GUILHERME BARBOSA OCHIKUBO

RELATÓRIO DE ESTÁGIO NA EMPRESA EQUALS - GESTÃO FINANCEIRA INTELIGENTE

LAVRAS – MG

2021

GUILHERME BARBOSA OCHIKUBO

RELATÓRIO DE ESTÁGIO NA EMPRESA EQUALS - GESTÃO FINANCEIRA INTELIGENTE

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

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG

2021

GUILHERME BARBOSA OCHIKUBO

RELATÓRIO DE ESTÁGIO NA EMPRESA EQUALS - GESTÃO FINANCEIRA INTELIGENTE

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

APROVADA em 09 de Novembro de 2021.

Prof. Dr. Maurício Ronny de Almeida Souza	UFLA
Profa. Dr. Renata Teles Moreira	UFLA
Profa. Dr. Juliana Galvani Greghi	UFLA

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG
2021

AGRADECIMENTOS

Agradeço à minha família por todo amor e apoio durante toda a minha vida. Aos amigos, que sempre estiveram ao meu lado, pela amizade incondicional e pelo apoio demonstrado ao longo de todo o período de tempo em que me dediquei a este trabalho.

Agradeço ao professor Maurício por todo o apoio durante este trabalho e também as professoras Renata e Juliana pela participação na banca e suas recomendações para este trabalho.

RESUMO

Este relatório tem como objetivo descrever as experiências obtidas dentro da empresa Equals durante o período de estágio supervisionado. A Equals é uma empresa de gestão financeira e durante este período o estagiário atuou como desenvolvedor de software em aplicações voltadas para web, ao longo deste relatório será descrito as tecnologias utilizadas, metodologias e atividades desenvolvidas durante o estágio.

Palavras-chave: Estágio supervisionado. Desenvolvimento web.

ABSTRACT

This report is intended to describe experience within the Equals company during the supervised internship period. Equals is a financial management company and during this period the intern worked as a software developer in web-oriented applications, throughout this report the technologies used, methodologies and activities developed during the internship will be described.

Keywords: Supervised internship. Web development.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este documento tem como objetivo descrever as atividades desenvolvidas pelo autor durante o estágio supervisionado dentro da empresa Equals ao longo do curso de ciências da computação na Universidade Federal de Lavras (UFLA).

O curso de Bacharelado em Ciência da Computação na Universidade Federal de Lavras possui duração de oito semestres e tem como objetivo capacitar o aluno a projetar, desenvolver e gerenciar sistemas computacionais. Assim, ao longo deste documento será correlacionado as matérias vistas em sala de aula com o que era praticado no cotidiano da empresa ao decorrer do estágio.

O estágio supervisionado teve duração de um ano, durante esse período o estagiário atuou como desenvolvedor de software na equipe de Web/API. Ao longo deste documento o estagiário irá descrever sobre a empresa Equals, as principais atividades desenvolvidas e tecnologias utilizadas ao longo do estágio.

Os capítulos posteriores a este estão organizados de forma que, o capítulo dois descreve sobre a empresa, o capítulo três contém a fundamentação teórica necessária para o entendimento deste documento, o capítulo quatro discorre sobre as atividades desenvolvidas ao longo do estágio e por fim o capítulo cinco contém a conclusão.

2 SOBRE A EMPRESA

A Equals é uma *fintech* [1] fundada em 2010, que ajuda outras empresas a controlarem suas vendas de cartão de crédito e débito, ela está situada no município de São Paulo-SP, e o polo tecnológico está situado na cidade de Lavras-MG.

A empresa é especializada em gestão e conciliação de vendas [2] com cartões de crédito, débito, boletos e outros meios de pagamento *online*. Os produtos oferecidos pela Equals tem como objetivo otimizar processos para ambientes de pagamento complexos e com grande volume de vendas, ajudando os lojistas a comparar controle financeiro interno com tudo o que foi lançado no extrato bancário em um determinado período de tempo.

Os principais produtos oferecidos pela empresa são o Equals Core que é uma aplicação de gestão financeira voltada para os grandes *players* do mercado com um alto volume de vendas e o Raio-x que é voltado para lojistas menores. Além desses dois produtos também há o portal interno utilizado pelos funcionários denominado retaguarda.

A plataforma Equals Core consiste em uma plataforma para os principais clientes do Equals. É através dela que os clientes consultam suas informações de vendas e conciliação por meio de diversos relatórios.

A retaguarda consiste no portal web utilizado internamente pelos funcionários da Equals. É o local onde é realizado o cadastro de clientes, das funcionalidades, dos contratos das adquirentes, dos operadores, entre outros. Os sub tópicos desta seção descrevem algumas das atividades desenvolvidas dentro de ambas as plataformas da empresa.

Dentro da empresa, os funcionários são divididos em equipes, também chamadas de times. O estagiário foi designado para o time de Web/API. Este time é responsável pela conciliação de vendas de cartões dos clientes e também pela plataforma web por onde os clientes consultam suas informações de vendas, pagamento, recebimentos e conciliações por meio de diversos relatórios. O time de Web/API também é responsável pelo portal interno, chamado de Retaguarda.

A equipe de Web/API tem como principais atividades garantir a conciliação das vendas de cartões dos clientes, criar novas funcionalidades e relatórios no sistema, realizar manutenções no sistema e prover suporte técnico de segundo nível aos clientes e times internos.

---

[1] Empresa de tecnologia que atua no ramo financeiro.

[2] Verificação entre as taxas contratadas e as taxas praticadas pela adquirente.

3 CONCEITOS E TECNOLOGIAS UTILIZADAS

Este capítulo apresenta a fundamentação teórica necessária para o entendimento dos conceitos utilizados neste relatório. Mais especificamente, este capítulo apresenta conceitos relacionados a desenvolvimento web, banco de dados, modelagem de software, metodologias ágeis e seu uso dentro do contexto de engenharia de software.

3.1 Desenvolvimento Web

O desenvolvimento web é a nomenclatura utilizada para descrever o processo de desenvolvimento de *sites*. Um site pode ser dividido em *front end* e *back end*. Pode-se definir *front end* como qualquer aspecto do processo de *design* que aparece ou está diretamente relacionado ao navegador (ROBBINS, 2012).

Butterfield, Ngondi e Kerr (2016) definem *back end* como a parte de um aplicativo que armazena e manipula dados. Ele recebe suas instruções e entradas e envia sua saída para o *front end* da aplicação, que é responsável por interagir com o usuário.

Ainda dentro do contexto de desenvolvimento Web, princípios como SOLID eram utilizados no cotidiano da equipe com o objeto de se manter um bom design de projeto. De maneira resumida, Martin (2014) define os cinco princípios do SOLID como:

- *The Single Responsibility Principle*: Uma classe deve ter um, e apenas um, motivo para mudar.
- *The Open Closed Principle*: Objetos ou entidades devem ser abertos para extensão, mas fechados para modificação.
- *The Liskov Substitution Principle*: Uma classe derivada ou subclasse deve ser substituivel por sua classe base.
- *The Interface Segregation Principle*: Uma classe não deve ser forçada a implementar interfaces e métodos que não irá utilizar.
- *The Dependency Inversion Principle*: Dependa de abstrações, não de implementações.

Tais princípios eram utilizados dentro do projeto para que o software se tornasse escalável e também flexível, deixando o projeto mais robusto e facilitando a implementação de novas funcionalidades e manutenção do sistema.

3.2 Metodologias Ágeis

A metodologia ágil é uma forma de gerenciamento de projetos proposto por Beck et al. (2001), também conhecida como manifesto ágil, que busca uma maior agilidade nos processos e nas entregas das tarefas sem comprometer a qualidade do produto. Segundo Valente (2020) o método ágil consiste em conseguir avançar, mesmo em ambientes com informações imperfeitas, parciais e sujeitas a mudanças.

Dessa forma, surgiram também *frameworks* como o Scrum, que se baseia nos princípios e valores expressos no manifesto ágil e busca gerar valor por meio de soluções adaptativas. De acordo com Schwaber e Sutherland (2020), o Scrum é baseado na afirmação de que o conhecimento vem da experiência e também da tomada de decisões com base no que é observado, além de focar em reduzir o risco e se concentrar no que é essencial.

O *framework* Scrum é organizado em papéis, artefatos e eventos. Os papéis do Scrum são: o *Product Owner*, o *Scrum Master* e o Time. Pinheiro (2017) define o *Product Owner* como o dono do produto, sendo aquele responsável por maximizar o valor do produto e do trabalho do time de desenvolvimento. A forma na qual ele irá fazer isso pode variar amplamente de acordo com as organizações, times e indivíduos. Além disso, o *Product Owner* é a única pessoa responsável por gerenciar o *Product Backlog*.

O *Product Backlog* é uma lista ordenada de tudo que deve ser necessário no produto, ele lista todas as características, funções, requisitos, melhorias e correções que formam as mudanças que devem ser feitas no produto nas versões futuras (SCHWABER; SUTHERLAND, 2020).

O *Scrum Master* é um líder servidor para o time Scrum. Sua responsabilidade é garantir que o Scrum seja entendido e aplicado, garantindo que o time Scrum aderiu às teorias, práticas e regras do Scrum (PINHEIRO, 2017). Schwaber e Sutherland (2020) afirma que o time consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto ao final de cada *Sprint*.

Os eventos ou cerimônias do Scrum são: a *Sprint*, a *Sprint Planning*, a *Sprint Review*, a *Sprint Retrospective* e as *Daily Meetings*. Conforme (REHKOPF, 2021) uma *sprint* pode ser definida como um período curto e fixo de tempo em que uma equipe trabalha para concluir uma quantidade definida de trabalho, sendo o cerne das metodologias Scrum. O ideal é que todas as *Sprints* ocorram em um

intervalo de tempo predeterminado, porém dentro da equipe de Web/API esse intervalo era variável de acordo com a quantidade de tarefas a serem realizadas em cada *Sprint*.

O *Sprint Planning* é um evento Scrum que ocorre juntamente com toda a equipe e ocorre sempre antes do início de uma nova *Sprint*. O objetivo do *Sprint Planning* é definir o que será entregue na *Sprint* e como esse trabalho vai ser alcançado (WEST, 2021).

Como afirma Schwaber e Sutherland (2020) a *Sprint Review* é uma reunião que acontece no final da Sprint para inspecionar o incremento e adaptar o *Product Backlog* se necessário. Durante a reunião de Revisão da *Sprint* o Time Scrum e as partes interessadas colaboram sobre o que foi feito na *Sprint*.

O propósito da *Sprint Retrospective* é planejar maneiras de aumentar a qualidade e a eficácia. Para isso a equipe inspeciona como foi a última *Sprint*, discutindo o que deu certo, quais problemas foram encontrados e como esses problemas foram (ou não) resolvidos, de forma a melhorar a próxima *Sprint* (SCHWABER; SUTHERLAND, 2020). Conforme Schwaber e Sutherland (2020) a *Daily Meeting* é uma reunião de 15 minutos que tem como objetivo inspecionar o progresso em direção a meta da *Sprint* e adaptar o *Sprint Backlog* conforme o necessário, ajustando o próximo trabalho planejado.

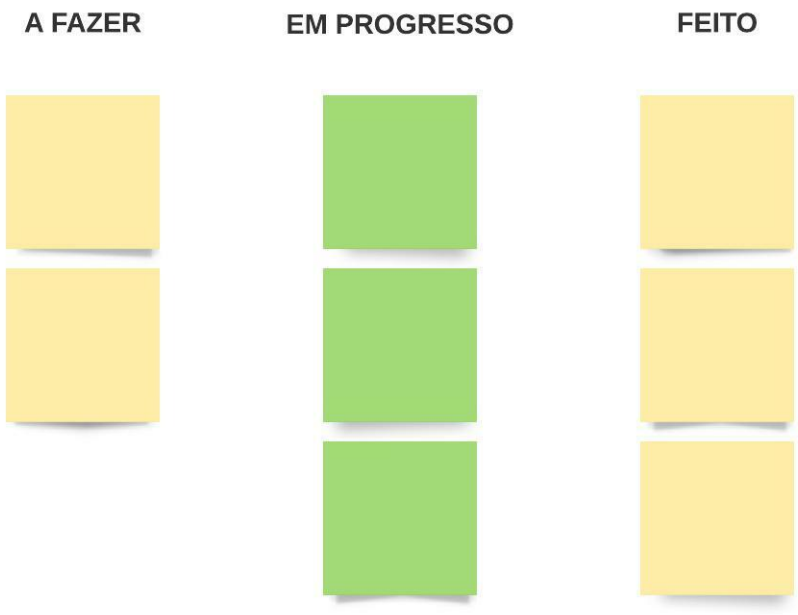
O Kanban é um quadro visual para controle de fluxo e gestão de tarefas. Segundo Mariotti (2012) o modelo Kanban, com seu mecanismo de sinalização, tem como objetivo apresentar uma atividade de trabalho em processo, ou seja, o número de atividades ou cartões em circulação é equivalente à capacidade do sistema. A Figura 3.1 representa um quadro Kanban simples.

A coluna A Fazer representa as tarefas que ainda não foram iniciadas, a coluna Em Progresso representa as tarefas que estão sendo realizadas e por fim a coluna Feito representa as tarefas que já foram finalizadas.

Durante o processo de desenvolvimento das atividades do time, a metodologia Scrum e Kanban eram utilizadas. Todos os dias durante a semana acontecia a *Daily Meeting*, onde todos os membros do time participavam e davam o parecer sobre o andamento das suas atividades atuais.

A *Sprint* tinha duração média de um mês, podendo variar dependendo do tamanho e nível de dificuldade das tarefas a serem desenvolvidas. O ideal é que todas as *Sprints* ocorram em um intervalo de tempo predeterminado, porém dentro da equipe de Web/API esse intervalo era variável de acordo com a quantidade de tarefas a serem realizadas em cada *Sprint*. Antes do início da *Sprint* acontecia a *Sprint Planning* com os desenvolvedores, analistas de qualidade e *Product Owner*, nessa reunião era

Figura 3.1 – Exemplo quadro Kanban



Fonte: Autoria própria

alinhado todos os pontos serem desenvolvidos e realizadas as estimativas de desenvolvimento, revisão e teste para cada atividade.

Ao final de cada *Sprint*, acontecia a *Sprint Retrospective*, na qual era discutido os pontos positivos e negativos que aconteceram durante o desenvolvimento da *Sprint*. Nesta reunião também era identificado como melhorar os pontos negativos definindo membros do time para ficarem responsáveis pela tentativa de melhora de algum ponto negativo durante a próxima *Sprint*.

3.3 Banco de Dados

De acordo com Oracle (2021) um banco de dados pode ser definido como uma coleção organizada de dados ou informações estruturadas, normalmente armazenado em um sistema de computador e controlado por um sistema de gerenciamento de banco de dados. Conforme Wade e Chamberlin (2012), para facilitar a interação com o banco de dados a linguagem SQL (*Structured Query Language*) foi criada. Ela é usada para executar comandos em bancos de dados relacionais.

Ainda dentro da plataforma Equals Core, alguns conceitos como *stored procedures*, *views* e *object relational mapper* eram utilizados. Um *stored procedure* é um conjunto de código SQL que pode ser salvo, para que possa ser reutilizado continuamente, sendo possível passar parâmetros para o *stored procedure* de modo que ele possa agir como uma função utilizando os valores dos parâmetros passados (W3SCHOOL, 2021).

Uma *view* pode ser determinada como uma tabela virtual, pois não necessariamente existe de forma física. Elmasri e Navathe (2019) explica que isso limita as possíveis operações de atualização que podem ser aplicadas as views, mas não oferece quaisquer limitações sobre a consulta em uma *view*, o que nos faz pensar em uma view como um modo de especificar uma tabela que precisamos referenciar com frequência, embora ela possa não existir fisicamente

ORM (*Object Relational Mapper*) é uma técnica utilizada na programação orientada a objetos que permite fazer uma relação dos objetos com os dados que os mesmos representam (DEVMEDIA, 2011). Dentro deste o contexto, o Mybatis é uma estrutura de persistência com suporte para SQL personalizado, procedimentos armazenados e mapeamentos avançados. Com o Mybatis é possível eliminar quase todo o código JDBC (*Java Database Connectivity*) e configuração manual de parâmetros usando xml simples ou annotations para configuração (MYBATIS, 2021).

3.4 Ferramentas de Apoio

Dentro do projeto no qual o estagiário atuou, algumas ferramentas de eram utilizadas, dentre elas as principais eram Git e Jira. O Git [1] é um sistema de controle de versão. Um sistema de controle de versão é um software projetado para acompanhar as alterações feitas nos arquivos ao longo do tempo. Mais especificamente, Git é um sistema de controle de versão distribuído, o que significa que todos que trabalham com um projeto em Git têm uma cópia do histórico completo do projeto, não apenas o estado atual dos arquivos (BELL; BEER, 2014).

Ainda dentro do contexto de Git, alguns conceitos como *branches*, *commits*, *merge* e *rebase* eram utilizados. Chacon e Straub (2014) definem uma *branch* como sendo um ponteiro móvel para um determinado *commit*, onde um *commit* representa o estado do projeto em um determinado ponto no tempo. Chacon e Straub (2014) ainda definem que o *merge* é o processo de mesclagem entre dois ou mais históricos de desenvolvimento. Já o *git rebase* é o processo de alterar a base da *branch* do *commit*

[1] https://git-scm.com/

para outra, fazendo parecer que você criou a *branch* a partir de um *commit* diferente (ATLASSIAN, 2021).

Para o monitoramento da tarefas dentro da equipe a ferramenta Jira era utilizada. De acordo com Atlassian (2021) Jira é uma ferramenta de gestão de projetos, através dela é possível criar quadros que permitem que times gerenciem seu trabalho utilizando metodologias como Scrum ou Kanban. Como é possível observa na Figura 4.1, cada tarefa a ser realizada pela equipe era representada por um *card*. Um *card* é um item no quadro Kanban da ferramenta Jira que contém informações como uma breve descrição da tarefa, código da tarefa, proprietário e status do item de trabalho.

A linguagem Java era utilizada no *back end* da aplicação, Java é uma linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems, sendo capaz de criar aplicativos para desktop, aplicações comerciais, aplicativos para a Web e softwares robustos (CLARO; SOBRAL, 2015). Schildt (2015) diz que a internet ajudou a impulsionar a linguagem Java para a dianteira da programação e assim java teve um efeito profundo sobre a internet, simplificando a programação web, em geral, e resolvendo problemas de portabilidade e segurança.

Além da linguagem Java, o Wildfly provia toda a infraestrutura de servidor para que a aplicação fosse executada. O WildFly, também conhecido como JBoss, é um servidor de aplicações open source. Escrito em Java, o Wildfly é baseado nos padrões definidos pela especificação Java EE, sendo mantido pela comunidade e também pela empresa Red Hat (SCHMIDT, 2015).

No *front end* eram utilizadas as linguagens JSP, JavaScript e a biblioteca Jquery. Segundo Caelum (2007) uma *JavaServer Page* (JSP) é um arquivo baseado em HTML com extensão .jsp, nele é possível escrever também código Java e transformá-lo em uma *servlet*. *Servlets* são uma forma de criar páginas dinâmicas com Java. O nome *servlet* vem da ideia de um pequeno servidor ou "servidorzinho"que tem como objetivo receber chamadas HTTP [2], processá-las e devolver uma resposta ao cliente (CAELUM, 2007).

JavaScript é uma linguagem de programação utilizada na Web, a ampla maioria dos sites e navegadores modernos utilizam JavaScript, a mesma é uma linguagem de alto nível, dinâmica, interpretada e não tipada e tem sua sintaxe derivada da linguagem Java (FLANAGAN, 2012).

Dentro da linguagem Javascript o jQuery é uma biblioteca que tem como objetivo tornar coisas como a passagem e manipulação de documentos HTML, manipulação de eventos, animação e Ajax muito mais simples (JQUERY, 2021).

[2] Protocolo utilizado para transmissão de documentos hipermídia.

4 ATIVIDADES DESENVOLVIDAS

Este capítulo descreve as principais atividades desenvolvidas durante o estágio na empresa Equals. As seções deste capitulo estão organizadas de acordo com as principais atividades: treinamento, desenvovimento no modulo Equals Core e no modulo retaguarda.

4.1 Treinamento

Ao entrar na organização o estágiario foi submetido a um processo de treinamento, composto por três etapas: (i) *Onboarding*, (ii) cursos e (iii) simulação do ciclo de vida de atividades. O treinamento durou aproximadamente quarenta dias. As subseções a seguir descrevem as atividades desenvolvidas.

4.1.1 *Onboarding*

Durante a primeira semana, o estagiário foi apresentado à toda a estrutura da empresa, participando de apresentações com cada uma das equipes presentes na Equals, onde foram abordadas quais são as responsabilidades de cada equipe e onde elas atuavam dentro da plataforma Equals. Uma vez que o produto da empresa se baseia na conciliação, o estagiário também foi apresentado a todo o processo de conciliação de vendas.

Após a primeira semana, o estagiário foi inserido dentro do time de Web/API. Assim que inserido no time, o estagiário também foi introduzido às atividades do time que consistem no suporte aos times internos da Equals, suporte de segundo nível ao cliente, correções de erros no sistema, desenvolvimento de melhorias e novas funcionalidades, migração e atualização de tecnologias.

4.1.2 Cursos

Após o *Onboarding*, o estagiário realizou uma série de cursos com foco em desenvolvimento web e banco de dados Oracle. Os cursos realizados foram feitos dentro da plataforma de cursos online Alura[1] e também W3schools[2].

[1] https://www.alura.com.br/
[2] https://www.w3schools.com/

Durante o curso de HTML e CSS da plataforma Alura, com duração de 32 horas, foram abordados tópicos como a estrutura básica de um arquivo HTML, como definir estilos para elementos utilizando CSS, navegação entre páginas, posicionamento de elemento e outros.

O curso de JavaScript, realizado também na plataforma Alura, com duração de de 20 horas, abordou desde variáveis e operações na linguagem, como também validação de formulários, filtragem de tabelas, Ajax e boas práticas em JavaScript. Ao terminar o curso de JavaScript, o estagiário seguiu para o curso de jQuery, com duração de 24 horas, onde foram estudados tópicos como criação e manipulação de elementos com jQuery, eventos, tratamento de erros e envio de dados.

Por fim, o estagiário realizou o curso SQL com o Oracle Database, com duração de 56 horas. Durante o curso foram apresentados temas já vistos na disciplina de Banco de Dados, como projeções, seleções, *joins* e *views*. O diferencial em relação a disciplina, foi o aprofundamento em tópicos como *procedures*. Todos os cursos realizados propocionaram uma base técnica para atividades que foram desenvolvidas ao longo do estágio.

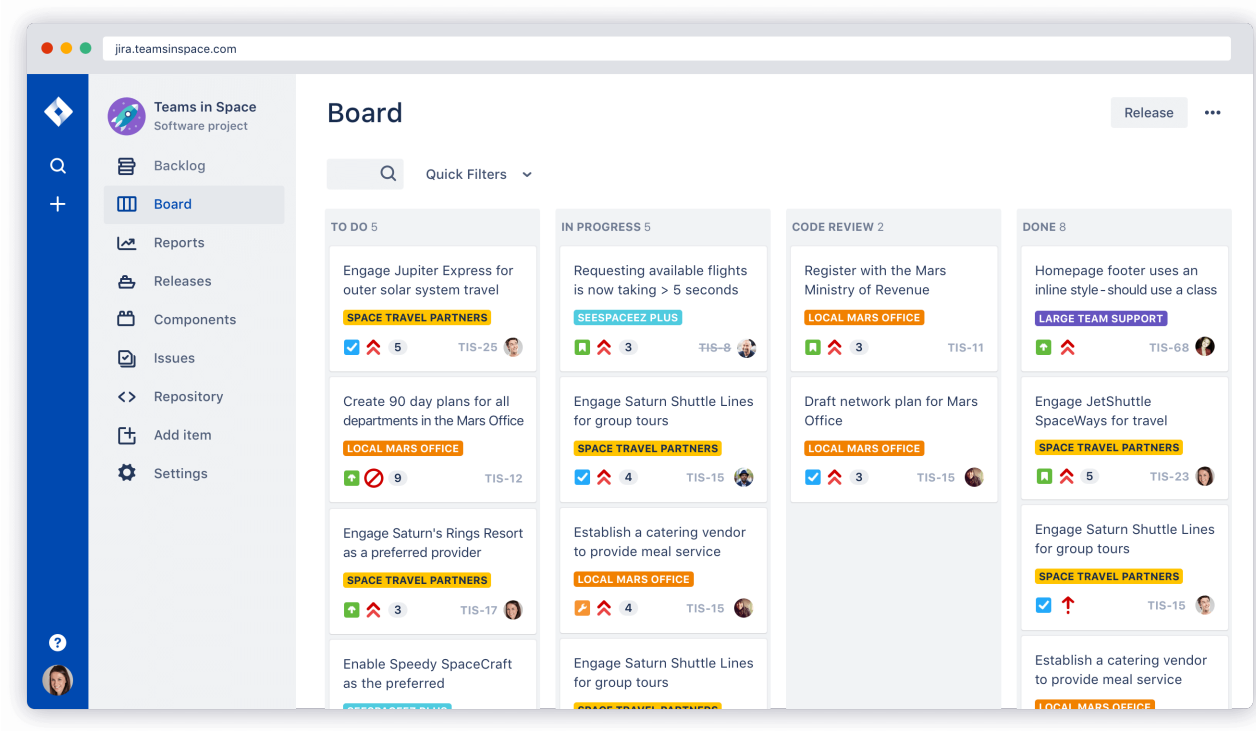
4.1.3 Simulação ciclo de vida de atividades

Após a realização dos cursos o estagiário começou a realizar tarefas dentro do time de Web/API, tendo como primeira tarefa a execução de uma simulação do ciclo de vida de atividades, simulando uma historia do *backlog* e passando por todo o ciclo de desenvolvimento. A ferramenta Jira (Figura 4.1) era usada para apoiar a implementação dos quadros de atividades da organização. O ciclo de vida das histórias é composto pelos seguintes estados:

- *Backlog* Priorizado: Itens que estão disponíveis para desenvolvimento na *Sprint* atual.
- Em Progresso: Itens que estão sendo desenvolvidos por algum membro do time.
- Impedimento: Itens que estão pausados por conta de algum impedimento.
- Liberado Para *Code Review*: Itens que já foram desenvolvidos e estão aguardando *code review* e revisão.
- *Code Review*/Revisão: Itens que estão em *code review* e revisão.
- Liberado Para Teste: Itens que já passaram por *code review*/revisão e estão aguardando o teste.
- Em Teste: Itens que estão em teste.

- Homologação: Itens que já foram testados e estão aguardando validação.
- Liberado Para Produção: Itens que já passaram por validação e estão aguardando *deploy* [3].

Figura 4.1 – Exemplo *board* Jira



Fonte: https://www.atlassian.com/br/software/jira

Para simular o ciclo de vida, o estagiário realizou uma tarefa de desenvolvimento que consistiu na criação de uma nova página onde o cliente é capaz de visualizar, cadastrar, editar e excluir motivos de cancelamentos de vendas. Nesta tarefa, além do contato com o processo de desenvolvimento, aconteceu também o primeiro contato com a arquitetura do projeto e sua organização. Neste ponto, apareceram as primeiras dificuldades, sendo necessário aprender temas ainda não vistos em sala de aula como SOLID, ORM e boas práticas de programação.

Ao final da tarefa, o estagiário passou a ter conhecimento da arquitetura do projeto, dos conceitos de desenvolvimento de software envolvidos na aplicação e também de todos os procedimentos e padronizações que envolvem o desenvolvimento de uma atividade do time Web/API. O treinamento foi de grande importância pois proporcionou a base das regras de negócio e também técnica para as futuras atividades desenvolvidas pelo estagiário.

[3] Fase do ciclo de vida de um software em que a nova versão do sistema é disponibilizada para o cliente.

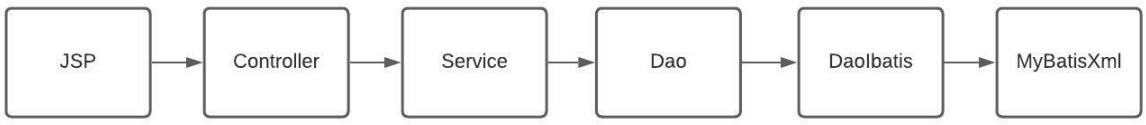
4.1.4 Fluxo de Dados e Arquitetura

O *front end* da aplicação era executado em JSP, que era responsável por enviar as requisições para a *controller* do *back end*, que por sua vez era disponibilizada pelo Wildfly. Dessa forma, ao receber um requisição a controller buscava e tratava as informações necessárias para serem exibidas em tela, em seguida essas informações eram redirecionadas para o JSP, que executava o HTML, CSS e Javascript.

O *back end* era responsável por conter as regras de negócio e fazer os devidos tratamentos para as requisições. Ele consistia da *controller* para o conciliador e retaguarda, que trata as requisições. Abaixo da camada de *controller* existia a camada de *service* que servia como uma camada intermediária entre a *controller* que recebe as requisições e a camada de persistência *DAO*, onde sua principal função era chamar os métodos da *DAO* e fazer as verificações que não são específicas da *controller* relacionadas a regra de negócio.

A camada *DAO* (*Data Access Object*) é a camada onde é centralizado o acesso ao banco de dados do sistema. Dentro do projeto Equals Core e Retaguarda a *DAO* é dividida em duas camadas sendo estas a camada *DAO* e *DaoIbatis*. A primeira camada era a *DAO*, ela é composta por uma interface contendo a assinatura dos métodos da camada de persistência, tendo como objetivo manter o baixo nível de acoplamento com a camada de persistência. A camada inferior é a *DaoIbatis*, nela é onde os métodos das assinaturas da camada *DAO* são de fato implementados. A camada *DaoIbatis* é responsável por também fazer as requisições XML (*Extensible Markup Language*) [4] do Mybatis para a camada MyBatisXml como ilustra a Figura 4.2.

Figura 4.2 – Fluxo de dados



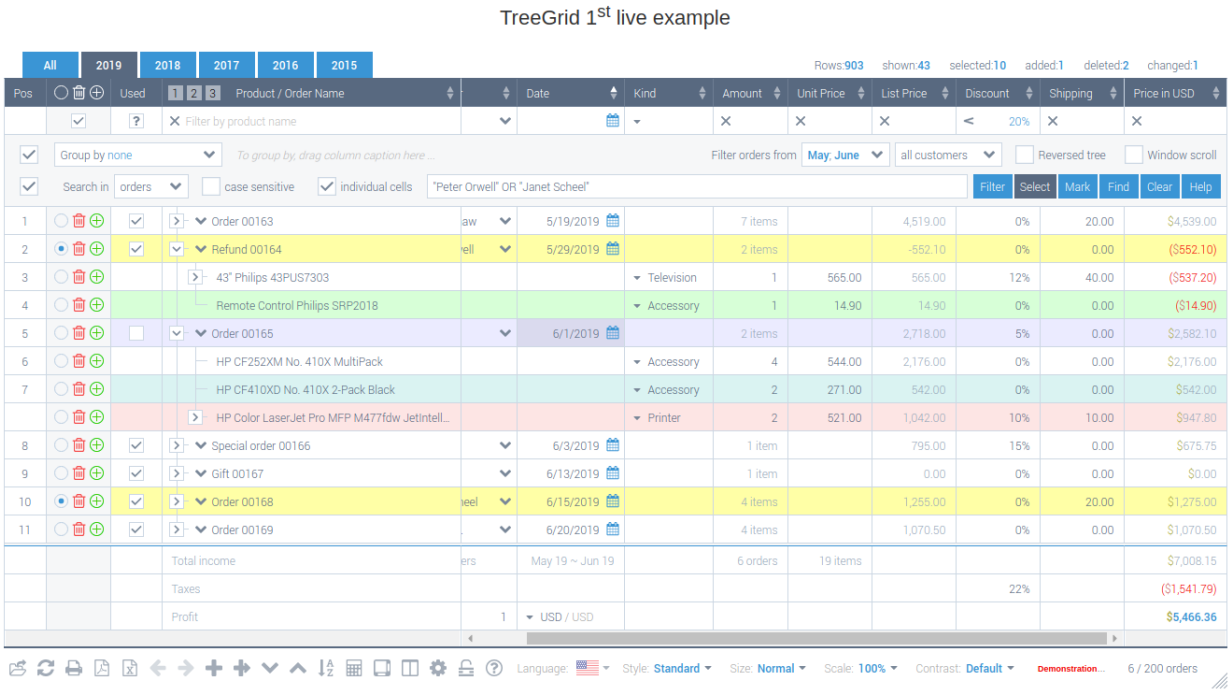
Fonte: Autoria própria

[4] Linguagem de marcação com regras para formatar documentos.

4.1.5 Alteração e Criação de Relatórios

Dentre as atividades desenvolvidas, uma das mais comuns era a alteração e criação de relatórios que exibem informações de conciliação para os clientes. Normalmente essas informações eram exibidas através de uma tabela utilizando a biblioteca Treegrid. A Treegrid (Figura 4.3) consiste em um componente DHTML[5] (*Dynamic Hyper Text Markup Language*) feito em JavaScript, utilizado para exibir os dados em tabela.

Figura 4.3 – Exemplo *TreeGrid*



Fonte: http://www.treegrid.com/Grid

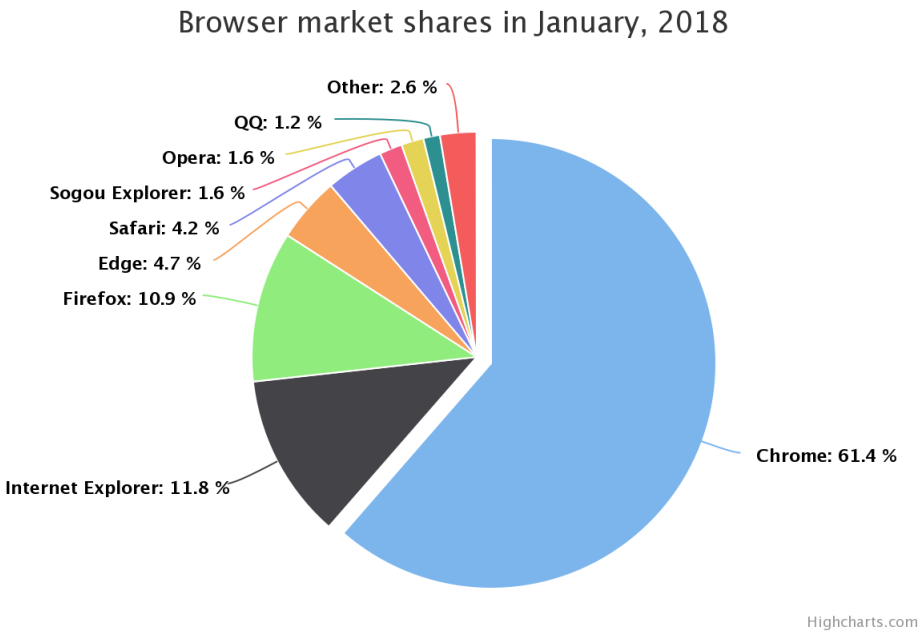
Procedures eram utilizadas com frequência para geração de relatórios, assim a manutenção e criação de *stored procedures*, criação de *scripts* e manipulação de tabelas, também faziam parte das tarefas do dia a dia.

[5] Conjunto de técnicas utilizadas entre HTML e JavaScript para tornar o HTML dinâmico.

4.1.6 Criação de Gráficos

Outro tipo de atividade recorrente era a criação de gráficos para exibição de dados relacionados a conciliação para os clientes. Neste ponto, geralmente era utilizada a biblioteca *Highcharts* (Figura 4.4), que é uma biblioteca de software para gráficos escrita em JavaScript puro.

Figura 4.4 – Exemplo *HighCharts*



Fonte: https://www.highcharts.com/demo/pie-basic

Como esse tipo de atividade regularmente envolvia tabelas com grandes quantidades de elementos, por questões de desempenho, era comum a criação de *views* para armazenamento dos dados que seriam apresentados nos gráficos.

4.1.7 Controle de Versão

Dentro da empresa é utilizado o sistema de controle de versão Git, através da plataforma de hospedagem Github. A plataforma Github é conectada com a ferramenta Jira, através dos prefixos com o código do *card* Jira adicionado na criação das *branches*, a ferramenta realiza a associação dos *cards* com as *branchs* criadas no Github. O padrão de *branches* utilizado no fluxo de desenvolvimento era o seguinte:

- *Feature*: É uma *branch* de melhoria ou nova funcionalidade do sistema. Refere-se ao tipo de item Software - Evolução do Jira. Esta *branch* inicia-se com o código do Jira da atividade, seguido de uma breve descrição da atividade. Exemplo: "EQ-1525_exemplo_feature".

- *Fix*: É uma *branch* de correção no ambiente de produção. Refere-se ao tipo de item *Software* - Manutenção do Jira. Esta *branch* inicia-se com o prefixo "fix/", seguido do código do Jira e uma breve descrição. Exemplo: "fix/EQ-2587_exemplo_fix".

- *Hotfix*: É uma *branch* de correção emergencial no ambiente de produção. Refere-se ao tipo de item *Software* - Manutenção do Jira com prioridade Emergencial. Esta *branch* inicia-se com o prefixo "hotfix/", seguido do código do Jira e uma breve descrição. Exemplo:"hotfix/EQ-899_exemplo_hotfix".

- *Develop*: É a *branch* comum de desenvolvimento entre todo os times.

- Pacote: É uma *branch* que contém um conjunto de features relacionadas. Exemplo: todas as *branchs* das atividades relacionadas com a nova tela de cadastro de clientes seriam criadas a partir da *branch* "PACOTE_CADASTRO_CLIENTES".

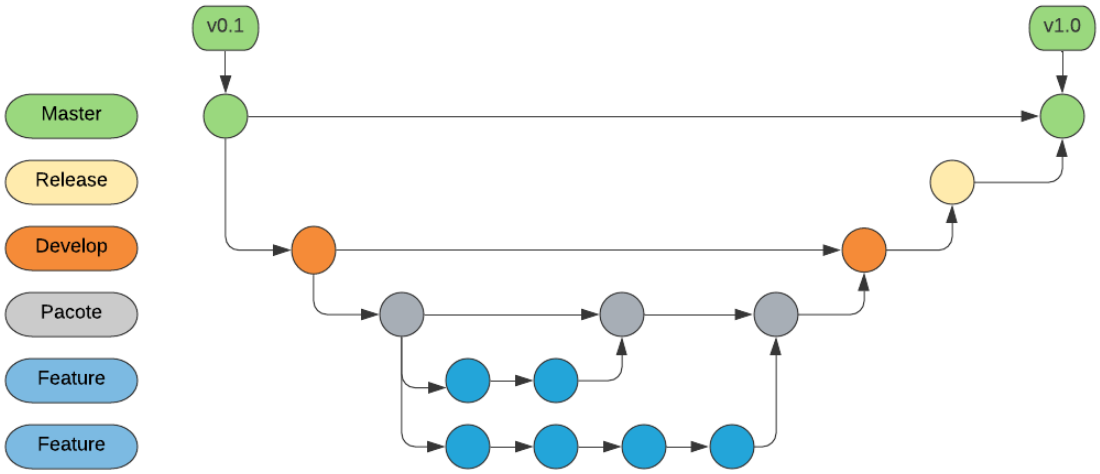
- *Release*: *Branch* com todas as atividades que irão fazer parte da próxima versão de produção.

- *Master*: *Branch* executada no ambiente de produção.

Assim, ao início de uma nova atividade, uma *branch* de pacote com todas as atividades relacionadas era criada. Deste modo as *features* eram criadas a partir do pacote. Ao final do desenvolvimento da *feature*, antes de realizar o *merge* com a *branch* do pacote era feito o *rebase*. O *rebase* tem a mesma função que o *merge*, porém ele trás as mudanças de outra *branch* de uma forma diferente, deixando o histórico de alterações mais linear, limpo e legível.

Após o *rebase*, era realizado o *merge* da *feature* com o pacote, *branch* na qual eram realizados os testes das atividades e validações. A validação era um processo onde depois dos testes, todos os pontos a serem desenvolvidos eram revisados para garantir que tudo que foi proposto foi desenvolvido corretamente. Posteriormente a validação, era realizado o *rebase* e *merge* com a *develop*, após isso realizava-se então o *rebase* e *merge* da *develop* com a *release*, em seguida o mesmo procedimento da *release* para *master*. A Figura 4.5 representa o fluxo de *branches*.

Figura 4.5 – Fluxo de *Branches*



Fonte: Autoria própria

5 CONCLUSÃO

O objetivo deste documento foi relatar as experiências obtidas durante o período de estágio supervisionado na empresa Equals, uma *fintech* que atua no nicho de conciliações financeiras. Nesse período, o estagiário atuou como desenvolvedor Web no time de Web/API, responsável por uma das principais aplicações da empresa que é o Equals Core.

Durante o período de estágio, foi possível assimilar tópicos vistos em matérias como estrutura de dados, engenharia de software, sistemas operacionais, banco de dados e programação web com o que era praticado no dia a dia da empresa. Além de aprofundar em tópicos vistos em sala de aula, também foi possível ter um contato mais aprofundado em tecnologias como Java, JavaScript, JSP e MyBatis, assim como, o funcionamento do fluxo de desenvolvimento de um grande projeto de software. Sendo capaz de compreender como um sistema de gerenciamento de versões (Git) é usado em um projeto de larga escala.

A ampla abrangência de responsabilidades da equipe de Web/API e também a alta complexidade das aplicações fez com que o estagiário evoluísse não só tecnicamente como também em habilidades interpessoais. Como ponto negativo, posso destacar o fato de o estágio ter ocorrido em sua maioria de forma remota devido a pandemia, o que levou a uma menor interação social com os colegas de trabalho e também uma maior dificuldade de assimilar conteúdos técnicos necessários para desenvolver as atividades cotidianas dentro da empresa.

Com relação ao curso de bacharelado em ciência da computação, acredito que o conteúdo sobre metodologias ágeis poderia ser abordado de forma mais aprofundada ao longo da graduação. Ainda assim, o curso de bacharelado em ciência da computação foi imprescindível para realizar as atividades dentro da empresa. Ao longo da graduação, tópicos vistos em sala de aula me proporcionaram toda a base necessária para realizar as atividades cotidianas como desenvolvedor de software dentro da empresa Equals.

REFERÊNCIAS

ATLASSIAN. 2021. Disponível em: <https://www.atlassian.com/br/software/jira/agile>.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>.

BELL, P.; BEER, B. **Introducing Github**. 1. ed. O'Reilly Media, 2014. Disponível em: <https://www.oreilly.com/library/view/introducing-github/9781491949801>.

BUTTERFIELD, A.; NGONDI, G. E.; KERR, A. **A Dictionary of Computer Science**. Oxford University Press, 2016. ISBN 9780191768125. Disponível em: <https://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975>.

CAELUM. **JAVA PARA DESENVOLVIMENTO WEB**. Caelum, 2007. Disponível em: <https://www.caelum.com.br/apostila-java-web/>.

CHACON, S.; STRAUB, B. **Pro Git**. 2. ed. Apress, 2014. Disponível em: <https://git-scm.com/book/pt-br/v2>.

CLARO, D. B.; SOBRAL, J. B. M. **Programação em JAVA**. [S.l.]: Pearson Education, 2015.

DEVMEDIA. **ORM : Object Relational Mapper**. 2011. Disponível em: <https://www.devmedia.com.br/orm-object-relational-mapper/19056>.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 7. ed. [S.l.]: Pearson Universidades, 2019. ISBN 8543025001.

FLANAGAN, D. **JavaScript O guia definitivo**. 6. ed. [S.l.]: Bookman, 2012.

JQUERY. **What is jQuery?** 2021. Disponível em: <https://jquery.com/>.

MARIOTTI, F. S. Kanban: o ágil adaptativo. engenharia de software magazine. 2012. Accessed: 2021-09-18. Disponível em: <https://www.devmedia.com.br/kanban-o-agil-adaptativo-revista-engenharia-de-software-magazine-45/23560>.

MARTIN, R. C. **Principles of OOD**. 2014. Disponível em: <http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod>.

MYBATIS. **O que é MyBatis?** 2021. Disponível em: <https://mybatis.org/mybatis-3/>.

ORACLE. 2021. Disponível em: <https://www.oracle.com/br/database/what-is-database/#WhatIsDBMS>.

PINHEIRO, R. N. **Agilidade nas Empresas: Guia Scrum para quem não é de TI**. 2. ed. [S.l.: s.n.], 2017.

REHKOPF, M. 2021. Disponível em: <https://www.atlassian.com/br/agile/scrum/sprints>.

ROBBINS, J. N. **Learning web design: a beginners guide to HTML, CSS, Javascript, and web graphics**. 4. ed. [S.l.]: OReilly, 2012.

SCHILDT, H. **Java para Iniciantes: Crie, Compile e Execute Programas Java Rapidamente**. 6. ed. [S.l.]: Pearson Universidades, 2015. ISBN 9788582603369.

SCHMIDT, A. Wildfly - do básico ao ambiente de produção. 2015. Accessed: 2021-09-18. Disponível em: <https://www.devmedia.com.br/wildfly-do-basico-ao-ambiente-de-producao/33653>.

SCHWABER, K.; SUTHERLAND, J. O guia do scrum. 2020.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. [s.n.], 2020. ISBN 978-65-00-01950-6. Disponível em: <https://engsoftmoderna.info/>.

W3SCHOOL. **What is a Stored Procedure?** 2021. Disponível em: <https://www.w3schools.com/sql/sql_stored_procedures.asp>.

WADE, B. W.; CHAMBERLIN, D. D. Ibm relational database systems: The early years. **IEEE Annals of the History of Computing**, v. 34, n. 4, p. 38–48, 2012.

WEST, D. **PLANEJAMENTO DO SPRINT**. 2021. Disponível em: <https://www.atlassian.com/br/agile/scrum/sprint-planning>.