LUCAS DE CARVALHO FELIZARDO

ESTÁGIO SUPERVISIONADO NA VEGA MONITORAMENTO

LAVRAS - MG
2022

LUCAS DE CARVALHO FELIZARDO

ESTÁGIO SUPERVISIONADO NA VEGA MONITORAMENTO

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

Prof. Dr. Janderson Rodrigo De Oliveira
Orientador

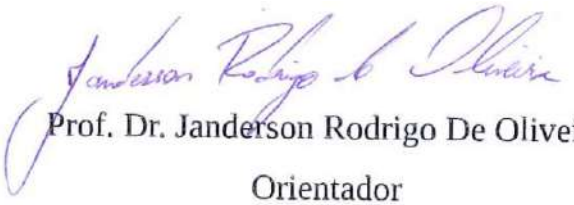
LAVRAS - MG
2022

LUCAS DE CARVALHO FELIZARDO

ESTÁGIO SUPERVISIONADO NA VEGA MONITORAMENTO

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

APROVADO em 16 de dezembro de 2022.
Janderson Rodrigo de Oliveira
Paula Christina Figueiredo Cardoso
Paulo Afonso Parreira Junior



Prof. Dr. Janderson Rodrigo De Oliveira
Orientador

LAVRAS - MG
2022

A todos os amigos conquistados ao longo dos anos de Universidade, que se tornaram grandes parceiros de vida.
A todos os profissionais da UFLA, em especial o corpo docente do Departamento de Ciência da Computação, que me auxiliaram na minha formação.

Dedico

AGRADECIMENTOS

A Universidade Federal de Lavras, especialmente ao Departamento de Ciência da Computação, pela oportunidade.

A Vega Monitoramento pela concessão do estágio.

Ao professor Dr. Janderson Rodrigo De Oliveira, pela orientação e disposição para ajudar.

A todos os funcionários do DCC/UFLA

A todos os colegas e amigos do departamento.

MUITO OBRIGADO!

“Raramente estou mais feliz do que quando passo um dia inteiro a programar o meu computador para fazer automaticamente uma tarefa que de outra forma demoraria uns bons dez segundos a fazer à mão.” (Douglas Adams)

RESUMO

Este documento apresenta as atividades desenvolvidas durante o período de estágio supervisionado na empresa Vega Monitoramento. O sistema de software que recebeu novas funcionalidades foi a plataforma de monitoramento Lyra, responsável por realizar o monitoramento agro, climático e socioambiental de territórios em todo o Brasil. Neste relatório de estágio, são apresentadas as principais tecnologias empregadas na construção da plataforma, juntamente com o desenvolvimento da funcionalidade de diagnóstico socioambiental. Ao final, como conclusão, é enfatizado a importância do período de estágio para aprimoração profissional, permitindo praticar as habilidades e conhecimentos adquiridos em sala de aula.

Palavras-chave: Desenvolvimento de software e Desenvolvimento Web.

ABSTRACT

This document presents the activities developed during the supervised training period at the company Vega Monitoramento. The software system that received new features was the Lyra monitoring platform, responsible for carrying out agro, climate and socio-environmental monitoring of territories throughout Brazil. In this training report, the main technologies used in the construction of the platform are presented, along with the development of the socio-environmental diagnosis functionality. At the end, as a conclusion, the importance of the training period for professional improvement is emphasized, allowing us to practice the skills and knowledge acquired in the classroom.

Keywords: Software development and Web development.

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é apresentada uma descrição sucinta sobre como está o mercado de tecnologia da informação, além de falar um pouco sobre a Vega Monitoramento e alguns problemas enfrentados durante o período de estágio supervisionado.

1.1 Contextualização

Segundo um estudo realizado trimestralmente pela ADVANCE Consulting[1], empresa focada em consultoria na área de TI, em 2021 o mercado de TI apresentou um recorde de crescimento de 23% em relação ao ano anterior (Tabela 1). A principal causa desse crescimento, apontada pelo estudo, foram os investimentos realizados em 2020 para que colaboradores trabalhassem em *home-office* e, como consequência, tendo um aumento de produtividade em 2021.

Tabela 1 — Crescimento trimestral, em relação ao período anterior, das empresas de TI.

Trimestre	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**
Primeiro	10%	8%	10%	2%	8%	16%
Segundo	3%	6%	8%	8%	5%	27%
Terceiro	5%	9%	11%	10%	10%	21%
Quarto	1%	15%	12%	14%	13%	28%
Total ano	**5%**	**11%**	**11%**	**10%**	**10%**	**23%**

Fonte: ADVANCE Consulting.

O mesmo estudo aponta um crescimento de 22% para o ano de 2022, porém um ponto de preocupação levantado pelas empresas é a falta de trabalhadores qualificados, inflacionando os salários dos profissionais de TI. Devido a este receio, as empresas vêm criando manobras como, por exemplo, capacitar novos profissionais através de estágios.

---

[1] ADVANCE Consulting | Dados do mercado de TI. advanceconsulting. Disponível em: <https://www.advanceconsulting.com.br/pesquisa>. Acesso em: 25 jun. 2022.

Neste contexto, o presente relatório de estágio deixa em exposição as atividades realizadas durante o estágio supervisionado na empresa Vega Monitoramento, no período de agosto/2021 à dezembro/2021. O trabalho ainda ressalta a importância da experiência prática, junto ao conhecimento teórico adquirido na UFLA durante o período de formação do autor.

1.2 Vega Monitoramento e plataforma Lyra

O estágio supervisionado relatado neste trabalho foi oferecido pela empresa Vega Monitoramento[2], uma startup que busca soluções inovadoras no agronegócio. A empresa possui matriz em São José dos Campos/SP e possui um escritório em Lavras/MG. Todo o período de estágio foi realizado remotamente.

A Vega Monitoramento foi fundada em 2018, por um ex-aluno da UFLA, com o propósito de monitorar a produção agrícola, atestando a sustentabilidade das práticas na cadeia produtiva do agronegócio, baseando-se em três pilares: monitoramento, tecnologia e sustentabilidade. Pertencendo ao Grupo Imagem, que possui mais de 32 anos e é líder no segmento de TI em geotecnologia, a empresa tem crescido rapidamente contando com 46 colaboradores distribuídos nos projetos Originar, Lyra e Vega-API atendendo grandes clientes como: Bayer, Governo do Estado de São Paulo, Santander, Sicredi, dentre outros.

Ao decorrer do estágio, todos os trabalhos realizados foram sobre a plataforma Lyra, sendo esta um sistema de software que, por sensoriamento remoto e inteligência de dados, rastreia a origem da cultura, realiza diagnóstico socioambiental de pessoas, empresas e territórios, monitoramento agroclimático e inteligência territorial. O sistema se baseia nas camadas públicas e privadas de riscos socioambientais e agrícolas, agregando valor e mitigando riscos das negociações. Algumas funcionalidades desenvolvidas pelo autor deste relatório foram:

a) Criação da página inicial da plataforma com a funcionalidade denominada pela equipe de “busca inteligente”, contemplando um mapa que apresenta todos os territórios cadastrados pela empresa e uma barra de busca dos mesmos onde o usuário pode pesquisar pelo nome do território ou pelo nome do responsável pela fazenda;

---

[2] Agtech do Agronegócio do Brasil - Conheça a Vega. Vega. Disponível em: <https://vegamonitoramento.com.br/>. Acesso em: 25 jun. 2022.

b) Criação do recurso para consulta de diagnóstico socioambiental, onde é possível pesquisar por um CPF ou código CAR (cadastro ambiental rural, código único referente a um território) e obter um relatório contendo os dados socioambientais como sobreposição do território em área de proteção, terra indígena, entre outras;
c) Alteração da regra de negócio relacionada a visualização dos dados, onde inicialmente os dados eram mostrados apenas para os usuários que os cadastraram, e, após a alteração, todos os usuários de uma mesma empresa tinham acesso às informações cadastradas por outro colega de trabalho.

A equipe responsável pelo Lyra utilizou o Scrum como facilitador no processo de desenvolvimento, contando um *Product Owner,* um *Scrum Master* e quatro desenvolvedores, além de um analista de qualidade e uma gerente de produto. O autor deste trabalho atuou como estagiário juntamente aos desenvolvedores, dos quais eram: um de nível pleno e dois de nível sênior, os quais foram de suma importância, dando suporte técnico ao autor deste documento quando necessário.

1.3 Objetivo

Visando a capacitação do autor deste relatório, o período de estágio teve como objetivo aplicar, junto a equipe de desenvolvimento do Lyra, os conceitos teóricos adquiridos durante o período acadêmico na UFLA. Para isso, o time de colaboradores responsáveis pela plataforma Lyra, teve como objetivo evoluir a plataforma, a qual inicialmente era uma prova de conceito, também conhecida como POC.

1.4 Estrutura do trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2, são apresentados conceitos básicos sobre as tecnologias e ferramentas utilizadas no decorrer do estágio; no Capítulo 3, é exposto um contexto maior sobre a plataforma Lyra e algumas atividades desenvolvidas durante o período de estágio, assim como os resultados obtidos; finalizando, no Capítulo 4, as considerações finais são elucidadas.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentadas, sucintamente, as principais tecnologias, linguagens e metodologias utilizadas durante o cumprimento do estágio.

2.1 Scrum

Segundo o Scrum Guide (2020) “Scrum é uma estrutura leve que ajuda pessoas, equipes e organizações a gerar valor através de soluções adaptáveis para problemas complexos”. Foi criado no início dos anos 1990 tendo como base o empirismo, observando os seus atores (*Scrum Master, Product Owner* e *Scrum Team)* e empregando uma abordagem incremental. Para isso o Scrum possui três pilares: transparência, inspeção e adaptação.

O *Scrum Master* é responsável por garantir que a equipe atente-se para os ritos e valores do Scrum. Em relação ao *Product Owner*, este assegura o entendimento das regras de negócio do produto além de entregar valor ao cliente, ou seja, é a pessoa que gerencia e prioriza as tarefas que serão realizadas. O *Scrum Team,* ou time de desenvolvimento, são as pessoas que desenvolvem as funcionalidades do produto. Cada ciclo de Scrum é chamado *Sprint* e as regras do Scrum propõem quatro eventos que ocorrem ao decorrer do ciclo, são eles: o planejamento da *Sprint*, reunião diária, revisão da *Sprint* e retrospectiva da *Sprint*.

O planejamento da *Sprint,* é realizado para definir o que será feito ao decorrer do período do ciclo, nesta etapa é gerado o *Product Backlog,* artefato do Scrum que carrega o entendimento necessário para atender os requisitos propostos. As *Dailys,* ou reuniões diárias, ocorrem diariamente e garantem a transparência do Scrum, nesta reunião todos da equipe falam o que fizeram desde a última reunião, o que vão realizar até a próxima e se há algum impedimento para o prosseguimento da tarefa. Para garantir a inspeção, a revisão da *Sprint* é realizada. Nesta reunião são apresentadas todos os itens concluídos pelo time, possibilitando conferir e avaliar o que foi proposto e o que foi entregue. O rito final da *Sprint* é a retrospectiva, nesta etapa tudo o que foi positivo e negativo durante o ciclo é listado, os pontos positivos devem ser mantidos e soluções para os pontos negativos devem ser propostas para a próxima *Sprint,* seguindo o pilar da adaptação.

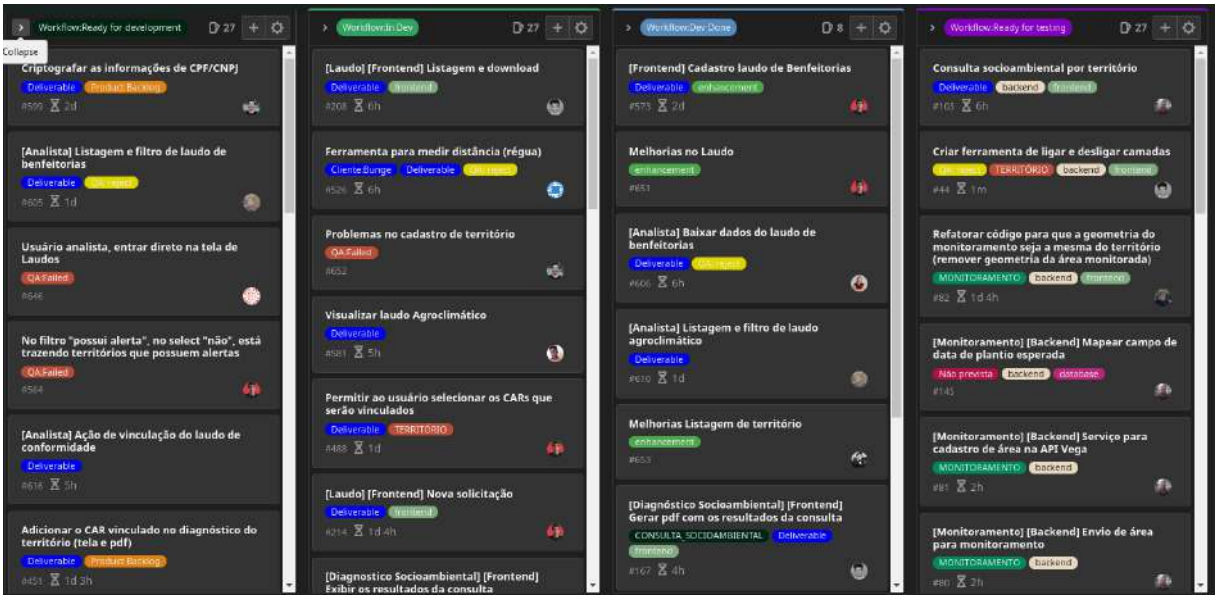
A disciplina “Modelagem e Implementação de Software — GCC132” foi o primeiro contato do autor com o *framework* Scrum. Na disciplina, a turma foi dividida em diversos

grupos e cada grupo se auto gerenciou tendo como base o Scrum para, no final do período, entregar um aplicativo para dispositivo Android.

Durante o período de estágio, o autor deste relatório atuou como parte do *Scrum Team* no desenvolvimento do Lyra. Para auxiliar na distribuição de terefas, a equipe utilizou a funcionalidade de *Issues* dentro do Gitlab, dessa forma foi possível garantir que nenhuma tarefa fosse realizada por mais de uma pessoa em simultâneo. O quadro de tarefas foi dividido em colunas de *workflow* para os desenvolvedores identificarem em qual etapa a tarefa se encontra, sendo:

a) *Ready for development*: nesta fase qualquer desenvolvedor pode assinar o cartão e definir a tarefa como sua.
b) *In dev: as* tarefas nesta coluna já estão sendo desenvolvidas.
c) *Dev Done:* as funcionalidades foram realizadas e o código está em análise para subir ao ambiente de teste.
d) *Ready for testing*: a partir desta coluna, as tarefas são de responsabilidade do analista de qualidade, o qual testa a funcionalidade desenvolvida. Caso haja algum erro, a tarefa volta para a coluna *ready for development.*
e) Também haviam colunas de responsabilidade do QA, como as colunas "*In test*" e "*Test done*"

Imagem 1 — Quadro de tarefas (*Issues)* da equipe responsável pelo Lyra

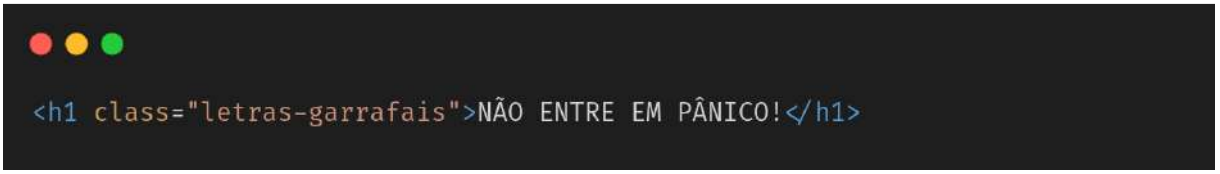


Fonte: Autor

2.2 HTML e Pug.js

A Mozilla Developer Network[3] descreve o HTML (*HyperText Markup Language*) como o bloco de construção mais básico da web, é a partir desta estrutura que os links entre páginas e conteúdo são criados, permitindo a navegação na *World Wide Web*. "HiperTexto" refere-se aos links que conectam os conteúdos, seja em uma página do mesmo site ou de um site externo, e "marcação" tem relação com o que é mostrado na tela, podendo ser um parágrafo, imagem, vídeos, dentro outras mídias.

Imagem 2 — Exemplo de um elemento HTML



Fonte: Autor

Observando a Imagem 2, é possível perceber que o HTML pode ser subdividido em várias estruturas chamadas elementos. Estes elementos podem ser subdivididos em:

a) *Tag* de abertura: refere-se ao nome do elemento, envolto em parênteses angulares. É utilizada para marcar o tipo e o início do elemento, no exemplo, onde começa o título. Ela pode ou não ter atributos, ou seja, informações que alteram o comportamento da estrutura, no caso, o título possui a classe "letras-garrafais" que poderá ser utilizada no CSS onde irá alterar o estilo do título.
b) *Tag* de fechamento: segue o mesmo padrão da *tag* de abertura, exceto que inclui uma barra antes do nome do elemento e não possui atributos, é utilizada para marcar onde o elemento acaba
c) Conteúdo: refere-se a todos os elementos entre as *tags* de abertura e fechamento. O conteúdo pode ser um texto simples, como no exemplo da Imagem 2, ou mais complexo como o exemplo da Imagem 3, o qual expõe todo o código para se fazer uma página com uma frase utilizando o HTML.

d) Elementos vazios: alguns elementos do HTML podem ser vazios, como uma imagem, desta forma não existe conteúdo ou *tag* de fechamento, restando apenas a *tag* de abertura com seus atributos.

Os elementos do HTML não são úteis por si só, uma página HTML é composta por vários elementos combinados de forma estruturada como no exemplo da Imagem 3, onde temos *tags* que definem o comportamento da página. Uma estrutura básica de um documento HTML é composta obrigatoriamente por um "<!DOCTYPE html>" sendo a *tag* inicial de todo documento HTML. Elemento "html" que envolve todo o conteúdo da página, também é conhecido como elemento raiz. Elemento "head" que age como recipiente de todo comportamento da página que não é um conteúdo, no exemplo, a utilização do UTF-8 como codificação dos caracteres do site. Por fim, o "body", que armazena todo o conteúdo que será exibido na página, no caso, um parágrafo definido pelo elemento "p".

Imagem 3 — Parágrafo simples utilizando HTML

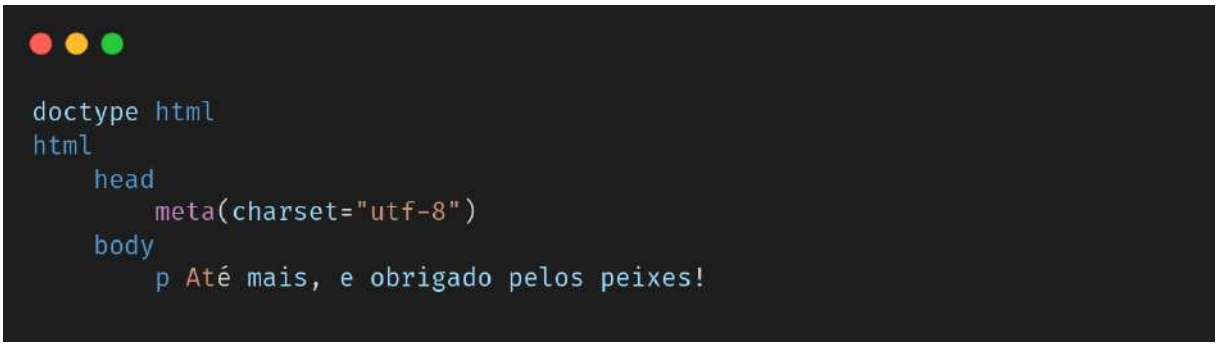
Fonte: Autor

Durante o período de estágio supervisionado a equipe responsável pelo desenvolvimento do Lyra optou por não usar o HTML diretamente, e sim o *template engine* Pug.js. Com esta biblioteca, o código fica mais legível à medida que a página se torna mais complexa e com mais conteúdo.

O Pug é um pré-processador que permite a criação de código HTML a partir de um documento JavaScript, ou seja, é possível escrever um código mais simples que será convertido para HTML no final do processo de compilação. Diferentemente do HTML, o

Pug.js não utiliza *tag* de fechamento, toda estrutura da página é controlada tendo como base a indentação, dessa forma o código se torna mais limpo e simples, sendo possível escrever o tipo do elemento (no caso, html) e logo a baixo, com a indentação, o conteúdo. Caso o elemento necessite de algum atributo, este é colocado em parênteses logo após a definição do tipo. A Imagem 4 demonstra a mesma página desenvolvida na Imagem 3, porém com a utilização do Pug.js.

Imagem 4 — Parágrafo simples utilizando Pug.js



Fonte: Autor

2.3 CSS

Cascading Style Sheets, popularmente chamado de CSS, é a linguagem utilizada para estilizar um documento HTML, ou seja, é onde estão contidas as instruções de como os elementos HTML devem ser exibidos na interface. Dessa forma se economiza trabalho, uma vez que uma folha de estilo pode controlar o *layout* de várias páginas da web de uma só vez ("CSS Introduction", 2022).

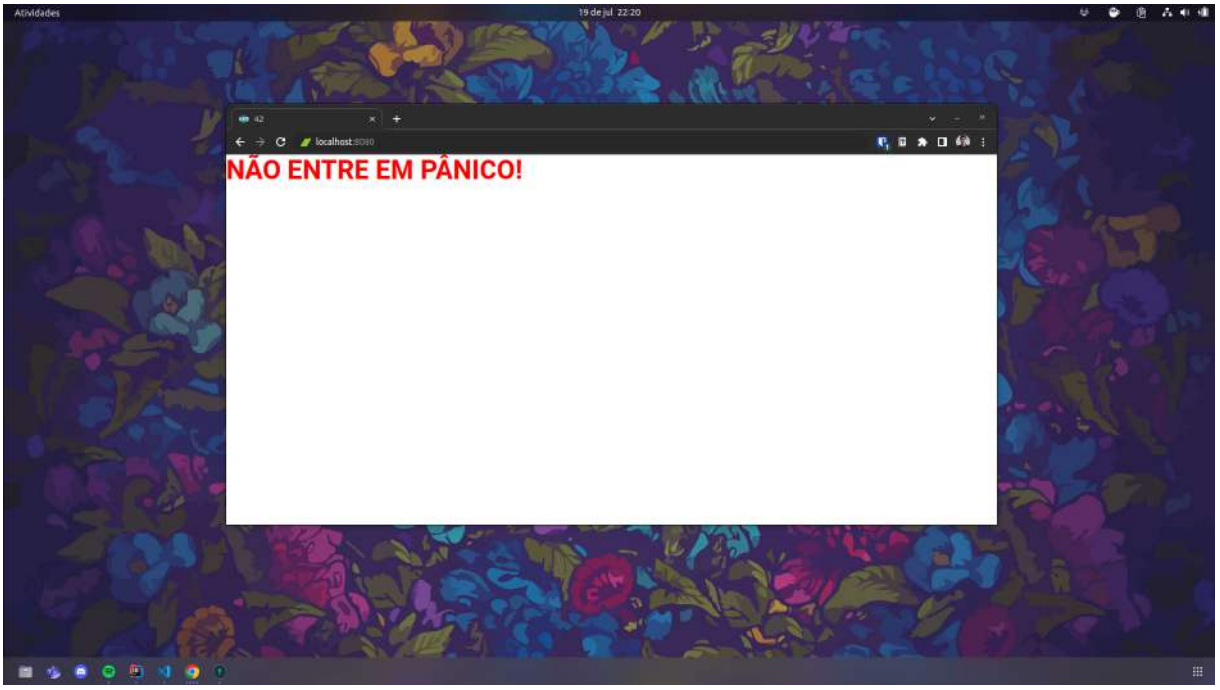
Imagem 5 — Exemplo código CSS

Fonte: Autor

Uma regra CSS é composta por um seletor e um bloco de declaração. O seletor aponta para o elemento HTML que será utilizado, já o bloco de declaração consiste em uma ou mais declarações entre chaves e separadas por ponto e vírgula. Por sua vez, cada declaração é realizada a partir de uma propriedade e seu valor separado por dois pontos.

A Imagem 5 exemplifica uma regra CSS, a regra é aplicada em cima do seletor h1. O bloco de declaração especifica, pela propriedade “color”, que o título terá cor vermelha e a propriedade “font-size” diz que o tamanho do texto exibido será de 42 píxeis.

Imagem 6 — Interface criada a partir do CSS da Imagem 5 aplicado ao HTML da Imagem 2



Fonte: Autor

2.4 JavaScript

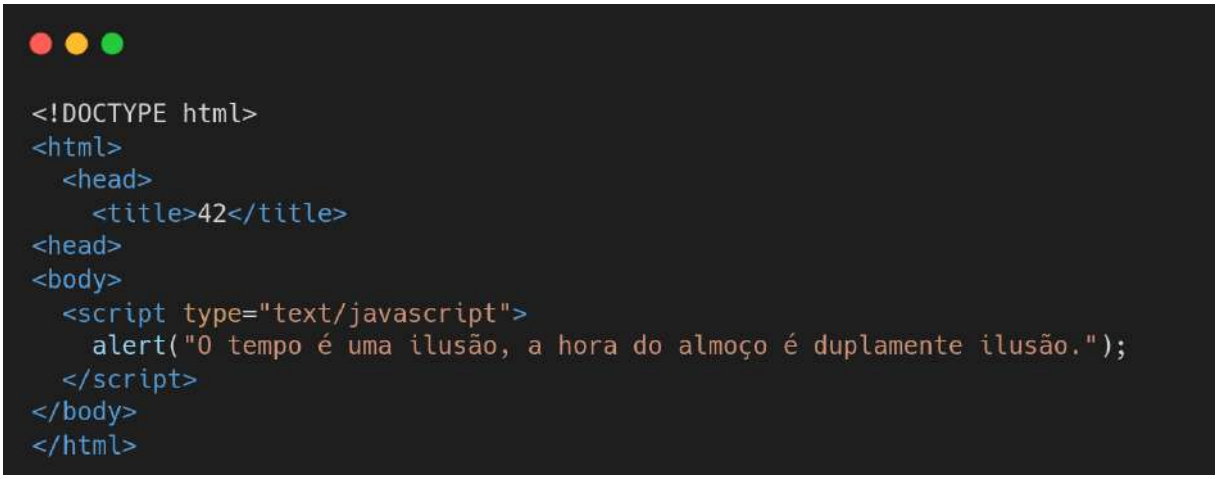
JavaScript é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas web. É uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (“JavaScript | MDN”, 2021).

Quase sempre que uma página web faz mais do que apenas mostrar informações estáticas, por exemplo, mostrar conteúdos atualizados em tempo real, mapas interativos,

animações gráficas em 2D/3D, vídeos, dentre outros conteúdos dinâmicos, estas funcionalidades possivelmente foram implementadas utilizando o JavaScript.

Junto ao HTML e o CSS, o JavaScript é um dos pilares do desenvolvimento front-end, os primeiros contatos do autor deste documento com as tecnologias voltadas para o desenvolvimento de interfaces gráficas foram nas disciplinas “GCC214 - Introdução a Sistemas de Banco de Dados” e “GCC188 - Engenharia de Software” onde ambos professores propuseram o desenvolvimento completo de uma aplicação web para o trabalho final.

Imagem 7 — Exemplo da utilização do JavaScript dentro de um documento HTML



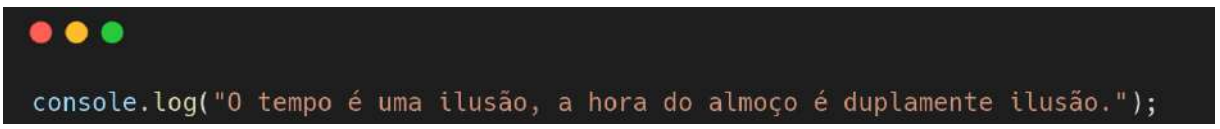
```
<!DOCTYPE html>
<html>
  <head>
    <title>42</title>
<head>
<body>
  <script type="text/javascript">
    alert("O tempo é uma ilusão, a hora do almoço é duplamente ilusão.");
  </script>
</body>
</html>
```

Fonte: Autor

2.5 Node.js e npm

O JavaScript foi criado sendo voltado para o lado do cliente, rodando direto no navegador. Porém, com sua popularização, foi desenvolvido um ambiente baseado no interpretador V8 do Google que permite a execução de códigos JavaScript fora de um navegador web, ou seja, do lado do servidor. O Node.js é um ambiente de tempo de execução JavaScript de código aberto e multiplataforma (NODEJS, 2022).

Imagem 8 — Exemplo de código utilizando JavaScript no ambiente Node.js



```
console.log("O tempo é uma ilusão, a hora do almoço é duplamente ilusão.");
```

Fonte: Autor

Com a possibilidade de se rodar o JavaScript tanto no cliente quanto no servidor, várias bibliotecas utilizando a linguagem surgiram, e como consequência houve a necessidade de se desenvolver algo para gerenciar essas bibliotecas dentro de um projeto para web. E é neste contexto que surge o npm, o maior registro de software do mundo, contendo mais de 800.000 pacotes de código (“What is npm”, 2022). O npm é instalado com o Node.js e inclui um CLI (Cliente de Linha de Comando) que pode ser utilizado para baixar e instalar um software contido no repositório do npm.

Imagem 9 — Exemplo do CLI no Windows

Fonte: “What is npm”, 2022

Imagem 10 — Exemplo do CLI no Mac OS

Fonte: “What is npm”, 2022

Pensando no escalonamento da aplicação, a base front-end da plataforma Lyra foi pensada utilizando o npm, reduzindo o atrito dos desenvolvedores ao incluir uma biblioteca nova no futuro, bastando apenas fazer a inclusão do pacote pelo CLI da ferramenta em Node.js.

2.6 Vue.js

Utilizando JavaScript, HTML e CSS, o Vue é um framework progressivo para a construção de interfaces de usuário (“Introdução | Vue.js”, 2022). Foi projetado desde sua concepção para ser adotado de forma incremental, sua biblioteca base é focada exclusivamente na camada visual, sendo facilmente integrada com outras bibliotecas ou projetos existentes. Segundo a documentação oficial, existem quatro formas principais de adicionar o Vue a um projeto:

a) Importar a partir de CDN na página desejada;
b) Baixar os arquivos JavaScript e hospedá-los no projeto;

c) Instalar usando npm;
d) Usar a ferramenta CLI oficial para criar o projeto, usando configurações previamente ajustadas para o desenvolvimento *frontend* moderno.

No núcleo do Vue está um sistema que permite renderizar dados no DOM (*Document Object Model*) de forma declarativa usando uma sintaxe de modelo simples. A partir dos atributos do objeto retornado na função data, é possível referenciar e utilizar seu valor na parte HTML do código utilizando o nome do atributo entre duas chaves. Baseando no exemplo da Imagem 2, a Imagem 11 usa a renderização declarativa do Vue para exibir o título na tela.

Imagem 11 – Exemplo renderização declarativa Vue

```
<html>
  <body>
    <div id="app">
      <h1> {{ titulo }} </h1>
    </div>
  </body>

  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

  <script type="text/javascript">

    const App = {
      data() {
        return {
          titulo: "NÃO ENTRE EM PÂNICO!"
        }
      }
    }

    Vue.createApp(App).mount('#app')
  </script>
</html>
```

Fonte: Autor

O Vue permite interligar os dados e o DOM, tornando tudo reativo, dessa forma não é necessário interagir diretamente com o HTML. Um app Vue acopla-se a um único elemento do DOM (#app, no exemplo) e então o controla completamente. O HTML é o ponto de entrada, mas o resto acontece dentro da recém criada instância do Vue (“Introdução — Vue.js”, 2022).

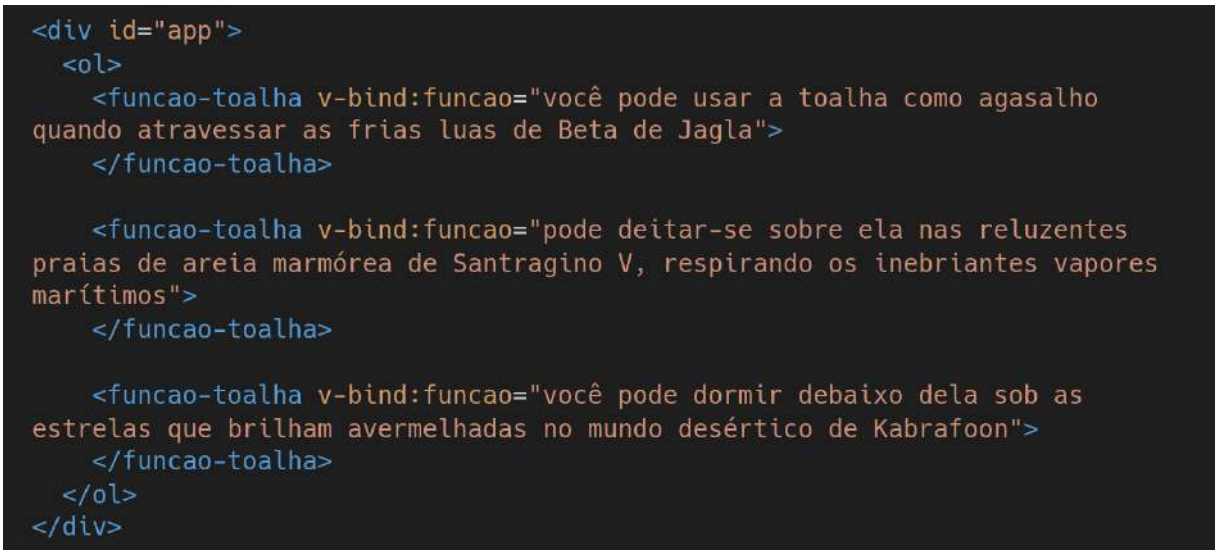
Outro importante conceito no Vue é o sistema de componentes, o qual proporciona uma abstração que permite a construção de uma interface baseada em pequenos conjuntos de funcionalidades auto contidas, possibilitando a reutilização do código. Por exemplo, em um componente de campo de texto é possível definir a lógica, estrutura e estilo desacoplado do código principal. O Vue.js também permite passar propriedades para os componentes, tornando-os mais dinâmicos e reutilizáveis.

Imagem 12 – Criação de um componente Vue

```
Vue.component('funcao-toalha', {
  props: ['funcao'],
  template: '<li>{{ funcao }}</li>'
})
```

Fonte: Autor

Imagem 13 – Utilização de um componente Vue



```
<div id="app">
  <ol>
    <funcao-toalha v-bind:funcao="você pode usar a toalha como agasalho
quando atravessar as frias luas de Beta de Jagla">
    </funcao-toalha>

    <funcao-toalha v-bind:funcao="pode deitar-se sobre ela nas reluzentes
praias de areia marmórea de Santragino V, respirando os inebriantes vapores
marítimos">
    </funcao-toalha>

    <funcao-toalha v-bind:funcao="você pode dormir debaixo dela sob as
estrelas que brilham avermelhadas no mundo desértico de Kabrafoon">
    </funcao-toalha>
  </ol>
</div>
```

Fonte: Autor

A Imagem 12 cria o componente “funcao-toalha” que recebe como parâmetro a propriedade “funcao” e cria um item de lista HTML. Já a imagem 13 recorre ao componente previamente criado para gerar uma lista ordenada, onde cada item da lista recebe um texto diferente através do “v-bind” que faz a transferência de dados para os componentes filhos. O

exemplo fictício apresentado é simples e poderia ser feito utilizando apenas elementos HTML, porém é possível observar o desacoplamento entre os códigos gerados. Em uma aplicação grande, como o exemplo da Imagem 14, é essencial dividir todo o aplicativo em componentes para tornar o desenvolvimento gerenciável (“Introdução — Vue.js”, 2022).

Imagem 14 – Exemplo da utilização real de componentes

```
<div id="app">
  <app-nav></app-nav>
  <app-view>
    <app-sidebar></app-sidebar>
    <app-content></app-content>
  </app-view>
</div>
```

Fonte: “Introdução — Vue.js”, 2022

Existem outras bibliotecas e *frameworks* que podem substituir o Vue.js, como o React[4] ou o Angular[5]. A utilização do Vue na plataforma Lyra se deu por sua facilidade de desenvolvimento, uma vez que sua estrutura é baseada fortemente em HTML, CSS e JavaScript, organizando o código destas tecnologias de forma intuitiva em um único arquivo de implementação.

2.7 Element UI

Element é uma biblioteca de componentes baseada em Vue 2.0 para desenvolvedores, designers e gerentes de produto (“Element - The world’s most popular Vue UI framework”, 2022). Ou seja, o Element é um conjunto de componentes que utiliza o Vue como base, acelerando o processo de desenvolvimento de algum software. Sua construção é pautada em 4 pilares, são eles:

a) Consistência: todos os elementos são alinhados com a lógica de utilização comum da internet e possuem o mesmo estilo de design, ícones e texto;

b) Retorno: permite o usuário perceber de forma clara a utilização do sistema através de estilo e efeitos interativos;
c) Eficiência: todos os componentes da biblioteca são criados de forma que o usuário consiga entender o fluxo que deve tomar, ajudando-o a tomar decisões mais rápido e com o menor atrito possível;
d) Controle: o estilo dos componentes criados pela biblioteca dão dicas das operações que o usuário pode tomar, mas não toma as decisões. Os usuários devem ter a liberdade para realizar a função, incluindo cancelar ou prosseguir com a operação a qualquer etapa.

A instalação recomendada é através do pacote "element-ui" gerenciado pelo npm, mas também é possível utilizar o Element UI pela sua versão via CDN, ou seja, importando o *script* diretamente no HTML.

2.8 Leaflet

Leaflet é a biblioteca JavaScript de código aberto líder para mapas interativos compatíveis com dispositivos móveis, contendo todos os recursos de mapeamento que a maioria dos desenvolvedores precisa (LEAFLET, 2022). A biblioteca é utilizada por grandes empresas como: Github, Meta, Evernote, dentre outras.

Imagem 15 – Mapa desenvolvido utilizando o Leaflet

Fonte: LEAFLET, 2022

O Leaflet foi desenvolvido desejando simplicidade, desempenho e usabilidade. Sendo assim, o código base não tenta fazer de tudo, focando em fazer as coisas básicas funcionarem, como, por exemplo: camadas de imagem que desenham o mapa, marcadores personalizados, controle básico de zoom e recursos visuais de animação dando resposta às interações do usuário.

Com a base sólida desenvolvida e aberta à comunidade de desenvolvedores, vários *plugins* foram criados para o Leaflet, adicionando recursos e aumentando o escopo de atuação da biblioteca. As extensões mais utilizadas são focadas em interações com o mapa, por exemplo, executar um evento a partir de um clique em algum ponto do mapa, mas também existem extensões ligadas a sobreposição de dados no mapa, como mapas de calor e marcadores mais complexos

2.9 Java e Spring Boot

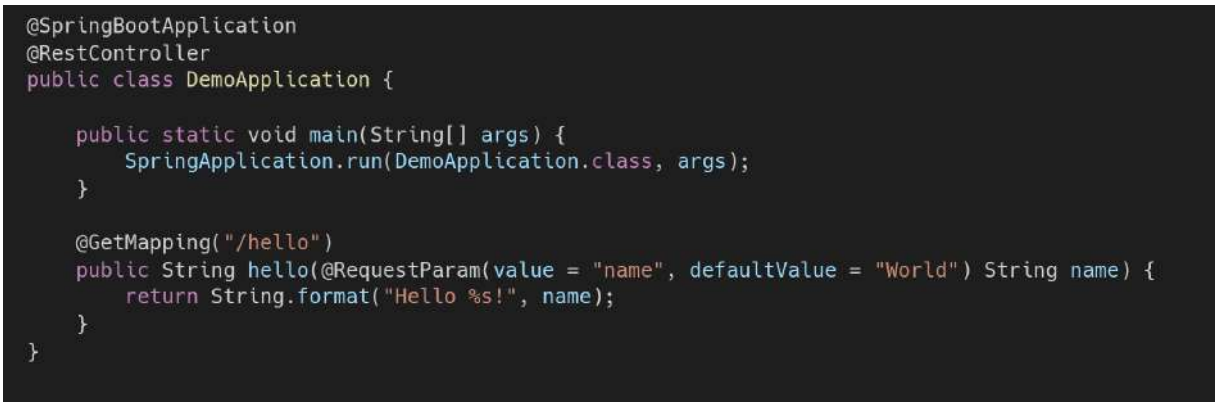
Java é uma linguagem de computador amplamente utilizada que foi criada com o propósito de ser multiplataforma através da JVM (*Java Virtual Machine*), capaz de interpretar o código, controlar a execução das pilhas, gerenciar memória, etc. Sendo assim, é possível rodar java em computadores, tablets, celulares, equipamentos de Blu-Ray, dentre outros. Porém, desde sua criação, existe um esforço dos desenvolvedores focando mais a linguagem para sistemas web.

O autor teve forte contato com a linguagem Java na disciplina “GCC178 - Práticas de Programação Orientada a Objetos”. Por ser uma linguagem baseada no paradigma orientado a objetos, o Java foi utilizado ao decorrer de todo o período de realização da disciplina.

> Com a popularização da rede Internet, os pesquisadores da Sun Microsystems perceberam que aquele seria um nicho ideal para aplicar a recém-criada linguagem de programação. A partir disso, adaptaram o código Java para poder ser utilizado em microcomputadores conectados à rede Internet, mais especificamente no ambiente da World Wide Web. Java permitiu a criação de programas batizados applets, que trafegam e trocam dados através da Internet e se utilizam da interface gráfica de um web *browser*. Implementaram também o primeiro *browser* compatível com a linguagem, o HotJava, que fazia a interface entre as aplicações Java e o sistema operacional dos computadores. (JAVA; SOARES INDRUSIAK, 1996).

Com a popularização do Java, alguns *frameworks* foram desenvolvidos para facilitar a construção de sistemas mais complexos utilizando a linguagem. O mais popular, atualmente, é o Spring Framework. ALURA (2021), “com o Spring é possível ter maior domínio do projeto que está sendo desenvolvido, tendo como maior característica o suporte à infraestrutura direto na aplicação, permitindo assim que os times de desenvolvimento possam se concentrar na parte lógica da aplicação”

Imagem 16 – Exemplo de aplicação “Hello World” utilizando o Spring Framework



```
@SpringBootApplication
@RestController
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @GetMapping("/hello")
    public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
        return String.format("Hello %s!", name);
    }
}
```

Fonte: “Spring Quickstart Guide”, 2022

A Imagem 16 refere-se a uma rota que, quando requisitada, irá retornar um dado na API com a *string* “Hello” e o nome passado na requisição. Para isso, a função “*main”* inicia a aplicação *Spring* e a função “*hello*” processa a requisição solicitada.

3 ATIVIDADES DESENVOLVIDAS

Neste capítulo são apresentados alguns dos trabalhos desenvolvidos durante o período de estágio supervisionado e um contexto maior do ecossistema da plataforma Lyra.

3.1 Vega API

A Vega API, como o próprio nome sugere, é uma API de uso interno da Vega Monitoramento. Foi desenvolvida no estilo arquitetural REST utilizando o Spring Boot como base de implementação. O estilo arquitetural REST enfatiza a escalabilidade das interações entre componentes, a generalidade das interfaces, o desenvolvimento independente de componentes e a utilização de componentes intermediários para reduzir a latência das interações, reforçar a segurança e encapsular sistemas legados (Fielding 2000).

Durante o período de estágio, a API ainda estava em desenvolvimento, se adequando às novas funcionalidades da plataforma Lyra. Por ser um serviço à parte, a ampliação da capacidade da API não afetou diretamente o processo de incremento da plataforma Lyra, necessitando apenas repassar aos desenvolvedores da plataforma a URL de acesso à funcionalidade criada na API. Alguns dos serviços oferecidos pela Vega API, para uso interno da empresa, são:

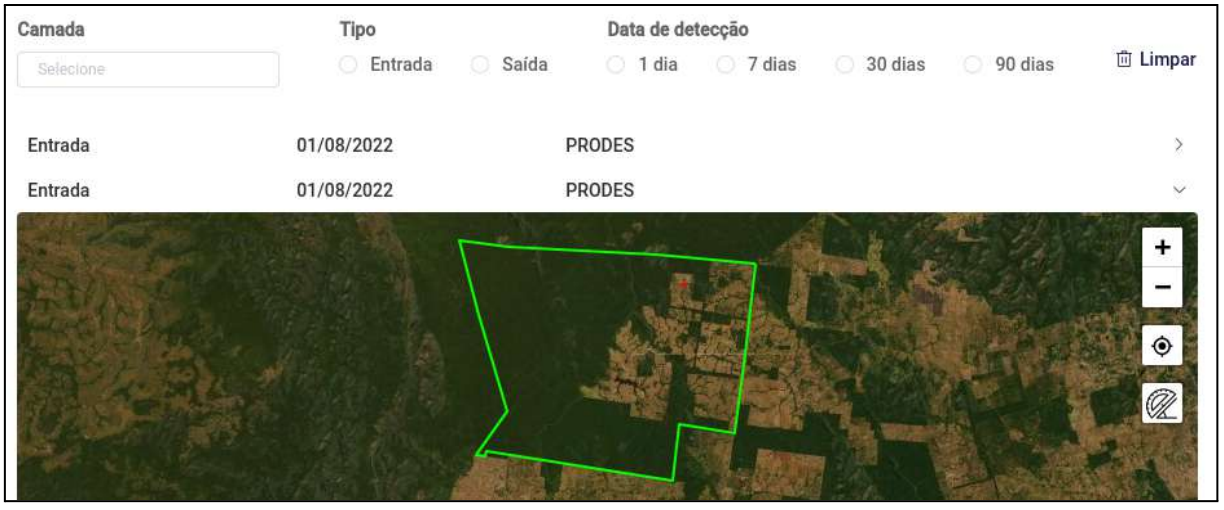
a) Realização do diagnóstico socioambiental por código CAR, CPF, CNPJ ou geometria espacial do território a ser analisado: nesta funcionalidade, é verificado se existem sobreposições (chamadas de camadas), tanto de dados privados quanto de dados públicos, gerando um relatório de todas as conformidades e inconformidades do valor buscado conforme a legislação. Por exemplo, se a fazenda consultada faz parte de um território indígena, há uma inconformidade segundo a lei e o relatório gerado discrimina a área, a geometria e o nome do território indígena sobreposto;

Imagem 17 – Exemplo fictício de uma consulta de diagnóstico socioambiental

Fonte: Autor

b) Geração de alerta: assim como no diagnóstico socioambiental, o território monitorado passa por validações legais de sobreposição de camadas, porém nesta funcionalidade a validação é feita de forma automática diariamente, gerando uma notificação caso haja uma irregularidade;

Imagem 18 – Exemplo fictício de uma consulta de alertas



Fonte: Autor

c) Monitoramento agroclimático e socioambiental: é a principal funcionalidade da Vega API. Gera relatórios agrupando vários dados, sendo estes dados das funcionalidades apresentadas anteriormente e dados gerados pela equipe de geoprocessamento da empresa. As informações obtidas através dos cruzamentos destes dados são úteis na tomada de decisão dos usuários da plataforma, uma vez que permite a rastreabilidade de todo processo agrícola.

3.2 Plataforma Lyra

Ao abrir o site que hospeda a plataforma Lyra, o usuário se depara com dois campos, e-mail e senha, utilizados para fazer o login no sistema. Ao entrar no sistema, é apresentado um resumo dos dados cadastrados pelos usuários da empresa do colaborador que está acessando o sistema e um menu lateral que permite acessar as outras funcionalidades que o usuário tem permissão.

Atualmente o menu lateral conta com sete botões nos perfis de usuários com a permissão de administrador, seis botões para usuários regulares e apenas um para analistas. São esses, de cima para baixo:

a) Botão “início”: apresenta o resumo do sistema, com territórios cadastrados, a soma das áreas destes territórios, territórios monitorados, a soma das áreas dos territórios monitorados e a quantidade de representantes cadastrados. Existe também um mapa com um campo de texto utilizado para fazer consultas rápidas dos territórios e representantes cadastrados, a partir desta busca é possível navegar para outras funcionalidades do sistema, por exemplo, realizar o diagnóstico socioambiental de um território;
b) Botão “representante”: nesta funcionalidade é exibido para o usuário uma lista paginada de no máximo vinte itens com os dados de todos os representantes cadastrados na empresa do usuário que está acessando o sistema. Estes dados são: nome, CPF ou CNPJ e e-mail. Também é possível cadastrar, pesquisar, editar, remover ou realizar uma consulta ambiental com o representante;
c) Botão “território”: assim como na funcionalidade de representantes, neste item do menu é apresentada uma lista paginada com dados dos territórios cadastrados.

Também é possível fazer o cadastro, pesquisa, edição e remoção de algum território, além de poder enviá-lo para monitoramento ou para a funcionalidade de diagnóstico socioambiental e o usuário também pode executar a ação de fazer o download da geometria de algum território no formato KML;

d) Botão “monitoramento”: apresenta um mapa semelhante ao botão inicial com um resumo dos territórios que se encontram em monitoramento, juntamente as suas notificações de alerta, separadas por cada tipo. Abaixo do mapa existe uma tabela que lista os dados dos territórios monitorados, caso seja do interesse do usuário é possível realizar um filtro nestes territórios. Apenas perfis de usuários de empresas que contrataram este serviço apresentam esta funcionalidade;
e) Botão “diagnóstico socioambiental”: nesta funcionalidade o usuário pode realizar uma consulta socioambiental de um CPF, CNPJ, CAR ou arquivos do formato KML, KMZ, WKT e Shapefile que contém a geometria a ser analisada. A interface apresenta uma coluna à esquerda com as informações obtidas na consulta e um mapa a direita com o território, caso o diagnóstico tenha sido em cima de um;
f) Botão “laudos” para usuários regulares: apresenta uma lista paginada com os laudos solicitados pela empresa do usuário logado no sistema, é possível fazer um filtro nesta tabela. Caso o usuário deseje, o mesmo pode solicitar um novo laudo, visualizar um resumo ou fazer o download de um laudo solicitado;
g) Botão “laudos” para usuários analistas: funcionalidade disponível apenas para funcionários dentro da Vega. Apresenta uma tabela listando todas as informações de todos os laudos solicitados, estas informações podem ser baixadas no formato CSV. Com o laudo verificado, o usuário pode fazer o upload de um arquivo PDF contendo o resultado da solicitação;
h) Botão “usuários”: funcionalidade disponível apenas para administradores, apresenta uma tabela com os usuários cadastrados no sistema e fazem parte da mesma empresa do usuário logado. É possível cadastrar um novo usuário no sistema, além de poder pesquisar, editar, inativar ou enviar um e-mail de primeiro acesso para usuários cadastrados.

Em todas as funcionalidades, o usuário pode sair da plataforma a partir de um botão no canto superior direito da interface. Caso queira mais informação, o usuário pode ainda expandir o menu lateral tendo acesso ao nome das funcionalidades do sistema.

3.3 Diagnóstico socioambiental - estrutura

Durante o período de estágio supervisionado, o autor deste relatório foi responsável por implementar o *frontend* da funcionalidade completa de diagnóstico socioambiental, ou seja, criação da interface, controle de rotas e comunicação com o *backend*.

Imagem 19 – Representação da funcionalidade de diagnóstico socioambiental

Fonte: Autor

Para facilitar a implementação da interface, a funcionalidade foi quebrada em componentes, a Imagem 19 demonstra como os componentes estão apresentados na interface, representados em:

a) Verde: representa o componente de mapa já implementado no sistema, sendo assim, o autor apenas fez uso. É utilizado para mostrar o território consultado e a geometria das camadas inconsistentes.

b) Laranja: é um componente do tipo *accordion*, ou seja, expande os componentes filhos conforme a necessidade. A Vega API apresenta muitas camadas de consulta, usando um *accordion* fica mais fácil para o usuário ver apenas o nome e, caso seja de seu interesse, basta expandir o componente para visualizar todos os dados da camada escolhida. Ao fazer isso, caso a camada apresente uma geometria que não é válida legalmente, essa geometria é mostrada no mapa.
c) Vermelho: são componentes simples que mostram os dados de uma camada específica.

Além dos componentes previamente descritos, existem ainda o título, um campo para o usuário inserir os dados da consulta e um botão para realizar a ação da mesma, que são componentes do Element UI. A página em si também é um componente.

Como os dados da Vega API são coletados de diversos lugares, a plataforma Lyra recebe um JSON com dados estruturados de forma diferente para cada camada. Por exemplo, a camada de queimadas apresenta os atributos de latitude e longitude, que não estão presentes na camada de terras indígenas. Portanto, existem componentes simples (representados em vermelho na Imagem 19) que fazem o mapeamento destes dados para cada camada, este componente recebe como parâmetro uma lista contendo as restrições da camada, estas restrições são os dados que serão mapeados. As Imagens 20 e 21 demonstram a implementação deste mapeamento para a camada de queimadas, o restante da implementação do código deste componente é a estrutura de um componente Vue.js, ou seja, os atributos “name”, “data” e “props”.

Imagem 20 – Parte da implementação da estrutura do componente “Queimadas”

```
<template lang="pug">
.container-restricao
    div(v-if="restricoes.length > 0" v-for="restricao in listaRestricoes")
        p
            b Latitude:
            |  {{ restricao.latitude || '-' }}
            br
            b Longitude:
            |  {{ restricao.longitude || '-' }}
</template>
```

Fonte: Autor

Imagem 21 – Implementação da função de mapeamento do componente “Queimadas”

```
computed: {
    listaRestricoes: function () {
        return this.restricoes.map((value) => {
            if (value.properties) {
                return {
                    ...value.properties
                };
            }
            return value;
        });
    }
}
```

Fonte: Autor

Para agrupar os componentes criados para cada camada, foi utilizado o componente “el-collapse” no modo *accordion*, disponível no Element UI. Dessa forma foi possível fazer o controle do item expandido através da propriedade “v-model” que recebe o índex da camada que está sendo mostrada.

O “el-collapse” permite definir um título simples para cada seção através da propriedade “title”. Porém, como foi necessário definir um ícone junto ao título, foi necessário criar um elemento do tipo “template”, que é utilizado pelo Vue para atribuir um trecho de código HTML a uma propriedade.

Imagem 22 – Estrutura do componente de listagem de camadas

```
<template lang="pug">
#diagnostico-socioambiental
    el-collapse(v-model="indexCollapseRow" accordion)
        el-collapse-item(
            v-if="showLayer(diagnostico)"
            v-for="(diagnostico, index)"
            :name="index")

            template(slot="title")
                i.mr-2.flow.flow-verified.verde.mr8(v-if="diagnostico.resultados.length")
                i.mr-2.flow.flow-warning-sign.vermelho.mr8
                    | {{diagnostico.camada.nome}}

            div(v-if="showLayerResults(diagnostico)")
                component(
                    v-bind:is="diagnostico.component"
                    v-bind="getProperties(diagnostico)")

            .container-restricao(v-if="diagnostico.resultados.length === 0")
                p Não há restrições!
</template>
```

Fonte: Autor

Como as camadas são trocadas frequentemente, existe uma constante chamada “camadasDiagnostico” que contém uma lista com os códigos de todas as camadas que podem ser mostradas ao usuário, assim é possível renderizar os componentes de camadas de forma dinâmica a partir de duas funções que fazem essa validação. A função “showLayer” apenas retorna verdadeiro ou falso, dizendo se a camada pode ou não ser apresentada. Já a função “showLayerResults” atribui um componente a essa camada caso haja restrições, caso contrário apenas é exibido o texto “Não há restrições!”.

Imagem 23 – Funções de exibição do componente de camada

```
showLayer (diagnostico) {
    if (diagnostico) {
        return camadasDiagnostico.includes(diagnostico.camada.codigo);
    }
    return false;
}

showLayerResults (diagnostico) {
    if (diagnostico && diagnostico.resultados.length) {
        diagnostico.component = diagnostico.camada.codigo.toLowerCase().replaceAll('_', '-');
        return true;
    }
    return false;
}
```

Fonte: Autor

Para realizar o controle da geometria desenhada no mapa, foi necessário comunicar para o componente pai qual a camada está aberta, isso foi feito através de uma função que observa a variável “indexCollapseRow”, toda vez que esta variável tem o valor alterado, o novo valor é passado para o componente pai.

O componente de nome “DiagnosticoSocioambiental” agrupa todos os componentes menores criados anteriormente. Para estilização da interface, o componente foi dividido em duas “divs”, uma com classe “card-socioambiental” e outra com a classe “mapa”, a primeira agrega todas as informações escritas da funcionalidade, enquanto a segunda contempla as informações gráficas, ou seja, o mapa com as geometrias.

Dentro da “div” de informações existem mais três subdivisões utilizando elementos HTML, o “header” contendo um ícone de seta e o título da página, ao clicar sob o título o usuário é levado a funcionalidade que estava sendo executada antes do diagnóstico. A “div” de pesquisa separa o componente de input do Element UI do resto da página, este componente

recebe alguns atributos como parâmetro, os mais importantes são: “v-mask” o qual cria uma máscara no valor inserido, “v-model” que referencia a variável que armazena o valor do texto e o “@change” que dispara uma função quando o usuário pressiona a tecla “enter”. Por fim, a terceira “div” contém os resultados do diagnóstico socioambiental, sendo assim, o componente de camadas é referenciado nesta parte da estrutura. A Imagem 24 apresenta parte do código da interface da funcionalidade.

Imagem 24 – Código do elemento com a classe “card-socioambiental”

```
.card-socioambiental
    header
        h1
            .el-icon-arrow-left.voltar(@click="clear")
            | Diagnóstico Socioambiental

    .pesquisa(v-if="!dadosConsulta")
          el-input(
              type="text"
              placeholder="Digite o CPF/CNPJ ou CAR"
              style = "width: 100%;"
              v-mask="['XXX.XXX.XXX-XX', 'XX.XXX.XXX/XXXX-XX', 'XX-XXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX']"
              v-model="pesquisa"
              @change="pesquisar")

          el-button(type="primary" icon="el-icon-search" @click="pesquisar")
                | Pesquisar

    .dados(v-if="dadosConsulta")
        header.mb-3
            p Resultados do Diagnóstico Socioambiental
                br
                b {{pesquisado}}

        DiagnosticoSocioambiental(
            :dadosConsulta="dadosConsulta"
            @index-aberto="mostrarAreasDiagnosticoNoMapa")
```

Fonte: Autor

Como o componente de mapa já havia sido desenvolvido, foi necessário apenas aplicá-lo conforme a necessidade da funcionalidade. Este componente pode receber alguns atributos por parâmetros, são eles:

a) mapLayersConfigs: recebe uma lista de objetos responsáveis por inserir desenhos no mapa. Para a funcionalidade de diagnóstico socioambiental, esse objeto recebe apenas a geometria e o estilo que será apresentado;
b) mapConfig: recebe um objeto contendo a configuração de apresentação do mapa, por exemplo, os controles que o mapa terá, as funcionalidades que poderão ser realizadas diretamente no mapa, dentre outras propriedades. Para a funcionalidade proposta, foi necessário apenas adicionar ao mapa os controles de zoom, camadas e permitir o modo tela cheia do mapa;

c) idMapa: recebe uma string que define o id que o mapa receberá, dessa forma cada mapa criado no sistema se torna único, evitando possíveis conflitos caso haja mais de um mapa na mesma funcionalidade.

Imagem 25 – Utilização do componente de mapa ano diagnóstico socioambiental

```
.mapa
    mapa(
        :mapLayersConfigs="layersMaps"
        :mapConfig="mapConfig"
        :idMapa="'mapaDiagnosticoSocioambiental'")
```

Fonte: Autor

3.4 Diagnóstico socioambiental - lógica

Ao realizar uma pesquisa, através de um clique no botão ou pressionando a tecla "enter", o usuário dispara a função "pesquisar" a qual irá buscar os dados da consulta e em seguida mostrar a geometria do território no mapa, caso seja uma consulta por código CAR.

A busca dos dados é realizada na função "buscarDadosConsulta". A primeira etapa da função é verificar o tipo de dados que está sendo consultado, CPF/CNPJ ou código CAR, para cada tipo é necessário passar valores diferentes para a consulta. Logo após os dados serem recebidos, esses dados são tratados para serem passados de forma homogênea para o componente de camadas. Por fim, o valor consultado é salvo. Em qualquer etapa do processo, caso ocorra algum problema, uma mensagem é mostrada alertando o usuário que ocorreu um erro ao buscar os dados.

Caso a consulta seja realizada em um código CAR, é necessário desenhar este território no mapa, a função responsável por isso é a "mostrarTerritorioNoMapa". Inicialmente ela limpa a lista de objetos desenhados no mapa, para ser exibido apenas o território, após isso, é criado um objeto com o estilo definido no sistema para áreas monitoradas, nome de "Território" e a geometria do imóvel CAR consultado. Este objeto é inserido na lista e, assim, sendo exibido no mapa.

Imagem 27 – Funções responsáveis pela consulta do diagnóstico socioambiental

```
async pesquisar () {
    await this.buscarDadosConsulta();
    this.mostrarTerritorioNoMapa();
},

async buscarDadosConsulta () {
    try {
        const consultaResponse = this.pesquisa.length === 43
            ? await DiagnosticoSocioambientalService.consultar('CODIGOCAR', this.pesquisa.toUpperCase())
            : await DiagnosticoSocioambientalService.consultar('CPF_CNPJ', formatters.cpfCnpjFormatToNumeral(this.pesquisa));

        this.dadosConsulta = consultaResponse.data[0].diagnosticos
            ? consultaResponse.data[0]
            : { diagnosticos: consultaResponse.data };

        this.pesquisado = this.pesquisa;
    } catch (error) {
        this.$notify.error({
            message:
                'Erro ao buscar dados'
        });
    }
},

mostrarTerritorioNoMapa () {
    this.layersMaps = [];

    if (!this.dadosConsulta.imovel) return;

    const layerTerritorio = {
        style: LayerStyle.AREA_MONITORADA(),
        name: 'Território',
        geoJson: this.dadosConsulta.imovel.geometria
    };

    this.layersMaps.push(layerTerritorio);
},
```

Fonte: Autor

A função “mostrarAreasDiagnosticoNoMapa”, executado a partir do índex aberto pelo usuário no componente que agrupa as camadas, exibe no mapa as áreas de restrições encontradas. A função começa limpando o mapa, deixando apenas o território consultado, para isso utiliza a função “mostrarTerritorioNoMapa”. Caso o usuário tenha fechado uma seção do componente, o índex informado é uma string vazia, isso ocorrendo, a função se encerra. Caso contrário, é feito um filtro nos dados retornados na consulta, gerando uma lista apenas com geometrias. Para cada geometria é criado um objeto para o componente de mapa para que ela seja exibida juntamente ao território.

Imagem 28 – Função para exibir as restrições no mapa

```
mostrarAreasDiagnosticoNoMapa (aberto) {
    this.mostrarTerritorioNoMapa();
    if (aberto === '') return;

    const areas = this.dadosConsulta.diagnosticos[aberto].resultados
        .filter((resultado) => resultado.geometria || resultado.geometry);

    areas.forEach((area, index) => {
        let layerAreaDiagnostico = {
            name: `AreaDiagnostico ${index}`,
            geoJson: area.geometria || area.geometry
        };

        this.layersMaps.push(layerAreaDiagnostico);
    });
}
```

Fonte: Autor

4 CONCLUSÃO

Além do valor financeiro, o período de estágio proporciona uma experiência enriquecedora para o crescimento da carreira e amadurecimento profissional. Também agrega valor aos currículos dos recém-formados, gerando novas oportunidades de aprimoramento das habilidades iniciadas no período de formação na universidade, sendo estas habilidades técnicas ou interpessoais.

4.1 Aplicação da graduação no período de estágio

O período de formação na universidade foi muito importante para que o desenvolvimento das atividades fossem realizadas de forma eficaz. As disciplinas focadas na base da programação, como Introdução aos Algoritmos e Estrutura de Dados, permitiram o desenvolvimento do raciocínio lógico aplicado em outras linguagens de programação utilizadas durante o período de estágio.

Disciplinas como Engenharia de Software e Modelagem e Implementação de Software tiveram papel fundamental no que se refere ao entendimento completo das regras de negócio, dando a base teórica para todo o ciclo de criação aplicado na plataforma Lyra. Plataforma a qual recorre a um banco de dados, sendo assim, disciplinas como a de Introdução aos Sistemas Gerenciadores de Banco de Dados, foram cruciais para a realização das tarefas propostas.

Por fim, cabe ressaltar que como boa parte da documentação das tecnologias utilizadas durante o período de estágio são em inglês, as disciplinas eletivas e cursos internos da universidade voltadas ao aperfeiçoamento da linguagem inglesa foram essenciais para a rápida compreensão das informações passadas nas documentações.

4.2 Consideração final

Com o período de estágio supervisionado é possível observar que existe uma defasagem entre o mercado de trabalho e a universidade, isso se deve ao fato das tecnologias de desenvolvimento estarem em constante evolução. Porém, durante o período de graduação em Ciência da Computação é passado aos alunos toda a base da computação utilizada pelas

novas tecnologias, fazendo com que haja um atrito menor no aperfeiçoamento de novas ferramentas para o desenvolvimento. Dessa forma, o período de estágio é fundamental para a aplicação dessa base e iniciação no mercado de trabalho, uma vez que complementa o que já foi passado na universidade.

5 REFERÊNCIAS

BITTENCOURT, J.; MURILO, C.; **Spring: Conheça esse framework Java**. Disponível em: <https://www.alura.com.br/artigos/spring-conheca-esse-framework-java>. Acesso em: 24 jul. 2022.

CSS Introduction. Disponível em: <https://www.w3schools.com/css/css_intro.asp>. Acesso em: 20 jul. 2022.

Element - The world's most popular Vue UI framework. Disponível em: <https://element.eleme.io/#/en-US>. Acesso em: 23 jul. 2022.

FERREIRA DE OLIVEIRA, J. **A utilização da metodologia Scrum sob a percepção da equipe de desenvolvimento em uma empresa privada de software: Um estudo de caso.** Monografia (Graduação) - Universidade Federal da Paraíba, 2014.

FIELDING, R. **Architectural Styles and the Design of Network-based Software Architectures**. 100 p. Tese (Doutorado) — University of California, 2000.

Getting Started – Pug. Disponível em: <https://pugjs.org/api/getting-started.html>. Acesso em: 10 jul. 2022.

Introdução | Vue.js. Disponível em: <https://vuejsbr-docs-next.netlify.app/guide/introduction.html>. Acesso em: 22 jul. 2022.

JavaScript | MDN. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 20 jul. 2022.

NODE.JS. **Docs | Node.js**. Disponível em: <https://nodejs.org/pt-br/docs/>. Acesso em: 20 jul. 2022.

NODEJS. **nodejs/node: Node.js JavaScript runtime**. Disponível em: <https://github.com/nodejs/node>. Acesso em: 31 jul. 2022.

SOARES INDRUSIAK, L. **Grupo JavaRS JUG Rio Grande do Sul. [s.l: s.n.]**. Disponível em: <https://www.cin.ufpe.br/~arfs/introjava.pdf>.

SOARES DA SILVA JUNIOR, V. **Relatório de estágio - Desenvolvimento e manutenção de software na empresa Technolog.** Monografia (Graduação) - Universidade Federal de Lavras, 2019.

Scrum Guide | Scrum Guides. Scrumguides.org. Disponível em: <https://scrumguides.org/scrum-guide.html>. Acesso em: 3 jul. 2022.

Spring Quickstart Guide. Disponível em: <https://spring.io/quickstart>. Acesso em: 24 jul. 2022.

What is npm. Disponível em: <https://www.w3schools.com/whatis/whatis_npm.asp>. Acesso em: 20 jul. 2022.