JOÃO PEDRO BATISTA FERREIRA

RELATÓRIO DE ESTÁGIO SOBRE A REFATORAÇÃO DE UMA ARQUITETURA MOBILE NA EMPRESA DTI DIGITAL

LAVRAS – MG

2019

JOÃO PEDRO BATISTA FERREIRA

RELATÓRIO DE ESTÁGIO SOBRE A REFATORAÇÃO DE UMA ARQUITETURA MOBILE NA EMPRESA DTI DIGITAL

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel

Prof. Dr. André Pimenta Freire
Orientador

BSc. Gustavo Lopes Dominguete
Coorientador

LAVRAS – MG

2019

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Ferreira, João Pedro Batista

Relatório de estágio sobre a refatoração de uma arquitetura mobile na empresa Dti Digital / João Pedro Batista Ferreira. 2^a ed. rev., atual. e ampl. – Lavras : UFLA, 2019.

41 p. : il.

Relatório de estágio(graduação)–Universidade Federal de Lavras, 2019.

Orientador: Prof. Dr. André Pimenta Freire.

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

JOÃO PEDRO BATISTA FERREIRA

RELATÓRIO DE ESTÁGIO SOBRE A REFATORAÇÃO DE UMA ARQUITETURA MOBILE NA EMPRESA DTI DIGITAL

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel

APROVADA em 05 de Dezembro de 2019.

Prof. Dr. Rafael Serapilha Durelli	UFLA
BSc. Jordann Alessander Rosa Almeida	Dti Digital

Prof. Dr. André Pimenta Freire
Orientador

BSc. Gustavo Lopes Dominguete
Co-Orientador

LAVRAS – MG
2019

Dedico este trabalho à toda minha família. Em especial aos meus pais Carlos e Aparecida e minhas irmãs Ana Flávia e Maria Amélia.

AGRADECIMENTOS

Agradeço primeiramente à Deus por me dar força e saúde para conseguir concluir este curso.

Aos meus pais por sempre me apoiarem nos momentos mais difíceis e por estarem sempre ao meu lado durante esta caminhada.

Aos meus falecidos avós, que estejam onde estiverem sei que estão orgulhosos.

À todos meus familiares que me ajudaram e apoiaram direta ou indiretamente.

Aos meus muitos amigos que fiz durante essa caminhada, obrigado por todas as experiências que vivemos.

Aos meus irmãos de coração da república ConCerva, obrigado por tudo que aprendi com vocês.

Ao professor André Pimenta por todos os ensinamentos e paciência ao me orientar durante a criação deste trabalho.

Ao meu querido amigo Gustavo Lopes por me abrir tantas portas que não consigo nem contar mais e me ajudar em vários momentos durante minha trajetória acadêmica e profissional.

À UFLA e todo o corpo docente por me proporcionar uma educação de qualidade, além de contribuir na construção do meu caráter como um cidadão.

”Em algum lugar, alguma coisa incrível está esperando para ser conhecida.” (Carl Sagan)

RESUMO

Este relatório de estágio descreve as atividades realizadas durante o período de estágio realizado na empresa Dti Digital. O projeto no qual o estagiário atuou consiste em atividades de desenvolvimento e manutenção de um aplicativo de uma rede bancária, que possibilita a seus clientes realizarem operações ordinárias por meio de seus *smartphones*. O estágio proporcionou oportunidades para que o aluno de graduação pudesse viver as experiências de trabalhar em uma empresa de modo a capacitá-lo a atuar em diversos projetos do mercado de tecnologia da informação e confirmar também a importância dos conceitos apresentados durante a graduação. Desta forma, o objetivo principal do estágio foi de possibilitar ao aluno a atuação com uma experiência no mercado de trabalho atuando em um projeto grande que busca solucionar um problema real. Como resultados, pode-se destacar que o estagiário pôde aplicar seus conhecimentos adquiridos ao longo da graduação e do período de estágio a fim de auxiliar no desenvolvimento da aplicação acima citada. A experiência de trabalhar em um projeto de tal porte, que tem por objetivo solucionar o problema de uma empresa que possui inúmeros clientes, foi de grande importância para contribuir com os conhecimentos e amadurecimento de um profissional que está iniciando sua carreira. Outro ponto muito importante a destacar foi a possibilidade de observar o uso dos conhecimentos adquiridos ao longo da graduação, pois a confirmação da teoria através da prática, além de muito gratificante, salienta ainda mais a importância do aprendizado dos conceitos de Ciência da Computação.

Palavras-chave: Processos de Software. Desenvolvimento de Software. Aplicações mobile.

ABSTRACT

This report describes the activities performed by an undergraduate student during his internship at Dti Digital. The project in which the intern worked consisted of development and maintenance activities of an application of a banking application that enables its customers to perform ordinary operations through their smartphones. It is important for an undergraduate student to experience working in a company in order to act in various projects in the information technology market. This also enables to practise important concepts studied during the degree. The main goal of the internship was to have an experience in the job market working on a large project that seeks to solve a real problem. As a result, the intern was able to apply the knowledge acquired during the degree and the internship period to assist in the development of the aforementioned application. The experience of working on such a project, which aims to solve the problem of a company that has many clients, was valuable to contribute with the knowledge and maturity of a professional who is starting his career. Another very important point to note was the possibility of observing the use of the knowledge acquired during the degree, as an application of theory through practice. In addition to being very rewarding, it also emphasizes the importance of learning the concepts of Computer Science.

Keywords: Software Process. Software Development. Mobile Application.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O presente documento tem o objetivo de descrever as atividades realizadas durante o período de estágio na empresa Dti Digital. O documento descreve tecnologias utilizadas para o desenvolvimento de uma aplicação móvel e uma aplicação de *web services*, e também os processos e metodologias de desenvolvimento de software aplicados no decorrer do projeto no qual o estagiário atuou.

Com a popularização do uso de aparelhos celulares *smartphones* em praticamente todas as atividades realizadas por uma pessoa em seu dia-a-dia, as empresas têm buscado cada vez mais oferecer soluções e produtos para seus clientes por meio da plataforma de dispositivos móveis (COMPILA, 2016).

A solução mais aplicada pelas empresas se dá através de aplicativos, que nada mais são do que softwares que são executados em aparelhos *smartphones* ou *tablets*. O desenvolvimento de tais aplicações pode ser classificado em dois tipos: o desenvolvimento mobile híbrido e o nativo. No método nativo, o software é desenvolvido utilizando uma linguagem específica para um sistema operacional, enquanto no método híbrido o aplicativo é construído geralmente através de linguagens como HTML, CSS e Javascript, de forma que o aplicativo gerado pelo código fonte escrito com essas linguagens possa ser executado em mais de um sistema operacional (BUDIU, 2013).

A demanda pelo desenvolvimento de sistemas de software para *smartphones* tem sido muito alta, uma vez que o engajamneto causado pela facilidade e comodidade oferecida pelos aplicativos é algo que as empresas estão explorando no mundo todo. Consequentemente, a demanda por profissionais capacitados também tem sido muito elevada.

Um profissional que deseja criar soluções por meio de aplicativos precisa ter conhecimentos em diversas áreas da Computação, como por exemplo: linguagens de programação, seja uma linguagem para desenvolvimento nativo ou híbrido; banco de dados, uma vez que a forma que os dados são armazenados, tanto interna quanto externamente, desempenha um papel fundamental para a performance do aplicativo; design e interação humano computador, além de resolver um problema do usuário é muito importante garantir o engajamento e satisfação do mesmo ao utilizar a aplicação; conhecimentos sobre arquitetura de computadores, considerando que o aplicativo será executado também é muito importante, a fim de explorar os recursos disponíveis para garantir uma aplicação que possua alto desempenho.

1.1 Objetivo geral do estágio

O estágio teve como objetivo possibilitar ao estagiário a aquisição de experiência com processos de desenvolvimento de software utilizados atualmente no mercado de trabalho e capacitá-lo para atuar na construção de sistemas, com foco no desenvolvimento de aplicações multiplataformas. Esse objetivo consistiu em permitir ao estagiário atuar no desenvolvimento de um aplicativo nas plataformas Android e iOS e uma aplicação de Web Services.

1.1.1 Objetivos específicos

Foram definidos os seguintes objetivos específicos:

- Adquirir habilidades com as plataformas Android e iOS e suas arquiteturas, bem como linguagens de programação como Java e Swift.
- Experienciar os processos organizacionais durante a construção de uma solução tecnológica para um problema do mundo real, utilizando metodologias ágeis durante o desenvolvimento das aplicações, como por exemplo o SCRUM.
- Descrever detalhadamente cenários de teste e realizar a implementação de testes unitários visando garantir a qualidade da aplicação.
- Utilizar dos conhecimentos adquiridos durante a graduação em Ciência da Computação e aplicá-los no mercado de trabalho.

2 DESCRIÇÃO DO LOCAL DE ESTÁGIO

Este capítulo apresenta uma descrição geral da empresa onde o estágio foi realizado, abordando informações como a localização física, seus escritórios, ramos de atuação e objetivo da empresa. É dado um destaque para o escritório de Lavras, onde o estagiário realizou suas atividades na maior parte do tempo.

A Dti Digital é uma empresa do ramo de tecnologia da informação fundada em 2009 com o principal objetivo de auxiliar outras empresas no processo de transformação digital, atuando principalmente através de soluções apresentadas por meio de software e levando a cultura ágil para o ambiente de seus clientes e parceiros.

A empresa atua em diversos ramos e desenvolve soluções que atendem os mais variados mercados como, por exemplo, o de logística, mobilidade, imobiliário, farmacêutico, mineração, bancário e até mesmo o mercado agropecuário. Embora a grande maioria dos produtos desenvolvidos pela Dti sejam sistemas de software, normalmente em forma de aplicativos para dispositivos móveis ou aplicações web, a empresa não se encaixa somente na definição de uma fábrica de software, uma vez que o foco principal da empresa é de oferecer soluções específicas para cada modelo de negócio a partir da cultura ágil, visando sempre a melhor experiência para o seu cliente.

A Dti possui atualmente três escritórios em duas cidades. Sua sede, também conhecida como Dti Place, localiza-se na cidade de Belo Horizonte, onde a empresa conta com mais um escritório chamado Dti Garden. O Dti Place é o escritório principal, onde estão alocadas as equipes do departamento administrativo, mas também conta com equipes de marketing e desenvolvimento. O Dti Garden possui três andares há pouco inaugurados, onde seu foco consiste em comportar equipes dos clientes da empresa juntamente com equipes de desenvolvimento. A Dti também possui um escritório na cidade de Lavras, que no momento conta apenas com equipes de desenvolvimento. Atualmente são mais de 500 colaboradores contribuindo com o crescimento e missão da empresa, com a grande maioria alocada nos escritórios de Belo Horizonte.

O estágio aconteceu quase que em sua totalidade no escritório de Lavras, que foi inaugarado em maio de 2019 e por isso possui um espaço menor quando comparado aos escritórios sedes. Atualmente, o espaço tem a capacidade para comportar 11 colaboradores em sua totalidade. As equipes alocadas no espaço da Dti em Lavras contam somente com desenvolvedores pertencentes a várias tribos. Portanto, toda a parte gerencial e de negócios é realizada nos es-

critórios localizados na capital mineira e a comunicação com as outras equipes é feita através de vídeo conferências ou ligações. Tanto o estagiário quanto os outros colaboradores tiveram a oportunidade de trabalhar por alguns dias em Belo Horizonte, podendo assim participar de reuniões e apresentações técnicas pessoalmente com os outros funcionários da empresa.

A duração do estágio totalizou 510 horas, tendo início em 12 de agosto de 2019 e término em 08 de novembro de 2019, com uma carga horária semanal de 40 horas.

3 PROCESSOS TÉCNICOS

Este capítulo tem como objetivo descrever os conceitos e processos técnicos utilizados pelo estagiário no decorrer do estágio, e também apresentar a fundamentação teórica das ferramentas e tecnologias utilizadas para o desenvolvimento das atividades. A Seção 3.1 descreve os conceitos de processo de software, seguida pela Seção 3.2, que aborda as metodologias e processos de construção de software aplicados na Dti, bem como a organização das equipes dentro da empresa. Na Seção 3.3 são expostos os tipos de teste utilizados durante o processos de construção do software e a importância deles para a garantir a qualidade da aplicação. Para finalizar, são apresentadas na Seção 3.4 as tecnologias e ferramentas utilizadas pelo estagiário durante a realização das atividades.

3.1 Processo de Software

De acordo com Pressman e Maxim (2016), a Engenharia de Software é baseada em camadas, sendo a camada de processo a responsável por possibilitar a organização do desenvolvimento de software de forma racional e dentro do prazo. O processo de software possibilita o gerenciamento de projetos e permite estabelecer quais métodos técnicos serão utilizados para a construção do software, bem como os artefatos e documentos que serão gerados.

Pressman e Maxim (2016) definem o processo de software como o conjunto de atividades, ações e tarefas realizadas com o objetivo de construir algum produto. Os autores defendem também que o processo de software não deve ser um padrão rígido a ser seguido rigorosamente, mas sim algo adaptável e que permita à equipe envolvida no desenvolvimento do software escolher quais atividades e ações realizar.

Um processo de Engenharia de Software completo pode ser estabelecido a partir de uma metodologia de software, que, ainda segundo os mesmos autores (PRESSMAN; MAXIM, 2016), possui cinco atividades principais, sendo elas:

- comunicação - é de vital importância comunicar-se com o cliente para se obter o entendimento sobre o projeto.
- planejamento - processo de mapeamento e descrição das atividades técnicas, dos possíveis riscos, recursos necessários e definição de cronograma.
- modelagem - a ideia principal é criar um esboço do que se pretende desenvolver para poder ter uma noção do projeto como um todo.

- construção - consiste basicamente na criação de código e teste das funcionalidades desenvolvidas.

- emprego - após finalizado o desenvolvimento, do projeto como um todo ou de uma parte entregável funcional, o produto é entregue ao cliente a fim de se obter uma avaliação da aplicação.

As atividades descritas anteriormente podem ser aplicadas a projetos dos mais variados portes. Os detalhes dos processos de software podem ser diferentes de acordo com o projeto. Contudo, as atividades continuarão tendo as mesmas finalidades, podendo ser implementas iterativamente de acordo com a construção do software.

3.2 Metodologias de desenvolvimento de software

Conforme descrito por Pressman e Maxim (2016), uma metodologia de desenvolvimento de software ou modelo de processo de software pode ser visto como um framework dos processos de software, sendo adaptáveis ao escopo do projeto, abordando o desenvolvimento de software de diferentes maneiras.

3.2.1 Metodologia ágil

O desenvolvimento ágil teve seu início quando a *Agile Alliance* (Aliança dos Ágeis), um grupo de 17 renomados desenvolvedores, autores e consultores de software publicaram o manifesto ágil, o qual começa da seguinte maneira:

> ”Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
>
> - Indivíduos e interações mais que processos e ferramentas
> - Software em funcionamento mais que documentação abrangente
> - Colaboração com o cliente mais que negociação de contratos
> - Responder a mudanças mais que seguir um plano
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda” (BECK et al., 2001).

Soares (2004) complementa que as metodologias ágeis nada possuem de novo além dos métodos e processos tradicionais de Engenharia de Software. Contudo, seu foco e preocupação é o que as diferencia. A metodologia ágil foca em pessoas e não em processos, preocupa-se em

gastar menos tempo com a documentação dos processos e mais tempo no software funcionando e agregando valor ao cliente.

Ainda segundo Soares (2004), as metodologias ágeis surgiram como uma resposta às tradicionais metodologias de Engenharia de Software, a fim de sanar problemas e fraquezas das metodologias tradicionais.

Ao se falar de metodologia ágil é muito interessante e importante definir o termo ágil. Para Pressman e Maxim (2016), uma equipe ágil é aquela capaz de se adaptar rapidamente às mudanças. Mudanças são comuns durante um desenvolvimento de software. Uma equipe ágil deve conseguir responder bem a mudanças, como mudanças de membros da equipe, das tecnologias utilizadas, do escopo do projeto, entre outras, e ainda assim manter a qualidade do produto em desenvolvimento e entregar dentro do prazo.

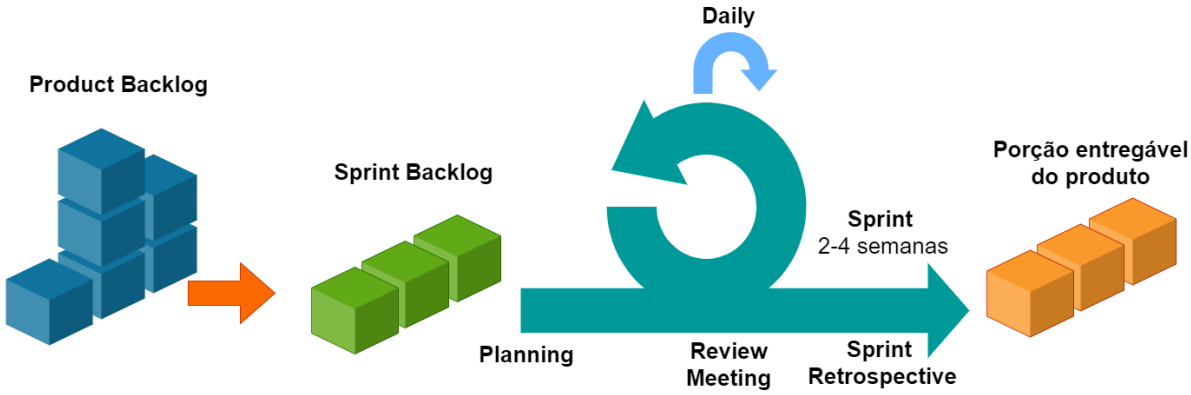
3.2.2 Metodologia SCRUM

Conforme descrito por Bissi (2007), o SCRUM teve sua origem em meados dos anos 90, tendo sua primeira concepção por Jeff Sutherland na Easel Corporation. Pouco depois, a metodologia foi refinada por Ken Schwaber, ao adaptar o conceito do SCRUM dentro do mundo de desenvolvimento de sistemas e processos. O SCRUM baseia-se principalmente no processo de desenvolvimento incremental, aplicado ao longo de curtos períodos de tempo - chamados de Sprint - e centrado na equipe responsável pela construção do produto.

Sabbagh (2014) descreve as atividades, ferramentas e ritos do SCRUM que auxiliam no processo de desenvolvimento de software, mantendo a equipe organizada, alinhada e sempre focada na entrega com valor agregado. Depois de levantados os requisitos em conjunto com o cliente, são definidas as funcionalidades da aplicação. Essas funcionalidades são organizadas em uma lista chamada *Product Backlog* e após serem definidos os níveis de prioridade de cada funcionalidade, as atividades necessárias para desenvolver as funcionalidades de maior prioridade são adicionadas à lista chamada *Sprint Backlog*. Antes de começar o desenvolvimento, a equipe realiza uma reunião, conhecida como *Planning*, para discutir sobre as atividades que serão desenvolvidas durante a *Sprint* atual (geralmente um período de duas a quatro semanas). Durante o andamento da *Sprint*, a equipe deve realizar reuniões diárias (*daily*) para alinhar o andamento do projeto e identificar possíveis riscos ou problemas não previstos durante a *Planning*. Ao final da *Sprint* , a porção do software desenvolvida é apresentada ao cliente em uma reunião chamada *Review Meeting* e a equipe também deve fazer uma análise da *Sprint* recém

terminada, buscando identificar quais foram os pontos positivos e negativos durante essas semanas. Esse rito é conhecido como *Sprint Retrospective*. O ciclo de execução dos ritos do SCRUM pode ser observado na Figura 3.1.

Figura 3.1 – Diagrama do ciclo de execução do SCRUM



Fonte: Adaptado de Sabbagh (2014)

3.2.3 Dti-flow

Buscando sempre manter a qualidade do código e gerar valor para o cliente, a Dti definiu um fluxo de desenvolvimento para as atividades a serem implementadas em todos os projetos executados por seus colaboradores. Esse processo é conhecido como Dti-flow. Ele define etapas pelas quais uma funcionalidade deve passar para ser considerada como finalizada e qual profissional é responsável por cada uma delas.

A Dti possui um sistema de Tribos. Cada tribo é responsável por determinados projetos e clientes, e possui vários *squads* (equipe com geralmente 4 a 7 pessoas), sendo que cada *squad* é responsável comumente por um determinado projeto. Um *squad* normalmente é composto por um ou mais desenvolvedores (Dev), um desenvolvedor líder (DL), um *Product Owner* (PO) e um *Designer* (Ux). Contudo, a composição das equipes pode variar de acordo com a necessidade do projeto. Cada um desses profissionais é responsável por uma ou mais atividades dentro do processo Dti-flow.

Existem duas versões do Dti-flow, uma para atender o desenvolvimento das atividades de *front-end* e outra para as de *back-end* de uma aplicação. Ambas possuem basicamente as mesmas etapas, apresentando apenas algumas diferenças.

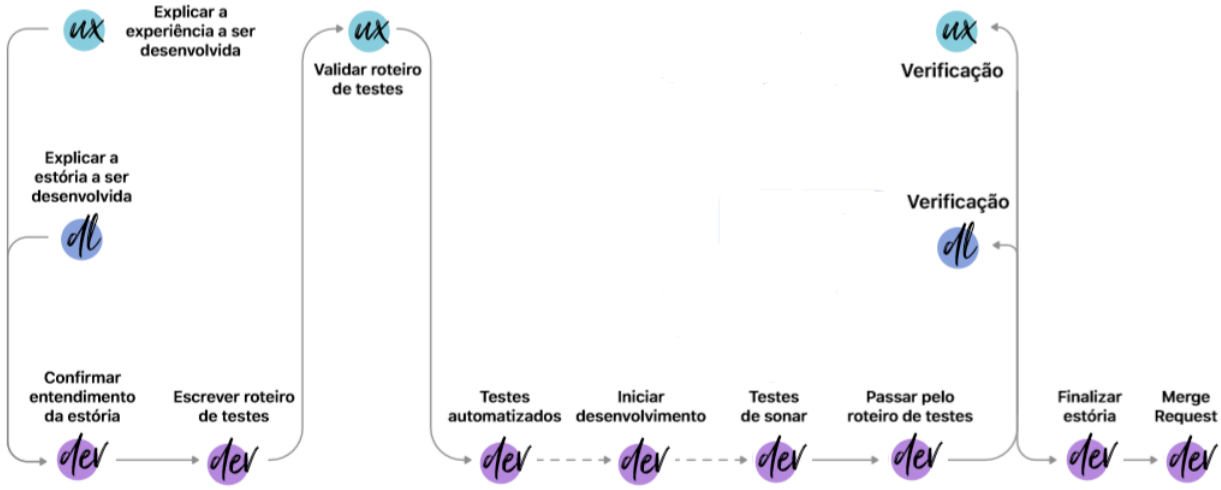
A seguir estão listadas as atividades a serem exercidas de acordo com a função de cada membro do *squad* dentro do Dti-flow para *front-end*. A Figura 3.2 representa a ordem em que as atividades devem ser executadas.

- DL
    1. Explicar a estória a ser desenvolvida.
    2. Verificação (realizar testes em dupla e fazer a revisão de código).
- UX
    1. Explicar a experiência a ser desenvolvida.
    2. Validar roteiro de testes.
    3. Verificação (realizar testes funcionais exploratórios e passar o roteiro de testes).
- Dev
    1. Confirmar o entedimento da estória.
    2. Escrever roteiro de testes.
    3. Escrever testes automatizados.
    4. Iniciar desenvolvimento das estórias.
    5. Realizar verificação de qualidade de código utilizando o SonarQube ou outra ferramenta.
    6. Passar pelo roteiro de testes.
    7. Finalizar estória.
    8. Solicitar *Merge Request*.

As atividades que devem ser realizadas no Dti-flow para *back-end* estão descritas a seguir e a ordem de ocorrência está exemplificada na Figura 3.3.

- DL
    1. Explicar a estória a ser desenvolvida.
    2. Validar roteiro de testes.
    3. Verificação (realizar testes em dupla e fazer a revisão de código).

Figura 3.2 – Diagrama das etapas do Dti-flow (front-end)



Fonte: Adaptado de Dti Digital (2019)

- UX

    1. Explicar os serviços que a tela irá consumir.
    2. Verificação (realizar testes funcionais exploratórios e passar o roteiro de testes).

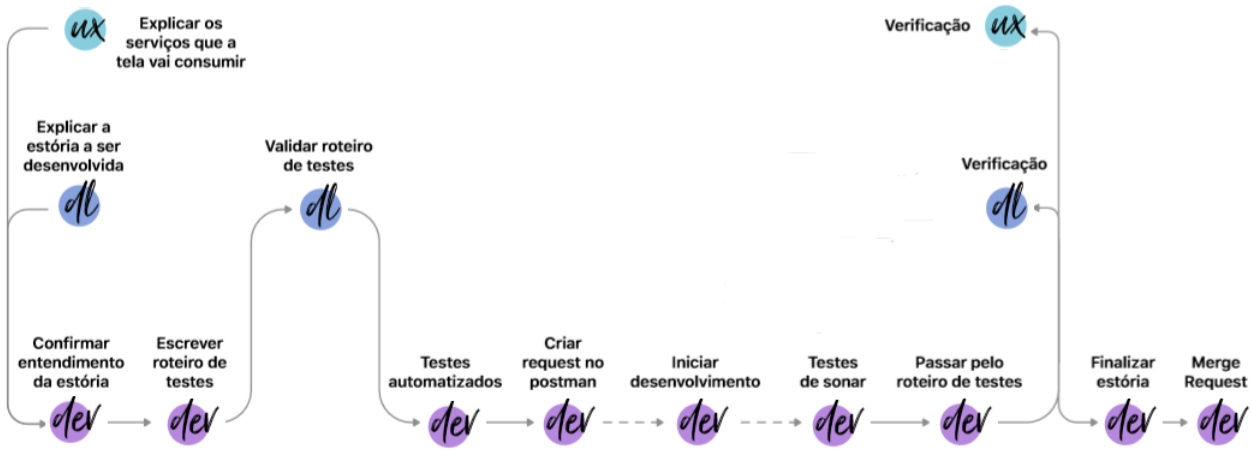
- Dev

    1. Confirmar o entedimento da estória.
    2. Escrever roteiro de testes.
    3. Escrever testes automatizados.
    4. Criar requisições no Postman ou ferramenta similar.
    5. Iniciar desenvolvimento das estórias.
    6. Realizar verificação de qualidade de código utilizando o SonarQube ou outra ferramenta.
    7. Passar pelo roteiro de testes.
    8. Finalizar estória.
    9. Solicitar *Merge Request*.

3.2.4 Gestão à vista

A Dti acredita que todos os colaboradores dentro da empresa devem ter acesso ao andamento de cada projeto, buscando manter uma boa organização no processo de desenvolvimento

Figura 3.3 – Diagrama das etapas do Dti-flow (back-end)



Fonte: Adaptado de Dti Digital (2019)

dos projetos, garantir a qualidade das soluções criadas e evitar o surgimento de problemas, como atrasos nas entregas. A metodologia de gestão à vista consiste em exatamente o que seu nome representa. O quadro de andamento das atividades, realização de ritos e *status* no qual o projeto se encontra devem estar sempre visíveis e acessíveis para todos. Por meio desse processo é possível identificar quais projetos não estão enfrentando problemas ou quais projetos precisam de atenção, pois apresentam riscos.

Além dos ritos pertencentes à metodologia SCRUM, outros ritos também são realizados durante a execução de um projeto, como a avaliação técnica e a avaliação de execução.

A avaliação técnica ou *check* arquitetural consiste em uma reunião na qual o responsável técnico pelo projeto, geralmente o desenvolvedor líder, apresenta o escopo macro a ser desenvolvido, buscando dar foco nos aspectos de risco, requisitos não funcionais e aspectos arquiteturais. Esta apresentação não é obrigatória, pois nem sempre é necessária, mas quando aplicável ao projeto ela deve ser executada. O indicado é que ela seja feita a outros profissionais com igual ou maior senioridade do apresentador, com o objetivo de sabatinar o responsável técnico, a fim de encontrar possíveis barreiras ou complicações técnicas e buscar ou revisar soluções para o escopo apresentado. É recomendado que esta avaliação seja feita em todo início de *sprint* ou quinzenalmente, caso a *sprint* possua mais de duas semanas de duração.

Na avaliação de execução ou *check* de execução, toda a equipe responsável pela construção do projeto deve ser sabatinada por um outro colaborador que não esteja envolvido diretamente com o projeto. A equipe apresenta o escopo do projeto e então é interrogada sobre o andamento do projeto. O objetivo dessa reunião é garantir que os processos apresentados no Dti-flow estão sendo seguidos, a fim de manter a qualidade do produto sendo desenvolvido e

identificar possíveis obstáculos à frente do andamento do projeto. Esta avaliação deve acontecer quinzenalmente ou ao final de cada *sprint*, independente do escopo do projeto.

3.3 Teste de Software

Conforme Pressman e Maxim (2016), teste de software consiste nas atividades definidas previamente que podem verificar e garantir que o software está sendo implementado de forma correta e funcionando conforme o esperado e planejado. O objetivo da realização dos testes é garantir a qualidade do produto desenvolvido e encontrar possíveis erros na aplicação. Para Crespo et al. (2004), o teste de software consiste basicamente no processo de executar o software de maneira controlada para verificar se o comportamento do mesmo será conforme o esperado.

Pressman e Maxim (2016) também ressaltam que o teste de software deve ser feito tanto em baixo nível, quanto em alto nível. O primeiro tem o objetivo de testar uma pequena porção do código, um componente específico ou componentes interligados, e o segundo consiste no teste mais amplo para verificar o funcionamento da aplicação conforme especificado pelo cliente.

3.3.1 Teste de unidade

De acordo com Maldonado et al. (2004), o principal foco do teste de unidade é encontrar erros de lógica e implementação em cada módulo do software. Pressman e Maxim (2016) reforçam que o teste de unidade deve concentrar em cada unidade de código, seja ela um componente, uma classe ou um objeto, garantindo seu funcionamento de forma correta como uma unidade.

3.3.2 Teste de integração

Como exposto por Maldonado et al. (2004), o teste funcional, também conhecido como teste de caixa-preta, busca validar as funcionalidades do software sem dar foco no código implementado, avaliando somente o resultado produzido de acordo com os dados de entrada.

3.3.3 Cenário de teste

Crespo et al. (2004) definem os cenários de teste como documentos que especificam quais funcionalidades serão testadas e as condições sobre as quais elas serão avaliadas, in-

cluindo os dados de entrada, resultados esperados e as ações que devem ser tomadas para disparar a funcionalidade a ser testada. Conforme Maldonado et al. (2004), é muito difícil utilizar de toda a base de dados e prever todos os casos em que a aplicação pode apresentar erros. Por isso, o objetivo dos casos de teste é descrever os fluxos a serem seguidos nos quais se tenha a maior probabilidade de se encontrar a maioria dos defeitos com o mínimo de tempo e esforço possível. Um teste bem sucedido é aquele que encontre a falha esperada para o caso em que foi testado.

3.4 Tecnologias utilizadas

Esta seção apresenta as tecnologias sobre as quais as aplicações que o estagiário atuou foram construídas e também descreve as ferramentas utilizadas para possibilitar o desenvolvimento das atividades durante o período de estágio.

São descritas as plataformas alvo do projeto, bem como as linguagens de programação com as quais os sistemas de software foram implementados, juntamente com as IDEs (*Integrated Development Environment*, do inglês) que apoiaram a escrita dos códigos fonte, e também os *frameworks* e bibliotecas que simplificaram o processo de criação das aplicações.

As ferramentas utilizadas para comunicação entre as equipes da própria Dti e equipes do cliente também são apresentadas, visto que o uso desses sistemas foi realizado diariamente no ambiente de trabalho, tendo como principal função a realização de vídeo chamadas e utilização do chat.

Uma definição das arquiteturas de software implementadas tanto nas aplicações para dispositivos móveis quanto nos Web Services desenvolvidos durante o estágio é apresentada ao longo desta seção.

3.4.1 Android

Conforme Lecheta (2013), o Android é uma plataforma *open-source* que conta com um sistema operacional baseado em Linux. A plataforma foi criada com o intuito de tornar o desenvolvimento de aplicações para aparelhos móveis mais simples e eficiente, buscando atender as exigências e expectativas do mercado. Além de ser mantido e desenvolvido pelo Google, a plataforma também conta com o apoio de um grupo chamado Open Handset Alliance (OHA), formado por diversas empresas do ramo de telefonia. De acordo com a documentação oficial (GOOGLE DEVELOPERS, 2019), a arquitetura da plataforma é dividida em uma pilha

de software, que contempla os seguintes componentes: aplicativos, *frameworks* e API Java, Bibliotecas Nativas em C/C++, Android Runtime (ART), Camada de Abstração de *Hardware* e o Kernel Linux.

3.4.2 Java

Segundo a documentação (ORACLE, 2019), Java é uma linguagem de programação de alto nível, multiplataforma e orientada a objetos. Ela é composta por dois componentes principais, que são a API (*Application Programming Interface*, do inglês) Java, uma biblioteca com comandos Java; e a JVM (*Java Virtual Machine*), que permite que a linguagem possa ser compilada e executada em várias plataformas. A linguagem conta com várias versões, como por exemplo a Java SE (*Standard Edition*), Java EE (*Enterprise Edition*), Java *Embbeded* e outras, cada qual focada para o desenvolvimento de soluções de software específicas, que vão desde aplicações para *desktop*, dispositivos móveis e embarcados, aplicações web até jogos.

3.4.3 Intellij IDEA e Android Studio

Ambos Intellij e Android Studio são ambientes de desenvolvimento integrados que oferecem inúmeras ferramentas aos desenvolvedores de software. O Intellij IDEA teve sua primeira versão lançada em janeiro de 2001. A IDE possui suporte para diversas linguagens de programação, *frameworks*, ferramentas, e por isso é utilizada para o desenvolvimento de projetos dos mais variados propósitos, contendo uma versão comunitária *open-source* e também uma versão comercial (KROCHMALSKI, 2014). A primeira versão estável do Android Studio foi lançada em dezembro de 2014 pela Google e sua criação foi baseada na versão comunitária do Intellij IDEA. Contudo, seu foco é o desenvolvimento para a plataforma Android (DRONGELEN, 2015).

3.4.4 Spring Boot

O Spring Boot é um *framework* para auxiliar no desenvolvimento de aplicações que utilizam a plataforma Java, buscando abstrair toda a parte de infraestrutura, possibilitando assim ao desenvolvedor concentrar-se na regra de negócio da aplicação. Conforme descrito por Walls (2016) e Johnson (2004), o Spring Boot faz parte de um conjunto de ferramentas disponibilizadas pelo *framework open-source* Spring, que surgiu como uma solução para simplificar o desenvolvimento de software. O Spring Boot baseia-se fundamentalmente em dois padrões de

projeto, que são: Inversão de Controle, o qual consiste na abstração para o desenvolvedor da criação dos componentes, início e fim da execução do programa, delegando essas atividades à um terceiro, geralmente um *framework*; e o padrão de Injeção de Dependência, que pode ser definido como a configuração de uma infraestrutura de software responsável por injetar em cada componente do software suas dependências (FOWLER, 2004).

3.4.5 Retrofit

Comumente as aplicações necessitam consumir serviços em APIs externas, e é exatamente isso que o Retrofit oferece às aplicações Android. Conforme Vogel, Scholz e Weiser (2018) explicam, o Retrofit é um cliente REST (*Representational State Transfer*, do inglês - padrão de arquitetura de software utilizado para a criação de *web services*) que permite às aplicações Android consumir e enviar dados através de operações por meio do protocolo HTTP (*HyperText Transfer Protocol*) . O formato dos dados nas requisições pode ser definido dentro da própria aplicação, sendo mais comum utilizar o formato JSON (*JavaScript Object Notation*), mas também permite utilizar conversores para o formato XML (*Extensible Markup Language*).

3.4.6 iOS

De acordo com Grønli et al. (2014), iOS é o sistema operacional que é executado nos celulares iPhone, no iPod e também iPad. Ele surgiu em 2007 com o lançamento do primeiro iPhone pela Apple, e primeiramente era o sistema operacional apenas para os *smartphones* da empresa. A plataforma foi muito importante, pois revolucionou o desenvolvimento de aparelhos e aplicativos em todo o mercado de telefonia. Tracy (2012) explica a composição da arquitetura do sistema operacional, a qual é dividida em quatro camadas organizadas da seguinte forma:

- Cocoa Touch - camada onde se encontram os *frameworks* para interação com a interface do aplicativo.
- Camada de Mídia - nesta camada se encontram as tecnologias de vídeo, áudio e gráfico.
- Core Services - contém os serviços fundamentais para o funcionamento das aplicações.
- Core OS - possui os *frameworks* para lidar com aspectos de baixo nível, como acesso à *hardware* ou questões mais técnicas de segurança.

3.4.7 Swift

De acordo com García et al. (2015), o Swift é uma linguagem de programação de propósito geral que foi criada pela Apple em 2014 com a finalidade de oferecer uma linguagem de programação mais simples, rápida, flexível e amigável aos desenvolvedores, buscando substituir a linguagem Objetive-C no desenvolvimento de aplicações para os sistemas operacionais iOS e OS X. O Swift possui tipagem forte, suporte a orientação a objetos e apresenta uma sintaxe com objetivo de ser clara e de fácil entendimento. É interessante citar também que a linguagem foi inspirada em linguagens como C++11, C#, F#, Go, JavaScript e outras.

3.4.8 Xcode

Outro ambiente de desenvolvimento integrado utilizado durante o desenvolvimento dos aplicativos foi a ferramenta Xcode, a IDE de software livre da Apple (2019). Ela é utilizada para o desenvolvimento e gerenciamento de projetos para todas as plataformas de aparelhos que possuem o sistema operacional Mac OS e iOS. O Xcode oferece suporte para as linguagens Objective-C, Apple-Script e Swift.

3.4.9 Alamofire

Similar à ferramenta Retrofit, o Alamofire também é uma biblioteca que permite às aplicações realizarem operações por meio do protocolo HTTP, contudo essa biblioteca é desenvolvida utilizando-se a linguagem de programação Swift e oferece uma alternativa à aplicações nas plataformas iOS e Mac OS de realizarem requisições à API externas utilizando também o formato de dados JSON. Conforme exposto por Kliffer (2018), o objetivo desta biblioteca, além de oferecer a possibilidade de realizar requisições REST, métodos de autenticação e outras funcionalidades, também está em tornar mais simples o processo de consumir os serviços de uma API externa, abstraindo os detalhes de configuração de operações HTTP do desenvolvedor.

3.4.10 Arquitetura MVC

Conforme descrito por Reenskaug (1979), um dos criadores do padrão MVC, a arquitetura *Model-View-Controller* (MVC) foi concebida no final dos anos 70 nos laboratórios da Xerox PARC. Inicialmente, a arquitetura se chamaria *Model-View-Editor*, mas posteriormente recebeu o nome que leva até hoje. O MVC foi concebido com o objetivo de auxiliar os de-

senvolvedores ao lidarem com um conjunto de dados muito grande e complexo. O propósito essencial da arquitetura é fornecer uma ponte entre a lacuna que existe entre o modelo mental humano e modelo digital no campo da computação.

Ramos (2015) acrescenta que o padrão de arquitetura de software MVC consiste na separação da aplicação em três camadas: a Model - responsável por representar e manipular os dados, seja na leitura, escrita ou até mesmo validação; a View - camada de interação com o usuário; e a Controller - responsável por lidar com todas as requisições do usuário.

3.4.11 Arquitetura MVP

O padrão de arquitetura MVP (*Model View Presenter*) é uma derivação do padrão MVC, também muito utilizado em projetos de software. Como retratado por Potel (1996), este padrão surgiu no início dos anos 90 e foi desenvolvido pela empresa Taligent, inicialmente usando a linguagem de programação C++ e depois migrado para Java. A arquitetura baseia-se nos três componentes que definem o nome do padrão, sendo a Model responsável por definir o modelo de dados que representará os objetos da aplicação; a View consiste na interface que exibe os dados e permite ao usuário interagir com o software; enquanto o papel do Presenter é atuar sobre tanto a camada de Model quanto a de View, recuperando os dados da primeira e controlando o que será exibido na segunda.

Comumente, na implementação desta arquitetura se utiliza um contrato, que nada mais é do que uma interface contendo a assinatura dos métodos da View e do Presenter. Desta forma, estas camadas precisam implementar os métodos definidos pelos seus contratos, o que permite estruturar o código em camadas e auxilia a evitar o acoplamento das classes.

Outra camada que costuma ser utilizada em conjunto com as já conhecidas Model, View e Presenter é a camada Interactor, que é responsável por trabalhar com os dados utilizados pela aplicação, geralmente através de requisições, seja por meio de serviços externos à aplicação ou acesso à uma base de dados. Segundo Ali (2017), o papel do Interactor é obter os dados e enviá-los para a Presenter, que irá processar esses dados e definir como eles serão exibidos.

3.4.12 Web Services

Comumente, aplicações em diferentes domínios e desenvolvidas em diversas tecnologias ou plataformas precisam se comunicar, ou seja, enviar e receber dados. Para isso, existem os Web Services, que permitem que a integração entre esses sistemas ocorra.

Conforme exposto por Newcomer e Lomow (2005), os Web Services são identificados por um URI (*Uniform Resource Identifier*, do inglês) e são baseados em protocolos e tecnologias padrões, como por exemplo o protocolo HTTP. Para consumir um Web Service é necessário ter conhecimento de seu identificador e os dados utilizados pelo recurso desejado. A requisição do serviço pode ser feita através dos Protocolos SOAP (*Simple Object Access Protocol*) ou REST (*Representational State Transfer*). O formato dos dados utilizados para a comunicação de aplicações através de Web Services podem ser tanto o padrão XML ou JSON.

3.4.13 Postman

Conforme descrito em seu site oficial, o Postman (2019) é uma plataforma para auxiliar no desenvolvimento de APIs e Web Services. A ferramenta permite realizar requisições nos padrões REST, SOAP e GraphQL, seja criando as consultas ou mesmo importando requisições já prontas.

3.4.14 VPN

Chin (1998) descreve a Rede Privada Virtual, ou *Virtual Private Network* (VPN) como túneis de comunicação criptografados entre pontos autorizados da rede. Como a transmissão dos dados é feita através da Internet, uma rede pública, os dados dentro da VPN devem estar protegidos de forma que não seja possível modificá-los ou interceptá-los. A VPN é comumente utilizada por corporações para fazerem conexão através da internet, pois assim podem realizar toda comunicação de forma segura mesmo que seus usuários ou colaboradores estejam remotos.

3.4.15 Microsoft Teams

O Microsoft Teams é uma ferramenta que auxilia na produtividade das equipes, oferecendo serviços como chats, vídeo-chamadas e chamadas apenas de áudio, além de ser integrado com outros serviços da Microsoft. De acordo com a documentação oficial (MICROSOFT, 2019), o Teams foi desenvolvido no mesmo ambiente da plataforma Office 365. Portanto, também confere acesso a outras ferramentas como o site SharePoint Online para armazenar arquivos, integração com o cliente de emails Outlook, além de outras ferramentas como por exemplo Power BI e Planner.

3.4.16 Slack

O Slack pode ser definido como uma plataforma de comunicação interna, permitindo aos times se comunicarem através de chat, chamadas de áudio e vídeo, compartilhar arquivos e mais. O objetivo da ferramenta é facilitar o processo de comunicação interno, o que por vezes pode se tornar complicado, como por exemplo ao se utilizar e-mail (PLUGA, 2016).

Como exposto pelo manual oficial do Slack (2019), a plataforma funciona através de canais de comunicação, os quais podem ser privados ou públicos. Também é permitido criar canais para equipes, para toda a empresa ou para um projeto específico.

3.4.17 Git e Gitlab

De acordo com o site oficial da ferramenta (GIT, 2019), o Git é um sistema de controle de versão *open-source* utilizado para auxiliar no controle de versões de projetos de qualquer porte. Em conjunto à utilização do git, também foi utilizado o software GitLab, o qual oferece funcionalidades tanto para o controle de versão de código, quanto o monitoramento de tarefas e atividades por meio de cards e listas, integração contínua, além de oferecer desenvolvimento nativo na nuvem por ser integrado com Kubernetes (BERTON, 2018).

3.4.18 SonarQube

O SonarQube (2019) é um revisor de código automático cujo objetivo é encontrar *bugs*, vulnerabilidades e *code smells* (trechos de código funcionais que podem apresentar problemas) no código fonte da aplicação. A ferramenta possui suporte para mais de 25 linguagens de programação e pode ser integrada a ferramentas de controle de versão para inspecionar a qualidade de todo código a ser adicionado ao repositório da aplicação em desenvolvimento.

3.4.19 Visual Studio Code

De acordo com a documentação oficial da ferramenta (VISUAL STUDIO CODE, 2019), o Visual Studio Code é um editor texto *open-source* que possui suporte para diversas linguagens de programação e oferece outras funcionalidades para edição de código fonte, como por exemplo um *debugger* para auxiliar na manutenção do código, uma ferramenta de controle de versão integrada, um terminal embutido dentro do próprio editor de texto, entre outras funcionalidades.

4 PROJETOS DESENVOLVIDOS

Este capítulo descreve o projeto em que o estagiário atuou durante o período de estágio, abordando aspectos como as tecnologias sobre as quais as aplicações foram contruídas, bem como as arquiteturas de software que foram escolhidas para estruturá-las.

4.1 Aplicativo Android e iOS

O projeto do qual o estagiário fez parte consiste em uma aplicação para dispositivos móveis cujo objetivo é oferecer aos clientes de uma rede bancária uma opção para realizar operações ordinárias referente a seus cartões, como por exemplo consultar o saldo disponível, solicitar aumento de limite de cartão, gerar boleto da fatura, entre outras funcionalidades. A aplicação também conta com funcionalidades básicas de configuração da conta digital do cliente, permitindo o mesmo a editar as informações do seu perfil.

O aplicativo foi desenvolvido tanto para a plataforma Android quanto para iOS, ambas desenvolvidas de forma nativa. Portanto, a primeira foi implementada utilizando-se a linguagem de programação Java enquanto a segunda foi criada através da linguagem Swift.

A arquitetura escolhida para estruturar as funcionalidades desenvolvidas pela equipe da Dti foi o padrão MVP, visto que esta arquitetura proporciona um alto desacoplamento entre classes, definindo bem a função de cada módulo do código, visando sempre manter a qualidade e facilitar no processo de manutenção da aplicação.

4.2 Web Service

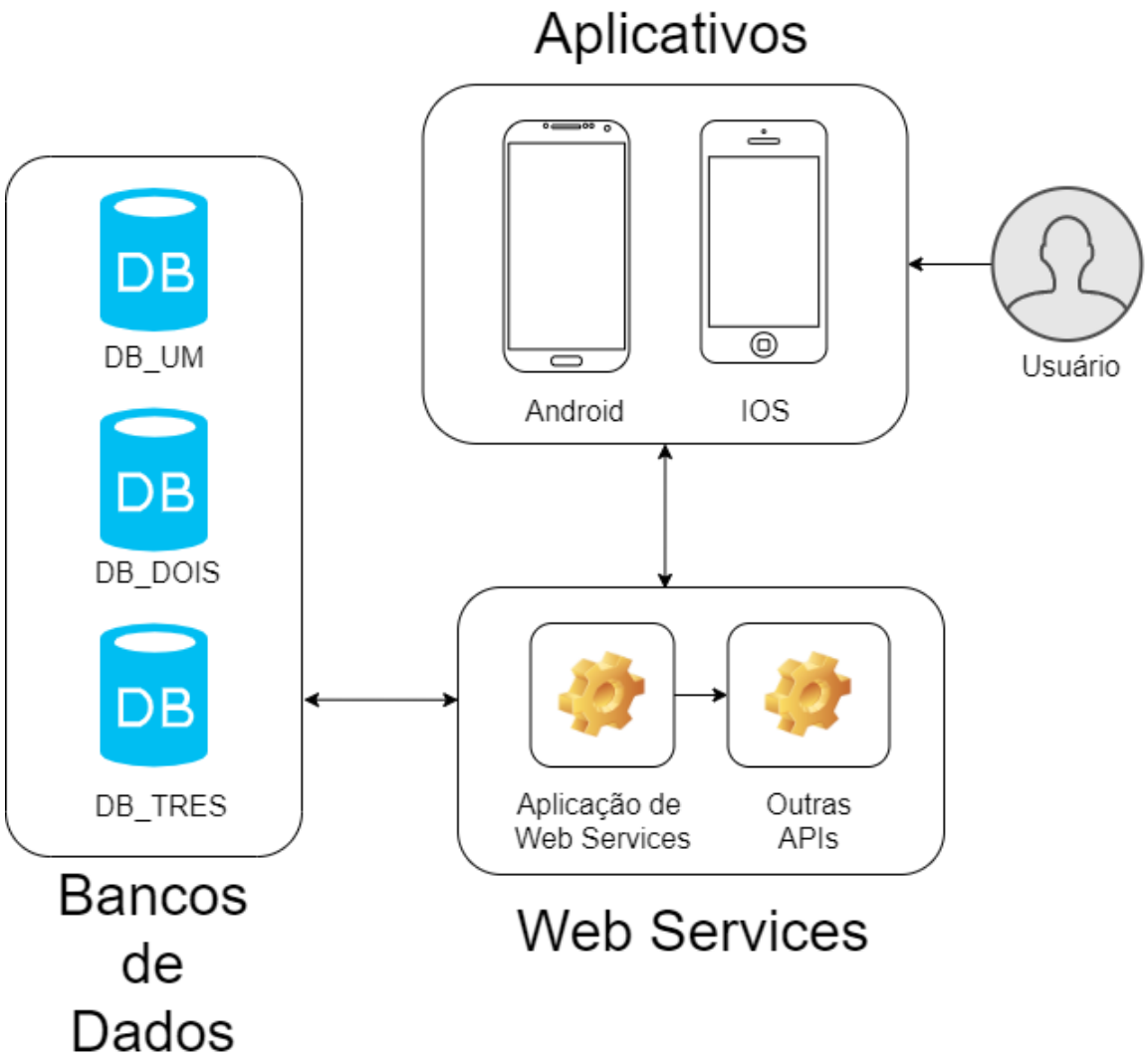
As aplicações móveis citadas anteriormente consomem diversos serviços externos, a fim de enviar e receber dados necessários para o funcionamento do aplicativo. Outro projeto no qual o estagiário atuou foi uma aplicação onde encontram-se vários *Web Services* consumidos pelo aplicativo.

A aplicação foi desenvolvida através da linguagem de programação Java em conjunto com o *framework* Spring Boot. A estruturação do código foi feita seguindo o padrão MVC, buscando manter o código organizado e cada camada da aplicação com uma única responsabilidade.

Apesar de ser consumido pela aplicação móvel, a aplicação de Web Services também realiza consultas a outros serviços externos, além de fazer acesso à bancos de dados para realizar

operações de persistência e consulta de dados. A Figura 4.1 ilustra a arquitetura da aplicação de uma forma mais ampla, apresentando seus componentes e conexões.

Figura 4.1 – Diagrama da arquitetura da aplicação



Fonte: Autor (2019)

5 ATIVIDADES DESENVOLVIDAS

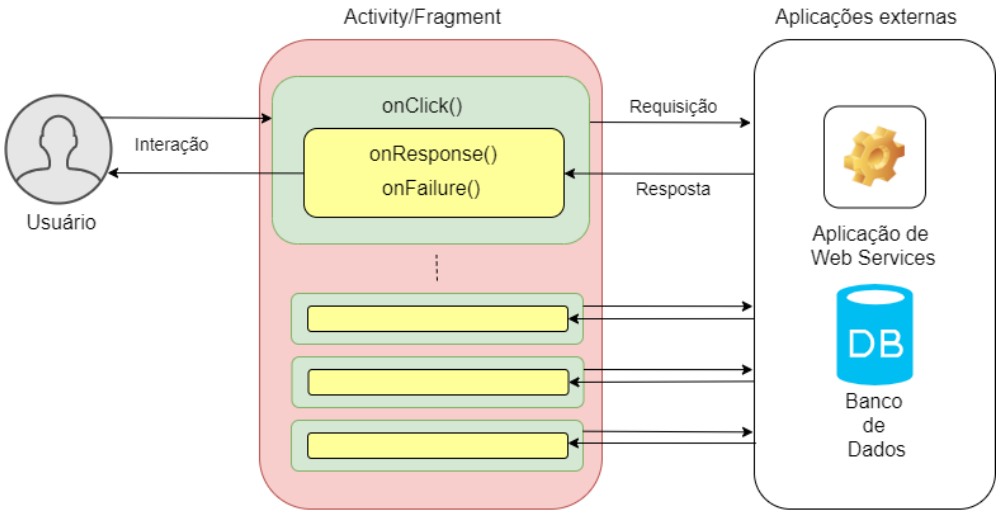
Neste capítulo são descritas as atividades realizadas pelo estagiário durante o período de estágio. É apresentada uma breve descrição sobre o escopo do projeto no qual o estagiário pode participar para um melhor entendimento das atividades desempenhadas pelo estagiário. As atividades estão divididas em seções, iniciando pela descrição do treinamento pelo qual o estagiário passou, e em seguida são expostas suas contribuições tanto no desenvolvimento dos aplicativos quanto na aplicação de *web services* e por fim é abordado a criação do roteiro de testes e a implementação dos testes de unidade para as funcionalidades desenvolvidas.

O escopo do projeto consistiu na sustentação do aplicativo citado no Capítulo 4, no qual a equipe atuou somente em atividades pontuais conforme especificado pelo cliente. Outra característica importante a ser citada é o fato da aplicação ser legada, ou seja, a equipe da Dti não participou de sua criação inicial.

As funcionalidades que seriam implementadas ou refatoradas pela equipe da Dti eram definidas pelo cliente e repassadas através de vídeo conferências, além de serem formalizadas por e-mail. Durante as reuniões de repasse eram levantadas possíveis dúvidas com relação às definições das atividades e também havia espaço para discussão e propostas de soluções que viessem a apresentar valor para o negócio do cliente. Após definidas as atividades, a equipe as analisava do ponto de vista técnico e o desenvolvedor líder definia a prioridade de cada atividade e as pessoas que iriam atuar em cada uma delas. Em seguida era dado início as implementações, sempre seguindo as etapas propostas pelo Dti-flow.

Um dos objetivos definidos para o projeto foi melhorar a performance do aplicativo, que apresentava baixo desempenho em algumas funcionalidades devido a diversas chamadas a serviços externos e uma implementação que não seguia um padrão de arquitetura de software, como ilustrado na Figura 5.1, cuminando em uma performance pobre e um código-fonte de difícil manutenção . Portanto, a maioria das atividades consistiram em refatoração de código, implementando o padrão de arquitetura MVP em diversas funcionalidades da aplicação. A criação da aplicação de *web services* foi uma ação tomada com a finalidade de concentrar o máximo de serviços possíveis consumidos pelos aplicativos em apenas uma localidade, buscando obter uma melhora no desempenho da aplicação móvel. A nova estrutura da aplicação pode ser observada na Figura 5.2.

Figura 5.1 – Estrutura da aplicação *mobile* antes da refatoração



Fonte: Autor (2019)

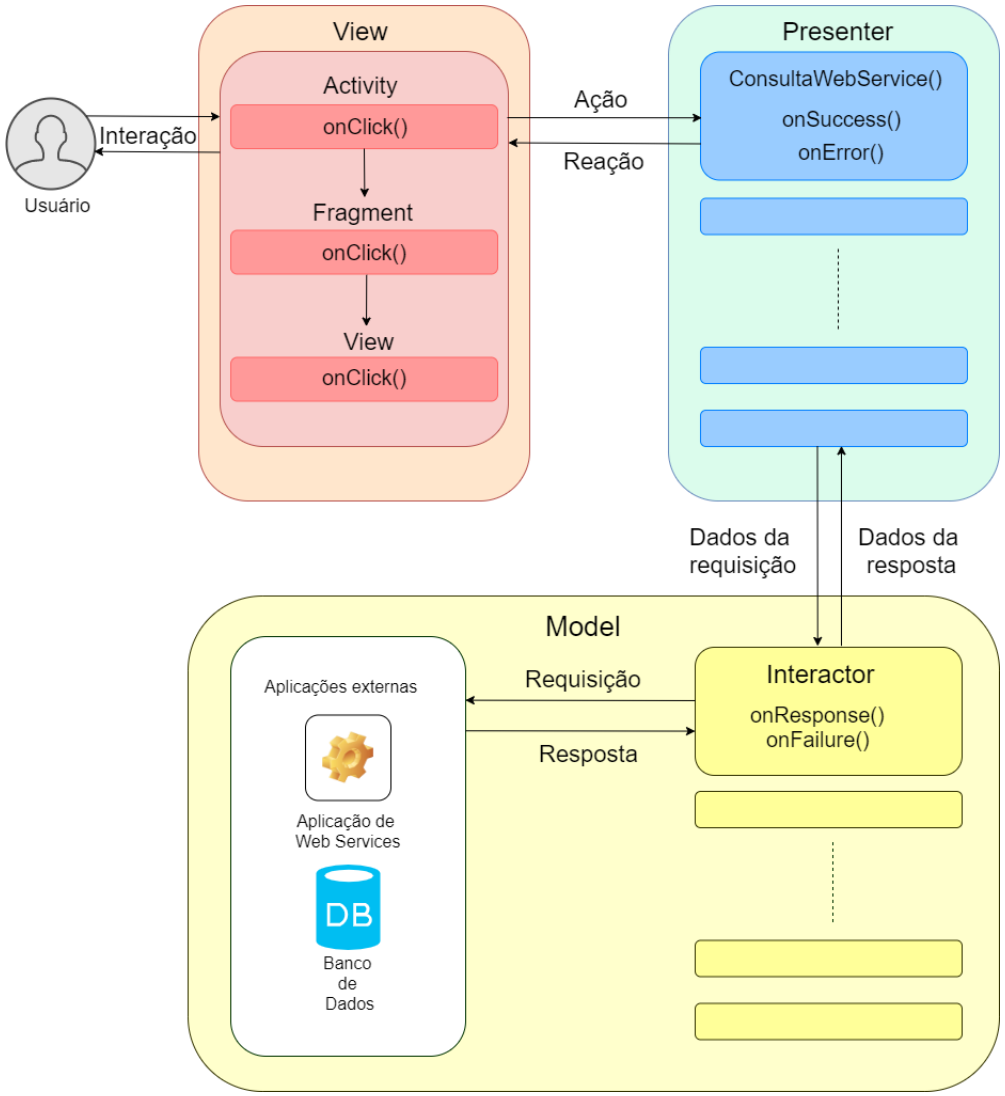
5.1 Treinamento

A primeira parte do estágio consistiu no treinamento, onde o estagiário teve que estudar sobre as tecnologias que seriam utilizadas nos projetos. Os estudos se deram em forma de pesquisa, através da leitura de vários artigos e documentações sobre os padrões de arquiteura de *software* que seriam implementados, as arquiteturas e catacterísticas de cada plataforma e os *frameworks* utilizados para auxiliar na implementação das aplicações. Também foram utilizados durante os estudos cursos *online* ofertados nas plataformas digitais Udemy e Udacity.

Durante o treinamento também foi apresentado ao estagiário os conceitos das metodologias e processos de desenvolvimento de *software* aplicadas nos projetos realizados na Dti Ditital, os quais foram descritos nas Seções 3.2.3 e 3.2.4.

Com a finalidade de fixar e colocar em prática os conhecimentos adquiridos durante a primeira etapa do estágio, o estagiário desenvolveu um projeto simples, porém que abordava diversos conceitos vistos durante o treinamento. O projeto consistiu na implementação de um aplicativo que possui uma tela de *login* com a função de salvar os dados inseridos pelo usuário para quando o mesmo quiser acessar a aplicação novamente. Também foi desenvolvida a validação dos dados de entrada, para certificar que o usuário possuía acesso à aplicação. Apesar de simples, todas as funcionalidades foram implementadas seguindo o padrão de arquitetura de *software* MVP e foram cobertas por testes de unidade, os quais seguiam os cenários descritos no documento de roteiro teste criado também pelo estagiário para testar e validar o funciona-

Figura 5.2 – Estrutura da aplicação *mobile* depois da refatoração



Fonte: Autor (2019)

mento da aplicação. Todos os processos realizados durante a criação do projeto de treinamento seguiram os conceitos do Dti-flow e da metodologia SCRUM.

Após finalizado o desenvolvimento do projeto de treinamento, o supervisor do estagiário realizou uma avaliação sobre a aplicação, onde foi possível encontrar possíveis melhorias, além de esclarecer dúvidas encontradas pelo estagiário durante o desenvolvimento e certificar se ele já estava apto para atuar em um projeto real.

5.2 Implementação de *Web Services*

Conforme citado anteriormente, a aplicação móvel possuia vários gargalos que causavam uma grande queda de desempenho. Em vista disso, após feita uma análise dos possíveis

motivos para essa performance pobre, foi detectado que a quantidade de serviços externos à aplicação consumidos de forma aninhada e síncrona era muito grande. A solução proposta para melhorar a performance diminuindo este gargalo, foi concentrar as chamadas aninhadas dos serviços externos em apenas um local, de forma que tais serviços pudessem ser executados com maior velocidade. Para tal foi desenvolvida uma aplicação contendo um conjunto de *web services* consumidos pelo aplicativo.

O estagiário pôde atuar na implementação de alguns desses serviços, onde teve uma grande contribuição principalmente no desenvolvimento de *web services* para a funcionalidade de solicitação de cartão virtual, além de outras, como por exemplo a solicitação de segunda via de boleto de fatura e atualização de senha da conta utilizada pelo usuário do aplicativo. Tais serviços foram implementados utilizando o padrão de arquitetura de *software* MVC com o auxílio do framework Spring Boot, permitindo dessa forma separar a responsabilidade de cada camada da aplicação (Model, View e Controller) evitando o acoplamento das classes e garantindo um código de qualidade que além de oferecer um ganho de desempenho também proporciona facilidade na manutenção.

5.3 Implementação das chamadas aos serviços nos aplicativos

A implementação da aplicação móvel, além de contar com o problema de possuir diversas consultas a serviços externos de forma aninhada, não seguia nenhum padrão de arquitetura de *software*. Após solucionado o primeiro problema através da criação da aplicação de *web services* descrita na seção anterior, foi necessário realizar as alterações no código do aplicativo. As mudanças consistiram na remoção dos serviços que foram tranferidos para os *web services* e na implementação das chamadas à esses serviços, contudo dessa vez utilizando a arquitetura de *software* MVP.

Dessa forma as funcionalidades refatoradas pela equipe da Dti passaram a possuir uma estrutura melhor organizada, onde apenas as interações com a tela do dispositivo móvel passaram a ficar na camada *View*, toda a parte de lógica se concentrou na camada *Presenter* e foi utilizada uma camada de *Interactor*, responsável por fazer toda a comunicação com os *web services*. Essa reestruturação resultou em aumento de performance e aumento da manutenibilidade do código.

O estagiário pôde atuar na reestruturação de algumas funcionalidades, dentre as que participou mais ativamente pode-se destacar as funcionalidades de alteração cadastral do cliente,

que permite ao usuário alterar dados como seu telefone, endereço residencial e e-mail, a atualização de senha da conta do aplicativo e também a função de solicitação de aumento do limite do cartão, esta última sendo a mais complexa, pois envolveu diversas validações e verificações referentes a ações fraudulentas.

5.4 Criação de roteiros de teste e implementação de testes de unidade

Conforme presente no Dti-flow, as etapas de criação de cenários de teste e implementação de testes de unidade são altamente recomendadas e fazem parte da definição de ”pronto” da atividade. Para todas as funcionalidades implementadas pela equipe da Dti durante o projeto foram desenvolvidos cenários de teste. Contudo, apenas a aplicação de web services contou com os teste de unidade.

O primeiro passo do desenvolvimento de uma atividade é a criação do cenário de teste, onde o responsável pela atividade deve descrever todas as ações dentro dos fluxos possíveis para a funcionalidade em questão e os resultados esperados após a realização de tais ações. As ações descritas nos cenários de teste precisam passar por uma validação feita pelo líder técnico da equipe, que dirá se o cenário está completo e correto ou se precisa de ajustes.

Os cenários de teste foram criados no formato de planilhas onde cada linha deveria representar uma ação dentro da funcionalidade. Cada ação possuía as colunas: ”descrição” - descrição da ação a ser realizada para causar o erro esperado; ”resultados esperados” - comportamento esperado da aplicação em função da ação realizada; ”*status* do desenvolvedor” - *status* do teste segundo o desenvolvedor responsável pela implementação; e ”*status* do desenvolvedor líder” - *status* do teste segundo avaliação do desenvolvedor líder, conforme mostra a Figura 5.3. Geralmente os valores para *status* do teste são "Aguardando teste"(teste ainda não realizado), "Aprovado"(obteve os resultados esperados) ou "Reprovado"(a ação não apresentou os resultados esperados).

Após finalizado o desenvolvimento da atividade, o membro da equipe responsável pela implementação executava os testes especificados no roteiro de teste. Depois que todos os testes estavam com status de desenvolvedor "Aprovado", o desenvolvedor líder os executava juntamente com o autor da implementação da estória a fim de confirmar se a funcionalidade não possuía realmente nenhum erro.

Os testes de unidade foram implementados somente nos serviços da aplicação de *web services*. Assim como estabelecido pelo Dti-flow, os testes unitários precisam obter uma co-

Figura 5.3 – Exemplo de roteiro de teste

Fluxo	Verificação do preenchimento do formulario		Aguardando Teste	
1	Descrição	Resultado Esperado	Status Desenvolvedor	Status Dev Líder
1.1	Tentar preencher o formulario sem apertar o botao "Alterar cadastro".	Não deve ser possível alterar os campos.	OK	OK
1.2	Clicar no botão "Alterar cadastro".	Deve habilitar os campos para ediçao e alterar a label do botão para "Salvar alterações".	OK	Aguardando Teste
1.3	Clicar no botão "Salvar alterações" sem preencher nenhum campo.	Nao deve realizar a alteração e deve exibir a mensagem de alerta "Nenhum campo alterado".	OK	Aguardando Teste

Fonte: Autor (2019)

bertura de código de pelo menos 80%. Para verificar a porcentagem de cobertura dos testes foi utilizada a ferramenta de análise de código SonarQube, descrita na Seção 3.4.18.

6 CONCLUSÃO

O estágio realizado na empresa Dti Digital como Trabalho de Conclusão de Curso para a obtenção do título de Bacharel em Ciência da Computação permitiu ao aluno de graduação ter a experiência de participar em um projeto de grande porte onde o mesmo pôde verificar os processos e etapas de construção de um *software* e também colocar em prática os conhecimentos adquiridos ao longo do curso.

A disciplina de Engenharia de Software foi fundamental no âmbito de entender os processos tradicionais de desenvolvimento de *software* e verificar na prática como e porque as novas metodologias surgiram como uma resposta à mudança pela qual o mercado digital está passando.

Os conhecimentos adquiridos nas disciplinas de Estrutura de Dados, Programação Orientada à Objetos e Banco de Dados puderam ser observados e utilizados na prática, sendo muito importantes pois contribuíram em muitas atividades desenvolvidas pelo estagiário.

Os conceitos apresentados na disciplina eletiva de Desenvolvimento de Aplicativos para Dispositivos Móveis foram amplamente utilizados, visto que o projeto no qual o estagiário atuou se trata de um aplicativo. A realização da disciplina se mostrou de grande valia pois apesar de apresentar apenas uma introdução ao desenvolvimento *mobile*, expôs ao aluno conceitos fundamentais que foram seguidos durante a implementação da aplicação para dispositivos móveis.

As disciplinas de Programação Paralela e Concorrente e Sistemas Distribuídos contribuíram para o melhor entendimento de conceitos chaves sobre o funcionamento de uma aplicação descentralizada. Esses conceitos puderam ser observados na integração do aplicativo com as outras aplicações de *Web Services* e bancos de dados.

Enquanto as disciplinas acima citadas contribuíram no desenvolvimento das aplicações, a disciplina de Interação Humano-Computador proporcionou ao estagiário uma visão mais crítica sob o ponto de vista de um usuário, fornecendo conceitos que possibilitaram identificar problemas de usabilidade e apresentar possíveis soluções de forma a proporcionar ao usuário uma boa experiência ao utilizar o aplicativo.

Além dos conceitos aprendidos durante o curso que puderam ser observados e aplicados dentro do projeto, foi possível adquirir um grande conhecimento durante o estágio com relação às ferramentas técnicas utilizadas para o desenvolvimento de um *software*. Foi possível observar também como a comunicação é um fator fundamental para o sucesso de um projeto. Tanto a

comunicação interna com os membros da equipe quanto a comunicação externa com o cliente deve ser realizada de forma que todos estejam sempre alinhados com o objetivo do projeto.

Outro ponto importante a se destacar foi a melhoria do código fonte gerado pela refatoração implementada pela equipe da Dti. Após a refatoração o código ficou com uma melhor organização e estruturação, o que ocasionou o aumento de performance do aplicativo, além de garantir a legibilidade e manutenibilidade do código. Com isso pôde-se concluir também como um planejamento e estudo prévio sobre o padrão de arquitetura de *software* a ser implementado na aplicação é fundamental para se desenvolver um aplicativo de qualidade, com alta performance e buscando mitigar o máximo de possíveis problemas.

REFERÊNCIAS

ALI, J. **Android MVC Architecture Extension with Interactors and Repositories**. 2017. Disponível online em: <https://blog.mindorks.com/android-mvp-architecture-extension-with-interactors-and-repositories-bd4b51972339>. Último acesso em 22 de outubro de 2019. Mindorks.

APPLE. **Xcode**. 2019. Disponível online em: <https://developer.apple.com/xcode/ide/>. Último acesso em 27 de outubro de 2019. Apple Developer.

BECK, M. B. K. et al. **Manifesto para o desenvolvimento ágil de software**. 2001. Disponível online em: <https://www.manifestoagil.com.br/index.html>. Último acesso em 4 de novembro de 2019. Agile Alliance.

BERTON, G. **Git, GitHub e GitLab: o que são e qual quais os ambientes de produtividade da programação**. 2018. Disponível online em: <https://introduceti.com.br/blog/git-github-e-gitlab-o-que-sao-e-qual-quais-os-ambientes-de-produtividade-da-programacao/>. Último acesso em 27 de outubro 2019. Introduce - tecnologia para crescer.

BISSI, W. Metodologia de desenvolvimento ágil. **Campo Digital**, v. 2, n. 1, 2007.

BUDIU, R. **Mobile: Native Apps, Web Apps, and Hybrid Apps.** 2013. Disponível online em: <https://www.nngroup.com/articles/mobile-native-apps/>. Último acesso em 19 de novembro de 2019. Nielsen Norman Group.

CHIN, L. K. Rede privada virtual–vpn. **Rede Nacional de Ensino e Pesquisa (RNP)**, 1998.

COMPILA. **Apps corporativos: por que mais e mais empresas criam aplicativos?** 2016. Disponível online em: <http://blog.compila.com.br/apps-corporativos/>. Último acesso em 11 de dezembro de 2019. Compila Soluções em Tecnologia.

CRESPO, A. N. et al. Uma metodologia para teste de Software no Contexto da Melhoria de Processo. **Simpósio Brasileiro de Qualidade de Software**, p. 271–285, 2004.

DRONGELEN, M. V. **Android studio cookbook**. [S.l.]: Packt Publishing Ltd, 2015.

DTI DIGITAL. **Fluxograma do processo do dti-flow.** 2019. Documento interno da empresa. Fornecido em comunicação pessoal com a gerência em 30/10/2019.

FOWLER, M. **Inversion of Control Containers and the Dependency Injection pattern.** 2004. Disponível online em: <https://martinfowler.com/articles/injection.html#FormsOfDependencyInjection>. Último acesso em 15 de novembro de 2019. Martin Fowler.

GARCÍA, C. G. et al. Swift vs. objective-c: A new programming language. **IJIMAI**, Universidad Internacional de La Rioja, v. 3, n. 3, p. 74–81, 2015.

GIT. **Git Documentation**. 2019. Disponível online em: <https://git-scm.com/>. Último acesso em 5 de novembro de 2019. Git.

GOOGLE DEVELOPERS. **Arquitetura da plataforma**. 2019. Disponível online em: <https://developer.android.com/guide/platform>. Último acesso em 21 de outubro de 2019. Android Developers.

GRØNLI, T.-M. et al. Mobile application platform heterogeneity: Android vs windows phone vs ios vs firefox os. In: IEEE. **2014 IEEE 28th International Conference on Advanced Information Networking and Applications**. [S.l.], 2014. p. 635–641.

JOHNSON, R. **Expert one-on-one J2EE design and development**. [S.l.]: John Wiley & Sons, 2004.

KLIFFER, R. **Alamofire Tutorial: Getting Started**. 2018. Disponível online em: <raywenderlich.com/35-alamofire-tutorial-getting-started>. Último acesso em 5 de novembro de 2019.

KROCHMALSKI, J. **IntelliJ IDEA Essentials**. [S.l.]: Packt Publishing Ltd, 2014.

LECHETA, R. R. **Google Android-3ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. [S.l.]: Novatec Editora, 2013.

MALDONADO, J. C. et al. **Introduçao ao teste de software**. 2004. 23 p. Nota Didática No. 65, ICMC-USP.

MICROSOFT. **Documentação técnica do Microsoft Teams**. 2019. Disponível online em: <https://docs.microsoft.com/pt-br/microsoftteams/microsoft-teams>. Último acesso em 28 de outubro de 2019. Microsoft.

NEWCOMER, E.; LOMOW, G. **Understanding SOA with Web services**. [S.l.]: Addison-Wesley, 2005.

ORACLE. **The Java Programming Language and the Java Platform**. 2019. Disponível online em: <https://www.oracle.com/technetwork/topics/newtojava/downloads/index.html>. Último acesso em 5 de novembro de 2019. Java Docs.

PLUGA. **O que é Slack? Menos emails e mais integração entre equipes!** 2016. Disponível online em: <https://pluga.co/blog/api/o-que-e-slack/>. Último acesso em 26 de outubro de 2019. Pluga.co.

POSTMAN. **Postman - What is Postman**. 2019. Disponível online em: <https://www.getpostman.com/>. Último acesso em 27 de outubro de 2019. Postman.

POTEL, M. MVP: Model-View-Presenter the Taligent programming model for C++ and Java. **Taligent Inc**, p. 20, 1996.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [S.l.]: McGraw Hill Brasil, 2016.

RAMOS, A. **O que é MVC?** 2015. Disponível online em: <https://tableless.com.br/mvc-afinal-e-o-que/>. Último acesso em 24 de outubro de 2019. Tableless.

REENSKAUG, T. M. H. The original mvc reports. 1979.

SABBAGH, R. **Scrum: Gestão ágil para projetos de sucesso**. [S.l.]: Editora Casa do Código, 2014.

SLACK. **O que é Slack?** 2019. Disponível online em: <https://slack.com/intl/pt-br/help/articles/115004071768>. Último acesso em 5 de novembro de 2019. Slack.

SOARES, M. dos S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 2004.

SONARQUBE. **SonarQube - Documentation (Docs 8.0)**. 2019. Disponível online em: <https://docs.sonarqube.org/latest/>. Último acesso em 31 de outubro de 2019. SonarQube Docs.

TRACY, K. W. Mobile application development experiences on apple¿ s ios and android os. **Ieee Potentials**, IEEE, v. 31, n. 4, p. 30–34, 2012.

VISUAL STUDIO CODE. **Visual Studio Code Docs**. 2019. Disponível online em: <https://code.visualstudio.com/docs>. Último acesso em 27 de outubro de 2019. Visual Studio Code.

VOGEL, L.; SCHOLZ, S.; WEISER, D. **Using Retrofit 2.x as REST client - Tutorial**. 2018. Disponível online em: <https://www.vogella.com/tutorials/Retrofit/article.html#what-is-retrofit>. Último acesso em 5 de novembro de 2019. Vogella.

WALLS, C. **Spring Boot in action**. [S.l.]: Manning Publications, 2016.