WITOR ÂNGELO BUENO

DESENVOLVIMENTO DE PLATAFORMA GEOESPACIAL PARA A WEB NO PROJETO DO OBSERVATÓRIO DA AGROPECUÁRIA BRASILEIRA

LAVRAS - MG

2021

WITOR ÂNGELO BUENO

DESENVOLVIMENTO DE PLATAFORMA GEOESPACIAL PARA A WEB NO PROJETO DO OBSERVATÓRIO DA AGROPECUÁRIA BRASILEIRA

Relatório de Estágio apresentado à Universidade Federal de Lavras como parte das exigências do curso de Sistemas de Informação, para a obtenção do título de Bacharel.

Prof. Dr. Ramon Gomes Costa

Orientador

LAVRAS - MG

2021

WITOR ÂNGELO BUENO

DESENVOLVIMENTO DE PLATAFORMA GEOESPACIAL PARA A WEB NO PROJETO DO OBSERVATÓRIO DA AGROPECUÁRIA BRASILEIRA

Relatório de Estágio apresentado à Universidade Federal de Lavras como parte das exigências do curso de Sistemas de Informação, para a obtenção do título de Bacharel.

APROVADA em 03 de Novembro de 2021.

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

Prof. Dr. Ramon Gomes Costa
Orientador

LAVRAS - MG
2021

AGRADECIMENTOS

Agradeço à minha família e amigos, especialmente aos amigos Igor e Hicaro pelo apoio e motivação oferecidos. Agradeço aos professores da minha graduação, em especial o meu orientador Ramon, pela paciência e por me instruir dentro do possível. Agradeço à minha companheira Samara pelo apoio e carinho durante todo o caminho.Por fim, agradeço a todos da empresa GT pela oportunidade de estágio, ensinamentos e amizade.

RESUMO

O presente trabalho descreve as atividades realizadas em um estágio supervisionado na empresa GT – Global Technologies, onde o aluno realizou atividades em análise e desenvolvimento de sistemas, em especial no projeto Observatório da agropecuária brasileira, um sistema de consulta e visualização de dados. Serão apresentadas as atividades realizadas, tecnologias e metodologias utilizadas e sua relação com o que foi aprendido no curso de Bacharelado em Sistemas de Informação, além do conhecimento adquirido durante o estágio.

Palavras-chave: Desenvolvimento Web; Geoprocessamento; Estágio; Java; JavaScript.

SUMÁRIO

1 INTRODUÇÃO

A ascensão da *internet*, e de quaisquer recursos digitais nos últimos anos, trouxe um aumento de demanda no mercado de desenvolvimento de *softwares*, tendo clientes desde cidadãos comuns até empresas e gestão pública. Os *softwares* podem atender a muitos fins e serem implementados de diversas maneiras diferentes, aqui se destaca a implementação de uma aplicação Web, que são sistemas projetados para utilização através de um navegador com acesso a internet.

O Curso de Bacharelado em Sistemas de Informação oferece uma base sólida para o desenvolvimento e entendimento de *softwares*. Assim como em muitas outras áreas, entretanto, com a demanda crescente e avanço da tecnologia, nota-se que o mercado tem exigências que não são possíveis de se encaixar no currículo de uma graduação.

Optando por um estágio, é possível colocar em prática o conhecimento já adquirido, e também expandi-lo. A prática dos conhecimentos resulta em ma evolução mais rápida, e a inserção em um ambiente com profissionais capacitados não só ajuda nesta evolução, como alavanca também as relações sociais e comunicativas.

Este documento tem o objetivo de relatar a experiência em um estágio supervisionado na área de análise e desenvolvimento de sistemas, durante o período de 02/07/2020 a 02/03/2021. O estagiário atuou em uma equipe onde foi responsável pelo desenvolvimento de interfaces para o usuário, API, Modelagem e manipulação de banco de dados e utilização do GeoServer para manipulação de geometrias espaciais, além de participar ativamente das discussões e planejamento dos projetos e atividades, atendendo ao cotidiano da equipe e empresa de acordo com uma metodologia ágil de desenvolvimento, o Scrum.

2 APRESENTAÇÃO DA EMPRESA

Neste capítulo, será apresentada a empresa GT – Global Technologies, onde foi realizado o estágio supervisionado.

2.1 Sobre a empresa

O estágio foi realizado na empresa GT – Global Technologies[1]. A empresa atua no desenvolvimento de sistemas, sendo especializada em sistemas com geoprocessamento para setores de gestão pública, ambiental, de ativos e territorial, com foco no desenvolvimento sustentável. Atualmente, a GT conta com cerca de 50 colaboradores e é sediada em Lavras - MG. A organização utiliza da metodologia Scrum para a gerência de seus projetos. Dentre os principais produtos desenvolvidos pela empresa, estão: Sistema de Cadastro e Regularização Fundiária (SICARF), Sistema de Gestão de Riscos e Desastres do Rio Grande do Sul (SEGIRD) e o Planejamento de Recursos Empresariais (*Enterprise Resource Planning*).

A empresa é dividida em diversas equipes que variam entre 4 a 6 participantes, os membros que compõem essa equipe são: desenvolvedores, que constroem e dão manutenção no software; tester, que testa a aplicação, relatando bugs e confirmando a conclusão de demandas; e um Product Owner, que atua como um representante do cliente, apresentando as demandas a equipe. As equipes utilizam metodologia ágil para o desenvolvimento das aplicações, onde acontecem iterações de planejamento, desenvolvimento, entrega e feedback do cliente.

[1] https://gtglobal.tech/

3 TECNOLOGIAS UTILIZADAS

Este capítulo tem o objetivo de descrever as tecnologias usadas durante o estágio, como, linguagens de programação, *frameworks* e ferramentas para a execução das tarefas diárias. Mesmo que algumas das citadas não terem sido apresentadas durante o curso, estas foram importantes para se ter a base necessária que possibilitou o aprendizado de novas tecnologias.

3.1 HTML e CSS

O HTML[1] (*HyperText Markup Language*) é uma linguagem de marcação utilizada no desenvolvimento de páginas Web. Ele define o significado e a estrutura do conteúdo da Web (BERNES-LEE, 2021). Utilizando um navegador, a linguagem consegue exibir diferentes tipos de conteúdo, como texto, imagens e vídeos. O HTML normalmente não trabalha sozinho na exibição dos conteudos que estão presentes em uma página. O CSS [2] (*Cascading Style Sheets*) é um mecanismo para adicionar estilo a um documento Web, e comumente usado junto com o HTML para dar uma visualização melhor dos elementos criados pelo HTML.

O HTML foi utilizado para desenvolver as páginas das aplicações durante o estágio, assim como a disposição de seus elementos.

3.2 JavaScript

O JavaScript[3] é uma linguagem de script, multiplataforma e orientada a objetos (MDN, 2021). Permite implementar funcionalidades mais complexas em páginas Web, podendo criar interações, efeitos visuais, ordenar tabelas, validar campos de formulários e etc. Na agência, é a linguagem utilizada para dar dinamicidade nas páginas Web, com foco na experiência e interações dos usuários, como efeitos de transições e carregamento de páginas de *sites*, filtros de buscas, menus interativos, entre outras funcionalidades dinâmicas.

[1] https://www.w3.org/html
[2] https://www.w3.org/Style/CSS
[3] https://www.javascript.com/

3.3 Vue JS

O Vue JS[4] é um *framework* progressivo para a construção de interfaces de usuário (VUE, 2021). Aplicações que utilizam Vue são constituídas de componentes criados com a sintaxe HTML, CSS e Javascript em um único arquivo .vue, como apresentado na Figura 3.1. Isso facilita o isolamento e a manutenção de funcionalidades. Os componentes definem as *views*, que são os elementos a serem dispostos para o usuário e modificados de acordo com a lógica desenvolvida. Os componentes podem receber propriedades de componentes "pai" e também conter componentes filhos, facilitando o desenvolvimento com o uso de modularização, além disso o vue conta com ferramentas como um *Router* para gerenciar as rotas da aplicação e com funções próprias que remetem ao seu ciclo de vida como uma função quando o componente é criado ou destruído.

O estagiário utilizou o Vue JS para a criação de páginas dinâmicas nas aplicações, além de manejamento de funções próprias da tecnologia.

3.4 Java

Java[5] é tanto uma linguagem quanto uma plataforma (DEVMEDIA,). A linguagem é baseada em classes e orientada a objetos. A implementação da API do projeto citado neste relatório foi desenvolvida em Java, para fazer a modelagem das classes necessárias, controle dos dados recebidos do frontend, conexão com o banco de dados e execução de serviços.

3.5 Spring

Spring Framework[6] é um *framework* de código aberto desenvolvido para a plataforma Java baseado nos padrões de projetos, inversão de controle e injeção de dependência. É constituído por módulos diversos e completos para aumentar o desempenho da aplicação.

[4] https://vuejs.org/
[5] https://www.java.com/pt-BR/
[6] https://spring.io/projects/spring-framework

Figura 3.1 – Exemplo de arquivo .vue

```
<template lang="pug">
  <div>
    <span class="spanUsuario"> {{usuario}} </span>
    <Map>
  </div>
</template>

<script>
import Map from '@/components/Map';

export default {
  data() {
    return {
      usuario: null,
    }
  },
  components: { Map },

  mounted() {
    this.setUsuario();
  },

  methods: {
    setUsuario(){
      this.usuario = 'Witor';
    }
  }
}
</script>

<style >
.spanUsuario {
  font-size: 12px;
}
</style>
```

Fonte: Autor

Nas API's trabalhadas no estágio foi utilizado o Spring Boot, que é um *framework* que torna fácil a criação de aplicações, possibilitando a execução imediata e o gerenciamento das dependências do projeto a partir do princípio da Convenção sobre Configuração, que busca reduzir o número de decisões a serem tomadas pelo programador no momento de configurar as diferentes áreas de uma aplicação.

Na Figura 3.2 é apresentado um exemplo de um código utilizando o *framework* spring.

Figura 3.2 – Exemplo spring

```
@RestController
@RequestMapping("/controller")
public class Controller {

    private final Service _service;

    public Controller(
            Service service,
    ) {
        this._service = service;
    }
    @GetMapping(path = "/findByid", produces = "application/json")
    public Json findByid(Int id) {
        return renderJSON(_service.findByid(id));
    }

}
```

Fonte: Autor

3.6 React

O React[7] é uma biblioteca Javascript para construção de interfaces de usuário (REACT, 2021). Este, possui muitas semelhanças com o concorrente Vue, como: Constituído por componentes, sintaxe em HTML, CSS e Javascript em único arquivo, funções de ciclo de vida, herança de componentes permitindo transição de dados entre eles. Um exemplo de um arquivo em React é apresentado na Figura 3.3.

O estagiário utilizou o React para a criação de páginas dinâmicas nas aplicações, além de manejamento de funções próprias da tecnologia.

3.7 Victory

O Victory[8] é um conjunto de componentes modulares de gráficos para React e React Native (FORMIDABLE, 2021). Com os objetivos de gerar gráficos e animações para demonstrar dados de maneira diferente e agradável, é extremamente personalizável a cada utilização.

[7] https://pt-br.reactjs.org/
[8] https://formidable.com/open-source/victory/

Figura 3.3 – Exemplo da estrutura do React

```
function ProdutosAgricolasList() {
  const dispatch = useDispatch();
  const [produtos] = useState([]);

  useEffect(() => {
    dispatch(headerActions.setHeaderImage(backgroundImage));
    dispatch(
      headerActions.setHeaderTitle({
        title: 'Produtos',
        boldTitle: 'Agrícolas',
        color: '#000',
        hasShadow: false,
        logoPreta: true,
        infoFoto: 'Foto: Tony Oliveira / CNA',
      }),
    );
  }, [dispatch]);

  const onClick = (produto) => {
    dispatch(filtroProdutoActions.setProdutoImage(produto.img));
    dispatch(headerActions.setShowStaticInfo(false));
  };

  return (
    <Container>
      {produtos.map((item) => (
        <CardProdutoAgricola
          key={item.produto}
          showForward
          isHover
          onClick={onClick}
          item={item}
          img={item.produto.toLowerCase()}
          itemData={item}
          showRedirect
        />
      ))}
    </Container>
  );
}

export default ProdutosAgricolasList;
```

Fonte: Autor

O estagiário utilizou o Victory para a criação de gráficos nas páginas desenvolvidas. Estes com o objetivo de representar visualmente dados e informações numéricas.

3.8 PostgreSQL

O PostgreSQL [9] é um poderoso sistema de banco de dados relacional de objeto, possui código aberto e, utiliza e estende a linguagem *Structured Query Language* (SQL), combinada com muitos recursos que armazenam e escalam com segurança as cargas de trabalho de dados mais complicadas (POSTGRESQL, 2021). Combinado com o SQL para executar comandos em bancos de dados relacionais, é principalmente usada para ler, inserir, editar e deletar registros em banco de dados.

Os bancos de dados PostgreSQL são necessários nas aplicações desenvolvidas, para persistir e alterar os dados utilizados e disponibilizados nas aplicações.

3.9 GeoServer

O GeoServer[10] é um servidor baseado em Java que permite aos usuários visualizar e editar dados geoespaciais (GEOSERVER, 2021). O Geoserver utiliza padrões abertos estabelecidos pelo *Open Geospatial Consortiume* (OGC), o que facilita muito a criação de mapas e a disposição dos dados.

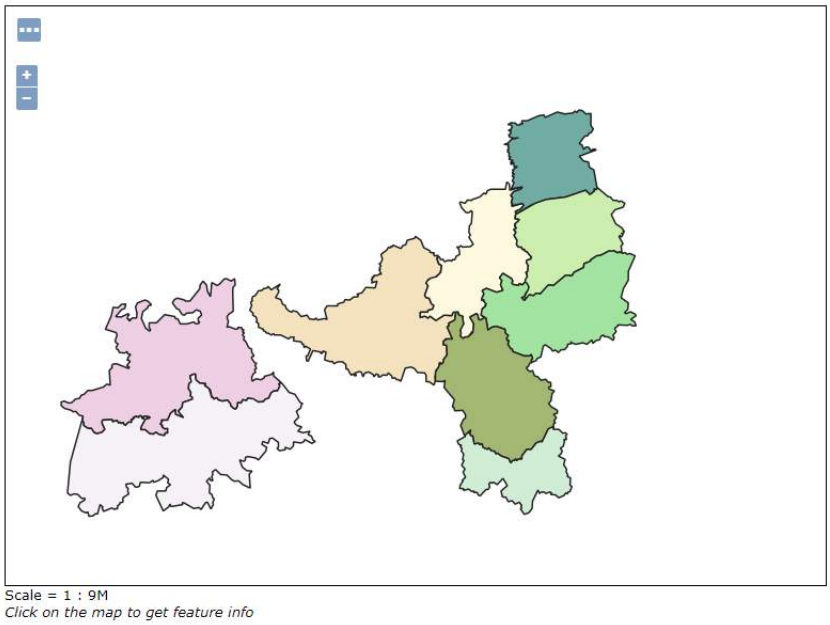
O estagiário utilizou o GeoServer para a manipulação de geometrias espaciais, bem como a visualização das mesmas. Na figura 3.4 é apresentada a visualização de camadas no GeoServer.

3.10 GitLab

O GitLab[11] é um gerenciador de repositórios baseado em Git, que permite o armazenamento de código em repositórios próprios (GITLAB, 2021). Possui diversas ferramentas para ajudar na metodologia do desenvolvimento, como: board para atividades, cadastro de solicitações de *merge* no código e controle de versões.

[9] https://www.postgresql.org/
[10] http://geoserver.org/
[11] https://about.gitlab.com/

Figura 3.4 – Exemplo de visualização de camadas no GeoServer



Fonte: Autor

O GitLab foi utilizado durante o desenvolvimento das aplicações para controle de código, através do controle de versões e *merge*.

3.11 Scrum

O Scrum [12] descreve um conjunto de reuniões, ferramentas e cargos que atuam juntos para ajudar as equipes a organizarem e gerenciarem o trabalho (DRUMOND, 2021). O funcionamento da metodologia ágil Scrum é feita em etapas e ciclos, para que dessa forma haja controle das atividades a serem desempenhadas e o produto final seja concluído com êxito. As etapas que compõem o Scrum são: *planning*, o trabalho que será realizado ao longo do *sprint* atual é planejado durante essa reunião por toda a equipe de desenvolvimento; *sprint*, é o período real em que a equipe do Scrum trabalha em conjunto para concluir um incremento; reunião diária, é uma reunião diária bem rápida que ocorre na mesma hora e serve para todos os integrantes da equipe estejam

[12] https://www.atlassian.com/br/agile/scrum

atualizados com as mesmas informações e alinhados; análise de *sprint*, uma sessão informal a fim de ver uma demonstração do incremento ou inspecioná-lo; retrospectiva de *sprint*, é o momento em que a equipe se reúne para documentar e discutir as qualidades e defeitos em uma *sprint*.

A equipe do estagiário seguiu a metodologia Scrum para o desenvolvimento de todas as aplicações, sendo feitas todas as reuniões citadas.

3.12 API REST

API REST, também chamada de API RESTful, é uma interface de programação de aplicações (API ou API Web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços Web RESTful. Quando um cliente faz uma solicitação usando uma API RESTful, essa API transfere uma representação do estado do recurso ao solicitante ou *endpoint*. Essa informação (ou representação) é entregue via HTTP utilizando um dos vários formatos possíveis (REDHAT, 2020).

O estagiário utilizou API REST para comunicação do *front-end* com o *back-end* nas requisições necessárias nas aplicações.

4 ATIVIDADES INICIAIS E COMUNS A TODOS OS PROJETOS

Este capítulo tem o objetivo de apresentar as atividades que foram realizadas em todos os projetos durante o período de estágio na GT - Global Technologies.

O estagiário trabalhou como desenvolvedor durante todo o período do estágio. As atividades relatadas neste capítulo não configuram a tarefa principal de criação de código do desenvolvedor, mas precisam ser realizadas quando se inicia um projeto ou precisam ser repetidas durante o cotidiano de um projeto.

As atividades neste capítulo relatadas foram realizadas pelo estagiário, antes ou durante sua atuação no projeto principal, relatado no capítulo 5.

4.1 Integração ao local de estágio

O estágio teve início no dia 02/07/2020 na GT. A primeira semana foi focada em informar o estagiário sobre a empresa e seus processos, além do treinamento nas tecnologias que viriam a ser utilizadas no projeto que o estagiário seria inserido. Após fazer cursos disponibilizados pela empresa, o estagiário foi integrado a uma equipe com 4 integrantes. A equipe já possuía algumas demandas e projetos, seu cotidiano era bem solidificado e o estagiário passou a frequentar suas reuniões diárias para discutir o que foi feito desde a última reunião e o que seria feito até a próxima.

4.2 Ritos do Scrum

Após a primeira semana, a equipe iniciou seu ciclo da metodologia Scrum. A *planning* foi feita, onde foi apresentado pela equipe a próxima demanda a ser executada. A mesma, discutiu e avaliou o esforço gasto em cada uma das atividades propostas foi utilizado o *planning poker*, onde cada membro da equipe escolhe uma carta entre os números primos de 1 a 13 para avaliar uma atividade. Se os números escolhidos pelos membros forem distantes, é feita uma discussão e é iniciada uma nova rodada de avaliação para a atividade em questão. Feita a avaliação de todas as atividades, a equipe analisa o peso das atividades e fecham o escopo definido para a próxima *sprint*,

que é um período de 10 dias corridos. Após o fim da *sprint*, a equipe executa uma revisão de todas as atividades feitas e uma retrospectiva, pontuando oque foi considerado positivo e negativo pelos integrantes nesse período. Por fim, há novamente uma *planning*.

4.3 Revisão de código

A equipe atualiza o repositório através de solicitações chamadas de *Merge Requests*, onde o ramo "filho" utilizado para executar a atividade é comparado e mesclado no ramo "pai", para realizar o controle de versões nestes ramos dados como estáveis. A solicitação é aberta por um desenvolvedor e um outro analisa as mudanças a serem mescladas ao código definitivo do projeto, podendo apontar erros e melhorias, visando a troca de conhecimento entre os desenvolvedores e a qualidade do código.

4.4 Controle de versões

O estagiário executou o controle de versões do projeto como atividades das *Sprints*, o controle foi feito através do Gitlab. Cada projeto do Observatório possui um repositório e três ambientes: Teste, onde o tester da equipe validava as atividades; Homologação, onde o cliente tem um acesso privado às últimas atividades realizadas e validadas; Produção, onde a aplicação é lançada e funciona em definitivo para seu público. Quando é necessário atualizar versões em cada ambiente, é gerada uma versão de um ramo específico do repositório, um acesso a máquina responsável pela hospedagem da aplicação em seu ambiente específico. Para os ambientes de homologação e produção, a cada versão gerada e atualizada é separada uma versão de *backup* e documentação, para garantir versões estáveis dos projetos.

4.5 Controle da evolução do banco de dados

Diversas atividades necessitavam de alterações no banco de dados, todos os códigos em SQL para a evolução do banco de dados foram colocados em pastas específicas, com nomeação

padronizada. Como cada ambiente possui um objetivo de utilização, cada um possui seu próprio banco de dados e é necessário ter o controle da evolução correta e condizente com o código em cada ambiente. O controle foi feito através de planilhas, como apresentado na Figura 4.1.

Figura 4.1 – Exemplo de tabela de evolução de banco

Nome	Criador	Data	Teste	Homologação	Produção
01.sql	Desenvolvedor 1	25/04/2019	X	X	X
02.sql	Desenvolvedor 2	30/05/2019	X	X	X
03.sql	Desenvolvedor 2	30/06/2019	X	X	
04.sql	Desenvolvedor 1	03/03/2020	X		
05.sql	Desenvolvedor 3	30/12/2020			

Fonte: Autor

4.6 Configuração de ambiente e desenvolvimento das atividades

Para começar a trabalhar em qualquer projeto, é necessário fazer a configuração do ambiente, que consiste na instalação de versões específicas de tecnologias e *softwares* que são necessárias para o projeto funcionar adequadamente na máquina local.

Durante o período na empresa, o estagiário participou de vários projetos. Os projetos se diferenciavam em tecnologias e bibliotecas utilizadas constantemente, mas o estagiário sempre executou as principais funções no desenvolvimento de aplicações Web. Independente do projeto, as atividades eram: o desenvolvimento de interfaces gráficas (*front-end*), o desenvolvimento de API REST (*back-end*), a modelagem e manipulação do banco de dados, e o *deploy* nos ambientes de teste, homologação e produção.

5 OBSERVATÓRIO DA AGROPECUÁRIA BRASILEIRA

O Observatório da Agropecuária Brasileira trata-se de uma plataforma que integra diversas bases de dados e disponibiliza para o usuário, através de painéis, os dados do setor de agropecuária brasileira. A plataforma contém informações acerca da safra agrícola, da previsão climática, do crédito rural, do setor pesqueiro e de imagens georreferenciadas da área rural brasileira. O projeto, desde seu lançamento, proporciona os benefícios de transparência dos dados do setor de agropecuária brasileira e o auxílio na tomada de decisões.

Este projeto teve atuação de várias instituições, mais recentemente, da empresa GT. Este capítulo visa relatar as atividades que o estagiário executou neste projeto e o que aprendeu durante o período de estágio na empresa.

O projeto do observatório aparenta ser um grande e único projeto, mas irei abordar cada uma das divisões, que, apesar de se completarem, têm objetivos e tecnologias diferentes.

5.1 Portal

O portal funciona como uma página inicial. É um local de acesso rápido e prático de informações sobre o Observatório da Agropecuária Brasileira, com intuito de divulgar, explicar e informar sobre os objetivos, impactos e benefícios do programa. Ele dá acesso aos painéis temáticos ZARC, Crédito Rural Público, Produtos Agrícolas, PronaSolos e a Plataforma Geoespacial, bem como divulgar os parceiros envolvidos no projeto, as coordenadorias e as secretarias. É um ambiente de interação e informação, no qual também se encontra uma biblioteca relacionada ao tema da agropecuária e informações de contato. A Figura 5.1 é apresentado o *banner* principal do *site*.

5.1.1 Tecnologias e atividades realizadas

O portal do observatório é um projeto que usa poucas tecnologias. Como sua função principal é divulgar e informar sobre o projeto, este foi feito usando HTML, CSS e Javascript. Devido a

Figura 5.1 – Página inicial do *site* do observatório

Fonte: Site Observatório

isso, as ações mais complexas do portal, como o cadastro do usuário para recebimento de e-mails, são o envio de requisições para a API de outro projeto que compõe o observatório. As demais ações são formadas por códigos CSS e HTML, fazendo o redirecionamento.

As atividades executadas neste projeto consistiam em correções de texto, imagens e ícones em sua maioria. Ao criar uma nova seção nos demais projetos, esta mesma precisava ser referenciada no portal, para que o usuário pudesse ter um caminho a ela, além de uma breve explicação, como todas as opções apresentadas na página de painéis temáticos, apresentado na Figura 5.2.

Figura 5.2 – Página de painéis temáticos do *site* do observatório

Fonte: Site Observatório

As correções de imagem e ícone, por sua vez, consistem em definir o local onde eram referenciadas as imagens e os ícones que deveriam ser retirados e substituir com os novos, que possuem o mesmo nome, mantendo a importação realizada ao disponibilizar os mesmos na página.

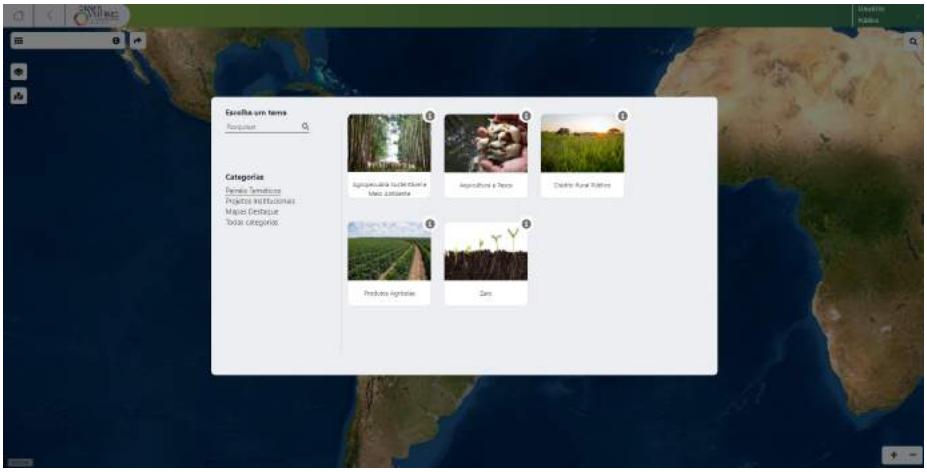
5.2 Plataforma Geoespacial

A plataforma geoespacial do Observatório tem o objetivo de integrar os dados da agropecuária brasileira com as informações territoriais, disponibilizando um mapa nacional que pode variar da vista por satélite e *Open Street Map* (Mapa Aberto de Ruas), com diversas camadas a serem aplicadas, a fim de uma visualização das informações em nível espacial. Além da visualização de camadas, a plataforma também disponibiliza relatórios gráficos, cruzamento de dados abertos para o público, opções de camadas e dados de acesso a usuários credenciados.

A plataforma geoespacial possui duas categorias que, por sua vez, contêm temas selecionáveis. Está disponível ao usuário navegar e visualizar as camadas, filtros e relatórios próprios de cada tema, além de informações referentes ao objetivo do tema e de suas fontes de dados para as informações. A categoria de Painéis Temáticos possui os seguintes temas lançados para o usuário: Agropecuária Sustentável e Meio Ambiente, Aquicultura e Pesca, Crédito Rural Público, Produtos Agrícolas e ZARC. A Categoria de Projetos Institucionais contém dois temas lançados ao público: AgroNordeste e Vertentes. Na Figura 5.3 é apresentada a página inicial da Plataforma Geoespacial e a categoria de Painéis Temáticos.

Em quase todos os temas disponíveis na plataforma, existe um menu de camadas que, por sua vez, contém uma lista de camadas agrupadas por temas. Estas camadas podem ser visualizadas no mapa simultaneamente, e cada uma possui um nome, legenda, informações sobre suas fontes, data de atualização e um filtro. O filtro normalmente apresenta as opções de filtrar por estratificação (municipal ou estadual) e por data. Para todos os temas disponíveis na plataforma, existe o menu de Limites e Camadas Base, que possui a mesma arquitetura de pastas que o menu de camadas, dessa vez visando camadas essenciais e que se repetem em todos os temas, como Terras indígenas,

Figura 5.3 – Página inicial da Plataforma Geoespacial do Observatório



Fonte: Site Observatório

Solos, Assentamentos e Unidades de conservação. A Figura 5.4 ilustra o menu de camadas e uma camada ativa no mapa.

Figura 5.4 – Menu de camadas da Plataforma Geoespacial do Observatório

Fonte: Site Observatório

O menu de relatórios, presente na maioria dos temas, se divide em duas possíveis opções: dados cruzados e dados quantitativos. A maioria dos temas possui o tipo dados cruzados, neste caso o menu de relatórios lista os cruzamentos disponíveis para serem visualizados no tema atual. Ao selecionar um cruzamento, o mesmo apresenta a soma dos registros que se cruzam em relação

ao espaço geográfico que ocupam, como por exemplo: os dados de unidades de conservação que possuem uma área em hectares (ha) podem ser cruzados com os hectares ocupados pelos municípios do projeto AgroNordeste.

Figura 5.5 – Relatório gráfico da Plataforma Geoespacial do Observatório

Fonte: Site Observatório

Os relatórios de camadas cruzadas apresentam a opção de visualizar no mapa os locais onde ocorrem os cruzamento das áreas

Quanto aos relatórios com dados quantitativos, quando a opção é implementada pelo tema atual, apresenta uma tela inicial com campos para que seja feita a filtragem dos dados, como por exemplo no tema de Crédito Rural Público, que apresenta filtros para estado, município e tecnologias, tal como mostrado na Figura 5.6.

Após selecionados, os filtros é mostrado na tela o resultado dos dados filtrados, então, é possível selecionar os campos que aparecem no resultado, baixar todas as informações em um arquivo .csv, excluir filtros e adicionar novos com outros parâmetros, como mostrado na Figura 5.7.

Todas as camadas inseridas no mapa, sejam elas por quaisquer dos menus descritos, tem uma funcionalidade ao clique do mouse do usuário, aparecendo um *pop-up* que mostra as informações específicas para a área clicada, como apresentado na Figura 5.4.

Figura 5.6 – Filtro do relatório quantitativo da Plataforma Geoespacial do Observatório

Fonte: Site Observatório

Figura 5.7 – Relatório quantitativo da Plataforma Geoespacial do Observatório

Fonte: Site Observatório

5.2.1 Tecnologias e atividades realizadas

A aplicação do mapa geo utiliza em seu *front-end* as tecnologias: HTML, CSS, Javascript e Vue. O *back-end*, que trata-se de uma *API REST* utiliza: Java e Spring Boot. Existe apenas um banco de dados e Sistema de gerenciamento de banco de dados (SGBD) para todas as aplicações do observatório, que utilizam PostgreSQL. O *back-end* além de fazer requisições ao SGBD, também fazia requisições ao Geoserver da aplicação, que guardava os dados de cada camada, assim como suas estilizações, legendas e geometrias, que eram necessárias para a disponibilização no mapa.

A equipe, a qual o estagiário fez parte, executou diversas atividades e correções solicitadas no projeto. Destaca-se a criação de um novo tema, o Vertentes, que se encontra na categoria de Projetos Institucionais.

Assim como a maioria das demandas executadas no projeto do Mapa-geo, a criação de um novo tema é quase inteiramente desenvolvido na linguagem SQL. Um arquivo em SQL é criado em uma pasta específica. Neste arquivo, é escrito o código em sql para as operações no banco de dados, que vão resultar na criação de um novo tema para a aplicação, sendo elas: Insert. - inserção de um nova linha em uma tabela; Update. - Atualização de uma linha; Delete. - remoção uma linha de uma tabela; Select. - busca de uma ou mais linhas de uma tabela.

Para a inicialização das atividades da equipe, o cliente precisa fornecer, previamente, os dados referentes ao projeto Vertentes para que o Administrador de Banco de Dados (DBA) possa inserir os dados agrupados da maneira correta em uma tabela no banco de dados do observatório em todos os ambientes.

Com o banco de dados devidamente populado, o próximo passo é o cadastro de cada nova camada no Geoserver, além de sua estilização. A manipulação do Geoserver pode ser feita através de uma aplicação Web comum, com uma aba de listar as camadas e adicioná-las. Adicionar uma camada é basicamente apontar para um banco de dados e fazer uma pesquisa em SQL para gerar uma nova tabela com os dados que serão utilizados na aplicação do mapa, sendo que o que torna a camada visível no mapa é a coluna de geometria, cujo o mapa espera e converte para a camada visível.

No Geoserver existe um menu para criação de estilos, onde as características da camada, como suas cores, formato e legenda podem ser customizados. O estilo é escrito em um arquivo XML, e é necessário seguir a estrutura de atributos do objeto esperado pelo geoserver, para que o estilo seja corretamente interpretado. Em resumo, no XML são estabelecidos objetos e seus atributos, onde cada objeto acessa a camada e com base nos atributos da mesma, adiciona uma estilização, por exemplo: Olhando para uma camada, temos uma tabela no Geoserver, onde cada linha possui uma geometria, ou seja, uma lista de formas geométricas que irão ser exibidas no

mapa. Cada linha desta tabela tem outros campos que podem ser usados para definir qual cor será visualizada e qual é o texto mostrado na legenda para essa cor. Se temos uma tabela agrupada por estados, cada linha possui um campo, que identifica qual o é o estado a qual seus demais campos se referem. Um desses campos é utilizado para decidir a cor que o estado irá assumir no mapa, um estado pode ter a cor azul por ter menos de 100 contratos cadastrados, e outro estado que irá assumir a cor vermelha por ter de 101 a 200 contratos. A Figura 5.8 apresenta um exemplo do arquivo em XML.

O cadastro de camadas e estilos do geoserver tem diversas opções e funcionalidades que em si geram um documento.

A adição de um novo tema no projeto é feito totalmente por inserções em tabelas no banco de dados. As funções que buscam e listam os temas e todos seus dados e camadas não apresentam problemas para disponibilizar de acordo com os parâmetros.

Foram criados códigos em SQL para inserção das entidades necessárias, sendo elas: tema, contendo título, categoria a qual o tema pertencia, foto e todas informações referentes ao projeto; grupos de camadas, que são as pastas com listas de camadas presentes nos menus acessados pelo usuário, indicando seus nomes; camadas, que compunham o grupo do Projeto Vertentes (a estrutura básica da camada se trata não só do nome e informações que aparecem na interface, mas também da referência do nome da camada condizente no geoserver); dicionário, esta entidade se trata do *pop-up* aparente quando se clica com o mouse no mapa; atributos do dicionário, esta tabela se trata de cada linha aparente no pop-up, contendo principalmente uma coluna para a adição de um código HTML, que será literalmente o que a interface do *front-end* irá renderizar no *pop-up*; tabelas de relacionamento, que relacionam as categorias com o tema, os grupos e subgrupos de camadas ao tema, e as camadas ao seus respectivos grupos.

As tabelas de relacionamento são pontos vitais que ajudam no reaproveitamento de trabalho. Como muitos temas compartilham diversos elementos, como o grupo de camadas complementares e o menu de camadas base, só é necessário fazer a relação registrando que os grupos já existentes também serão referenciados por um outro objeto. Neste caso, há uma relação com a parte teórica

Figura 5.8 – XML exemplo de estilo de camada

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <NamedLayer>
    <Name>estilo_testeo</Name>
    <UserStyle>
      <Name>estilo_testeo</Name>
      <Title>TESTE</Title>
      <Abstract>teste</Abstract>
      <FeatureTypeStyle>
        <Transformation>
          <ogc:Function name="vec:PointStacker">
            <ogc:Function name="parameter">
              <ogc:Literal>data</ogc:Literal>
            </ogc:Function>
            <ogc:Function name="parameter">
              <ogc:Literal>cellSize</ogc:Literal>
              <ogc:Literal>30</ogc:Literal>
            </ogc:Function>
            <ogc:Function name="parameter">
              <ogc:Literal>outputWidth</ogc:Literal>
              <ogc:Function name="env">
                <ogc:Literal>wms_width</ogc:Literal>
              </ogc:Function>
            </ogc:Function>
          </ogc:Function>
        </Transformation>
        <Rule>
          <Name>rule1</Name>
          <Title>Soja - Produção (t)</Title>
          <ogc:Filter>
            <ogc:PropertyIsLessThanOrEqualTo>
              <ogc:PropertyName>count</ogc:PropertyName>
              <ogc:Literal>1</ogc:Literal>
            </ogc:PropertyIsLessThanOrEqualTo>
          </ogc:Filter>
          <PointSymbolizer>
            <Graphic>
              <Mark>
                <WellKnownName>circle</WellKnownName>
                 <Fill>
                  <CssParameter name="fill">#f7fcc0</CssParameter>
                  <CssParameter name="fill-opacity">0.8</CssParameter>
                </Fill>
                <Stroke>
                  <CssParameter name="stroke">#000000</CssParameter>
                  <CssParameter name="stroke-width">0.7</CssParameter>
                </Stroke>
              </Mark>
              <Size>8</Size>
            </Graphic>
          </PointSymbolizer>
        </Rule>

        <Rule>
          <Title>2 até 50</Title>
          <ogc:Filter>
            <ogc:PropertyIsBetween>
              <ogc:PropertyName>count</ogc:PropertyName>
              <ogc:LowerBoundary>
                <ogc:Literal>2</ogc:Literal>
              </ogc:LowerBoundary>
              <ogc:UpperBoundary>
                <ogc:Literal>50</ogc:Literal>
              </ogc:UpperBoundary>
            </ogc:PropertyIsBetween>
          </ogc:Filter>
          <PointSymbolizer>
            <Graphic>
              <Mark>
                <WellKnownName>circle</WellKnownName>
```

Fonte: Autor

vista durante o curso, onde entidades possuem uma relação de muitos para muitos mantendo a persistência e consistência dos dados.

Toda a parte do menu de relatórios, assim como as camadas, funcionam como camadas no Geoserver. Porém, para os relatórios que contêm gráficos, existem colunas na tabela de camada no banco de dados, onde pode ser inserido o tipo do gráfico e os campos que serão utilizados no componente no *front-end*. O mesmo processo é feito para os relatórios que não apresentam visualização de camada no mapa, as camadas possuem os dados quantitativos, mas não possuem as geometrias.

Com tudo inserido no banco de dados, o *front-end* faz a requisição para o *back-end*, que, por sua vez, faz a requisição ao SGBD para as informações básicas, e ao geoserver para as informações realmente interessantes ao usuário. Como todos os temas, grupos e camadas, seguem a mesma estrutura, a estrutura do *front-end* feita de maneira genérica, consegue tratar todas da mesma maneira.

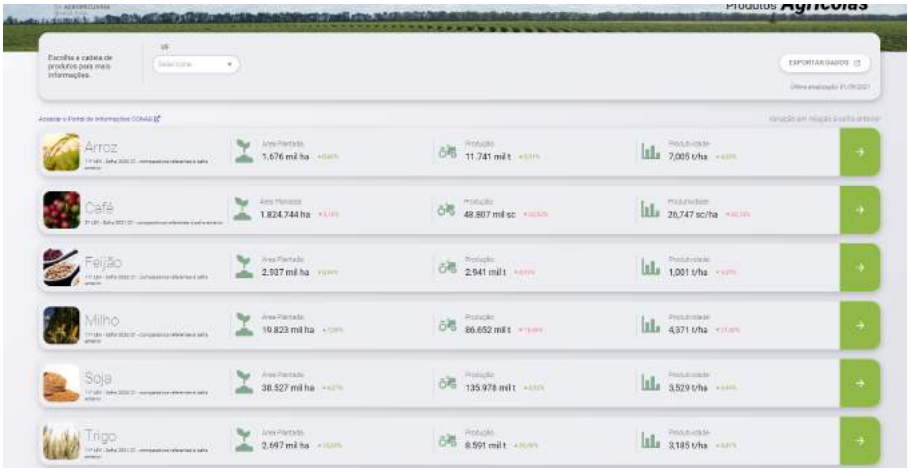
5.3 Plataforma Estatística

A plataforma estatística do Observatório, tem o objetivo de disponibilizar informações sobre a Agropecuária brasileira em formato de dados quantitativos, em tabelas e gráficos. A plataforma conta com filtro por período e com estratificação com foco no cenário nacional, estadual e municipal. A plataforma é dividida em 4 painéis temáticos ao usuário: Credito Rural, que oferece dados referentes ao Crédito Rural e do Proagro; Produtos Agrícolas, que centraliza informações sobre o cultivo, mercado e crédito para cada uma das principais culturas agrícolas do Brasil; Pronasolos, o Programa Nacional de Solos do Brasil, que apresenta de forma interativa e espacial os dados referentes a solos de todo o Brasil; ZARC, painel de Zoneamento Agrícola de Risco Climático, que tem o intuito de apresentar informações das portarias de zoneamento para as culturas, grupos e tipos de solo.

Apesar de cada painel temático não ser interligado nas interfaces, todos fazem parte do mesmo projeto. Este documento foi escrito com o foco no painel temático de Produtos Agrícolas (onde o estagiário executou refatorações e mudanças) e no painel do Crédito Rural (onde foram executadas trabalhos de criação de página Web).

O Painel de Produtos Agrícolas, na plataforma estatística, se restringe às culturas agrícolas: Arroz, Café , Feijão, Milho, Soja e Trigo. A página inicial do painel, como apresentado na Figura 5.9, exibe as principais informações sobre cada cultura agrícola. Nesta página, há a opção de exportar tais informações em uma planilha (arquivo .csv), há um filtro para filtrar as informações por Unidades federativas e, é possível clicar em cada uma de suas culturas para acessar a página de detalhes sobre a mesma.

Figura 5.9 – Página inicial do painel temático Produtos Agrícolas da plataforma estatística

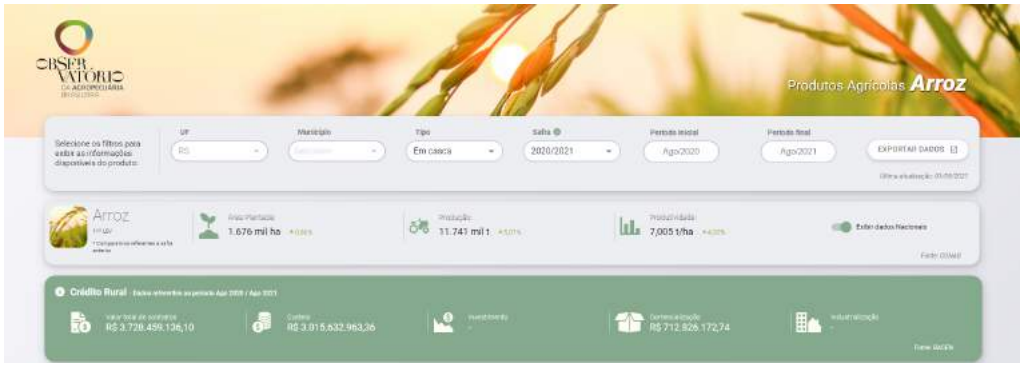


Fonte: Site Observatório

A página de detalhes de cada cultura possui os mesmos elementos. Apenas os dados são variáveis de acordo com a cultura selecionada. A página de detalhes é composta por um filtro, como apresentado na Figura 5.10, que possui as opções de período, Unidade Federativa, Município, Tipo do produto e Safra agrícola. Este filtro tem efeito em todos os dados da página, que estão dispostos de maneira quantitativa e gráfica. Estes dados estão divididos em diversos quadros referentes ao mercado, cultivo e crédito do produto, os dados em cada quadro são filtrados pelos filtros selecionadas no filtro geral e também pelos filtros selecionados dentro do próprio quadro, que normalmente são referentes a datas, mas podem ter outros objetivos específicos. Na página, há também um quadro com as últimas notícias sobre a cultura agrícola e um botão para a exportação de todos os dados disponibilizados na tela, no momento da exportação, ou seja, com todos os fil-

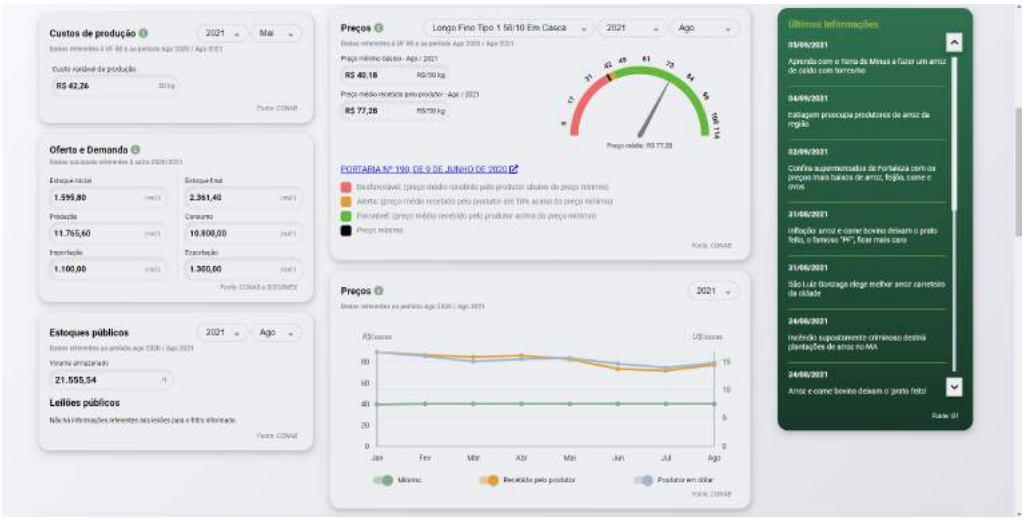
tros aplicados na exportação. Na Figura 5.11 são apresentados alguns dos quadros que compõem a página de detalhes.

Figura 5.10 – Filtro e informações gerais sobre o produto na pagina de detalhes da cultura



Fonte: Site Observatório

Figura 5.11 – Parte da página de detalhes da cultura com os quadros de dados e o quadro de notícias



Fonte: Site Observatório

O Painel de Crédito Rural na Plataforma estatística é formado por uma página, que inicialmente exibe uma tabela com os dados referentes ao Crédito Rural, um botão para a exportação dos dados, um *header* com diversos interruptores e um filtro lateral com as opções: período, Região, Unidade Federativa, opção para agrupamento de todos os dados, entre outros. Na Figura 5.12 é apresentado os elementos iniciais desta página.

Figura 5.12 – Página inicial do Crédito Rural

Fonte: Site Observatório

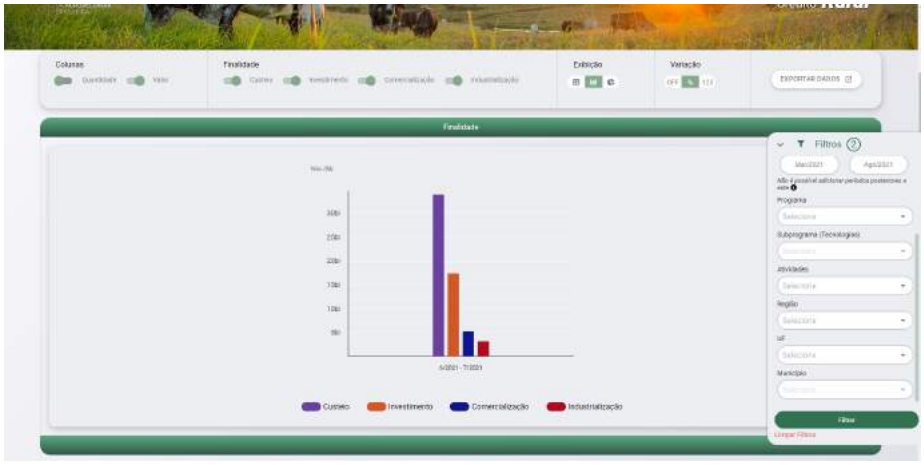
O *header* contém interruptores que controlam as informações apresentadas na página. A primeira divisão contém interruptores referentes à quantidade e valor. As opções de interruptores para a divisão de exibição mudam a forma no qual os dados são apresentados. A exibição varia entre a tabela inicial (ver Figura 5.12), um gráfico de barras agrupados pela Finalidade do crédito. (ver Figura 5.13), e uma exibição com dois gráficos em fatias. (um referente a quantidade e outro ao valor do crédito), (ver Figura 5.14). Existe uma divisão com interruptores referentes ao formato da variação exibida e a maior divisão de opções, referentes à finalidade, que filtra as informações que aparecem em cada exibição. Cada exibição tem suas configurações possíveis de finalidades e alguns interruptores indisponíveis para modificação por padrão.

5.3.1 Tecnologias e atividades realizadas

Nos Projetos da plataforma estatística foram utilizados no *front-end* as tecnologias HTML, CSS, Javascript e React. O *back-end*, que se trata de uma *API REST* utiliza: Java e Spring Boot, o SGBD usado nesta aplicação é o mesmo usado na aplicação geoespacial.

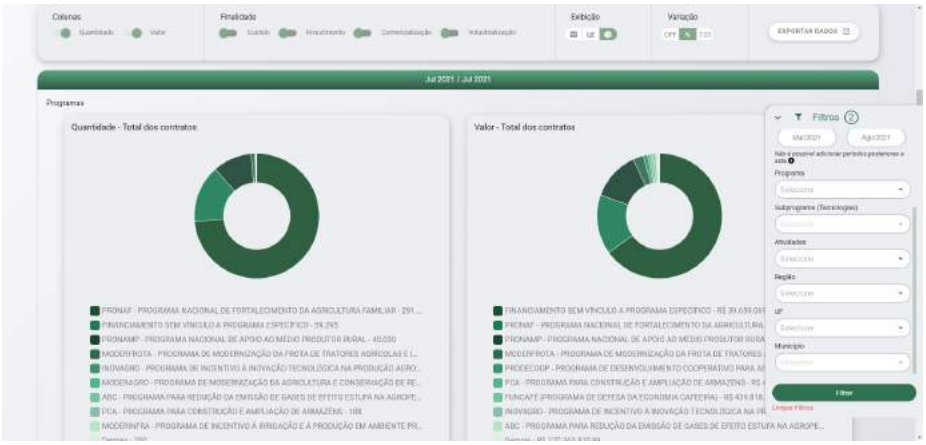
Em equipe, o estagiário realizou diversas manutenções e acréscimos no Painel de Produtos Agrícolas, especialmente na tela de detalhes da cultura. A principal foi a adição dos filtros individuais dos quadros.

Figura 5.13 – Exibição por gráfico de barras



Fonte: Site Observatório

Figura 5.14 – Exibição por gráfico de pizza

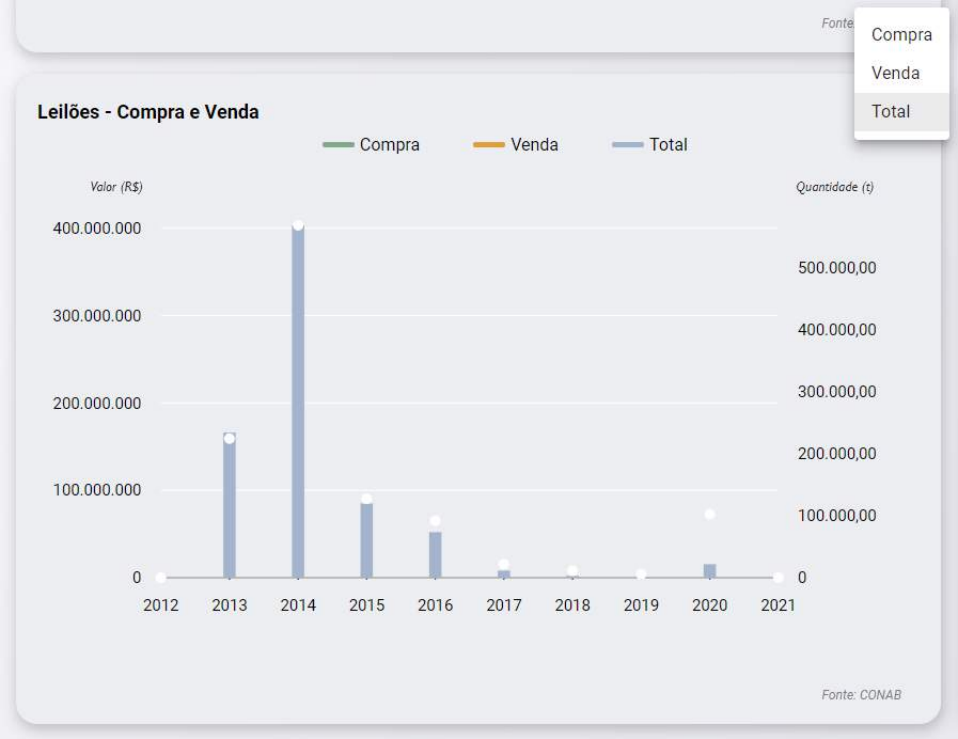


Fonte: Site Observatório

A adição de um novo filtro individual, como apresentado na Figura 5.15, para o quadro de leilões é divida em algumas partes. Primeiramente, é necessário inserir um componente de multi seleção no topo do quadro. Este é um componente básico e já existe no sistema, sua implementação e estilização foi replicada. Existe uma variável declarada no arquivo que guarda o valor selecionado pelo usuário no campo. Uma das ferramentas do React é uma função que, quando variáveis selecionadas são modificadas, é executado um bloco código (ver Figura 5.16). Esta função é acionada

pela mudança da variável, que por sua vez, guarda o valor do filtro, e realiza uma requisição para a API.

Figura 5.15 – Quadro de leilões da página de detalhes da cultura



Fonte: Site Observatório

Figura 5.16 – Exemplo de Effect Hook função disparada por consequência no React

```
useEffect(() => {
  document.title = `Você clicou ${count} vezes`;
}, [count]); // Apenas re-execute o efeito quando o count mudar
```

Fonte: Documentação React

A API recebe a requisição e a redireciona para a função correta. A função recebe como parâmetro a escolha do usuário e faz uma pesquisa no banco de dados pelos dados de leilões, considerando o que foi selecionado no filtro. Para a pesquisa no banco de dados, é utilizada a declaração SQL SELECT, que retorna um conjunto de resultados de uma determinada tabela. Para aplicação do filtro à cláusula WHERE é adicionada, filtrando o retorno com base em uma condição

como a da Figura 5.17. A requisição à API e pesquisa feita ao banco de dados, carregam os parâmetros do filtro superior, que são aplicados na mesma cláusula WHERE.

Figura 5.17 – Uso do cláusula WHERE

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Fonte: Autor

Após o retorno dos dados devidamente filtrados e formatados, o *front-end* atribui os resultados às variáveis que o código de implantação dos gráficos aguarda. No caso do quadro de Leilões, usado como exemplo, é necessário o controle da visibilidade das barras do gráfico de acordo com o que está marcado no campo de filtro de multi seleção. É feita a criação das barras possíveis de cores diferentes, é feita a criação de todas em código e, é feito o controle de suas respectivas visibilidades, usando uma condicional relacionada a variável do campo de filtro. O exemplo de visibilidade condicional é apresentado na Figura 5.18.

Figura 5.18 – Exemplo de uso da visibilidade condicional

```
<div>
  { count && <h1>Messages: {count}</h1>}
</div>
```

Fonte: Autor

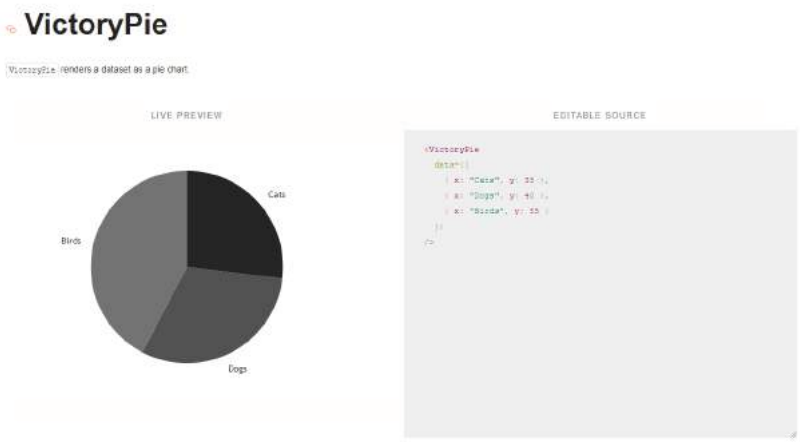
O Painel do Crédito Rural foi criado em equipe. O estagiário foi incumbido da exibição em gráficos do tipo pizza apresentado na Figura 5.14. Só existe um componente que renderiza toda página. Então, foram codificados componentes separadamente e controlada sua renderização condicional usando a variável presente no interruptor de exibição no *header*.

Para implementação dos gráficos, foi utilizada a biblioteca Victory. Como todas as exibições utilizam os mesmos dados e apenas é alterada sua maneira de visualização, não foi preciso modificar o mecanismo de busca para essa atividade em específico, método de busca esse que é muito semelhante em seu funcionamento aos métodos citados no painel de Produtos Agrícolas.

Com os dados já disponíveis no *front-end*, é necessário importar os componentes da biblioteca e posicioná-los no código HTML. No caso do gráfico em formato de pizza é necessária a

importação do componente “VictoryPie”. Todo o controle do componente é feito através de propriedades que o mesmo recebe. Na Figura 5.19 é apresentado um exemplo da documentação da biblioteca em como renderizar o gráfico e passar seus dados para cada uma de suas divisões.

Figura 5.19 – Exemplo de gráfico do victory



Fonte: Documentação Victory

Em posse da lista de dados retornados do banco de dados, é criada uma cópia semelhante com o formato esperado pela biblioteca, para que a mesma possa renderizar corretamente todos os dados originalmente mostrados na Tabela. Este modo de exibição, por ter dois gráficos, só consegue espelhar duas colunas da tabela por vez. Como padrão, foi adotado e implementado que ao entrar na visualização, todos os botões interruptores da divisão de Finalidades iriam ser desligados, e os gráficos renderizam os dados da soma de todas as Finalidades. Foi necessário desenvolver funções, que são ativadas quando os botões interruptores da divisão de finalidades são modificados, para que as variáveis que são passadas por propriedades para os gráficos tomem o valor específico de sua opção. As legendas abaixo dos gráficos são componentes da mesma biblioteca. O VictoryLegend é um componente que recebe uma lista de objetos que possuem uma cor e um texto a ser renderizado.

Para a estilização e animação dos componentes da biblioteca, a mesma tem propriedades específicas onde valores pré definidos são listados na documentação.

5.4 Relação teoria e prática

Durante todo o estágio foi possível colocar em prática diversos conceitos aprendidos durante o curso de Sistemas de Informação. Entretanto, algumas disciplinas se destacam, sendo elas:

- **GCC224 - Introdução aos Algoritmos:** Esta disciplina foi o primeiro contato do aluno com programação. Ao longo dela foi trabalhado desde o básico dos algoritmos até soluções mais complicadas de problemas. O desenvolvimento do pensamento lógico e resolução de problemas foi de muita importância no ambiente de trabalho.

- **GCC178 - Práticas de Programação Orientada a Objetos:** Transmitindo um conhecimento mais próximo do que era exigido e usado no mercado de trabalho. A linguagem Java, juntamente com os ensinamentos sobre o paradigma de programação e padrões de código, foram utilizados no *back-end* de todos os projetos desenvolvidos.

- **GCC214 - Introdução a Sistemas de Banco de Dados:** Em todos os projetos desenvolvidos foi utilizado o conhecimento adquirido em banco de dados e a linguagem SQL.

- **GCC219 - Interação Humano-Computador:** O conhecimento sobre como a experiência do usuário (ao interagir com um sistema) deve influenciar a produção de software é muito útil para um desenvolvedor, possibilitando ao mesmo desenvolver um produto que seja prazeroso e de fácil manuseio para o usuário final.

- **GCC175 - Sistemas Gerenciadores de Banco de Dados:** utilizado para gerenciar todos os bancos de dados trabalhados na empresa. Gerenciando o acesso, persistência, manipulação e organização dos dados.

6 CONSIDERAÇÕES FINAIS

O período de estágio foi essencial para o desenvolvimento profissional e pessoal do estagiário. A convivência com profissionais experientes, oferecendo o suporte para a execução de atividades em projetos complexos, não só consolidou conteúdos do curso como também proporcionou o aprendizado e aprofundamento em outros conteúdos, estes mais ligados ao mercado de trabalho e sua demanda.

A empresa sempre adotou uma postura flexível com o cumprimento do horário de trabalho, o que foi proveitoso ao estagiário, devido à sua rotina ser sempre dividida com a universidade.Em contrapartida ao horário de trabalho flexível, houve uma constante cobrança na entrega de resultados, principalmente, em relação a um projeto grande quanto o Observatório da Agropecuária Brasileira, aqui abordado como o principal projeto participado. Esta cobrança contribuiu para o desenvolvimento da responsabilidade, mesmo com liberdade no planejamento de horas de trabalho, este havia de ser estritamente cumprido. Acredito que a responsabilidade de entrega de trabalho próprio é uma grande qualidade para um profissional.

Vale ressaltar que o período de estágio na empresa supera o registrado no estágio, e que houve reconhecimento por parte da empresa, aumento de responsabilidades e benefícios constantes. A empresa conseguiu dar constante *feedback* sobre meu desempenho e me direcionar ao caminho de aprimoramento da profissão.

Após algum tempo na empresa o estagiário conseguiu identificar alguns pontos a serem melhorados na organização. Mesmo o aumento de responsabilidades sendo acompanhado por uma recompensa financeira, através de aumento salarial, a empresa parece não saber administrar seu contingente de funcionários. A mesma, executa o treinamento e logo após isso (para cumprir prazos) acaba sobrecarregando o funcionário, que agora com experiência profissional acaba sendo procurado por outras organizações, como ocorrido com o próprio estagiário. Por consequência a empresa agrava a falta de profissionais experientes e qualificados para liderar os projetos mais complexos, fazendo com que a sobrecarga do bom funcionário deixe de ser uma escolha e passe a

ser uma consequência, impactando diretamente na quantidade e qualidade de entrega de valor das equipes e da empresa.

Por fim, o período de estágio foi de muita importância, tanto para a área pessoal quanto profissional do aluno, tendo um aprimoramento tanto na área técnica quanto na área social.

REFERÊNCIAS

BERNES-LEE. **Documentação HTML**. 2021. (Acessado em 09/09/2021). Disponível em: <https://devdocs.io/html/>.

DEVMEDIA. **Guia Completo de Java**.

DRUMOND, C. **O que é Scrum**. 2021. (Acessado em 20/09/2021). Disponível em: <https://www.atlassian.com/br/agile/scrum>.

FORMIDABLE. **Sobre o Victory**. 2021. (Acessado em 28/09/2021). Disponível em: <https://formidable.com/open-source/victory/about/>.

GEOSERVER. **Agência especializada em Marketing Digital**. 2021. (Acessado em 09/09/2021). Disponível em: <http://geoserver.org/about/>.

GITLAB. **O que é GitLab**. 2021. (Acessado em 10/09/2021). Disponível em: <https://about.gitlab.com/what-is-gitlab/>.

MDN. **Introdução JavaScriot**. 2021. (Acessado em 10/09/2021). Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction>.

POSTGRESQL. **O que é PostgreSQL**. 2021. (Acessado em 09/09/2021). Disponível em: <https://www.postgresql.org/about/>.

REACT. **Introdução React**. 2021. (Acessado em 28/09/2021). Disponível em: <https://pt-br.reactjs.org/docs/getting-started.html>.

REDHAT. **O que é API REST**. 2020. (Acessado em 20/09/2021). Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>.

VUE. **O que é Vue.Js**. 2021. (Acessado em 28/09/2021). Disponível em: <https://br.vuejs.org/v2/guide/index.html>.