**MINISTÉRIO DA DEFESA**

**EXÉRCITO BRASILEIRO**

**DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA**

**INSTITUTO MILITAR DE ENGENHARIA**

**Seção de Engenharia de Computação / SE8**

**FELIPE DA COSTA RASINHAS**

**JONATHAN CORREIA CARVALHOSA**

**PROJETO DE ANÁLISE E DESENVOLVIMENTO DE UMA SOLUÇÃO MÓVEL INTEGRADORA**

**Rio de Janeiro**

**2013**

**INSTITUTO MILITAR DE ENGENHARIA**

**FELIPE DA COSTA RASINHAS**

**JONATHAN CORREIA CARVALHOSA**

**PROJETO DE ANÁLISE E DESENVOLVIMENTO DE UMA SOLUÇÃO MÓVEL INTEGRADORA**

Projeto de fim de curso apresentado ao Curso de Graduação de Engenharia de Computação como requisito parcial para a obtenção do título de Engenheiro.

Orientador: Prof. Ricardo Choren Noya

Rio de Janeiro

2013

c2013

INSTITUTO MILITAR DE ENGENHARIA

Praça General Tibúrcio, 80 – Praia Vermelha

Rio de Janeiro - RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

XXXX CARVALHOSA, Jonathan Correia. RASINHAS, Felipe da Costa.

Projeto de análise e desenvolvimento de uma solução móvel integradora / Jonathan Correia Carvalhosa, Felipe da Costa Rasinhas. Rio de Janeiro: Instituto Militar de Engenharia, 2013.

47 f.

Projeto de fim de curso (graduação) – Instituto Militar de Engenharia, 2013.

1. ###PALAVRAS CHAVE###

**INSTITUTO MILITAR DE ENGENHARIA**

**JONATHAN CORREIA CARVALHOSA**

**FELIPE DA COSTA RASINHAS**

**PROJETO DE ANÁLISE E DESENVOLVIMENTO DE UMA SOLUÇÃO MÓVEL INTEGRADORA**

Projeto de fim de curso do Curso de Engenharia de Computação do Instituto Militar de Engenharia, como requisito para colação de grau no curso de Engenharia de Computação.

Orientador: Prof. Ricardo Choren Noya

Aprovada em \_\_ de \_\_\_\_\_\_\_\_de 2013 pela seguinte Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Ricardo Choren Noya, D.C. , do IME– Presidente

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Maj QEM Anderson Fernandes P. dos Santos, D.Sc.., do IME

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Profa. Raquel Coelho Gomes Pinto, D.Sc., do IME

Rio de Janeiro

2013

SUMÁRIO

LISTA DE ILUSTRAÇÕES 5

LISTA DE SIGLAS 6

1 INTRODUÇÃO 9

1.1 MOTIVAÇÃO 9

1.2 OBJETIVO 9

1.3 METODOLOGIA 10

1.4 ORGANIZAÇÃO DA MONOGRAFIA 10

2 FERRAMENTAS UTILIZADAS 11

2.1 RIO DATAMINE 11

2.1.1 AUTENTICAÇÃO 11

2.1.2 REQUISIÇÃO 11

2.1.3 RESPOSTA 12

2.2 IOS 12

2.3 OBJECTIVE C 12

2.4 PYTHON 13

2.5 DJANGO 13

2.6 MYSQL 13

2.7 SOUTH 13

2.8 ASI-HTTP-Request 14

2.9 SBJSON 14

2.10 GIT 14

2.11 GITHUB 15

3 DESENVOLVIMENTO 16

3.1 DISPOSITIVO MÓVEL 17

3.2 SERVIDOR WEB 18

3.2.1 COMUNICAÇÃO COM A API 18

3.2.2 BANCO DE DADOS 19

4 DOCUMENTAÇÃO 20

4.1 USUÁRIOS 20

4.1.1 REGISTRAR USUÁRIO 20

4.1.2 LOGIN 21

4.1.3 LOGOUT 23

4.1.4 EDITAR DADOS 23

4.2 PESQUISAS 24

4.2.1 UTILIDADE PÚBLICA 26

4.2.2 RESTAURANTES 28

4.2.3 HOSPEDAGEM 29

4.2.4 ENTRETENIMENTO 30

3.3 GERENCIAMENTO DE PREFERÊNCIAS 31

5 CONCLUSÃO 33

6 REFERÊNCIAS BIBLIOGRÁFICAS 35

7 ANEXOS 36

7.1 ANEXO 1 36

# LISTA DE ILUSTRAÇÕES

FIG. 3.1 Arquitetura do Sistema.................................................................................16

FIG. 4.1.1.1 Cadastro.................................................................................................21

FIG. 4.1.2.1 Login.......................................................................................................22

FIG. 4.1.2.2 Menu inicial do sistema..........................................................................23

FIG. 4.1.4.1 Edição de dados básicos.......................................................................24

FIG. 4.2.1 Menu de pesquisas do sistema.................................................................25

FIG. 4.2.1.1 Filtro de pesquisa por serviços de utilidade pública...............................26

FIG. 4.2.1.2 Resultados de uma pesquisa.................................................................27

FIG. 4.2.1.3 Mapa com o resultado............................................................................28

FIG. 4.2.2.1 Filtro de pesquisa de restaurantes.........................................................29

FIG. 4.2.3.1 Filtro de pesquisa de hospedagens.......................................................30

FIG. 4.2.4.1 Filtro de pesquisa de entretenimento.....................................................31

# LISTA DE SIGLAS

API Application programming interface

HTTP Hypertext Transfer Protocol

JSON Javascript Object Notation

IDE Integrated Development Environment

iOS iPhone Operating System

IP Internet Protocol

MVC Model-View-Controller

OS Operating System

SQL Structured Query Language

URL Uniform Resource Locator

XIB Xcode Interface Builder

XML Extensible Markup Language

**RESUMO**

Nos próximos anos, o Rio de Janeiro sediará eventos de caráter global, e por este motivo, haverá um grande número de turistas na cidade durante este período. Nesse contexto, é importante o desenvolvimento de *softwares* de apoio, que possam ajudar essas pessoas, desde a escolha do transporte a ser utilizado até a chegada no local e a localização de pontos estratégicos, como polícia, bombeiros e hospitais.

Este trabalho consiste no desenvolvimento de uma aplicação *mobile* que fornece essas informações através de consultas a uma base de dados de apoio do governo, o Rio DataMine. Em paralelo a isso, será configurado um servidor *web* para centralizar as consultas e armazenar informações sobre os usuários.

Com esse trabalho, espera-se facilitar o deslocamento dos turistas pela cidade, disponibilizando informações que tornem sua estadia mais prazerosa.

**ABSTRACT**

In the next years, Rio de Janeiro will host global scale events, and for that reason, there will be an enormous number of tourists in town at this period. Within this context, the development of assist softwares is very important, so it can help them choosing the transportation, finding the event places and localizing strategic points, like police departments, fire departments, hospitals and public bathrooms.

This project consists in the development of a mobile application that provides this information through queries to an auxiliary database from the government, the Rio DataMine. In parallel to the mobile application, a web server will be configured, so it centralizes the queries and stores data from the users.

This work is expected to improve the mobility of tourists around town, providing information to make their stay more enjoyable.

# 1 INTRODUÇÃO

## 1.1 MOTIVAÇÃO

Prestes a protagonizar o papel de centro das atenções mundiais, o Rio de Janeiro precisa oferecer aos turistas informações claras, concisas e corretas. Não é o que se vê hoje. Entidades públicas e privadas raramente fornecem dados de forma sistêmica. Com isso, informações essenciais ficam soltas no ar. Por outro lado, é muito difícil ao turista incauto ter acesso às mesmas.

Com uma segurança pública que ainda deixa bastante a desejar e uma sinalização das vias públicas ineficiente, o turista fica perdido. A principal motivação para esse trabalho é disponibilizar ao turista as informações de que ele precisa para que sua estadia na cidade seja mais prazerosa.

Outra grande motivação é o desafio de aprender mais sobre o desenvolvimento de aplicativos para iPhone, que é um mercado que está em grande ascensão. Através da *app store*, a Apple permite que os desenvolvedores do aplicativo ganhem 70% de todo o lucro gerado pela venda. Além disso, a documentação presente no site destinado aos desenvolvedores é amigável e possui muitos exemplos que facilitam o aprendizado.

## ****1.2 OBJETIVO****

Este trabalho teve como objetivo fazer o desenvolvimento de um sistema integrador para dispositivos móveis. O sistema tem o propósito de permitir a aplicação de conhecimentos em duas frentes: (i) desenvolvimento para dispositivos móveis, mais especificamente para a plataforma iOS, e; (ii) desenvolvimento com reuso através da integração com um sistema de informação já existente e não mantido pelo grupo. O domínio de aplicação escolhido para o sistema desenvolvido é o de mobilidade na cidade do Rio de Janeiro. Desta forma, o sistema visa ajudar os turistas a se deslocarem com mais facilidade pela cidade do Rio de Janeiro, tirando dúvidas quanto a localização dos serviços de utilidade pública, restaurants, hotéis e atividades de lazer, e sugerindo locais a serem visitados de acordo com as suas preferências.

## 1.3 METODOLOGIA

A base de dados utilizada foi a do site Rio DataMine [2]. As principais vantagens que influenciaram nessa decisão foram a facilidade de uso e a alta disponibilidade, além de ser uma ferramenta criada pela própria prefeitura com o objetivo de estimular a criação de aplicativos. A primeira parte do trabalho foi o estudo do formato exigido nos métodos HTTP para as requisições e para as respostas do site e, para a próxima etapa será desenvolvido um serviço para automatizar o acesso a base de dados.

O núcleo da aplicação foi desenvolvido a partir do modelo de casos de uso e será posteriormente revisado para que seja alcançada a versão final.

O dispositivo escolhido para hospedar o aplicativo foi o iPhone. Entre as principais vantagens que influenciaram esta decisão estão a grande quantidade de usuários do iPhone em todo o mundo e a facilidade de acesso a documentação da Apple.

## 1.4 ORGANIZAÇÃO DA MONOGRAFIA

O trabalho está distribuído em um capítulo abordando as tecnologias utilizadas para o desenvolvimento do aplicativo, descrevendo todas as ferramentas a serem utilizadas ao longo do projeto e a arquitetura do sistema; seguido de um capítulo onde são descritas as funcionalidades do aplicativo. A seção 3.1 descreve as funcionalidades relacionadas ao cadastro e gerenciamento dos dados básicos do usuário. A seção 3.2 descreve e exemplifica as principais funcionalidades do sistema, que são as consultas a base de dados da prefeitura. O relatório é finalizado na seção 4 com a conclusão do trabalho.

# 2 FERRAMENTAS UTILIZADAS

Para a execução do projeto, foi necessária a escolha de uma plataforma *mobile*, uma base de dados contendo informações sobre a cidade e um *framework* de desenvolvimento *web*. Esta seção tem como objetivo esclarecer a escolha de cada um destes componentes do sistema, além de citar algumas das características dos mesmos.

## 2.1 RIO DATAMINE

O Rio DataMine [1] é uma iniciativa do governo para melhorar a acessibilidade, a transparência e a mobilidade dos dados  públicos  pra os moradores da cidade. Para isso, o site fornece acesso a um servidor composto por bases de dados legíveis por computadores e dispositivos similares.

O acesso aos dados pode ser feito por qualquer pessoa através de requisições HTTP.  O formato esperado da requisição e da resposta será brevemente descrito, de acordo com os padrões especificados pela API do Rio DataMine.

### 2.1.1 AUTENTICAÇÃO

Para um aplicativo acessar os serviços da API do Rio DataMine, é necessária a posse de um *token* conhecido com *access token*, cuja validade é de seis horas. Para obter o *access token*, o aplicativo deve realizar um HTTP GET passando os parâmetros app\_id e app\_secret, que são identificadores obtidos assim que o aplicativo é cadastrado no site.

### 2.1.2 REQUISIÇÃO

O formato esperado é: http://api.riodatamine.com.br/rest/DATASET?QUERY, onde *dataset*  e *query* são, respectivamente, o serviço e o conjunto de filtros que a serem aplicados na consulta.

### 2.1.3 RESPOSTA

O formato da resposta é, por padrão, o formato json, que consiste em uma lista de pares (chave, valor) e é muito comum em aplicações *web*, devido a facilidade na manipulação dos dados de um objeto json.

## 2.2 IOS

O iOS [2] é o sistema operacional móvel desenvolvido pela Apple. Ele foi desenvolvido originalmente para iPhone, mas acabou sendo usado também em iPod, iPad e Apple TV. Ele é dividido em quatro camadas básicas:

* Core OS Layer: É a camada mais básica, que se comunica diretamente ao hardware e serve diretamente às camadas superiores.
* Core Services Layer: Contém os serviços mais fundamentais, que todas as outras aplicações podem usar. Ela cuida, por exemplo, da conexão com a *internet*, do contador de referência das variáveis, da criptografia e do serviços de compartilhamento de arquivos.
* Media Layer: Contém as ferramentas gráficas e as tecnologias de vídeo, áudio disponíveis no aparelho.
* Cocoa Touch Layer: É a camada mais externa, e contém as tecnologias necessárias para o contato direto entre o aparelho e o usuário, como captura dos toques na tela e envio de notificações por *push*.

## 2.3 OBJECTIVE C

O Objective C é uma linguagem de alto nível orientada a objeto, que adiciona eventos e troca de mensagens à tradicional linguagem C. Essa é principal linguagem utilizada pela Apple e é nela que devem ser desenvolvidos os aplicativos para iPhone.

## 2.4 PYTHON

Python [3] é uma linguagem de programação interpretada, de alto nível e orientada a objetos. Esta linguagem foi desenvolvida com o objetivo de priorizar a legibilidade do código e sua sintaxe possibilita expressar conceitos em menos linhas de código do que em linguagens como C e Java.

Os principais motivos da escolha do Python como linguagem base para o desenvolvimento da plataforma *web* foram a facilidade e velocidade de se desenvolver um código na linguagem, além do conhecimento de vários *frameworks* de apoio para o desenvolvimento de uma aplicação *web* como o Django e o South.

## 2.5 DJANGO

O Django [4] é um *framework* *open source* para o desenvolvimento de aplicações *web* utilizando Python. Este framework é baseado no padrão MVC e fornece facilidades como uma modelagem de dados ORM, uma linguagem de *templates* extensa e amigável, um sistema de serialização de dados que possibilita a representação de XML e json como instancias de objetos e um sistema de internacionalização.

## 2.6 MYSQL

O MySQL [5] é um sistema de gerenciamento banco de dados que utiliza a linguagem SQL. Além de ser uma aplicação *open source*, é um dos sistemas de bancos de dados mais populares do mundo. A escolha desta ferramenta deu-se principalmente por fatores como a baixa necessidade de recursos, a facilidade de uso, o bom desempenho e a compatibilidade com a linguagem de escolha através do MySQLdb.

## 2.7 SOUTH

South [6] é uma biblioteca de migração de bases de dados para o *framework* Django. Essa ferramenta verifica se algo foi modificado em algum dos modelos do Django e gera migrações automáticas para que os bancos de dados continuem sempre sincronizados com os modelos.

## 2.8 ASI-HTTP-Request

Trata-se de uma biblioteca que visa facilitar a comunicação de aplicações para iPhone com servidores *web* através de requisições HTTP.

## 2.9 SBJSON

O SBJSON é uma API voltada para a manipulação de objetos do tipo json (javascript object notation) em objective-C. Esta ferramenta é utilizada basicamente para codificação e decodificação de objetos json, o que é fundamental para a comunicação dos dispositivos com o servidor *web*.

## 2.10 GIT

O Git [7] é uma ferramenta de controle de versão desenvolvida pelo criador do Linux, Linus Torvalds. Entre suas principais características estão o suporte à criação e mesclagem de *branches*, o desenvolvimento descentralizado (cada desenvolvedor tem acesso a todo o projeto e ao histórico completo das mudanças), a presença de mecanismos de autenticação criptográfica e a eficiência das estratégias de mesclagem automática.

O Git é utilizado para versionar diversas ferramentas *open source*, entre elas o GitHub. Entre os principais motivos para a escolha do Git estão o fato de ele não ser pago, a facilidade de uso e a qualidade da documentação e das ferramentas de mesclagem de tarefas.

## 2.11 GITHUB

O GitHub [8] é o repositório *open source* mais popular do mundo e foi escolhido para hospedar todo o nosso projeto. Por meio de uma interface *web* excelente, todos os documentos do projeto e o histórico de mudanças podem ser diretamente acessados pelo site.

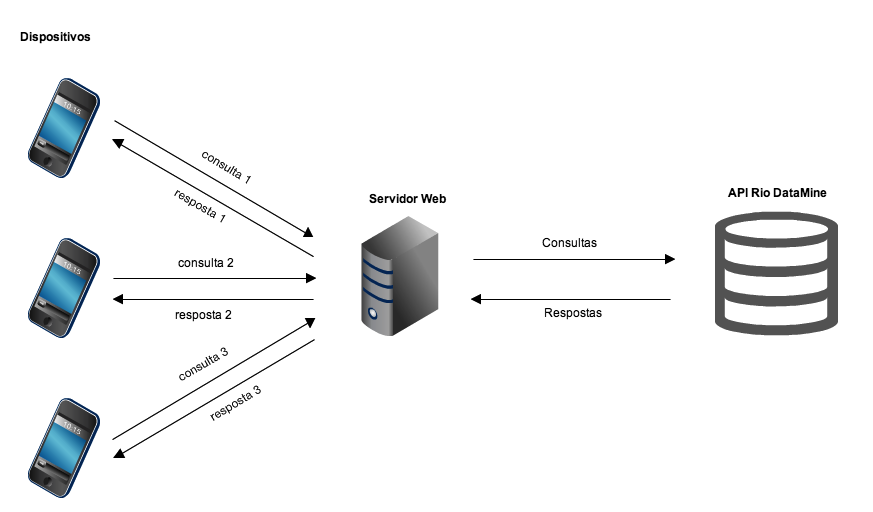
Além disso, o projeto fica guardado no servidor do GitHub e pode ser copiado para qualquer computador através do uso da API. Existem planos pagos para projetos restritos, mas nós preferimos usar o plano não pago e manter o projeto aberto para visualização de todos. O endereço do nosso projeto no GitHub é “<https://github.com/Rasinhas/PFC>”.

# 3 DESENVOLVIMENTO

Nesta seção serão apresentadas todas as etapas do desenvolvimento do sistema, incluindo a arquitetura, o modelo de banco de dados e a implementação da aplicação *mobile (*Objective-C*)* e do servidor *web* (Python).

Inicialmente foi considerada a utilização exclusiva da plataforma móvel. No entanto, mais tarde viu-se que isso não seria bom, já que sem a centralização das consultas seria impossível ter um histórico sobre a utilização ou saber como a API retornou os resultados para cada celular.

Dessa forma, o sistema foi dividido em duas partes: o dispositivo móvel, que apresenta as informações para o cliente, e o servidor *web*, que faz as consultas para a API do Rio Datamine, gerencia contas de usuários e suas preferências e mantém o histórico da utilização do sistema, segundo a imagem a seguir:



**FIG. 3.1 –** Arquitetura do Sistema

A seguir serão abordadas detalhadamente as implementações do aplicativo para o dispositivo móvel e para o servidor *web*.

## 3.1 DISPOSITIVO MÓVEL

A implementação do aplicativo para iPhone foi feita usando o xcode, que é uma IDE disponível apenas para o MAC OS da Apple. O XIB foi uma ferramenta muito importante para a construção das telas, pois ajudou no posicionamento e na customização dos elementos na tela do aplicativo.

A requisição ao servidor *web* foi feita usando a biblioteca ASIHTTPRequest, que permite o envio de requisições HTTP assíncronas. Através de uma API interna todas as consultas são realizadas diretamente através do endereço IP do servidor, que é especificado no arquivo de configuração do aplicativo. Elas seguem sempre o mesmo padrão, onde são passados os parâmetros necessários para a consulta ao Rio DataMine seguidos dos parâmetros necessários para um filtro posterior a essa consulta, que é feito no servidor web para refinar os resultados. Para montar a URL das consultas e decodificar os resultados, foi usada a biblioteca SBJSON, que faz a manipulação de objetos json em Objective-C.

Os resultados de uma busca são mostrados em formato de tabela e cada resultado pode ser visualizado em um mapa disponível através de uma aplicação interna do iPhone*,* oMapView. As coordenadas geográficas necessárias para mostrar o mapa são passadas pelo Rio DataMine e as coordenadas do posicionamento do usuário, se permitido, são obtidas no próprio aparelho.

Durante a implementação do dispositivo móvel, surgiram problemas com relação ao compilador do projeto, pois as bibliotecas estavam escritas em um padrão mais antigo que permitia uma gerência mais detalhada de memória e o resto do código foi escrito em um padrão mais novo. Para resolver isso, foi necessária a inclusão de alguns parâmetros no compilador para permitir a compilação correta das bibliotecas.

Outro problema foi a customização de certos elementos, como os botões de escolha do tipo de restaurante. Para que fosse possível a escolha de mais de um tipo, era necessário que os botões selecionados ficassem de uma cor diferente dos demais, o que não era possível em nenhum dos botões padrão oferecidos pela interface. Isso nos levou a criar um botão personalizado e a definir manualmente uma série de características de forma que ele ficasse igual ao botão padrão.

## 3.2 SERVIDOR WEB

O servidor *web* foi implementado na linguagem Python utilizando o *framework* Django. Este trabalho foi dividido em duas partes que foram feitas separadamente e depois unidas: a comunicação com a API do Rio DataMine e a criação do banco de dados usado para gerenciar as preferências dos usuários e o armazenamento do histórico.

A seguir serão abordadas detalhadamente as implementações da comunicação com a API do Rio DataMine e do banco de dados, bem como as dificuldades encontradas.

### 3.2.1 COMUNICAÇÃO COM A API

A comunicação com a API do Rio DataMine foi considerada a etapa mais difícil do desenvolvimento do sistema devido à enorme dificuldade na integração dos dados fornecidos. As respostas da API tinham formatos bem diferentes dependendo da consulta, o que exigiu um tratamento de casos extremamente detalhado, de forma a eliminar essas diferenças e apresentar uma informação mais uniforme ao usuário no iPhone*.*

Para realizar uma consulta, a API exige uma autenticação, que é verificada através de um *token* que deve ser inserido no cabeçalho de cada requisição HTTP. Como esse *token* tem uma validade de seis horas, o último *token* válido é mantido no banco de dados para uma acesso mais rápido, com a finalidade de minimizar as requisições por novos *tokens*.

Alguns filtros necessários para o sistema não são disponibilizados pela API, como faixa de preço e tipo de restaurante. Isso nos levou a criar rotinas customizadas para refinar os resultados a partir de campos extras que são passados pelo dispositivo móvel.

Outro problema foi a instabilidade na comunicação com a API. Durante a realização do projeto, por várias vezes a API ficou fora do ar, sendo algumas delas por mais de duas semanas. Isso tornou fundamental a presença de mecanismos de *backup* para que as requisições não fiquem sem respostas nessas ocasiões.

O *backup* foi feito salvando as respostas em json de algumas consultas sem filtros em arquivos de texto. Quando a comunicação com a API falha, esses arquivos são usados e os filtros que deveriam ser feitos pela API também passam a ser feitos pelo servidor *web*.

### 3.2.2 BANCO DE DADOS

Para a implementação do aplicativo foi necessária a criação de um esquema de banco de dados [9] com o principal objetivo de armazenar informações sobre os usuários e suas preferências.

A principal entidade do sistema é a entidade Usuário que contém as informações: nome, senha e e-mail.

Além deste modelo principal foram criadas outras três entidades: Uma entidade de autenticação, que armazena o *access token* do Rio Datamine e sua validade e é usada para minimizar as requisições ao servidor do Rio DataMine; uma entidade de pesquisas, que tem como objetivo armazenar os filtros usados em cada requisição; e uma entidade de Preferências, que armazena informações relativas às preferências de cada Usuário.

# 4 DOCUMENTAÇÃO

Neste capítulo serão apresentadas detalhadamente todas as funcionalidades do aplicativo. Uma descrição mais formal, contendo a especificação do sistema em casos de uso [10], pode ser encontrada no ANEXO 1 deste relatório.

O projeto é composto de duas partes fundamentais: o código embarcado no iPhone de cada cliente, que é visto como um aplicativo de celular, e o servidor *web* usado para centralizar as requisições e armazenar o histórico. Dessa forma, a documentação abrangerá tanto o aplicativo de iPhone como o servidor *web*, descrevendo as funcionalidades envolvidas e a utilização de cada uma delas.

## 4.1 USUÁRIOS

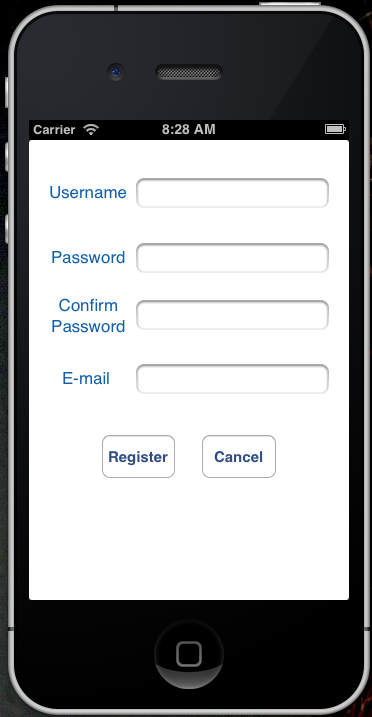
O projeto inclui a gerência de usuários, que é feita da seguinte maneira:

1. O cliente cria uma conta usando seu celular através da opção Registrar Usuário
2. O cliente faz *login* utilizando seu *username* e *password*.
3. O cliente pode modificar seus dados através da opção Editar Dados.

A seguir serão abordadas detalhadamente cada uma dessas funcionalidades.

### 4.1.1 REGISTRAR USUÁRIO

Quando o cliente obtém o aplicativo, essa é a única funcionalidade disponível. As informações necessárias são *username*, *password* e e-mail, com um campo para a confirmação do *password*, conforme mostrado na imagem a seguir:



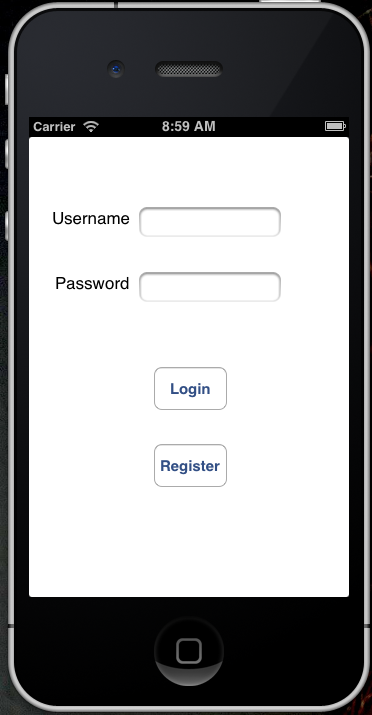
**FIG. 4.1.1.1 –** Cadastro

Assim que o botão *Register* é pressionado, o iPhone verifica apenas se todos os campos foram preenchidos e a confirmação do *password* está correta e envia uma requisição com os dados para o servidor *web*.

A requisição é validada se o campo *username* for diferente do de todos os outros usuários anteriormente cadastrados. Nesse caso, um novo usuário será criado e armazenado no banco de dados. Com a resposta da requisição do servidor, o iPhone notifica ao cliente de que o usuário foi criado, ou mostra os erros de validação que foram encontrados.

### 4.1.2 LOGIN

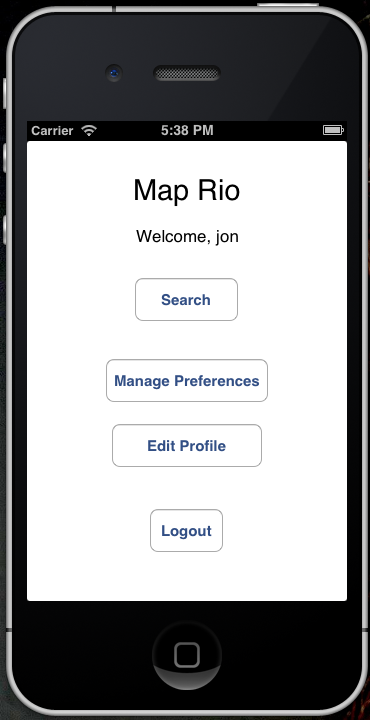
O *login* é a porta de entrada para o aplicativo e a sua execução pelo cliente permite que todas as outras funcionalidades sejam liberadas. A seguir, uma imagem com a tela de *login* no iPhone:



**FIG. 4.1.2.1 –** Login

Assim que o botão *Login* é pressionado, o celular verifica se os dados foram preenchidos e envia a requisição ao servidor, que faz uma busca no banco de dados para recuperar as informações do usuário e responde à requisição com o identificador do usuário ou uma mensagem de erro.

Assim que o aparelho receber o identificador do usuário, ele salva esse número em uma variável acessível em todo o ambiente do sistema, de forma que essa informação possa ser futuramente recuperada, e redireciona o cliente para uma tela com o menu principal.



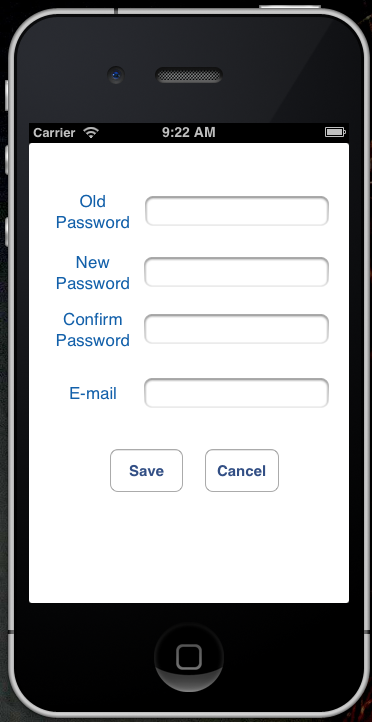
**FIG. 4.1.2.2 –** Menu principal

### 4.1.3 LOGOUT

Quando o cliente quiser desconectar seu usuário do sistema ele pode utilizar a funcionalidade de *logout*. Diretamente no menu inicial, ele aperta o botão *logout*, que funciona sem o auxílio do servidor. Assim que o botão é pressionado, o iPhone libera o identificador de usuário que estava armazenado e retorna para a tela de *login*.

### 4.1.4 EDITAR DADOS

Caso tenha o interesse de mudar alguma informação relacionada ao seu usuário, o cliente pode usar a funcionalidade *Edit Profile*, cuja tela está representada na imagem a seguir:



**FIG. 4.1.4.1 –** Edição de dados básicos

O cliente preenche os campos, o iPhone verifica se todos eles estão preenchidos e se a confirmação da senha nova está correta e envia a requisição ao servidor, acrescentando a informação do identificador do usuário, que passa a ficar disponível em todas as telas após a realização do *Login*.

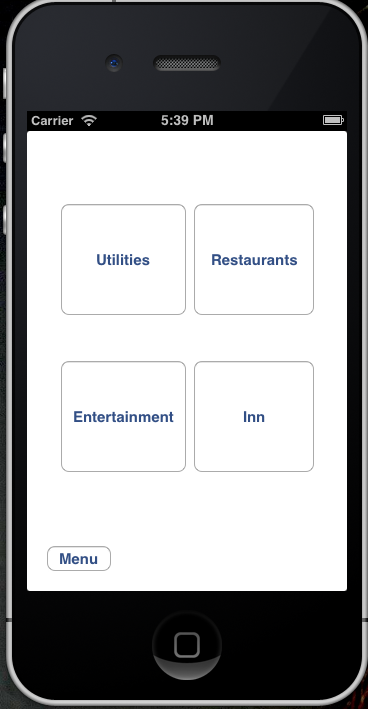
O servidor, por sua vez, verifica se a senha antiga está correta e então atualiza as informações do usuário, retornando uma mensagem de sucesso para o celular.

## 4.2 PESQUISAS

Esta é a principal funcionalidade do aplicativo e pode ser acessada através do botão *Search* localizado no menu inicial do sistema. Através desta ferramenta o usuário consegue informações sobre serviços e locais na cidade além de um mapa plotando o local de interesse e a posição do usuário. Os pontos de interesse são divididos em quatro grupos:

1. Utilidade Pública: Neste grupo o usuário consegue acessar informações sobre hospitais, delegacias e corpos de bombeiros.
2. Restaurantes: Este grupo fornece ao usuário informações sobre restaurantes na cidade.
3. Lazer: São fornecidas informações sobre eventos realizados na cidade.
4. Hospedagem: O usuário tem acesso a informações sobre hotéis e outros tipos de hospedagens na cidade.

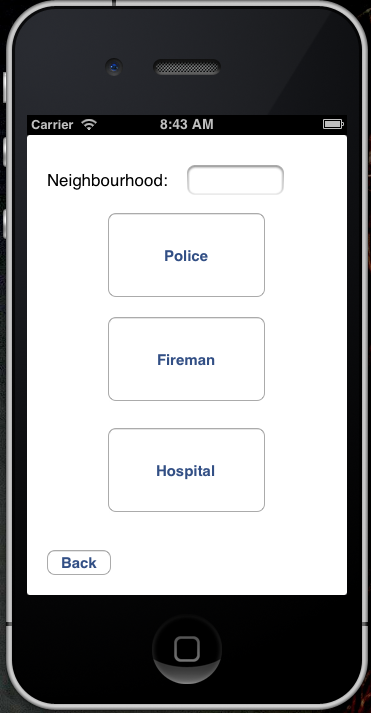
A seguir serão abordados cada um destes pontos de interesses.



**FIG. 4.2.1 –** Menu de pesquisas

### 4.2.1 UTILIDADE PÚBLICA

Caso o usuário necessite da ajuda de algum serviço de utilidade pública, o mesmo deve selecionar a opção *Utilities* no menu de pesquisas. O formulário onde deve ser preenchido o nome do bairro em que se deseja realizar a busca está representado na imagem a seguir:



**FIG. 3.2.1.1 –** Filtro de pesquisa por serviços de utilidade pública

Assim que o usuário preencher o bairro (*neighbourhood*) e clicar na opção desejada, o sistema fará uma requisição ao servidor *web* que manterá no histórico a pesquisa feita e tentará fazer uma consulta através da API do Rio DataMine. Se a ela não estiver funcionando, o servidor realizará uma busca no banco de dados local para responder à consulta do usuário.

A resposta em json será então transferida para o aplicativo, que apresentará uma lista com os resultados na tela. Na lista de resultados, serão apresentadas várias informações sobre os pontos encontrados, como telefone, nome e endereço. Além disso, o usuário também poderá clicar nos resultados para ter acesso a um mapa que indicará a posição do local selecionado e, caso seja permitido, a sua localização atual.

A seguir serão apresentadas duas imagens representando, respectivamente, a lista de resultados e o mapa:



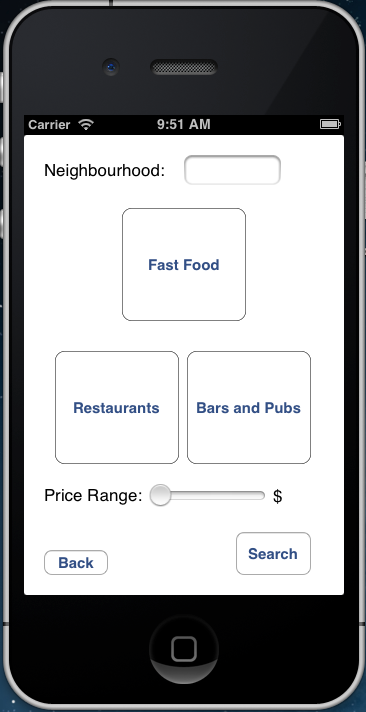
**FIG. 4.2.1.2 –** Resultados de uma pesquisa



**FIG. 4.2.1.3 –** Mapa com o resultado

### 4.2.2 RESTAURANTES

Se o usuário estiver procurando um local para se alimentar, basta selecionar a opção *Restaurants*  no menu de pesquisas. O sistema apresentará um formulário onde o usuário deverá selecionar o bairro em que deseja procurar o serviço, os tipos de restaurantes que deseja encontrar (fast food, restaurantes e bares), e também a faixa de preço que está disposto a pagar. O formulário será representado na imagem a seguir:



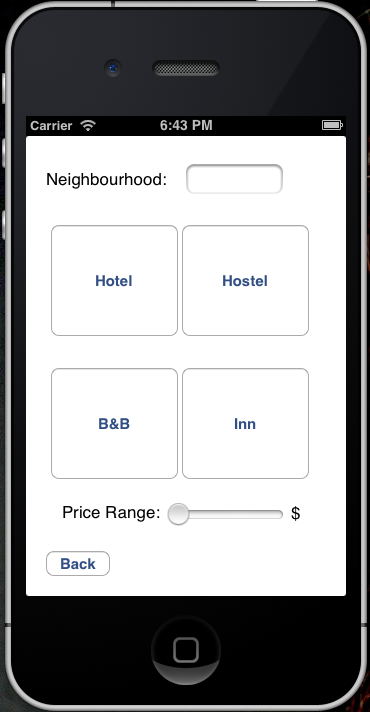
**FIG. 4.2.2.1 –** Filtro de pesquisa de restaurantes

Assim que o usuário clicar no botão *Search*, o sistema fará uma requisição ao servidor *web*, que armazenará o histórico e retornará uma lista de resultados, exatamente como na funcionalidade anterior.

A lista de resultados conta com dados como: nome, faixa de preço, endereço e uma breve descrição do estabelecimento. O usuário pode selecionar o item da lista que mais se interessou e visualizar o mapa. A seguir será apresentada uma imagem como uma possível lista de resultados:

### 4.2.3 HOSPEDAGEM

Se o usuário estiver interessado em encontrar um local para se hospedar, como um hotel ou uma pousada, poderá selecionar a opção *Inn* no menu de pesquisas. A seguir será apresentada uma imagem com o formulário da consulta:



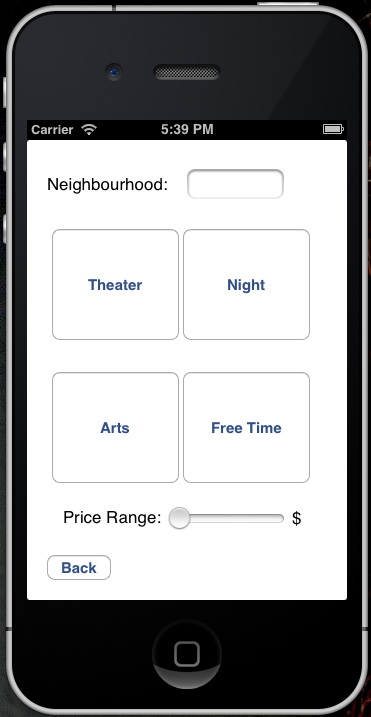
**FIG. 4.2.3.1 –** Filtro de pesquisa de hospedagens

Quando o usuário clicar no tipo de hospedagem desejado, o sistema fará uma requisição ao servidor *web* e assim como na funcionalidade de busca por serviços de utilidades públicas apresentará uma lista com os locais que atendem aos filtros especificados pelo usuário.

A lista de resultados conta com dados como: nome, endereço, faixa de preço, telefone. O usuário pode selecionar o item da lista que mais lhe interessar e visualizar o mapa.

### 4.2.4 ENTRETENIMENTO

Caso o usuário esteja interessado em realizar atividades de lazer, ele pode selecionar a opção *Entertainment* no menu de pesquisas. A imagem a seguir representa o formulário de pesquisa de atividades:



**FIG. 4.2.4.1 –** Filtro de pesquisa de entretenimento

O sistema fará uma requisição ao servidor *web* e assim como na funcionalidade de busca por serviços de utilidades públicas apresentará uma lista com os locais que atendem aos filtros especificados pelo usuário.

A lista de resultados conta com dados como: nome, endereço e data. O usuário pode selecionar o item da lista que mais lhe interessar e visualizar o mapa.

## 3.3 GERENCIAMENTO DE PREFERÊNCIAS

Esta funcionalidade tem como objetivo facilitar a utilização das opções de busca do usuário. No menu principal, o usuário poderá selecionar a opção *Manage Preferences*  para poder mudar as suas preferências de busca.

Nesta tela será apresentado um formulário onde o usuário deverá preencher os campos relativos aos grupos de busca como: *Neighbourhood* (para todos os grupos), tipos e faixa de preço para restaurantes, e tipos para eventos. Para salvar os dados, o usuário deverá pressionar o botão *save*. Os dados serão enviados ao servidor para serem salvos no banco de dados além de ser salvos no dispositivo.

Sendo assim, em qualquer momento que o usuário entrar em sua conta (em qualquer dispositivo) essas preferencias serão salvas no novo dispositivo até o *logout* do usuário.

Tendo preenchido e salvo estas informações, para todas as buscas que o usuário fizer a partir deste instante, os formulários estarão pré-preenchidos com as informações das preferencias do cliente.

# 5 CONCLUSÃO

Este trabalho é muito importante dada a situação atual da cidade do Rio de Janeiro. O Rio foi recentemente escolhido para ser sede de eventos de caráter mundial, que trazem à cidade um grande número de turistas. Dessa forma, aplicativos para ajudar na locomoção e na localização de pontos na cidade são extremamente úteis.

Foi utilizada uma grande variedade de ferramentas *open source* que foram essenciais para o desenvolvimento do trabalho. O código foi versionado através de um repositório criado no GitHub, o servidor de dados foi feito com o Django, que é um *framework* para desenvolvimento *web* extremamente completo para Python e os *frameworks* para *Objective-c*:TDBadgedCell, ASIHttpRequest e json nos ajudaram a construir o código para o aplicativo no iPhone. O trabalho agora está disponível no GitHub para qualquer pessoa que se interesse, podendo ainda ser utilizado como fonte de consulta para outros trabalhos nessa área.

As dificuldades encontradas dizem respeito principalmente à base de dados da prefeitura, o Rio Datamine. O serviço prestado fica fora do ar com muita frequência, o que nos levou a fazer um *backup* das informações em nosso servidor para não deixar o erro chegar aos usuários.

Além disso, algumas informações não tinham o nível de detalhe que nós esperávamos. Por exemplo: as informações sobre eventos esportivos nos diziam apenas os lugares que sediavam os eventos, mas não o momento que esses eventos iam acontecer. Por isso acabamos por desistir da ideia de notificar os usuários quando a data de algum evento esportivo que fosse do interesse dele estivesse próxima, o que nos fez desistir de utilizar tal informação.

Outro problema com o Rio Datamine foi a codificação usada por eles. Alguns caracteres comuns na língua portuguesa, como ‘ã’, ‘á’, ‘â’, ‘ç’ apresentam problemas de codificação, o que prejudica a leitura das informações e faz com que os usuários questionem a qualidade do aplicativo.

Para trabalhos futuros de caráter geral, seria muito interessante um trabalho para melhorar o visual do aplicativo. As telas poderiam ter mais imagens e botões personalizados que facilitassem a utilização e tornassem o aplicativo mais bonito. Além disso, as buscas poderiam ser mais inteligentes, para tratar possíveis erros de digitação, o que é comum para uma pessoa que não está ambientada com os nomes dos bairros ou expressões locais. O mapa que mostra a localização dos resultados também poderia ser melhorado, permitindo que os usuários buscassem maneiras de sair do local em que eles estão para ir para o local desejado.

# 6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] RIO DATAMINE. Disponível em <riodatamine.com.br >

Acesso em 12 de Outubro de 2012.

[2] APPLE DEVELOPER CENTER. Disponível em <developer.apple.com>

Acesso em 12 de Outubro de 2012.

[3] PYTHON. Disponível em <www.python.org>

Acesso em 14 de Março de 2013

[4] DOCS DJANGOPROJECT.

Disponível em < docs.djangoproject.com/en/1.5>

Acesso em 13 de Março de 2013.

[5] MYSQL. Disponível em < www.mysql.com>

Acesso em 14 de Março de 2013.

[6] SOUTH. Disponível em <south.readthedocs.org/en/0.7.6>

Acesso em 14 de Março de 2013.

[7] GIT. Disponível em <git-scm.com/documentation>

Acesso em 2 de Junho de 2013.

[8] GITHUB. Disponível em <github.com>

Acesso em 2 de Junho de 2013.

[9] *ELMASRI, R. and NAVATHE, S.B. Sistemas de Banco de Dados, 6a Ed., Pearson/Prentice Hall 2011.*

[10] SOMMERVILLE, I. Engenharia de Software, 9 *a* ed. Pearson Education, 2011.

# 7 ANEXOS

## 7.1 ANEXO 1

A utilização de casos de uso é uma técnica relativamente recente, mais flexível apoiado num formato novo e mais ágil para capturar requisitos de *software* que contrasta com a documentação extensiva que tenta, mas falha em registrar todos os requisitos possíveis de um sistema antes deste começar a ser construído.

Os casos de uso podem ser facilmente adicionados e removidos do projeto de software assim que as prioridades mudam. Os casos de uso podem também servir como base para estimar, escalonar e validar esforços. Uma razão porque os casos de uso se tornaram populares é que são fáceis de entender por pessoas da área de negócio, e assim provaram ser uma excelente ponte entre quem desenvolve o software e os usuários finais.

Caso de uso 1:

- Nome: Cadastrar usuário.

- Ator Primário: Usuário.

- Ator Secundário: Não há.

- Pré-condições: Não há.

- Fluxo Básico de Eventos:

1. Este caso de uso começa quando o usuário seleciona a opção cadastrar usuário.

2. O sistema apresenta um formulário de configuração.

3. O usuário preenche o formulário e confirma.

4. O sistema cadastra o usuário.

5. O sistema apresenta uma mensagem de sucesso e o caso de uso termina.

- Fluxo Alternativo de Eventos:

- OS DADOS PREENCHIDOS SAO INVÁLIDOS OU JÁ EXISTENTES

1. No passo 3 do fluxo básico de eventos, se o e-mail do usuário estiver inválido ou se os dados já pertencerem a outro usuário cadastrado.

2. O sistema apresenta uma mensagem de erro e retorna ao passo 2 do fluxo básico de eventos.

- Pós-condições: O sistema cadastra uma conta para o usuário.

- Outras informações: As informações que devem ser preenchidas pelo usuário são:

  - Nome, senha e e-mail.

Caso de uso 2:

- Nome: Efetuar login.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-Condições: Não há.

- Fluxo básico de eventos:

1. O caso de uso começa quando o usuário deseja se logar no sistema.

2. O sistema apresenta um formulário pedindo login e senha do usuário.

3. O usuário preenche os dados e confirma.

4. O sistema valida os dados.

5. O sistema redireciona o usuário para a página inicial e o caso de uso termina.

-Fluxo alternativo de eventos:

- DADOS INVÁLIDOS

1. No passo 4 do fluxo básico de eventos, se a senha ou o login do usuário não conferem.

2. O sistema retorna uma mensagem de erro e retorna ao passo 2 do fluxo básico de eventos.

- Pós-condições: O sistema está exibindo a página inicial.

Caso de uso 3:

- Nome: Alterar dados básicos.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-condições: O sistema está exibindo a página inicial.

- Fluxo Básico de eventos:

1. Este caso de uso começa quando o usuário seleciona a opção editar perfil

2. O sistema apresenta um formulário com os dados do usuário.

3. O usuário preenche o formulário e confirma.

4. O sistema armazena as novas configurações do usuário.

5. O sistema apresenta uma mensagem de sucesso e o caso de uso termina.

- Fluxo Alternativo de eventos: Não há

- Pós-condições: O sistema salvou as novas configurações do usuário

Caso de uso 4:

- Nome: Alterar senha.

- Ator Primário: Usuário.

- Ator Secundário: Não há.

- Pré-condições: O sistema está exibindo a página inicial.

- Fluxo Básico de eventos:

1. Este caso de uso começa quando o usuário seleciona a opção editar perfil

2. O sistema apresenta o formulário para troca se senha

3. O usuário preenche o formulário e confirma

4. O sistema valida os dados

5. O sistema salva a nova senha do usuário

6. O sistema apresenta uma mensagem de sucesso e o caso de uso termina

- Fluxo Alternativo de eventos:

- OS DADOS PREENCHIDOS SÃO INVÁLIDOS

1. No passo 3 do fluxo básico de eventos, se a senha antiga do usuário estiver errada ou se a senha nova  e a confirmação forem diferentes.

2. O sistema apresenta uma mensagem de erro e retorna ao passo 2 do fluxo básico de eventos.

- O USUARIO DESISTIU DE TROCAR A SENHA

1. No passo 3 do fluxo básico de eventos, se o usuário desistir de preencher o formulário.

2. O sistema apresenta uma mensagem ao usuário e o caso de uso termina.

- Pós-condições: O sistema salvou uma nova senha para o usuário.

- Outras informações: As informações que devem ser preenchidas pelo usuário são:

Senha antiga, senha nova e confirmação da senha nova.

Caso de uso 5:

- Nome: Gerenciar preferências.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-Condições: O sistema está exibindo a página inicial.

- Fluxo básico de eventos:

1. O caso de uso começa quando o usuário deseja modificar suas preferências.

2. O sistema apresenta uma lista com os tipos de preferências que podem ser alteradas e o caso de uso termina.

- Fluxo alternativo de eventos: Não há.

- Outras informações: Os tipos de preferências são:

Gerais, restaurantes, lazer e hospedagens.

- Pós-condições: O sistema está exibindo a lista de tipos de preferências.

Caso de uso 6:

- Nome: Gerenciar preferências de restaurantes.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-Condições: O sistema está exibindo a lista de tipos de preferências.

- Fluxo básico de eventos:

1. O caso de uso começa quando o usuário seleciona a opção restaurantes na lista de tipos de preferências.

2. O sistema apresenta um formulário contendo o tipo de restaurante e a faixa de preço esperada.

3. O usuário preenche o formulário e confirma.

4. O sistema salva os dados e o caso de uso termina.

- Fluxo alternativo de eventos: Não há.

- Outras informações: Os tipos de comida são:

Fast food, restaurantes e bares.

- Pós-condições: O sistema atualizou as preferências de restaurantes do usuário.

Caso de uso 7:

- Nome: Gerenciar preferências gerais.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-Condições: O sistema está exibindo a lista de tipos de preferências.

- Fluxo básico de eventos:

1. O caso de uso começa quando o usuário seleciona a opção gerais na lista de tipos de preferências.

2. O sistema apresenta um formulário contendo um campo para selecionar o bairro de preferência.

3. O usuário escolhe o bairro preferido e confirma.

4. O sistema armazena o bairro escolhido pelo usuário e o caso de uso termina.

- Fluxo alternativo de eventos: Não há.

- Pós-condições: O sistema atualizou as preferências gerais do usuário.

Caso de uso 8:

- Nome: Gerenciar preferências de lazer.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-Condições: O sistema está exibindo a lista de tipos de preferências.

- Fluxo básico de eventos:

1. O caso de uso começa quando o usuário seleciona a opção lazer na lista de tipos de preferências.

2. O sistema apresenta um formulário contendo os tipos de atividades de lazer e a faixa de preço esperada.

3. O usuário seleciona os tipos de eventos preferidos e confirma.

4. O sistema armazena os tipos de atividades escolhidos pelo usuário e o caso de uso termina.

- Fluxo alternativo de eventos: Não há

- Outras informações: Os tipos de atividades são:

Noite, teatros, artes e outras atividades.

- Pós-condições: O sistema atualizou as preferências de lazer do usuário.

Caso de uso 9:

- Nome: Gerenciar preferências de hospedagem.

- Ator primário: Usuário.

- Ator secundário: Não há.

- Pré-Condições: O sistema está exibindo a lista de tipos de preferências.

- Fluxo básico de eventos:

1. O caso de uso começa quando o usuário seleciona a opção hospedagem na lista de tipos de preferências.

2. O sistema apresenta um formulário contendo os tipos de hospedagens e a faixa de preço esperada.

3. O usuário seleciona os tipos de hospedagens prefere e confirma.

4. O sistema armazena os tipos de hospedagens selecionados pelo usuário e o caso de uso termina.

- Fluxo alternativo de eventos: Não há.

- Outras informações: Os tipos de eventos hospedagens são:

Hotéis, pousadas, albergues e cama e café.

- Pós-condições: O sistema atualizou as preferências de hospedagem do usuário.

Caso de uso 10:

- Nome: Requisitar informações de restaurantes.

- Ator Primário: Usuário.

- Ator Secundário: BD Rio DataMine.

- Pré-condições: O aparelho está exibindo a tela inicial.

- Fluxo básico de eventos:

1. Este caso de uso começa quando o usuário clica na opção requisitar informações sobre restaurantes.

2. O aparelho apresenta um formulário com uma lista de opções a serem incluídas no filtro.

3. O usuário preenche o formulário e confirma.

4. O aparelho valida o formulário.

5. O aparelho envia uma requisição ao servidor.

6. O servidor envia uma requisição ao BD RioDatamine.

7. O servidor envia os dados para o aparelho.

8. O aparelho apresenta os resultados ao usuário e o caso de uso termina

- Fluxo Alternativo de Eventos:

- O USUÁRIO PERMITE QUE O SISTEMA UTILIZE SUAS PREFERÊNCIAS

1. No passo 2 do fluxo básico de eventos, se o usuário permite que o sistema utilize suas preferências.

2. O sistema acrescenta as preferências do usuário no filtro da busca.

3. O sistema retorna ao passo 3 do fluxo básico de eventos.

- Outras informações: Os campos do filtro são:

Tipo de restaurante (restaurante, bar e *fast food*), faixa de preço e bairro.

- Pós-condições: O sistema está exibindo os resultados da busca.

Caso de uso 11:

- Nome: Requisitar informações de utilidades públicas.

- Ator Primário: Usuário.

- Ator Secundário: BD Rio DataMine.

- Pré-condições: O aparelho está exibindo a tela inicial.

- Fluxo básico de eventos:

1. Este caso de uso começa quando o usuário clica na opção requisitar informações sobre utilidades públicas.

2. O aparelho apresenta um formulário com uma lista de opções a serem incluídas no filtro.

3. O usuário preenche o formulário e confirma.

4. O aparelho valida o formulário.

5. O aparelho envia uma requisição ao servidor.

6. O servidor envia uma requisição ao BD RioDatamine.

7. O servidor envia os dados para o aparelho.

8. O aparelho apresenta os resultados ao usuário e o caso de uso termina

- Fluxo Alternativo de Eventos:

- O USUÁRIO PERMITE QUE O SISTEMA UTILIZE SUAS PREFERÊNCIAS

1. No passo 2 do fluxo básico de eventos, se o usuário permite que o sistema utilize suas preferências.

2. O sistema acrescenta as preferências do usuário no filtro da busca.

3. O sistema retorna ao passo 3 do fluxo básico de eventos.

- Outras informações: Os campos do filtro são:

Tipo (delegacias de polícia, bombeiros e hospitais) e bairro.

- Pós-condições: O sistema está exibindo os resultados da busca.

Caso de uso 12:

- Nome: Requisitar informações de lazer.

- Ator Primário: Usuário.

- Ator Secundário: BD Rio DataMine.

- Pré-condições: O aparelho está exibindo a tela inicial.

- Fluxo básico de eventos:

1. Este caso de uso começa quando o usuário clica na opção requisitar informações sobre lazer.

2. O aparelho apresenta um formulário com uma lista de opções a serem incluídas no filtro.

3. O usuário preenche o formulário e confirma.

4. O aparelho valida o formulário.

5. O aparelho envia uma requisição ao servidor.

6. O servidor envia uma requisição ao BD RioDatamine.

7. O servidor envia os dados para o aparelho.

8. O aparelho apresenta os resultados ao usuário e o caso de uso termina

- Fluxo Alternativo de Eventos:

- O USUÁRIO PERMITE QUE O SISTEMA UTILIZE SUAS PREFERÊNCIAS

1. No passo 2 do fluxo básico de eventos, se o usuário permite que o sistema utilize suas preferências.

2. O sistema acrescenta as preferências do usuário no filtro da busca.

3. O sistema retorna ao passo 3 do fluxo básico de eventos.

- Outras informações: Os campos do filtro são:

Tipo (noite, teatro, artes e tempo livre), faixa de preço e bairro.

- Pós-condições: O sistema está exibindo os resultados da busca.

Caso de uso 13:

- Nome: Requisitar informações de hotéis.

- Ator Primário: Usuário.

- Ator Secundário: BD Rio DataMine.

- Pré-condições: O aparelho está exibindo a tela inicial.

- Fluxo básico de eventos:

1. Este caso de uso começa quando o usuário clica na opção requisitar informações sobre hotéis.

2. O aparelho apresenta um formulário com uma lista de opções a serem incluídas no filtro.

3. O usuário preenche o formulário e confirma.

4. O aparelho valida o formulário.

5. O aparelho envia uma requisição ao servidor.

6. O servidor envia uma requisição ao BD RioDatamine.

7. O servidor envia os dados para o aparelho.

8. O aparelho apresenta os resultados ao usuário e o caso de uso termina

- Fluxo Alternativo de Eventos:

- O USUÁRIO PERMITE QUE O SISTEMA UTILIZE SUAS PREFERÊNCIAS

1. No passo 2 do fluxo básico de eventos, se o usuário permite que o sistema utilize suas preferências.

2. O sistema acrescenta as preferências do usuário no filtro da busca.

3. O sistema retorna ao passo 3 do fluxo básico de eventos.

- Outras informações: Os campos do filtro são:

Tipo (hotel, pousada, albergue e “cama e café”), faixa de preço e bairro.

* Pós-condições: O sistema está exibindo os resultados da busca.

Caso de uso 14:

- Nome: Visualizar mapa com resultados.

- Ator Primário: Usuário.

- Ator Secundário: Não há.

- Pré-condições: O aparelho está exibindo a tela de resultados de uma busca.

- Fluxo básico de eventos:

1. Este caso de uso começa quando o usuário clica em um resultado da busca realizada.

2. O aparelho apresenta um mapa contendo a posição do local de escolhido e o caso de uso termina.

- Fluxo Alternativo de Eventos:

- O USUÁRIO PERMITE QUE O SISTEMA UTILIZE SUA LOCALIZAÇÀO GEOGRÁFICA

1. No passo 2 do fluxo básico de eventos, se o usuário permite que o sistema utilize suas preferências, o sistema evidencia a posição do usuário no mapa.

Pós-condições: O sistema está exibindo o mapa.