



RODRIGO PINTO HERCULANO

**RELATÓRIO DE ESTÁGIO - DESENVOLVIMENTO DE
SISTEMA WEB PARA MANEJO FLORESTAL NA AGÊNCIA
ZETTA**

LAVRAS - MG

2022

RODRIGO PINTO HERCULANO

**RELATÓRIO DE ESTÁGIO - DESENVOLVIMENTO DE SISTEMA WEB PARA
MANEJO FLORESTAL NA AGÊNCIA ZETTA**

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

Prof. Dr. Bruno de Abreu Silva

Orientador

LAVRAS - MG

2022

RODRIGO PINTO HERCULANO

**RELATÓRIO DE ESTÁGIO - DESENVOLVIMENTO DE SISTEMA WEB PARA
MANEJO FLORESTAL NA AGÊNCIA ZETTA**

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

APROVADA em 25 de abril de 2022.

Prof. Dr. Paulo Afonso Parreira Junior UFLA
Prof. Dra. Renata Teles Moreira UFLA

Prof. Dr. Bruno de Abreu Silva
Orientador

**LAVRAS - MG
2022**

*Dedico aos meus pais Adalberto e Elaine que me proporcionaram essa etapa importante da
minha vida.*

AGRADECIMENTOS

Agradeço a todos que estiveram do meu lado me dando suporte durante essa caminhada.

RESUMO

Atualmente, boa parte do sistema de manejo florestal realizado pelo Estado de Mato Grosso do Sul consiste em processos manuais burocráticos e lentos. No intuito de facilitar e agilizar tais processos, o sistema web Licenciamento Ambiental, mantido pela Agência Zetta, permite que o usuário faça o cadastro da sua solicitação de licenças ambientais online. O presente relatório apresenta o estágio supervisionado na Agência Zetta, onde o discente atuou desenvolvendo este sistema. O discente atuou em atividades frontend e backend utilizando tecnologias como Play Framework, AngularJS e PostgreSQL. Com esse sistema, espera-se que a emissão de licenças ambientais ocorra de forma mais eficiente no Estado de Mato Grosso do Sul.

Palavras-chave: Manejo florestal. Sistema web. Play framework. AngularJS. PostgreSQL

ABSTRACT

Currently, most of the forest management system carried out by the State of Mato Grosso do Sul consists of manual, bureaucratic and slow processes. In order to facilitate and speed up such processes, the Environmental Licensing web system, maintained by Agência Zetta, allows users to register their request for environmental licenses online. This report presents the supervised internship in the Zetta Agency system, where the student advertises this. The student acted in frontend and backend activities using technologies such as Play FrameworkJS and PostgreSQL. With this system, it is expected that the issuance of environmental licenses will occur more efficiently in the State of Mato Grosso do Sul.

Keywords: Forest management, Web system. Play framework. AngularJS. PostgreSQL

SUMÁRIO

1	Introdução	9
1.1	Contextualização	9
1.2	Objetivos	9
1.3	Organização do texto	10
2	Agência Zetta	11
2.1	Organização e processos de desenvolvimento	11
3	Tecnologias utilizadas no Estágio	17
3.1	Frontend	17
3.1.1	HTML	17
3.1.2	CSS	18
3.1.3	Pug	19
3.1.4	Javascript	20
3.1.5	AngularJS	20
3.1.6	Vuejs	21
3.2	Back-end	22
3.2.1	Java	22
3.2.2	Play framework	22
3.2.3	Spring Framework	23
3.2.4	PostgreSQL	23
3.3	Scrum	23
3.3.1	Scrum Team	24
3.3.2	Scrum Events	24
3.3.3	Scrum Artifacts	25
3.4	Gitlab	26
4	Atividades desenvolvidas no Estágio	27
4.1	Desenvolvimento das Atividades	27
4.2	Licenciamento Ambiental do Mato Grosso do sul	28
4.2.1	Remoção de empreendimentos do empreendedor	28
4.2.2	Implementação do cálculo de distância dos municípios	33
4.2.3	Integração com API de pagamentos	35
5	Considerações Finais	41

LISTA DE FIGURAS

Figura 2.1 – Zetta	11
Figura 2.2 – Squad	13
Figura 2.3 – kanban	14
Figura 2.4 – Branches	16
Figura 3.1 – Código html apresentado no navegador	18
Figura 3.2 – Código HTML estilizado	19
Figura 3.3 – Código em JavaScript	20
Figura 3.4 – Scrum Framework	24
Figura 4.1 – Licenciamento Ambiental	28
Figura 4.2 – Função do <i>frontend</i> de remover empreendimento	30
Figura 4.3 – Função do <i>frontend</i> de confirmar remoção empreendimento	30
Figura 4.4 – Método de exclusão de empreendimentos	31
Figura 4.5 – Clicando no botão de ações do empreendedor	31
Figura 4.6 – Selecionando exclusão do empreendedor	32
Figura 4.7 – Empreendedor/empreendimento deletado	32
Figura 4.8 – Método para encontrar o município	33
Figura 4.9 – Método para encontrar o município a partir do <i>centroid</i>	34
Figura 4.10 – Método para calcular distância da capital	34
Figura 4.11 – Método de gerar boleto	35
Figura 4.12 – Método gerador de token	36
Figura 4.13 – Método de download da guia	37
Figura 4.14 – <i>Service</i> de download da guia	37
Figura 4.15 – Baixar guia	38
Figura 4.16 – Boleto gerado	39
Figura 4.17 – Job de verificação de pagamentos	40

LISTA DE LISTAGENS

3.1	Exemplo em HTML.	17
3.2	Exemplo em CSS.	18
3.3	Exemplo em PUG.	19
3.4	Exemplo em JavaScript.	20

1 INTRODUÇÃO

O intuito deste Capítulo é descrever e contextualizar os objetivos do estágio supervisionado realizado pelo discente na Agência Zetta, onde o mesmo atuou como desenvolvedor no projeto de Licenciamento Ambiental do estado do Mato Grosso do Sul. Além disso, o Capítulo fornece os antecedentes do estágio supervisionado e descreve brevemente a agência Zetta, os projetos envolvidos e as atividades desenvolvidas durante o estágio.

1.1 Contextualização

Durante a graduação, o discente obtém conhecimento e experiências que são de grande importância. Na sala de aula é apresentada ao discente uma grande variedade de disciplinas e conteúdos que são essenciais para a formação do mesmo. No entanto, é necessário solidificar esse aprendizado fora da sala de aula, para isso, durante a formação são disponibilizados grupos de estudos, estágio supervisionado e iniciação científica.

Porém a entrada para o mercado de trabalho não é algo trivial, pois as empresas buscam profissionais amplamente qualificados e experientes. Portanto, o estágio supervisionado é de grande valia, onde o discente é alocado para obter e adquirir conhecimento prático do conteúdo teórico e experiência, enquanto mostra domínio perante tudo aquilo que aprendeu durante toda a formação.

Neste documento, serão apresentadas as atividades e os resultados do discente que foram realizadas em seu estágio na Agência Zetta, situada dentro do campus da UFLA, onde conheceu e trabalhou no projeto de manejo florestal que tem como foco facilitar a realização de cadastros de licenciamento ambiental *online*.

O licenciamento ambiental online é importante para reduzir a burocracia e agilizar o processo de criação de licenças ambientais, permitindo que o empreendedor faça o cadastro da sua solicitação de licenças ambientais online.

1.2 Objetivos

Levando em conta que a Agência Zetta desenvolve sistemas no âmbito florestal, o discente teve como objetivo desenvolver e manter a aplicação web Licenciamento Ambiental do Mato Grosso do Sul, que tem como foco facilitar o cadastro de solicitações de licenças ambientais.

No momento em que o discente foi alocado no projeto de Licenciamento Ambiental, o sistema já estava em processo de desenvolvimento e necessitava de melhorias e da criação de novas funcionalidades. Portanto o discente foi alocado para desenvolver e manter o sistema, realizando todas as atividades em conjunto com a sua equipe. Durante o projeto, a equipe seguiu a metodologia Scrum e o discente participou junto com a equipe de todos os eventos e, além disso, utilizou sistemas de controle de versão para auxiliar na construção do sistema. E como resultado desse trabalho é apresentado neste documento as implementações realizadas que agregaram valor para o sistema final e para o conhecimento do discente.

1.3 Organização do texto

Os próximos Capítulos estão divididos em: O Capítulo 2 apresenta descrições da Agência Zetta e como é organizado o processo de desenvolvimento de *software* utilizando ferramentas de controle de versões e metodologias ágeis e a divisão de tarefas dentro da equipe em que o discente foi alocado. No Capítulo 3, são apresentadas as tecnologias, linguagens de programação e *frameworks* utilizados no projeto durante o desenvolvimento do *backend* e *frontend* da aplicação durante o estágio. Já no Capítulo 4, são apresentadas as atividades desenvolvidas pelo discente no período do estágio e, por fim, no Capítulo 5 estão as considerações finais deste documento.

2 AGÊNCIA ZETTA

A Zetta é a Agência UFLA de Inovação em Geotecnologias e Sistemas Inteligentes no Agronegócio, tendo como referência o LEMAF (Laboratório de estudos e projetos em manejo florestal). Localizada dentro do Campus da Universidade (Figura 2.1), a Agência tem o objetivo voltado para a área de agricultura digital, com foco em geotecnologias, ciência de dados e implementação de sistemas inteligentes, conduzindo inúmeros projetos relacionados ao manejo florestal. A agência é composta por mais de 170 colaboradores, tem mais de 15 anos de expertise, com mais várias soluções entregues.

De acordo com Zetta (2020), são promovidas pela Agência soluções que impactam a sociedade diretamente. São gerados conhecimentos e tecnologia para aumentar a produção de alimentos, a sustentabilidade e a eficiência de serviços para a população. Quatro pilares sustentam o propósito da Agência: criatividade, inovação, tecnologia e sociedade.

Figura 2.1 – Zetta



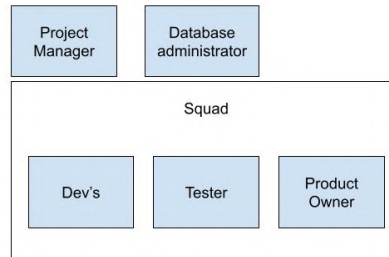
Fonte: <https://agenciazetta.ufla.br/>

2.1 Organização e processos de desenvolvimento

As equipes de desenvolvimento na agência são separadas por tribos. Atualmente existem 3 tribos distintas, sendo elas: ARA, YBY, IG. Cada tribo é responsável por seus projetos,

organizada em grupos de profissionais que são chamados de *squad*, composto por desenvolvedores, *testers* e 1 *Product Owner*. Também existem os profissionais que não são alocados para um *squad* específico mas agem em prol de todos eles, como o Gerente de Projetos e o *Database Administrator* como mostrado na Figura 2.2.

- **Desenvolvedores:** São responsáveis pelo desenvolvimento do sistema. Vários profissionais da área trabalham na empresa, mas nem todos atuam na mesma área de trabalho ou utilizam a mesma tecnologia.
- **Tester:** São responsáveis por encontrar falhas e outros tipos de problemas que não foram detectados durante o desenvolvimento do software.
- **Product Owner:** São responsáveis por apresentar novas funcionalidades, melhorias ou erros para a sua equipe, para que todos os membros sigam uma visão definida para o projeto. Ele também é responsável pela qualidade final das entregas.
- **Database Administrator:** São responsáveis por manter o banco de dados das aplicações. Eles são responsáveis pela criação, monitoramento, análise das estruturas e tabelas e reparos do banco de dados.
- **Gerente de Projetos:** São responsáveis por monitorar projetos e garantir que o projeto fique dentro do escopo da empresa. Exemplos de escopo são custo do projeto, prazo para entrega de atividades, obter recursos humanos, financeiros e gerenciar a equipe para que não haja conflitos.

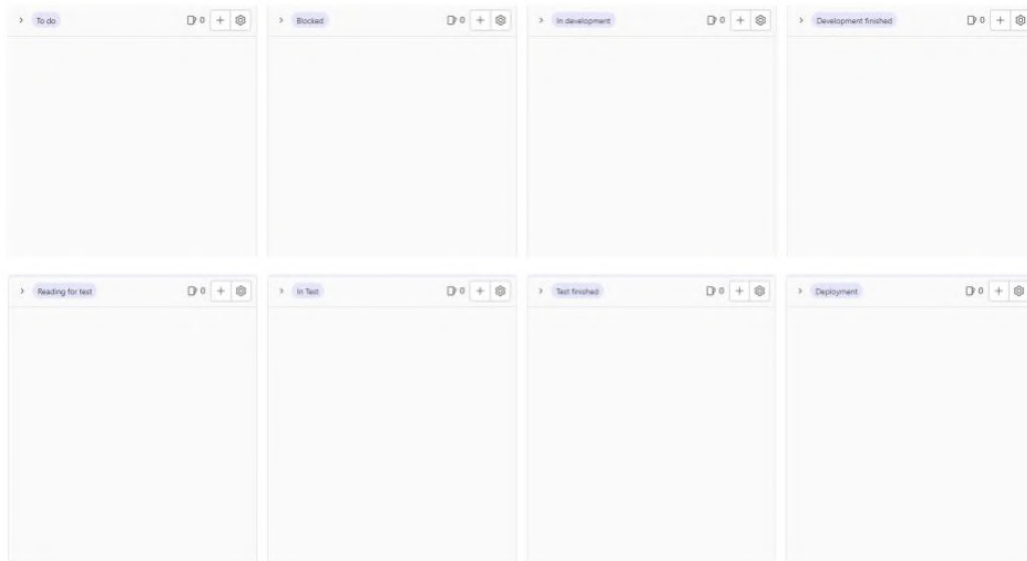
Figura 2.2 – Squad

Fonte: Autor

Com cada equipe dividida, suas atividades em conjuntos são ditadas pelo Scrum, metodologia citada mais a frente neste trabalho. No começo de cada *sprint*, os membros fazem suas reuniões de planejamentos, e adicionam atividades que serão desenvolvidas nesse tempo.

Para se organizar toda a tribo YBY faz o utilização do *framework Scrum*, onde são realizados sprints com duração de dez dias. No primeiro dia da *sprint*, a equipe mantém o foco na reunião de *planning*, onde o *Product Owner* apresenta as atividades desejadas para serem adicionadas ao *sprint backlog*. Nesta reunião, os integrantes do *squad* jogam suas estimativas em pontos utilizando o *Scrum Poker*.

Todas as atividades adicionadas na *sprint* são disponibilizadas em um *board kanban* localizado na ferramenta gitlab como mostrado na Figura 2.3.

Figura 2.3 – kanban

Fonte: Zetta (2021)

Para auxiliar os integrantes do *squad*, o *board* é organizado de forma bem intuitiva, onde cada integrante é capaz de vincular uma atividade para si. A seguir é exemplificado como o *board* é composto e o significado de cada coluna:

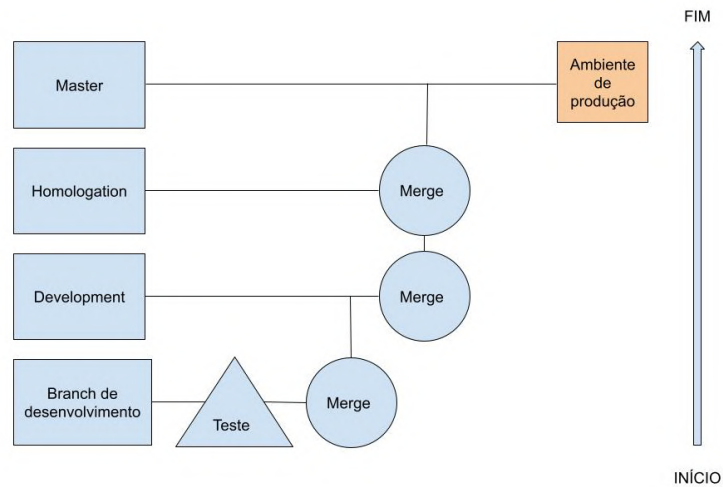
- **To do** :Nesta coluna são inseridas todas as atividades jogadas na *planning* e que estão prontas para serem desenvolvidas.
- **Blocked** : Nessa coluna são inseridas atividades que estão com algum impedimento para serem desenvolvidas.
- **In development** : Nessa coluna são inseridas atividades que já estão designadas a algum desenvolvedor e estão em andamento.
- **Development finished** : Nesta coluna estão as atividades finalizadas pelos desenvolvedores.
- **Ready for test** : Nesta coluna estão as atividades finalizadas pelos desenvolvedores e que já estão prontas no ambiente de teste.

- ***In test*** : Nesta coluna estão as atividades em que o analista de qualidade está trabalhando.
- ***Test finished*** : Nesta coluna estão as atividades finalizadas pelo *tester*.
- ***Deployment*** : Nesta coluna estão as atividades devidamente finalizadas, ou seja, foram desenvolvidas e já estão testadas, prontas para serem levadas ao ambiente de homologação.

O código do projeto é mantido em *branches* separadas utilizando a ferramenta de controle de versão gitlab, cada uma com seu propósito único, como mostrado na Figura 2.4

- ***Master*** : A *branch master* armazena o código estável da aplicação. O código armazenado nesta *branch* é o mesmo que está no ambiente de produção.
- ***Homologation*** : Esta *branch* mantém a versão estável que está em ambiente de homologação.
- ***Development*** : Esta é a *branch* principal que auxilia no desenvolvimento. Os desenvolvedores utilizam esta *branch* como base para criar as *branches* que serão utilizadas para criar novas funcionalidades e correções. É nesta *branch* que os desenvolvedores criam suas *branches* para desenvolvimento.
- ***Branch de desenvolvimento da atividade*** : Esta *branch* é utilizada pelos desenvolvedores para inserir as mudanças feitas para uma atividade específica. É nela onde o desenvolvedor insere suas mudanças no código, ao término do desenvolvimento, o desenvolvedor faz um *merge request* para a *branch* principal *development*. Esta é uma maneira de facilitar a verificação das alterações feitas, e uma forma de manter o padrão do código da aplicação, pois o *merge request* pode ser feito somente por outro desenvolvedor, seguindo um padrão de regras pré definidas.

Figura 2.4 – Branches



Fonte: Autor

Merge request é uma maneira de facilitar a verificação das alterações feitas, onde o desenvolvedor une a *branch* com as alterações feitas por ele para a *branch* principal.

O ambientes de desenvolvimentos são divididos em:

- **Local** : O ambiente local é o ambiente de desenvolvimento, onde os desenvolvedores implementam as funcionalidades.
- **Teste** : Este ambiente fica disponível apenas para o *squad* e é onde o analista de qualidade faz a avaliação das atividades desenvolvidas pelos desenvolvedores.
- **Homologação** : Este ambiente é disponibilizado para o *squad* e para o cliente. Neste ambiente são disponibilizadas as atividades já testadas do ambiente de teste, e que estão prontas para serem disponibilizadas no ambiente de produção.
- **Produção** : Neste ambiente a aplicação é disponibilizada ao público. Para chegar neste estágio as atividades desenvolvidas devem passar por todos os ambientes citados anteriormente.

3 TECNOLOGIAS UTILIZADAS NO ESTÁGIO

Este Capítulo tem como objetivo descrever e demonstrar as tecnologias, estruturas e ferramentas usadas no processo de desenvolvimento do projeto durante o período do estágio.

3.1 Frontend

Nesta seção, serão abordadas as tecnologias que foram utilizadas no desenvolvimento *frontend* e que executam em um navegador *web* no lado do cliente.

3.1.1 HTML

HTML (*HyperText Markup Language*) é a Linguagem de Marcação de HiperTexto, usada para compor a estrutura de *sites web*. De acordo com Mozilla (2021b), o HTML usa “Marcação” para anotar texto, imagem e outros conteúdos para exibição em um navegador da *Web*. A marcação HTML inclui “elementos” especiais, como `<head>`, `<title>`, `<body>`, `<header>`, `<footer>`, `<article>`, `<section>`, `<p>`, `<div>`.

O discente teve contato pela primeira vez com a linguagem na graduação, nas disciplinas de Engenharia de *Software* e *Programação Web*. O estágio incrementou os conhecimentos durante o desenvolvimento dos projetos.

A figura 3.1 é apresenta como algumas *tags* do *html* são hierarquicamente organizadas.

Na Figura 3.1 é apresentado como o código da Listagem 3.1 é exibido no navegador.

Listagem 3.1 – Exemplo em HTML.

```

1 <html>
2   <head>
3     <title>Titulo da pagina</title>
4   </head>
5   <body>
6     <h1>Titulo</h1>
7     <p>Paragrafo</p>
8   </body>
9 </html>
```

Fonte: Autor

Figura 3.1 – Código html apresentado no navegador

Titulo

Paragrafo

Fonte: Autor

3.1.2 CSS

CSS (*Cascading Style Sheets*) ou Folha de estilo em cascata são usadas para estilizar e arranjar páginas *web* — por exemplo, para alterar a fonte, cor, tamanho e espaçamento do seu conteúdo, separá-lo em múltiplas colunas, ou então adicionar animações e outras implementações decorativas (MOZILLA, 2021a).

A Listagem 3.2 exemplifica o código CSS fazendo alterações visuais em *tags* html.

Listagem 3.2 – Exemplo em CSS.

```
1 body{
2   font-style: italic;
3   font-size: 12px;
4 }
5 p{
6   color: blue;
7 }
```

Fonte: Autor

Na Figura 3.3 é apresentado como o exemplo da Figura 3.1 ficaria com o código CSS da Listagem 3.2.

Figura 3.2 – Código HTML estilizado

Titulo

Paragrafo

Fonte: Autor

3.1.3 Pug

O Pug é um mecanismo rico em recursos de alto desempenho, implementado usando JavaScript, o Pug substitui as variáveis do projeto pelos valores atuais e, a seguir, envia ao cliente a *string* HTML gerada.

O processo geral de renderização do Pug é simples. A função *pug.compile()* irá compilar o código fonte do Pug em uma função JavaScript que recebe um objeto de dados (chamado “locals”) como argumento. Chame essa função resultante com seus dados e ela retornará uma *string* em HTML renderizada com seus dados (PUGJS, 2022).

De acordo com (PUGJS, 2022) (IMASUL, 2022) o Pug também fornece a família de funções *pug.render()* que combinam a compilação e a renderização em uma única etapa. No entanto, a função do modelo será recompilada toda vez que o *render* for chamado, o que pode afetar o desempenho.

Na Listagem 3.3 é apresentada como a Listagem ?? aparenta ao utilizar-se o pug. Repara-se que o código é menos verboso e de fácil entendimento, utilizando indentação para substituir as *tags*.

Listagem 3.3 – Exemplo em PUG.

```

1 head
2   title Titulo da pagina
3 body
4   h1 Titulo
5   p Paragrafo

```

Fonte: Autor

3.1.4 Javascript

JavaScript é uma linguagem de programação leve, interpretada e orientada a objetos com funções de primeira classe, conhecida como a linguagem de *scripting* para páginas *Web*, mas também utilizada em muitos ambientes fora dos navegadores, segundo o Mozilla (2021c).

O JavaScript é uma linguagem muito poderosa que consegue manipular as ações das páginas *web* a partir de eventos disparados no navegador. Ela é responsável pelas alterações que acontecem nestas páginas, juntamente com as linguagens já citadas HTML e CSS.

Na Listagem 3.4 é apresentado é apresentado um exemplo de código na linguagem JavaScript.

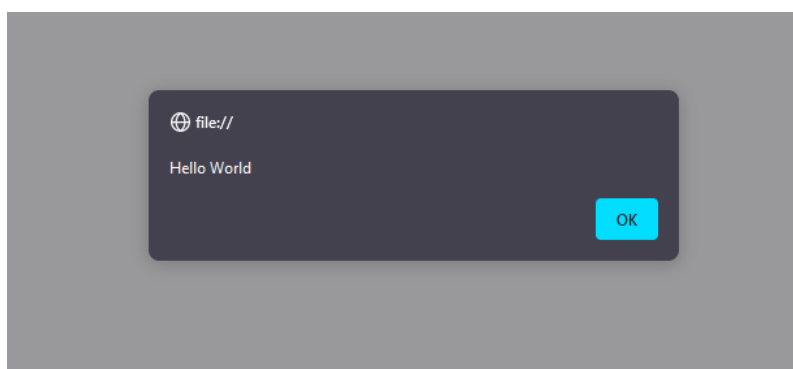
Listagem 3.4 – Exemplo em JavaScript.

```
1 function HelloWorld() {  
2     alert("Hello World");  
3 }
```

Fonte: Autor

Na Figura 3.3 é mostrada a função em JavaScript apresentada na Listagem 3.4 sendo chamada no navegador. Observe que ele cria uma caixa de alerta.

Figura 3.3 – Código em JavaScript



Fonte: Autor

O Javascript foi a linguagem mais utilizada no *frontend* pelo discente durante o estágio. No começo, o discente passou por treinamento e foi auxiliado pelos companheiros de equipe no desenvolvimento das atividades para se familiarizar com a linguagem.

3.1.5 AngularJS

AngularJS é um *framework* estrutural para aplicativos da *web* dinâmicos. Ele permite que você use HTML como sua linguagem de modelo e permite que você estenda a sintaxe do

HTML para expressar os componentes do seu aplicativo de forma clara e sucinta, segundo o AngularJS (2021).

A seguir estão listados alguns dos recursos do Angular e que foram utilizados durante o desenvolvimentos das atividades do discente:

- **Data binding** : É uma maneira de atualizar automaticamente o modelo quando a visualização muda sem a necessidade da manipulação do DOM (*Document Object Model*) da página *web*. O DOM é a página *web* em uma representação orientada a objetos, com isso podemos manipular a página utilizando a linguagem JavaScript citada anteriormente .
- **Controller** : Permite a manipulação do comportamento de uma forma clara e legível, sem a necessidade de atualizar o DOM manualmente.
- **Directives** : Diretivas possibilitam a criação de uma nova sintaxe HTML, única para a sua aplicação.

O discente utilizou o *framework* no desenvolvimento de suas atividades *frontend*. Juntamente com o Javascript, o discente foi capaz de criar telas e ações para o sistema em que foi designado. A versão utilizada pelo discente foi a versão 1.6 do *framework*, uma versão antiga que gerou alguns transtornos durante o desenvolvimento, por falta de documentações de fácil acesso. Esta versão foi utilizada pois já havia sido desenvolvido um código principal para o sistema nesta versão, e o custo para atualizar todo o código para versão mais nova do *framework* seria muito alto.

3.1.6 Vuejs

De acordo com Vuejs (2021), Vue é um *framework* progressivo para a construção de interfaces de usuário. Ao contrário de outros *frameworks* monolíticos, Vue foi projetado desde sua concepção para ser adotável incrementalmente. A biblioteca principal é focada exclusivamente na camada visual (*view layer*), sendo fácil adotar e integrar com outras bibliotecas ou projetos existentes.

O discente teve seu primeiro contato com o *framework* durante o treinamento no estágio, e completou todo o treinamento utilizando-o.

3.2 Back-end

Nesta seção será abordado as tecnologias que foram utilizadas no desenvolvimento *back-end*, executado no lado do servidor das aplicações.

3.2.1 Java

O Java¹ é uma linguagem de programação utilizada em diversos ambientes computacionais. Isso se deve ao fato de que ela foi construída para ser capaz de produzir programas que podem ser executados em diferentes sistemas computacionais. Com o crescimento da *web*, o Java também cresceu, por conta de sua alta flexibilidade e capacidade de adaptação, de acordo com Deitel e Deitel (2015).

O discente teve contato com a linguagem durante a graduação nas matérias de Paradigmas de Linguagens de Programação e Práticas de Programação Orientada a Objetos por isso já estava familiarizado. Esta linguagem foi a mais utilizada pelo discente durante todo o período do estágio.

3.2.2 Play framework

Segundo o PlayFramework (2021), o Play é baseado em uma arquitetura leve, sem estado e amigável para a web. Construído no Akka, o Play fornece consumo previsível e mínimo de recursos (CPU, memória, *threads*) para aplicativos altamente escaláveis.

O Play faz o uso da arquitetura MVC (*Model, View, Controller*) e com ele pode-se encontrar todos os componentes necessários para criar aplicativos da *Web* e serviços *REST*, como manipulação de formulários *web*, segurança de comunicação entre aplicações e outras funções. O Play auxilia no desenvolvimento utilizando carregamento instantâneo, para que o trabalho possa ser compilado e exibido imediatamente.

A arquitetura MVC e seus conceitos:

- **Model** : A sua responsabilidade é gerir e controlar o comportamento dos dados através de funções, lógicas e regras de negócio estabelecidas pelo desenvolvedor. A *model* é responsável pelo mapeamento dos dados do banco de dados para o *backend* da aplicação.

¹ <https://www.java.com/pt-BR/>

- **Controller** : A sua responsabilidade é intermediar a solicitação enviada pela *View* e a resposta fornecida pelo *Model*, processando os dados que foram enviados do *frontend* pelo usuário e transferindo para as demais camadas.
- **View** : A sua responsabilidade é apresentar as informações de forma visual, renderizando a resposta da *Model* em um formato que seja de fácil entendimento para as interações do usuário final.

3.2.3 Spring Framework

O *Spring* é um *framework open source* utilizado para plataformas Java. Segundo Spring.io (2021) o *Spring Boot* transforma a maneira como você aborda as tarefas de programação Java, otimizando radicalmente sua experiência. O *Spring Boot* combina necessidades como um contexto de aplicativo e um servidor da *Web* incorporado e configurado automaticamente para tornar o desenvolvimento de microsserviços uma tarefa fácil.

Além de ter uma vasta opções de recursos, os mais utilizados pelo discente durante o treinamento foram:

- **Spring Data JDBC** : Implementação da interação do *backend* com o banco de dados.
- **Spring Data JPA**: Interface de persistência de dados.
- **Spring MVC**: Integração com controladores padrão de arquitetura MVC.

3.2.4 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional.

O PostgreSQL é o SGBD utilizado no sistema de banco de dados da Agência e durante o estágio foi utilizado para o desenvolvimento e manutenção do banco de dados das aplicações. Além de ser um SGBD robusto, tem funções de dados geográficos que são muito utilizados pela agência e possui documentação completa e pode ser facilmente acessado pela internet.

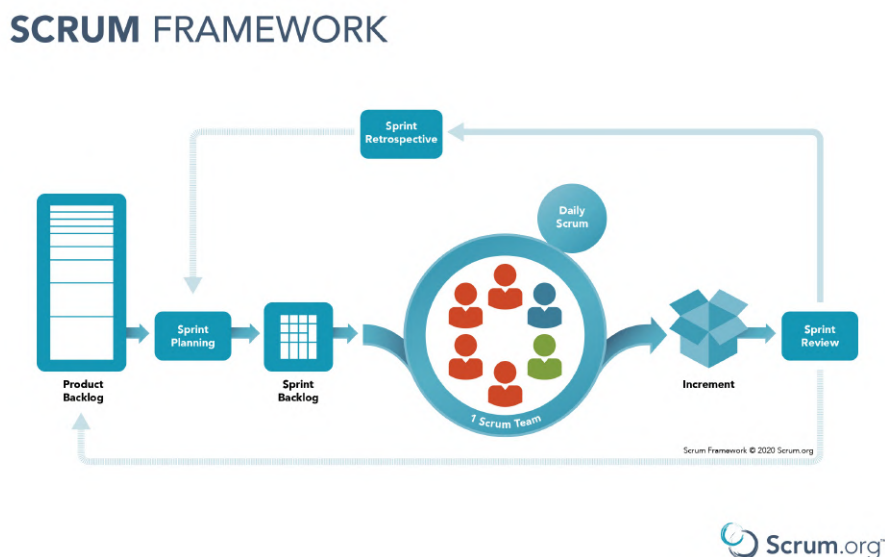
3.3 Scrum

De acordo com Scrum.org (2021), Scrum é um *framework* leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.

O discente teve seu primeiro contato teórico durante a graduação na disciplina Modelagem e Implementação de *Software*, porém a primeira experiência na prática foi durante o desenvolvimento do estágio, realizando os ritos do *scrum* do dia dia na agência.

A Figura 3.4 apresenta O *scrum* que é composto por eventos, artefatos e papéis.

Figura 3.4 – Scrum Framework



Fonte: <https://www.scrum.org/resources/scrum-framework-poster>

3.3.1 Scrum Team

A unidade fundamental do Scrum é um pequeno time de pessoas, um *Scrum Team*. O *Scrum Team* consiste em um *Scrum Master*, um *Product owner* e *Developers*. Dentro de um *Scrum Team*, não há sub-times ou hierarquias. É uma unidade coesa de profissionais focados em um objetivo de cada vez, a Meta do Produto, de acordo com Scrum.org (2021).

3.3.2 Scrum Events

- **Sprint:** De acordo com Scrum.org (2021) *Sprints* são o coração do *Scrum*, onde ideias são transformadas em valor. São eventos de duração fixa de um mês ou menos para criar consistência. Uma nova *Sprint* começa imediatamente após a conclusão da *Sprint* anterior.

Uma *Sprint* é um ciclo de duração fixa, onde todos os integrantes do time *scrum* realizam as tarefas que foram planejadas durante a *Planning*. Durante a *sprint* é feito o refinamento do *product backlog* e qualquer mudança no escopo da mesma deve ser negociado com o

Product Owner, porém nenhuma mudança que coloque o objetivo da *sprint* em risco pode ser feita, mas caso isso aconteça a *sprint* poderá ser cancelada, mas somente pelo *Product Owner*.

- **Planning:** A *Sprint Planning* inicia a *Sprint* ao definir o trabalho a ser realizado na *Sprint*. Este plano resultante é criado pelo trabalho colaborativo de todo o *Scrum Team*, segundo Scrum.org (2021).

A *Planning* é realizada no início da *sprint*, neste rito todos os integrantes do time *scrum* planejam os passos para realizar a *sprint*. Para a realização da *planning* é necessário que o *Product Owner* tenha em mãos o *Product Backlog*. Ao final do rito estarão definidas todas as atividades que serão realizadas durante toda a *sprint*. Estas atividades planejadas formam o *Sprint Backlog*.

- **Daily Scrum:** Segundo Scrum.org (2021) O propósito da *Daily Scrum* é inspecionar o progresso em direção a Meta da *Sprint* e adaptar o *Sprint Backlog* conforme necessário, ajustando o próximo trabalho planejado.

A *Daily* é uma reunião rápida que dura em torno de 15 minutos, onde os integrantes discutem as atividades que estão sendo feitas e como está o decorrer da *sprint*, identificando impedimentos dos integrantes e auxiliando na autogestão da equipe.

- **Sprint Retrospective:** De acordo com Scrum.org (2021) O propósito da *Sprint Retrospective* é planejar maneiras de aumentar a qualidade e a eficácia. O *Scrum Team* inspeciona como foi a última *Sprint* em relação a indivíduos, interações, processos, ferramentas e sua Definição de Pronto.

Nesse rito, são discutidos todos os eventos que aconteceram na última *sprint*, levantando não somente os problemas mas também as metodologias usadas que deram certo.

3.3.3 Scrum Artifacts

De acordo com Scrum.org (2021) Os artefatos do *Scrum* representam trabalho ou valor. Eles são projetados para maximizar a transparência das principais informações. Assim, todos os que os inspecionam têm a mesma base para adaptação.

- **Product Backlog:** O *Product Backlog* é uma lista ordenada e emergente do que é necessário para melhorar o produto. É a única fonte de trabalho realizado pelo *Scrum Team*, segundo Scrum.org (2021).
- **Sprint Backlog:** O *Sprint Backlog* é composto pela Meta da *Sprint* (por que), o conjunto de itens do *Product Backlog* selecionados para a *Sprint* (o que), bem como um plano de ação para entregar o Incremento (como), ainda segundo Scrum.org (2021).

3.4 Gitlab

Gitlab é um repositório de gerenciamento de *software* baseado em git. Segundo o Git (2021), Git é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar, desde projetos pequenos a muito grandes com velocidade e eficiência.

Durante todo o desenvolvimento do projeto foi utilizada a ferramenta para fazer o controle de versões. A agência tem o seu próprio servidor do Gitlab, é mantido dentro do campus da universidade.

A interface do GitLab é acessada através da *web*, tendo assim um fácil acesso das *branches* e versões utilizadas nos projetos da agência. A ferramenta contempla várias ferramentas como *wiki* dos projetos e *kanban* para auxiliar no desenvolvimentos das atividades.

4 ATIVIDADES DESENVOLVIDAS NO ESTÁGIO

Esta seção apresenta as atividades realizadas durante o estágio, assim como as dificuldades e aprendizados que cada atividade trouxe.

4.1 Desenvolvimento das Atividades

Ao iniciar o estágio, o discente foi alocado a um *squad* de treinamento onde permaneceu durante 2 meses. No primeiro mês foi apresentado ao discente palestras e cursos focados nos *frameworks* e tecnologias que seriam utilizadas no desenvolvimento dos projetos. O *squad* era liderado por um *Scrum Master* para guiar e tirar dúvidas. Os cursos apresentados ao discente como cursos introdutórios de lógica de programação, cursos focados em css e html, orientação a objetos, programação *web* utilizando Vuejs, programação *web* utilizando *Spring Boot* e curso focado em banco de dados utilizando PostgreSQL foram de suma importância para o aprendizado do mesmo.

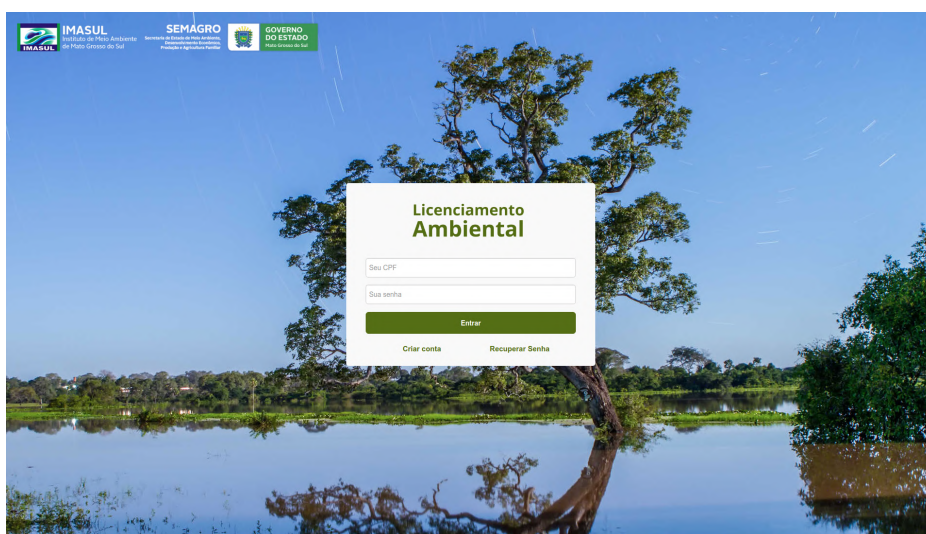
No segundo mês, com auxílio do *Scrum Master*, o discente e seu *squad* começaram a colocar em prática tudo o que aprenderam durante o primeiro mês. A equipe desenvolveu um sistema do início ao fim para fins de treinamento e aprendizado, onde o discente participou ativamente das atividades entregues ao time. O sistema teve como ideia inicial contabilizar horas trabalhadas pelos funcionários da agência e foi construído a princípio com apenas duas telas onde o usuário é capaz de clicar em um botão para começar o seu turno e outro botão para finalizar seu turno. Ao final o sistema armazena a quantidade de horas trabalhadas e apresenta os dados armazenados em uma tabela para o usuário final. As tecnologias utilizadas no projeto foram, Java como a linguagem para o *backend* utilizando o *framework Spring Boot* e a linguagem JavaScript no *frontend* utilizando o *framework* Vuejs, e o banco de dados utilizado para armazenar os dados foi o Postgres.

No término destes dois meses o discente foi alocado ao *squad* 1 da tribo YBY, responsáveis pelo projeto de licenciamento ambiental do Mato Grosso do Sul. O *squad* era formado por um *tester*, um *Product Owner* e três desenvolvedores, sendo um deles o *Scrum Master* da equipe.

4.2 Licenciamento Ambiental do Mato Grosso do sul

O sistema de licenciamento ambiental é um módulo público (Figura 4.1) que permite que o usuário cadastre-se como empreendedor e cadastre seus empreendimentos. Após o seu cadastro o empreendedor tem a opção de cadastrar solicitações de licenças ambientais para as atividades que seus empreendimentos executam. Ao fim desse processo é gerado uma guia de pagamento com os valores das taxas para ser feita a análise da solicitação de licenciamento, que são calculadas a partir de várias variáveis, como o porte do empreendimento, a distância do empreendimentos para Campo Grande e o grau de poluição que a atividade exercida irá gerar. O sistema foi implementado utilizando o *framework* AngularJS versão 1.6 no *frontend* e o *framework* Play no *backend* usando a versão 1.5. Os dados são mantidos e manuseados no banco Postgres na versão 9.5.

Figura 4.1 – Licenciamento Ambiental



Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Zetta (2021)

4.2.1 Remoção de empreendimentos do empreendedor

A primeira atividade com um pouco mais de complexidade que foi designada ao discente foi a atividade de exclusão do empreendedor.

Ao iniciar no projeto, o módulo de licenciamento ambiental estava passando por uma reestruturação que consistia na refatoração de mapeamento entre as principais entidades do sistema, sendo elas, empreendedor e empreendimento. Nessa refatoração foram alteradas a relação que antes eram mantidas um pra um, para permitir a relação um para N. Dessa forma, antes das mudanças um empreendedor podia cadastrar apenas um empreendimento no sistema

e após as mudanças um empreendedor consegue ter vários empreendimentos, com isso muitos métodos ficaram obsoletos. A exclusão do empreendedor foi uma delas, pois o método não conseguia tratar quando o empreendedor tem mais de um empreendimento.

Nesta atividade o discente precisou entender todo o fluxo do sistema, desde o *frontend* até o *backend*.

O discente optou por utilizar *Promise* (Figura 4.2) para auxiliar no desenvolvimento da atividade. Uma *Promise* é utilizada para obter uma operação assíncrona, ou seja, ela é um objeto que representa a conclusão da operação, uma *Promise* possui dois parâmetros, ambos chamados de *call back functions*, uma representando sucesso e outra o fracasso da *Promise*. Para auxiliar nesse processo foi utilizado o método *then*, método da própria biblioteca da *Promise*, o seu uso permite o encadeamento de outros métodos durante a sua execução, como apresentado em sua documentação Mozilla (2022).

Como a alteração no mapeamento permite um empreendedor poder ter vários empreendimentos, a requisição de busca dos empreendimentos pode ser uma requisição demorada por causa da quantidade de empreendimento que o empreendedor pode ter, por isso a importância de criar a operação assíncrona.

O discente preocupou-se em manter a regra de negócio do sistema, onde um empreendedor que contém um empreendimento com uma solicitação de licenciamento criada, não é possível excluir o mesmo. Caso o empreendedor tenha solicitações já criadas é apresentada na tela uma mensagem informando que não é possível finalizar a ação (Figura 4.3), caso contrário é apresentado na tela uma *modal* de confirmação de exclusão.

A *service* de deletar o empreendedor foi inserida dentro do método *then*, o método *then* retorna uma *Promise* de sucesso ou de fracasso. Então caso ocorra erro na exclusão de algum empreendimento, neste caso é lançado uma exceção e o empreendedor não será alterado.

Figura 4.2 – Função do *frontend* de remover empreendimento

```
function removerEmpreendedor(cpfcnpj, idEmpreendedor) {
  empreendedorService.buscarEmpreendimento(cpfcnpj).then(function(response){
    var empreendimentos = response.data;
    var podeSerExcluido = true;
    var listaIds = [];

    empreendimentos.forEach(empreendimento => {
      listaIds = listaIds.concat(empreendimento.id);

      if(empreendimento.possuiCaracterizacoes == true){
        podeSerExcluido = false;
      }
    });
    if (podeSerExcluido) {
      var configModal = {
        titulo: 'Confirmar exclusão do empreendedor',
        conteudo: 'Tem certeza que deseja excluir esse empreendedor? Ao selecionar a opção "Sim", todos os empreendimentos vinculados também serão excluídos.'
      };
      var instanciaModal = modalSimpleService.abrirModal(configModal);

      instanciaModal.result
        .then(function(){
          listagem.confirmarRemoverEmpreendedor(listaIds, idEmpreendedor);
        })
        .catch(function(){
          mensagem.error("Não foi possível excluir o empreendedor pois o mesmo possui empreendimentos com solicitações vinculadas");
        });
    }
  });
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Figura 4.3 – Função do *frontend* de confirmar remoção empreendimento

```
function confirmarRemoverEmpreendedor(listaIds, idEmpreendedor) {
  if(listaIds.length > 0) {
    empreendimentoService.deleteEmpreendimentos(listaIds)
      .then(function(response){
        empreendedorService.delete(idEmpreendedor)
          .then(function(response){
            listagem.pesquisar(null, response.data.texto);
          })
          .catch(function(response){
            mensagem.error(response.data.texto);
          });
        listagem.pesquisar(null, response.data.texto);
      })
      .catch(function(response){
        mensagem.error(response.data.texto);
      });
  } else {
    empreendedorService.delete(idEmpreendedor)
      .then(function(response){
        listagem.pesquisar(null, response.data.texto);
      })
      .catch(function(response){
        mensagem.error(response.data.texto);
      });
  }
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

No *backend* é feito uma busca para cada *id* pertencente a lista passada por parâmetro. Para cada empreendimento buscado é chamado o método de exclusão e caso dê erro em alguma exclusão do empreendimento, é lançado uma exceção (Figura 4.4).

Figura 4.4 – Método de exclusão de empreendimentos

```

public static void deleteEmpreendimentos(RemoveEmpreendimentoV0 removeV0){

    verificarPermissao(Acao.EXCLUIR_EMPREENDIMENTO);

    try {
        if(removeV0.ids != null) {
            removeV0.ids.forEach(id -> {

                Empreendimento empreendimento = Empreendimento.findById(id);
                empreendimento.delete();
            });
        }else{

            throw new ValidationException("Ocorreu um erro, não existem empreendimentos para serem excluídos");
        }
    }catch (Exception e){

        throw new ValidacaoException(Mensagem.EMPREENDEDOR_POSSUI_EMPREENDIMENTO_COM_CARACTERIZACAO);
    }

    renderMensagem(Mensagem.EMPREENDIMENTO_EXCLUIDO_SUCESSO);
}

```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Na Figura 4.5 é apresentado a tela inicial do sistema, ao selecionar o botão de ações é apresentado a modal de confirmação de exclusão (Figura 4.6), e ao final é retornado a mensagem de sucesso para o usuário (Figura 4.7).

Figura 4.5 – Clicando no botão de ações do empreendedor

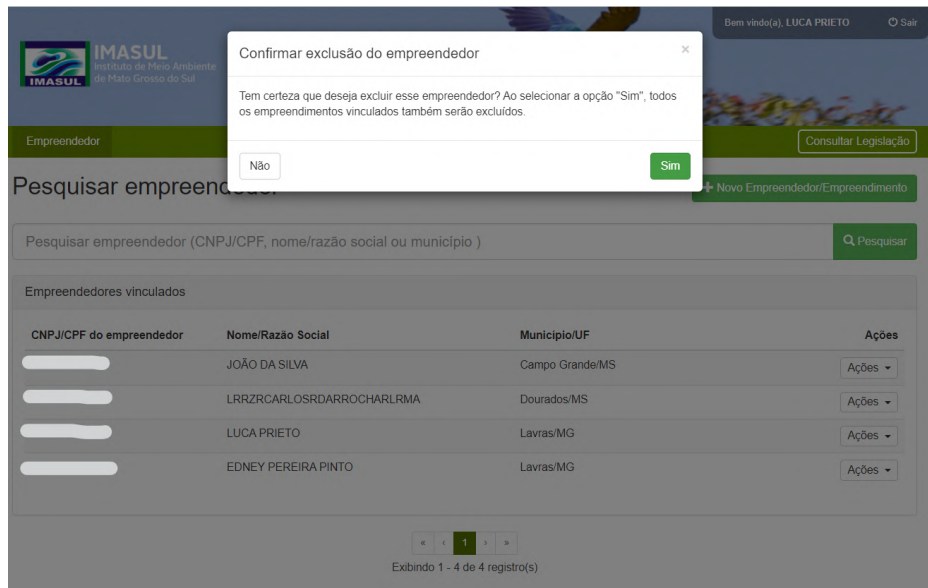
The screenshot displays the user interface of the IMASUL system. At the top, there are logos for IMASUL, SEMAGRO, and GOVERNO DO ESTADO Mato Grosso do Sul. The user is logged in as 'LUCA PRIETO'. Below the header, there is a search bar for entrepreneurs and a table of linked entrepreneurs. The table has columns for CNPJ/CPF, Nome/Razão Social, Município/UF, and Ações. A dropdown menu is open for the 'Ações' column of the first row, showing options: 'Listar empreendimentos', 'Visualizar empreendedor', and 'Excluir empreendedor'.

CNPJ/CPF do empreendedor	Nome/Razão Social	Município/UF	Ações
[Redacted]	JOÃO DA SILVA	Campo Grande/MS	Ações ▾
[Redacted]	LRRZRCARLOSRDARROCHARLRMA	Dourados/MS	[Redacted]
[Redacted]	LUCA PRIETO	Lavras/MG	[Redacted]
[Redacted]	EDNEY PEREIRA PINTO	Lavras/MG	Ações ▾

Exibindo 1 - 4 de 4 registro(s)

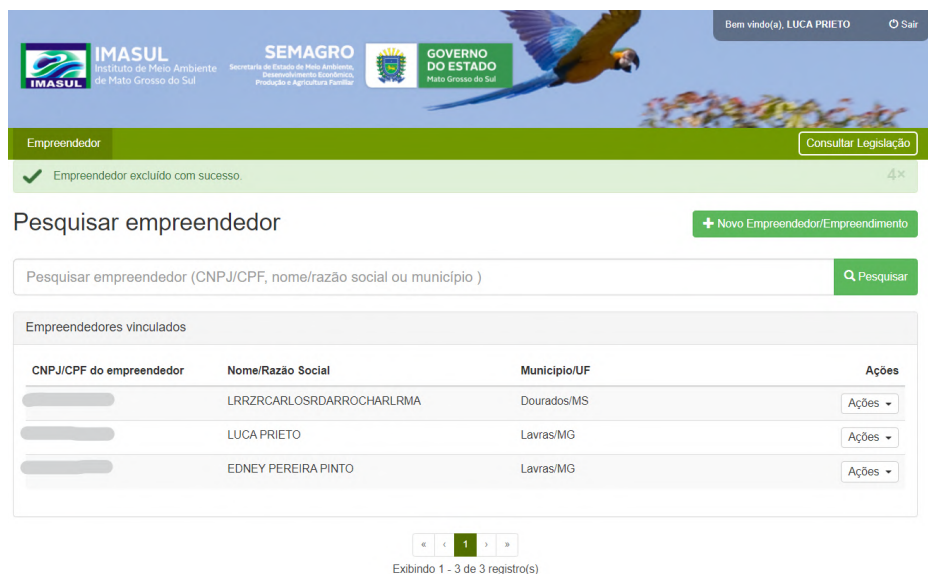
Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Figura 4.6 – Selecionando exclusão do empreendedor



Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Figura 4.7 – Empreendedor/empreendimento deletado



Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

A principal dificuldade encontrada nesta atividade foi entender a regra de negócio, pois o discente não estava habituado com o sistema e suas tecnologias. Durante o desenvolvimento do projeto o discente teve acompanhamento da equipe, contudo, esta atividade foi suma de importância para a adaptação ao sistema, por ser a primeira atividade em contato com ambas partes do projeto, *frontend* e *backend*.

4.2.2 Implementação do cálculo de distância dos municípios

Uma das principais funcionalidades do sistema é a parte de cadastro da geometria da atividade do empreendimento, onde o empreendedor consegue desenhar e montar a geometria de suas atividades.

O cálculo da distância dos municípios foi uma demanda que apresentava a necessidade de cálculo de distâncias entre as cidades onde eram criadas as atividades no sistema e a cidade de Campo Grande.

Foi necessária a implementação de um método capaz de fazer o cálculo da distância da atividade para a central de Campo Grande no Mato Grosso do Sul. Para que fosse possível a realização do cálculo de custo de viagens dos técnicos responsáveis pela validação dos parâmetros e atividades levantadas pelo empreendedor no sistema de licenciamento ambiental. Não foi utilizado APIs ou Interface de Programação de Aplicações para a solução deste problema porque o projeto do Licenciamento Ambiental possui a sua própria biblioteca de geolocalização desenvolvido pela agência.

A dificuldade encontrada pelo discente foi saber em qual cidade estava a geometria da atividade, pois a regra de negócio do sistema permite que algumas atividades sejam plotadas fora da área do empreendimento, fazendo com que a atividade possa ser inserida em qualquer cidade do estado do Mato Grosso do sul. sabendo disso, foi necessário criar um método onde fosse capaz de fazer a validação e verificação das geometrias das atividades.

O discente fez o uso da função *centroid* da classe *Point* (Figura 4.8) . Este método recebe uma geometria por parâmetro e é capaz retornar a coordenada (x, y) que representa o ponto central da geometria. Utilizando esse ponto central encontrado para fazer o reconhecimento de onde a atividade está estabelecida, para fazer o cálculo de distância deste ponto para Campo grande no Mato Grosso do Sul.

Figura 4.8 – Método para encontrar o município

```
public static Municipio getMunicipioAtividadeForaEmpreendimento(Geometry geo) {  
  
    Point centroide = geo.getCentroid();  
    Municipio municipio = Municipio.findByCoordenada(centroide.getX(),centroide.getY());  
  
    return municipio;  
  
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Utilizando esse *centroid* da geometria calculado anteriormente, e a classe *GeometryFactory* que fornece métodos para a criação de coordenadas, o discente implementou a verificação (Figura 4.9) utilizando o método que calcula o limite da geometria do município para encontrar onde a atividade está plotada.

Figura 4.9 – Método para encontrar o município a partir do *centroid*

```
public static Municipio findByCoordenada(Double latitude, Double longitude) {
    List<Municipio> municipios = Municipio.findByEstado(uf: "MS");
    GeometryFactory gf = new GeometryFactory();
    for (Municipio municipio : municipios) {
        Geometry coordenada = gf.createPoint(new Coordinate(latitude, longitude));
        if(municipio.getLimite().contains(coordenada)) {
            return municipio;
        }
    }
    return null;
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Para o cálculo da taxa o discente atentou se que o cálculo para atividade que estão em área urbana e rural são diferentes. Em ambos era necessário considerar a ida e a volta dos técnicos, porém o cálculo para atividades rurais teriam um acréscimo de vinte por cento na distância, como mostrado na Figura 4.10.

Figura 4.10 – Método para calcular distância da capital

```
public static Double calcularDistanciaCapital(AtividadeCaracterizacao atividadeCaracterizacao, Caracterizacao caracterizacao){
    Double distancia = null;
    if(atividadeCaracterizacao.atividade.dentroEmpreendimento || atividadeCaracterizacao.geometriasAtividade.isEmpty()){
        Empreendimento e = Empreendimento.findById(caracterizacao.empreendimento.id);
        distancia = DistanciaCapital.findDistanciaByCodigoIbge(e.municipio.id).distancia * 2;
    }else{
        Geometry geo = atividadeCaracterizacao.geometriasAtividade.get(0).geometria;
        Municipio municipio = AtividadeCaracterizacao.getMunicipioAtividadeForaEmpreendimento(geo);
        distancia = DistanciaCapital.findDistanciaByCodigoIbge(municipio.id).distancia * 2;
    }
    if(caracterizacao.empreendimento.localizacao == TipoLocalizacao.ZONA_RURAL){
        distancia = distancia + (distancia * 0.2);
    }
    return distancia;
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

4.2.3 Integração com API de pagamentos

Uma das regras do módulo de licenciamento ambiental são as taxas de licenciamento, que são obrigatórias para o empreendedor para dar continuidade no fluxo da criação da licença. Essas taxas são disponibilizadas aos usuários como boletos.

Para a criação dos boletos foi disponibilizada uma *API web*, responsável por criar e manter os boletos gerados para fins do licenciamento ambiental.

A implementação foi feita a partir de uma documentação disponibilizada pelos responsáveis. Onde o documento apresenta as rotas e os parâmetros necessários para fazer a conexão com a *api* remota, utilizando uma conexão segura. Sendo necessário a utilização de um *bearer token*, este *token* caracteriza uma autorização emitida pelo lado do *client*. O *client* deve possuir credenciais únicas necessárias para identificação de quem está fazendo a requisição para ser efetuado a utilização da *api* de pagamentos.

Na Figura 4.11 é apresentado uma das conexões estabelecidas pelo discente para gerar um boleto, onde o discente utiliza a interface disponibilizada pelo *framework* Play *WSRequest*, que é a principal interface de criação de *WS request*.

Figura 4.11 – Método de gerar boleto

```
public static Boletov0 gerarBoleto(GerarBoletoDTO boletoDTO) throws FileNotFoundException {
    String json = gson.toJson(boletoDTO);

    WS.WSRequest request = WS.url(GERAR_BOLETO);

    request.setHeader( name: "Content-Type", value: "application/json");
    request.setHeader( name: "Authorization", value: "Bearer "+ SiriemaWS.getToken());
    request.body(json);

    WS.HttpResponse response = request.post();

    if (!response.success())
        throw new WebServiceException(response);

    Type type = new TypeToken<Boletov0>().getType();

    return gsonBuilder.create().fromJson(response.getJson(), type);
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

A Figura 4.12 está presente o método desenvolvido para a geração do *bearer token*, onde é necessário ter em mãos, *client id* e *client secret*, credenciais únicas de identificação.

Figura 4.12 – Método gerador de token

```
private static SiriemaResponseTokenVO gerarTokenAcesso(){
    Map<String, String> paramsAuth = new HashMap<>();

    paramsAuth.put("grant_type", SIRIEMA_GRANT_TYPE);
    paramsAuth.put("client_id", SIRIEMA_CLIENT_ID);
    paramsAuth.put("client_secret", SIRIEMA_CLIENT_SECRET);

    WS.WSRequest request = WS.url(SIRIEMA_TOKEN_AUTENTICACAO);

    request.setHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
    request.setHeader( name: "Accept", value: "application/json");
    request.setParameters(paramsAuth);

    WS.HttpResponse responseAuth = request.post();

    if (!responseAuth.success())
        throw new WebServiceException(responseAuth);

    Type type = new TypeToken<SiriemaResponseTokenVO>().getType();

    return gsonBuilder.create().fromJson(responseAuth.getJson(), type);
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Para vincular uma guia a uma solicitação de licença o discente criou uma coluna no banco de dados na tabela caracterização. Essa tabela representa a solicitação de licença e a coluna criada com nome de “codigoGuia” armazena o código da guia gerada e retornada pela *api*, fazendo assim um vínculo do boleto com a solicitação como mostrado na Figura 4.13.

Figura 4.13 – Método de download da guia

```

public void downloadGuia(Long id) throws FileNotFoundException {

    verificarPermissao(Acao.CADASTRAR_CARACTERIZACAO);

    Caracterizacao caracterizacao = Caracterizacao.findById(id);

    if(caracterizacao.valorTotalTaxaLicenciamento == 0.00) {
        throw new ApplicationException(Mensagem.CARACTERIZACAO_TAXALICENCIAMENTO_ZERO);
    }else {
        caracterizacao.empreendimento.validarSeUsuarioCadastrante();

        GerarBoletoDTO boleto = convertToBoleto(caracterizacao);

        BoletoVO boletoVO = PagamentosWS.gerarBoleto(boleto);

        try {
            caracterizacao.status = StatusCaracterizacao.findById(StatusCaracterizacao.AGUARDANDO_PAGAMENTO_DA_GUIA);
            caracterizacao.codigoGuia = boletoVO.codigoBoleto;
            caracterizacao.save();

            renderBinaryFile(byteArrayToFile(boletoVO.dados, boletoVO.codigoBoleto));
        } catch (Exception e) {

            Logger.error(e.getMessage());

            renderMensagem(Mensagem.CARACTERIZACAO_ERRO_EMISSAO_DAE);
        }
    }
}

```

Fonte: Análise do Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Com a integração concluída foi necessário criar a *service* no sistema para fazer a comunicação com o *frontend*, que é mostrada na Figura 4.14.

Figura 4.14 – *Service* de download da guia

```

this.downloadDaeLicenciamentoCaracterizacao = function(idCaracterizacao) {
    window.location.href = config.BASE_URL() + "caracterizacoes/" + idCaracterizacao + "/daeLicenciamento/arquivo";
};

```

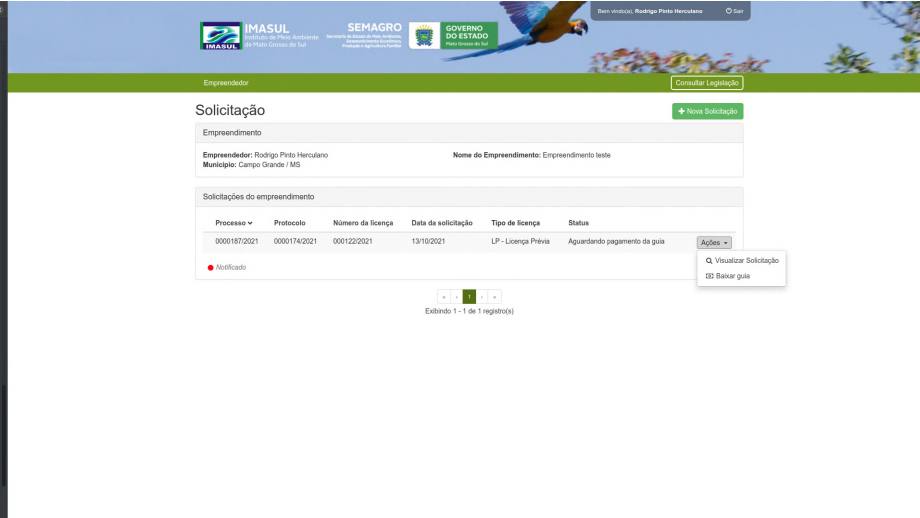
Fonte: Análise do Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Na Figura 4.15 encontra-se a tela do sistema onde a listagem das solicitações são apresentadas. Ao finalizar o cadastro da solicitação da licença a solicitação irá apresentar o status aguardando emissão da guia.

Para criar a funcionalidade de emitir a licença, o discente preocupou-se em mostrar o botão de emitir licença apenas no *status* final da solicitação, evitando que o solicitante emita antes da finalização do cadastro da solicitação. Para isso foi utilizado o *ng-show*, funcionalidade do *framework* Angular que permite a manipulação do que realmente deve aparecer na tela de acordo com o status da caracterização.

Ao concluir a solicitação o botão de emissão aparece ao empreendedor para que ele seja capaz de baixar a guia como na Figura 4.16.

Figura 4.15 – Baixar guia



The screenshot displays the IMASUL (Instituto de Meio Ambiente e Recursos Hídricos do Estado do Mato Grosso do Sul) web interface. The header includes logos for IMASUL, SEMAGRO, and the Government of Mato Grosso do Sul, along with a user profile for Rodrigo Pinto Herculano. The main content area is titled 'Solicitação' and contains a form for 'Empreendimento' with fields for 'Empreendedor' (Rodrigo Pinto Herculano) and 'Município' (Campo Grande / MS). Below the form is a table of 'Solicitações do empreendimento' with columns for 'Processo', 'Protocolo', 'Número da licença', 'Data da solicitação', 'Tipo de licença', and 'Status'. A single entry is shown with a status of 'Notificado'. A dropdown menu for 'Ações' is open, showing options for 'Visualizar Solicitação' and 'Baixar guia'.

Processo	Protocolo	Número da licença	Data da solicitação	Tipo de licença	Status
0000187/2021	0000174/2021	000122/2021	13/10/2021	LP - Licença Prévia	Aguardando pagamento da guia

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Zetta (2021)

Figura 4.16 – Boleto gerado

GOVERNO DE ESTADO DE MATO GROSSO DO SUL		SECRETARIA DE ESTADO DE MEIO AMBIENTE E DESENVOLVIMENTO ECONÔMICO, PRODUÇÃO E AGRICULTURA FAMILIAR – SEMAGRO		INSTITUTO DE MEIO AMBIENTE DE MATO GROSSO DO SUL – IMASUL	
Rua Desembargador Leão Neto do Carmo, Bl.06, P. dos Poderes, CEP:79.031-902, Campo Grande/MS, Tel.:(67) 3318-5600					
GUIA DE RECOLHIMENTO				Nº 695039	
Contribuinte			CNPJ/CPF		
Endereço			Cidade		UF
RUA RUA, 85, AVENIDA AVENIDA			LAVRAS		MG
Especificação da Receita		Empreendimento			
LICENÇA DE OPERAÇÃO		Squad1			
Nr. Processo	Data de Emissão	Data do Vencimento	Valor A Recolher		
0000063/2021	24/02/2021	26/03/2021	R\$ 7.206,82		
Informações Complementares					
Guia referente à solicitação de Licenciamento Ambiental. Após o pagamento a solicitação tramitará para a análise do setor responsável.					
Este documento só terá validade autenticado pelo Banco do Brasil. Via Contribuinte					
GOVERNO DE ESTADO DE MATO GROSSO DO SUL		SECRETARIA DE ESTADO DE MEIO AMBIENTE E DESENVOLVIMENTO ECONÔMICO, PRODUÇÃO E AGRICULTURA FAMILIAR – SEMAGRO		INSTITUTO DE MEIO AMBIENTE DE MATO GROSSO DO SUL – IMASUL	
Rua Desembargador Leão Neto do Carmo, Bl.06, P. dos Poderes, CEP:79.031-902, Campo Grande/MS, Tel.:(67) 3318-5600					
GUIA DE RECOLHIMENTO				Nº 695039	
Contribuinte			CNPJ/CPF		
Cidade	UF	Nr. Processo	Data Emissão	Data Vencimento	Valor A Recolher
LAVRAS	MG	0000063/2021	24/02/2021	26/03/2021	R\$ 7.206,82
Especificação da Receita		Empreendimento			
LICENÇA DE OPERAÇÃO		Squad1			
8566000072 8 06820169202 4 1032600000 7 0000695039 8					
					
GOVERNO DE ESTADO DE MATO GROSSO DO SUL		SECRETARIA DE ESTADO DE MEIO AMBIENTE E DESENVOLVIMENTO ECONÔMICO, PRODUÇÃO E AGRICULTURA FAMILIAR – SEMAGRO		INSTITUTO DE MEIO AMBIENTE DE MATO GROSSO DO SUL – IMASUL	
Rua Desembargador Leão Neto do Carmo, Bl.06, P. dos Poderes, CEP:79.031-902, Campo Grande/MS, Tel.:(67) 3318-5600					
GUIA DE RECOLHIMENTO				Nº 695039	
Contribuinte:			CNPJ/CPF		
Endereço			Cidade		UF
RUA RUA, 85, AVENIDA AVENIDA			LAVRAS		MG
Especificação da Receita		Empreendimento			
LICENÇA DE OPERAÇÃO		Squad1			
Nr. Processo	Data de Emissão	Data do Vencimento	Valor A Recolher		
0000063/2021	24/02/2021	26/03/2021	R\$ 7.206,82		
Informações Complementares					
Guia referente à solicitação de Licenciamento Ambiental. Após o pagamento a solicitação tramitará para a análise do setor responsável.					
Este documento só terá validade autenticado pelo Banco do Brasil. Via Processo					

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Zetta (2021)

A fim de automatizar todo o processo, o discente realizou a desenvolvimento de um *job* de verificação, criado para fins de verificação de pagamentos.

O *job* faz a consulta no banco de dados e seleciona todas as solicitações de licença que estão no status aguardando pagamento da taxa, ou seja, toda caracterização que foi finalizada e o empreendedor emitiu a guia de pagamento. Ao encontrar todas as solicitações o *job* faz a verificação um a um para cada solicitação utilizando o “codigoGuia” armazenado no primeiro passo. Para fazer o *job* executar repetidas vezes o discente fez a utilização de cron, uma ex-

pressão lógica que permite que o *job* seja manipulado de tempo em tempo, assim fazendo a verificação de todas as solicitações de licenças emitidas, fazendo a verificação de pagamentos efetuados, para que o sistema permita fazer a tramitação do processo para o módulo de análise. Módulo onde os técnicos responsáveis irão fazer as verificações necessárias para a validação da licença ambiental.

Na Figura 4.17 é apresentado o *job* de verificação de pagamentos:

Figura 4.17 – Job de verificação de pagamentos

```
public void processarComApiPagamentos(){
    List<Caracterizacao> caracterizacoes = Caracterizacao.getCaracterizacaoComGuiaEmitida();
    Logger.info( message: "ProcessamentoGuia:: Quantidade de Guias a serem processadas: " + caracterizacoes.size());
    caracterizacoes.forEach(caracterizacao -> {
        situacao = PagamentosWS.situacaoBoleto(caracterizacao.codigoGuia).situacaoBoleto;
        if(!situacao.equals("EMITIDO")) {
            if(situacao.equals("PAGO")) {
                caracterizacao.status = StatusCaracterizacao.findById(StatusCaracterizacao.EM_ANALISE);
                Logger.info( message: "ProcessamentoGuia:: Guia : " + caracterizacao.codigoGuia + " mudou seu status para PAGO");
            } else {
                caracterizacao.status = StatusCaracterizacao.findById(StatusCaracterizacao.GUIA_VENCIDA);
                Logger.info( message: "ProcessamentoGuia:: Guia : " + caracterizacao.codigoGuia + " mudou seu status para VENCIDA");
            }
            caracterizacao.save();
        }
    });
}
```

Fonte: Licenciamento Ambiental do Mato Grosso do Sul - Agência Zetta (2021)

Alguns detalhes da integração impactaram direto no sistema, como a parâmetro necessário nome do empreendimento para a geração do boleto, onde a *api* tratava apenas nomes com um tamanho bem reduzido (80 caracteres). Mais tarde foi adaptado no sistema licenciamento ambiental para que não houvesse divergência nos dados nas nomenclaturas do empreendimentos e nos boletos gerados pela api.

5 CONSIDERAÇÕES FINAIS

Durante o estágio supervisionado o discente atuou ativamente na realização de diversas atividades e correções de *bugs* que contribuíram para o seu crescimento. Como o projeto de licenciamento já estava em estágio avançado, foi necessário que o discente entendesse completamente o sistema e suas regras de negócio antes de começar o desenvolvimento. A evolução do mesmo foi gradativa, onde começou com pequenas correções até começar a desenvolver atividades complexas que demandam experiência para a sua solução.

Indubitavelmente o estágio é uma valiosa etapa, onde o discente adquire experiência nas quais somente nesse ambiente o mesmo conseguiria adquirir esses resultados, onde o discente conseguiu aplicar as teorias vistas anteriormente em sala de aula. Os conceitos aprendidos durante a graduação foram de suma importância para o desenvolvimento do discente no estágio. O conteúdo apresentado na sala de aula serviram de base teórica para o desenvolvimento das atividades, fazendo com que o discente apresentasse soluções para os problemas encontrados durante o decorrer do estágio.

Considerando que o discente adquiriu não somente o conhecimento técnico mas também habilidades de trabalho em equipe, organização, entrega de prazos, que são de suma importância para o profissional, pois entender e saber se comunicar com os colegas de trabalho também é uma parte crucial para o seu crescimento.

Foram observados alguns pontos que podem melhorar os processos internos de desenvolvimento da agência, sendo eles:

- A falta da atualização dos documentos referentes ao produto que estão em desenvolvimento. Em alguns casos os integrantes do time de desenvolvimentos levantaram algumas dúvidas, porém a documentação de regras do sistema estava desatualizada, e o profissional que tinha o conhecimento da regra de negócio já não estava mais presente na agência, deixando dúvidas sobre o que é realmente deveria ser feito. Logo, manter uma documentação atualizada impede de gerar transtornos futuros.
- Troca constante de membros do *squad* por falta de pessoas disponíveis, que impacta diretamente a equipe, pois ao sair um membro experiente para ser alocado a outro projeto, eram alocados novos membros sem experiência impactando na entrega da sprint.

Em conclusão, disciplinas como Introdução aos Algoritmos, Estruturas de Dados, Introdução a Sistemas de Banco de Dados, Práticas de Programação Orientada a Objetos, Enge-

nharia de Software, Programação WEB, vistas durante a graduação foram de suma importância para a realização do estágio, proporcionando uma base de conhecimento para que o mesmo conseguisse desempenhar as atividades. O estágio supervisionado proporcionou benefícios extremamente significativos ao discente, pois o mesmo obteve um crescimento técnico e pessoal, desenvolvendo habilidades em desenvolvimento de sistemas *web* em ambos lados, *backend* e *frontend*, resolução de problemas complexos e comunicação.

REFERÊNCIAS BIBLIOGRÁFICAS

ANGULARJS. *AngularJS*. 2021. Disponível em: <<https://docs.angularjs.org/guide/introduction>>. Acesso em: 30/08/21.

DEITEL, P.; DEITEL, H. *Java - Como Programar*. 10. ed. Londres: Pearson, 2015. 13–13 p. ISBN 9788543004792.

GIT. *Git*. 2021. Disponível em: <<https://git-scm.com/>>. Acesso em: 07/09/21.

IMASUL. *Licenciamento Ambiental do Mato Grosso do Sul*. 2022. Disponível em: <<http://licenciamento.imasul.ms.gov.br/licenciamento-simplificado/>>. Acesso em: 19/04/22.

MOZILLA. *CSS*. 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/CSS/First_steps>. Acesso em: 21/08/21.

MOZILLA. *HTML*. 2021. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/HTML>>. Acesso em: 20/08/21.

MOZILLA. *Javascript*. 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/About_JavaScript>. Acesso em: 24/08/21.

MOZILLA. *Then*. 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise/then>. Acesso em: 05/04/22.

PLAYFRAMEWORK. *PlayFramework*. 2021. Disponível em: <<https://www.playframework.com/>>. Acesso em: 11/09/21.

PUGJS. *Pug*. 2022. Disponível em: <<https://pugjs.org/api/getting-started.html>>. Acesso em: 05/04/22.

SCRUM.ORG. *Scrum*. 2021. Disponível em: <<https://www.scrum.org/resources/what-is-scrum>>. Acesso em: 06/09/21.

SPRING.IO. *Spring is productive*. 2021. Disponível em: <<https://spring.io/why-spring>>. Acesso em: 16/09/21.

VUEJS. *Vuejs*. 2021. Disponível em: <<https://br.vuejs.org/v2/guide/index.html>>. Acesso em: 14/09/21.

ZETTA, A. *Agência Zetta/UFLA*. 2020. Disponível em: <<https://agenciazetta.ufla.br/>>. Acesso em: 12/12/21.