WILIAN HENRIQUE DE SOUZA

DESENVOLVIMENTO DE SISTEMAS WEB NA AGÊNCIA ZETTA

LAVRAS – MG

2021

WILIAN HENRIQUE DE SOUZA

DESENVOLVIMENTO DE SISTEMAS WEB NA AGÊNCIA ZETTA

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

Bruno de Abreu Silva
Orientador

LAVRAS – MG

2021

WILIAN HENRIQUE DE SOUZA

DESENVOLVIMENTO DE SISTEMAS WEB NA AGÊNCIA ZETTA
WEB SYSTEMS DEVELOPMENT AT ZETTA AGENCY

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

APROVADA em 23 de Abril de 2021.

Prof. Bruno de Abreu Silva	UFLA
Prof. Paulo Afonso Parreira Júnior	UFLA
Thiago Vilela Paranhos Nunes	CRIA

Bruno de Abreu Silva
Orientador

LAVRAS – MG
2021

Dedico este trabalho ao meu querido pai, Vanderlei Henrique de Souza (in memoriam), que sempre me apoiou em todos os momentos de minha vida, e sem o qual, eu nunca teria chegado aonde cheguei. Gratidão eterna.

AGRADECIMENTOS

Primeiramente gostaria de agradecer aos meus pais Vanderlei Henrique de Souza e Marcinea da Silva, que sempre estiveram ao meu lado, me apoiando e incentivando ao longo de toda a minha vida.

Agradeço ao meu orientador Bruno de Abreu Silva por aceitar conduzir o meu trabalho.

E também a todos os meus amigos que me incentivaram durante esta jornada.

RESUMO

O presente relatório busca descrever as rotinas, atividades, experiências e desafios vivenciados pelo discente no desenvolvimento de dois projetos durante a realização do estágio na Agência ZETTA (Agência UFLA de Inovação em Geotecnologias e Sistemas Inteligentes no Agronegócio). O primeiro projeto desenvolvido foi um sistema de gerenciamento de reserva de salas, o qual foi executado durante o período de treinamento fornecido pela agência, e teve como objetivo preparar o estagiário para projetos futuros. O segundo projeto foi um sistema de emissão de títulos de direito de outorga desenvolvido para o Estado do Pará, o qual era um projeto legado que já estava próximo da sua entrega para o cliente. Este trabalho também visa descrever a primeira experiência profissional do discente, em que houve a oportunidade de se aplicar os conhecimentos adquiridos durante a graduação, bem como foi possível aprender novas tecnologias, como Java, Angular, Banco de Dados, o Framework Spring Boot, além de conhecer e utilizar a metodologia ágil Scrum.

Palavras-chave: Scrum. Estágio. Sistemas Web.

ABSTRACT

This report aims to describe the routines, activities, experiences and challenges experienced by the student in the development of two projects during the internship at Agência ZETTA (UFLA Agency for Innovation in Geotechnologies and Intelligent Systems in Agribusiness). The first project developed was a room reservation management system, that was executed during the training period provided by the agency, and aimed to prepare the intern for future projects. The second project was a system of grant right titles emission, developed for the State of Pará, which was a legacy project that was already close to its delivery to the client. This work also aims to describe the student's first professional experience, in which there was an opportunity to apply the knowledge acquired during graduation, as well as it aims to describe that it was possible for the intern to learn new technologies, such as Java, Angular, Database, the Spring Boot Framework, in addition to the opportunity to know and to use the agile Scrum methodology.

Keywords: Scrum. Internship. Web Systems.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, o mercado de trabalho está cada vez mais exigente e competitivo, o que leva as empresas a se tornarem cada vez mais exigentes em seus processos seletivos, de modo que exigem dos seus candidatos mais experiência e qualificação. Nesse contexto, a realização de estágios proporciona aos estudantes uma das melhores formas de ingresso no mercado de trabalho.

Este trabalho é o relatório das atividades desenvolvidas pelo estagiário durante o seu período de estágio na Agência UFLA de Inovação em Geotecnologias e Sistemas Inteligentes no Agronegócio (ZETTA), uma organização que atua principalmente no desenvolvimento de sistemas relacionados ao agronegócio.

Nesse intervalo de tempo, o estagiário atuou em dois sistemas. O primeiro projeto, realizado durante o período de treinamento, foi um sistema de reserva de salas que seria utilizado nos mais diversos fins, como reuniões e confraternizações, este sistema seria usado internamente na agência, e sua finalidade era preparar o estagiário para atuar em projetos futuros. O segundo foi um sistema para analisar pedidos de outorga de direito do estado do Pará, um sistema legado para clientes externos à agência.

1.1 Objetivos

Este relatório tem como objetivo relatar as atividades desenvolvidas pelo estagiário durante o período de 01 de julho de 2020 até 31 de dezembro de 2020.

Posto que o estágio foi dividido em dois projetos, são apresentados, a seguir os objetivos de cada um deles.

O objetivo principal do Sistema de Reserva de Salas foi facilitar o modo que eram feitas as reservas de salas, tornando esse processo digital e automático. Além disso, objetivou-se estabelecer contato com tecnologias e rotinas que seriam utilizadas após o período de treinamento. Esse sistema é uma demanda da agência para realização de reservas de salas para os mais diversos fins, como por exemplo

para reuniões, eventos, coffee breaks, etc. A versão desenvolvida pelo estagiário foi um versão inicial que seria continuada por outros colaboradores.

Em relação ao Sistema de Outorga de Direito do Estado do Pará, o objetivo principal foi implementar melhorias no sistema já existente, enquanto os objetivos secundários foram corrigir suas inconformidades e implementar, nele, novas funcionalidades.

Pode-se ressaltar, do ponto de vista do crescimento profissional do estagiário, que o objetivo principal do estágio foi obter experiência profissional, de modo que fosse possível aplicar, na prática, os conhecimentos adquiridos durante a graduação. Para se alcançar esse objetivo principal, diversos objetivos secundários foram estabelecidos:

- Desenvolver habilidades de trabalho em equipe;
- Ter contato com metodologias e ferramentas de trabalhado utilizadas no desenvolvimento Web;
- Colaborar com a agência, utilizando os conhecimentos obtidos durante a graduação;
- Experimentar uma possível área de atuação, bem como, se desejado, seguir neste ramo.

Haja vista esses objetivos, as principais atividades desenvolvidas durante o estágio foram as seguintes:

- Desenvolvimento de um sistema de gerenciamento de reserva de salas;
- Participação em workshops, cursos e treinamentos;
- Participação no desenvolvimento de um sistema de outorga de direito de recursos hídricos.

1.2 A agência

Localizada na Universidade Federal de Lavras (UFLA), a Agência UFLA de Inovação em Geotecnologias e Sistemas Inteligentes no Agronegócio (ZETTA) foi credenciada, em 2020, como Unidade de Agricultura Digital da Empresa Brasileira de Pesquisa e Inovação Industrial (Embrapii), uma organização social que atua por meio da cooperação com instituições de pesquisas científicas e tecnológicas, públicas e privadas, tendo como foco as demandas empresariais e como alvo o compartilhamento de riscos na fase pré-competitiva da inovação.

O propósito dessa agência é o de promover soluções que impactam a sociedade de modo direto, gerando conhecimento e tecnologia para aumentar a produção de alimentos, a sustentabilidade e a eficiência de serviços para a população. Suas áreas de expertise são:

- Inovação: explorar novas ideias, permitindo a criação de algo novo e a agregação de valor para o cliente;

- Geotecnologias: conjunto de técnicas e métodos que permitem a coleta, o processamento, a análise e a disponibilização de dados com referência geográfica;

- Sistemas inteligentes: processamento de bilhões de informações, de modo a estruturá-las em dados úteis e estratégicos centralizados em um único lugar, gerando informações úteis para a tomada de decisão;

- Agricultura digital: uso das tecnologias digitais na agricultura, promovendo a otimização da produção, a redução de falhas, o aumento da produtividade, um melhor aproveitamento de insumos e de recursos naturais, a redução do custo de produção e a promoção do bem-estar animal e da sustentabilidade.

1.2.1 Organização

Na Agência ZETTA, os colaboradores são divididos em times de desenvolvimento que são alocados em tribos, sendo cada uma delas composta por vários times de desenvolvimento e liderada por pelo menos um(a) Gerente de Projetos. Na estrutura organizacional dessa agência, os principais cargos são:

- Gerente de Projetos: responsável pelos projetos atribuídos à tribo que ele lidera; sua principal função é a de planejar e coordenar a execução dos projetos;
- Scrum Master: responsável pelo time de desenvolvedores; sua principal função é a de remover impedimentos e facilitar o trabalho dos integrantes do time;
- Product Owner: responsável por conduzir o projeto de acordo com a necessidade do cliente, implementando novas funcionalidades e melhorias para o produto;
- Desenvolvedor: responsável pelo desenvolvimento dos sistemas, que são o produto principal da agência;
- Analista de qualidade: responsável pela qualidade do produto; sua principal função é analisar as inconformidades do sistema;
- Administrador do banco de dados: administra, mantém, evolui e gerencia o banco de dados.

1.3 Estrutura do trabalho

Os capítulos subsequentes possuem os seguintes conteúdos:

- O Capítulo 2 apresenta o referencial teórico, que contém alguns conceitos, ferramentas e tecnologias que foram utilizadas e são relevantes para o embasamento deste trabalho;

- o Capítulo 3 contém uma descrição pormenorizada das atividades desenvolvidas durante o período de treinamento e no decorrer do desenvolvimento do sistema de outorga;
- Ao final, o Capítulo 4 reúne as conclusões relativas ao período de estágio elaboradas pelo estagiário.

2 REFERENCIAL TEÓRICO

Este capítulo tem o objetivo de apresentar os conceitos importantes para um melhor entendimento deste trabalho, bem como as ferramentas e tecnologias que foram utilizadas. Neste serão abordados o framework de gerenciamento de projetos SCRUM, conceitos sobre bancos de dados, HTML, CSS e JavaScript, os frameworks Angular, Spring Boot, Spring Data JPA, a linguagem de programação Java, a ferramenta Planning Poker, o ambiente de desenvolvimento Intellij IDEA, e por fim o editor de código Visual Studio Code.

2.1 SCRUM

Metodologias de gestão de projetos tradicionais seguem um modelo sequencial, no qual uma etapa deve ser executada após a outra. Assim, uma tarefa não pode ser iniciada enquanto a antecedente não estiver concluída. Também é esperado que o itinerário planejado no início do projeto seja seguido à risca, buscando-se o resultado final dentro do prazo, da qualidade e do orçamento definidos. (GUEDES, 2019). O principal receio de quem utiliza as metodologias tradicionais é a falta de flexibilidade em relação às mudanças percebidas no decorrer do projeto. Nesse contexto, surgem os métodos ágeis, como o Scrum, que será descrito nesta seção.

Desenvolvido por Jeff Sutherland em 1993, o Scrum é um framework de gerenciamento de projetos. Inicialmente, seu foco estava somente no desenvolvimento de software, mas hoje em dia ele também é aplicado no desenvolvimento de produtos de uma maneira geral (CARVALHO; MELLO, 2012).

O termo Scrum deriva-se de uma jogada bastante conhecida do Rúgbi, em que é necessária a colaboração de todos os jogadores trabalhando em conjunto para atingir um objetivo em comum. Esse trabalho em equipe é uma das principais características do framework Scrum (CRUZ, 2013).

O Scrum controla processos empíricos com uma abordagem incremental e iterativa para otimizar a previsibilidade e diminuir os riscos, em comparação a uma entrega única e total de métodos tradicionais. Para fazer esse controle, o Scrum segue três pilares de sustentação (CRUZ, 2013):

- Transparência: visibilidade completa do que está acontecendo, de modo que todos os responsáveis por alguma parte da entrega devem ser capazes de visualizar o processo de que fazem parte;
- Inspeção: observar os processos com frequência para garantir a qualidade da entrega;
- Adaptação: de posse das observações coletadas durante a inspeção, o próximo passo é agir para se adaptar a alguma alteração no processo.

2.1.1 O Time Scrum

O framework Scrum é formado por pequenos times que possuem seus respectivos papéis e responsabilidades, os quais, juntos, realizam eventos com uma duração fixa. Cada time Scrum é composto pelo Product Owner, o Scrum Master e o time de desenvolvimento. A seguir, será descrito o papel de cada membro do time Scrum (CRUZ, 2013):

- Scrum Master: É quem vai estar à frente da resolução de problemas enfrentados pelo time. Caso algum integrante do time esteja enfrentando algum tipo de resistência ou algum tipo de impedimento, é o Scrum Master quem deve tomar a frente para que a solução seja encontrada. Também é de responsabilidade do Scrum Master fazer com que os novos integrantes do time, que ainda não têm familiaridade com o framework, adotem a cultura e os ritos do Scrum (FERREIRA, 2020);
- Product Owner (PO): Também chamado de dono do produto, é o integrante do time responsável por manter o Backlog do produto sempre atualizado. O

Backlog do produto é o principal artefato do Scrum, já que se trata de uma lista com todos os requisitos do produto a ser entregue ao cliente. O PO também é responsável por conhecer a solução do produto, e deve sempre procurar entregar mais valor ao cliente. Sua relação com o cliente final é muito forte, de modo que ele sempre troca informações com o cliente quando necessário (FERREIRA, 2020);

- Time de desenvolvimento: São os integrantes do time responsáveis por transformar os itens do Backlog do Produto em incrementos de funcionalidade para que possam ser entregues ao cliente. (CRUZ, 2013).

2.1.2 Eventos Scrum

Os eventos são usados para minimizar reuniões não planejadas no Scrum. Todos eles são eventos *time-boxed*, de modo que todo evento tem uma duração máxima. Uma vez que a Sprint é iniciada, sua duração não deve ser alterada, e os eventos restantes podem terminar quando o seu objetivo for cumprido. Os eventos citados acima são: *Sprint Planning Meeting* (Reunião de Planejamento da Sprint), *Daily Scrum* (Reunião Diária), *Sprint Review Meeting* (Reunião de Revisão da Sprint), *Sprint Retrospective* (Retrospectiva da Sprint). A seguir será descrito cada um desses itens (FERREIRA, 2020).

- *Sprint Planning Meeting*: O tempo de duração da reunião de planejamento deve ser levado em conta na duração do Sprint. Para uma Sprint com duas semanas de duração, a *Sprint Planning Meeting* durará aproximadamente quatro horas. Nessa reunião, o PO fornecerá para o time os itens do Backlog do Produto com maior grau de prioridade. Assim, o time pode tirar suas dúvidas e também pode sugerir a quebra dos itens por especialidades. Os itens selecionados para a Sprint passam, então, para o Backlog da Sprint e devem ser suficientes para durar por todo o seu tempo de duração. A quantidade estimada de itens leva em conta a experiência das Sprints passadas,

assim como a capacidade do time de entregar as tarefas. Após a primeira etapa de planejamento da Sprint, o time de desenvolvimento irá se reunir para analisar e discutir a viabilidade da entrega dos itens propostos pelo PO, dando retorno a ele sobre a possível diminuição ou não do Backlog da Sprint. Assim, o time se compromete com as entregas propostas.

- *Daily Scrum* (Reunião Diária): Uma reunião curta que não deve durar mais do que 15 minutos. Nela, todos os integrantes do time Scrum podem estar presentes, mas, em essência, é o time de desenvolvimento que participa desta reunião. Cada um relata o que foi feito no dia anterior, o que planeja fazer neste dia e se há algum tipo de impedimento no seu trabalho. Caso exista algum impedimento, é papel do Scrum Master removê-lo. As soluções do impedimento serão feitas em um momento posterior, do qual participam apenas aqueles que podem contribuir com a formação da solução.

- *Sprint Review Meeting* (Reunião de Revisão da Sprint): Esta reunião acontece no último dia da Sprint e normalmente tem duração máxima de quatro horas, para uma Sprint de duas semanas. Nela, o time Scrum irá discutir se os objetivos da Sprint foram cumpridos e se todos os itens do Backlog da Sprint foram entregues. Um aspecto muito importante é a aceitação dos itens como prontos pelo PO. É do PO ou de alguém que represente o cliente a palavra final sobre se o que foi entregue está de acordo com o que foi planejado;

- *Sprint Retrospective* (Retrospectiva da Sprint): Esta reunião acontece logo após a Reunião de Revisão da Sprint, e, para uma Sprint de duas semanas, ela deve ter duração máxima de duas horas. Seu objetivo não é o de analisar os Backlogs, mas, sim, o de perceber se a rotina foi seguida e se as ferramentas estão sendo utilizadas. É uma avaliação do time para identificar pontos de melhoria que devem ser aplicados na próxima Sprint.

Os eventos descritos anteriormente fazem parte do ciclo de vida de uma Sprint. Em uma Sprint de duas semanas, são utilizados cerca de dois dias para o planejamento e para as revisões. Ainda que isso possa parecer um tempo perdido, há um grande ganho de produtividade, porque o projeto terá um planejamento efetivo daquilo que realmente será entregue e o time trabalhará com a melhoria contínua no resultado das entregas e dos processos (FERREIRA, 2020).

Na ZETTA, o Scrum não é aplicado em atendimento integral às práticas, já que alguns ajustes são realizados para haver adaptação às necessidades da agência. O tempo de duração de uma Sprint na ZETTA era de duas semanas, e o tempo de duração dos outros ritos eram proporcionais ao tempo da Sprint como explicado nesta seção.

Uma característica mais notadamente observada pelo estagiário nos times de desenvolvimento da ZETTA, foi o espirito colaborativo e o esforço conjunto para atingir um objetivo em comum.

2.2 Banco de dados

Um banco de dados é uma coleção de dados relacionados que são armazenados e organizados de forma a suprir as necessidades dos seus usuários. Em outras palavras, o banco de dados deve possuir alguma fonte que lhe forneça os dados, um público interessado no seu conteúdo e certo grau de interação com ocorrências do mundo real (VICCI, 2015).

Para gerenciamento de uma base dados é utilizado o Sistema de Gerenciamento de Banco de Dados (SGBD), que é um sistema computadorizado que permite que o usuário desenvolva e mantenha um banco de dados. Ele é um sistema de software de uso geral que facilita o processo de definição, construção, manipulação e compartilhamento de bancos de dados (ELMASRI; NAVATHE, 2018).

2.2.1 Modelo de dados relacional

Em um banco de dados relacional, cada tabela deve possuir um campo, denominado chave primária. Esse campo permite identificar de forma exclusiva um determinado registro, pois ele não pode ser repetido (LEAL, 2015).

Além da chave primária, outro conceito importante é o de chave estrangeira, que consiste em um campo usado para estabelecer uma relação de dependência entre as relações, ou seja, é por meio de chaves estrangeiras que se estabelece o relacionamento entre as relações (LEAL, 2015).

2.2.2 Modelo Entidade-Relacionamento

O modelo Entidade-Relacionamento (ER), é um modelo de dados conceitual de alto nível e muito popular, que descreve os dados como entidades, relacionamentos e atributos, os quais serão explanados a seguir (ELMASRI; NAVATHE, 2018).

O conceito básico que o modelo ER representa é uma entidade, que é algo do mundo real com uma existência independente, como, por exemplo, um objeto com uma existência física, como um carro ou uma casa, ou um objeto com uma existência conceitual, como uma empresa ou um cargo. Cada entidade possui atributos, que são propriedades que a descrevem. Por exemplo, uma entidade CACHORRO pode ser descrita pelo nome, pela idade, pela raça e pela cor do pelo (ELMASRI; NAVATHE, 2018).

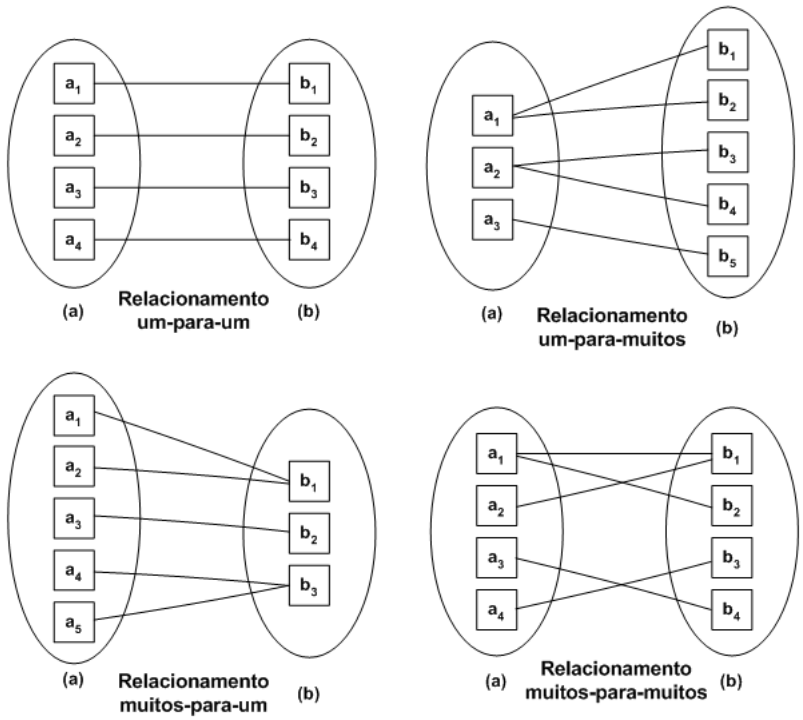
Sempre que um atributo de determinado tipo de entidade se refere a outro tipo de entidade, existe um tipo de relacionamento. Antes de se fazer uma explicação sobre os tipos de relacionamentos, um conceito essencial a se saber é o de cardinalidade. A razão de cardinalidade determina o número máximo de instâncias de relacionamento de que uma entidade participa. Como exemplo, uma pessoa pode ter vários carros, mas um carro pode ter apenas uma pessoa como dono (VICCI, 2015).

Com base na cardinalidade, existem quatro tipos básicos de relacionamentos (RODRIGUES, 2014; SIQUEIRA,):

- Relacionamento um para um (1:1) - Uma entidade de A pode estar relacionada a no máximo uma entidade de B, e uma entidade de B pode estar relacionada a no máximo um entidade de A;
- Relacionamento um para muitos (1:M) - Uma entidade de A pode estar relacionada a qualquer número de entidades de B, e uma entidade de B pode estar relacionada a no máximo uma entidade de A;
- Relacionamento muitos para um (N:1) - Uma entidade de A pode estar relacionada a no máximo uma entidade de B, e uma entidade de B pode estar relacionada a qualquer número de entidades de A;
- Relacionamento muitos para muitos (N:M) - Uma entidade de A pode estar relacionada a qualquer número de entidades de B e uma entidade de B pode estar relacionada a qualquer número de entidades de A.

A Figura 2.1 ilustra o comportamento de cada uma das relações citadas acima.

Figura 2.1 – Tipos de relacionamentos entre entidades.



Fonte: Material elaborado pelo professor André Rodrigo Sanches (SANCHES, 2004).

2.3 HTML, CSS e JavaScript

HTML, CSS e JavaScript são a tríade de tecnologias que grande parte dos desenvolvedores Web precisam conhecer. Esse conjunto de tecnologias é usado para criação de páginas Web. O HTML cria as estruturas das páginas, o CSS faz a estilização da página, e o JavaScript especifica o comportamento delas (FLANAGAN, 2013).

O HTML (*HyperText Markup Language*) é uma linguagem de marcação, usada para estruturar páginas web e seu conteúdo. Como exemplo, podemos estru-

turar conteúdos em parágrafos, listas com marcadores ou a partir do uso de tabelas e imagens (MOZILLA, b).

CSS (*Cascading Style Sheets*) é uma linguagem de estilização usada para descrever como os elementos HTML são apresentados na tela. Em outras palavras, é ele quem cuida da aparência da página Web (MOZILLA, a).

JavaScript é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe. Apesar de ser utilizada principalmente como uma linguagem de script para páginas Web, ela também é usada em outros ambientes sem browser, como node.js, Apache CouchDB e Adobe Acrobat (MOZILLA, c).

O estagiário utilizou as tecnologias citadas acima pois elas se configuram como a base para o desenvolvimento de páginas Web. Essas ferramentas foram usadas através do framework Angular 5, que será descrito na próxima seção.

2.4 Angular 5

O Angular é um framework de código aberto desenvolvido e mantido pela Google e usado na construção de SPA (*Single Page Applications* ou Aplicações de Página Única). Um SPA é uma aplicação web construída em apenas uma página, na qual a navegação e a interação entre seções de uma página não ocorre de maneira que toda a página seja recarregada a cada mudança na mesma.

Trata-se de uma ferramenta que tem como finalidade a criação de aplicações SPA de maneira otimizada e menos complexa. Com o Angular, é possível criar aplicações web voltadas para resoluções desktop e também para resoluções mobile, o que o torna uma ferramenta dinâmica, moderna e escalável (GUEDES, 2020).

Na ZETTA, o estagiário utilizou o Angular 5 durante o seu período de estágio, porque essa já era a tecnologia que estava sendo utilizada pela Agência. Em virtude de ela ser uma versão mais antiga, muitas facilidades que as novas versões do Angular possuem ainda não estavam disponíveis. Por essa razão, o

tempo de aprendizado do estagiário para o manejo dessa ferramenta foi um pouco mais longo.

2.5 Java

A linguagem orientada a objetos Java foi desenvolvida na década de 1990 por desenvolvedores da empresa Sun Microsystems. Um dos principais pontos do Java é a capacidade de escrever programas a serem executados em uma grande gama de sistemas computacionais. Em 1993, a Web explodiu em popularidade e a Sun Microsystems viu potencial em utilizar o Java também nela. A Sun Microsystems foi comprada pela Oracle em 2010 (DEITEL; DEITEL; PEARSON, 2016). O Java, hoje, é utilizado para o desenvolvimento de aplicativos corporativos de grande porte, bem como para fornecer aplicativos voltados para o consumo popular e para vários outros propósitos.

Na ZETTA, o estagiário utilizou o Java 8 para desenvolvimento do back-end. Entretanto, não se utilizou o Java em sua forma pura; utilizou-se o framework Sprint Boot, que será descrito posteriormente.

2.6 Spring Boot

O Sprint Boot é um projeto da Spring usado para facilitar o processo de configuração. Como exemplo, basta indicar quais módulos serão considerados, como Web, Persistência, Segurança ou qualquer outro, e o Spring Boot os reconhece e os configura. O principal benefício do Spring Boot é permitir que o desenvolvedor se preocupe apenas com as regras de negócio (WEISSMANN, 2021).

2.7 Spring Data JPA

O Spring Data JPA é um framework que foi criado para facilitar a implementação de repositórios baseados em JPA (*Java Persistence API*), uma interface

de persistência de dados. Implementar uma camada de acesso a dados de uma aplicação envolve codificação extensa para se fazer consultas simples. O Spring Data JPA tem a finalidade de melhorar a implementação de camadas de acesso a dados. O desenvolvedor escreve apenas as interfaces de repositório e o Spring Data JPA fará a sua implementação automaticamente (SPRING, 2015).

2.8 Hibernate

O framework Hibernate é uma ferramenta para mapeamento objeto/relacional, ele faz o mapeamento de classes Java para tabelas do banco de dados, e também os tipos de dados em Java para os tipos de dados em SQL (ORM, 2021).

2.9 Visual Studio Code

Visual Studio Code é um editor de código-fonte muito poderoso. Ele já vem com suporte integrado para Node.js, JavaScript e TypeScript e também possui uma vasta gama de extensões que facilitam o desenvolvimento (CODE, 2021). Na ZETTA, esse editor foi utilizado para desenvolvimento do front-end dos projetos, devido à produtividade por ele proporcionada.

2.10 Intellij IDEA

IntelliJ IDEA é um ambiente de desenvolvimento integrado para linguagens baseadas em JVM, criado para otimizar a produtividade no desenvolvimento. Ele fornece auxílio em tarefas repetitivas e rotineiras, fornecendo o preenchimento de código inteligente, a análise de código estático e as refatorações. Isso permite que o desenvolvedor tenha um grande ganho de produtividade (S.R.O, 2021). Para o desenvolvimento do back-end, o estagiário utilizou o IntelliJ IDEA, em virtude das grandes vantagens por ele oferecidas como assistência para codificação, navegação rápida, análise inteligente de erros e refatorações.

2.11 *Planning Poker*

Planning Poker é uma técnica de estimativa utilizada em metodologias ágeis que consiste na obtenção de estimativa de estórias de usuário através de um jogo de cartas. Sua principal ideia é permitir que todos os membros do time de desenvolvimento participem expondo sua visão de complexidade, levando em conta o fator de tempo e esforço para pontuar a estória de usuário, e após juntos chegarem a um consenso. Os seguintes procedimentos devem ser tomados para estimar uma estória (RITTER, 2014).

1. Pontua-se a estória que mais se aproxima do valor três, para servir de referência;

2. O Product Owner descreve para o time Scrum a estória fornecendo informações a respeito de seu valor de negócio, e em seguida pergunta qual o esforço necessário para concluir a estória aos membro do time;

3. Cada membro do time devem estimar a estória considerando todas as tarefas envolvidas pelos responsáveis em desenvolver, testar, criar design, etc. Então, deve-se escolher uma carta no baralho de acordo com o valor de sua estimativa e vira-lá para baixo;

4. Deve-se revelar as cartas simultaneamente quando todos os membros terminarem o procedimento acima;

5. Todos avaliam os resultados e verificam se houve convergência entre as cartas reveladas;

6. Caso haja divergência entre os valores, o Scrum Master solicita aos membros que expliquem o motivo que os levaram a tal estimativa. Então, uma nova rodada é realizada até que as estimativas de esforço cheguem a uma convergência;

7. A estimativa final da estória será o valor que tiver maior ocorrência, e então, uma nova rodada se inicia com a leitura da próxima estória dos itens do Product Backlog.

Para usar o *planning poker*, os integrantes do time de desenvolvimento davam uma nota para a atividade usando a sequência de fibonacci. A estimativa do tempo gasto para cada atividade era feita pelos integrantes do time de desenvolvimento mais experientes.

3 ATIVIDADES DESENVOLVIDAS

Este capítulo aborda as atividades realizadas pelo estagiário durante o período de estágio na agência ZETTA. Descreve-se, aqui, o desenvolvimento vivido pelo estagiário, que contempla desde o período de treinamento até a atuação no SOUT-PA (Sistema de Outorga de Direito do Pará). Apresentam-se mais detalhadamente alguns aspectos específicos de implementação que, do ponto de vista do estagiário, foram importantes para o desenvolvimento das atividades e também para o seu crescimento profissional.

3.1 Sistema de gerenciamento de reserva de salas

Inicialmente, o estagiário passou por um período de treinamento que tinha o objetivo de prepará-lo para trabalhar no sistema de outorga do Estado do Pará. Ele participou de diversos workshops que visavam transmitir a cultura da agência e, em seguida, realizou uma série de cursos relacionados às tecnologias e metodologias que seriam usadas posteriormente. Também, como parte do treinamento, foi desenvolvida uma aplicação para gerenciamento das reservas de salas para os mais diversos fins, como reuniões e palestras. Esse sistema seria utilizado internamente na agência.

Para reservar uma sala, antes do desenvolvimento dessa aplicação, era necessário ir até um colaborador responsável por fazer a reserva, o qual verificava quais salas e horários estavam disponíveis em uma agenda de papel para, então, realizar a reserva. O projeto desenvolvido tinha a finalidade de tornar esse processo totalmente digital e automático. Essa aplicação beneficiaria tanto o responsável por fazer as reservas de sala, quanto todos os outros colaboradores que necessitam desse serviço.

Para a realização desse projeto, o estagiário foi alocado para um time de desenvolvimento que tinha como integrantes um Scrum Master, um Product Owner, um desenvolvedor júnior, dois testers e outros dois estagiários. O estagiário,

ao lado do desenvolvedor júnior, ficaram responsáveis pelo desenvolvimento do back-end. As regras de negócio e as histórias de usuário foram fornecidas pelo Product Owner. Com isso, foi possível definir os passos para o desenvolvimento da aplicação. Esse projeto foi desenvolvido em duas Sprints, e cada uma teve duração de quinze dias.

3.1.1 Back-end

Para o desenvolvimento do back-end, as tecnologias utilizadas foram o framework Spring Boot e o SGBD PostgreSQL. A escolha dessas tecnologias deve-se ao fato de que elas seriam posteriormente utilizadas no desenvolvimento do SOUT-PA (Sistema de Outorga do Estado do Pará) e também para futuros projetos.

Para gerar uma estrutura inicial da aplicação, foi usado o Spring Initializr[1] que é uma API que fornece uma interface simples e rápida para a criação de projetos Spring, pois ela já vem com várias pré-configurações.

3.1.2 Configurações do banco de dados

Depois de criado o projeto, o próximo passo foi fazer as configurações para se conectar ao banco de dados. Isso, inicialmente, foi um problema para o estagiário, pois, pela sua falta de experiência em executar projetos desde o inicio, foi necessário um estudo para o entendimento do que são as dependências, por que são necessárias e como utilizá-las. Depois desse aprofundamento teórico, adicionou-se uma dependência ao arquivo pom.xml como mostrado na Figura 3.1. Essa dependência adiciona o driver de conexão do banco de dados PostgreSql.

[1] https://start.spring.io/

Figura 3.1 – Dependências para configuração e conexão com o banco de dados

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
<dependency>
```

Fonte: do autor (2021).

Em seguida, dentro do arquivo 'application.properties', foram feitas as configurações para se conectar ao banco de dados. Esse também foi outro problema enfrentado pelo estagiário, uma vez que era a primeira vez que ele precisou realizar esse tipo de configuração. As configurações estão ilustradas na Figura 3.2. Na primeira linha, está a URL de conexão do banco de dados PostgreSQL. Na segunda e na terceira linhas estão, respectivamente, o nome de usuário e senha para se conectar ao banco de dados.

Figura 3.2 – Configurações para conexão com o banco de dados

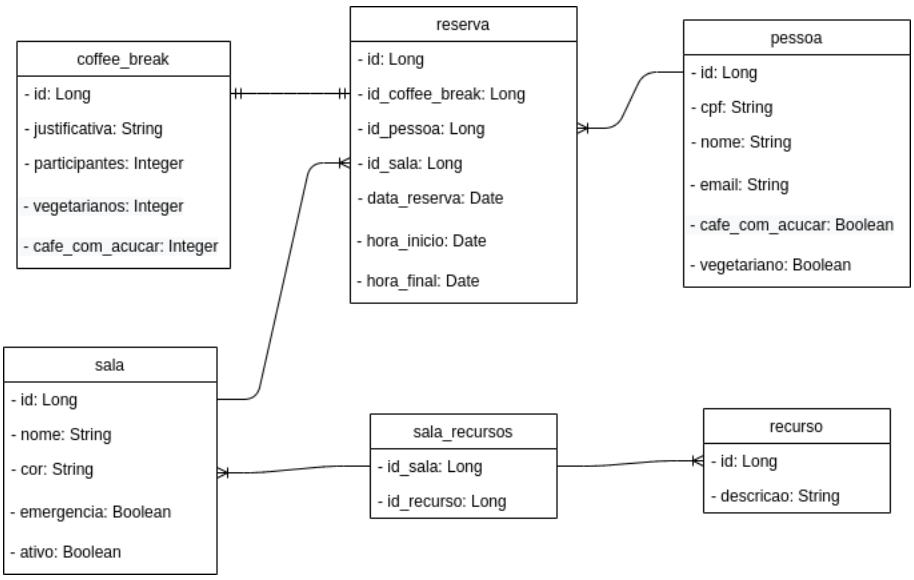
```
spring.datasource.url=jdbc:postgresql:
                        //localhost:5432/reserva_sala
spring.datasource.username=postgres
spring.datasource.password
```

Fonte: do autor (2021).

3.1.3 Criando e mapeando entidades

O terceiro passo foi criar as entidades e fazer o seu mapeamento com o banco de dados. No decorrer do andamento do projeto, novas ideias surgiram, assim como foram percebidas alterações para ser realizadas. Por isso, o estagiário recorrentemente fazia adaptações nas entidades. A Figura 3.3 ilustra o diagrama entidade-relacionamento ao final do desenvolvimento do Sistema de Reserva de Salas. Nela constam os nomes das classes e do seus atributos, bem como as chaves estrangeiras usadas para fazer o relacionamento entre as tabelas do banco de dados.

Figura 3.3 – Diagrama entidade relacionamento do sistema de reserva de salas



Fonte: do autor (2021).

A Figura 3.4 exemplifica como foi mapeada uma das entidades usadas no sistema. A anotação '@Entity' indica que está classe é uma entidade, a anotação '@Table', com o atributo 'name', mostra em qual tabela do banco de dados essa entidade é mapeada, e a anotação '@Column' relaciona os atributos da classe com as colunas do banco de dados.

Figura 3.4 – Exemplo de mapeamento de uma entidade

```
1   @Entity
2   @Table(name = sala)
3   public class Pessoa {

4       @Column(name="nome")
5       private String nome
6   }
```

Fonte: do autor (2021).

O relacionamento entre as entidades foi feito utilizando as chaves estrangeiras das tabelas do banco de dados. Esse foi um aspecto sobre o qual o estagiário não tinha experiência, portanto essa parte do desenvolvimento levou um

tempo maior para ser executada, já que um estudo foi necessário para um maior entendimento desse tópico.

A Figura 3.5 ilustra como foram feitos os relacionamentos relativos à entidade 'Reserva'. Para a relação entre 'Reserva' e 'Sala' foi usada a anotação @ManyToOne, pois várias reservas podem ser feitas para uma mesma sala. Para a relação entre 'Reserva' e 'Pessoa' também se utilizou a anotação @ManyToOne, pois várias reservas podem ser feitas por uma mesma pessoa. Já para a relação entre 'Reserva' e 'CoffeeBreak' foi utilizada a anotação @OneToOne, pois, para cada reserva feita, poderia haver apenas um coffee break.

Figura 3.5 – Mapeamento da classe Reserva

```
1   @ManyToOne
2   @JoinColumn(name="id_sala", referencedColumnName="id")
3   private Sala sala;

4   @OneToOne
5   @JoinColumn(name="id_coffee_break",
6               referencedColumnName="id")
7   private CoffeeBreak coffeeBreak;

8   @ManyToOne
9   @JoinColumn(name="id_pessoa", referencedColumnName="id")
10   Private Pessoa pessoa;
```

Fonte: do autor (2021).

A Figura 3.6 ilustra como foi feito o relacionamento entre as entidades 'Sala' e 'Recurso'. A anotação utilizada foi a @ManyToMany, porque uma sala pode ter vários recursos e um recurso pode ser utilizado em várias salas. Esse caso é um pouco diferente dos demais, pois uma tabela intermediária foi criada para manter os relacionamentos entre as chaves. O framework Hibernate cria esta tabela automaticamente, porém mais atributos são inseridos na anotação '@JoinTable' para personalizar o nome da tabela e dos seus atributos.

Figura 3.6 – Relacionamentos da classe Sala

```
1   @ManyToMany
2   @JoinTable(schema="reserva_sala",
3   name="sala_recursos",
4   joinColumns = @JoinColumn(name="id_sala",
5   referencedColumnName="id"),
6   inverseJoinColumns = @JoinColumn(name="id_recurso",
7   referencedColumnName="id"))
8   private List<Recurso> recursos;
```

Fonte: do autor (2021).

A anotação @JoinTable recebeu os seguintes atributos:

- schema: Esse atributo indicou qual esquema estava sendo utilizado;
- name: Com esse atributo, foi possível passar o nome da tabela intermediária 'sala_recurso';
- joinColumns: Por meio deste atributo, foi passado o nome da coluna da tabela intermediária 'id_sala', bem como o nome da coluna da tabela "sala"a que queríamos referenciar, que, no caso específico, foi a coluna 'id';
- inverseJoinColumns: Por meio deste atributo, também foi passado o nome da coluna da tabela intermediária 'id_recurso', bem como se identificou qual coluna da tabela secundária "recursos"foi referenciada por 'id'.

3.1.4 Repositórios

Para inserir, consultar, atualizar e deletar os dados no banco de dados, foi utilizado o framework Spring Data JPA. Esse framework facilita essas operações, pois já vem com algumas funcionalidades implementadas, além de auxiliar no processo de criação de outras operações sem o uso de comandos SQL.

Para utilizar o Spring Data JPA, foi necessário criar interfaces anotadas com '@Repository' e elas deveriam estender a interface JpaRepository para se poder utilizar as suas funcionalidades.

A Figura 3.7 exemplifica como o repositório da entidade 'Sala' foi criado. A anotação '@Repository' indica que aquela interface é um repositório. Na segunda linha, a interface 'PessoaRepository' estende a interface 'JpaRepository' para herdar seus métodos. Fazer isso permite que alguns métodos comuns do banco de dados já possam ser usados. Na quarta linha, foi criado um método para consultar um objeto 'Pessoa' usando o e-mail, o que foi realizado a partir do uso de pré-configurações do Spring Data JPA.

Figura 3.7 – Exemplo de criação de repositório

```
1  @Repository
2  public interface PessoaRepository
3                   extends JpaRepository<Pessoa, Long> {

4      Optional <Pessoa> findByEmail(String email);

5  }
```

Fonte: do autor (2021).

3.1.5 Controle, Serviços e rotas

A fim de que aplicações externas consumam os serviços criados no backend, foi necessário disponibilizar rotas para o uso desses serviços. Para isso, foram criadas classes Controller com seus devidos métodos e rotas. A função das classes Controller é a de mediar as requisições recebidas em um de seus métodos e, posteriormente, delegar para que as classes Services façam toda a lógica do serviço.

A Figura 3.8 exemplifica como foi criada a Controller 'ReservaController'. A anotação '@RestController', usada na primeira linha, indica que aquela classe é uma Controller. Na terceira linha, é utilizada a anotação '@PostMapping', juntamente com a rota de acesso a esse serviço. Por esse método utilizado, uma requisição post, que é um verbo HTTP, é usado para criar um novo registro. Na quinta linha, é chamada a Service para realizar toda a lógica de cadastrar ou editar um dado.

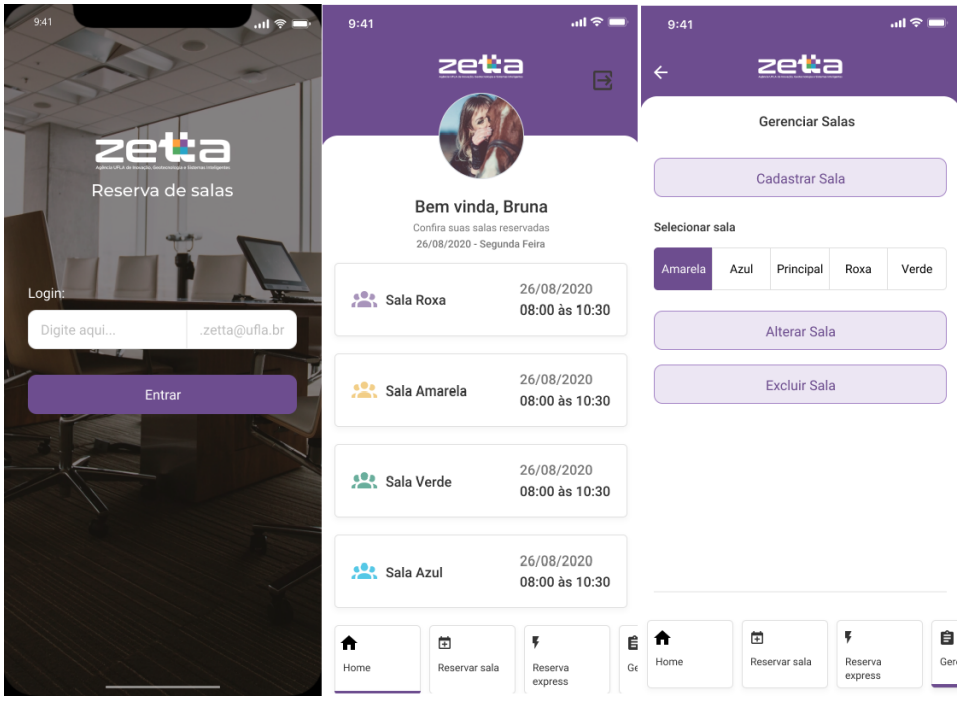
Figura 3.8 – Exemplo de classe controladora

```
1   @RestController
2   public class ReservaController {

3       @PostMapping(value = "reserva/inserir")
4       public Reserva inserir(@RequestBody Reserva reserva){

5           return serviceReservaSalaImp
6                   .cadastrarOuEditarReservaSala(reserva);
7       }
8   }
```

Fonte: do autor (2021).

A Figura 3.9 ilustra o resultado final do sistema de reserva de salas, depois da integração com o front-end desenvolvido pelos outros integrantes do time de desenvolvimento. À esquerda está a tela de login para acesso ao sistema, no centro se encontra a tela de boas vindas e à direita está a tela de gerenciamento de salas.

Figura 3.9 – Sistema de Reserva de Salas



Fonte: ZETTA (2020).

Todo esse período de treinamento teve como objetivo preparar o estagiário para trabalhar no SOUT-PA e em futuros projetos. Ao término do treinamento, o estagiário foi alocado para outro time de desenvolvimento que era composto por dois testers, um desenvolvedor e um Product Owner.

3.2 Sistema de outorga do Estado do Pará (SOUT-PA)

Antes que o estagiário começasse a trabalhar no SOUT-PA, ele passou um tempo entendendo o sistema e a rotina do novo time de desenvolvimento. Primeiramente, ele participou de um workshop, apresentado pelo Product Owner. Nele, foram apresentados todos os fluxos do sistema e uma ideia geral de como o mesmo deveria ficar. Como a demanda estava perto do prazo de entrega, muitas funcionalidades e regras de negócio já estavam implementadas. Portanto, os problemas recebidos pelo estagiário envolviam, em sua maioria, realizar pequenos ajustes e solucionar erros no SOUT-PA.

Após o workshop, o estagiário participou do planejamento das atividades a serem executadas na próxima Sprint. Para isso, todos os integrantes do novo time de desenvolvimento se reuniram para analisar as atividades contidas na Product Backlog e escolher quais delas passariam para a Sprint Backlog. Nessa etapa, as atividades eram explicadas com bastante profundidade e qualquer dúvida era imediatamente respondida.

Depois de selecionadas as atividades para a Sprint, passa-se à estimativa da complexidade e do tempo gasto para se realizar cada atividade. Para se medir a complexidade da tarefa, foi utilizada uma ferramenta chamada planning poker. Essa ferramenta já vinha sendo usada pelo novo time de desenvolvimento e tinha o intuito de priorizar as tarefas mais importantes e fazer estimativas do esforço exigido para executá-las.

O estagiário também participava de reuniões diárias com todos os integrantes do time de desenvolvimento. O objetivo dessas reuniões era fazer com

que todos falassem sobre as atividades realizadas no último dia de trabalho, o que pretendiam fazer no próximo dia e se tinha algo impedindo seu trabalho.

3.2.1 Visão geral do SOUT-PA

O SOUT-PA é um sistema desenvolvido para o Estado do Pará que tem como objetivo analisar pedidos de Outorga de Direito. A Outorga de Direito é o instrumento legal que tem como objetivo realizar o controle quantitativo e qualitativo do uso das águas. Ele também assegura ao usuário o direito de utilizar os recursos hídricos. No entanto, essa autorização não dá ao usuário a propriedade da água, mas, sim, meramente o direito de seu uso.

Os usuários realizam o cadastro do pedidos de outorga de direito através do SIGERH-PA (Sistema de Gerenciamento de Recursos Hídricos do Pará) [2]. Após esse cadastro, o pedido é enviado para o SOUT-PA fazer a sua análise. O sistema conta com dois tipos de perfis, cada um com suas responsabilidades dentro da aplicação:

- Perfil Técnico: Esse perfil tem como finalidade analisar os dados do pedido recebido e elaborar um parecer técnico favorável ou não;
- Perfil Gestor: Esse perfil analisa o parecer emitido pelo técnico, podendo aprová-lo ou não.

No SOUT-PA, há quatro tipos de processos, cada qual com uma finalidade especifica. São eles:

- Análise: Processo para um novo pedido de título de outorga de direito;
- Alteração: Pedido de alteração de alguma característica de um título de outorga de direito;

[2] Disponível em: http://sistemas.semas.pa.gov.br/sigerhpa/

- Renovação: Pedido para prolongar o uso de um título de outorga de direito que esteja com seu tempo de uso em vias de escoamento;
- Cancelamento: Pedido para cancelar um título de outorga de direito existente.

À medida que os processos são analisados, eles têm os seus estados alterados, podendo estar em uma das seguintes formas:

- Aguardando análise: aguardando para que o técnico inicie a análise do processo;
- Em análise: O processo está sendo analisado pelo técnico;
- Aguardando aprovação: aguardando que o gestor aprove ou não o parecer técnico;
- Aguardando reanálise: aguardando que o processo seja reanalisado pelo técnico;
- Análise finalizada: gestor aprovou o parecer do técnico;
- Aguardando informações complementares: aguardando maiores informações do usuário que solicitou o pedido de outorga de direito.
- Deferido aguardando pagamento: o processo foi aprovado pelo gestor e está aguardando pagamento para a emissão de título.

A Figura 3.10 mostra a tela principal do SOUT-PA, na qual todos os processos são listados e onde se encontram os filtros para cada tipo e estado dos processos. Os filtros podem ser combinados para uma melhor filtragem. Por exemplo, pode-se selecionar os filtros de 'Renovação' e 'Análise finalizada' para listar todos os processos que já possuam a análise de renovação finalizada.

Figura 3.10 – Tela principal do Sistema de Outorga do Pará.

Fonte: ZETTA (2020).

3.2.2 Fluxos

Processos do tipo "análise"são os mais básicos do sistema, pois todos os outros se baseiam neles. Quando um processo é recebido, ele fica em espera até que um técnico inicie a sua análise. Ao analisá-lo, o técnico pode dar um parecer favorável ou desfavorável, o qual é encaminhado para o gestor aprová-lo ou desaprová-lo. Nesse cenário, podem haver quatro possibilidades:

- Parecer técnico favorável e aprovação do gestor: nesse caso, a análise é finalizada e o processo fica à espera do pagamento do DAE (Documento de Arrecadação Estadual) para emissão do titulo de direito da outorga;

- Parecer técnico favorável e reprovação do gestor: nesse caso, quando o gestor reprova uma análise técnica, uma justificativa é gerada, e o processo retorna ao técnico para reanálise;

- Parecer técnico desfavorável e aprovação do gestor: nesse caso, a análise do processo é finalizada e o título de direito de outorga não é emitido;

- Parecer técnico desfavorável e reprovação do gestor: nesse caso, processo também volta ao técnico para reanálise.

Quando um pedido de direito de outorga é aprovado, esse título possui um tempo de validade. Caso o usuário queira estender esse tempo, ele precisará fazer um pedido de renovação. Em processos do tipo "renovação", o fluxo será o mesmo dos processos do tipo "análise"anteriormente citados, com a diferença de que, ao fazer o pedido de renovação, o usuário poderá fazer alterações ou não em relação ao título original. Caso seja aprovado, o título de direito de outorga será estendido por mais um tempo.

Também há processos do tipo "alteração", que têm a finalidade de alterar alguma característica do título de direito de outorga. Por exemplo, se, em relação a um título de outorga permite o uso de 10.000 m³/dia de água, o seu titular deseje aumentar esse uso para 20.000 m³/dia, ele terá que dar início a um processo de alteração.

Por fim, existem os processos do tipo "cancelamento"para o caso em que o usuário queira cancelar seu título de direito de outorga. Os processos do tipo cancelamento passam exatamente pelas mesmas etapas de um processo do tipo análise.

3.2.3 Atividades desenvolvidas

Antes de iniciar as atividades, o estagiário levou aproximadamente uma semana para preparar seu ambiente de desenvolvimento, o que incluiu instalar o Intellij IDEA como ambiente de desenvolvimento do back-end e o Visual Studio Code para desenvolvimento do front-end, popular o banco de dados, além de instalar vários outros programas necessários e fazer as configurações do projeto em sua máquina.

No início, em alguns momentos do estágio, o estagiário fez programação em par com um desenvolvedor mais experiente para aprender como solucionar

problemas mais complexos. Para isso, uma atividade era resolvida em conjunto. Em alguns momentos o estagiário acompanhava o desenvolvedor mais experiente desenvolver alguma solução, e em outros era o inverso, onde o estagiário resolvia a atividade com o auxílio do desenvolvedor mais experiente. Como o estágio foi realizado em home office, uma sala virtual do meet[3] era aberta, e cada um compartilhava sua tela do computador a medida em que codificava.

As primeiras atividades indicadas para o estagiário eram relativamente simples, como correções de erros no sistema, que incluíam troca de tooltip, correções ortográficas e alterações no comportamento do cursor. O principal objetivo dessas atividades iniciais era fazer com que o estagiário conhecesse melhor o sistema e também permitir que ele aprendesse como identificar os problemas na interface do sistema e localizá-los no código-fonte. À medida que o estagiário foi ganhando experiência, as atividades recebidas por ele foram gradualmente ficando mais complexas.

O próximo problema resolvido pelo estagiário foi um problema que se caracterizava pelo fato de que, ao se fechar uma modal de preenchimento de um formulário e depois abri-lo novamente, os dados do formulário já vinham preenchidos.

Para resolver esse problema, o estagiário criou o método ilustrado na Figura 3.11. Na segunda linha, todos os dados do formulário são apagados e na terceira linha a modal é fechada. Isso faz com que, antes do fechamento da modal, os dados sejam apagados. Os métodos 'formulario.reset()' e 'closeModal.emit()' foram reaproveitados, pois já existiam. Essa solução é bastante simples, porém, para que o estagiário chegasse até ela, foi necessário um longo processo de entendimento do código e de aplicação de técnicas de depuração. Sendo assim, foi possível aprimorar habilidades que em raras oportunidades haviam sido colocadas à prova.

[3] Dispónivel em: https://meet.google.com/

Figura 3.11 – Método para limpar o formulário.

```
1   fecharLimparModal() {
2    this.formulario.reset();
3    this.closeModal.emit();
4  }
```

Fonte: do autor (2021).

Outro problema resolvido pelo estagiário foi o de criar um serviço que buscasse o título de direito de outorga para processos do tipo cancelamento. Essa tarefa envolveu inicialmente a necessidade de se encontrar no banco de dados qual tabela possuía o título de direito de outorga e identificar os relacionamentos entre essa tabela para, então, saber o caminho a se percorrer para fazer essa consulta no banco de dados.

Após essa etapa, foi criado um método, conforme se ilustra na Figura 3.12. Esse método recebe como parâmetro o número do identificador único do processo cujo número do título se deseja. Em seguida, foi criado o objeto do tipo 'Análise', o qual recebeu a análise do processo através de uma consulta no banco de dados usando o identificador único do processo. E, por fim, de posse do identificador único da análise, uma nova consulta é feita para se obter o objeto "título", que é retornado na sétima linha.

Figura 3.12 – Serviço para busca de título de direito de outorga.

```
1   public Titulo findByProcessoId(Integer id) {

2   Analise analise = analiseRepository
3                     .findByProcessoId(processoRepository
4                     .findById(id).getId());

5   Titulo titulo = tituloRepository
6                   .findByAnaliseId(analise.getId());

7   return titulo;
8   }
```

Fonte: do autor (2021).

Para consumir o serviço de busca de título de direito de outorga criado no back-end, no front-end foi criado um método, conforme se mostra na Figura 3.13. Esse método consiste em passar como parâmetro o identificador único para o serviço que irá fazer uma requisição ao serviço criado no back-end citado anteriormente. Na quarta linha, a variável 'this.titulo' recebe os dados resultantes da requisição. Com isso, esses dados ficam disponíveis para uso do front-end de modo a serem utilizados na interface do usuário.

Figura 3.13 – Método para consumir o serviço de busca de título de direito de outorga.

```
1   buscarTitulo() {
2       this.tituloProcessoService
3           .getByProcessoId(this.processo.id)
4           .subscribe( data => this.titulo = data,
5     );
6  }
```

Fonte: do autor (2021).

O próximo problema resolvido pelo estagiário foi relacionado à importação de processos. O SOUT-PA recebe processos do SIGERH-PA para fazer a sua análise. Então, ele precisa chamar um serviço de importação dos processos do SIGERH-PA de maneira regular. Um serviço que importava processos do tipo cancelamento deixou de funcionar depois de um merge realizado no SOUT-PA.

Ao depurar o código, o estagiário descobriu que o problema estava na rota que chamava o serviço de importação de processos, pois essa rota não estava completa. Para a correção desse problema, o estagiário entrou em contato com os integrantes do SIGERH-PA, que é o sistema de onde o SOUT-PA recebe os processos. Assim, a rota correta foi fornecida e o problema foi corrigido.

Houve outro problema ao gerar um documento do parecer técnico. Uma imagem necessária no documento aparecia no ambiente de desenvolvimento mas não no ambiente de teste do SOUT-PA. Esse problema era causado devido ao diretório em que a imagem estava. Para fazer com que essa imagem fosse carregada independentemente de onde estava localizada, foi usada a Base64 da ima-

gem. Base64 é um método para codificação de dados para transferência de dados binários em forma de texto pela Internet.

4 CONSIDERAÇÕES FINAIS

O período do estágio foi de suma importância para o discente, pois, nele, foi possível adquirir experiência profissional e também que se vivenciasse, na prática, vários conhecimentos adquiridos durante a graduação. A partir da experiência de estágio na Agência ZETTA, o estagiário pôde observar alguns aspectos do estágio que chamaram a sua atenção e houve a possibilidade de se fazer uma comparação entre os conhecimentos adquiridos durante a graduação e a sua utilidade nas atividades práticas em um ambiente profissional.

O primeiro ponto observado foi o aprendizado que o desenvolvimento de um novo sistema a partir do início propiciou. Embora o Sistema de Reserva de Salas, desenvolvido na fase de treinamento, fosse um projeto de pequena escala, ele ajudou de maneira substancial o estagiário a fortalecer o conhecimento envolvido nos cursos e workshops promovidos pela empresa, bem como nos conceitos aprendidos em algumas disciplinas estudadas na graduação. Tais habilidades foram essenciais para a realização do estágio como um todo.

Projetos práticos propostos por algumas disciplinas ajudaram o estagiário a desenvolver o pensamento critico e voltado à busca por soluções. Isso também foi um fator essencial, pois permitiu que o estagiário buscasse soluções para os desafios que surgiam de forma mais autônoma e independente. Entretanto, vale salientar que os colegas de equipe sempre davam suporte quando necessário, o que sempre se configurou como imprescindível.

O trabalho em equipe foi um dos aspectos do estágio que serviu de grande aprendizado. Trabalhar com pessoas mais experientes acelerou o aprendizado na solução das demandas técnicas e das atividades realizadas. Outro ponto observado foi o esforço conjunto do time para atingir um objetivo em comum, melhorando consideravelmente a produtividade da equipe.

Um ponto diferente vivenciado pelo estagiário foi o home office, pois, ao se trabalhar em casa, muitos desafios foram enfrentados. Além disso, adaptações

necessárias, tanto mentalmente quanto estruturalmente, como a autodisciplina, tiveram de ser desenvolvidas para manter o foco no trabalho e não ceder às distrações. Estruturalmente, os desafios percebidos foram o espaço alocado para o trabalho, problemas com conexão de internet e os ruídos diversos.

Os conceitos de orientação a objetos explorados na disciplina de Programação Orientada a Objetos auxiliaram no desenvolvimento do back-end dos sistemas, pois o trabalho com os frameworks utilizados para desenvolvimento do back-end dos sistemas utilizam os conceitos do paradigma de orientação a objetos.

O primeiro contato com uma metodologia de gerenciamento de projetos ocorreu somente através do estágio, usando o Scrum. Uma disciplina obrigatória sobre gerenciamento de projetos poderia ser uma melhoria na matriz curricular do curso de Engenharia de Controle e Automação, uma vez que a grande gama de campos de atuação para os formandos deste curso envolve, em algum nível, o trabalho com projetos.

Uma disciplina obrigatória de introdução aos conceitos básicos de bancos de dados também poderia ser uma melhoria na matriz curricular do curso de Engenharia de Controle e Automação, pois tanto no desenvolvimento Web, quanto na automação em geral, muitas aplicações envolvem, de certa forma, armazenamento e recuperação de dados.

Outro aspecto importante foi a quantidade de ferramentas que o estagiário precisou aprender. A vivência na ZETTA e a utilização de ferramentas usadas por uma agência de inovação ligada à Universidade e que se relaciona com empresas públicas e privadas forneceu ao estagiário a oportunidade de enriquecer a sua formação profissional. As ferramentas, linguagens e metodologias usadas no estágio, como Scrum, Spring Boot, Intellij IDEA, Java, Angular, entre outros, possuem grande importância para o momento atual na área de tecnologia da informação.

O estágio foi a primeira experiência profissional formal do discente e serviu como porta de entrada para o mercado de trabalho. Devido ao bom desempenho do estagiário e às suas contribuições para o funcionamento da agência, houve o reconhecimento da mesma, efetivando o estagiário como assistente de analista de sistemas.

Levando-se em conta o que foi observado, pode-se concluir que o estágio propiciou um grande crescimento, tanto pessoal como profissional, para o discente, pois, além de abrir as portas para o mercado de trabalho, o discente agora carrega consigo toda uma bagagem das experiências práticas vividas na área de desenvolvimento Web, o que tem papel crucial para ajudar estagiário a decidir qual carreira seguir dentro de suas possibilidades.

REFERÊNCIAS

CARVALHO, B. V. d.; MELLO, C. H. P. **Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica**. **Gestão Produção**, scielo, v. 19, p. 557 – 573, 00 2012. ISSN 0104-530X. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000300009&nrm=iso>.

CODE, V. S. **IntelliJ IDEA overview**. 2021. Disponível em: <https://spring.io/projects/spring-data-jpa#overview>. Acesso em: 18 fev. 2021.

CRUZ, F. **Scrum e PMBOK unidos no Gerenciamento de Projetos**. 1. ed. [S.l.]: BRASPORT, 2013. ISBN 9788574526102.

DEITEL, H.; DEITEL, P.; PEARSON, E. **Java®: COMO PROGRAMAR**. 10. ed. [S.l.]: PEARSON UNIVERSIDADES, 2016. ISBN 8543004799.

ELMASRI, R.; NAVATHE, S. **Sistemas de banco de dados**. São Paulo: Pearson, 2018. ISBN 9788543025001.

FERREIRA, M. B. **Metodos ageis e melhoria de processos**. 1. ed. Curitiba: Contentus, 2020. ISBN 9786557452639.

FLANAGAN, D. **JavaScript: O Guia Definitivo**. 6. ed. Porto Alegre: Bookman Editora, 2013. ISBN 9788565837484.

GUEDES, M. **Metodologias ágil x tradicional: Quais as diferenças?** 2019. Disponível em: <https://www.treinaweb.com.br/blog/metodologias-agil-x-tradicional-quais-as-diferencas/>. Acesso em: 28 fev. 2021.

GUEDES, M. **O que é o Angular e para que serve?** 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-o-angular-e-para-que-serve/>. Acesso em: 17 fev. 2021.

LEAL, G. C. L. **Linguagem, programação e banco de dados: guia prático de aprendizagem**. 1. ed. Curitiba: INTERSABERES, 2015. ISBN 9788544302583.

MOZILLA. **CSS**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em: 17 fev. 2021.

MOZILLA. **HTML básico**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Aprender/Getting_started_with_the_web/HTML_basico>. Acesso em: 17 fev. 2021.

MOZILLA. **JavaScript**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 17 fev. 2021.

ORM, H. **Guides and such**. 2021. Disponível em: <https://hibernate.org/orm/documentation/5.4/>. Acesso em: Mai. 2021.

RITTER, R. **Planning Poker e Ideal Day: Técnicas de Abordagem de Estimativa Ágil**. 2014. Disponível em: <https://www.devmedia.com.br/planning-poker-e-ideal-day-tecnicas-de-abordagem-de-estimativa-agil/31220>. Acesso em: Apr. 2021.

RODRIGUES, J. **Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER)**. 2014. Disponível em: <https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>. Acesso em: 27 fev. 2021.

SANCHES, A. R. **Disciplina: Fundamentos de Armazenamento e Manipulação de Dados**. abr. 2004. Disponível em: <https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula7.html>. Acesso em: fev. 2021.

SIQUEIRA, F. de. **Tipos de Relacionamento**. Disponível em: <https://sites.google.com/site/uniplibancodedados1/aulas/aula-7---tipos-de-relacionamento>. Acesso em: 27 fev. 2021.

SPRING. **Spring Data JPA**. 2015. Disponível em: <https://www.devmedia.com.br/spring-boot-simplificando-o-spring/31979>. Acesso em: 18 fev. 2021.

S.R.O, J. **Comparando o NoSQL ao modelo relacional**. 2021. Disponível em: <https://www.jetbrains.com/help/idea/discover-intellij-idea.html>. Acesso em: 18 fev. 2021.

VICCI, C. **Banco de Dados**. São Paulo: Pearson, 2015. ISBN 9788543006833.

WEISSMANN, H. L. **Spring Boot: simplificando o Spring**. 2021. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 18 fev. 2021.