UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

DESENVOLVIMENTO DO APLICATIVO SPEEK

CURITIBA – PR
2017

LARISSA PEREIRA RAMOS DE OLIVEIRA

DESENVOLVIMENTO DO APLICATIVO SPEEK

Monografia apresentada como requisito parcial à obtenção do título de Especialista em Desenvolvimento para Dispositivos Móveis, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Campus Curitiba.

Orientador: Prof. Adriano Francisco Ronszcka

CURITIBA – PR

2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

“Desenvolvimento do Aplicativo Speek”

por

“Larissa Pereira Ramos de Oliveira”

Este Trabalho de Conclusão de Curso foi apresentado às 19:45 do dia 22 de novembro de 2017 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

_____ Prof. Robson Ribeiro Linhares **(Presidente - UTFPR/Curitiba)**	_____ Profa. Maria Claudia Figueiredo Pereira Emer **(Avaliador 1 – UTFPR/Curitiba)**
_____ Prof. Adriano Francisco Ronszcka **(Avaliador 2/Orientador – Externo)**	

“A Ata de Aprovação assinada encontra-se na Coordenação do Curso.”

RESUMO

Observa-se que nos últimos anos houve um aumento no número de pessoas que se alimentam fora de seus domicílios e, consequentemente, um aumento no número de estabelecimentos comerciais no ramo alimentício. Sendo assim, o desenvolvimento de um aplicativo de localização de restaurantes torna-se viável uma vez que o número de usuários em potencial é crescente. O objetivo geral deste trabalho consiste em desenvolver um aplicativo eficiente, rápido e focado em usabilidade, por meio do emprego das teorias apresentadas durante o curso de Desenvolvimento para Dispositivos Móveis. As atividades realizadas neste trabalho culminaram no desenvolvimento do aplicativo *SPeek*, o qual armazena informações relevantes de estabelecimentos alimentícios para que o usuário tenha ao seu alcance uma base de dados de restaurantes, na qual este poderá pesquisar sobre restaurantes abertos em um dado momento, restaurantes perto de sua localização ou os mais bem avaliados.

Palavras-chave: Android; Aplicativo; Restaurantes; Avaliações.

ABSTRACT

It has been observed that in recent years there was an increase in the number of people who feed outside their homes and, consequently, an increase in the number of commercial establishments in the food industry. Therefore, this application becomes feasible since the number of potential users is increasing. The general objective of this work consists of developing an efficient, fast and usability focused application, through the use of theories presented during the course of Development for Mobile Devices. The activities carried out in this work culminated in the development of the *SPeek* application, which stores relevant information from food establishments in order to deliver to the user a database of restaurants in which he can search about restaurants open at any given time, restaurants near his location or the best evaluated options.

Keywords: Android; App; Restaurants; Rates.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

APK – Android Package

IBGE – Instituto Brasileiro de Geografia e Estatística

IDC – International Data Corporation

IxD – Interaction Design

OMT – Object Modeling Technique

OOSE – Object-Oriented Software Engineering

REST – Representational State Transfer

SOAP – Simple Object Access Protocol

TDD – Test Driven Development

TI – Tecnologia da Informação

UML – Unified Modeling Language

XP – Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Segundo dados do IBGE, os gastos com alimentação representam mais de 16% da despesa total de famílias brasileiras. Este estudo, que foi elaborado entre os anos de 2008 e 2009, aponta também que, dentre os gastos com alimentação, o percentual de despesas da população com alimentação fora do domicílio, na área urbana, alcança mais de 33%, sendo que o maior percentual atingido com este tipo de alimentação ocorre na região Sudeste, ultrapassando 37% (IBGE, 2010).

O estilo de vida da população é um dos fatores que poderia justificar os dados obtidos nesse estudo, uma vez que habitantes de centros urbanos passam muito tempo em função do trabalho e/ou estudo, reduzindo assim, o tempo para cuidados com alimentação. Ademais, muitos habitantes não têm tempo para se deslocarem até suas casas para almoçarem, então o hábito de comer fora se torna conveniente para a economia de tempo.

Além disso, conforme os dados do censo demográfico de 2010 do IBGE, pode-se notar que a estrutura das famílias brasileiras tende a mudar em virtude do aumento do número de domicílios de pessoas solteiras e de casais sem filhos (IBGE, 2012). Pessoas com esse tipo de estrutura familiar, muitas vezes, preferem se alimentar fora ao invés de cozinhar para somente uma ou duas pessoas.

De acordo com a Abrasel (Associação Brasileira de Bares e Restaurantes), o setor de bares e restaurantes, atualmente, reúne aproximadamente um milhão de empresas e representa quase 3% do PIB brasileiro (Abrasel, 2017).

Dessa forma, com muitos estabelecimentos alimentícios disponíveis e com alta demanda de clientes em potencial, surge para o estabelecimento o problema de como lidar com a concorrência em relação aos demais lugares. Para o cliente, por outro lado, muitas opções pode ser tanto uma vantagem quanto um fardo no momento da escolha do local para se alimentar.

1.1 MOTIVAÇÃO

O aplicativo *SPeek* desenvolvido para este trabalho visa auxiliar a população em geral a obter informações sobre restaurantes bem como localizar e avaliar restaurantes. Sendo possível, assim, descobrir novos restaurantes, verificar

se o restaurante está aberto em dado momento, conferir o preço médio dos pratos, além de permitir que o usuário elogie ou critique um estabelecimento.

O nome *SPeek* é uma variação da expressão americana “*sneak peek*” que significa “espiadinha”, ou seja, no contexto do aplicativo, o cliente em potencial tem a oportunidade de obter informações sobre o restaurante antes de visitá-lo para tomar uma decisão se compensaria ou não se deslocar até o local.

Além disso, com esse aplicativo, donos de restaurantes têm a conveniência de divulgar o estabelecimento para atrair novos clientes, além de mensurar a popularidade do mesmo.

Apesar de ser possível buscar por restaurantes no Google Maps, o aplicativo *SPeek* é totalmente focado em restaurantes, assim não se faz necessário buscar por restaurantes em seu mapa, pois estes são apresentados ao usuário de acordo com sua localização assim que o aplicativo é aberto.

Outro diferencial do aplicativo *SPeek* é a quantidade de informações sobre os restaurantes, como a categoria em que se enquadra, bem como as formas de pagamento aceitas, o que não é apresentado de maneira clara e completa no Google Maps.

Além disso, nas próximas versões do aplicativo *SPeek*, haverá informações sobre os pratos dos restaurantes com fotos, valores e o tamanho do prato para que os usuários tenham noção de quantas pessoas um prato serve. Essas funcionalidades também não estão disponíveis no Google Maps.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho consiste em conciliar as teorias apresentadas durante o curso de Desenvolvimento para Dispositivos Móveis com o intuito de desenvolver um aplicativo de localização de restaurantes eficiente, rápido e focado em usabilidade.

1.2.2 Objetivos Específicos

Os objetivos específicos desse trabalho são:

a. Planejar uma interface intuitiva para o aplicativo para garantir a usabilidade;
b. Usar o perfil do Facebook para autenticação do usuário;
c. Integrar a funcionalidade de mapa nativa do Android para exibir restaurantes por localização;
d. Avaliar o aplicativo com respeito à usabilidade.

1.3 METODOLOGIA

Com a finalidade de desenvolver e concluir este trabalho, uma metodologia foi adotada composta das seguintes atividades:

- Consolidação do tema:
  - Nesta fase inicial, um questionário foi disponibilizado em redes sociais para validar a viabilidade da ideia do aplicativo. A partir disso foi possível definir o escopo da primeira versão do aplicativo.
- Levantamento de requisitos:
  - Com um escopo fechado, fez-se possível o levantamento de requisitos do aplicativo para que todas as funcionalidades deste escopo fossem contempladas.
- Arquitetura do sistema:
  - A partir do levantamento dos requisitos, foi possível desenhar uma arquitetura para o sistema para facilitar sua visão como um todo com o auxílio de diagramas da UML.
- Elaboração de *layout*:
  - Para garantir uma interface amigável e uma boa usabilidade do sistema, protótipos foram elaborados auxiliando na escolha do *layout* final.

- Implementação:
    - Por fim, após todas essas fases, deu-se início a implementação do sistema.

1.4 ORGANIZAÇÃO

Este trabalho está descrito em cinco capítulos. No Capítulo 2 são apresentados os conceitos fundamentais estudados para o desenvolvimento do aplicativo. O Capítulo 3, por sua vez, apresenta a descrição do projeto e as atividades desenvolvidas para a concepção do aplicativo. No Capítulo 4, particularmente, são apresentadas as futuras implementações caso seja comprovada a aceitação da ideia do aplicativo. Por fim, o Capítulo 5 apresenta as conclusões obtidas a respeito do desenvolvimento da primeira versão do aplicativo.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o embasamento teórico estudado pertinente para o desenvolvimento do aplicativo. Neste âmbito, a primeira seção apresenta a teoria dos métodos ágeis, focando no quadro *kanban* e no *Scrum*. Em seguida, explica-se o tema Design de Interação, destacando a utilização de *surveys* na etapa inicial do projeto, bem como protótipos de baixa fidelidade e de alta fidelidade. Mais adiante, apresenta-se uma visão geral de *Unified Modeling Language* (UML) e seus principais diagramas. Posteriormente, contextualiza-se sobre a linguagem de programação Java e serve como uma breve introdução para a seção seguinte, a qual apresenta o Sistema Operacional Android, que tem suas raízes na linguagem Java. E, por fim, contempla-se o emprego da linguagem de programação Java em *Web Services*.

2.1 MÉTODOS ÁGEIS

O modelo de ciclo de vida de desenvolvimento de sistemas chamado cascata ou, do inglês, *waterfall*, foi documentado em 1970 por Winston Walker Royce. Este modelo descreve uma metodologia linear para o curso do projeto de *software* com uma sequência de eventos como análise de requisitos, projeto e arquitetura do sistema, codificação e testes. Sendo assim, geralmente uma etapa somente se inicia quando a etapa anterior termina (LOTZ, 2013).

Porém, projetos de *software* raramente seguem um fluxo sequencial como descrito neste modelo, gerando contratempos e complicações para realização do projeto como um todo. Mudanças solicitadas pelo cliente na fase de implementação, por exemplo, seriam mais complexas de tratar, visto que as fases de análise e projeto já terminaram (RASMUSSON, 2017).

Como uma alternativa a esse modelo tradicional de gerenciamento de projetos, os métodos ágeis auxiliam a equipe de um projeto a reagir à imprevisibilidade do cotidiano profissional por meio do andamento incremental e iterativo do trabalho.

No ano de 2000, um grupo de pessoas envolvidas nas áreas de gestão de projetos e desenvolvimento de *software* se reuniu para discutir o processo de desenvolvimento seguindo o modelo tradicional e levantaram as consequências

geradas pela burocratização do processo e o excesso de formalização com documentações existentes nestas áreas. Isso, então, os fez debater sobre os benefícios que novos métodos poderiam oferecer (Project Builder, 2017).

A partir disso, em uma segunda reunião de 17 pessoas em 2001 originou-se o manifesto ágil que é atualmente um guia para a indústria de *software* sobre o que é considerado ágil e o que não é (BERNARDO, 2015). Este manifesto possui quatro valores, sendo eles:

- Indivíduos e interação entre eles mais que processos e ferramentas;
- *Software* em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

Além disso, o manifesto ágil reúne um conjunto de abordagens de desenvolvimento de *software* que é evidenciado por 12 princípios que são:

- A maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de *software* de valor;
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas;
- Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;
- Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho;
- Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara;
- *Software* funcional é a medida primária de progresso;

- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes;
- Contínua atenção à excelência técnica e bom *design*, aumenta a agilidade;
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito;
- As melhores arquiteturas, requisitos e *design* emergem de times auto organizáveis;
- Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajusta e aperfeiçoa seu comportamento de acordo.

Apesar do manifesto ágil ter sido criado somente em 2001, metodologias ágeis já existiam como, por exemplo, as metodologias de desenvolvimento *Extreme Programming* (XP) e *Scrum*. No entanto, o método cascata ainda prevalecia na época e somente começou a ser substituído por métodos ágeis após a criação do manifesto ágil (VARHOL, 2015).

A metodologia de desenvolvimento XP é uma metodologia ágil que possui um foco maior nas atividades de desenvolvimento. Esse método foi formulado em 1996 por Kent Beck e considera que o cliente é um elemento indispensável na implementação de um *software*, visto que ele é o maior interessado no produto final. Portanto, uma equipe XP trabalha rigorosamente com a prioridade definida pelo cliente (COHN, 2007).

Além disso, essa metodologia estabelece algumas práticas específicas de trabalho como TDD (*Test Driven Development*), *design* simples e *pair programming*, o que pode confundir a equipe uma vez que trabalhando com metodologias ágeis a mesma deveria ter mais autonomia e ser auto organizada ao invés de seguir estritamente práticas e prioridades (COHN, 2007).

2.1.1 Scrum

Outro método ágil e mundialmente conhecido e usado é o *Scrum*, o qual foi fundado em 1995 por Ken Schwaber e Jeff Sutherland. O *Scrum* é um *framework*

no qual é possível empregar diversos processos e técnicas para construção de produtos, porém ele é mais aplicado na área de tecnologia da informação para gerenciar desenvolvimento de *softwares* (KNIBERG & SKARIN, 2010).

O método *Scrum* propõe que as equipes de trabalho sejam pequenas, multifuncionais e auto organizadas. Além disso, sugere que o trabalho a ser realizado seja fragmentado em porções “entregáveis” classificadas em ordem de prioridade e com esforço estimado. Ademais, recomenda que o tempo de trabalho seja dividido em iterações de duração fixa – entre uma e quatro semanas – e ao final de cada iteração a porção “entregável” seja demonstrada ao cliente para que este possa validar o resultado e colaborar para otimização do plano de entrega por meio de seu *feedback* e, por fim, norteia que o processo seja otimizado por meio de uma retrospectiva após cada iteração (KNIBERG & SKARIN, 2010).

Portando, “... ao invés de um grande grupo gastando um monte de tempo construindo uma grande coisa, temos uma equipe pequena gastando um tempo curto construindo uma pequena coisa. Mas integrando regularmente para ver o todo.” (KNIBERG & SKARIN, 2010, p. 24).

Seguindo a sugestão que o trabalho deve ser fragmentado em porções “entregáveis”, no *Scrum*, existem fases chamadas *Sprints*, as quais representam um período de tempo em que as tarefas devem ser executadas, realizando assim um trabalho iterativo (Desenvolvimento Ágil, 2014).

Todas as tarefas a serem realizadas no projeto são adicionadas em uma lista denominada *Product Backlog*. Essa lista é priorizada pelo *Product Owner* em uma reunião chamada *Planning Meeting*, a qual acontece no início de cada *Sprint*. Nessa reunião, o time analisa quais tarefas são viáveis para implementação durante a próxima *Sprint* (Desenvolvimento Ágil, 2014).

Para que toda a equipe tenha conhecimento das atividades que estão sendo realizadas na *Sprint*, uma rápida reunião é realizada, geralmente no início da manhã. Essa reunião é chamada de *Daily Meeting* e com ela se faz possível “identificar impedimentos e priorizar o trabalho do dia que se inicia” (Desenvolvimento Ágil, 2014).

Com o intuito de apresentar ao *Product* Owner as funcionalidades desenvolvidas na *Sprint*, faz-se uma reunião chamada *Review Meeting*. E, por fim, faz-se uma *Retrospective Meeting*, na qual a equipe faz o levantamento de lições aprendidas na *Sprint*, o que poderia ser melhorado, o que deu certo para ser

mantido e a equipe parte para o planejamento da próxima *Sprint*, reiniciando, assim, o clico, conforme ilustra a Figura 1 (Desenvolvimento Ágil, 2014).

Figura 1 – Metodologia Scrum

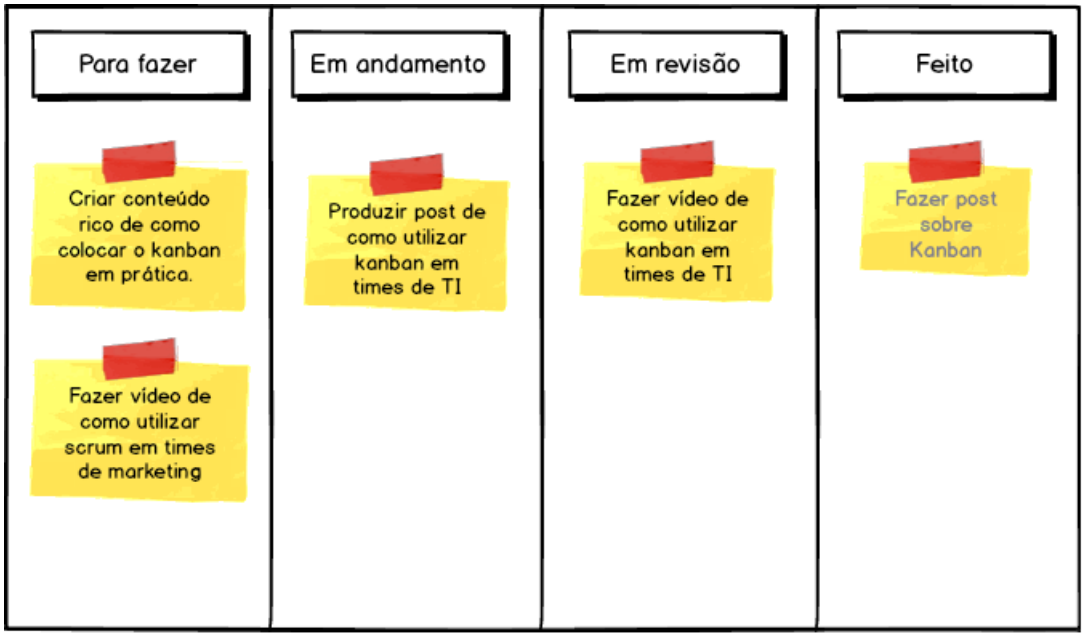
Fonte: Adaptado de Gomes (2017)

Geralmente, utilizando a metodologia *Scrum* faz-se uso também do quadro *kanban*, que é uma ferramenta para visualização da situação trabalho durante a *Sprint*. A palavra *kanban* é uma palavra da língua japonesa que significa “cartão”, então o quadro *kanban* é um quadro de cartões que auxilia a visualização do fluxo de trabalho (leankit, 2016).

O quadro *kanban* pode ser dividido em quatro colunas que são: a realizar, realizando, em revisão e pronto, conforme ilustra a Figura 2. Cada item de trabalho deve ser colocado em um cartão no quadro e na coluna que se adequa a sua situação. Com o quadro elaborado é possível acompanhar o tempo de execução de cada tarefa (leankit, 2016).

Segundo Bernardo (2014, p. 1), “ao olhar para um quadro *kanban* é fácil enxergar como o trabalho seu e de sua equipe fluem, permitindo não só comunicar o status, mas também dar e receber *feedbacks*”.

Figura 2 – Exemplo de Quadro Kanban



Fonte: Oliveira (2016)

2.2 DESIGN DE INTERAÇÃO

A disciplina *design* de interação estuda a interação entre produtos e usuários para garantir que essa relação seja adequada, simples e rápida. O objetivo é criar produtos com os quais o usuário atinja seus objetivos da melhor maneira possível. A interação entre um usuário e um produto geralmente envolve elementos como estética, movimento, som, espaço, entre outros (SIANG, 2017).

> O *design* de interação (IxD) define a estrutura e o comportamento de sistemas interativos. Os *designers* de interação se esforçam para criar relacionamentos significativos entre as pessoas e os produtos e serviços que eles usam, desde computadores até dispositivos móveis... (IxDA, 2017, tradução nossa).

A partir da observação das experiências e testes realizados com usuários, pode-se tomar decisões sobre conceitos e sobre a forma de apresentar informações para o usuário, criando assim, uma experiência significativa para este (SIANG, 2017).

Estas experiências e testes são realizados utilizando principalmente dois métodos de *design* de interação que são os questionários, do inglês, *survey* e a prototipação de baixa e de alta fidelidade (SIANG, 2017).

2.2.1 Survey

A pesquisa *survey* ou questionário é um método de pesquisa empírica para fazer levantamentos sobre um determinado tema. Esse é um método de pesquisa quantitativo para obtenção de informações sobre um tema definido por meio da coleta de dados de um grupo de pessoas (BABBIE, 2005).

> Pesquisa de *survey* se refere a um tipo particular de pesquisa social empírica, mas há muitos tipos de *survey*. O termo pode incluir censos demográficos, pesquisas de opinião pública, pesquisas de mercado sobre preferências do consumidor, estudos acadêmicos sobre preconceito, estudos epidemiológicos, etc (BABBIE, 2005, p. 95).

Segundo Pinsonneault & Kraemer (1993), o método de pesquisa *survey* é classificado como explanatório, uma vez que tem como objetivo testar uma teoria do autor do *survey*, como exploratório já que também tem a finalidade de familiarização com um determinado tema e, por fim, é também classificado como descritivo, pois procura identificar quais atitudes ou opiniões estão expostas em uma população.

A Figura 3 mostra um exemplo de uma pesquisa *survey* feita para avaliar a satisfação do cliente. A partir do resultado desta pesquisa, o autor poderá, por exemplo, obter informações sobre a qualidade de seu produto ou serviço e, assim, tomar decisões para melhorá-lo, caso seja necessário (Survio, 2017).

Figura 3 – Exemplo de Survey

Fonte: Survio (2017)

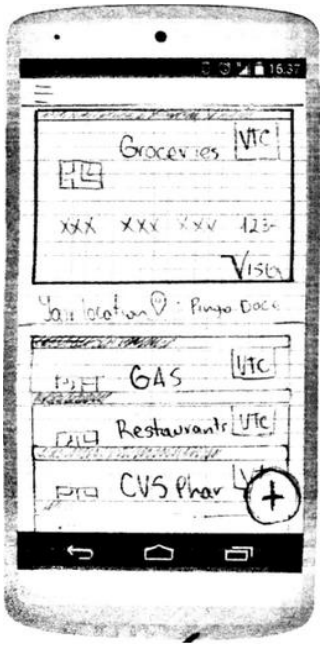
2.2.2 Protótipos de baixa fidelidade

Normalmente, no início de um projeto, o cliente expõe diversas ideias sobre como ele pretende que seja o resultado final do produto, mas frequentemente ainda não tem uma noção clara do que ele realmente deseja e precisa. Assim, para interpretar essas ideias, são criados protótipos de baixa fidelidade, que são um esboço da ideia inicial do *design* de um sistema (usability.gov, 2017).

Esse tipo de protótipo geralmente é criado de forma rápida e não é utilizado nenhum tipo de *software* nesse levantamento, mas sim papel e caneta. Desta forma, o processo de validação de ideias se torna mais natural e prático, uma vez que o cliente pode se sentir mais confortável para sugerir mudanças com esboços mal-acabados (usability.gov, 2017).

Quando finalizado, esse tipo de protótipo é utilizado para validar a interação do usuário com o sistema por meio de testes com pessoas que não estão envolvidas no desenvolvimento. O objetivo destes testes é verificar se a pessoa conseguiria manejar o sistema de forma correta. Esse tipo de protótipo, portanto, auxilia a identificar rapidamente os maiores obstáculos de usabilidade de um projeto (usability.gov, 2017).

Figura 4 – Exemplo de Protótipo de Baixa Fidelidade



Fonte: Kim (2017)

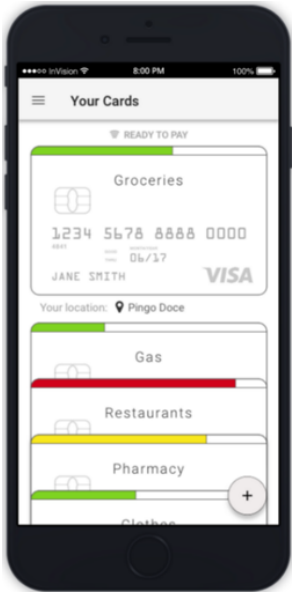
Neste âmbito, a Figura 4 mostra um protótipo de baixa fidelidade criado para uma idealização de um aplicativo de controle financeiro que mostra os cartões do usuário de forma virtual e separados por categorias, como compras, restaurantes e viagens. O aplicativo determina um limite de gastos para cada cartão e mostra quanto o usuário está economizando e gastando. Com esse protótipo, o autor pôde ter uma noção de como seria a interface de seu aplicativo e, além disso, pôde conduzir testes com usuários para evoluir seu protótipo (KIM, 2017).

2.2.3 Protótipos de alta fidelidade

Com o decorrer da fase de prototipação, as ideias originais do cliente se aperfeiçoam e se estabilizam, pois o cliente se familiariza com seu conceito de negócio e, consequentemente, com o produto final. Além disso, a partir dos testes com os usuários e protótipos de baixa fidelidade, são levantados pontos de melhorias e, assim, é possível evoluir tais protótipos (usability.gov, 2017).

Com essa evolução, são criados os chamados protótipos de alta fidelidade, que são praticamente o *design* final de um sistema e são criados utilizando algum *software*, ou seja, necessitam de mais tempo e recurso para serem construídos (usability.gov, 2017).

Figura 5 – Exemplo de Protótipo de Média Fidelidade



Fonte: Kim (2017)

Frequentemente, entre a criação do protótipo de baixa fidelidade e o de alta fidelidade pode haver uma versão intermediária que é chamada de protótipo de média fidelidade, conforme ilustra a Figura 5, que demonstra uma melhoria da versão esboçada na Figura 4. Essa nova versão apresenta algumas cores e estilos, porém ainda não é uma versão final do *design* do aplicativo de controle financeiro. Com essa versão intermediária, é possível fazer uma nova rodada de testes com os usuários para levantar novos pontos de melhorias (usability.gov, 2017).

Os protótipos de alta fidelidade, por outro lado, são os que mais se aproximam do *design* da versão final do produto, conforme apresenta a
Figura 6, que ilustra o resultado final do *design* do aplicativo de controle financeiro (KIM, 2017).

Por meio do *feedback* a cada iteração de testes com cada protótipo, o autor do aplicativo pôde constatar que os usuários preferiram uma cor sólida na barra indicadora de gastos, um texto alinhado a direita para as categorias de compras e cores escuras variadas para o cartão, como mostra a
Figura 6 (KIM, 2017).

Figura 6 – Exemplo de Protótipo de Alta Fidelidade

Fonte: Kim (2017)

No caso de protótipos de algum *software*, pode-se apresentar ao cliente um protótipo funcional fiel ao *design* do produto final e, assim, o cliente pode

interagir com esse protótipo validando, por exemplo, a navegabilidade neste *software*.

2.3 UML

Na década de 90, existiam diversos métodos de modelagem orientada a objeto como OMT (*Object Modeling Technique*) criado por Rumbaugh em 1991, Booch criado em 1994 por Grady Booch e OOSE (*Object-Oriented Software Engineering*) criado em 1992 por Jacobson.

Nessa época, muitos usuários destes métodos tiveram dificuldades em suprir suas necessidades com um único método. O OMT era o melhor método para análise e sistemas de informação em dados intensivos, porém o Booch era melhor para desenho do projeto e implementação (Visual Paradigm, 2017).

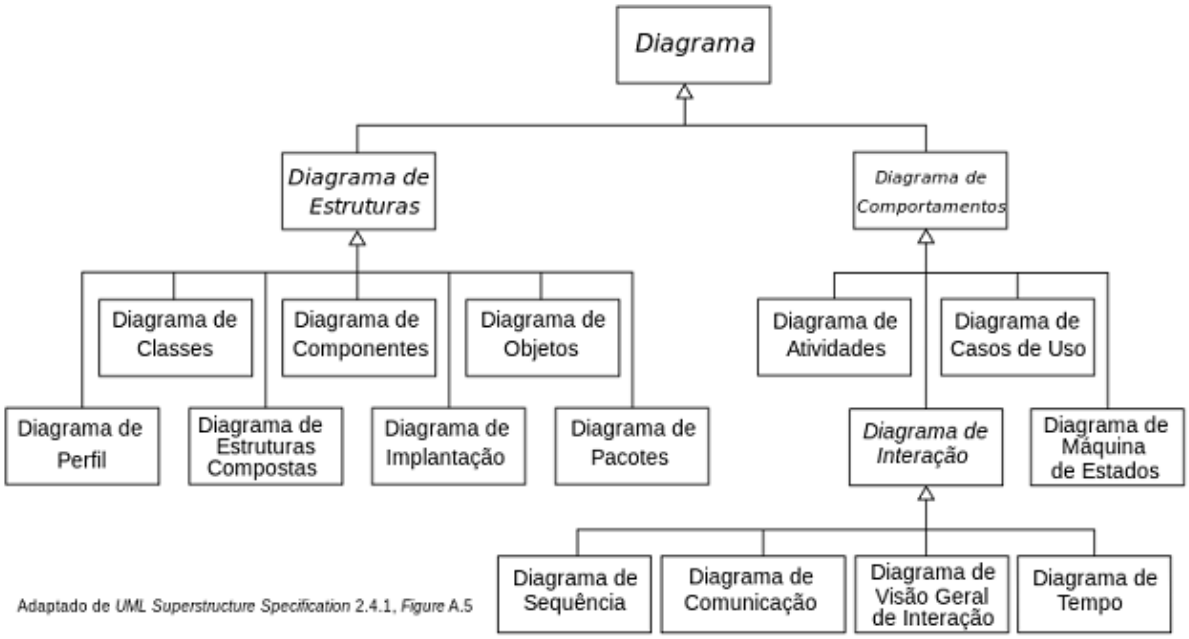
Devido a essa dificuldade, a necessidade de uma notação única e padronizada se tornou mais evidente e assim começou o desenvolvimento da UML no final de 1994, quando Jim Rumbaugh saiu da empresa General Electric e começou a trabalhar na Rational Corp juntamente com Grady Booch.

A UML representa um conjunto de melhores práticas para garantir o sucesso na modelagem de sistemas complexos. A linguagem UML é, então, uma linguagem padronizada de modelagem construída por um conjunto de diagramas e elaborada para auxiliar desenvolvedores de *software* a especificar, visualizar, construir e documentar os artefatos do sistema (Visual Paradigm, 2017).

Os diagramas da UML estão divididos em duas categorias, os diagramas comportamentais e os diagramas estruturais, conforme ilustra a Figura 7.

Na categoria de diagramas de estruturas existem sete diagramas, que são: diagrama de perfil, de classe, de estruturas compostas, de componentes, de implantação, de objetos e de pacotes. Na categoria de diagramas de comportamentos também existem sete diagramas, que são: diagrama de atividades, de casos de uso, de máquina de estados, de sequência, de comunicação, de visão geral de integração e de tempo.

Figura 7 – Diagramas da UML



Fonte: Ventura (2016)

2.3.1 Diagramas Comportamentais

Os diagramas comportamentais mostram o comportamento dinâmico dos objetos em um sistema, o qual pode ser descrito como transições no sistema ao longo do tempo (FAKHROUTDINOV, 2009).

Eles são responsáveis por demonstrar as funcionalidades de um sistema computacional em tempo de execução, como por exemplo, um fluxo de autenticação em um sistema.

2.3.1.1 Diagrama de Casos de Uso

O diagrama de casos de uso se enquadra na categoria de diagramas comportamentais. O objetivo desse diagrama é descrever o conjunto de funções que um sistema computacional pode realizar através da interação de um ator (MALCHER, 2016).

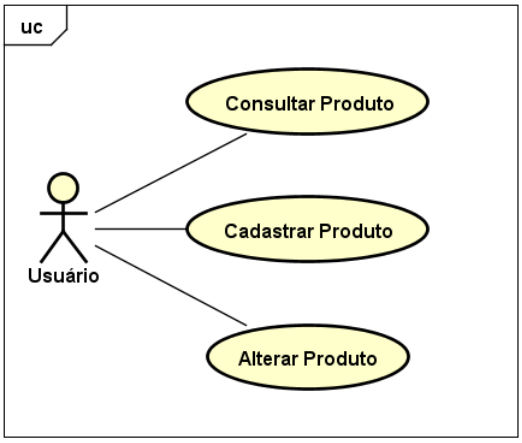
Um caso de uso especifica uma sequência de ações que o sistema desempenha, gerando resultados visíveis de interesse para um ator. Um ator

representa os papéis desempenhados pelos usuários de um caso de uso. Esse ator pode ser uma pessoa, um outro sistema ou um dispositivo (NOGUEIRA, 2005).

A Figura 8 exemplifica um diagrama de casos de uso no qual o ator é o usuário de um sistema que tem as funcionalidades de consultar uma base de produtos, cadastrar um novo produto ou realizar alguma alteração em um determinado produto.

Cada caso de uso pode ser mais especificado em um documento que descreve o caso de uso, o ator principal – e secundário, se necessário – as pré e pós-condições do caso de uso, as regras de negócio e os fluxos principal e alternativos do cenário (NOGUEIRA, 2005).

Figura 8 – Exemplo de Diagrama de Casos de Uso



Fonte: Elaborada pela autora

2.3.1.2 Diagrama de Atividades

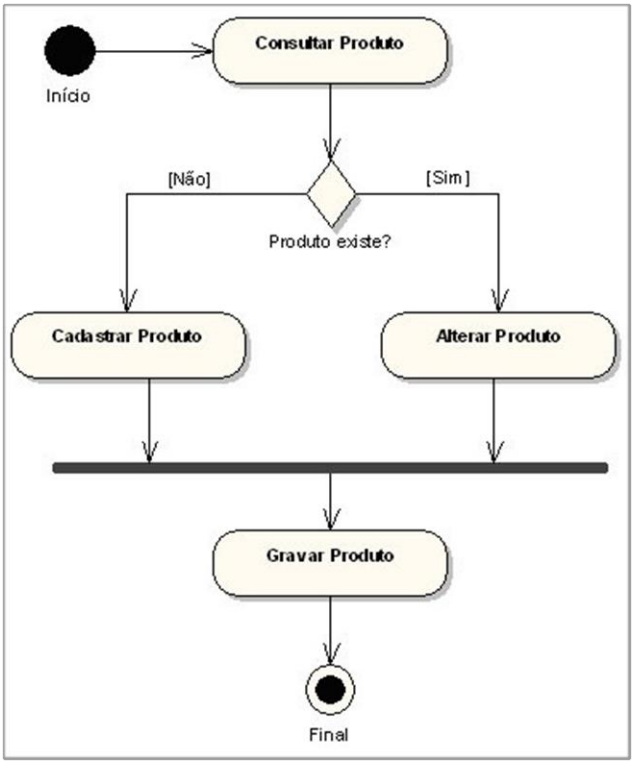
O diagrama de atividades também se enquadra na categoria de diagramas comportamentais. O objetivo desse diagrama é apresentar o fluxo de um sistema computacional com ênfase na sequência de atividades e condições deste fluxo (VENTURA, 2016).

> Um diagrama de atividade é uma maneira alternativa de se mostrar interações, com a possibilidade de expressar como as ações são executadas, o que elas fazem (mudanças dos estados dos objetos), quando elas são executadas (sequência das ações), e onde elas acontecem (os divisores ou *swimlanes*) (NOGUEIRA, 2005).

Um diagrama de atividades segue a convenção de desenhos para determinar o que representa o início e o final do fluxo, a atividade, condições, entre outros (VENTURA, 2016).

Neste âmbito, a Figura 9 apresenta um diagrama de atividades, no qual o fluxo se inicia ao se consultar um produto no sistema, passando por uma validação da condição se o produto consultado existe no sistema ou não. Caso o produto não exista, o fluxo é direcionado para cadastrá-lo. Caso contrário, o fluxo é direcionado para a alteração do produto. Porém, independente da direção do fluxo, após a operação de cadastro ou atualização, ocorre a atividade de armazenar os dados do produto finalizando o fluxo (VENTURA, 2016).

Figura 9 – Exemplo de Diagrama de Atividades



Fonte: Adaptado de Nogueira (2005)

2.3.2 Diagramas Estruturais

Os diagramas estruturais retratam a organização estática de um sistema que está sendo modelado, limitando-se aos componentes do sistema. Essa organização é demonstrada pelos tipos e instâncias no sistema (BELL, 2016).

> Existem para visualizar, especificar, construir e documentar os aspectos estáticos de um sistema, ou seja, a representação de seu esqueleto e estruturas “relativamente estáveis”. Os aspectos estáticos de um sistema de *software* abrangem a existência e a colocação de itens como classes, interfaces, colaborações, componentes (VARGAS, 2008, p. 34)

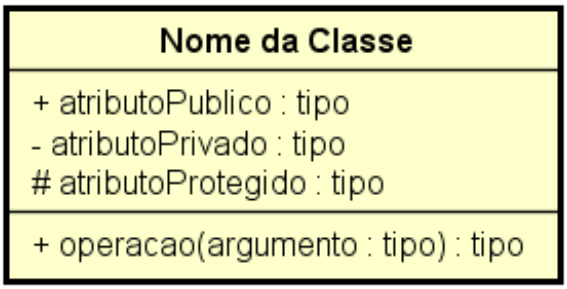
Esses diagramas mostram a estrutura estática de um sistema e seus elementos em diferentes níveis de abstração e implementação, além de mostrar como esses elementos estão relacionados (FAKHROUTDINOV, 2009).

2.3.2.1 Diagrama de Classes

O diagrama de classes é um dos diagramas estruturais mais fundamentais para modelagem de um sistema, uma vez que “o propósito do diagrama de classes é mostrar os tipos que estão sendo modelados no sistema” (BELL, 2016).

Este é o diagrama que modela a visão estática de um sistema. Nele, uma classe é representada por um retângulo com três fragmentos, conforme ilustra a Figura 10. O primeiro fragmento corresponde ao nome da classe, o segundo aos atributos dessa classe e o terceiro às assinaturas das operações realizadas por essa classe.

Figura 10 – Representação de Classe



Fonte: Elaborada pela autora

2.4 JAVA

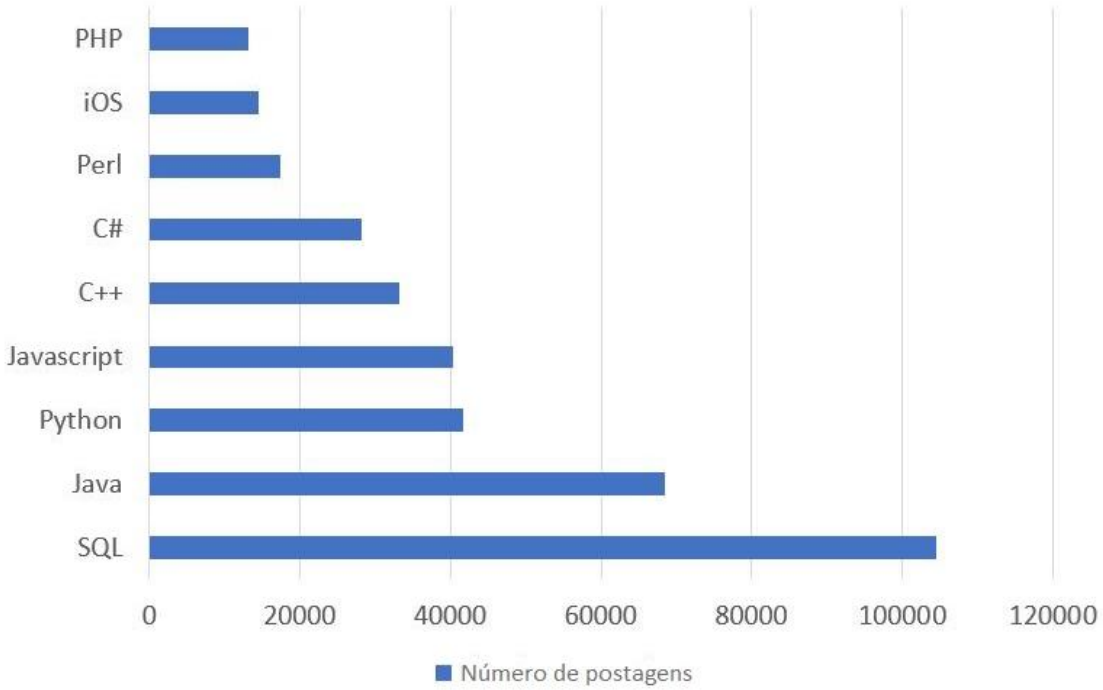
A Sun Microsystems era uma empresa americana que vendia serviços de tecnologia da informação (TI), além de *software*, computadores e seus componentes. Em 1991, um grupo de engenheiros desta empresa chamado “Green

Team” acreditava que a união de dispositivos de consumo digital e computadores era o futuro da computação e, assim, liderados por James Gosling, esse grupo de engenheiros iniciou o projeto da linguagem de programação (Oracle, 2017).

Esse grupo demonstrou para a indústria de televisão a cabo digital a nova linguagem através de um controlador de entretenimento doméstico interativo e portátil. Porém, não obteve sucesso, pois, infelizmente, o conceito era muito avançado para a equipe na época, mas era apropriado para a Internet. E então, em 1995, a equipe anunciou que a tecnologia Java estaria incorporada no navegador da Internet *Netscape Navigator*. Atualmente, a linguagem Java é aplicada não somente a Internet, mas em diversas aplicações e dispositivos comumente utilizados pela população (Oracle, 2017).

A linguagem de programação Java é orientada a objetos e amplamente utilizada no desenvolvimento de sistemas *web*, *desktop* e *mobile*, conforme apresenta a Figura 11, pois aplicações Java são rápidas e estáveis, suprindo, assim, as necessidades de tais sistemas. Além disso, o Java usufrui da vantagem de ser uma linguagem portável, podendo, então, ser utilizada em vários sistemas operacionais.

Figura 11 – Ranking de Linguagens de Programação



Fonte: Adaptado de Coding Dojo (2017)

A Figura 11 apresenta uma classificação de linguagens de programação feita pelo *site* Coding Dojo (2017). Existem diversas formas de classificar linguagens de programação, como número de *sites* feitos com dada linguagem, resultados de buscas no Google (2017), projetos disponíveis no GitHub (2017) ou questionamentos feitos por meio do site StackOverflow (2017), porém a classificação feita pelo *site* foi gerada através da análise dos dados do mecanismo de pesquisa de vagas de trabalho Indeed (2017). Eles analisaram o número de postagens de vagas de trabalho que continham o nome das linguagens de programação (Coding Dojo, 2017). Assim, pode-se concluir, portanto, que em 2017 a linguagem de programação Java é a segunda linguagem em maior demanda no mercado de trabalho em escala mundial.

O mesmo levantamento foi realizado pelo *site* Coding Dojo em 2016 e foi constatado que o número de vagas de trabalho que continham o nome da linguagem de programação Java aumentou em quase 30 mil em 2107. Isso ocorreu, provavelmente, por causa do aumento de usuários no mercado do sistema operacional Android, abordado em seguida na Seção 2.5. Uma vez que todo aplicativo nativo Android utiliza a linguagem Java para ser desenvolvido, o número de vagas de trabalho que têm esse requisito também aumentou (Coding Dojo, 2017).

2.5 ANDROID

O sistema operacional móvel Android foi desenvolvido inicialmente por uma empresa chamada Android Inc, a qual em 2005 foi comprada pela empresa Google (2017). Em 2007, o Android foi lançado pela Google, em uma época que o iOS, sistema operacional móvel da empresa Apple (2017), prevalecia no mercado (SpinFold, 2016).

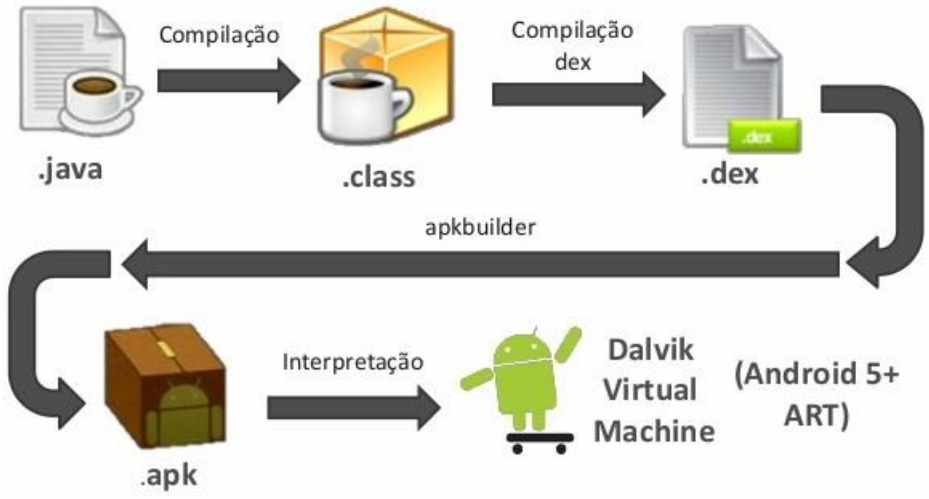
Android é um sistema operacional móvel baseado no núcleo do sistema operacional Linux. Atualmente, o Android é o sistema operacional móvel mais utilizado no mundo, com estimativas de aumento no número de usuários na casa de milhões diariamente (GOOGLE, 2017).

O desenvolvimento de aplicativos nativos para o sistema operacional móvel Android é baseado na linguagem de programação Java, uma vez que

diversas bibliotecas Java estão disponíveis na plataforma Android. Já as bibliotecas Java que não são suportadas na plataforma Android, ou possuem uma substituição melhor, ou não são necessárias, como por exemplo, as bibliotecas de impressão.

A Figura 12 apresenta o processo de compilação de um aplicativo Android. O código-fonte é compilado e empacotado em um Android Package (APK) junto com recursos, ativos e dependências (PATRICK, 2017). O arquivo de extensão .apk gerado a partir deste processo de compilação pode, então, ser distribuído para uso principalmente através da loja Google Play (2017), que é a loja oficial de aplicativos Android da empresa Google (2017).

Figura 12 – Visão Geral da Compilação Android



Fonte: Adaptado de Macedo (2014)

2.5.1 Versões Android

Atualmente, existem diversas versões disponíveis do sistema operacional móvel Android. Desenvolver aplicativos que funcionem em todas as versões seria um trabalho que demandaria muito tempo e esforço, pois versões mais antigas do Android podem não ter certas funcionalidades necessárias para o desempenho adequado do aplicativo que está sendo desenvolvido (HELPPI, 2014).

Assim, faz-se necessário escolher uma versão base para o aplicativo a ser desenvolvido. De modo a auxiliar os desenvolvedores na escolha da versão base para seus aplicativos, a empresa Google mantém um painel de controle de

versões do Android e o respectivo percentual de dispositivos que utilizam cada versão, como mostra a Figura 13.

Figura 13 – Uso das Versões do Android

Versão	Nome	API	Distribuição
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.3%
4.2.x		17	3.3%
4.3		18	1.0%
4.4	KitKat	19	14.5%
5.0	Lollipop	21	6.7%
5.1		22	21.0%
6.0	Marshmallow	23	32.0%
7.0	Nougat	24	15.8%
7.1		25	2.0%
8.0	Oreo	26	0.2%

Fonte: Adaptado de Google (2017)

Os dados são coletados durante um período de sete dias e os últimos dados foram coletados em outubro de 2017. Além disso, versões do Android que representam menos de 0.1% de dispositivos não são mostradas no painel (GOOGLE, 2017).

2.6 WEB SERVICES

Além de Java ser a linguagem nativa do Android, segundo o site Coding Dojo (2017), 90% das empresas que estão classificadas na Fortune 500 (2017) usam esta linguagem de programação para desenvolvimento *server-side*.

O desenvolvimento de aplicativos móveis muitas vezes envolve a comunicação entre duas partes – *server* e *client* – por meio, por exemplo, do protocolo HTTP. O *server* é a parte responsável por fornecer as informações e o *client* é a parte que requisita tais informações do *server* para as exibir ao usuário.

Assim, a parte *server* é comumente chamada de *server-side* e a parte *client* é chamada de *client-side*. E, no caso do desenvolvimento de um aplicativo Android, o aplicativo em si seria o *client-side* e um *web service* que busca informações de um banco de dados, por exemplo, seria o *server-side*.

Um *web service*, por sua vez, é a parte *server* responsável pela integração e comunicação entre diferentes aplicações através de um conjunto de operações previamente definidas utilizando, por exemplo, o protocolo de comunicação SOAP (*Simple Object Access Protocol*) ou o estilo arquitetural REST (*Representational State Transfer*).

O termo REST foi definido por Roy Thomas Fielding em sua tese de Doutorado (FIELDING R. T., 2000). REST é um estilo arquitetural de se projetar sistemas distribuídos e contempla um conjunto de restrições como ser *stateless*, ter um relacionamento *client/server* e uma interface uniforme (Spring, 2017). Além disso, esse estilo arquitetural segue quatro princípios que são:

- Os recursos expõem URIs de estrutura de diretório facilmente compreensíveis;
- Representações transferem JSON ou XML para representar objetos e atributos de dados;
- As mensagens usam métodos HTTP explicitamente (por exemplo, GET, POST, PUT e DELETE);
- As interações não mantêm contexto algum de cliente no servidor entre solicitações. O cliente possui o estado da sessão.

O estilo REST utiliza os métodos HTTP para fazer requisições ao servidor. Os métodos definidos no documento RFC 2616 (FIELDING, et al., 1999) são: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE e PATCH. Porém, os métodos mais comumente utilizados são: GET, POST, PUT e DELETE.

O método GET é usado para recuperar qualquer informação. Já o método POST é usado para criar ou atualizar alguma informação, assim como o método PUT. E, por fim, o método DELETE é usado para solicitar que um recurso seja removido (Spring, 2017).

3 PROJETO

Este capítulo apresenta a fase de desenvolvimento do aplicativo. Para isso, primeiramente, a Seção 3.1 apresenta a idealização do aplicativo bem como o porquê da escolha da plataforma Android e sua versão base. Em seguida, a Seção 3.2 retrata os resultados de uma pesquisa de mercado criada para validar a ideia do aplicativo. Já a Seção 3.3 apresenta a metodologia usada no desenvolvimento do aplicativo. A Seção 3.4, por sua vez, aponta os requisitos funcionais e não funcionais levantados para este desenvolvimento, o qual segue a modelagem descrita na Seção 3.5 para implementar as funcionalidades do aplicativo listadas na Seção 3.7. E, por fim, a Seção 3.6 retrata a análise de *layout* feita para a interface do aplicativo.

3.1 APLICATIVO SPEEK

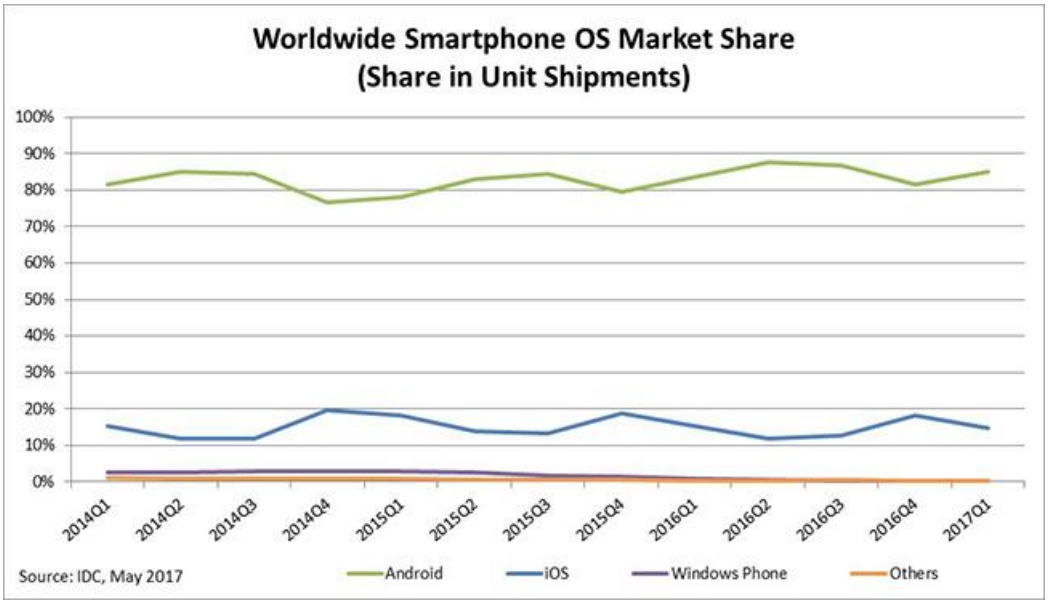
O aplicativo *SPeek* surgiu a partir da percepção da falta de informações sobre restaurantes da cidade de Curitiba, como também das demais cidades observadas pela autora.

A partir desta percepção, foram idealizadas funcionalidades para o aplicativo que auxiliariam pessoas a obter informações sobre restaurantes antes de visitá-los. Dentre estas funcionalidades estão a localização do restaurante, o preço médio dos pratos e a avaliação do restaurante feita pelos usuários do aplicativo.

Para conseguir um maior número de usuários a partir do lançamento do aplicativo, tomou-se a decisão de desenvolvê-lo inicialmente para Android e, futuramente, caso se comprove a aceitação do conceito, iniciar o desenvolvimento para iOS.

Esta decisão se baseou no fato de existirem mais usuários de Android do que de iOS, conforme ilustra o gráfico da Figura 14. De acordo com a *International Data Corporation* (IDC), a discussão sobre a quota de mercado do sistema operacional (SO) Android se tornou irrelevante alguns anos atrás quando ficou claro que esse sistema continuará a capturar aproximadamente 85% do volume mundial de smartphones (International Data Corporation, 2017).

Figura 14 – Quota de Mercado de SOs



Fonte: International Data Corporation (2017)

Além disso, conforme apresentado na Seção 2.5.1, apenas 22.3% dos usuários do sistema Android utilizam versões abaixo da versão *Lollipop*, a qual sozinha é utilizada por 27.7% dos usuários deste sistema. Sendo assim, tomou-se a decisão de desenvolver o aplicativo *SPeek* com a *Lollipop* como a versão mínima suportada, ou seja, dispositivos com essa ou uma versão superior são capazes de rodar o aplicativo deste trabalho.

3.2 PESQUISA

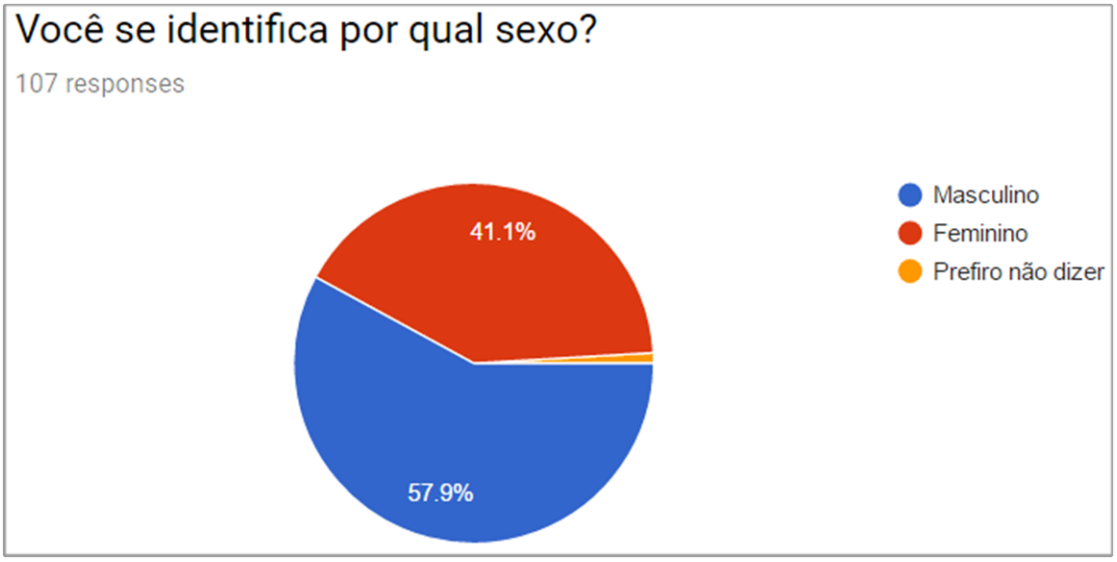
Para validar a ideia do aplicativo foi aplicado um questionário seguindo o conceito de *"survey"*, conforme apresentado na Seção 2.2.1. O questionário foi criado utilizando o Google Forms, possui dez questões e está acessível em: https://goo.gl/forms/3XoI7yASdiCBcKYl1.

O questionário foi disponibilizado em redes sociais, como o Facebook e o LinkedIn, para tentar atingir um público de diferentes faixas etárias, de diferentes localidades e, principalmente, não relacionado a área de tecnologia da informação.

Até o momento, existem 107 respostas. No questionário há perguntas para tentar identificar o público atingido, como por exemplo, as perguntas "Você se

identifica por qual sexo?”, “Qual sua idade?” e “Qual estado você reside?”, conforme Figura 15, Figura 16 e Figura 17.

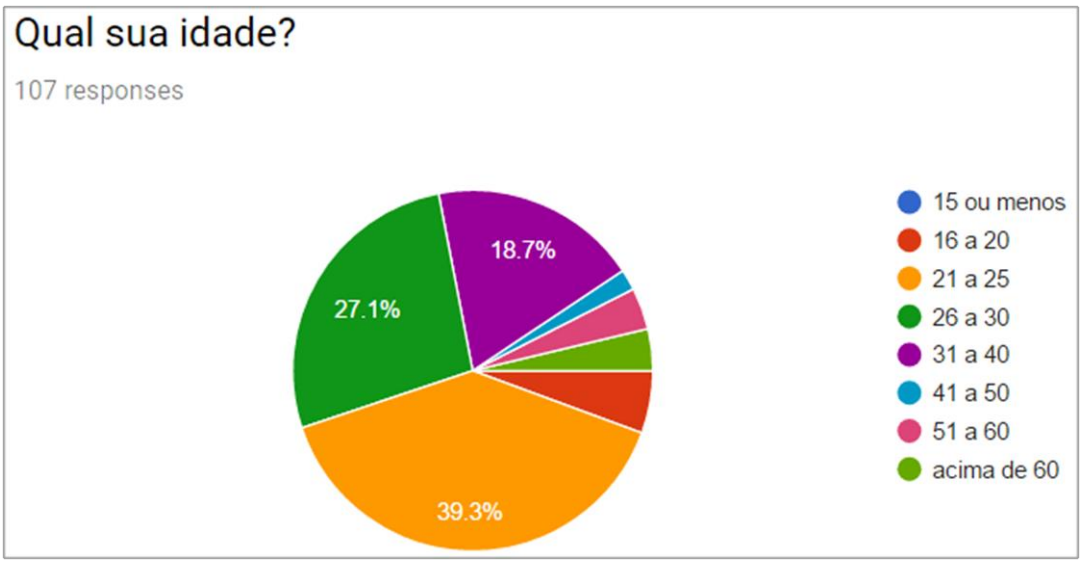
Figura 15 – Respostas da Pergunta Um



Fonte: Elaborada pela autora

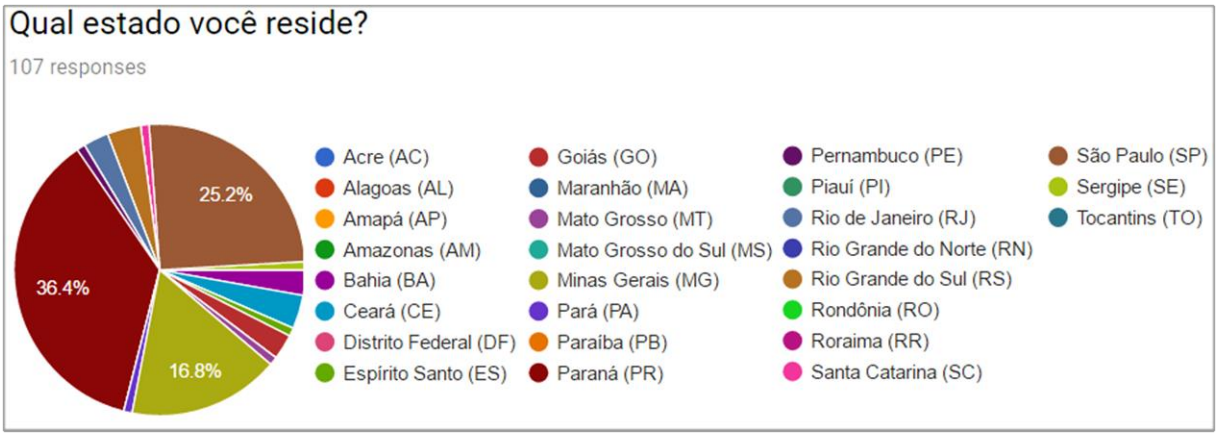
Com essas respostas, é possível, por exemplo, criar uma campanha publicitária mais focada para o público do sexo masculino, de 21 a 30 anos, residente das regiões Sudeste e Sul.

Figura 16 – Respostas da Pergunta Dois



Fonte: Elaborada pela autora

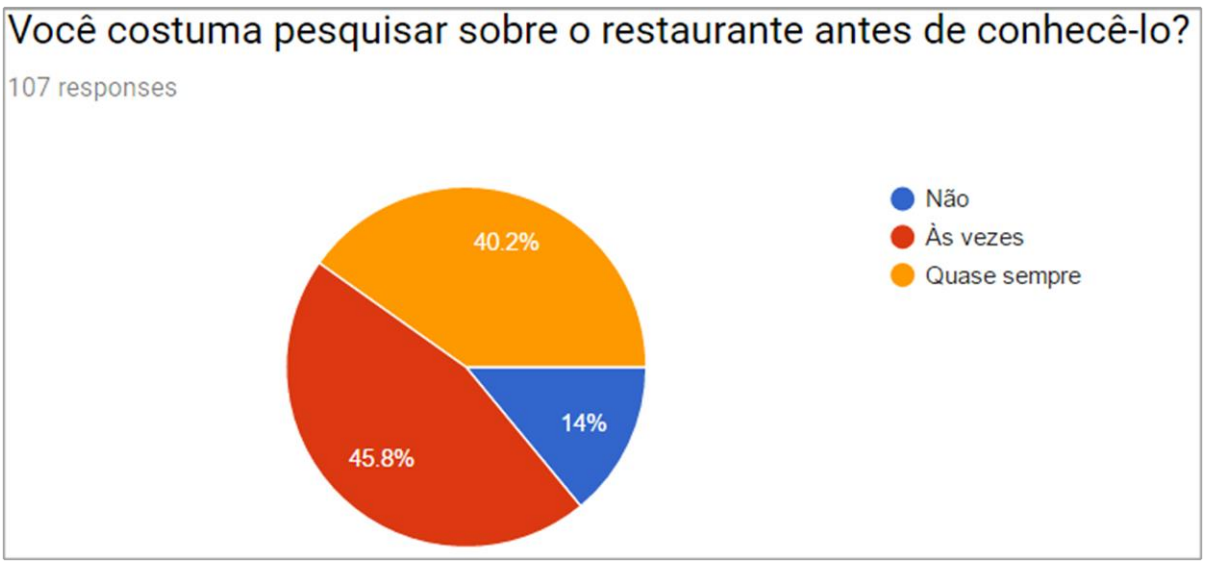
Figura 17 – Respostas da Pergunta Três



Fonte: Elaborada pela autora

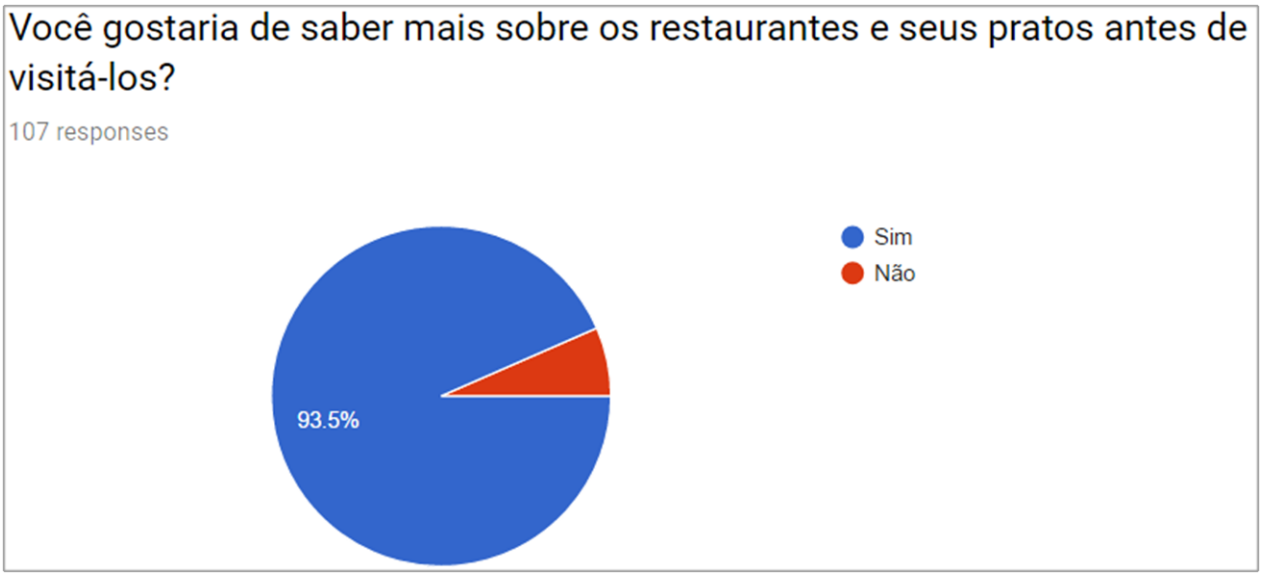
Além disso, há também perguntas para validar se existiriam usuários para o aplicativo, como por exemplo, as perguntas “Você costuma pesquisar sobre o restaurante antes de conhecê-lo?”, “Você gostaria de saber mais sobre os restaurantes e seus pratos antes de visitá-los?” e “Você gostaria de saber os restaurantes em funcionamento próximo a sua localização?” conforme Figura 18, Figura 19 e Figura 20.

Figura 18 – Respostas da Pergunta Sete



Fonte: Elaborada pela autora

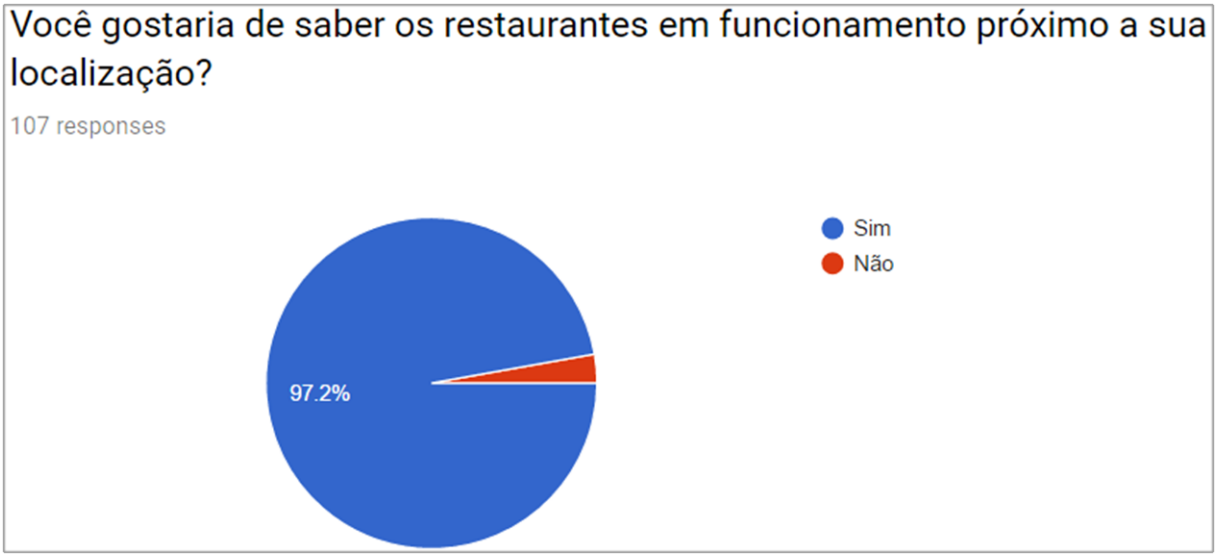
Figura 19 – Respostas da Pergunta Nove



Fonte: Elaborada pela autora

Com os dados obtidos pela pesquisa, foi possível analisar as informações e concluir que a maioria das pessoas que responderam o questionário são usuários em potencial do aplicativo, uma vez que 93.5% das pessoas têm interesse em obter informações sobre o restaurante antes de visitá-lo, conforme Figura 19 e 97,2% das pessoas gostariam de localizar restaurantes próximos a elas, conforme Figura 20. Assim, as principais funcionalidades do aplicativo atenderiam esse público.

Figura 20 – Respostas da Pergunta Dez



Fonte: Elaborada pela autora

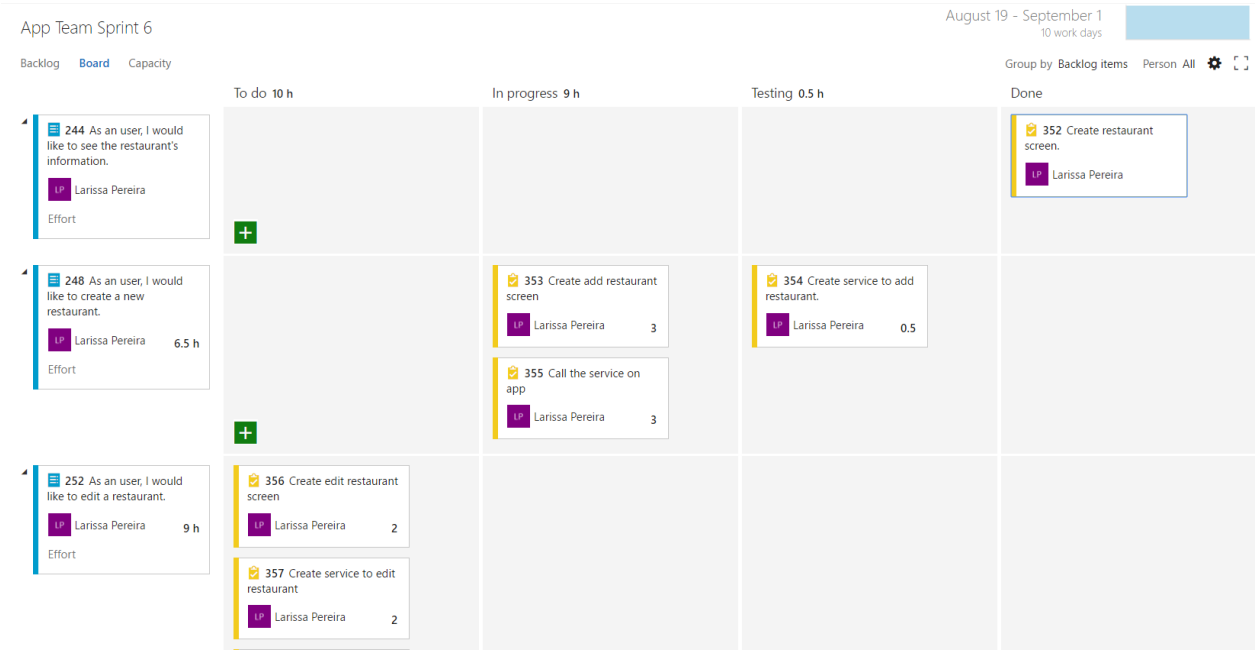
3.3 METODOLOGIA

A partir da validação positiva da ideia do aplicativo através da pesquisa realizada pelo Google Forms, iniciou-se a fase de desenvolvimento do aplicativo *SPeek*. Para isso, foi utilizada a metodologia ágil *Scrum* com auxílio do quadro *kanban* como ferramenta de gestão.

O escopo para a primeira versão do aplicativo consiste em cadastrar e editar restaurantes, avaliá-los e exibi-los no mapa, bem como em forma de lista, além da integração com o Facebook.

Assim, todas as tarefas a serem realizadas para contemplar esse escopo de modo a concluir o projeto foram cadastradas no Visual Studio Team Services, uma ferramenta da Microsoft gratuita para até cinco usuários no mesmo time. Essas tarefas foram divididas em *Sprints*, conforme descrito na Seção 2.1.1, e cada *Sprint* teve a duração de 14 dias totalizando aproximadamente 20 horas por *Sprint*.

Figura 21 – Quadro Kanban SPeek



Fonte: Elaborada pela autora

A Figura 21 mostra o quadro *kanban* para o desenvolvimento das tarefas de uma das *Sprints* do projeto, na qual a meta era implementar as funcionalidades de cadastrar e editar restaurantes para seguir um cronograma estipulado no início do

projeto, no qual estas duas funcionalidades deveriam estar concluídas até o final do mês de agosto.

3.4 REQUISITOS

Nesta seção estão indicados os requisitos funcionais e não funcionais levantados para o desenvolvimento do aplicativo *SPeek*.

3.4.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais do aplicativo são:

- O usuário pode se autenticar no aplicativo utilizando um endereço de e-mail e senha ou utilizando a integração com o Facebook;
- O usuário pode se cadastrar no aplicativo utilizando um endereço de e-mail e senha ou utilizando a integração com o Facebook;
- O aplicativo deve manter o usuário autenticado no aplicativo até que o usuário escolha sair do aplicativo;
- O usuário pode fazer *logoff*;
- O aplicativo deve mostrar os restaurantes no mapa;
- O aplicativo deve mostrar uma lista de restaurantes;
- O aplicativo deve mostrar detalhes do restaurante selecionado pelo usuário na lista ou no mapa;
- O usuário pode cadastrar um novo restaurante;
- O usuário pode editar um restaurante cadastrado;
- O usuário pode avaliar e comentar sobre um restaurante cadastrado;
- O usuário pode compartilhar informações do restaurante cadastrado no Facebook.

3.4.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais do aplicativo são:

- O *web service* deve ser implementado utilizando a linguagem Java;
- O aplicativo deve ser desenvolvido de forma nativa Android;

- O aplicativo poderá ser utilizado em qualquer *smartphone* que tenha a versão *Lollipop* (5.0) ou superior.

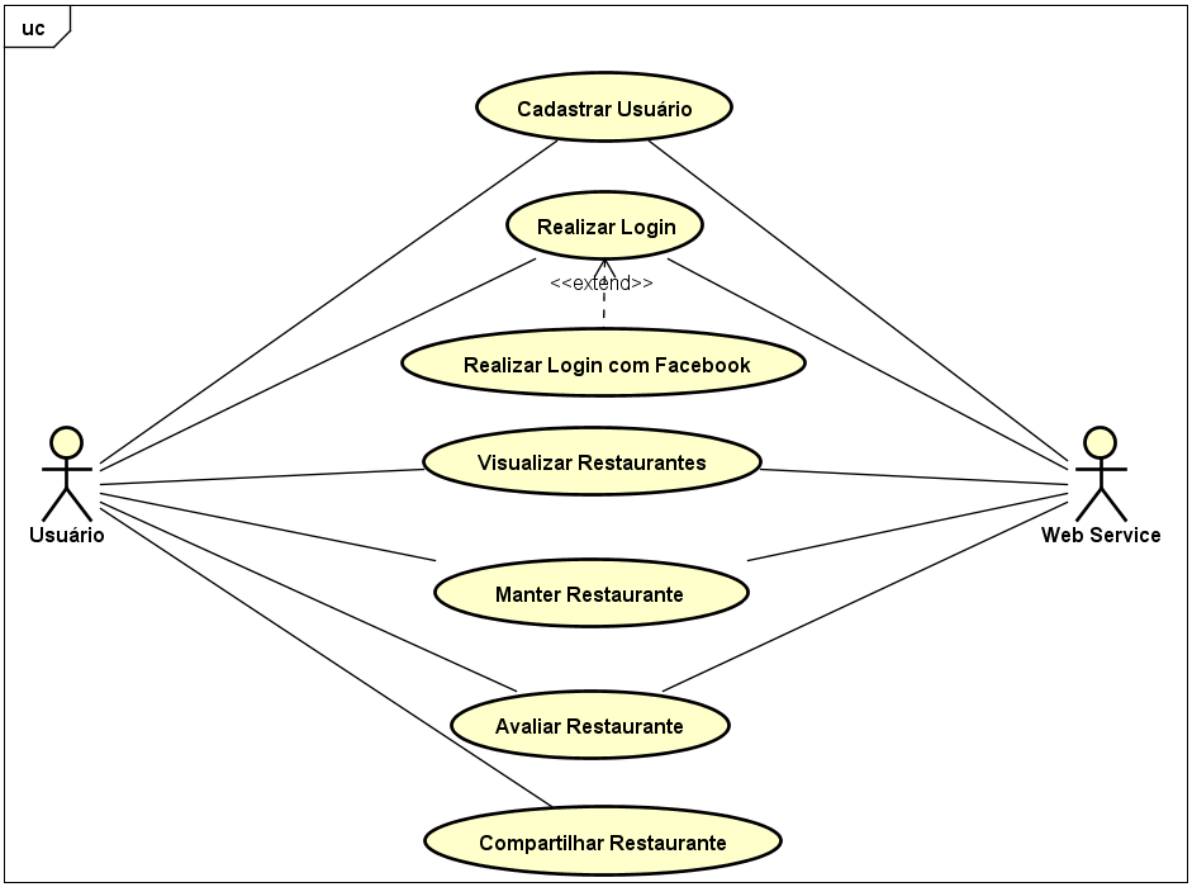
3.5 MODELAGEM

Nesta seção são apresentados os diagramas de casos de uso, de atividades e de classes elaborados utilizando a modelagem UML.

3.5.1 Diagrama de Casos de Uso

A partir do levantamento de requisitos funcionais foi modelado o diagrama de casos de uso do aplicativo para especificar as ações que o sistema pode desempenhar através da interação de um ator, conforme apresentado na Seção 2.3.1.1.

Figura 22 – Diagrama de Casos de Uso



Fonte: Elaborada pela autora

O diagrama de casos de uso exibido na Figura 22 contempla todas as operações que podem ser realizadas pelo ator que, neste caso, é o usuário do aplicativo. Entre estas operações estão o cadastro e a autenticação de usuário, além de visualização e avaliação de restaurantes, as quais serão detalhadas na Seção 3.7.

3.5.1.1 Casos de Uso

Nome do Caso de Uso: UC001 – Cadastrar Usuário

Descrição:

Permite que o cadastro de um usuário para o acesso ao sistema.

Eventos:

- Usuário abre o aplicativo no dispositivo móvel.

Atores:

- Usuário

Pré-Condições:

- O usuário deve ter o aplicativo instalado em seu dispositivo.
- O usuário deve estar conectado à Internet.

Pós-Condições:

1. Conclusões com sucesso:
- O usuário é cadastrado com sucesso no sistema.
2. Conclusões sem sucesso:
- O aplicativo verifica que os dados de cadastro estão incorretos e informa o usuário.

Fluxo básico:

1. O usuário abre o aplicativo;
2. O usuário pressiona o botão "Cadastre-se";
3. O aplicativo apresenta da tela de cadastro;
4. O usuário preenche os dados de cadastro (E1);
5. O aplicativo verifica se os dados estão corretos ;
6. O aplicativo cadastra o usuário e apresenta a tela de login;
7. Fim do caso de uso.

Fluxo de exceção:

E1 - O aplicativo verifica que os dados não estão corretos e apresenta uma mensagem de erro.

Nome do Caso de Uso: UC002 – Realizar Login

Descrição:

Permite que o usuário realize login no aplicativo, possibilitando o acesso ao sistema.

Eventos:

- Usuário abre o aplicativo no dispositivo móvel.

Atores:

- Usuário
- Facebook

Pré-Condições:

- O usuário deve ter o aplicativo instalado em seu dispositivo.
- O usuário deve estar habilitado a utilizar o sistema através de informações de login.
- O usuário deve estar conectado à Internet.

Pós-Condições:

1. Conclusões com sucesso:
- O usuário realiza com sucesso o login no aplicativo.
2. Conclusões sem sucesso:
- O aplicativo verifica que os dados de login estão incorretos e informa o usuário.

Fluxo básico:

1. O usuário abre o aplicativo;
2. O aplicativo solicita nome de usuário e senha (A1);
3. O aplicativo verifica se os dados estão corretos (E1);
4. O aplicativo realiza login e apresenta a tela principal da aplicação;
5. Fim do caso de uso.

Fluxo alternativo:

A1:

2.1. O usuário pressiona no botão "Login com Facebook";
2.2. O aplicativo Facebook valida os dados de login do usuário (E2);
2.3. O aplicativo realiza login e apresenta a tela principal da aplicação;
2.4. Fim do caso de uso.

Fluxos de exceção:

E1 - O aplicativo verifica que os dados não estão corretos e apresenta uma mensagem de erro.

E2 - O Facebook verifica que os dados não estão corretos e apresenta uma mensagem de erro.

Nome do Caso de Uso: UC003 – Visualizar Restaurantes

Descrição:

Permite que o usuário visualize todos os restaurantes cadastrados no sistema.

Eventos:

- Usuário abre o aplicativo no dispositivo móvel.

Atores:

- Usuário

Pré-Condições:

- O usuário deve ter o aplicativo instalado em seu dispositivo.
- O usuário deve estar habilitado a utilizar o sistema através de informações de login.
- O usuário deve estar conectado à Internet.
- O usuário deve estar autenticado no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- O usuário visualiza todos os restaurantes cadastrados no sistema.
2. Conclusões sem sucesso:
- O aplicativo verifica que os dados de login estão incorretos e informa o usuário.

Fluxo básico:

1. O usuário abre o aplicativo;
2. O aplicativo apresenta o mapa marcado com os restaurantes;
3. Fim do caso de uso (A1).

Fluxo alternativo:

A1:

3.1. O usuário abre o menu principal do aplicativo e seleciona a opção "Restaurantes";

3.2. O aplicativo apresenta a tela de listagem de restaurante;

3.3. Fim do caso de uso.

Nome do Caso de Uso: UC004 – Manter Restaurante

Descrição:

Permite que o usuário gerencie restaurantes no sistema.

Eventos:

- Usuário abre o aplicativo no dispositivo móvel.

Atores:

- Usuário

Pré-Condições:

- O usuário deve ter o aplicativo instalado em seu dispositivo.
- O usuário deve estar habilitado a utilizar o sistema através de informações de login.
- O usuário deve estar conectado à Internet.
- O usuário deve estar autenticado no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- O usuário gerencia restaurantes cadastrados no sistema.

2. Conclusões sem sucesso:

- O aplicativo não recupera ou salva dados do restaurante e informa o usuário.

Fluxo básico:

1. O usuário seleciona um restaurante a partir da listagem ou do mapa (A1);
2. O aplicativo apresenta a tela de detalhes do restaurante selecionado;
3. Fim do caso de uso (A2).

Fluxo alternativo:

A1:

1.1. O usuário pressiona o botão de adicionar restaurante na tela de listagem;
1.2. O aplicativo apresenta a tela de cadastro de um restaurante;
1.3. O usuário preenche os dados do restaurante e pressiona o botão "Salvar";
1.4. O aplicativo valida os dados preenchidos e salva o restaurante no sistema (E1);
1.5. Fim do caso de uso.

A2:

3.1. O usuário pressiona o botão de editar restaurante na tela de detalhes;
3.2. O aplicativo apresenta a tela de edição de um restaurante;
3.3. O usuário preenche os dados do restaurante e pressiona o botão "Salvar";
3.4. O aplicativo valida os dados preenchidos e salva o restaurante no sistema (E1);
3.5. Fim do caso de uso.

Fluxos de exceção:

E1 - O aplicativo verifica que os dados não estão corretos e apresenta uma mensagem de erro.

Nome do Caso de Uso: UC005 – Avaliar Restaurante

Descrição:

Permite que o usuário avalie restaurantes no sistema.

Eventos:

- Usuário abre o aplicativo no dispositivo móvel.

Atores:

- Usuário

Pré-Condições:

- O usuário deve ter o aplicativo instalado em seu dispositivo.
- O usuário deve estar habilitado a utilizar o sistema através de informações de login.
- O usuário deve estar conectado à Internet.
- O usuário deve estar autenticado no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- O usuário avalia restaurantes cadastrados no sistema.
2. Conclusões sem sucesso:
- O aplicativo não salva avaliação do restaurante e informa o usuário.

Fluxo básico:

1. O usuário pressiona o botão de avaliação de restaurantes na tela de detalhes;
2. O aplicativo apresenta a tela de avaliação do restaurante selecionado;
3. O usuário preenche os dados de avaliação do restaurante;
4. O aplicativo valida os dados preenchidos e os salva no sistema (E1);
5. Fim do caso de uso.

Fluxos de exceção:

E1 - O aplicativo verifica que os dados não estão corretos e apresenta uma mensagem de erro.

Nome do Caso de Uso: UC006 – Compartilhar Restaurante

Descrição:

Permite que o usuário compartilhe restaurantes no Facebook.

Eventos:

- Usuário abre o aplicativo no dispositivo móvel.

Atores:

- Usuário

Pré-Condições:

- O usuário deve ter o aplicativo instalado em seu dispositivo.
- O usuário deve estar habilitado a utilizar o sistema através de informações de login.
- O usuário deve estar conectado à Internet.
- O usuário deve estar autenticado no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- O usuário compartilha no Facebook um restaurante cadastrado no sistema.
2. Conclusões sem sucesso:
- O aplicativo Facebook não compartilha os dados e informa o usuário.

Fluxo básico:

1. O usuário pressiona o botão de compartilhar restaurante na tela de detalhes;
2. O aplicativo Facebook apresenta a tela de compartilhar o restaurante selecionado;
3. O usuário pressiona o botão "Publicar";
4. O aplicativo Facebook compartilha o restaurante no perfil do usuário autenticado (E1);
5. Fim do caso de uso.

Fluxos de exceção:

E1 - O aplicativo Facebook não compartilha as informações e apresenta uma mensagem de erro.

3.5.2 Diagrama de Atividades

Conforme exposto na Seção 2.3.1.2, o diagrama de atividades apresenta o fluxo de atividades de um sistema se atentando para a sequência e condições das mesmas.

A Figura 23 apresenta um dos diagramas de atividades do aplicativo *SPeek*, que é o diagrama de atividades do fluxo de autenticação e cadastro do usuário. Este diagrama detalha a sequência de atividades dos casos de uso “Cadastrar Usuário” e “Realizar Login”.

O fluxo se inicia quando o usuário abre o aplicativo, o qual recupera as informações do usuário. Caso o usuário esteja autenticado, o sistema exibe a tela principal e o fluxo se encerra. Caso contrário, o sistema exibe a tela de autenticação do usuário. Se o usuário já estiver cadastrado, ele pode se autenticar com sua conta

do Facebook ou com seu e-mail e senha para, então, o sistema apresentar a tela principal e o fluxo se encerrar.

Porém, se o usuário ainda não estiver cadastrado, ele poderá se cadastrar também de duas maneiras, utilizando um e-mail e senha ou sua conta no Facebook. Desta segunda maneira, o usuário deve conceder permissão de acesso do aplicativo à sua conta no Facebook. Após o cadastro, o sistema apresenta a tela principal e o fluxo se encerrar.

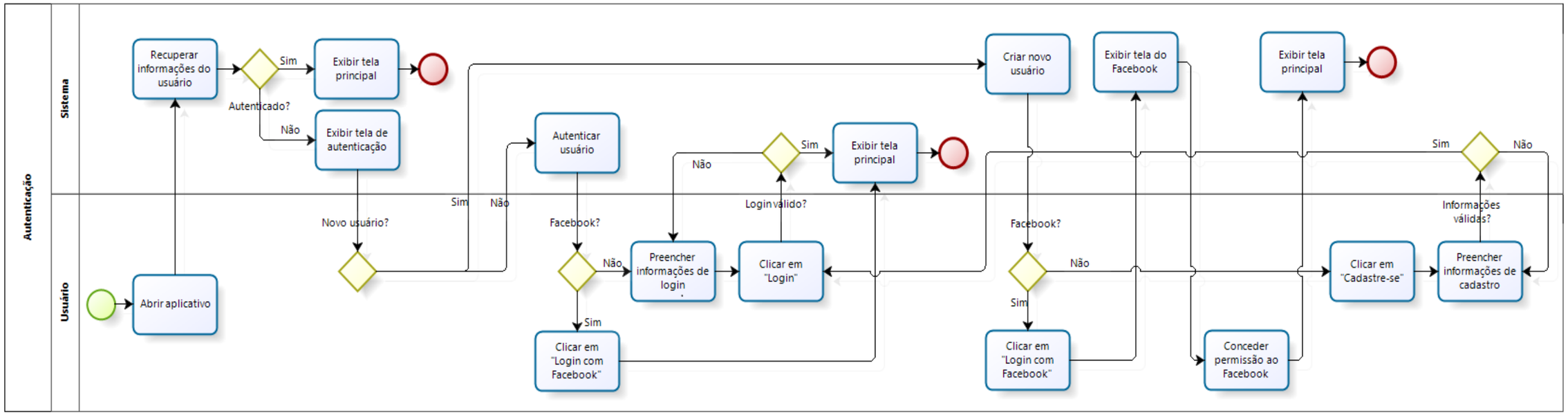
A Figura 24 apresenta o diagrama de atividades que detalha a sequência de atividades dos casos de uso “Visualizar Restaurantes no Mapa”, “Visualizar Restaurantes na Lista”, “Visualizar Restaurante” e “Compartilhar Restaurante”.

Neste diagrama, o fluxo se inicia quando o sistema apresenta a tela principal do aplicativo, em seguida busca as informações dos restaurantes cadastrados e os exibe no mapa do aplicativo. Caso o usuário escolha visualizá-los em forma de lista, ele irá clicar na opção “Restaurantes” do menu principal e o sistema apresentará a tela de listagem de restaurantes.

Nestes dois cenários, o usuário poderá decidir se deseja visualizar mais informações sobre algum restaurante. No caso de visualização de restaurantes no mapa, ele irá clicar em um marcador no mapa que representa um restaurante. Já no caso de visualização em forma de listagem, ele irá clicar em um item da lista. O sistema, então, irá exibir a tela de detalhes de um determinado restaurante.

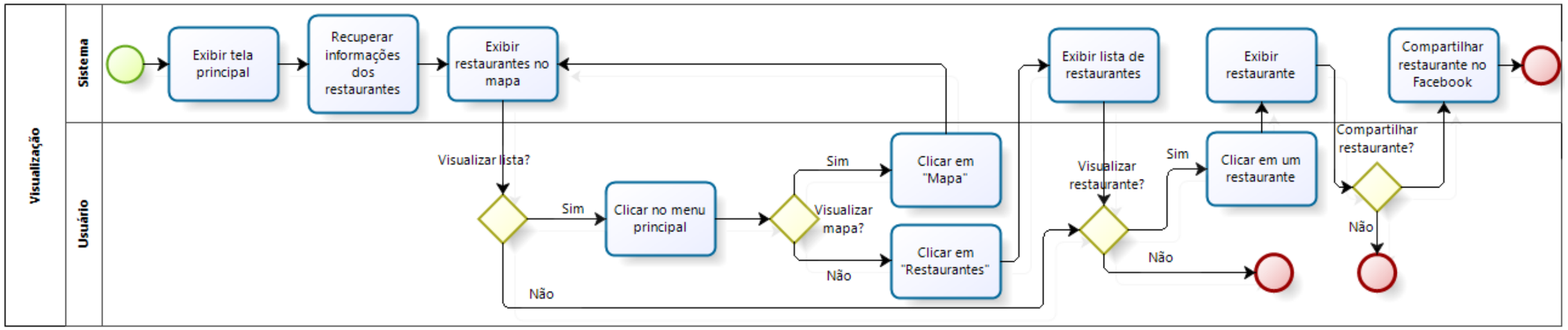
Se o usuário não desejar ver mais informações, o fluxo se encerra. Caso contrário, ele visualizará as informações do restaurante e poderá decidir se irá ou não compartilhar estas informações sobre o restaurante no Facebook e, assim, o fluxo se encerra.

Figura 23 – Diagrama de Atividades Autenticação



Fonte: Elaborada pela autora

Figura 24 – Diagrama de Atividades Visualização



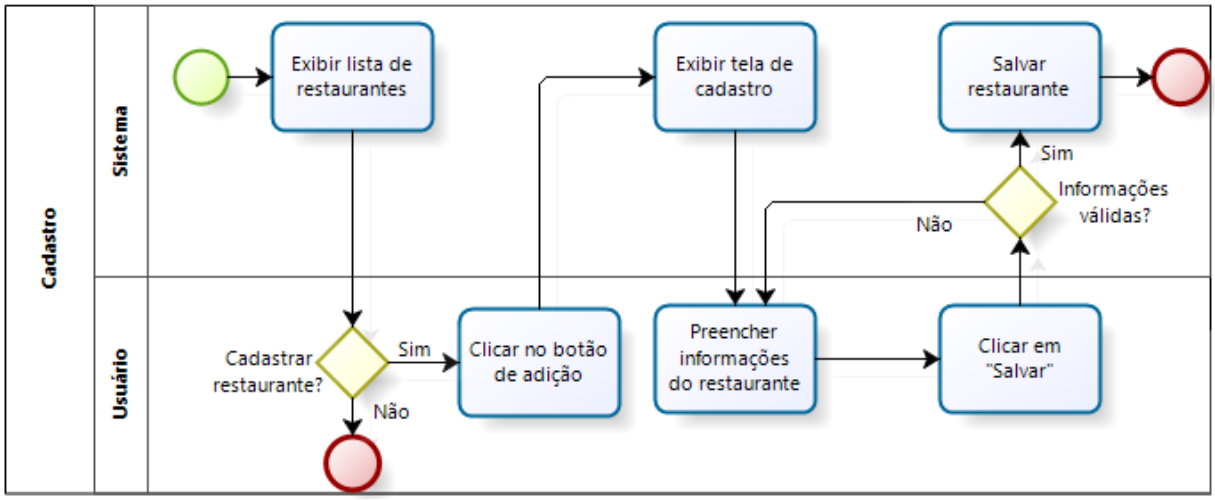
Fonte: Elaborada pela autora

A Figura 25 apresenta o diagrama de atividades que descreve as atividades do caso de uso “Cadastrar Restaurante”, no qual o fluxo se inicia quando o sistema exibe a tela de listagem de restaurantes e a partir disso o usuário pode escolher se irá cadastrar um novo restaurante.

Caso ele escolha cadastrar um novo restaurante, o sistema apresenta a tela de cadastro para que o usuário possa preencher as informações de cadastro e, então, clicar em “Salvar”. Se as informações forem válidas, o sistema armazena o novo restaurante na base de dados e o fluxo se encerra.

Caso contrário, uma mensagem de erro é apresentada para que o usuário corrija o dado errado e clique novamente em “Salvar”. Se as informações estiverem válidas, o sistema cadastra o restaurante e o fluxo se encerra.

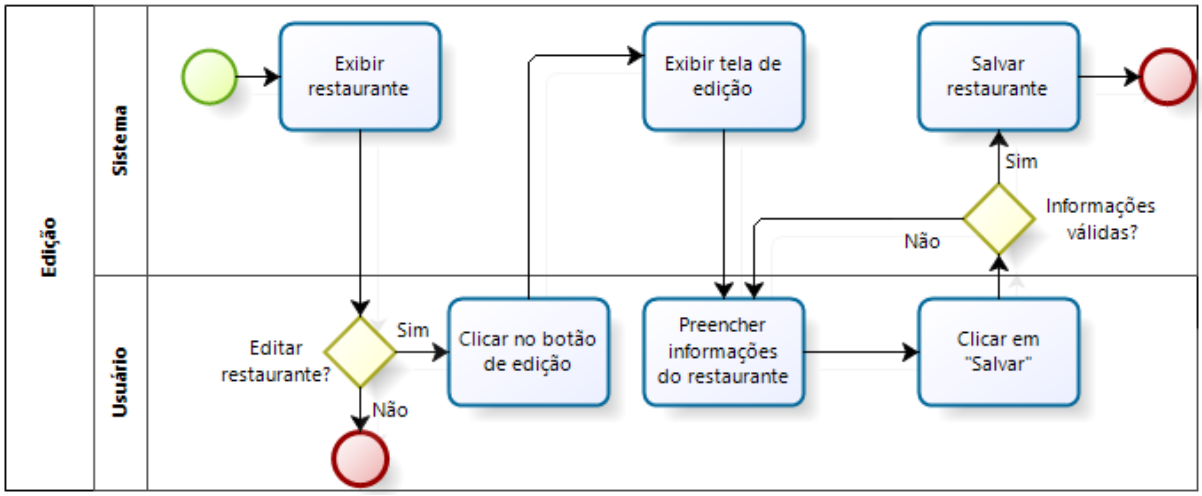
Figura 25 – Diagrama de Atividades Cadastro



Fonte: Elaborada pela autora

A Figura 26 exibe o diagrama de atividades que detalha as atividades do caso de uso “Editar Restaurante”. O fluxo de edição é semelhante ao fluxo de cadastro, porém este se inicia a partir da tela de visualização de detalhes do restaurante e o sistema não armazena um novo restaurante na base de dados, mas sim atualiza um restaurante já existente.

Figura 26 – Diagrama de Atividades Edição

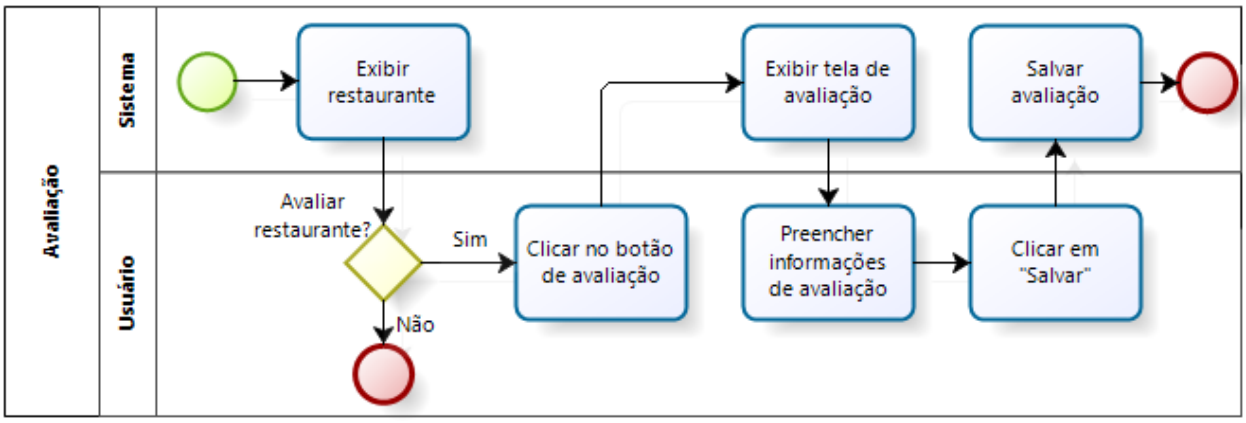


Fonte: Elaborada pela autora

A Figura 27 mostra o diagrama de atividades que descreve o fluxo de atividades do caso de uso “Avaliar Restaurante”. O fluxo tem seu início quando o sistema apresenta a tela de visualização de detalhes do restaurante. A partir disso o usuário pode escolher se ele irá avaliar um determinado restaurante ou não.

Em caso afirmativo, o usuário clicará no botão de avaliação, o sistema exibirá a tela de avaliação e, então, o usuário poderá preencher as informações sobre sua avaliação deste restaurante. Após preencher as informações, o usuário irá clicar no botão “Salvar” para que o sistema armazene a avaliação do restaurante e, assim, o fluxo se encerra.

Figura 27 – Diagrama de Atividades Avaliação



Fonte: Elaborada pela autora

3.5.3 Diagrama de Classes

Conforme descrito na Seção 2.3.2.1, o diagrama de classes fornece uma visão geral de como a aplicação está estruturada, ele simplifica a manutenção do sistema, pois com ele podemos analisar a arquitetura antes de codificar uma funcionalidade podendo reduzir, assim, o tempo de manutenção.

A Figura 28 mostra o diagrama de classes do *server-side* do aplicativo *SPeek*, já a Figura 29 apresenta o diagrama de classes resumido do *client-side* do aplicativo. O diagrama de classes completo encontra-se no Apêndice A. Nesse diagrama, a classe de entrada é a classe *SplashActivity*, a qual utiliza a classe *LoginService* que é responsável por validar se já existe ou não um usuário autenticado para utilizar aplicativo. A classe *SplashActivity* é uma especialização da classe *BaseActivity*, a qual contém métodos comuns a todas as classes de interface com usuário.

A classe *LoginActivity* constrói a tela de autenticação do usuário e também utiliza a classe *LoginService* que tenta autenticar o usuário através da classe *LoginRemoteDao.* Esta classe monta uma requisição para o *web service*, a qual é tratada na classe *UserRest* do *server-side* que utiliza a classe *UserService* do *server-side*, que faz uma chamada para a classe *UserDao* responsável por tentar recuperar um usuário do banco de dados.

Além disso, a classe *UserService* do *cliente-side* faz uma chamada para a classe *UserRemoteDao* para armazenar as informações do usuário no banco de dados local. Estas classes são utilizadas também para cadastrar um novo usuário. Esse conjunto de classes é responsável por contemplar o caso de uso “Realizar Login” e “Cadastrar Usuário”.

As classes *RestaurantRest*, *RestaurantService* e *RestaurantDao* do *server-side* são responsáveis por recuperar informações dos restaurantes do *web service*. Já no *cliente-side*, a classe *RestaurantRemoteDao* faz uma requisição para este *web service* e retorna estas informações para a classe *RestaurantService*.

Esta classe é utilizada na classe *LoginActivity* para recuperar informações do restaurante no momento da autenticação, na classe *MapFragment* para mostrar os restaurantes no mapa, na classe *RestaurantListActivity* para listar os restaurantes, na classe *RestaurantViewActivity* para exibir detalhes de um

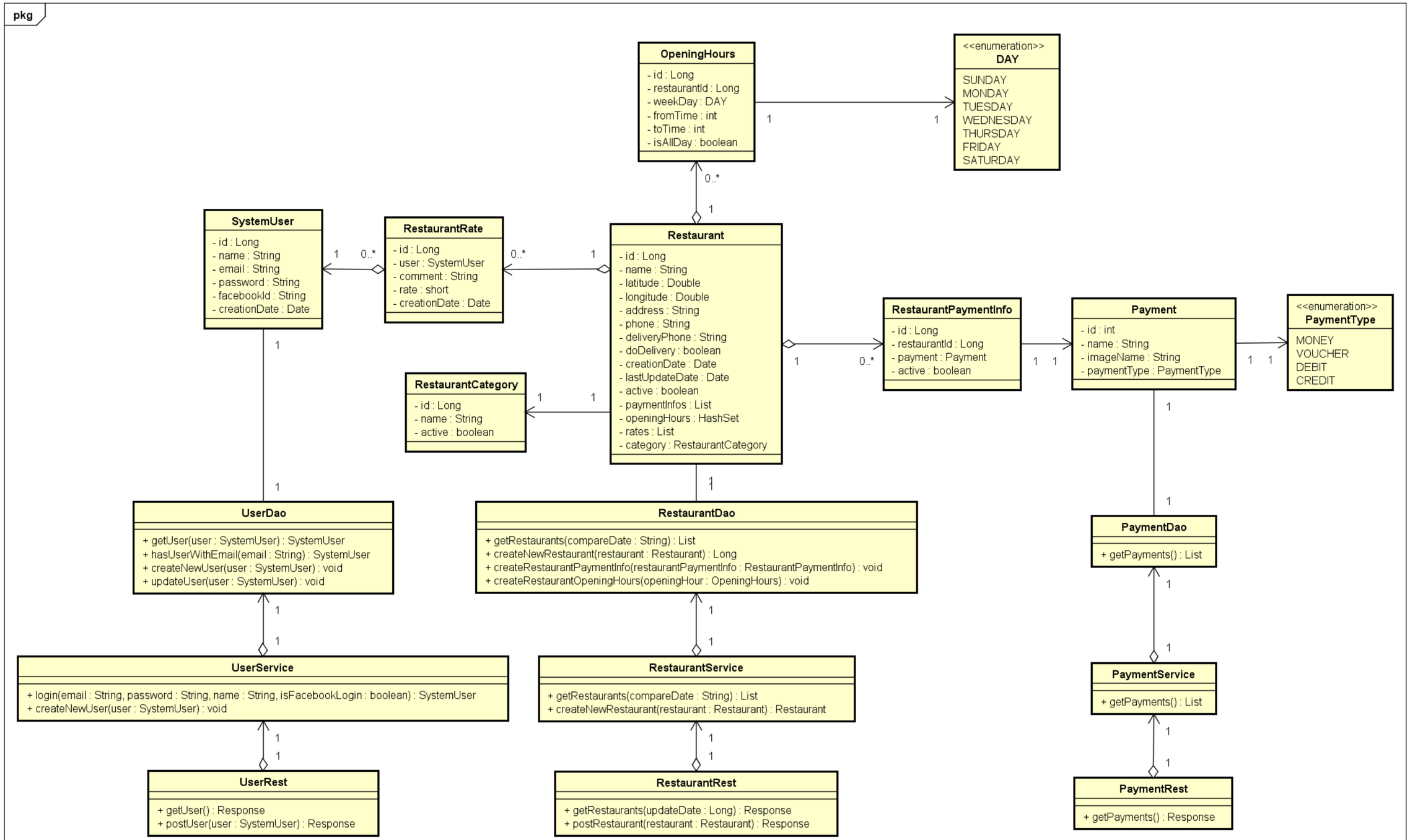
restaurante, na classe *RestaurantRateActivity* para cadastrar uma avaliação e na classe *RestaurantFormActivity* para cadastrar e editar um restaurante.

Esse conjunto de classes é responsável por contemplar os casos de uso “Visualizar Restaurantes no Mapa”, “Visualizar Restaurantes na Lista”, “Visualizar Restaurante”, “Avaliar Restaurante”, “Cadastrar Restaurante” e “Editar Restaurante”.

Por fim, as classes *PaymentRest*, *PaymentService* e *PaymentDao* do *server-side* são utilizadas pelo *web service* responsável por recuperar informações de formas de pagamento de restaurantes. Este *web service* é consumido no *cliente-side* através das classe *PaymentRemoteDao* que cria a requisição para buscar essas informações e as retorna para a classe *PaymentService*.

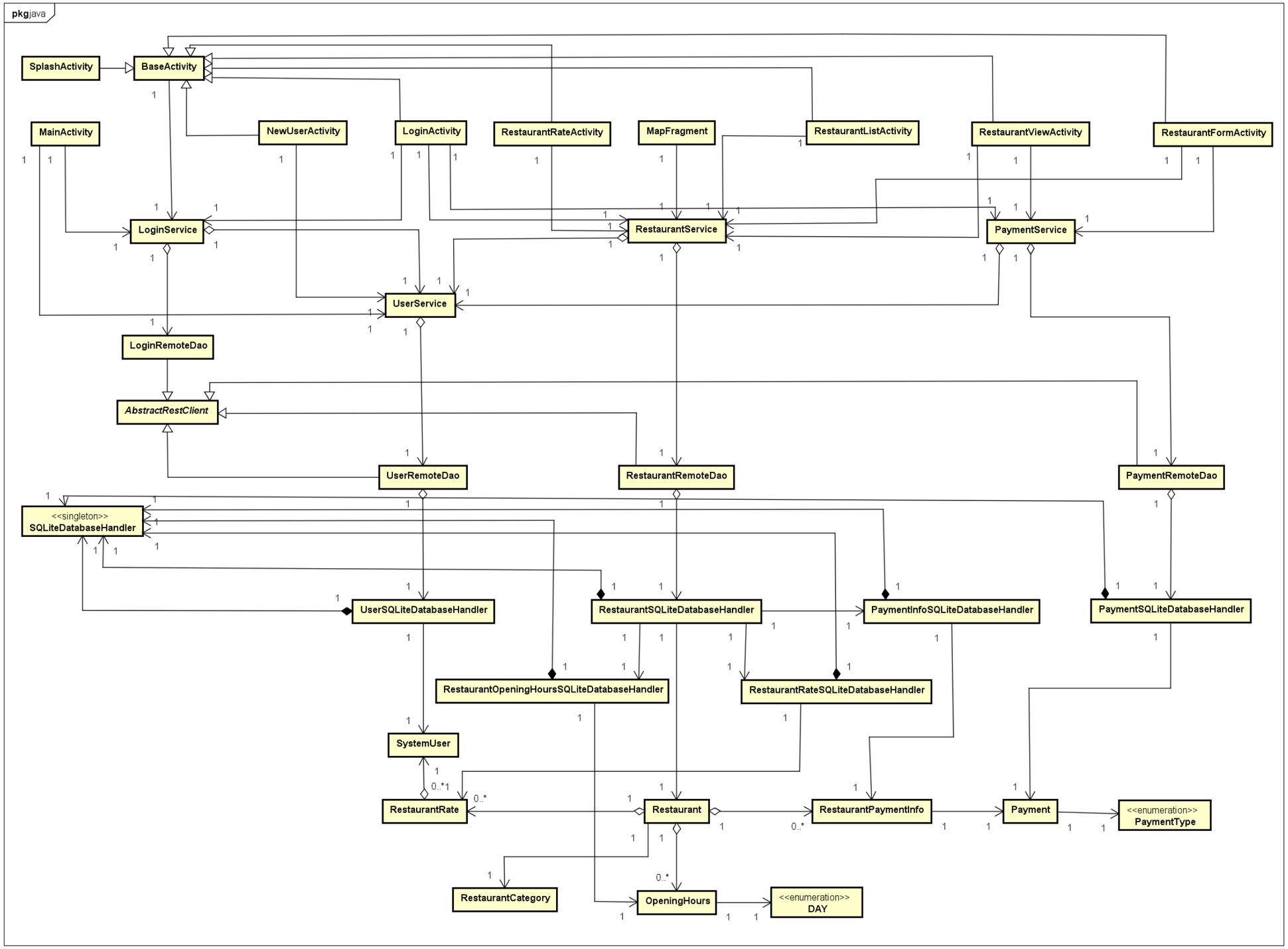
Esta classe é utilizada na classe *LoginActivity* para recuperar informações de formas de pagamento dos restaurantes no momento da autenticação, na classe *RestaurantViewActivity* para exibir detalhes das formas de pagamento de um restaurante e na classe *RestaurantFormActivity* para cadastrar um novo restaurante.

Figura 28 – Diagrama de Classes *Server-Side*



Fonte: Elaborada pela autora

Figura 29 – Diagrama de Classes *Client-Side*



Fonte: Elaborada pela autora

3.6 LAYOUT

Esta seção apresenta o desenvolvimento da interface do aplicativo *SPeek* iniciando pelos protótipos de baixa fidelidade e evoluindo até a versão final do *layout* do aplicativo.

3.6.1 Baixa Fidelidade

Conforme descrito na Seção 2.2.2, foi criado o protótipo de baixa fidelidade para validar tanto a navegabilidade do aplicativo quanto a disposição dos componentes na tela.

A Figura 30 apresenta a tela de autenticação do aplicativo *SPeek* do protótipo de baixa fidelidade, ou seja, esta tela foi criada de maneira rápida utilizando apenas papel e lápis.

Figura 30 – Protótipo de Baixa Fidelidade da Tela de Autenticação

Fonte: Elaborada pela autora

O protótipo de baixa fidelidade foi apresentado para algumas pessoas como esboço das ideias iniciais para o aplicativo para determinar de pontos

negativos e positivos destas ideias e a partir disso foi possível evoluir este protótipo para as demais versões.

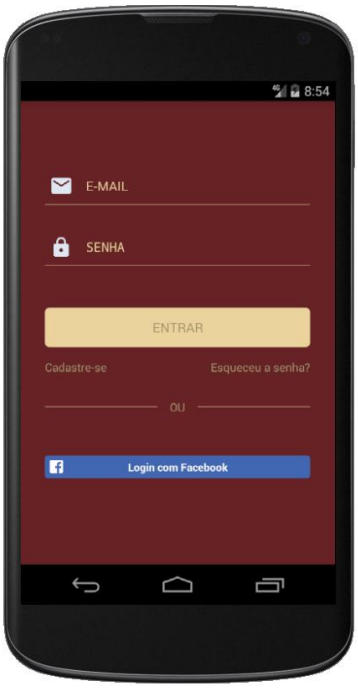
3.6.2 Alta Fidelidade

Conforme detalhado na Seção 2.2.3, a partir da conclusão do protótipo de baixa fidelidade podemos levantar pontos de melhorias para evoluir a prototipação criando, assim, o protótipo de alta fidelidade.

Após a apresentação do protótipo de baixa fidelidade, analisando a Figura 30 foi levantado que existiam muitos botões na tela, o que a polui visualmente e compromete a navegação da principal funcionalidade desta tela, que é a autenticação do usuário.

Assim, uma nova versão foi proposta, conforme Figura 31, que deixou a tela mais limpa visualmente focando na sua principal funcionalidade sem inutilizar os demais fluxos.

Figura 31 – Primeira Versão do Protótipo de Alta Fidelidade da Tela de Autenticação



Fonte: Elaborada pela autora

Nessa primeira versão, contudo, cores não foram ponderadas. Esse trabalho foi realizado somente na segunda versão do protótipo de alta fidelidade com o auxílio de uma pessoa que trabalha com *design* atingindo, assim, a versão final do protótipo do aplicativo *SPeek*, conforme apresentado na Figura 32.

Figura 32 – Versão Final do Protótipo de Alta Fidelidade da Tela de Autenticação

Fonte: Elaborada pela autora

3.7 FUNCIONALIDADES

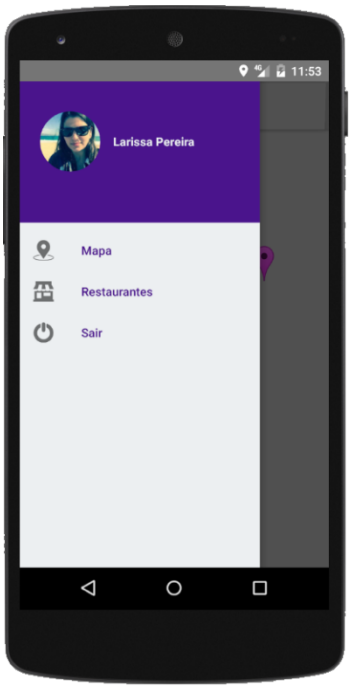
De maneira geral, as funcionalidades da primeira versão do aplicativo *SPeek* são autenticação, visualizar restaurantes, cadastrar restaurantes e avaliar restaurantes, além de se integrar com o Facebook, conforme apresentado a seguir.

3.7.1 Autenticação

Os usuários do aplicativo *SPeek* terão uma conta para acessar os dados do aplicativo. Eles poderão se cadastrar de duas maneiras, a primeira é com um e-mail e senha da escolha do usuário e a segunda é com seu perfil do Facebook, conforme apresentado na Figura 32.

Uma vez autenticado, o usuário não precisará fazer esse processo novamente desde que ele não utilize a funcionalidade de sair do aplicativo, pois os dados de autenticação do usuário são armazenados no banco de dados local do dispositivo. Os dados do usuário autenticados são exibidos no menu principal do aplicativo, conforme Figura 33.

Figura 33 – Tela do Menu Principal

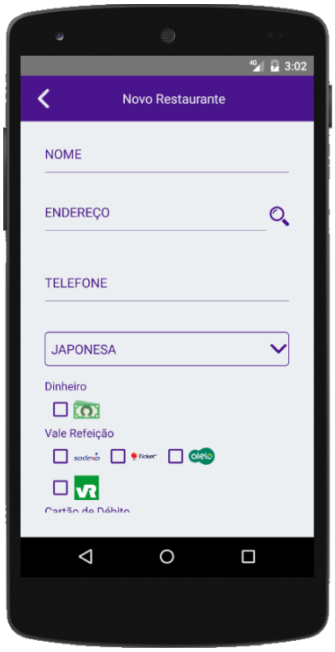


Fonte: Elaborada pela autora

3.7.2 Cadastrar Restaurantes

Os usuários do aplicativo *SPeek* poderão cadastrar novos restaurantes informando inicialmente e obrigatoriamente somente o nome do estabelecimento e sua localização, mas pode também inserir, de forma opcional, informações de formas de pagamento, telefone e a categoria na qual o restaurante se enquadra, conforme Figura 34.

Figura 34 – Tela de Cadastro de Restaurante



Fonte: Elaborada pela autora

Uma vez que o restaurante esteja na base de dados do aplicativo, qualquer usuário poderá editar suas informações nesta primeira versão, conforme Figura 35.

Figura 35 – Tela de Edição de Restaurante

Fonte: Elaborada pela autora

3.7.3 Visualizar Restaurantes

Os restaurantes cadastrados no aplicativo *SPeek* são apresentados de duas maneiras, a primeira é em forma de lista, divididos entre restaurantes abertos e fechados no momento, conforme Figura 36, exibindo, na própria lista, o nome do restaurante, seu endereço, a sua distância em relação a localização do usuário, bem como sua avaliação média.

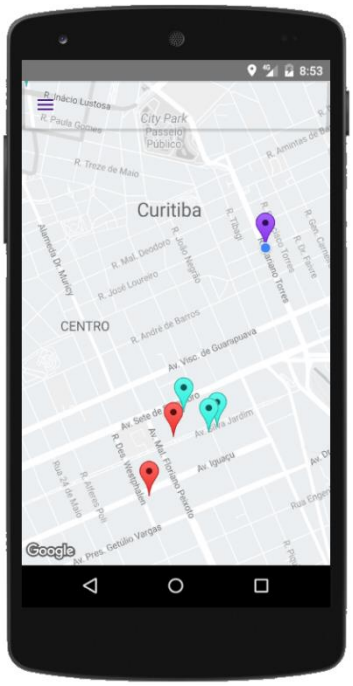
Figura 36 – Tela de Listagem de Restaurantes

Fonte: Elaborada pela autora

E a segunda maneira de apresentação é em forma de mapa, na qual os restaurantes são marcados no mapa de acordo com sua localização e horário de funcionamento, conforme Figura 37.

Ao selecionar um restaurante, a partir da tela de listagem ou da tela de mapa, uma outra tela é exibida, conforme Figura 38, para detalhar as informações do restaurante, como sua categoria, suas avaliações e suas formas de pagamento.

Figura 37 – Tela do Mapa de Restaurantes



Fonte: Elaborada pela autora

Figura 38 – Tela de Visualização de Detalhes

Fonte: Elaborada pela autora

Além disso, a partir da integração do aplicativo *SPeek* com o Facebook os usuários podem compartilhar o restaurante no Facebook através de seu perfil.

3.7.4 Avaliar Restaurantes

Os usuários do aplicativo *SPeek* serão capazes também de avaliar os restaurantes cadastrados na base de dados do aplicativo, classificando-os por meio de uma nota de um a cinco para cada um dos itens, que são o preço dos itens vendidos no restaurante, o nível do serviço prestado e a qualidade da culinária, conforme ilustra a Figura 39. Além disso, os usuários do aplicativo *SPeek* poderão fazer comentários em suas avaliações.

Figura 39 – Tela de Avaliação

Fonte: Elaborada pela autora

3.8 AVALIAÇÃO

Após a implementação da primeira versão do aplicativo, este foi disponibilizado para um grupo de pessoas afim de avaliar sua usabilidade e interface. Alguns pontos de melhoria foram levantados e serão tratados nas versões futuras antes da publicação do aplicativo na Play Store para que a primeira versão oficial já contemple detalhes de experiência do usuário.

4 CONCLUSÕES

Este trabalho contribuiu para o desenvolvimento profissional da autora, uma vez que proporcionou a oportunidade de aprofundar o aprendizado em desenvolvimento mobile e mais especificamente no sistema operacional Android. Isso proporcionou atingir o objetivo geral deste trabalho que era desenvolver um aplicativo eficiente, rápido e focado em usabilidade que proporcionaria aos seus usuários informações mais pontuais sobre restaurantes disponíveis na região.

Além disso, foi possível também atingir os objetivos específicos, já que foi criada uma interface intuitiva com auxílio do trabalho de um *designer* e as integrações com Google Maps e com o Facebook foram desenvolvidas. Além disso, foi possível avaliar a primeira versão do aplicativo com respeito à usabilidade.

As disciplinas do curso foram de suma importância para fornecer conhecimento para a conclusão deste trabalho. Como, por exemplo, a disciplina Métodos Ágeis que contribuiu para a escolha da metodologia de trabalho em prol de organizar as tarefas que deveriam ser realizadas.

A disciplina Design de Interação contribuiu com a mentalidade voltada para a criação de uma melhor experiência para o usuário e, assim, desenvolver um aplicativo com uma interface amigável ao usuário final.

Além disso, a disciplina UML ajudou a relembrar os diagramas da UML, bem como norteou a criação dos principais diagramas para esclarecer as funcionalidades do aplicativo e seus fluxos de atividades, bem como a modelagem do sistema como um todo.

A disciplina Linguagem de Programação Java colaborou para relembrar conceitos da mesma, que foi de extrema importância para o desenvolvimento deste trabalho, uma vez que esta linguagem é a base para o desenvolvimento Android nativo e foi a linguagem escolhida para a criação dos *Web Services*.

E, por fim, as disciplinas relacionadas ao desenvolvimento Android contribuíram com o resultado final obtido neste trabalho, visto que este foi desenvolvido para o sistema operacional Android.

4.1 TRABALHOS FUTUROS

A primeira versão do aplicativo *SPeek* é um protótipo que contém apenas funcionalidades básicas, porém futuras implementações irão aprimorar o aplicativo de forma iterativa.

Dentre estas implementações estão:

- Criação de ícones intuitivos para o mapa para mostrar informações de horário de funcionamento do restaurante;
- Permissão de adição de fotos do restaurante e de seus pratos;
- Busca por nome do restaurante;
- Atualização da lista de restaurantes puxando-a para baixo;
- Realizar *check in* no Facebook na localização do restaurante;
- Cadastro de pratos;
- Sistema de pontuação do usuário que permita que este tenha acessos de acordo com sua pontuação, como por exemplo editar restaurantes;
- Publicação na Play Store;
- Versão para iOS;
- Versão off-line.

5 REFERÊNCIAS

Abrasel. (2017). *Perfil da Abrasel*. Acesso em 04 de 10 de 2017, disponível em Abrasel: http://www.abrasel.com.br/a-abrasel/perfil-da-abrasel.html

Apple. (2017). Acesso em 30 de 09 de 2017, disponível em Apple: https://www.apple.com

BABBIE, E. (2005). Métodos de Pesquisas em Survey. Belo Horizonte, MG: UFMG.

BELL, D. (2016). *O diagrama de classes*. Acesso em 28 de 08 de 2017, disponível em IBM - developerWorks: https://www.ibm.com/developerworks/br/rational/library/content/RationalEdge/sep04/bell/index.html

BERNADO, K. (2014). *Kanban: Do início ao fim!* Acesso em 09 de 09 de 2017, disponível em Cultura Ágil: https://www.culturaagil.com.br/kanban-do-inicio-ao-fim/

BERNARDO, K. (2015). *O que são métodos ágeis?* Acesso em 01 de 09 de 2017, disponível em Cultura Ágil: https://www.culturaagil.com.br/o-que-sao-metodos-ageis/

Coding Dojo. (2017). Acesso em 30 de 09 de 2017, disponível em Coding Dojo: http://www.codingdojo.com

COHN, M. (06 de 04 de 2007). *Differences Between Scrum and Extreme Programming*. Acesso em 29 de 10 de 2017, disponível em Mountain Goat Software: https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming

Desenvolvimento Ágil. (2014). *Scrum*. Acesso em 30 de 11 de 2017, disponível em Desenvolvimento Ágil: http://www.desenvolvimentoagil.com.br/scrum/

FAKHROUTDINOV, K. (2009). Acesso em 31 de 08 de 2017, disponível em uml-diagrams: http://www.uml-diagrams.org/uml-24-diagrams.html

FIELDING, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures.

FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., et al. (1999). *Hypertext Transfer Protocol -- HTTP/1.1*. Acesso em 09 de 08 de 2017, disponível em RFC 2616: http://tools.ietf.org/html/rfc2616

Fortune 500. (2017). Acesso em 30 de 09 de 2017, disponível em Fortune 500: http://fortune.com/fortune500/

GitHub. (2017). Acesso em 30 de 09 de 2017, disponível em GitHub: http://github.com/

GLOVER, A. (2008). *Construa um Serviço da Web RESTful*. Acesso em 09 de 08 de 2017, disponível em IBM - developerWorks: https://www.ibm.com/developerworks/br/library/j-rest/index.html

GOMES, C. (2017). *Scrum: A Metodologia Ágil Simplificada*. Acesso em 16 de 12 de 2017, disponível em Europneumaq: http://blog.europneumaq.com/scrum-metodologia-agil-simplificada

Google. (2017). Acesso em 30 de 09 de 2017, disponível em Google: http://www.google.com

Google. (2017). Acesso em 01 de 10 de 2017, disponível em Google Play: https://play.google.com/store

GOOGLE. (2017). *Android*. Acesso em 09 de 09 de 2017, disponível em Developers: https://developer.android.com/about/index.html

GOOGLE. (2017). *Dashboards*. Acesso em 04 de 11 de 2017, disponível em Developers: https://developer.android.com/about/dashboards/index.html

HELPPI, V. (2014). *What Every App Developer Should Know About Android*. Acesso em 16 de 12 de 2017, disponível em Smashing Magazine: https://www.smashingmagazine.com/2014/10/what-every-app-developer-should-know-about-android/

IBGE. (2010). *Pesquisa de Orçamentos Familiares 2008-2009.* Acesso em 04 de 10 de 2017, disponível em IBGE: https://biblioteca.ibge.gov.br/visualizacao/livros/liv45130.pdf

IBGE. (2012). *Censo Demográfico - 2010.* Rio de Janeiro.

Indeed. (2017). Acesso em 30 de 09 de 2017, disponível em Indeed: Indeed.com

International Data Corporation. (2017). *Smartphone OS Market Share, 2017 Q1.* Acesso em 20 de 08 de 2017, disponível em IDC: http://www.idc.com/promo/smartphone-market-share/os

IxDA. (2017). *About & History*. Acesso em 24 de 09 de 2017, disponível em IxDA: http://ixda.org/ixda-global/about-history/

JOB, J. (07 de 12 de 2015). *Scrum Diagram*. Acesso em 29 de 10 de 2017, disponível em Jordan Job: https://jordanjob.me/2015/12/07/scrum-diagram/

KIM, J. W. (2017). *PRISM*. Acesso em 25 de 09 de 2017, disponível em JAE WON KIM: http://www.jae-wonkim.com/prism/

KNIBERG, H., & SKARIN, M. (2010). *Kanban e Scrum obtendo o melhor de ambos.* InfoQueue.

leankit. (2016). *What is Kanban?* Acesso em 01 de 12 de 2017, disponível em leankit: https://leankit.com/learn/kanban/what-is-kanban/

LOTZ, M. (2013). *Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?* Acesso em 16 de 12 de 2017, disponível em Segue Technologies: https://www.seguetech.com/waterfall-vs-agile-methodology/

MACEDO, J. (2014). *Desenvolvimento de Aplicativos para a Plataforma Android*. Acesso em 16 de 12 de 2017, disponível em Slide Share: https://pt.slideshare.net/joseamacedo/desenvolvimento-de-aplicativos-para-a-plataforma-android

MALCHER, J. (2016). *Resumo – Diagramas UML*. Acesso em 16 de 12 de 2017, disponível em José Malcher Jr.: http://josemalcher.net/resumo-diagramas-uml/

NOGUEIRA, A. (2005). *UML - Unified Modeling Language - Atores, Atividades e Componentes*. Acesso em 31 de 08 de 2017, disponível em Linha de Código: http://www.linhadecodigo.com.br/artigo/853/uml-unified-modeling-language-atores-atividades-e-componentes.aspx

OLIVEIRA, L. (08 de 10 de 2016). *KANBAN: UMA BREVE INTRODUÇÃO*. Acesso em 29 de 10 de 2017, disponível em Diferencial TI: http://blog.diferencialti.com.br/uma-breve-introducao-ao-kanban/

Oracle. (2017). *The History of Java Technology*. Acesso em 30 de 09 de 2017, disponível em Oracle: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html

PATRICK, T. (15 de 02 de 2017). *The Relationship Between Android and Java*. Acesso em 01 de 10 de 2017, disponível em THE ICONIC Tech: https://theiconic.engineering/android-java-fdbd55aadc51

PINSONNEAULT, A., & KRAEMER, K. (1993). *Survey Research Methodology in Management Information Systems.* Journal of Management Information System.

Project Builder. (2017). *Manifesto Ágil: conheça os 12 princípios do Agile*. Acesso em 24 de 09 de 2017, disponível em Project Builder: https://www.projectbuilder.com.br/blog-pb/entry/conhecimentos/manifesto-agil-conheca-os-12-principios-do-agile

RASMUSSON, J. (2017). *Agile vs Waterfall*. Acesso em 29 de 10 de 2017, disponível em Agile In a Nutshell: http://www.agilenutshell.com/agile_vs_waterfall

SIANG, T. (09 de 2017). *What is Interaction Design?* Acesso em 2017, disponível em Interaction Design Foundation: https://www.interaction-design.org/literature/article/what-is-interaction-design

SpinFold. (2016). *History of first android phone*. Acesso em 30 de 09 de 2017, disponível em SpinFold: http://www.spinfold.com/first-android-phone/

Spring. (2017). *Understanding REST*. Acesso em 01 de 10 de 2017, disponível em Spring: https://spring.io/understanding/REST

StackOverflow. (2017). Acesso em 30 de 09 de 2017, disponível em StackOverflow: http://stackoverflow.com

Survio. (2017). *Inquérito Satisfação Cliente*. Acesso em 29 de 10 de 2017, disponível em Survio: https://www.survio.com/pt/pesquisa-de-satisfacao-de-clientes

usability.gov. (2017). *Prototyping*. Acesso em 16 de 12 de 2017, disponível em usability.gov: https://www.usability.gov/how-to-and-tools/methods/prototyping.html

VARGAS, T. C. (2008). Suporte à Edição de UML 2 no Ambiente SEA. Florianópolis, SC.

VARHOL, P. (26 de 08 de 2015). *To agility and beyond: The history - and legacy - of agile development*. Acesso em 29 de 10 de 2017, disponível em Tech Beacon: https://techbeacon.com/agility-beyond-history%E2%80%94-legacy%E2%80%94-agile-development

VENTURA, P. (2016). *Entendendo o Diagrama de Atividades da UML*. Acesso em 29 de 10 de 2017, disponível em Até o Momento: http://www.ateomomento.com.br/uml-diagrama-de-atividades/

Visual Paradigm. (2017). *What is Unified Modeling Language (UML)?* Acesso em 01 de 10 de 2017, disponível em Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

APÊNDICE A – DIAGRAMA DE CLASSES CLIENT-SIDE COMPLETO