UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL MENDES COSTA

DASHBOARD PARA ANÁLISE DE DADOS CLIMÁTICOS DE BAURU

BAURU
Janeiro/2023

RAFAEL MENDES COSTA

DASHBOARD PARA ANÁLISE DE DADOS CLIMÁTICOS DE BAURU

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. João Pedro Albino

BAURU
Janeiro/2023

C837d

Costa, Rafael Mendes
Dashboard para análise de dados climáticos de Bauru / Rafael Mendes Costa. -- Bauru, 2023
53 f.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: João Pedro Albino
Coorientadora: Simone das Graças Domingues Prado

1. Previsão de temperatura. 2. Dashboard. 3. Análise de dados. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Rafael Mendes Costa

Dashboard para Análise de Dados Climáticos de Bauru

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. João Pedro Albino
Orientador
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Profª. Drª. Simone das Graças Domingues Prado
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof. Dr. Kelton Augusto Pontara da Costa
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Bauru, 17 de Janeiro de 2023.

Dedico esta monografia à minha mãe e avó por todo esforço que me permitiu chegar até aqui.

Agradecimentos

Agradeço primeiramente à minha mãe, que tornou possível o sonho de fazer uma faculdade e me apoio desde o primeiro dia da faculdade e em todos estes anos de vida.

Agradeço também aos meus amigos que estiveram do meu lado me dando suporte e compartilhando dos mementos bons e ruins, em especial ao João Renato Ribeiro Manesco, Gabriel Dadamos Rossetto, Bruna Lika Tamake, Pedro Barros, Rafael Kawagoe Gomes Muller e Lucca Vieira Batistão que estarão sempre em meu coração.

Também deixo meu agradecimento aos professores que tive, em especial ao meu orientador Prof. Assoc. João Pedro Albino, pela confiança em aceitar esse projeto e orientação e a Profª. Simone das Graças Domingues Prado por conduzir todo o processo do trabalho de conclusão de curso.

Por fim, agradeço a todos aqueles que convivi no LEPEC durante todos esses anos, em especial a Maria Cristina de Campos e aos calouros que tive o prazer de ter contato no ano de 2022.

Now this is not the end. It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.
Winston Churchill

Resumo

A energia elétrica pode ser comercializada pelo mercado livre ou pelo mercado regulado pelo governo, onde os consumidores não possuem a liberdade de escolha de fornecedor nem negociação. Já no mercado livre consumidores e fornecedores de energia podem interagir através das comercializadoras para negociar energia elétrica. Isso acontece através da compra de energia do fornecedor pelo comercializador e a venda dessa energia para o consumidor. No Brasil a matriz energética depende de 70% de energia gerada por hidrelétricas e na região de Bauru existe a hidrelétrica de Iacanga. Sistemas para Análise de Dados Climáticos são importantes para a comercialização de energia e *commodities* no mercado. O painel de controle para análise de dados em Bauru permite que o usuário tenha acesso aos dados climáticos da região de Bauru, dando mais segurança nas negociações de energia gerada pela hidrelétrica de Iacanga feitas pelos comercializadores de energia. A aplicação foi desenvolvida usando *React*, *Javascript*, *Python* e *Pandas*. Além disso, o projeto foi desenvolvido seguindo técnicas de boas práticas adotadas por cada um das linguagens e princípiosSOLID.

Palavras-chave: Previsão de temperatura, Previsão de chuva, Análise de dados, *Dashboard*.

Abstract

Electricity can be traded in the free market or in the market regulated by the government, where consumers do not have the freedom of choice of supplier or negotiation. In the free market, consumers and energy suppliers can interact through the trading companies to negotiate electric power. This happens through the purchase of energy from the supplier by the trader and the sold to the consumer. In Brazil the energy matrix depends on 70% of energy generated by hydroelectric power plants and in the Bauru region there is the hydroelectric plant of Iacanga city. Dashboards for Climatic Data Analysis are important for the commercialization of energy and commodities in the market. The dashboard allows the user to have access to the climatic data of the Bauru region, giving more security for energy negotiations generated by the hydroelectric plant of Iacanga. The application was developed using React, Javascript, Python and Pandas. In addition, the project was developed following best practice techniques adopted by each of the languages and SOLID principles.
Keywords: Temperature Forecast, Rain Forecast, Data Analysis, Dashboard.

Lista de figuras

Lista de códigos

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CNN	*Convolutional Neural Networks*
CSV	*Comma-Separated Values*
DOM	*Document Object Model*
FAQ	*Frequently Asked Questions*
GOES	*Geostationary Operational Environmental Satellite*
HTML	*Hyper Text Markup Language*
MetOp	*Meteorological Operational satellite*
PEP	*Python Enhancement Proposals*
NWP	*Numerical Weather Prediction*
PIB	*Produto Interno Bruto*
SOLID	Single Responsibility; Open Closed ; Liskov Substitution ; Interface Segregation ; e Dependence Inversion
UI	*User Interface*
VS Code	*Visual Studio Code*

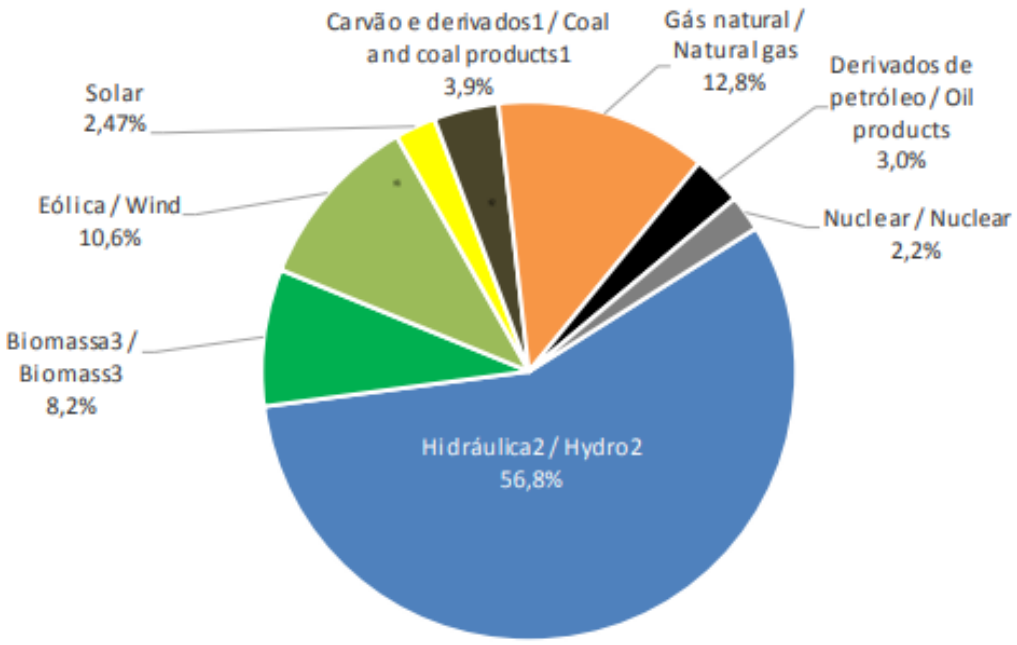
Sumário

1 Introdução

O Brasil é um país de grande potencial econômico, tendo como principais atividades o agronegócio, o mercado de energia elétrica e o mercado de *commodities*(MUNDIAL, 2021). No entanto, essas atividades estão fortemente ligadas ao clima, que pode ser um fator determinante para o sucesso ou fracasso desses setores (KFOURY, 2020).

O mercado de energia elétrica no Brasil é dividido em dois, o regulado pelo Governo Federal (PIRES, 2019) e o mercado livre de energia elétrica, que permite a negociação, por parte de fornecedores e consumidores, por intermédio de uma comercializadora (WALVIS; MARTINS, 2014). Segundo o relatório do Balanço Energético Nacional (Ministério de Minas e Energia, 2022), o Brasil conta com uma matriz energética diversificada, sendo que sua principal fonte é hídrica, com cerca de 56,8% de participação. Esta distribuição é representada na Figura 1 .

Figura 1 – Matriz energética brasileira.



Fonte: Ministério de Minas e Energia (2022).

O agronegócio é um setor de grande importância para a economia brasileira, responsável por cerca de 23% do Produto Interno Bruto (PIB) nacional (Ministério da Agricultura, Pecuária e Abastecimento, 2019), com destaque para a produção de grãos, carnes e açúcar. No entanto, o clima pode ser um fator de grande influência para a produção agrícola, sendo que eventos climáticos extremos, como secas ou chuvas excessivas, podem afetar a qualidade e quantidade dos produtos agrícolas.

O mercado de *commodities* também é afetado pelo clima, uma vez que muitas matérias-primas são sensíveis às condições climáticas. Por exemplo, a produção de café (ESTADãO, 2019) e o preço do petróleo (S.PAULO, 2017) podem ser afetados pelo clima.

Diante disso, torna-se evidente a importância de se ter uma plataforma de acompanhamento climático, que possa fornecer informações precisas e atualizadas sobre as condições climáticas, possibilitando que os diferentes setores da economia possam se preparar e tomar decisões informadas. Sendo assim, por meio da utilização de técnicas de Engenharia de *Software* e Análise de Dados, foi desenvolvida uma plataforma de acompanhamento climático.

1.1 Problema

O mercado de energia elétrica no Brasil é um setor de grande movimentação financeira, com investimentos anuais que superam os R$ 50 bilhões (ELéTRICA, 2020). Além disso, esse campo é responsável por abastecer a maior parte da demanda energética do país, que atingiu a marca de 324 mil GWh em 2020 (ELéTRICA, 2021). No entanto, ele enfrenta diversos problemas, como a dependência de fontes energéticas poluentes, como a térmica, e a falta de investimentos em fontes renováveis, como a solar e eólica. Além disso, a falta de conhecimento sobre as condições climáticas pode ser um problema, uma vez que eventos climáticos extremos, como secas ou chuvas excessivas, podem afetar a produção e distribuição de energia.

O agronegócio também é um setor de grande movimentação financeira no Brasil, gerando um faturamento anual de aproximadamente R$ 600 bilhões (Ministério da Agricultura, Pecuária e Abastecimento, 2019). Além disso, este departamento emprega cerca de 16 milhões de pessoas no país (Ministério da Agricultura, Pecuária e Abastecimento, 2019). No entanto, ele também enfrenta diversos problemas, como a falta de investimentos em tecnologia e a dependência de práticas agrícolas pouco sustentáveis, como o uso excessivo de agrotóxicos. A falta de informações climáticas precisas também pode ser um problema para o agronegócio visto que o clima afeta a qualidade e quantidade dos produtos agrícolas.

Ambos os setores são de extrema importância para a economia brasileira, mas enfrentam problemas que precisam ser resolvidos para garantir a sustentabilidade e o crescimento a longo prazo. A falta de conhecimento sobre as condições climáticas pode agravar esses problemas e dificultar a tomada de decisões informadas.

1.2 Justificativa

O uso de uma *dashboard* para o controle de dados climáticos pode ser muito útil para o mercado de energia elétrica e para o agronegócio no Brasil, visto que ela permite o acompanhamento e a análise dos dados climáticos em tempo real, permitindo a tomada de decisões informadas e rápidas.

No caso do mercado de energia elétrica, por exemplo, a esta plataforma pode ser utilizada para monitorar as condições climáticas e antecipar possíveis problemas de produção e distribuição de energia. Isso é especialmente importante em um país como o Brasil, que enfrenta eventos climáticos que podem afetar a produção e distribuição de energia. Além disso, o uso de uma *dashboard* também pode permitir a implementação de medidas de contingência para minimizar os efeitos desses eventos.

No caso do agronegócio, a *dashboard* também pode ser utilizada para acompanhar as condições climáticas e garantir a qualidade e quantidade dos produtos agrícolas.

1.3 Objetivos

Objetivo Geral: Desenvolver uma *dashboard* que mostre dados de temperatura, chuva, anomalia de temperatura, temperatura de solo e umidade com telas de adição de tarefas e perfil de usuário

Objetivos Específicos:

- Estudar técnicas e ferramentas de engenharia de software a partir do desenvolvimento de uma *dashboard*, aplicando fazendo de linguagens de programação focadas do desenvolvimento *web*;
- Estudar técnicas e ferramentas de análise de dados focado no uso de ferramentas como *Pandas* e *Python*, coletando, tratando e disponibilizando os dados;
- Desenvolver um aplicativo para análise de dados climáticos que una os estudos em engenharia de software e análise de dados;

2 Fundamentação Teórica

A execução deste trabalho exige conhecimento de conceitos relacionados a análise de dados e engenharia de *software*, portanto, nesta seção estão apresentados conceitos-chave referentes às áreas de análise de dados e engenharia de *software*, incluindo informações acerca das técnicas utilizadas para as tarefas de construção de gráficos, manipulação de dados e disponilibização de dados utilizados no trabalho.

2.1 Engenharia de *software*

A engenharia de *software* é a aplicação de teorias, métodos e técnicas práticas para o desenvolvimento de *software* de maneira sistemática e disciplinada. Ela envolve a criação de processos eficientes para a construção e manutenção de *software* de alta qualidade, considerando fatores como tempo, custo, recursos e requisitos de negócio. A engenharia de *software* também envolve a aplicação de conhecimento e técnicas de gerenciamento de projetos para garantir que o *software* seja desenvolvido de maneira eficiente e eficaz.

2.1.1 Qualidade do Código

A qualidade de código é a medida em que um código é bem escrito, fácil de entender, de manter e de mudar. Alguns fatores que podem afetar a qualidade do código incluem a clareza do código, a estrutura e organização do código, a legibilidade, a reutilização de código, a eficiência do código e a robustez. A qualidade de código é importante porque pode afetar a confiabilidade, a manutenibilidade e o desempenho do sistema que o código implementa.

Uma das principais referências em qualidade de código é o conjunto de boas práticas chamado *Clean Code*, escrito por Robert C. Martin (MARTIN, 2008) e os pricípios *SOLID* elaborados pelo mesmo autor (MARTIN, 2000)

2.1.1.1 *Clean Code*

Clean Code é um livro escrito por Robert C. Martin Martin (2008) que define as boas práticas de escrita de código de qualidade. O livro oferece dicas e técnicas para escrever código limpo, bem estruturado e fácil de manter. Ele também apresenta exemplos de código "sujo"e mostra como refatorá-los para torná-los mais legíveis e mantíveis. É considerado uma referência importante para desenvolvedores de *software* e é amplamente utilizado como um guia para escrever código de qualidade. Os principais pontos abordados pelo livro estão descritos a seguir.

- Nomes Significativos Escolha nomes claros e precisos para suas variáveis, métodos e classes, para que outros desenvolvedores possam entender facilmente o que seu código está fazendo.
- Código Limpo Remova o código desnecessário e mantenha seu código bem organizado e estruturado, para que seja mais fácil de ler e entender.
- Código Repetido Evite escrever código que faz a mesma coisa em vários lugares. Em vez disso, use funções ou métodos para encapsular a funcionalidade comum.
- Boas Práticas Siga as boas práticas de design de código e utilize princípios de design de código, como *SOLID*, para escrever código que seja fácil de entender, manter e mudar.
- Consistência Siga um estilo de codificação consistente ao longo do seu projeto, para que o código seja mais fácil de ler e entender.

  Seja consistente: Siga um estilo de codificação consistente ao longo do seu projeto, para que o código seja mais fácil de ler e entender.

2.1.1.2 *SOLID*

SOLID são um conjunto de princípios de design de código desenvolvidos por Robert C. Martin (MARTIN, 2000) que estão descritos a seguir.

- Princípio da Responsabilidade Única Cada módulo ou classe deve ter uma única responsabilidade, e essa responsabilidade deve ser completamente encapsulada pelo módulo ou classe.
- Princípio Aberto-Fechado Você deve ser capaz de estender um comportamento de um módulo ou classe sem modificá-lo.
- Princípio da Substituição de Liskov As classes derivadas devem ser substituíveis por suas classes base.
- Princípio da Segregação de Interface Muitas interfaces específicas são melhores do que uma interface única genérica.
- Princípio da Inversão de Dependência Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

2.2 Análise de Dados

De acordo com Carvalho e Mamede (2017), análise de dados é o processo de examinar, limpar, transformar e modelar dados com o objetivo de descobrir informações úteis, tirar

conclusões e tomar decisões. É uma área da ciência de dados que envolve a aplicação de técnicas estatísticas, algoritmos de aprendizado de máquina e técnicas de visualização de dados para entender e extrair *insights* valiosos dos dados.

Existem diferentes etapas envolvidas na análise de dados, incluindo:

- Coleta de dados: Neste passo, os dados são coletados de diferentes fontes, como bancos de dados, sensores, questionários ou raspagem de sites da *web*;
- Limpeza de dados: Os dados coletados podem ser incompletos, incorretos ou inconsistentes, o que pode afetar a precisão das análises. Por isso, é necessário limpar os dados para remover erros, duplicatas e dados irrelevantes;
- Transformação de dados: Os dados podem precisar ser transformados para serem compatíveis com as ferramentas de análise ou para facilitar a análise. Por exemplo, os dados podem ser agregados, pivôtados ou transformados em um formato diferente;
- Exploração de dados: Neste passo, os dados são examinados para descobrir padrões, tendências e relações. Isso pode ser feito por meio de gráficos, tabelas ou técnicas estatísticas, como testes de hipóteses ou análise de componentes principais;
- Modelagem de dados: Os dados são usados para treinar um modelo de aprendizado de máquina ou outro tipo de modelo matemático para prever resultados ou detectar padrões.
- Visualização de dados: Os resultados da análise são apresentados de maneira clara e visualmente atraente por meio de gráficos, tabelas e outros tipos de visualização de dados;
- Tomada de decisão: Os *insights* obtidos a partir da análise de dados são usados para tomar decisões informadas ou para orientar a tomada de ações;

A análise de dados é amplamente utilizada em vários campos, como negócios, ciência, saúde e governo, para ajudar a entender e a resolver problemas complexos (SIGELMAN; BROWN, 2021).

2.3 *JavaScript*

JavaScript é uma linguagem de programação de *script* leve e interpretada, com tipagem dinâmica e baseada em protótipos. Ela foi criada para ser usada principalmente em páginas web, mas pode ser usada em qualquer ambiente que suporte a execução de código (MOZILLA DEVELOPER NETWORK, 2022).

JavaScript é executada principalmente no lado do cliente, o que significa que o código é executado pelo navegador do usuário, em vez de pelo servidor. Isso permite que ela faça

coisas como validar formulários de entrada de dados no navegador, criar animações e interações com o usuário e muito mais (SINGH; PATEL; GUPTA, 2020).

Além disso é orientada a objetos, o que significa que ela usa objetos para representar dados e funcionalidades. Também é baseada em eventos, o que significa que ele permite que você escreva código que é executado quando um determinado evento ocorre, como quando o usuário clica em um botão ou quando uma página é carregada (ZACHARY, 2022).

2.4 *React*

React é uma biblioteca *JavaScript* para criar interfaces de usuário, criado pelo *Facebook* e é amplamente utilizado para criar aplicativos web e mobile por muitas empresas de tecnologia, incluindo o *Facebook*, *Instagram*, *Netflix* e *Airbnb* SimilarTech (2022).

O *React* funciona criando componentes de interface de usuário que são renderizados como HTML. Cada componente é escrito como uma função ou classe que retorna o HTML que deve ser renderizado. O *React* usa o conceito de *virtual DOM* para determinar as diferenças entre o HTML atual e o HTML que deve ser renderizado e, em seguida, atualiza apenas os elementos que precisam ser alterados, o que o torna eficiente para aplicativos de grande escala.

2.4.1 *Virtual DOM*

O *Virtual DOM* é uma representação em memória de um *DOM* real que é usada pelo *React* para atualizar eficientemente a *IU* quando os dados mudam. Quando os dados de um componente mudam, o *React* cria uma nova representação do *Virtual DOM* para esse componente. Em seguida, ele compara a nova representação com a representação anterior e determina quais elementos do *DOM* precisam ser alterados para refletir as alterações nos dados. Isso é muito mais eficiente do que atualizar o DOM diretamente, pois evita a necessidade de percorrer todo o DOM para encontrar as alterações. Ao invés disso, o *React* pode atualizar apenas os elementos que foram alterados, o que pode tornar as aplicações *React* mais rápidas e responsivas.

2.4.2 Componentes

Em *React*, um componente é uma função ou classe que retorna um elemento do *DOM*. Os componentes são os blocos básicos de construção de *IU* em uma aplicação *React*. Eles podem ser reutilizados em vários lugares na *IU* e podem conter outros componentes como filhos (SMITH, 2022b).

Existem dois tipos principais de componentes em *React*: componentes de classe e componentes de função.

Componentes de classe são componentes que são implementados como classes do *JavaScript* e que possuem estado (um conjunto de dados que é mantido pelo componente) e ciclo de vida (métodos que são chamados em diferentes momentos durante o ciclo de vida do componente).

Componentes de função são componentes que são implementados como funções do *JavaScript* e não possuem estado. Eles são mais simples do que os componentes de classe e geralmente são usados para componentes que não precisam de um estado próprio.

2.5 *Python*

Python é uma linguagem de programação de alto nível, interpretada, dinâmica e orientada a objetos. Ela foi criada por Guido van Rossum em 1989, e tem uma sintaxe simples e legível que se parece com o idioma inglês. É uma linguagem versátil e pode ser usada para muitos propósitos diferentes, como desenvolvimento web, análise de dados e ciência de dados, automatização de tarefas, desenvolvimento de aplicativos de *desktop*, desenvolvimento de jogos. *Python* possui uma grande comunidade ativa e uma ampla variedade de bibliotecas e *frameworks* disponíveis para uso. Alguns exemplos populares incluem *NumPy* e *Pandas* para análise de dados (SMITH; WILLIAMS; PATEL, 2022).

Python tem suporte a:

- Tipagem dinâmica: as variáveis em Python não precisam ser declaradas com um tipo específico, pois o tipo é determinado automaticamente quando o valor é atribuído;
- Suporte a módulos: Python possui um sistema de módulos que permite dividir o código em vários arquivos e usá-los em diferentes projetos;
- Suporte a exceções: Python possui um sistema de exceções que permite lidar com erros e exceções de maneira organizada e controlada;

2.6 *Pandas*

Pandas é um pacote de código aberto para a linguagem de programação *Python* que fornece ferramentas de análise de dados e manipulação de dados. Ele foi criado para tornar mais fácil o trabalho com dados e ser rápido, fácil de usar e versátil (SMITH, 2022a).

Foi desenvolvido a partir do pacote *Numpy*, que fornece suporte para arrays multidimensionais e operações matemáticas avançadas. O *Pandas* adiciona capacidades de manipulação de dados, como indexação, agrupamento, junção e agregação, e também fornece ferramentas para ler e escrever dados em vários formatos, como CSV, Excel e SQL.

Algumas das principais vantagens do Pandas incluem:

- Manipulação de dados fácil e rápida: possui uma sintaxe intuitiva e uma ampla variedade de funções de manipulação de dados que tornam o trabalho com dados mais fácil e rápido (WILLIAMS; PATEL; KIM, 2020);
- Integração com outras bibliotecas: o pode ser facilmente integrado a outras bibliotecas de análise de dados, como o *Numpy*, o *Matplotlib* e o scikit-learn, o que torna mais fácil a realização de tarefas de análise avançada;
- Suporte a vários formatos de dados: é capaz de ler e escrever dados em vários formatos, incluindo CSV, Excel, JSON e SQL, o que o torna útil em muitos cenários diferentes;
- Operações de junção e agregação avançadas: possui suporte para operações de junção e agregação avançadas, como junção de tabelas baseadas em chaves e agregação de dados com base em funções especificadas pelo usuário (PATEL, 2021);

2.7 Redes Neurais Convolucionais

Redes Neurais Convolucionais como descritas no livro *Neural Networks and Deep Learning* (NIELSEN, 2019) são uma classe de redes neurais que foram especialmente projetadas para lidar com dados bidimensionais,. Elas são chamadas de "convolucionais"porque utilizam uma técnica de operação matemática chamada de *kernel* que é multiplicação de matrizes por pedaços do *input* de dados e a soma dos resultados. Isso permite que as CNNs aprendam padrões específicos. As CNNs foram inspiradas pelo funcionamento do córtex visual do cérebro humano e têm sido muito bem-sucedidas em muitas tarefas de classificações e reconhecimento de objetos (LECUN et al., 1998).

Uma das principais diferenças entre uma CNN e outras redes neurais é a forma como os pesos são compartilhados. Em uma rede neural normal, cada neurônio recebe entrada de todos os neurônios na camada anterior e tem seus próprios pesos. Isso significa que, para uma imagem de entrada de tamanho NxN, haverá NxNxC pesos onde C é o número de canais (por exemplo, 3 para imagens RGB). Isso pode levar a redes muito grandes e complexas, especialmente quando se trata de imagens de alta resolução.

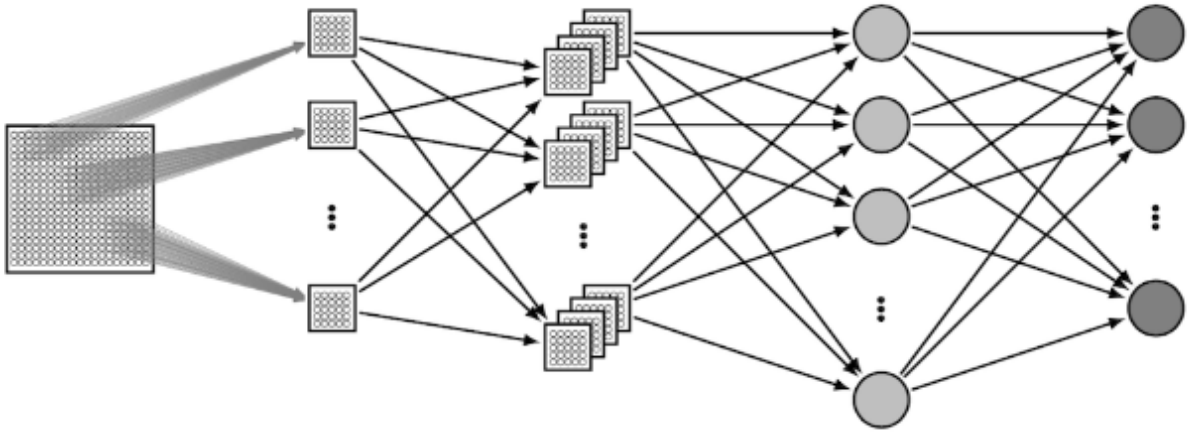
Para lidar com isso, as CNNs compartilham os pesos através de uma camada de convolução. Em vez de cada neurônio ter seus próprios pesos, todos os neurônios na camada de convolução compartilham os mesmos pesos. Isso significa que, em vez de ter NxNxC pesos, há apenas CxFxKxK pesos, onde F é o número de filtros (também conhecidos como *kernels*) e K é o tamanho do *kernel*. Isso reduz significativamente o número de pesos na rede e torna a rede mais fácil de treinar (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Além da camada de convolução, as CNNs também geralmente incluem camadas de *pooling*, que são usadas para reduzir a dimensionalidade da imagem e tornar a rede mais robusta a translações. Há vários tipos diferentes de camadas de *pooling*, mas a mais comum é

a camada de *pooling* máximo, que seleciona o valor máximo em um determinado tamanho de janela. Outros tipos de camadas de *pooling* incluem a camada de *pooling* médio, que seleciona a média dos valores em uma janela, e a camada de *pooling* L2, que seleciona o valor mínimo da norma L2 em uma janela (KIM, 2014).

Depois de várias camadas de convolução e *pooling*, as CNN's geralmente têm uma ou mais camadas *fully-connected* (também conhecidas como camadas densas), que são semelhantes às camadas encontradas em outras redes neurais.

Figura 2 – Modelo de camadas de uma Rede Neural Convolucional.



Fonte: Nielsen (2019)

.

2.8 *Numerical Weather Prediction*

Numerical weather prediction é o processo de prever o tempo futuro usando modelos matemáticos de previsão do tempo. Esses modelos são baseados em equações que descrevem como a atmosfera se comporta e são executados em computadores para calcular previsões do tempo em um determinado local (SCAIFE; MARAUN; MAROTZKE, 2020).

Para realizar a previsão numérica do tempo, os modelos precisam de dados de entrada atuais, como temperatura, pressão, umidade e vento. Esses dados são coletados por uma variedade de fontes, incluindo sondas meteorológicas, satélites e estações meteorológicas. Esses dados são usados pelos modelos para calcular como a atmosfera se comportará no futuro e gerar previsões do tempo para um determinado período de tempo.

A precisão da previsão numérica do tempo melhora à medida que nos aproximamos do presente, mas mesmo assim a previsão do tempo a longo prazo pode ser bastante precisa. No entanto, é importante lembrar que o tempo é imprevisível em um nível fundamental e que as previsões do tempo são apenas uma aproximação do que pode acontecer.

2.9 Anomalia de chuva

Anomalia de chuva é a diferença entre a quantidade de chuva esperada em um determinado período de tempo em um local específico e a quantidade de chuva realmente observada nesse mesmo período de tempo. Por exemplo, se a quantidade média de chuva para um determinado mês é de 100 milímetros e a quantidade de chuva observada nesse mês é de 50 milímetros, a anomalia de chuva seria de -50 milímetros. Isso indica que houve uma deficiência de chuva em relação à quantidade esperada. Se a quantidade de chuva observada tivesse sido de 150 milímetros, a anomalia de chuva seria de +50 milímetros, indicando que houve um excesso de chuva em relação à quantidade esperada (FRANÇA; KOUSKY; DIAS, 2009).

Anomalias de chuva são comumente usadas para avaliar o impacto da mudança climática e para monitorar eventos meteorológicos extremos, como secas ou inundações. Elas também são úteis para agricultores e outros profissionais que dependem de condições climáticas ideais para suas atividades.

3 Metodologia

Neste capítulo é apresentada a metodologia utilizada para o desenvolvimento da *dashboard* de análise de dados climáticos em Bauru usando a biblioteca *React* do *JavaScript* e os pacotes necessários para desenvolvimento de uma UI agradável ao usuário e Pandas para manipulação de dados que será mostrado na *dashboard* seguindo os principios de qualidade de software *SOLID* e *Clean Code*.

3.1 Bases de Dados

Trabalhar com dados de qualidade, que sejam fidedignos, completos e não enganosos é necessário para a análise de dados, pois deste modo o usuário que for consumir desses dados terá uma resposta correspondente com o ambiente observado e as previsões serão referentes ao que apresentado pelos dados obtidos.

> "Os dados são a base da ciência de dados. Sem dados de qualidade, os resultados das análises podem ser imprecisos ou enganosos, o que pode levar a conclusões erradas e decisões ruins. Por isso, é fundamental garantir que os dados sejam precisos, completos e relevantes para a tarefa em questão. Isso envolve a coleta, a limpeza, a integração e a validação de dados, bem como a gestão de sua qualidade ao longo do tempo."(CARVALHO; MAMEDE, 2017, tradução nossa).

Para obter dados de qualidade a base de dados ultilizada foi a base de dados de municípios do Brasil do Instituto Nacional de Metereologia, (Ministério da Agricultura, Pecuária e Abastecimento, 2021) que contém dados referentes a precipitação total, pressão atmosférica máxima, pressão atmosférica mínima, radiação global, temperatura máxima, temperatura mínima, umidade máxima, umidade mínima, direção dos ventos, velocidade máxima dos ventos e velocidade mínima dos ventos de Bauru desde 2021.

3.1.1 Manipulação dos Dados

Com os dados obtidos é necessário filtrar quais informações serão úteis para o projeto e quais podem ser descartadas, além de agrupar todo o conjunto de dados em uma base única de fácil acesso para a aplicação, para essa etapa será necessário o uso da biblioteca Pandas, já que os dados são entregues em pacotes anuais em formato CSV.

Além de filtrar os dados é necessário manipula-los a fim de gerar um padrão ao valores, como por exemplo deixar todos os valores de temperatura em números de ponto flutuante, excluir linhas com dados faltando e até mesmo converter valores de data para um mesmo tipo como *datetime*.

3.1.2 Climatologia

Segundo Valente et al. (2006), a climatologia é "o ramo da meteorologia que estuda o clima, incluindo sua distribuição espacial e temporal e os fatores que o influenciam". O artigo também destaca que a climatologia é uma ciência interdisciplinar, que se relaciona com outras áreas do conhecimento, como a geologia, a biologia, a oceanografia e a antropologia, entre outras.

Durante a manipulação dos dados é possível criar a climatologia do clima de Bauru através das ferramentas de manipulação de dados que o Pandas fornece como a criação de *DataFrames* e operações matemáticas entre linhas e colunas de um *DataFrame*

3.1.3 *DataFrame*

Um dataframe é uma estrutura de dados bidimensional, similar a uma planilha ou tabela, que armazena os dados em linhas e colunas. Ele é amplamente utilizado em linguagens de programação para análise de dados, como o Python, o R e o Julia.

De acordo com o artigo "Introdução ao uso de dataframes em Python para análise de dados"os *dataframes* são:

> "A principal estrutura de dados para manipulação e análise de dados em Python". O artigo explica que os dataframes permitem que os dados sejam facilmente indexados, filtrados, agrupados e transformados, o que os torna uma ferramenta poderosa para a análise de dados."(MIRANDA; PERIN; SOUZA, 2018).

3.1.4 *PyCharm*

PyCharm é uma IDE para a linguagem de programação *Python*. Ele foi desenvolvido pela JetBrains e é projetado para tornar mais fácil o desenvolvimento de aplicativos *Python*, oferecendo recursos como destaque de sintaxe, verificação de erros, sugestão de código e integração com controle de versão.

Alguns dos recursos do *PyCharm* incluem:

Suporte para várias linguagens de programação: o *PyCharm* suporta várias linguagens de programação, incluindo *Python*, *JavaScript*, HTML, CSS e SQL, e oferece recursos específicos para cada uma delas.

- *Debugger*: permite pausar a execução do código para examinar variáveis e expressões que estão sendo executadas.
- *Versionamento*: é integrado com o GIT para os desenvolvedores controlarem o versionamento do código.

3.2 *Dashboard*

Uma *dashboard* é um tipo de interface gráfica que exibe informações importantes de maneira resumida e fácil de entender (WIJK; KLAMMA, 2010). Ela é usada para monitorar e controlar indicadores-chave de desempenho em tempo real, ajudando os usuários a tomar decisões informadas e a identificar problemas ou oportunidades de melhoria. Podem ser utilizadas em várias áreas, como negócios, saúde, tecnologia e governo, e podem exibir informações de diversas fontes, como bases de dados, planilhas, sensores ou APIs. Elas podem ser acessadas por meio de computadores, *tablets* ou *smartphones*, e podem ser customizadas para atender às necessidades específicas de cada usuário ou organização, podem ser incluídos em uma *dashboard* incluem gráficos, tabelas, indicadores de progresso, alertas e mapas.

A *dashboard* foi planejada para ser feita usando a biblioteca *React* com auxílio dos pacotes *Nivo* para geração de gráficos com os dados recebidos, *Axios* para realizar requisições HTTP, *FullCalendar* para gerar uma interface de tarefas e calendário de usuário, a API *OpenWeatherMap* para buscar os dados de clima correntes e previstos, a API *OpenAgro* para buscar os dados de soja correntes e previstos e *Material UI* para deixar a interface de acordo os padrões de exibição do *Google*.

3.2.1 *FullCalendar*

FullCalendar é uma biblioteca *JavaScript* que fornece um calendário de arrastar e soltar em tamanho real. Ele permite que os usuários criem e gerenciem eventos e os mostrem em um calendário é personalizável e pode ser integrado a outras bibliotecas ou estruturas, como *jQuery*, *Angular* e *Vue.js*. É frequentemente usado para criar aplicativos baseados em calendário, como agendamento de eventos, gerenciamento de tarefas e sistemas de reservas. O *FullCalendar* pode ser usado em aplicativos da Web e móveis e oferece suporte a uma ampla variedade de plataformas de navegadores e dispositivos.

3.2.2 *Nivo*

Nivo é uma biblioteca *JavaScript* que fornece um conjunto de componentes reutilizáveis para criar aplicativos de visualização de dados com *React*. Baseia-se na biblioteca D3, que é uma biblioteca popular para a criação de tabelas e gráficos, e inclui uma ampla variedade de visualizações, como gráficos de barras, gráficos de linha, gráficos de pizza, gráficos de dispersão e muito mais.

3.2.3 *Axios*

Axios é uma biblioteca *JavaScript* que fornece uma maneira de fazer solicitações HTTP a partir do navegador. É uma biblioteca baseada em *promisses* que utiliza recursos modernos

do *JavaScript* e é executada tanto no cliente quanto no servidor. O *Axios* facilita o envio de solicitações HTTP e o processamento da resposta, e pode ser usado com *React*.

Uma das principais características do *Axios* é que ele permite fazer solicitações HTTP a domínios diferentes, o que não é possível com o objeto de *XMLHttpRequest built*-in do *JavaScript*. Ele também oferece suporte ao cancelamento de solicitações e gerenciamento de concorrência de solicitações, e possui uma API simples de usar.

3.2.4 *OpenWeatherMap*

A *OpenWeatherMap* API é uma API de previsão do tempo que fornece dados meteorológicos atualizados para todo o mundo. A API pode ser usada para obter informações sobre o clima atual em uma localidade específica, bem como previsões meteorológicas futuras. Alguns exemplos de informações que podem ser obtidas através da API incluem temperatura, umidade, pressão atmosférica, velocidade do vento e condições climáticas atuais, como chuva ou neve.

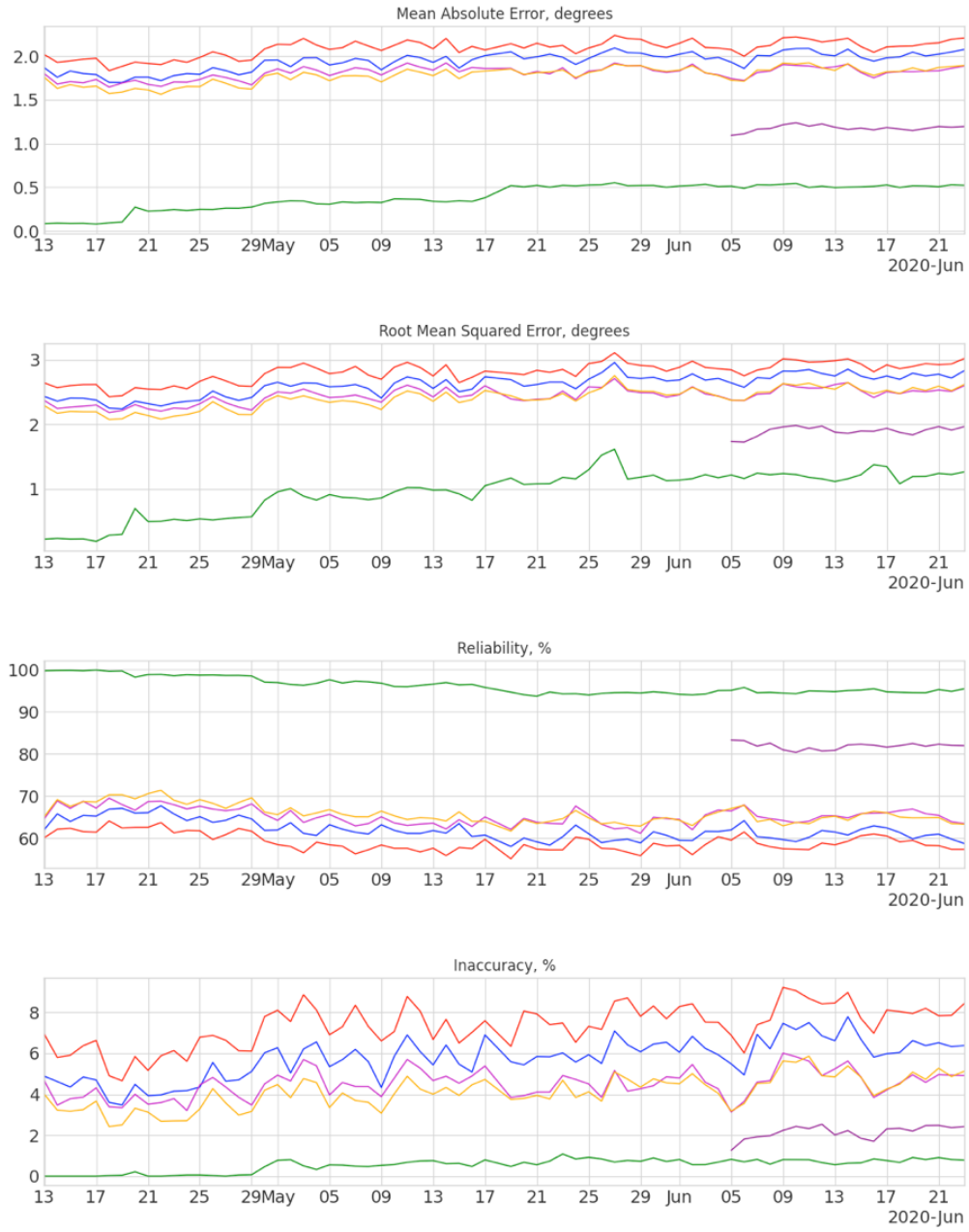
A *OpenWeatherMap* API é baseada em tecnologia de *web services* e pode ser acessada através de uma solicitação HTTP. A API é fornecida gratuitamente, mas também há opções pagas que oferecem acesso a dados mais detalhados e atualizados mais frequentemente.

Para realizar as previsões a API utiliza CNN e NWP para garantir a alta acurácia de seus resultados usando imagens do satélite que incluem o GOES, MetOp e Aqua (satelite da NASA que monitora a água no sistema terra-atmosfera), para verificar a precisão da previsão são usadas duas métricas de erro estatístico mais comuns para a previsão da temperatura: o Erro Absoluto Médio (MAE) avalia o erro médio, enquanto que o Erro Quadrático Médio (RMSE) se concentra nos erros maiores. Com essas métricas de confiabilidade e imprecisão obtém-se uma descrição qualitativa da precisão em termos percentuais. Todos os cálculos abaixo foram realizados em graus Celsius.

- MAE - diferença absoluta das estações, em graus; menor é melhor.
- RMSE - em graus; menor é melhor.
- Confiabilidade - percentual de tempo em que os valores do modelo estiveram dentro de ± 2 graus da verdade terrestre; maior é melhor. Considera-se um erro de até 2 graus na previsão como aceitável.
- Imprecisão - percentual de tempo em que os valores do modelo não estiveram dentro de ± 5 graus da verdade terrestre; menor é melhor. Considera-se um erro de mais de 5 graus na previsão como inaceitável.

O período de análise é de 13 de abril de 2020 00:00 até 26 de junho de 2020 00:00 (fuso horário UTC).

Figura 3 – Dados de precisão de previsão da api openweathermap.



modelo, NOAA GFS05, GFS025, NOAA GFS05 corrigido pelo *openweathermap*, NOAA GFS025 corrigido pelo openweathermap, dados observados

Fonte: OpenWeatherMap (2020).

A Figura 3 mostra que o MAE é de cerca de 0,5 graus, o RMSE é menor que 2 graus, a confiabilidade está entre 90% e 100% e a incorreção é de cerca de 1% (menor é melhor). Fica claro que o modelo OpenWeather fornece o resultado preciso. No entanto, deve-se lembrar que a diferença real entre o pronóstico e a situação real em um lugar e momento específicos pode ser maior do que esses erros médios.

3.2.5 *OpenAgro*

A *OpenAgro* é uma API fornecida peal *Extreme Electronics* que fornece dados estatísticos sobre uma ampla gama de produtos agrícolas. A API pode ser usada para obter informações sobre temperatura do solo, umidade e dados agrícolas importantes para uma ampla variedade de produtos, incluindo frutas, verduras, grãos, laticínios e outros.

A API é baseada em tecnologia de *web services* e pode ser acessada através de uma solicitação HTTP. Os dados são fornecidos em formato JSON ou CSV e podem ser facilmente integrados em aplicativos ou *websites*. A API é gratuita e está disponível para uso comercial e não comercial.

3.2.6 *Material UI*

Material UI é um conjunto de componentes de UI para o *React* que seguem o *design* do *Google*. É um conjunto de diretrizes de *design* que visa criar aplicações e *sites* com uma aparência consistente e intuitiva, fornecendo uma ampla variedade de componentes pré-construídos, como botões, caixas de seleção, menus de navegação e muito mais, que podem ser usados para criar IU com o *design* de material *design*. Além disso, o Material UI oferece uma ampla personalização, permitindo que os desenvolvedores ajustem a aparência dos componentes para atender às suas necessidades específicas.

3.2.7 *Visual Studio Code*

Visual Studio Code é um editor de código-fonte desenvolvido pela *Microsoft* para *Windows*, Linux e *macOS*. Ele é projetado para ser leve e extensível, e vem com uma variedade de recursos para ajudar os desenvolvedores a escrever e depurar código. Alguns dos recursos incluem:

- Destaque de sintaxe: o VS Code destaca a sintaxe de várias linguagens de programação, o que facilita a leitura e a compreensão do código.
- Depurador: o VS Code inclui um depurador integrado que permite aos desenvolvedores pausar o código, examinar variáveis e avaliar expressões enquanto o código está sendo executado.
- Integração com controle de versão: o VS Code tem integração com o Git e outras ferramentas de controle de versão, o que permite aos desenvolvedores gerenciar o histórico de alterações do código e trabalhar em projetos em equipe.
- Extensões: o VS Code é extensível através de um amplo conjunto de extensões disponíveis na loja de extensões do VS Code, que adicionam novas funcionalidades e integrações ao editor.

O VS Code é amplamente utilizado por desenvolvedores de todo o mundo e é compatível com uma ampla variedade de linguagens de programação, incluindo *JavaScript* e *Python*.

3.3 Arquitetura do projeto

Aqui será explicado as escolhas de arquitetura de pastas, *interface* e *design* do projeto.

3.3.1 Arquitetura de pastas

Ao criar um projeto *React*, ele automaticamente cria um esquema de pastas onde a parte programável fica dentro da pasta *source*, sendo assim toda a arquitetura de projeto parte dessa pasta raiz. O desenvolvimento aplicações em *React* se dá pelo uso de *components* e *scenes* que são conceitos fundamentais no desenvolvimento de aplicações com *React*. Eles são amplamente utilizados por desenvolvedores de todo o mundo e são considerados uma das principais vantagens da biblioteca.

A utilização de *components* e *scenes* ajuda a promover a reutilização de código e a manutenção de aplicações de grande escala, pois permite que o código seja dividido em partes mais gerenciáveis e facilmente testáveis. Além disso, a divisão da interface do usuário em componentes independentes também facilita a colaboração em equipe, já que diferentes desenvolvedores podem trabalhar em diferentes partes da aplicação de forma isolada. Esse projeto se baseia na arquitetura de pastas de *components* e *scenes* para o desenvolvimento da aplicação por ser a arquitetura mais usada no desenvolvido de aplicações *React*.

3.3.1.1 *Components*

Components de acordo com a documentação da biblioteca (FACEBOOK,) são os blocos de construção fundamentais da *interface* do usuário em uma aplicação *React*. Eles são responsáveis por representar visualmente os dados e permitir a interação do usuário com eles.

Cada componente é um pedaço de código *JavaScript* que renderiza algum conteúdo HTML. Os componentes podem ter seus próprios estados, que são variáveis que determinam como o componente se comporta e como ele se apresenta.

3.3.1.2 *Scenes*

Scenes segundo a documentação (FACEBOOK,) são simplesmente uma coleção de componentes que se relacionam para formar uma tela ou página da aplicação. Em geral, uma *scene* é composta por um ou mais componentes de *layout*, que por sua vez, podem conter outros componentes menores.

3.3.2 *Design*

O design da aplicação foi pensado para o mais intuitivo possível e oferecer o máximo de informações possíveis para quem visualizar. A tela inicial deverá conter todos os dados oferecidos de forma bem diagramada para que todos os blocos e gráficos se encaixem bem para o usuário ter menos cliques ao navegar e ter as informações de fácil acesso. Além disso para melhorar a visualização do usuário será implemtado um *dark mode* e *light mode* e terá funcionalidades de agendar compromisso em um calendário, além de uma tela com informações de usuários do sistema.

As cores predominantes serão tons de azul já que segundo o artigo *The connotations of blue: A cross-cultural study* (SMITH; KIM, 2015) a cor azul remete a confiança, estabilidade e inteligência, adicionado a cor azul, além de tons de azul o laranja foi escolhido como cor secundária já a cor complementar segundo o site *Color Matters* (COLORMATTERS, 2022).

4 Desenvolvimento

O desenvolvimento deste trabalho foi divido em três partes, na primeira foi a colheita e manipulação de dados históricos usando *Python*, *Pandas* na IDE *PyCharm* para a criação da climatologia que vai ser usada na Anomalia de Chuva, a segunda parte foi a criação da *dashboard* usando *React*, *JavaScript*, *Nivo*, *Material UI* e *FullCalendar* apenas usando *mocks*, sem fazer nenhum integração com API. Já na terceira parte foi realizado o *input* de dados das APIs *OpenWeatherMap*, *OpenAgro* e os dados de climatologia na aplicação para ela mostrar dados reais e confiáveis.

4.1 Tratamento de dados

O tratamento de dados é uma área da ciência da computação que se ocupa da coleta, armazenamento, processamento, análise e visualização de dados. É uma parte importante de muitas áreas da ciência, da tecnologia e da indústria, pois permite que as informações sejam transformadas em conhecimento útil e tomadas decisões informadas. Entre as etapas do tratamento de dados estão a coleta, processamento e visualização dos dados.

Para a implementação, a linguagem *Python* foi utilizada, por ser uma linguagem de fácil acesso, altamente documentada e que oferece um conjunto amplo de ferramentas de análise de dados, que são vantajosas para este trabalho. Como discutido na sessão de Base de Dados 3.1 os dados históricos foram retirados do Instituto Nacional de Meteorologia. Durante a primeira etapa, foram realizados testes com as técnicas de adaptação de domínio seguindo o protocolo e os conjuntos de dados apresentados no Capítulo 3. Para colher esses dados foi implementado um *scraper* em *Python* usando a biblioteca *BeautifulSoup*

4.1.1 Coleta de Dados

Segundo Rao (2015) a biblioteca *BeautifulSoup* é ideal para fazer raspagem de dados na internet e trabalhar com eles em formato de *DataFrame* posteriormente, O Código 1 mostra a implementação.

Código 1 – Scraper para aquisição de dados

```
1 import requests
2 import zipfile
3 from bs4 import BeautifulSoup
4
5 for i in range(1, 22):
```

```
6       url = "https://portal.inmet.gov.br/uploads/dadoshistoricos
           /20{i:02d}.zip"
7       response = requests.get(url)
8
9       open("2000.zip", "wb").write(response.content)
10
11      with zipfile.ZipFile("2000.zip", "r") as zip_ref:
12          zip_ref.printdir()
```

4.1.2 Processamento dos Dados

Nessa área os dados coletados são processados a fim de produzir um conjunto de dados que sejam úteis para serem visualizados em forma de climatologia para os comercializadores.

4.1.2.1 Concatenação de *DataFrames*

Como os dados são divididos e pacotes anuas de arquivos CSV é necessário unificar todos os aquivos em um *DataFrame* para fazer somente uma leitura com todos os dados, para fazer isso é necessário ler todos os arquivos e juntar todos os *dataFrames* em um só, O Código 2 mostra a implementação.

Código 2 – Concatenação de todos os *DataFrames*

```
1  def unificar_dados(self):
2      df = pd.DataFrame()
3      for i in range(2001, 2022):
4          weather = pd.read_csv(f'data/rain_data/BAURU_{i}.CSV',
               sep=';', on_bad_lines='skip', encoding='latin-1')
5          df = pd.concat([df, weather], axis=0)
6      df.reset_index(inplace=True, drop=True)
7      del df['Unnamed: 19']
8      return df
```

4.1.2.2 Tratamento do *Dataframe*

Para facilitar a manipulação do *DataFrame* no código e prevenir erros, as colunas e linhas da base de dados deve ser tratada, para isso são feitos alguns ajustes, como renomeação de colunas, ajuste no formato de datas e tratamento e filtragem de valores nas células que fogem do padrão do resto do *DtaFrame*

4.1.2.2.1 Renomeação de Colunas

Como as colunas possuem nomes que não possuem o padrão PEP 8 (PYTHON-SOFTWARE-FOUNDATION, 2001) de nomeação de objetos do *Python*, assim como aparece

no Código 3.

Código 3 – Renomeação das colunas

```
def renomear_base(df: pd.DataFrame):
        df = df.rename({
            'DATA (YYYY-MM-DD)': 'date',
            'HORA (UTC)': 'time',
            'PRECIPITA  O TOTAL, HOR RIO (mm)': '
              total_precipitation',
            'PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA
              (mB)': 'atmospheric_pressure_station_level',
            'PRESS O ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)'
              : 'max_atmospheric_pressure',
            'PRESS O ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)
              ': 'min_atmospheric_pressure',
            'RADIACAO GLOBAL (KJ/m  )': 'global_radiation',
            'TEMPERATURA DO AR - BULBO SECO, HORARIA ( C )': '
              air_temperature',
            'TEMPERATURA DO PONTO DE ORVALHO ( C )': '
              dew_temperature',
            'TEMPERATURA M XIMA NA HORA ANT. (AUT) ( C )': '
              max_hourly_temperature',
            'TEMPERATURA M NIMA NA HORA ANT. (AUT) ( C )': '
              min_hourly_temperature',
            'TEMPERATURA ORVALHO MAX. NA HORA ANT. (AUT) ( C )
              ': 'dew_max_hourly_temperature',
            'TEMPERATURA ORVALHO MIN. NA HORA ANT. (AUT) ( C )
              ': 'dew_min_hourly_temperature',
            'UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)': '
              max_moisture_hourly',
            'UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)': '
              min_moisture_hourly',
            'UMIDADE RELATIVA DO AR, HORARIA (%)': '
              relative_moisture_hourly',
            'VENTO, DIRE  O HORARIA (gr) (  (gr))': '
              wind_direction_hourly',
            'VENTO, RAJADA MAXIMA (m/s)': 'wind_max_burst',
            'VENTO, VELOCIDADE HORARIA (m/s)': 'wind_max_speed
              ',
        }, axis=1)

        return df
```

4.1.2.2.2 Formatação de Datas

Como os dados de Climatologia serão horários, a forma correta de se trabalhar com datas é usar o formato *datetime*, para isso é aplicado o seguinte método no Código 4.

Código 4 – Formatação das datas

```
    def ajustar_date(self, df):
        df['time'] = df['time'].apply(lambda x: x.split()[0])
        df['time'] = df['time'].apply(lambda x: self.
            formatar_hora(x))
        df['time'] = pd.to_datetime(df['time'], format='%H:%M'
            )
        df['time'] = df['time'].dt.strftime('%Y%m%d%H%M')
        df['date'] = pd.to_datetime(df['date'], dayfirst=True)
        df['date'] = pd.to_datetime(df['date'], format='%Y-%m
            -%d %H:%M:%S')
        df['date'] = df['date'].dt.strftime('%Y%m%d%H%M')
        df['date'] = df.apply(lambda x: x.date[0:8] + x.time
            [8: 12], axis=1)
        df['date'] = pd.to_datetime(df['date'], dayfirst=True)
        del df['time']

        return df
```

4.1.2.2.3 Filtrar Valores

Como os dados de Climatologia serão horários, a forma correta de se trabalhar com datas é usar o formato *datetime*, para isso é aplicado o seguinte método no Código 5.

Código 5 – Filtragem de dados

```
def filtrar_valores(self, df: pd.DataFrame, coluna, i, mean):
    if i == 0:
        x = mean
    else:
        x = df[coluna][i - 1]
    j = i + 1
    while True:
        if df[coluna][j] != -9999.0:
            y = df[coluna][j]
            result = (x + y) / 2
            for k in range(i, j):
                df[coluna][k] = result
```

```
13              return j
14          j = j + 1
15  def ajustar_valores(self, df):
16      colunas = [item for item in df.columns if item not in ['
            date']]
17      for coluna in colunas:
18          df[coluna] = df[coluna].apply(lambda x: self.
                formatar_valor(x))
19          mean = pd.DataFrame(filter((-9999.0).__ne__, df[coluna
                ])).mean()
20          for i in range(len(df[coluna])):
21              if df[coluna][i] == -9999.0:
22                  i = self.filtrar_valores(df, coluna, i, mean
                        [0])
23      return df
```

4.1.3 Criação da Climatologia

Após estar com os dados tratados a manipulação desse *DataFrame* já pode ser feita para se criar um novo *DataFrame* de Climatologia. Para se obter uma climatologia que seja menos ruidosa para com os dados ao redor, aplica-se uma Suavização de Gauss (LIU; DONG; MA, 2012) de modo a deixar os dados mais suavisador com relação a vizinhança, Código 6.

Código 6 – Climatologia

```
1   def climatology(self, df: pd.DataFrame) -> pd.DataFrame:
2       climatology_df = df[['date', 'total_precipitation']]
3       for idx in df.index - 8:
4           if idx < 7:
5               continue
6           climatology_valor = 0
7           for j in range(idx - 7, idx + 8):
8               climatology_valor = climatology_valor + df['
                    total_precipitation'][j]
9           climatology_valor = climatology_valor / 15
10          climatology_df['total_precipitation'][idx] =
                climatology_valor
11
12      return climatology_df
```

Após as operações de manipulação de dados a fim de construir uma base de climatologia, esse foi o resultado para temperatura máxima e precipitação. O resultado é mostrado na Figura 4.

Figura 4 – Climatologia de Chuva e Temperatura.

	date	total_precipitation
0	2001-08-30 10:00:00	0.034078
1	2001-08-30 11:00:00	0.029210
2	2001-08-30 12:00:00	0.024341
3	2001-08-30 13:00:00	0.019473
4	2001-08-30 14:00:00	1.014605
5	2001-08-30 15:00:00	1.009737
6	2001-08-30 16:00:00	1.004868
7	2001-08-30 17:00:00	1.000000
8	2001-08-30 18:00:00	1.000000
9	2001-08-30 19:00:00	1.000000

Climatologia de Chuva.

	date	max_hourly_temperature
0	2001-08-30 10:00:00	30.112794
1	2001-08-30 11:00:00	30.389207
2	2001-08-30 12:00:00	30.512286
3	2001-08-30 13:00:00	30.542032
4	2001-08-30 14:00:00	30.401778
5	2001-08-30 15:00:00	30.091524
6	2001-08-30 16:00:00	29.714603
7	2001-08-30 17:00:00	29.284349
8	2001-08-30 18:00:00	28.840762
9	2001-08-30 19:00:00	28.517175

Climatologia de Temperatura.

Fonte: Elaborada pelo autor.

4.2 *Dashboard*

O desenvolvimento da aplicação foi feito em *React* e devido a amplo documentação da biblioteca para aplicações *web*. O início do desenvolvimento da aplicação começa pelo arquivo *app.js* onde os componentes globais são configurados como a *sidebar* e a *topbar* com os componentes que serão inseridos nos componentes globais. O Código 7 traz a implementação do componente inicial.

Código 7 – App.js

```
1
2  function App() {
3    const [theme, colorMode] = useMode();
4    const [isSidebar, setIsSidebar] = useState(true);
5
6    return (
7      <ColorModeContext.Provider value={colorMode}>
8        <ThemeProvider theme={theme}>
9          <CssBaseline />
10         <div className="app">
11           <Sidebar isSidebar={isSidebar} />
12           <main className="content">
13             <Topbar setIsSidebar={setIsSidebar} />
14             <Routes>
15               <Route path="/" element={<Dashboard />} />
16               <Route path="/team" element={<Team />} />
```

```
17                <Route path="/contacts" element={<Contacts />}
                     />
18                <Route path="/invoices" element={<Invoices />}
                     />
19                <Route path="/form" element={<Form />} />
20                <Route path="/faq" element={<FAQ />} />
21                <Route path="/calendar" element={<Calendar />}
                     />
22              </Routes>
23            </main>
24          </div>
25        </ThemeProvider>
26      </ColorModeContext.Provider>
27    );
28  }
29
30  export default App;
```

Após a codificação do arquivo *app.js* que é o *entry point* de uma aplicação *react* ou outros componentes podem ser programados.

4.2.1 Tela Inicial

A *home* da *dashboard* deve conter todas as informações sobre os dados climáticos a fim de diminuir a quantidade de cliques do usuário até o dado pretendido, ter a opção de troca de modo de cores e acesso ao menu. Como a tela inicial terá todos os dados climáticos da região de Bauru, a implementação de todos os componentes da *home* são descritos a baixo.

4.2.1.1 Bloco de clima

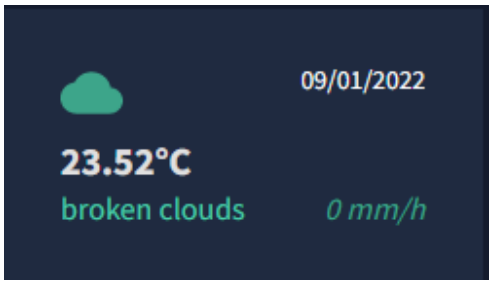
O bloco com informações de temperatura atuais e previstas são feitas usando o componente *Box* e *StatBox* que permite a personalização fácil usando CSS. O Código 8 é a implementação do primeiro bloco com os dados atuais de chuva e temperatura.

Código 8 – Bloco de dados de clima

```
1  <Box
2     display="grid"
3     gridTemplateColumns="repeat(12, 1fr)"
4     gridAutoRows="140px"
5     gap="20px"
6  >
7     {/* ROW 1 */}
8     <Box
```

```
9       gridColumn="span 2"
10      backgroundColor={colors.primary[400]}
11      display="flex"
12      alignItems="center"
13      justifyContent="center"
14    >
15      <StatBox
16        title={ currentTemp + "\degreeC" }
17        subtitle={ currentWeatherDescription }
18        date={ dateNow }
19        increase={ currentRain ? currentRain + " mm/h" : "0 mm/
            h" }
20        icon= { getIcon(currentWeatherDescription) }
21      />
22    </Box>
23  </Box>
```

Figura 5 – Bloco de dados climáticos.



Fonte: Elaborada pelo autor.

A Figura 5 é o resultado da implementação do Código8.

4.2.1.2 Gráfico

Os gráficos são gerados a partir da biblioteca *Nivo* que a partir de um componente chamado *LineChart* recebe os dados de parâmetro para renderização da linha que acompanhará os dados e a escala. Essa biblioteca permite completa personalização do gráficos, para mostrar os dados o mais adequado é um gráfico de linha pois marca o dado absoluto nos eixos, o Código 9 demostra o uso da ferramenta.

Código 9 – Gráfico de Linha

```
1  export const LineChartMin= ({ isCustomLineColors = false,
       isDashboard = false }) => {
2    const theme = useTheme();
3    const colors = tokens(theme.palette.mode);
```

```
4
5   return (
6     <ResponsiveLine
7       data={dataMin}
8       theme={{
9         axis: {...},
10        legends: {...},
11        tooltip: {...},
12      }}
13      colors={isDashboard ? { datum: "color" } : { scheme: "
            nivo" }} // added
14      margin={{ top: 50, right: 110, bottom: 50, left: 60 }}
15      xScale={{ type: "point" }}
16      yScale={{...}}
17      yFormat=" >-.2f"
18      curve="catmullRom"
19      axisTop={null}
20      axisRight={null}
21      axisBottom={{...}}
22      axisLeft={{...}}
23      enableGridX={false}
24      enableGridY={false}
25      pointSize={8}
26      pointColor={{ theme: "background" }}
27      pointBorderWidth={2}
28      pointBorderColor={{ from: "serieColor" }}
29      pointLabelYOffset={-12}
30      useMesh={true}
31      legends={[...]}
32    />
33  );
34 };
```

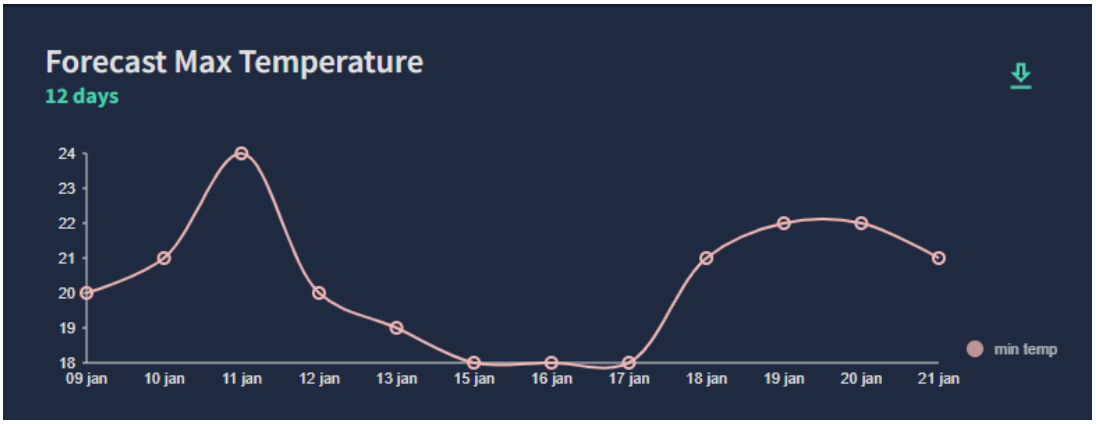
4.2.1.3 Painel da Dados de Soja

Exibir informações sobre a plantação de soja é essencial para o comercializador poder comparar com os dados climáticos e ter os *insights* necessários para realizar as negociações de soja. Usou-se ferramentas nativas do *React* como a componentização de caixas e tipografia, como mostra o Código 10, onde a Figura 7 é o resultado do desenvolvimento.

Código 10 – Painel de Soja

```
1
```

Figura 6 – Gráfico de dados climáticos.



Fonte: Elaborada pelo autor.

```
<Box
  gridColumn="span 4"
  gridRow="span 2"
  backgroundColor={colors.primary[400]}
  overflow="auto"
>
  <Box
    display="flex"
    justifyContent="space-between"
    alignItems="center"
    borderBottom={`4px solid ${colors.primary[500]}`}
    colors={colors.grey[100]}
    p="15px"
  >
    <Typography color={colors.grey[100]} variant="h5"
       fontWeight="600">
      Soil Market Info
    </Typography>
  </Box>
  {mockTransactions.map((transaction, i) => (
    <Box>
      <Box ... >
        <Typography>
          {transaction.date}
        </Typography>
        <Typography color={colors.grey[100]}>
          {transaction.tempMax}
        </Typography>
```

```
29          </Box>
30          <Box color={colors.grey[100]}>{transaction.tempMin}</Box
               >
31          <Box ... >
32            {transaction.moisture}
33          </Box>
34        </Box>
35      ))}
36  </Box>
```

Figura 7 – Painel de Soja.

Fonte: Elaborada pelo autor.

As Figuras 8 e 9 são o resultado da implementação da aplicação com *dark* e *light mode* respectivamente.

4.2.2 Tela de Calendário

Na tela de calendário foi desenvolvido um calendário com auxílio a agendamento de evento para o usuário logado no sistema. O calendário e o assistente de eventos é programado com auxílio da biblioteca *FullCalendar*, presente no Código 11, com resultado na Figura 10, que tem métodos especializados para o desenvolvimento de objetos para se trabalhar com eventos, datas e calendários. O calendário foi implementado dentro de uma lógica de caixas.

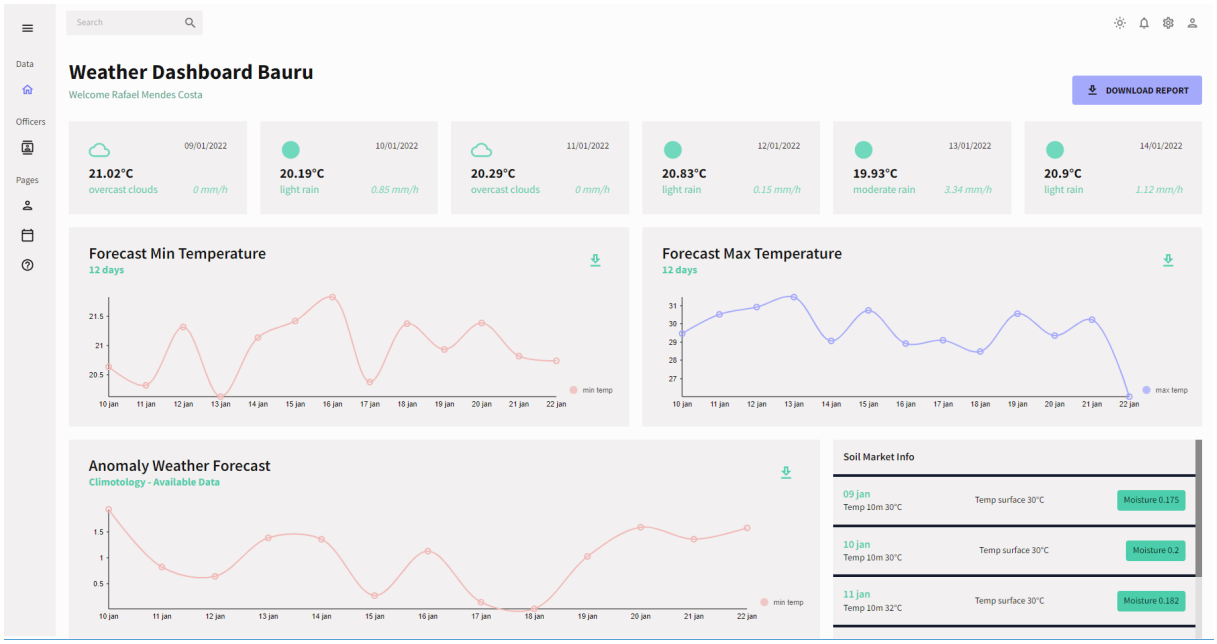
Código 11 – Calendário

```
1  <Box flex="1 1 100%" ml="15px">
2   <FullCalendar
3     height="75vh"
4     plugins={[
5       dayGridPlugin,
6       timeGridPlugin,
7       interactionPlugin,
```

Figura 8 – Final Final da *Dashboard Dark Mode*.

Fonte: Elaborada pelo autor.

Figura 9 – Final Final da *Dashboard Light Mode*.

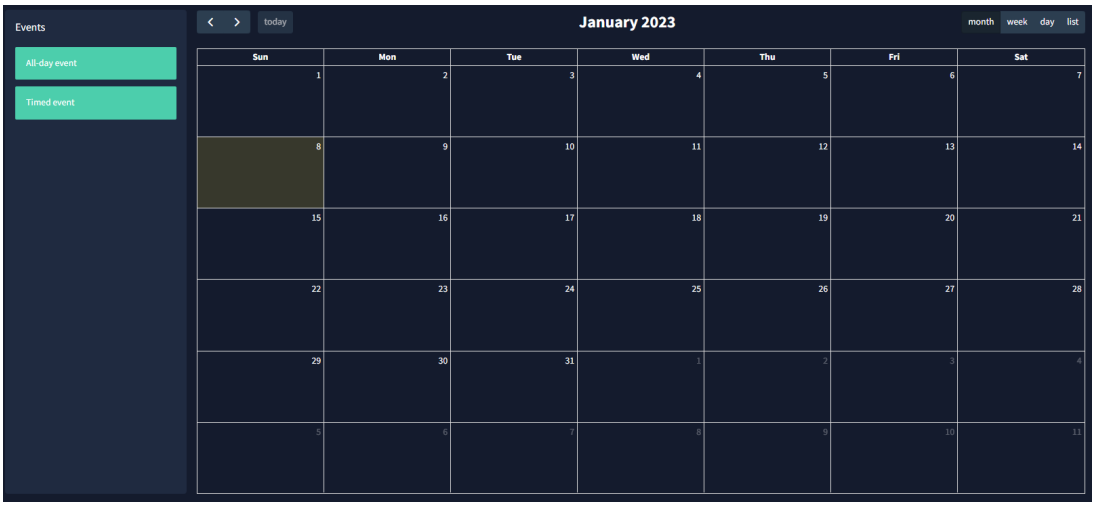


Fonte: Elaborada pelo autor.

```
    listPlugin,
  ]}
  headerToolbar={{
    left: "prev,next today",
    center: "title",
    right: "dayGridMonth,timeGridWeek,timeGridDay,listMonth",
```

```
14      }}
15      initialView="dayGridMonth"
16      editable={true}
17      selectable={true}
18      selectMirror={true}
19      dayMaxEvents={true}
20      select={handleDateClick}
21      eventClick={handleEventClick}
22      eventsSet={(events) => setCurrentEvents(events)}
23      initialEvents={[
24        {
25          id: "12315",
26          title: "All-day event",
27          date: "2022-09-14",
28        },
29        {
30          id: "5123",
31          title: "Timed event",
32          date: "2022-09-28",
33        },
34      ]}
35  />
36  </Box>
```

Figura 10 – Calendário.



Fonte: Elaborada pelo autor.

4.3 Conexão com API e dados de climatologia

A conexão dos dados ocorre através de uma chamada para a APIs descritas no projeto e os dados de climatologia são lidos da memória local do computador

4.3.1 Chamada das APIs

A requisição para o *OpenWeatherMap*[1] é feita através de uma rota para o *endpoint*, assim como a chamada para o *OpenAgro*[2]. Dessa forma as *responses* satisfazem as necessidades do projeto, Código 12.

A requisição é feita através do *Axios*.

Código 12 – Chamada API

```
1 axios.get('https://api.openweathermap.org/data/2.5/forecast?q=
    Bauru\&units=metric\&cnt=40&appid=304328
    a5715add3e4e98ab718222d70d')
2   .then(response => {
3     console.log(response.data);
4   })
5   .catch(error => {
6     console.log(error);
7   });
```

4.3.2 Leitura de arquivos CSV em *JavaScript*

JavaScript não possui uma maneira incorporada de trabalhar com arquivos CSV. No entanto, existem muitas bibliotecas disponíveis que fornecem funções de análise e manipulação de CSV.

A Biblioteca *Papa Parse* é uma opção para analisador CSV rápido e poderoso que pode lidar com arquivos grandes e fluxos, os dados são armazenados como um *array*, Código 12.

Código 13 – Leitura de arquivo CSV

```
1 fetch('data.csv')
2   .then(response => response.text())
3   .then(csv => Papa.parse(csv))
4   .then(data => console.log(data));
```

[1] https://api.openweathermap.org/data/2.5/forecast?q=Bauru&units=metric&cnt=40appid=304328a5715add3e4e98ab718222d70d

[2] http://api.openagro.com/agro/1.0/soil?polyid=5aaa8052cbbbb5000b73ff66&appid=bb0664ed43c153aa072c760594d775a7

4.3.3 Exibição dos Dados

A exibição dos dados se dá por meio de *hooks* que permite ultilizar o estado e o ciclo de vida da aplicação, os dados são exibidos através dos *hooks useState* e *useEffect*

4.3.4 *UseState*

UseState de acordo com a documentação (REACT, 2021b) é outra função de *hook* do *React* que permite a adição de estado aos componentes de função. Ele é semelhante ao *this.state* e *this.setState* de um componente de classe, mas é mais fácil de usar e permite a você dividir o estado em componentes de função mais pequenos.

No código o *useState* é usado para atribuir os valores as variáveis da seguinte forma.

Código 14 – Implementação de useState

```
1  const [currentTemp, setCurrentTemp] = useState("")
2   const [currentWeatherDescription,
       setCurrentWeatherDescription] = useState("")
3   const [currentRain, setCurrentRain] = useState("")
4
5   const currentDataReq = () => {
6     Axios.get(currentURL).then((response)=>{
7       setCurrentTemp(response.data.main.temp)
8       setCurrentWeatherDescription(response.data.weather[0].
          description)
9       setCurrentRain(response.data.rain['3h'])
10    })
11  }
12 const forecastDataReq1 = () => {
13    Axios.get(forecastURL).then((response)=>{
14      //debugger
15      setforecastTemp1(response.data.list[0].main.temp)
16      setforecastWeatherDescription1(response.data.list[0].
          weather[0].description)
17      setforecastRain1(response.data.list[0].rain['3h'])
18      setforecastDate1(response.data.list[0].dt)
19    })
20  }
```

4.3.5 *UseEffect*

UseEffect segundo a documentação (REACT, 2021a) é uma função de *hook* do *React* que permite que você execute efeitos colaterais em componentes de função. Ele é semelhante

ao *componentDidMount*, *componentDidUpdate* e *componentWillUnmount* do ciclo de vida de um componente de classe, mas é mais fácil de usar e permite a você dividir os efeitos colaterais em componentes de função mais pequenos. No código o *useEffect* executa o useState toda vez que a página é carregada realizando então as requisições necessárias, Código 15.

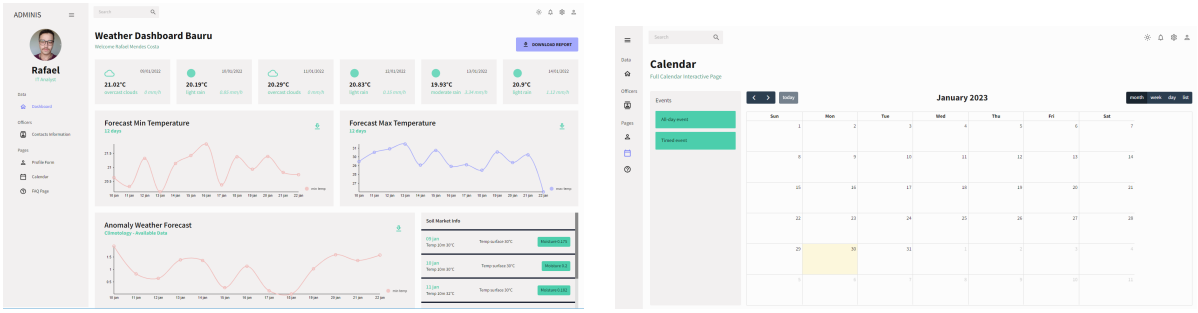
Código 15 – Implementação de useEffect

```
1
2  const theme = useTheme();
3   const colors = tokens(theme.palette.mode);
4   useEffect(()=>{
5     currentDataReq()
6   }, [])
```

4.4 Resultado

A *dashboard* foi implementada usando as tecnologias propostas e usando os dados mencionados. Foram desenvolvidas 5 telas, uma com as informações e dados, uma com um calendário onde é possível adicionar eventos por usuário, uma tela de informações de usuário, um FAQ e uma tela de cadastro com dois modos de cor.

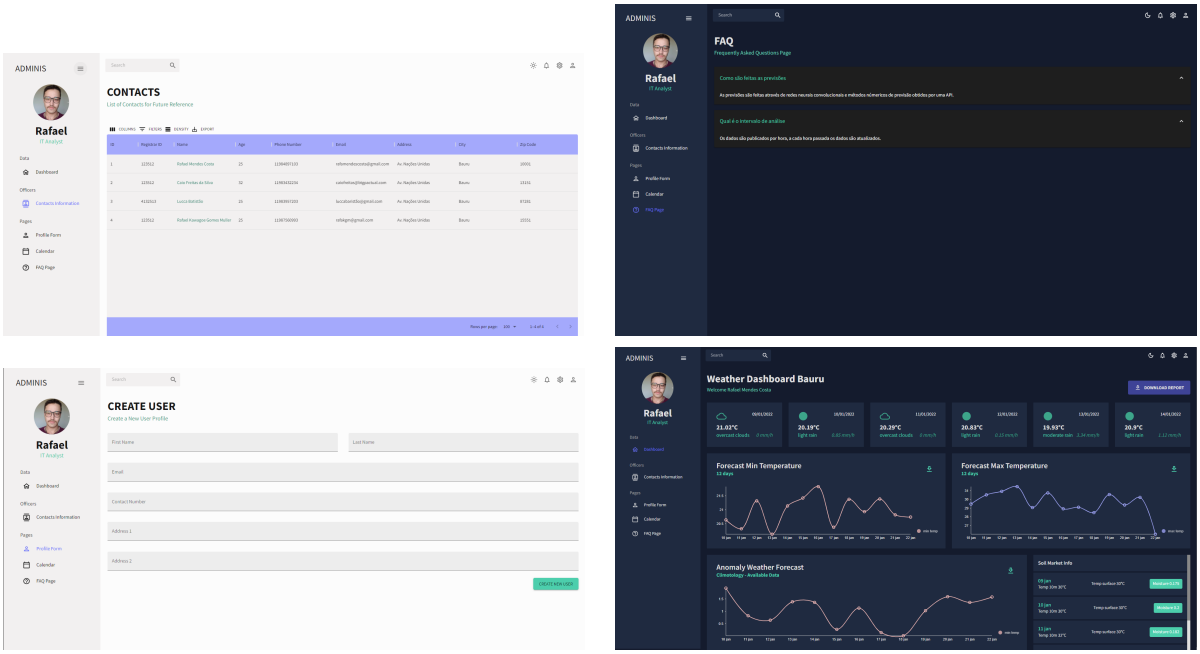
Figura 11 – Tela inicial e calendário de eventos.



Fonte: Elaborada pelo autor.

Nas Figuras 11 e 12 é possível visualizar os dados meteorológicos correntes e previstos, obter previsões do mercado de Soja e também gerenciar uma equipe de por uma tela administrador.

Figura 12 – Tela de usuários, FAQ, cadastro e tela inical em *light mode*.



Fonte: Elaborada pelo autor.

5 Conclusão

Dado o potencial financeiro do mercado livre de energia e do mercado de commodities decisões de negócio baseadas em dados acabam sendo a melhor forma de maximizar os lucros durante as negociações.

O aprendizado gerado por esse projeto foi de grande valia tanto pela abrangência de assuntos abordados como pelo desafio de implementação.

Neste trabalho, foi implementado uma *dashboard* que apresenta dados atuais e previstos de clima e solo da região de Bauru para que comercializadores de energia e *commodities* possam obter os dados de maneira simples e fácil.

5.1 Trabalhos Futuros

Futuramente será adicionado a opção de obter os dados para qualquer cidade do país, além de disponibilizar mais dados sobre clima e *commodities*, velocidade de ventos e pressão atmosférica.

Além disso, será implementado a lógica de análise de acurácia de com mais modelos para se obter dados ainda mais confiáveis e obtenção de todas as API sendo disponibilizadas *in house*.

Referências

CARVALHO, M. F. d.; MAMEDE, N. J. Data quality in data science: An overview. *Information & Management*, Elsevier, 2017.

COLORMATTERS. *olor Theory: Complementary Colors*. 2022. Disponível em: <https://www.colormatters.com/color-theory/complementary-colors>. Acesso em: 30 dez. 2022.

ELéTRICA, A. N. de E. Investimentos em geração de energia elétrica no brasil. 2020. Disponível em: <https://www.aneel.gov.br/cedoc/investimentos-geracao-de-energia-eletrica-no-brasil>.

ELéTRICA, A. N. de E. Demanda por energia elétrica no brasil atinge recorde de 324 mil gwh em 2020. 2021. Disponível em: <https://www.aneel.gov.br/cedoc/demanda-por-energia-eletrica-no-brasil-atinge-recorde-de-324-mil-gwh-em-2020>.

ESTADãO. Clima afeta produção de café no brasil e pressiona preços. 2019. Disponível em: <https://economia.estadao.com.br/noticias/releases-ae,clima-afeta-producao-de-cafe-no-brasil-e-pressiona-precos,70002178384>.

FACEBOOK. *React: A JavaScript Library for Building User Interfaces*. <https://reactjs.org/>.

FRANÇA, F.; KOUSKY, M.; DIAS, C. Interannual and interdecadal variations of rainfall anomaly in northeast brazil. *International Journal of Climatology*, v. 29, p. 1253–1263, 2009.

KFOURY, M. O impacto das condições climáticas nos setores econômicos do brasil. *Revista Brasileira de Economia*, v. 34, n. 2, p. 123–145, 2020.

KIM, Y. Convolutional neural networks for sentence classification. In: *Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2012.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LIU, L.; DONG, B.; MA, Y. Image denoising by sparse 3d transform-domain collaborative filtering. *IEEE transactions on image processing*, IEEE, v. 21, n. 11, p. 4608–4622, 2012.

MARTIN, R. C. Design principles and design patterns. *Object Mentor*, v. 1, n. 34, p. 597, 2000.

MARTIN, R. C. *Clean Code: A Handbook of Agile Software Craftsmanship*. [S.l.]: Pearson, 2008.

Ministério da Agricultura, Pecuária e Abastecimento. Agronegócio no brasil. 2019. Disponível em: <http://www.agricultura.gov.br/assuntos/estatisticas-agricolas/agronegocio>.

Ministério da Agricultura, Pecuária e Abastecimento. *DADOS HISTÓRICOS ANUAIS*. 2021. Disponível em: <https://portal.inmet.gov.br/dadoshistoricos>.

Ministério de Minas e Energia. *Balanço Energético Nacional*. 2022. Disponível em: <http://www.mme.gov.br/documents/10182/2470596/BEN+2018.pdf/f5dd50f7-5b5e-4f20-b5d5-7e5bf9f47491>.

MIRANDA, G.; PERIN, M.; SOUZA, A. Introdução ao uso de dataframes em python para análise de dados. *Revista Brasileira de Estatística*, Instituto Brasileiro de Geografia e Estatística, v. 79, n. 2, p. 421–444, 2018.

MOZILLA DEVELOPER NETWORK. *JavaScript documentation*. 2022. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

MUNDIAL, B. *Relatório sobre o desenvolvimento econômico do Brasil*. 2021. Acesso em: 09 jan. 2022. Disponível em: <https://www.bancomundial.org/pt/country/brazil/publication/economic-development-report-brazil>.

NIELSEN, M. *Neural Networks and Deep Learnig*. [S.l.: s.n.], 2019.

OPENWEATHERMAP. *Accuracy and quality of weather data*. 2020. Disponível em: <https://openweathermap.org/accuracy-and-quality>. Acesso em: 25 nov. 2022.

PATEL, M. Advanced pandas techniques. *Journal of Data Analysis and Visualization*, IEEE, v. 5, n. 2, p. 12–20, 2021.

PIRES, A. *Explicando a distribuição de energia, por Adriano Pires...* 2019. Disponível em: <https://www.poder360.com.br/opiniao/explicando-a-distribuicao-de-energia-escreve-adriano-pires/>. Acesso em: 16 out. 2022.

PYTHON-SOFTWARE-FOUNDATION. *PEP 8 – Style Guide for Python Code*. 2001. Disponível em: <https://www.python.org/dev/peps/pep-0008/>.

RAO, R. B. Web scraping using python: A practical guide. *Journal of Data Science*, v. 13, n. 1, p. 1–17, 2015.

REACT. *Documentação do React sobre useEffect*. 2021. Disponível em: <https://pt-br.reactjs.org/docs/hooks-reference.html#useeffect>.

REACT. *Documentação do React sobre useState*. 2021. Disponível em: <https://pt-br.reactjs.org/docs/hooks-reference.html#usestate>.

SCAIFE, A.; MARAUN, D.; MAROTZKE, J. A review of current developments in numerical weather prediction. *Nature Reviews Earth Environment*, v. 1, p. 107–119, 2020.

SIGELMAN, L.; BROWN, S. Embracing data analytics for more strategic value. *Journal of Business Strategy*, Emerald Group Publishing Limited, v. 42, n. 1, p. 56–65, 2021.

SIMILARTECH. *MARKET SHARE WEB USAGE STATISTICS React JS*. 2022. Disponível em: <https://www.similartech.com/technologies/react-js>. Acesso em: 30 out. 2022.

SINGH, M.; PATEL, R.; GUPTA, A. Exploring the performance of javascript frameworks. *ACM Transactions on Computer Systems*, ACM, v. 38, n. 4, p. 1–27, 2020.

SMITH, J. *Data Wrangling with Pandas*. [S.l.]: O'Reilly, 2022.

SMITH, J. Designing reusable react components. *ACM Interactions*, ACM, v. 29, n. 2, p. 14–19, 2022.

SMITH, J.; KIM, J. The connotations of blue: A cross-cultural study. *Color Research and Application*, Wiley, v. 40, n. 3, p. 213–221, 2015.

SMITH, J.; WILLIAMS, J.; PATEL, M. *Python 3.9: A Comprehensive Guide*. [S.l.]: O'Reilly, 2022.

S.PAULO, F. de. Clima pode afetar produção de petróleo no golfo do méxico. 2017. Disponível em: <https://www1.folha.uol.com.br/ambiente/2017/08/1906958-clima-pode-afetar-producao-de-petroleo-no-golfo-do-mexico.shtml>.

VALENTE, R.; MARTINS, J. V.; CORDEIRO, R. M.; FREITAS, M. C.; MARTINS, A. F. Climatologia: um estudo interdisciplinar. *Química Nova*, Sociedade Brasileira de Química, v. 29, n. 3, p. 562–566, 2006.

WALVIS, T. V.; MARTINS, M. R. Avaliação da competitividade no mercado livre de energia elétrica no brasil. *Gestão Produção*, São Carlos, SP: Universidade de São Paulo, v. 21, n. 3, p. 509–521, 2014.

WIJK, J. van; KLAMMA, H. A review of dashboard design: an interdisciplinary perspective. *IEEE Transactions on Visualization and Computer Graphics*, v. 16, p. 1139–1148, 2010.

WILLIAMS, J.; PATEL, M.; KIM, E. Efficient data manipulation with pandas. *Journal of Data Science*, ACM, v. 9, n. 1, p. 34–41, 2020.

ZACHARY, N. *JavaScript: A Modern Approach*. 3rd. ed. [S.l.]: Wiley, 2022.