologias dinamicamente escaláveis e virtuais, providas em forma de serviços sobre a Internet. A escalabilidade, o modelo de utilização "pagar por utilização" e a virtualização de serviços de *hardware* e *software* definem os principais aspectos deste modelo computacional, onde o produto de *software* é apresentado em forma de serviço ou *Software as a Service* (SaS) com o objetivo de satisfazer as necessidades do usuário. Os serviços são geralmente providos através de centros de dados e construídos a partir de diferentes níveis de tecnologias de virtualização (e.g., discos, memória, unidade de processamento). A "nuvem" representa a possibilidade de desenvolvimento, instalação e acesso a estes serviços de qualquer local a partir de um acesso em rede.

O modelo computacional da computação nas nuvens apresenta outras características chave tais como a independência de dispositivo e local de acesso, o compartilhamento de recursos com uma grande quantidade de usuários, a

Tabela 2.2. Padrões utilizados pelo modelo de computação nas nuvens.

Características	Padrões
Comunicação de dados	HTTP, XMPP
Segurança	<i>Secured Socket Layer (SSL)/ Transport Layer Security (TLS)</i>
Clientes de acesso	<i>Assynchronous JavaScript and XML (AJAX), XML e HTML</i>

confiabilidade, escalabilidade e segurança. A Tabela 2.2 apresenta diversos padrões adotados na concepção e utilização de serviços disponibilizados por provedores de recursos em nuvem. Os componentes que fazem parte do contexto do modelo de computação nas nuvens são:

- **Aplicações:** P2P, *Google Apps*, aplicações Web;
- **Clientes:** móveis (e.g., *Google Android*, JavaME, *Symbian OS*, *Windows Mobile*), clientes leves e pesados;
- **Infra-estrutura:** computação em grade, virtualização de infra-estrutura de computadores;
- **Plataformas:** arcabouços Web (AJAX, *Python Django*, *Java Enterprise Edition* (JavaEE)), hospedagem de sítios;
- **Serviços:** identificação, integração, pagamentos, mapeamento e consulta a informações;
- **Armazenamento:** bases de dados remotas (*BigTable*⁴), *Web Services*.

O arcabouço *Google Web AppEngine Framework* (GAE)⁵ exemplifica o modelo computacional mencionado acima. Ele define uma plataforma de desenvolvimento de aplicações Web que permite o desenvolvimento, instalação e execução de aplicações na linguagem *Python* ou Java sobre a infra-estrutura de servidores Web da Google. Dos vários benefícios apresentados, alguns incluem a praticidade no desenvolvimento, provimento e administração de aplicações Web (e.g., infra-estrutura virtual) e de APIs, acesso ao armazém de dados, integração com outros serviços Web tais como *GMail*, *YouTube*, *GTalk*, *GDocs*, dentre outros. Entretanto, algumas restrições existem na execução de aplicações Web sobre esta infra-estrutura, tais como o protocolo HTTP sendo o único meio de acesso ao servidor de aplicações e dados até o momento da criação dos protótipos, acesso somente de leitura ao sistema de arquivos do servidor, suporte a somente duas linguagens de programação (e.g., *Python* e *Java*) e quotas

⁴Google Query Language: <http://code.google.com/intl/pt-BR/apis/base/docs/2.0/query-lang-spec.html>

⁵<http://code.google.com/intl/en/appengine/docs/whatisgoogleappengine.html>

limitadas para o uso gratuito da infra-estrutura que inclui o espaço utilizado e recursos computacionais como a largura de banda e o processamento de requisições de clientes.

2.6.4 Tecnologias Web para Aplicações Móveis

Aplicações móveis em geral utilizam componentes construídos a partir de cada uma das plataformas existentes no mercado atualmente. Cada um destes componentes está adaptado para sua plataforma específica no que diz respeito ao tamanho, comportamento e aparência na interface de usuário. Porém, devido à esta exclusividade, não é possível utilizarmos determinadas aplicações em um conjunto maior de dispositivos, já que cada um possui características bem definidas, tais como resolução, dispositivos de entrada de dados, poder computacional, dentre outros. Aplicações móveis que possuem uma necessidade em executar em diferentes dispositivos precisam ser adaptadas ou até reconstruídas para cada uma das plataformas específicas, mesmo em casos onde elas possuam um denominador em comum, como no caso do JavaME.

A partir desta visão, utilizamos uma abordagem diferente no que diz respeito à construção de aplicações móveis. Ao invés de utilizarmos uma API específica, desenvolvemos uma aplicação para ser executada em dispositivos móveis utilizando algumas das tecnologias Web mais utilizadas, tais como o HTML e a linguagem *JavaScript*. O principal objetivo foi possibilitar a execução da aplicação em diferentes dispositivos com o mínimo de adaptação possível utilizando um navegador Web móvel. Podemos destacar vários benefícios nesta abordagem, como por exemplo, o desenvolvimento de uma interface única, utilização de serviços existentes na Web, dentre outros.

Diante dos benefícios existentes na construção de aplicações móveis utilizando tecnologias Web, destacamos a facilidade de execução de clientes de usuários simultâneos através do uso de navegadores Web e/ou emuladores de dispositivos móveis com suporte a navegação Web, tais como o *Android SDK*. No caso do *Android SDK*, a classe *WebView* junto com as permissões de acesso correspondentes oferecem uma atividade de navegação Web para a aplicação em execução. Entretanto para outras plataformas, tais como o JavaME e o *Symbian*, a simulação deste protótipo tornou-se mais difícil devido à falta de suporte a recursos utilizados pelo protótipo. As tecnologias Web utilizadas neste trabalho incluem:

- ***JavaScript***: linguagem de *script* usada no desenvolvimento de clientes Web através de funções que são embutidas ou inclusas no documento HTML, possibilitando uma alta responsividade da interface no lado cliente. O código em *JavaScript* pode ser executado tanto no lado cliente quanto no lado servidor.

Navegadores Web em geral oferecem o ambiente hospedeiro para a execução de código nesta linguagem para clientes, incluindo a existência de alguns destes em dispositivos móveis. Alguns dispositivos, porém, podem não prover suporte a execução de código em *JavaScript*. Na transmissão de dados entre o servidor e o cliente Web, utiliza-se a notação *JavaScript Object Notation* (JSON), definida pela RFC 4627⁶.

- **HTML**: linguagem de marcação de hiper-texto na Web, provendo a estrutura de informações baseadas em texto em forma de um documento. O HTML oferece a interface de visualização de conteúdo para o usuário em um navegador Web.
- **Google Maps API⁷**: interface de programação utilizada na integração de mapas do Google com aplicações Web. O serviço utiliza *JavaScript* e JSON [22] para o envio de dados para clientes Web. Com este recurso, foi possível utilizarmos mapas disponibilizados pela Google para a apresentação de informações cientes de localização. No processamento de informações sobre os mapas, o serviço pode ser acessado de três formas: (a) através da API utilizando código em *JavaScript*, (b) através da API disponível para o arcabouço *Google Web Toolkit* (GWT) e (c) através da API disponível no *Android*.
- **Asynchronous JavaScript and XML**: conjunto de técnicas utilizadas na criação de aplicações Web interativas ou aplicações Internet ricas. Utilizando AJAX, aplicações Web acessam dados de um servidor Web remoto através de requisições assíncronas sem interferir com a visão ou com o comportamento da página sendo exibida, possibilitando um aumento na interatividade de animações em páginas Web. É necessário que o navegador Web possua suporte a *JavaScript* e também ao AJAX.

2.6.5 Serviços Móveis e Ubíquos

As áreas de projeto e algoritmos definem as duas principais áreas de conhecimento no desenvolvimento de sistemas móveis. Cada uma delas apresenta soluções para os desafios e limitações impostos pelas tecnologias (algoritmos, *hardware* ou *software*) e pelas funcionalidades de sistema (requisitos, recursos). Na área de sistemas móveis, destacam-se a computação distribuída, RSSFs, IHC, computação centrada ao usuário ou *Context-Aware Computing*, Inteligência Artificial (IA) e serviços ubíquos.

⁶JavaScript Object Notation (JSON): <http://tools.ietf.org/html/rfc4627>

⁷<http://code.google.com/apis/maps/>

A computação centrada no usuário ou *Context-Aware Computing* tem como objetivo prover soluções tecnológicas que satisfaçam o usuário final, utilizando seu comportamento como uma forma de aprimorar estas soluções e adaptá-las apropriadamente. Esta filosofia define uma nova maneira de se construir sistemas, se opondo à forma clássica onde o usuário final precisa se adaptar para a utilização da tecnologia. O objetivo é que o sistema saiba das condições do usuário e consiga propor atividades para o mesmo de acordo com estas informações coletadas. O sistema realiza para o usuário suas atividades de forma transparente e automatizada, provendo assim serviços ubíquos.

Os serviços ubíquos, em geral, buscam prover características importantes para dispositivos móveis, tais como a disponibilidade, transparência na utilização e na transição (e.g., localização e tecnologia), sensibilidade e confiança. A disponibilidade define para aplicações acessibilidade independente do contexto e mudanças de estado, necessidades e preferências do usuário, levando em consideração o contexto, a conectividade com os serviços e a energia do dispositivo. O foco do usuário deve ser maior na atividade ao invés da ferramenta, e a execução de tarefas deve ser feita sem a necessidade de atenção e ciência da tecnologia por trás. A transparência na transição permite o provimento de serviços para o usuário independente de sua localização ou seu meio de acesso. Isto inclui a disponibilidade de sessões de serviços sobre quaisquer conexões e em quaisquer tipos de dispositivos, reconhecimento do usuário em qualquer ponto de autenticação (e.g., sistema, dispositivo) a qualquer instante, o controle de sessão de estados em serviços e a necessidade de adaptação da aplicação a partir do contexto. Para isto, requisitos de *Quality of Service* (QoS), usabilidade, mobilidade, dentre outros devem ser definidos para sistemas ubíquos em questão.

A sensibilidade provê a extensão dos sentidos humanos, fornecendo um melhor conhecimento do ambiente, através de uma interação entre o usuário (e.g., contexto) e o serviço (e.g., realimentação). Para isto, é necessário que o sistema efetue: (a) a identificação do usuário e seu redor (vizinhos), (b) as atividades e comportamentos associados, (c) a localização para o provimento de funcionalidades, (d) a gestão do tempo, (e) inferências e (f) a identificação de razões para determinadas ações do usuário. Também relacionada com a identificação do usuário, a confiabilidade define diversos fatores tais como a integridade e unicidade entre entidades de um sistema ubíquo (dispositivos, usuários e serviços), o comportamento baseado em um contexto e a definição de relacionamentos baseados em identidade e/ou contexto físico entre entidades.

2.7 Cenários de Uso

Devido à característica pervasiva da computação ciente de contexto, podemos destacar diversos cenários de uso em aplicações ubíquas. Estes cenários em geral possuem características que justificam e incentivam o uso de aplicações ubíquas [47], tais como a coleta e compartilhamento de informações de perfil e contexto do usuário ou da aplicação, adaptação no que diz respeito às intenções do usuário, da interface gráfica e de serviços, privacidade, confiança e mobilidade. De todos estes cenários, destacam-se alguns trabalhos na área de turismo, em serviços de emergência, em escritórios e em ambientes educacionais [30]. De todos os cenários possíveis, optamos por explorar dois destes neste trabalho: um guia turístico eletrônico e um serviço de emergências.

2.7.1 Turismo

O cenário de uso envolvendo aplicações turísticas apresenta diversos modelos e protótipos existentes em trabalhos relacionados. Takeuchi et. al [56] apresenta um guia turístico sensível a localização que utiliza dados históricos de localização para a estimativa de interesses individuais através de um algoritmo de aprendizado de localização, além de oferecer mecanismos de busca por atrações baseadas em seus desejos. Cheverst et. al [16] apresenta um protótipo de um guia turístico adaptativo de hipermídia com a apresentação de conteúdo relacionado ao contexto do usuário. Long et. al [33] apresenta um guia turístico baseado em componentes, provendo uma alternativa expansível e modular para a aplicação móvel. A alternativa expansível possibilita a adição de serviços enquanto a componentização minimiza o impacto no sistema a partir de alterações em componentes da aplicação. Yue et. al [66] apresenta um guia turístico construído a partir de estudos de interação entre o usuário e a interface do dispositivo, buscando uma melhor interação entre as partes através do uso de comandos por voz, dentre outros. Podemos citar outros trabalhos relacionados como em Hertzog et. al [26] e em Bessho et. al [4] para ambientes externos e trabalhos como em Kuffik et al [32] e em Kuffik et al [28] para suporte a ambientes internos como museus. Todos estes trabalhos utilizaram dispositivos móveis como clientes, inclusive em Chen et al [15] com um estudo de caso na utilização de redes de sensores sem fio para estes tipos de aplicações.

2.7.2 Emergências e Desastres

O segundo cenário de uso utilizado neste trabalho engloba o contexto de gerenciamento de emergências e desastres. Tecnologias de comunicação e informação têm colaborado

de forma significativa e pervasiva em organizações de saúde, emergências e desastres com o objetivo de aumentar a eficiência de suas operações. Estes sistemas são responsáveis por prover o atendimento de emergências envolvendo um conjunto de atividades distribuídas e interdependentes realizadas de forma colaborativa por indivíduos com diferentes papéis e responsabilidades na saúde. Alguns trabalhos têm sido publicados nas áreas relacionadas, como por exemplo, em Jiang et al [29] apresentando o gerenciamento de emergências por bombeiros, em Camp et al [7] apresentando um sistema de comunicação para uso em situações críticas de segurança, a difusão de informações de contexto na saúde em Pallapa et al [40] e a importância da segurança da informação (i.e., privacidade, anonimato, autorização) em Wang et al [60].

Poulymenopoulou et al [43] apresenta a importância da qualidade e o grau de colaboração e coordenação entre indivíduos e elementos automatizados como sendo essenciais no fornecimento de serviços de alta qualidade a pacientes e na otimização de gastos em sistemas de comunicação e informação voltados para a saúde. Profissionais responsáveis pelo atendimento destas vítimas precisam estar altamente coordenados entre si e também possuem acesso a sistemas móveis capazes de fornecer serviços de auxílio para algumas de suas atividades, tais como informações de posição e condição clínica da vítima, acesso às instituições respectivas, um hospital, uma delegacia ou uma área segura designada em caso de desastres naturais ou diretamente gerados pela ação humana.

Na área de sistemas de informação, o uso de fluxos de trabalho tem contribuído na melhor compreensão de processos e abstrações de atividades de indivíduos ou outros tipos de elementos. O fluxo de trabalho (em inglês *workflow*) define uma seqüência de operações de uma pessoa, um grupo de pessoas, mecanismos simples ou complexos, organização ou máquinas. Ele também auxilia na descrição de sistemas de processamento de dados complexos de uma forma visual, sem a necessidade de se conhecer tópicos específicos da computação como programação. Poulymenopoulou et al [43] define um fluxo de trabalho para serviços de emergência como uma abstração composta por atividades com o principal objetivo de atender vítimas de forma mais eficiente possível, levando em consideração o tempo, o espaço e o custo. Algumas destas atividades incluem o fornecimento de cuidados emergenciais (ambulância em rota para o local do acidente), despacho de ambulâncias (recepção da ligação, seleção de ambulância, seleção de hospitais e roteamento de ambulâncias) e o gerenciamento da frota de ambulâncias (disponibilidade e escalonamento de pessoal). Um modelo de alto nível pode ser composto pelas seguintes atividades:

1. Recepção de dados iniciais do evento e a definição da urgência do evento no momento em que este ocorre;
2. Seleção da unidade móvel de atendimento baseada em fatores como a distância, tempo e condições de tráfego, notificação da mesma e o seu deslocamento até a posição do evento;
3. Informação de detalhes do incidente e o provimento de atendimento no local pela unidade móvel;
4. Seleção do marco mais apropriado para atendimento (hospital ou delegacia) baseada em fatores como a distancia, tempo, condições de tráfego e disponibilidade de vagas;
5. Notificação do marco (hospital, corpo de bombeiros ou delegacia) sobre informações referentes ao evento;
6. Informações de transporte da vítima ao marco;
7. Recepção da vítima no marco;

Capítulo 3

O Serviço Proposto

Este capítulo apresenta o servidor de eventos proposto neste trabalho. A seção um apresenta uma visão geral do serviço, suas principais fases e componentes. A seção dois apresenta uma definição conceitual de sistemas de eventos. Seção três apresenta a arquitetura do sistema, componentes e interfaces de acesso que compõem o serviço. Seção quatro apresenta uma classificação do serviço proposto a partir de uma taxonomia predefinida. Seção cinco apresenta o ciclo de vida de execução do serviço de eventos proposto para uso em aplicações móveis e ubíquas. Finalmente, a seção seis apresenta um resumo das aplicações de referência implementadas neste trabalho.

3.1 Visão Geral

O servidor de eventos possui as responsabilidades de: (a) gerenciar eventos em decorrer de mudanças nas informações de perfil e contexto de aplicações e usuários móveis e de serviços remotos, (b) fornecer a subscrição por eventos e serviços remotos através de canais de tópicos para aplicações e serviços móveis e (c) prover a notificação de eventos gerados em função de mudanças nas informações de perfil e contexto para consumidores interessados. O servidor de eventos é acessível a partir de um servidor remoto de dados onde clientes móveis o acessam utilizando requisições sobre o protocolo HTTP. A partir das requisições enviadas, o servidor de eventos envia respostas utilizando mensagens que contém metadados no formato XML sobre HTTP.

Como uma de suas principais responsabilidades, o servidor de eventos monitora as mudanças nas informações de perfil e contexto presentes no dispositivo móvel (local) e global (remoto). O sistema fornece este monitoramento em cinco deferentes fases ou etapas, conforme apresentado na Figura 3.1. Em cada uma das fases, diferentes

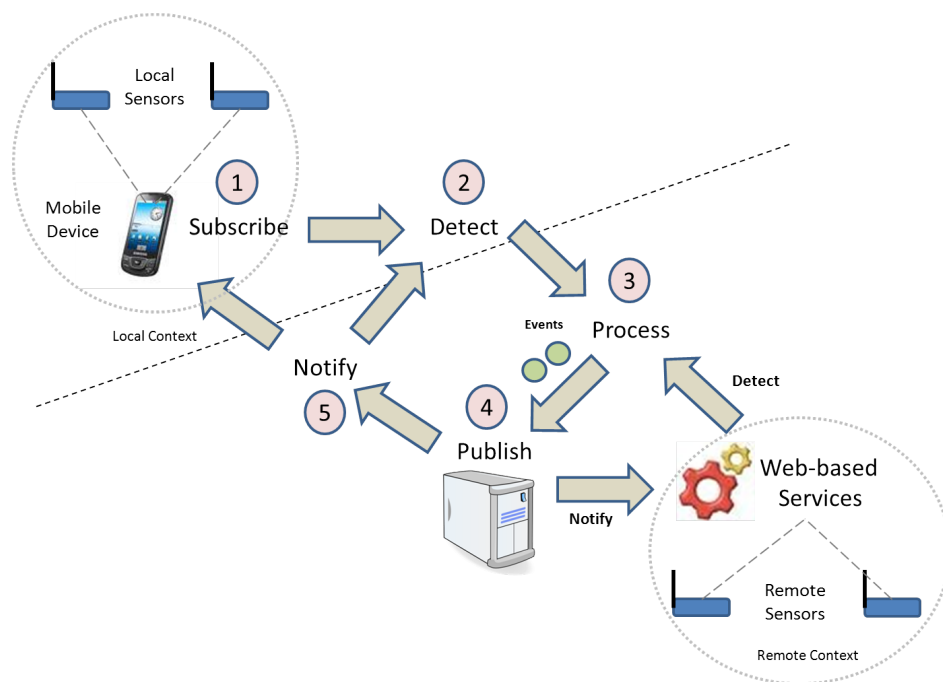


Figura 3.1. Principais fases de execução do modelo de gestão de eventos proposto.

componentes processam informações de perfil e contexto coletadas tanto no lado cliente quanto no lado servidor do sistema. As principais fases são:

1. **Subscrição:** A subscrição por canais de interesse pelo cliente (tópico, serviço ou atividade);
2. **Detecção:** A detecção de mudança e coleta de informações de perfil e contexto no cliente móvel;
3. **Processamento:** O processamento destas informações no lado do servidor, que corresponde à criação de eventos representando as mudanças detectadas no cliente;
4. **Publicação:** A publicação destes eventos no servidor através do modelo *Pub/Sub*;
5. **Notificação:** A notificação dos eventos publicados através de mensagens para consumidores subscritos.

O servidor de eventos provê a subscrição de mensagens através de tópicos de interesse e/ou serviços remotos baseados em informações de perfil e contexto local e

global. Desta forma, o sistema permite que um usuário móvel ou serviço localizado no dispositivo subscreva por tópicos de interesse para o recebimento e consumo de notificações geradas em função de outras aplicações, serviços e componentes no sistema, como por exemplo, o serviço de gestão de informações de perfil e contexto ou *Profile and Context Manager* (PCM) em uma aplicação turística. Este serviço está interessado em receber informações de perfil e contexto coletadas a partir de sistemas remotos a fim de auxiliar o usuário nas suas diversas tarefas. Neste caso, a assinatura por serviços de informações de interesse relativas à localização no ponto de vista turístico possibilita um rastreamento de mudanças externas de contexto físico contendo informações sobre o clima, condições de tráfego em uma determinada rota, dentro outras informações.

Aplicações e serviços em execução no dispositivo móvel são responsáveis em iniciar o processo de envio e notificação de eventos no sistema. Neste caso, a aplicação localizada no dispositivo móvel capta as mudanças nas informações de perfil e contexto do usuário móvel ou dispositivo. Para cada mudança detectada no perfil ou contexto local, a aplicação registra uma ocorrência e a prepara para envio para o servidor. Na recepção de novas ocorrências, a aplicação móvel as converte em objetos representando os eventos detectados, enviando-os para o servidor remoto de dados. Além do processamento de eventos no lado do cliente, a aplicação móvel também recebe e processa os eventos criados e enviados pelo servidor remoto de dados em função de mudanças detectadas no contexto global/externo, transformando-os em mensagens de notificação para o usuário.

No lado do servidor, o servidor de eventos recebe as requisições vindas de clientes móveis e as processa de acordo com as assinaturas por canais ou tópicos realizadas pelo usuário móvel. Após o processamento, o servidor de eventos repassa os eventos recebidos para o *container* de eventos, componente responsável pelo armazenamento de eventos criados no sistema. O *Container* de Eventos ou *Event Container* (CE/EC) armazena eventos criados em dois grupos: eventos por usuários subscritos e por tópicos. A primeira subdivisão permite que consumidores de eventos acessem de forma direta o conjunto de eventos gerados a partir do consumidor móvel (identificador do usuário móvel), enquanto a segunda permite o container organizar o conjunto de eventos em função de canais ou tópicos disponíveis no servidor de eventos. Na publicação de novos eventos, o servidor de eventos os repassa ao *container* que irá enfileirá-los de acordo com os consumidores subscritos e canais ou tópicos relacionados. Além de receber as requisições a partir de clientes móveis, o servidor de eventos também envia para clientes móveis mensagens de notificação de eventos criados em função de mudanças em informações de perfil e contexto presentes no lado do servidor. Estas mudanças podem ocorrer a partir de serviços Web remotos ou a partir de sensores remotos que

detectaram estas mudanças.

O gestor de subscrições gerencia a subscrição de elementos tais como tópicos de interesse, serviços Web remotos e atividades. O gestor organiza as subscrições de duas maneiras: em função do usuário móvel e de tópicos. A primeira maneira permite o acesso à lista de subscrições efetuada por um determinado usuário móvel, enquanto a segunda permite visualizar todas as subscrições relacionadas a um determinado tópico ou canal. Esta estruturação facilita o processo de correlação entre os eventos publicados no sistema e as subscrições disponíveis no gestor. Eventos publicados possuem um ou mais atributos responsáveis em definir um ou mais tópicos com subscrições interessadas em consumi-los. Estas subscrições definem os usuários interessados por eventos armazenados no *container* de eventos, permitindo que eles sejam agrupados de três formas: (a) em função das subscrições, (b) dos elementos de subscrição e (c) de cada usuário móvel no sistema.

Além do *container* de eventos, o servidor de eventos utiliza dois outros *containers*: o *container* de aplicações e de serviços Web. O *container* de aplicações gerencia componentes responsáveis pela lógica de negócio das aplicações em execução no servidor de dados remoto. Alguns exemplos de aplicações no servidor incluem o gestor de usuários, de informações turísticas, de emergências, de serviços de informação e marcos geográficos. O *container* de serviços Web gerencia os serviços localizados no servidor de dados remoto que são acessíveis por clientes móveis. Em conjunto com o gestor de subscrições, este *container* permite que usuários móveis subscrevam em serviços Web sensíveis à informações de perfil e contexto, permitindo assim que estes (os serviços) recebam informações de perfil e contexto coletadas a partir dos usuários. Em geral, entidades externas (empresas, corporações, prestadoras de serviços) representam a existência e funcionalidade dos serviços (i.e., interface e implementação) cadastrando-as no *container*. Após este cadastro inicial, usuários móveis assinam por estes serviços e usufruem de suas funcionalidades. Da mesma forma, estes serviços estarão aptos em receber informações de perfil e contexto do usuário móvel coletadas.

A fim de detectar mudanças explícitas nas informações de perfil e contexto, o servidor de eventos requisita de forma periódica (e.g., *push* periódico) informações localizadas no dispositivo, tais como localização, interesses e estado do usuário, dentre outros. Esta requisição periódica é também utilizada no envio de eventos do servidor para a aplicação móvel, possibilitando assim a notificação de eventos ocorridos no lado do servidor. Durante cada coleta, a aplicação móvel cria eventos representando as mudanças detectadas e os envia ao servidor de eventos localizado no servidor remoto de dados. Apesar de não termos tratado este recurso neste trabalho, este ciclo pode ser configurável e adaptável de acordo com as informações de contexto da aplicação móvel

e servidora (i.e., frequência na quantidade de eventos gerados, conectividade de rede, largura de banda, energia disponível, prioridade em eventos).

A partir do envio e armazenamento de eventos ao servidor e no *container* de eventos, respectivamente, o servidor de eventos aciona serviços Web sensíveis às informações de perfil e contexto que foram subscritos pelo usuário. Acionando estes serviços, o servidor efetua a notificação de eventos através da filtragem pelo gestor de subscrições. Esta filtragem utiliza o modelo baseado em tópicos que permite correlacionar canais de interesse, eventos ocorridos, consumidores e produtores. Neste caso, as referências presentes no *container* de serviços Web representam consumidores e produtores de eventos. Estas referências comunicam diretamente com o servidor de eventos localizado no servidor remoto de dados e com aplicações móveis localizadas no dispositivo.

No servidor de eventos proposto neste trabalho, serviços Web subscritos por usuários móveis consomem eventos gerados a partir destes usuários. Quando os usuários estiverem em um contexto que seja de interesse a algum serviço remoto, o servidor de eventos entrega os eventos armazenados para cada um dos serviços relacionados. Além de receber eventos, serviços Web podem também criar novos eventos em função das informações coletadas do usuário móvel ou da mudança no contexto global. Neste caso, o servidor de eventos responderá para o cliente em forma de mensagens de notificação apresentando os eventos relacionados ao contexto encontrado e informações dos serviços relacionados. Os clientes consomem os eventos a partir das mensagens de notificação enviadas pelo servidor, informando assim ao servidor que os eventos foram consumidos com sucesso.

3.2 Definição Conceitual

Apresentamos nesta seção uma definição conceitual de um servidor de eventos baseado no modelo *Pub/Sub*. Nestes sistemas, utilizam-se como entidades produtor, evento, tópico, consumidor e subscrição. Na geração de eventos, um ou mais produtores $p \in P$ são responsáveis por gerar um ou mais eventos $e \in E$ no sistema. Cada evento gerado é composto por um conjunto de propriedades $a \in A$, onde cada uma está relacionada à uma característica ou atributo observável existente nas informações de perfil e contexto do sistema (e.g., usuário móvel, dispositivo, serviços Web e contexto global). Cada característica observável possui uma relação com um ou mais canais ou tópicos $t \in T$, denominada de relação contexto-tópico ou *Context-Topic Relation* (CTR). Esta relação possibilita a associação entre as informações de perfil e contexto observados pelo sistema (e.g., produtores de eventos) com os tópicos disponíveis nas assinaturas por notificações

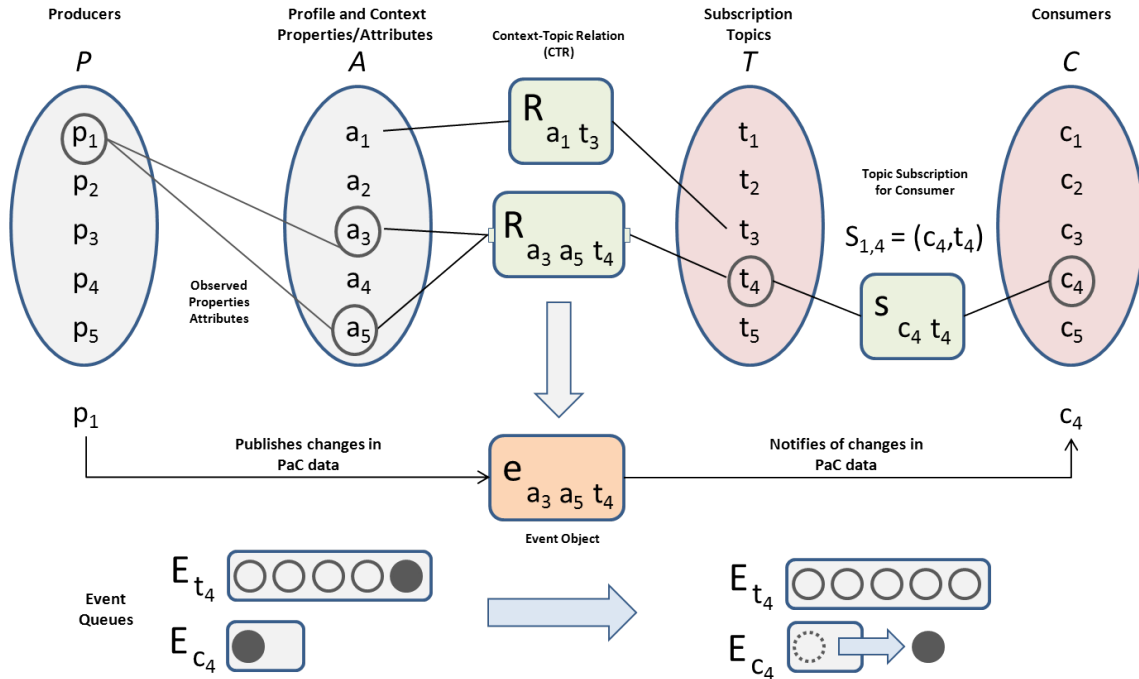


Figura 3.2. Representação das relações referentes a um sistema baseado em eventos sensível a informações de contexto.

que foram criadas em função de eventos gerados no sistema, conforme apresentado na Figura 3.2. Neste trabalho, utilizamos relações simples, isto é, uma propriedade ou atributo observável se relaciona com apenas um tópico de interesse, como por exemplo, nível de umidade e clima, condição das vias de tráfego, dentre outras alterações de estado nas informações de perfil e contexto do usuário móvel. Uma visão geral destas relações pode ser visualizada na Figura 3.2.

Na subscrição em notificações de eventos, o sistema de eventos disponibiliza para o consumidor uma listagem de tópicos de interesse relacionados aos eventos gerados e notificações enviadas. Sendo assim, a listagem de tópicos é usada diretamente por consumidores na subscrição em eventos. Cada consumidor $c \in C$ possui acesso ao conjunto de tópicos T para a criação de uma nova subscrição ou assinatura. O processo de subscrição em tópicos define uma função que retorna uma subscrição $s_{c_i t_j}$ em função do consumidor c_i e tópico t_j , armazenando informações do consumidor e do tópico subscrito. Estas informações são úteis no processo de publicação e notificação de eventos pelo sistema, possibilitando o acesso à listagem de tópicos subscritos por um determinado consumidor e a listagem dos consumidores subscritos a um determinado tópico para um evento sendo publicado pelo sistema.

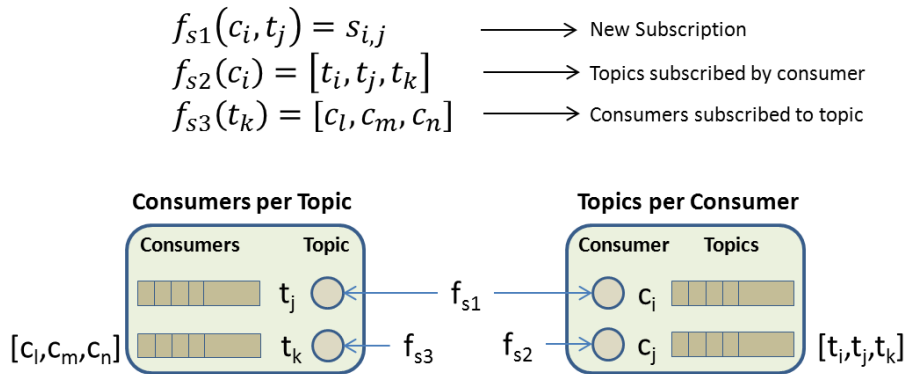


Figura 3.3. Funções definidas no gestor de subscrições.

A partir das relações apresentadas, um sistema de eventos provê serviços de subscrição em elementos, publicação e notificação de eventos ocorridos no sistema. Podemos organizar estes serviços em funções responsáveis por prover cada uma das funcionalidades necessárias. Conforme apresentado na Figura 3.3, um sistema de eventos utiliza três principais funções: f_{s1} , f_{s2} e f_{s3} . Em f_{s1} , o sistema define uma subscrição em um tópico t_i por um consumidor de eventos c_j . Em f_{s2} , o sistema retorna a listagem de tópicos subscritos por um determinado consumidor. Em f_{s3} , o sistema retorna a listagem de consumidores subscritos em um determinado tópico. Os componentes disponíveis em um sistema de eventos utiliza as três funções acima para fornecer serviços baseados em eventos ocorridos no sistema a clientes ou consumidores.

A subscrição em um sistema de eventos tem um importante papel na relação entre interesses de consumidores e eventos publicados. Além de permitir subscrições por tópicos de eventos gerados no sistema, ela também fornece métodos de consulta por subscrições em função do tópico e do consumidor. Desta forma, é possível sabermos duas importantes informações: (a) as subscrições realizadas por usuários ou consumidores e (b) as subscrições realizadas em determinados tópicos. A primeira será útil no momento em que se deseja saber quais tópicos um determinado consumidor subscreveu, enquanto a segunda é útil no momento em que produtores publicam eventos relacionados à tópicos com subscrições. A partir da desta listagem, o sistema de eventos conhece os consumidores que deverão receber as notificações dos eventos gerados por produtores no sistema.

A publicação de eventos tem como tarefa enfileirar eventos gerados por produtores nas filas de eventos E de acordo com as assinaturas realizadas por consumidores. Estes eventos podem ser armazenados em uma fila global ou especificamente alocada por

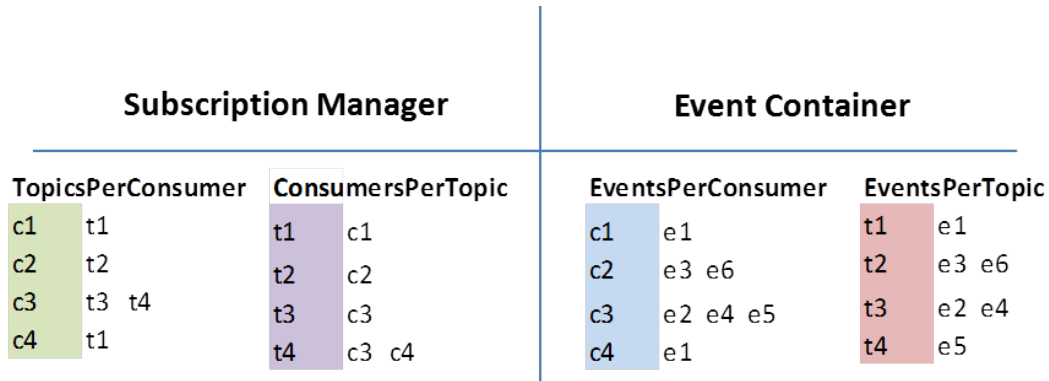


Figura 3.4. Ilustração das filas de subscrições e eventos em um sistema baseado em eventos.

elemento (e.g., tópico, consumidor, serviço, atividade, etc.). Em filas específicas, utiliza-se a fila de eventos E_t de tópico t para adicionar o novo evento gerado. Em filas globais, o sistema armazena todos os eventos publicados no sistema, independente da existência de subscrições para o tópico relacionado. Após o armazenamento dos eventos gerados, o sistema busca por consumidores subscritos interessados nos eventos (tópico, serviço, atividade), correspondendo à função f_{s3} na Figura 3.3. Agora com a lista de subscrições S_t ao elemento t sendo publicado, é possível obtermos a listagem dos consumidores subscritos ao mesmo. O sistema obtém as filas de eventos (E_{c1} , E_{c2} , ... E_{cn}), onde cada uma destas está associada a um consumidor c_i subscrito no tópico e adiciona o novo evento e_k em cada uma das filas. No final deste processo, os consumidores subscritos aos tópicos relacionados aos eventos gerados terão uma representação do evento adicionada em suas filas. O sistema utiliza as filas (E_{c1} , E_{c2} , ... E_{cn}) no momento do envio das notificações dos eventos publicados para os clientes, enviando assim para os clientes os eventos publicados nas filas respectivas dos consumidores. Uma ilustração destas filas existentes no sistema pode ser visualizada na Figura 3.4.

3.2.1 Algoritmos Relacionados

Esta subseção tem como objetivo apresentar a análise de complexidade dos algoritmos de publicação e notificação de eventos no sistema de eventos proposto. Neste trabalho, realizamos a análise em função do tempo e do espaço em função do número de consumidores inscritos em tópicos publicados, o número de eventos publicados e a quantidade de tópicos disponíveis para assinatura por consumidores. Para isto, utilizamos as seguintes variáveis: (1) n que representa o número de consumidores

inscritos em um tópico publicado, (2) m sendo o número de eventos publicados e (3) t sendo o número de tópicos disponíveis para assinatura. A Tabela 3.1 apresenta um resumo da análise de complexidade dos principais algoritmos usados no servidor de eventos.

Durante a assinatura e publicação de eventos, o sistema utiliza tabelas do tipo *hash* para armazenar quatro conjuntos de dados: elementos (serviços, tópicos, atividades) subscritos por consumidor, consumidores subscritos por elemento, eventos por consumidor e eventos por elemento. A utilização de tabelas do tipo *hash* possibilita um acesso de ordem constante em função do tempo ($O(1)$) às listas de eventos e de assinaturas presentes no sistema, facilitando assim o acesso às informações necessárias nos processos de publicação e notificação dos eventos gerados por produtores para consumidores.

3.2.1.1 Assinatura de Tópicos de Interesse

Apresentaremos nesta seção uma análise de complexidade dos dois principais algoritmos utilizados na assinatura de tópicos, serviços e atividades em sistemas baseados em eventos: a assinatura por tópicos e o cancelamento de assinaturas pelo usuário móvel. Nesta avaliação, definimos n como sendo o número total de consumidores existentes no sistema e t o número de elementos disponíveis para assinatura. Elementos em geral podem representar tópicos, serviços ou atividades de interesse. Na assinatura de tópicos por um determinado consumidor, o algoritmo terá complexidade constante ($O(1)$) em função do tempo.

Utilizando de uma tabela do tipo *hash* no processo de assinatura por elementos, o sistema fornece acesso a duas tabelas do tipo *hash* para a consulta e adição de novas assinaturas: consumidores por elemento e elementos subscritos por consumidores. A consulta por assinaturas possui complexidades diferentes para cada uma das operações a serem executadas sobre a estrutura de dados. No caso da consulta por consumidores subscritos em um elemento, o algoritmo apresenta complexidade constante ($O(1)$), já que utiliza a função *hash* para o acesso a lista de consumidores de um determinado elemento. O mesmo se aplica na consulta por elementos subscritos por consumidores, já que esta consulta também utiliza uma tabela do tipo *hash*. Na consulta por uma determinada assinatura, i.e., se um consumidor específico possui assinatura em um elemento, a complexidade será linear em função do tempo ($O(n)$), já que para cada um dos elementos na tabela *hash* podemos ter diversas assinaturas, sendo necessárias n operações de consulta em uma das duas tabelas *hash* existentes. Na adição de assinaturas, teremos um custo constante em função do tempo ($O(1)$) devido à

Algorithm SubscribeToTopic

Input: Consumer c and topic t
Output: Subscription s (c , t)
Sets:
 T_c : topics per consumer c
 C_t : consumers per topic t

	Times	Cost	Times x Cost	Space	Function Complexity	
					Time	Space
$T_c = \text{getSubscriptionsPerUser}(c)$	1	C_1	$1 \times C_1$	n	$O(1)$	$O(n)$
1 $C_t = \text{getSubscriptionsPerTopic}(t)$	1	C_2	$1 \times C_2$	n	$O(1)$	$O(n)$
2 $\text{create subscription } s \rightarrow (c, t)$	1	C_3	$1 \times C_3$	1		
3 $\text{add subscription } s \text{ to } T_c$	1	C_4	$1 \times C_4$	1	$O(1)$	$O(n)$
4 $\text{add subscription } s \text{ to } C_t$	1	C_5	$1 \times C_5$	1	$O(1)$	$O(n)$
5 $\text{return } s$	1	C_6	$1 \times C_6$	1		

$f(n) = C_1 + C_2 + C_3 + C_4 + C_5 + C_6$ **Total** **$O(1)$** **$O(n)$**
Time Space

Figura 3.5. Algoritmo de assinatura por elementos.

complexidade da função *hash* ($O(1)$) utilizada para a busca por cada uma das listas mais a adição a ser realizada sobre a lista ($O(1)$). Neste caso, é necessário adicionar consumidores e elementos (neste caso, tópicos, serviços ou atividades) em cada uma das duas filas ($O(1) + O(1) = O(1)$). O algoritmo e o cálculo de sua análise de complexidade podem ser visualizados na Figura 3.5.

Em função do espaço, o algoritmo de assinatura possui complexidade linear ($O(n) + O(t) = O(n)$). O algoritmo utiliza duas listas de elementos presentes nas duas tabelas *hash* no armazenamento dos consumidores e elementos subscritos. Na medida em que o sistema recebe novas assinaturas por elementos, cada uma das listas cresce de forma linear em função do espaço ($O(n)$). Sendo assim, considerando as duas listas, o custo computacional será linear, já que $O(n) + O(t) = O(2n) = 2O(n) = O(n)$.

No cancelamento da assinatura por um consumidor do sistema, a complexidade será linear ($O(n)$) em função do tempo e do espaço, respectivamente. Em função do tempo, através da utilização de uma tabela do tipo *hash*, o sistema fornece acesso a ambas as listas (i.e., consumidores por elemento e elementos subscritos por consumidores) com custo constante ($O(1)$) para a exclusão de uma assinatura com custo linear ($O(n)$) em função do tempo. Neste caso, é necessário remover consumidores e elementos em cada uma das duas filas ($O(1)$), assumindo que utilizamos um algoritmo

Algorithm UnsubscribeToTopic							
Input: Consumer c and topic t							
Sets:							
T_c : topics per consumer c							
C_t : consumers per topic t							
		Times	Cost	Times x Cost	Space	Function Complexity	
						Time	Space
1	$C_t = \text{getSubscriptionsPerTopic}(t)$	1	C_1	$1 \times C_1$	n	$O(1)$	$O(n)$
2	for each s in C_t	$n+1$	C_2	$(n+1) \times C_2$			
3	if $s.\text{consumer}$ equals c	$1 \times n$	C_3	$n \times C_3$			
4	remove s from C_t	$1 \times n$	C_4	$n \times C_4$			
5	$T_c = \text{getSubscriptionsPerUser}(c)$	1	C_5	$1 \times C_5$	n	$O(1)$	$O(n)$
6	for each s in T_c	$n+1$	C_6	$(n+1) \times C_6$			
7	if $s.\text{topic}$ equals t	$1 \times n$	C_7	$n \times C_7$			
8	remove s from T_c	$1 \times n$	C_8	$n \times C_8$			
$f(n) = C_1 + (n+1)(C_2 + C_6) + n(C_3 + C_4 + C_7 + C_8) + C_5$			Total	$O(n)$	$O(n)$		
				Time	Space		

Figura 3.6. Algoritmo de cancelamento de assinatura de elementos.

eficiente para a remoção de elementos de uma lista. O algoritmo e o cálculo de sua análise de complexidade podem ser visualizados na Figura 3.6.

3.2.1.2 Publicação de Um Evento

Neste trabalho, optamos por separar a publicação de eventos em duas partes: processamento e notificação. No processamento, o sistema recebe eventos de produtores e os armazena em filas específicas a fim de preparar a operação de notificação. Na notificação, o sistema cria mensagens em função dos eventos recebidos para serem enviadas a consumidores interessados. Estas mensagens serão então consumidas por consumidores (i.e., usuários móveis ou serviços Web) que informarão ao sistema a recepção destas. Sendo assim, o processo de publicação utiliza dois principais algoritmos, sendo cada um responsável pelas operações de processamento e de notificação de mensagens de eventos gerados. Esta divisão da operação de publicação foi meramente lógica, a fim de facilitar a implementação das duas partes no sistema.

Na publicação de eventos gerados, o algoritmo usado no sistema possui um custo linear ($O(n)$) em função do tempo. O processamento de eventos depende diretamente do número de elementos subscritos por consumidores relacionados ao evento sendo

Algorithm PublishEventInput: Event e with topic t

Sets:

 E_t : Events per topic t C_t : Consumers subscribed in E_c : Events per consumer c

		Times	Cost	Times x Cost	Space	Function Complexity	
						Time	Space
1	$E_t = \text{getEventQueueForTopic}(t)$	1	C_1	$1 \times C_1$	n	$O(1)$	$O(n)$
2	add event e to E_t	1	C_2	$1 \times C_2$			
3	$C_t = \text{getUsersSubscribedToTopic}(t)$	1	C_3	$1 \times C_3$	n	$O(1)$	$O(n)$
4	for each c in C_t do	$n+1$	C_4	$(n+1) \times C_4$		$O(n)$	$O(n)$
5	$E_c = \text{getEventsForConsumer}(c)$	$1 \times n$	C_5	$n \times C_5$	n	$O(1)$	$O(n)$
6	add event e to E_c	$1 \times n$	C_6	$n \times C_6$			
		$f(n) = C_1 + C_2 + C_3 + (n+1) C_4 + n C_5 + n C_6$		Total		$O(n)$	$O(n)$
						Time	Space

Figura 3.7. Algoritmo de publicação de um evento.

publicado. Neste processo, a busca pelas filas de eventos de cada consumidor subscrito pode ser realizada com um custo constante ($O(1)$) através do uso de tabelas do tipo *hash*. Em função do espaço, a ordem de complexidade também é linear ($O(m) + O(n) = 2O(n) = O(n)$), já que a quantidade de dados armazenados em cada entrada das tabelas crescerá linearmente em função do número de consumidores subscritos em diferentes elementos (i.e., tópicos, serviços e atividades). À medida que o número de publicações de eventos cresce, as filas de eventos por elemento e eventos por consumidor também cresce n vezes. Sendo assim, o número de eventos por consumidor aumenta na ordem $O(n) = O(n)$. O algoritmo e o cálculo de seu custo computacional podem ser visualizados na Figura 3.7.

Após o processamento dos eventos gerados por produtores, o sistema notifica consumidores sobre os eventos ocorridos. Durante a operação de processamento, o sistema armazena os eventos gerados em duas listas: (a) em função dos elementos de assinatura existentes e (b) em função dos consumidores com assinaturas nestes elementos. O objetivo neste caso é possibilitar a busca por eventos em função de um determinado elemento de assinatura e em função de consumidores cadastrados no sistema, otimizando assim o acesso aos eventos no processo de notificação em função do tempo.

Algorithm NotifyEventsForConsumerInput: Consumer c

Output: list of message notification for consumer

Sets:

 E : Notification messages to be sent for consumer E_c : Events published per consumer c

		Times	Cost	Times x Cost	Space	Function Complexity	
						Time	Space
1	$E = []$	1	C_1	$1 \times C_1$			
2	$E_c = \text{getEventsForConsumer}(c)$	1	C_2	$1 \times C_2$	n	$O(1)$	$O(n)$
3	for each e in E_c do	$n+1$	C_3	$(n+1) \times C_3$			
4	$n = \text{createNotificationMsg}(e, c)$	$1 \times n$	C_4	$(1 \times n) \times C_4$		$O(1)$	$O(1)$
5	add n in E	$1 \times n$	C_5	$(1 \times n) \times C_5$	n		
6	return E	1	C_6	$1 \times C_6$			
$f(n) = C_1 + C_2 + (n+1)C_3 + nC_4 + nC_5 + C_6$			Total	$O(n)$	$O(n)$		
				Time	Space		

Figura 3.8. Algoritmo de notificação de eventos para um determinado consumidor.

Algoritmo	Tempo	Espaço
Assinatura	$O(1)$	$O(n)$
Cancelamento de assinatura	$O(n)$	$O(n)$
Publicação de um evento	$O(n)$	$O(n)$
Notificação de um consumidor	$O(n)$	$O(n)$

Tabela 3.1. O custo computacional dos algoritmos utilizados no serviço de eventos.

Na notificação de eventos para um determinado consumidor, o algoritmo utilizado possui um custo linear ($O(n)$) em função do tempo e do espaço. Apesar de utilizarmos uma tabela do tipo *hash* no armazenamento das assinaturas, a complexidade continua sendo linear em função do tempo. Isto ocorre devido à necessidade do sistema buscar por cada evento publicado na fila do consumidor e criar uma mensagem de notificação para cada evento, dependendo assim do número de eventos gerados para um dado consumidor. Em função do espaço, o algoritmo utiliza duas listas: a lista de eventos para um consumidor e a lista de mensagens de notificação a ser enviada para o consumidor. Neste caso, ambas listas possuem um custo computacional linear ($O(n)$) em função do espaço. O algoritmo e o cálculo de seu custo computacional podem ser visualizados na Figura 3.8.

3.3 Arquitetura

O servidor de eventos proposto neste trabalho utiliza uma arquitetura cliente-servidor composta por clientes móveis e uma infra-estrutura de aplicações Web em nuvem que provê o servidor de eventos proposto neste trabalho. As interfaces de acesso ao servidor utilizam mensagens sobre o protocolo HTTP, permitindo que o usuário utilize os recursos do serviço tanto a partir de um dispositivo móvel quanto de um navegador Web disponível em um computador *desktop*. Através da conectividade de uma rede de dados WLAN, *General Packet Radio Service* (GPRS), *High-Speed Downlink Packet Access* (HSDPA) ou LAN, o usuário móvel acessa os recursos disponíveis no servidor de eventos disponível sobre uma infraestrutura de aplicações em nuvem. A Figura 3.9 apresenta uma visão geral da arquitetura utilizada pelo serviço.

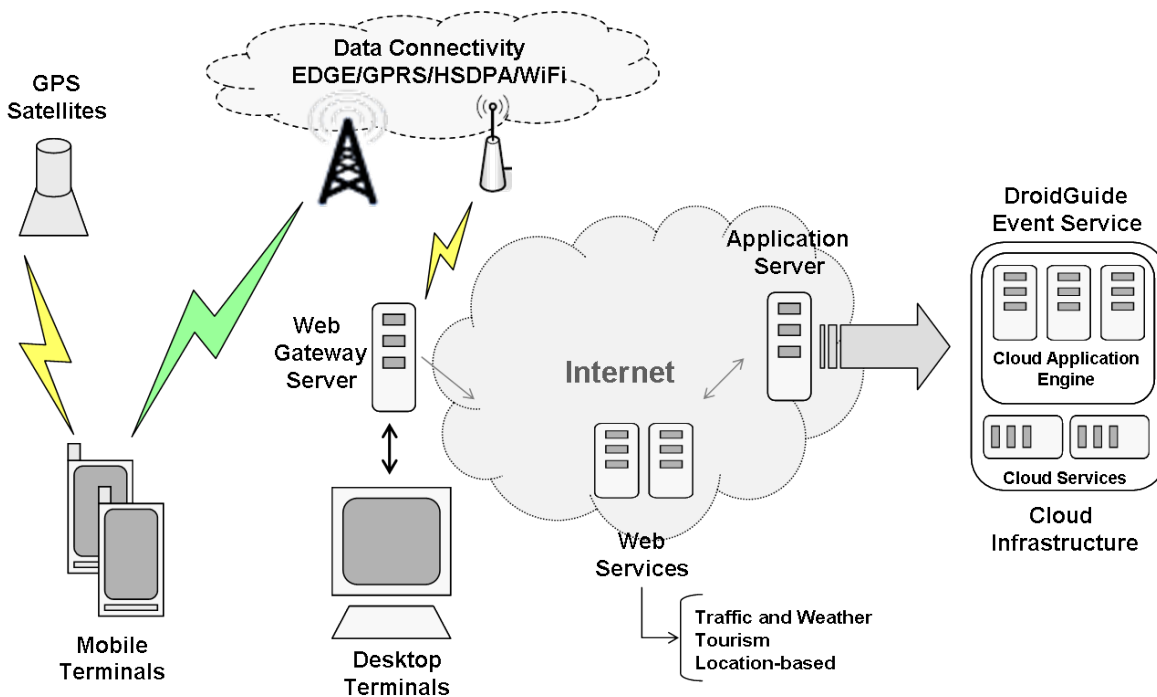


Figura 3.9. A visão geral do sistema de serviço de notificação de eventos para aplicações ubíquas.

Diferente da arquitetura distribuída proposta em Carzaniga et al [9], o sistema proposto neste trabalho utiliza uma solução onde clientes (*peers*) conectam a um único servidor de dados disponível em nuvem. Conforme apresentado em Carzaniga et al [9], uma das desvantagens no uso de uma solução centralizada está na incapacidade do tratamento de falhas no serviço de forma distribuída, caso ele falhe. Apesar disto, optamos por utilizar esta solução, devido ao fato dela simplificar tanto o processo de

desenvolvimento do serviço quanto o provimento da infra-estrutura no lado do servidor. A responsabilidade de montagem e gerenciamento dos recursos necessários reside sobre a própria nuvem, não sendo necessária uma criação da infra-estrutura servidora em *hardware* e *software*. A computação em nuvem propõe um estilo de computação onde recursos dinamicamente escaláveis e normalmente virtualizados são providos em forma de serviços sobre a Internet [58]. Desta forma, aplicações e componentes de infra-estrutura não residem localmente, mas em uma localização remota sobre a Internet, removendo assim a necessidade de usuários terem conhecimento ou experiência ou controle sobre a infra-estrutura da "nuvem" que os suporta.

Com o objetivo de separar as responsabilidades necessárias em um sistema de gestão de eventos especificamente para aplicações móveis, dividimos o serviço proposto neste trabalho em módulos com responsabilidades e relacionamentos bem definidos, conforme apresentado na Figura 3.10. Sendo assim, o servidor de eventos é composto pelos seguintes componentes agrupados no cliente e servidor, respectivamente:

- No dispositivo móvel:
 - **Interface gráfica do usuário ou GUI:** Apresenta ao usuário os recursos disponíveis pela aplicação móvel.
 - **Aplicações e serviços:** Aplicações e serviços disponíveis no dispositivo, tais como os gerenciadores de perfil e contexto (PCM), adaptação e escalonamento de atividades ao usuário móvel.
 - **O processador de eventos:** processa as informações vindas de aplicações e serviços, as envia para o servidor de dados e processa respostas enviadas do servidor.
- No servidor remoto de dados:
 - **O processador de eventos:** processa informações vindas de clientes móveis, compartilha os eventos gerados com serviços remotos interessados nestes.
 - **O *container* de eventos:** armazena os eventos gerados no sistema.
 - **O gestor de subscrições ou *Subscription Manager* (SM):** gerencia as subscrições por elementos (tópicos, serviços e atividades).
 - **O *container* de aplicações:** gerencia as aplicações presentes no servidor contendo as regras de negócio da aplicação.
 - **O *container* de serviços:** gerencia o cadastro e acesso à serviços remotos disponíveis para uso em clientes móveis.

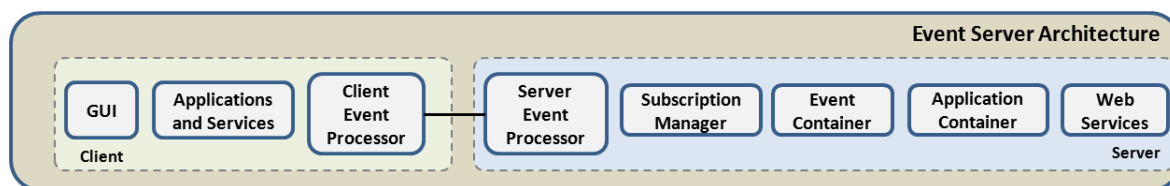


Figura 3.10. Principais componentes do servidor de eventos.

A arquitetura apresentada na Figura 3.10 define alguns componentes hipotéticos neste trabalho. Estes componentes foram implementados ou utilizados de uma forma abstrata, sem a utilização de dispositivos de *hardware* específicos. Por exemplo, entidades do sistema utilizam o serviço de mapas para o rastreamento de suas posições durante um trajeto. Podemos considerar este serviço como uma instanciação de um sensor de GPS (*hardware*) que coleta informações da posição do usuário móvel. Estes componentes "abstratos" foram propostos e implementados com o objetivo de cobrir requisitos do sistema no que diz respeito ao acesso a informações de contexto físico do ambiente em que o usuário móvel se encontra. Estes componentes fornecem informações que incluem a detecção de mudanças de dados a partir de sensores localizados no ambiente local e remoto, observando determinados tipos de dados de contexto. Exemplos de dados observáveis incluem a localização (dispositivo GPS), medidores de temperatura (termômetro), humidade e de luminosidade, dentre outros. Além destes, dispositivos móveis também podem coletar determinados tipos de dados observáveis através do uso de sensores disponíveis no próprio aparelho. Por exemplo, dispositivos móveis com um acelerômetro ou sensor GPS possibilitam coletar informações de inclinação do aparelho e de localização do usuário móvel.

3.3.1 Interfaces de acesso

Com o objetivo de padronizar o acesso aos principais recursos presentes no servidor de eventos proposto, as interfaces de acesso definem as principais funcionalidades dos componentes responsáveis por gerir os eventos coletados no sistema. Esta definição inclui o agrupamento e definição das principais operações responsáveis pela gestão de eventos (i.e., *container* de eventos e o servidor de eventos), de subscrições, de serviços Web e de aplicações. Destes quatro grupos, somente o pacote de aplicações utiliza interfaces específicas que variam de acordo com a aplicação em execução, tais como o *ActivityManager* usado no guia turístico *DroidGuide* e o gestor de emergências (*EmergencyManager*) usado no protótipo DECS. Além dos gestores específicos por

aplicação, o sistema utiliza gestores genéricos que provêm diversos serviços em comum, tais como os gestores de usuários, marcos e de contexto, independente da aplicação cliente em execução.

Utilizando uma visão *top-down*, componentes do servidor de eventos podem ser divididos em cinco camadas: (a) cliente, (b) gestão de eventos, (c) gestão de aplicações, (d) gestão de serviços e (e) serviços Web ou remotos. Na camada cliente, dispositivos com e/ou sem fio acessam o sistema por meio de uma rede de dados. Na camada de gestão de eventos, estão presentes os três principais componentes responsáveis pela gestão de eventos no servidor: o *container* de eventos, o servidor de eventos e o gestor de subscrições. Na camada de gestão de aplicações, componentes orientados por aplicações específicas fornecem serviços no nível de aplicação para clientes móveis e serviços remotos. Na camada de gestão de serviços, componentes fornecem a clientes móveis e serviços remotos o acesso à serviços Web sensíveis à mudanças em informações de perfil e contexto. A camada de serviços Web ou remotos fornece a instância dos próprios serviços acessíveis e também usufrui das informações de perfil e contexto coletadas a partir do dispositivo móvel. A Figura 3.11 apresenta uma visão macro das camadas e seus componentes que fornecem as principais interfaces que compõem o sistema. Uma visão detalhada das interfaces que compõem o sistema proposto pode ser visualizada na Figura 3.12.

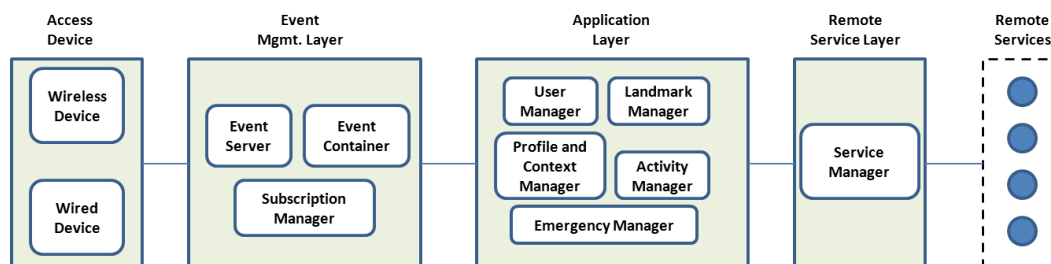
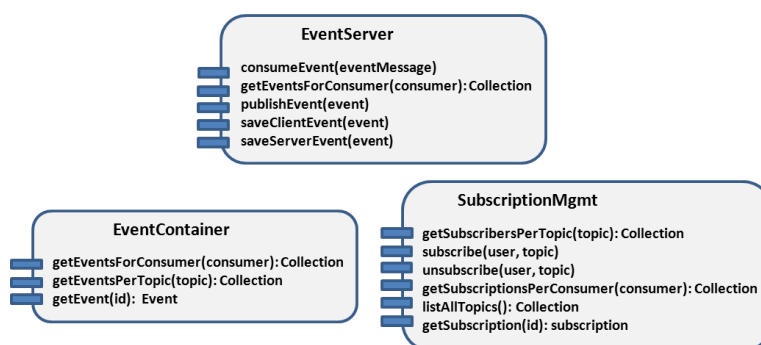


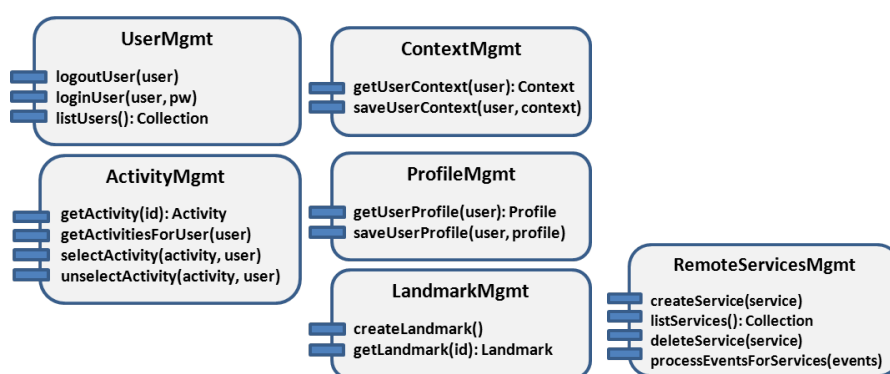
Figura 3.11. Visão macro das interfaces de acesso.

3.3.1.1 Gestão de Eventos

Conforme apresentado na Figura 3.12(a), a camada de gestão de eventos é composta por três principais componentes que armazenam e processam eventos originados de clientes móveis e de serviços remotos. O *container* de eventos armazena e fornece a consulta por eventos gerados no sistema em função de um determinado elemento (e.g. tópico, serviço e/ou atividade) e em função de cada consumidor subscrito neste elemento. O



(a) Gerenciamento de eventos.



(b) Gerenciamento de aplicações.

(c) Gerenciamento de serviços Web.

Figura 3.12. As interfaces providas pelos componentes propostos no servidor de eventos.

servidor de eventos provê funcionalidades de criação, publicação, notificação e consumo de eventos. No caso deste trabalho, a fim de separarmos o processamento de eventos vindos do cliente e do servidor, optamos por tratar as operações separadas na criação de eventos nestes dois pontos do sistema. As Tabelas 3.2 e 3.3 apresentam a composição das interfaces do *container* de eventos e o processador de eventos, respectivamente.

3.3.1.2 Assinatura em Eventos

Na parte de assinatura em eventos, o gestor de subscrições fornece acesso à operações de subscrita, listagem e consulta por assinaturas a partir de um tópico ou consumidor. O servidor de eventos utiliza boa parte destes métodos na publicação e notificação dos eventos gerados à clientes com subscrições associadas. No processo de assinatura em tópicos no sistema, o usuário móvel solicita ao servidor de eventos a listagem de tópicos subscritos por ele e/ou a listagem completa de tópicos disponíveis para assinatura.

Operação	Responsabilidades	Clientes
getEventsPerConsumer(Consumer c): Collection	Retorna uma lista de eventos relacionados à um determinado consumidor.	O processador de eventos no servidor.
getEventsPerTopic(String topic): Collection	Retorna uma lista de eventos relacionados à um determinado tópico.	O processador de eventos no servidor.
getEvent(Long id): Event	Retorna um evento específico.	O processador de eventos no servidor e clientes móveis.
listAllEvents(): Collection	Retorna a lista completa de eventos armazenados no container.	O processador de eventos no servidor

Tabela 3.2. As interfaces do processador de eventos e do *container* de eventos.

Operação	Responsabilidades	Clientes
getEventsForConsumer(Consumer c): Collection	Retorna a lista de notificações em função de eventos gerados para um determinado consumidor.	Clientes móveis
publishEvent(Event evt)	Fornece a funcionalidade de publicação de um evento ocorrido no sistema.	Serviços remotos e clientes móveis
saveClientEvent(Map params): ClientEvent	Cria um novo evento baseado em informações enviadas por um cliente móvel.	Clientes móveis
saveServerEvent(Map params): ServerEvent	Cria um novo evento baseado em informações enviadas por um serviço Web remoto.	Serviços remotos

Tabela 3.3. As interfaces do processador de eventos e do *container* de eventos.

A Tabela 3.4 apresenta os métodos que compõem a interface de acesso ao gestor de subscrições.

Operação	Responsabilidades	Clientes
getSubscribersPerTopic(String topic): Collection	Retorna a lista de subscrições em um determinado tópico.	O processador de eventos no servidor
getSubscriptionsPerConsumer(Consumer c): Collection	Retorna a lista de subscrições de um determinado consumidor.	O processador de eventos no servidor
subscribe(Map params)	Permite a subscrição em um determinado elemento a partir de parâmetros (Tópico, serviço ou atividade).	Serviços remotos e clientes móveis
unsubscribe(Map params)	Permite o cancelamento de uma subscrição existente em um determinado elemento a partir de parâmetros.	Serviços remotos e clientes móveis
listAllTopics(): Collection	Lista todos os tópicos disponíveis para subscrição.	Serviços remotos e clientes móveis
getSubscription(Long id): Subscription	Retorna uma subscrição específica no sistema.	Serviços remotos e clientes móveis

Tabela 3.4. As interfaces presentes no gestor de subscrições.

3.3.1.3 Gestão de Serviços Web

No acesso à serviços Web sensíveis à informações de perfil e contexto, o gestor de serviços fornece operações de criação, listagem e invocação de serviços remotos a partir do cliente móvel ou componentes residentes no lado do servidor, conforme apresentado na Figura 3.12(c). A listagem de serviços possibilita ao usuário móvel a subscrita por estes serviços no lado do servidor através do gestor de subscrições. A invocação permite a execução direta de um serviço a fim de prover informações sob demanda para o usuário móvel. O servidor de eventos também repassa os eventos gerados no lado cliente para os serviços de acordo com as subscrições existentes. Além destas funcionalidades, definimos algumas operações adicionais, tais como a criação de parâmetros, a fim de estendermos o uso de destes serviços via parametrização. Apesar da definição desta operação na interface, esta funcionalidade não foi implementada por completo no trabalho. A Tabela 3.5 apresenta as operações definidas na interface definida pelo gestor de serviços.

Operação	Responsabilidades	Clientes
createService(Map params)	Cria uma nova referência para um serviço remoto.	Provedores de serviços e usuários externos
getListOfServices():Collection	Retorna a lista de referências a serviços remotos disponíveis no sistema.	Clientes móveis
deleteService(Long id)	Deleta uma referência de um serviço remoto existente.	Provedores de serviços e usuários externos
processEventsForServices(Collection events):Collection	Repassa os eventos vindos de clientes móveis para as referências de serviços remotos de acordo com as subscrições definidas. Retorna uma lista de novos eventos caso algum serviço remoto necessite criá-los.	O processador de eventos no servidor
invokeService(Map params):Collection	Permite a invocação de serviços remotos via passagem de parâmetros. Retorna uma lista de valores retornados pelo serviço.	Clientes móveis
addServiceParameter(Map params)	Cadastra um parâmetro a ser associado a um serviço remoto. Parâmetros são usados na invocação de serviços.	Provedores de serviços e usuários externos
removeServiceParameter(Long id)	Remove um parâmetro associado a um serviço remoto.	Provedores de serviços e usuários externos

Tabela 3.5. As interfaces da camada de serviços disponíveis no servidor de eventos.

3.3.1.4 Gestão de Aplicações Remotas

A fim de prover as principais funcionalidades da aplicação no lado do servidor, a camada de aplicações define componentes responsáveis em gerenciar a lógica de negócio e o ciclo

de vida de aplicações e funcionalidades genéricas de sistema, conforme apresentado na Figura 3.12(b). Alguns exemplos de operações lógicas incluem a autenticação de usuários móveis, gestão de entidades e marcos geográficos definidos no sistema, dentre outras. No que diz respeito ao ciclo de vida da aplicação servidora, componentes de aplicação tais como o *EmergencyManager* e o *ActivityManager* compõem o serviço DECS e o *DroidGuide*, respectivamente. O *EmergencyManager* fornece diversas operações referentes à gestão de eventos de emergencia, enquanto o *ActivityManager* provê funcionalidades ligadas às atrações turísticas disponíveis para visitação. A Tabela 3.6 apresenta as principais interfaces que fazem parte dos componentes de aplicação do sistema.

3.3.2 Os Serviços Remotos

Os serviços Web baseados em informações de perfil e contexto apresentam recursos que fazem uso das informações de perfil e contexto coletadas em dois pontos iniciais: a partir do usuário móvel e a partir de provedores de informações de contexto global. A partir da publicação destas informações coletadas, estes serviços poderão reagir e publicar eventos em função de determinados estados ou condições de usuários, da aplicação móvel ou de dispositivos. Neste trabalho, assume-se que os serviços já estão disponíveis no servidor remoto de dados para uso por aplicações móveis. Estes serviços são acessíveis via processo de assinatura a partir de dispositivos móveis através da requisição pela listagem de serviços disponíveis no servidor de dados por usuários. Estes serviços podem ser acionados a partir de eventos gerados tanto no lado cliente quanto no lado servidor.

3.4 Comunicação entre Peers

No servidor de eventos proposto neste trabalho, a comunicação entre clientes é realizada através de requisições HTTP síncronas ou assíncronas utilizando *server request polling* e AJAX, respectivamente. No primeiro caso, por meio de uma requisição síncrona gerada no cliente móvel, o servidor de eventos utiliza uma requisição baseada em estilo *Push (request polling)*. Neste caso, o cliente envia periodicamente uma requisição ao servidor, solicitando-o por novas informações a serem notificadas. Caso haja informações de interesse a serem enviadas, o servidor as envia para o cliente, transmitindo as mensagens de notificação de eventos gerados no servidor de eventos. Este processo permite que clientes móveis recebam notificações periódicas a partir da aplicação em execução no servidor remoto de dados. No caso da comunicação por meio de requisições assíncronas,

Interface	Operação	Responsabilidades	Clientes
Usuário	login(String name, String pw)	Permite a autenticação do usuário no sistema.	Clientes móveis
Usuário	logout(String name)	Permite a saída do usuário do sistema.	Clientes móveis
Usuário	listUsers(): Collection	Lista os usuários conectados no sistema.	Clientes móveis
Perfil	getUserProfile(Long id): Profile	Obtém os dados de perfil do usuário.	Clientes móveis (<i>DroidGuide</i>)
Perfil	saveUserProfile(Map params)	Salva os dados de perfil do usuário.	Clientes móveis (<i>DroidGuide</i>)
Contexto	saveContext(Map params)	Salva os dados de contexto do usuário.	Clientes móveis (<i>DroidGuide</i>)
Contexto	getContext(Long id): Context	Obtém os dados de contexto do usuário.	Clientes móveis (<i>DroidGuide</i>)
Marcos	createLandmark(Map params)	Cria um novo marco no sistema.	Clientes móveis
Marcos	getLandmark(Long id): Landmark	Obtém um marco específico.	Clientes móveis
Marcos	getAllLandmarks(): Collection	Obtém a lista de todos os marcos existentes no sistema.	Clientes móveis
Atividades	getActivity(Long id): Activity	Retorna uma atividade específica.	Clientes móveis (<i>DroidGuide</i>)
Atividades	getActivitiesForUser(Consumer c): Collection	Retorna uma lista de atividades associadas a um usuário do sistema.	Clientes móveis (<i>DroidGuide</i>)
Atividades	selectActivity(Map params)	Permite o usuário selecionar uma atividade. A ser substituído pelo método subscribe do gestor de subscrições.	Clientes móveis (<i>DroidGuide</i>)
Atividades	unselectActivity(Map params)	Permite o usuário desmarcar uma atividade. A ser substituído pelo método unsubscribe do gestor de subscrições.	Clientes móveis (<i>DroidGuide</i>)
Emergências	createEmergency(Map params): EmergencyEvent	Cria um novo evento de emergência.	Clientes móveis (DECS)
Emergências	searchForClosestEMU(Map params): Collection	Retorna a UMA/EMU mais próxima ao evento de emergência.	Clientes móveis (DECS)
Emergências	updateEMUDispatchedToEvent(Map params)	Sinaliza que a UMA/EMU foi despachada para o evento.	Clientes móveis (DECS)
Emergências	updateEMUArrivedAtEvent(Map params)	Sinaliza que a UMA/EMU chegou ao local do evento.	Clientes móveis (DECS)
Emergências	updateEMUReadyToGoToLandmark(Map params)	Sinaliza que a UMA/MEU está preparada para se deslocar para o marco.	Clientes móveis (DECS)
Emergências	selectClosestLandmarkToEMU(Map params): Collection	Retorna o marco mais próxima ao evento de emergência.	Clientes móveis (DECS)
Emergências	updateEMUArrivedAtLandmark(Map params)	Sinaliza que a UMA/EMU chegou no marco.	Clientes móveis (DECS)
Emergências	updateEMUDispatchedToLandmark(Map params)	Sinaliza que a UMA/EMU foi despachada para o marco.	Clientes móveis (DECS)
Emergências	updateEmergencyEvent(Map params)	Atualiza informações do evento de emergência tais como o seu estado, localização, responsável, etc.	Clientes móveis (DECS)
Emergências	updateEmergencyTransferredToLandmark(Map params)	Sinaliza que a UMA/EMU transferiu o evento de emergência para o marco.	Clientes móveis (DECS)
Emergências	updateEMUCannotReachLocation(Map params)	Sinaliza que a Uma/EMU não conseguiu ser despachada para o local onde o evento ocorreu.	Clientes móveis (DECS)

Tabela 3.6. As interfaces da camada de aplicação disponíveis no servidor de eventos.

<i>JavaSE API</i> ¹	<i>Networking</i>	API padrão de comunicação em rede
<i>Jakarta HTTP Client</i> ²	<i>Commons</i>	Disponível no <i>Android</i> como uma API alternativa de comunicação em rede

Tabela 3.7. As APIs de comunicação utilizadas no protótipo *DroidGuide* na linguagem Java.

o cliente envia em *background* uma solicitação por informações disponíveis ou não no servidor. A grande vantagem na requisição assíncrona está no fato da aplicação poder continuar o seu processamento (interface gráfica, entrada e saída, etc.) enquanto envia as solicitações ao servidor, sem a necessidade de paralisar o processamento, como no caso da requisição síncrona.

Para o provimento da comunicação entre clientes móveis e o servidor remoto de dados, desenvolvemos um módulo de comunicação exclusivo para o protótipo do guia turístico *DroidGuide*. Este módulo de comunicação localizado no cliente é responsável pelo envio de requisições para o servidor remoto de dados e a recepção das respostas vindas do servidor para o cliente. Este módulo prepara o envio das requisições ao servidor remoto de dados através da criação de mensagens HTTP síncronas e assíncronas do tipo *GET*. Ele também é responsável por processar as respostas vindas do servidor de dados em formato XML (e.g., XML sobre HTTP) para serem consumidas pela aplicação móvel. Este módulo permite também uma abstração a nível de protocolo de comunicação para aplicações móveis na linguagem Java, permitindo que protocolos e/ou APIs de comunicação diferentes possam ser usados no lado do cliente.

Nos dois protótipos desenvolvidos neste trabalho, apenas o guia turístico *DroidGuide* utiliza o módulo de comunicação. Diferente do *DroidGuide*, o DECS não necessita de um módulo de comunicação próprio devido ao fato deste utilizar o próprio navegador Web para a transmissão dos dados via HTTP para o servidor. Desta forma, a comunicação é realizada de forma direta via requisições assíncronas em AJAX. Os processos de serialização e deserialização (e.g., conversão de dados entre o formato XML e objetos Java e vice versa) são realizados pelo módulo de comunicação no caso do *DroidGuide* e pela API de processamento de documentos XML fornecida pelo GWT no protótipo DECS. Apesar de utilizarmos diferentes mecanismos de comunicação nos protótipos, o processo de serialização e deserialização é idêntico, diferenciando apenas nas interfaces utilizadas em cada um dos protótipos. Esta semelhança possibilitou o reuso de algumas das classes e métodos de processamento de dados previamente implementadas no *DroidGuide* para uso no DECS.

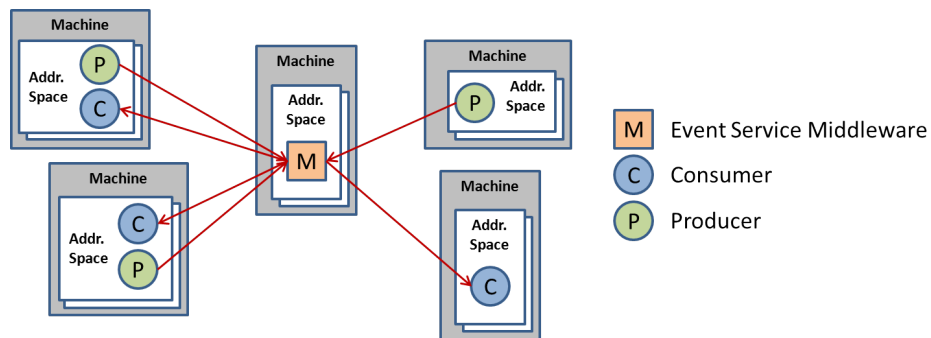
3.5 Classificação

Utilizando a taxonomia apresentada em Meier et al. [36], classificamos o servidor de eventos proposto neste trabalho a fim de compará-lo com outros sistemas relacionados. Em relação ao modelo de eventos proposto neste trabalho, o servidor de eventos utiliza um único mediador entre produtores e consumidores presentes no sistema. Ao contrário do servidor de eventos, outras implementações suportam o uso de mediadores múltiplos funcionalmente equivalentes (i.e., CORBA) ou diferentes tais como o SECO [24]. No que diz respeito à organização do serviço de eventos no sistema, o servidor de eventos utiliza uma organização distribuída e localizada em diferentes endereços em entidades do sistema, conforme apresentado na Figura 3.13(a). Como no contexto deste trabalho utilizamos clientes móveis acessando um único servidor remoto de dados, o serviço de eventos centralizado em um único ponto fornece os serviços de publicação e subscrição à eventos para clientes móveis localizados em diferentes endereços. No modelo de interação, o servidor de eventos define uma interação centralizada e intermediária entre as entidades do sistema. Produtores enviam eventos para o servidor e este os repassa para consumidores de acordo com as subscrições criadas, conforme apresentado na Figura 3.13(b). A Tabela 3.8 apresenta um resumo da comparação entre o servidor de eventos proposto neste trabalho e outros sistemas baseados em eventos classificados em Meier et al. [36].

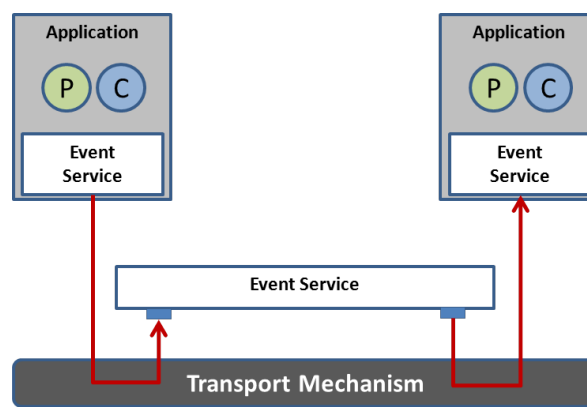
	CORBA	SIENA	SECO	Hermes	SdE
Event model	Single mediator or multiple, nonfunctionally equivalent mediators	Single or multiple mediators	Implicit	Multiple mediators	Single mediator
Event Service Organisation	Single or multiple distributed, separated middleware	Single or multiple distributed, separated middleware	Distributed, collocated middleware	Multiple distributed, separated middleware	Distributed, separated middleware
Event Service Interaction Model	Centralised intermediate or partitioned, distributed intermediate	Centralised intermediate or cooperative, distributed intermediate	No Intermediate, named (uSECO) or implicit (mSECO)	Cooperative, distributed intermediate	Single centralized intermediate

Tabela 3.8. Categorização do modelo e do serviço de eventos.

Avaliamos também alguns dos recursos funcionais do serviço proposto em relação à taxonomia proposta por Meier et al. [36]. No modelo de propagação de eventos, o servidor de eventos utiliza o padrão *push* periódico, onde consumidores e produtores periodicamente comunicam com o servidor de eventos a fim de consumirem ou publicarem eventos, respectivamente. Em comparação com as outras implementações,



(a) Organização



(b) Modelo de interação

Figura 3.13. Organização do serviço de eventos.

	CORBA	SIENA	SECO	Hermes	SdE
Event Propagation Model	Sporadic push and pull	Sporadic push	Sporadic push	Sporadic push	Periodic pull
Event Type	Typed	Typed	Typed	Typed	Typed
Expressive Power	Application specific attributes	Application specific attributes	Application specific object	Application specific object	Application specific attributes
Type Hierarchies	Omitted	Omitted	Omitted	Supported	Omitted

Tabela 3.9. Modelo de propagação e tipos de eventos suportados.

elas utilizam o *push* esporádico, onde eventos são propagados somente quando ocorrem mudanças de estado nas entidades. Em relação à tipagem de eventos, o servidor de eventos proposto utiliza eventos tipados utilizando atributos específicos de aplicação (e.g., nome, tipo e valor), sem utilizar uma hierarquia pré-definida. A Tabela 3.9 apresenta uma comparação entre alguns sistemas baseados em eventos referentes ao modelo de propagação e tipagem de eventos.

	CORBA	SIENA	SECO	Hermes	SdE
Location	Producer, consumer and intermediate	Intermediate	Producer and consumer	Intermediate	Intermediate
Definition	Constraint language	Constraint language	Programming language	Programming language	Programming language
Implementation	String	String	Object	Object	Object (server) and String (clients)
Evaluation mechanism	Implicit interpreted	Implicit interpreted	Implicit compiled	Implicit compiled	Implicit interpreted
Evaluation time	Propagation	Propagation	Propagation	Propagation	Propagation
Mobility	Static	Static and nomadic entity	Static	Static	Collaborative entity
Composite events	Omitted	Omitted	Omitted	Omitted	Supported programatically

Tabela 3.10. Recursos funcionais do serviço de eventos.

Em relação à filtragem de eventos, o servidor de eventos proposto utiliza o mediador como responsável pela filtragem de eventos, denominado de intermediário. Produtores e consumidores não possuem a capacidade de filtrar eventos, diferente de implementações como o CORBA [35] e o SIENA [9]. A definição de filtros é possível através da linguagem de programação e definição de objetos interpretados de forma implícita. Estes são utilizados diretamente no processamento de eventos durante a propagação, selecionando ou não eventos específicos de interesse para consumidores. Na mobilidade, o servidor de eventos utiliza entidades colaborativas, já que clientes móveis e serviços remotos baseados em informações de perfil e contexto podem publicar e consumir eventos gerados no sistema. Apesar destes se moverem, seus endereços são mantidos durante o ciclo de execução do sistema de eventos. A colaboração é possível através de envio de mensagens e publicação de eventos entre entidades conectadas ao servidor de eventos. Apesar do serviço desenvolvido neste trabalho não suportar diretamente eventos compostos, a flexibilidade no modelo de criação de objetos de eventos do sistema permite a criação de eventos compostos, permitindo o reconhecimento e notificação de determinados padrões gerados por produtores. A Tabela 3.10 apresenta alguns recursos funcionais presentes no servidor de eventos e em outros sistemas de eventos.

Meier et al. [36] apresenta algumas características de classificação de sistemas de eventos relativas à recursos não funcionais. Algumas destas incluem a confiabilidade, processamento em tempo real, suporte a prioridade, ordenação de eventos, segurança e gerenciamento de falhas. A Tabela 3.11 apresenta um resumo dos recursos não funcionais suportados pelo servidor de eventos e por outros sistemas de eventos. O servidor de eventos proposto utiliza a abordagem *best-effort* sem prioridade, onde não é possível associar um tempo limite e prioridades para o consumo de eventos no sistema. Devido ao fato do servidor de eventos utilizar o protocolo HTTP para comunicação

entre entidades e duas abordagens de armazenamento, o serviço de eventos possui uma conexão confiável e armazenamento temporário no caso do uso de tabelas do tipo *hash* e persistente no caso do uso de um SGBD. Eventos são ordenados na ordem cronológica de ocorrência para o envio a seus consumidores durante o processo de notificação. Conforme apresentado anteriormente, decidimos não fornecer recursos de segurança tais como a autenticação, autorização e criptografia na transmissão de dados entre entidades do sistema.

O recurso de armazenamento de eventos no servidor de eventos proposto oferece algumas configurações no que diz respeito à persistência e gerenciamento de filas. A característica *store occupancy* definida por Meier et al. [36] descreve requisitos de armazenamento necessários por um servidor de eventos durante seu tempo de execução, tais como o tamanho máximo em memória a ser usado. Este tamanho pode ser implícito ou configurável, dependendo dos requisitos apresentados por uma determinada aplicação. O armazenamento implícito impõe um tamanho máximo fixo de memória ou aloca o tamanho necessário de forma dinâmica, como por exemplo, através do uso de uma tabela do tipo *hash*. O armazenamento configurável utiliza determinados parâmetros, tais como o tamanho máximo das filas de eventos, quantidade máxima de produtores, consumidores e mediadores que podem ser suportados por um serviço de eventos. Além de suportar o armazenamento implícito, o servidor de eventos também permite o uso do armazenamento persistente, onde eventos são salvos em um SGBD caso a aplicação exija este tipo de persistência. A seleção do tipo de armazenamento (e.g., *hash* ou via SGBD) é configurável durante a inicialização do sistema de eventos. Por exemplo, nos casos de uso implementados, o *DroidGuide* utiliza o armazenamento persistente enquanto o DECS utiliza o armazenamento através de tabelas *hash*. A escolha por determinados tipos de armazenamento foi realizada apenas com o objetivo de avaliar ambas as abordagens. Neste caso, apesar da abordagem de armazenamento persistente ser mais confiável, ela possui um desempenho inferior do que a abordagem usando tabelas *hash*.

Além das características não funcionais apresentadas acima, Meier et al. [36] apresenta possíveis recursos no tratamento de falhas no sistema de eventos em diferentes níveis, tais como nas entidades, no *middleware* e na rede de dados. O servidor de eventos proposto pode ser caracterizado como falha parcial, já que na ocorrência de uma falha em uma de suas entidades, o sistema continuará funcionando como um todo. No caso de falhas no *middleware* responsável por gerenciar os eventos, a falha poderá ser parcial funcional e em alguns casos uma falha total, dependendo do tipo de falha ocorrido. A falha parcial funcional pode afetar determinadas partes do sistema, sendo estas divididas de forma geográfica ou funcional, isolando determinadas partes do sistema. O

	CORBA	SIENA	SECO	Hermes	SdE
Real time	Soft	Best effort	Best effort	Best effort	Best effort
Priority	Multiple	No	No	No	No
Store occupancy	Configurable	Implicit	Implicit	Implicit	Configurable
Reliability	Best effort, reliable connection or persistent	Best effort	Best effort (uSECO) or reliable connection (mSECO)	Reliable connection (temporarily) and then best effort	Best effort, Reliable connection and (temporarily) and persistent
Ordering	Any, FIFO, priority or deadline	Any	Any	Any	Any
Security	Omitted	Omitted	Omitted	Omitted	Omitted

Tabela 3.11. Recursos não funcionais do serviço de eventos.

	CORBA	SIENA	SECO	Hermes	SdE
Entity	Partial system failure	Partial system failure	Partial system failure	Partial system failure	Partial system failure
Middleware	Functional partial system failure or total system failure	Geographical partial system failure or total system failure	Results in failed entity	Geographical or functional partial system failure (temporarily)	Functional partial system failure or total system failure
Network	Partial system failure	Redundant or partial system failure	Partial system failure	Redundant or partial system failure	Partial system failure

Tabela 3.12. Recursos não funcionais do serviço de eventos.

servidor de eventos não provê suporte a mecanismos de redundância de falhas. Em casos de falhas no serviço de eventos, o servidor apresentará uma falha total, já que este utiliza uma arquitetura centralizada e mediadora no serviço de eventos. No caso da perda de conectividade de dados entre entidades e o serviço de eventos, o servidor terá uma falha parcial, já que os eventos não entregues a consumidores ficam armazenados até que as entidades restabeleçam um determinado tempo futuro a conectividade de dados com o servidor. Da mesma forma, eventos detectados por entidades desconectadas são armazenados no dispositivo móvel até a obtenção da conectividade de dados, sendo então enviados para o servidor de dados remoto. A Tabela 3.12 apresenta um resumo da categorização de sistemas de eventos em relação ao gerenciamento de falhas.

3.6 Ciclo de Vida

A aplicação móvel utiliza o servidor de eventos em três fases: (a) na captação e processamento de eventos internos no dispositivo, (b) captação de eventos externos no servidor de dados e (c) na entrega de mensagens de notificação a partir de serviços Web para aplicações e serviços interessados. Localizado no dispositivo móvel, o processador de eventos recebe as requisições de mudança nas informações de perfil e contexto, as processa, criando assim um conjunto de eventos a serem enviados ao servidor de

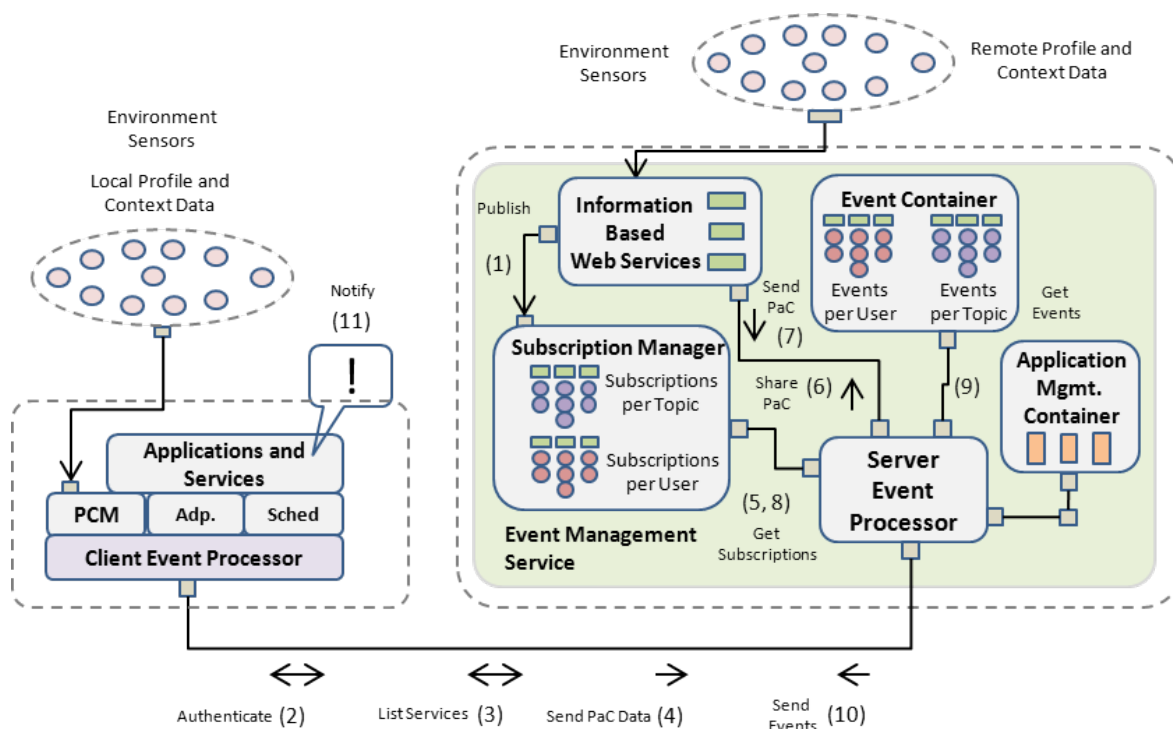


Figura 3.14. O fluxo de execução do processamento de eventos ocorrendo no dispositivo e no servidor de dados.

dados. Após o envio de eventos para o servidor, o serviço de processamento de eventos localizado no servidor de dados recebe estes eventos e os despacha para serviços Web interessados por receber eventos vindos do cliente. O diagrama descrito na Figura 3.14 apresenta a interação entre componentes no servidor e cliente no serviço proposto neste trabalho.

Para a disponibilidade de serviços remotos e atividades ao usuário móvel, é necessário o cadastramento destes elementos no servidor de eventos. Neste caso, entidades externas (usuários externos, provedores de serviços) cadastram os serviços remotos a serem disponibilizados pelo servidor de eventos. O servidor de eventos disponibiliza estes serviços para clientes no momento da assinatura por serviços e eventos de interesse. Após o cadastro, o sistema disponibiliza a assinatura por elementos à usuários móveis. O sistema proposto oferece dois modos de assinatura: o explícito e o implícito. A assinatura explícita é utilizada em casos onde o próprio usuário assina por elementos de seu interesse, como por exemplo, no guia turístico *DroidGuide*. No guia turístico, o usuário móvel solicita por listas de elementos tais como a de serviços móveis, de atividades e tópicos de eventos disponíveis no servidor de eventos, conforme mostrado na Figura 3.15. O usuário subscreve por elementos

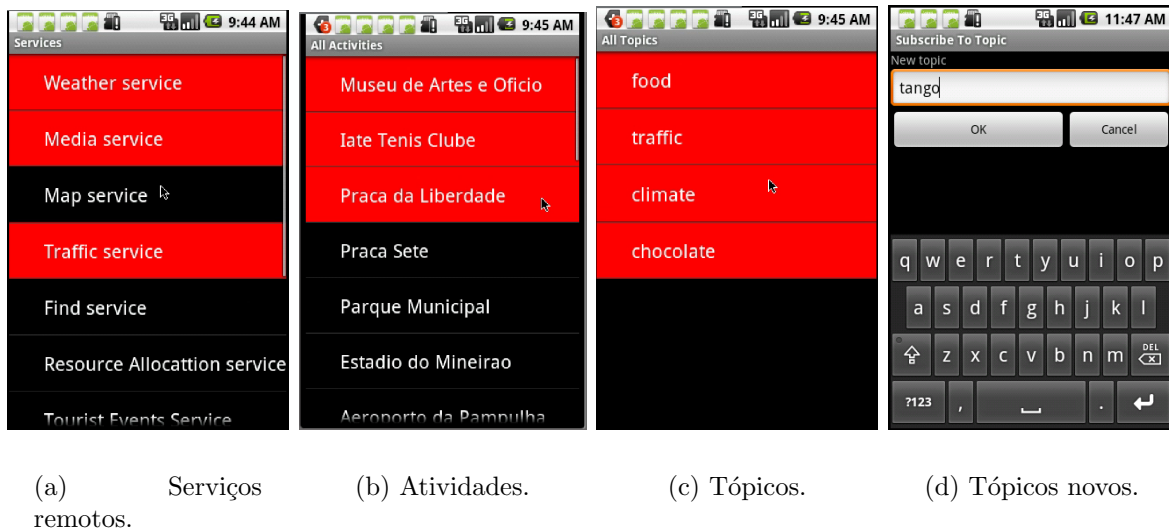


Figura 3.15. Subscrição por elementos de interesse pelo usuário móvel no *DroidGuide*.

em cada um destes grupos com o objetivo de consumir eventos, receber mensagens de notificação destes elementos e também disponibilizar suas informações de perfil e contexto lógico a eles. A assinatura implícita, entretanto, é realizada em aplicações onde aplicações e serviços clientes assinam diretamente a serviços remotos sem a necessidade de intervenção do usuário móvel. O DECS utiliza este modo de assinatura, já que os clientes móveis assinam por serviços e tópicos no momento da inicialização do serviço de emergencias.

O sistema disponibiliza a listagem de tópicos de eventos de interesse de uma forma diferenciada em relação à listagem de serviços e atividades, já que o próprio usuário móvel define os tópicos em que deseja receber informações de eventos publicados. Na assinatura explícita, a listagem de tópicos para assinatura é constituída a partir da criação de tópicos por cada um dos consumidores existentes no sistema. Neste caso, o primeiro consumidor logado no sistema não terá tópicos existentes para assinatura enquanto o último receberá a lista de todos os tópicos criados até o momento. No guia turístico *DroidGuide*, turistas podem obter a lista de tópicos disponíveis no sistema e assinar por novos tópicos ou existentes, conforme apresentado nas Figuras 3.15(d) e 3.15(c), respectivamente. Na assinatura implícita, clientes móveis assinam por tópicos específicos dos serviços a serem executados no sistema. No DECS, clientes móveis assinam por tópicos predeterminados que são utilizados durante a execução do serviço de contexto de emergências. Alguns tópicos utilizados incluem "unicast", "multicast" e "broadcast" usados para o envio de mensagens entre clientes móveis e "location" usado para representar as mudanças de posição dos clientes no serviço de emergências. No

caso do protótipo DECS, o cliente móvel poderá também publicar eventos baseados em um ponto específico no mapa geográfico apresentado na aplicação Web cliente, permitindo que este publique eventos em função de marcos ou acontecimentos ocorridos em determinadas localizações na região, como por exemplo, acidentes, lentidão no trânsito, dentre outras.

Na inicialização do sistema, o servidor de eventos solicita a autenticação do usuário móvel para que ele tenha acesso aos recursos disponíveis no sistema, tais como a listagem e assinatura em elementos (serviços remotos, atividades e tópicos disponíveis) e o consumo de eventos através de mensagens de notificação. No guia turístico *DroidGuide*, o usuário efetua a autenticação no sistema de forma manual, já que realizamos uma "emulação" a partir de uma aplicação móvel. No caso do protótipo DECS, a autenticação é realizada de forma automática para cada um dos clientes móveis que compõem o sistema de atendimento de emergências, já que neste caso realizamos uma "simulação" do serviço de contexto de emergências sobre uma determinada região. No guia turístico *DroidGuide*, caso o usuário já possua o perfil e contexto lógico definidos em sessões prévias, a aplicação móvel requisita estas informações para visualização e atualização por ele.

A partir de mudanças detectadas no ambiente onde o usuário e o dispositivo se encontram, o gerenciador de dados de perfil e contexto entra em ação no dispositivo móvel. Ele armazena cada uma das mudanças coletadas e periodicamente as repassa para o processador de eventos no dispositivo. O processador de eventos no lado cliente cria objetos representando cada um dos eventos coletados e os envia para o servidor de dados remoto para serem processados pelo servidor de eventos e armazenados no *container* de eventos no lado do servidor. Através do gerenciador de subscrições, o servidor de eventos busca por serviços Web interessados pelos eventos recebidos e armazenados do cliente. Caso haja serviços remotos subscritos pelo cliente, eles recebem os eventos relacionados às mudanças de perfil e contexto detectadas na aplicação móvel. Estes serviços remotos também podem ocasionar na criação de novos eventos em duas situações. A primeira delas envolve a necessidade destes em publicar mensagens para o cliente devido à mudanças de dados de perfil e contexto global. A segunda envolve a necessidade do envio de notificações para o cliente em função da mudança de estado nas informações do próprio serviço remoto, tais como a disponibilidade de um novo conteúdo a ser consumido pelo cliente móvel. Os serviços remotos enviam estes novos eventos para o servidor de eventos para processamento e armazenamento de acordo com os clientes subscritos nestes e em tópicos relacionados.

Na fase de retorno da requisição do servidor remoto para o cliente, o servidor de eventos busca por eventos que sejam de interesse do cliente a partir de elementos

Tabela 3.13. Visão geral dos protótipos desenvolvidos no trabalho.

Características	<i>DroidGuide</i>	DECS
Cenário de uso	Guia turístico eletrônico (atividades turísticas, conteúdo digital e serviços)	Controle de unidades móveis e fixas de emergência
Plataforma do cliente	<i>Google Android</i> e tecnologias Web (HTML, <i>JavaScript</i>)	Tecnologias Web (HTML, <i>JavaScript</i> e <i>Assynchronous JavaScript and XML</i>) sobre <i>Google Web Toolkit</i>
Plataforma do servidor	<i>Google Web AppEngine Framework</i> (Python)	<i>Google Web AppEngine Framework</i> (Python e JavaEE)
Interface de informações georeferenciadas	<i>Google Maps API</i> (GMaps API) e dados de GPS	GMaps API e dados de GPS
Informações gerenciadas	Informações sobre atividades turísticas, clima e tráfego, serviços Web	Informações de unidades móveis e de emergências, tráfego, seleção de marcos (hospitais e delegacias)
Número de classes implementadas	123 (1.0), 85 (2.0)	184
Linhas de código	12286 (1.0), 7497 (Cliente Android 2.0), 5504 (Servidor Android 2.0)	14195 (Cliente Web), 10053 (Servidor)

subscritos (serviços, atividades ou tópicos) pelo usuário móvel. Caso haja eventos a serem enviados ao cliente, o servidor de eventos os recupera do *container* e os envia na resposta para o cliente em formato XML. O processador de eventos no cliente recebe os eventos transmitidos do servidor de dados remoto, os transforma em objetos de mensagens de notificação e os despacha para componentes ouvintes, tais como aplicações do usuário e serviços em execução no dispositivo. No final, a aplicação consumidora apresenta a mensagem de notificação ao usuário em forma de aviso, para que ele possa tomar alguma decisão ou reação em função do evento ocorrido.

3.7 Implementações de Referência

Com o objetivo de fornecer exemplos de cenários de uso que utilizam o servidor de eventos proposto neste trabalho, desenvolvemos dois protótipos para fins de apresentação para implementações de referência: o primeiro focado em uma aplicação turística e o segundo na gestão de eventos de emergências. Ambos os protótipos se basearam nos cenários apresentados na seção 2.7 e têm como objetivo apresentar de forma concreta o uso do servidor de eventos em aplicações móveis, sua viabilidade, desafios e soluções propostas.

Apesar de ambos os protótipos utilizarem o servidor de eventos proposto neste trabalho, eles utilizam o servidor proposto utilizando tecnologias diferentes. O principal motivo desta abordagem foi a tentativa em avaliar novas tecnologias no desenvolvimento de aplicações móveis, possibilitando assim uma avaliação mais abrangente no que diz respeito a cada uma das tecnologias utilizadas. Enquanto o guia turístico utiliza um cliente móvel sobre a plataforma *Android*, o serviço de contexto de emergências foi

concebido a partir de uma aplicação utilizando tecnologias Web. Optamos por utilizar novos tipos de interfaces e clientes a fim de avaliar o serviço proposto neste trabalho. O capítulo 4 apresenta o guia turístico *DroidGuide* enquanto o capítulo 5 apresenta o serviço de contexto de emergências DECS. Nas seções 4.5 e 5.6, apresentaremos o mapeamento da arquitetura proposta na Figura 3.14 em função dos protótipos desenvolvidos neste trabalho.

Capítulo 4

Guia Turístico DroidGuide

Este capítulo tem como objetivo apresentar o primeiro estudo de caso do servidor de eventos proposto por este trabalho. Para este estudo, desenvolvemos um guia turístico eletrônico denominado de *DroidGuide*. Neste capítulo, apresentaremos a aplicação turística em forma de emulação que utiliza os recursos disponibilizados pelo servidor de eventos proposto.

4.1 Visão Geral

Esta seção apresenta uma visão geral do guia turístico eletrônico *DroidGuide* desenvolvido em forma de uma aplicação móvel. O *DroidGuide* utiliza uma arquitetura cliente/servidor composta de clientes móveis conectados a um servidor de dados. Os clientes móveis comunicam com o servidor através de mensagens de requisição/resposta utilizando o protocolo HTTP a partir do processador de eventos e do módulo de comunicação presentes na aplicação móvel. A comunicação tem como objetivo compartilhar informações de perfil e contexto do usuário móvel, atividades turísticas e de serviços Web para os diversos componentes do sistema. Ao receber as requisições vindas do cliente móvel, o servidor as processa e responde para o cliente através do envio de documentos XML sobre HTTP contendo diversas informações, tais como notificações geradas por serviços Web para o usuário, informações turísticas de perfil e contexto armazenadas, informações de contexto físico global, dentre outras.

Neste protótipo, assume-se que o dispositivo móvel possui capacidade de se conectar ao servidor utilizando um ponto de acesso em diferentes tipos de redes sem fio disponíveis (e.g., WLAN ou GSM/GPRS/EDGE/HSDPA). Desenvolvemos o lado cliente do *DroidGuide* utilizando a plataforma de software e sistema operacional *Android* baseado no *kernel* do Linux. No caso do servidor remoto de dados, utilizamos

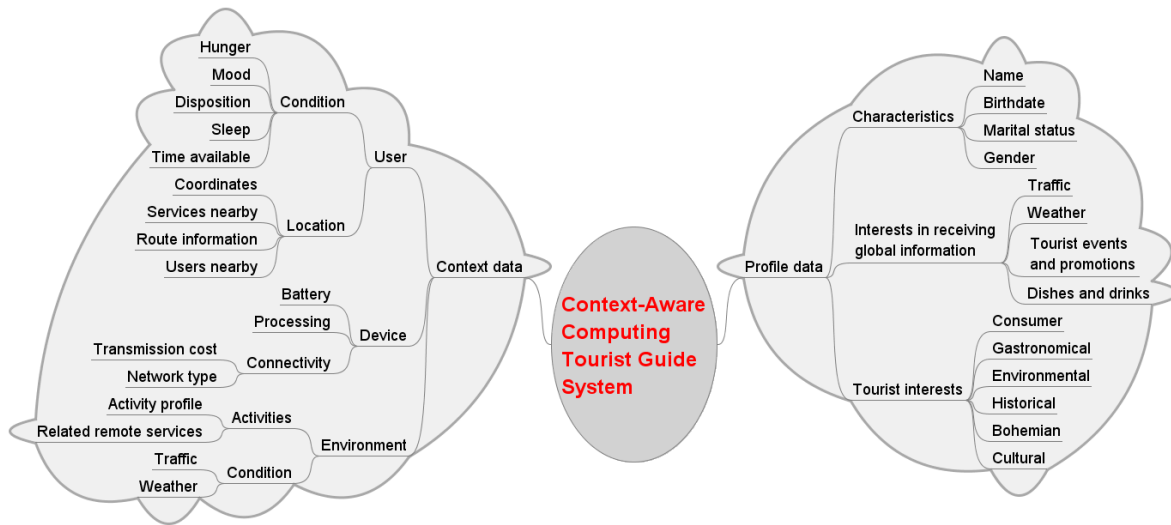


Figura 4.1. A taxonomia das informações de perfil e contexto do usuário definida para a aplicação turística.

a plataforma de desenvolvimento de aplicações Web GAE, executando sobre o ambiente computacional em nuvem nas linguagens *Python* e *Java*.

A fim de classificar as informações turísticas de perfil e contexto a serem coletadas pela aplicação móvel e servidor remoto de dados, definimos uma taxonomia de informações para o sistema, conforme demonstrado na Figura 4.1. Nesta taxonomia, definimos as duas principais classes de informações de perfil e contexto: informações locais ou individuais e informações remotas ou globais. As informações locais ou individuais representam estados e situações que estão próximas do usuário e do dispositivo, tais como a localização, interesses e perfil do usuário. As informações remotas ou globais representam estados ou situações mais distantes do usuário móvel e do dispositivo, tais como as condições climáticas, de tráfego e serviços disponíveis em função dos interesses e localização do usuário móvel.

O *DroidGuide* foi inicialmente desenvolvido como tema de trabalho prático na disciplina de computação ubíqua oferecida no segundo semestre de 2008 pelo Departamento de Ciência da Computação da Universidade Federal de Minas Gerais. No início do projeto, definimos algumas características em relação ao projeto, dentre estas o local a ser explorado pelo turista, a definição dos protótipos de interfaces gráficas de acesso à aplicação e a arquitetura de sistema da aplicação. Em relação ao local escolhido, optou-se por utilizar a região da Lagoa da Pampulha em Belo Horizonte por oferecer atrações turísticas e conteúdo acessível e disponível para acesso por aplicações móveis, tais como texto, fotos e vídeos de atrações turísticas. Desta forma, criou-se um

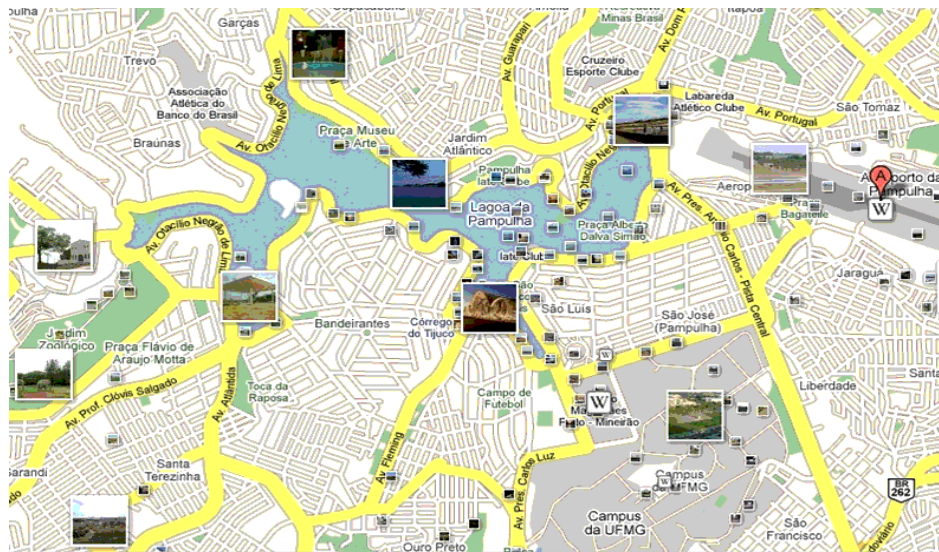


Figura 4.2. O mapa georeferenciado da região da Pampulha utilizado no protótipo *DroidGuide*.

mapa geográfico da região utilizando o serviço *Google Maps* (*Google Maps*)¹ em forma de interface de navegação para acesso aos serviços da aplicação pelo turista. Uma visão macro deste mapa pode ser visualizada na Figura 4.2.

Com a disponibilidade de versões mais recentes das tecnologias utilizadas neste trabalho, desenvolvemos uma nova versão contemplando as atualizações da API 2.0 do *Android*, suporte à GMaps API e a API do JavaEE pelo GAE. Nesta atualização, fomos capazes de desenvolver uma quantidade maior de testes unitários no lado do cliente e no servidor a fim de garantir a implementação das funcionalidades da aplicação e melhorar a qualidade dos serviços oferecidos ao usuário móvel. Algumas funcionalidades previstas e não implementadas na versão 1.0 foram também incorporadas neste nova versão, tais como a assinatura em atividades turísticas e tópicos, a sugestão de atividades turísticas em função dos interesses do turista definidos em seu perfil no cliente *Android* e recursos de pesquisa por serviços em função da localização do usuário móvel. As classes utilitárias (i.e., utilidades do *Android* e processamento de documentos em XML) e algumas definições de leiaute de telas na versão 1.0 foram reaproveitadas nesta nova versão no lado cliente.

¹<http://code.google.com/apis/maps>

4.2 Módulos do DroidGuide

Como o objetivo de simplificar o desenvolvimento e facilitar a integração das diversas funcionalidades do sistema proposto, dividimos logicamente o *DroidGuide* em módulos com responsabilidades e relacionamento entre si bem definidos, conforme apresentado na Figura 4.3. Em geral, cada módulo é composto por duas principais partes: uma localizada no dispositivo móvel e outra no servidor de dados. Uma listagem dos principais módulos da aplicação podem ser visualizados na Tabela 4.1.

Módulo	Descrição	Composição
Aplicação	Responsável por gerenciar as principais telas, menus, comandos e navegação entre as telas da aplicação cliente	Telas de login (Figura 4.4(a)), menus principal (Figura 4.4(b)) e secundários (Figura 4.6(a)), mapas (Figura 4.4(c)), e configuração (Figura 4.5(c))
Informações de Perfil e Contexto (PCM e PaC)	Responsável por gerenciar as informações, interesses, estado e condição do usuário e	Tela de perfil do usuário, conforme apresentado na Figura 4.5(a) e Tela de contexto lógico, conforme apresentado na Figura 4.5(b)
Comunicação	Provê a comunicação dos módulos presentes no cliente e no servidor, serialização e deserialização no envio e recepção dos dados	Processador de requisições e de documentos XML
Escalonador de atividades (AS e Activities)	Responsável em sugerir atrações turísticas baseadas nas informações de perfil e contexto do usuário móvel	Tela de sugestão de atividades turísticas, conforme apresentado na Figura 4.11(a)
Processador/servidor de eventos (ES/P)	Responsável por obter informações de mudança nas informações de perfil e contexto local	
Servidor de eventos (EMS)	Responsável por notificar o usuário de mudanças nas informações de contexto remotas e de serviços remotos	Telas de notificação e processador de informações de perfil e contexto, conforme apresentado nas Figuras 4.6(b) e 4.6(c)
Adaptação de conteúdo (Adp)	Responsável em apresentar o conteúdo ao usuário móvel baseado nas informações de perfil e contexto.	Telas de mapas com conteúdo, conforme apresentado nas Figuras 4.4(d), 4.4(e) e 4.4(f)

Tabela 4.1. Módulos que compõem o guia turístico *DroidGuide*.

Dos módulos apresentados acima, podemos destacar alguns destes devido à sua relevância neste trabalho: o módulo de aplicação, o de comunicação e o servidor de eventos. O módulo de aplicação define o relacionamento funcional entre os outros módulos do sistema no lado cliente e permite que o usuário interaja com as funcionalidades disponíveis no *DroidGuide*. O servidor de eventos possui a sua importância já que ele oferece informações relativas à mudanças nas informações de perfil e contexto do usuário móvel, da aplicação e de eventos externos relativos às atividades turísticas. O módulo de comunicação permite interligar os componentes localizados no cliente com os serviços disponibilizados no lado servidor, oferecendo um canal de transferência de objetos e mensagens entre os dois extremos da aplicação.

No protótipo desenvolvido, o servidor de eventos capta mudanças nas informações de perfil e contexto da aplicação através da definição de ouvintes nas entidades

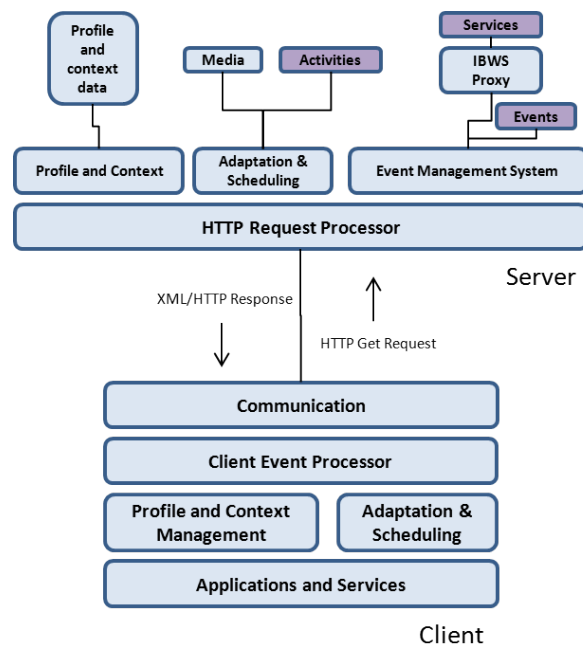
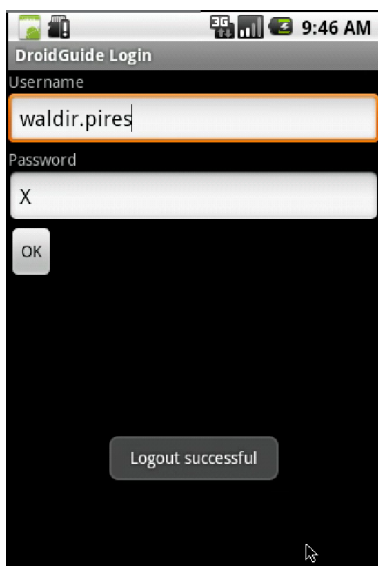


Figura 4.3. Arquitetura do protótipo *DroidGuide*.

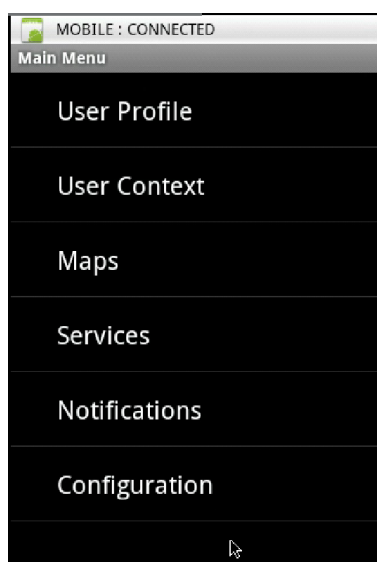
responsáveis em armazenar estas informações na aplicação. A aplicação móvel notifica ao processador de eventos do dispositivo as mudanças detectadas, que incluem condições lógicas do usuário móvel, mudança de interesses em receber notificações relativas ao clima, tráfego, serviços turísticos e mudanças no perfil turístico. As informações de perfil e contexto são também armazenadas no servidor de dados de tal forma que, ao logar em um momento futuro, a aplicação movel terá disponível o último estado de perfil e contexto do usuário. Um exemplo de requisição de informações de contexto do usuário pela aplicação móvel pode ser visualizado na Figura 4.7.

O PCM definido na aplicação móvel é responsável por gerenciar os interesses e estados do usuário móvel. Dividimos neste trabalho os interesses em seis principais áreas, conforme apresentadas na taxonomia na Figura 4.1: *Bohemian*, *Cultural*, *Gastronomical*, *Historical*, *Environmental* e *Consumer*. A partir destas categorias, classificamos as atrações turísticas disponíveis no sistema em cada uma das áreas apresentadas através de uma pontuação entre zero (nenhuma relação) e dez (fortemente relacionado). Desta forma, com a definição do perfil turístico do usuário, o sistema é capaz de propor ao usuário móvel sugestões de atrações turísticas na região que melhor se encaixam com o seu perfil.

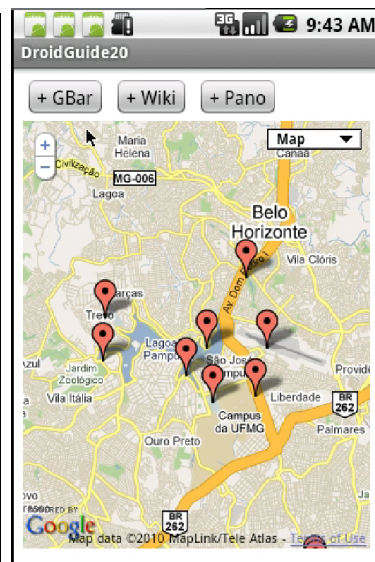
Um exemplo de sugestão de atividades turísticas pode ser visualizado na Figura 4.11(a). O usuário móvel em questão possui um perfil definido conforme a Figura



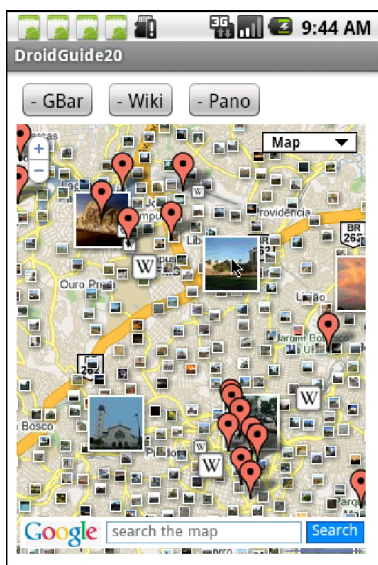
(a) Login.



(b) Menu principal.



(c) Mapa de atividades.



(d) Mapa com conteúdo.



(e) Mapa com conteúdo 2.



(f) Mapa com pesquisa.

Figura 4.4. Telas do módulo de aplicação de *login*, menu principal e telas de mapas.



(a) Perfil de usuário.

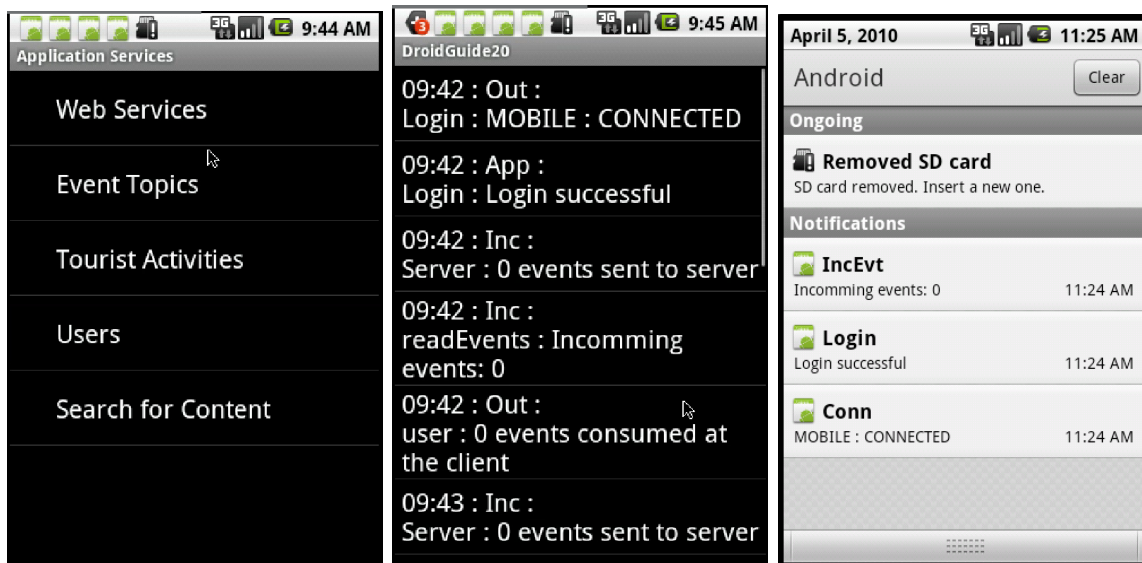
(b) Contexto lógico.

(c) Configuração.

Figura 4.5. Telas do módulo de perfil (modo estendido), contexto lógico e de configuração.

4.5(a). Neste perfil, podemos notar a relevância de dois estilos turísticos: *environmental* e *gastronomical*. A partir do processo de seleção de atrações turísticas, o sistema sugere atividades que se encaixam exatamente com o perfil do usuário ou oferece sugestões mais flexíveis, conforme apresentado neste caso, a seleção de atividades com uma pontuação superior ao valor sete. Conforme a disponibilidade e pontuação de atrações turísticas no sistema, o usuário recebe uma lista de atividades em que poderá se inscrever para seu entretenimento e também receber eventos relacionados à estas.

Similar ao que foi realizado no perfil, definimos os estados do contexto lógico do usuário móvel como sendo: *Sleep*, *Mood*, *Explore*, *Time* e *Hunger*, conforme apresentado na Figura 4.5(b). A aplicação móvel utiliza cada um destas características para representar o estado atual do usuário móvel. Na mudança de uma destas, por exemplo, o processador de eventos localizado no dispositivo móvel envia para o servidor de eventos um evento representando a mudança detectada, repassando-o para os serviços Web subscritos pelo usuário móvel. Por exemplo, na detecção de um estado de "fome" do usuário, o sistema permite que serviços Web ofereçam sugestões de restaurantes e lanchonetes na proximidade em que o usuário se encontra naquele instante. Sendo assim, o usuário móvel é responsável por informar à aplicação seu atual estado nas



(a) Menu principal de serviços.

(b) Notificação.

(c) Aba de notificações.

Figura 4.6. Telas do módulo de servidor de eventos, apresentando as mensagens de comunicação, a barra de notificações e as mensagens de notificação recebidas a partir do servidor de dados.

cinco características destacadas, permitindo assim que a aplicação detecte mudanças nas informações de contexto e informando ao servidor de eventos os eventos coletados.

4.3 Execução da Aplicação Móvel

A partir da inicialização da aplicação pelo usuário no dispositivo, o sistema inicia a captura e o envio de eventos para o servidor de eventos localizado no servidor de dados remoto. Na primeira execução da aplicação, o usuário define suas informações de perfil e contexto para serem utilizadas pelo servidor. Após a definição dos interesses definidos no perfil, a aplicação móvel estará preparada para acionar o escalonador de atividades no servidor de dados a fim de sugerir atividades turísticas que melhor se enquadram com os interesses do turista. O escalonador de atividades no lado cliente é responsável por apresentar as atividades sugeridas ao usuário e gerenciar a execução de cada uma das atividades na ordem definida ou pelo usuário ou pelo próprio escalonador localizado no servidor. O usuário móvel também assina por serviços Web disponíveis no servidor, possibilitando assim o uso destes durante o seu trajeto turístico. Através do servidor de eventos proposto neste trabalho, os serviços Web são capazes de enviar mensagens de notificação para o dispositivo móvel através da criação de eventos relacionados aos

Request

```
http://droidguide.appspot.com/context/process?operation=get&droidGuideUser_key=agpkcm9pZGd1aWRlchUL E g5Ecm9pZEd1aWRlVXNlchigJAw
```

Response

```
<droidguide-message datetime="2008-12-09 10:43:57.116770">
  <entity kind="UserContext" key="agpkcm9pZGd1aWRlchUL E g5Ecm9pZEd1aWRlVXNlchigJAw">
    <key>
      tag:droidguide.gmail.com,2008-12-09:UserContext [agpkcm9pZGd1aWRlchUL E g5Ecm9pZEd1aWRlVXNlchigJAw]
    </key>
    <property name="datetime" type="gd:when">2008-12-09 10:43:57.167082</property>
    <property name="droidGuideUser key" type="string">
      agpkcm9pZGd1aWRlchUL E g5Ecm9pZEd1aWRlVXNlchigJAw</property>
    <property name="hunger" type="int">0</property>
    <property name="mood" type="int">3</property>
    <property name="provision" type="int">2</property>
    <property name="sleep" type="int">2</property>
    <property name="time" type="int">3</property>
  </entity>
</droidguide-message>
```

Figura 4.7. Informações de contexto sendo requisitadas do servidor de eventos pela aplicação móvel.

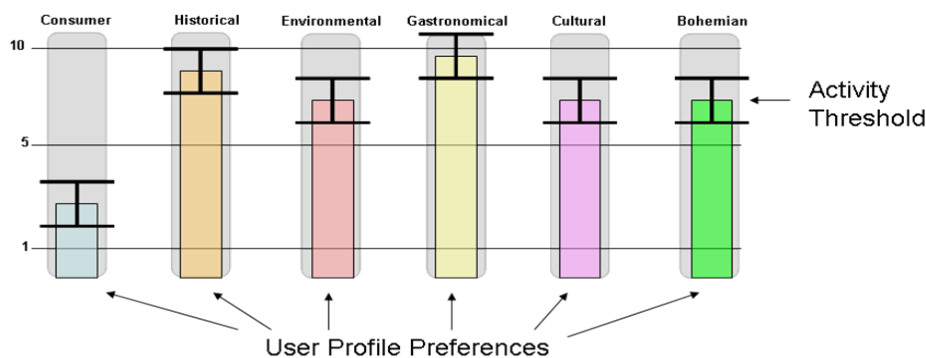
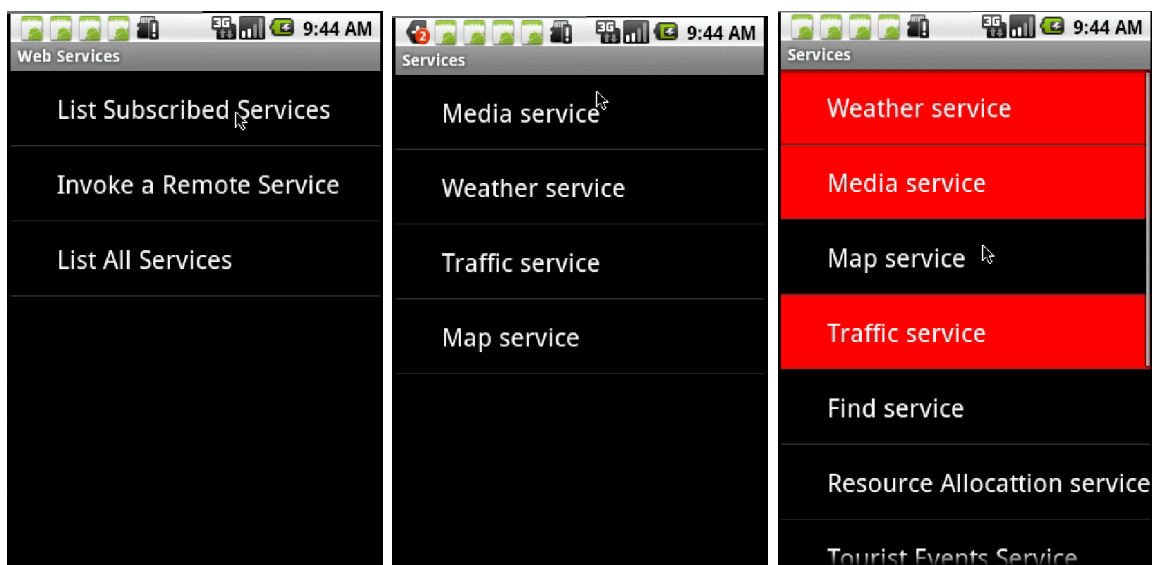


Figura 4.8. Categorização e seleção de atividades turísticas realizadas pelo escalonador de atividades.

interesses e estados do usuário móvel no âmbito global.

4.3.1 Seleção de Serviços Web

Após a definição das informações do perfil e contexto lógico, o usuário requisita pela lista de serviços Web disponíveis pelo servidor de dados para assinatura, conforma apresentado na Figura 4.9(a). Estes serviços utilizam as informações de perfil e contexto do usuário para a notificação de eventos ocorridos local e remotamente detectados pela aplicação e/ou servidor de dados. Ao subscrever em serviços localizados no servidor, o usuário móvel estará apto a receber notificações destes serviços em função dos interesses e estados definidos no perfil e contexto da aplicação móvel. O sistema permite que a assinatura seja feita de forma individual (e.g., por serviço) ou múltipla (e.g., mais de



(a) Menu de serviços remotos.

(b) Serviços remotos subscritos.

(c) Todos os serviços remotos.

Figura 4.9. Acesso e subscrição de serviços Web sensíveis em informações pela aplicação móvel.

um serviço), conforme demonstrado na Figura 4.9(b). Após a assinatura em serviços Web acessíveis no servidor de dados, o usuário móvel estará apto a compartilhar suas informações de perfil e contexto para estes serviços, além de receber notificações de eventos ocorridos nestes serviços em função de mudanças no contexto global.

4.3.2 Seleção de Atrações Turísticas

Além da seleção e assinatura em serviços Web, o *DroidGuide* permite a consulta, seleção e sugestão de atividades turísticas de interesse, conforme apresentado na Figura 4.11. Nesta funcionalidade, a aplicação apresenta a listagem completa de atividades disponíveis no sistema para consumo, as atividades sugeridas de acordo com o perfil do usuário definido e as atividades já subscritas pelo usuário móvel. Neste caso, a assinatura em atividades é realizada da mesma forma em que o usuário subscrive por serviços Web e tópicos ou canais de interesse em eventos. Neste caso, utilizamos tópicos para representar a assinatura em atividades, permitindo também o consumo de eventos relacionados à estas atividades por consumidores no sistema.

A sugestão de atividades turísticas tem como principal objetivo auxiliar o turista no consumo de atrações turísticas que melhor se enquadram com o seu perfil e contexto. Após o usuário móvel definir seus interesses na aplicação móvel a partir do perfil (Figura



(a) Menu de atividades.

(b) Atividades subscritas.

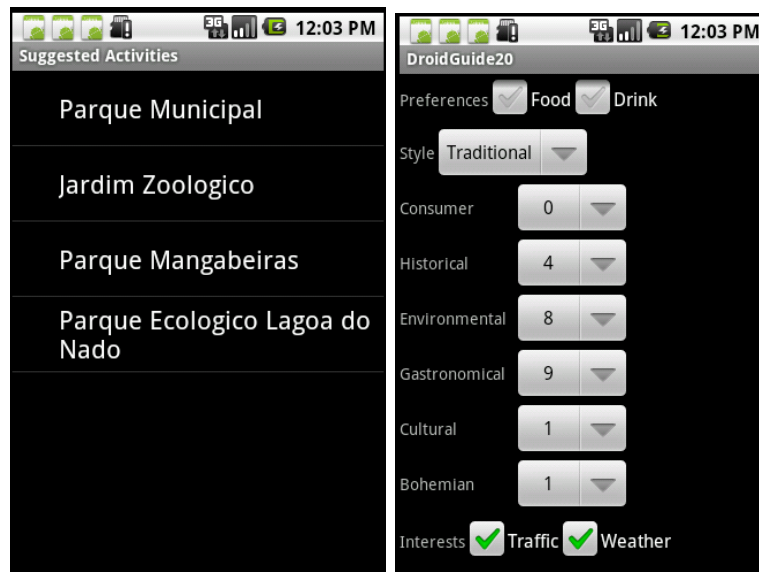
(c) Todos as atividades.

Figura 4.10. Funcionalidades de seleção de atividades turísticas.

4.5(a)), o sistema fornece uma sugestão de atividades turísticas baseadas nas categorias pré-definidas e na pontuação de cada uma. A Figura 4.11(a) apresenta uma sugestão enviada do servidor de dados à aplicação móvel.

Na seleção de atividades em função das características do perfil do usuário, a aplicação funciona da seguinte forma. Atividades turísticas foram categorizadas utilizando uma "assinatura" que representa os seis interesses definidos no *DroidGuide*. A assinatura é uma composição dos valores representando cada um dos interesses do usuário, como por exemplo, uma assinatura p_j de "[2, 6, 8, 5, 8, 4]". Inicialmente, o escalonador de atividades localizado no servidor remoto de dados busca pela assinatura do perfil do usuário. Após obter a assinatura do perfil, o sistema compara cada uma dos interesses com a listagem das atividades turísticas disponíveis para o turista. Para cada uma das atividades turísticas cadastradas, o sistema executa uma verificação a fim de encontrar atividades turísticas que estejam mais próximas dos interesses do usuário móvel. Caso haja atividades a serem sugeridas ao turista, estas são inseridas em uma lista de sugestões a ser apresentada para o mesmo, conforme demonstrado na Figura 4.11(a) e Tabela 4.3.2. O usuário poderá então aceitar a sugestão através da assinatura na mesma, da mesma forma que realizado em serviços Web e tópicos de interesse.

A busca por atividades turísticas de interesse pode também ser customizada a fim de flexibilizar ou não a seleção de atividades. Dependendo da rigidez na seleção, um limite (e.g., *threshold*) diferencial é definido para a nota de cada um



(a) Atividades sugeridas.

(b) Perfil do usuário.

Figura 4.11. Funcionalidades de sugestão de atividades turísticas.

Signatures	Boh	Cult	Gast	Hist	Ecol	Cons	Selected
User Profile (U1)	2	6	8	5	8	4	-
Activity 1 (A1)	3	5	9	7	7	5	Yes
Delta (U1 - A1)	1	1	1	2	1	1	-
Activity 2 (A2)	5	1	4	3	7	8	-
Delta (U1 - A2)	3	5	4	2	1	4	No

Tabela 4.2. Verificação de assinaturas de atividades turísticas com os interesses do usuário ($threshold = 2$).

dos interesses definidos. Por exemplo, suponhamos uma atividade turística a_j com assinatura "[3, 5, 9, 7, 7, 5]" para cada um dos interesses disponíveis. Definindo um threshold de dois, o sistema calcula a diferença absoluta para cada um dos interesses durante a verificação e, neste caso, o limite não pode ser ultrapassado para a seleção da atividade. Caso algum dos interesses ultrapasse o limite, o sistema não selecionará a atividade para o turista. Neste exemplo, o sistema selecionaria a atividade a_j como sugestão para o usuário com o perfil p_j , conforme apresentado na Figura 4.3.2. Os algoritmos de seleção de atividades e da verificação da assinatura de perfil podem ser visualizados nas Figuras 4.12 e 4.13, respectivamente.

O algoritmo apresentado na Figura 4.13 é responsável pela verificação do perfil de atividades turísticas em função do perfil do usuário. Esta verificação tem como objetivo informar se uma determinada atividade turística se enquadra nos interesses do usuário móvel, possibilitando assim a sugestão de atividades turísticas que melhor se enquadram com os interesses do turista. A seleção de atividades turísticas pode ser

Algorithm GetActivitiesForUser		Times	Cost	Times x Cost	Space
Input: User u					
Output: list of activities					
1	$p_u = \text{getUserProfile}(u)$	1	C_1	$1 \times C_1$	
2	$L = []$	$n+1$	C_2	$(n+1) \times C_2$	
3	$A = \text{getAllActivities}()$	1	C_3	$1 \times C_3$	n
4	for each a in A	$n+1$	C_4	$(n+1) \times C_4$	
5	if $\text{checkSignatureThreshold}(a, p_u)$	n	C_5	$n \times C_5$	
6	add a in L	n	C_6	$n \times C_6$	n
7	return L	1	C_7	$1 \times C_7$	
$f(n) = C_1 + (n+1)C_2 + C_3 + (n+1)C_4 + nC_5 + nC_6 + C_7 = O(n)$ (Time)			Total	$O(n)$	$O(n)$
$O(n)$ (Space)					

Figura 4.12. Algoritmo de busca por atividades em função do perfil do usuário.

realizada em quatro abordagens diferentes:

- **AND:** todas as notas compondo as assinaturas devem atender ao mínimo estabelecido (i.e., *threshold*). Esta abordagem é a mais restrita das quatro existentes, porém mais precisa em relação aos interesses do turista. Neste caso, as atividades devem possuir uma assinatura bem próxima dos interesses do turista.
- **OR:** pelo menos uma das notas deve atender ao mínimo estabelecido. Esta abordagem é a mais flexível de todas, porém menos precisa em relação aos interesses do turista. Pelo menos uma das notas que compõem a assinatura deve estar próxima da nota definida no perfil do turista.
- **AND/OR:** uma combinação das abordagens AND e OR, de tal forma que teremos uma quantidade mínima de notas atendendo ao mínimo estabelecido em relação ao perfil do turista. Esta opção possui a vantagem de utilizar as características de ambas abordagens, que incluem a vantagem da restrição da alternativa AND e a flexibilidade da alternativa OR.
- **TOP:** uma seleção das atividades que possuam a maior nota nas categorias ou interesses em comparação ao perfil do turista. Desta forma, as atividades em geral terão uma das categorias definidas como sua principal classificação em função dos interesses do turista, podendo ser definida como sendo a maior nota nos perfis presentes.

Suponhamos, por exemplo, um turista com o perfil que possua os interesses conforme apresentado na Figura 4.14. A partir dos interesses definidos pelo usuário,

Algorithm CheckActivitySignature	Times	Cost	Times x Cost	Space
Input: Profile p and Activity a				
Output: boolean (true if within threshold)				
1 $s_p = p.signature$	1	C_1	$1 \times C_1$	
2 $s_a = a.signature$	1	C_2	$(n+1) \times C_2$	
3 $result = false$	1	C_3	$1 \times C_3$	
4 for each v_p, v_a in s_p, s_a	$n+1$	C_4	$(n+1) \times C_4$	
5 if $abs(v_p - v_a) < MINIMUM$	n	C_5	$n \times C_5$	
6 $result = true$	n	C_6	$n \times C_6$	
7 return result	1	C_7	$1 \times C_7$	
$f(n) = C_1 + C_2 + C_3 + (n+1)C_4 + nC_5 + nC_6 + C_7 = O(n)$ (Time)		Total	$O(n)$	$O(1)$
$O(1)$ (Space)				

Figura 4.13. Algoritmo de verificação da assinatura do perfil e da atividade.

	Consumer	Historical	Envorinmental	Gastronomical	Cultural	Bohemian
Tourist01	1	6	8	10	7	2
Activity01	0	9	7	2	8	0
Activity02	2	6	9	2	8	0
Activity03	0	9	6	0	10	0
Activity04	0	8	10	7	8	0
Activity05	9	7	0	8	7	8
Activity06	8	6	0	9	4	10

Figura 4.14. Tabela contendo o perfil de um turista e de atividades disponíveis.

dependendo da abordagem escolhida, o algoritmo seleciona ou não atividades para sugestão. Neste exemplo, procuramos definir atividades que possuam pelo menos uma das características possíveis em destaque. A partir desta definição, podemos avaliar a seleção de atividades em função de cada uma das abordagens propostas.

A partir dos perfis definidos na Figura 4.14, avaliamos a seleção de atividades para cada uma das abordagens apresentadas (i.e., AND, OR, AND/OR, TOP). Variando o limite (e.g., *threshold* entre 1 e 3 pontos, podemos verificar a capacidade de seleção e filtragem de atividades para o turista em cada uma das abordagens. As Figuras 4.15(a), 4.15(b) e 4.15(c) apresentam os resultados da seleção de atividades para cada um dos limites (i.e., 1 a 3), respectivamente.

A partir das Figuras 4.15(a), 4.15(b) e 4.15(c), podemos notar o comportamento da seleção de atividades para cada uma das abordagens e em limites diferentes. Quanto menor o limite, mais preciso será a seleção de atividades para o turista para todas as abordagens selecionadas, conforme apresentado na Figura 4.15(a). Com o limite

	Activity Selection in Relation to Tourist01 Profile						Count		Activity Selection Approaches			
	Consumer	Historical	Envorinmental	Gastronomical	Cultural	Bohemian	TRUE	FALSE	AND	OR	AND/OR	TOP
	Activity01	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	3	3	No	Yes	Yes
Activity02	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	4	2	No	Yes	Yes	Yes
Activity03	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	1	5	No	No	No	No
Activity04	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	2	4	No	Yes	No	Yes
Activity05	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	2	4	No	Yes	No	Yes
Activity06	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	2	4	No	Yes	No	Yes

Threshold: 1

(a) *threshold = 1*

	Activity Selection in Relation to Tourist01 Profile						Count		Activity Selection Approaches			
	Consumer	Historical	Envorinmental	Gastronomical	Cultural	Bohemian	TRUE	FALSE	AND	OR	AND/OR	TOP
	Activity01	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	4	2	No	Yes	Yes
Activity02	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	5	1	No	Yes	Yes	Yes
Activity03	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	3	3	No	Yes	Yes	No
Activity04	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	5	1	No	Yes	Yes	Yes
Activity05	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	3	3	No	Yes	Yes	Yes
Activity06	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	2	4	No	Yes	No	Yes

Threshold: 2

(b) *threshold = 2*

	Activity Selection in Relation to Tourist01 Profile						Count		Activity Selection Approaches			
	Consumer	Historical	Envorinmental	Gastronomical	Cultural	Bohemian	TRUE	FALSE	AND	OR	AND/OR	TOP
	Activity01	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	5	1	No	Yes	Yes
Activity02	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	5	1	No	Yes	Yes	Yes
Activity03	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	5	1	No	Yes	Yes	No
Activity04	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	0	Yes	Yes	Yes	Yes
Activity05	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	3	3	No	Yes	Yes	Yes
Activity06	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	3	3	No	Yes	Yes	Yes

Threshold: 3

(c) *threshold = 3*

Figura 4.15. Os resultados da seleção de atividades para cada uma das abordagens.

menor, entretanto, o algoritmo será mais restritivo, possibilitando uma seleção menor de atrações turísticas para o usuário. Para todos os limites avaliados, apenas com valores maiores que três, foi possível selecionar uma atividade na abordagem AND. Por exemplo, no caso da Figura 4.15(a), a abordagem AND foi consideravelmente restritiva, onde o algoritmo selecionou nenhuma das atividades disponíveis. Porém, a abordagem OR selecionou a maior quantidade de atividades (i.e., 5/6), porém com um grau menor de precisão em relação aos interesses do turista. As abordagens AND/OR (utilizando o mínimo de interesses similares para 3) e TOP (utilizando a atividade com a maior pontuação entre as categorias) selecionaram duas e quatro atividades, respectivamente.

Um dos fatores que também influencia a seleção de atividades está na quantidade

e diversidade de atividades turísticas na cidade ou local em questão. Quanto maior a quantidade e diversidade de atrações, maiores serão as opções de seleção de atividades para o turista, possibilitando optarmos na utilização de abordagens mais restritivas, tais como a AND. No caso da existência de poucas atividades e/ou atividades com pouca diversidade, as abordagens TOP e AND/OR são as mais recomendadas.

Com o objetivo de cobrir a maioria dos casos possíveis, é possível também utilizarmos uma solução híbrida utilizando mais de uma única abordagem na seleção de atividades. Esta solução inicia-se com a abordagem mais restritiva (i.e., AND), e caso não haja um mínimo desejado de atividades selecionadas para o turista, o sistema altera a abordagem para a próxima menos restritiva (i.e., AND/OR). Caso ainda não tenha o mínimo desejado, o sistema altera a abordagem para a TOP, e assim por diante. Além do sistema, o próprio turista poderá avaliar as seleções realizadas, informando ao sistema se ele/ela deseja uma maior restrição ou flexibilidade na seleção de atividades.

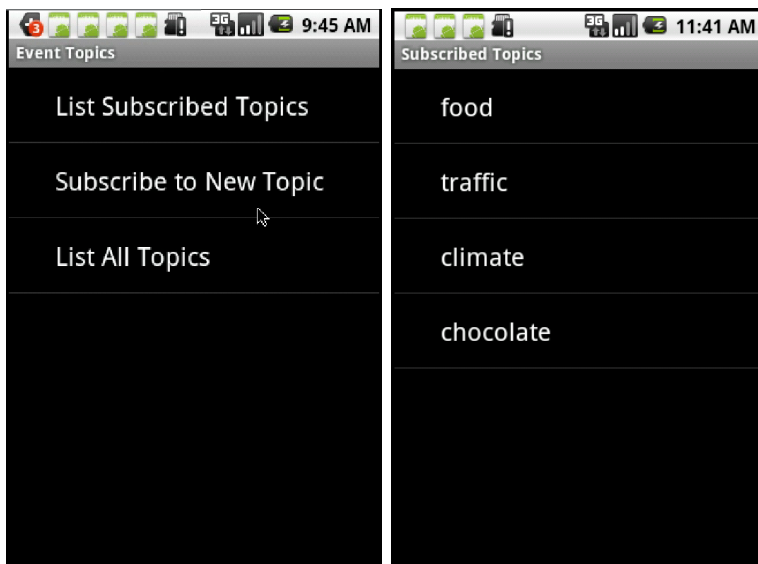
4.3.3 Seleção de Tópicos ou Canais de Interesse

Além da seleção e assinatura em serviços Web e atividades turísticas, o *DroidGuide* permite a seleção de tópicos ou canais de interesse, conforme apresentado na Figura 4.16. A assinatura por tópicos permite que o servidor de eventos notifique usuários inscritos em eventos ocorridos relacionados com tópicos. A aplicação móvel permite que o usuário móvel efetue a busca por tópicos de eventos disponíveis no sistema, a listagem de tópicos inscritos por ele e a assinatura em um novo tópico. O usuário possui acesso aos tópicos inscritos por ele e a listagem completa de todos os tópicos disponíveis do sistema, inclusive de outros usuários móveis logados na aplicação.

4.4 Detalhes de Implementação

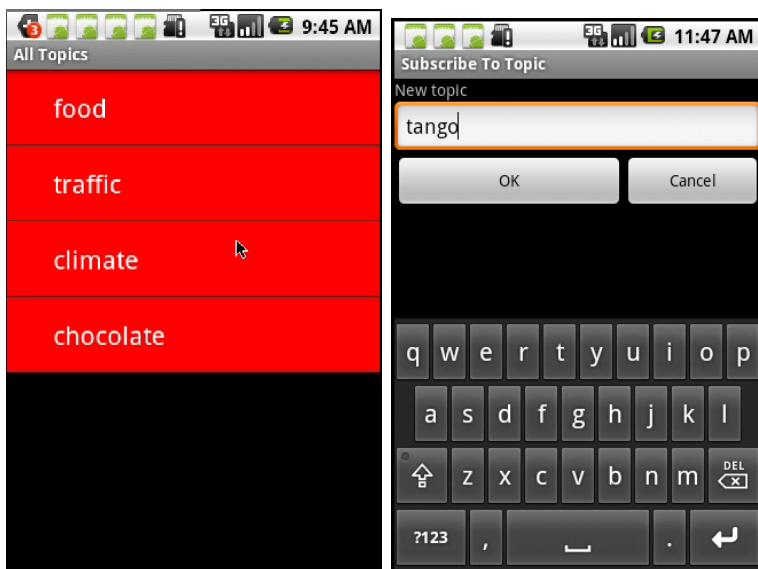
O guia turístico *DroidGuide* foi desenvolvido utilizando a API definida na plataforma *Android* utilizada na criação de aplicações móveis em Java. Com o objetivo de facilitar o desenvolvimento dos componentes necessários, dividimos o projeto em três subprojetos: cliente, comum e servidor. O primeiro subprojeto é responsável pelas classes e interfaces Java que utilizam diretamente a plataforma *Android*.

O segundo projeto possui as classes Java que não possuem dependências com a plataforma *Android*. O principal objetivo desta separação foi um desenvolvimento simplificado dos componentes, possibilitando, por exemplo, a codificação de testes unitários sobre os componentes desenvolvidos. Boa parte das classes referentes ao processamento de eventos e ao módulo de comunicação do cliente está contida dentro



(a) Menu de tópicos.

(b) Tópicos subscritos.



(c) Todos os tópicos.

(d) Novo tópico.

Figura 4.16. Funcionalidades de seleção de tópicos para notificações de eventos.

deste projeto. Na abordagem de desenvolvimento utilizada, é possível utilizarmos o processador de eventos em qualquer plataforma de software que utilize Java como linguagem de programação.

O terceiro subprojeto contém as classes e definições de operações a serem executadas no lado servidor definidos na linguagem *Python* (versão 1.0) e Java (versão 2.0). Enquanto o terceiro projeto é completamente independente dos outros subprojetos no que diz respeito à compilação de classes e funções, o primeiro projeto (i.e., cliente) possui uma dependência direta com o segundo (e.g., comum), já que as classes de interface de usuário utilizam entidades e serviços definidos no subprojeto comum.

4.4.1 Sequência de Execução

Em relação ao envio e recepção de eventos pela aplicação móvel, o protótipo executa da seguinte maneira. No início da execução da aplicação, o usuário se autentica no sistema informando seu *username* e *password*, conforme apresentado na Figura 4.4(a). Logo após a autenticação, o usuário então define suas informações de perfil, que incluem seus interesses relacionados às atividades turísticas, conforme apresentado na Figura 4.5(a). A aplicação móvel envia as informações definidas pelo usuário para o servidor, informando de que o perfil foi atualizado. Na inicialização da aplicação móvel, o processador de eventos inicia o ciclo periódico de execução do processo de envio e solicitação por eventos cadastrados no servidor remoto de dados. No envio de eventos, o processador de eventos localizado no cliente envia os dados de novos eventos coletados no dispositivo para o servidor de eventos, a fim destes serem compartilhados com serviços Web subscritos pelo usuário. Após o envio, o processador de eventos no dispositivo requisita por novos eventos que eventualmente tenham sido criados no servidor, a fim de receber notificações dos mesmos na aplicação. O processamento das requisições enviadas e das respostas recebidas do servidor é realizado pelo módulo de comunicação. Um diagrama de sequência da operação acima pode ser visualizado na Figura 4.17.

O processo de requisição por eventos no *DroidGuide* utiliza a execução de tarefas agendadas (*TimerTask*) através de um processo dedicado na aplicação móvel. Este processo possibilita o acesso à informações de eventos ocorridos no servidor de dados remoto e o compartilhamento dos eventos ocorridos no lado cliente. A aplicação móvel também permite a configuração do intervalo do ciclo do processo de comunicação, definido em segundos. Desejamos futuramente que este parâmetro seja gerenciado pela própria aplicação, de tal forma que o período seja calibrável em situações onde o sistema gera uma quantidade grande, média ou pequena de eventos. Por exemplo, em casos

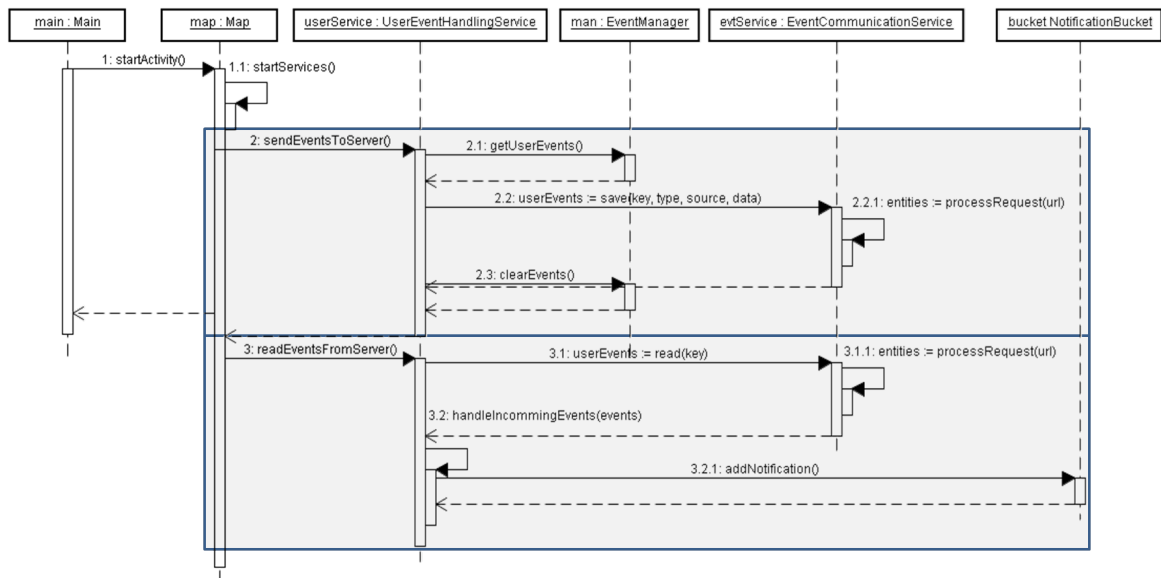


Figura 4.17. Diagrama de sequência de execução do mecanismo de envio e recepção de eventos pela aplicação móvel.

onde exista um número alto de eventos gerados em ambos os lados, o intervalo do ciclo poderá ser diminuído gradual e automaticamente pela aplicação.

Na execução do ciclo de atualização por notificações, o *DroidGuide* utiliza duas possíveis abordagens de desenvolvimento de processos disponíveis na plataforma *Android*: através de *threads* e *broadcast receivers*, conforme apresentado na seção 2.6.2. A primeira abordagem apresenta um desenvolvimento mais simplificado já que esta utilizou conceitos já existentes no *JavaSE* em outras plataformas de desenvolvimento. Entretanto, a segunda abordagem forneceu um conjunto maior de recursos e um melhor tratamento de notificações pelo *Android* devido ao fato deste mecanismo estar incorporado na arquitetura da plataforma em si, apesar de prover uma complexidade maior no desenvolvimento da solução.

4.4.2 Desafios e Limitações

Podemos destacar alguns desafios ou limitações encontrados no desenvolvimento do protótipo apresentado. Em relação aos serviços Web suportados pela aplicação, não foi possível emularmos um conjunto maior de serviços remotos, tais como de localização, tráfego, emergências, dentre outros. A integração do servidor de eventos com outros módulos também foi comprometida devido a não conclusão de algumas funcionalidades dependentes, conforme apresentado na Figura 4.3. Alguns módulos

Característica	JME	Android
Modelo de <i>Graphical User Interface</i> (GUI)	Componentes do <i>Liquid Crystal Display GUI</i> (LCDUI)	Atividades e componentes de interface (Widgets)
Mecanismo de persistência local	<i>Record Management System</i>	Provedores de conteúdo
Ativação de componentes	Programático	Intenções
Conectividade de dados	<i>Generic Connection Framework</i>	<i>HTTP Commons</i> e <i>Java Net API</i>
Concorrência	Processos (<i>threads</i>)	Provedores e consumidores de serviços
Ciclo de vida da aplicação	<i>MIDP</i>	Proprietário
Persistência permanente	Não possui	<i>SQLLite</i>
Mecanismo de permissões	<i>MIDP</i>	Proprietário
Suporte a Mapas	Não implícito	Sim

Tabela 4.3. Comparativo de recursos entre o *JavaME* e o *Android*.

que foram postergados incluem: (a) adaptação (Adp) no que diz respeito ao consumo de conteúdo (e.g., texto, áudio e vídeo), (b) de contexto físico (PaC) no que diz respeito à informações de tráfego e de localização e (c) de escalonamento de atividades (AS) com informações relativas a atividades executadas ou não executadas pelo turista e sugestões de atividades levando em consideração a localização do usuário.

O modelo de desenvolvimento de aplicações móveis proposto pela plataforma *Android* trouxe também um desafio significativo ao trabalho, já que ela difere de todos os outros modelos de desenvolvimento de aplicações móveis existentes antes deste, tais como o *Mobile Information Device Profile* (MIDP) 2.0 [53]. O *Android* utiliza padrões de desenho e de desenvolvimento diferentes do MIDP na criação das aplicações e serviços executados sobre a plataforma. Algumas diferenças incluem o modelo de interface gráfica baseado em atividades, mecanismo de persistência local via provedores de conteúdo e a ativação de componentes através de intenções, conforme apresentado na Tabela 4.3. Esta nova metodologia de desenvolvimento exigiu um maior tempo para o entendimento de como as aplicações móveis são construídas sobre plataforma a partir dos conceitos definidos pelo servidor de eventos e aplicação móvel propostos.

Além dos desafios já apresentados acima sobre a plataforma *Android*, outras dificuldades foram também encontradas na utilização de sua API. A API do *Android* sofreu frequentes atualizações durante o desenvolvimento deste trabalho. Nos primeiros protótipos implementados, utilizamos a versão 0.8 para a construção das interfaces gráficas e do mecanismo de comunicação com o servidor. Durante o desenvolvimento, a versão 1.0 e 1.1 foram disponibilizadas, com diversas alterações em sua API, resultando em problemas de compatibilidade entre a nova versão e componentes já desenvolvidos na versão anterior. Optamos em migrar o protótipo para a versão 1.1 e, logo após, para a versão 2.0, já que as migrações poderiam ser um risco para a finalização da aplicação móvel. Atualmente, o *Android* está na versão 2.1 de sua API.

Componente	DroidGuide
<i>Processador de eventos no servidor</i>	Processamento de eventos de clientes móveis e serviços Web
<i>Container de eventos</i>	Armazenamento de eventos turísticos
<i>Gestor de subscrições</i>	Assinatura por eventos turísticos
<i>Processador de eventos no servidor</i>	Coordena os componentes acima no guia turístico
<i>Container de serviços Web</i>	Container de serviços Web turísticos
<i>Container de aplicações</i>	Container da aplicação turística
<i>Processador de eventos no cliente</i>	Processa eventos no <i>Android</i>
<i>Gestor de perfil e contexto</i>	Gerencia informações do usuário e turísticas no <i>Android</i>
<i>Aplicações e serviços</i>	Aplicação cliente desenvolvida no <i>Android</i>
<i>Sensores no ambiente</i>	Sensores presentes no dispositivo móvel e ambiente

Tabela 4.4. Componentes da arquitetura do servidor de eventos aplicada no protótipo *DroidGuide*.

4.5 Aplicação da Arquitetura Proposta

Esta seção tem como objetivo apresentar a relação do protótipo desenvolvido neste trabalho com a arquitetura do servidor de eventos proposta. Esta seção também discute como a arquitetura proposta foi utilizada no estudo de caso do guia turístico desenvolvido neste trabalho. Conforme apresentado na Figura 3.14, a arquitetura do sistema proposto neste trabalho é composta por diversos componentes e módulos. Cada um destes componentes possui responsabilidades definidas para o protótipo desenvolvido neste trabalho. Utilizamos a arquitetura definida para o servidor de eventos para o desenvolvimento dos serviços necessários na aplicação. Um mapeamento entre os componentes definidos na arquitetura do servidor de eventos e do protótipo *DroidGuide* pode ser visualizado na Tabela 4.4.

No guia turístico *DroidGuide*, o servidor de eventos tem como responsabilidade a captação dos eventos vindos da aplicação móvel e de serviços Web. Neste processo, o servidor utiliza os componentes de assinatura de eventos, processamento e armazenamento de eventos presentes no lado do servidor remoto de dados, conforme a Figura 4.18. O gerenciador de subscrições é responsável por captar as solicitações de assinatura em eventos, serviços Web e atrações turísticas através de tópicos ou canais de interesse. Estes tópicos são definidos por atrações, serviços ou pelo próprio usuário em casos de eventos de interesse turístico. O processador de eventos é responsável por captar eventos vindos do lado do cliente e também do servidor, armazená-los no *container* de eventos ou *Event Container* e enfileirar eventos para notificação em função das assinaturas existentes no sistema. No guia turístico, quando eventos são criados por serviços remotos, o processador de eventos efetua o enfileiramento de notificações para o usuário móvel em função de seus interesses em receber notificações referentes

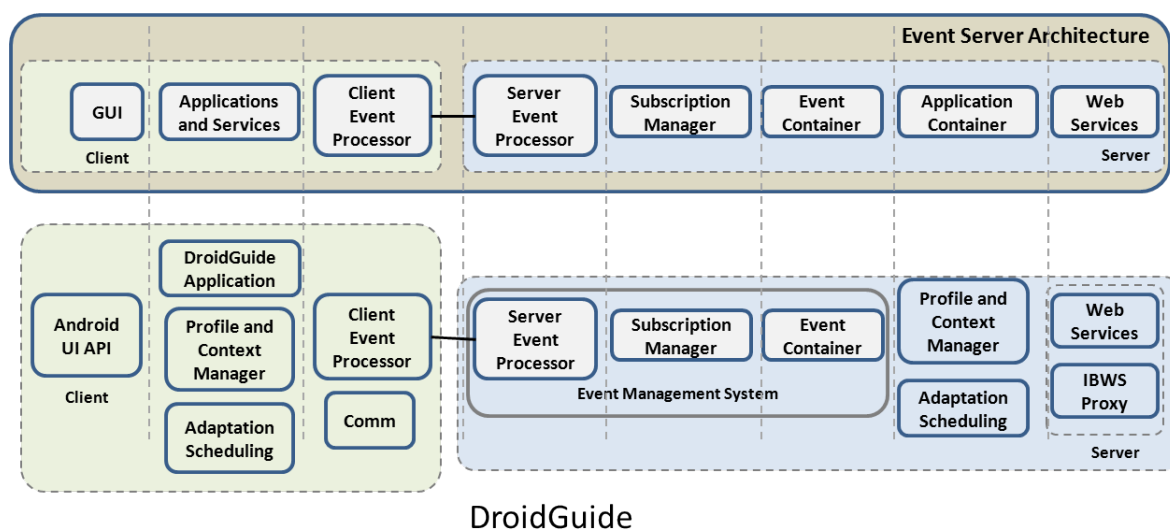


Figura 4.18. Mapeamento dos componentes no protótipo *DroidGuide* em função da arquitetura proposta.

aos eventos.

Dois outros componentes adicionais oferecem recursos a aplicação servidora do guia turístico. O *container* de serviços Web baseados em informações (IBWS Proxy) é responsável por agrupar os serviços Web disponíveis para usuários móveis, tais como serviços meteorológicos, de tráfego, de serviços sensíveis à localização, dentre outros. No caso do guia turístico, este componente fornece ao gerenciador de subscrições a listagem de serviços remotos disponíveis para o usuário móvel e o processamento de eventos vindos do cliente. Desta forma, os serviços Web utilizam as informações de perfil e contexto do usuário móvel providas através da publicação e notificação de eventos a fim de prover atividades e recursos relacionados ao turismo. O *container* de aplicações localizado no servidor gerencia as instancias de aplicações remotas utilizadas pelo guia turístico, como por exemplo, autenticação, seleção de atividades turísticas, comunicação entre usuários móveis, localização do usuário e gestão de marcos turísticos. O *container* de aplicações comunica diretamente com o servidor de eventos a fim de oferecer diversas informações do usuário móvel e de atrações turísticas disponíveis no sistema.

No lado cliente, a aplicação móvel possui o processador de eventos, o gerenciador de informações de perfil e contexto e a aplicação contendo diversos serviços a serem oferecidos ao usuário móvel e a aplicação móvel. O processador de eventos no cliente recebe requisições de mudanças de informações de perfil e contexto vindas do gerenciador e as envia para serem processadas pelo servidor de eventos no servidor.

Por exemplo, quando o usuário solicita uma mudança no seu contexto lógico, ela é detectada pelo gerenciador de perfil e contexto e, após a detecção, ela é enviada para o processador de eventos que irá finalmente enviá-la para o servidor para armazenamento e processamento.

Em relação aos sensores de ambiente, o guia turístico utiliza recursos no cliente e servidor para "emular" informações coletadas para serem utilizadas pelo serviço e aplicação. No caso do cliente, as informações coletadas pelo dispositivo móvel e por sensores presentes no ambiente são utilizadas na definição do perfil e contexto local do usuário móvel. Alguns exemplos de informações de contexto local incluem estado do usuário, localização, temperatura ambiente, dentre outros. No lado do servidor, informações coletadas por sensores e serviços remotos compõem as informações de perfil e contexto remoto do usuário móvel. As informações remotas incluem condição de tráfego, clima, proximidade entre usuários e serviços turísticos, dentre outros. A união das informações locais e remotas compõe o contexto representado ao usuário móvel, contendo informações locais e remotas referentes ao seu estado e interesses.

4.6 Resultados obtidos

Os resultados obtidos a partir do protótipo *DroidGuide* podem ser visualizados no capítulo 6. Estes resultados incluem a criação de um evento climático e a notificação deste evento no dispositivo móvel através do servidor de eventos proposto.

Capítulo 5

Serviço de Contexto para Emergências

Este capítulo apresenta o segundo protótipo desenvolvido neste trabalho a utilizar o servidor de eventos, denominado DECS. Ele foi construído com o objetivo de validar o desenvolvimento do servidor de eventos proposto neste trabalho em um segundo cenário de uso. Este cenário apresenta um serviço de contexto para emergências envolvendo pacientes, unidades móveis e fixas de atendimento que enviam, recebem e processam eventos de emergência relacionados à saúde, segurança ou incêndios.

5.1 Visão Geral

O serviço de gerenciamento de eventos de emergência DECS tem como objetivo fornecer o acompanhamento e alocação de recursos de emergência, tais como unidades móveis e fixas, pacientes, dentre outros. O serviço DECS possui as seguintes funcionalidades:

- Visualização de dados georeferenciados: cadastro de emergências e visualização de informações relacionadas das unidades móveis;
- Sugestões de caminhos para pontos de interesse: disponibilidade da rota detalhada em formato gráfico e em texto, conforme apresentado na Figura 5.1;
- Visualização de usuários *online*: acesso à lista de usuários móveis e fixos conectados no sistema;
- Atualização de localização das unidades móveis: Atualização periódica da localização das unidades móveis de emergência;
- Visualização dos eventos gerados o sistema: listagem dos eventos gerados no sistema para o usuário;



Figura 5.1. Visualização de informações sobre rotas e estatísticas pela aplicação.

- Publicação de eventos em função da localização: criação de eventos sensíveis a localização pelo usuário.
- Acesso às informações estatísticas: dados estatísticos da execução da aplicação no cliente Web e no servidor remoto de dados, conforme apresentado na Figura 5.1. A Figura 5.1 apresenta dados estatísticos coletados após o atendimento a um evento de emergência.

5.2 O Fluxo de Gerenciamento de Emergências

O serviço DECS permite gerenciar o fluxo no atendimento de emergências em uma determinada região. Este gerenciamento envolve a utilização de entidades móveis (automóveis ou usuários em ambulâncias, viaturas de polícia e bombeiros) e entidades fixas (e.g., marcos como hospitais, delegacias de polícia e corpo de bombeiros). As entidades comunicam entre si e com o serviço de emergências através do envio e do consumo de eventos criados por produtores, publicados pelo servidor de eventos e consumidos por entidades fixas e móveis. O sistema fornece serviços de informações georeferenciadas e alocação de recursos móveis e fixos durante o atendimento a um evento de emergência.

A aplicação móvel inicia o ciclo de execução a partir de uma requisição responsável em buscar e renderizar as entidades responsáveis pelo tratamento de eventos de

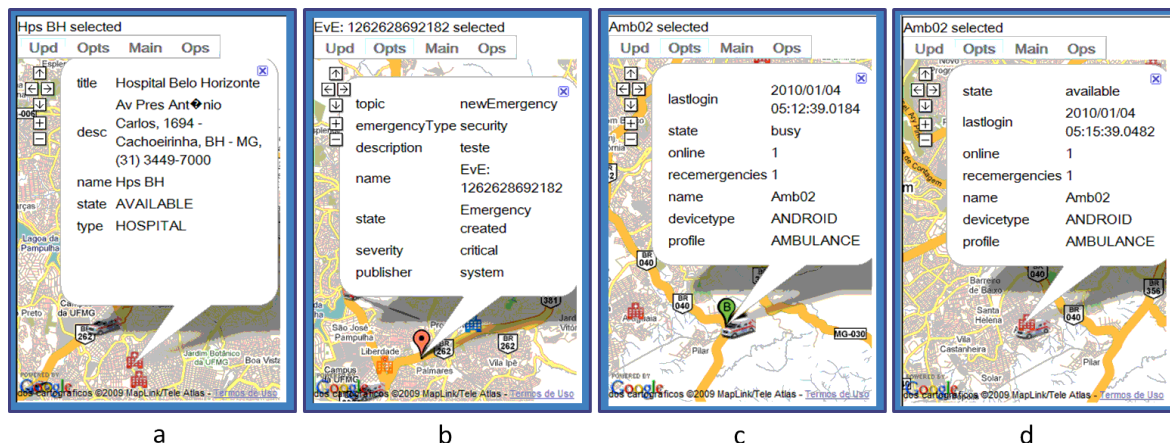


Figura 5.2. Visualização de detalhes de marcos apresentados sobre o mapa. Em (a), a visualização dos detalhes de um marco do tipo **HOSPITAL**. Em (b), a visualização de um evento de emergência criado sobre uma localização. Em (c), visualização da UnM em atendimento a um evento de emergência. Em (d), a UnM após entregar a emergência a um marco fixo (i.e., hospital).

emergência que podem surgir dentro de uma área pré-definida em um espaço de tempo. As entidades responsáveis neste caso incluem as entidades móveis, neste caso as ambulâncias, as viaturas móveis (e.g., policiais e do corpo de bombeiros) e os marcos fixos tais como os hospitais, delegacias e corpos de bombeiros. Na resposta da requisição, o servidor de eventos envia para o cliente uma listagem das entidades a serem renderizadas no mapa em formato XML, conforme apresentado na Figura 5.2.

Durante o ciclo de atendimento de um evento de emergência, o evento em si contém um "proprietário" associado, podendo ser uma UnM ou marco fixo. No caso da UnM, enquanto ela estiver relacionada a um determinado evento de emergência, ela estará ocupada e impossibilitada de atender outros eventos cadastrados no sistema naquele instante. Somente após a entrega do evento à um marco fixo, a UnM estará disponível para um atendimento de uma nova emergência. A disponibilidade de uma UnM depende diretamente de seu estado (ocupado ou disponível) e seu tipo, já que para determinados tipos de emergências precisaremos de UnMs específicas (e.g., saúde-ambulância, segurança-polícia, incêndio-bombeiro). Após a listagem e renderização dos marcos no mapa apresentado ao usuário móvel, a aplicação está preparada para receber notificações de novos eventos de emergência a serem gerenciados pelo serviço.

O serviço DECS possui um fluxo de execução composto de estados e operações a serem executadas sobre as entidades relacionadas (evento de emergência, marcos fixo, unidade móvel de emergência e rota) para a mudança destes estados, conforme

apresentado na Figura 5.3. Uma unidade de emergência (UnM) armazena as informações de perfil e contexto de uma ocorrência, tais como localização, estado do paciente/vítima, elemento responsável, marco fixo de atendimento, dentre outras.

A Unidade Móvel de Emergência inicia o fluxo de atendimento através do envio de uma requisição para serviço de emergência. Esta requisição pode ser realizada de diversas formas, como por exemplo, a partir de um telefonema para uma central, uma mensagem SMS ou uma requisição HTTP. O dispositivo móvel envia na requisição informações relevantes tais como a localização e o tipo de evento (e.g., segurança, emergência ou incêndio) e informações adicionais de auxílio ao serviço. O servidor de dados recebe a requisição vinda do usuário (PoV) e cria um Evento de Emergência (EvE) no sistema representando a respectiva ocorrência. Este evento armazena diversas informações sobre a emergência, tais como sua localização, seu tipo (i.e. saúde, segurança ou incêndio) e uma descrição sobre o estado da(s) vítima(s), quando disponível. Após o cadastro da emergência no sistema, o serviço de emergências pesquisa e seleciona a UnM disponível mais próxima do evento para o atendimento. Após a seleção, o serviço requisita a UnM selecionada para se deslocar até o destino definido pelo evento.

Na chegada da unidade de emergência ao evento, ela torna-se responsável em: (a) recolher o paciente ou vítima, (b) enviar informações adicionais quando aplicáveis e disponíveis e (c) se preparar para o transporte do paciente até o seu destino. O servidor de dados então efetua a seleção do marco mais próximo de acordo com tipo de emergência. Neste fluxo, a UnM coleta e transporta o paciente para o respectivo marco, podendo ser um hospital em caso de acidentes ou uma delegacia de polícia em casos de distúrbio da ordem e/ou segurança. No caso do corpo de bombeiros, a UnM de incêndio respectiva (e.g., caminhão de bombeiros) não transporta pacientes, mas o sistema desenvolvido permite que UnMs notifiquem outras UnMs (e.g., ambulância ou viatura policial) caso exista uma necessidade de apoio.

Os estados apresentados na Figura 5.3 representam as operações a serem executadas pelas entidades envolvidas no sistema. Os estados existentes em um atendimento de emergência providos no DECS são:

1. ***Emergency Detection:*** Uma emergência é informada por um usuário (paciente/vítima) através da marcação do ponto ou local onde esta está localizada. Outras formas de detecção de emergências podem ser utilizadas, tais como por telefone, serviço Web, dentre outras. O serviço de emergências recebe a requisição de detecção por uma emergência.
2. ***Emergency Event Created:*** O serviço de emergências cria uma nova instância

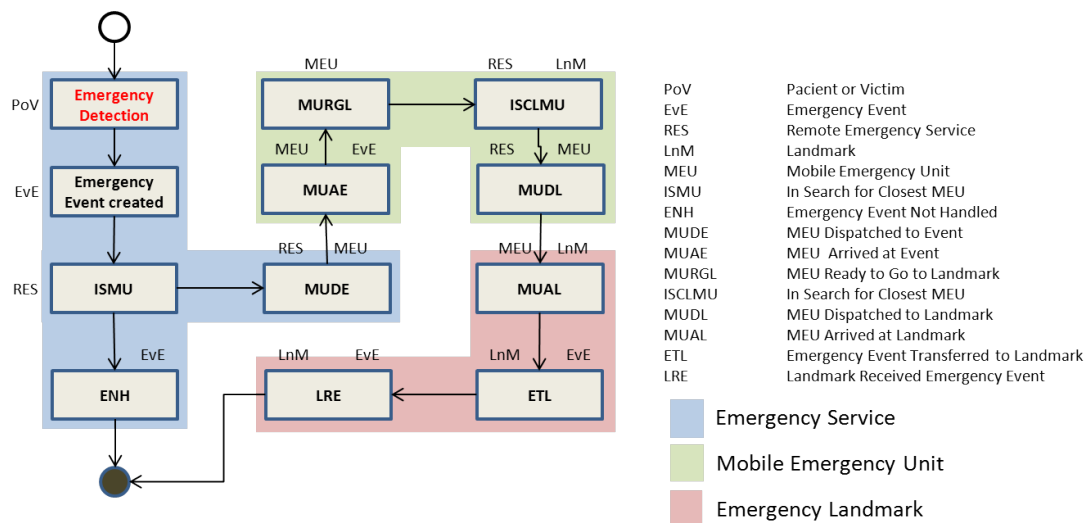


Figura 5.3. Diagrama de estados de um evento de emergência gerenciado pelo DECS.

de emergência (evento) para representar a ocorrência do evento submetida pelo cliente móvel.

3. **ISMU (In Search for Mobile Users):** O serviço busca por UnMs disponíveis que estejam mais próximas do evento de emergência. Caso haja uma UnM disponível, o serviço informa seu identificador para que ela possa ser chamada para o atendimento. Caso contrário, o evento de emergência não será atendido, já que não haverá uma UnM disponível para atendê-lo. Em versões futuras, deseja-se criar um processo que permita a verificação periódica da disponibilidade de UnMs para o atendimento a eventos de emergência sem atendimento no momento.
4. **ENH (Emergency Not Handled):** Este estado é utilizado em duas ocasiões: (a) o serviço não encontrou uma UnM disponível para o atendimento ou (b) a posição informada do evento de emergência não pode ser alcançada por uma UnM. No primeiro caso, uma indisponibilidade das UnMs pode ocorrer se, por exemplo, todas elas estiverem ocupadas no atendimento a outras emergências. No segundo caso, a posição informada não pode ser alcançada por que não existe uma rota em que a UnM possa utilizar para chegar ao destino definido na emergência.
5. **MUDE (Mobile User Dispatched to Emergency):** Ao encontrar uma UnM disponível, o serviço envia uma notificação informando de que este deverá atender ao evento de emergência, incluindo informações sobre a ocorrência, local e rota para chegar ao ponto. A rota é apresentada à UnM através da aplicação

móvel no dispositivo conectado ao sistema. A rota pode ser apresentada de duas maneiras: através do mapa e em forma de listagem de passos, conforme apresentado na Figura 5.1. Nesta fase, o cliente Web simula a movimentação da UnM pelo mapa através da apresentação de duas informações: (a) da animação do marco móvel na interface de mapa da rota apresentada e (b) da atualização de sua localização informando o serviço de emergencias suas novas coordenadas. Na chegada ao destino, a animação se encerra.

6. ***MUAE (Mobile User Arrived at Event)***: A UnM chegou ao local designado, informando o serviço maiores detalhes do evento de emergência, caso necessário.
7. ***MURGL (Mobile User Ready to Go to Landmark)***: A UnM está pronta para ir ao marco de emergência respectivo (e.g., hospital, delegacia ou corpo de bombeiros), dependendo do tipo de emergência sendo atendida (saúde, segurança, incêndio). O serviço de emergencias recebe o evento e se prepara para a próxima fase.
8. ***ISCLMU (In Search for Closest Landmark for Mobile User)***: O serviço busca pelo marco fixo (LnM) disponível mais próximo do local onde a UnM está localizada. Ao encontrar um marco disponível, o serviço informa a rota para a unidade móvel até o destino. A rota é apresentada à UnM através da aplicação móvel no dispositivo conectado ao sistema.
9. ***MUDL (Mobile User Dispatched to Landmark)***: A UnM é despachada para o marco escolhido. Nesta fase, o protótipo simula a movimentação da UnM pelo mapa de duas formas: (a) através da animação do marco móvel pela rota apresentada e (b) da atualização de sua localização informando o serviço suas novas coordenadas. Na chegada ao destino, a emulação se encerra.
10. ***MUA (Mobile User Arrived at Landmark)***: A unidade móvel de atendimento chegou ao marco designado (LnM) com o paciente ou vítima.
11. ***ETL (Emergency Transferred to Landmark)*** o evento de emergência é transferido da UnM para o marco, liberando assim a UnM para o atendimento de uma nova emergência.
12. ***LRE (Landmark Received Emergency)***: O marco fixo (e.g., hospital ou delegacia) LnM notifica o serviço que o evento de emergência está sobre o seu gerenciamento. A UnM é liberada para o atendimento a ocorrências futuras no sistema.

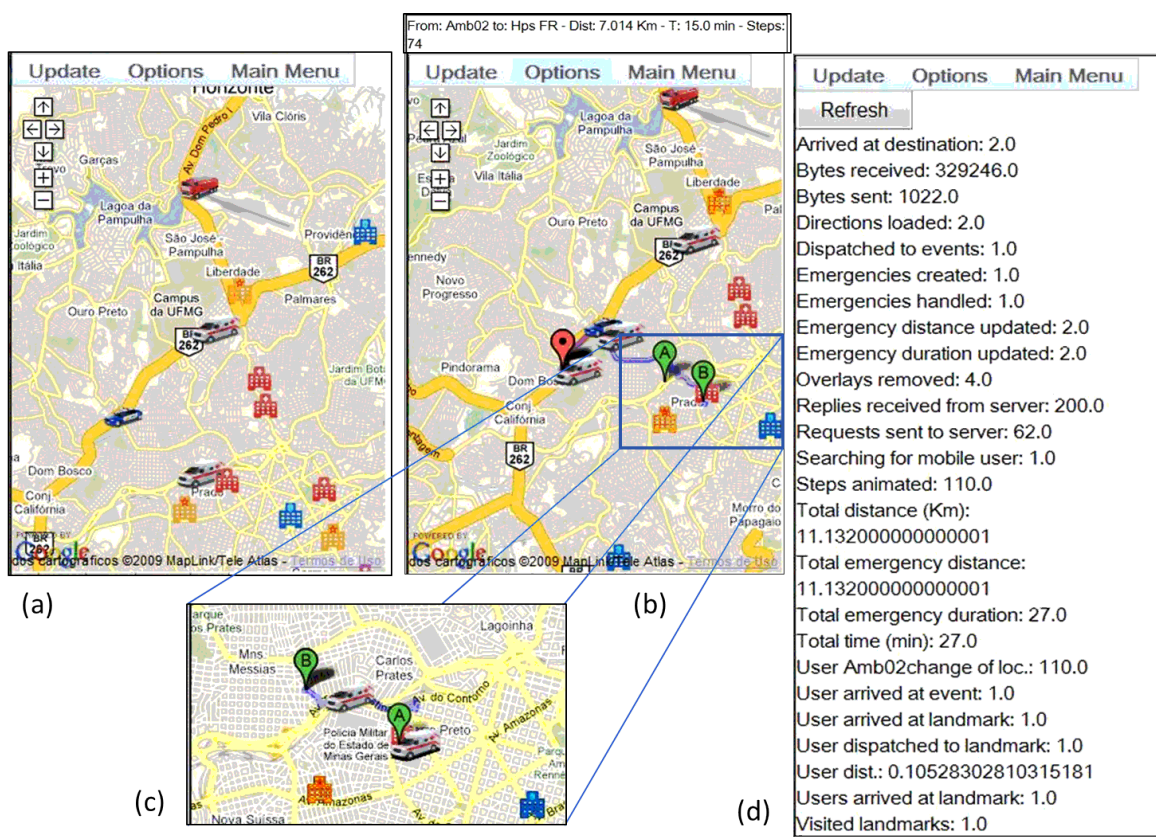


Figura 5.4. Execução do serviço DECS durante o tratamento de um evento de emergência. Em (a), o cliente é inicializado em um navegador Web; em (b) a UnM é despachada para o evento de emergência; em (c) um zoom da área onde a UnM é despachada para uma segunda emergência; e em (d) a apresentação de dados estatísticos coletados durante a execução do cliente.

Durante o processo de execução descrito acima, mudanças de perfil e contexto ocorrem tanto no cliente Web quanto no serviço de emergências. Informações tais como a localização da unidade móvel de atendimento, as condições do paciente/vítima, disponibilidade de marcos e a comunicação entre unidades são compartilhadas através da publicação e notificação de eventos ocorridos no sistema. Neste caso, o serviço de emergências utiliza o servidor de eventos para a publicação e notificação de eventos ocorridos durante o processo de execução, especificamente nas mudanças de estados dos eventos de emergência, das UnMs (e.g., disponibilidade), de suas posições (e.g., latitude e longitude) e dos marcos fixos. A Figura 5.4 apresenta a execução de um atendimento a um evento de emergência.

5.2.1 Protótipos Desenvolvidos

Conforme apresentado no capítulo 4, o guia turístico eletrônico *DroidGuide* utiliza a plataforma de desenvolvimento *Android* para a execução da aplicação móvel residente no dispositivo. Este cliente é responsável por comunicar com o servidor remoto de dados, onde está localizado o servidor de eventos proposto. Porém, devido à algumas características e restrições apresentadas nesta plataforma, optamos por avaliar novas alternativas de desenvolvimento para clientes leves (e.g., em inglês, *thin clients*) para o uso em sistemas móveis e ubíquos. Sendo assim, optamos em avaliar o desenvolvimento de interfaces e aplicativos leves utilizando tecnologias Web. Apresentaremos nesta seção os dois protótipos desenvolvidos para o serviço de contexto de emergencias utilizando estas tecnologias.

5.2.1.1 Protótipo 1.0: Desenvolvimento de Clientes Leves Web

A arquitetura do primeiro protótipo pode ser visualizada na Figura 5.5. O cliente é composto pelo navegador Web, documentos HTML com funções definidas em *JavaScript* e recursos adicionais de acesso ao serviço de mapas (GMaps) e de comunicação assíncrona (G-AJAX). O servidor remoto de dados utiliza um processador de requisições HTTP disponibilizado pelo arcabouço *Web AppEngine*, um servidor de eventos e de emergências. O servidor de eventos tem como objetivo receber eventos vindos de clientes Web e publicá-los a consumidores subscritos. O gerenciador de emergencias implementa as funcionalidades referentes ao tratamento de emergencias em uma determinada região que incluem operações de cadastro de eventos de emergencia até a alocação de recursos em marcos fixos para tais emergências. Através do uso da comunicação assíncrona sobre o protocolo HTTP, clientes Web enviam e recebem informações referentes aos marcos fixos e móveis relacionados ao serviço, tais como a localização, estado e informações de rota (i.e., distância e tempo). Estas informações coletadas são renderizadas na interface de mapas disponibilizada pela aplicação, conforme apresentado na Figura 5.6.

Diferente do utilizado no guia turístico *DroidGuide*, utilizamos tecnologias Web no desenvolvimento da interface gráfica de acesso ao DECS. Utilizamos as tecnologias também na interação do usuário móvel com o servidor de eventos já desenvolvido em *Python* e usado pelo *DroidGuide*. Entretanto, com o aumento da complexidade no desenvolvimento do cliente Web, algumas limitações surgiram, tais como a dificuldade no tratamento de requisições assíncronas no lado do cliente, dificuldade na reusabilidade de funções desenvolvidas em *JavaScript*, a complexidade no processamento de documentos em XML enviados pelo servidor de dados e a dificuldade

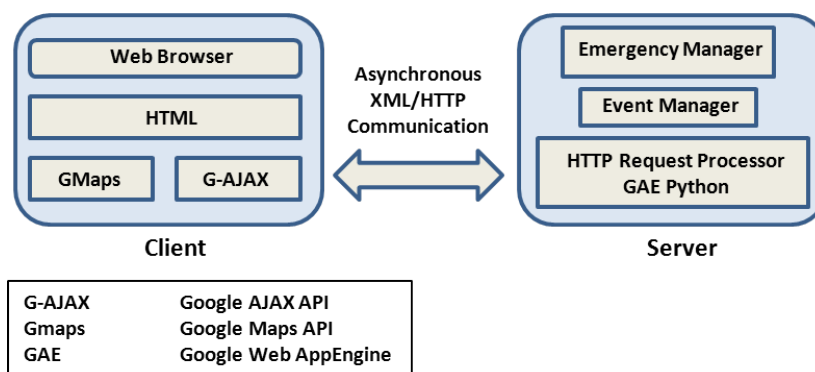


Figura 5.5. Arquitetura do primeiro protótipo desenvolvido para o DECS.

na realização de testes automatizados (e.g., testes unitários) nas funcionalidades implementadas.

A compatibilidade da interface Web desenvolvida em dispositivos móveis e navegadores Web móveis disponíveis tornou-se também um outro desafio. A interface Web desenvolvida poderia não ser renderizável pelos principais navegadores Web móveis no mercado, tornando assim um grande risco para o trabalho. Desta forma, efetuamos testes de renderização de páginas em HTML e *JavaScript* no *Android* de forma emulada e no navegador *SkyFire* em um dispositivo real utilizando a plataforma *Symbian OS S60*. O *Android* utiliza o renderizador de páginas HTML baseado no *WebKit*, responsável em converter o documento HTML em componentes de interface existentes na plataforma. O *Skyfire* utiliza uma abordagem diferente, utilizando um servidor Web intermediário que re-renderiza toda a página Web para ser apresentada no navegador, abordagem similar a outros navegadores Web para dispositivos móveis tais como o *Opera Mini*. Os resultados dos testes de rederização foram satisfatórios, possibilitando uma renderização completa e funcional da interface desenvolvida a partir de tecnologias Web em ambos dispositivos móveis. A Figura 5.6 demonstra a renderização no *Android* de duas telas desenvolvidas no DECS. Com os testes de renderização bem sucedidos, decidimos em continuar o desenvolvimento utilizando tecnologias Web.

Apesar das dificuldades apresentadas acima, a utilização da interface Web no desenvolvimento do cliente também trouxe alguns benefícios, tais como a facilidade de acesso à aplicação (e.g., em praticamente qualquer computador *Personal Computer* (PC)), flexibilidade no desenvolvimento (e.g., qualquer navegador Web e em qualquer PC com acesso a Internet) e a possibilidade de execução em sessões simultâneas de usuários (e.g., contexto coletivo), conforme apresentado na Figura 5.7. O grande ganho neste benefício está na facilidade em executarmos diversos clientes da aplicação, onde

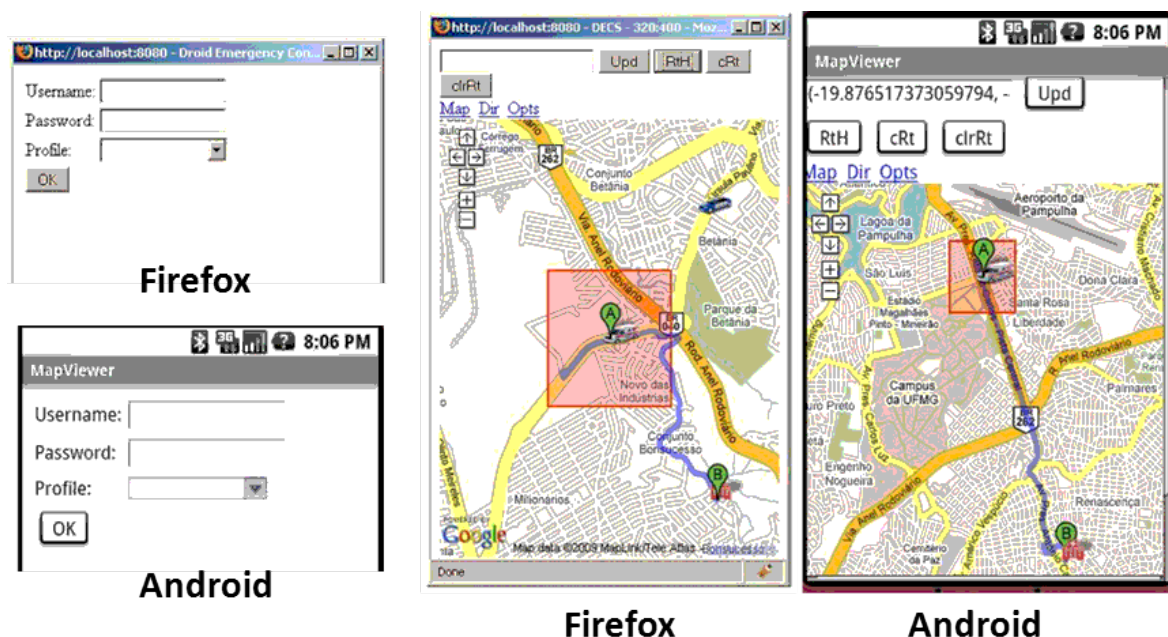


Figura 5.6. Testes de renderização do cliente do serviço DECS no *Android* e no navegador *Firefox*.

cada um destes pode residir em um mesmo espaço de endereço (vários navegadores em um PC) ou em espaços diferentes (diferentes navegadores em diferentes PCs).

As Figuras 5.6 e 5.7 apresentam alguns dos testes realizados de renderização e de execução coletiva do cliente Web, respectivamente. O primeiro teste tem como objetivo validar a renderização de componentes Web de interface gráfica a fim de possibilitar o uso destes no navegador Web do *Android*. Conforme apresentado na Figura, as telas de *login* e de mapas foram renderizadas corretamente, possibilitando também a interação do usuário com os componentes de interface como se estivessem sendo executados diretamente no lado cliente. O teste da interface de mapas teve como objetivo avaliar a capacidade do navegador do *android* em renderizar mapas e também permitir a interação do usuário com elementos sobre estes componentes. No final dos testes, fomos capazes de interagir com todos os elementos disponíveis, sendo possível a visualização de informações de rota entre pontos no mapa.

5.2.1.2 Protótipo 2.0: Utilização de Arcabouços Web

Até o início de 2009, o servidor de dados em nuvem utilizado pelos protótipos deste trabalho provia suporte a apenas uma única linguagem de programação (e.g., *Python*). Apesar do servidor de dados em nuvem prover algumas vantagens já

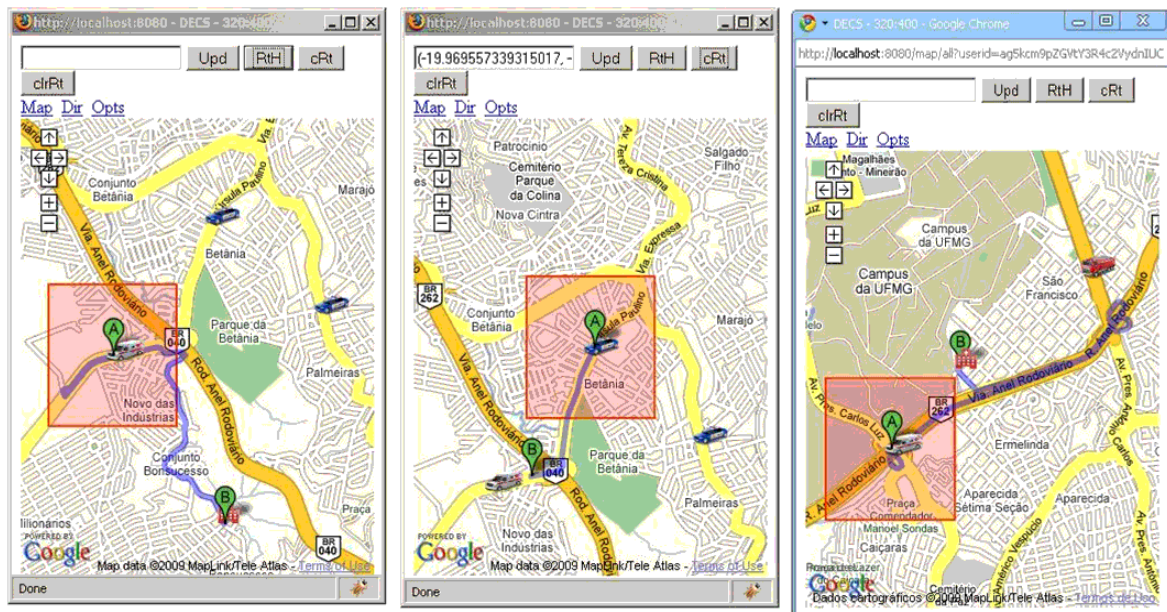


Figura 5.7. Execução do protótipo DECS em contexto coletivo.

apresentadas neste trabalho, a limitação na linguagem de programação apresentou algumas limitações ao projeto. Uma destas inclui a dificuldade no desenvolvimento de testes de implementação em módulos e funcionalidades específicas. A linguagem *Python* não provia suporte à implementação de testes automatizados sobre as funcionalidades desenvolvidas, dificultando assim o desenvolvimento do sistema como um todo. Um outro fator está no número limitado de ferramentas de desenvolvimento para a linguagem *Python* em comparação ao ambiente de desenvolvimento Java. As limitações acima também se apresentaram durante o processo de integração e testes dos módulos desenvolvidos no lado do servidor do guia turístico *DroidGuide*.

No início de 2009, uma nova versão do GAE foi disponibilizada com suporte à linguagem de programação Java. Este suporte adicional possibilitou o desenvolvimento do servidor de dados nesta linguagem, facilitando seu desenvolvimento e possibilitando o desenvolvimento de testes automatizados. Além deste suporte, o GAE foi integrado com outro arcabouço de desenvolvimento Web: o *Google Web Toolkit*, arcabouço que permite a construção de aplicações Web baseadas em *JavaScript*. Desta forma, isto possibilitou o uso do GWT no desenvolvimento de clientes Web leves para acesso ao servidor de eventos proposto neste trabalho.

Durante o desenvolvimento da segunda versão do protótipo DECS, migramos o lado cliente para o GWT. Esta migração trouxe vários benefícios no que diz respeito ao desenvolvimento das interfaces necessárias para a aplicação, já que a integração

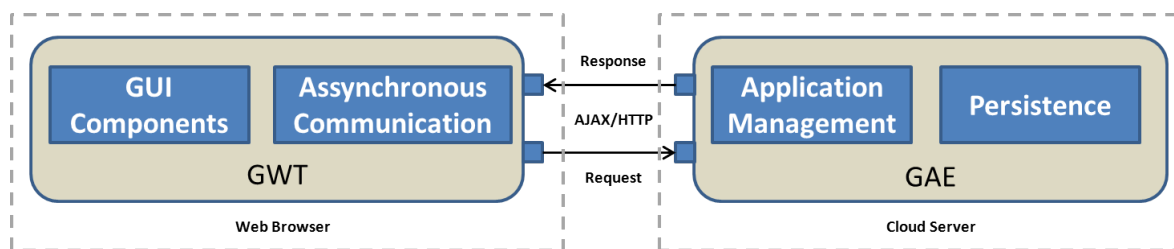


Figura 5.8. Interação entre os arcabouços GWT e GAE no protótipo DECS.

entre os arcabouços acima é bastante direta. O GWT fornece as interfaces Web e o mecanismo de comunicação assíncrono no lado cliente para a execução de chamadas remotas de procedimentos no servidor. O GAE fornece os serviços Web necessários (e.g., gerenciadores de aplicações, persistência, servidor de eventos e tratamento de requisições assíncronas) para a aplicação no lado servidor.

Outro benefício está no uso da API disponível no GWT de mapas *Google Maps API* disponível na linguagem de programação Java. Ela é utilizada na construção e gestão de marcos e rotas em uma determinada região geográfica definida no mapa. Além desta fornecer uma quantidade maior de recursos no GWT em relação à API em *JavaScript* usada no primeiro protótipo, a sua utilização no desenvolvimento tornou-se mais simples e gerenciável, já que ela utiliza o mesmo ambiente de desenvolvimento do GWT. Com a utilização da API de mapas, foi possível a aquisição de informações de rotas importantes entre marcos ou pontos definidos no mapa. Algumas das informações obtidas incluem a distância, tempo médio gasto para chegar ao destino e, mais importante, a listagem dos passos necessários para a chegada do marco móvel até o seu destino. Estes passos são compostos por um conjunto de linhas ou "polilinhas" (em inglês, *polylines*) criadas entre um ponto de partida e chegada. Com estas "polilinhas", foi possível a simulação da movimentação de marcos na interface de mapas a partir das rotas informadas pelo serviço de mapas na Web.

Consideramos neste protótipo a comunicação entre clientes como sendo uma importante funcionalidade a ser disponibilizada, com o objetivo de enfatizarmos a interação entre usuários no sistema. No DECS, a comunicação entre os *peers* é feita indiretamente através de um encaminhamento de mensagens pelo servidor de dados remoto e pelo servidor de eventos. Neste caso, o cliente solicita o envio de uma mensagem através da definição do(s) usuário(s) receptor(es), do tópico (título da mensagem) e do conteúdo da mesma. A partir do tópico definido, o servidor de eventos recebe o evento e notifica o cliente destino em forma de mensagem. O sistema transmite as mensagens entre clientes a partir de requisições e respostas submetidas entre o cliente e o servi-

dor remoto de dados. Neste caso, o cliente é responsável por remeter as mensagens enquanto o servidor de dados armazena e repassa as mesmas para os clientes respectivos. Todo este procedimento ocorre durante o processamento de eventos no servidor, apresentado na seção 3.6.

5.3 Arquitetura

O DECS utiliza uma arquitetura cliente/servidor composta de clientes leves construídos a partir de tecnologias Web e um servidor de dados em nuvem. Estes clientes leves executam sobre um navegador Web suas funcionalidades e acessam o servidor através do envio de requisições HTTP assíncronas (AJAX), conforme mostrado na Figura 5.9. O servidor de dados retorna os dados requisitados para o cliente em forma de mensagens em XML sobre a resposta HTTP. Os principais módulos que compõem a arquitetura serão apresentados na subseção seguinte.

5.3.1 Componentes

O primeiro componente utilizado pelo DECS é o navegador Web ou *Web Browser*. Ele é responsável por prover a infra-estrutura de interface gráfica e comunicação de dados entre os clientes e o servidor de dados. A partir do navegador Web, o usuário móvel utiliza o serviço de contexto de emergências, o servidor de eventos e a interface de mapas provida pela API do *Google Maps*. Um dos grandes benefícios em se utilizar o navegador Web como interface com o usuário está na sua disponibilidade em diversos sistemas computacionais e a escalabilidade, já que cada janela de um navegador Web tem o potencial de se empenhar como um cliente utilizando os serviços oferecidos pelo DECS.

O *DECSContext* ou contexto do DECS é o principal componente presente na aplicação cliente. Ele é responsável pela inicialização e controle de estados e objetos localizados no cliente Web. Ele fornece acesso ao *proxy* de acesso aos serviços disponíveis no servidor e controla os estados dos componentes de interface gráfica (e.g., painéis, comandos, ações, dentre outros) da aplicação no cliente. Relacionado com o contexto do DECS, o objeto *EntryPoint* define o ponto de entrada ou inicialização de toda a aplicação cliente que utiliza o arcabouço GWT. No caso do DECS, o *EntryPoint* aciona o *DECSContext* responsável pela inicialização da aplicação cliente.

No lado do cliente, diversas classes auxiliam nas operações de interface gráfica e comandos. O componente *Panel* é responsável por prover as interfaces gráficas de acesso ao usuário, tais como mapas, listagem de eventos e *log*, mensagens, usuários

conectados, detalhes de rota, dentre outros. Todos os componentes de painel utilizados pela aplicação herdam deste componente. O *ClickHandler* é responsável pelo gerenciamento de eventos de botões gerados pelo usuário móvel. Todos os eventos de *click* herdam atributos e operações deste componente, tais como o método *onClick()* que deve ser implementado. O Componente *Action* executa ações em função de eventos gerados pelo usuário ou pela própria aplicação. Todas as ações executadas no cliente leve herdam deste componente. Em alguns casos o componente de ação também é responsável por tratar a resposta (e.g., *callback*) enviada a partir do servidor remoto de dados. Apresentaremos na seção 5.4.2 maiores detalhes referentes a execução de ações no protótipo DECS. O *Callback* é responsável por tratar respostas assíncronas enviadas por componentes no servidor de dados para serem processadas no cliente leve. O tratamento pode possuir dois resultados: bem sucedido ou mal sucedido. O resultado bem sucedido demonstra que a requisição foi processada de forma correta no servidor de dados. O resultado mal sucedido ocorre em casos de erros ocorridos no cliente ou servidor.

O *proxy* de serviços e as filas de objetos também cumprem papéis importantes na aplicação. O *ServiceProxy* é responsável por prover acesso aos serviços (gerenciadores) disponíveis no servidor de dados de forma assíncrona. Os métodos disponíveis para acesso são definidos a partir de interfaces síncronas e assíncronas usadas tanto no cliente Web quanto nos serviços disponibilizados no servidor de dados. As interfaces síncronas definem os métodos em si no lado do servidor enquanto as interfaces assíncronas definem os métodos no lado do cliente, utilizando mecanismos de *callback* para o tratamento do retorno de requisições assíncronas solicitadas pelo cliente.

As filas de objetos ou *Queue list* são responsáveis por manter uma listagem de objetos no lado do cliente para processamento local. Estes elementos incluem eventos gerados, mensagens recebidas do servidor remoto de dados, dados estatísticos, rotas requisitadas, usuários conectados no serviço e eventos de emergência referentes ao usuário. O principal objetivo das listas é permitir que o cliente tenha acesso à informações referentes ao contexto da aplicação sem a necessidade de requisitar frequentemente ao servidor remoto de dados por estas informações. Para cada um dos tipos apresentados acima, o cliente possui uma lista onde elementos são inseridos ou removidos durante a execução da aplicação.

Na arquitetura apresentada, alguns atores atuam sobre o sistema de atendimento a emergências. Conforme apresentado na Figura 5.9, os principais atores são: Paciente ou Vítima (PoV), Unidade Móvel de Atendimento (UnM) e Marco Fixo ou *Landmark* (LnM). O PoV representa a entidade a ser resgatada pela unidade móvel de atendimento de emergência (UnM), podendo ser um indivíduo ou um grupo de indi-

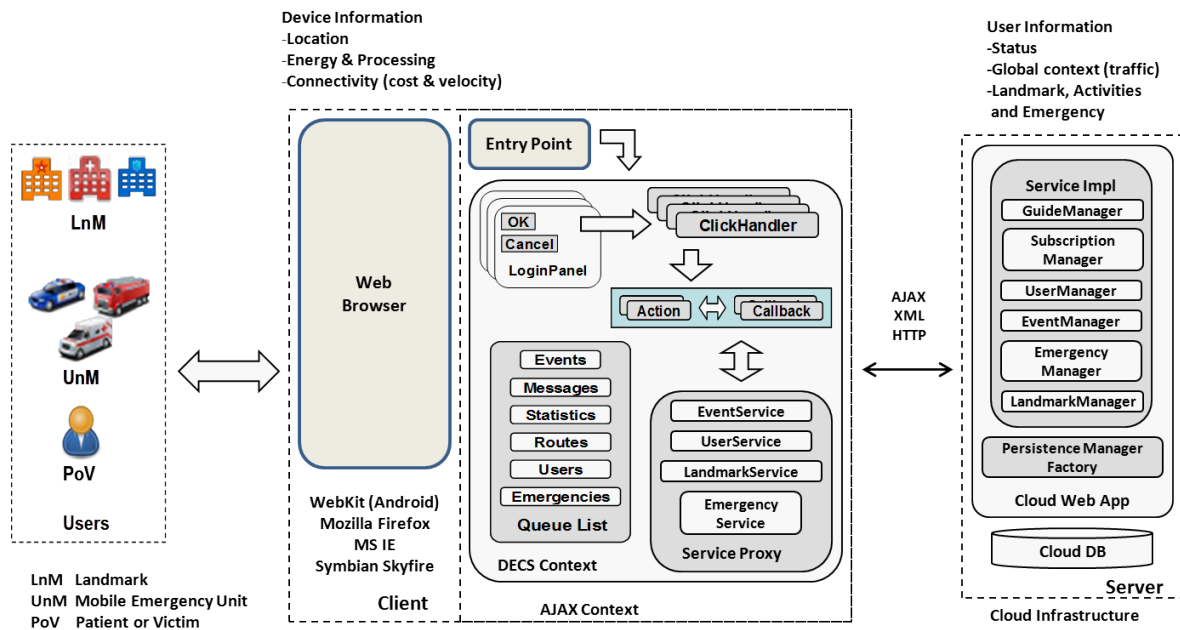


Figura 5.9. Arquitetura do protótipo DECS 2.0.

vídeos. A UnM é responsável por se locomover até ao local da emergência de suas responsabilidade, categorizadas em três tipos (e.g., saúde, segurança e incêndios): ambulância, viatura policial e viatura dos bombeiros. O LnM representa os marcos fixos no sistema responsáveis em receber PoVs de UnMs.

5.4 Detalhes de Implementação

Esta seção tem como objetivo apresentar os detalhes de implementação do serviço DECS, incluindo seus componentes que incluem o servidor de eventos proposto no capítulo 3 deste trabalho.

5.4.1 Localização de Marcos Fixos, Móveis e Rotas

Neste protótipo, era necessário que as UnMs pudessem enviar informações atualizadas de sua localização para o servidor de eventos a fim de possibilitar a execução de diversas atividades, tais como a seleção da UnM mais próxima do evento de emergência, a seleção do marco fixo mais próximo a UnM, dentre outras. Para isto, utilizamos o próprio serviço do *Google Maps* para a aquisição das coordenadas de cada um dos pontos e, desta forma, foi possível a seleção de marcos baseados em sua localização e

a definição de rotas entre os pontos do mapa. A partir da definição da posição dos marcos (fixos e móveis), foi possível renderizá-los na interface de mapa utilizada pela aplicação cliente.

A definição de rotas do serviço do Google Maps funciona de forma bem simples. Dados dois pontos como sendo a origem e o destino, o cliente envia uma *query* em *String* contendo as coordenadas destes pontos no formato específico (e.g., latitude e longitude) para o serviço. O serviço envia a resposta para o cliente contendo diversas informações, tais como a distância e a duração entre os pontos considerando a rota apresentada, a quantidade de linhas que compõem a rota (e.g., *polylines*) e uma renderização gráfica da rota retornada em um painel da interface do tipo *MapWidget* no arcabouço GWT. O serviço de mapas também oferece a busca a partir do nome do ponto, podendo este ser o nome da rua ou avenida. Isto é possível graças ao serviço de *GeoLocation* (*Google Geolocation API*) responsável por converter nomes de locais em coordenadas (e.g., latitude e longitude).

5.4.2 Ações, Tratamento de Eventos e de Retorno

Na emulação do serviço de atendimento a emergências, implementamos o comportamento a ser executado pelas diversas entidades no sistema através de ações. Dividimos o tratamento deste comportamento em duas partes: (a) no lado do cliente e (b) no lado do servidor. No cliente, as entidades possuem o comportamento de movimentação e o processamento de eventos implementado. Desenvolvemos no protótipo os comportamentos dos componentes utilizando uma comunicação de forma assíncrona (AJAX) através de três principais conceitos: (a) ações ou comandos de execução (*Action*), (b) tratadores de eventos (*Handler*) e (c) tratadores de retorno (*Callback*).

No DECS, cada ação é responsável por executar uma tarefa ou atividade e tratar sua resposta, caso haja alguma requisição a ser enviada para o servidor remoto. Por exemplo, para carregar as entidades do sistema na interface do mapa, a ação responsável pela execução desta tarefa solicita ao servidor remoto de dados a listagem de todas as entidades que devem ser renderizadas no mapa. Ao receber a listagem das entidades no formato XML, a ação processa a mensagem recebida de tal forma a possibilitar a leitura dos dados recebidos do servidor e a renderização das entidades na interface da mapa localizada no cliente. Uma visão geral das classes e interfaces presentes no sistema pode ser visualizada nas Figuras 5.10, 5.11 e 5.12.

No caso do serviço de contexto para emergências, a cada requisição e mudança de estado, o cliente Web recebe uma mensagem no formato XML. Esta mensagem contém os dados necessários para a execução das atividades a serem empenhadas, como por

exemplo, a movimentação da entidade móvel de um ponto a outro no mapa. Além da recepção dos dados vindos do servidor remoto, o cliente também envia informações de perfil e contexto da entidade móvel localizada no mapa. Por exemplo, ao chegar em um destino, podendo ser o evento de emergência ou marco fixo, o cliente Web envia para o servidor remoto de dados informações para que o servidor possa determinar os próximos passos a serem executados pelas entidades apresentadas no cliente Web. A Figura 5.11 apresenta as principais classes que fazem parte das ações de emergência do serviço DECS.

Os tratadores de eventos ou *Handlers* são responsáveis por tratar cliques de comandos efetuados pelo usuário. Estes tratadores podem acionar ou não ações definidas no sistema, como por exemplo, a atualização das entidades no mapa. Esta atualização exigirá a execução da ação responsável por requisitar a lista de entidades do servidor e processar a mensagem recebida do servidor. Além dos tratadores de comandos, existem também os tratadores relacionados à eventos presentes na interface do mapa. Estes tratadores têm como objetivo capturar eventos gerados pelo mapa e seus elementos. Exemplos de eventos no mapa incluem o clique na área do mapa para a criação de um novo evento e o clique direto em uma entidade renderizada no mapa. A figura 5.10 apresenta as principais classes que compõem os tratadores de eventos.

Os tratadores de retorno ou *Callbacks* são responsáveis por tratar mensagens de retorno vindas do servidor remoto. Eles são utilizados em casos onde a lógica de processamento da mensagem recebida é um pouco mais complexa, exigindo uma separação lógica entre a ação de envio de dados e o tratamento da resposta entre o cliente Web e o servidor remoto de dados. Neste caso, para cada ação de execução existe um objeto que implementa a interface *Action* e um outro objeto que implementa a interface *AsyncCallback*. Ao criar o objeto responsável pela ação, informamos o objeto responsável pelo tratamento do retorno de dados do servidor remoto, passando assim a referência dele para o objeto responsável pela ação.

5.4.3 Processamento de Mensagens em XML

Na comunicação entre o cliente o servidor remoto de dados, o DECS utiliza na requisição a passagem de parâmetros do tipo *String* e no retorno o envio de documentos no formato XML sobre a resposta HTTP. Para o processamento das mensagens enviadas do servidor para o cliente, utilizamos o processador de XML *XMLParser* disponível no GWT para a leitura das propriedades de entidades representadas no documento. A partir da leitura do documento XML retornado, transferimos as propriedades encontradas no documento para objetos no cliente responsáveis por representar cada uma

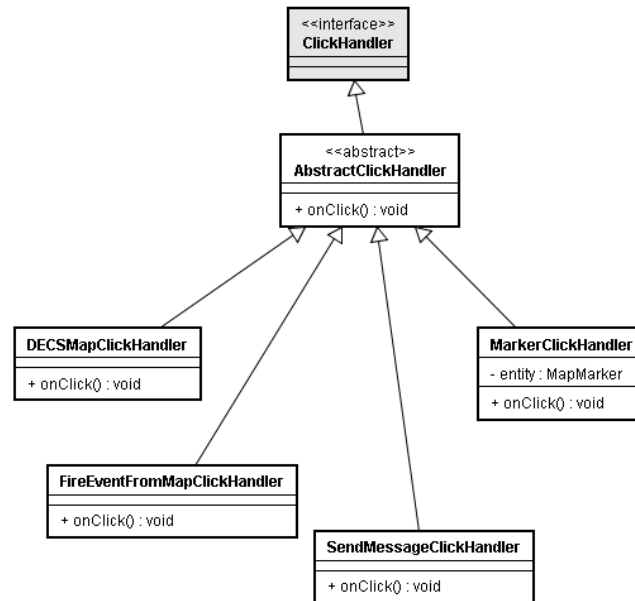


Figura 5.10. Diagrama resumido de classes do Sistema DECS relativo ao tratamento de eventos.

das entidades. Estes objetos podem ser eventos gerais do sistema (*EventMessage*), eventos de emergência (*EmergencyEvent*), usuários móveis e fixos. No tratamento das emergências, por exemplo, cada um dos objetos (e.g., evento de emergência, usuário móvel e marco fixo) estão associados a uma ação de execução, representados na classe *AbstractEmergencyAction*, conforme visualizado na Figura 5.11. Isto possibilita a busca pelo evento de emergência corrente, usuário móvel responsável e marco fixo relacionado e mantém informações de contexto atualizadas para cada ação em execução.

Para a construção de uma aplicação ubíqua qualquer utilizando o conjunto de tecnologias utilizadas pelo serviço DECS, é necessária a definição das operações tanto no lado cliente quanto no lado servidor. Devido ao fato de cada ação (*Action*) ou retorno (*Callback*) possuir um objetivo claro e específico, é necessário que o desenvolvedor implemente seu comportamento, já que o tratamento de mensagens deve ser implementado de forma explícita na recepção dos dados vindos do servidor. Por exemplo, no caso do DECS, quando se espera a seleção de uma unidade móvel para o atendimento de um evento de emergência, teremos duas situações possíveis. A primeira define um retorno de um identificador de uma unidade móvel para casos onde haja uma unidade móvel de emergência disponível e a segunda um retorno sem um nome definido. Sendo assim, caso haja um retorno bem sucedido, o tratamento da resposta da ação deverá executar a próxima ação que despachará a unidade móvel para o evento. Caso

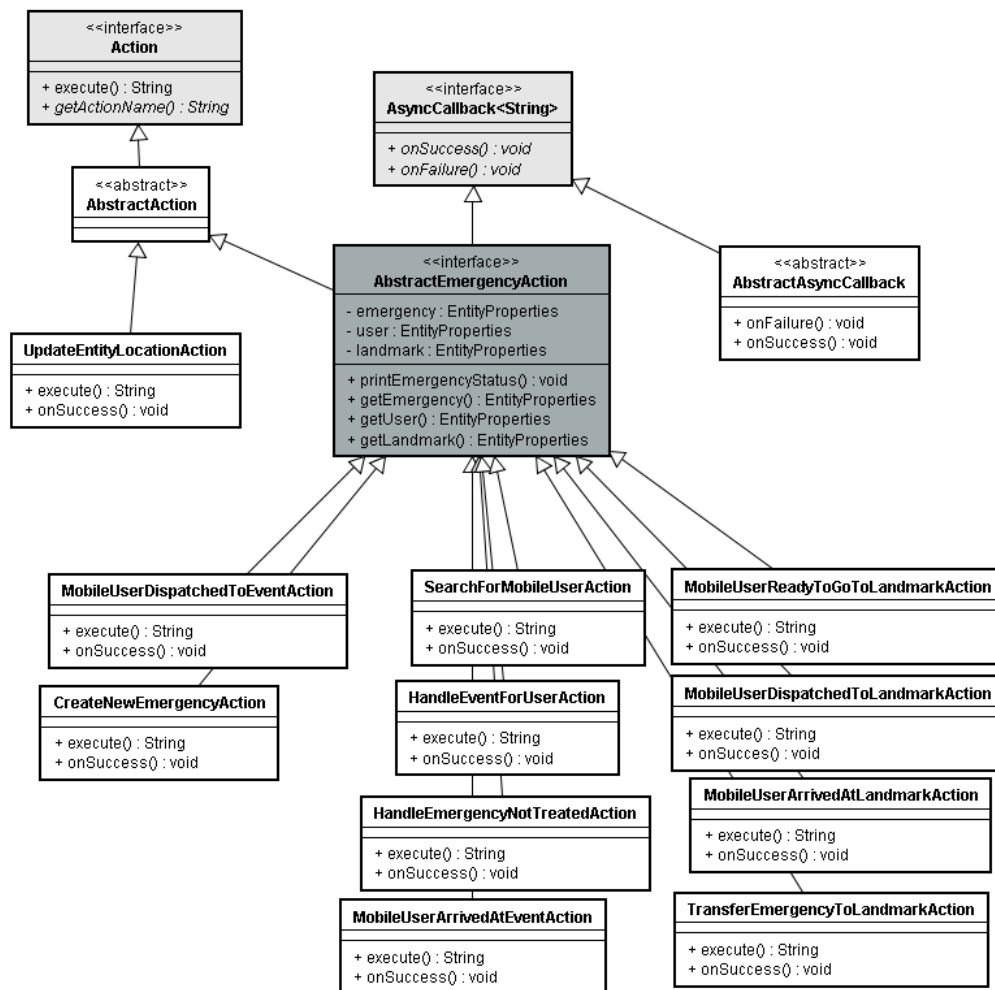


Figura 5.11. Diagrama resumido de classes do Sistema DECS relativo às ações de emergência.

contrário, o tratamento da resposta deve notificar o serviço de contexto de emergências de que não tratará o evento de emergência neste momento, por exemplo, devido à indisponibilidade das unidades móveis logadas no sistema naquele momento.

5.4.4 Modos de Execução

Para fins de depuração e testes do serviço DECS e do servidor de eventos proposto neste trabalho, implementamos a execução do cliente Web de duas formas: (a) manual, onde o usuário informa à aplicação o local onde o evento de emergência ocorreu e (b) automático, onde o cliente Web gera de forma pseudo-aleatória as posições dos eventos de emergência a serem tratados pelo serviço de contexto de emergências.

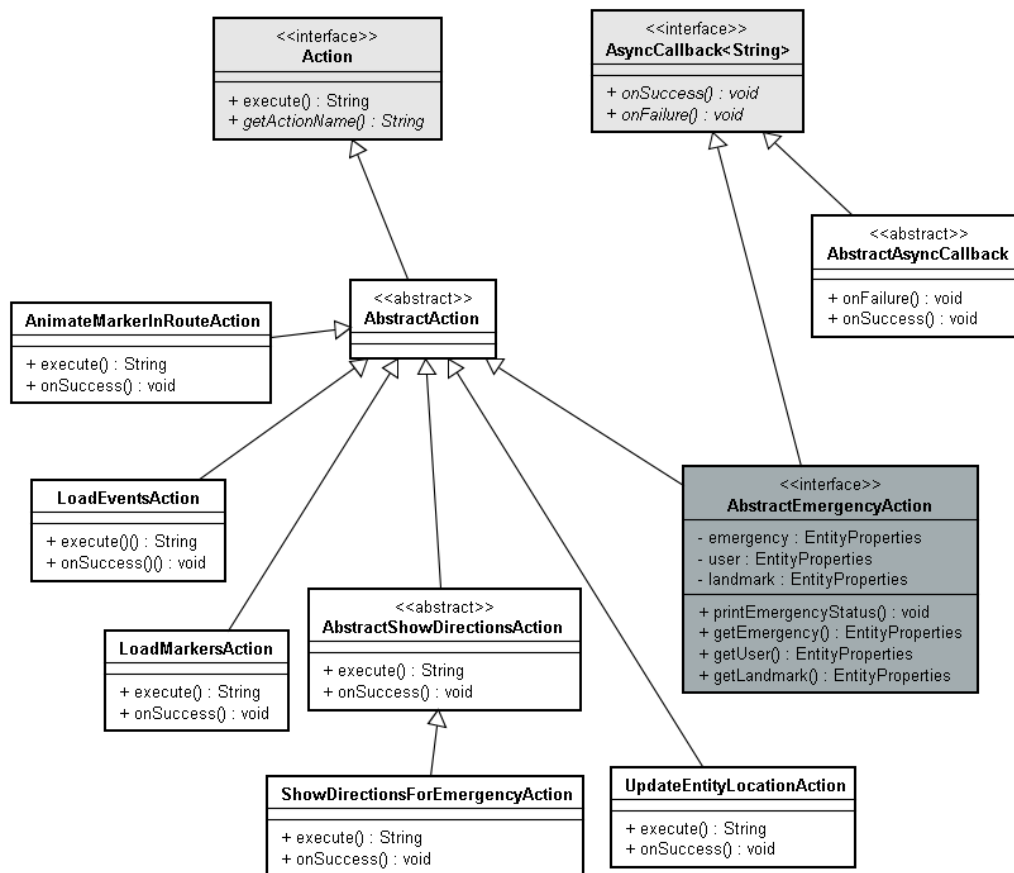


Figura 5.12. Diagrama resumido de classes do Sistema DECS relativo aos outros tipos de ações.

Para facilitar a avaliação do serviço no modo manual, criamos um menu de aplicação que permite que o usuário crie um evento de emergência de uma maneira bem simplificada. Primeiro, é necessário que o usuário marque uma posição no mapa apresentado e logo após selecionar a opção *Option* no menu e depois uma das três opções disponíveis: a) *New Emergency - Health*, b) *New Emergency - Security* e c) *New Emergency - Fire*. Cada uma destas iniciará o processo de gerenciamento de eventos de emergência criados para a busca e depois a entrega dos mesmos nos marcos apropriados (e.g., hospital ou delegacia).

No serviço de gerenciamento de emergências, o servidor de eventos é responsável por gerar os eventos apropriados e acionar determinadas operações definidas do serviço de emergências, que vão desde a criação de um evento de emergência até a transferência deste para o marco respectivo. Para cada evento de emergência criado, o servidor de eventos cria no mínimo dez diferentes tipos de mensagens de notificação, cada uma representando uma fase sendo processada pelo serviço de emergências. Estas men-

Tabela 5.1. Coordenadas da área de cobertura do serviço DECS no modo automático de execução.

Ponto	Latitude	Longitude	Latitude Real	Longitude Real
1-NW (Noroeste)	-19.726634950986692	-44.138946533203125	-19° 43' 35.8824"	-44° 8' 20.205"
2-NE (Nordeste)	-19.744085416705005	-43.87596130371094	-19° 44' 38.7054"	-43° 52' 33.459"
3-SW (Sudoeste)	-19.992707943572263	-44.149932861328125	-19° 59' 33.7446"	-44° 8' 59.7582"
4-SE (Sudeste)	-20.01464544534135	-43.8848876953125	-20° 0' 52.722"	-43° 53' 5.5962"

sagens são criadas pelo serviço de emergências durante a execução das fases do fluxo de atendimento à emergência. No final de cada atendimento, o serviço de emergências atualiza as informações apresentadas no mapa georeferenciado, de tal forma a possibilitar um novo atendimento a uma futura emergência. A futura emergência deve ser criada novamente clicando no local onde esta é solicitada e acionando o comando de criação da mesma.

No modo automático, uma ação (i.e., comando de execução no cliente) inicia o fluxo de atendimento de emergências através da criação de um ponto no mapa utilizando coordenadas geradas de forma pseudo-aleatória. Este ponto é criado sobre uma área pré-determinada do mapa na região em foco no nosso caso, a cidade de Belo Horizonte. As coordenadas definidas no trabalho estão apresentadas na Tabela 5.1 e a área de cobertura respectiva pode ser visualizada na Figura 5.13.

Após a criação do evento de emergência sobre o mapa de forma automática, o fluxo de execução entra em ação da mesma forma que no modo manual. Após a conclusão do atendimento da emergência pelo serviço, o gerador automático de pontos no mapa é novamente chamado para criar um novo evento para ser tratado pelo serviço. É possível, por exemplo, para alguns casos que a coordenada gerada não seja alcançável por unidades móveis de emergência. Isto ocorre devido ao fato do serviço de rotas do *Google Maps* não encontrar uma rota ou caminho até a posição indicada. Nestes casos, a aplicação cliente notifica o serviço de atendimento de emergências a impossibilidade de atendimento do evento. O serviço então libera a alocação da UnM previamente selecionada e encerra o ciclo de atendimento deste evento de emergência em questão.

5.4.5 Publicação de Eventos e Mensagens

Além da tarefa de gerenciamento das fases do serviço de emergências, o servidor de eventos também efetua duas operações: a) publicação de eventos baseados em uma determinada localização no mapa e b) envio de mensagens por usuários logados no sistema. O envio de mensagens permite que o usuário defina o tópico a ser publicado e o conteúdo do evento associado, que inclui um conjunto de propriedades e valores. Baseado no tópico publicado, o servidor de eventos publica o evento para os clientes

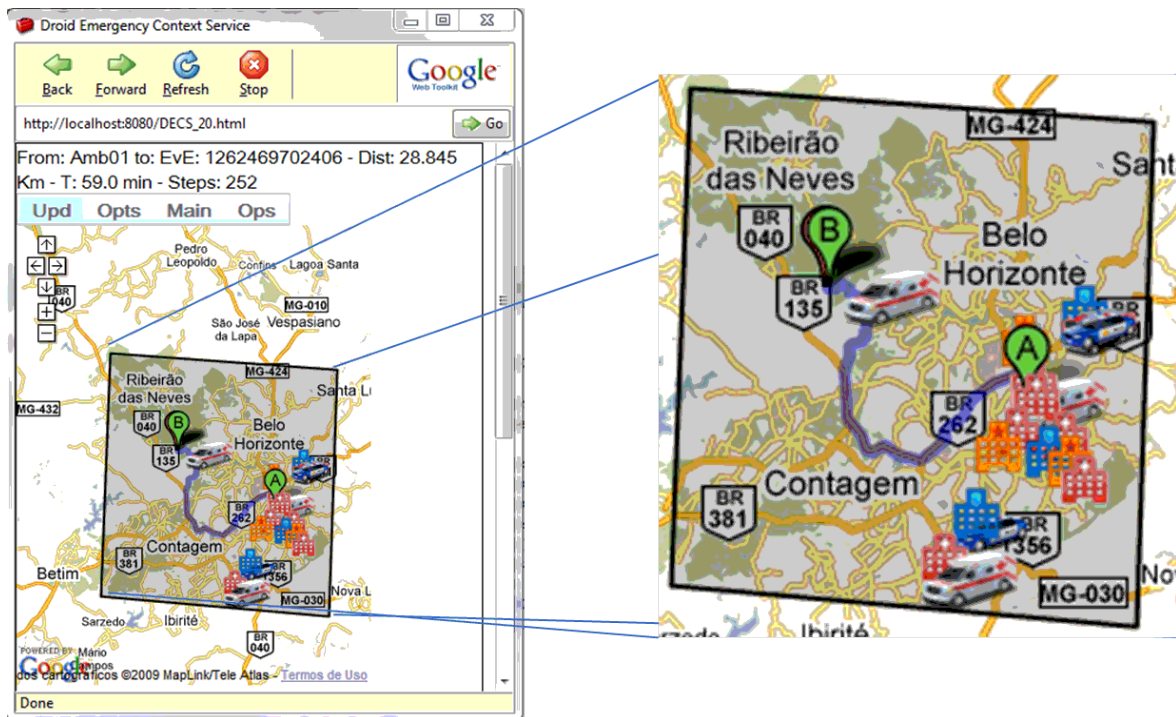
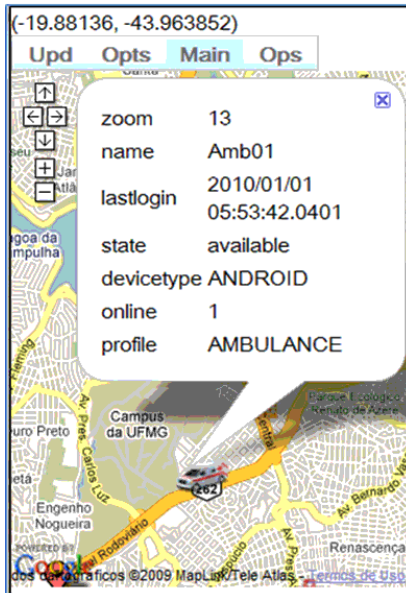


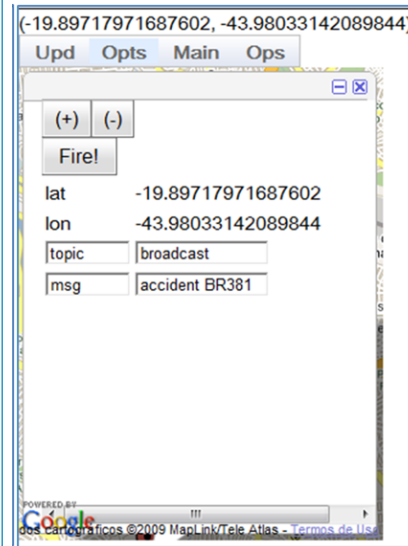
Figura 5.13. Visualização da área de cobertura do serviço de atendimento de emergências no modo automático de execução.

que possuem assinatura ao tópico relacionado. Por exemplo, este recurso permite que um determinado usuário do sistema (e.g., ambulância) possa informar outros usuários sobre um determinado evento, acontecimento ou problema em uma das rotas utilizadas por unidades móveis de emergência para atender pacientes ou vítimas. Neste caso, as unidades móveis relacionadas ao tópico publicado (consumidores) recebem uma mensagem contendo informações relevantes enviadas pelo usuário do sistema, neste caso o produtor do evento. A publicação de eventos por entidades móveis no DECS pode ser visualizada na Figura 5.14.

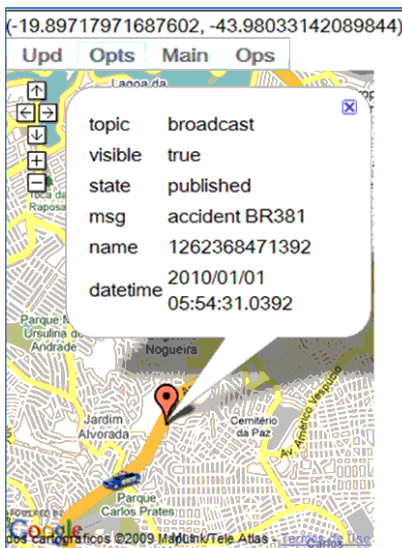
Além da publicação de eventos baseados em localização, o protótipo desenvolvido também suporta o envio de mensagens entre usuários, também apresentado na Figura 5.15. O envio direto de mensagens para usuários ou para um grupo de usuários fornece um importante recurso de comunicação entre as entidades no sistema, permitindo que unidades móveis enviem mensagens entre si ou até para os marcos fixos relacionados, tais como hospitais ou delegacias. No envio de mensagens, o servidor de eventos processa a solicitação de envio de uma mensagem em forma de evento a ser publicado no sistema, como um outro evento qualquer. A grande vantagem desta abordagem está na simplicidade do envio de mensagens e no fato de não diferenciarmos o en-



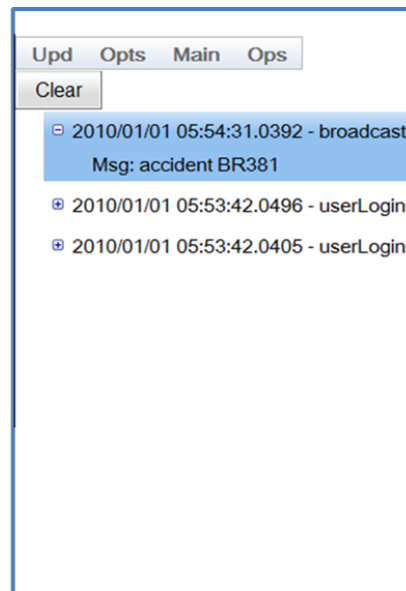
(a) Detalhes de uma UnM.



(b) Novo evento criado na localização.



(c) Detalhes de um evento publicado.



(d) Listagem dos eventos gerados.

Figura 5.14. Publicação e visualização de eventos relacionados à localização.

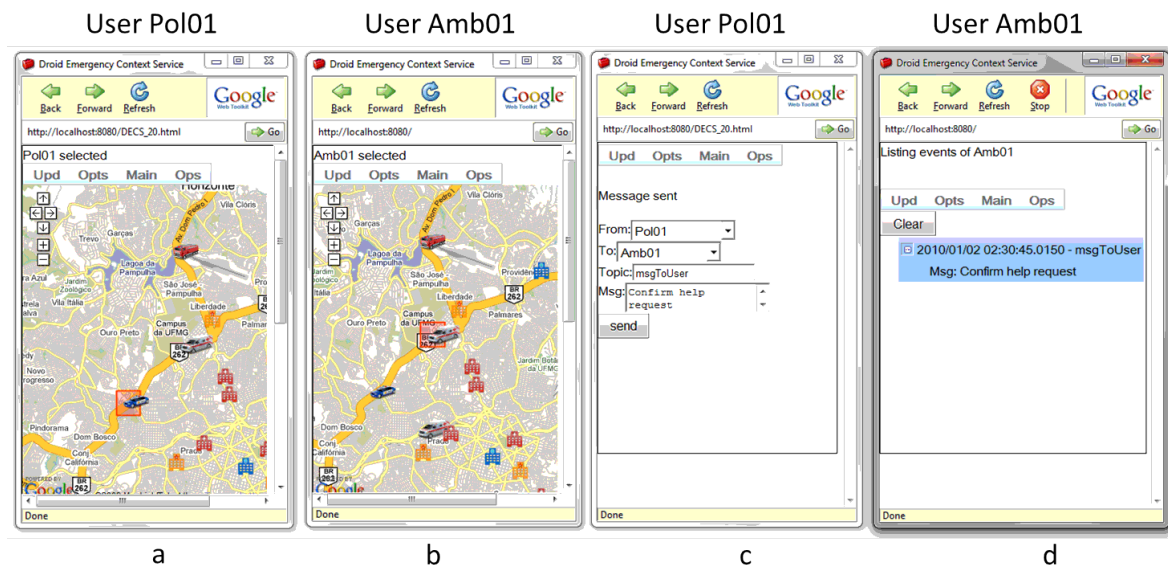


Figura 5.15. Demonstração do envio de mensagens entre usuários. Em (a) e (b), a seleção dos usuários no contexto de execução. Em (c), o envio da mensagem para um usuário específico. Em (d), a listagem de eventos apresentando um evento relacionado à mensagem enviada para o usuário específico.

vio de mensagens ou eventos para consumidores. Sendo assim, o sistema permite que produtores publiquem eventos e que consumidores os consumam, independente de serem mensagens ou eventos. Para isto, todos os usuários logados no sistema se inscrevem em diversos tópicos de forma implícita ou explícita. A inscrita implícita é realizada pelo próprio serviço DECS, já que estes deverão desempenhar atividades pré-estabelecidas, tais como receber mensagens de usuários específicos ou mensagens do tipo *broadcast/unicast* e eventos relacionados às mudanças de estados durante o tratamento da emergência tanto pela UnM quanto pelo marco fixo durante a execução da aplicação.

No processo de publicação e notificação de mensagens entre usuários, o servidor de eventos é responsável por gerir a mensagem, publicando-a para os consumidores respectivos. A solicitação de envio de uma mensagem é recebida pelo servidor de eventos no lado servidor, onde processa e publica eventos de acordo com o tópico informado pelo produtor da mensagem. Neste protótipo, três formas de envio de mensagens entre usuários são suportadas: (a) o envio para um usuário específico (e.g., *unicast*), (b) o envio para um grupo de usuários (e.g., *multicast*) e (c) o envio para todos os usuários logados do sistema (e.g., *broadcast*). Para o envio de mensagens com cada um dos tópicos apresentados acima, o produtor utiliza três tópicos pré-definidos: *msgToUser*,

multicast e *broadcast*. Destes três tópicos, apenas o último não necessita informar o destino, já que a mensagem se destina para todos os usuários no sistema, conforme apresentado na Figura 5.15. Para mensagens com o tópico *msgToUser*, o usuário destino deve ser informado (e.g., *Amb01*, *Pol01*, *Fireman01*), enquanto para o tópico *multicast* é necessário informar o grupo de usuários destino (e.g., *AMBULANCE*, *FIREMAN*, *POLICE*).

5.4.6 Execução em Contexto Coletivo

Além da execução em um único cliente do serviço DECS, o protótipo desenvolvido também permite a execução simultânea de mais de um cliente Web, conforme mostrado na Figura 5.16. Neste cenário, definimos três clientes: uma ambulância, uma viatura do corpo de bombeiros e da polícia militar. Para cada um dos clientes, criamos um evento de emergência para cada tipo de cliente: uma emergência relacionada à saúde, outra relacionada à segurança e a última relacionada à um incêndio. Após a criação de cada um dos eventos, o serviço de atendimento de emergências inicia o fluxo de execução para cada um dos clientes em execução. No contexto coletivo, o cliente Web trata as emergências e os dados estatísticos de forma isolada, compartilhando somente o servidor de dados que contém o servidor de eventos, os containeres de eventos e de subscrições. Neste caso, os dados estatísticos no lado do servidor são únicos, já que os clientes Web acessam o único servidor remoto. A execução simultânea de clientes Web permite a publicação de eventos e o envio de mensagens entre usuários, conforme já apresentado nas Figuras 5.14 e 5.15.

No final de cada fluxo de execução do serviço de atendimento de emergências, o cliente Web requisita uma atualização das entidades no mapa georeferenciado, possibilitando uma visualização da posição atual das UnMs do sistema. A Figura 5.17 apresenta o resultado final da execução de cada um dos clientes após o atendimento a cada um dos tipos de eventos de emergências e a entrega destes aos seus respectivos marcos fixos. Nesta figura, apresentamos a localização final de cada uma das UnMs do sistema, incluindo as que foram acionadas para atender aos seus respectivos eventos de emergência. É possível notar, por exemplo, a localização de outros clientes presentes na tela do cliente em foco, tais como o cliente *Amb01* na tela do cliente *Pol01*, o cliente *Bom01* presente na tela do cliente *Amb01* e vice-versa.

O módulo estatístico utilizado neste trabalho é composto de um *container* denominado *Statistics Container* (SC), conforme apresentado no diagrama de classes da Figura 5.18(a). Este *container* é formado por uma tabela do tipo *hash* composta por chaves que representam as variáveis a serem contabilizadas e valores representados pela

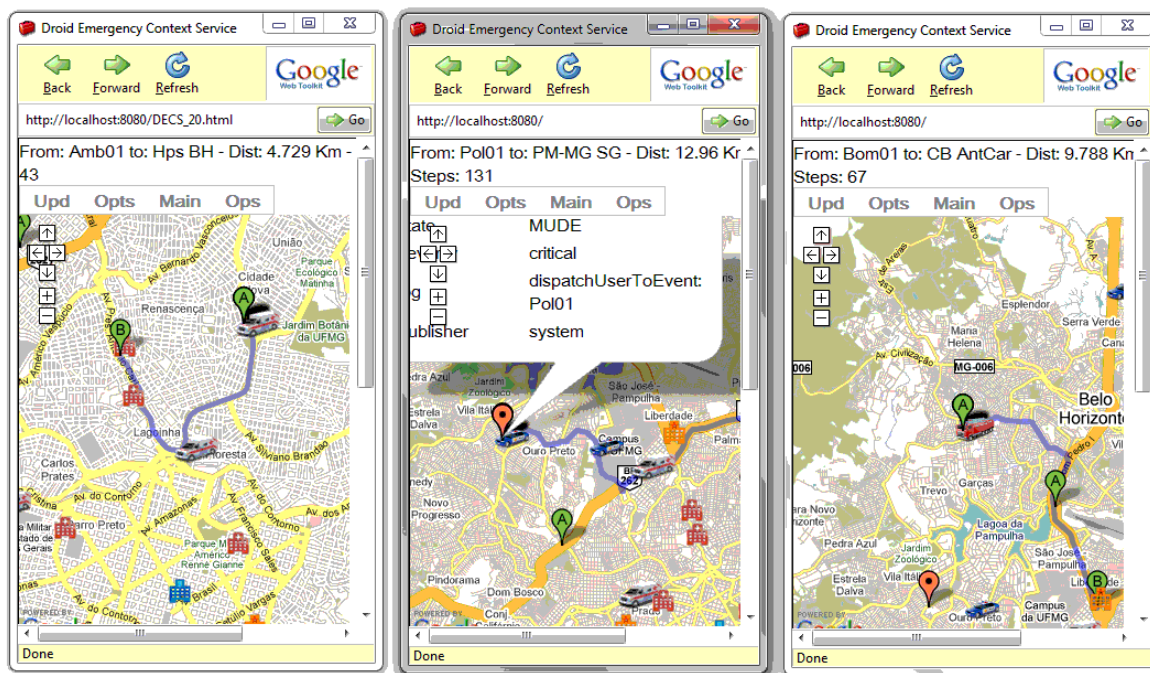


Figura 5.16. Execução do serviço DECS no modo coletivo com três clientes acessando um servidor.

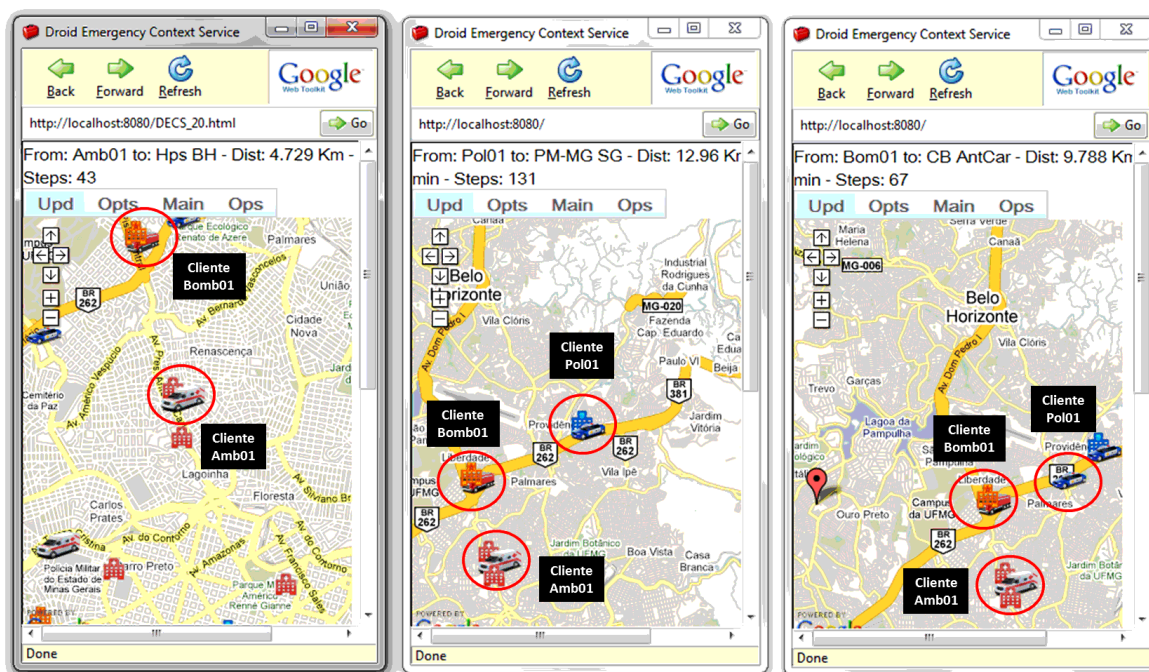
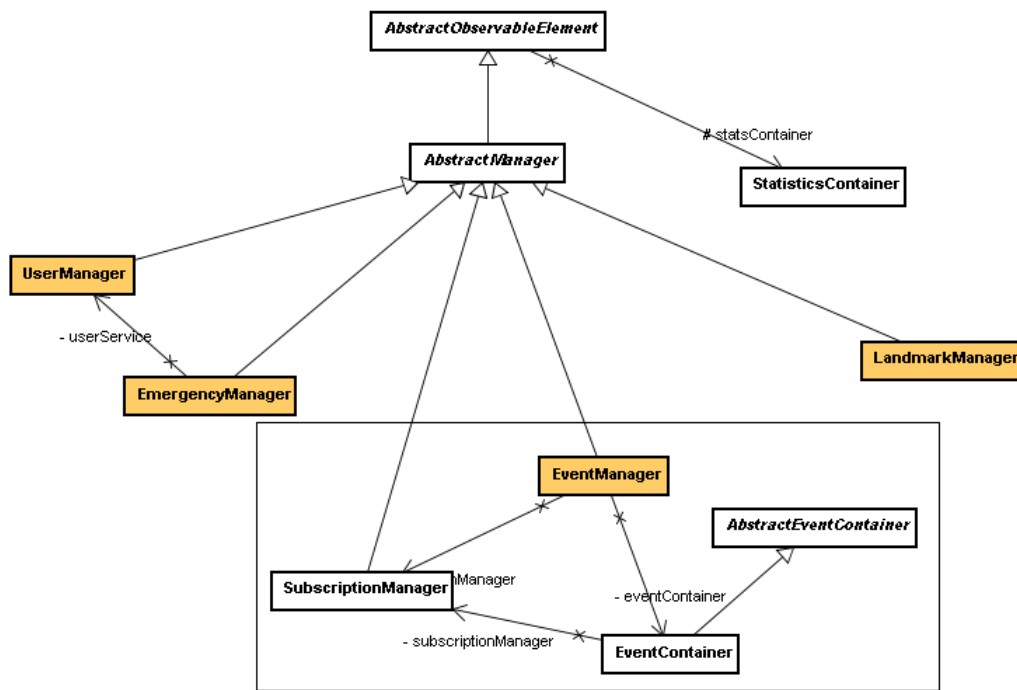


Figura 5.17. Final da execução do serviço DECS no modo coletivo com três clientes acessando um servidor, após o atendimento a um evento de emergência.

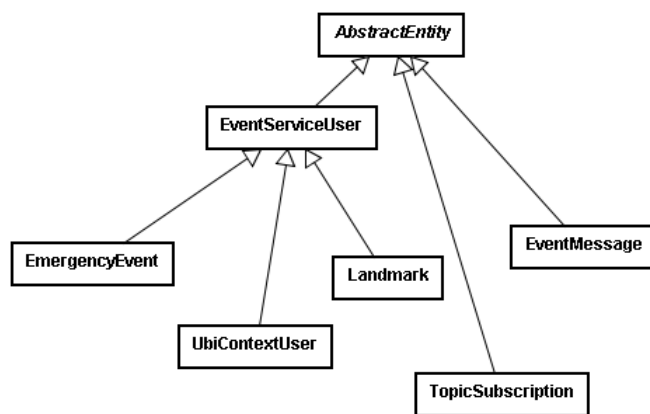
quantidade ou valor a ser armazenado, como por exemplo, o número de usuários criados no sistema, o número total de eventos publicados, dentre outros. Os principais componentes do protótipo DECS possuem uma instância do SC, tais como o cliente Web (classe *DECSContext*) e os componentes no lado do servidor de dados (classes *EmergencyManager*, *EventManager*, *SubscriptionManager*, *EventContainer*, *LandmarkManager*, e *UserManager*), conforme apresentado na Figura 5.4.6. Na execução em contexto coletivo, cada um dos clientes Web possui sua própria instância do

No lado servidor, componentes utilizam objetos que representam as principais entidades do serviço DECS: o marco fixo, o usuário, o evento de emergência e a mensagem de evento, conforme apresentado na Figura 5.18(b). Estas entidades herdam da classe *AbstractEntity* todas as propriedades e atributos, que inclui um mapa (tabela *hash*) do tipo *HashMap* onde o sistema armazena propriedades e valores, tais como coordenadas (e.g., latitude e longitude), identificador, dentre outros. Neste protótipo, o sistema representa as UnMs por meio de objetos do tipo *UbiContextUser*, responsáveis por gerenciar os atributos de entidades ubíquas móveis. Todas as entidades que fazem uso do servidor de eventos herdam da classe *EventServiceUser*, cuja classe é responsável por armazenar informações de eventos tais como as assinaturas por tópicos realizadas pelos usuários.

A partir dos dados estatísticos gerados e visualizados na Figura 5.19, é possível analisarmos a execução de cada um dos clientes Web representados. Podemos avaliar diversos aspectos, tais como o tempo de duração do atendimento em função da distância e também a velocidade em cada um dos clientes, já que estes fatores são considerados de grande importância para o melhor atendimento às vítimas em acidentes, distúrbios da ordem e/ou incêndios. Além dos dados relativos à distância e duração, a quantidade de dados transmitidos em *bytes* entre o cliente e o servidor nos informa o custo operacional da aplicação em termos de transmissão e o consumo de energia. Esta informação é de extrema importância, já que a transmissão é considerada o maior consumidor de energia em um dispositivo móvel que utiliza a transmissão de dados em redes sem fio. No protótipo DECS, conforme esperado, o número de *bytes* recebidos pelo cliente do servidor foi bem maior que o número de *bytes* enviados. A razão disto está no fato do cliente Web necessitar enviar apenas parâmetros de requisição para o servidor. No caso do servidor, ele envia mensagens em formato XML sobre a resposta HTTP para o cliente, sem a utilização de mecanismos de compressão de dados. Para futuras versões deste protótipo, desejamos utilizar este mecanismo a fim de diminuir a quantidade de dados sendo transmitidos, principalmente na recepção e processamento pelo cliente móvel.



(a) Classes de Gerenciamento



(b) Classes de Entidade

Figura 5.18. As principais classes no lado do servidor no serviço DECS.

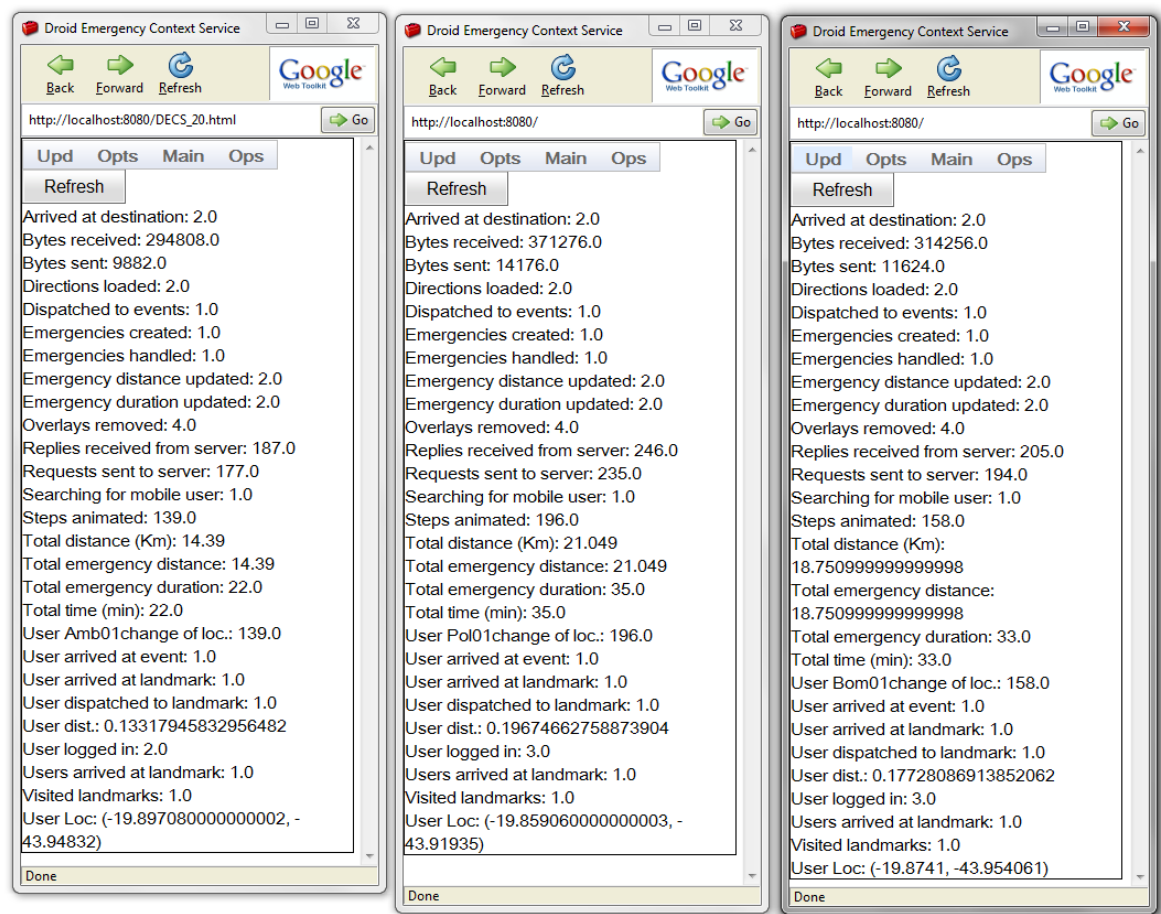


Figura 5.19. Visualização dos dados estatísticos de cada um dos clientes no modo coletivo.

5.5 Comparativo entre Protótipos 1.0 e 2.0

Apesar de ambos os protótipos desenvolvidos para o serviço de contexto para emergências utilizarem tecnologias similares (e.g., AJAX, GMaps API), podemos destacar algumas diferenças. Boa parte destas diferenças está ligada na facilidade de programação das funcionalidades necessárias para o funcionamento do sistema e na compatibilidade entre tipos diferentes de navegadores Web para computadores desktop e para dispositivos portáteis. No caso do protótipo 1.0, apesar das dificuldades apresentadas, não podemos desconsiderar sua notável capacidade de adaptação quando executado em diferentes tipos de clientes Web (e.g., PCs e dispositivos móveis). Algumas razões contribuem para esta característica, tais como o fato de utilizarmos tecnologias Web mais elementares como o HTML, o *JavaScript* e mensagens no formato XML para a interface gráfica de usuário e acesso a serviços Web de localização e navegação.

Componente	DECS
<i>Processador de eventos no servidor</i>	Processamento de eventos de clientes Web e de serviços relacionados à gestão de emergências
<i>Container de eventos</i>	Armazenamento de eventos de emergência
<i>Gestor de subscrições</i>	Assinatura por eventos relacionados ao atendimento de emergências
<i>Servidor de gerenciamento de eventos</i>	Coordena os componentes acima no serviço DECS
<i>Container de serviços Web</i>	Container de serviços Web relacionados à emergências
<i>Container de aplicações</i>	Container da aplicação de emergências
<i>Processador de eventos no cliente</i>	Processa eventos no cliente Web
<i>Gestor de perfil e contexto</i>	Gerencia informações referentes as entidades e emergências
<i>Aplicações e serviços</i>	Aplicação e serviços desenvolvidos no navegador Web
<i>Sensores no ambiente</i>	Sensores presentes na aplicação Web cliente

Tabela 5.2. Componentes da arquitetura do servidor de eventos utilizada no protótipo DECS.

5.6 Aplicação da Arquitetura Proposta

Esta seção tem como objetivo apresentar a relação do protótipo DECS com a arquitetura do servidor de eventos proposta. Esta seção também discute como a arquitetura proposta foi utilizada no estudo de caso desenvolvido neste capítulo. Conforme apresentado na Figura 3.14, a arquitetura do sistema proposto neste trabalho é composta por diversos componentes e módulos. Cada um destes componentes possui responsabilidades definidas para o protótipo DECS desenvolvido neste trabalho. Utilizamos a arquitetura definida neste trabalho para o desenvolvimento do servidor de eventos utilizado no DECS. Um mapeamento entre os componentes definidos na arquitetura do servidor de eventos e dos protótipos desenvolvidos neste trabalho pode ser visualizado na Tabela 5.2.

Conforme apresentado na Figura 5.20, o DECS possui dos componentes presentes na arquitetura proposta no servidor de eventos. No lado servidor, o gerenciador de assinatura em eventos é responsável por gerenciar assinaturas de tipos de eventos por usuários móveis e fixos existentes no sistema. O mecanismo de envio de mensagens desenvolvido no DECS utiliza este gerenciador a fim de selecionar os remetentes que receberão as mensagens publicadas por clientes móveis e também por marcos fixos. Tópicos de mensagens também podem ser criados a fim de possibilitar o envio de mensagens para grupos específicos de usuários, tais como hospitais, ambulâncias, viaturas policiais, dentre outros. Da mesma forma, o processador e *container* de eventos efetua operações de processamento e armazenamento dos eventos criados no sistema, respectivamente. No tratamento de eventos de emergência pelo DECS, o serviço de emergências localizado no servidor recebe informações referentes aos eventos criados e processados

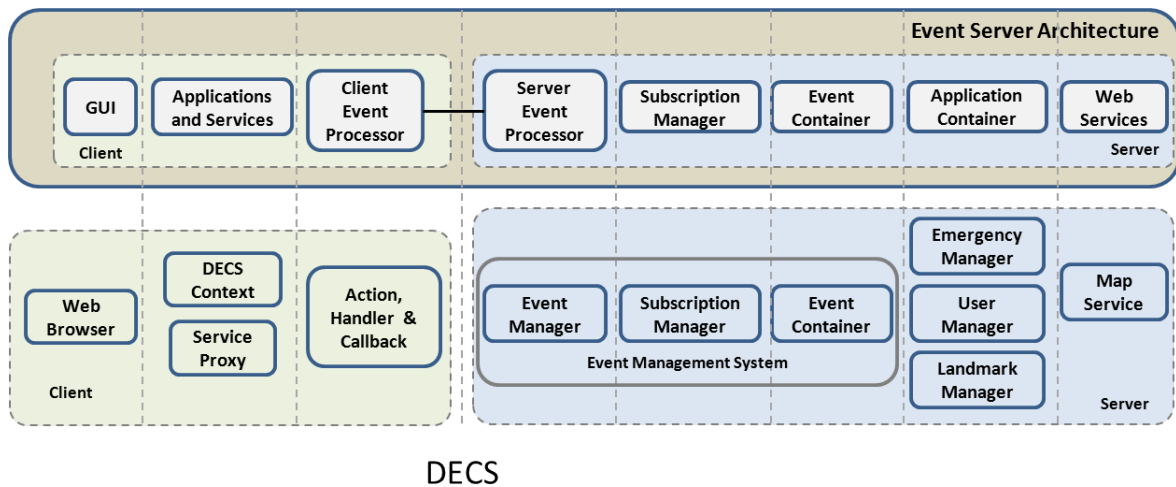


Figura 5.20. Mapeamento dos componentes no protótipo DECS em função da arquitetura proposta.

pele processador de eventos a fim de definir e realizar as próximas operações referentes à emergência em si.

Os *containers* de serviços Web e de aplicações foram parcialmente utilizados no DECS, devido a algumas características específicas do serviço em si. No atendimento à emergências, apesar de não termos implementado nenhum serviço Web, utilizamos o serviço Web de mapas para representar os marcos fixos e móveis da aplicação em questão, além de obter informações de rotas, distância percorrida e tempo médio gasto entre trajetos. Além deste, poderíamos utilizar outros serviços, tais como o de informações de tráfego, de alocação de recursos no marco fixo, como por exemplo, a alocação de leitos em hospitais em função da situação do paciente ou vítima (i.e., contexto lógico). No caso do armazenamento da aplicação servidora, o serviço DECS gerenciado pelo *container* de aplicações localizado no servidor remoto é responsável por todo o processamento de emergências, alocação de unidades móveis de emergência e de marcos fixos.

Devido a características específicas do tipo de cliente utilizado no DECS, alguns componentes foram adaptados a fim de adequar aos requisitos e limitações da interface Web. O cliente Web possui um processamento de eventos similar ao usado pelo *DroidGuide*, como por exemplo, a possibilidade de publicar eventos em função de uma localização no mapa. Desenvolvemos o gerenciador de informações de perfil e contexto a fim de gerenciar informações referentes aos eventos de emergência e os marcos móveis e fixos existentes na aplicação Web. Algumas informações gerenciadas

incluem a localização e a rota percorrida por unidades móveis, o tempo gasto, distância percorrida, informações de eventos de emergência e informações estatísticas (i.e., dados transmitidos entre cliente e servidor). Representada pelo cliente Web, a aplicação móvel oferece serviços oferecidos pelo servidor de dados tais como o monitoramento de unidades móveis de emergência, publicação de eventos, recepção de notificações a partir de eventos, assinatura de eventos através de tópicos e apresentação de informações de perfil e contexto referentes ao usuários tais como a rota, localização, distância e tempo de trajeto até o destino definido pelo serviço DECS.

5.7 Resultados obtidos

A fim de concentrarmos todos os resultados obtidos neste trabalho em um único capítulo, os resultados coletados a partir do protótipo DECS podem ser visualizados no capítulo 6. Nestes resultados, utilizamos dois modos de execução (manual e automático), a geração de eventos de emergência de forma pseudo-aleatória e a geração de uma amostra de eventos de emergência a fim de obtermos informações referentes ao comportamento do serviço de contexto para emergências proposto.

Capítulo 6

Resultados Obtidos

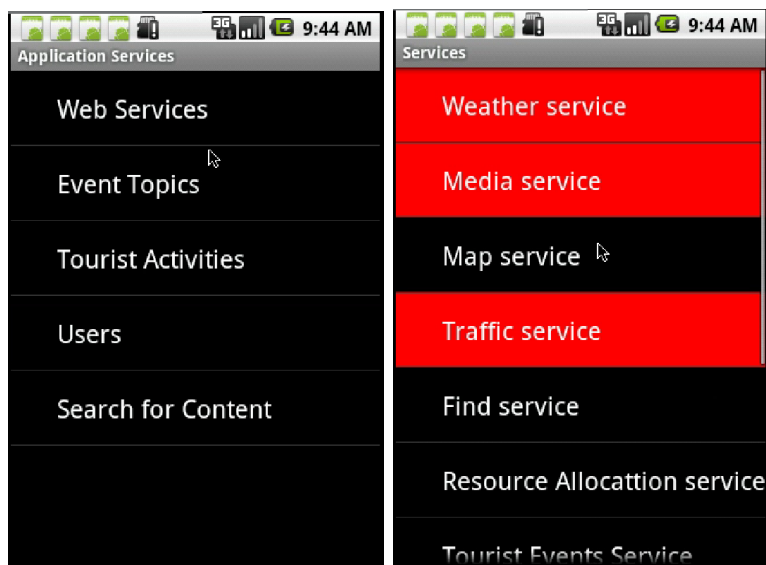
Este capítulo tem como objetivo apresentar os resultados técnicos obtidos a partir dos protótipos especificados e desenvolvidos a fim de validar a especificação e implementação do servidor de eventos para uso em aplicações móveis. Resultados de desempenho das aplicações em si não foram coletados, já que o objetivo principal dos protótipos foi validar o uso do servidor de eventos proposto neste trabalho para uso em aplicações e serviços móveis.

6.1 Guia Turístico DroidGuide

Esta seção tem como objetivo apresentar os resultados obtidos na execução do protótipo *DroidGuide*. Neste protótipo, realizamos três testes: (a) assinatura de serviços disponíveis no servidor pelo cliente móvel, (b) notificação de eventos no cliente móvel ocorridos a partir do servidor e (c) a seleção de atividades turísticas em função das informações definidas no perfil (i.e., interesses) do usuário móvel.

6.1.1 Assinatura de Serviços Web

Para a assinatura de serviços Web pelo usuário, criamos algumas entradas representando o acesso a estes serviços. Conforme apresentado na Figura 6.1, o usuário móvel assinou por três serviços: (a) uma que fornece dados climáticos da sua região, (b) outra que fornece conteúdo (imagens, vídeo, foto, som) referente a sua localização e (c) a que fornece dados referentes ao tráfego. O objetivo dos testes foi possibilitar ao usuário a assinatura por serviços disponíveis no servidor remoto de dados. Após a assinatura, estes serviços enviam para o usuário notificações referentes à eventos ocorridos que se relacionam com as informações de perfil e contexto definido na aplicação móvel.



(a) Menu principal de serviços.

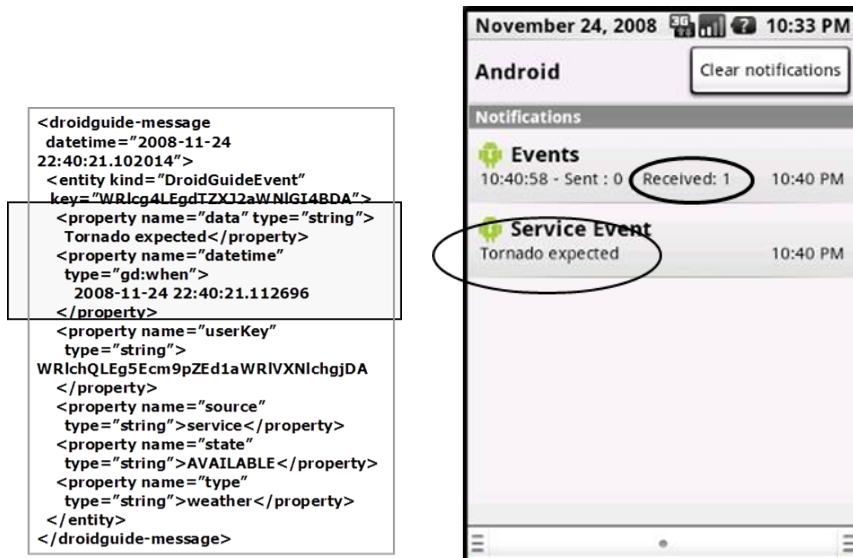
(b) Assinatura de serviços remotos.

Figura 6.1. Acesso e subscrição de serviços remotos pela aplicação móvel.

6.1.2 Notificação de Eventos

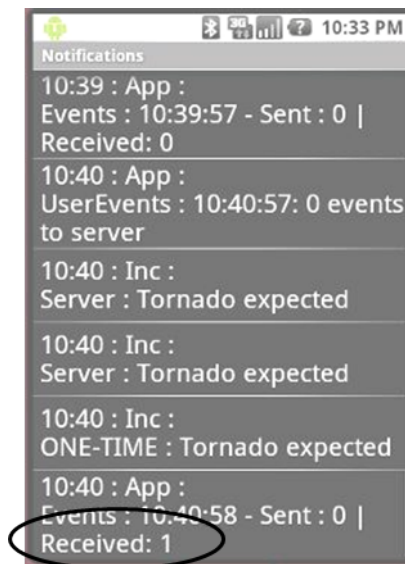
Nos testes de notificação de eventos no Guia Turístico *DroidGuide*, efetuamos a criação manual de um evento climático a fim de possibilitar a geração de uma notificação pelo servidor de eventos. Neste teste, o serviço remoto responsável pelo clima cria um evento climático no lado do servidor, já que não utilizamos sensores reais de condições climáticas para a criação de eventos. Através de uma requisição HTTP ao serviço remoto, criamos o evento relacionado ao clima, armazenado-o no *container* de eventos presente no servidor de eventos. Após a criação deste evento, o servidor remoto de dados envia uma resposta em formato XML para a aplicação móvel, conforme apresentado na Figura 6.2(a). A partir do evento criado, o servidor de eventos gera mensagens de notificação enviadas para clientes móveis subscritos no tópico relacionado ao evento.

Conforme descrito na seção 4.4.1, a aplicação móvel requisita de forma periódica por eventos detectados e coletados tanto no cliente quanto no servidor. Neste processo, a aplicação móvel recebe os eventos gerados no servidor em função de mudanças detectadas tanto no dispositivo móvel quanto em serviços remotos, no nosso caso, o serviço climático. Na composição da resposta ao cliente, o serviço adiciona a mensagem referente ao evento gerado e a envia para a aplicação móvel. Ao receber as mensagens de notificação do servidor, a aplicação as apresenta em duas formas: na tela e na aba de notificações, conforme apresentado na Figura 6.1.2.



(a) Criação do evento climático no servidor

(b) Aba de notificações do dispositivo móvel



(c) Tela de notificações

Figura 6.2. Notificação de eventos remotos detectados no lado do servidor.

Através da utilização do servidor de eventos proposto e desenvolvido neste trabalho, o dispositivo móvel foi capaz de submeter às mudanças de dados de perfil e contexto para o servidor em forma de requisições sobre o protocolo HTTP, desta forma compartilhando seu atual estado ou condição durante a execução do guia turístico para os serviços Web subscritos pelo turista. Estes serviços Web puderam, quando necessário, enviar notificações de eventos ocorridos em função de mudanças de dados de perfil e contexto locais recebidos pelo dispositivo e também remotos, tais como estados ou condições climáticas e de tráfego nas proximidades do local onde o usuário móvel se localiza ou no seu destino pré-determinado. Através das requisições HTTP vindas de clientes móveis, o servidor de eventos utilizou as respostas sobre o HTTP para o envio de notificações a partir de eventos coletados no lado do servidor para o cliente, sendo estas apresentadas ao usuário por meio de mensagens de notificação da aplicação móvel. Além dos serviços disponibilizados na aplicação móvel em execução pelo turista, alguns recursos foram desenvolvidos no lado do servidor com o objetivo de gerenciar os dados relacionados aos turistas e suas atividades. Algumas das informações acessíveis a partir do servidor de dados incluem:

- Listagem de usuários logados no servidor: informações do estado ou condição do perfil e contexto de usuários em um dado momento;
- Gerenciamento dos serviços Web disponíveis no servidor de dados: listagem, criação, edição e desativação. Um exemplo de listagem pode ser visualizado na Listagem 6.3.
- Listagem dos serviços subscritos por usuários móveis: estatísticas de subscrição de serviços por usuários;
- Gerenciamento de eventos de serviço: criação, alteração e exclusão de eventos para testes no envio de notificações para o usuário móvel;
- Listagem dos eventos relacionados com o usuário: quantidade de eventos gerados por usuário móvel, tipos de eventos gerados, dentre outros.

6.1.3 Seleção de Atividades Turísticas em Função do Perfil do usuário

A seleção de atividades turísticas descreve uma funcionalidade no guia turístico em propor atividades ao usuário móvel em função de seus interesses e desejos em conhecer um determinado local. A seleção de atividades foi desenvolvida neste trabalho em forma de


```
<droidguide-message datetime="2010-01-06 17:51:33.114789">
  <entity kind="Service" key="agpkcm9pZGd1aWR1cg4LEgdTZXJ2aWN1GKADDA">
    <key> tag:droidguide.gmail.com,2010-01-06:Service[agpkcm9pZGd1aWR1cg4LEgdTZXJ2aWN1GKADDA]
  </key>
  <property name="data" type="string">25 Degrees</property>
  <property name="description" type="string">Weather data Pampulha</property>
  <property name="state" type="string">AVAILABLE</property>
  <property name="type" type="string">weather</property>
</entity>

  <entity kind="Service" key="agpkcm9pZGd1aWR1cg4LEgdTZXJ2aWN1GN8EDA">
    <key>
      tag:droidguide.gmail.com,2010-01-06:Service[agpkcm9pZGd1aWR1cg4LEgdTZXJ2aWN1GN8EDA]
    </key>
    <property name="data" type="string">18 Degrees</property>
    <property name="description" type="string">Weather data Ouro Preto</property>
    <property name="state" type="string">AVAILABLE</property>
    <property name="type" type="string">weather</property>
  </entity>

  <entity kind="Service" key="agpkcm9pZGd1aWR1cg4LEgdTZXJ2aWN1GOAEDA">
    <key>
      tag:droidguide.gmail.com,2010-01-06:Service[agpkcm9pZGd1aWR1cg4LEgdTZXJ2aWN1GOAEDA]
    </key>
    <property name="data" type="string">Slow</property>
    <property name="description" type="string">Traffic data Pampulha</property>
    <property name="state" type="string">AVAILABLE</property>
    <property name="type" type="string">traffic</property>
  </entity>
</droidguide-message>
```

Figura 6.3. Uma resposta do servidor remoto de dados contendo a listagem dos serviços disponíveis (SWBIs) no servidor remoto de dados para subscrita.

componente compondo os serviços de turismo disponíveis no servidor remoto de dados em nuvem. Outros serviços de turismo existentes no sistema incluem o acesso a mapas, assinatura por elementos (tópicos, atividades e serviços remotos), contexto lógico (estados do usuário móvel) e perfil (interesses e informações referentes ao usuário).

Durante uma execução de alguns testes iniciais da seleção de atividades, notamos uma dificuldade em selecionar atividades turísticas para o usuário. Isto ocorre devido ao fato de utilizarmos uma quantidade pequena de atividades turísticas no sistema e estas atividades possuíam uma pouca diversidade de interesses. Desta forma, com uma quantidade pequena de atividades em determinados interesses, a seleção destas para o usuário tornou-se mais difícil. Um exemplo de classificação de perfis de atividades turísticas e usuário pode ser visualizada na Figura 6.1. O algoritmo apresentado na Figura 4.13 foi consideravelmente rígido na seleção de atividades (e.g., checagem AND), retornando quase que nenhuma atividade para o usuário em boa parte dos testes realizados. Optamos então em flexibilizar a seleção de atividades de tal forma que, ao encontrar pelo menos um interesse em comum e dentro do limite (e.g., checagem OR),

Tabela 6.1. Um Exemplo de classificação de perfis para usuários e atividades turísticas.

Entity	Bohemian	Consumer	Cultural	Ecological	Gastronomic	Historical
Tourist01	2	4	4	7	9	7
St. Francis Church	0	1	6	4	0	8
City Zoo	0	0	5	8	0	4
Central Market	0	0	3	5	9	4
Night Club	9	8	1	5	0	0

Tabela 6.2. Classificação do perfil de turistas usado na emulação.

Interest	Tour01	Tour02	Tour03	Tour04	Tour05
Bohemian	0	9	10	0	0
Consumer	4	9	8	2	4
Cultural	4	2	6	4	8
Ecological	7	0	4	6	8
Gastronomic	9	9	2	8	4
Historical	7	0	0	10	0

a atividade seria selecionada. Com o aumento do limite mínimo (*threshold*), o sistema retorna um número maior de atividades turísticas, porém com um grau maior de erro em relação ao perfil do usuário móvel, possivelmente diminuindo assim a satisfação do turista referente as atividades sugeridas.

Para avaliarmos o algoritmo de seleção de atividades de uma forma automática, optamos por utilizar uma infra-estrutura de aplicação Web similar ao protótipo DECS. Com pequenas modificações no protótipo DECS, fomos capazes de executar uma emulação do algoritmo de seleção de atividades turísticas sobre a interface de mapas disponível na aplicação Web. Nesta emulação, o serviço de turismo seleciona atividades turísticas em função do perfil dos usuários definidos conforme a tabela 6.2 e realiza a animação da visita a cada uma das atrações turísticas selecionadas. Para cada um dos turistas criados na emulação, o emulador coletou informações referentes a seleção de atividades e a visita destas atividades pelo usuário de forma sequencial. A Tabela 6.5 apresenta os resultados coletados durante a execução da emulação.

A partir das atividades turísticas definidas em uma cidade, executamos a seleção de atividades turísticas para turistas cujos perfis podem ser visualizados na tabela 6.2. Os resultados da execução podem ser visualizados na Figura 6.5. Em função das atrações disponíveis no mapa, os turistas 01 e 04 consumiram a maior quantidade de atividades turísticas. A partir das rotas definidas entre atividades, calculamos a distância percorrida, o tempo total gasto entre trajetos sem considerar o tempo gasto pelo turista na atividade em si e a velocidade média do turista. A emulação também coletou a quantidade de dados transmitidos para cada um dos turistas em função de *bytes* transmitidos entre o cliente Web e o servidor. Estes dados transmitidos entre o cliente e o servidor incluem a publicação de informações referentes a localização do turista durante o trajeto e sua chegada/saída da atração turística. O objetivo da coleta

foi avaliar o consumo de dados entre clientes Web e o servidor remoto de dados e não realizar uma avaliação de desempenho da aplicação cliente, do servidor de dados e do serviço de contexto de emergências.

A Figura 6.1.3 apresenta a execução da emulação utilizando o mesmo arcabouço desenvolvido para o protótipo DECS. Ressaltamos que os dados apresentados na Figura 6.1.3 servem apenas para avaliar a emulação do serviço de seleção e visita a atividades turísticas pelo usuário móvel. A interface gráfica e os dados apresentados nas Figuras 6.4(c) e 6.4(d) não estão formatados e apresentados de forma amigável para serem apresentados por uma aplicação móvel ao usuário final (turista). Futuramente, desejamos trabalhar na formatação das informações referentes a eventos a serem apresentadas ao usuário móvel final, a fim de tornar a interface gráfica mais amigável.

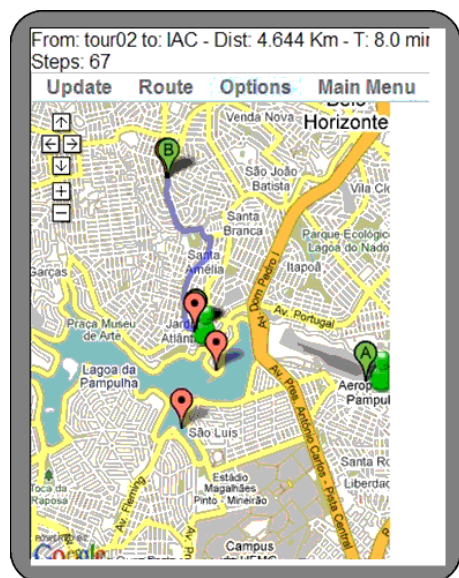
No modo manual de execução da ferramenta, conseguimos também executar um disparo de um evento relacionado ao clima e visualizar a publicação e consumo do mesmo pelo turista, conforme apresentado nas Figuras 6.4(b), 6.4(c) e 6.4(d), respectivamente.

A Figura 6.5 apresenta os resultados referentes à simulação da seleção de atrações turísticas em função do perfil de turistas. Neste resultado, apresentamos o número total de atrações selecionadas e visitadas, a quantidade de dados transferida entre o cliente Web e o servidor de dados, a razão entre os dados recebidos e enviados pelo cliente Web, o número total de passos ou saltos efetuados pelo turista no mapa em função das rotas apresentadas, a distância total percorrida, o tempo total gasto em trajetos entre atrações, a velocidade média de locomoção e o número de passos efetuados pelos turistas por atividade.

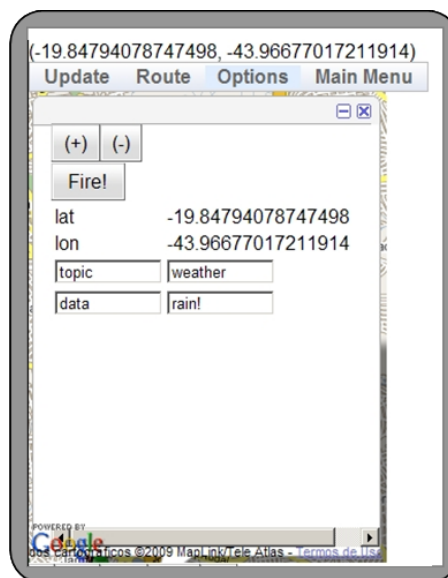
Referente ao número total de atrações visitadas, os turistas 01 e 04 obtiveram um melhor êxito visitando um total de 7 atrações. O turista 03 obteve uma quantidade menor de atrações, devido baixa compatibilidade de seu perfil com as atrações disponíveis na cidade. Conforme esperado, a quantidade de dados trafegada entre o cliente e o servidor é proporcional a quantidade de atrações visitadas, já que uma boa parte das informações compartilhadas se refere a posição dos turistas em trajetórias definidas pelo serviço de mapas. Da mesma forma, o número de saltos realizados foi maior para os turistas 01 e 04.

6.2 Serviço de Contexto para Emergências DECS

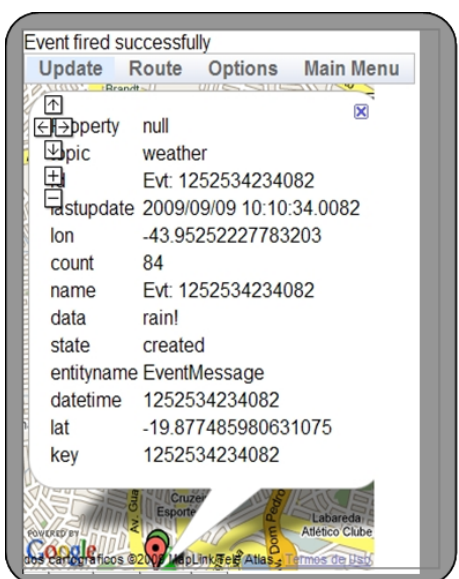
Conforme apresentado na seção 5.4.6, o protótipo DECS utiliza um mecanismo de coleta de informações estatísticas durante a execução dos serviços nele disponibilizados. O objetivo deste mecanismo é coletar dados relevantes durante a execução da aplicação,



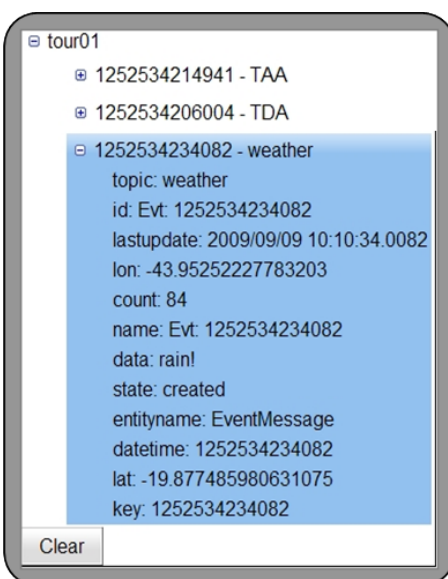
(a) Mapa do turista



(b) Disparo de um evento



(c) Visualização dos detalhes do evento



(d) Consumo do evento pelo turista

Figura 6.4. Execução da emulação da seleção de atividades turísticas em função do perfil do usuário móvel.

	Activities visited	KBytes sent	KBytes received	Sent/Received Ratio	Total Number of Steps	Distance Traveled (km)	Duration of Travel (hours)	Speed (km/h)	Steps/Activity Ratio
Tour01	7	1,02	972,99	0,10%	511	54	1,75	31,09086	73
Tour02	5	0,78	558,80	0,14%	288	33,168	1,05	31,58857	57,6
Tour03	1	0,21	103,38	0,20%	44	4	0,13	30	44
Tour04	7	1,00	1162,89	0,09%	618	65,143	2,02	32,30231	88,286
Tour05	4	0,57	696,78	0,08%	364	30,53	0,92	33,30545	91
Average	4,8	0,71	698,97	0,12%	365	37,45	1,17	31,65744	70,777

Figura 6.5. Resultado da execução da emulação do algoritmo 4.13.

tais como o número de requisições e respostas efetuadas, quantidade de dados em *bytes* enviados e recebidos, distância percorrida pelo marco móvel e a duração do atendimento do evento de emergência, conforme apresentado na Figura 5.4. Durante cada ciclo de execução do fluxo de atendimento a emergências, o serviço coleta as informações e as armazena em um container de dados estatísticos localizado em cada aplicação cliente em execução. O sistema gera dados estatísticos das seguintes formas: (a) individual para cada cliente e (b) coletiva, incorporando todos os clientes Web em execução.

6.2.1 Cenários de Execução

Utilizando os modos manual e automático de execução do serviço DECS, coletamos diversos dados estatísticos com o objetivo de avaliar o comportamento dos serviços desenvolvidos e utilizados pelo protótipo, em especial o servidor de eventos e o servidor de dados. Um resumo do total de dados coletados pode ser visualizado na figura 6.3. Executamos o protótipo em cenários baseados nas seguintes variáveis:

- **Modo de execução:** Manual, onde o usuário define a posição de cada emergência e automático, onde as posições são geradas de forma pseudo-aleatória. O que nos mais interessa neste trabalho é o modo automático de execução, já que poderemos gerar eventos de emergência de forma aleatória. O modo manual de execução foi utilizado para fins de demonstração do serviço sendo acionado por um usuário utilizando um dispositivo móvel.
- **Número de eventos de emergência gerados:** A quantidade total de emergências atendidas no cenário, variando de 1 a 500 eventos.

Tabela 6.3. Resumo das informações coletadas a partir do protótipo DECS.

Variável	Descrição
Quantidade de unidades móveis disponíveis	9 (3 de cada um dos tipos)
Quantidade de marcos fixos disponíveis	10 (hospitais, delegacias e corpo de bombeiros)
Tamanho da amostra	660 eventos de emergência gerados de forma pseudo-aleatória
Distância total percorrida por clientes móveis	22,990 Km
Tempo total gasto por clientes móveis em rotas	35,336 minutos
Número de movimentações no mapa	162,713
Dados transmitidos	513 MB (do cliente para o servidor), 24 MB (do servidor para o cliente)
Número de requisições/respostas submetidas	340,525 requisições/respostas
Total de eventos disparados	324,731
Total de eventos armazenados em tabelas <i>hash</i>	2,86 GB

- **Tamanho da amostra:** Coletamos 660 eventos de emergência subdivididos em cada um dos três tipos possíveis, com o objetivo de avaliarmos o comportamento e a distribuição das variáveis coletadas.

6.2.2 Resultados Obtidos

Antes do início da execução da aplicação, é necessário a definição dos parâmetros de coleta e análise, listados na Tabela 6.4. Após a definição dos parâmetros, iniciamos a execução dos cenários de tratamento de eventos de emergência a fim de obtermos dados estatísticos de execução do serviço DECS. A partir dos dados coletados, foi possível efetuarmos uma avaliação quantitativa do servidor de eventos de outros serviços ou recursos tais como o gerenciador de usuários e de emergências. Algumas informações coletadas incluem a quantidade de dados trafegados entre o cliente e o servidor, distância percorrida e tempo gasto médio no atendimento à emergências. Os resultados coletados estão divididos em três partes: (a) servidor de eventos, (b) lado cliente Web e (c) lado servidor remoto de dados.

6.2.3 Servidor de Eventos

No que diz respeito ao servidor de eventos, os valores coletados variaram de forma linear em função do aumento do número de eventos de emergência atendidos. O número de eventos disparados, a quantidade de mudanças de posições das UnMs e a quantidade de acessos ao *Container* de Eventos cresceram de forma linear, conforme demonstrado na Figura 6.6. Devido à alta variabilidade na geração de coordenadas para o tratamento de eventos de emergência, o desvio padrão para o número de eventos disparados nas simulações foi consideravelmente alto (45,6 por cento da média). Enquanto a instância que obteve o menor número de eventos disparados foi 47, a instância que obteve o maior

Parâmetro de configuração	Descrição
Intervalo de animação de marcos móveis	Define o intervalo de atualização da posição dos marcos móveis durante a animação da aplicação.
Frequência de atualização do centro do mapa	Número de passos necessários para a aplicação atualizar o centro do mapa no cliente Web (10 passos).
Coordenadas do centro do mapa (Lat, Lon)	Define as coordenadas iniciais do centro do mapa da aplicação Web (-43.963852, -19.88136).
Resolução da aplicação cliente (altura, largura)	Tamanho da tela em pixels (390px, 300px)
Modo de execução	Define o modo de execução da aplicação (Cliente/Simulação)
Atraso de carga de marcos fixos e móveis	Define o atraso da carga dos marcos móveis e fixos no mapa durante a atualização do mapa (3s).
Número total de emergências geradas automaticamente	Define o número total de emergências geradas automaticamente.
Nível de depuração	Define o nível de depuração a ser adotado na aplicação.
Intervalo entre eventos de emergência	Tempo de intervalo entre a geração de emergências.

Tabela 6.4. Parâmetros de configuração do DECS.

número foi de 1227. No histograma gerado na Figura 6.7, observamos que, com a alta variabilidade na geração de coordenadas no sistema, os valores foram mais distribuídos pela curva do que o normal, resultando em um histograma com frequências maiores.

6.2.3.1 Cliente Web

Durante a execução do serviço DECS, coletamos informações no lado do cliente Web com o objetivo de avaliarmos a quantidade de dados transmitidos entre o cliente e o servidor remoto de dados. No nível de transmissão de dados, temos os seguintes valores: (a) o número de *bytes* transmitidos do cliente para o servidor (i.e., durante requisições) e (b) do servidor para o cliente (i.e., respostas). Conforme esperado, os dois valores obtiveram um comportamento similar em função das instâncias executadas do serviço de atendimento de emergências, conforme apresentado na Figura 6.8. Comparando as duas variáveis, podemos observar que o número de *bytes* recebidos do servidor é bem maior do que o número de *bytes* enviados para o servidor, conforme esperado. Isto demonstra a necessidade de transmitirmos uma quantidade bem maior de dados do servidor para o cliente comparado com o sentido contrário. Neste caso, calculamos a média da razão entre *bytes* enviados/recebidos em torno de 3,5 a 4 por cento, considerando todas as 660 amostras coletadas. Uma sumarização deste e de outros resultados pode ser visualizada na Tabela 6.10.

A tabela 6.10 apresenta um resumo das principais variáveis coletadas durante a execução do serviço DECS. A partir dos resultados coletados, constatamos que para algumas variáveis, o valor do desvio padrão foi consideravelmente alto. Acreditamos

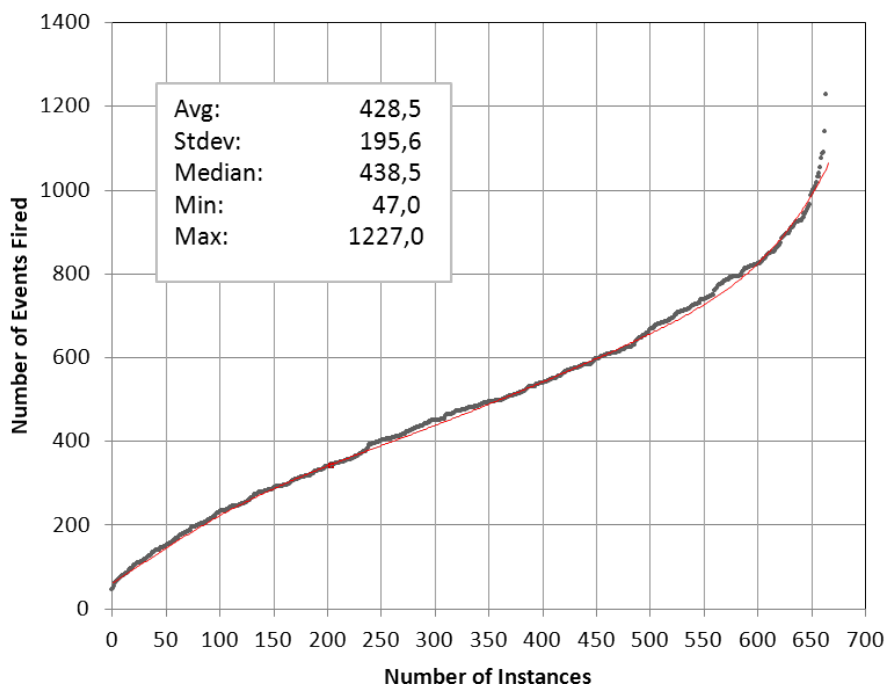


Figura 6.6. Comportamento da variável do número de eventos disparados.

que, no caso do serviço DECS, a variabilidade na geração de eventos de emergência causado pela variação nas distâncias e tempos percorridos entre eventos de emergência seja o principal motivo dos valores encontrados neste caso. Podemos citar, por exemplo, a variação da distância que, no menor caso foi de apenas 1.27 km, enquanto para o maior foi de 83,4 Km. Esta diferença corresponde a uma variação de 656,692.91 por cento. Isto implica que, para a modelagem, implementação e implantação do serviço de emergências no mundo real, este deverá estar apto a lidar com eventos de diferentes distâncias e durações, necessitando de, por exemplo, alocação de marcos fixos mais bem distribuídos na sua região de atuação. Um possível trabalho futuro para este serviço seria um estudo para avaliar a melhor distribuição de marcos fixos em função dos principais corredores de transporte existentes em uma região.

A partir da coleta dos dados realizada sobre o serviço DECS apresentada na Tabela 6.10, podemos destacar algumas observações na relação entre variáveis durante a execução do serviço. Podemos concluir que, a distância percorrida pelas unidades móveis é diretamente proporcional ao número de eventos disparados no sistema, já que boa parte destes eventos disparados se deve em função da mudança de posição das entidades. Quanto maior a distância, maior o número de eventos disparados no

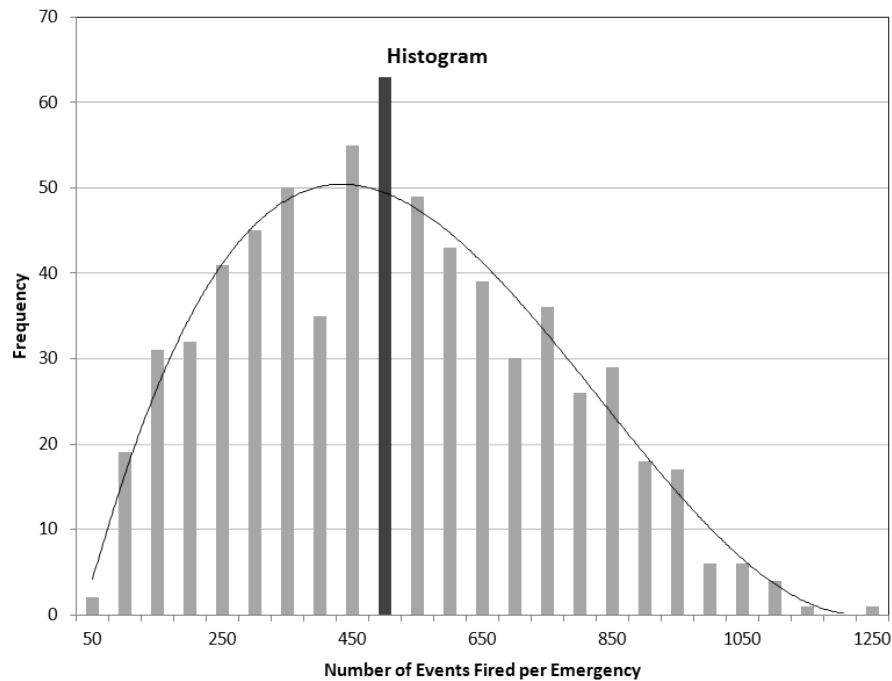


Figura 6.7. Histograma do número de eventos disparados por evento de emergência tratado no DECS.

sistema. O mesmo se aplica ao tamanho da fila de eventos presente no servidor de eventos em número de mensagens e na quantidade de *bytes* armazenados no *container* de eventos.

A partir dos dados coletados no lado do cliente, podemos destacar algumas das instâncias executadas pelo serviço, cujos detalhes podem ser visualizados na Figura 6.11. Estas instâncias apresentaram boa parte dos valores extremos coletados durante as execuções. Por exemplo, a instância 179 obteve os valores mínimos de algumas variáveis, em função da curta distância percorrida pela UnM. Por outro lado, a instância 292 obteve boa parte dos máximos em função da distância e também da complexidade das rotas definidas para a UnM. Os detalhes das instâncias 179 e 292 podem ser visualizados nas Figuras 6.12 e 6.13, respectivamente.

Na análise da quantidade de dados transmitidos em *bytes* entre o cliente e o servidor, observamos também que o número de *bytes* transmitidos cresce em função do aumento da distância percorrida. Isto se deve ao fato da necessidade de transmissão de um número maior de eventos relacionados à posição das unidades móveis. Isto causa um número maior de requisições por Km rodado, e por consequência um número maior de *bytes* transmitidos do cliente para o servidor e vice-versa. A complexidade da rota

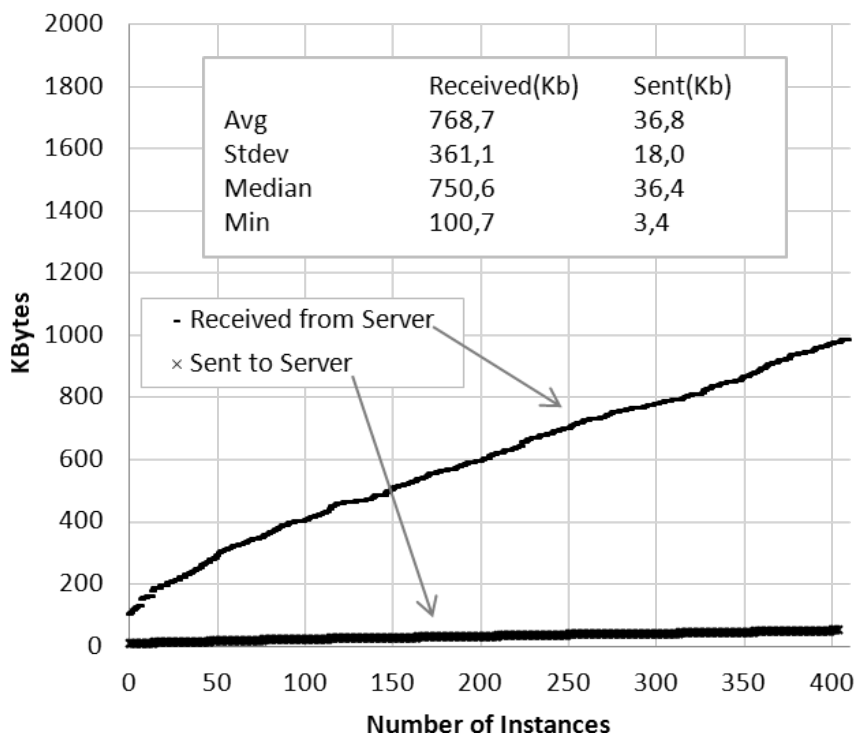


Figura 6.8. Comportamento da transmissão de dados no cliente.

também reflete no número de requisições enviadas para o servidor, sendo composta pelo número de passos ou saltos e a sua distância. Em rotas mais longas, as unidades móveis necessitam de um número maior de passos ou saltos para chegar ao seu destino. Nos resultados coletados, por exemplo, a instância 179 que já possui a menor distância percorrida obteve o menor número de requisições por emergência (53 requisições), enquanto a instância 292 obteve o maior número de requisições por emergência (1251 requisições) e também pela distância percorrida em Km (41,8 requisições/Km). Apesar da instância 292 não apresentar a maior distância percorrida entre os dados coletados (i.e., instância 456), ela é considerada mais complexa já que possui um número maior de passos ou saltos até os destinos alcançados. Cada instância é composta pela rota da origem até o evento de emergência e a rota do evento até ao marco fixo mais próximo.

A partir dos dados coletados, foi possível avaliarmos as relações entre as variáveis coletadas durante a execução do serviço DECS, obtendo assim uma amostra de 660 instâncias. A primeira delas é a quantidade de requisições solicitadas por evento de emergência em função da quantidade de *bytes* transmitidos pelo cliente Web. Como era de se esperar, quando maior a quantidade de requisições solicitadas para um dado

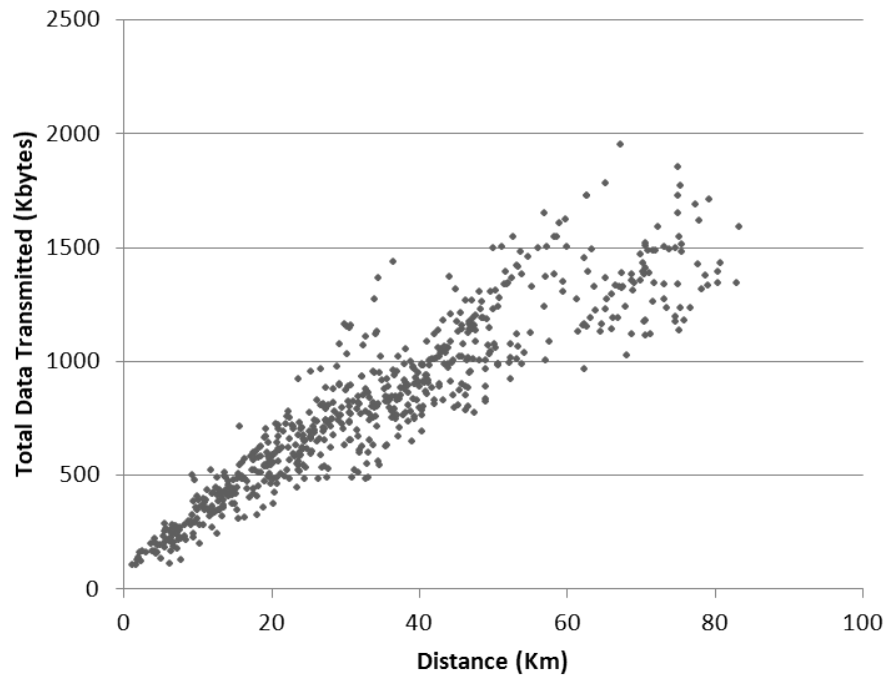


Figura 6.9. Transmissão de dados no cliente em função da distância percorrida.

evento de emergência, maior a quantidade de dados transmitidos entre o cliente Web e o servidor remoto de dados. Neste caso, o crescimento foi praticamente linear, conforme apresentado na Figura 6.15.

A segunda relação avaliada foi a distância percorrida em função do número de requisições solicitadas pelo cliente Web. Uma representação desta relação pode ser visualizada no segundo gráfico da Figura 6.14. Com o crescimento da distância percorrida, o número de requisições solicitadas pelo cliente Web também aumentou. Isto também era esperado, já que com uma distância maior a percorrer, a UnM necessitará de enviar um número maior de requisições contendo a atualização de sua posição na rota definida. Com o aumento da distância percorrida, a variabilidade no número de requisições também cresce, causando uma dispersão dos dados no formato de um cone.

Duas outras relações avaliadas a partir dos dados coletados foram o número de movimentações, a distância e a velocidade média das UnMs. Conforme esperado, com o aumento da distância percorrida, as UnMs necessitaram de um número maior de movimentos por rota, conforme apresentado na Figura 6.16. Diversos fatores podem contribuir para uma alta complexidade da rota percorrida, sendo os principais a distância e o número de movimentações necessárias para percorrê-la. Nos resultados

Component	Variable	Average	Standard deviation	Percentage deviation	Median	Minimum	Maximum
Event Container	Events published per emergency	492,25	237,37	48,22%	482,5	47 ^A	1227 ^B
	Event message size (Bytes)	369,9	3,53	0,95%	371	358	379
	Event queue size (messages)	493,94	238,20	48,22%	484	47	1224 ^B
	Hash size (KBytes)	180,36	89,79	49,79%	177,65	17,34 ^A	452,88 ^B
Client	Data sent (KBytes)	37,26	17,97	48,24%	36,54	3,43 ^A	90,21 ^B
	Data received (KBytes)	777,2	360,71	46,41%	760,75	100,76 ^A	1858,19 ^B
	Ratio sent/received	0,05	0,00	4,85%	0,05	0,03	0,05
	Total distance (Km)	34,9	19,63	56,25%	33,07	1,27 ^A	83,4 ^D
	Total time (mins)	53,61	28,03	52,29%	50	3 ^{*A}	185
	Speed (Km/h)	38,42	10,31	26,84%	37,65	14,92	71,17
	Requests per emergency	515,73	247,25	47,94%	503	53 ^A	1251 ^B
	Requests per km	16,18	4,21	26,01%	15,6	8,64	41,8 ^B
	Data sent/request (Bytes)	72,07	1,64	2,28%	72,21	64,52 ^{*B}	76,34
	Data received/req (Bytes)	1529,17	67,33	4,40%	1510,19	1440,78	1977,47 ^{*B}

A Instance 179

B Instance 292

C Instance 553

D Instance 456

Figura 6.10. Resumo dos resultados obtidos durante a execução do serviço DECS.

Instances	Distance (Km)	Time (mins)	Steps	Data Transmission	Requests	Events fired
179	1,2	3	28	100KB↓, 3,4 kB↑	53	47
292	67,3	106	1205	1800KB↓, 90,2KB↑	1251	1227
553	67,3	106	1037	1629,0KB↓, 79,1KB↑	1083	1038
456	83,4	125	987	1514,5KB↓, 75,6KB↑	1037	987

Figura 6.11. Detalhes relativos às instâncias destacadas nos resultados coletados.

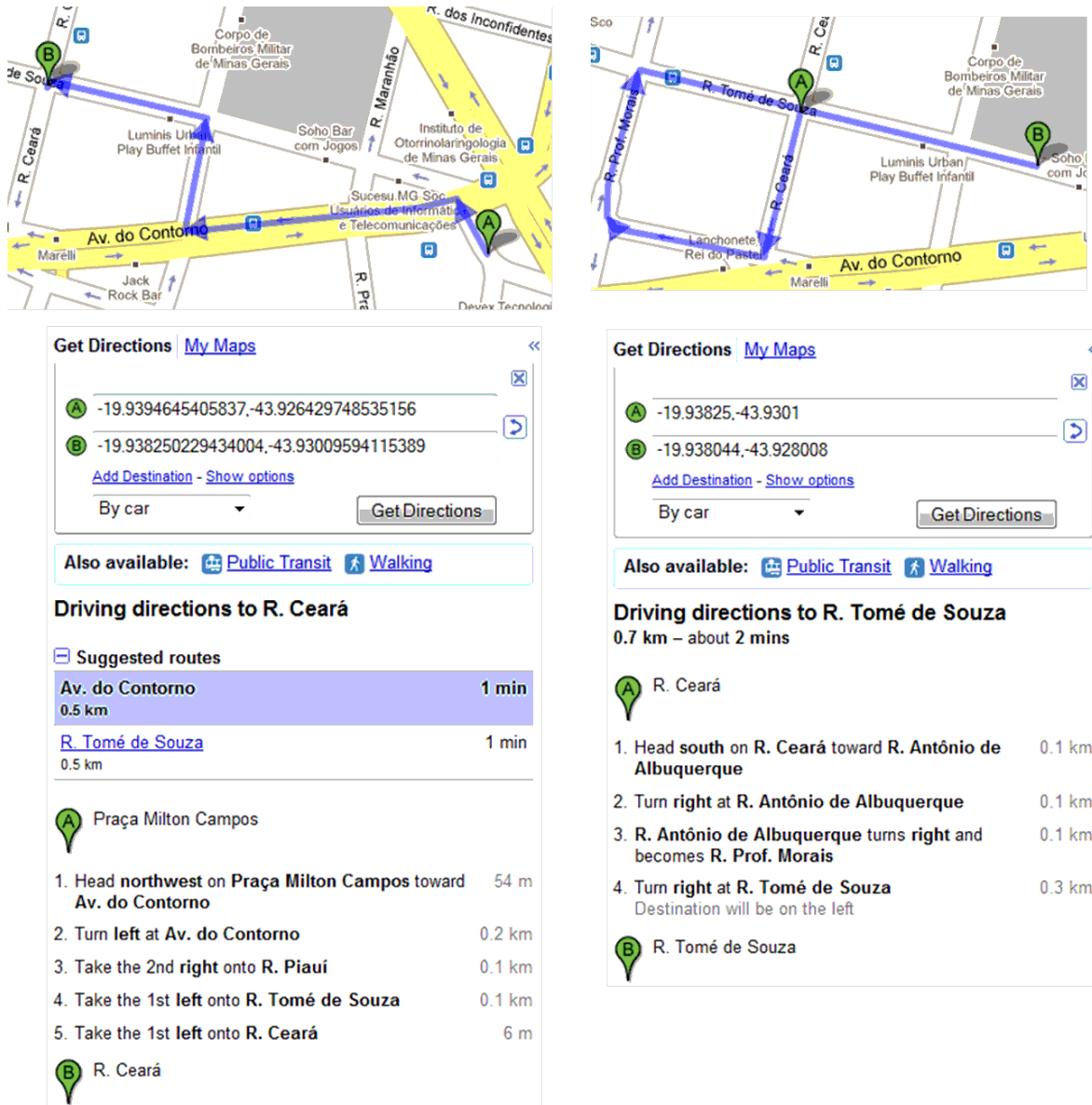


Figura 6.12. Informações das rotas executadas pela instancia 179.

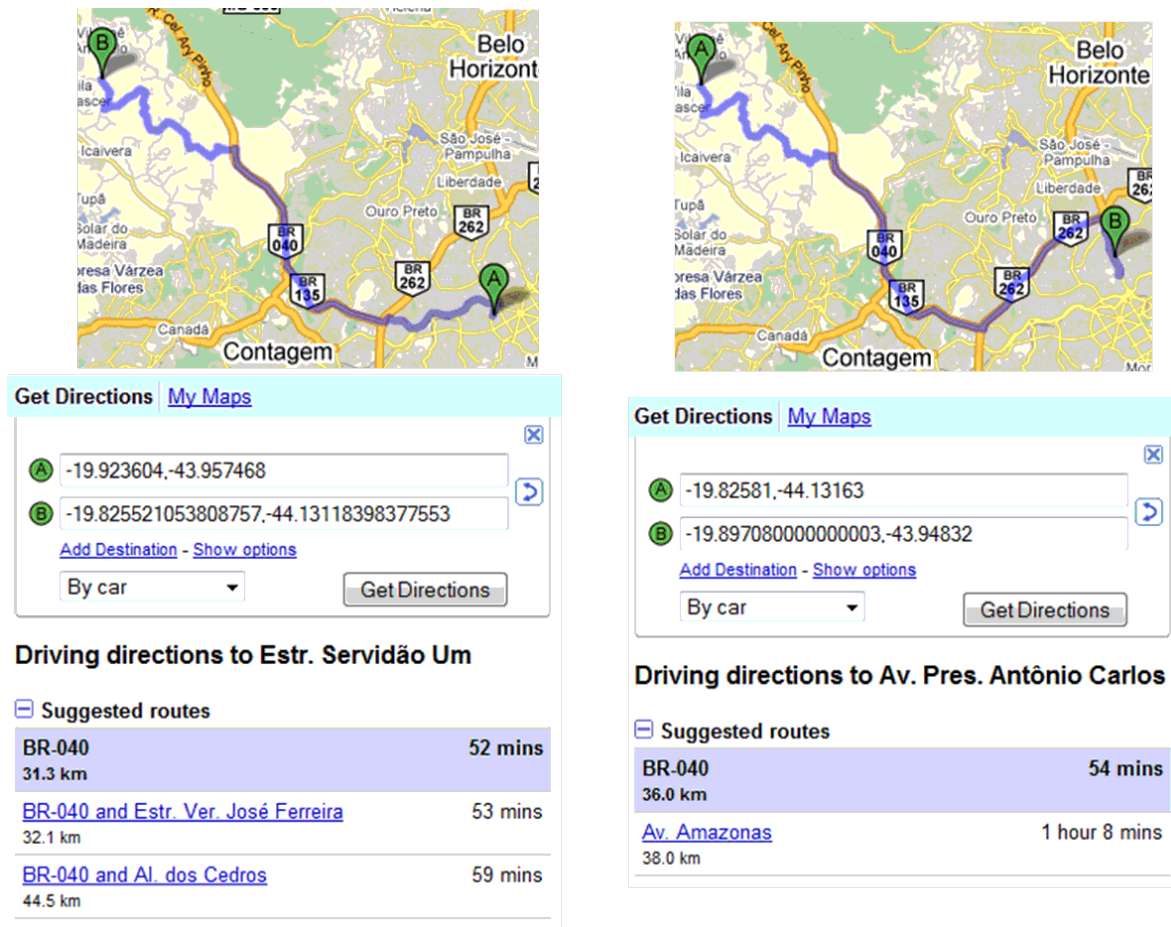


Figura 6.13. Informações das rotas executadas pela instancia 292.

coletados, obtivemos instâncias onde a distância era relativamente curta (+/- 30Km), porém com o mesmo número de movimentações que uma rota contendo o dobro da distância (+/- 60 Km). As curvas representando as distribuições da distância e da duração do atendimento podem ser visualizadas na Figura 6.17.

6.2.3.2 Servidor Remoto de Dados

A fim de identificarmos detalhes do consumo em número de *bytes* no sistema DECS, analisamos neste trabalho duas variáveis no lado do servidor para cada um dos componentes existentes: (a) o número de *bytes* enviados pelo cliente e (b) o número de *bytes* recebidos pelo cliente do servidor. O principal objetivo está na busca pelo componente que transmite a maior quantidade de dados para o cliente e que mais recebe dados do cliente. Os resultados da coleta podem ser visualizados na Figura 6.18.

Após a coleta dos dados, identificamos que no envio do cliente para o servidor

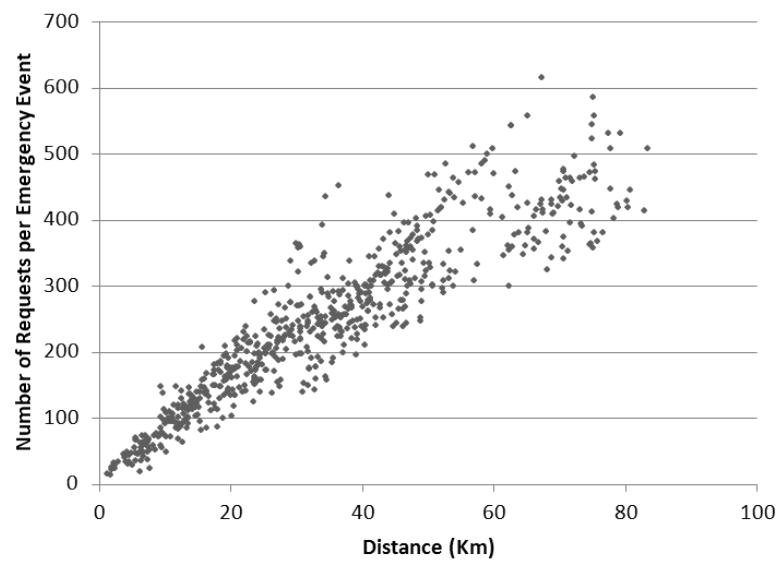


Figura 6.14. Gráfico apresentando a relação entre o número de requisições por evento e a distância.

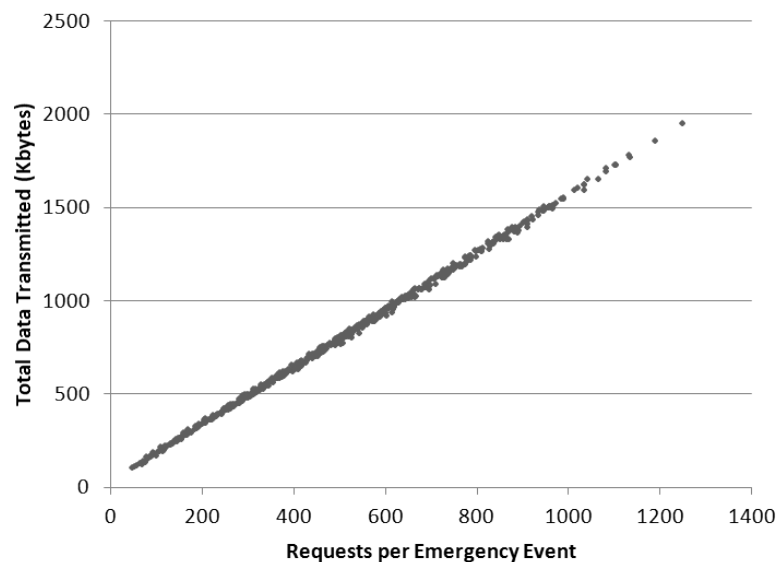


Figura 6.15. Gráfico apresentando a relação entre o número de requisições por evento de emergencia e a quantidade de dados transmitidos.

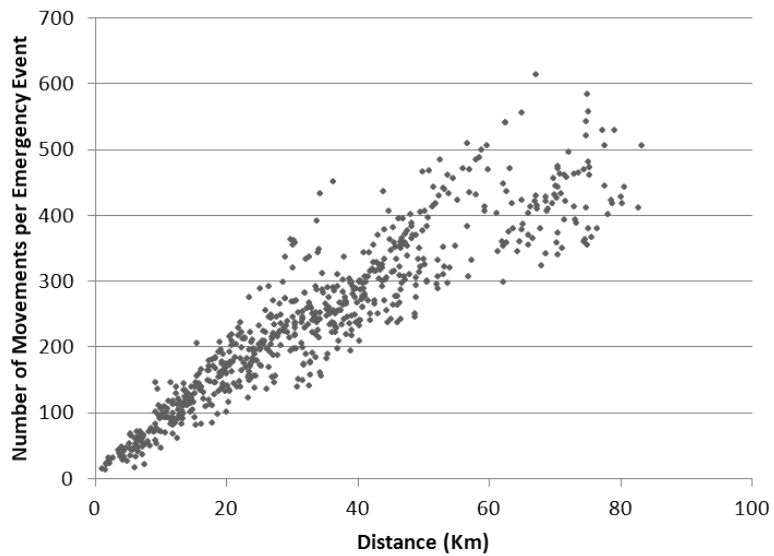


Figura 6.16. Gráfico apresentando a relação entre o número de movimentações realizadas pelas unidades móveis de emergência e a distância.

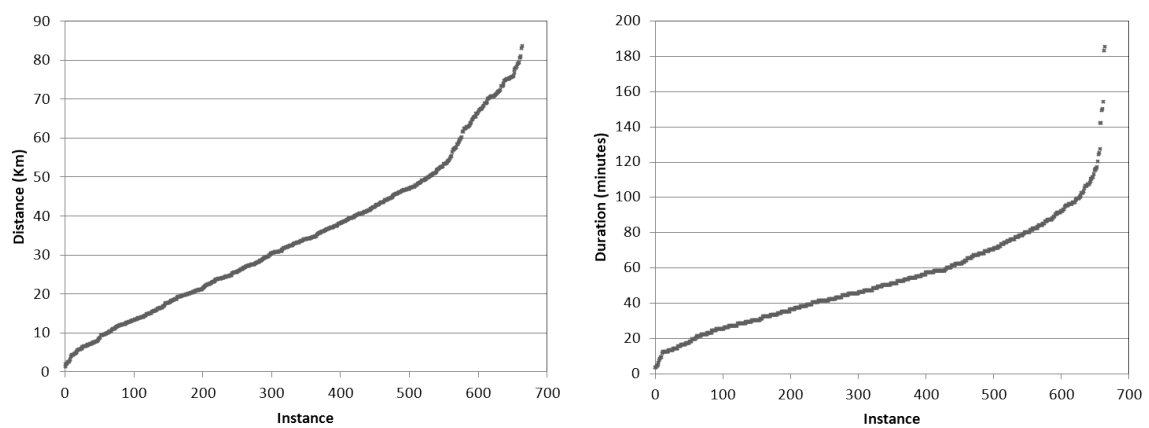


Figura 6.17. Gráfico apresentando a distribuição da distância e a duração das instâncias coletadas.

(i.e., requisição), o componente *EventManager* (i.e., o gerenciador de eventos no lado do servidor) possui o maior consumo em *bytes* de todos os componentes avaliados. Porém o envio do servidor para o cliente (i.e., resposta), o componente *UserManager* (i.e., o gerenciador de usuários) obteve um maior consumo em *bytes* transmitidos. Este resultado é explicável pelos motivos apresentados abaixo:

1. **Dados recebidos do cliente para o servidor:** Devido ao fato do servidor de eventos fornecer serviços para os outros componentes do sistema, inclusive componentes também no lado do servidor, ele recebe uma quantidade maior de dados comparando com outros componentes. Calculamos a participação por volta de 79 por cento do total de dados enviados para o servidor. Concluimos que este valor se deve ao fato de boa parte dos dados recebidos pelo *EventManager* se originarem de componentes no lado do servidor, tais como o *UserManager* (e.g., mudança de localização dos marcos móveis) e *EmergencyManager* (e.g., mudança de estados nos eventos de emergência).
2. **Dados enviados do servidor para o cliente:** Identificamos o componente *UserManager* como o componente que envia a maior quantidade de dados para o cliente, com 65 por cento do total dos dados enviados. A razão disto está no fato deste enviar a cada finalização do fluxo de atendimento a emergências a listagem dos marcos presentes na área georeferenciada, com o objetivo de atualizar as posições de cada um dos marcos móveis. Apesar de não termos aplicado métodos de compressão de dados neste caso, sugere-se a utilização do mesmo para uma diminuição na quantidade de dados recebidos pelo cliente. Um dos motivadores para isto está no fato das mensagens enviadas para o cliente utilizarem o formato XML. Mensagens no formato XML podem ser facilmente comprimidas para a otimização do envio de dados do servidor para o cliente Web. É necessário neste caso avaliarmos também o custo computacional da compressão de dados no lado do cliente em função do processamento e da energia. Isto se deve a diversas limitações de hardware e software já destacadas neste trabalho, tais como limitações no poder de processamento e energia do dispositivo móvel.

A partir dos resultados coletados (Figura 6.18), o componente *EmergencyManager* participa com uma porcentagem muito pequena no envio e recebimento de dados do cliente, já que este possui o objetivo de receber os identificadores das entidades envolvidas (i.e., usuário móvel, evento de emergência e marco fixo), atualizá-las de acordo com o estado da emergência e enviar uma mensagem de retorno ao cliente. A quantidade de dados transmitidos é bem menor comparado com os outros componentes no lado do servidor.

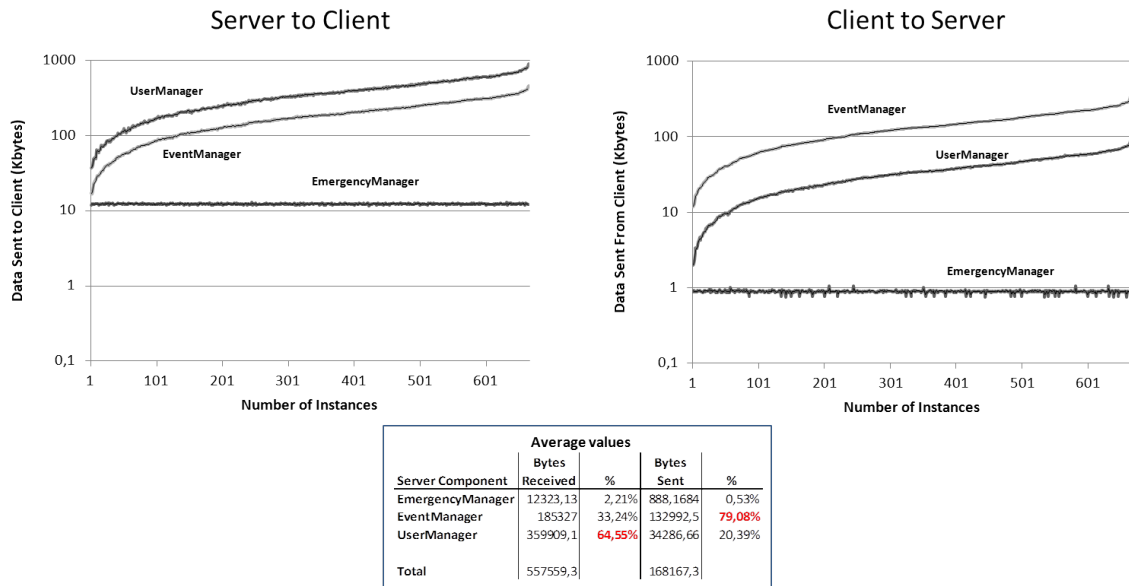


Figura 6.18. Dados transmitidos do lado do servidor.

6.2.4 Cobertura do Serviço DECS

Com o objetivo de avaliarmos a cobertura realizada pelo serviço DECS no atendimento de eventos de emergência, coletamos informações das localizações de todos os eventos gerados no sistema, que inclui as coordenadas da posição respectiva. Utilizando os limites definidos na tabela 5.1, calculamos a posição dos 660 eventos de emergência gerados de forma pseudo-aleatória na amostra utilizada. O objetivo desta avaliação foi garantir que a distribuição dos pontos gerados esteja bem distribuída sobre a área pré-definida do serviço. A Figura 6.19 apresenta as posições onde o gerador pseudo-aleatório criou eventos de emergência para serem tratados pelo serviço. Conforme esperado e desejado, observamos que houve uma boa distribuição dos eventos gerados sobre a área definida de cobertura do serviço de emergências.

Investigando um pouco mais além, associamos as áreas que obtiveram as maiores densidades de eventos de emergência com as regiões no mapa georeferenciado, conforme apresentado na Figura 6.20. Com este mapa, foi possível localizarmos as regiões que obtiveram as maiores densidades de eventos gerados. Coincidentemente, as áreas que possuíam a menor quantidade de ruas e avenidas tiveram as menores densidades de eventos gerados. Em alguns casos, uma pequena quantidade de eventos de emergência não puderam ser tratados pelo serviço (e.g., 2,25 por cento), devido à impossibilidade da UnM em se deslocar até o local por não existir uma estrada ou caminho até o ponto. Ao não encontrar uma rota entre dois pontos, o serviço GMaps API informa ao cliente

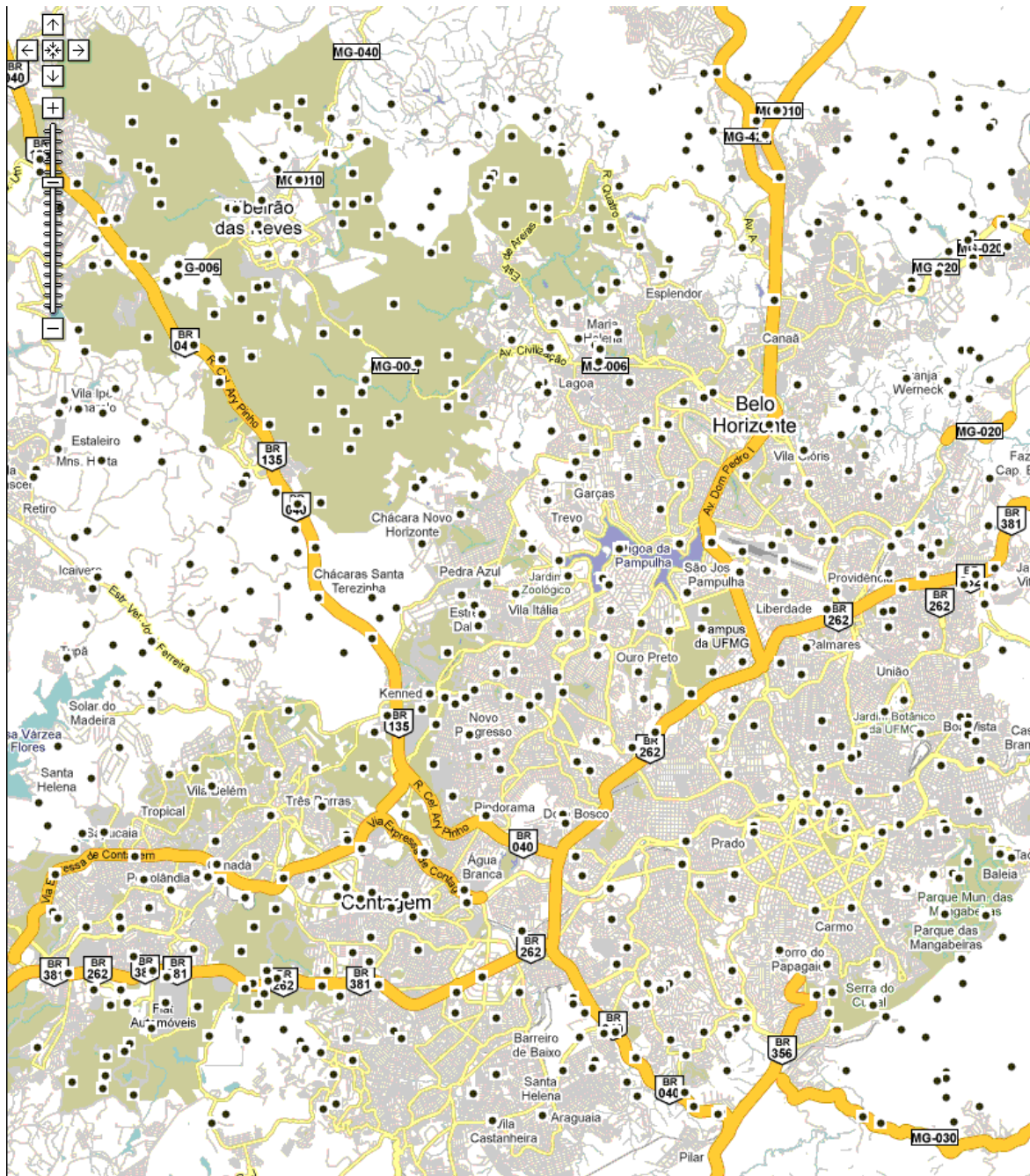


Figura 6.19. Cobertura do Serviço DECS na área geográfica pré-definida no trabalho.

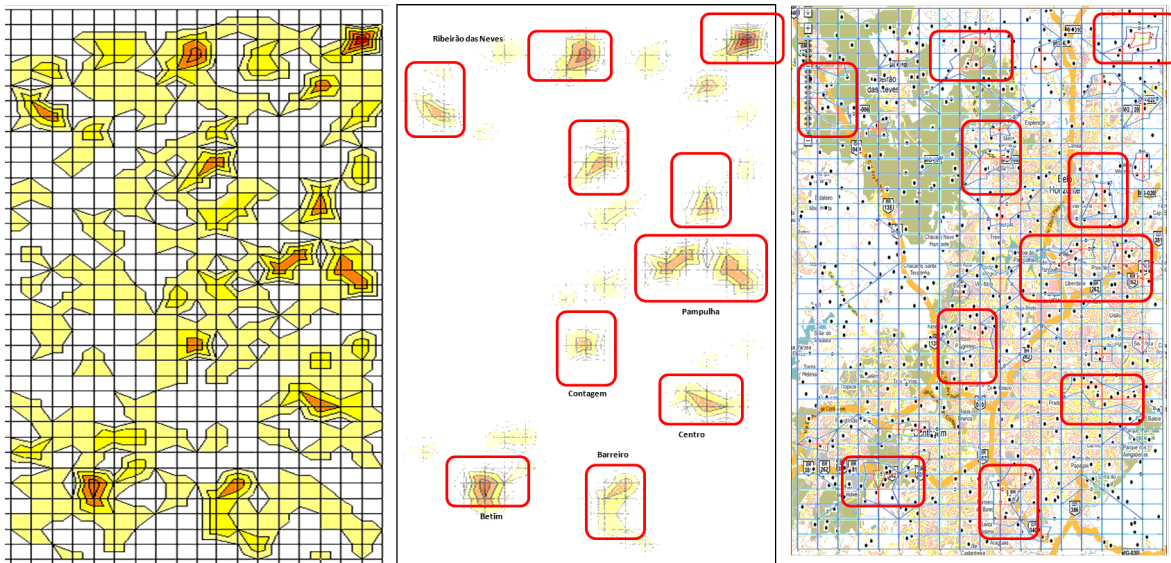


Figura 6.20. Resultado da cobertura do Serviço DECS.

Tabela 6.5. Distribuição dos eventos de emergência por tipo.

Tipo de Emergência	Quantidade
Incêndio	208
Saúde	213
Segurança	221

Web de que não foi possível definir um caminho até o destino.

Além de avaliarmos a distribuição dos pontos gerados no mapa, verificamos também a distribuição dos eventos de emergência gerados para cada um dos tipos disponíveis (e.g., saúde, incêndio e segurança). Conforme mostrado na Tabela 6.5, o gerador pseudo-aleatório foi capaz de gerar eventos para cada um dos tipos de forma distribuída. Com estas informações, seria possível, por exemplo, verificarmos a distribuição dos eventos de emergência em função da localização, a fim de descobrirmos pontos em que ocorreram uma quantidade maior de eventos para cada um dos tipos.

6.2.5 Testes Em Dispositivos Reais

Devido à dificuldade de acesso a um dispositivo real com o sistema operacional *Android* para a execução dos protótipos desenvolvidos neste trabalho, avaliamos outras alternativas. A tabela 6.6 descreve o estudo realizado sobre a compatibilidade de recursos em navegadores Web para dispositivos móveis. O principal objetivo deste estudo é prover um dispositivo real capaz de executar o serviço DECS utilizando tecnologias Web para o desenvolvimento de aplicações. Neste estudo, consideramos as plataformas

Tabela 6.6. Compatibilidade de navegadores Web para a execução do DECS.

Navegador Web	WebView	Opera Mobile	Skyfire	Web Browser for S60
Plataforma	Google Android	Sun JavaME	S60/Symbian OS, Windows Mobile	S60/Symbian OS
Tipo de avaliação	Emulado via SDK	Dispositivo real	Dispositivo real	Dispositivo real
Máquina de leiaute	WebKit	Presto	Proprietária	WebKit
Renderização por servidor remoto	Não	Opera Binary Markup Language (OBML) [39]	Gecko Runtime Environment	Não
Tipo de saída	JavaScript e HTML	OBML	Imagem	Imagem
Suporte a GMaps	Sim	Não	Sim	Não
Suporte a JavaScript	Limitado	Limitado	Sim	limitado
Suporte a Adobe Flash	Não	Não	Sim	Não

Android, JavaME e *Symbian OS* e navegadores Web em cada uma das plataformas para os testes. Para os testes reais, utilizamos um dispositivo *Nokia E65* com acesso a redes WLAN contendo as plataformas S60/Symbian e JavaME.

De todos os navegadores apresentados na tabela 6.6, o primeiro protótipo DECS foi capaz de ser executado apenas no navegador *Skyfire* instalado no dispositivo *Nokia E65*. Apesar da menor resolução de tela em comparação a um dispositivo da plataforma *Android*, o dispositivo real foi capaz de acessar a aplicação Web localizada no servidor em nuvem e prover os recursos assíncronos de eventos e mensagens.

Em relação ao segundo protótipo do serviço DECS, a aplicação cliente não executou em nenhum dos navegadores apresentados. Uma das principais razões que acreditamos estar relacionada a este fator está no fato de utilizarmos uma quantidade muita grande de recursos do GWT. O GWT converte o código desenvolvido inicialmente em Java para *JavaScript*.

6.3 Análise de Desempenho e Escalabilidade

O objetivo desta seção é apresentar a avaliação do desempenho e escalabilidade do servidor de eventos proposto neste trabalho. A avaliação foi realizada em termos da capacidade de gerenciamento de um número considerável de usuários móveis concorrentes (5 a 1000, 5000). Os testes de desempenho têm como meta avaliar o tempo de resposta das principais operações definidas e realizadas pelo servidor de eventos, enquanto os testes de escalabilidade de carga possuem como meta avaliar a escalabilidade do servidor em situações de aumento de carga. Tínhamos o desejo de avaliar um número maior de interfaces definidas no servidor, porém isto não foi possível devido

Quotas	Quota Gratuita		Quota Paga	
	Limite Diário	Taxa Máxima	Limite Diário	Taxa Máxima
Requisições	1.3M reqs/dia	7.4K reqs/min	43M reqs/dia	30K reqs/min
Dados transmitidos	1 GB	7.4 GB/min	1 TB	10 GB/min
Dados recebidos	1 GB	56 MB/min	1 TB	10 GB/min
Tempo de CPU	6.5 CPU horas	15 CPU-mins/min	1729 CPU horas	72 CPU-mins/min

Tabela 6.7. As quotas providas pela infraestrutura do servidor em nuvem.

às limitações impostas pelo plano gratuito de quota disponível no servidor em nuvem¹. Sendo assim, fomos capazes de avaliar um subconjunto de todas as interfaces definidas e disponíveis pelo servidor de eventos. A Tabela 6.7 apresenta as limitações impostas pela quota gratuita utilizada neste trabalho.

Na gestão do consumo de recursos de aplicações em nuvem, o servidor em nuvem utiliza um conjunto predefinido de variáveis. Estes recursos estão sujeitos às quotas, e em casos em que eles são totalmente consumidos, o servidor em nuvem paralisa² os serviços utilizados na aplicação por clientes Web. Além da quota gratuita, o servidor em nuvem fornece quotas pagas para o provimento adicional de recursos às aplicações em nuvem. De todas as variáveis e recursos disponíveis no servidor, consideramos os seguintes recursos para os nossos testes:

- **Requisições:** O número total de requisições processadas pela aplicação e o tempo de resposta das requisições.
- **Largura de banda:** A quantidade de dados enviados e recebidos pela aplicação nas requisições.
- **Tempo de CPU:** O tempo total de processamento da CPU no servidor no processamento de requisições, incluindo o tempo gasto pela aplicação e em operações no *DataStore*. O *DataStore* é responsável por gerir operações de dados (CRUD³, requisições SQL) sobre o SGBD utilizado pelo servidor.
- **Tempo de execução de ciclos:** O tempo médio, mínimo e máximo de execução dos ciclos de testes em clientes.
- **Timeouts:** A quantidade média de *timeouts* ocorridos nos clientes em função de erros ocorridos no servidor.
- **Eventos publicados e consumidos:** O número total de eventos publicados e consumidos pela *thread* cliente.

¹Informações de Quota: <http://code.google.com/appengine/docs/quotas.html>

²Negação de quota ou '*quota denial*'

³CRUD = *Create, Retrieve, Update, Delete*.

- **Negação de quotas e erros:** O número total de erros ocorridos no servidor devido a negações de quota ou erros internos.

Apesar de utilizarmos o plano gratuito de quotas, fomos capazes de avaliar o desempenho de uma boa parte das interfaces no servidor de eventos proposto neste trabalho. Dependendo dos tipos de testes realizados, os clientes foram capazes de consumir o limite diário imposto pelo plano em poucas horas. Isto nos obrigou a realizar os testes em intervalos regulares em diferentes dias, já que as limitações de quota são renovadas a cada 24 horas. Definimos neste trabalho três principais testes:

- **Teste 1:** Avaliação de desempenho das operações providas pelo servidor de eventos.
- **Teste 2:** Avaliação da escalabilidade de carga na publicação de eventos por clientes móveis.
- **Teste 3:** Avaliação da escalabilidade no provimento de acesso por múltiplos clientes móveis.

Para o provimento dos clientes móveis nos testes acima, utilizamos a criação e execução de *threads* de processamento concorrente contendo as funcionalidades avaliadas. Estas *threads* foram acionadas a partir de um único computador a fim de garantir as características únicas do ambiente de testes tais como a disponibilidade de banda, condições de tráfego de rede, características e desempenho de *hardware/software*. Nos clientes de teste, utilizamos um computador *quad-core* 2.8 GHz CPU com 3GB de RAM e uma conexão de rede *Ethernet* de 1GB. A Figura 6.21 apresenta um diagrama descrevendo a configuração definida nos testes de desempenho e escalabilidade.

6.3.1 Teste 1: Desempenho na Execução de Operações do Servidor de Eventos

Avaliamos neste teste o tempo de resposta necessário para executar as principais operações providas pelo servidor de eventos proposto neste trabalho. As operações avaliadas incluem: *login* do usuário, subscrição do usuário a um tópico, publicação de eventos, listagem de eventos e consumo de eventos pelo usuário. Algumas destas operações estão diretamente relacionadas entre si como, por exemplo, para que o cliente consuma um evento, ele deverá executar o *login* e se inscrever em tópicos para que ele receba eventos para consumo. Desta forma, o consumo de eventos irá utilizar mais recursos no servidor do que, por exemplo, subscrição a tópicos ou *login* no sistema. Utilizamos

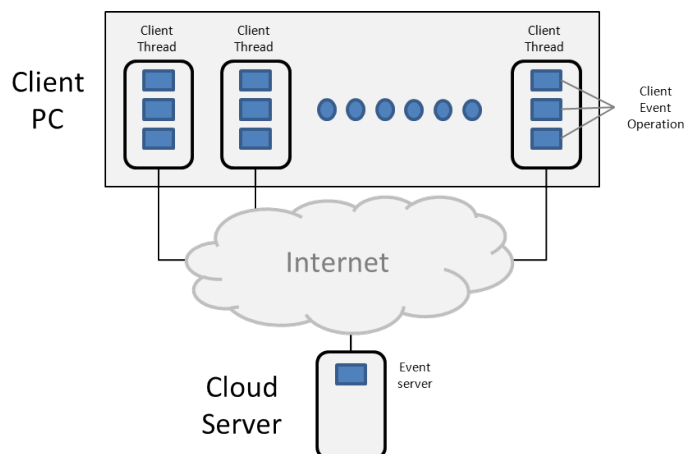


Figura 6.21. A configuração utilizada nos testes de desempenho e escalabilidade.

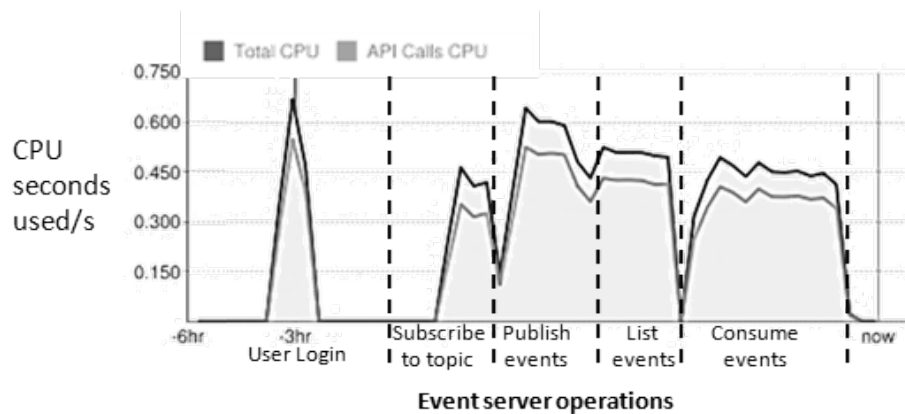
Operações	Média (s)	Mínimo (s)	Máximo (s)
<i>login</i>	0,55	0,39	9,58
<i>subscribe</i>	0,42	0,23	7,48
<i>publish</i>	0,50	0,36	5,7
<i>listEvents</i>	0,31	0,25	5,9
<i>consumeEvent</i>	1,15	0,87	12,97

Tabela 6.8. Resultados do teste 1.

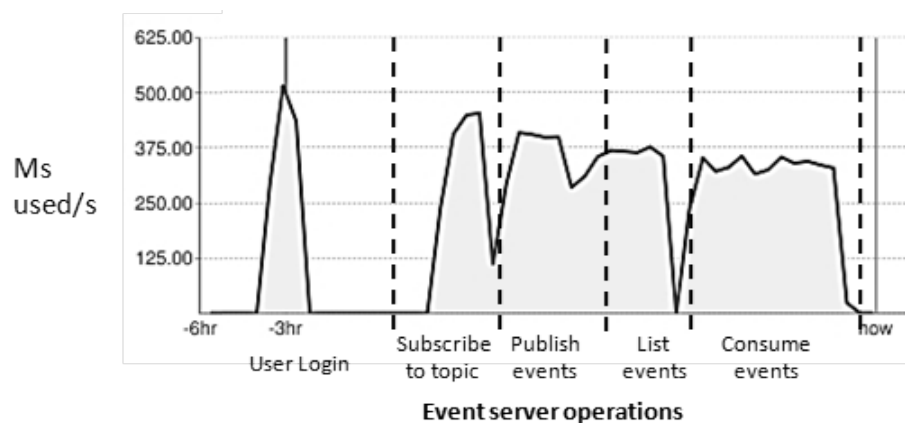
uma amostra de 1000 instâncias que foram executadas sobre o servidor. A Tabela 6.8 apresenta os resultados obtidos.

Baseado nos resultados apresentados na Tabela 6.8, podemos apresentar algumas conclusões. Considerando os casos médios, o servidor de eventos respondeu bem dentro do tempo de resposta desejado (abaixo de 500ms), com exceção da operação do consumo de eventos que obteve um tempo médio de resposta acima de 1s. A operação de consumo de evento recupera o objeto de evento a ser consumido e altera seu estado para 'consumido'. O fato desta operação depender de outras operações para sua execução pode ter contribuído para o tempo adicional de resposta coletado durante os testes.

Em relação ao consumo de recursos, podemos visualizar como as operações avaliadas consumiram o processamento de CPU no servidor, conforme mostrado na Figura 6.22(a). Este gráfico provê uma medida de consumo de recursos de todas as atividades, incluindo atividades especificamente relacionadas à API do GAE, incluindo requisições de dados no *DataStore*. A Figura 6.22(b) também apresenta o tempo utilizado pela aplicação para cada segundo disponível no servidor. Na medida que as operações se tornaram mais complexas, a aplicação exigiu uma quantidade maior de segundos de CPU. A área de cada divisão no gráfico apresentado na Figura 6.22(b) oferece o tempo total em que a operação foi executada no servidor, já que $ms/sec \times sec = ms$.



(a) Segundos de CPU usados por segundo.



(b) Milissegundos usados do servidor por segundo.

Figura 6.22. Teste 1: uso de CPU no servidor.

6.3.2 Teste 2: Escalabilidade na Publicação de Eventos

Avaliamos neste teste a capacidade do servidor em nuvem e do servidor de eventos em tratar requisições concorrentes ao receber eventos e publicá-los para notificação em clientes móveis. Neste cenário, as *threads* clientes enviam informações de eventos ao servidor e recebem de volta o objeto de evento criado no servidor de eventos. Variamos a quantidade de instancias de 100 à 1000 *threads* clientes simultâneos. Medimos o tempo de resposta médio no processamento das requisições (ms/req), o número de *timeouts* dado pelo servidor e medidas adicionais de consumo de recursos providos pelo servidor em nuvem. Estas medidas incluem a quantidade máxima de requisições processadas por segundo, tempo máximo de processamento de requisições, quantidade máxima de dados transmitidos (*Bytes* enviados/recebidos por segundo) e a quantidade

Número de <i>Threads</i>	100	200	500	700	800	900	1000
Processamento de requisições (Req/s) Máx	8	15	7	20	26	33	26
Tempo em requisições (S/req) Máx	5,0	3,6	5,0	7,5	7,5	7,5	6,0
Dados recebidos (KB/s) Máx	5,8	9,0	5,8	9,0	3,2	9,0	7,0
Dados enviados (KB/s) Máx	5,0	8,8	5,0	8,0	2,6	8,0	5,8
Tempo de CPU (CPU s/s)	3,6	4	3,2	3,6	2,2	3,6	4,5
Tempo de resposta Médio (s)	1,3	1,4	2,2	2,5	2,6	2,7	2,7
Tempo de resposta Mín (s)	0,34	0,33	0,31	0,31	0,36	0,33	0,33
Tempo de resposta Máx (s)	9,8	10,2	11,5	12,5	13,5	12,2	12,6
Número de <i>Timeouts</i>	0	5	17	213	212	376	517
Número de erros/s	2,9	4	0,3	1,8	1	3,2	4,5
Número de <i>threads</i> criadas	300	1000	2500	3500	2400	4500	5000

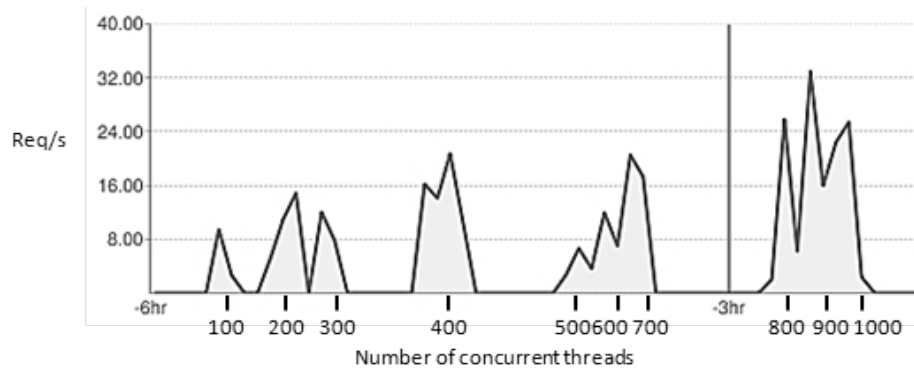
Tabela 6.9. Resultados do teste 2.

de segundos de CPU utilizados por segundo no servidor. A Tabela 6.9 apresenta os resultados coletados nos testes. As figuras 6.23, 6.24 e 6.25 apresentam os gráficos gerados pelo servidor em nuvem utilizado nos testes.

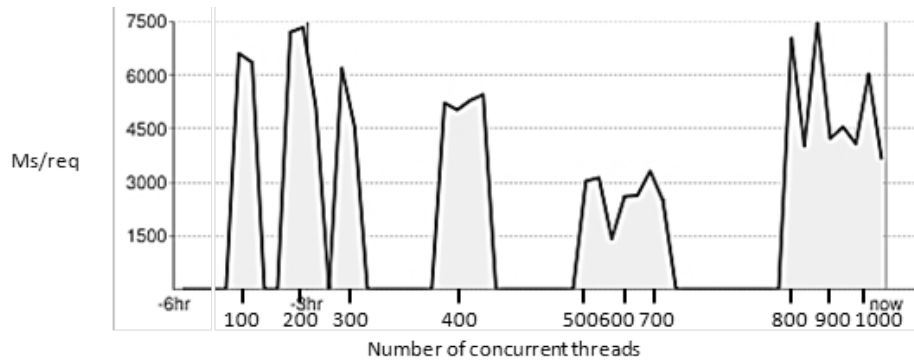
A partir dos resultados apresentados na Tabela 6.9, podemos observar que o servidor em nuvem não foi capaz de tratar uma quantidade maior que 33 reqs/s. A medida que o número de clientes concorrentes aumentou, o número de *timeouts* e erros aumentou, conforme mostrado na Figura 6.24(b). Os erros foram causados por dois motivos: a não-resposta do servidor e o gasto total de recursos no *DataStore* e tempo de CPU. O consumo de tempo de CPU pode ser visualizado na Figura 6.24(a). Em alguns casos, o servidor em nuvem informou através de *logs* que a aplicação requisitou uma quantidade maior de recursos do que a quantidade disponível no momento do consumo. O tempo de resposta para requisições ficou estável (entre 3.6s e 7.5s) para clientes concorrentes, entretanto, com o aumento de *timeouts* e erros no servidor devido a limitações de quota impostas pelo servidor. O número de *timeouts* aumentou significativamente devido a restrições de quota, causando uma não-resposta do servidor a uma quantidade grande de requisições a partir de testes com 500 clientes concorrentes.

6.3.3 Teste 3: Escalabilidade em Clientes Concorrentes

Este teste de escalabilidade apresenta uma avaliação mais interessante comparada aos testes realizados previamente. Neste cenário, cada *thread* cliente se comporta virtualmente como um cliente móvel acessando o servidor em nuvem e requisitando as operações de serviço de eventos. Cada cliente executa em ciclos contendo um conjunto de operações que incluem *login*, assinatura por tópicos de eventos, publicação de eventos do cliente, recepção de eventos do servidor e consumo de eventos no cliente. Variamos o número de ciclos entre 5 e 14 para cada *thread* cliente, de forma pseudo aleatória. Em cada ciclo, utilizamos um fator de probabilidade de 0.4 para a pub-



(a) Requisições processadas por segundo.

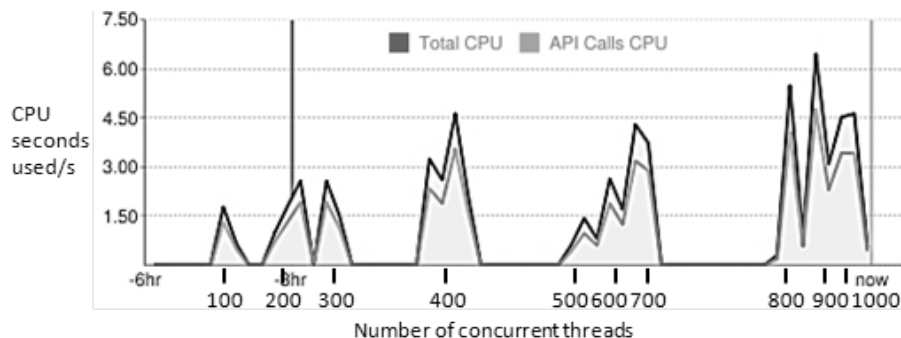


(b) Milissegundos por requisição.

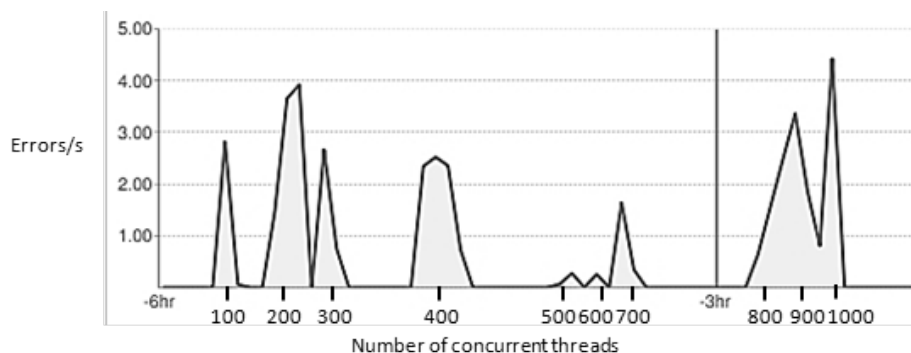
Figura 6.23. Teste 2: processamento de requisições.

licação de eventos do cliente para o servidor. Desta forma, para cada 10 clientes, 4 destes publicavam eventos no servidor. Este fator foi utilizado porque não tínhamos o desejo de clientes publicarem eventos no servidor em todos os ciclos de execução. Em casos comuns, incluindo o que foi observado na aplicação DroidGuide, a publicação de eventos não ocorre em todos os ciclos de execução no dispositivo móvel. A Tabela 6.10 apresenta os resultados coletados a partir do servidor em nuvem em função dos testes de escalabilidade realizados. A Figura 6.28 apresenta os gráficos gerados pelo servidor em nuvem.

Em relação a recursos necessários para iniciar e processar requisições, quando estes estão totalmente gastos, o servidor em nuvem retorna uma mensagem HTTP 403 '*Forbidden status*' como resposta ao invés de repassar a requisição para o tratador de requisições. Os seguintes recursos no servidor possuem este comportamento: requisições, tempo de CPU e largura de banda (entrada e saída). Para



(a) Segundos de CPU usadas por segundo.

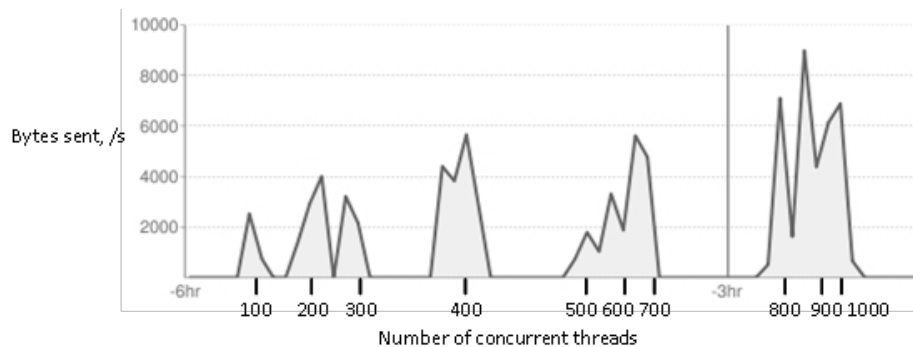


(b) Número de erros por segundo.

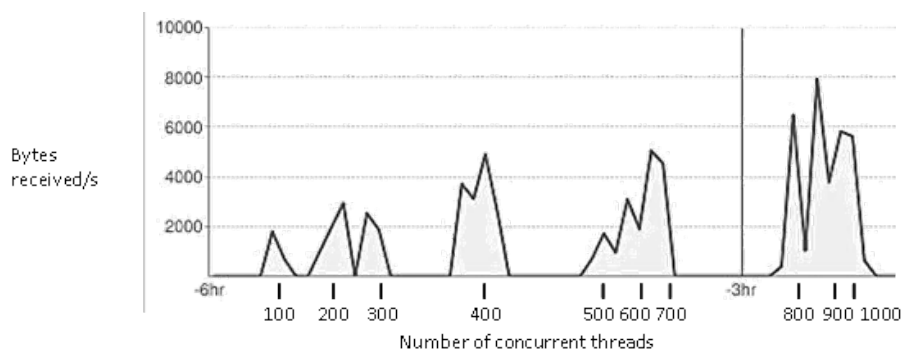
Figura 6.24. Teste 2: número de erros e uso de CPU.

todos os outros recursos, quando estes estão totalmente gastos, uma tentativa da aplicação em consumir o recurso resulta em uma exceção no servidor do tipo *com.google.apphosting.api.ApiProxy.OverQuotaException*, conforme mostrado pelo do servidor na Figura 6.29.

Baseado nos resultados coletados, o servidor respondeu às requisições de acordo até 100 clientes concorrentes. Alguns problemas ocorreram quando o número de clientes passou de 500, onde o servidor respondeu com mensagens de negação de quota e erros, causando diversos *timeouts*. Tentamos executar testes com até 5000 clientes concorrentes, conforme mostrado na Tabela 6.10. Entretanto, a configuração definida no computador cliente não foi capaz de executar esta quantidade alta de requisições de forma simultânea. Obtivemos algumas informações a respeito desta instância de teste (5000 clientes), observando que houveram uma enorme quantidade de erros ocorridos no lado do servidor. Devido à grande quantidade de *timeouts*, o número de requisições processadas no servidor diminuiu, diminuindo assim o uso de segundos na CPU,



(a) Dados enviados por segundo.



(b) Dados recebidos por segundo.

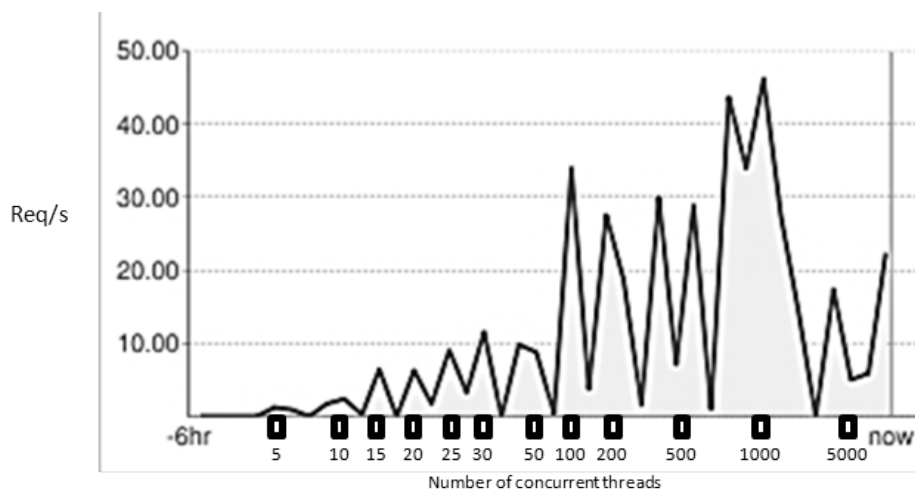
Figura 6.25. Teste 2: transmissão de dados.

conforme mostrado na Figura 6.28.

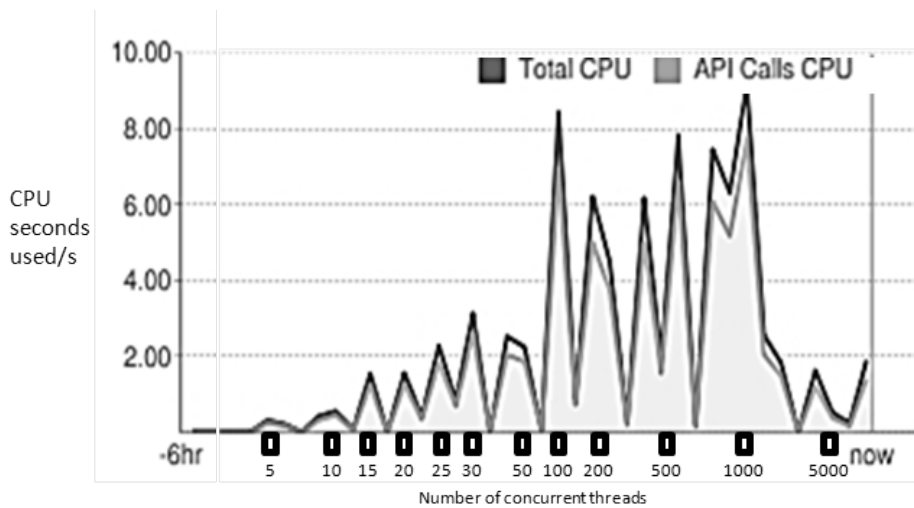
Número de <i>Threads</i>	5	10	15	20	25	30
Processamento de requisições (Reqs/s)	1,3	2,5	6,5	6,2	11	9
Tempo em requisições (S/req)	0,4	0,6	0,4	0,6	0,7	0,7
Dados recebidos (KB/s)	0,32	0,62	1,6	1,5	2,8	2,3
Dados enviados (KB/s)	0,3	0,5	1,4	1,3	2,5	1,9
Tempo de CPU (CPU s/s)	0,3	0,6	1,5	1,6	3,2	2,3
Número de negações de quota/s	0	0	0	0	0	0
Taxa de erros (Erros/s)	0	0	0	0	0	0
Tempo de execução de ciclo Méd. (s)	51,4	54,3	12,4	13,1	15,3	16,1
Tempo de execução de ciclo Mín. (s)	23,2	15,9	5,2	5,9	4,4	6,5
Tempo de execução de ciclo Máx. (s)	90,7	98,4	23,2	26,7	27,9	29,6
Número de Timeouts	0	0	0	0	0	5
Número total de eventos publicados	125	245	349	437	629	669
Número total de ciclos executados	2186	4701	6871	9460	13766	13153

Número de <i>Threads</i>	50	100	200	500	1000	5000
Processamento de requisições (Reqs/s)	10	25	28	30	45	47
Tempo em requisições (S/req)	0,7	1,9	1,0	5,8	4,0	3,0
Dados recebidos (KB/s)	2,4	7,0	6,8	7,0	11,0	12,0
Dados enviados (KB/s)	2,0	6,0	7,0	7,6	12,0	8,0
Tempo de CPU (CPU s/s)	2,5	8	6	6	7,5	9
Número de negações de quota/s	0	0	1,2	2,5	5	0,5
Taxa de erros (Erros/s)	0	0,5	2	4,5	13	8
Tempo de execução de ciclo Méd. (s)	13,5	17,9	9,1	3,8	2,1	-
Tempo de execução de ciclo Mín. (s)	6,4	7,1	6,5	9,5	8,1	-
Tempo de execução de ciclo Máx. (s)	27,5	41,8	36,7	37,5	44,8	-
Número de Timeouts	32	19	484	417	901	-
Número total de eventos publicados	997	1960	2382	1438	3329	-
Número total de ciclos executados	20321	39909	49939	28983	69754	-

Tabela 6.10. Resultados do teste 3.

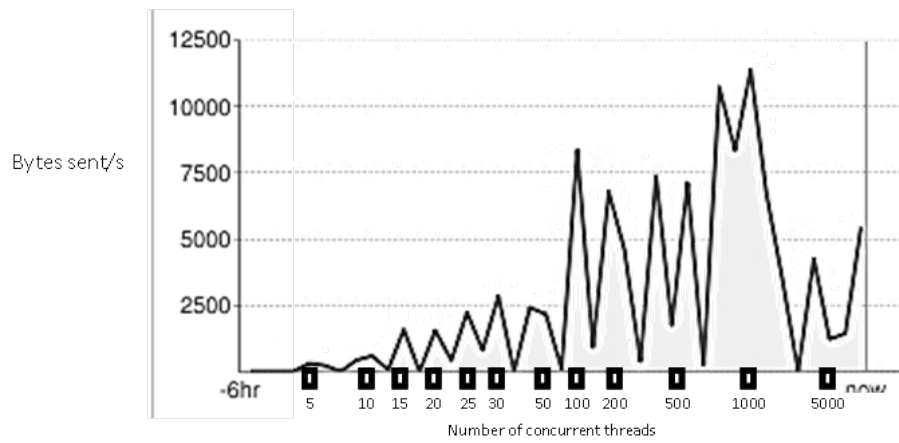


(a) Requisições processadas por segundo.

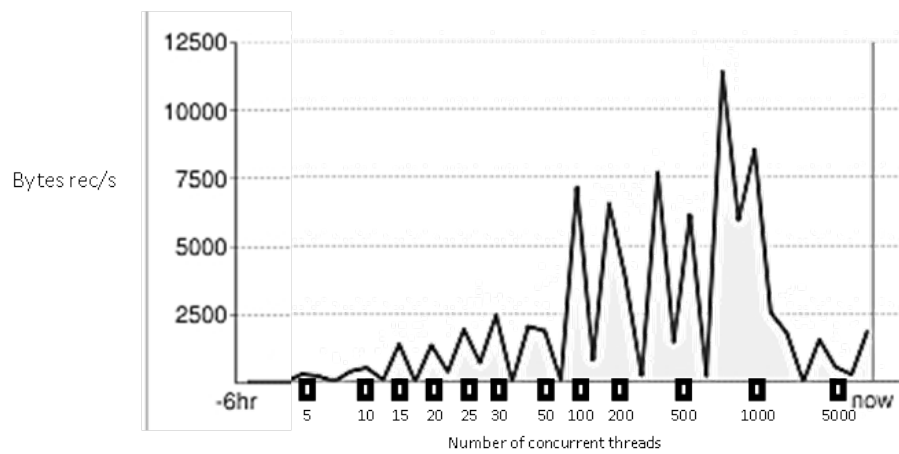


(b) Segundos de CPU usados por segundo.

Figura 6.26. Teste 3: processamento de requisições e uso de CPU.

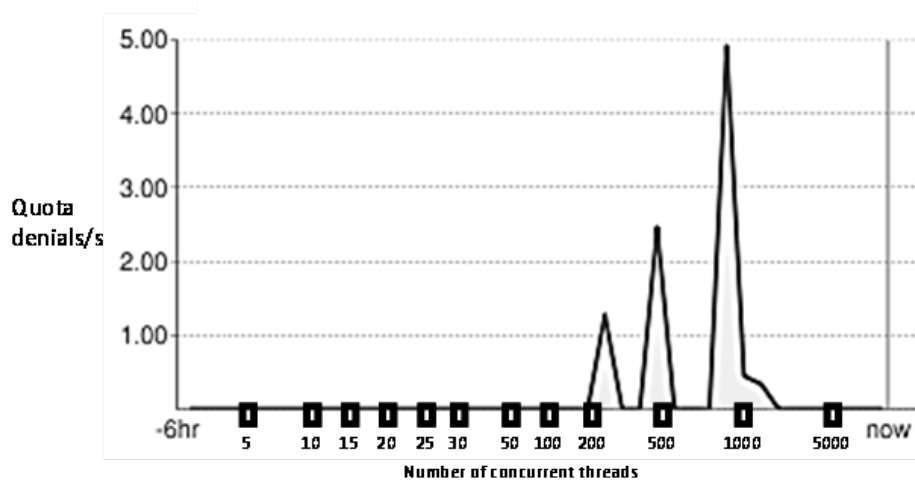


(a) Datos enviados por segundo.

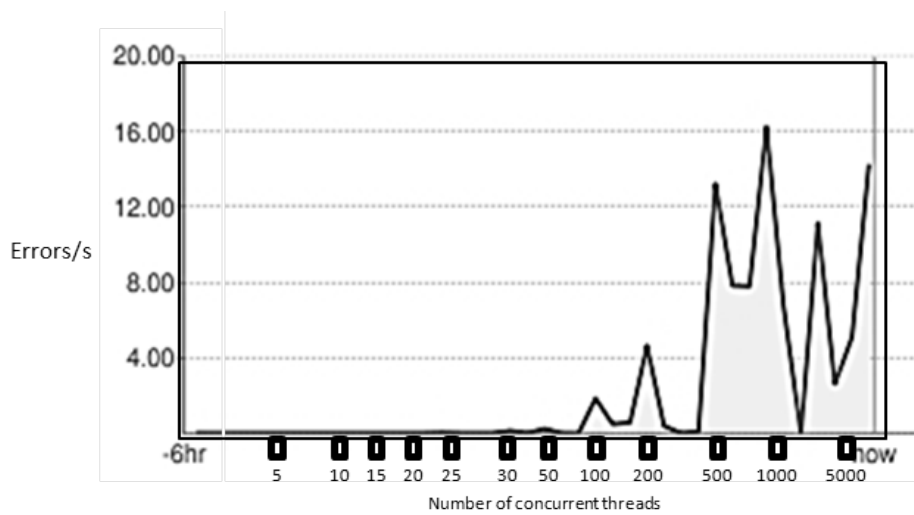


(b) Datos recibidos por segundo.

Figura 6.27. Teste 3: datos transmitidos.



(a) Negações de quota por segundo.



(b) Erros por segundo.

Figura 6.28. Teste 3: erros e negações de quota.

```

07-29 02:16PM 22.538 /event?operation=CONSUME_EVENT_MESSAGE&key=267021 500 2840ms 66cpu_ms 8api_cpu_ms 0kb Java/1.6.0_16,gzip(gfe)
150.164.5.121 - - [29/Jul/2010:14:16:25 -0700] "GET /event?operation=CONSUME_EVENT_MESSAGE&key=267021 HTTP/1.1" 500 0 -
"Java/1.6.0_16,gzip(gfe)" "droidguide20.appspot.com" ms=2841 cpu_ms=67 api_cpu_ms=8 cpm_usd=0.001873 pending_ms=2791
throttle_code=4

07-29 02:16PM 25.367
Uncaught exception from servlet
com.google.apphosting.api.ApiProxy$OverQuotaException: The API call datastore_v3.Get() required more quota than is available.
    at com.google.apphosting.runtime.ApiProxyImpl$AsyncApiFuture.rpcFinished(ApiProxyImpl.java:273)
    at com.google.net.rpc.RpcStub$RpcCallbackDispatcher$1.runInContext(RpcStub.java:1025)
    at com.google.tracing.TraceContext$TraceContextRunnable$1.run(TraceContext.java:444)
    at com.google.tracing.TraceContext.runInContext(TraceContext.java:684)
    at com.google.tracing.TraceContext$AbstractTraceContextCallback.runInInheritedContextNoUnref(TraceContext.java:322)
    at com.google.tracing.TraceContext$AbstractTraceContextCallback.runInInheritedContext(TraceContext.java:314)
    at com.google.tracing.TraceContext$TraceContextRunnable.run(TraceContext.java:442)
    at com.google.net.rpc.RpcStub$RpcCallbackDispatcher.rpcFinished(RpcStub.java:1046)
    at com.google.net.rpc.RPC.internalFinish(RPC.java:2038)
    at com.google.net.rpc.impl.RpcNetChannel.finishRpc(RpcNetChannel.java:2352)
    at com.google.net.rpc.impl.RpcNetChannel.messageReceived(RpcNetChannel.java:1279)
    at com.google.net.rpc.impl.RpcConnection.parseMessages(RpcConnection.java:319)
    at com.google.net.rpc.impl.RpcConnection.dataReceived(RpcConnection.java:290)
    at com.google.net.async.Connection.handleReadEvent(Connection.java:474)
    at com.google.net.async.EventDispatcher.processNetworkEvents(EventDispatcher.java:831)
    at com.google.net.async.EventDispatcher.internalLoop(EventDispatcher.java:207)
    at com.google.net.async.EventDispatcher.loop(EventDispatcher.java:103)
    at com.google.net.async.GlobalEventRegistry$2.runLoop(GlobalEventRegistry.java:95)
    at com.google.net.async.LoopingEventDispatcher$EventDispatcherThread.run(LoopingEventDispatcher.java:378)

```

Figura 6.29. Erro de disponibilidade de quota no *DataStore* do servidor.

Capítulo 7

Considerações Finais

O objetivo deste capítulo é apresentar os principais resultados obtidos neste trabalho e algumas propostas de trabalhos futuros que podem ser trabalhadas a partir dos resultados deste.

7.1 Resultados

Nos protótipos desenvolvidos, o servidor de eventos foi capaz de capturar mudanças de perfil e contexto da aplicação. Os eventos foram capturados no dispositivo e enviados para o serviço de eventos remoto. A partir dos interesses definidos pelo usuário no perfil, o escalonador de atividades no servidor foi acionado para definir atividades turísticas de acordo com os interesses do turista. A subscrição de SWBIs pela aplicação móvel e a notificação de eventos ocorridos nestes tipos de serviços também apresentou a viabilidade na utilização destes em aplicações ubíquas.

No que diz respeito aos serviços remotos utilizados neste trabalho, os SWBIs foram capazes de enviar mensagens de notificação para o dispositivo em duas situações: local e remotamente. Localmente, o processador de eventos em execução no dispositivo detectou mudanças de dados de perfil e contexto e enviou eventos o servidor remoto de dados, onde foram publicados e compartilhados nos SWBIs sensíveis a dados de perfil e contexto. Remotamente, os SWBIs criaram eventos remotos em razão de mudanças de dados de perfil e contexto remotos ou devido aos eventos criados e publicados no servidor de eventos pelo cliente móvel. Este trabalho apresentou os seguintes resultados:

- **Especificação e implementação do servidor de eventos:** Apresentação de uma especificação e implementação de um servidor de eventos para uso em aplicações móveis;

- **Desenvolvimento de Protótipos:** Desenvolvimento de dois protótipos, neste caso o guia turístico *DroidGuide* e o serviço de contexto de emergências DECS;
- **Uso de um servidor de dados em nuvem:** Avaliação do uso de um servidor de dados em nuvem para o provimento dos serviços utilizados neste trabalho, tais como o servidor de eventos e os serviços Web;
- **Uso de tecnologias Web:** Desenvolvimento de protótipos de aplicações móveis utilizando tecnologias Web, tais como o AJAX, serviço de visualização de mapas *Google Maps* e os arcabouços GAE e GWT.

7.2 Trabalhos Futuros

Esta seção apresenta alguns dos possíveis trabalhos futuros a serem desenvolvidos a partir dos resultados obtidos neste trabalho. As propostas de trabalhos futuros foram divididas nos seguintes temas: modelagem de dados e plataformas de software, segurança e transmissão de dados, processamento de eventos em aplicações móveis, e localização e informações georeferenciadas.

7.2.1 Modelagem de Dados e Plataformas de Software

A modelagem e disponibilização de dados de serviços externos (dados de vôos, clima, atrações turísticas, restaurantes, atrações culturais, tráfego, segurança e emergência) a serem utilizados pelo servidor de eventos pode ser um trabalho futuro que agrega de forma direta em aplicações ubíquas. Enquanto neste trabalho utilizaram-se dados simulados, a utilização de dados reais externos beneficiaria o uso da Web em aplicações móveis através de seus serviços, muitos destes já disponíveis para computadores *desktop*.

A adaptação do arcabouço desenvolvido em Java e *Python* para os protótipos apresentados neste trabalho para outras plataformas móveis, tais como o JavaME e *Windows Mobile*, também pode oferecer resultados interessantes, já que será possível a avaliação dos protótipos e outras aplicações móveis em diferentes plataformas de software para dispositivos. A grande vantagem deste trabalho que incentiva esta avaliação está no fato dos protótipos apresentados usarem tecnologias Web para o processamento de eventos, notificação por mensagens e serviços remotos, tais como localização por mapas georeferenciados e informações sensíveis ao contexto, tais como localização, estado e condição do usuário/aplicação, identidade, atividade e tempo.

7.2.2 Segurança e Transmissão de Dados

Apesar de não estarem no principal foco deste trabalho, alguns requisitos de segurança tais como a criptografia, anonimato, autorização e autenticação possuem um importante papel em aplicações ubíquas. Estes garantem a proteção dos dados através da utilização de métodos de criptografia e de identificação garantindo assim a autenticidade não só do usuário, mas também de serviços externos a serem oferecidos para aplicações móveis. Esta garantia na transmissão e processamento de informações do usuário fortalece a confiança do usuário em utilizar serviços remotos disponíveis a ele e também a aplicações móveis no dispositivo. A otimização na transferência de dados em aplicações móveis/ubíquas é de extrema importância, já que o custo de transmissão está diretamente relacionado à quantidade de dados a serem transferidos, quantificado na forma de custo (em reais ou dólares) por bytes transferidos e energia necessária para transmitir e receber estes dados. Recursos neste caso como a compressão, simplificação de mensagens XML, adaptabilidade do sistema de requisições/respostas entre o cliente e o servidor e o uso de requisições HTTP *POST* podem diminuir direta ou indiretamente o custo na transmissão e processamento de informações oriundas de um servidor de dados remoto. Na área de comunicação de dados entre o servidor e o cliente, outros trabalhos futuros podem ser sugeridos. O suporte a novos padrões de comunicação, tais como a tecnologia *push* que permite a comunicação oriunda do servidor com o cliente, possibilitando assim uma melhor interação entre estes durante a execução dos serviços e a geração de eventos e notificações no servidor remoto. O suporte a outros protocolos de mensagens tais como o XMPP também contribuiria para a padronização na comunicação de mensagens em XML, permitindo também a utilização de outros serviços que utilizam o mesmo padrão tais como os de mensagens instantâneas.

7.2.3 Processamento de Eventos em Aplicações Móveis

No que diz respeito ao servidor de eventos, vários trabalhos futuros podem ser sugeridos. O primeiro deles seria a inclusão de outros tipos de SWBIs, tais como localização e tráfego. A integração do SWBIs com outros módulos dos protótipos apresentados neste trabalho pode também ser estendida, como por exemplo, na adaptação durante o consumo de conteúdo, no contexto na coleta de dados de tráfego e localização e no escalonador de atividades obtendo eventos de atividades selecionadas, executadas e não executadas pelo usuário, sugestões de atividades, dentre outros. Neste trabalho, podemos destacar três perfis de lógica de processamento de eventos: uma distribuída onde o processamento de eventos está presente no cliente e no servidor remoto, outra centralizada apenas no servidor e uma apenas no dispositivo. Destes perfis, apenas a

primeira foi abordada nos dois protótipos apresentados.. Um possível trabalho futuro está na exploração de cenários de aplicações móveis/ubíquas que poderiam usufruir de cada um destes perfis. O sistema possivelmente poderá identificar e traçar diferentes perfis de aplicações móveis de tal forma a associá-los com os perfis de lógica de processamento de eventos identificados e propostos neste trabalho. Outra possibilidade inclui a configuração do perfil de processamento de eventos no dispositivo e no servidor remoto com o objetivo de aperfeiçoar a utilização dos recursos presentes nos dispositivos móveis como a bateria e o processador e o custo de transmissão de dados na rede. Este perfil de processamento incluiria a opção de definir os nós responsáveis pelo processamento de eventos mediante as condições do dispositivo, tais como o acesso á rede (e.g., qualidade do canal, tipo), custo de transmissão (e.g., valor, consumo de energia), densidade de eventos sendo gerados.

7.2.4 Localização e Informações Georeferenciadas

Os métodos de rastreamento da localização dos usuários em aplicações móveis/ubíquas cientes de contexto também podem ser avaliados, de tal forma a possibilitar a seleção ou adaptação ao método mais preciso e prático (e.g., custo em termos de energia) para um dado momento na execução de serviços ao usuário. Alguns destes métodos que também foram apresentados neste trabalho incluem a interface GPS do *Android* presente no emulador e outros métodos de localização por triangulação, mapeamento de pontos de acesso sem fio, dentre outros. Na apresentação de conteúdo voltado ao contexto para o usuário, os mapas georeferenciados foram de grande utilidade na disponibilização de informações sensíveis ao contexto, especialmente relacionadas à localização. Estudos aprimorados da usabilidade de diferentes tipos de interfaces para a apresentação de informações voltadas ao contexto do usuário/aplicação podem contribuir para garantir uma melhor interatividade entre o usuário, a aplicação móvel e o servidor de dados remoto. Outros estudos incluem a adaptabilidade de interfaces Web em dispositivos móveis e a navegabilidade/acessibilidade de aplicações móveis sensíveis ao contexto.

7.2.5 Composição de Serviços Web

Utilizamos neste trabalho SWBIs capazes de utilizar dados de perfil e contexto para notificar o usuário móvel de informações referentes eu seu estado ou durante mudanças de estado em dados de perfil e contexto remoto. Desejamos também como trabalho futuro a utilização da composição de serviços, onde cada parte compartilha informações coletadas a fim de prover serviços mais complexos ao usuário móvel, conforme proposto em Urbietta et al [57] e Bronsted et al [6].

Podemos citar, por exemplo, um serviço móvel de acompanhamento de viagens aéreas. Este serviço composto pode ser composto por diversos serviços, cada um responsável em obter as seguintes informações: (a) dados do voo (e.g., horário, situação) e condições do aeroporto de origem/destino, (b) informações de tráfego até o aeroporto, (c) informações climáticas na origem e destino, (d) alocação de recursos de transporte (e.g., de um local até o aeroporto de origem e do aeroporto destino até o local desejado). Cada um destes serviços simples seriam servidores de informações para um serviço ainda maior, podendo assim prover informações ainda mais úteis para o usuário móvel. Desta forma, seria possível provermos SWBIs ainda mais complexos e mais significativos ao usuário do que os utilizados neste trabalho.

Referências Bibliográficas

- [1] AMQP Group (2010). Advanced Message Queuing Protocol. <http://www.amqp.org/confluence/display/AMQP/Advanced+Message+Queuing+Protocol>.
- [2] Apple Inc. (2010). iPhone Software Development Kit. <http://developer.apple.com/iphone/>.
- [3] Baldoni, R.; Querzoni, L. & Virgillito, A. (2005). Distributed event routing in publish/subscribe communication systems: a survey. Technical report.
- [4] Bessho, M.; Kobayashi, S.; Koshizuka, N. & Sakamura, K. (2008). A space-identifying ubiquitous infrastructure and its application for tour-guiding service. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1616–1621, New York, NY, USA. ACM.
- [5] Bishop, D. (1992). The marble answering machine. <http://design.cca.edu/graduate/uploads/pdf/marbleanswers.pdf>.
- [6] Bronsted, J.; Hansen, K. M. & Ingstrup, M. (2010). Service composition issues in pervasive computing. *IEEE Pervasive Computing*, 9:62–70.
- [7] Camp, P. J.; Hudson, J. M.; Keldorph, R. B.; Lewis, S. & Mynatt, E. D. (2000). Supporting communication and collaboration practices in safety-critical situations. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pp. 249–250, New York, NY, USA. ACM.
- [8] Caporuscio, M. & Inverardi, P. (2005). Uncertain event-based model for egocentric context sensing. In *SEM '05: Proceedings of the 5th international workshop on Software engineering and middleware*, pp. 25–32, New York, NY, USA. ACM.
- [9] Carzaniga, A.; Rosenblum, D. S. & Wolf, A. L. (2001). Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383.

- [10] Chakraborty, D.; Joshi, A.; Finin, T. & Yesha, Y. (2005). Service composition for mobile environments. *Mob. Netw. Appl.*, 10(4):435–451.
- [11] Chandy, K. M. (2006). Event-driven applications: Costs, benefits and design approaches. <http://www.infospheres.caltech.edu/sites/default/files/Event-Driven%20Applications%20-%20Costs,%20Benefits%20and%20Design%20Approaches.pdf>.
- [12] Chang, H. & Lee, K. (2009). Quality-driven web service composition for ubiquitous computing environment. *New Trends in Information and Service Science, International Conference on*, 0:156–161.
- [13] Chen, G. & Kotz, D. (2000). A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA.
- [14] Chen, H. (2004). *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, Department of Computer Science, University of Maryland.
- [15] Chen, P. Y.; Chen, W. T.; Wu, C. H.; Tseng, Y.-C. & Huang, C.-F. (2007). A group tour guide system with rfids and wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 561–562, New York, NY, USA. ACM.
- [16] Cheverst, K.; Davies, N.; Mitchell, K.; Friday, A. & Efstratiou, C. (2000). Developing a context-aware electronic tourist guide: some issues and experiences. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 17–24, New York, NY, USA. ACM.
- [17] Cho, Y.; Choi, J. & Choi, J. (2007). A context-aware workflow system for a smart home. In *ICCIT '07: Proceedings of the 2007 International Conference on Convergence Information Technology*, pp. 95–100, Washington, DC, USA. IEEE Computer Society.
- [18] Christensen, J. H. (2009). Using restful web-services and cloud computing to create next generation mobile applications. In *OOPSLA '09: Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pp. 627–634, New York, NY, USA. ACM.
- [19] Debaty, P.; Goddi, P. & Vorbau, A. (2005). Integrating the physical world with the web to enable context-enhanced mobile services. *Mob. Netw. Appl.*, 10(4):385–394.

- [20] Dey, A. K. (2000). *Providing architectural support for building context-aware applications*. PhD thesis, Atlanta, GA, USA.
- [21] Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7.
- [22] ECMA International (1999). JavaScript Object Notation. <http://json.org/>.
- [23] Greenfield, A. (2006). *Everyware: The Dawning Age of Ubiquitous Computing*. Peachpit Press, Berkeley, CA, USA.
- [24] Haahr, M.; Meier, R.; Nixon, P.; Cahill, V. & Jul, E. (2000). Filtering and scalability in the eco distributed event model. In *PDSE '00: Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems*, p. 83, Washington, DC, USA. IEEE Computer Society.
- [25] Hans-Werner Gellersen, Michael Beigl, H. K. (1999). The media cup. <http://mediacup.teco.edu>.
- [26] Hertzog, P. & Torrens, M. (2004). Context-aware mobile assistants for optimal interaction: a prototype for supporting the business traveler. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*, pp. 256–258, New York, NY, USA. ACM.
- [27] Jabber Open Source Community (1999). XMPP Standards Foundation. <http://xmpp.org/>.
- [28] Jbara, S.; Kuffik, T.; Soffer, P. & Stock, O. (2007). Context aware communication services in active museums. *IEEE International Conference on Software Science, Technology and Engineering*, 0:127–135.
- [29] Jiang, X.; Hong, J. I.; Takayama, L. A. & Landay, J. A. (2004). Ubiquitous computing for firefighters: field studies and prototypes of large displays for incident command. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 679–686, New York, NY, USA. ACM.
- [30] Johanson, B.; Fox, A. & Winograd, T. (2002). The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):67–74.
- [31] Joohyun Han¹ Contact Information, Y. C. C. I. & Choi, J. (2005). Context-aware workflow language based on web services for ubiquitous computing. *Computational Science and Its Applications ICCSA 2005*, 3481/2005(4):1008–1017.

- [32] Kuffik, T.; Sheidin, J.; Jbara, S.; Goren-Bar, D.; Soffer, P.; Stock, O. & Zancanaro, M. (2007). Supporting small groups in the museum by context-aware communication services. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pp. 305–308, New York, NY, USA. ACM.
- [33] Long, S.; Kooper, R.; Abowd, G. D. & Atkeson, C. G. (1996). Rapid prototyping of mobile context-aware applications: the cyberguide case study. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pp. 97–107, New York, NY, USA. ACM.
- [34] Maamar, Z.; Benslimane, D. & Narendra, N. C. (2006). What can context do for web services? *Commun. ACM*, 49(12):98–103.
- [35] Mann, S. (1997). Wearable computing: A first step toward personal imaging. *Computer*, 30(2):25–32.
- [36] Meier, R. & Cahill, V. (2002). Taxonomy of distributed event-based programming systems. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pp. 585–588, Washington, DC, USA. IEEE Computer Society.
- [37] Mühl, G.; Fiege, L. & Pietzuch, P. (2006). *Distributed Event-Based Systems*. Springer, 1st edição.
- [38] Microsoft Inc. (2009). .NET Compact Framework. <http://msdn.microsoft.com/en-us/netframework/aa497273.aspx>.
- [39] Opera SW (2006). Opera Binary Markup Language. <http://dev.opera.com/articles/view/opera-binary-markup-language/>.
- [40] Pallapa, G.; Roy, N. & Das, S. (2007). Precision: Privacy enhanced context-aware information fusion in ubiquitous healthcare. In *SEPCASE '07: Proceedings of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, p. 10, Washington, DC, USA. IEEE Computer Society.
- [41] Patrick Eugster, Benoit Garbinato, A. H. (2009). Middleware support for context-aware applications. In *Middleware for Network Eccentric and Mobile Applications*, pp. 305–322. Springer Berlin Heidelberg.
- [42] Pietzuch, P. R. & Bacon, J. (2002). Hermes: A distributed event-based middleware architecture. In *ICDCSW '02: Proceedings of the 22nd International Conference on*

- Distributed Computing Systems*, pp. 611–618, Washington, DC, USA. IEEE Computer Society.
- [43] Poulymenopoulou, M.; Malamateniou, F. & Vassilacopoulos, G. (2003). Specifying workflow process requirements for an emergency medical service. *J. Med. Syst.*, 27(4):325–335.
- [44] Python Software Foundation (2010). Python Programming Language. <http://www.python.org/>.
- [45] Rossi, P. & Tari, Z. (2006). Software adaptation for service-oriented systems. In *MW4SOC '06: Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*, pp. 12–17, New York, NY, USA. ACM.
- [46] Sacramento, V.; Endler, M.; Rubinsztein, H. K.; Lima, L. S.; Goncalves, K.; Nascimento, F. N. & Bueno, G. A. (2004). Moca: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10):2.
- [47] Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17.
- [48] Schilit, B.; Adams, N. & Want, R. (1994). Context-aware computing applications. In *WMCSA '94: Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pp. 85–90, Washington, DC, USA. IEEE Computer Society.
- [49] Schmidt, A. (2003). *Ubiquitous Computing - Computing in Context*. PhD thesis, Computing Department, Lancaster University, UK.
- [50] Starner, T. E. (2002). Wearable computers: No longer science fiction. *IEEE Pervasive Computing*, 1(1):86–88.
- [51] Sun Microsystems Inc. (2001). Java Message Service. <http://java.sun.com/products/jms/>.
- [52] Sun Microsystems Inc. (2004). JSR 172: J2ME Web Services Specification. <http://jcp.org/en/jsr/detail?id=172>.
- [53] Sun Microsystems Inc. (2006). Mobile Information Device Profile 2.0. <http://java.sun.com/products/midp/>.
- [54] Sun Microsystems Inc. (2010). Java Micro Edition. <http://java.sun.com/j2me/>.
- [55] Symbian Foundation (2010). Symbian OS. <http://www.symbian.com/>.

- [56] Takeuchi, Y. & Sugimoto, M. (2009). A user-adaptive city guide system with an unobtrusive navigation interface. *Personal Ubiquitous Comput.*, 13(2):119–132.
- [57] Urbietta, A.; Barrutieta, G.; Parra, J. & Urizarren, A. (2008). A survey of dynamic service composition approaches for ambient systems. In *SOMITAS '08: Proceedings of the 2008 Ambi-Sys workshop on Software Organisation and MonIToring of Ambient Systems*, pp. 1–8, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [58] Vaquero, L. M.; Rodero-Merino, L.; Caceres, J. & Lindner, M. (2009). A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55.
- [59] Violet (2009). Nabaztag. <http://www.nabaztag.com/en/index.html>.
- [60] Wang, K.; Sui, Y.; Zou, X.; Durrezi, A. & Fang, S. (2008). Pervasive and trustworthy healthcare. In *AINAW '08: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, pp. 750–755, Washington, DC, USA. IEEE Computer Society.
- [61] Want, R.; Borriello, G.; Pering, T. & Farkas, K. I. (2002). Disappearing hardware. *IEEE Pervasive Computing*, 1(1):36–47.
- [62] Want, R.; Hopper, A.; Falcao, V. & Gibbons, J. (1992). The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102.
- [63] Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11.
- [64] Weiser, M. & Brown, J. S. (1995). Designing calm technology. <http://nano.xerox.com/weiser/calmtech/calmtech.htm>.
- [65] World Wide Web Consortium (W3C) (2007). Simple Object Access Protocol. <http://www.w3.org/TR/soap/>.
- [66] Yue, W.; Mu, S.; Wang, H. & Wang, G. (2005). Tgh: a case study of designing natural interaction for mobile guide systems. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pp. 199–206, New York, NY, USA. ACM.