UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUIZ ALEXANDRE ZAPCHAU

APLICATIVO ANDROID PARA BUSCA DE LOCAIS DE VENDA DE ALIMENTOS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

LUIZ ZALEXANDRE ZAPCHAU

APLICATIVO ANDROID PARA BUSCA DE LOCAIS DE VENDA DE ALIMENTOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO

2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e
Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

APLICATIVO ANDROID PARA BUSCA DE LOCAIS DE VENDA DE ALIMENTOS

por

LUIZ ALEXANDRE ZAPCHAU

Este trabalho de conclusão de curso foi apresentado no dia 24 de novembro de 2016, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Banca examinadora:

Profª. Drª. Beatriz Terezinha Borsoi
Orientador

Profª. Drª. Kathya Silvia Collazos Linares

Prof. Me. Robison Cris Brito

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª. Me. Soelaine Rodrigues Ascari
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

AGRADECIMENTOS

Aos meus pais Elton e Sonia e a minha irmã Vanessa, pelo apoio, pela orientação, dedicação, compreensão e incentivo durante toda minha vida e pelo apoio durante todas as minhas escolhas de curso e que incluíram várias tentativas até encontrar o presente curso. Minha eterna gratidão.

A UTFPR e a todos os professores, pela formação profissional e ética.

Agradeço a todas as amizades que fiz neste longo período na Instituição, por todos os momentos compartilhados e pela troca de experiências.

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it.

Steve Jobs

RESUMO

ZAPCHAU, Luiz Alexandre. Aplicativo Android para busca de locais de venda de alimentos. 2016. 66f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2016.

A utilização de aplicativos online com finalidade de fornecer buscas de estabelecimentos que forneçam serviços específicos é cada vez maior. Levando em consideração esse fato, constatou-se a oportunidade de oferecimento de um aplicativo mais dinâmico para esse tipo de busca, mais especificamente para o segmento de alimentação. Este trabalho tem como objetivo desenvolver um aplicativo móvel, para o sistema operacional Android para auxiliar na localização de estabelecimentos alimentícios e seus respectivos cardápios de forma mais dinâmica, exibindo fotos e detalhando os pratos com seus ingredientes. O detalhamento dos principais ingredientes dos pratos visa auxiliar na escolha do alimento e respectivo estabelecimento, bem como fornecer uma alternativa para as pessoas que são intolerantes a determinados ingredientes ou não podem ou não querem ingerir ingredientes específicos. E, mesmo, para pessoas que buscam pratos com determinados ingredientes. Esse aplicativo poderá ser utilizado tanto por consumidores quanto por estabelecimentos para encontrar ou disponibilizar tais produtos.

Palavras-chave: Android. Aplicativo Móvel. Aplicativo de busca de locais de alimentação.

ABSTRACT

ZAPCHAU, Luiz Alexandre. Android Application for search food sales establishments. 2016. 66f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

The use of online applications that provide search results of establishments which provide a specific service is growing exponentially today. Taking into consideration this fact it became clear the relevance of a more dynamic application for one of its goals, the food commercialization. This paper has as goal the development of a mobile application for Android devices, which would be able to find food sales points as restaurants, as well as its menus using a more dynamic approach, showing pictures and a more detailed explanation of the plates with its ingredients. The detailing of the main ingredients aims to help the plate choice as well as provide an alternative for people that are intolerant to specific ingredients or are not able to or do not want to eat specific ingredients. Also for people who do seek plates with specific ingredients. This application would be used by consumers and by providers as well, to find or make available these products.

Keywords: Android. Mobile Application. Application for Search of Food Sales Points.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGOS

LISTA DE SIGLAS

API	*Application Program Interface*
IDE	Integrated Development Environment
GPS	*Global Positioning System*
GSM	*Global System for Mobile Communications*
XML	Ex*tensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa do trabalho. No final está a organização do texto por meio de uma breve apresentação dos seus capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

O desenvolvimento tecnológico visa, tem entre outros objetivos, o de facilitar tarefas comuns realizadas no cotidiano das pessoas. Controle de agenda pessoal, de contas a pagar e localização de locais para realizar refeições são exemplos de atividades rotineiras e que têm sido suportadas pelos avanços dos recursos tecnológicos. Aplicativos para dispositivos móveis que fazem o uso de técnicas de Inteligência Artificial e de recursos como *Global Positioning System* (GPS) têm se tornado comuns como auxiliares nas atividades do dia a dia das pessoas, como profissionais ou cidadãos. Essas atividades incluem trabalho, tarefas domésticas e entretenimento.

Atualmente, a tecnologia móvel vem, notoriamente, crescendo, popularizando-se e tornando o ser humano cada vez mais dependente dos recursos e facilidades que ela oferece. A tecnologia móvel, por meio de dispositivos como *tablets* e *smartphones*, pode ser aplicada para facilitar e melhorar a realização de muitas atividades praticadas diariamente. Alcântara e Vieira (2011) destacam que,

> "A cada dia, um número maior de pessoas interessa-se pela mobilidade, o fácil acesso às informações em qualquer lugar, com alcance amplo a qualquer hora, se conectando de forma fácil e rápida a outros dispositivos móveis, localizando pessoas, produtos e serviços personalizados" (ALCANTARA; VIEIRA p. 2, 2011).

Mesmo que os dados estando disponíveis aos usuários por meio de dispositivos móveis, facilitando o acesso, eles precisam, muitas vezes, ser compartilhados entre os usuários. Para isso, esses dados precisam estar centralizados em servidores, que farão o compartilhamento entre os usuários. Para que isso seja possível é necessário que os servidores estejam disponíveis via *web* e

um dos meios de executar esse tipo de operação é pela utilização de computação em nuvem.

> “A Computação em Nuvem torna possível o acesso de arquivos, aplicativos e serviços disponíveis na web por diversos usuários simultaneamente e de diversos tipos de aparelhos que tenham acesso à Internet e navegadores compatíveis com tais serviços e aplicações” (VANDERSEN; MAGALHÃES, p. 2, 2013).

Este trabalho busca desenvolver um aplicativo para o Sistema Operacional Móvel Android, utilizando computação em nuvem e codificação na linguagem de programação Java e a *Integrated Development Environment* (IDE) de programação Android Studio para auxiliar em uma atividade bastante rotineira para muitas pessoas: a busca de locais de alimentação.

1.2 OBJETIVOS

O objetivo geral remete ao resultado final a ser obtido com a realização deste trabalho e os objetivos específicos o complementam explicitando as funcionalidades do aplicativo desenvolvido.

1.2.1 Objetivo Geral

Desenvolver um aplicativo para o Sistema Operacional Android, que visa facilitar a busca por locais de alimentação contendo pratos ou ingredientes específicos em estabelecimentos cadastrados.

1.2.2 Objetivos Específicos

- Oferecer filtros na aplicação Android visando possibilitar ao usuário encontrar locais com pratos ou ingredientes específicos;

- Ajudar consumidores a encontrar determinado local utilizando sistema de rotas via recurso externo (Google Maps);
- Permitir ao usuário a possibilidade de adicionar comentários aos pratos oferecidos por estabelecimentos cadastrados.

1.3 JUSTIFICATIVA

O aplicativo que é resultado da realização deste trabalho foi desenvolvido utilizando um sistema de banco de dados com armazenamento em nuvem. A disponibilização de dados em nuvem facilita o acesso pelo usuário e consequentemente o auxilia na tomada de decisão no momento de consumir alimentos fora de sua residência.

O resultado do desenvolvimento do trabalho consiste de um aplicativo para o Sistema Operacional móvel Android. A escolha dessa tecnologia decorre do fato de o mesmo conter maior popularidade entre 101 países, segundo o *website* Proxxima:

> [...]é o sistema operacional mais popular no mundo, com predominância em 67 países, enquanto a utilização do iOS é predominante em 34 nações. É o que mostra o estudo realizado pela empresa de tecnologia móvel dotMobi em 101 países. Em vez do número de vendas, os dados da pesquisa foram obtidos com a ferramenta goMobi da empresa, que monitora pageviews de dispositivos móveis (PROXXIMA, p. s.n, 2014).

O aplicativo desenvolvido auxiliará nas dificuldades de encontrar estabelecimentos como restaurantes, bares, lanchonetes que apresentem cardápios diversificados ou específicos, em uma determinada cidade ou localização delimitada pelo usuário. Os estabelecimentos poderão cadastrar seus cardápios para ofertar pratos com fotos e descrições de cada produto. Os clientes poderão buscar tais pratos por meio de filtros fornecidos pelo aplicativo, por local, ou utilizando busca por nome do estabelecimento, nome do prato ou ingrediente, por exemplo.

1.4 ESTRUTURA DO TRABALHO

A sequência deste texto está organizada em capítulos. O Capítulo 2 apresenta o referencial teórico que se refere às aplicações para dispositivos móveis. No Capítulo 3 são apresentados os materiais e o método utilizados para a modelagem e a implementação do sistema. O resultado da realização deste trabalho é apresentado no Capítulo 4. No Capítulo 5 estão as considerações finais, seguidas das referências utilizadas no texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico que fundamenta a aplicação desenvolvida que é para o Sistema Operacional Android.

2.1 HISTÓRICO

Em 2005 a *Google* adquiriu uma pequena empresa de Palo Alto na California, conhecida como Android Inc. A empresa desenvolvia aplicativos para celular. Com a compra, a Google começou o desenvolvimento do Android, um Sistema Operacional gratuito baseado em *Linux* (UOL, p. s.n., 2016).

O primeiro dispositivo Android a ser produzido foi o HTC G1, que teve suas vendas iniciadas em 2008 nos Estados Unidos. Chegando ao Brasil um ano depois, equipando o modelo HTC Magic (UOL, p. s.n., 2016).

Dados da Android Developer (p. s.n., 2016) indicam que a partir de então, o Android começou a popularizar-se e hoje está presente em mais de 80% dos dispositivos móveis do mercado. Android é atualmente a plataforma móvel mais popular, sendo o sistema operacional encontrado em centenas de milhares de dispositivos e em mais de 190 países. É estimado que a cada dia surge um milhão de novos usuários Android que procuram jogos, aplicativos e outros conteúdos digitais.

Em 2014, o sistema operacional Android detinha 84,7% do mercado de dispositivos móveis (HAMANN, 2014). Em decorrência do seu elevado número de parceiros nas áreas de software, hardware e operadoras, o Android tornou-se o Sistema Operacional móvel com o crescimento mais rápido do mercado.

2.2 DEFINIÇÃO

A plataforma Android é uma pilha (conjunto de camadas) de software para dispositivos móveis que consiste de um Sistema Operacional, *middleware* e aplicações chave (ANDROID, 2016). Android é um sistema operacional visto como

um conjunto de serviços de software especialmente desenvolvidos para dispositivos móveis (GUANA et al., 2012).

Maji et al. (2010) citam diferentes características das áreas de desenvolvimento de aplicações, Internet, mídias e conectividade que caracterizam os aplicativos Android. Essas características incluem *framework* de aplicação, máquina virtual Dalvik, navegador integrado, gráficos otimizados, SQLite para armazenamento de dados estruturados, suporte a mídia para áudio, vídeo, formatos de imagens, telefonia, *Global System for Mobile Communications* (GSM), bluetooth, 3G, wifi, câmera, GPS e ambiente de desenvolvimento rico.

2.3 ARQUITETURA DO ANDROID

A arquitetura da plataforma Android é baseada no *kernel* do Linux, que funciona como uma camada de abstração entre o *hardware* e os aplicativos que executam na plataforma. Sua arquitetura é baseada em cinco camadas com funcionalidades e comportamentos específicos (GUANA et al. 2012). Essas camadas são: aplicações, *framework*, bibliotecas, *runtime* e *kernel* Linux.

Na plataforma Android, as aplicações são escritas em Java e executadas em sua própria máquina virtual, a Dalvik, que por sua vez é executada em seu próprio processo no Linux, isolando-a de outras aplicações e facilitando o controle de recursos (ANDROID DEVELOPER, 2012).

A Figura 1 apresenta a organização da arquitetura do Android.

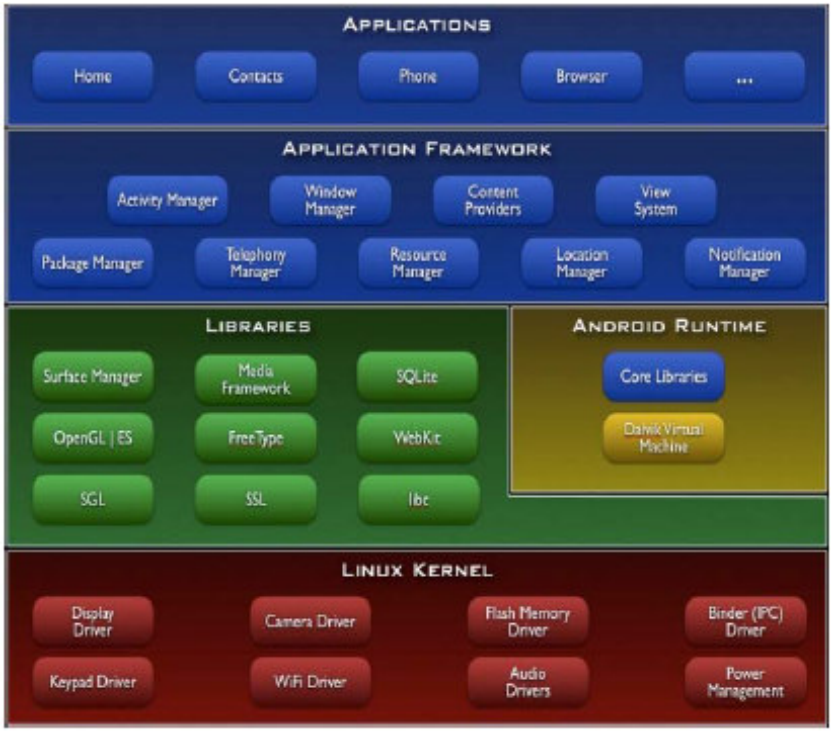

Figura 1 – Arquitetura do sistema operacional Android
Fonte: Android Developer (2016, p.s.n.).

De acordo com a representação da Figura 1 (GUANA et al., 2012, MAJI et al., 2010):

a) *Applications* – a camada de aplicação consiste de um conjunto de aplicações como clientes de *emails*, aplicativos de mensagens, calendário, navegador de mapa, navegadores os jogos que estendem as funcionalidades do sistema operacional. Essas aplicações vêm junto com a distribuição do Android;

b) Application framework – essa camada visa facilitar o reuso de componentes no Android. Com a ajuda dos elementos desse *framework*, tais como *intents, content providers, views* e *managers*, os desenvolvedores podem construir aplicações para executar no *kernel* Android e interoperar entre si e com outras aplicações;

c) *Libraries* – a camada de bibliotecas configura um conjunto de pacotes C/C++ usados pelo *framework* de aplicação para gerenciar a renderização da interface, segurança de dispositivos, persistência de aplicações e outros. Essa camada inclui bibliotecas C do sistema, gerenciadores de

interface, *engine* gráfica 2D e 3D, *codecs* de mídia, a base de dados SQLite e a *engine* LibWebCore do navegador web;

d) Android Runtime – essa camada é composta pela máquina virtual Dalvik e pelas bibliotecas básicas Android que especificam o ambiente de execução da aplicação dentro do sistema operacional. As bibliotecas básicas provem a maioria das funcionalidades disponíveis em Java. Dalvik opera como um tradutor entre a aplicação e o Sistema Operacional. Toda aplicação Android executa em seu próprio processo, com sua própria instância da máquina virtual Dalvik;

e) Linux *kernel* – é um *kernel* Linux customizado usado para prover as funcionalidades de baixo nível do sistema operacional tais como gerenciamento de memória e agendamento de processos.

2.4 VERSÕES DO ANDROID

Desde a versão 1.6 lançada em 2009, a equipe responsável pelo desenvolvimento do Sistema Operacional Android, vem nomeando as suas versões com nomes de sobremesas. As versões para as quais há suporte são as lançadas a partir de 2010, a versão 2.2, conhecida como Froyo. O Quadro 1 contém informações sobre as principais versões do Android lançadas até o momento.

Versão	**Codinome**	***Application Program Interface* (API)**
2.2	Froyo	8
2.3.3 - 2.3.7	Gingerbread	10
4.0.3 - 4.0.4	Ice Cream Sandwich	15
4.1.x	Jelly Bean	16
4.2.x		17
4.3		18
4.4	KitKat	19
5.0	Lollipop	21
5.1		22
6.0	Marshmallow	23
7,0	Nougat	24

Quadro 1 – Versões do Android e distribuição
Fonte: Platform Versions (ANDROID DEVELOPER, 2016, p. s. n.).

3 MATERIAIS E MÉTODO

Este capítulo apresenta as ferramentas e as tecnologias e o método utilizados para a modelagem e a implementação da solução proposta, o aplicativo desenvolvido para atender os objetivos definidos.

3.1 MATERIAIS

As ferramentas e as tecnologias constantes do Quadro 2 foram utilizadas para desenvolver a modelagem e a implementação do aplicativo.

Ferramenta/ Tecnologia	Versão	Referência	Finalidade
MySQL Workbench	9.4	http://dev.mysql.com/doc/workbench/en/	Modelagem do banco de dados.
DataGrip	2016.2	https://www.jetbrains.com/datagrip/	Construção e manipulação do banco de dados.
Visual Paradigm	13.1	https://www.visual-paradigm.com/	Modelagem do sistema.
Android Studio	2.2	https://developer.android.com/studio	IDE de desenvolvimento Android.
SQLite	3.14.2	https://www.sqlite.org/	Armazenar informações específicas no dispositivo Android.
MySQL	5.7	https://www.mysql.com/	Banco de dados para armazenamento de informações gerais.
Amazon RDS	2016	https://aws.amazon.com/rds/	Fornecer local de armazenamento e Web Service para a conexão com o banco de dados (nuvem).
Material design		https://material.google.com/	Linguagem de design da Google.

Quadro 2 – Ferramentas e tecnologias

O aplicativo foi desenvolvido utilizando a interface Android Studio desenvolvida pela Google que de acordo com o site Tecmundo " [...] é um Ambiente de Desenvolvimento Integrado feito para facilitar a vida de quem quer desenvolver aplicativos para a plataforma móvel da Gigante das Buscas. " (TECMUNDO, p. s.n, 2014).

3.2 MÉTODO

A modelagem e a implementação do sistema têm como base o modelo sequencial linear, também conhecido como modelo cascata. Esse modelo foi introduzido em 1970 por Royce e é o modelo mais antigo utilizado na Engenharia de Software. Esse modelo sofreu diversos ajustes (MACORATTI, 2016) e é composto pelas etapas representadas na Figura 2.

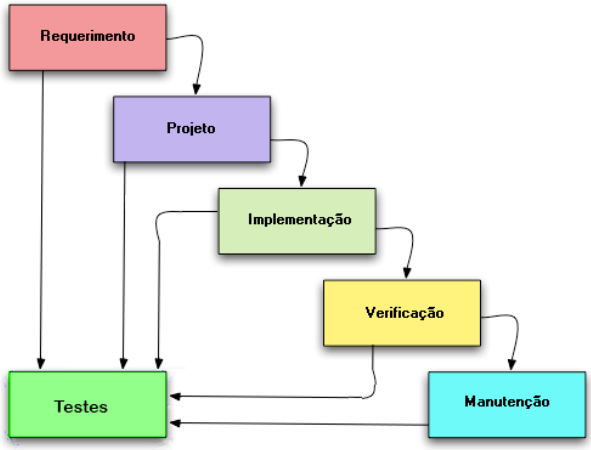

Figura 2 – Etapas do modelo cascata
Fonte: baseado em Infoccomp (2016, p.s.n).

A seguir estão descritas as etapas definidas para o desenvolvimento do aplicativo.

a) Análise de requisitos (Requerimento)

Na fase inicial da análise foram realizadas conversas informais com consumidores para determinar as expectativas deles em relação à proposta de sistema. Essas pessoas sustentaram a importância desse tipo de um aplicativo pela dificuldade que as pessoas podem ter para encontrar novos locais de alimentação ou de conhecer os diversos locais desse tipo que uma cidade ou região possui. E, ainda, os locais que oferecem os alimentos desejados.

Em seguida, foram pesquisados vários artigos acadêmicos e sites que abordam o assunto de buscas utilizando dispositivos móveis, visando entendimento e subsídios para definir o projeto do aplicativo.

Iniciou-se então o levantamento de requisitos para o aplicativo visando facilitar ao consumidor a localização de estabelecimentos e pratos desejados para

seu consumo, tornando possível a busca por ingredientes, para que o consumidor possa encontrar exatamente o que quer consumir ou evitar consumo.

Com base nas informações coletadas e nos requisitos estabelecidos foram definidos os casos de uso do sistema, que foram ajustados e complementados à medida que a modelagem do sistema era realizada.

b) Projeto

A partir dos casos de uso, foi definido utilizada a ferramenta MySQL Workbench o diagrama de entidades e relacionamentos para a modelagem do banco de dados contendo as informações sobre estabelecimentos, pratos e demais informações essenciais para atendimento à funcionalidade do aplicativo. Neste diagrama, foram definidos tipo dos campos (como numérico, texto e data), tamanho e funcionalidade.

Esse diagrama também foi alterado várias vezes durante o processo, sempre visando melhorar a eficácia do aplicativo e atender necessidades mais específicas.

c) Implementação

A fase de desenvolvimento começou com a implementação do banco de dados. A partir do modelo de entidade e relacionamento criado na fase de projeto, foi então desenvolvido o banco de dados utilizando a tecnologia MySQL como sistema de banco de dados e o software DataGrip para a construção e a manipulação deste banco via codificação.

Com o banco pronto, utilizou-se o diagrama de casos de uso como base para o desenvolvimento do aplicativo. O desenvolvimento do aplicativo foi realizado com a utilização da IDE de desenvolvimento Android Studio.

A implementação foi o processo mais longo deste projeto e contou com a criação de mais de quarenta classes para que o sistema funcionasse de maneira a atender aos objetivos. Essas classes têm como objetivo receber e processar informações inseridas pelo usuário, fazer a pesquisa no banco e devolver dados aos usuários bem como armazenar dados no banco de dados para possibilitar futuras consultas.

O desenvolvimento da interface do aplicativo teve como base as diretrizes de *design* do Material Design da Google.

d) Verificação

Os testes foram informais, realizados durante a implementação e sem um plano ou planejamento definido. Esses testes visaram encontrar erros de codificação, verificar as validações definidas como necessárias e o atendimento aos requisitos.

e) Manutenção

Considerando que o sistema não foi colocado em produção, ou seja, implantado para uso, não há manutenção.

4 RESULTADOS

Este capítulo apresenta o resultado deste trabalho que é a modelagem e o desenvolvimento de um aplicativo para busca de alimentos e seus respectivos estabelecimentos.

4.1 ESCOPO DO SISTEMA

O sistema permite o cadastro de estabelecimentos e os pratos que oferecem com os respectivos ingredientes principais. A busca poderá ser realizada por pratos que contenham ou não determinado ingrediente. O aplicativo permitirá a utilização de rotas para encontrar os locais que oferecem tais alimentos utilizando recursos do site Google Maps.

Poderão se conectar ao sistema, consumidores e fornecedores de alimentos. No caso de consumidores, poderão além da busca, fazer *check-in* no estabelecimento publicando suas atividades em determinadas redes sociais. O consumidor também terá a possibilidade de avaliar e fazer comentários sobre os produtos consumidos. Acessando como fornecedor, o usuário pode adicionar pratos ao estabelecimento, adicionar fotos e ingredientes aos pratos, pode visualizar os comentários sobre pratos e consultar os pratos mais vendidos de seu estabelecimento, por exemplo.

4.2 MODELAGEM DO SISTEMA

O Quadro 3 apresenta os requisitos funcionais identificados para o sistema.

Identificação	Nome	Responsável	Descrição
RF01	Cadastrar estabelecimento	Fornecedor	O cadastro é composto basicamente por nome, endereço, descrição, foto e telefone.
RF02	Cadastrar prato	Fornecedor	O cadastro de prato contém nome, descrição, foto, quantidade de pessoas que o prato serve, preço, um campo de observação e relação com o estabelecimento.

RF03	Cadastrar ingredientes	Fornecedor	Os ingredientes são utilizados para serem relacionados aos pratos.
RF04	Buscar prato	Usuário	A busca de pratos é feita por meio de seu nome.
RF05	Buscar estabelecimento	Usuário	A busca de estabelecimento é feita por nome ou endereço e pode haver uma rota que é obtida por meio de recursos do site Google Maps.
RF06	Buscar ingrediente	Usuário	Os ingredientes podem ser encontrados por nome.
RF07	Avaliar prato	Usuário	Os pratos são avaliados pelos consumidores por meio de nota e/ou comentário.

Quadro 3 – Requisitos funcionais

O diagrama de casos de uso apresentado na Figura 2, exibe as principais funcionalidades do sistema e seus dois atores: cliente e fornecedor.

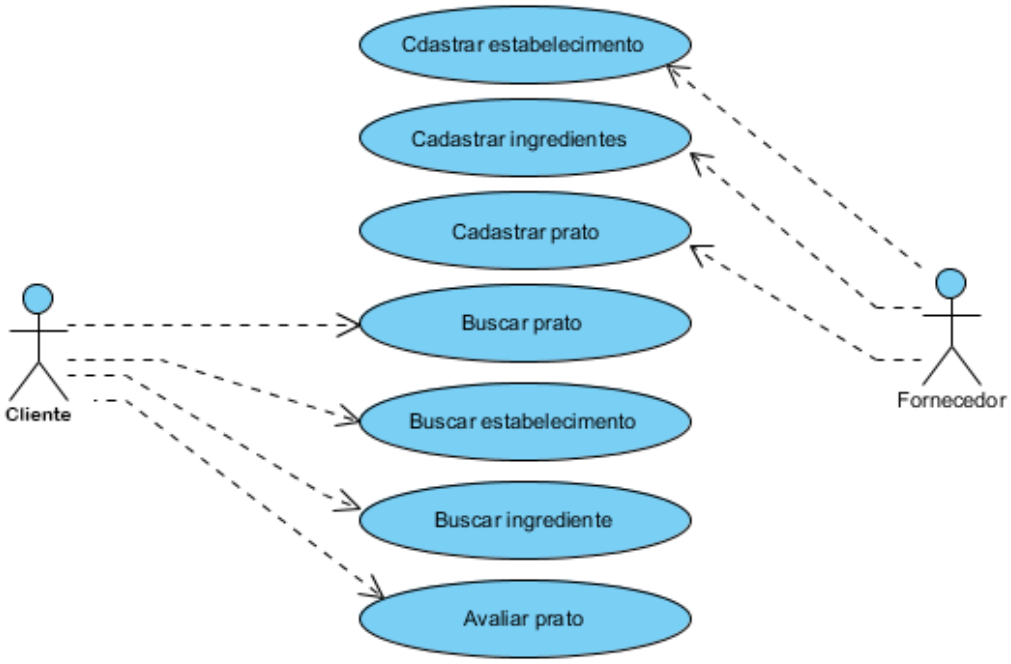

Figura 3 – Diagrama de casos de uso
Fonte: Elaborada pelo autor

O Quadro 4 descreve as operações de cadastrar estabelecimentos, ingredientes e pratos. Tais cadastros se referem às tabelas *establishment*, *ingredient* e *plate* da Figura 4, respectivamente.

Caso de uso	Cadastrar estabelecimento, ingredientes e pratos.
Descrição	O fornecedor cadastrará seu estabelecimento, os respectivos pratos contendo os respectivos ingredientes principais.
Evento indicador	Acessar o menu de cadastro de estabelecimento, prato ou ingrediente.
Atores	Fornecedor.
Pré-condição	Ter o aplicativo instalado em seu dispositivo.
Sequência de eventos	1 – O fornecedor acessa o menu de cadastro de estabelecimento, ingrediente ou prato.
	2 – O fornecedor insere os dados necessários.
	3 – Em seguida pressiona o botão salvar.
Pós-condição	Dados do cadastro inseridos no banco.
Fluxo alternativo	**Descrição**
3 Dados inválidos	3.1 – No momento da inclusão dos dados, o sistema faz a validação dos dados e busca inconsistências. É então exibida uma mensagem, especificando a inconsistência e o formulário permanece disponível para alteração.

Quadro 4 – Operação de inclusão de estabelecimento, ingrediente ou prato

Quadro 5 descreve o processo de buscar pratos, ingredientes ou estabelecimentos. Tais buscas são efetuadas nas tabelas *plate*, *ingredient* e *place* respectivamente. Tais tabelas são exibidas na Figura 4.

Caso de uso	Buscar prato, ingrediente ou estabelecimento.
Descrição	O consumidor buscará por pratos, ingredientes ou estabelecimentos por nome. Pratos podem estar contidos em qualquer estabelecimento. Ingredientes podem estar contidos em quaisquer pratos. E estabelecimentos podem estar contidos em local especifico se tal opção for selecionada.
Evento indicador	Acessar a busca de pratos, ingredientes ou estabelecimentos.
Atores	Consumidor.
Pré-condição	Ter o aplicativo instalado em seu dispositivo.
Sequência de eventos	1 – O consumidor acessa a busca por pratos, ingredientes ou estabelecimentos.
	2 – O consumidor insere os dados necessários.
	3 – Em seguida pressiona o botão buscar.
Pós-condição	Pratos, ingredientes ou estabelecimentos condizentes com os dados de busca serão exibidos, desde que haja registros correspondentes aos critérios de busca no banco de dados.
Fluxo alternativo	**Descrição**
3 Dados inválidos	3.1 – No momento da busca, caso não haja pratos, ingredientes ou estabelecimentos que atendam os critérios de busca, será exibida uma mensagem informativa para o consumidor. Opção de busca continuará disponível.

Quadro 5 – Operação de busca por prato, ingrediente ou estabelecimento

A descrição do processo de avaliação de pratos é apresentada no Quadro 6.

Caso de uso	Avaliar Prato
Descrição	O consumidor poderá avaliar pratos consumidos por ele em determinado estabelecimento.
Evento indicador	Acessar o menu de avaliação de pratos.
Atores	Consumidor.
Pré-condição	Ter o aplicativo instalado em seu dispositivo.
Sequência de eventos	1 – O consumidor acessa o menu de avaliação de pratos.
	2 – O consumidor insere uma nota a ser atribuída ao prato no estabelecimento em que foi consumido.
	3 – Em seguida faz um comentário sobre o prato consumido.
	4 – Na sequência, o consumidor pressiona o botão enviar avaliação.
Pós-condição	A avaliação será exibida para outros consumidores que buscarem o mesmo prato no mesmo estabelecimento.

Quadro 6 – Operação avaliar prato

A Figura 4 apresenta o diagrama de entidades e relacionamentos que representam a estrutura do banco de dados do sistema. Essa modelagem será aplicada ao Microsoft SQL Server.

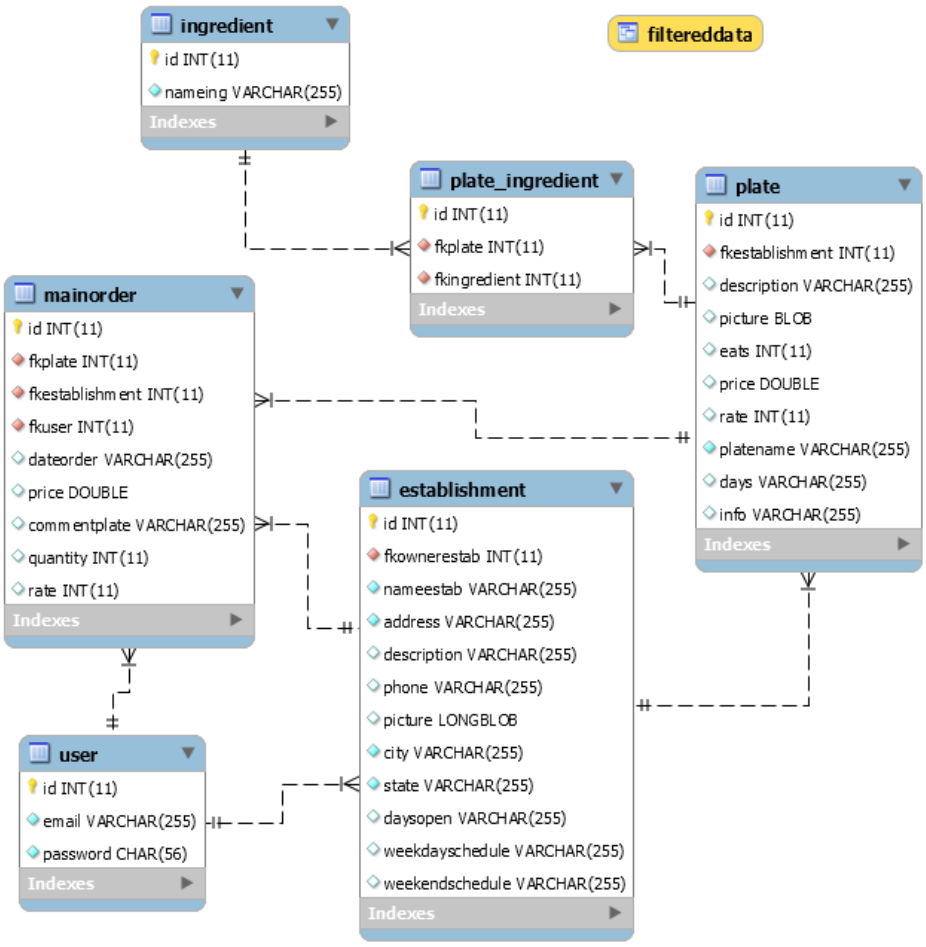

Figura 4 – Diagrama de entidades e relacionamentos do banco em MYSQL
Fonte: Elaborada pelo autor.

As chaves primárias são identificadas pelo campo “id”, seguindo um padrão, as chaves estrangeiras são identificadas pela sigla “fk” presentes no início do nome do campo seguido pelo nome da tabela a qual fazem referência, também seguindo um padrão.

No Quadro 7 estão os campos da tabela de ingredientes (*ingredient*). Um ingrediente pode estar contido em vários pratos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
NAMEING	varchar	Não	Não	Não

Quadro 7 – Campos da tabela *ingredient*

O Quadro 8 apresenta os campos da tabela usuários (*user*). Clientes e proprietários são os usuários do sistema que fazem buscas, podem registrar opinião sobre pratos que se acredita tenham provado. Um usuário pode ter um ou mais estabelecimentos, tornando-se então um proprietário, podendo adicionar pratos com detalhamento de ingredientes e tornando-os disponíveis para a busca.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
EMAIL	varchar	Não	Não	Não
PASSWORD	nvarchar	Não	Não	Não

Quadro 8 – Campos da tabela *user*

No Quadro 9 são apresentados os campos da tabela estabelecimento (*establishment*). Um estabelecimento pode conter vários pratos e avaliações realizadas pelos usuários. Na tabela estabelecimento, também são especificados os dias da semana e horários em que os estabelecimentos funcionam.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
FKOWNERESTAB	integer	Não	Não	Sim
NAMEESTAB	varchar	Não	Não	Não
ADDRESS	varchar	Não	Não	Não
CITY	varchar	Não	Não	Não
STATE	varchar	Não	Não	Não
DESCRIPTION	varchar	Sim	Não	Não
PHONE	varchar	Sim	Não	Não
PICTURE	blob	Sim	Não	Não
DAYSOPEN	varchar	Sim	Não	Não
WEEKDAYSCHEDULE	Varchar	Sim	Não	Não
WEEKENDSCHEDULE	varchar	Sim	Não	Não

Quadro 9 – Campos da tabela *establishment*

O Quadro 10 exibe os campos da tabela de pratos (*plate*). Um prato pode estar contido em vários estabelecimentos e conter várias avaliações. O prato pode possuir a informação sobre quais dias ele é servido. Também há local para informar restrições do prato (light, diet, sem glúten...).

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
FKESTABLISHMENT	integer	Não	Não	Sim
PLATENAME	varchar	Não	Não	Não
DESCRPTION	varchar	Sim	Não	Não
PICTURE	blob	Sim	Não	Não
EATS	integer	Sim	Não	Não
PRICE	double	Sim	Não	Não
RATE	integer	Sim	Não	Não
DAYS	varchar	Sim	Não	Não
INFO	varcahr	Sim	Não	Não

Quadro 10 – Campos da tabela *plate*

O Quadro 10, contém os campos da tabela *plate_ingredient*. Esta tabela foi criada com o propósito de resolver a relação muitos para muitos: um ingrediente pode estar em muitos pratos e um prato é composto por muitos ingredientes.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
FKPLATE	integer	Não	Não	Sim
FKINGREDIENT	integer	Não	Não	Sim

Quadro 11 – Campos da tabela *plate*

O Quadro 122, mostra informações sobre os campos da tabela avaliações (*mainorder*). Essa tabela contém informações que indicam o prato que o cliente consumiu, o nome do estabelecimento, o dia, o comentário e a nota atribuída pelo cliente.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
FKPLATE	integer	Não	Não	Sim
FKESTABLISHMENT	integer	Não	Não	Sim
FKUSER	integer	Não	Não	Sim
DATEORDER	varchar	Não	Não	Não
PRICE	double	Não	Não	Não
QUANTITY	double	Não	Não	Não
COMMENTPLATE	varchar	Sim	Não	Não
RATE	double	Sim	Não	Não

Quadro 122 – Campos da tabela *mainorder*

O Quadro 13 descreve a tabela *view filtereddata*. A *view* é uma tabela dinâmica de seleção, que se ajusta automaticamente com as inserções feitas nas tabelas que ela contém. Esta *view* faz a seleção dos dados contidos nas tabelas plate_*ingredient*, *ingredient*, *plate* e *establishment*. Tem como objetivo vincular os dados de um prato com seus ingredientes e seu estabelecimento para facilitar a obtenção de tais informações pelo aplicativo.

Campo	Tabela de referência	Campo de referência
IDPLATEINGREDIENT	plate_ingredient	ID
FKPLATE	plate_ingredient	FKPLATE
FKINGREDIENT	plate_ingredient	FKINGREDIENT
IDPLATE	plate	ID
PLATENAME	plate	PLATENAME
FKESTABLISHMENT	plate	FKESTABLISHMENT
DESCPLATE	plate	DESCRIPTION
PICPLATE	plate	PICTURE
EATS	plate	EATS
PRICE	plate	PRICE
RATE	plate	RATE
DAYS	plate	RATE
INFO	plate	INFO
IDING	ingredient	ID
NAMEING	ingredient	NAMEING
IDESTAB	establishment	ID
OWNERESTAB	establishment	FKOWNERESTAB
ADDRESS	establishment	ADDRESS
DESCESTAB	establishment	DESCRIPTION
PHONE	establishment	PHONE
PICESTAB	establishment	PICTURE
CITY	establishment	CITY
STATE	establishment	STATE
DAYSOPEN	establishment	DAYSOPEN
WEEKDAYSCHEDULE	establishment	WEEKDAYSCHEDULE
WEEKENDSCHEDULE	establishment	WEEKDAYSCHEDULE

Quadro 13 – Campos da view filtereddata

Além do banco em MYSQL há, ainda, o banco em SQLite, que é o banco de dados interno da aplicação. Esse banco contém somente as informações referentes ao usuário para controle de sessão do aplicativo. A Figura 5 descreve o diagrama de entidades e relacionamentos deste banco.

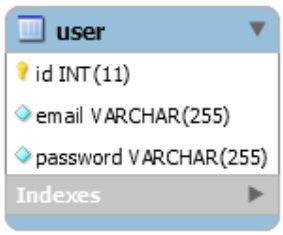

Figura 5 – Diagrama de entidades e relacionamentos do banco em SQLite
Fonte: Elaborada pelo autor

O Quadro 14 contém informações sobre os campos da tabela de usuário (user).

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
ID	integer	Não	Sim	Não
EMAIL	varchar	Não	Não	Não
PASSWORD	varchar	Não	Não	Não

Quadro 14 – Campos da tabela user

4.3 APRESENTAÇÃO DO SISTEMA

Uma aplicação para o sistema operacional Android possui em sua base, vários componentes divididos em interfaces gráficas (.xml) escritas no vocabulário *Extensible Markup Language* (XML) próprio do Android, no qual são definidos os componentes de interação com o usuário e classes de código (.class) na linguagem Java. Juntos, um componente de interface e uma classe de código se complementam formando uma atividade.

A interface gráfica de uma aplicação Android é dividida em dois setores: a barra de tarefas (*toolbar*) na qual se encontram as informações e opções de cada tela, como menus e títulos; e a área de conteúdo, que apresenta os demais componentes de interação com o usuário que dispõe de operações mais complexas e contextualizam o real objetivo do aplicativo.

Os aplicativos desenvolvidos para Android, seguem as normativas de design fornecidas e padronizadas pela Google. Juntas essas normativas formam um padrão chamado de Material Design. O aplicativo desenvolvido neste trabalho segue as normativas de *design* estipuladas pela Google que podem ser encontradas na página web de desenvolvimento Android (https://design.google.com/ e https://material.google.com/).

A Figura 6 representa a primeira tela do sistema. Nessa tela o usuário pode selecionar entre entrar ou criar uma nova conta no aplicativo.

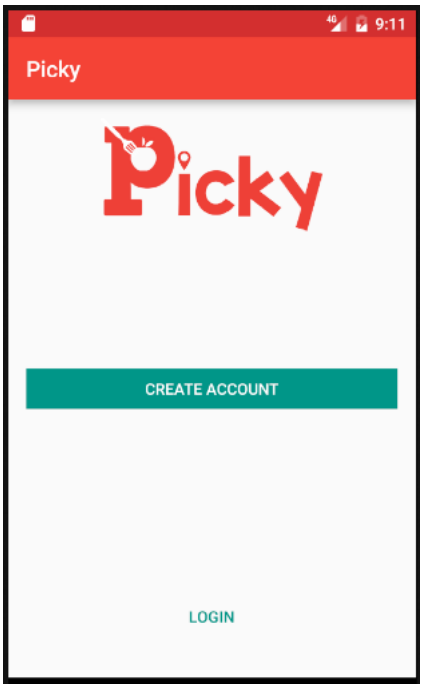

Figura 6 – Primeira tela do aplicativo

Após entrar com seu e-mail e senha no aplicativo é então exibida a tela principal do programa. Essa tela contém em sua barra de tarefas o título da atividade e dois ícones de opções. O primeiro ícone é de acesso aos estabelecimentos cadastrados, bem como a opção de cadastrar e em seguida gerenciar um novo estabelecimento. O segundo ícone oferece ao usuário a opção de sair da sua conta no sistema, voltando para a tela inicial.

Na área de conteúdo da tela principal, é exibida uma lista com o histórico de pratos avaliados pelo usuário. A lista dispõe uma imagem do prato, cadastrada pelo dono do ou responsável pelo estabelecimento, o nome do prato avaliado, o nome do estabelecimento que possui o prato sendo avaliado, a data em que a avaliação foi realizada e a nota atribuída ao prato na avaliação. Essa tela é exibida na Figura 7.

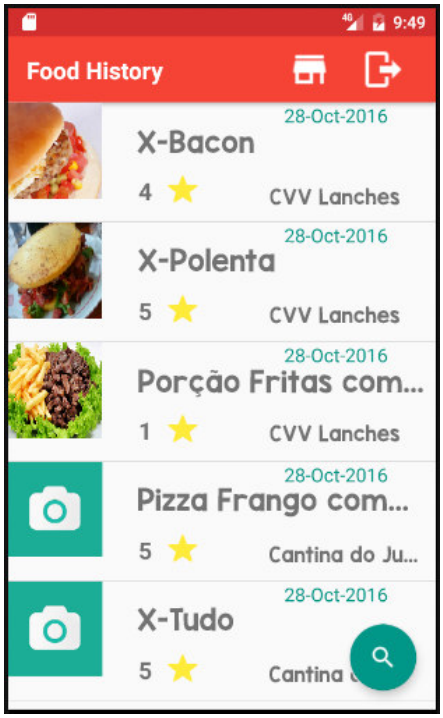

Figura 7 – Tela principal

No canto inferior direito da tela principal, há um botão flutuante. Esse botão exibe a tela de busca de locais por meio de pratos, estabelecimentos e/ou ingredientes. Essa tela possui um sistema gerenciador de abas composto por três abas. Na primeira aba (Figura 8) é possível inserir as informações que o usuário gostaria de encontrar. Com o uso de duas caixas de seleção opcionais, o usuário informa se deseja buscar um prato, ingrediente ou estabelecimento. A descrição como o nome do prato, pode ser inserida no campo de entrada de texto correspondente à caixa de seleção encontrado logo abaixo da mesma.

Entre as caixas de seleção, há dois botões de opção. Esses botões especificam se o usuário deseja que as opções filtrem os dados juntos ou separados fazendo uso dos rótulos ‘e’ e ‘ou’.

Ainda na primeira aba é possível escolher um local para a busca. Essa escolha é realizada por meio de uma caixa de seleção que traz as cidades dos estabelecimentos cadastrados na aplicação.

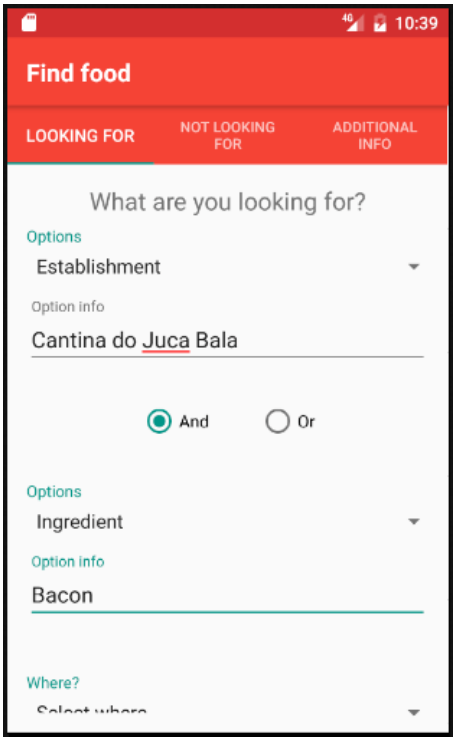

Figura 8 – Tela de busca – aba 1

Na segunda aba (Figura 9), utilizando também duas caixas de seleção contendo as mesmas opções da aba um e seus respectivos campos, contando também com os botões de opção ‘e’ e ‘ou’, o usuário tem a opção de informar o que não gostaria de encontrar em sua busca.

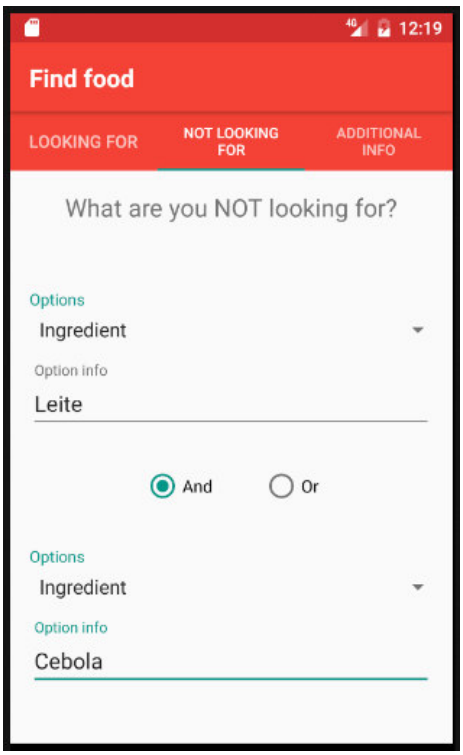

Figura 9 – Tela de busca – aba 2

A Figura 10 mostra a terceira aba dos filtros de busca. Nessa aba o usuário pode escolher para quantas pessoas deseja encontrar um determinado prato, a variação de preço e a nota mínima do mesmo.

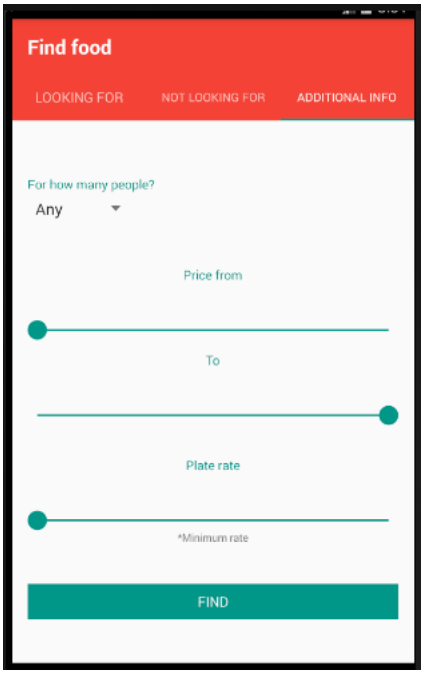

Figura 10 – Tela de busca – aba 3

Como resultado dessa busca é exibida uma lista com os pratos encontrados. Selecionando qualquer dos itens listados é exibida a tela de descrição em duas abas.

A primeira destas abas (Figura 11) mostra a descrição do prato, contendo uma imagem do prato, seu nome, nota média, descrição, até cinco dos seus ingredientes, os dias da semana em que é servido nesse estabelecimento, quantas pessoas o prato serve, preço, informações nutricionais e três comentários feitos por outras pessoas que consumiram este prato. O sistema apresenta os três últimos comentários realizados para o respectivo prato.

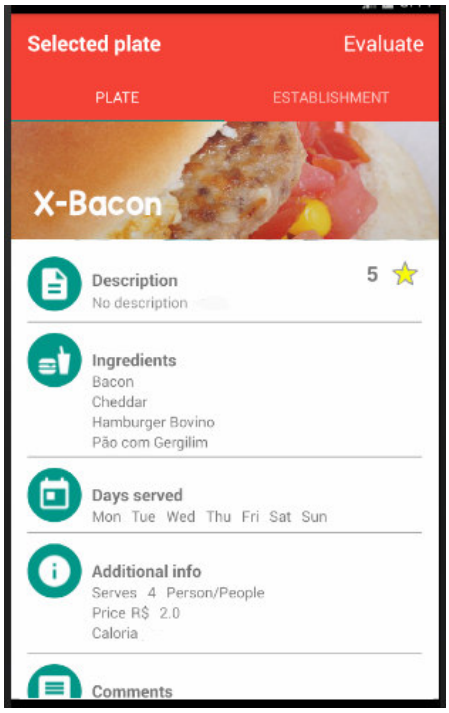

Figura 11 – Descrição de produto encontrado – aba 1

Na aba seguinte (Figura 12), são exibidas informações do estabelecimento que contém o prato selecionado. Nesta aba é possível visualizar uma foto do estabelecimento, a descrição deste, os dias da semana em que o mesmo se encontra em funcionamento e demais informações, como endereço, telefone e horário de atendimento durante a semana e nos finais de semana.

Ao final dessa tela encontra-se um botão com o texto “Me leve até lá” (“Get me there”, em Inglês), que abre o aplicativo ou website do serviço de localização e rotas da Google, o Google Maps, que fornece ao usuário as rotas até o estabelecimento escolhido.

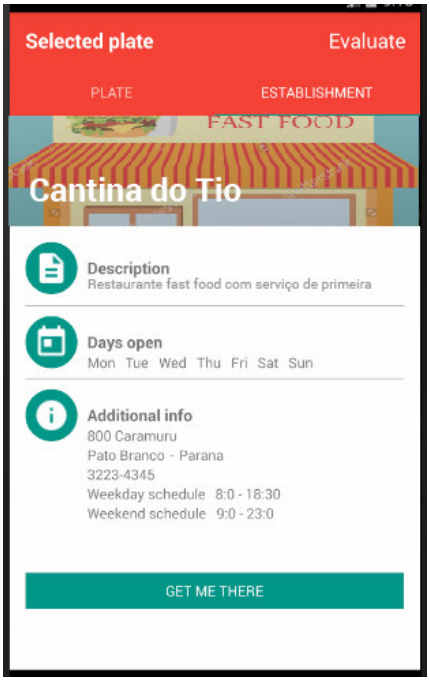

Figura 12 – Descrição de produto encontrado – aba 2

Ao pressionar o botão “Me leve até lá” (“Get me there”, em Inglês), será invocada a aplicação Google Maps sendo passado por parâmetro o endereço do estabelecimento selecionado. O aplicativo calculará, então, as possíveis rotas do local em que o usuário se encontra até o local do estabelecimento. A Figura 13 apresenta a tela do aplicativo com a rota indicada por meio do GoogleMaps.

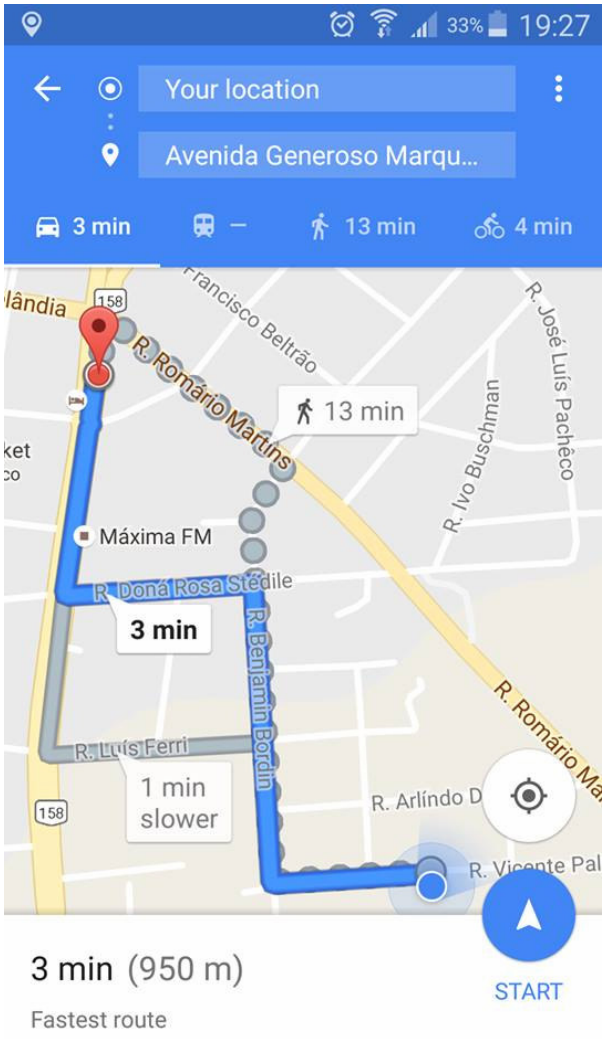

Figura 13 – Apresentação da rota no Google Maps

Na barra de ferramentas dessa tela, no lado direito encontra-se o botão “Avaliar” (“Evaluate”, no Inglês). Este botão chama a tela de avaliação do prato escolhido.

Na tela de avaliação (Figura 14), o usuário visualiza o nome do prato que está avaliando e o nome do estabelecimento no qual o referido prato é oferecido. Logo abaixo, o usuário tem a opção de informar a quantidade de porções deste prato que consumiu, dar uma nota ao prato e adicionar um comentário. O comentário será exibido a próximas pesquisas por este prato.

Na barra de ferramentas, no lado direito, existe o botão “Enviar” (“Send”, no Inglês), que envia a avaliação ao banco de dados e adiciona a mesma na lista de avaliações deste usuário na tela principal do aplicativo.

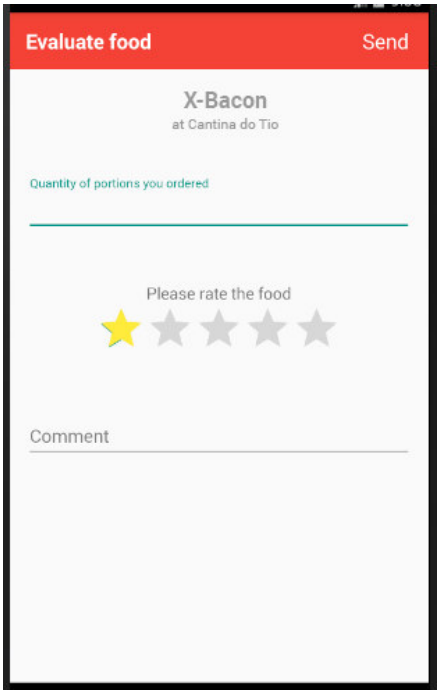

Figura 14 – Tela de avaliação de produto

Além das opções de busca, na tela principal o usuário tem a opção de gerenciar seus estabelecimentos, caso possua algum, por meio do ícone na barra de ferramentas (Figura 15).

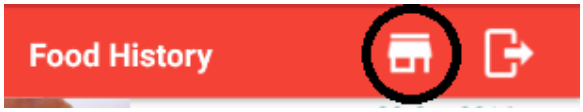

Figura 15 – Barra de ferramentas da tela principal

Tocando neste ícone, será exibida a tela de gerenciamento de estabelecimentos do usuário ativo (Figura 16). Essa tela mostra uma lista com todos os estabelecimentos do usuário atual, bem como uma foto, nome e cidade de cada estabelecimento.

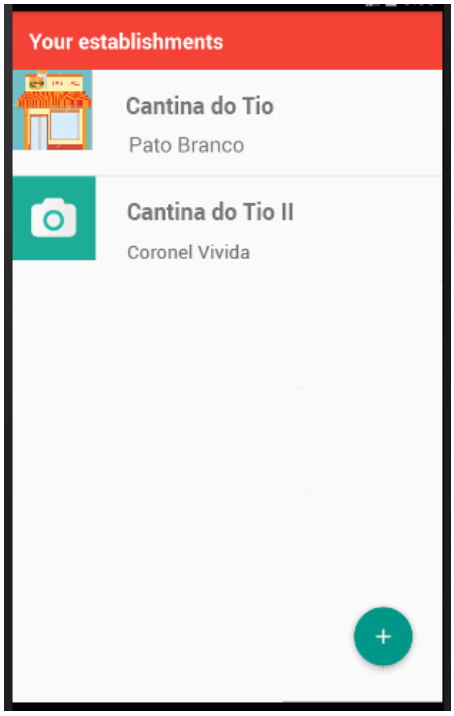

Figura 16 – Tela de gerenciamento de estabelecimentos

No canto inferior direito desta tela, há um botão flutuante, tocando neste botão o usuário pode inserir um novo estabelecimento. A tela de adição de estabelecimento é composta por três abas. A primeira aba (Figura 17) possui 5 campos de entrada de texto. Os campos recebem as informações do nome, endereço (rua e número), cidade, estado e telefone do estabelecimento.

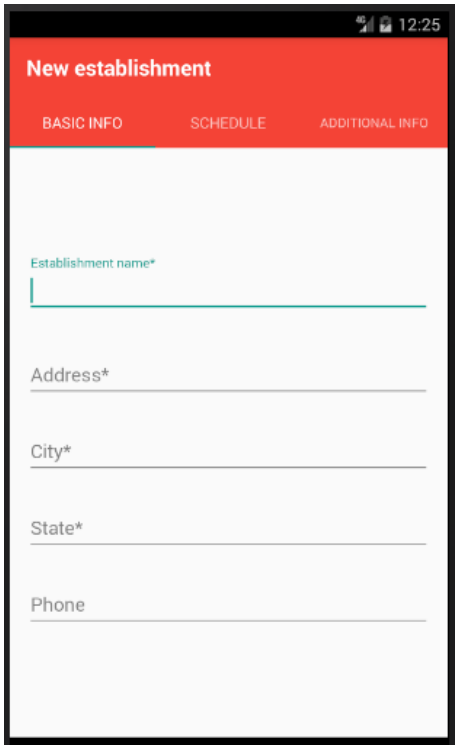

Figura 17 – Tela de novo estabelecimento – aba 1

Na segunda aba (Figura 18) existem cinco botões de ativação para marcar ou desmarcar os dias da semana em que o estabelecimento estará operando. São encontradas, ainda, duas caixas de entrada de texto para que seja possível informar os horários de início e fim do expediente. Tais caixas de inserção de texto possuem dois métodos de entrada de texto, o primeiro é via teclado e o segundo ocorre pela seleção de horário via relógio digital que é ativado se pressionado uma segunda vez a mesma caixa de inserção de texto.

Seguindo esses componentes, se encontram dois botões de seleção seguidos de duas caixas de inserção de texto para que seja informado os dias e horários de atendimento nos fins de semana.

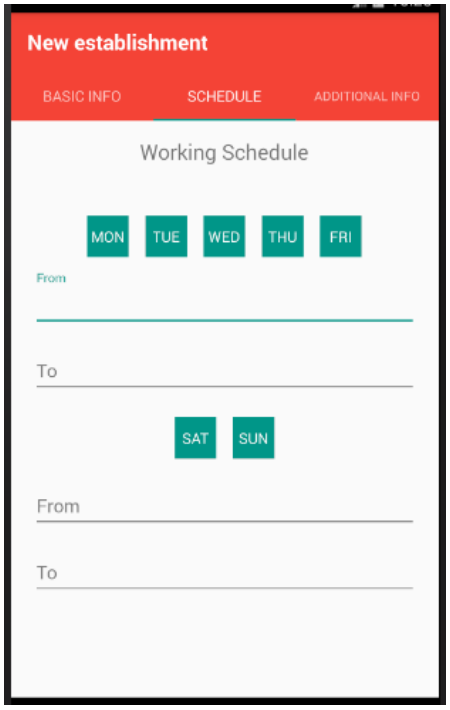

Figura 18 – Tela de novo estabelecimento – aba 2

Na terceira e última aba (Figura 19), se encontra um componente que exibe a foto do estabelecimento que é opcionalmente selecionada pelo botão flutuante logo abaixo deste componente. Ao tocar no botão flutuante, um diálogo é exibido para saber se o usuário deseja escolher uma foto existente na memória ou acessar a câmera fotográfica do seu dispositivo para obter nova foto.

Logo abaixo há uma caixa de inserção de texto para que seja informada a descrição do estabelecimento. Em seguida está o botão de texto “Salvar” (“Save” no Inglês), que envia o novo estabelecimento para o banco de dados e o vincula com o usuário que o cadastrou.

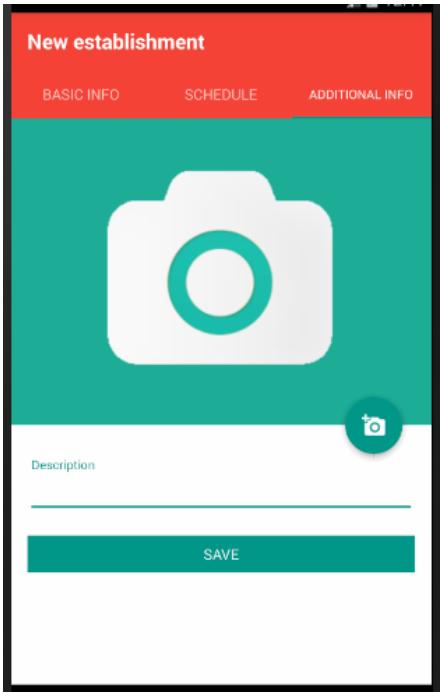

Figura 19 – Tela de novo estabelecimento – aba 3

Após salvo o estabelecimento, ele é exibido na lista de estabelecimentos gerenciáveis na tela de gerenciamento de estabelecimentos.

Um toque leve sobre os itens desta lista permite visualizar a descrição completa deste estabelecimento juntamente com sua lista de pratos (Figura 20). Um toque longo sobre os itens desta lista exibe duas opções, editar ou excluir o estabelecimento.

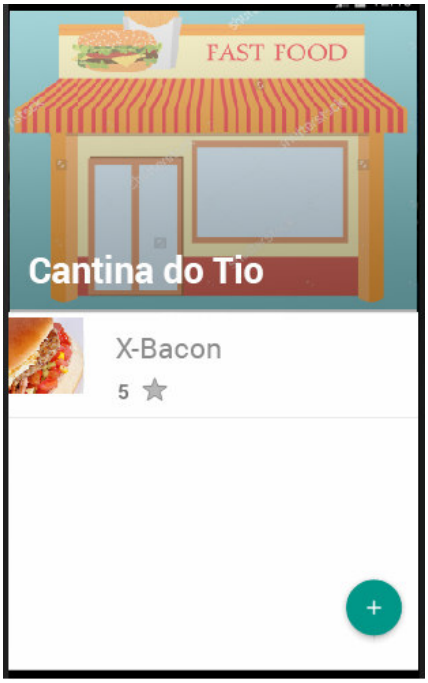

Figura 20 – Tela de descrição de estabelecimento

Tocando em um item da lista de pratos, é possível visualizar o prato cadastrado. E um toque longo exibe um diálogo que permite selecionar a opção de editar o prato selecionado ou excluí-lo.

Essa tela também possui um botão flutuante no canto inferior direito, o qual possibilita cadastrar novos pratos para o estabelecimento.

A tela de cadastro de novos pratos também possui três abas. A primeira (Figura 21) exibe cinco caixas de inserção de texto que recebem obrigatoriamente a opção de nome do estabelecimento e opcionalmente as opções que informam o número de pessoas que tal prato serve, seu preço, informações nutricionais e uma descrição do prato.

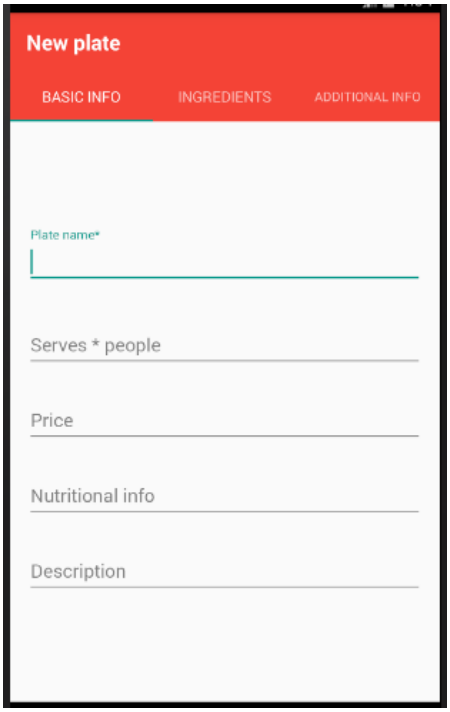

Figura 21 – Tela de cadastro de prato – aba 1

A segunda aba (Figura 22) é composta de sete botões de seleção para serem informados os dias em que este prato é servido. Após isto são dispostas em sequência cinco caixas de seleção contendo os ingredientes pré-cadastrados no banco de dados, que podem opcionalmente compor os ingredientes do prato a ser cadastrado.

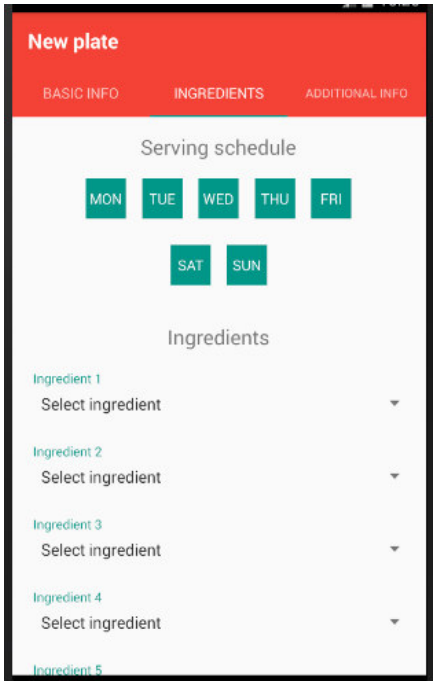

Figura 22 – Tela de cadastro de prato – aba 2

Na terceira e última aba do cadastro de pratos, como no cadastro de estabelecimentos (Figura 23), exibe um campo para visualização de uma imagem do prato que pode ser opcionalmente adicionada pelo botão flutuante logo abaixo deste componente. Há ainda, uma caixa de inserção de texto que recebe a descrição do prato a ser cadastrado. E, por último, um botão com o texto “Salvar” (“Save”, no Inglês) que envia esses dados ao banco de dados e os vincula com o estabelecimento, o que permite que este prato seja encontrado por futuras buscas.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nos arquivos .class, chamados de classes, são encontradas as rotinas de negócio implementadas em Java, também chamado de back-end. Esses arquivos definem a lógica e as funções que cada componente da interface desempenha. Também define como e quando acontecerá a comunicação do sistema com o banco de dados. Já os arquivos .xml possuem um código em XML que define os componentes presentes no sistema que foram previamente apresentados na Seção 4.3. Os arquivos escritos em XML para Android são conhecidos como front-end do aplicativo.

Tais atividades são dispostas em uma estrutura de pacotes (Figura 23), a estrutura é composta pelos pacotes principais Java e res.

O pacote Java possui as classes do sistema divididas em sub-pacotes. O sub-pacote Activities possui todas as classes de atividades da aplicação. O sub-pacote Controller possui todas as classes de controle como os *adapters* de lista personalizadas. Já sub-pacote Dao possui todas as classes de conexão com bancos de dados. O sub-pacote Fragments possui todos os fragmentos que compõem as abas das atividades do sistema. E o sub-pacote Model possui todos os modelos de estruturas como a classe Evaluation (Listagem 6).

O pacote res possui todos os elementos gráficos da aplicação e também é dividido em sub-pacotes. O sub-pacote *color* contém as variáveis de cores do sistema. O sub-pacote *drawable* possui todas as imagens e formas utilizadas na aplicação. O sub-pacote *layout* possui todos os arquivos XML que constituem as atividades, fragmentos e listas personalizadas. O sub-pacote *mipmap* possui o ícone da aplicação. E, por fim, o sub-pacote *values* armazena todos os arquivos de valores que são utilizados na execução da aplicação, como o arquivo de *strings* em que se encontram todas as frases exibidas ao usuário em Inglês (língua base para a aplicação) bem como os arquivos *string* de internacionalização, que traduzem todas estas frases para outros idiomas.

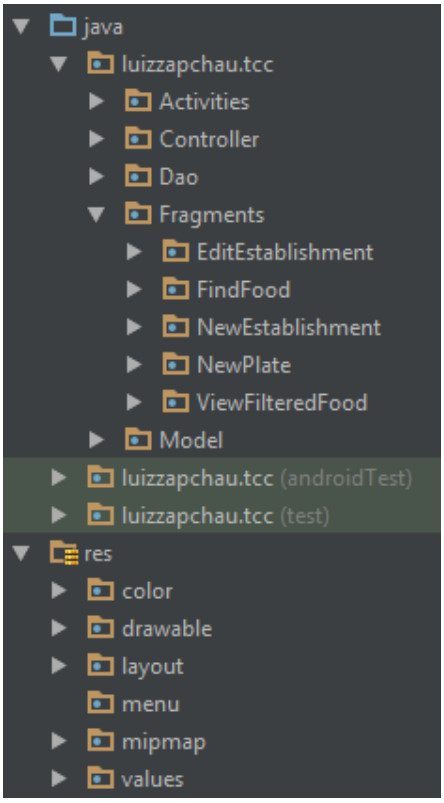

Figura 23 – Estrutura de pacotes do sistema

As Listagens 1 e 2 mostram o código da XML da atividade principal. Esta atividade do tipo Basic Activity é dividida em dois arquivos: activity_main e content_main. O arquivo activity_main (Listagem 1) contém as informações da barra de ferramentas da atividade.

Essa barra de ferramentas possui um componente de texto (TextView) para o título da atividade e dois componentes de botão com imagem (ImageButton), para exibir os ícones de acesso a atividade de gerenciamento de estabelecimentos e o ícone para desconectar a conta do usuário do aplicativo.

```
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="luizzapchau.tcc.Activities.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">
```

```
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15dp"
                android:layout_marginRight="15dp"
                android:layout_marginTop="15dp"
                android:text="@string/food_history"
                android:textSize="20sp"
                android:background="@null"
                android:textColor="@android:color/background_light"
                android:textStyle="bold" />

            <ImageButton
                android:id="@+id/btLogout"
                android:textAlignment="center"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="btLogoutOnClick"
                android:background="@null"
                android:clickable="true"
                android:visibility="visible"
                android:layout_gravity="right|top"
                android:src="@drawable/logout"
                style="?android:attr/borderlessButtonStyle"/>

            <ImageButton
                android:id="@+id/btEstablishment"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="btEstablishmentOnClick"
                android:background="@null"
                android:src="@drawable/store"
                android:clickable="true"
                android:layout_gravity="right|top"
                android:visibility="visible" />

        </android.support.v7.widget.Toolbar>

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

</android.support.design.widget.CoordinatorLayout>
```

Listagem 1 – Código XML da atividade principal – parte 1

O arquivo “content_main” (Listagem 2) codifica o conteúdo da atividade principal. O conteúdo principal é constituído de um componente de lista (ListView) para a construção da lista de estabelecimentos e um componente FloatingActionButton que é o botão flutuante que exibirá a atividade de busca.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="luizzapchau.tcc.Activities.MainActivity">
    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15dp"
                android:layout_marginRight="15dp"
                android:layout_marginTop="15dp"
                android:text="@string/food_history"
                android:textSize="20sp"
                android:background="@null"
                android:textColor="@android:color/background_light"
                android:textStyle="bold" />

            <ImageButton
                android:id="@+id/btLogout"
                android:textAlignment="center"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="btLogoutOnClick"
                android:background="@null"
                android:clickable="true"
                android:visibility="visible"
                android:layout_gravity="right|top"
                android:src="@drawable/logout"
                style="?android:attr/borderlessButtonStyle"/>

            <ImageButton
                android:id="@+id/btEstablishment"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="btEstablishmentOnClick"
                android:background="@null"
                android:src="@drawable/store"
                android:clickable="true"
                android:layout_gravity="right|top"
                android:visibility="visible" />
        </android.support.v7.widget.Toolbar>
    </android.support.design.widget.AppBarLayout>
    <include layout="@layout/content_main" />
</android.support.design.widget.CoordinatorLayout>
```

Listagem 2 – Código XML da atividade principal – parte 2

Para que esses componentes funcionem existe a classe “MainActivity” (Listagem 3) que contém a lógica desta atividade.

Logo no início da classe é instanciada a classe SQLiteHelper (Listagem 4) que contém as informações para acesso e troca de informações com o banco local do dispositivo.

Após isso, é instanciada uma variável da classe Context que é um ponto de acesso para informações globais sobre um ambiente de aplicativo.

Então é chamada a anotação @Bindview que faz parte da API Butter Knife que possibilita o uso de anotações no ambiente Android. Essa anotação vincula um componente do arquivo XML dessa atividade a uma variável no código. Neste caso está vinculando o componente lvFood que é o componente lista no XML, à variável lvFoodHistory que é a variável no código Java.

No método OnCreate que é inicializado automaticamente toda vez que a atividade é iniciada, é indicado com o arquivo XML que será vinculado com esta classe. Após isso é inicializada a barra de ferramentas da atividade. Em seguida é inicializado o componente para a utilização da API Butter Knife nesta atividade. Então são inicializadas as variáveis de acesso ao banco SQLite e a variável da classe Context. Por último, o método chama o método onLvFoundFoodHistoryItemClick que inicializa o evento de toque em um item da lista de avaliações.

O método onResume é iniciado automaticamente toda vez que a atividade é iniciada ou resumida. Esse método chama o método startEvaluationList que preenche a lista de avaliações do usuário atual.

O método onBackPressed é acionado toda vez que o botão de voltar da atividade é pressionado. Neste caso o método finaliza a aplicação evitando que a mesma volte para a primeira tela do sistema.

Em seguida há o método btFindFoodOnClick que é vinculado ao botão flutuante desta atividade por meio da anotação @OnClick. Caso o botão seja pressionado, esse método é acionado, exibindo a atividade de busca.

O método seguinte, de nome startEvaluationList, que é inicializado pelo método onResume, busca no banco SQLite o campo id da tabela usuário, que contém o código de identificação do usuário conectado no sistema no momento. Esse método então inicializa uma variável do tipo JSONArray (evaluationsListArray) que vai receber um vetor de objetos JSON do banco de dados MYSQL que se

encontra na nuvem. Este vetor é solicitado por meio do método listEvaluationByUser (Listagem 5) que lista as avaliações pelo código do usuário conectado. Após isso, o método por fim chama o método addEvaluationsToList passando como parâmetro o vetor de avaliações recebido do banco de dados. Esse método adiciona os itens do vetor ao componente lista da atividade.

O método addEvaluationsToList recebe um vetor de avaliações por parâmetro. Em seguida cria um vetor da classe Evaluations que está descrito na Listagem 6. E, ainda, instancia e inicializa uma variável da classe EvaluationAdapter (Listagem 7) que é um *adapter* para que cada item da lista seja personalizado a partir de outro arquivo XML, passando por parâmetro o vetor de da classe Evaluations criado. Então, o método vincula o *adapter* com a lista e adiciona o vetor de avaliações recebido por meio do método fromJson da classe Evaluation. Após fazer isto, chama o método addAll da classe EvaluationAdapter para que os itens sejam adicionados à lista.

Em seguida o método onDestroy que é ativado toda vez que a atividade é encerrada finaliza a conexão com o baco de dados SQLite.

O método btEstablishmentOnClick inicializa a atividade de gerenciamento de estabelecimentos quando tocado.

O método seguinte btLogoutOnClick limpa as informações da conta de usuário conectada do banco local do dispositivo e então inicializa a primeira atividade de aplicativo.

Por último, o método onLvFoundFoodHistoryItemClick inicializa a atividade que exibe a avaliação feita para o prato selecionado passando por parâmetro o código de identificação da avaliação que está contido no item da lista correspondente.

```
package luizzapchau.tcc.Activities;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;
import org.json.JSONArray;
import java.util.ArrayList;
import butterknife.BindView;
import butterknife.ButterKnife;
```

```
import butterknife.OnClick;
import luizzapchau.tcc.Controller.EvaluationAdapter;
import luizzapchau.tcc.Dao.EvaluateFoodDao;
import luizzapchau.tcc.Dao.SQLiteHelper;
import luizzapchau.tcc.Model.Evaluation;
import luizzapchau.tcc.R;

public class MainActivity extends AppCompatActivity {

    private SQLiteHelper sqLiteHelper;
    private Context mContext;

    @BindView(R.id.lvFood) ListView lvFoodHistory;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayShowTitleEnabled(false);

        ButterKnife.bind(this);

        sqLiteHelper = new SQLiteHelper(this);
        mContext = this;

        onLvFoundFoodHistoryItemClick();
    }

    @Override
    public void onResume() {
        super.onResume();

        startEvaluationList();
    }

    @Override
    public void onBackPressed(){
        Intent intent = new Intent(Intent.ACTION_MAIN);
        intent.addCategory(Intent.CATEGORY_HOME);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

    @OnClick(R.id.btFindFood)
    public void btFindFoodOnClick(){
        Intent i = new Intent(this, FindFoodActivity.class);
        startActivity(i);
    }

    public void startEvaluationList() {

        final String USER = sqLiteHelper.userId();

        JSONArray evaluationsArray =
EvaluateFoodDao.listEvaluationsByUser(USER);

        addEvaluationsToList(evaluationsArray);
    }
```

```
    public void  addEvaluationsToList(JSONArray evaluationsList){
        ArrayList<Evaluation> arrayOfEvaluations = new ArrayList<>();
        EvaluationAdapter adapter = new EvaluationAdapter(this,
arrayOfEvaluations);
        lvFoodHistory.setAdapter(adapter);
        ArrayList<Evaluation> newEvaluation =
Evaluation.fromJson(evaluationsList);
        adapter.addAll(newEvaluation);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        sqLiteHelper.close();
    }

    @OnClick(R.id.btEstablishment)
    public void btEstablishmentOnClick(){
        Intent i = new Intent(this, EstablishmentManagementActivity.class);
        startActivity(i);
    }

    @OnClick(R.id.btLogout)
    public void btLogoutOnClick(){

        sqLiteHelper.deleteUserData();
        Intent i = new Intent(this, FirstActivity.class);
        startActivity(i);
    }

    private void onLvFoundFoodHistoryItemClick() {
        lvFoodHistory.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                System.out.println("POSITION"+position);
                TextView tvEvaluationId = (TextView)
view.findViewById(R.id.tvEvaluationId);
                String evaluationId     =
tvEvaluationId.getText().toString();

                Intent i = new Intent(mContext,
ViewEvaluationActivity.class);
                i.putExtra("id", evaluationId);
                startActivity(i);
            }
        });
    }
}
```

Listagem 3 – Código Java da atividade principal

A classe SQLiteHelper (Listagem 4) declara variáveis que farão interação com o banco de dados local de forma final e estática, recebendo os valores com os nomes dos campos no banco de dados para facilitar a codificação.

Após isso o método onCreate que é chamado automaticamente quando o banco é criado, cria a tabela que armazenará as informações do usuário conectado.

O método onUpgrade exclui a tabela e a recria (caso exista) quando o banco é atualizado.

O método insertUserData recebe por parâmetro o código de identificação do usuário por meio da primeira atividade do sistema, seu e-mail e sua senha e as adiciona ao banco de dados local.

O método deleteUserData exclui as informações inseridas no banco de dados local pelo método insertUserData.

O método userId recupera o código de identificação do usuário contido no banco local.

```
public class SQLiteHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "tcc.db";

    private static final String TB_USER       = "user";
    private static final String USER_ID       = "_id";
    private static final String USER_EMAIL    = "email";
    private static final String USER_PASSWORD = "password";

    public SQLiteHelper(Context context) {
        super(context, DB_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        StringBuilder createTableUser = new StringBuilder();
        createTableUser.append("CREATE TABLE ");
        createTableUser.append(TB_USER);
        createTableUser.append(" (");
        createTableUser.append(USER_ID);
        createTableUser.append(" INTEGER PRIMARY KEY, ");
        createTableUser.append(USER_EMAIL);
        createTableUser.append(" TEXT NOT NULL);");
        createTableUser.append(USER_PASSWORD);
        createTableUser.append(" TEXT NOT NULL);");

        db.execSQL(createTableUser.toString());
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion){
        db.execSQL("DROP TABLE IF EXISTS " + TB_USER);
        onCreate(db);
    }

    public boolean insertUserData(Integer id, String email, String
password){
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
```

```
        contentValues.put(USER_ID, id);
        contentValues.put(USER_EMAIL, email);
        contentValues.put(USER_PASSWORD, password);

        Long result = db.insert(TB_USER, null, contentValues);

        if (result == -1)
            return false;
        else
            return true;
    }

    public void deleteUserData() {
        SQLiteDatabase db = this.getWritableDatabase();

        db.delete(TB_USER, null, null);
    }

    public String userId() {
        SQLiteDatabase db = getWritableDatabase();

        StringBuilder select = new StringBuilder();
        select.append("SELECT ");
        select.append(USER_ID);
        select.append(" FROM ");
        select.append(TB_USER);
        Cursor mcursor = db.rawQuery(select.toString(),null);

        mcursor.moveToFirst();
        String userId = mcursor.getString(0);

        return userId;
    }
}
```

Listagem 4 – Código Java da classe SQLiteHelper

O método listEvaluationByUser (Listagem 5) da classe EvaluateFoodDao, lista as avaliações feitas por um usuário recebendo o código de identificação desse usuário por parâmetro.

Um serviço executor é iniciado, em seguida é instanciado um objeto da classe Future que espera um resultado de uma operação do tipo Callable. Nesse caso a operação retornará um JSONArray. O método então inicializa o método *call* da classe Callable. Em seguida uma conexão com o banco de dados MYSQL na nuvem é iniciada através de um método da classe MYSQLHelper (Listagem 8). Após isso é criada uma string contendo o comando SQL para buscar as avaliações no banco de dados. O método então envia esse comando ao banco e caso haja resultado coloca cada variável em um JSONObject que é então colocado em um JSONArray. Por fim o Método retorna o JSONArray esperado pela variável da classe

Future que é retornado a atividade principal do aplicativo para ser usada na lista de avaliações.

```
public static JSONArray listEvaluationsByUser(final String USER) {

    ExecutorService executor = Executors.newCachedThreadPool();
    Future<JSONArray> res = executor.submit(new Callable<JSONArray>() {

        @Override
        public JSONArray call() throws Exception {

            final JSONArray evaluationList = new JSONArray();

            try {
                Connection conn = MYSQLHelper.connect();
                Statement stmt = conn.createStatement();
                StringBuilder sql = new StringBuilder();
                sql.append("SELECT * ");
                sql.append(" FROM ");
                sql.append(TB_MAIN_ORDER);
                sql.append(" WHERE ");
                sql.append(MAIN_USER);
                sql.append(" = ");
                sql.append(USER);

                try {
                    ResultSet rs = stmt.executeQuery(sql.toString());
                    while (rs.next()) {
                        JSONObject evaluation = new JSONObject();
                        try{
                           evaluation.put("id",  rs.getInt   (MAIN_ID));
                           evaluation.put("plate", rs.getInt
(MAIN_PLATE));
                           evaluation.put("estab", rs.getInt
(MAIN_ESTAB));
                           evaluation.put("date",
rs.getString(MAIN_DATE));
                           evaluation.put("price",
rs.getDouble(MAIN_PRICE));
                           evaluation.put("comment",
rs.getString(MAIN_COMMENT));
                           evaluation.put("quantity",rs.getInt(MAIN_QUANTITY));
                           evaluation.put("rate",  s.getInt   (MAIN_RATE));
                           evaluationList.put(evaluation);
                        }catch (Exception e){
                            e.printStackTrace();
                        }
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            } catch (Exception e) {
                e.printStackTrace();
                System.out.println("fetch action error" +
e.getLocalizedMessage());
            }
            return evaluationList;
        }

    });
```

```
    try {
        return res.get();
    } catch (InterruptedException | ExecutionException e) {
        e.printStackTrace();
    }
    JSONArray ret = new JSONArray();
    ret.put("empty");

    return ret;
}
```

Listagem 5 – Código Java do método *listEvaluationByUser* da classe *EvaluateFoodDao*

A classe Evaluations (Listagem 6) lê um JSONObject. O método fromJson recebe por parâmetro um JSONArray o transforma em um vetor de *strings* e o retorna à classe que o chamou.

```
package luizzapchau.tcc.Model;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;

import luizzapchau.tcc.Dao.EstablishmentDao;
import luizzapchau.tcc.Dao.PlateDao;

/**
 * Created by Luiz Z on 10/28/2016.
 */

public class Evaluation {
    public  String id;
    public  String plateId;
    public  String estabId;
    public  String date;
    public  String price;
    public  String comment;
    private String quantity;
    public  String plateRate;

    private Evaluation(JSONObject object){
        try {
            this.id        = object.getString("id");
            this.plateId   = object.getString("plate");
            this.estabId   = object.getString("estab");
            this.date      = object.getString("date");
            this.price     = object.getString("price");
            this.comment   = object.getString("comment");
            this.quantity  = object.getString("quantity");
            this.plateRate = object.getString("rate");
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    public static ArrayList<Evaluation> fromJson(JSONArray jsonObjects) {
```

```
        ArrayList<Evaluation> food = new ArrayList<>();

        for (int i = 0; i < jsonObjects.length(); i++) {
            try {
                food.add(new Evaluation(jsonObjects.getJSONObject(i)));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return food;
    }
}
```

Listagem 6 – Código Java da classe *Evaluation*

A classe EvaluationAdapter (Listagem 7) recebe um JSONArray com os vetores das avaliações vindos do banco de dados na nuvem os lê e os vincula com os elementos da lista personalizada (Listagem 9), por meio do método getItem para que sejam adicionados à lista de avaliações da atividade principal.

O método getBitMapFromString recebe uma *string* do banco de dados com o código Base64 contendo a imagem e a converte para Bitmap para que seja adicionada a lista.

```
public class EvaluationAdapter extends ArrayAdapter<Evaluation>{

    public EvaluationAdapter(Context mContext, ArrayList<Evaluation>
evaluations){
        super(mContext, 0, evaluations);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent){

        Evaluation evaluation = getItem(position);
        if (convertView == null){
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.list_evaluation_history,
parent, false);
        }
        TextView tvId        = (TextView)
convertView.findViewById(R.id.tvEvaluationId);
        TextView tvPlateId   = (TextView)
convertView.findViewById(R.id.tvEvaluationPlateId);
        TextView tvPlateName = (TextView)
convertView.findViewById(R.id.tvEvaluationPlateName);
        ImageView ivPlatePic = (ImageView)
convertView.findViewById(R.id.ivEvaluationPicture);
        TextView tvEstabId   = (TextView)
convertView.findViewById(R.id.tvEvaluationEstablishmentId);
        TextView tvEstabName = (TextView)
convertView.findViewById(R.id.tvEvaluationEstablishmentName);
        TextView tvPlateRate = (TextView)
convertView.findViewById(R.id.tvEvaluationPlateRate);
        TextView tvDate      = (TextView)
convertView.findViewById(R.id.tvEvaluationDate);
```

```
        String plate = PlateDao.findPlateById(evaluation.plateId);
        String platePic = PlateDao.findPictureById(evaluation.plateId);
        String establishment =
EstablishmentDao.findEstablishmentById(evaluation.estabId);

        tvId       .setText(evaluation.id);
        tvPlateId  .setText(evaluation.plateId);
        tvPlateName.setText(plate);
        tvEstabId  .setText(evaluation.estabId);
        tvEstabName.setText(establishment);
        tvPlateRate.setText(evaluation.plateRate);
        tvDate     .setText(evaluation.date);
        if (!platePic.equals("nopic")){
            ivPlatePic.setImageBitmap(getBitmapFromString(platePic));
        }
        return convertView;
    }

    private Bitmap getBitmapFromString(String stringPicture) {

        byte[] decodedString = Base64.decode(stringPicture,
Base64.DEFAULT);
        return BitmapFactory.decodeByteArray(decodedString, 0,
decodedString.length);
    }
```

Listagem 7 – Código Java da classe *EvaluationAdapter*

A classe MYSQLHelper (Listagem 8) efetua e retorna a conexão com o banco de dados MYSQL na nuvem.

```
class MYSQLHelper {
    static Connection connect() throws ClassNotFoundException {
        Class.forName("com.mysql.jdbc.Driver");
        String connectionUrl = "jdbc:mysql://tccproject.cjysntudvjwc.sa-
east-1.rds.amazonaws.com:3306/tccdb";
        String dbUser = "root";
        String dbPwd = "senha";
        Connection conn = null;

        try {
            conn = DriverManager.getConnection(connectionUrl, dbUser,
dbPwd);
            System.out.println("conn Available");

        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("fetch action
error"+e.getLocalizedMessage());
        }
        return conn;
    }
}
```

Listagem 8 – Código Java da classe *MYSQLHelper*

A lista personalizada chamada de list_evaluation_history (Listagem 9) que define cada elemento a ser adicionada à lista de avaliações na tela principal da

aplicação é composta de um elemento ImageView para exibir a foto do prato que foi avaliado, um ícone em forma de uma estrela simbolizando a nota do prato e quatro componentes TextView exibindo o nome do prato, a data da avaliação, a nota da avaliação e o estabelecimento a qual este prato pertence.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_marginRight="10dp"
        android:layout_width="100dp"
        android:layout_height="100dp">

        <ImageView
            android:id="@+id/ivEvaluationPicture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="fitStart"
            android:paddingBottom="20dp"
            android:src="@drawable/camera"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvEvaluationPlateName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:background="@null"
            android:text="@string/plate_name"
            android:textSize="25sp"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/tvEvaluationPlateRate"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:background="@null"
                android:text="5.0"
                android:textStyle="bold"
                android:textSize="20sp"/>

            <CheckBox
                android:id="@+id/cbRate"
                style="?android:attr/starStyle"
```

```
                android:textSize="20sp"
                android:layout_marginTop="8dp"
                android:layout_marginLeft="10dp"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:buttonTint="#FFEB3B"
                android:checked="false"
                android:focusable="false"
                android:clickable="false"/>

        </LinearLayout>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvEvaluationDate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="16sp"
            android:layout_gravity="end"
            android:layout_marginRight="40dp"
            android:textColor="@color/colorAccent"
            android:text="@string/date" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="end|bottom"
            android:paddingBottom="12dp"
            android:paddingRight="10dp">

            <TextView
                android:id="@+id/tvEvaluationEstablishmentName"
                android:layout_width="130dp"
                android:layout_height="20dp"
                android:ellipsize="end"
                android:textSize="16sp"
                android:text="Establishment" />

        </LinearLayout>

    </LinearLayout>

    <TextView
        android:id="@+id/tvEvaluationPlateId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="id"
        android:visibility="gone"/>

    <TextView
        android:id="@+id/tvEvaluationEstablishmentId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="id"
        android:visibility="gone"/>
```

```
    <TextView
        android:id="@+id/tvEvaluationId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="id"
        android:visibility="gone"/>

</LinearLayout>
```

Listagem 9 – Código XML da lista personalizada *list_evaluation_history*

As demais classes da aplicação seguem o mesmo padrão de codificação dos códigos apresentados nas Listagens 1 a 9.

5 CONCLUSÃO

O objetivo deste trabalho foi propor uma solução para dispositivos móveis Android para que pessoas encontrem a localização de alimentos e locais para consumi-los. A escolha pelo Android foi decorrente de sua ampla disseminação no mercado para dispositivos móveis.

A escolha de armazenamento em nuvem dos dados necessários para o adequado funcionamento do aplicativo permitiu a criação de uma base de dados mais ampla para consulta. A segurança proporcionada pelos servidores de computação em nuvem, a facilidade e a praticidade com que os dados serão disponibilizados para a utilização e manipulação dos mesmos por parte do aplicativo, favorece ainda mais o objetivo da aplicação.

No aplicativo desenvolvido, o filtro aplicado nas buscas facilita ao usuário evitar conteúdos indesejáveis e encontrar o alimento ideal com base nos ingredientes desejados, preço, qualidade do prato avaliado por outros usuários, bem como encontrar o local especificado pelo usuário.

O desenvolvimento do aplicativo resultado deste trabalho ajudou a perceber a lacuna que existe no mercado de venda de alimentos neste sentido e o quão difícil é para clientes desses estabelecimentos evitarem ingredientes indesejados.

Melhorias podem ser desenvolvidas como, por exemplo, a aplicação de técnicas para a mineração dos dados e de Inteligência Artificial fornecendo dados que possam melhor auxiliar a decisão do consumidor.

REFERÊNCIAS

ALCÂNTARA, Carlos Augusto Almeida; VIEIRA, Anderson Luiz Nogueira. **Tecnologia móvel: uma tendência, uma realidade**. 2011. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1105/1105.3715.pdf>. Acesso em: 11 jun. 2016.

ANDROID DEVELOPER. **Android, the world's most popular mobile platform**. 2012. Disponível em: <https://developer.android.com/about/android.html>. Acesso em: 08 mai. 2016.

ANDROID DEVELOPER. **Dashboards**. 2012. Disponível em: <https://developer.android.com/about/dashboards/index.html>. Acesso em: 08 mai. 2016.

GUANA, Victor; ROCHA, Fabio; HINDLE, Abram; STROULIA, Eleni. **Do the stars align? multidimensional analysis of android's layered architecture**. In: Conference on Mining Software Repositories, 2012, p. 124-127.

HAMANN, Renan. **iOS, Android e Windows Phone: números dos gigantes comparados**. 2014. Disponível em: <http://www.tecmundo.com.br/sistema-operacional/60596-ios-android-windows-phone-numeros-gigantes-comparados-infografico.htm>. Acesso em: 27 set. 2016.

INFOCCOMP. **Modelos de processo de engenharia de software**. Disponível em: <https://infoccomp.wordpress.com/modelos-de-processo-de-engenharia-de-software/>. Acesso em: 23 ago. 2016.

MACORATTI, José. **O processo de software**. Disponível em: <http://www.macoratti.net/proc_sw1.htm>. Acesso em: 08 mai. 2016.

MAJI, Amiya Kumar; HAO, Kangli; SULTANA, Salmin; BAGCHI, Saurabh. **Characterizing failures in mobile oses: a case study with Android and Symbian.** In: 21st International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2010, p. 1-10.

PROXXIMA. **Pesquisa: Android domina navegação mobile em 101 países**. Disponível em: <http://www.proxxima.com.br/home/proxxima/noticias/2014/03/26/pesquisa-android-domina-navegacao-mobile-em-67-paises.html>. Acesso em: 08 mai. 2016.

TECMUNDO. **Android Studio: ferramenta de criação de apps da Google ganha versão 1.0**. 2014. Disponível em: <http://www.tecmundo.com.br/android/69111-android-studio-ferramenta-criacao-apps-google-ganha-versao-1-0.htm>. Acesso em: 08 mai. 2016.

UOL. **Sistema operacional do Google conquista espaço no mercado de smartphones**. Disponível em: <http://android.uol.com.br/o-que-e-google-android.jhtm>. Acesso em: 08 mai. 2016.

VANDERSEN, Rogério Schueroff; MAGALHÃES, Willian Barbosa. **Conceitos e aplicações da computação em nuvem**. 2014. Disponível em: <http://ftp.unipar.br/~seinpar/2013/artigos/Rogerio%20Schueroff%20Vandresen.pdf>. Acesso em: 11 jun. 2016.