UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

INFOCITY: SELEÇÃO CONTEXTUALIZADA DA INFORMAÇÃO DE EVENTOS EM UMA CIDADE INTELIGENTE.

TRABALHO DE GRADUAÇÃO

Bruno Romero de Azevedo

Santa Maria, RS, Brasil

2013

INFOCITY: SELEÇÃO CONTEXTUALIZADA DA INFORMAÇÃO DE EVENTOS EM UMA CIDADE INTELIGENTE.

Bruno Romero de Azevedo

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientadora: Profª Drª Iara Augustin

348
Santa Maria, RS, Brasil

2013

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

INFOCITY: SELEÇÃO CONTEXTUALIZADA DA INFORMAÇÃO DE EVENTOS EM UMA CIDADE INTELIGENTE.

elaborado por
Bruno Romero de Azevedo

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Iara Augustin, Dr[a]
(Presidente/Orientadora)

Ana Winck, Dr[a] (UFSM)

Eduardo Kessler Piveta, Dr (UFSM)

Santa Maria, 20 de Fevereiro de 2013.

“Você esperando respostas, olhando pro espaço;
E eu tão ocupado vivendo. Eu não me pergunto, eu faço!”
— Raul Seixas

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer à minha família por todo o apoio que sempre foi demonstrado (e exercido) por ela. Meus pais e irmãos incessantemente me dando forças e incentivando durante todo o percurso que tenho trilhado, servindo como base para a minha formação pessoal.

Gostaria, também, de agradecer aos meus professores sempre humildes e dispostos a ajudar. Em especial à professora Iara Augustin, responsável por me guiar e auxiliar desde que entrei para o seu grupo de pesquisa em computação móvel no segundo semestre de 2009.

E, por fim, aos meus colegas, meu muito obrigado por todas as experiências compartilhadas e pelo convivío agradável e edificante que partilhamos durante tantos anos.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

INFOCITY: SELEÇÃO CONTEXTUALIZADA DA INFORMAÇÃO DE EVENTOS EM UMA CIDADE INTELIGENTE.

AUTOR: BRUNO ROMERO DE AZEVEDO
ORIENTADORA: IARA AUGUSTIN
Local da Defesa e Data: Santa Maria, 20 de Fevereiro de 2013.

Este trabalho tem por objetivo desenvolver um sistema capaz de permitir a integração dos cidadãos com o intuito de auxiliar no desenvolvimento e descobrimento de suas cidades. A maneira com a qual pensou-se em elaborar este sistema, chamado de *InfoCity* (aglutinação, em inglês, de *informação* e *cidade* para dar a ideia de cidade informada), foi a partir do compartilhamento da informação de eventos em uma cidade inteligente. De modo que, a partir de um aplicativo para dispositivos móveis inteligentes, os *smartphones*, baseados no sistema *Android*, fosse possível a visualização e adição destes eventos em um mapa. Porém, como o número de eventos pode ser elevado, grande parte deles seriam de pouca utilidade ao usuário, por isso, se for de seu interesse, os eventos retornados devem estar de acordo com o contexto em que ele se encontra. Assim, seriam-lhe apresentados apenas os fatos que tem maiores chances de serem relevantes a ele.

Para o armazenamento das informações de eventos compartilhados, um servidor teve de ser criado. Dessa forma, a obtenção e registro dos dados por parte do aplicativo deve ser feita através de requisições GET e POST. É do servidor, também, a responsabilidade de definir que eventos são relevantes ao usuário a partir do contexto em que ele se encontra.

Portanto, percebe-se que o aplicativo tem um uso social muito prático. Ao facilitar a utilização do sistema, podemos inferir um maior conforto de uso e, assim, maior número de usuários. O que torna-o capaz de agregar um grande número de informações de eventos que acontecem pela cidade e, também, uma maior quantidade de discussões sobre os caminhos que ela está tomando. Além, também, de oferecer aos usuários informações mais confiáveis e variadas.

Palavras-chave: Cidade Inteligente. Sensibilidade ao Contexto. Recuperação de Texto. Computação Móvel. Android.

ABSTRACT

Undergraduate Final Work
Undergraduate Program in Computer Science
Federal University of Santa Maria

CONTEXTUAL SELECTION OF INFORMATION FROM EVENTS IN AN INTELLIGENT CITY.

AUTHOR: BRUNO ROMERO DE AZEVEDO
ADVISOR: IARA AUGUSTIN
Defense Place and Date: Santa Maria, February 20st, 2013.

This work has as objective the development of a system capable of allowing the citizen integration with the intention of help the development and discovery of their cities. The way that was thought in order to elaborate this system, called *InfoCity* (informed city), was through the sharing of the information of events in an intelligent city. In a way that, by using an application for smartphones based on *Android*, was possible to visualize and add them on a map. However, because the number of events can be high, many of them could be useless for the user. So, if it was his interest, the returned events should be according to the context of where he is at. Thus, would be shown to him only the facts that has more chances in being relevant to him.

For the storage of the shared event information, a server had to be created. Thereby, the acquiring and registration of the data by the application should be made through GET and POST requests. It's the server, also, the responsability of defining which events are relevant for the user from the context of where he is at.

Therefore, we can realize that the app has a very practical social usage. With the facilitation of the usage of the system we can infer a greater comfort of use and, thus, a greater number of users. This way it becomes capable of aggregate a large number of information of events that happens around the city and, also, a larger quantity of discussions about the paths that it's taking. As well to offer the users more informations that are trustable and variable.

Keywords: Intelligent City, Context Awareness, Text Retrieval, Mobile Computing, Android.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CI	Cidade Inteligente
GPS	*Global Positioning System*
IDE	*Integrated Development Environment*
IDF	*Inverse Document Frequency*
JSON	*JavaScript Object Notation*
MVC	*Model-View-Controller*
ORM	*Object/Relational Mapping*
OS	*Operating System*
SDK	*Software Development Kit*
TF	*Term Frequency*
TIC	Tecnologia da informação e comunicação
UML	*Unified Modeling Language*
VCS	*Version Control System*

SUMÁRIO

1 INTRODUÇÃO

Nosso dia-a-dia frequentemente nos apresenta diferentes desafios. Seja por desconhecer lugares que estejamos indo ou por algo fora do normal ter acontecido. Muitos destes fatos, que ocorrem pela cidade, são anunciados e discutidos pelos cidadãos somente quando estes se encontram em bares ou em ônibus, por exemplo. Esse tipo de informação que alguns individuos possuem e compartilham com poucos, poderia ser de uso para grande parte da população, ou seja, é de relevância a todos. Como não foi encontrado um meio de fácil acesso que permita esse tipo de divulgação que poderia ser utilizado pelos cidadãos, a informação acaba se restringindo a poucos e, com o tempo, se perdendo. Impedindo, assim, que problemas não cheguem ao ouvido de pessoas capazes de resolvê-los ou, então, que não se tenham discussões que gerariam tais soluções.

Com isso, é proposta neste trabalho a criação de um sistema capaz de permitir o compartilhamento de informações de acontecimentos (ou eventos) em uma cidade inteligente (CI). Existem diversas definições sobre o que é uma CI (KOMNINOS, 2006). Tais como comunidades que utilizam tecnologia da informação para transformar a vida dentro de suas regiões e, também, ambientes cujas tecnologias da informação e comunicação (TICs) estão presentes de forma transparente. Para este trabalho, CIs serão pensadas, ainda, como "territórios que trazem sistemas de inovação e TICs para dentro da mesma localidade, combinando a criatividade de indivíduos talentosos que fazem parte da população da cidade (...)" (KOMNINOS, 2006).

Cidades inteligentes também pressupõe uma cobertura ampla de conexão e uma cultura de uso da tecnologia. Tendo em vista o constante crescimento da utilização de dispositivos móveis inteligentes -os *smartphones*-, em especial os baseados no sistema *Android*[1] (ALEXANDER, 2012), percebe-se que estes dispositivos móveis podem ser instrumentos para a publicação e visualização dos acontecimentos pela cidade, não só pela questão da mobilidade, mas também pela facilidade de seu uso.

Ainda, este sistema teria um grande apelo social, uma vez que os próprios cidadãos seriam os responsáveis pela criação e discussão de eventos. Dessa forma, cria-se um senso de responsabilidade cívica em cada um, pois eles vão se sentir co-responsáveis pelos problemas e soluções que envolvem suas cidades.

Sabe-se que ocorrem inúmeros eventos em uma cidade e que cada um teria seus próprios

[1] Disponível em: http://www.android.com. Acesso em janeiro de 2013.

interesses quanto a eles. Isto é, nem todos os eventos da cidade seriam de importância para um cidadão em determinado momento. Buscando trazer uma experiência de uso do sistema mais agradável, propõe-se que os eventos apresentados para a visualização do usuário estejam de acordo com o contexto em que este se encontra no momento em que acessa as informações.

1.1 Escopo

Sabe-se que para melhor aproveitar o conhecimento de todos cidadãos, deve-se ter, além do fluxo de informações entre cidadão e governo, um fluxo entre os próprios cidadãos de maneira integrada para uma democracia mais participativa. Atualmente, isto não acontece nos sistemas implementados na área de governo eletrônico (e-gov), havendo apenas interação entre cidadão-governo ou governo-cidadão.

Este trabalho se encaixa justamente abrindo um fluxo cidadão-cidadão, permitindo a eles mesmos a divulgação, discussão e resolução de problemas de interesse de um grupo social. Em adição, permite que qualquer órgão governamental verifique as informações de eventos e, juntamente com a população, resolva as adversidades. Criando, assim, um sistema e-gov mais dinâmico e reativo. Isto é, o governo adquire a capacidade de exercer suas funções de forma democrática, mais efetiva e cuja performance envolveria menos custos (RELYEA, 2002).

1.2 Motivação

Tendo em vista que ocorrem muitos eventos nas cidades e, em nossas pesquisas, não termos encontrado um meio de visualizá-los de forma centralizada, participativa e prática, o presente trabalho foi pensado em ser desenvolvido. Capacitando, assim, os cidadãos a compartilhar as informações sobre os diversos acontecimentos em suas cidades.

A importância deste tipo de sistema reside no fato que estes eventos compartilhados seriam de relevância a uma comunidade. Sendo assim, cada um poderia ajudar a encontrar uma solução para os problemas apresentados ou, então, poderiam descobrir, através das informações disponibilizadas por outros, fatos interessantes sobre a sua cidade.

O que este sistema gera, de fato, é a integração cidadã no auxílio do exercício pleno de sua cidadania.

1.3 Objetivos

O objetivo geral deste trabalho é modelar e prototipar uma plataforma para o descobrimento e desenvolvimento de eventos em uma cidade, através da integração cidadã com o compartilhamento de acontecimentos importantes que acontecem nela.

Como objetivos específicos, podemos explorar e introduzir questões que envolvem o dia-a-dia das pessoas na modelagem do sistema, tais como permitir a elas tomar decisões baseadas em informações obtidas através deste sistema de forma rápida, uma vez que os eventos apresentados estariam de acordo com o seu contexto. Além disso, tem-se a possibilidade de utilização do sistema por parte do governo para facilitar a gestão da cidade, com a averiguação de problemas e sugestões dos cidadãos.

1.4 Estrutura do Texto

Este trabalho está estruturado da seguinte maneira: o capítulo 2 apresenta trabalhos relacionados e uma técnica de recuperação de textos utilizada no sistema. O capítulo 3 mostra como foi feita a modelagem para o desenvolvimento do sistema nos seus diferentes aspectos. Já o seguinte, capítulo 4, apresenta as ferramentas utilizadas, bem como detalhes da implementação. Por fim, o capítulo 5 finaliza este trabalho com as considerações finais e trabalhos futuros.

2 PROJETOS PARA CIDADES INTELIGENTES

Começam a surgir ferramentas para permitir a construção de Cidades Inteligentes, dentre estes citam-se: PortoAlegre.cc, Cambridge iReport, Waze, Cyberguide e Guide, analisados neste capítulo.

2.1 PortoAlegre.cc

O projeto PortoAlegre.cc[2] tem por objetivo "oferecer um espaço virtual por meio do qual os moradores da cidade de Porto Alegre possam conhecer, debater, inspirar e transformar a própria cidade" (ARRUDA FREIRE et al., 2012). Ele foi criado com o intuito de trazer fatos relevantes para sociedade através dela mesma. Gerando, assim, uma verdadeira mobilização virtual para tentar mudar o mundo real, de forma a "ouvir a cidade e a partir daí buscar co-criação, co-responsabilização e engajamento cívico" (GERSON; VALIATI, 2012).

Com ele, surgiu o conceito de *wikicidade*: um lugar físico que funciona como um verbete do website *Wikipedia*[3] ao qual qualquer pessoa pode acrescentar informações sobre ele para torná-lo mais rico. Isso acontece, pois este projeto permite a criação e visualização de causas, além de permitir comentários relacionados a cada uma delas. Este projeto pode ser visto como a evolução de um outro, o Redencao.cc[4], cujo objetivo é similar, porém não tão abrangente quanto ele pois é restrito ao Parque Farroupilha, da cidade de Porto Alegre/RS, também conhecido como Parque da Redenção.

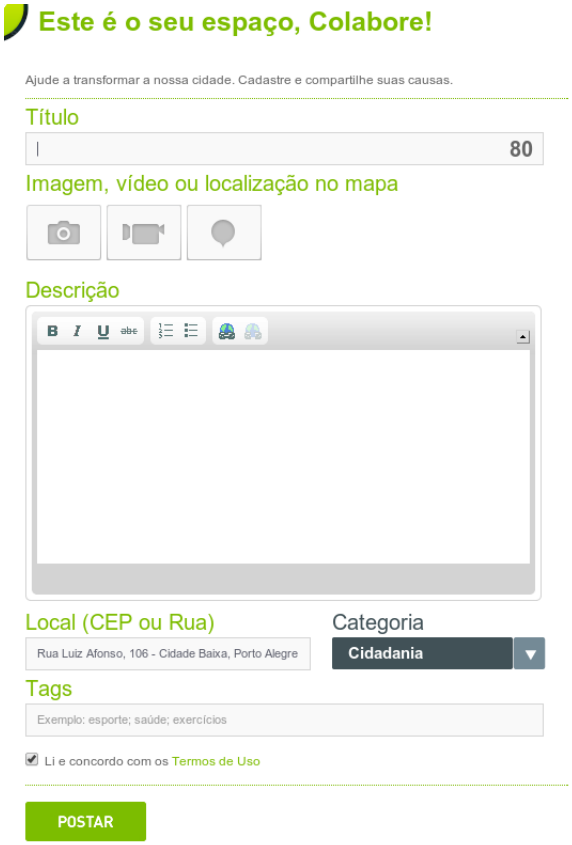
Existem dois modos de participação: como usuário, capaz de acessar e visualizar as informações e; como colaborador, sendo este um gerador de conteúdo. Este conteúdo pode ter diversas formas, tais como fotos, vídeos e textos. Estes dados devem ser autenticados por um moderador, isto para evitar informações irregulares que não se adequam aos Termos de Uso do projeto.

Todas essas causas são apresentadas em um mapa e podem ser facilmente identificadas, pois existem categorias pré-definidas para elas, e cada uma tem sua própria imagem, como pode ser observado na figura 2.1a. Já a figura 2.1b mostra o formulário a ser preenchido para a adição de causas. Podemos perceber no projeto a possibilidade de adicionar fotos e vídeos, além de

[2] Disponível em http://www.portoalegre.cc/. Acesso em: janeiro de 2013.
[3] Disponível em http://pt.wikipedia.org/. Acesso em: janeiro de 2013
[4] Disponível em http://www.redencao.cc. Acesso em: janeiro de 2013.

determinar o local no mapa em que esta causa deve ser registrada.

(a) Mapa com causas cadastradas de diferentes tipos.



(b) Formulário para cadastro de causa.

Figura 2.1 – Na figura 2.1a e figura 2.1b podemos ver um pouco de como é o sistema do PortoAlegre.cc.

O PortoAlegre.cc, atualmente, funciona apenas através de seu website, o que limita sua usabilidade em dispositivos móveis. Porém, segundo os autores, há planos para desenvolver aplicativos nativos para os smartphones mais atuais. Essa etapa faz parte do quarto dos cinco módulos planejados, sendo eles:

1. Módulo 1: sistema em funcionamento;
2. Módulo 2: envolve a colaboração dos usuários através dos conteúdos publicados (sendo esta a fase em andamento);
3. Módulo 3: mensuramento de que tipo de informações estão sendo compartilhadas, através de nuvens com os fatos com maior número de menções;
4. Módulo 4: disponibilização do projeto para os dispositivos móveis;

5. Módulo 5: projeção de cenários futuros para a cidade, a fim de estimular discussões sobre os rumos que estão sendo tomados.

Um dos problemas do PortoAlegre.cc é que, mesmo que se utilize filtros ou busca por palavras-chave, pode ser que o usuário não encontre as informações que, de fato, seriam relevantes para ele naquele momento. Isso pode ser ainda mais problemático após o desenvolvimento do projeto nos dispositivos móveis, uma vez que o contexto da pessoa varia conforme a sua posição.

2.2 Cambridge iReport

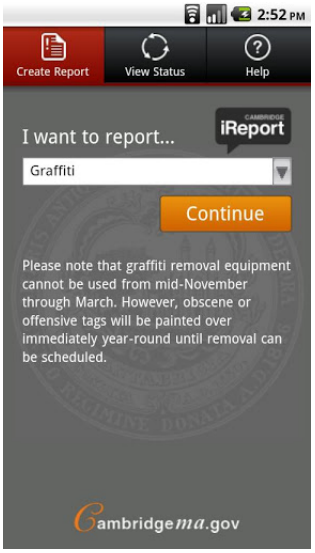
Com o intuito de fazer um uso maior da tecnologia para auxiliar no desenvolvimento e na melhoria da cidade de Cambridge, Massachusetts/EUA, seu governo desenvolveu um aplicativo para smartphones que permite aos seus cidadãos o registro de denúncias.

Para iniciar a denúncia, o usuário deve escolher um tipo de problema dentre os pré-definidos (Figura 2.2a), tais como: buracos nas ruas, semáforos defeituosos, pixações, ratos, calçadas sujas, entre outros. As outras etapas deste processo seguem como mostrado na figura 2.2b e figura 2.2c.

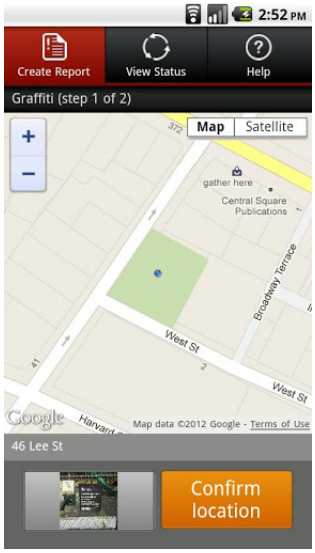
Este aplicativo, apelidado de Cambridge iReport[5], serve como um meio de interação direta entre o Estado e seus cidadãos. A denúncia de problemas na cidade é feita de forma prática, uma vez que as informações são repassadas ao governo de forma mais eficiente do que se fosse uma ligação telefônica. Elas consistem na localização geográfica, foto e uma breve descrição do caso (Figura 2.2b e Figura 2.2c). Também é possível verificar o status atual do problema que foi registrado, para acompanhamento (Figura 2.2d).

As denúncias, como, por exemplo, pixações em placas, são enviadas diretamente para os órgãos responsáveis, os quais averiguam e tentam resolver o problema. O que acaba restringindo a participação social. Se fosse possível a todos os cidadãos visualizar estes problemas e discutir eles, o sistema poderia ser melhor aproveitado, uma vez que cada um poderia sugerir soluções para a denúncia em questão e, assim, gerar uma discussão sobre a melhor maneira de resolvê-la.

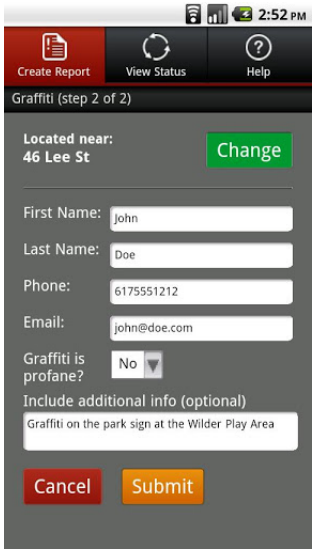
[5] Disponível em: http://www.ci.cambridge.ma.us/iReport. Acesso em: janeiro de 2013.



(a) Escolha do tipo de problema o qual a denúncia consiste.



(b) Localização geográfica e foto do problema sendo reportado.



(c) Informações pessoais do denunciante e adicionais sobre o caso.

(d) Averiguando o status do problema.

Figura 2.2 – Etapas para registro de uma denúncia no Cambridge iReport. Parte da escolha do tipo de problema (Figura 2.2a) para a localização (Figura 2.2b) e informações adicionais e do denunciante (Figura 2.2c). Também pode-se verificar o andamento da solução para o problema (Figura 2.2d).

2.3 Waze

O Waze[6] é um aplicativo para smartphones desenvolvido com o foco nos motoristas e o compartilhamento de informações que envolvem o tráfego nas vias da cidade. Como apresentado, por exemplo, na figura 2.3b em que mostra o registro de que determinada rua está

[6] Disponível em: http://www.waze.com/. Acesso em: janeiro de 2013.

congestionada. Auxiliando, portanto, o usuário para que ele utilize um outro caminho, tendo em vista que aquela via está problemática.

O aplicativo nos permite adicionar alguns tipos de alertas diferentes. Como visto na figura 2.3a, é possível informar sobre acidentes, perigos na via, radares e preços da gasolina em diferentes postos.

(a) Possíveis alertas que podem ser registrados.

(b) Informação compartilhada sobre um congestionamento.

(c) Pode-se marcar reuniões com amigos.

Figura 2.3 – O Waze permite o registro de alertas (Figura 2.3a) como, por exemplo, congestionamentos (Figura 2.3b) e, também, marcar reuniões com amigos (Figura 2.3c).

Tendo em vista que se trata de um sistema que depende da comunidade para manter seu funcionamento e sua relevância, algumas questões sociais foram adicionadas com o intuito de estimular a interação entre seus usuários para mantê-los ativos. É possível marcar com seus amigos um ponto de encontro, definir um percurso e até dar carona (Figura 2.3c). Todas essas ações funcionam compartilhando informações de localização em tempo real e percursos entre os envolvidos para que todos possam saber onde cada um se encontra. Também é possível conversar em um chat com qualquer usuário ativo.

Pelo fato de o foco deste aplicativo estar limitado ao trânsito e às informações em tempo real, perde-se a capacidade de criar discussões acerca de um assunto e, assim, possíveis soluções ou ideias para melhorias na cidade, tanto como um todo, quanto apenas do seu tráfego, não são desenvolvidas ou são perdidas rapidamente.

2.4 Cyberguide

Cyberguide e Guide foram projetos que influenciaram bastante o desenvolvimento dos atuais aplicativos para cidades inteligentes, por isso estão destacados aqui.

No final do ano de 1996, Abowd (ABOWD et al., 1997) argumentava que os usuários, no futuro, se veriam livres das limitações dos desktops, voltando-se para tecnologias de computação móvel. Sendo que, essas aplicações para dispositivos móveis deveriam fazer uso do contexto atual do usuário para oferecer melhores serviços a eles. Dessa maneira, desenvolveram protótipos (para dispositivos como PDAs e Tablets-PC) que, a partir dessas informações de contexto (tanto a posição atual, quanto lugares já visitados), funcionavam como guias turísticos capazes de trazer informações relevantes sobre o local em que o usuário se encontrava.

A fim de realizar a tarefa destinada, Abowd et al (ABOWD et al., 1997) definiu uma arquitetura a qual serviu de base para o desenvolvimento dos protótipos. Esta arquitetura é composta de quatro componentes, os quais foram definidos como os papéis que um guia turístico desenvolve, sendo eles:

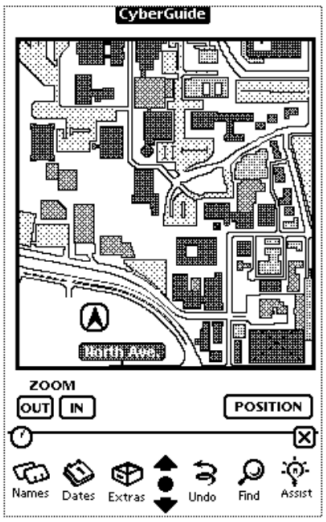
1. *Componente do Mapa*: Um guia turístico deve saber os pontos interessantes de um determinado lugar e como chegar até eles por qualquer caminho. No sistema do Cyberguide, essas informações são providas através de um mapa (Figura 2.4a e Figura 2.4b).

2. *Componente da Informação*: As informações relativas a cada elemento do passeio turístico devem ser disponibilizadas ao turista para que este possa sanar suas dúvidas e vir a conhecer novos fatos (Figura 2.4c). Para adquirir esses dados, o Cyberguide utiliza um repositório com essas informações. Porém, são informações que não podem ser expandidas pelos usuários. Uma vez que não é possível a eles cadastrar comentários e atribuí-los a um ponto de interesse. Isso acaba por limitar a experiência social, uma vez que tais comentários podem conter informações interessantes para as diferentes pessoas, enriquecendo-as com conhecimentos que vão além dos dados básicos encontrados no repositório.

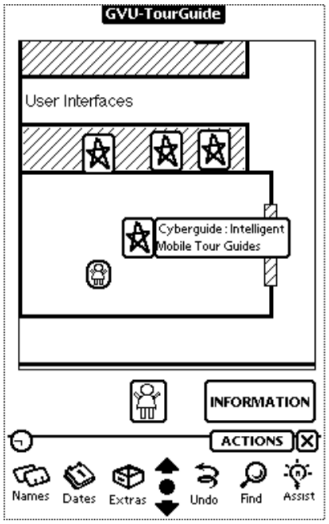
3. *Componente de Posição*: A localização do turista geralmente está próxima dos objetos de seu interesse, por isso é importante saber onde ele se encontra para poder providenciar os dados sobre o que está ao seu redor. Para a posição do usuário em um lugar fechado, como GPS não seria preciso o suficiente (e, talvez, até não fosse possível obter sinal),

utilizaram um sistema com diversos pontos instalados pelo local. Eles disparavam sinais em infravermelho que eram recebidos pelo dispositivo do usuário e, assim, sua posição era definida, pois cada ponto tinha um identificador único.

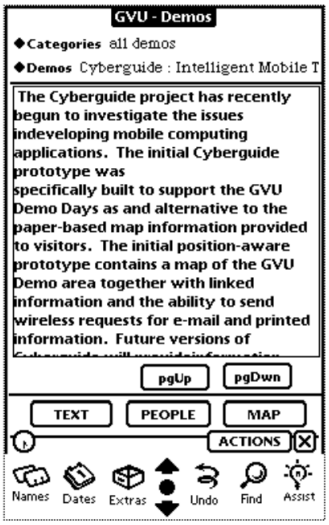
4. *Componente de Comunicação*: A comunicação com outras pessoas pode, também, ser uma necessidade. Por isso, este componente foi acrescentado. Para que mensagens possam ser enviadas e recebidas entre as pessoas que estiverem por perto ou, inclusive, para responder questionários (Figura 2.4d).



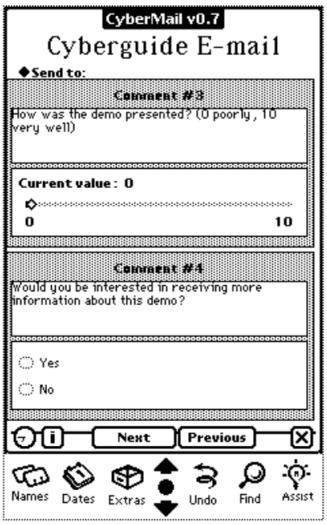
(a) Componente de Mapa em um lugar aberto. A posição do usuário, nesse caso, é obtida através de um GPS.



(b) Componente de Mapa em um lugar fechado.



(c) Componente de Informação.



(d) Componente de Comunicação.

Figura 2.4 – Os componentes que estruturam o sistema do Cyberguide são o mapa (Figura 2.4a e Figura 2.4b), dados sobre os pontos de interesse (Figura 2.4c) e comunicação (Figura 2.4d)

2.5 GUIDE

No ano de 2000, o GUIDE (CHEVERST et al., 2000), assim como o Cyberguide (Subseção 2.4), foi um sistema desenvolvido com o intuito de ser utilizado como um guia turístico personalizado aos interesses do usuário. Percebeu-se que a maneira tradicional de se fazer turismo, ou seja, com um grupo de turistas e um único guia dando informações que satisfazem apenas a maioria dos visitantes, poderia ser aperfeiçoada. Uma vez que cada turista tem seus próprios interesses, viu-se a necessidade de criar um sistema sensível ao contexto capaz de oferecer capacidades desejadas e personalizadas ao se fazer um tour por uma cidade. No caso do GUIDE, a cidade em que se desenvolveu este projeto foi Lancaster, Lancashire/Reino Unido.

Alguns requisitos foram levantados para este sistema, dentre outros:

- *Flexibilidade*: O sistema deve ser capaz de permitir ao visitante explorar a cidade à sua propria maneira, permitindo a ele a alteração de forma dinâmica do seu tour. Por exemplo, a qualquer momento o usuário pode decidir descansar e ir a um restaurante ou ele pode decidir ficar mais tempo do que o esperado em um lugar ou sair mais cedo de outro;

- *Informação sensível ao contexto*: As informações apresentadas devem estar de acordo com o contexto do usuário. Foram identificadas duas classes de contexto:

  - *Contexto pessoal*: como, por exemplo, localização e preferências do usuário e;
  - *Contexto de ambiente*: tais como, hora do dia e hora de abertura de atrações.

  A localização do visitante não é obtida através de um GPS, mas sim por um sistema que faz uso de estações-base posicionadas estrategicamente, responsáveis por transmitir mensagens aos dispositivos móveis que, assim, descobrem sua posição.

O dispositivo móvel escolhido para utilização do sistema foi um tablet baseado em caneta, como pode ser visto na figura 2.5a. E sua interface gráfica foi desenvolvida baseada na forma de web browser. Como pode ser visto na figura 2.5b, temos algumas ações possíveis, tais como:

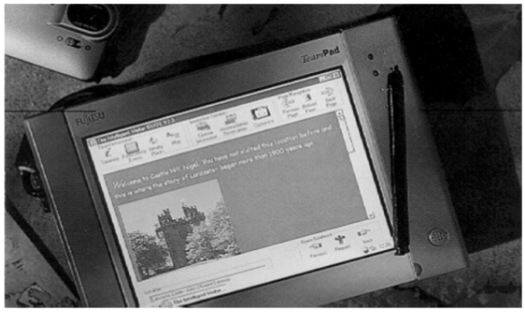
1. Visualização de informações: ao clicar no botão de informações, o usuário é levado para uma página em que ele deve escolher que tipo de informação deseja. A figura 2.5d nos mostras as opções disponíveis. Os dois primeiros hyperlinks retornam dados relativos ao contexto do usuário, ou seja, sua posição, horário, etc. Sendo suas funções apresentar

conteúdo a respeito da área em que o usuário se encontra e responder questões acerca de algo que ele consegue ver;

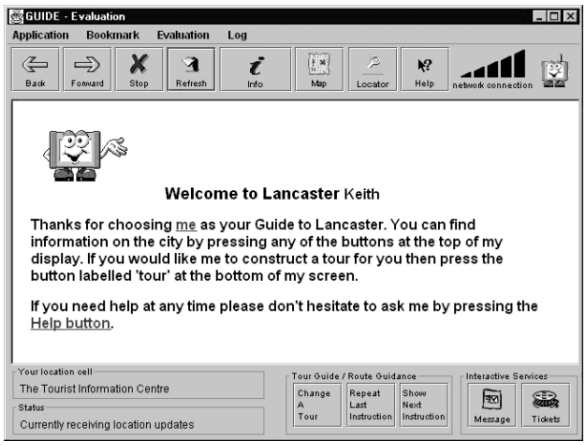
2. Navegar pela cidade através de um mapa;

3. Criar e seguir um tour pela cidade (Figura 2.5c);

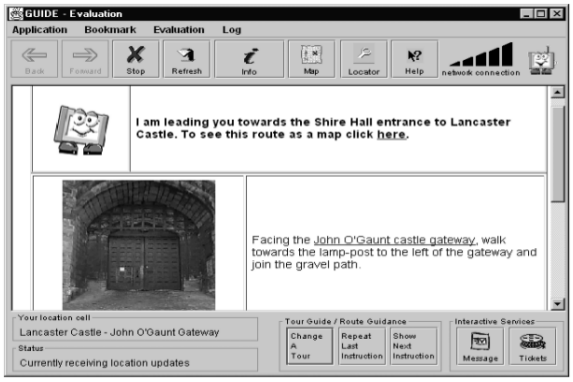
4. Comunicar-se com o centro de informação ao turista ou com outros visitantes. Essa comunicação entre os turistas é a única maneira de expressar os sentimentos do usuário sobre determinados assuntos. Não é possível a ele criar algum tipo de evento que ficará disponível para outros verem e comentarem no futuro.



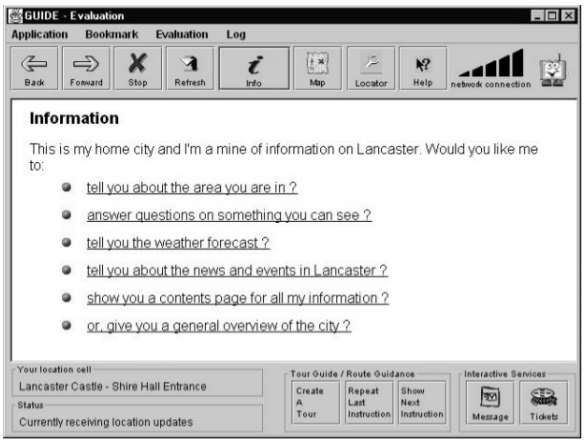
(a) Dispositivo móvel utilizado para desenvolvimento.



(b) Interface gráfica baseada em Web Browsers.



(c) Apresentação da informação de um tour criado.



(d) Página para seleção do tipo de informação.

Figura 2.5 – No GUIDE, utilizou-se de um tablet baseado em caneta (Figura 2.5a) e uma interface gráfica baseada em web browsers (Figura 2.5b). Ao fazer um tour (Figura 2.5c), o usuário vai sendo navegado pelo ambiente e é permitido a ele buscar informações (Figura 2.5d) sobre o que está ao seu redor.

3 MODELAGEM

É apresentado neste capítulo os objetivos gerais e requisitos deste sistema, bem como a modelagem das soluções encontradas para o seu desenvolvimento. De modo que se tenham motivações específicas para determinadas metas, foram exploradas, juntamente delas, justificativas para a sua abordagem. Contribuindo, assim, para um entendimento mais preciso do que se busca.

Em primeiro lugar, são expostos os objetivos e requisitos pertinentes. Em seguida, é abordada, através da modelagem de diagramas e estruturas, a maneira com a qual o sistema foi planejado.

3.1 Objetivos e Requisitos

Este trabalho busca, como objetivo geral, ser uma ferramenta de integração cidadã, permetindo a todos ajudar no desenvolvimento e descobrimento de sua cidade, exercendo, assim, a sua cidadania.

A maneira com a qual cada um pode auxiliar neste desenvolvimento é através do compartilhamento de informações de eventos que tenham ocorrido, ou estejam ocorrendo, pela sua cidade. Para permitir que essa troca de informações seja feita de maneira prática, o cidadão deve ter posse de um telefone celular inteligente, chamado de *smartphone*, que seja capaz de acessar a internet e descobrir sua localização no globo, juntamente com um aplicativo responsável por permitir ações relacionadas ao compartilhamento de informações de eventos. Assim, por ser um dispositivo móvel, o smartphone traz a capacidade, ao sistema, de garantir um meio de visualizar e adicionar eventos nos momentos mais pertinentes a isso, ou seja, o cidadão recebe os eventos na hora em que precisa e compartilha um novo evento no momento em que o presencia.

Dessa forma, o aplicativo para o smartphone deve permitir a qualquer um de seus usuários:

1. Visualizar eventos compartilhados que, se for sua escolha, estejam de acordo com o contexto em que se encontra. Isso é feito, pois o número de eventos cadastrados na cidade pode ser muito elevado e, desta forma, a maioria destes, possivelmente, não seriam de seu interesse;

2. Registrar eventos na posição em que se encontra, evidenciando informações pertinentes e

atribuindo-as a ele.

Como se trata de um sistema para a integração cidadã, não basta que seja possível aos usuários praticar apenas ações de visualização e adição de eventos. É necessária, também, a comunicação (de forma não-anônima) entre os mesmos, para:

- Garantir que a qualidade das informações de um evento esteja em um nível apropriado e não tenha dados equivocados. Além de dar maior destaque aos eventos que tenham atraído maior interesse (através de votos positivos ou negativos);
- Expandir o conteúdo de um evento. Uma vez que cada cidadão tem suas próprias experiências sobre os assuntos sendo relatados, é interessante que se possa compartilhá-las para que todos tenham um melhor entendimento do que está acontecendo e;
- Gerar discussões sobre os eventos. Permitindo, assim, que propostas de inovações para a cidade, ou soluções de problemas, sejam elaboradas em conjunto.

Estas nuances estão diretamente relacionadas aos objetivos deste trabalho. Uma vez que, através da integração cidadã, elas auxiliam no descobrimento (com a expansão do conteúdo dos eventos) e desenvolvimento (com as discussões) da cidade.

3.2 Modelagem

De modo a nortear o desenvolvimento deste sistema, foram feitas modelagens básicas de funcionalidades e do funcionamento de seus diferentes aspectos.

Um caso de uso foi esquematizado (Figura 3.1) para apresentar o conjunto de ações possíveis de um cidadão dentro do InfoCity. Com ele podemos ver de forma clara as atividades, citadas na seção anterior (Seção 3.1), que o usuário é capaz de realizar.

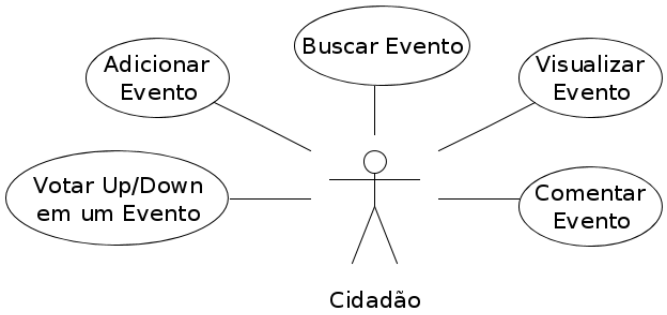


Figura 3.1 – Caso de uso das ações possíveis ao cidadão.

Vale notar que qualquer cidadão pode tanto adicionar quanto buscar por eventos de forma anônima. Somente é requerida identificação quando há algum tipo de interação com outros usuários ou com o evento. O impedimento de comentários e votos anônimos se dá devido ao fato de que, caso contrário, poderia haver um abuso do sistema. Tanto por postagens de comentários irrelevantes e ofensivos, quanto pela possibilidade de que qualquer um poderia votar mais de uma vez num mesmo evento, devido à falta de controle.

Para entender de forma mais detalhada como se pensou que seria o processo de busca (Figura 3.2) e adição (Figura 3.3) de eventos, criou-se modelos das etapas de cada um.

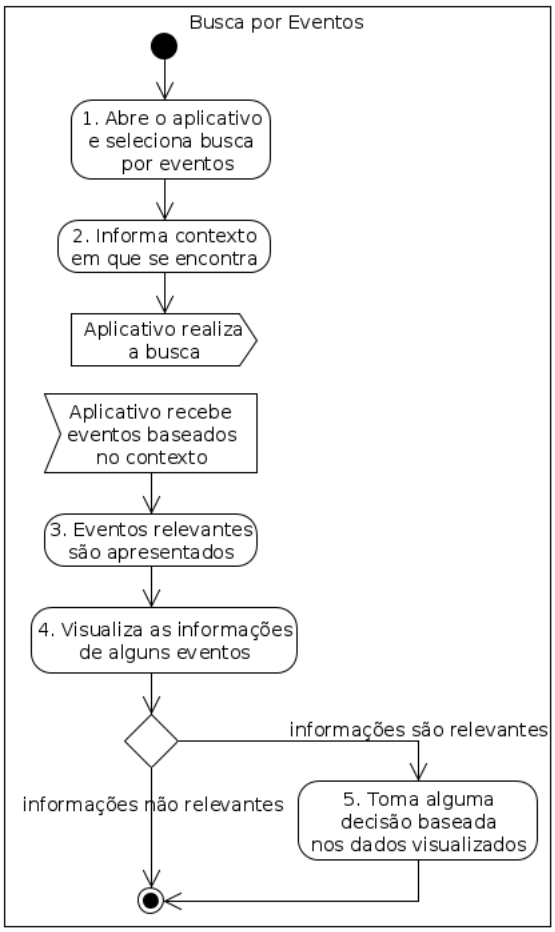


Figura 3.2 – Diagrama de atividades de como deve ser a busca por eventos.

Exemplificando o funcionamento do processo de busca de eventos (Figura 3.2), podemos imaginar a seguinte situação: *"Ao passear por uma praça de sua cidade, Maria decide verificar o que as pessoas estão comentando sobre o lugar. Para isso, abre o aplicativo do* InfoCity *no seu smartphone e seleciona a busca por eventos* (etapa 1). *Após o aplicativo identificar o contexto em que ela se encontra, através de diferentes ferramentas,* (etapa 2) *ele a apresenta com eventos relevantes* (etapa 3). *Como estes eventos são exibidos em suas localizações exatas*

sobre um mapa, ela consegue identificar quais eventos estão relacionados à praça em que se encontra. Assim, ela verifica alguns desses eventos e descobre que esta praça é perigosa e está mal iluminada. Sabendo disso, ela visualiza um evento cadastrado em uma outra praça perto dali e averigua que aquele local é bem iluminado e ótimo para passear com a família. Pelo evento ter bastante votos positivos e comentários (etapa 4)*, Maria decide deslocar-se para lá, pois prefere um local seguro para o seu lazer* (etapa 5)*".*

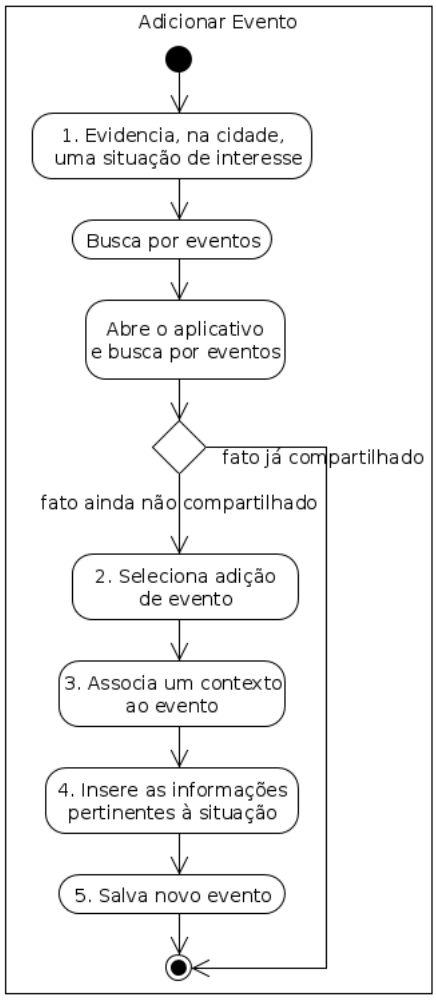


Figura 3.3 – Diagrama de atividades de como deve ser a adição de um evento.

Utilizando a mesma ideia do exemplo recém citado, a adição de um evento (Figura 3.3) poderia acontecer da seguinte forma: *"Ao andar pela nova praça, Maria verifica que há poucos bancos disponíveis para descanso* (etapa 1)*. Indignada com o fato, ela deseja compartilhar essa informação com outros cidadãos. Assim, abre o aplicativo do* InfoCity *no seu smartphone e decide adicionar um evento* (etapa 2)*. O aplicativo, através de diferentes ferramentas, descobre o contexto em que ela se encontra* (etapa 3) *e o associa ao evento sendo criado. Ela, então, adiciona os dados ao evento, falando do problema que encontrou* (etapa 4) *e o disponibiliza a*

todos (etapa 5)*"*.

3.2.1 Servidor

Por se tratar de um sistema que necessita o armazenamento de dados e que os mesmos possam ser acessados por todos usuários, percebeu-se a necessidade do desenvolvimento de um servidor central que pudesse ser acessado via *web* e disponibilizasse de um banco de dados para o registro das informações.

Para isso, foi desenvolvida uma estrutura para o servidor que pudesse comportar este sistema (Figura 3.4). Ela é composta, basicamente, de métodos (formando, assim, uma Interface de Programação de Aplicação, API, em inglês) responsáveis por tarefas como adicionar novos eventos ao sistema, buscar eventos que estejam dentro de um raio, buscar comentários, salvar comentário, dentre outras. Como pode-se notar, as requisições, enviadas pelo aplicativo instalado em um smartphone, são feitas para executar tais métodos. E, assim, manipular o banco de dados para tanto receber, quanto registrar informações.

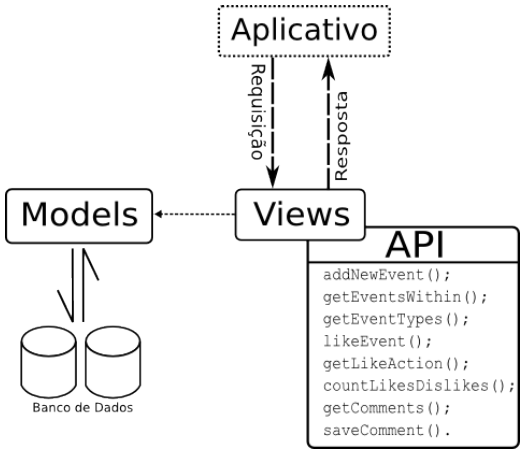


Figura 3.4 – Estrutura global do servidor.

O banco de dados foi modelado (Figura 3.5) de forma a garantir que os objetivos e requisitos relacionados ao armazenamento de informações referidos na seção 3.1 pudessem ser atendidos.

Desta forma, chegou-se à seguinte estrutura de tabelas:

- *EventType*: Tabela que armazena todos os possíveis tipos de eventos pré-definidos, sendo que cada evento pode (e deve) ter apenas um único tipo;
- *EventKeyword*: Tabela com todas as palavras-chave registradas pelos usuários. Cada evento pode ter, no banco de dados, qualquer número delas;

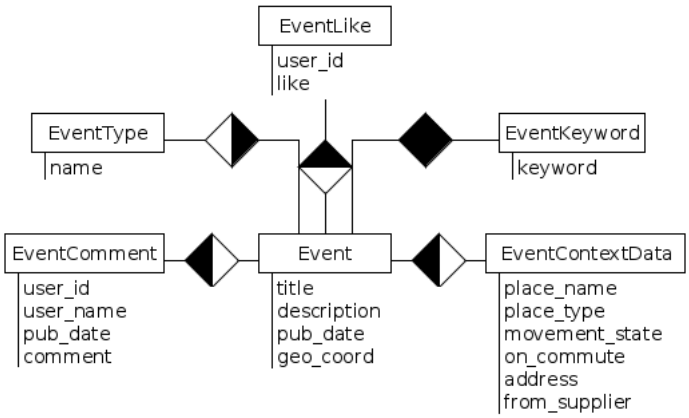


Figura 3.5 – Modelo do banco de dados que deve ser armazenado no servidor.

- *EventLike*: Cada usuário único tem a possibilidade de votar em um evento. Este voto pode ser tanto positivo quanto negativo e cada evento pode ter qualquer número de votos;
- *EventComment*: Os eventos podem ter diversos comentários, porém um comentário deve estar associado a um único evento. Eles são registrados com:
  1. user_id: Identificação única de usuário. Para saber exatamente quem realizou o comentário;
  2. user_name, pub_date e comment: Informações a serem apresentadas quando o comentário for ser visualizado.
- *EventContextData*: Armazena as informações relacionadas ao contexto em que determinado evento foi registrado. É composta por:
  1. place_name: Nome do lugar em que o evento aconteceu (e.g. Santa Maria Shopping);
  2. place_type: Tipo do local (e.g. shopping);
  3. movement_state: Estado da movimentação de quando o usuário registrou o evento (e.g. caminhando);
  4. on_commute: Se um usuário estava indo de um lugar a outro;
  5. address: Endereço do local (e.g. Av. Presidente Vargas) e;
  6. from_supplier: O fornecedor de onde as informações de contexto vieram.
- *Event*: Tabela central para o registro de eventos. Cada um deles, além das informações relacionadas indicadas pelas conexões com as outras tabelas (como pode ser visto na

figura 3.5), tem informações básicas como título, descrição, data de publicação e coordenada geográfica para posterior visualização. É interessante notar que o campo da coordenada geográfica é importante, pois ele determina a localização (na forma de latitude e longitude) em que o evento foi registrado. Com essa informação, é possível saber quais os eventos que estão mais perto do usuário para, assim, quando uma requisição ao servidor for feita, retornar apenas estes, diminuindo o tráfego com a remoção de eventos que seriam, a princípio, de pouco uso ao cidadão.

Pode-se observar que a modelagem do banco de dados apresentado na figura 3.5 respeita os objetivos e requisitos propostos na seção 3.1, uma vez que ele não faz restrição nenhuma com relação a quem pode visualizar ou adicionar eventos. Essa limitação existe, através do campo user_id, apenas com a adição de novos registros nas tabelas que armazenam os votos e comentários. Como usuários anônimos não dispõem de identificadores únicos, se torna impossível, a eles, adicionar novas instâncias nas tabelas *EventComment* e *EventLike* que, como já mencionado, listam os comentários e votos de cada evento, respectivamente.

3.2.2 Aplicativo para Android

O desenvolvimento do aplicativo para smartphones baseados no sistema *Android* foi fundamentado a partir do diagrama de classes da figura 3.6. Sendo este modelado com base em UML, *Unified Modeling Language* (BOOCH; RUMBAUGH; JACOBSON, 1999).

Assim, podemos verificar a existência de quatro pacotes principais:

1. *Model*: Composto de classes que representam as estruturas básicas do sistema, tendo relação direta com as tabelas, e suas colunas, no banco de dados (apresentado na subseção 3.2.1);

2. *Context*: A fim de gerenciar o contexto em que o usuário se encontra, este pacote é composto pelas classes:

   - *ContextData*: representação dos dados de contexto;
   - *ContextSupplier*: interface que deve ser implementada por qualquer provedor de contexto incorporado pela aplicação e;
   - *Provedores de Contexto*: classes que implementam a interface ContextSupplier. Elas são responsáveis por preencher os dados de contexto (representados pela classe

ContextData), utilizando seus próprios métodos, para que eles possam ser empregados onde forem cabíveis.

3. *Internet*: Seu principal objetivo é o de servir como interface entre o servidor do sistema e a aplicação. A classe InfoCityServer mapeia todos os métodos possíveis de serem chamados no lado do servidor (apresentados na figura 5ffig:serverstructure). Assim, qualquer ação, que necessite informações armazenadas no servidor, é feita através destes métodos e;

4. *Map*: A classe InfoCityMap funciona como a interface principal entre o aplicativo e o usuário. É através dela que as interações com o mapa são feitas, tais como buscar ou adicionar eventos. E é ela, também, quem apresenta tais eventos. Permitindo ao usuário a visualização das informações de cada um.

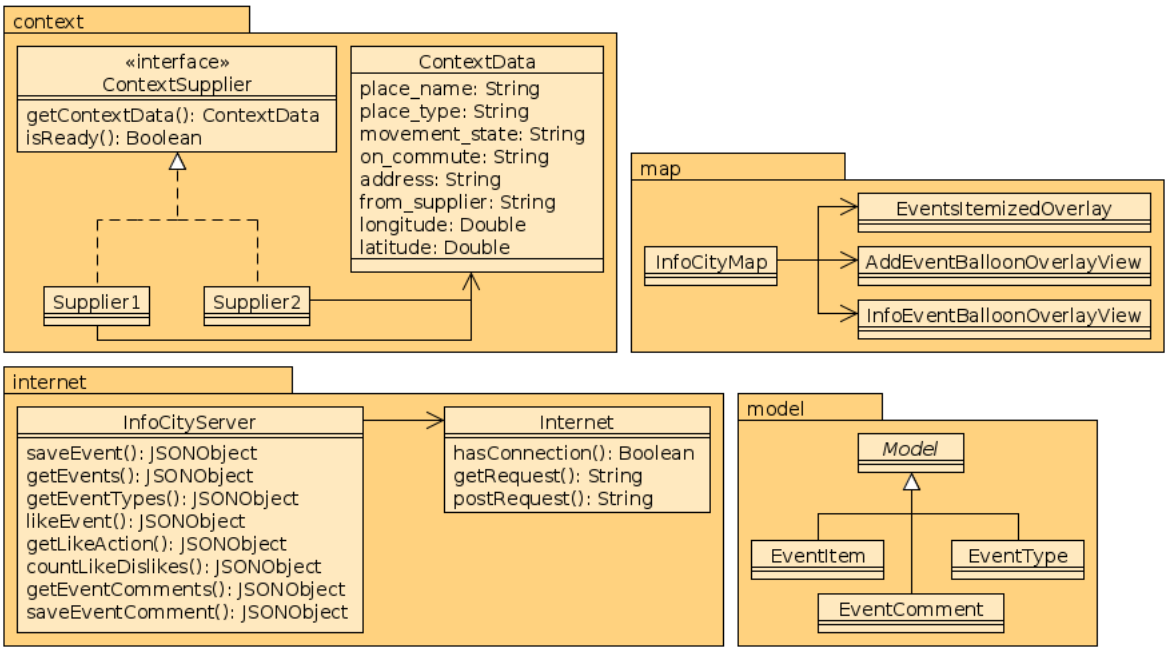


Figura 3.6 – Diagrama de classes base para o desenvolvimento do aplicativo do InfoCity no sistema Android.

Para exemplificar como seria a utilização destas classes, o diagrama de sequência na figura 3.7 nos mostra o processo pelo qual a atualização de eventos deve ocorrer. Este fluxo consiste nas seguintes etapas:

1. Após iniciar o aplicativo, o usuário decide que quer visualizar os eventos relevantes a ele naquele momento. Para isso, ele seleciona a opção de buscar por eventos, cuja *callback* está implementada na classe InfoCityMap;

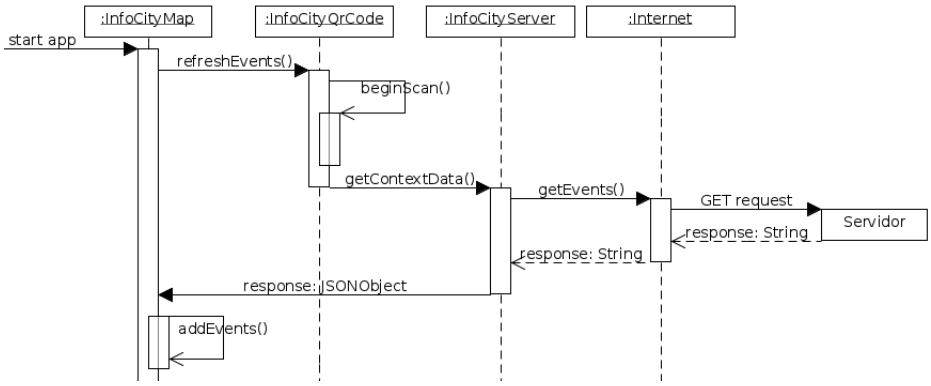


Figura 3.7 – Diagrama de sequência do fluxo de como a atualização de eventos deve ser feita.

2. A relevância dos eventos que devem ser retornados é dada pelo contexto em que ele se encontra. Então, após decidir que seu contexto será adquirido através da leitura de um Qr-Code, ele aponta sua câmera para o código de barras e o aplicativo faz o seu reconhecimento através da classe InfoCityQrCode, que implementa a interface ContextSupplier;

3. Estes dados de contexto são, então, passados à classe InfoCityServer que os repassa, na forma de String, à classe Internet, a qual formata os dados e faz a requisição do tipo GET;

4. Por fim, a resposta do servidor é analisada para verificar se está no formato correto (isto é feito pela classe InfoCityServer). Uma vez estando no formato correto, a própria classe InfoCityServer transforma ela em um objeto do tipo JSONObject[7] e o repassa para a classe InfoCityMap. Nela, ele é transformado em objetos do tipo EventItem para serem adicionados ao mapa e, finalmente, apresentados ao usuário.

[7] Detalhes podem ser vistos em: http://www.json.org/javadoc/org/json/JSONObject.html. Acesso em: janeiro de 2013.

4 IMPLEMENTAÇÃO

Podemos dividir o sistema InfoCity em duas partes:

1. Aplicativo para smartphones baseados em Android e;
2. Servidor responsável pelo armazenamento e manipulação dos dados.

Com isso, a implementação deste sistema se deu em duas etapas, uma para cada parte. Nas próximas seções são apresentadas as ferramentas utilizadas para o desenvolvimento dessas etapas (Seção 4.1), a maneira com a qual o servidor se comunica com o aplicativo para Android (Seção 4.2), como é que o servidor determina quais são os eventos relevantes ao usuário (Seção 4.3), uma demonstração com telas de como, de fato, o aplicativo para Android é utilizado (Seção 4.4) e, por fim, alguns testes para a avaliação do sistema (Seção 4.5).

4.1 Ferramentas

Ao longo de todo o desenvolvimento, manteve-se as estruturas modeladas (Capítulo 3) em mente para que todos os objetivos deste trabalho pudessem ser alcançados. Ele foi feito com o auxílio de diferentes ferramentas para questões de contexto, gerenciamento de código e do sistema em si.

Buscando-se uma produção segura dos códigos fonte, utilizou-se o Sistema de Controle de Versão (VCS, em inglês) *Git*[8] juntamente com o website *GitHub*[9] para o armazenamento dos repositórios [10] [11]. Segundo (CHACON; HAMANO; PEARCE, 2009), VCSs podem ser definidos como sistemas capazes de gravar toda e qualquer alteração em algum arquivo, ou conjunto de arquivos, no decorrer do tempo para que, assim, seja possível retornar a uma versão mais antiga, caso se faça necessário. Utilizar um VCS, geralmente, significa que "se você estragar as coisas ou perder arquivos, pode facilmente recuperar-se." (CHACON; HAMANO; PEARCE, 2009). Ou seja, obtém-se grande controle sobre os arquivos de um projeto que se encontram como parte de um repositório sendo gerenciado por um VCS.

[8] Disponível em: http://git-scm.com. Acesso em: janeiro de 2013.
[9] Disponível em: https://github.com/. Acesso em: janeiro de 2013.
[10] Repositório do aplicativo disponível em: https://github.com/brunodea/infocity. Acesso em: janeiro de 2013.
[11] Repositório do servidor disponível em: https://github.com/brunodea/infocity_server. Acesso em: janeiro de 2013.

As próxima subseções abordam como é o processo de desenvolvimento de um aplicativo para o sistema Android e como foi o desenvolvimento do servidor utilizando o framework para web Django, respectivamente. Por fim, é apresentada uma técnica para recuperação de textos utilizada pelo servidor para determinar a relevância de eventos.

4.1.1 Desenvolvimento de Aplicativos para Android

Os aplicativos para Android são codificados na linguagem de programação Java. Para facilitar seus desenvolvimentos, foi criado um *plugin*[12] para o Ambiente de Desenvolvimento Integrado (IDE, em inglês) *Eclipse*[13], o qual foi utilizado neste trabalho. Na figura 4.1 podemos visualizar como é este ambiente.

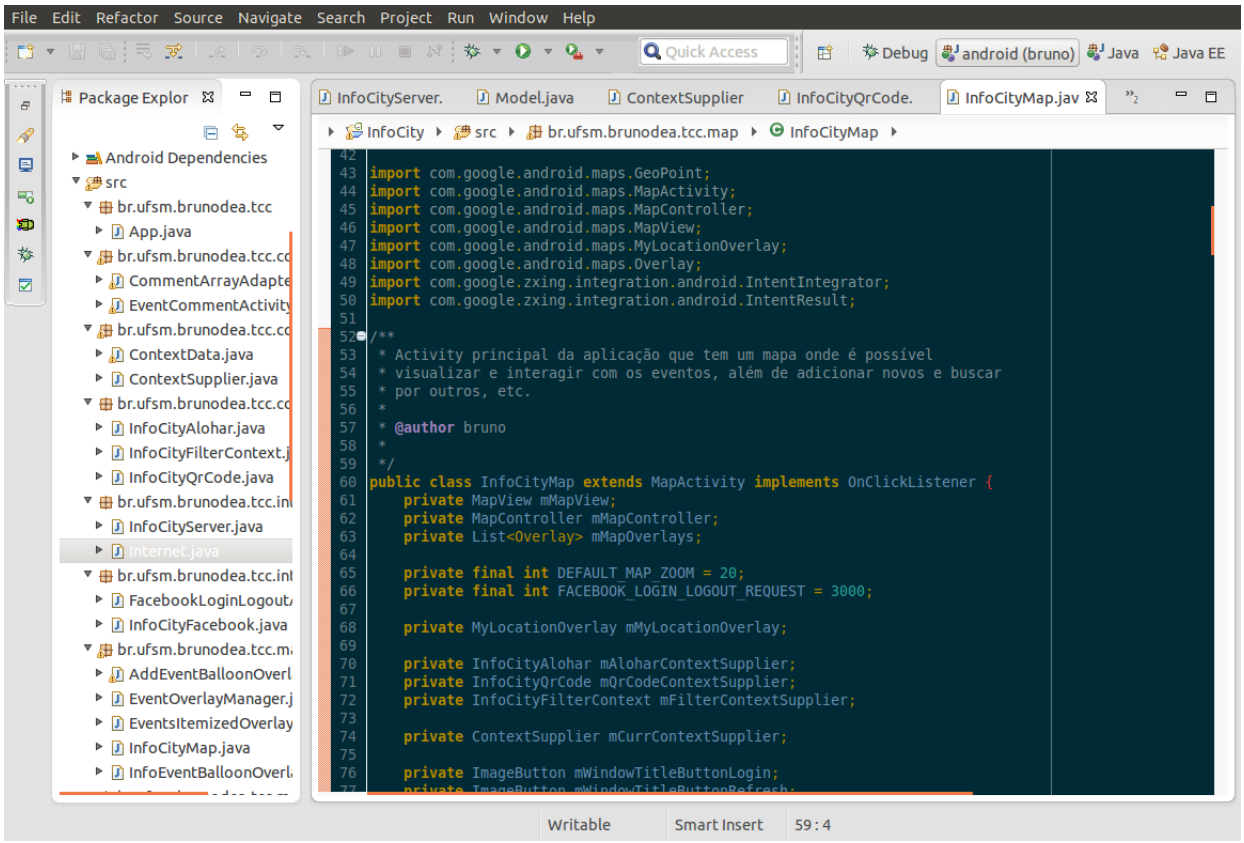


Figura 4.1 – Ambiente de desenvolvimento Eclipse.

Como apontado por (DEVELOPERS, 2011), o Sistema Operacional (OS, em inglês) Android foi construído em cima do kernel do Linux. Sendo que ele funciona na forma de pilha,

[12] Disponível em: http://developer.android.com/tools/sdk/eclipse-adt.html. Acesso em: janeiro de 2013.
[13] Disponível em: http://www.eclipse.org. Acesso em: janeiro de 2013.

de modo que as aplicações fiquem em seu topo e os módulos do kernel do Linux fiquem na parte inferior (A figura 4.2 nos dá uma ideia de como é estruturada esta pilha).

Figura 4.2 – Estrutura em pilha do Android OS.

Quando está sendo desenvolvido um aplicativo para esta plataforma, é possível trabalhar com diferentes componentes. Alguns deles, como apresentado em (MURPHY, 2009), são:

- *Activity*: Funciona, essencialmente, como a interface gráfica com a qual o usuário vai interagir;
- *Intent*: Sistema de mensagens rodando dentro do dispositivo. Pode-se tanto interceptar diferentes Intents que estão funcionando (e.g. mensagem SMS recebida), quanto criar seus próprios. Por sinal, é através de um Intent que se faz a troca de Activities;
- *Service*: Services são códigos que podem ser executados independentemente da aplicação. Ou seja, mesmo que ela seja fechada, um Service pode ficar rodando em *background*. Isto é útil, por exemplo, quando se está checando por algum tipo de atualização (e.g. novas entradas em um banco de dados utilizado pelo aplicativo).

Ainda nos é disponibilizado diferentes ferramentas para tornar nossos aplicativos mais úteis, tais como a possibilidade de armazenar dados internamente no dispositivo (ou no cartão SD), fazer requisições através da internet, acessar a câmera e a localização via GPS, dentre outros.

4.1.2 Django Web Framework

Existem diversas maneiras de se definir o que é um *framework*. É mostrado por (PEREIRA et al., 2009) que frameworks são ferramentas cujos componentes são dedicados a resolver problemas de forma flexível e extensível, e que também podem ser tratados como pequenas aplicações completas que resolvem problemas restritos. Se tratando de frameworks voltados para web, (HOLOVATY; KAPLAN-MOSS, 2009) entende que eles devem ser capazes de compartilhar funções entre páginas, abstrair fluxos que atrasam o desenvolvimento (e.g. fechar uma conexão com o banco de dados), utilizar o mesmo código nos diferentes ambientes e separar o sistema na forma de *Model-View-Controller* (MVC), ou seja, "separar o código da definição e acesso aos dados (o modelo) da lógica de requisições de rotinas (o controlador), o qual é separado da interface do usuário (a visualização)." (HOLOVATY; KAPLAN-MOSS, 2009).

Pelo fato de que, no presente trabalho, o acesso às informações no servidor são feitas através de um aplicativo para smartphones baseados em Android, não foi necessária a preocupação com a parte visual no servidor. Com isso, o desenvolvimento dele utilizou-se apenas da parte de requisições e acesso aos dados do framework Django.

Assim, definiu-se que banco de dados seria utilizado (no caso, o *PostgreSQL*[14]), bem como quais tabelas, e suas respectivas colunas, seriam criadas. Por se tratar de um sistema que faz uso da tecnologia de *Mapeamento de Objeto Relacional* (ORM, em inglês), o Django permitiu que estas tabelas e colunas fossem produzidas a partir dos modelos (classes) especificados. Esta tecnologia funciona de forma a transformar o esquema de objetos (tais como, classes, diagramas, etc) em esquemas no banco de dados (O'NEIL, 2008).

Estas classes, no Django, são especificadas em um arquivo chamado models.py. Para o sistema InfoCity, elas foram baseadas no modelo especificado na figura 3.5, ou seja, elas são: EventType, EventKeyword, Event, EventContextData, EventLike e EventComment. Após a elaboração dessas classes e seus atributos, foi necessário rodar o comando da figura 4.3. Ele faz a ação de, de fato, transformar as classes e atributos dos modelos criados em suas respectivas tabelas e colunas.

```
$ python manage.py syncdb
```

Figura 4.3 – Linha de comando para sincronizar os modelos do Django com as tabelas no banco de dados.

[14] Disponível em: http://www.postgresql.org/. Acesso em: janeiro de 2013.

Utilizando um módulo adicional ao Django, o GeoDjango[15], pode-se adicionar funcionalidades espaciais ao sistema. Segundo (GÜTING, 1994), não existe uma definição genérica do que é um banco de dados espacial, porém ele afirma que banco de dados espaciais oferecem tipos de dados espaciais e permitem a sua utilização nas suas implementações, ou seja, devem prover, ao menos, indexação espacial e algorítmos eficientes para uma junção espacial.

O GeoDjango foi utilizado neste trabalho devido ao fato de que os eventos são registrados com um ponto georreferencial (latitude e longitude). Com isso, as *queries* feitas ao banco de dados, que tenham cunho espacial, são facilitadas pelo sistema. Um exemplo é quando se deseja buscar eventos que estejam dentro de um raio. Como o próprio banco de dados espacial tem funções que fazem este tipo de busca, o cálculo de distância de um ponto até um evento é feito automaticamente. No Django isto é feito na forma apresentada na figura 4.4.

```
Event.objects.filter(geo_coord__distance_lt=(point,radius))
```

Figura 4.4 – *Query*, em python com o Django e GeoDjango, para a busca de eventos no banco de dados que estejam dentro de um raio.

Onde point é um tipo de dado espacial chamado POINT, ou seja, tem apenas uma latitude e longitude; radius é a distância máxima que um evento deve estar de point; geo_coord é a coluna espacial na tabela de eventos e; distance_lt é a função que verifica se a distância dos eventos, através da coluna geo_coord, até o ponto point, é menor que radius. Este método retorna uma lista com todos os eventos dentro do raio.

4.1.3 Recuperação de Texto

Podemos dizer que a recuperação de texto "está preocupada em identificar documentos, em uma coleção, que melhor atendem a uma descrição da informação que um usuário necessita." (TURTLE; CROFT, 1992). Sendo ela – a descrição da informação –, chamada, no processo de recuperação da informação, de *query*, "um conjunto ordenado de palavras" (ISBELL, 1998) e uma coleção sendo "um conjunto de documentos em linguagem natural." (ISBELL, 1998).

Existem diversos modelos para a recuperação da informação em textos não-estruturados. Como apontado por Turtle e Croft (1992), um modelo de recuperação é a especificação de como os documentos e a informação precisam estar de modo que seja possível, de fato, recuperar os documentos com informações relevantes. Dentre eles, temos o *Modelo de Espaço Vetorial*, o

[15] Disponível em: http://geodjango.org/. Acesso em: janeiro de 2013.

qual trabalha com um conjunto de documentos d e um conjunto de termos t (HAN; KAMBER, 2006). Modelando cada documento como um vetor v em um espaço t-dimensional R^t, podemos criar uma *matriz de frequência de termos*, $TF(d,t)$ (frequência de termos, em inglês), a qual "mede a associação do termo t com dado documento d."(HAN; KAMBER, 2006). Uma maneira de calcular essa associação é através da *frequência* de t em d. Essa frequência é calculada como "o número de ocorrências do termo t no documento d, ou seja, $freq(d,t)$." (HAN; KAMBER, 2006).

Porém, esta não é a única maneira de calcular essa associação. O mensuramento pode ocorrer de forma a levar diferentes fatores em conta, um deles sendo a normalização da frequência do termo no documento e o outro, a importância do termo entre o conjunto de documentos, para, assim, ter valores que representam de forma mais confiável o nível de relevância do texto. Esta técnica é chamada de *TF-IDF* e será melhor detalhada na próxima subseção, pois foi a escolhida para ser utilizada nesse trabalho.

Vale ressaltar que, a fim de obter melhores resultados, um pré-processamento deve ser feito nos documentos e nos termos da *query*. Ele consiste, basicamente, na remoção de *palavras de parada*[16] e na transformação de todas as palavras restantes em seus *stems*[17]. O que resulta em uma *query* e documentos compostos apenas de palavras-chave. Dessa forma, a busca é feita, efetivamente, através da similaridades entre as palavras dos documentos e da *query*.

4.1.3.1 TF-IDF

(HAN; KAMBER, 2006) indica uma fórmula para normalizar a frequência do termo no documento. Então, ao invés de utilizarmos simplesmente $freq(d,t)$, utiliza-se

$$TF(d,t) = \begin{cases} 0, & \text{se} freq(d,t) = 0 \\ 1 + \log(1 + \log(freq(d,t))), & \text{caso contrário.} \end{cases}$$

para obtenção de melhores resultados.

Não bastando a medida recém apresentada, utiliza-se também a *frequência inversa de documento* (IDF, em inglês). Ela é importante, pois mensura a importância de um termo t. Ou seja, "se um termo t ocorre em muitos documentos, sua importância será diminuída devido ao seu reduzido poder discriminativo." (HAN; KAMBER, 2006). Como exemplo, (HAN;

[16] Como pode ser visto em http://www.searchengineshowdown.com/defs/stop.html (Acessado em: janeiro de 2013), são palavras de uso comum, tal como o, a, é, de, para, etc.

[17] Raíz da palavra. "Por exemplo, o grupo de palavras *droga, drogado e drogas* tem um *stem* em comum, *droga*, e pode ser visto como diferentes ocorrências da mesma palavra." (HAN; KAMBER, 2006).

KAMBER, 2006) afirma que o termo *database systems* possivelmente será pouco importante caso ele ocorra em muitos artigos em uma conferência de sistemas de banco de dados. Assim, $IDF(t)$ é definido como

$$IDF(t) = \log \frac{1 + |d|}{|d_t|},$$

onde d é a coleção de documentos e d_t é o conjunto de documentos contendo o termo t. Então, TF e IDF são combinados na forma

$$TF\text{-}IDF(d, t) = TF(d, t) \times IDF(t),$$

que representa o nível de similaridade e importância do termo t no documento d.

4.2 Comunicação do Servidor com o Aplicativo para Android

Para que o banco de dados possa ser acionado pelo usuário, no aplicativo para smartphones, é necessário que ele faça requisições GET ou POST ao servidor. Essas requições devem chamar funções definidas na parte de visualizações (no Django, estes métodos são implementados no arquivo views.py), as quais são responsáveis por exercer alguma lógica para a manipulação do banco de dados. No sistema InfoCity, conforme o modelo da figura 3.4, os seguintes métodos foram desenvolvidos:

1. addNewEvent(): Adiciona um novo evento ao banco de dados;
2. getEventsWithin(): Retorna todos os eventos relevantes (de acordo com um contexto ou filtro) que estão dentro de determinado raio;
3. getEventTypes(): Retorna todos os tipos de eventos cadastrados;
4. likeEvent(): Responsável por dar voto positivo ou negativo a um evento ou cancelar ele;
5. getLikeAction(): Verifica se determinado usuário votou tanto positiva, quanto negativamente ou não votou em um evento;
6. countLikesDislikes(): Retorna o número de votos positivos e negativos que determinado evento recebeu;
7. getComments(): Retorna todos os comentários associados a um evento e;
8. saveComment(): Salva um comentário sobre um evento.

Estes métodos são chamados pelo aplicativo para smartphone quando ele manda uma requisição para o servidor através de uma URL, a qual é formatada de forma a identificar que método deve ser executado e quais devem ser os valores de seus parâmetros (se for uma requisição do tipo GET. Requisições do tipo POST ficam invisíveis na URL). No Django, a definição de como as URLs devem ser formatadas é feita no arquivo urls.py. A título de exemplo, uma requisição para buscar eventos relevantes seria formatada da maneira apresentada na figura 4.5.

```
127.0.0.1/getWhithin/latitude/longitude/raio/limite de eventos/contexto
```

Figura 4.5 – Formato da URL utilizada para fazer a requisição de busca de eventos.

Onde os parâmetros devem ter valores válidos (e.g. raio deve ser um número maior do que zero).

A formatação básica com a qual as requisições, exceto alguns parâmetros delas, e respostas são editadas é a Notação de Objetos Javascript (JSON, em inglês). Uma explicação um pouco mais detalhada sobre JSON é feita na próxima subseção (Subseção 4.2.1).

4.2.1 Formato JSON

O JSON é uma forma leve para a troca de informações pela rede, se comparado ao XML. É independente de linguagem e fácil para leitura e escrita tanto para seres humanos quanto para computadores (ZYP et al., 2010). Um exemplo de esquema em JSON representando um evento é apresentado na figura 4.6.

```
{
"pk": 8,
"model": "events.event",
"fields": {
        "description": "T.neves estragou",
        "title": "onibus quebrado",
        "dislikes": 0,
        "geo_coord": "POINT (-53.8392 -29.7000)",
        "likes": 0,
        "keywords": ["onibus", "quebrado", "atraso"],
        "pub_date": "2012-11-28T04:57:53Z",
        "event_type": "bus"
        }
}
```

Figura 4.6 – Exemplo do formato de um evento em JSON.

Percebe-se que fields é um objeto JSON dentro do objeto JSON principal, sendo que seu campo keywords é uma lista de strings. Como notado por (SHIN, 2010), no Java é possível fazer a conversão de strings formatadas em JSON para objetos do tipo JSONObject facilmente (Figura 4.7).

```
JSONObject event_jsonobject = new JSONObject(event_json);
```

Figura 4.7 – Transformação de uma string formatada em JSON para um JSONObject.

Quando o aplicativo no smartphone busca por eventos no servidor, a resposta que ele recebe é uma string formatada em JSON com, dentre outras informações, vários eventos formatados, também, em JSON. Desse modo, cada um destes eventos é transformado em um JSONObject para, em seguida, ser transformado em um objeto do tipo EventItem.

Um exemplo de requisição, cujo um dos parâmetros deve estar formatado em JSON, é a busca de eventos (apresentado na figura 4.5). Neste caso, o parâmetro que o servidor espera que esteja formatado em JSON é o último, o contexto. Isto é feito para facilitar a transformação deste parâmetro em um objeto do modelo de tipo EventContextData para que, assim, seja manipulado com mais facilidade. Um exemplo de contexto formatado em JSON pode ser visto na figura 4.8.

```
{
"place_type": "praca",
"address": "Praca Farroupilha Porto Alegre Rio Grande do Sul",
"place_name": "Redencao",
"from_supplier": "Qr-Code",
"longitude":-53.8383,
"latitude":-29.7002,
"movement_state": "",
"on_commute": false
}
```

Figura 4.8 – Exemplo de dados de contexto formatado em JSON.

4.3 Aquisição de Eventos Relevantes ao Contexto

Como apresentado em seções anteriores no presente trabalho, a necessidade de que os eventos que um usuário recebe em seu smartphone estejam de acordo com o contexto em que ele se encontra existe, principalmente, pelo fato de que o número de eventos registrados ao seu

redor pode ser muito elevado e, com isso, poucos deles seriam realmente de seu interesse.

Para elucidar o que, de fato, é um contexto, utilizamos a seguinte definição:

> *"Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e aplicação."* (ABOWD et al., 1999).

Com ela em mente é que se definiu as informações que um contexto deveria ter (apresentadas na figura 4.8 e na tabela EventContextData da figura 3.5).

A modelagem realizada, na qual o InfoCity foi baseado, prevê o modo como o sistema deve se comportar a fim de incluir os contextos dos usuários nos seus diferentes aspectos. Nas próximas subseções, serão apresentadas de forma mais detalha como ocorre tanto a obtenção deste contexto, quanto a caracterização do nível de relevância de um evento a partir dele.

4.3.1 Obtenção do Contexto

Um ponto-chave para o sucesso do sistema, no que diz respeito à obtenção de contexto, é o de que ele deve exigir o mínimo, ou nenhum, esforço por parte do usuário para descobrir informações sobre o ambiente em que ele se encontra (ABOWD et al., 1999). Assim, como a utilização deste sistema é feita através de dispositivos móveis, o contexto do cidadão é sempre dinâmico. Dessa forma, é importante que o InfoCity seja capaz de adquirir diversas informações de ambiente diferentes para que ele possa atribuir um contexto mais preciso.

São apresentados na figura 4.8 os dados que devem ser preenchidos pelos provedores de contexto de forma a representar o ambiente ao redor do usuário. Essa aquisição é feita por parte do aplicativo para Android. Dessa forma, os provedores de contexto, apresentados nas subseções seguintes, foram desenvolvidos na forma de classes na linguagem java que implementam a interface ContextSupplier (como pode ser visto na figura 3.6).

As duas próximas subseções (Subseção 4.3.1.1 e Subseção 4.3.1.2) nos mostram como estes contextos são obtidos através de Qr-Codes e do Alohar[18] SDK[19], respectivamente. A seguinte (Subseção 4.3.1.3) mostra como o usuário pode interferir na aquisição de eventos a partir de filtros.

[18] Disponível em: https://www.alohar.com/developer/. Acesso em: janeiro de 2013.
[19] Kit de Desenvolvimento de Software, em inglês.

4.3.1.1 Qr-Code

Qr-Codes são, basicamente, códigos de barra de duas dimensões (2D). Devido a esta característica, eles são capazes de armazenar um número, de certa forma, elevado de informações. Com isso, existem diversas aplicações diferentes cuja utilização de Qr-Codes beneficiaria o seus sistemas. Uma delas é a explorada por (AL-KHALIFA, 2008). Cujo objetivo é auxiliar pessoas com problemas de visão para a identificar e apresentar as características de objetos.

Na figura 4.9, é apresentado um Qr-Code cuja a informação é um contexto formatado em JSON (tal como na figura 4.8).

Figura 4.9 – Exemplo de um Qr-Code cujo conteúdo é uma informação de contexto no formato JSON.

Supondo que o Qr-Code da figura 4.9 estivesse disponível em algum local de uma praça e o usuário estivesse buscando por eventos relevantes, o fluxo para a obtenção de contexto seria, internamente, como apresentado na figura 3.7. Porém, do ponto de vista do cidadão, seguiria na forma do modelo da figura 3.2. Representado, aqui, pela figura 4.10.

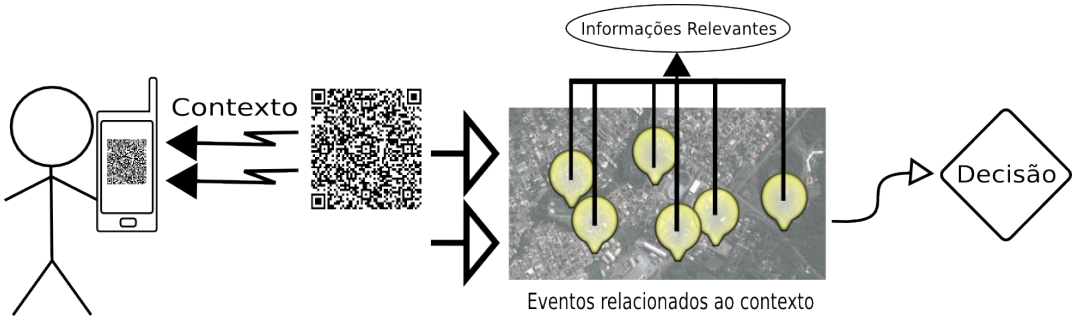


Figura 4.10 – Processo de busca de eventos relevantes ao contexto via Qr-Code do ponto de vista do usuário.

4.3.1.2 Alohar SDK

Ao contrário da aquisição de contexto utilizando Qr-Codes, o qual necessita que tais códigos de barras estejam dispostos nos diferentes pontos da cidade e que os usuários se di-

girijam ao local e façam a sua leitura com seus smartphones, o framework Alohar não requer nenhum tipo de ação por parte dos cidadãos.

A partir dos diversos sensores disponíveis pelos smartphones, o Alohar é capaz de detectar automaticamente o estado de movimentação de um usuário, o nome e o tipo de lugar em que ele se encontra, o número de vezes que ele visitou determinado lugar e o tempo que gastou nele (ALOHAR DEVELOPER OVERVIEW, 2012). Com isso, justifica-se o seu uso para a obtenção de contexto.

A diferença no fluxo apresentado na figura 4.10 é apenas a de que, como já dito, o usuário não teria que fazer ação alguma (como, por exemplo, se deslocar e ler um Qr-Code). A tabela 4.1 compara os métodos de aquisição de contexto: Qr-Code e o Alohar SDK.

Tabela 4.1 – Tabela comparativa entre os métodos de obtenção de contexto Qr-Code e Alohar SDK.

Qr-Code	Alohar SDK
Necessário que esteja disponível em diversos pontos da cidade.	Não requer nenhum tipo extra de informação.
Usuário precisa deslocar-se até o local onde o Qr-Code se encontra e apontar seu smartphone a ele.	O usuário não precisa fazer ação alguma.
Os dados obtidos são bastante precisos.	Dados não são tão precisos. Inclusive, podem ser de um local diferente, caso a obtenção da localização do usuário não for boa o suficiente.

4.3.1.3 Filtros

Com a finalidade de permitir ao próprio usuário a definição de que tipos de eventos ele deseja receber, criou-se uma maneira para que isso seja possível. Através da busca por eventos utilizando uma filtragem pelos seus tipos, o cidadão decide qual tipo de eventos são relevantes a si.

Atualmente, no sistema, existem os seguintes tipos pré-definidos de eventos: ônibus, saúde, denúncia, compras e sugestão.

Para que esta utilização de filtros fosse mais flexível, permitiu-se ao usuário não só buscar todos os eventos de um tipo escolhido, mas também buscar eventos que estão de acordo com o seu contexto e que sejam do tipo determinado por ele.

4.3.2 Determinação da Relevância de um Evento

Após a descoberta das informações sobre o ambiente em que o cidadão se encontra e, após este decidir buscar por eventos a partir de seu contexto, resta ao servidor determinar quais são os eventos que devem ser retornados ao aplicativo. Ou seja, o servidor, a partir do contexto do usuário, deve decidir quais são os possíveis eventos relevantes a ele.

Este cálculo de relevância é feito a partir das seguintes variáveis:

1. A primeira delas, e mais importante, é o nível de similaridade dos dados de um evento com relação aos dados de um contexto. Essa similaridade é calculada através da técnica apresentada na seção sobre recuperação de textos (Seção 4.1.3). A forma com que o sistema foi adaptado para utilizar-se deste método, foi a transformação dos eventos e contextos em textos, da seguinte forma:

    - Todas as informações relativas a um evento são transformadas em uma única string, sendo que cada informação é separada por um espaço. Vale lembrar que cada evento tem um contexto associado e, com isso, este contexto também é adicionado na representação textual do evento (Figura 4.11).

```
#evento transformado em texto
"titulo do evento" + "descricao do evento"
+ "tipo do evento" + "palavras-chave"
#contexto transformado em texto
+ "nome do lugar" + "tipo do lugar"
```

Figura 4.11 – Forma textual de um evento para ser analisada na recuperação de texto.

    - Como podemos ver na figura 4.11, quando o contexto é transformado em texto, apenas as informações de place_name e place_type são utilizadas.

    Após feita esta transformação, os eventos começam a ser tratados como documentos e o contexto do usuário se transforma nos termos de busca textual. Passando, desta forma, uma lista de eventos (documentos) e um contexto (termos) ao algorítmo TF-IDF (Subseção 4.1.3.1). O qual, retorna um valor de similaridade para cada evento;

2. Quantidade de votos positivos e negativos (*likes* e *dislikes*). De modo que, quanto mais votos positivos, mais relevante o evento. E quanto mais votos negativos, menos relevante;

3. Data de publicação do evento, dando prioridade aos eventos criados mais recentemente.

A fórmula utilizada para cálculo da relevância é a que se segue:

$$\frac{tf\_idf + (likes - dislikes) \times tf\_idf \times 0.001}{dias + segundos}$$

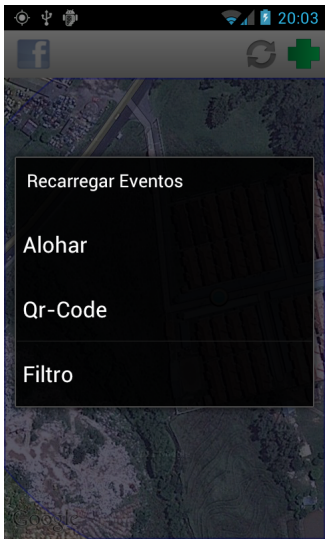
A multiplicação de $(likes - dislikes) \times tf\_idf \times 0.001$ é feita para serem levados em conta os votos. Sendo que, devido ao $\times 0.001$, cada voto que o evento recebeu aumenta ou diminui (se for positivo ou negativo, respectivamente) em 0.1% o valor do tf_idf.

4.4 Demonstração do Aplicativo para Android

Ao iniciar o aplicativo, o usuário tem a opção de buscar por eventos ou adicionar um novo no ponto em que se encontra (Figura 4.12a). Tanto a adição quanto a busca por eventos, como visto no capítulo de modelagem (Capítulo 3), requer uma fonte que indique o contexto em que o cidadão se encontra. Sendo assim, uma lista com as possíveis fontes de contexto é apresentada a ele para que escolha o provedor mais pertinente a si naquele momento (Figura 4.12b e Figura 4.12c).

(a) Tela inicial do aplicativo.

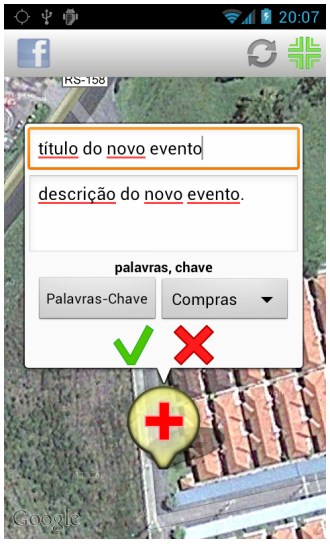
(b) Aquisição de contexto para adição de evento.



(c) Aquisição de contexto para buscar eventos.

Figura 4.12 – Ao iniciar o aplicativo, o usuário se depara com a tela da figura 4.12a (o ponto em destaque é a sua localização atual). Então, ele pode decidir buscar por eventos ou adicionar um novo. Cada uma destas ações exige a obtenção de seu contexto (Figura 4.12c e Figura 4.12b).

Após ter obtido seu contexto, é permitido ao usuário finalizar as ações desejadas. Se for a de adição de um novo evento, um breve formulário é exibido para que as informações sobre tal evento sejam cadastradas (Figura 4.13a). Agora, se a ação foi a busca por eventos, são mostrados num mapa todos os eventos cadastrados que estejam ao redor do usuário e de acordo



(a) Após obter o contexto, o usuário pode adicionar um novo evento.

(b) Após ler um Qr-Code, foram recebidos eventos relacionados ao contexto adquirido.

(c) Balão com as informações de um evento, além das possibilidades de votar e comentar nele.

Figura 4.13 – Após adquirir o contexto, o usuário pode adicionar um evento (Figura 4.13a) ou buscar por eventos (Figura 4.13b). Para visualizar as informações de um evento, o cidadão deve selecioná-lo (Figura 4.13c). Vale notar que, como o usuário não está identificado, não é possível a ele votar ou comentar no evento. Ele pode apenas visualizar as suas informações e comentários.

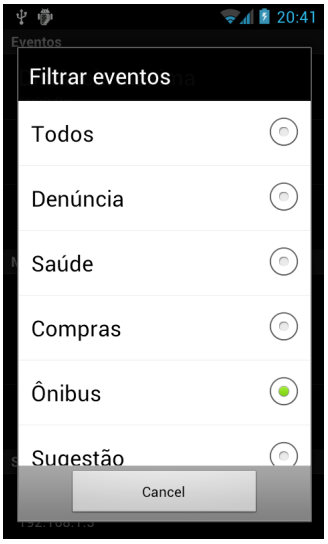
com o seu contexto (Figura 4.13b). Sendo que basta ao cidadão selecionar um destes que suas informações são apresentadas em um balão acima de sua localização (Figura 4.13c).

São disponibilizadas, também, algumas configurações básicas para um controle maior do usuário (Figura 4.14a) sobre o que ele visualiza. Uma delas é a de seleção de filtros para a busca de eventos (Figura 4.14b). Como visto no capítulo de modelagem (Capítulo 3), ela permite que usuários busque eventos de um único tipo e que não estejam de acordo com seu contexto (Figura 4.14c), pois nem sempre as informações de contexto são precisas o suficiente.

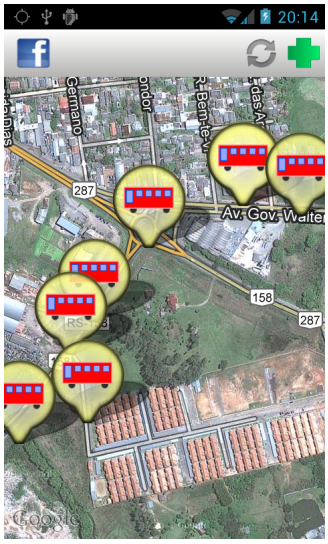
Como parte relevante ao sistema, é permitido ao usuário tanto votar (Figura 4.15b), quanto comentar (4.15c) em eventos. Desde que o usuário esteja identificado no sistema (Figura 4.15a). Para que o usuário se identifique no sistema, foi utilizada a conexão com o *Facebook*[20] através do *Facebook SDK*[21].

[20] Disponível em: http://facebook.com. Acesso em: janeiro de 2013.
[21] Disponível em: http://developers.facebook.com/. Acesso em: janeiro de 2013.

(a) Tela de preferências.

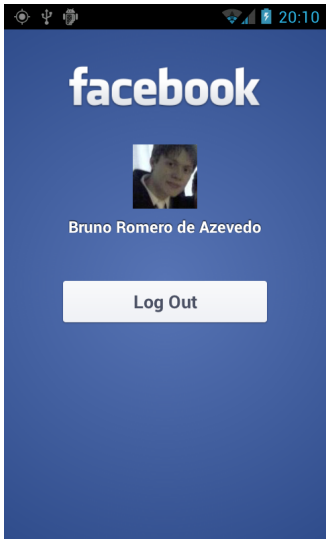


(b) Lista de tipos de eventos para filtragem.



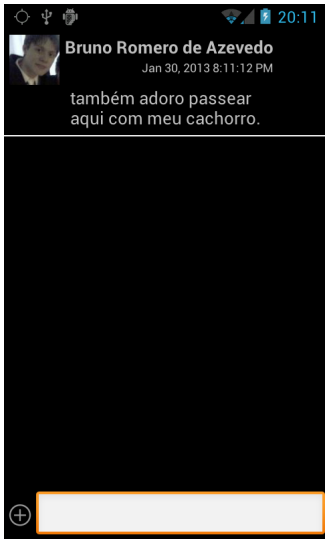
(c) Busca de eventos utilizando como filtro o tipo de evento 'ônibus'.

Figura 4.14 – A figura 4.14a mostra as possíveis preferências que um usuário pode determinar. Uma delas, é com relação ao filtro de busca (Figura 4.14b). Para que, se for de sua escolha, os eventos retornados sejam apenas do tipo especificado (Figura 4.14c).



(a) Usuário identificado no sistema.

(b) Usuário votou positivamente no evento.



(c) Comentários de um evento.

Figura 4.15 – Uma vez identificado no sistema (Figura 4.15a), ele pode tanto votar (Figura 4.15b), quanto comentar (Figura 4.15c) nos eventos.

4.5 Testes e Avaliação

Percebeu-se a necessidade de testar o sistema com relação à obtenção de eventos relevantes pelo usuário. Ou seja, verificar se a fórmula definida na subseção 4.3 funciona de fato,

bem com o algorítmo de recuperação de texto TF-IDF (Subseção 4.1.3.1).

A subseção a seguir (Subseção 4.5.1) apresenta o modo como estas informações foram testadas e faz uma análise dos resultados obtidos.

4.5.1 Teste de Contexto

Por ser uma parte importante do presente trabalho, é preciso garantir que as informações de eventos sejam, de fato, relevantes ao contexto em que o cidadão se encontra. Dessa forma, o ambiente no qual os testes de contexto foram aplicados, é o seguinte:

- No servidor, todos os eventos e informações de contextos, que não relacionadas ao teste, foram removidos;
- Determinou-se três contextos, ctx_1, ctx_2 e ctx_3, diferentes para tanto simular o ambiente em que o usuário se encontra, quanto para serem associados aos eventos;
- Foram criados nove eventos diferentes, $e_1..e_9$. Cada contexto foi associado a três deles e, cada um destes três tem, em suas informações, somente palavras relacionadas a um dos contextos (Figura 4.16).

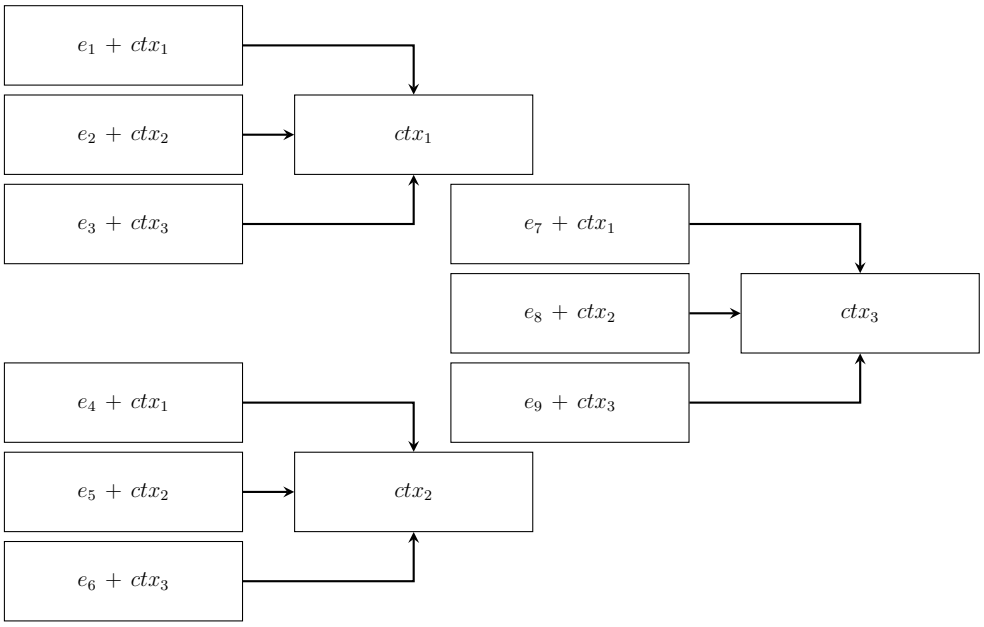


Figura 4.16 – Resumo de como foi feito o ambiente de testes de contexto.

As informações de cada contexto podem ser vistas na tabela 4.2. Como o servidor verifica somente o nome do lugar e o tipo de lugar para fazer as suas verificações de relevância, apenas estas variáveis são apresentadas. Já as dos eventos, podem ser vistas na tabela 4.3.

Tabela 4.2 – Informações dos contextos no ambiente de testes.

	Contextos		
Informação	ctx_1	ctx_2	ctx_3
place_name	Praça Farroupilha	Parada de Ônibus	Hospital de Caridade
place_type	praça	parada de ônibus	hospital

Após termos feitos estas determinações, podemos partir para o teste em si. Ele consiste em enviar um contexto ao servidor (cada um dos três anteriormente apresentados) e verificar qual o valor de relevância (calculado pela fórmula apresentada na subseção 4.3.2) que cada evento recebeu para cada contexto enviado. Os resultados são apresentados na tabela 4.4.

4.5.1.1 Análise dos Resultados

É interessante notar que os eventos apresentados na tabela 4.4 com maior valor para relevância (valores destacados), em cada nova iteração, são aqueles cujos textos (título, descrição, etc.) e contextos associados são iguais ao contexto de busca. Significando que os eventos vão ser mais relevantes, quanto mais relacionado ao contexto associado seus textos estiverem, sendo este o mesmo contexto da busca. Tanto é que todos os eventos, cujos contextos associados eram o mesmo da busca, tiveram valores elevados.

Outro fator válido de ser mencionado, é o de que eventos que não tem relação alguma com o contexto de busca também podem ter algum valor de relevância maior do que 0. Por exemplo, com o contexto de busca ctx_1 o evento e_6 teve um valor de relevância maior que 0, sendo que seu contexto associado é o ctx_2 e seu texto não tem qualquer referência a dados do ctx_1. Vale destacar, porém, que seu valor de relevância é o menor dentre os maiores que 0 para o contexto de busca ctx_1. Ou seja, em um ambiente com vários eventos relacionados ao contexto, ele ficaria para o fim da fila. De modo que ele só seria apresentado ao usuário, caso o número de eventos que este desejasse receber fosse bastante grande.

Portanto, podemos concluir, a partir dos resultados apresentados na tabela 4.4, que a fórmula utilizada (mostrada na subseção 4.3.2) satisfaz as necessidades do sistema. Uma vez que os eventos mais relevantes são aqueles cujos textos e contexto associado estão de acordo com o contexto do usuário que está buscando as informações. Percebe-se isso mesmo que não tenham sido feitos testes para eventos com votos positivos ou negativos, pois fica fácil notar, apenas olhando para a fórmula, que quanto mais votos positivos, maior o valor de relevância e quanto mais votos negativos, menor o valor de relevância.

Tabela 4.3 – Informações dos eventos no ambiente de testes.

Eventos associados ao contexto ctx_1			
Informação	Evento e_1	Evento e_2	Evento e_3
título	Praça muito bonita.	Faltando uma parada aqui.	Faltando hospital.
descrição	Essa Praça Farroupilha está cada vez melhor.	Estamos tendo que caminhar muito para o ônibus.	Este bairro ainda não tem hospital.
tipo	sugestão	denúcia	denúncia
palavras-chave	praça, bonita	parada, ônibus, problema	hospital, bairro, problema
Eventos associados ao contexto ctx_2			
Informação	Evento e_4	Evento e_5	Evento e_6
título	Praça está suja.	Ônibus atrasado.	Hospital com pouca gente.
descrição	Quando o governo vai limpá-la?	Estou há 2 horas na parada.	Praticamente vazio.
tipo	denúncia	ônibus	saúde
palavras-chave	praça, suja, limpar	parada, ônibus, atraso	hospital, vazio
Eventos associados ao contexto ctx_3			
Informação	Evento e_7	Evento e_8	Evento e_9
título	Praça muito barulhenta.	Cadê a parada?.	Hospital intransitável.
descrição	Podiam parar com o barulho, né?	Por que tiraram a parada daqui?	Hospital lotado, não venham!
tipo	denúncia	ônibus	saúde
palavras-chave	praça, barulhenta	parada, ônibus, sumiu	hospital, cheio

Tabela 4.4 – Valores de relevância para cada contexto (estão multiplicados por 100000 para facilitar a visualização).

	Contextos		
Eventos	ctx_1	ctx_2	ctx_3
e_1	**113.1727**	0	0
e_2	104.5595	127.05737	0
e_3	108.8578	0	80.6116
e_4	37.2341	137.0573	0
e_5	0	**165.6810**	0
e_6	36.2059	149.5342	85.2191
e_7	42.8494	31.6157	159.6656
e_8	0	148.0298	164.1559
e_9	0	0	**184.0463**

5 CONCLUSÃO

A integração cidadã a partir do uso de tecnologias móveis, para o descobrimento e desenvolvimento de suas cidades, é uma alternativa que traz diversos ganhos devido à sua facilidade de uso e mobilidade (dentro de um contexto de cidades inteligentes). Por permitir ao usuário a criação de eventos quando estes ocorrem e sua busca, quando é necessária, o encoraja a exercer sua cidadania. Tanto é, que um sistema semelhante já disponibilizado às pessoas da cidade de Porto Alegre, PortoAlegre.cc, engajou um grande número de cidadãos para a sua utilização mesmo sem mobilidade (atualmente, sua utilização se faz apenas via navegadores *web*). Do mesmo modo, se disponibilizado ao público em geral, os próprios usuários perceberiam a capacidade de desenvolvimento que o sistema InfoCity é capaz de trazer às suas cidades.

Como explicado neste trabalho, o InfoCity foi um sistema prototipado para permitir aos cidadãos o exercício de suas cidadanias, compartilhando e discutindo fatos que ocorrem pelas suas cidades. Além de auxiliá-los na tomada de decisões com o descobrimento de eventos que estão ocorrendo ao seu redor, estando eles relacionados ao ambiente em que se encontram. Reparamos, através do teste realizado que a utilização da técnica de recuperação de textos TF-IDF, por parte do servidor, teve um desempenho satisfatório para encontrar os eventos relevantes ao contexto.

Com isso, podemos perceber que o sistema implementado atinge os objetivos propostos.

5.1 Trabalhos Futuros

Por mais que os objetivos estejam sendo cumpridos, ainda há espaço para melhorias no sistema. Uma delas seria a personalização da relevância de eventos para cada pessoa em separado. Não bastando apenas o contexto como meio de determinar os possíveis eventos relevantes a um usuário. Um método que permitisse a ele distinguir quando certos eventos retornados são relevantes e, a partir disso, desenvolver um modelo de aprendizagem do sistema específico para cada cidadão, traria ganhos nas questões de usabilidade e utilidade do sistema.

Ainda com relação à relevância de eventos, seria interessante a utilização, de forma inteligente, de todos os dados de contexto para melhorar, também, a determinação desta relevância. Atualmente, apenas as informações de nome e tipo do local em que a pessoa se encontra estão sendo verificadas.

Alguns pontos específicos foram levantados, a fim de melhorar o sistema. Eles não

foram executados por motivos de tempo:

1. Integrar votos positivos e negativos aos comentários. E, assim, organizá-los de forma a favorecer os mais bem votados;
2. Utilizar estes comentários quando a técnica de recuperação de textos é aplicada;
3. Testar diferentes fórmulas para a determinação da relevância de contexto, levando alguns outros fatores em conta, como a hora do dia, por exemplo;
4. Adicionar outros tipos de filtros. Por exemplo, por palavras-chave e;
5. Testes de usabilidade e aplicabilidade com um número elevado de pessoas.

REFERÊNCIAS

ABOWD, G. et al. Cyberguide: a mobile context-aware tour guide. **Wireless networks**, [S.l.], v.3, n.5, p.421–433, 1997.

ABOWD, G. et al. Towards a better understanding of context and context-awareness. In: HANDHELD AND UBIQUITOUS COMPUTING. **Anais...** [S.l.: s.n.], 1999. p.304–307.

AL-KHALIFA, H. Utilizing qr code and mobile phones for blinds and visually impaired people. **Computers Helping People with Special Needs**, [S.l.], p.1065–1069, 2008.

ALEXANDER, A. Smartphone Usage Statistics 2012 [Infographic]. **http://ansonalex.com/infographics/smartphone-usage-statistics-2012-infographic/**, [S.l.], 2012. Acesso em: janeiro de 2013.

ALOHAR Developer Overview. [S.l.]: Alohar Mobile Inc., 2012.

ARRUDA FREIRE, G. de et al. **O ciberativismo na construção da ciberdemocracia**: análise do portal wikicidade de porto alegre. 2012.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling language user guide**. [S.l.]: Pearson Education India, 1999.

CHACON, S.; HAMANO, J.; PEARCE, S. **Pro Git**. [S.l.]: Apress, 2009. v.288.

CHEVERST, K. et al. Developing a context-aware electronic tourist guide: some issues and experiences. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS. **Proceedings...** [S.l.: s.n.], 2000. p.17–24.

DEVELOPERS, A. What is Android? **http://developer.android.com/guide/basics/what-is-android.html**, [S.l.], v.2, 2011. Acesso em: janeiro de 2013.

GERSON, D.; VALIATI, V. **PortoAlegre.cc, Shoot the Shit e Espiral positiva**: administração público-criativa através das redes sociais. 2012.

GÜTING, R. An introduction to spatial database systems. **The VLDB Journal**, [S.l.], v.3, n.4, p.357–399, 1994.

HAN, J.; KAMBER, M. **Data mining**: concepts and techniques. [S.l.]: Morgan Kaufmann, 2006.

HOLOVATY, A.; KAPLAN-MOSS, J. **The Definitive Guide to Django**: web development done right. [S.l.]: Apress, 2009.

ISBELL, C. L. **Sparse Multi-Level Representations for Text Retrieval**. 1998. Tese (Doutorado em Ciência da Computação) — Massachusetts Institute of Technology, Cambridge, MA.

KOMNINOS, N. The architecture of intelligent clities: integrating human, collective and artificial intelligence to enhance knowledge and innovation. In: INTELLIGENT ENVIRONMENTS, 2006. IE 06. 2ND IET INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. v.1, p.13–20.

MURPHY, M. **The busy coder's guide to Android development**. [S.l.]: Commonsware, LLC, 2009.

O'NEIL, E. Object/relational mapping 2008: hibernate and the entity data model (edm). In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2008. **Proceedings...** [S.l.: s.n.], 2008. p.1351–1356.

PEREIRA, A. et al. Frameworks para Desenvolvimento Rápido de Aplicações Web: um estudo de caso com cakephp e django. **10º Fórum Internacional de Software Livre (FISL 10)**, [S.l.], 2009.

RELYEA, H. E-gov: introduction and overview. **Government Information Quarterly**, [S.l.], v.19, n.1, p.9–36, 2002.

SHIN, S. Introduction to JSON (JavaScript Object Notation). **Presentation** `www.javapassion.com`, [S.l.], 2010. Acesso em: janeiro de 2013.

TURTLE, H.; CROFT, W. A comparison of text retrieval models. **The computer journal**, [S.l.], v.35, n.3, p.279–290, 1992.

ZYP, K. et al. A JSON media type for describing the structure and meaning of JSON documents. `http://tools.ietf.org/html/draft-zyp-json-schema-01`, [S.l.], 2010. Acesso em: janeiro de 2013.