# INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

# DE SÃO PAULO – *CAMPUS* SÃO CARLOS

# 

# Pós-Graduação *Lato Sensu* em

# Desenvolvimento de Sistemas para Dispositivos Móveis

# 

# 

# 

# 

# 

# 

# MATHEUS SIMÕES MINGUINI

# 

# 

# 

# 

# CONTROLE E ANÁLISE DE CRÉDITO EM APLICATIVO MULTIPLATAFORMA

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# São Carlos – SP

# 2020

**MATHEUS SIMÕES MINGUINI**

# CONTROLE E ANÁLISE DE CRÉDITO EM APLICATIVO MULTIPLATAFORMA

Monografia apresentada ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – *campus* São Carlos, como parte dos requisitos exigidos para a obtenção do título de especialista em Desenvolvimento de Sistemas para Dispositivos Móveis.

Orientador: Danilo Augusto Moschetto

Data da aprovação: \_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_

MEMBROS DA BANCA EXAMINADORA:

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

Prof. Dr. Danilo Augusto Moschetto (orientador)

Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo

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

Prof. Dr. Rodrigo Elias Bianchi

Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo

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

Prof. Dr. Pedro Northon Nobile

Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo

# AGRADECIMENTOS

Primeiramente, gostaria de agradecer aos meus professores, em especial ao meu orientador, por todo o conhecimento que fora transmitido. O conhecimento - assim como costumavam dizer os filósofos iluministas - é libertador. E vocês, professores, por o transmitirem de forma tão bonita, ajudam a emancipar o futuro e a mente de cada aluno.

Seria muito egoísmo não deixar nessas humildes linhas o meu reconhecimento e profundo agradecimento a todos os matemáticos, físicos, cientistas e pesquisadores que devotaram as suas vidas para o avanço da tecnologia. Tal tecnologia convergiu para que eu pudesse fazer uso na ferramenta desenvolvida nessa monografia.

Não é exagero dizer que não teria concluído essa pós-graduação sem o apoio da minha namorada, Bruna. Ela me ajudou a ser mais resiliente e estoicista, iluminando o meu mundo como mais ninguém.

Finalmente, agradeço a minha grande inspiração: Silvia, minha amada mãe, nós acreditamos. E fizemos.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

*Se eu vi mais longe, foi por estar sobre ombros de gigantes.*

***Sir Isaac Newton***

# RESUMO

A concessão de crédito é uma ferramenta muito importante economicamente, pois o empréstimo monetário aumenta o poder de compra da população, aumentando consequentemente o consumo e assim, também a produção. Entretanto, tal concessão deve ser feita cautelosamente, pois tal empréstimo possui certos riscos que podem ser minimizados com o acesso contínuo a dados que forneçam um panorama do negócio e mercado. Considerando tanto a importância do modelo de crediário no cenário econômico contemporâneo quanto o potencial das ferramentas computacionais para o campo de controle e análise de crédito, este manuscrito apresenta o desenvolvimento de uma aplicação híbrida para o controle e a análise de crédito objetivando auxiliar microempreendedores a examinarem em tempo plausível a saúde do negócio, a fim de minimizar os riscos que a concessão de crédito fornece e, ao mesmo tempo, ajudar na comunicação com o cliente. Primeiro, foi realizado um estudo de viabilidade do projeto; em seguida, escolheu-se a *stack* tecnológica para ir ao encontro do escopo da ferramenta, após isso, elaborou-se a prototipação de telas e fluxo de navegação a fim de entender a jornada do usuário e melhorar a usabilidade da aplicação; e então após o desenvolvimento e fases de testes, o *software* multi-plataforma foi utilizado durante um mês em uma loja de roupas na cidade de Taquaritinga, São Paulo. Os resultados do uso da ferramenta obtidos através de *feedbacks* e de um questionário de aceitação respondido pelos *stakeholders* da loja apontam que o *software* para de dispositivos móveis tem potencial para uso prático no setor de compra e venda utilizando crediário, tendo em sua maioria, aceitação e avaliação positivas dos primeiros usuários.

**Palavras-chaves:** Dispositivos Móveis, Aplicação Híbrida, Crédito

# ABSTRACT

The granting of credit is an important tool economically wise, for the monetary loan increases the population purchase power, increasing therefore the consumption and therefore, the production of goods in the market. However, this act of loaning ought to be done cautiously because it comes with some risks that may be minimized making use of continuous access to data that provides a full picture of the business as well as the market. Considering both the importance of the credit model in the current scenario and potential of computational tools for the credit analyses field, this manuscript presents the development of a hybrid application for the control and analysis of credit aiming to help micro-entrepreneurs to examine in a plausible time the health of their business, in order to minimize the risks that come with making credit available to the customers and at the same time, improve the customer services. First, a study of the viability of the project was drawn up, then the technological stack was chosen to meet the application scope, after that, the screens prototypes were manufactured in order to understand the user journey in the app, to work in its usability betterment and after the development phase as well as the testing phase was finished, the software was used in a clothes store in a city called Taquaritinga, that is situated in São Paulo state, Brazil. The results from using the tool - obtained through feedbacks and also over an acceptance quiz answered by the stakeholders of the store - show that the software for disposable devices has potential to practical using in the sell and buying sector making use of credit granting, having in its majority, acceptance of its first users. The development scope was achieved, once that by making use of the developed application, owners and also employees of the store managed to decrease significantly the access and delivery of information to their clients, optimizing daily tasks and grating credit in a more accurate way to their business.

**Keywords:** Mobile Devices, Hybrid Application, Credit

**LISTA DE FIGURAS**

**Figura 1 - Fluxo atual na web 22**

**Figura 2 - Implementação de Model View Controller para aplicações web. 28**

**Figura 3 - Tipos, operações e endereços em uma abordagem orientada a Serviços. 32**

**Figura 4 - Tipos, operações e endereços em uma abordagem REST 32**

**Figura 5 - Maior valor gerado com a utilização na opinião dos entrevistados 39**

**Figura 6 - Melhor maneira de pagar pelo EasyStore 40**

**Figura 7 - Formas de controles utilizadas atualmente 41**

**Figura 8 - Modelo de negócio da EasyStore 44**

**Figura 9 - Análise hierárquica do fluxo de venda 48**

**Figura 10 - Análise hierárquica do fluxo de recebimento 48**

**Figura 11 - Análise hierárquica do fluxo de compartilhamento de crediário 49**

**Figura 12 - Prototipação do fluxo de entrada no aplicativo 53**

**Figura 13 - Escopo sistêmico da aplicação 55**

**Figura 14 - Camadas da arquitetura do serviço web 56**

**Figura 15 - Divisão de camadas do aplicativo. 57**

**Figura 16 - Fluxo de entrada e autenticação no aplicativo. 59**

**Figura 17 - Quadro Kanban utilizado para a correção dos testes iniciais do aplicativo. 65**

**Figura 18 - Avaliação dos stakeholders após os testes de usabilidade 70**

**LISTA DE ILUSTRAÇÕES**

**Ilustração 1 - Componentes da ferramenta Canvas 42**

**Ilustração 2 - Modelo de negócio Canvas 43**

**LISTA DE QUADROS**

**Quadro 1 - Os nove componentes da ferramenta Canvas 42**

**Quadro 2 - Resultados dos testes de usabilidade condensados 69**

**Quadro 3 - Fluxo de busca do valor do crediário antes e depois do uso da ferramenta 71**

**Quadro 4 - Histórico do crediário antes e depois do uso da ferramenta 71**

**Quadro 5 - Comparação da visibilidade do total de crédito concedido com o uso da ferramenta 72**

# LISTA DE ABREVIATURAS E SIGLAS

# API Application Programming Interface CEP Código de Endereçamento Postal CPF Cadastros de Pessoas Físicas CSS Cascading Style Sheets DML Data Manipulation Language HTA Hierarchical Task Analysis HTML Hypertext Markup Language HTTP Hyper Text Transfer Protocol IDE Integrated Development Environment IoC Inversion of Control JVM Java Virtual Machine JWT Json Web Token MVC Model View Controller ORM Object Relational Mapping REST Representational State Transfer RFC Request for Comments SaaS Solution as a Service SDK Software Development Kit SOA Service-Oriented Architecture SoC Separation of Concerns SQL Structure Query Language SWOT Strengths, Weaknesses, Opportunities and Threats TI Tecnologia da Informação URL Uniform Resource Locator

**SUMÁRIO**

[**1. INTRODUÇÃO 15**](#_Toc43239382)

[**2. FUNDAMENTAÇÃO TEÓRICA 17**](#_Toc43239383)

[**2.1 Tipos de experiência móvel 17**](#_Toc43239384)

[**2.1.1 Aplicativos móveis nativos 17**](#_Toc43239385)

[**2.1.2 Websites móveis (*Web apps*) 18**](#_Toc43239386)

[**2.1.3 Aplicativos híbridos 18**](#_Toc43239387)

[**2.2 Tecnologias 19**](#_Toc43239388)

[**2.2.1 Tecnologias de desenvolvimento híbrido com Ionic 19**](#_Toc43239389)

[**2.2.1.1 Angular 19**](#_Toc43239390)

[**2.2.1.2 Cordova 20**](#_Toc43239391)

[**2.2.1.3 WebView 21**](#_Toc43239392)

[**2.2.1.4 Ionic framework 21**](#_Toc43239393)

[**2.2.2 Tecnologias de desenvolvimento *Backend* 22**](#_Toc43239394)

[**2.2.2.1 HTTP 22**](#_Toc43239395)

[**2.2.2.2 Serviços *Web* 24**](#_Toc43239396)

[**2.2.2.3 Java 24**](#_Toc43239397)

[**2.2.2.4 Spring Boot 25**](#_Toc43239398)

[**2.2.2.5 MySQL 26**](#_Toc43239399)

[**2.2.3 Padrões de projeto 27**](#_Toc43239400)

[**2.2.3.1 Padrão Model View Controller 27**](#_Toc43239401)

[**2.2.3.2 Inversão de Controle (IoC) 28**](#_Toc43239402)

[**2.2.3.3 Object Relational Mapping - ORM 30**](#_Toc43239403)

[**2.2.3.4 REST 31**](#_Toc43239404)

[**2.3 Considerações finais 33**](#_Toc43239405)

[**3. ESTUDO DE VIABILIDADE 34**](#_Toc43239406)

[**3.1 Trabalhos Relacionados 34**](#_Toc43239407)

[**3.1.1 Meu Crediário 34**](#_Toc43239408)

[**3.1.2 QuantoSobra 35**](#_Toc43239409)

[**3.1.3 Pagthos 35**](#_Toc43239410)

[**3.1.4 Excel 36**](#_Toc43239411)

[**3.2 Coleta de dados 37**](#_Toc43239412)

[**3.2.1 Aplicação do questionário 38**](#_Toc43239413)

[**3.2.2 Resultados gráficos 39**](#_Toc43239414)

[**3.3 Modelo de negócios Canvas 41**](#_Toc43239415)

[**3.4.1 Canvas EasyStore 44**](#_Toc43239416)

[**3.4 SWOT 44**](#_Toc43239417)

[**3.4.1 Oportunidades 45**](#_Toc43239418)

[**3.4.2 Ameaças 45**](#_Toc43239419)

[**3.4.3 Pontos fortes 45**](#_Toc43239420)

[**3.4.4 Pontos fracos 45**](#_Toc43239421)

[**3.5 Considerações finais 46**](#_Toc43239422)

[**4. MODELAGEM E DESENVOLVIMENTO DO APLICATIVO 47**](#_Toc43239423)

[**4.2 Projeto de interface 47**](#_Toc43239424)

[**4.2.1 Análise hierárquica de tarefas 47**](#_Toc43239425)

[**4.2.1.1 Fluxo de venda 47**](#_Toc43239426)

[**4.2.1.2 Receber 48**](#_Toc43239427)

[**4.2.1.3 Compartilhar valor do crediário no WhatsApp 49**](#_Toc43239428)

[**4.2.2 Modelos conceituais 49**](#_Toc43239429)

[**4.2.2.1 Tipos de *interface* 50**](#_Toc43239430)

[**4.2.3 Princípios de usabilidade 51**](#_Toc43239431)

[**4.2.4 Prototipação 52**](#_Toc43239432)

[**4.3 Projeto de Software 53**](#_Toc43239433)

[**4.3.1 Arquitetura e design do *Software* 54**](#_Toc43239434)

[**4.3.2 Camadas da aplicação *Backend* 55**](#_Toc43239435)

[**4.3.2.1 Camada de controle 56**](#_Toc43239436)

[**4.3.2.2 Camada de serviço 56**](#_Toc43239437)

[**4.3.2.3 Camada de persistência 57**](#_Toc43239438)

[**4.3.3 Desenvolvimento do aplicativo 57**](#_Toc43239439)

[**4.4 Apresentação do aplicativo 58**](#_Toc43239440)

[**4.4.1 Fluxo de autenticação 58**](#_Toc43239441)

[**4.4.2 Inserção de crediário 59**](#_Toc43239442)

[**4.4.3 Pesquisar cliente 59**](#_Toc43239443)

[**4.4.4 Vender 60**](#_Toc43239444)

[**4.4.5 Receber 60**](#_Toc43239445)

[**4.4.6 Detalhar a venda 61**](#_Toc43239446)

[**4.4.7 Excluir a venda 61**](#_Toc43239447)

[**4.4.8 Pesquisar o pagamento 61**](#_Toc43239448)

[**4.4.9 Excluir o pagamento 62**](#_Toc43239449)

[**4.4.10 Geração de crédito 62**](#_Toc43239450)

[**4.4.11 Histórico de crédito 62**](#_Toc43239451)

[**4.4.12 Compartilhamento de crediário por WhatsApp 63**](#_Toc43239452)

[**4.4.13 Geração de relatórios 63**](#_Toc43239453)

[**4.5 Considerações finais 63**](#_Toc43239454)

[**5. ANÁLISE E DISCUSSÃO DOS RESULTADOS 64**](#_Toc43239455)

[**5.1 Teste de usabilidade 64**](#_Toc43239456)

[**5.1.1 Testes iniciais 64**](#_Toc43239457)

[**5.1.2 Testes de usabilidade com usuários potenciais 66**](#_Toc43239458)

[**5.1.1 Resultados obtidos na loja de testes 71**](#_Toc43239459)

[**5.4 Limitações do trabalho 74**](#_Toc43239460)

[**5.5 Considerações finais 74**](#_Toc43239461)

[**6. CONCLUSÃO 75**](#_Toc43239462)

[**6.1 Trabalhos futuros 76**](#_Toc43239463)

[**REFERÊNCIAS BIBLIOGRÁFICAS 77**](#_Toc43239464)

1. INTRODUÇÃO

Com a revolução tecnológica, as empresas encontram-se inseridas em um cenário de concorrência acirrada, buscando sempre as melhores práticas para conquistar e fidelizar clientes. Para Silva (1997) a concessão de crédito é considerada um dos principais motivos de crescimento das empresas e tal decisão de concessão está relacionada ao volume de vendas que se pretende atingir em determinados produtos e épocas.

No comércio, segundo Pereira (2000), o crédito assume o papel de facilitador da venda, possibilitando ao cliente adquirir o bem para atender sua necessidade, ao mesmo tempo em que incrementa as vendas do comerciante. Em algumas atividades comerciais, o ganho no financiamento das mercadorias chega a ser maior que a própria margem praticada em sua atividade principal.

Blatt (1999) pontua que a concessão de crédito deve basear-se em informações creditícias íntegras e seguras, no histórico de pagamento, na identificação do cliente, na análise das demonstrações contábeis, na qualidade da cobrança e no fortalecimento da venda a crédito.

Ao conceder um crédito, segundo Santos (2015), as empresas se expõem ao risco da inadimplência, ou seja, a probabilidade de não recebimento do valor parcial ou total do valor do financiamento. O autor ainda pontua que para minimizar esse risco, os empreendedores devem acompanhar e analisar minuciosamente o histórico do credor. Pereira (2000) vai além ao dizer que a clareza e confiabilidade na decisão de concessão de crédito dependerão do detalhamento e qualidade das informações e de que forma estas serão avaliadas dentro do contexto econômico que está em constante mutação.

Para controlar e dominar a informação de crediário, como afirma Reynolds, W. G (2018), precisa-se de um sistema de informação. Segundo o autor, informação em si só possui um valor real para continuar-se competitivo se as organizações possuírem um fluxo firme de dados que evidenciem o negócio. Visto isso, ainda segundo o autor, a tecnologia é uma forte aliada nesses processos organizacionais, contribuindo com a fidelização de clientes.

Acompanhando a revolução tecnológica, a indústria de aplicativos móveis está florescendo ainda mais, o *hardware* dos dispositivos está ficando melhor e com isso, a capacidade dos *softwares* cresceram de maneira exponencial. Soluções antes utilizadas somente em aparelhos considerados imóveis tendem a serem acessados a partir de dispositivos móveis, como Tablets e celulares. (BAALEN et al, 2015).

Os resultados divulgados na 30ª Pesquisa Anual da FGV/EAESP (2019) comprovam o processo de informatização e digitalização das empresas e da sociedade. Segundo a pesquisa, existem 230 milhões de celulares inteligentes (*smartphones*) em uso no Brasil, analogamente para cada habitante, existem dois dispositivos digitais, sendo computador, *laptop*, *tablet* e/ou *smartphone*. Os dados da pesquisa apontam como o uso da TI bem como os seus respectivos gastos e investimentos nas empresas continuam crescendo e amadurecendo, mesmo com a economia retraída nos últimos anos.

Com toda essa revolução tecnológica, Gois (2017) diz que uma grande quantidade de Sistemas Operacionais surgiu, o que por consequência aumentou a dificuldade de padronizar uma linha de desenvolvimento de aplicações que fossem portáveis ou multiplataforma. Por este motivo, surgiram os *frameworks* para compilação híbrida de aplicativos sendo desenvolvidos em uma única linguagem. Tais tecnologias ainda segundo o autor deixam o desenvolvimento mais dinâmico para atender o mercado e, diferentemente de como era antigamente, não existe a limitação para validação de um modelo de negócio. Antes do desenvolvimento híbrido, tudo era limitado a uma plataforma para desenvolver e validar a ideia, e a partir do sucesso e aceitação da mesma, era feita a migração, principalmente para as duas principais plataformas (Android e iOS), o que segundo o autor, levava tempo e também dinheiro, podendo perder uma parcela de usuários do mercado que possuísse uma plataforma não compatível com o aplicativo que até então, era de teste.

Neste cenário, o presente trabalho objetiva desenvolver uma aplicação móvel de apoio ao controle da concessão de crédito para lojas que trabalham com crediário utilizando o paradigma de desenvolvimento híbrido, acessando dados em um serviço *web* externo com autenticação JWT.

A justificativa para a criação desta ferramenta é fundamentada na percepção de que a má qualidade das informações utilizadas na análise de crédito comumente levam a decisões que geram prejuízos para a empresa que concedeu o crédito. Além disso, os prejuízos decorrentes de uma má gestão de crediário também inflacionam o custo das despesas administrativas levando a empresa à perda de competitividade. (Completar dizendo os motivos pelos quais a ferramenta traz melhores informações)

Este trabalho está organizado em capítulos, sendo o segundo o capítulo a respeito da fundamentação teórica, isto é, uma introdução e apresentação das tecnologias e paradigmas utilizados no desenvolvimento da ferramenta deste objeto de estudo. O terceiro capítulo discorre sobre a viabilidade de negócio, capítulo este que aborda a aceitação e desejos dos potenciais usuários. No quarto capítulo são relatadas a modelagem e todas as etapas do desenvolvimento da aplicação. No último capítulo exibe-se os resultados obtidos após a instalação e uso por parte de gestores de uma loja de roupas em um estudo de caso prático.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados conceitos técnicos relacionados ao desenvolvimento do aplicativo com relação à arquitetura e às tecnologias.

**2.1 Tipos de experiência móvel**

De acordo com Wilken (2016), é importante entender que existem vários modos de construir aplicações para dispositivos móveis, e cada um possui seus pontos fortes e fracos. Existem três tipos básicos: *Apps* nativos, *mobile* *websites* e *apps* híbridos. Abaixo, discorre-se brevemente sobre os três tipos básicos.

2.1.1 Aplicativos móveis nativos

Segundo Wilken (2016), para criar-se aplicações nativas, desenvolvedores escrevem o código na linguagem padrão para aquela plataforma específica, como Objective C ou Swift para iOS e Java ou Kotlin para Android. Os desenvolvedores compilam aquele código e o instalam em um dispositivo, usando um SDK para acessar os dados no dispositivo ou carregá-los de um servidor externo usando requisições HTTP (seção 2.2.2.5). A SDK é, *e*m termos simples, umum grupo de ferramentas que permite a programação de aplicativos móveis.

Griffith (2017) relata que tipicamente o código nativo é a solução que a maioria dos desenvolvedores pensam quando precisam criar uma aplicação móvel porque possui muitas vantagens:

1. Os desenvolvedores trabalham em um ambiente de desenvolvimento integrado (do Inglês, *Integrated Development Environment*, ou ainda, IDE) desenvolvido especificamente para uma determinada plataforma: Android Studio para o Android e XCode para o iOS;
2. Como a codificação está sendo feita utilizando a API disponibilizada para uma determinada plataforma, não precisa-se de "pontes extras" para ter acesso a tais bibliotecas;
3. O fato da aplicação estar sendo executada em uma plataforma nativa ajuda no melhoramento da performance pois não existe camadas intermediárias de código que podem afetar tal performance.

2.1.2 Websites móveis (*Web apps*)

De acordo com Wilken (2016), *websites* móveis funcionam bem em um dispositivo móvel e são acessados através de um *browser* móvel. Tais aplicações nada mais são do que *websites* vistos em um *browser* para dispositivos móveis, desenvolvidos especialmente para caber nas telas de tais dispositivos.

Griffith (2017) nos diz que a grande vantagem desse paradigma de desenvolvimento é a abrangência de mercado. Segundo ele, além de iOS e Android, plataformas móveis adicionais se tornam disponíveis e dependendo do mercado alvo, tal abrangência pode tornar-se um fator crítico. Além disso, as atualizações são mais rápidas e fáceis, pois a única dependência é a atualização do servidor.

2.1.3 Aplicativos híbridos

Wilken (2016) diz que um aplicativo híbrido é um *app* móvel que contém uma instância isolada de um *browser*, comumente chamado de *WebView (seção* 2.2.1.3) para rodar uma aplicação *web* de dentro de um *app* nativo. Tal tecnologia faz uso de um empacotador de *apps* nativos que consegue comunicar-se com uma a plataforma nativa do dispositivo e também com a *WebView*. Isso significa que aplicações *webs* podem rodar em um dispositivo e ter acesso a recursos como câmera e funcionalidades de GPS, por exemplo.

Como bem nos assegura Cheng (2017), antes de fazer uma decisão entre aplicações nativas ou híbridas, o time de desenvolvimento precisa entender a natureza da ferramenta a ser construída. Aplicações híbridas fazem sentido em um contexto de conteúdo mais centrado, como os de leitura de notícias, fórum online ou portfólio de produtos. O autor ainda destaca que é importante considerar a habilidade do time, isso porque muitas empresas de aplicativos precisam contratar desenvolvedores para Android e iOS para suprir a demanda dessas duas plataformas; entretanto, para aplicativos nativos, desenvolvedores *Frontend* já suprem a necessidade e normalmente é mais fácil encontrá-los do que desenvolvedores Java, Kotlin, Swift e Objective-C.

O desenvolvimento multiplataforma propícia, de acordo com Griffith (2017), outras vantagens:

1. O projeto possui um orçamento menor, visto que não é necessária uma equipe dedicada para cada plataforma;
2. A curva de aprendizagem para essa tecnologia é menor, uma vez que um desenvolvedor com experiência em desenvolvimento *web* não transmuta totalmente de escopo, gerando assim, uma oferta maior de profissionais no mercado.

Entretanto, Griffith (2017) relata uma grande desvantagem dessa solução quando comparada com a uma solução nativa: a performance. Como a aplicação é uma *web app*, a mesma fica limitada pela performance e capacidades do *browser* do dispositivo, ou seja, a performance pode variar de acordo com a capacidade computacional do dispositivo do usuário.

Enfim, diversos fatores podem influenciar na escolha entre o desenvolvimento híbrido e, o nativo. Como relata Cheng (2017), comparado com apps nativos, aplicações híbridas tem seus benefícios e desvantagens. Segundo o autor, o benefício mais considerável é a possibilidade que os desenvolvedores possuem de utilizarem tecnologias e habilidades do desenvolvimento *web* e usarem também, apenas uma base de código para diferentes plataformas.

**2.2 Tecnologias**

2.2.1 Tecnologias de desenvolvimento híbrido com Ionic

Ionic é uma SDK (*Software Development Kit*) de código aberto que usa um conceito chamado *native-feeling* *mobile* apps, que basicamente sugere a ideia de desenvolver aplicativos móveisutilizando das tecnologias *web* (Gois, 2017).

A seguir são apresentadas as tecnologias utilizadas pelo *framework* bem como um o contexto de criação e o cenário atual do mesmo.

2.2.1.1 Angular

De acordo com Branas (2014), criado por Misko Hevery e Adam Abrons em 2009, o AngularJS é um framework de código aberto, do lado do cliente (navegador) que promove uma alta produtividade no desenvolvimento *web*. Ainda segundo o autor, o resultado de utilizar uma tecnologia como o Angular é o desenvolvimento de componentes expressivos, reusáveis e de fácil manutenção, deixando de lado muito código desnecessário e mantendo o time focado no modelo negocial do *software*.

O Angular, segundo Branas (2014), é um framework que adota um padrão de desenvolvimento cujo objetivo é prover uma separação clara de preocupações entre camadas de aplicação (mais sobre tal arquitetura pode ser visto no capítulo 4), provendo modularização, flexibilidade e testes mais assertivos e fáceis de desenvolver.

Segundo Khanna (2017), o framework é baseado na linguagem interpretada denominada JavaScript, a mesma fornece uma forte base para aplicações *webs* escaláveis e complexas. A ferramenta proporciona um modo de criar componentes reusáveis, colocando *templates* padrão no HTML, e reutilizar lógica de negócio com a habilidade de ligardados dinamicamente - ou também conhecido pela expressão em Inglês *two-way-data-binding*.

De acordo com Lim (2017), Angular é um framework para criar *Single Page* *Application* - Aplicações que não fazem requisição ao servidor a cada interação dousuário uma vez que exibem os dados obtidos assíncronamente através da manipulação dinâmica do documento utilizando Javascript.

2.2.1.2 Cordova

Como bem nos assegura Camden (2016), Cordova é um framework de código aberto que permite o convertimento de HTML, JavaScript e CSS em uma aplicação nativa que pode rodar em iOS, Android e em outras plataformas móveis. A tecnologia usa um embrulhador de app nativo na *WebView* (seção 2.2.1.1). Além disso, a ferramenta provê acesso a funcionalidades de *hardware* como câmera e fotos. Diferentemente de simples páginas *web*, aplicações que implementam o Cordova podem ser encontradas e vendidas em lojas de aplicativos como qualquer outro app desenvolvido nativamente.

Segundo Phan (2014), o Apache Cordova é a ponte entre aplicativos nativos e HTML/JavaScript. Precisa-se dessa ferramenta para “buildar” o aplicativo para submissão na *Google Play* e *Apple Store*.

De acordo com Lopes (2016), o Cordova utiliza o ponto forte da *web* por ter linguagens padronizadas e um ambiente de execução para construir aplicativos. O autor ainda diz que só escrever HTML, CSS e JS não é o suficiente para ter um aplicativo no fim, pois o que o Cordova faz é prover uma casca nativa para a aplicação, sendo responsável por subir um *browser* que fará a execução do código, ou seja, o papel da tecnologia é apenas criar essa janela de navegador e fazer a comunicação de chamadas de código para acesso de funcionalidades nativas do App.

Phan (2014) relata que o antigo nome do Cordova era Phonegap. Depois de ser comprado pela Adobe, foi doado para a Apache e o nome original da plataforma mudou para Cordova.

2.2.1.3 WebView

Segundo Khanna (2017), cada plataforma mobile nativa tem um componente comum chamado WebView, que nada mais é que um *browser* sem cromo. Esse componente é utilizado para abrir conteúdos *web* armazenados localmente, por exemplo: páginas HTML, arquivos de CSS e código Javascript.

Para Looper (2018), WebView é um browser com configurações mínimas que entrega conteúdo *Web*.

2.2.1.4 Ionic framework

A empresa por que desenvolvedora do Ionic chama-se Drift e já criou outros projetos no passado, como Jetstrap (um construtor de interfaces) e Codiqa, uma ferramenta de arrastar e soltar para construir *websites* móveis. A principal missão da Drift atualmente é ajudar na mudança do desenvolvimento de *software* para a plataforma móvel, especialmente através da melhoria das aplicações híbridas através dos componentes oferecidos no Ionic (Baalen et al. 2015).

Segundo Griffith (2017), Ionic é o um framework construído com HTML, CSS e JavaScript para o desenvolvimento de aplicações híbridas. A ferramenta ainda segundo o autor é simplesmente uma combinação de tecnologias para construir aplicações móveis mais rápido e mais facilmente. A camada principal da pilha é o próprio framework, provendo a interface do usuário para a aplicação.

Logo abaixo está o Angular e de acordo com Branas (2014), esse framework faz parte de uma nova geração de bibliotecas que vieram para apoiar o desenvolvimento de aplicações *webs* mais produtivas, flexíveis, testáveis e com uma melhor manutenibilidade (seção 2.2.1.1). A ferramenta utiliza o Apache Cordova, permitindo com que a aplicação *web* utilize as capacidades nativas do aparelho.

Segundo Wilken (2016), um aplicativo híbrido é um tipo de app móvel que usa uma janela de browser para mostrar a sua interface. O Ionic é uma combinação de ferramentas para construir aplicativos móveis híbridos usando as mesmas tecnologias usadas para criar *Web Services* e *Web Applications* (seção 2.2.2.2) dentro de um aplicativo nativo através do Cordova.

Phan (2018) diz que o Ionic framework possui muitas vantagens sobre os competidores, o autor cita os mesmos como segue:

1. O Ionic é baseado no Angular, que é um framework para aplicações *web* robusto e com uma grande comunidade no universo do desenvolvimento atual;
2. O framework fornece uma lista de componentes de tela que podem ser adaptados de acordo com a necessidade de desenvolvimento, além de serem flexíveis às plataformas;
3. A ferramenta é código aberto e possui uma comunidade consolidada de desenvolvedores;
4. Vasta documentação com amostras de código;
5. Fórum que permite uma comunicação diretamente com a equipe do Ionic.

2.2.2 Tecnologias de desenvolvimento *Backend*

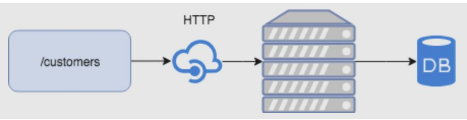
Para Duarte (2017), todo o núcleo de uma aplicação fica longe do usuário, em um servidor *web*, que tratará as requisições de todos os usuários em um único lugar, com um único código fonte. Tal núcleo pode ser escrito em qualquer linguagem de programação desde que o servidor de aplicação a suporte. Atualmente, segundo o autor, existem muitas tecnologias populares para esse tipo de desenvolvimento que permite com que as aplicações sejam dinâmicas no modelo conectado atual. A essa gama de tecnologias cunhou-se o termo *Backend* e a seguir são apresentadas as utilizadas no desenvolvimento da ferramenta deste trabalho.

2.2.2.1 HTTP

Segundo Kurose (2010), HTTP é um acrônimo para Protocolo de Transferência de Hipertexto, estes, como pontua Décio (2001), são documentos que podem conter textos, gráficos, vídeos, áudios e também informações embutidas sobre outros documentos enquanto que protocolos são um conjunto de regras que administram os procedimentos para troca de informações.

O conteúdo na internet é armazenado em servidores *webs* que utilizam o protocolo HTTP como porta de entrada e por isso, são comumente chamados de servidores HTTPs. Tais servidores acessam base de dados, retornando os dados obtidos para aplicações que estabelecem uma conexão com o propósito de solicitar dados (aplicações clientes). Juntos, clientes e servidores fazem os componentes básicos da internet. (Gourley e Totty, 2015).

A figura 1 abaixo ilustra tal fluxo.

Figura 1 - fluxo atual na web

Fonte: Araújo (2019).

O protocolo é utilizado para transmitir recursos e não apenas arquivos. Um recurso, como enfatiza Décio (2001), é qualquer trecho de informação que possa ser identificado por um URL (*Uniform Resource Locator*). O tipo mais comum de recurso é o arquivo, contudo, pode ser uma mídia qualquer, o resultado de uma consulta a um banco de dados que fora gerado dinamicamente e qualquer outro recurso que esteja disponível no servidor.

Segundo Araújo (2019), toda requisição HTTP possui um verbo e tais verbos são úteis no esclarecimento sobre o que tal requisição almeja realizar. No geral, os verbos mais utilizados são POST, GET, PUT, PATCH e DELETE, que querem dizer respectivamente: criar, ler, alterar (PUT e PATCH) e remover. Ainda segundo o autor, existem muitos outros verbos, contudo não são tão frequentemente utilizados.

De acordo com Molinari (2016) todas as respostas HTTPs possuem um *status,* o autor pontua que tais status são formas mais simplificadas de oferecer dicas sobre o que aconteceu na requisição para quem está recebendo a resposta. Existem vários grupos de status, sendo que cada um possui uma classe de números que têm significados diferentes. A tabela 1 ilustra as classes de status mais comuns em um procedimento requisição-resposta na *web*:

**Tabela 1: Classes de status HTTP**

|  |  |
| --- | --- |
| 1xx | Informativo, a requisição foi recebida e o processo continua. |
| 2xx | Ocorreu tudo certo com a requisição. |
| 3xx | Redirecionamento, alguma ação precisará ser tomada para completar a requisição |
| 4xx | Erro no cliente, possivelmente a requisição foi mal feita |
| 5xx | Erro no servidor, a requisição parece válida, mas o servidor não consegue processar. |

Fonte: Molinari (2016).

Todos os detalhes sobre os status apresentados e os vários outros disponíveis para utilização estão descritos na RFC 7231, que pode ser usada como referência, caso necessário.

O HTTP, como pontua Stallings (2005) foi desenvolvido para ser usado na Internet, embora seu desenho definitivo permita que ele possa ser usado em outras formas de aplicação. O autor ainda pontua uma das principais características do protocolo: sem estado (termo conhecido como *stateless*), ou seja, cada comunicação é tratada independentemente da anterior. Por conta disso, tecnologias como JWT (*Json Web Tokens*) e *cookies* de sessão são implementados como meios de autenticação, isso é, porquê o protocolo nem o servidor guardam estados, o cliente precisa obter um algo a ser enviado em todas as futuras requisições, informando assim, que já autenticou-se anteriormente.

2.2.2.2 Serviços *Web*

“Em termos técnicos, um serviço *web* nada mais é que um *software* distribuído cujos componentes podem ser acessados em qualquer dispositivo físico autenticado.” (KALIN, 2013).

Segundo Saudate (2014), cada vez mais, sistemas corporativos exigem o reaproveitamento de funcionalidades, e consequentemente a integração entre diferentes sistemas se torna uma realidade. Essa é uma tendência natural das arquiteturas modernas. O autor pontua que é muito comum uma aplicação, digamos A, acessar outra, digamos B, para buscar dados específicos. Como por exemplo: Ao digitar um CEP em um formulário, precisa-se buscar o respectivo endereço do mesmo, para isso, geralmente acessa-se um serviço com *expertise* de dado um CEP, encontrar seu respectivo endereço. Esses cenários comuns tornam o serviços na *web* muito importantes atualmente, afinal são uma solução prática e de baixo custo para solucionar a incompatibilidade de sistemas e garantir a sua comunicação. Estes permitem ligar diferentes aplicações que integram um sistema, ultrapassando barreiras como o tipo de plataforma ou linguagens de programação utilizadas.

Gourley e Totty (2015) informam que os sistemas, por não serem construídos com a mesma tecnologia, precisam se integrar através de um padrão de comunicação, também conhecido como protocolo. O protocolo usado atualmente entre serviços na *web* como dito na seção 2.2.2.1 é o HTTP.

Concluindo, ressalta-se que todo serviço na *web* é a representação lógica de repetidas ações de negócio que possuem um resultado específico, tais como: Obter a previsão do tempo de uma cidade, acessar preço de ações de um mercado, atualizar um registro de algum cliente, obter dados para relatórios, etc. Tais serviços funcionam como uma caixa preta, isto é, uma camada de abstração para o cliente que está consumindo (SUBRAMANIAN e PETHURU, 2019).

2.2.2.3 Java

Como pontua Herbert (2015), a linguagem foi concebida por James Gosling, Patrick Naughton, Chris Warth, Ed Frank e Mike Sheridan na *Sun Microsystems*, em 1991. No início, a linguagem se chamava “Oak”, mas foi renomeada como “Java” em 1995. Herbert ainda pontua que a motivação original da linguagem não foi a internet, mas sim a necessidade de uma linguagem independente de plataforma. O problema da época era o seguinte: Se a empresa precisa-se desenvolver um *software* para operar em duas máquinas com sistemas operacionais distintos, por exemplo uma executando Linux e outra Windows, teria que desenvolver dois executáveis - um para cada plataforma.

Deitel e Harvey (2017) são incisivos ao afirmar que a principal característica do Java é ser capaz de escrever programas que podem ser executados em uma grande variedade de sistemas computacionais e dispositivos controlados por computador. Uma das principais diferenças entre a plataforma Java e as demais linguagens existentes na época é que o Java é executado sobre uma JVM, ou *Java Virtual Machine*. Qualquer plataforma de *hardware* ou equipamento eletrônico que possa executar uma máquina virtual conseguirá executar o Java. Dado essa característica da linguagem, o *slogan* da mesma é: “Escreva uma vez, execute em qualquer lugar” (LUCKOW, D. H; MELO, A. A, 2010).

“Por uma feliz casualidade, a *web* explodiu em popularidade em 1993 e a Sun viu o potencial de utilizar o Java - que até então era conhecido como Oak - para adicionar conteúdo dinâmico às páginas *web*. Atualmente, a linguagem é utilizada para desenvolver aplicativos corporativos de grande porte, bem como aprimorar e desenvolver servidores *web*” (DEITEL; HARVEY, 2017).

2.2.2.4 Spring Boot

Segundo Karanam (2019), o Spring vem sendo o framework número um de escolha para o desenvolvimento de aplicações no mundo Java por mais de uma década. O framework tornou-se muito famoso por ser *open-source* e se basear no padrão de projeto denominado inversão de controle (IoC), padrão esse abordado na seção 2.2.3.2 desse capítulo.

Como pontua Walls (2019), uma aplicação normalmente é feita de vários componentes, cada qual responsável por uma parte da funcionalidade de toda a aplicação. Quando a aplicação é executada, esses componentes precisam ser criados e integrados, e é nesse ponto que o Spring aparece: Idealizada por Rod Johnson, o framework surgiu para aumentar a produtividade e diminuir o acoplamento.

Porém, de acordo com Boaglio (2017), desde de 2003, o ecossistema do *framework* cresceu muito e se modularizou, isto é, vários partes com escopos bem definidos foram criados. Por exemplo: Para segurança, existe o Spring Security (Ajuda na implementação de autenticação e verificação de controle de acesso), para acessar e persistir dados, tem-se o Spring Data, para criar processos com data e hora agendados, usa-se o Spring Batch e essa lista é enorme. Esse crescimento todo trouxe alguns problemas: Com muitos módulos, vieram muitas dependências e configurar uma aplicação em Spring já não era tão trivial assim e é então que surge o Spring Boot.

O Spring Boot nasceu em 2014, e a sua ideia inicial veio da necessidade de o Spring *Framework* ter suporte a servidores *web* embutidos. A tecnologia, como afirma Rajput (2018), é uma maneira eficiente de criar uma aplicação utilizando os módulos do Spring e colocá-la no ar, funcionando sem depender de um servidor de aplicação previamente instalado na máquina. O objetivo com isso é acelerar e simplificar a parte de desenvolvimento e configuração das aplicações que fazem uso da stack do Spring.

2.2.2.5 MySQL

De acordo com Date (2003), um sistema de banco de dados é basicamente um sistema computadorizado de manutenção de registros. O banco de dados, por si só, pode ser analogamente considerado como um armário de arquivamento, isto é, um repositório de coleção de dados computadorizados. Os usuários de tais bancos de dados, ainda segundo o autor, podem solicitar que o mesmo realize diversas operações envolvendo os arquivos persistidos, como:

1. Acrescentar um registro;
2. Inserir dados em arquivos existentes;
3. Buscar dados de arquivos existentes;
4. Excluir dados de arquivos existentes;
5. Alterar dados em arquivos existentes;
6. Remover arquivos do banco de dados.

De acordo com Davis e Phillips (2008), MySQL é um banco de dados relacional gratuito criado nos anos 90 para sanar a crescente necessidade dos computadores de gerenciar informações de maneira inteligente.

Bento (2017) aborda que o MySQL é o banco de dados do tipo relacional, isto é, no qual guardamos informações em estruturas no estilo de tabelas, sendo que cada linha da tabela é um novo registro. O motivo de ser chamado “relacional”, como nos assegura Beighley e Morrison (2010), é porque linhas entre as tabelas se relacionam entre si através de registros identificadores.

2.2.3 Padrões de projeto

Freeman et al. (2007) diz que padrões de projetos são formas de descrever problemas que acontecem rotineiramente, informando também a solução raiz dos problemas atacados de uma maneira que permite que a solução seja utilizada com a mesma frequência que os seus respectivos problemas aparecem. Gamma et al. (1994) também informa que padrões de projetos ajudam a reutilizar e testar códigos com mais simplicidade.

A seguir são apresentados alguns dos padrões de projetos utilizados neste trabalho.

2.2.3.1 Padrão Model View Controller

Para a construção do aplicativo, usou-se como framework de desenvolvimento *Frontend* o Angular (Seção 2.2.1.1) e tal *framework* implementa o padrão aqui explicado.

O MVC é um dos padrões mais citados nos projetos de *software*. Começou como um *framework* desenvolvido por Trygve Reenskaug para uma plataforma chamada *SmallTalk* no fim da década de 1970. Desde então, o padrão vêm sendo muito utilizado em *frameworks* de componentes de telas e ao pensar no *design* da *interface* com o usuário (FOWLER, 2003).

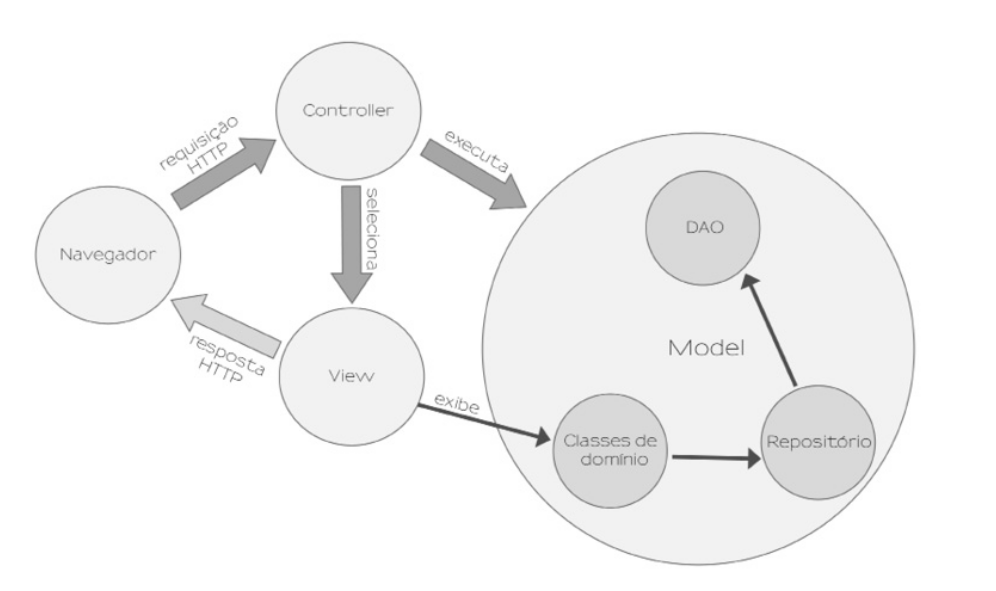
De acordo com Silveira et al. (2012) um conceito amplo e difundido como excelente prática arquitetural é o da Separação de Responsabilidades, ou SoC, do inglês, *Separation of Concerns*. Para diminuir o acoplamento, aumentar a coesão, promover a flexibilidade e garantir responsabilidades únicas é preciso saber separar essas responsabilidades.

Conforme complementa Weissmann (2012), o padrão MVC surgiu, então, justamente para facilitar a separação da aplicação em responsabilidades enxutas, o mesmo os divide em três papéis (comumente chamados de camadas) fundamentais: o modelo, a visualização e o controle. Ainda segundo o autor, as camadas possuem as seguintes semânticas.

1. Camada de Modelo: objeto não visualizável que contém todos os dados e comportamentos do domínio da aplicação.
2. Camada de Visualização: representa a disponibilização de um modelo na *interface* do usuário. O único papel dessa camada é disponibilização de dados na tela, qualquer mudança nos dados dispostos são gerenciadas pelo terceiro membro da trindade: O controlador da camada de controle.
3. Camada de Controle: obtém as informações inseridas, manipula o modelo e faz com que a camada de visualização atualize e mostre os dados na tela.

Abaixo, na figura 2, exemplifica-se o uso do modelo através de um diagrama.

Figura 2 - Implementação de Model View Controller para aplicações web.



Fonte: Silveira et al. (2012).

Concluindo, Silveira et al. (2012) informa que independentemente da definição formal desse padrão ou das interpretações que o mercado adotou do MVC, a característica de maior valor é a de separar os componentes de apresentação do resto da aplicação (conhecido como *Separated Presentation*).

2.2.3.2 Inversão de Controle (IoC)

O Spring framework (abordado na seção 2.2.2.4) tornou-se muito famoso no universo Java para web porque implementa o padrão de inversão de controle (WEISSMANN, 2012).

Abaixo discorre-se sobre o problema atacado por esse padrão bem como a solução.

Segundo Fowler (2003), ao pensar no design de uma aplicação orientada a objetos, o foco não deve ser apenas na relação entre dois objetos, mas também entre grupos de objetos que trabalham em conjunto para uma determinada finalidade. Ao desenhar pequenas classes que, juntas, realizam uma tarefa maior, o projeto ganha em simplicidade e modularidade.

Como nos assegura Silveira et. al em sua obra Introdução à Arquitetura de Design de Software (2012) , mesmo com o baixo acoplamento haverá sempre uma ligação entre duas classes (ou modelos) que precisam trabalhar em conjunto. Normalmente, há um modelo que necessita dos serviços oferecidos por outro. Qualquer mudança nesse modelo que está sendo acessado pode afetar o comportamento da primeiro. Há aqui uma relação natural de dependência.

Weissmann (2012) aborda que o IoC é analogamente conhecido como a aplicação do princípio de Hollywood, cujo nome é baseado em um clichê comum naquela cidade: quando atores concorrem por um papel, os produtores responsáveis pela peça/filme costumam dizer aos participantes do processo seletivo: “não nos chame, deixe que nós chamamos você”.

Para ilustrar o conceito, seria interessante imaginar o seguinte exemplo: existe uma aplicação que possui a classe A que usa a classe B, conforme apresentado abaixo:

*public class A*

*private B b;*

*public A( ){*

*b = new B();*

*}*

*}*

No exemplo acima, percebe-se que a classe A depende de uma instância da classe B, isso é conhecido como forte acoplamento entre classes, pois a classe A é quem solicita e controla o objeto b.

Com isso surge um problema: mudanças nas classes que compõem uma dependência irão refletir diretamente nas classes que possuem uma referência dela. Visando resolver esse problema, surge o IoC. Ao aplicar o padrão na classe A de exemplo, a mesma fica como segue.

*public class A*

*private B b;*

*public A( ){};*

*public void setB(B instance)*

*this.b = instance*

*}*

*}*

Com essa mudança, a classe A obtém uma referência da classe B usando o método *setB(),* com isso, as seguintes afirmações são verdades:

1. A classe A pode obter uma referência da classe B sem saber como a classe B é instanciada;
2. A classe B pode ser uma interface, classe abstrata ou mesmo uma classe concreta;
3. Não existe mais nenhum acoplamento entre a classe A e a classe B;
4. Qualquer alteração feita na classe B não afeta a classe A e vice-versa;

Ao dizer que os objetos não vão mais atrás de suas dependências, mas que agora devem apenas recebê-las de alguém, inverte-se quem está no controle de gerenciar a dependência. Em vez de fazer os objetos irem atrás daquilo que precisam, o programador faz com que eles recebam os objetos já inicializados e preparados para uso (SILVEIRA et al., 2012).

2.2.3.3 Object Relational Mapping - ORM

Atualmente, como nos assegura Oliveira (2014), existem dois paradigmas muito utilizados no desenvolvimento de *software*: Orientado a objeto e relacional.

O paradigma orientado a objeto possibilita criar unidades de código mais próximas da forma como os seres humanos pensam e agem, facilitando o processo de transformação das necessidades diárias para uma linguagem de programação (LEITE, 2018).

Conforme Date (2003), o paradigma relacional é baseado na Álgebra Relacional. Em tal paradigma os dados são armazenados em tuplas em conjunto de atributos constituídos por tipo de dado e valor. A definição de um domínio ou combinação de domínios que identifica individualmente cada elemento da relação é chamada de chave primária, enquanto a identificação da relação entre os itens únicos da tabela é dada por uma chave estrangeira. Conceito também abordado na seção sobre MySQL (2.2.2.4).

Persistir o estado de um objeto, isto é, uma representação abstrata de algo tangível, é um requisito fundamental para a grande parte das aplicações. A fim de manter os dados, precisa-se de um armazenamento que não seja volátil, como por exemplo, as bases de dados (KONDA, 2014).

Ao desenvolver uma aplicação, como aborda Konda (2014) normalmente espera-se que os dados armazenados ultrapassem o tempo de vida útil da própria aplicação que os salvou, afinal, uma aplicação pode migrar de uma plataforma para outra e mesmo assim manipular o mesmo conjunto de dados. Atualmente, grande parte do número dos bancos de dados do mercado fazem uso do paradigma relacional e portanto, quando utilizados levam consigo tal conceito.

Nesse cenário, como bem defende Cordeiro (2014), temos dois paradigmas diferentes que precisam de uma integração. Ao persistir um dado de uma aplicação em um banco de dados, as diferenças entre ambos os modelos precisam ser ajustadas. Some isso ao fato da existência de conceitos como polimorfismo, herança, composição e diversas outras características do modelo orientado a objeto que não estão presentes no modelo relacional. A essas diferença de paradigmas dá-se um nome: Impedância de objeto relacional, ou também conhecido no Inglês como *Impedance Mismatch.*

*Object Relational Mapping* (ORM), segundo Oliveira (2014), é um conceito que objetiva resolver a problemática da impedância citada no parágrafo anterior, isto é, transformar as informações de um banco de dados, que estão no modelo relacional para classes, no paradigma Orientado a Objetos de um modo fácil.

“Como toda especificação, existem as implementações e toda linguagem orientada a objetos possui muitas delas. Atualmente, no universo Java, o mais utilizado é a ferramenta denominada Hibernate” (CORDEIRO, 2014).

No aplicativo desenvolvido para o presente trabalho, usou-se o framework ORM Hibernate.

Como afirma Konda (2014), um *framework* ORM torna-se muito útil na hora de realizar consultas, manipular dados e até mesmo a gerar relatórios complexos. A verbosidade de um SQL puro não é necessária, nem mesmo os loops feitos para popular objetos. Tudo isso já está abstraído pela ferramenta ORM.

Outra vantagem, segundo Leite (2018), é a portabilidade da aplicação entre banco de dados e tratamento de acesso simultâneo.

2.2.3.4 REST

Sistemas sem integração com outros sistemas externos vivem isolados em silos de dados, algo cada vez mais raro, já que a necessidade de se integrar com outros sistemas é crescente. Por isso, ao desenvolver uma nova aplicação, pensa-se em como será sua integração com outras no futuro (SILVEIRA et al., 2012).

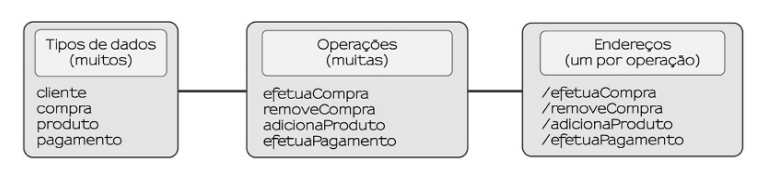
REST é um acrônimo para *Representational State Transfer* (Representação do estado de transferência). O mesmo é um conceito arquitetural de *software* e não uma linguagem de programação ou tecnologia (SUBRAMANIAN e PETHURU, 2019).

Como já visto na seção 2.2.2.2, serviços *web* são funcionalidades de *software* providas por um conjunto de sistemas computacionais para um outro conjunto distinto. Tais *softwares* comunicam-se usando o protocolo abordado na seção 2.2.2.1 (SUBRAMANIAN e PETHURU, 2019).

Segundo Araújo, o estilo REST é um conjunto de práticas de engenharia de *software* que contém requisitos que devem ser usados para criar serviços *webs* em sistemas distribuídos. O padrão arquitetural provê um conjunto de regras de design a fim de criar serviços mostrados como recursos e, em alguns casos, fontes de informações específicas como dados e funcionalidades.

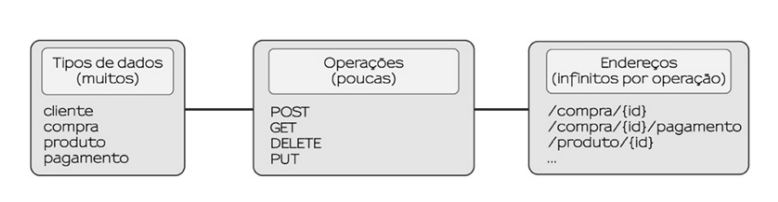
Segundo Silvera et al (2012), na abordagem orientada a serviços tradicional, a tendência é forçar os desenvolvedores a criarem múltiplas operações, além de diversos tipos de dados. Por exemplo, um sistema simples, com tipos representando clientes, compras, produtos e pagamentos, possui também diversas operações (serviços), como mostrado na figura 3.

Figura 3 - Tipos, operações e endereços em uma abordagem orientada a Serviços.

****Fonte: Silveira et al. (2012).

No padrão REST, ainda segundo o autor, é possível alterar essa forma de exposição do sistema ao disponibilizar mais endereços em uma URI que tenha como base a entidade (recurso) acessado, a figura abaixo mostra a mudança.

Figura 4 - Tipos, operações e endereços em uma abordagem REST

****

Fonte: Silveira et al. (2012)

Conforme visto na figura 4 e enfatizado por Araújo (2019), no padrão REST não é incluída na URI a ação que quer-se performar. A operação é inferida através do método HTTP (seção 2.2.2.1) enviado.

Para ilustrar o conceito, seria interessante imaginar o seguinte exemplo: o usuário acessa o recurso */compra/id* passando um método HTTP *PUT*, tal método, de acordo com RFC 2616 Fielding, et al. é conhecido por mudar a representação de um recurso no servidor, logo pode-se induzir que ao acessar o recurso */compra/id* através desse método quer-se, na verdade, atualizar dados da compra com o id informado.

Com a utilização do padrão apresentado, o dado acessado no serviço possui uma representação mais abstrata orientada ao método passado e não somente a nomenclatura da operação, com isso, a URI torna-se mais fácil de lembrar, menor e fala por si só.

**2.3 Considerações finais**

Neste capítulo foram contextualizadas as principais tecnologias de desenvolvimento bem como os padrões empregados escolhidos para o desenvolvimento da aplicação.

No Capítulo seguinte, serão discutidos algumas ferramentas disponíveis no mercado atualmente e apresentada a coleta de dados para a elaboração da proposta da ferramenta.

3. ESTUDO DE VIABILIDADE

**3.1 Trabalhos Relacionados**

Com o objetivo de investigar o que existe disponível atualmente no mercado para o controle de crediário, foram levantados quatro exemplos de soluções tecnológicas para o estudo comparativo, as ferramentas são: MeuCrediário, QuantoSobra, Pagthos e o sistema de planilhas Excel da Microsoft. Abaixo, explica-se o que cada ferramenta é, bem como suas respectivas funcionalidades, preços e pontos fracos.

3.1.1 Meu Crediário

O aplicativo é uma plataforma SaaS (Solution as a service) de gestão de cobrança e análise de crédito que é indicado para lojas de vestuário, calçados e acessórios, onde o valor médio das vendas fica entre R$100 e R$ 800. Para a utilização da ferramenta existe um custo, os valores variam dependendo do tamanho, modelo de negócio e volume de vendas. Os planos mensais para contratação custam a partir de R$14,90 e o preço da análise de crédito a partir de R$2,99 (Fonte: https://meucrediario.com.br/)

São funcionalidades do Meu Crediário:

1. Análise de crédito:Consulta os principais órgãos de proteção ao crédito simultaneamente para calcular o risco de cada cliente e sugerir limites seguros;
2. Cobrança: Envio de mensagens pelo celular em diferentes datas após o vencimento da parcela;
3. Vendas:Persistência dos dados da venda;
4. Negativação: Negativa os devedores.

São pontos negativos do Meu Crediário:

1. A venda não possui a inclusão de itens. Para realizar uma venda, o usuário apenas digita o valor total, o que pode prejudicar a rastreabilidade do que foi comprado ao gerar a dívida;
2. Cada consulta de análise de crédito possui um valor.

3.1.2 QuantoSobra

O QuantoSobra é Sistema de Gestão que visa facilitar o controle do negócio de pequenas empresas. A ferramenta é paga mensalmente e conta com vários planos, sendo o mais básico R$ 59,00 e o mais caro R$ 199,00. Ambos os planos proporcionam o uso de três usuários, caso o contratante queira mais usuários, um valor de R$ 10,00 por usuário é acrescentado na mensalidade (Fonte: https://www.quantosobra.com.br).

São funcionalidades do QuantoSobra:

1. Recibos e carnês: Emite recibos não fiscais para o cliente e caso a compra seja no crediário, um carnê também é impresso;
2. Contas a receber: Realiza a inclusão de juros por dia de atraso, possui filtros por clientes, por contas que possui atrasos, etc;
3. Limite de crédito: Sistema notifica quando o limite estiver estourado;
4. Ponto de venda: Tela responsável por registrar uma venda no sistema;
5. Controle de estoque: Ao realizar uma venda, o estoque é baixado e a venda registrada no histórico do cliente. No fluxo de caixa automaticamente já são lançadas as parcelas recebidas e criadas as contas a receber.

São pontos negativos do QuantoSobra:

1. O preço, pois entre os três sistemas pesquisados, o mesmo é o que possui o maior entre eles;
2. Não possui a versão para dispositivos móveis.

3.1.3 Pagthos

Pagthos é um sistema online que ajuda na segurança e disponibiliza ferramentas para redução da inadimplência, automatizando ações de cobrança quando necessário. A ferramenta é indicada para lojas de calçados e confecções, lojas de móveis e eletro, lojas de óticas e saúde, clínicas odontológicas e exames, academias, representantes comerciais, distribuidores, outras empresas de varejo e prestadores de serviço. O sistema possui vários planos, cada plano com certas funcionalidades: O plano mais básico é R$39,00 e te dá direito a até 50 vendas mensais enquanto que o plano mais caro é R$99,00 e dá direito a até 200 vendas mensais (Fonte: https://pagthos.com.br).

São funcionalidades básicas do Pagthos:

1. Vendas;
2. Emissão do contrato das vendas;
3. Emissão de carnês;
4. Boleto bancário (R$2,45 por recebimento);
5. Suporte por email / chat online.

São pontos negativos do Pagthos:

1. Preço não é muito acessível;
2. Não possui a versão para dispositivos móveis.

3.1.4 Excel

Excel é um programa de planilha eletrônica desenvolvido pela Microsoft (MANZANO, J; MANZANO, A, 2016).

O Microsoft Excel é uma planilha eletrônica que pode ser imaginada como uma grande folha de papel, dividida em 256 colunas e 65.536 linhas, nas quais é possível o armazenamento de textos e números. Uma funcionalidade que aumentou o uso da ferramenta foi a possibilidade de manipulação dos valores e textos inseridos na planilha através de fórmulas disponíveis para organizar os dados, gerando uma informação mais especializada com o contexto do usuário (MANZANO, J; MANZANO, A, 2016).

São funcionalidades do Excel:

1. Persistência dos dados em formato tabular;
2. Plataforma programável, ou seja, o usuário pode, desde que possua um conhecimento técnico prévio, gerar planilhas, realizar somas automáticas e fórmulas aritméticas a fim de realizar um cruzamento de dados e gerar informação;
3. Os dados, quando utilizados na nuvem, são armazenados automaticamente com backups;
4. Caso a planilha seja armazenada em nuvem, pode ser acessada de dispositivos como Tablet, *Smartphone* e *Desktops* desde que haja conexão com a internet.

São pontos negativos do Excel:

1. Planilhas eletrônicas, de forma generalizada, possuem um alto risco de integridade e imprecisão de dados. Isso acontece porque os criadores das planilhas tem que lidar com muitas fórmulas e códigos e qualquer erro de digitação ou de transferência na hora de criar uma nova célula ou coluna gera uma informação que pode resultar em um desvio nos valores apresentados;
2. Existe uma limitação de espaço no Excel e seus similares. Uma planilha não pode ter um número muito grande de registros (linhas). Uma solução paliativa seria desenvolver mais planilhas quando necessário, contudo, nesse caso, o trabalho se torna difícil e ainda mais arriscado;
3. Não existe uma pesquisa com filtros e que seja rápida, pois não existe uma possibilidade de manipulação de dados relacionais ou não-relacionais fornecidos pelos bancos de dados atuais.

**3.2 Coleta de dados**

Para o levantamento de dados deste projeto foram escolhidos os instrumentos de questionário com potenciais usuários utilizando a técnica estatística de amostragem. Para Peterson (2016), amostragem é uma técnica onde escolhe-se uma pequena parte de um grupo grande para estimar características do grupo todo, por exemplo, existe um grupo de mil pessoas e queremos descobrir quantas gostam de Matemática. Levaria um tempo enorme para questionar todos eles. Ao invés disso, entrevista-se apenas uma parcela daquele grupo e utiliza-se os resultados dessa parcela menor para elaborar conclusões aproximadas do grupo todo. O autor ainda pontua a importância de garantir que a parcela escolhida para a amostra seja a representação mais aproximada possível do grupo todo.

A escolha do método de questionário ocorreu pelo dinamismo, visto que uma entrevista demandaria agendamento e possivelmente tornaria-se um gargalo no processo de desenvolvimento. A pesquisa contém perguntas qualitativas e quantitativas a fim de entender as principais necessidades do grupo da amostra, entendendo o perfil do público-alvo e gerando assim valor e viabilidade de negócio na construção de um aplicativo como ferramenta de controle de crediário.

Recursos extras como a apresentação de um rápido protótipo de telas foi elaborado em Wireframe a fim de possibilitar uma noção mais tangível da proposta sistêmica.

O período de coleta das respostas foi no primeiro semestre de 2020.

3.2.1 Aplicação do questionário

O questionário elaborado conteve sete perguntas que podem ser vistas no anexo A, tais perguntas objetivavam a validação da ideia e modelo de negócio a ser adotado (seção 3.3). Proprietários de lojas de roupas e comerciantes individuais se propuseram a responder o questionário que em média, demorou 15 (quinze) minutos para ser completado. A amostra contou com 10 (dez) pessoas.

Abaixo seguem os resultados condensados.

1. 70% das pessoas responderam que estariam dispostas a instalar um aplicativo no *smartphone* para lançar as vendas e os pagamentos dos clientes;
2. 60% utilizam Android enquanto 40% iOS;
3. 100% das pessoas da amostra disseram que possuem acesso a internet móvel;
4. Quando perguntado sobre o - Variação do tempo - levado para saber o quanto o cliente está devendo (Seja por motivos de levantamento de dívidas ou até mesmo quando perguntado pelo cliente), as respostas foram bem variáveis, como segue: 2 minutos, 2 minutos, 3 minutos, 4 minutos, 5 minutos, 5 minutos, 7 minutos, 9 minutos, 12 minutos e 15 minutos. A partir do exposto, tirou-se a mediana da amostra - Segundo Peterson (2016), a mediana é o número central de um conjunto de dados ordenados do menor para o maior. Concluiu-se então que em média, leva-se 3,5 minutos para encontrar o valor devido por um cliente;
5. A moda da funcionalidade extra que um aplicativo de controle de crédito deve possuir foi “Compartilhamento do valor devido no *WhatsApp*” com 5 respostas. Tal informação é relevante pois ajuda no entendimento de como gerar mais valor negocial e qual funcionalidade extra priorizar para a versão inicial do aplicativo.

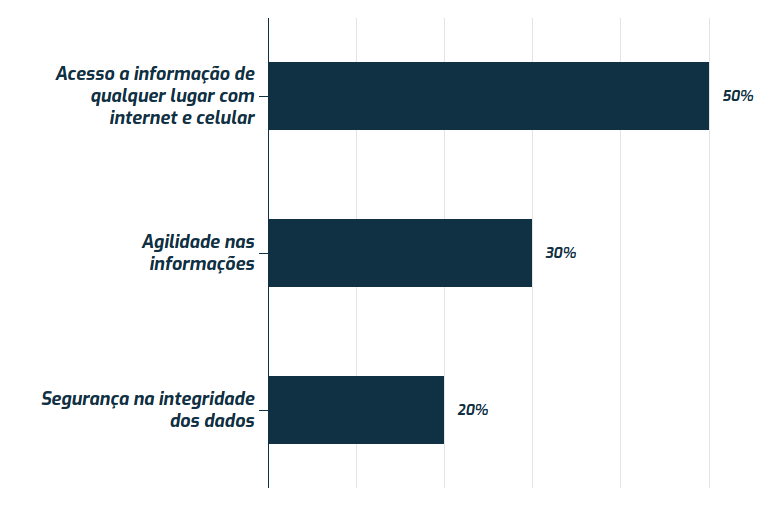
A ideia condensada permite que algumas inferências sejam feitas: O segundo resultado (item b, seção 3.2.1) homologa a ideia da construção de um aplicativo híbrido, visto que nessa amostra a parcela de sistemas operacionais utilizados está bem dividida. O terceiro ponto (item c, seção 3.2.1) homologa a ideia de utilizar um serviço *Web* para manipulação de dados, pois 100% dos participantes alegaram possuir acesso à internet em seus aparelhos móveis. O primeiro levantamento (item a, seção 3.2.1) indica uma adesão da ferramenta uma vez que a mesma já esteja inserida no mercado. Por fim, o último ponto (item e, seção 3.2.1) convalida um dos valores oferecidos pelo *App*, isto é, a obtenção de informação a respeito do crediário de maneira rápida, pois em média, com uma internet razoável, demora-se 30 segundos para obter-se o valor devido pelos clientes na plataforma.

3.2.2 Resultados gráficos

Uma vez que os dados tenham sido coletados e organizados, eles podem ser dispostos em gráficos, tabelas e diagramas. Abaixo, encontram-se gráficos originados a partir da resposta dos *stakeholders* questionados. Os gráficos estão dispostos no formato de Histograma - Um histograma, segundo Peterson (2016) apresenta a frequência dos dados através de eixos, apresentando visualmente as diferentes frequências e características.

Conforme demonstrado na figura 5, 50% dos questionados afirmam que o valor mais significativo gerado pelo aplicativo seria o acesso a informação de qualquer lugar com o auxílio de um *smartphone* e acesso à internet. Informação relevante que nos permite concluir que, para alcançar uma grande adesão, deve-se desenvolver aplicativos que consumam dados centralizados em um servidor de aplicação que faz uso da infraestrutura proveniente da rede mundial de computadores.

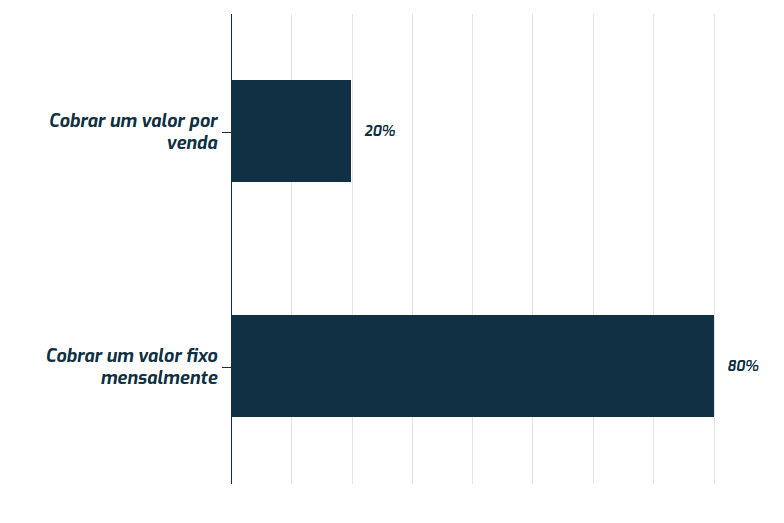
Figura 5 - Maior valor gerado com a utilização na opinião dos entrevistados

****

Fonte: Elaborado pelo autor.

Com o objetivo de entender a melhor prática de monetização do aplicativo, perguntou-se qual seria o melhor modelo de cobrança e de acordo com a figura 6 abaixo, 80% preferem a cobrança de um valor fixo mensal pelo uso da ferramenta.

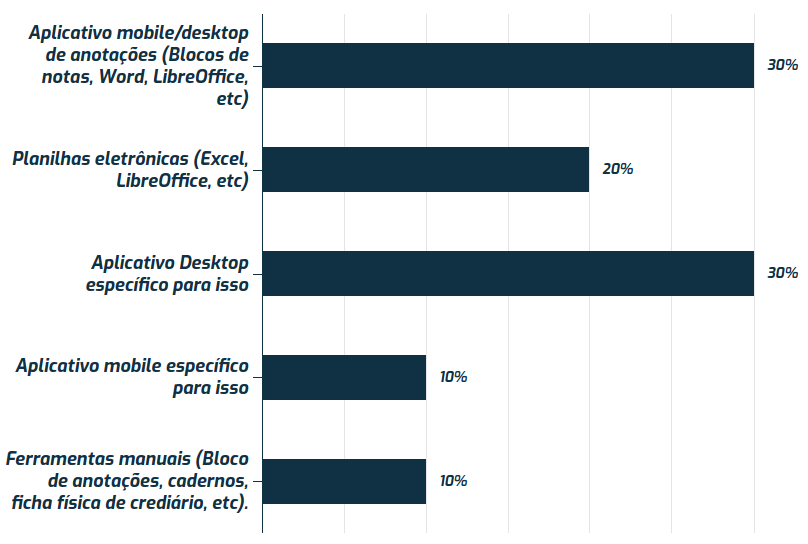
Figura 6 - Melhor maneira de pagar pelo EasyStore

****

Fonte: Elaborado pelo autor.

Atualmente, como pode ser visualizado na figura 7, 30% dos participantes disseram usar algum tipo de aplicativo *mobile*/*desktop* de anotações (Blocos de notas, Word, LibreOffice, etc) e 10% relatam usar ferramentas manuais para controle, tais como: cadernos e fichas físicas de crediário, por exemplo. Esses dados levam a refletir que existe espaço atualmente no mercado para ganhar o público que ainda executa o controle de crediário de uma forma manual ou pouco sistêmica.

Figura 7 - Formas de controles utilizadas atualmente

****

Fonte: Elaborado pelo autor.

**3.3 Modelo de negócios Canvas**

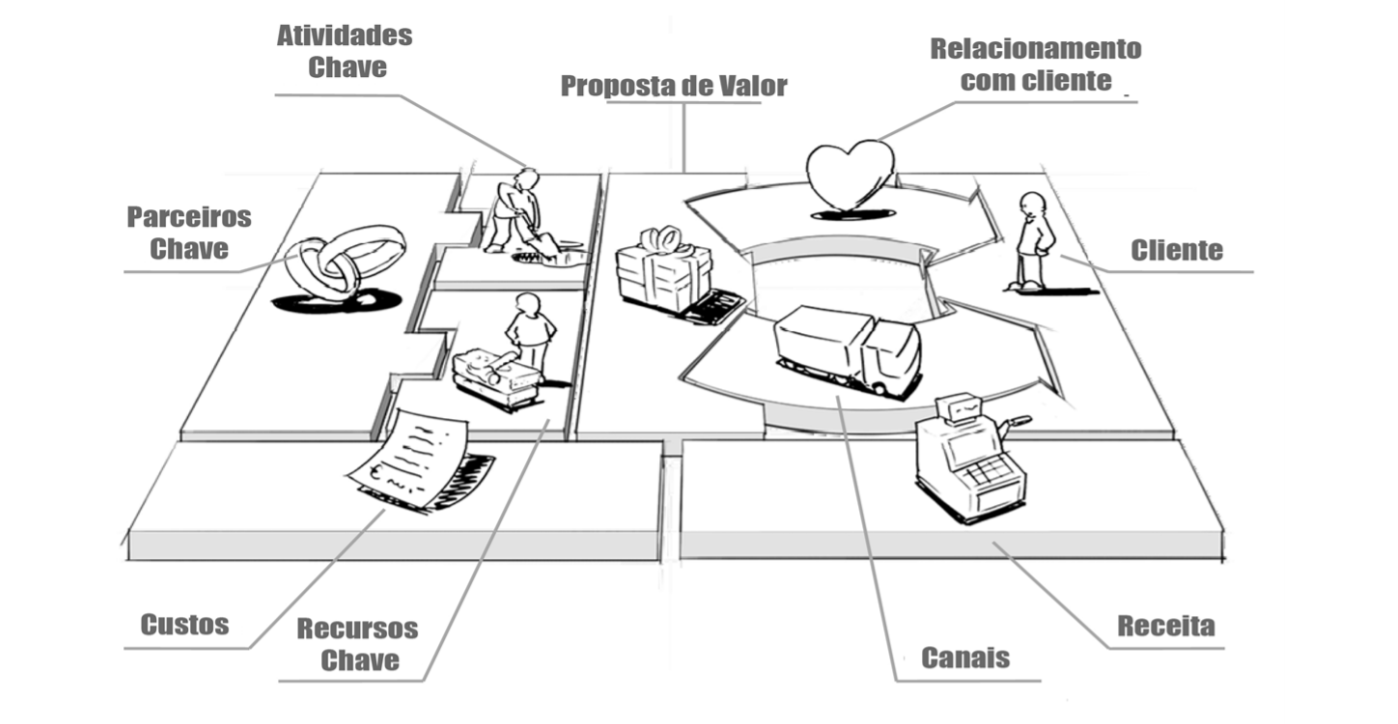
O conceito de modelo de negócios surgiu na literatura acadêmica em 1975, contudo, foi ganhar destaque na década de 90 devido ao surgimento da internet e do *e-commerce* (MORRIS; SCHINDEHUTTE; ALLEN, 2005). Os autores apontam que tal conceito não possui uma concordância no meio acadêmico no que diz respeito à definição, natureza, estrutura e evolução dos modelos de negócio.

Orofino (2011) ainda nos diz que a empresa precisa estabelecer um modelo adequado ao seu negócio, adaptado às exigências do mercado e que tente prever e gerenciar ao máximo as incertezas.

“Um modelo de negócio descreve a lógica de criação, entrega e captura de valor por parte de uma organização” (OSTERWALDER, 2011). O autor nos diz acreditar que um modelo de negócios pode ser melhor descrito com nove componentes básicos, que mostram a lógica de como uma organização pretende gerar valor. Os nove componentes cobrem as quatro áreas principais de um negócio: clientes, oferta, infraestrutura e viabilidade financeira. Tais componentes e suas respectivas descrições podem ser visualizados na ilustração 1.

Osterwalder (2011) nos apresenta esses nove componentes em formato de peças de um quebra-cabeças (conforme ilustração 1), isso é, cada componente sendo uma peça e quando encaixados, geram um todo. Esse todo é uma tabela que pode ser visualizada na ilustração 2. Ainda segundo o autor, tal tabela segue o funcionamento dos hemisférios cerebrais: o lado esquerdo processa a lógica enquanto o direito, emoções. Dessa forma, o quadro funciona assim: no lado direito mostra-se a eficiência do negócio enquanto que no lado esquerdo apresenta-se o valor do mesmo.

Ilustração 1 - Componentes da ferramenta Canvas

****

Fonte: Adaptado de Osterwalder (2011).

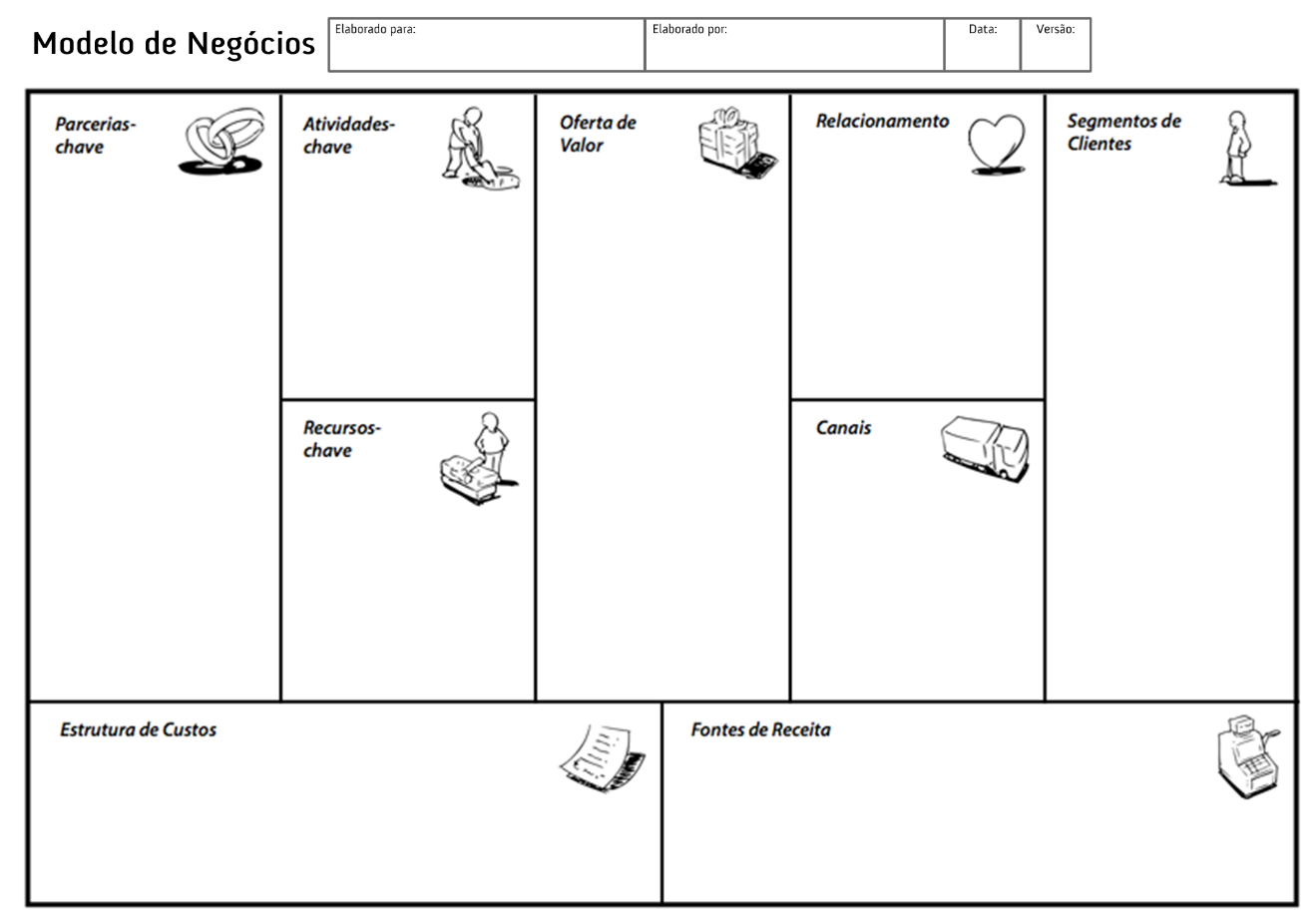
Quadro 1 - Os nove componentes da ferramenta Canvas

|  |  |
| --- | --- |
| **Componente** | **Descrição** |
| Segmento de clientes | Possível grupo de clientes ou organização a ser servido. |
| Oferta de Valor | Conjunto de benefícios oferecidos. |
| Canais | Como alcançar o cliente e entregar o valor proposto. |
| Relacionamento | O modo de se relacionar com o cliente. |
| Fontes de Receita | Mecanismos de precificação. |
| Recursos-chave | Recursos físicos, financeiros, intelectuais ou humanos. |
| Atividades-chave | Ações a serem realizadas. |
| Parcerias-chave | Alianças para otimizar o modelo, reduzir riscos ou adquirir recursos. |
| Estrutura de Custo | Custos envolvidos na operação do modelo de negócio. |

Fonte: Elaborado pelo autor.

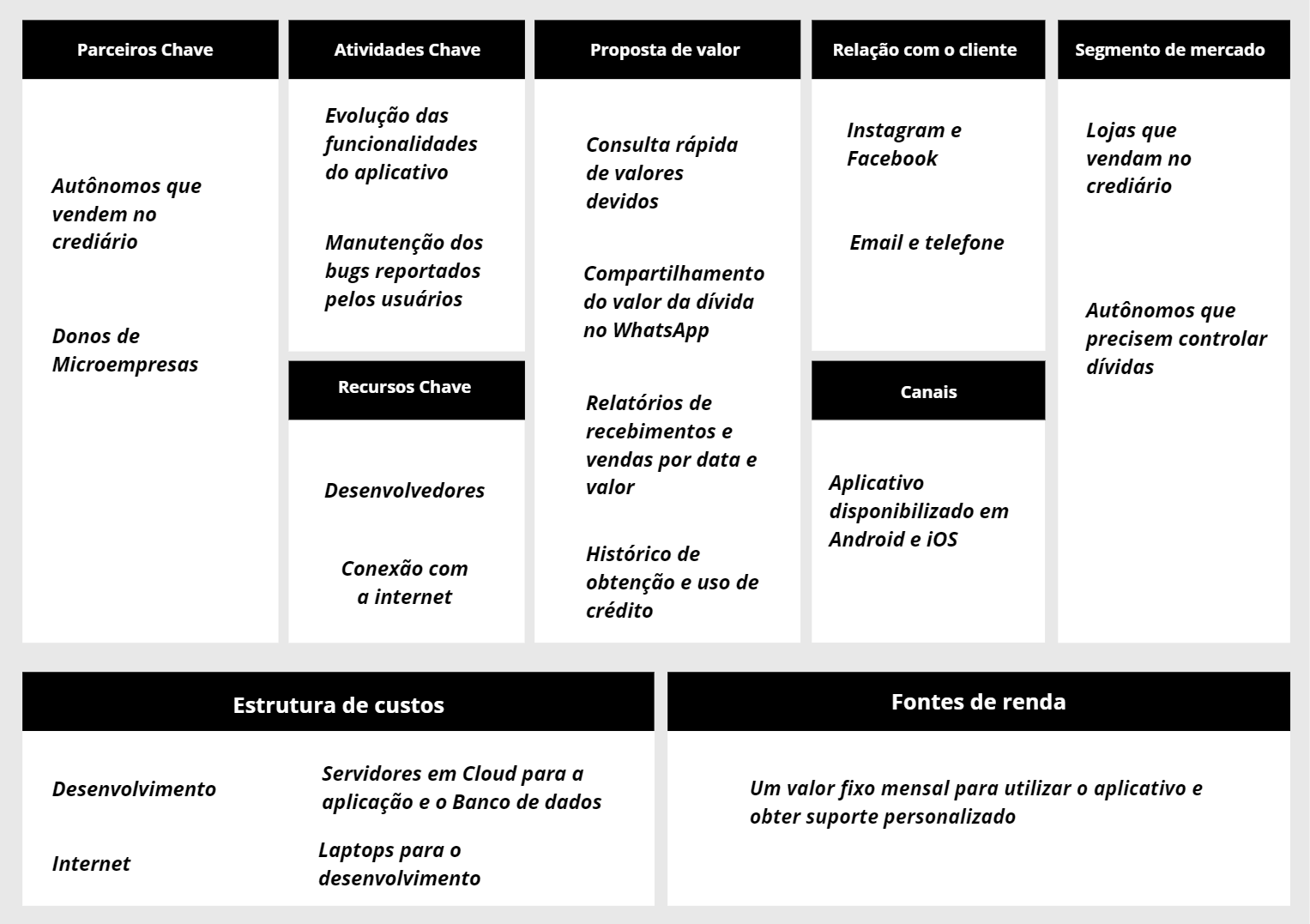
Nesta linha de raciocínio, o Canvas apresenta-se como uma ferramenta de modelo de negócios com uma fácil aplicabilidade. O Sebrae (2019) informa que com o “quadro”, a empresa reflete sobre cada função dentro dela e descobre o que precisa ser feito para conquistar clientes. Ainda segundo o Sebrae (2019), a análise e reflexão sobre o modelo possibilitam a elaboração de um plano bem estruturado e com maior potencial de sucesso, isto é, o modelo permite a criação de hipóteses a serem validadas e após isto se torna um recurso para a determinação do planejamento e a execução do negócio. Desta forma, propõe-se que tal modelo de negócios seja utilizado como uma ferramenta estratégica de apoio ao planejamento estratégico, auxiliando o dirigente a definir o seu negócio, buscando, assim, um melhor cenário dentro do ambiente em que está inserido.

Ilustração 2 - Modelo de negócio Canvas

Fonte: Adaptado de Osterwalder (2011).

3.4.1 Canvas EasyStore

Pensando no que foi falado na seção 3.3 e 3.4, o modelo de negócio do aplicativo EasyStore foi desenvolvido usando como base a ferramenta do Canvas. O questionário validou a ideia de construir um aplicativo híbrido que dependa da infraestrutura da internet para obter os dados, pois como visto nos resultados apresentados em 3.4, as pessoas utilizam em média mais Android, porém uma grande parcela do público também utiliza o iOS e todas as pessoas da amostra possuem acesso a internet. O resultado foi o quadro que pode ser visualizado na figura 8.

Figura 8 - Modelo de negócio da EasyStore

Fonte: Elaborado pelo autor

**3.4 SWOT**

Segundo Speth (2015), a análise ou também método SWOT é um acrônimo dos termos *Strengths* (Força), *Weaknesses* (Fraquezas), *Opportunities* (Oportunidades) e *Threats* (Ameaças). O autor pontua que tal método é usado em um contexto de análise decisória e facilita o desenvolvimento de planos estratégicos.

Azevedo e Costa (2001) pontuam que o SWOT é uma ferramenta utilizada na análise do ambiente interno e externo, objetivando o desenvolvimento de planos de médio e longo prazo e pode ser aplicado em design de projetos e produtos em geral.

Os pontos internos têm por finalidade colocar em evidência as deficiências e qualidades do contexto que está sendo analisado, segundo Oliveira (2001). Tais informações estão dentro do contexto e tem implicação imediata e específica na administração do mesmo.

Com o objetivo de expandir entendimento de negócio do aplicativo EasyStore, uma análise SWOT foi desenvolvida internamente.

* + 1. Oportunidades

### Uso de novas tecnologias;

### Alinhar o aplicativo ao mercado e com a possibilidade de concorrência;

### Grande oferta de mão de obra, pois existem muitas faculdades e universidades com o cunho de Ciência da Computação na região.

* + 1. Ameaças
       1. Nicho de mercado depende do cenário econômico e da inflação;
       2. O aplicativo ainda não possui uma equipe de Suporte 24 horas;
       3. Poucos desenvolvedores, o que gera uma baixa manutenibilidade.
    2. Pontos fortes

Software pronto, dependendo apenas de ajustes e adaptações;

Plataforma tecnologicamente moderna;

Portabilidade – banco de dados padrão SQL relacional;

Possibilidade de ampliar e sustentar novos serviços;

Possibilidade de atendimento a novos requisitos de negócio.

* + 1. Pontos fracos
       1. Baixa Governança de TI;
       2. Falta de cultura e ações de segurança da informações de maneira geral;
       3. Incrementos no software são feitos com uma certa latência visto que o aplicativo não possui muitos desenvolvedores.

**3.5 Considerações finais**

Este capítulo apresentou a investigação de viabilidade do projeto, com estratégias de coleta de dados que resultaram na elaboração da proposta de um aplicativo multiplataforma de controle de crediário que acessa dados remotos de qualquer lugar a partir de um dispositivo com conexão à internet.

O próximo capítulo abordará as etapas de modelagem e desenvolvimento da ferramenta, bem como discutirá as principais estratégias adotadas para atender às necessidades aqui levantadas.

# 

# 

# 

# 

4. MODELAGEM E DESENVOLVIMENTO DO APLICATIVO

**4.2 Projeto de interface**

O objetivo deste projeto foi construir uma *interface* minimalista que agregasse os recursos de usabilidade discutidos na seção 4.2.3. Para isso foram utilizadas as estratégias de design centrado no usuário descritas a seguir.

4.2.1 Análise hierárquica de tarefas

Para Preece, Rogers e Sharp (2013): “Uma das grande preocupações no design de interação é desenvolver produtos que sejam usáveis, o que genericamente significa produtos fáceis de aprender a usar, eficazes e que proporcionem ao usuário uma experiência agradável.”

Uma tarefa é qualquer parte do trabalho que precisa ser realizada. Toda tarefa pode ser definida em termo de seu(s) objetivo(s). Tarefas complexas são definidas em termos de objetivos e subobjetivos, num desdobramento hierárquico (BARBOSA e SILVA, 2010).

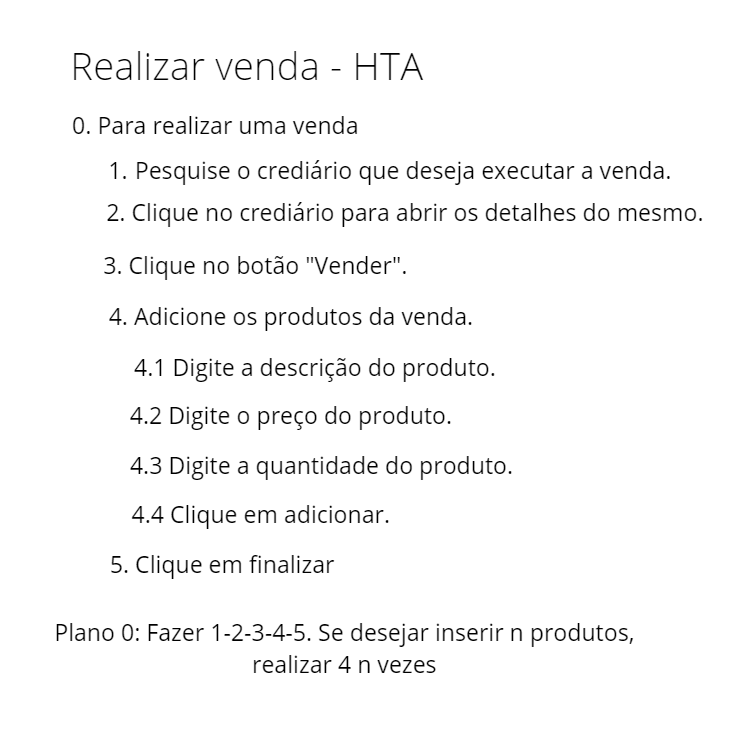
Como pontuam Annett e Duncan (1967), para ajudar no entendimento da interação criou-se a Análise Hierárquica de Tarefas (*HTA – Hierarchical Task Analysis*) na década de 1960. Tal análise objetiva entender as competências e habilidades exibidas em tarefas complexas e não repetitivas, bem como para auxiliar na identificação de problemas de desempenho. Os autores ainda pontuam que tal procedimento ajuda a relacionar o que as pessoas fazem (ou se recomenda que façam), por que o fazem, e quais as consequências caso não o façam corretamente. Ela se baseia em psicologia funcional, e não comportamental, como eram as abordagens da época em que foi criada.

Considerando as três principais funcionalidades definidas no escopo do *EasyStore*, foram elaboradas as análises de tarefas que se seguem.

4.2.1.1 Fluxo de venda

Permite que o usuário registre uma venda com seus respectivos produtos. Tal tarefa culmina com o incremento do valor devido pelo cliente na ficha.

Figura 9 - Análise hierárquica do fluxo de venda

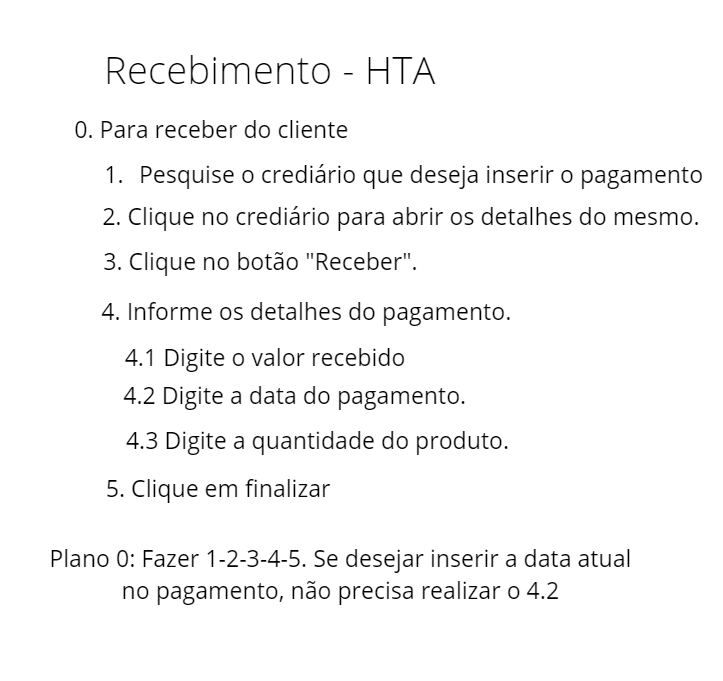
****

Fonte: Elaborado pelo autor.

4.2.1.2 Receber

Permite que o usuário registre o recebimento de uma quantia monetária. Tal quantia é subtraída do valor devido no crediário do cliente.

Figura 10 - Análise hierárquica do fluxo de recebimento

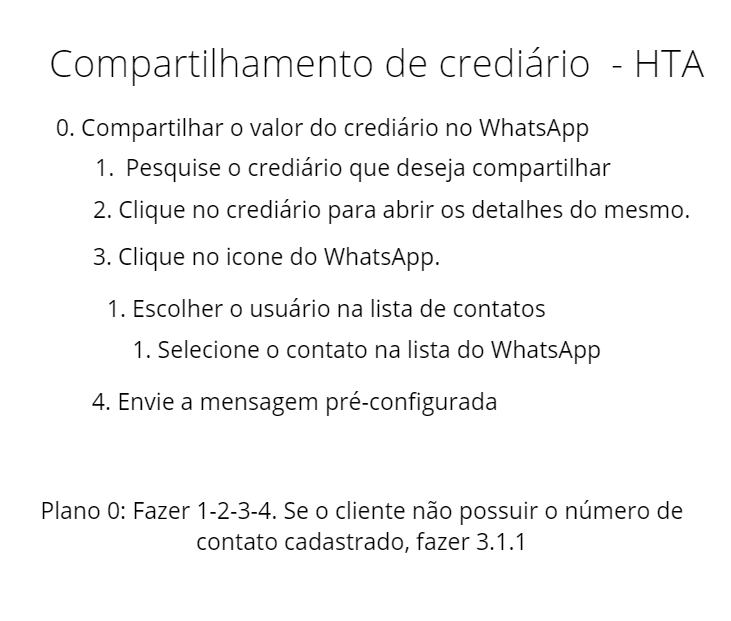
****

Fonte: Elaborado pelo autor.

4.2.1.3 Compartilhar valor do crediário no WhatsApp

Geralmente o cliente pergunta para o lojista o valor devido, visto isso, tal fluxo permite que o usuário compartilhe o valor devido com uma mensagem amigável ao mesmo de forma rápida e prática, diminuindo significamente o tempo de resposta ao cliente.

Figura 11 - Análise hierárquica do fluxo de compartilhamento de crediário

****

Fonte: Elaborado pelo autor.

4.2.2 Modelos conceituais

Metáforas são parte integrante de nosso pensamento e linguagem. Aparecendo não somente na poesia ou na literatura, como a maioria acredita, mas também em nossa linguagem cotidiana (ROCHA e BARANAUSKAS, 2010). As expressões “Gastar dinheiro” e “trânsito engarrafado” são alguns exemplos comuns. As metáforas funcionam como modelos naturais, permitindo o uso da familiaridade de objetos concretos e experiências para dar estrutura a conceitos mais abstratos.

Da mesma forma que metáforas invisíveis permeiam nossa linguagem cotidiana, elas o fazem nas interfaces que usamos e projetamos (Erickson, 1990). Por exemplo, um usuário quando arrasta um documento de um diretório para outro nos sistemas gerenciadores de arquivos de ambientes Windows, ele efetivamente acredita que está mudando o documento de lugar, contudo o que efetivamente ocorre é que o apontador para o arquivo mudou (apontador também é uma metáfora).

De acordo com Ausubel (1982), utilizar-se de uma analogia ou metáfora é considerar as idéias já existentes na estrutura cognitiva do sujeito possibilitando que sirvam como âncora para introdução de novos conceitos. O autor afirma que quando estas idéias se relacionam, ocorre uma aprendizagem significativa.

Com o objetivo de acelerar a aprendizagem do usuário no aplicativo *EasyStore*, bem como eliminar qualquer distanciamento semântico entre a regra de negócio e o universo em que o usuário está inserido, o desenvolvimento fez uso de metáforas no aplicativo, como seguem:

1. Uma das subtarefas que o usuário deverá realizar para alcançar um objetivo no aplicativo será a de autenticação, na qual será adotada a metáfora “Entrar” na tecla onde o usuário, após informar o login e senha, deverá pressionar para conseguir acessar as próximas subtarefas do sistema. A escolha dessa metáfora tem o objetivo de fazer com que o usuário se sinta à vontade e familiarizado para poder entrar no ambiente e obter as funcionalidades desejadas;
2. Outra subtarefa realizada pelo usuário é a de pesquisa de crediário, nessa subtarefa será adotada a metáfora “Buscar”, o objetivo de tal metáfora é causar a informalidade no sistema, deixando a linguagem mais coloquial, trazendo o aplicativo para a realidade do usuário;
3. O aplicativo utiliza muitos ícones indicando a ação do respectivo botão, como por exemplo, na tela de detalhamento do crediário, caso o usuário queira ver as vendas registradas para aquele cliente, existe um ícone cuja imagem é um carrinho de compra, metáfora essa indicando a ação de compra. Muitos outros ícones indicando a ação semântica do botão ou campo são utilizados durante toda a experiência do usuário com o aplicativo.

O estilo de interação adotado é o de instrução, ou seja, o usuário realiza comandos. Os tipos de interface envolvidos são os de entrada de dados via texto por preenchimento de formulários.

4.2.2.1 Tipos de *interface*

Para Domingues (2001), a *interface* é um espaço tradutor, isto é, uma face intermediária entre a linguagem nativa da máquina e as linguagens naturais do homem. Fazendo uma analogia, poderíamos dizer que a interface teria a função de um intérprete bilíngue: O usuário clica em um ícone, por exemplo, e o intérprete traduz para o sistema operacional o que ele deve fazer. Em suma a interface é todo o conjunto de recursos - janelas, ícones, barras de rolagem, barras de ferramentas e links, por exemplo – que permitem a interação entre o homem e o computador.

A interface escolhida foi a gráfica, com as seguintes características:

1. Exibir mensagens relativas às ações executadas no sistema usando uma semântica de cores. Exemplo: Se a mensagem for de erro sistêmico, um alerta com as cores padrões do componente do dispositivo é exibido, precisando de um *click* no botão para ser fechado. Caso o erro seja por parte do usuário (Informar um CPF inválido, por exemplo) a mensagem é um *toast* vermelho que é exibido durante três segundos. Nas mensagens de sucesso, um *toast* verde é exibido também durante três segundos;
2. Navegação intuitiva e simples. A navegação das telas não é muito aprofundada, isto é, o usuário não empilhará mais do que quatro telas nos fluxos do aplicativo. Tal característica auxilia nos botões de voltar, pois ao clicar muitas vezes em voltar o usuário sente-se perdido;
3. O teclado torna-se numérico quando o campo é relativo a algum valor monetário, não permitindo assim, que o usuário fique confuso ou digite letras;
4. O usuário tem acesso a tudo que digita, inclusive a senha na tela de credenciais, quando o mesmo seleciona o ícone de visualização de senha digitada;

4.2.3 Princípios de usabilidade

### Para favorecer a usabilidade do aplicativo, alguns pontos foram considerados, são eles:

1. Conhecer a necessidade dos usuários finais por meio de questionário de pesquisa;
2. Momentos de uso do celular com apenas uma mão, aplicando interface minimalista;
3. Poucos cliques para atingir o objetivo, ou seja, possuindo propósito bem definido;
4. Suporte a diferentes tipos de telas, usando dispositivos de vários modelos e marcas;
5. Inclusão do usuário no processo de desenvolvimento, disponibilizando funcionalidades para serem testadas durante a homologação do projeto.

Em 1990, Jakob Nielsen e Rolf Molich definiram dez princípios gerais de design de interface, os quais são chamados de heurísticas de Nielsen e focam no desenho de telas com uma boa navegação, interação e experiência (NIELSEN, 1993).

Abaixo estão listadas algumas das dez heurísticas escolhidas bem como os seus respectivos requisitos. Os princípios abaixo foram escolhidos para o aplicativo pois os mesmos auxiliam no uso de tarefas e ajudam na aprendizagem dos fluxos de negócio. Por ser um aplicativo de controle de crédito, é muito importante garantir a integridade das informações bem como exibi-las de maneira simples e menos confusa possível.

1. Equivalência entre o sistema e o mundo real: Utilizar ícones relacionados ao mundo real, por exemplo: Uma chave no campo de senha;
2. Prevenção de erros: Ações drásticas como deletar e sair devem ser confirmadas antes de serem executadas, evitando assim erros;
3. Visibilidade do estado do sistema: Para que o usuário sempre saiba o que está acontecendo em determinado momento (Exemplo: Um loading quando uma tela estiver iniciando);
4. Reconhecimento de erros:Quando algo sair errado, deixe claro para o usuário o que está acontecendo e o que ele deve fazer para resolver;
5. Permitir o cancelamento de ações:Oferecer ao usuário sempre a opção de “retornar” à tela anterior, cancelar a atividade sendo executada (Desistir de cadastrar algo no meio do preenchimento das informações, por exemplo), porque é fundamental que o usuário tenha total controle das ações que estiver executando.

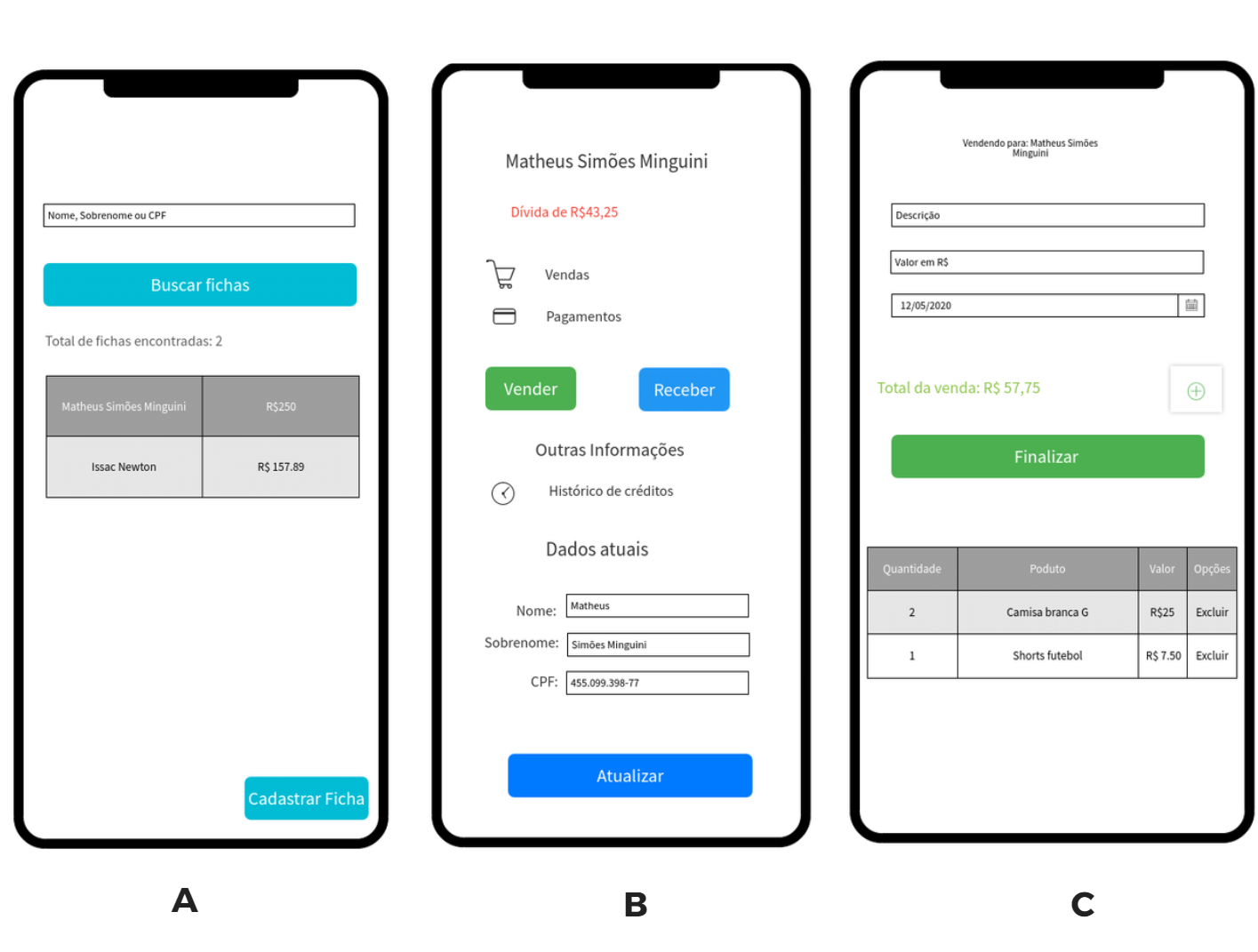
4.2.4 Prototipação

Com a intenção de elaborar e validar as telas que fariam parte do aplicativo, protótipos dos fluxos principais foram desenvolvidos utilizando uma ferramenta de *Wireframe*.

O protótipo gerado foi disponibilizado através de um link e o primeiro contato com os usuários potenciais de uma loja de roupas da cidade de Taquaritinga, no estado de São Paulo, foi realizado. Os usuários interagiram livremente, sem auxílio nenhum e explicando verbalmente o que estavam pensando a respeito de quão simples e intuitivo as telas bem como os fluxos estavam.

Esse processo de testes permitiu que cenários fossem previstos, sugestões de *interface* fossem apontadas e fluxos de tarefas revistos, como por exemplo ao cadastrar uma ficha, já abrir a tela de detalhamento da mesma. A seguir, está a figura dos protótipos do de venda.

Figura 12 - Prototipação do fluxo de entrada no aplicativo

****

Fonte: Elaborado pelo autor

Na figura apresentada acima, os pontos possuem uma marcação de sequenciamento, visando mostrar o passo a passo da tarefa, sendo eles:

1. Pesquisa-se o crediário e seleciona-se o mesmo;
2. Na tela de detalhe, clica-se em vender;
3. Informa-se os produtos da venda e seus detalhes como: descrição, valor,quantidade. Clica-se em finalizar para encerrar o fluxo.

**4.3 Projeto de Software**

O código fonte da ferramenta foi desenvolvido em dois repositórios distintos, sendo o primeiro para a aplicação instalada no celular e o segundo para o desenvolvimento da aplicação que ficará no servidor atendendo requisições HTTP.

Caso queira, o leitor pode acessar tais repositórios nos links abaixo, visto que estão disponibilizados publicamente.

Repositório aplicativo: *https://github.com/MatheusMinguiniPostGraduation/storeRecordApp*

Repositório aplicação: *https://github.com/MatheusMinguiniPostGraduation/storeRecord.git*

Nos capítulos a seguir, explica-se qual o modelo de arquitetura escolhido, o porquê dessa escolha, bem como as camadas que juntamente com as suas respectivas tecnologias compõem a aplicação desenvolvida. Ao final, na seção 4.4 mostra-se e explica-se a lógica de negócio dos fluxos do *app*.

4.3.1 Arquitetura e design do *Software*

Desenhar sistemas é uma tarefa difícil. E, ainda fazer com que sejam escaláveis e performáticos, mantendo uma alta qualidade interna e externa, é um desafio (SILVEIRA, et al., 2011). O autor ainda pontua que a arquitetura pode ser definida como uma possível interpretação da implementação do sistema. Sendo a análise de como diferentes partes dos sistemas afetam-se entre si.

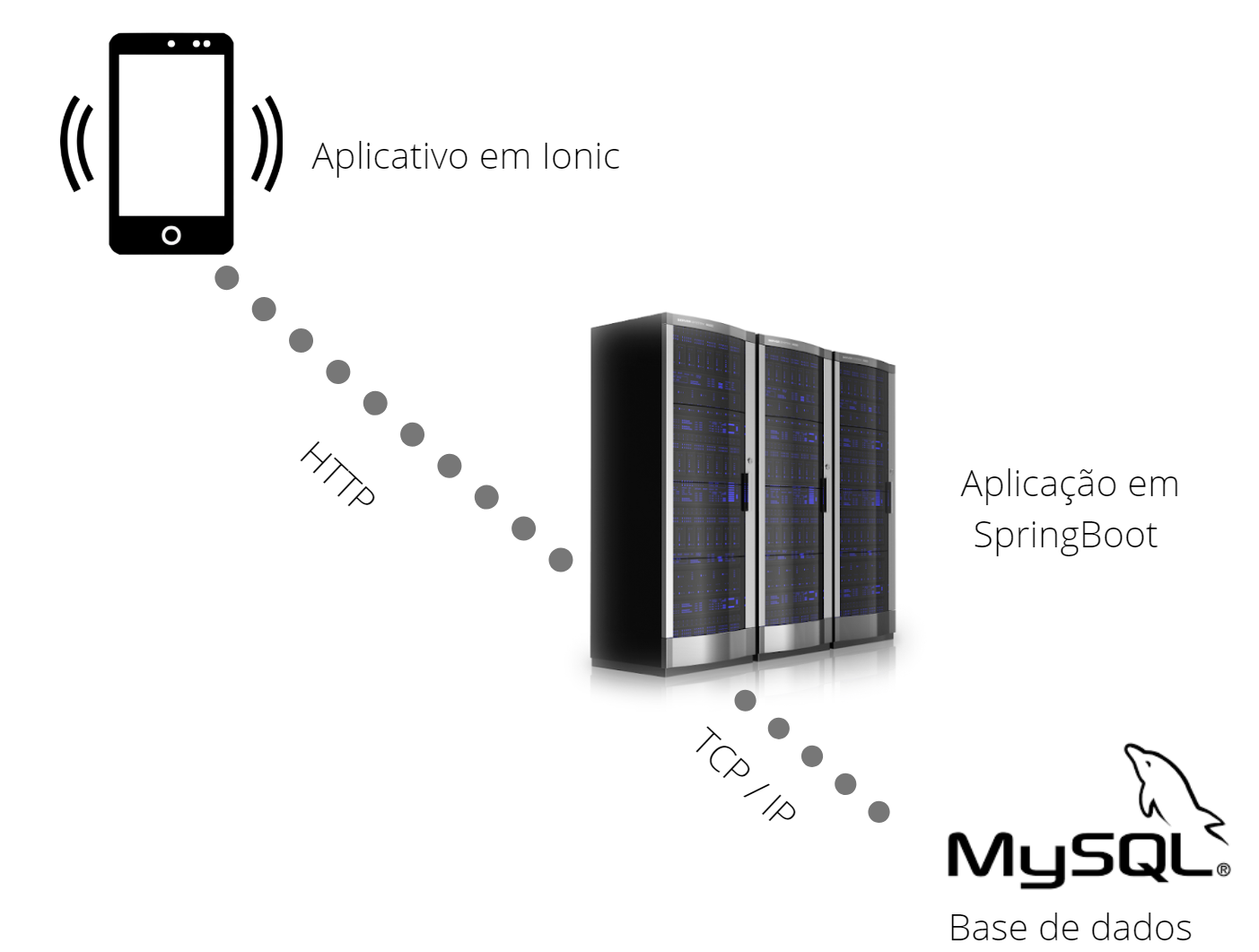
Segundo Fowler (2003), uma boa arquitetura ajuda no crescimento e futura modificação da aplicação, diminuindo o custo de manutenção a longo prazo.

Pensando nisso, com a definição do escopo da ferramenta tendo como base pesquisas com usuário potenciais (terceiro capítulo), o próximo passo foi a escolha da arquitetura do modelo de software. Para tanto, o primeiro passo desta etapa foi a modelagem das entidades de negócio (*Beans*) utilizando um modelo de diagrama de entidade e relacionamento, tal diagrama pode ser visto no anexo B.

Com o entendimento e visualização do domínio estrutural da aplicação, optou-se pelo desenvolvimento de dois sistemas com escopos bem definidos, sendo um projeto para o aplicativo instalável no dispositivo do usuário e o outro, um serviço exposto na *web*, seguindo o modelo SOA (*Service-oriented architecture*).

Segundo Saudate (2014), SOA é um conceito, não uma tecnologia. O modelo define diretrizes para o desenvolvimento de sistemas com funcionalidades expostas na *web* a fim de integrar sistemas com domínios diferentes, o que implica em reutilização de lógica de negócio e desacoplamento de linguagens de programação.

Figura 13 - Escopo sistêmico da aplicação

****

Fonte: Elaborado pelo autor.

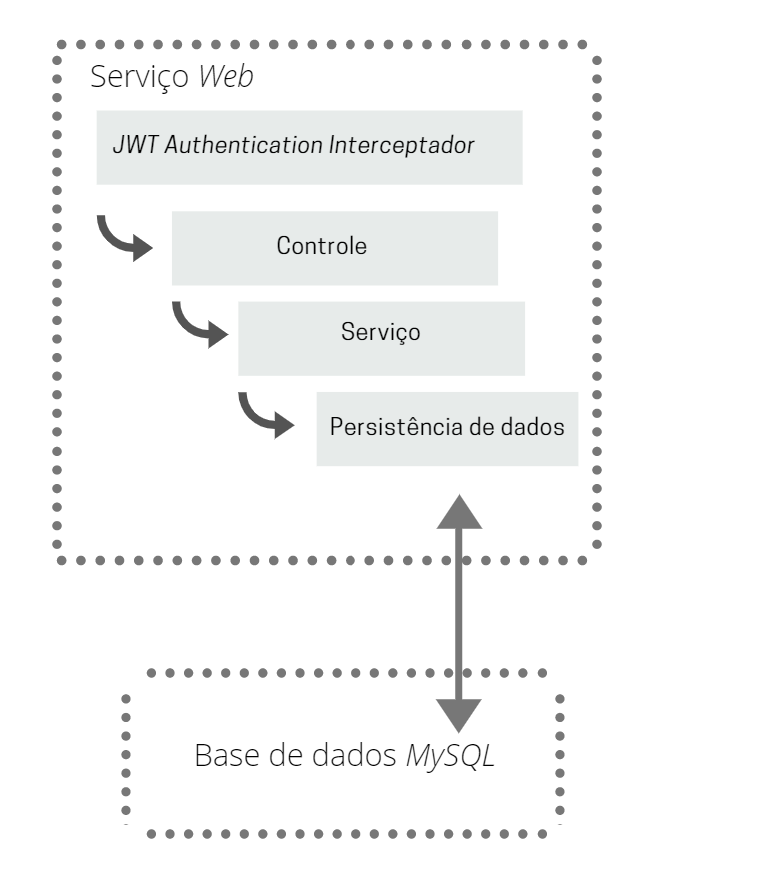
Em suma, a aplicação instalada no dispositivo do usuário não persiste dados localmente, mas consome dados em um serviço *web* através de requisições HTTP (seção 2.2.2.1). Tal serviço foi construído utilizando-se do padrão arquitetural REST (seção 2.2.3.4). A seguir, apresenta-se as camadas de arquitetura da ferramenta construída do lado do servidor.

4.3.2 Camadas da aplicação *Backend*

Silveira et al. (2012) explica que ao referir-se a *layer*, normalmente pensa-se em separações lógicas, isto é, em como organizar o código da aplicação de maneira a diminuir o acoplamento e facilitar mudanças.

Abaixo, na figura 14, pode ser verificado visualmente como a aplicação está construída. A seguir, o presente trabalho aborda as camadas unitariamente.

Figura 14 - Camadas da arquitetura do serviço web

****

Fonte: Elaborado pelo autor.

4.3.2.1 Camada de controle

A camada de controle é responsável por expor os endereços que podem ser acessados bem como o verbo HTTP que irá permitir o acesso do mesmo. Nessa camada, mapeia-se o recurso bem como os dados passados para o mesmo.

Visto que é a porta de entrada, é a partir da camada de controle que as demais camadas são acessadas.

4.3.2.2 Camada de serviço

A camada de serviço é a responsável por toda a lógica negocial, a mesma é a única com acesso a camada de persistência de dados justamente por possuir a *expertise* necessária das regras e especificações das atividades do nicho de mercado do aplicativo.

Apresenta-se, no anexo C, um exemplo de código da camada de serviço do fluxo de cadastro de crediário. Deve notar-se que antes de persistir a nova ficha, verifica-se se a existência de uma ficha cujo cliente possua o mesmo nome, sobrenome e quando informado, CPF que está tentando-se cadastrar no momento.

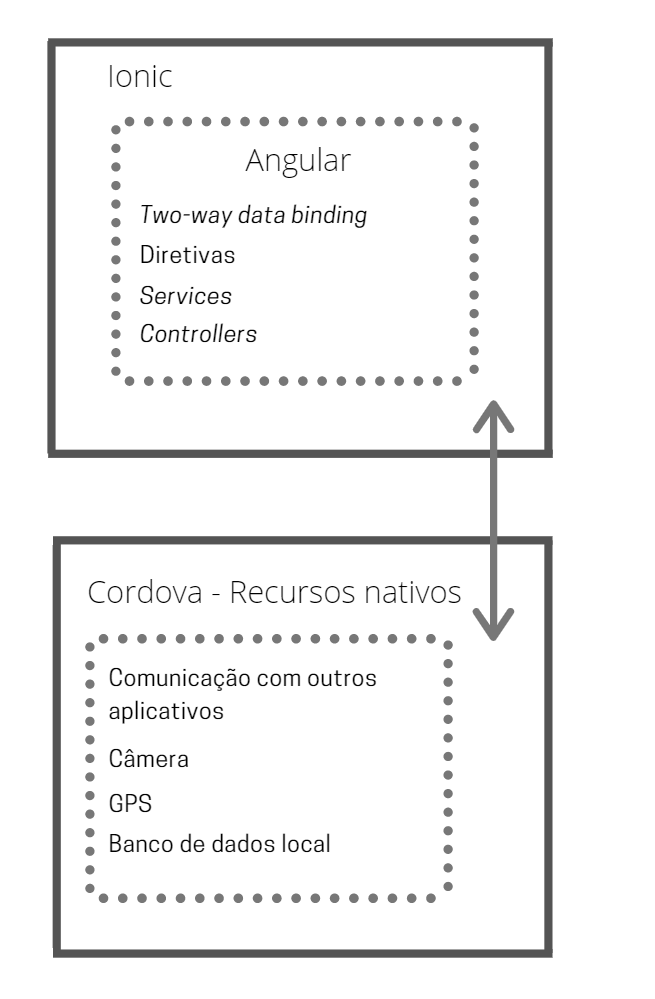
4.3.2.3 Camada de persistência

A camada de persistência fica responsável por conectar e executar comandos DML (*Data Manipulation Language*) no banco de dados MySQL, isto é: *SELECT*, *INSERT*, *UPDATE* e *DELETE*. No anexo C apresenta-se um exemplo de uma consulta na camada de dados do crediário.

4.3.3 Desenvolvimento do aplicativo

O aplicativo foi desenvolvido utilizando a tecnologia do Ionic, tanto o Ionic quanto às suas respectivas tecnologias foram explicadas no capítulo de Fundamentação Teórica - Seção 2 deste trabalho. O diagrama de camadas abaixo apresenta uma visualização clara de como funciona o *App,* bem como onde o agrupamento das tecnologias por ele utilizadas.

Figura 15 - Divisão de camadas do aplicativo.

****

Fonte: Elaborado pelo autor.

**4.4 Apresentação do aplicativo**

Nesta seção serão explicados os elementos de interação e funcionalidades da aplicação desenvolvida.

Todos os fluxos nessa seção explicados, possuem imagens da implementação das suas respectivas telas, tais imagens podem ser encontradas no anexo C.

4.4.1 Fluxo de autenticação

Como forma de identificar e validar os usuários do aplicativo, foi escolhido o método de autenticação utilizando dois dados sensíveis: usuário e a senha.

Tais dados são cadastrados administrador sistêmico através de uma *interface* online na *web*. A tecnologia de *backend* para garantir a segurança de acesso aos dados na aplicação foi o *Spring Security*. Segundo Cosmina (2017), a tecnologia provê um suporte compacto para autenticação. O autor ainda pontua que o processo de autenticação é importante pois é como verifica-se a identidade do cliente que está tentando acesso a um recurso particular.

O usuário informa as credenciais o login e os envia para autenticação. Após isso, acontece o seguinte algoritmo:

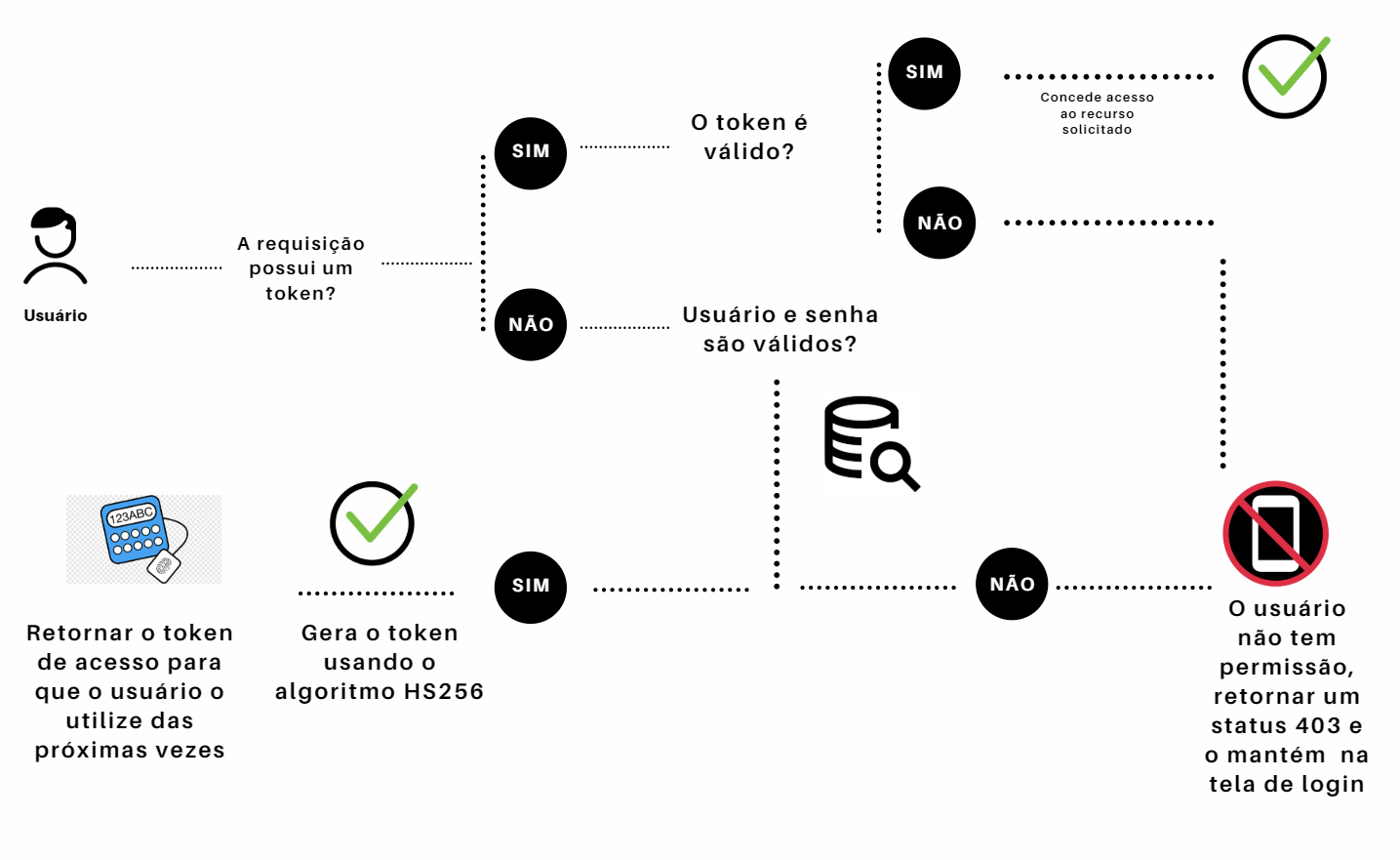
1. Visto que o servidor de aplicação é *stateless*, isto é, não guarda o antigo status do usuário, um token precisa ser enviado em todas as requisições como forma de autenticação e permissão de acesso ao recurso solicitado;
2. Um *Interceptor* (Classe interceptadora que captura todas as requisições que chegam à API) foi desenvolvida. Ao receber uma requisição HTTP, a instância de controle verifica se existe um *token* no atributo *Authorizer* no cabeçalho da requisição.
3. Caso exista, então a ferramenta concede acesso ao recurso requisitado;
4. Caso não exista um token, então a aplicação verifica no *body* da requisição se as credenciais para autenticação (usuário e senha) foram fornecidas;

b.1) Se as informações forem válidas, então um token é gerado utilizando o algoritmo HS256, retornando o token a fim de ser utilizado pelo usuário nos próximos 60 minutos;

b.2) Se as informações não forem válidas ou nada tenha sido informado, então retorna-se um estado 403 (seção 2.2.2.1) de forma a expressar semanticamente ao requisitante que o mesmo não está autorizado a acessar os dados;

A sequência de eventos pode ser visualizada, em formato de fluxograma, na figura abaixo.

Figura 16 - Fluxo de entrada e autenticação no aplicativo.

****

Fonte: Elaborado pelo autor.

4.4.2 Inserção de crediário

O usuário, ao acessar o sistema, seleciona o botão “Fichas” no menu inferior, posteriormente clicando no *floating button* com o símbolo ‘+’ - metáfora remetendo a adição de dados. Ao fazer isso, o mesmo é encaminhado para o formulário com os dados do cliente.

Ao preencher as informações requisitadas no formulário e apertar o botão “Salvar” o sistema então valida se o CPF foi informado (Visto que não é um campo obrigatório), caso tenha sido informado e esteja certo, o sistema analisa se não existe um outro usuário com o mesmo nome e sobrenome, a fim de evitar a duplicidade de crediário. Caso o CPF esteja certo e o nome não exista para aquele usuário, cadastra-se então o mesmo.

4.4.3 Pesquisar cliente

Caso o lojista queira pesquisar um crediário já existente, o mesmo acessa a tela de “Fichas” e no campo de filtro, opta por informar o nome, sobrenome ou o CPF do cliente cujos dados queiram ser encontrados.

Caso o aplicativo não encontre a conta com os filtros informados, então exibe-se uma imagem de recurso não encontrado, conforme princípio de usabilidade abordado na seção 4.2.3.

No entanto, quando o cliente é encontrado, exibe-se na listagem de retorno o total devido bem como o nome completo do mesmo.

4.4.4 Vender

O usuário pesquisa o crediário e após selecionar o desejado é então enviado para a tela de detalhes do crediário; nessa tela, existe um botão “Vender” e quando clicado, faz com que o lojista acesse a tela de ponto de venda.

Então, informa-se a descrição, o preço unitário e a quantidade que está sendo comprada do produto.

Validações são feitas no momento de adicionar o produto na lista de compra: A quantidade deve ser maior que zero, a descrição precisaestar preenchida (espaços em branco são validados) e o valor unitário deve ser entre R$1,00 e R$2.000,00.

No caso de validadas e atendidas todas as restrições, a peça é adicionada. O usuário pode visualizar os produtos através de uma tabela abaixo do botão “Adicionar Produto”.

Caso opte por remover o artefato, então, clica-se no botão de remoção (ícone de uma lixeira, seguindo o modelo de metáforas apresentado na seção 4.2.2) que o mesmo será removido e o total da venda atualizado.

Após manipular e decidir o que será comprado, o usuário aperta o botão finalizar e o sistema persiste a venda com a data e hora atual.

Para efeitos de auditoria, a venda possui o usuário que a realizou - No caso, o usuário logado que cadastrou a venda. O fluxo termina com o usuário sendo levado para a tela de detalhes do crediário com o total devido já atualizado e uma mensagem de confirmação da operação que é exibida durante 5 segundos, desaparecendo posteriormente.

4.4.5 Receber

O lojista pesquisa o crediário desejado, selecionando o mesmo. Então, na tela seguinte pressiona-se o botão “Receber”. Assim, na tela subsequente, o usuário informa o valor que está sendo recebido, a data que foi recebido e método de pagamento: dinheiro, cartão de crédito ou débito.

Algumas verificações são feitas ao finalizar o recebimento, são elas: A data de recebimento não pode ser futura, o valor recebido precisa ser positivo e menor do que o valor devido - Um pagamento não pode gerar um crédito.

Para efeitos de auditoria, o ato de receber persiste o usuário que a efetivou no sistema - No caso, o usuário logado que cadastrou a venda. O fluxo termina com o usuário sendo levado para a tela de detalhes do crediário com o total devido já atualizado e uma mensagem de confirmação da operação que é exibida durante 5 segundos, desaparecendo posteriormente.

4.4.6 Detalhar a venda

O usuário pesquisa o crediário cuja venda queira ser encontrada. Ao entrar na tela, o usuário clica no ícone de "Vendas”.

Uma vez que o lojista esteja na tela de pesquisa de venda, carrega-se inicialmente todas as vendas realizadas para o cliente daquele crediário. Caso queira, o usuário pode filtrar a venda por data e por valor. Funciona assim: Coloca-se a data inicial e a final da venda, exemplo: Vendas de 02/03/2020 a 15/04/2020 e o valor da aquisição segue o mesmo pensamento, ou seja, vendas de R$15,70 a R$45,68, por exemplo.

Ao encontrar e selecionar a venda desejada, a tela de detalhamento da venda é exibida; nesta tela visualiza-se os dados gerais e os produtos comprados adquiridos.

4.4.7 Excluir a venda

Caso haja a necessidade, o lojista pode excluir uma venda, a mesma é excluída logicamente no banco de dados.

Para tanto, acessa-se o detalhamento da venda, conforme já explicado acima e na tela de dados gerais, pressiona-se o botão *"Sim, quero cancelar"*. Então, um aviso é exibido solicitando a atenção do usuário bem como a confirmação do mesmo.

Após a confirmação da operação, a venda é excluída. Caso o usuário acesse os detalhes daquela venda posteriormente, ao entrar na tela já verá a mensagem em vermelho "Venda excluída" acompanhada de quem excluiu, a data e hora, conforme princípio de usabilidade abordado na seção 4.2.3.

4.4.8 Pesquisar o pagamento

O usuário pesquisa o crediário cujo pagamento queira ser encontrado. Ao entrar na tela, o usuário clica no ícone de *"Pagamentos”*.

Então carrega-se inicialmente todos os pagamentos realizados pelo cliente daquele crediário.

Pensando em melhorar a usabilidade, a funcionalidade de filtro por data e valor foi desenvolvida: Informa-se uma data inicial e final, assim como um valor inicial e final. No caso em que o lojista queira, por exemplo, saber pagamentos de 02/03/2020 a 15/04/2020 e/ou valores de R$15,70 a R$45,68.

4.4.9 Excluir o pagamento

O usuário pode excluir um pagamento, o mesmo é excluído fisicamente do banco de dados. Para excluir, o usuário acessa a busca de pagamento, conforme descrito na seção 4.4.8e na listagem, o usuário escolhe o pagamento a ser removido clicando sobre o mesmo e arrastando para a esquerda.

Ao fazer isso, o lojista visualiza botão de exclusão com o ícone de uma lixeira, seguindo o princípio de metáforas abordado na seção 4.2.2. Então,quando clicado, o botão exibe um aviso solicitando a atenção do usuário bem como a confirmação do mesmo.

Após a confirmação da operação, o pagamento é excluído. Após a remoção, o pagamento não será mais encontrado pois o mesmo não existirá na base de dados.

4.4.10 Geração de crédito

Caso o cliente cancele uma compra após já ter pago a mesma, o dinheiro não é devolvido, então esse pagamento entra como crédito. Exemplo: No dia 17/03/2020 o cliente realizou uma compra no valor de R$ 50,00 e deu uma entrada de R$ 20,00, passando a dever R$ 30,00. No dia 18/03/2020, o cliente retorna a loja solicitando a devolução da roupa. Os R$ 20,00 não são devolvidos mas sim, computados como crédito na ficha do cliente.

O usuário do aplicativo acessa o fluxo de exclusão da venda (Seção 4.4.7) e cancela a mesma, após fazer isso, a diferença é automaticamente colocada como crédito na ficha, deixando inclusive o número disposto em verde, indicando assim um valor positivo - Seguindo o princípio de metáforas abordado na seção 4.2.2.

4.4.11 Histórico de crédito

Todas as vezes que um crédito é gerado ou utilizado, tais transações são persistidas a fim de guardar um histórico seguindo um modelo de extrato para futuros esclarecimentos ao cliente.

Como no caso em que o cliente possui um crédito de R$ 25,00 (vinte e cinco reais) e efetua uma compra de R$ 20,00 (vinte reais). Então um registro de uso de R$ 20,00 (vinte reais) é persistido na base. Tal extrato pode ser visualizado ao pressionar-se o botão “Histórico” na tela do detalhamento do crediário. Na tela de histórico, o lojista pode visualizar o extrato dos valores debitados e creditados para o cliente em questão.

4.4.12 Compartilhamento de crediário por WhatsApp

Caso o cliente solicite remotamente um resumo atual da ficha, então o lojista encontra o detalhamento do crediário e pressiona um botão de compartilhamento na tela. Fazendo isso, o extrato é enviado no WhatsApp do solicitante.

Para que o fluxo ficasse mais performático, adotou-se a integração do número cadastrado no aplicativo com o número a ser enviado no WhatsApp, isto é, caso o cliente em questão possua o número de contato cadastrado no *EasyStore*, ao pressionar-se o botão de compartilhamento, a mensagem de resumo de crediário é então enviada automaticamente, não sendo necessária outras ações. Por outro lado, quando o solicitante não possuir um número de contato cadastrado, então, ao pressionar-se o botão de compartilhamento, o lojista precisa selecionar o contato no WhatsApp.

4.4.13 Geração de relatórios

Caso os administradores da loja queiram acessar os relatórios disponíveis no aplicativo, basta dirigir-se à tela inicial do aplicativo e no menu inferior pressionar o botão “Relatórios”. Então a tela com a lista dos relatórios disponíveis será exibida: O percentual de pagamentos com os métodos disponíveis (dinheiro, cartão de débito e crédito), total de clientes no aplicativo e também um gráfico de Pareto mostrando o total das vendas e pagamentos mensais do ano.

**4.5 Considerações finais**

Neste Capítulo foi descrito o processo de desenvolvimento do aplicativo *Easystore*, que se propõe a organizar crediários de clientes de lojas de itens diversos.

O produto gerado é um aplicativo multiplataforma (Android, iOS e Windows Phone) para testes e avaliações, cujos resultados serão discutidos no próximo capítulo.

5. ANÁLISE E DISCUSSÃO DOS RESULTADOS

Neste capítulo, serão analisados os resultados obtidos com os testes e uso da ferramenta em uma loja de roupas, na cidade de Taquaritinga, estado de São Paulo.

A análise visa validar a aceitação do modelo de negócio e desenvolvimento da ferramenta, atestando empiricamente como a mesma fez-se útil no cotidiano de pessoas que lidam diretamente com a manipulação de crediários.

**5.1 Teste de usabilidade**

O processo de desenvolvimento de software envolve uma série de atividades em que, apesar dos métodos, técnicas e ferramentas empregados, erros no produto ainda podem ocorrer (PRESSMAN, 1997).

Para a avaliação da primeira versão do aplicativo, criou-se um servidor utilizando a plataforma do Google *Cloud* para que a aplicação *Backend* pudesse conectar-se ao banco de dados e começasse a receber e responder requisições HTTP (seção 2.2.2.1). Com a base de dados criada no servidor, um usuário beta foi inserido manualmente e a aplicação passou por duas fases de testes descritas nos subcapítulos a seguir: iniciais com testadores e de usabilidade com potenciais usuários.

O aplicativo não foi disponibilizado na loja de aplicativos, portanto, foi instalado presencialmente nos celulares dos envolvidos com a validação da ferramenta.

5.1.1 Testes iniciais

Antes da liberação para testes com os potenciais usuários, dois testadores profissionais se disponibilizaram a realizar os testes iniciais da ferramenta desenvolvida a fim de encontrar fluxos essenciais com problemas e também, contribuir com sugestões.

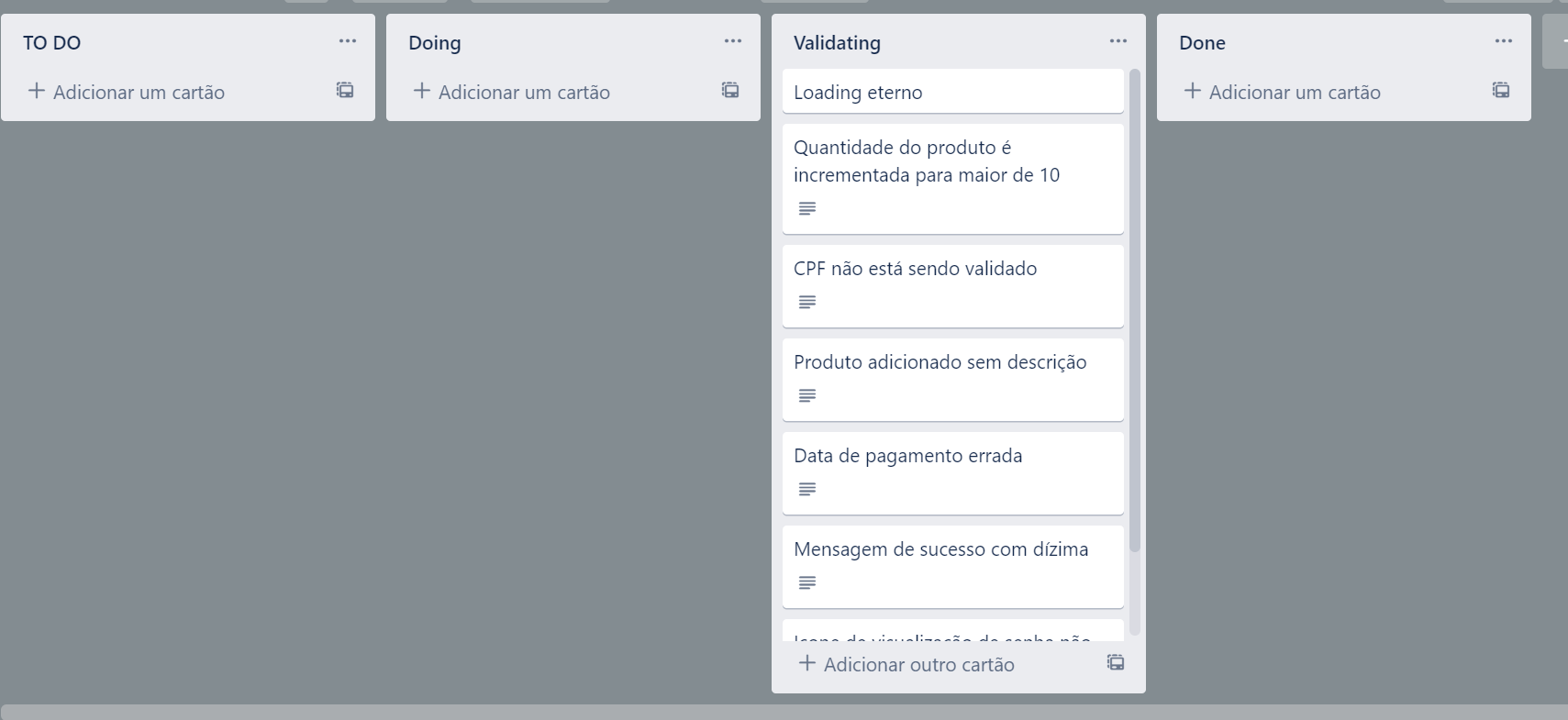
Para essa parte do trabalho utilizou-se o modelo ágil denominado Kanban. Tal modelo foi escolhido porque segundo Brechner (2015), o Kanban é um sistema com uma abordagem relacionada à produção, o mesmo controla o fluxo de trabalho de vários processos usando um conjunto de regras simples com o intuito de maximizar a eficiência do projeto. Anderson (2011) complementa que em um sistema Kanban, as tarefas são escritas em um *card* (bloco de notas, adesivo) e depois vinculados a um quadro com várias colunas. Conforme as pessoas envolvidas finalizam as tarefas, eles movem as tarefas para a coluna ao lado, representando que um processo do sistema de produção foi finalizado e que a tarefa progrediu para o próximo estágio. Fazendo isso, economiza-se energia, tempo, dinheiro e desperdício.

Abaixo encontra-se o quadro Kanban utilizado. O mesmo divide-se em cinco colunas, são elas:

1. *TO DO*: Os bugs encontrados pelos testadores foram inicialmente colocados nessa coluna para o desenvolvedor avaliar a urgência de cada um e começar a resolução;
2. *Doing:* O desenvolvedor ao escolher um bug para resolver, arrasta o *card* para esta coluna e o mesmo permanece ali até que o processo de desenvolvimento da correção (análise do código causador do problema, desenvolvimento de algoritmo de solução e testes de desenvolvimento) seja finalizado;
3. *Validating:* Ao finalizar a correção e atualizar o repositório com código, o desenvolvedor arrasta o *card* para essa coluna. Então, quando um dos testadores estiver disponível, irá pegar o mesmo e validar se o fluxo foi realmente corrigido. Caso o testador identifique que o erro ainda persiste, retorna o *card* para a coluna *TO DO,* fazendo com que o desenvolvedor revise o problema bem como a solução previamente estipulada;
4. *Done:* Caso o testador aprove a correção do problema, o mesmo coloca o *card* nesta coluna. Tal coluna faz se essencial pois é através da mesma que se mantém um histórico de todos os problemas evidenciados, corrigidos e avaliados até o momento na ferramenta de estudo.

O quadro com os bugs encontrados e corrigidos na primeira etapa de testes pode ser visualizado abaixo, na Figura 17.

Figura 17 - Quadro Kanban utilizado para a correção dos testes iniciais do aplicativo.



Fonte: Elaborado pelo autor.

Após a validação da ferramenta pelos testadores, pôde-se adquirir uma maior confiança no funcionamento dos requisitos funcionais e não-funcionais da ferramenta, podendo seguir então, para a próxima etapa de validação - Com os *stakeholders* potenciais.

5.1.2 Testes de usabilidade com usuários potenciais

Para o primeiro contato com o aplicativo, testes de usabilidade foram desenvolvidos a fim de familiarizar os potenciais usuários de uma loja de roupas na cidade de Taquaritinga com a aplicação bem como colher *feedbacks* sobre a ferramenta. Os testes descritos abaixo foram realizados com oito pessoas (Sendo quatro delas funcionárias da loja que posteriormente utilizaria o aplicativo por um mês e quatro outras pessoas que se disponibilizaram a realizar os testes fim de aumentar a amostra da pesquisa de usabilidade).

Todos os testes foram realizados com o desenvolvedor presente fisicamente no local.

1. Registrar um novo cliente (Ficha de crediário)
   1. Autenticar-se no aplicativo utilizando usuário e senha fornecidos;
   2. Acessar a tela de pesquisa de crediário;
   3. Acessar o formulário de cadastro de crediário;
   4. Informar os dados para a criação do crediário

Para garantir a integridade dos dados, realizar os cenários abaixo:

1. O CPF precisa ser válido e único, isto é, não pode haver um cadastro na base de dados utilizando aquele CPF;

2. Nome e o sobrenome não podem estar previamente cadastrados.

* 1. Registrar a informação e visualizar a mensagem de confirmação.

1. Registrar uma venda
   1. Autenticar-se no aplicativo utilizando usuário e senha fornecidos;
   2. Acessar a tela de pesquisa de crediário;
   3. Na tela de pesquisa, digite no campo o nome, sobrenome ou CPF do cliente e pressione o botão “Pesquisar”;
   4. O cliente será retornado. Então, clique sobre as informações do crediário dispostas na tela, ao fazê-lo os detalhes do crediário serão acessados. Nesta tela, pressione o botão “Vender”;
   5. Na tela de vendas, insira no mínimo três produtos no carrinho de compras;
   6. Pressione o botão finalizar e espere pela mensagem de sucesso.
   7. Confira na tela de detalhamento do crediário se o total devido foi acrescido com o valor da compra.
2. Registrar um pagamento
3. Autenticar-se no aplicativo utilizando usuário e senha fornecidos;
4. Acessar a tela de pesquisa de crediário;
5. Na tela de pesquisa, digite no campo o nome, sobrenome ou CPF do cliente e pressione o botão “Pesquisar”;
6. O cliente será retornado. Então, clique sobre as informações do crediário dispostas na tela, ao fazê-lo os detalhes do crediário serão acessados. Nesta tela, pressione o botão “Receber”;
7. Na tela de pagamentos, informe o valor, a data e a forma de pagamento;
8. Pressione o botão finalizar e espere pela mensagem de sucesso.
9. Confira na tela de detalhamento do crediário se o total devido foi debitado com o valor do pagamento.

4. Compartilhar total devido com o cliente no WhatsApp

1. Autenticar-se no aplicativo utilizando usuário e senha fornecidos;
2. Acessar a tela de pesquisa de crediário;
3. Na tela de pesquisa, digite no campo o nome, sobrenome ou CPF do cliente e pressione o botão “Pesquisar”;
4. O cliente será retornado. Então, clique sobre as informações do crediário dispostas na tela, ao fazê-lo os detalhes do crediário serão acessados. Nesta tela, pressione o botão “Vendas”;
5. Na tela de vendas, pressione o *float button* com o ícone do WhatsApp
6. O WhatsApp irá abrir e procurar o cliente informado, inserindo posteriormente a mensagem padrão do sistema para exibição de dívida.
7. Clique no ícone de “Enviar” do WhatsApp
8. A mensagem deverá ter sido enviada e o aplicativo EasyStore exibido novamente na tela de detalhamento de crediário.

5. Filtrar vendas por data e valor

1. Autenticar-se no aplicativo utilizando usuário e senha fornecidos;
2. Acessar a tela de pesquisa de crediário;
3. Na tela de pesquisa, digite no campo o nome, sobrenome ou CPF do cliente e pressione o botão “Pesquisar”;
4. O cliente será retornado. Então, clique sobre as informações do crediário dispostas na tela e ao fazê-lo os detalhes do crediário serão acessados. Nesta tela, pressione o botão “Vendas”;
5. Na tela, clique no *float button* com o ícone de pesquisa. Ao fazê-lo, um formulário para informar os filtros será exibido;
6. Informe a data inicial e final, bem como o valor mínimo e máximo das vendas pesquisadas;
7. Pressione o botão “Buscar”;
8. As vendas atendendo os filtros informados devem ser exibidas em uma listagem da tela.

6. Cancelar venda

1. Autenticar-se no aplicativo utilizando usuário e senha fornecidos;
2. Acessar a tela de pesquisa de crediário;
3. Na tela de pesquisa, digite no campo o nome, sobrenome ou CPF do cliente e pressione o botão “Pesquisar”;
4. O cliente será retornado. Então, clique sobre as informações do crediário dispostas na tela e ao fazê-lo os detalhes do crediário serão acessados. Nesta tela, pressione o botão “Vendas”;
5. Então, a tela de detalhamento de vendas será apresentada.
6. Abaixo da frase “Quer cancelar essa venda?”, clique em “Sim, quero”. Ao fazê-lo, uma mensagem de confirmação aparecerá;
7. Confirme o cancelamento e aguarde a mensagem de confirmação da transação;
8. Confirme na tela de informações gerais do crediário se o valor da venda foi subtraído do total devido.

7. Acessar relatório do total de crédito disponibilizado no mês atual

1. Na tela inicial do aplicativo, acesse a tela de relatórios através do botão “Relatórios” no menu inferior;
2. Então, na tela com as opções de relatório, acesse a opção “Créditos concedidos no mês”;
3. Então o valor de crédito concedido no mês atual deverá aparecer na tela para o usuário.

Quadro 2 - Resultados dos testes de usabilidade condensados

|  |  |
| --- | --- |
| **Tempo médio de execução das tarefas** | Tarefa 1: 3 minutos;  Tarefa 2: 5 minuto;  Tarefa 3: 7,5 minutos;  Tarefa 4: 10 minutos;  Tarefa 5: 4,5 minutos;  Tarefa 6: 3 minutos;  Tarefa 7: 1 minuto. |
| **Média de falhas no entendimento das representações e operações** | 7 falhas de entendimento |
| **Erros mais comuns dos**  **usuários** | * Por não haver um botão de carregamento, o usuário clica repetidas vezes no botão de finalizar; * O usuário coloca o nome inteiro do cliente na barra de pesquisa. Atualmente ou você coloca o nome ou o sobrenome; * O usuário clicava em “Venda” na tela de detalhamento de crediário esperando que abrisse o ponto de vendas quando na verdade, abre-se o histórico das vendas; |
| **Obstáculos da interface para**  **a interação** | * O usuário tenta inserir pontos de decimal nos valores monetários, não entendendo que existe uma máscara para isso; * O usuário tenta inserir pontuação na digitação do número de telefone e CPF do novo cliente; * O usuário confunde o botão de abrir o formulário de filtro da pesquisa com o botão que realmente realiza a pesquisa; * Estando na tela de histórico de venda, não existe botão de *home screen*, tendo o usuário que voltar duas telas para acessar a primeira tela da aplicação; * O botão de *logout* do aplicativo encontra-se apenas na primeira tela da aplicação; * Alguns usuários relataram não gostar do componente de digitação da data, principalmente os usuários de Iphone; |

Um questionário online com onze afirmações foi aplicado com as pessoas envolvidas nos testes a respeito da experiência com o aplicativo e solicitado que respondessem o quanto concordavam ou discordavam de cada uma delas.

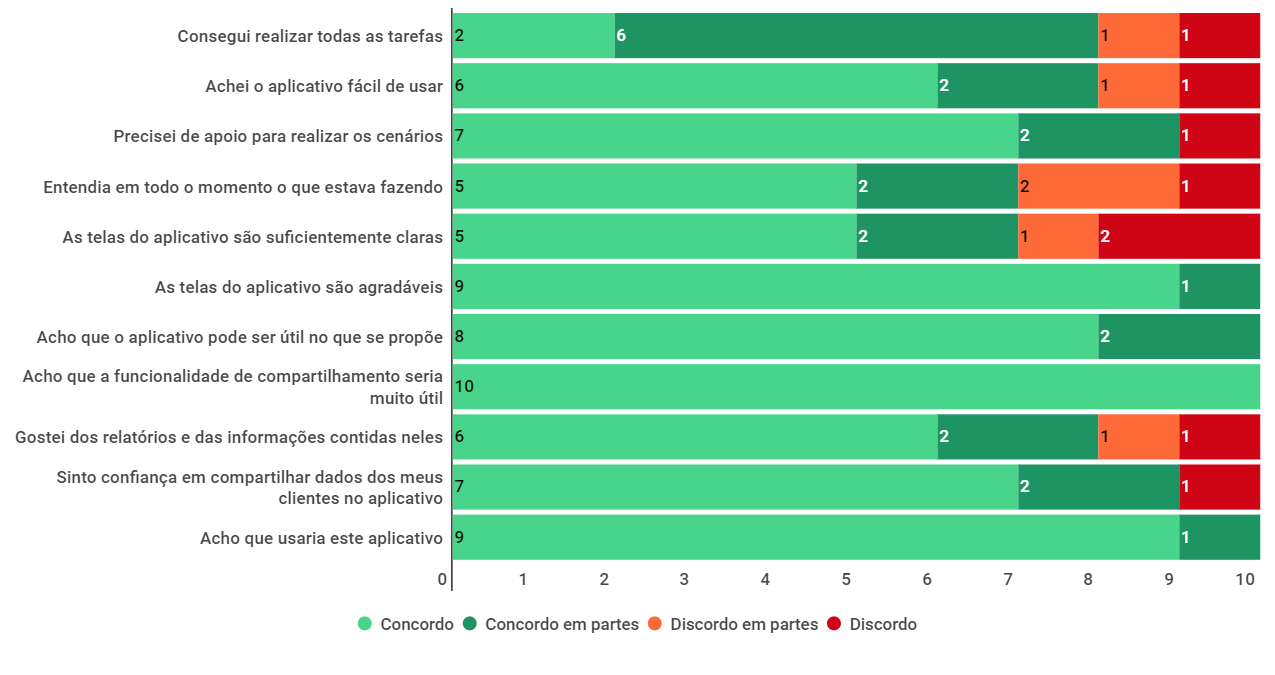
Os resultados deste questionário podem ser vistos na Figura 18.

Figura 18 - Avaliação dos stakeholders após os testes de usabilidade

Fonte: Elaborado pelo autor.

Além do questionário, solicitou-se também aos *stakeholders* do teste que descrevessem qualquer dificuldade na realização de alguma tarefa. As principais dificuldades relatadas foram:

1. *“Encontrei dificuldade para entender como um crédito é lançado. Achei o fluxo e o modelo de negócio confuso”*;
2. *“Apertei várias vezes o botão com o texto “Vendas” com a expectativa de acessar a tela para eu informar uma nova venda quando na verdade, abriu-se a tela de histórico e pesquisa de venda”*;
3. *“Na pesquisa de venda, cliquei por muitas vezes no floating button com o ícone de pesquisa a fim de pesquisar as vendas, mas tal ação não realiza a busca de vendas, mas sim abre o formulário para informar os filtros da pesquisa”;*
4. *“Ao querer encerrar a minha sessão no aplicativo, precisei voltar para a tela inicial do aplicativo, o que é oneroso. Seria muito melhor se o botão de saída estivesse presente na tela”.*

5.1.1 Resultados obtidos na loja de testes

A loja em potencial utilizou os serviços do aplicativo durante um mês e após tal período retornou alguns feedbacks:

1. Com o aplicativo, a busca do valor devido pelo cliente tornou-se mais rápida. A descrição do tempo médio bem como o fluxo antigo e atual encontram-se no quadro abaixo.

Quadro 3 - Fluxo de busca do valor do crediário antes e depois do uso da ferramenta

|  |  |  |
| --- | --- | --- |
|  | **Anteriormente** | **Atualmente** |
| **Tempo médio** | 15 minutos. | 30 segundos, isto é, aproximadamente 97% mais rápido. |
| **Descrição** | O vendedor dirigia-se ao cômodo da loja onde as fichas ficavam guardadas. As fichas eram ordenadas alfabeticamente. Então, o mesmo fazia uma busca manual do crediário requisitado e após encontrá-lo, digitava os itens da última compra bem como o total atual devido no WhatsApp do cliente. Esse processo manual, levava em média, dez minutos. | O usuário autentica-se na aplicação e pesquisa o cliente por nome, sobrenome ou CPF. A busca retorna inicialmente o valor devido pelo cliente. Caso o vendedor precise de mais detalhes do crediário, seleciona-se o mesmo. |

Fonte: Elaborado pelo autor.

1. O histórico com a data de compra, os produtos e o vendedor ficaram mais acessíveis. O compartilhamento de tais informações tornou-se bem trivial, que segundo os proprietários, facilita muito. No quadro abaixo, mostra-se o que foi relatado sobre o fluxo anteriormente e posteriormente ao uso da ferramenta EasyStore.

Quadro 4 - Histórico do crediário antes e depois do uso da ferramenta

|  |  |  |
| --- | --- | --- |
|  | **Anteriormente** | **Atualmente** |
| **Tempo médio** | 15 minutos. | 4 minutos, isto é, aproximadamente 74% mais rápido. |
| **Descrição** | O vendedor dirigia-se até onde localizavam as fichas de crediário físicas e após encontrar a ficha do cliente, analisava a mesma tentando entender a letra das pessoas que anotaram os dados das vendas anteriores. Após isso, tirava-se uma foto com o celular para o envio no WhatsApp do cliente. Caso o cliente estivesse na loja, mostrava-se pessoalmente os dados escritos no crediário. | Autentica-se no aplicativo e com alguns cliques encontra-se o histórico completo de vendas registradas. Podendo filtrar por data e valor. Podendo clicar no botão de compartilhamento presente na tela e enviar ao cliente diretamente no *WhatsApp.* |

Fonte: Elaborado pelo autor.

1. O acesso ao total de crédito concedido no mês atual foi citado como melhoria no dia a dia das atividades da loja. Os resultados podem ser acompanhados no quadro abaixo.

Quadro 5 - Comparação da visibilidade do total de crédito concedido com o uso da ferramenta

|  |  |  |
| --- | --- | --- |
|  | **Anteriormente** | **Atualmente** |
| **Tempo médio** | 50 minutos. | 15 segundos, isto é, aproximadamente 99.5% mais rápido. |
| **Descrição** | O vendedor dirigia-se até onde localizavam as fichas de crediário físicas e somava todas as vendas do mês corrente. Tal processo é oneroso e passível de erros aritméticos oriundos de desatenção e cansaço da atividade enfadonha. | Autentica-se no aplicativo e ao pressionar dois botões acessa-se o total vendido no mês atual. |

Fonte: Elaborado pelo autor.

Os *stakeholders* também ficaram a vontade para deixar suas sugestões, críticas ou qualquer comentário adicional, listados abaixo:

1. *“O compartilhamento das informações do crediário é muito legal. Me economiza um tempo enorme!”;*
2. *“Quando incluía um produto errado na venda, eu não conseguia devolver o mesmo. Tinha que cancelar toda a venda e fazer de novo sem aquele produto. Seria muito legal se houvesse a funcionalidade de devolução de itens”*;
3. *“Ao cancelar a venda e refazê-la, a data persistida é a atual. Já aconteceu de eu precisar cancelar a venda e refazer no outro dia, o cliente me questionou sobre a data da venda, pois ele não havia estado na loja naquele dia em particular”;*
4. *“Ao buscar um cliente e digitar nome e sobrenome, o aplicativo se perde”;*
5. *“Se pudesse fazer uma funcionalidade de análise de crédito, iria incrementar muito o valor do aplicativo”;*
6. *“Na tela de detalhamento da venda, existem dois botões relacionados a venda. Um leva para a pesquisa das vendas e outro, para a tela de ponto de venda. Muitas vezes eu clico em um querendo clicar em outro”.*

De acordo com os *feedbacks* recebidos pelo uso do aplicativo, os usuários indicam a necessidade de uma nova funcionalidade de devolução parcial de itens da venda após a confirmação e finalização ao invés de cancelar a venda como um todo.

A sugestão de melhoria do item 6 vem no sentido de inserir um ícone que aumente a familiaridade com a ação daquele botão, visto que existem dois botões relacionados ao mesmo domínio da aplicação na tela: venda. Outra solução seria inserir o botão de ponto de venda na tela de busca de vendas, mas nesse caso, o número de botões pressionados para se chegar até a tela de venda (principal tela do sistema) aumentaria em um.

Outra ponto foi um defeito na busca de crediário: atualmente ou o usuário digita o nome ou o sobrenome, caso digite os dois, o aplicativo não busca usando os dois como filtro. Para isso, existem duas possibilidades:

1. Separar cada dado em um *input* e então, realizar a busca utilizando os dados como filtro;
2. Desenvolver alguma lógica de quebra de *strings* em espaço, aplicando consequentemente cada parte da *string* como filtro no campo de nome e sobrenome do cliente.

Apesar das melhorias sugeridas, o retorno quanto a utilização da ferramenta no dia a dia foi positivo, conforme pode ser visto nos quadros 3, 4 e 5.

Das cinco pessoas iniciais que testaram o aplicativo, 80% afirmaram que a aplicação possui uma fácil usabilidade, a porção que discorda relaciona-se aos comentários abertos que indicam as necessidades de melhoria apontadas acima.

Relacionando a modelagem feita no capítulo 4 e as informações retornadas pela amostra da utilização beta da ferramenta, pôde-se constatar que a elaboração aqui descrita cumpriu o objetivo proposto: elaboração de uma aplicação híbrida para controle de crediário com uma fácil usabilidade e disponibilidade.

**5.4 Limitações do trabalho**

Por questões de limitação técnica e recursos financeiros, o aplicativo não foi disponibilizado na loja de aplicativos para os testes da versão beta, o que limitou a amostra de testes realizados, isto é, caso o aplicativo estivesse na loja, pessoas poderiam instalar facilmente nos seus *smartphones* e realizarem os primeiros testes sem a necessidade de ter um desenvolvedor ao lado para a instalação da ferramenta.

Ainda sobre os testes da versão beta da aplicação, a amostra foi pequena porque não houveram lojas dispostas a realizar a validação da aplicação nessa etapa do desenvolvimento. Seria interessante realizar novos testes porém com um maior número de pessoas e estabelecimentos na amostra a fim de certificar-se quanto a qualidade técnica, negocial e de usabilidade da ferramenta.

O tempo de uso da aplicação pela loja escolhida com estudo prático foi de apenas um mês, o que é relativamente pouco para que os usuários tenham *feedbacks* sobre a ferramenta. Tais testes empíricos não puderam ser mais longos nesta monografia devido ao breve intervalo de tempo entre o término da implementação e a escrita da mesma, bem como a escassez de recursos monetários, pois utilizou-se o servidor da Google *Cloud*, onde o primeiro mês de testes era gratuito.

**5.5 Considerações finais**

Neste Capítulo foram apresentadas as estratégias para o teste de usabilidade com o aplicativo *EasyStore* e seus principais resultados. Também foram relacionadas as respostas da avaliação de usabilidade. Por fim, foi feita uma análise crítica sobre os resultados e relatadas algumas limitações que se impuseram durante o desenvolvimento deste trabalho.

6. CONCLUSÃO

O uso de aplicativos para dispositivos móveis cresce exponencialmente ao passar dos anos e tal crescimento ocorre devido ao aumento da capacidade computacional para as ferramentas.

Empresas de todos os portes apostam cada vez mais - devido ao aumento e disponibilidade computacional - em aplicações móveis para administrar os seus negócios, pois a mesma proporciona praticidade ao acessar e manipular dados a fim de gerar uma informação. Com a internet móvel, desde que se possua um dispositivo inteligente conectado, consegue-se acessar dados armazenados em qualquer continente.

O mercado de crédito é de suma importância pois é um instrumento para promover o desenvolvimento econômico de qualquer nação. A concessão de crédito aumenta o poder de compra da população, de forma a aumentar o consumo, estimular a produção e aquecer a economia.

Pensando nisso, o presente trabalho, através da avaliação de soluções existentes e da coleta de dados composta de um questionário com oito perguntas envolvendo proprietários de micro-empresas que trabalham com o modelo de crediário, foram levantadas as necessidades primordiais desse grupo mercadológico e identificados as funcionalidades potenciais que definiram a proposta de trabalho: Desenvolver uma aplicação móvel híbrida que sirva como uma ferramenta de apoio a controle e concessão de crédito.

Para o desenvolvimento da solução tecnológica, paradigmas de desenvolvimento multiplataformas contemporâneos foram empregados, com destaque para as técnicas e padrões de projetos arquiteturais que garantem uma alta disponibilidade da aplicação, integridade de dados e usabilidade.

Após quatro meses de desenvolvimento e testes, a versão final foi disponibilizada e então, utilizada por uma loja de roupas que trabalha com um sistema manual de controle de fichas de crediário. Após um mês empregando a aplicação no cotidiano das atividades, a mesma apresentou *feedbacks* referentes aos valores intangíveis e tangíveis que o uso da ferramenta *EasyStore* trouxe*,* isto é, os *feedbacks* foram positivos e ressaltaram a diminuição no tempo depositado para manipular informações pertinentes ao controle de crediário da loja, o que, segundo os envolvidos, resulta em um aumento de credibilidade e satisfação dos clientes. Além disso, aumenta e facilita a visualização do estado atual do estabelecimento, facilitando tomadas de decisões referentes a concessão de crédito.

Sendo assim, no presente trabalho, obteve-se êxito no desenvolvimento de uma funcionalidade de controle de crediário em aplicativo para todas as plataformas atualmente no mercado.

O código-fonte da aplicação *EasyStore* está disponível no GitHub¹, bem como a aplicação *Backend* que acessa e dispõem os dados². Ambos os repositórios são *Open source*.

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

1 https://github.com/MatheusMinguiniPostGraduation/storeRecordApp

2 https://github.com/MatheusMinguiniPostGraduation/storeRecord

**6.1 Trabalhos futuros**

Como trabalhos futuros, dever se-ia incluir um requisito funcional de análise de crédito acessando alguma *API* externa como a do Serasa, a fim de melhorar o entendimento do perfil econômico/social do cliente que está obtendo o crédito e, então baseando-se nessa informação, conceder um limite de crédito na loja.

Gráficos evidenciando o cliente com mais dívidas na loja e o cliente que mais compra agrupado pelos meses do ano são requisitos funcionais que podem aumentar significamente o valor do produto final.

REFERÊNCIAS BIBLIOGRÁFICAS

Ausubel, D. P. **Aprendizagem significativa: A teoria de David Ausubel.** São Paulo: Moraes, 1982.

Annett, J; Duncan, K. (1967). **Task analysis and training design.** Journal of Occupational Psychology, 41, 211-221.

Araújo, B, M. **Hands-On RESTful Web Services with TypeScript 3: Design and develop scalable RESTful APIs for your applications.**  Birmingham: Editora Packt Publishing, 2019.

Azevedo, M; Costa, H. **Métodos para Avaliação de Postura Estratégica.** Caderno de Pesquisas em Administração. São Paulo, 18p. abril 2001.

Anderson, D, J. **Kanban: Mudança evolucionária de sucesso para o seu negócio de tecnologia.** Sequim, WA, USA: Editora Blue Hole Press, 2011.

Baalen, V.; Levicki, A.; Moore, K.; Netto, D.; Shevchenko, A. **Developing an Ionic Edge.** California: Editora Bleeding Edge Press, 2015.

Barbosa, S. D. J; Silva, B. S. **Interação Humano Computador.** Rio de Janeiro: Elsevier, 2010.

Beighley, L; Morrison, M. **Use a cabeça: PHP & MySQL.** Rio de Janeiro: Alta

Books, 2008.

Bento, E. J. **Desenvolvimento web com PHP e MySQL**. São Paulo: Casa do código, 2017.

Blatt, A. **Avaliação de Risco e Decisão de Crédito: Um Enfoque Prático**. São Paulo: Nobel, 1999.

Branas, R. **Angular Essentials**. Birmingham: Editora Packt Publishing, 2014.

Brechner, E. **Agile Project Management with Kanban.** Microsoft Express, 2015.

Boaglio, Fernando. **Spring Boot - Acelere o desenvolvimento de microsserviços.** São Paulo: Casa do Código, 2017.

Camden, R. **Apache Cordova in Action.** Nova York: Editora Manning Publications, 2016.

Cheng, F. **Build Mobile Apps with Ionic 2 and Firebase.** Nova Zelândia: Editora Apress, 2017.

# Cordeiro, G. Aplicações Java para a web com JSF e JPA. São Paulo: Casa do código, 2014.

# Cosmina, L. et al. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools. 5. ed. Nova York: Apress, 2017.

Date, C, J. **Introdução a sistemas de banco de dados.** Rio de Janeiro: Elsevier, 2003.

Davis, M.E; Phillips, J.A. **Aprendendo PHP & MySQL.** Rio de Janeiro: Alta Books,

2008.

Décio , J. **Guia de consulta rápida HTTP.** São Paulo: Novatec, 2001.

Deitel, P.; Harvey, D. **Java - Como programar.** 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Domingues, D. G.  **O Uso de Metáforas na Computação** – Dissertação apresentada para obtenção do título de Mestre em Ciências da Comunicação – Escola de Comunicações e Artes da Universidade de São Paulo. Orientador: Prof. Dr.Martin Grossmann – São Paulo, 2001. Disponível em: <http://www.teses.usp.br/teses/disponiveis>. Último acesso em: Março de 2020.

# Duarte, L. Programação Web com Node.js: Completo, do Front-end ao Back-end. Gravataí: LuizTools, 2017.

Fowler, M. **Patterns of Enterprise Application Architecture.** Boston: Pearson Education, 2003.

Freeman, E. et al. **Use a cabeça: Padrões de projetos.** Rio de Janeiro: Alta Books, 2007.

Gamma, E. et al. **Design Patterns – Elements of Reusable Object-Oriented Software.** Boston: Addison-Wesley, 1994.

Gois, A. Ionic Framework: **Construa aplicativos para todas as plataformas mobile**. São Paulo: Casa do Código, 2017.

Gourley, D; Totty, B. **HTTP: The definitive Guide.** California: O’reilly Media, 2015.

Griffith, C. **Mobile App Development with Ionic.** California: O’Reilly Media, 2017.

Kalin, M. **Java Web Services.** Cambridge:O’Reilly Media, 2013.

Karanam, R. R. **Mastering Spring 5.** 2. ed. Birmingham: Packt Publishing, 2019.

Kurose, J. F. **Redes de computadores e a internet: Uma abordagem top-down.** São Paulo: Pearson Education do Brasil, 2010.

Peterson, A. **Everything you need to Ace Math in one big fat notebook.** Nova York: Workman, 2016.

Preece, R; Rogers, Y; Sharp, H. **Design de interação: Além da interação humano-computador.** 3. ed. Porto Alegre: Bookman, 2013.

Khanna, R; Sani, Y; Hoc, P. **Ionic: Hybrid Mobile App Development.** Birmingham: Editora Packt Publishing, 2017.

# Konda, M. Just Hibernate: A Lightweight Introduction to the Hibernate Framework. California: O'Reilly Media, 2018.

Kurose, J. F. **Redes de computadores e a internet: Uma abordagem top-down.** São Paulo: Pearson Education do Brasil, 2010.

Lim, G. **Beginning Angular with TypeScript.** Nova York: Edição do autor, 2017.

Looper, J. **What is a Web View?** Disponível em: &lt;https://developer.telerik.com/featured/what-is-a-webview/&gt;. Último acesso em: agosto de 2018.

Manzano, A. L. N. G; Manzano, J. A. N. G. **Estudo dirigido: Microsoft Excel 2016.** São Paulo: Érica, 2015.

Molinari, W. **Desconstruindo a Web: As tecnologias por trás de uma requisição.** São Paulo: Casa do Código, 2016.

Morris, M.; Schindehutte, M.; Allen, J. **The entrepreneur's business model: toward a unified perspective.** Journal of business research, v. 58, n. 6, p. 726-735, 2005.

Nielsen, J. **Usability engineering**. San Francisco: Morgan Kaufmann, 1993.

Oliveira, H, C. **JPA Eficaz: As melhores práticas de persistência de dados em Java.** São Paulo: Casa do código, 2014.

Orofino, M. A. R. **Técnicas de criação do conhecimento no desenvolvimento de modelos de negócio.** 233 p. Dissertação (Mestrado). Centro Tecnológico, Programa de PósGraduação em Engenharia e Gestão do Conhecimento., Universidade Federal de Santa Catarina. 2011.

Osterwalder; A. **Business Model Generation: Inovação em Modelos de Negócios.** Rio de Janeiro: Alta Books, 2011.

Phan, H. **Full Stack Mobile App with Ionic Framework.** Estados Unidos: Editora Hoc Phan, 2014.

Pereira, S. J. **Gestão e Análise de Risco de Crédito.** 4a ed. São Paulo: Atlas, 2000.

Peres, R. **Mastering ASP.NET Core 2.0.** Birmingham:Packt Publishing, 2017.

Pressman, **R. S. Software Engineering - A practitioner's approach.** 4ª Edição, McGraw-Hill, 1997.

Rajput, Dinesh. **Mastering Spring Boot 2.0.** Birmingham: Packt Publishing, 2018.

Reynolds, M. S. **Principles of information systems.** Boston: Cengage Learning, 2018.

Rocha, H. V; Baranauskas, M. C. C. **Design e avaliação de interfaces humano-computador.** Campinas: Unicamp, 2003

Santos, J. O. **Análise de crédito.** 6. ed. São Paulo: Atlas, 2015.

Saudate, A. **SOA aplicado: Integrando com web services e além.** São Paulo: Casa do código, 2014.

Sebrae. **Cartilha: O quadro de Modelo de Negócios.** Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/bis/quadro-de-modelo-de-negocios-para-criar-recriar-e-inovar,a6df0cc7f4217410VgnVCM2000003c74010aRCRD>

Acesso em: Março de 2020.

Silva, J. P. **Análise e Decisão de Crédito.** São Paulo: Atlas, 1988.

Silveira; et al. **Introdução à Arquitetura de Design de Software.** Rio de Janeiro: Elsevier, 2012.

Schildt, Herbert. **Java para Iniciantes: Crie, Compile e Execute Programas Java Rapidamente.** 6. ed. Porto Alegre: Bookman, 2015.

# Speth, C. The SWOT Analysis: A key tool for developing your business strategy. Nova York: 50Minutes.com, 2015

Stallings, W. **Redes e sistemas de comunicação de dados: teorias e aplicações corporativas.** 5 ed. Rio de Janeiro: Elsevier, 2005.

Walls, C. **Spring in action**. 5. ed. Nova York: Manning, 2019.

Wilken, J. **Ionic in Action: Hybrid Mobile Apps with Ionic and AngularJS.** Nova York: Editora Manning Publications, 2016.

Meirelles, F. C. **Pesquisa anual do FGVcia - Centro de Tecnologia de Informação Aplicada da FGV-EAESP.** Disponível em: <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/pesti2019fgvciappt\_2019.pdf >

Acesso em: Março de 2020.

# Weissmann, H, L. Vire o Jogo com Spring Framework. São Paulo: Casa do Código, 2012.

# Anexo A – Questionário de validação das ideias do aplicativo

1. **Qual ferramenta abaixo você utiliza para o controle de valores devidos na sua loja?**

( ) Aplicativo mobile específico para isso

( ) Aplicativo Desktop específico para isso

( ) Aplicativo mobile/desktop de anotações (Blocos de notas, Word, LibreOffice, etc)

( ) Planilhas eletrônicas (Excel, LibreOffice, etc)

( ) Ferramentas manuais (Bloco de anotações, cadernos, ficha física de crediário, etc).

1. **Caso o cliente queira saber o quanto deve, em quanto tempo você obtém essa informação em média?**
2. **Qual o sistema operacional do seu celular?**

( ) Android ( ) iOS

1. **Você estaria disposta a instalar um aplicativo no seu celular para lançar as vendas e os pagamentos dos clientes?**

( ) Sim ( ) Não

1. **Escolha a funcionalidade extra que um aplicativo de controle de créditos deve possuir**

( ) SMS automático para as pessoas que não pagaram nos últimos 30 dias.

( ) Compartilhamento do valor devido no WhatsApp.

( ) Consulta da situação atual do CPF perante órgãos de proteção ao crédito.

1. **Caso o aplicativo seja pago, qual o modelo lhe agradaria mais?**

( ) Pagar uma porcentagem pequena em cada venda registrada.

( ) Pagar um valor constante todo o mês.

1. **Na sua opinião, qual seria o maior benefício de possuir um aplicativo controlando os créditos e débitos dos clientes na sua loja?**

( ) Agilidade nas informações

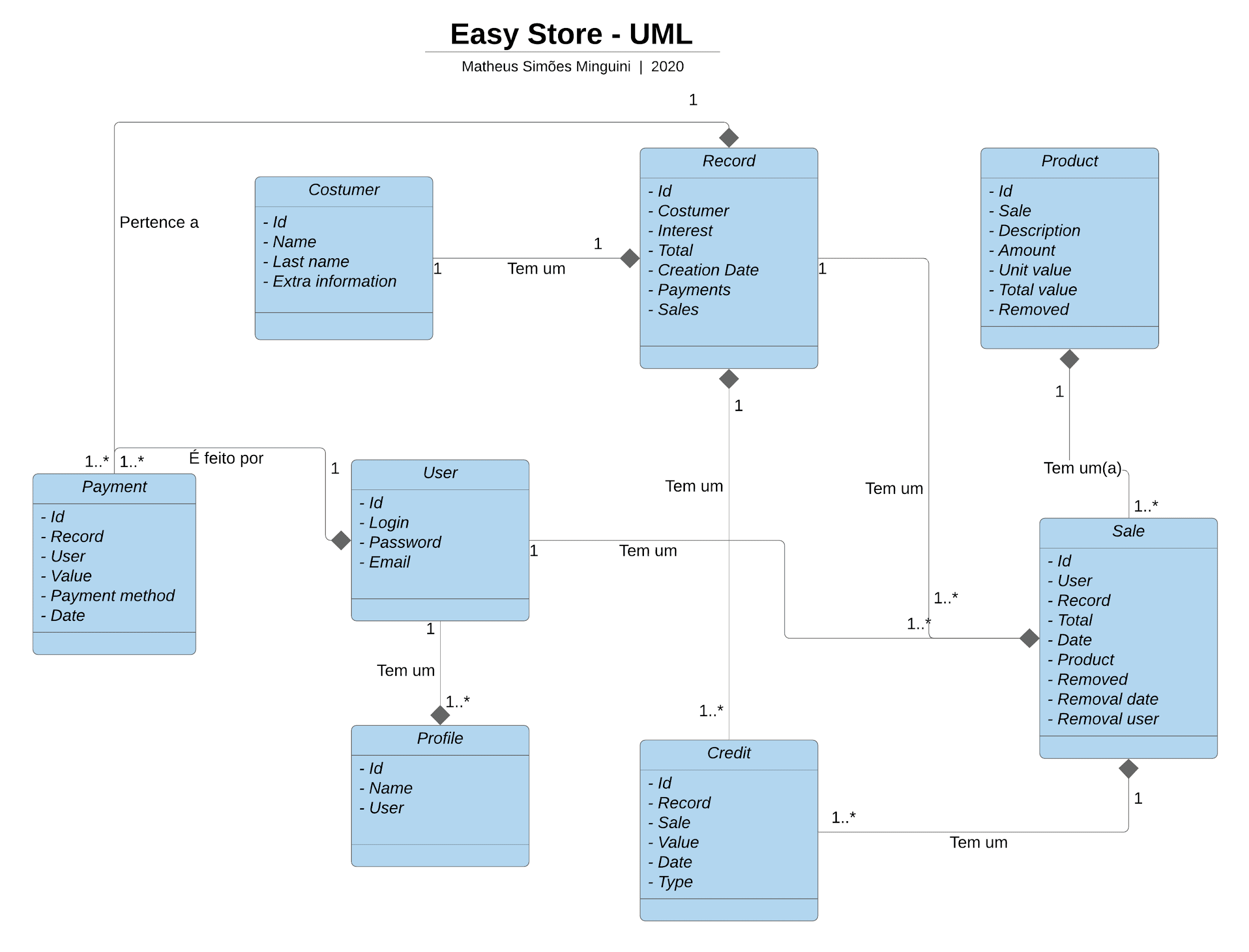
( ) Segurança na integridade dos dados

( ) Acesso a informação de qualquer lugar com internet e celular

1. **Você possui acesso a internet móvel**

( ) Sim ( ) Não

**Anexo B – Diagrama UML**



A screenshot of a cell phone

Description automatically generated**Anexo C – *Interface* do aplicativo**

