Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Sistema de previsão climática baseado em *Machine Learning*

Autor: Guilherme de Oliveira Aguiar, Victor Rodrigues Silva
Orientador: Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF
2022

Guilherme de Oliveira Aguiar, Victor Rodrigues Silva

Sistema de previsão climática baseado em *Machine Learning*

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF

2022

Guilherme de Oliveira Aguiar, Victor Rodrigues Silva
Sistema de previsão climática baseado em *Machine Learning*/ Guilherme de Oliveira Aguiar, Victor Rodrigues Silva. – Brasília, DF, 2022-
117 p. : il. (algumas coloridas) ; 30 cm.

Orientador: Dr. Vandor Roberto Vilardi Rissoli

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2022.

1. *machine learning*. 2. meteorologia. I. Dr. Vandor Roberto Vilardi Rissoli. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Sistema de previsão climática baseado em *Machine Learning*

CDU 02:141:005.6

Guilherme de Oliveira Aguiar, Victor Rodrigues Silva

Sistema de previsão climática baseado em *Machine Learning*

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 04 de Novembro de 2022.

Dr. Vandor Roberto Vilardi Rissoli
Orientador

Dr. André Barros de Sales
Convidado 1

Dr. Maurício Serrano
Convidado 2

Brasília, DF
2022

Resumo

Desde a antiguidade, com os babilônios, a humanidade tem buscado entender como o clima funciona e como realizar previsões do tempo, com várias finalidades, como planejamento de agricultura e de riscos. Com o passar dos anos, esse tema sofreu um grande crescimento em sua relevância. Com o envolvimento de novas tecnologias, tem sido desenvolvidas novas formas de previsões mais assertivas. Este trabalho propõe o desenvolvimento de uma aplicação capaz de realizar previsões climáticas com técnicas de aprendizado de máquina. Com a funcionalidade de fornecer alarmes aos seus usuários, além de possibilitar a consulta climática sob demanda, proporcionando aos usuários uma percepção mais segura sobre a situação climática, que poderá contribuir com tomadas de decisões acerca de atividades que podem ser afetadas pelo clima. Assim, trata-se de uma pesquisa aplicada, com pesquisa bibliográfica e produção tecnológica com base nas metodologias de aprendizado de máquina e desenvolvimento de software. Dessa forma, foi realizada uma comparação entre várias técnicas de previsão e foi construído um modelo de *machine learning* para previsão climática que alimenta uma aplicação web em que seu usuário tem acesso ao histórico e a situação meteorológica das horas seguintes.

Palavras-chave: Previsão de Chuva; Meteorologia; Aplicativo; *Machine Learning*; *Deep Learning*.

Abstract

Since ancient times, with the Babylonians, humanity has been pursuing to understand how the weather works and how to make weather forecasts for various purposes, such as agriculture and risk planning. Over the years, this topic has been seeing a great increase in its relevance. With the development of new technologies, forecasts have become more assertive. This work proposes the development of an application capable of performing weather forecasts with machine learning techniques. It has the objective of contributing even more. It will be possible to transmit alarms to the user and enable the weather query on demand, providing the user with a more secure perception of the climate situation in order to contribute to the decisions that may be taken that has the climate as important. Thus, this work is applied research, with bibliographic research and technological production based on Machine Learning and development of a software. Therefore, a comparison was made between several forecasting techniques and a *machine learning* model was built to forecast rainfall that feeds a web application in which the user has access the weather history and the weather situation of the following hours.

Key-words: Rainfall Prediction; Meteorology; App; Machine Learning; Deep Learning.

Lista de Figuras

Lista de tabelas

Lista de Equações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CASE	*Computer-Aided Software Engineering*
CSV	*Comma Separated Values*
DE-R	Diagrama Entidade-Relacionamento
DLD	Diagrama Lógico de Dados
FGA	Faculdade do Gama
IA	Inteligência Artificial
IAS	Instituto de Estudos Avançados
JSON	*JavaScript Object Notation*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MVC	*Model View Controller*
MVP	Produto Mínimo Viável
ORM	*Object-Relational Mapping*
PNT	Previsão Numérica de Tempo
PVI	Problema de Valor Inicial
PWA	*Progressive Web App*
RHN	Rede Hidrometeorológica Nacional
RMSE	Raiz Quadrada do Erro Médio
RN	Redes Neurais
RNN	*Recurrent Neural Network*
SBM	Sistemas Baseados em Conhecimento
SGBD	Sistema Gerenciador de Banco de Dados

SQL	*Structured Query Language*
SVC	*Support Vector Classifier*
SVM	*Support Vector Machine*
UnB	Universidade de Brasília

Sumário

1 Introdução

A meteorologia e a previsão do tempo são fundamentais para o modo em que a humanidade vive atualmente, como a decisão de como uma pessoa vai se vestir para sair de casa, escolhas de qual será o cultivo na agricultura e até mesmo a previsão de possíveis catástrofes naturais.

Existem inúmeras agências meteorológicas em todo o mundo que são únicas para cada país, no entanto, também existem agências meteorológicas globais que fornecem serviços em escala internacional, semelhante ao papel das Nações Unidas (TEAGUE; GALLICCHIO, 2017).

Segundo Teague e Gallicchio (2017), a meteorologia pode ser definida como o estudo da atmosfera e seus fenômenos. O interesse da humanidade com fenômenos atmosféricos não é recente, existindo vários registros na antiguidade, como os babilônios, que no ano de 1792 a.C. desenvolveram um calendário de 360 dias que correspondia às estações com base no ciclo solar. Eles também tentaram prever mudanças climáticas de curto prazo com base no aparecimento de nuvens e fenômenos óticos, como halos (GRAHAM; PARKINSON; CHAHINE, 2002).

Um dos grandes precursores do que seria a ciência da meteorologia foi Aristóteles. Seu tratado intitulado *Meteorologica* e publicado em 340 a.C. é o mais antigo tratado sobre meteorologia (FRISINGER, 1972). Durante a maior parte da história da humanidade a previsão do tempo foi um empreendimento qualitativo, isto é, com base na observação, sem se atentar a medições estritas. Essa realidade só mudaria mais tarde com invenções de instrumentos de medição como o termômetro, barômetro e o anemômetro (TEAGUE; GALLICCHIO, 2017).

No final da década de 1940, usando um dos primeiros computadores digitais, um progresso significativo em direção à previsões numéricas foi feito por uma equipe de meteorologistas e matemáticos do Instituto de Estudos Avançados (IAS) em Princeton, Nova Jersey. O grupo de cientistas liderados por Jule Charney fez uma série de previsões bem-sucedidas de 24 horas na América do Norte e, a partir de 1950 as previsões numéricas do tempo (PNT) começaram a ser feitas continuamente (GRAHAM; PARKINSON; CHAHINE, 2002).

De acordo com Hakim e Patoux (2017), transformar a previsão do tempo de uma arte qualitativa em uma ciência quantitativa é uma das maiores realizações humanas do século XX.

Com a revolução tecnológica iniciada em 1950, enormes melhorias na precisão da

previsão do tempo ocorreram, particularmente o avanço de computadores e satélites meteorológicos, e a disponibilidade de dados fornecidos por redes coordenadas de observação meteorológica (GRAHAM; PARKINSON; CHAHINE, 2002). Paralelamente iniciava-se a área de Aprendizado de Máquina do inglês *Machine Learning* (ML). Em 1960 Frank Rosenblatt criou a máquina intitulada *Perceptron*, que usava sinais analógicos e discretos e incluiu um elemento de limiar que converteu sinais analógicos em discretos (FRADKOV, 2020).

Além disso, Fradkov (2020) descreve três tendências que explicam por que o início da primeira década do século XXI acabou sendo um ponto de virada na história da ML:

- ***Big Data***: A quantidade de dados tornou-se tão grande que novas abordagens foram trazidas à vida por necessidade prática e não por curiosidade dos cientistas.
- **Redução do custo de computação e memória paralela**: Essa tendência foi popularizada em 2004, quando o Google lançou sua tecnologia MapReduce, seguida por seu equivalente *open source* Hadoop (2006). Juntas, estas tecnologias possibilitaram a distribuição do processamento de grandes quantidades de dados entre processadores simples.
- ***Deep Machine Learning***: Concepção de novos algoritmos de *Deep Machine Learning* que herdam e desenvolvem a ideia do Perceptron em combinação com uma campanha de publicidade científica bem-sucedida.

Machine Learning é uma subárea de Inteligência Artificial (IA), segundo Rosebrock (2017), sendo objetivo central da IA fornecer um conjunto de algoritmos e técnicas que podem ser usados para resolver problemas que os humanos executam intuitivamente e quase automaticamente, mas são desafiadores para computadores. Enquanto isso, o ML tende a estar especificamente interessado em reconhecimento de padrões e aprendizado de dados.

1.1 Problema

A produção de previsões meteorológicas é uma atividade praticada em escala internacional e é relevante para diversos setores da sociedade, como órgãos governamentais, exército, meios de comunicação, indústrias, agricultura e também para o uso pessoal. Essa tarefa movimenta bilhões de dólares mundo afora, fomentando a criação de novas tecnologias como redes de computação e telecomunicações, satélites e sistemas de observação terrestres (INNESS; DORLING, 2012).

O Brasil é um país com um alto índice de desastres naturais, segundo o artigo de Watanabe (2019), pelo menos 116 milhões de pessoas já foram afetadas por desastres

naturais nos últimos 120 anos. Esses desastres vêm sendo ampliados pelos efeitos do aquecimento global, pela pesquisa da empresa *Uswitch*, o Brasil é o 10º país mais afetado por desastres naturais no mundo, com cerca de 13 mil fatalidades entre o período de 1902 e 2021 (GALLIZZI, 2022), sendo que a maior parte dos desastres são enchentes, seguido de deslizamentos, que são geralmente ligados a chuvas (WATANABE, 2019).

A previsão do tempo é tradicionalmente tratada como um problema da física atmosférica, com muitos processos físicos diferentes contribuindo para o resultado final da previsão. Sendo assim, todos os agentes que impactam o resultado de um evento climático devem ser identificados para o melhor resultado da previsão.

Inness e Dorling (2012) define NWP como um modelo computadorizado da atmosfera que, dado um estado inicial para a atmosfera, derivada de observações, pode gerar previsões de como o tempo vai evoluir para o futuro.

Como denota Bjerknes (2009), o problema da previsão numérica pode ser caracterizado como um problema do valor inicial, já que os estados atmosféricos evoluem a partir dos estados antigos seguindo as regras da física. Enquanto previsões de 1 a 2 dias tendem ser mais precisas, à medida que o tempo aumenta, a confiabilidade da previsão diminui. Isso se deve em parte por que, como reconhece Hakim e Patoux (2017), não há como saber exatamente como todas as variáveis estão nas condições iniciais. Assim uma pequena imprecisão nas grandezas atmosféricas do estado causa um erro no cálculo. Basicamente, esses pequenos erros crescem com o tempo para se tornarem grandes erros.

Além disso, Hakim e Patoux (2017) apontam outra dificuldade das previsões numéricas: a representação do modelo da atmosfera também é imperfeita, assim, não é possível representar todas as escalas de movimento exatamente. Certos processos, como a microfísica de nuvens, ocorrem em escala molecular e não podem ser resolvidos explicitamente. A radiação eletromagnética é importante para a evolução da temperatura, mas não se pode simular fótons individuais.

Sendo assim, uma pequena perturbação na atmosfera pode resultar em uma evolução muito diferente dos sistemas climáticos ao seu redor. O exemplo mais citado disso é que uma borboleta batendo as asas no Brasil pode causar um tornado no Texas, o que significa que todas as previsões numéricas do tempo que serão feitas estão fadadas a se tornarem erradas em algum momento (INNESS; DORLING, 2012).

Isso não se deve às limitações humanas em observar ou modelar a atmosfera, é devido à própria natureza da atmosfera. No final, a amplificação dos erros iniciais se deve à natureza caótica da atmosfera, que é descrita pela teoria do caos (HAKIM; PATOUX, 2017). Nesse contexto, caos significa que a previsão do estado da atmosfera em tempos futuros depende sensivelmente dos detalhes da atmosfera no momento inicial da previsão.

O aprendizado de máquina (ML), pelo contrário, é relativamente resistente a per-

turbações, e não requer uma total compreensão dos processos físicos que governam a atmosfera. Portanto, o aprendizado de máquina pode representar uma alternativa viável aos modelos físicos de previsão do tempo (HOLMSTROM; LIU; VO, 2016).

Com a crescente disponibilidade de *big data* meteorológico, os pesquisadores perceberam que a introdução de abordagens baseadas em dados na meteorologia pode alcançar um sucesso considerável, visto que vários métodos de aprendizado de máquina foram aplicados à previsão do tempo (WANG et al., 2018).

Tendo em vista as dificuldades que o Brasil vem enfrentando com desastres naturais, muitos deles relacionados com chuvas e as dificuldades que foram apresentadas ao desafio que é a previsão do tempo além da baixa quantidade de estações meteorológicas existentes no país e a existência de esforços executados com o objetivo de resolver o problema usando técnicas de ML. Percebe-se a relevância da construção de uma aplicação de previsão que utiliza um modelo de aprendizado de máquina, utilizando dados históricos para prever a condição atmosférica de chuvas, tendo como base variáveis temporais próprias do escopo do problema, como precipitação, temperatura, umidade e pressão atmosférica.

1.2 Questão de Pesquisa

Com o objetivo de nortear o desenvolvimento deste trabalho, a seguinte questão de pesquisa foi proposta:

Como, diante das limitações dos modelos de previsão numérica do tempo frente às alterações atmosféricas e perturbações, elaborar uma aplicação de previsão de chuvas, utilizando técnicas de aprendizado de máquina?

1.3 Justificativa

Este trabalho reconhece a importância da previsão do tempo para o funcionamento ideal dos vários setores da sociedade, sendo determinante para tomadas de decisão e assistência à vigilância civil e outros órgãos governamentais. Desta forma, a criação de uma aplicação de previsão do tempo empregando técnicas de aprendizado de máquina para remediar as limitações das técnicas tradicionais, que envolvem modelos matemáticos para simulação da atmosfera, consistiria em uma aplicação relevante à realidade da sociedade.

Se realizado com sucesso, a aplicação de previsão climática irá proporcionar mais informação para o usuário tomar decisões e se preparar, ajudando na segurança e previsibilidade nas situações cotidianas.

Apesar de ser uma atividade muito frutuosa, que recebe muita atenção dos setores

público e privado mundo afora, a dificuldade de representar a atmosfera fielmente e realizar previsões de longo prazo confiáveis ainda é um desafio a ser superado. Segundo o relatório da Organização Meteorológica Mundial e do Escritório da Organização das Nações Unidas para a Redução do Risco de Desastres (WMO, 2019), os 10 maiores desastres registrados na América Latina foram responsáveis por 60% das 34.854 vidas perdidas e 38% das perdas econômicas equivalentes a US$ 39,2 bilhões.

O artigo do meteorologista Haby (2015) também cita alguns benefícios da previsão do tempo, sendo eles:

- Auxilia empresas e pessoas a planejar a produção de energia e quanta energia usar (ou seja, empresas de energia, onde definir o termostato);
- Ajuda as pessoas a se prepararem em caso de necessidade de equipamento extra para se proteger do clima (por exemplo, guarda-chuva, capa de chuva, protetor solar);
- Assiste as empresas a planejar os riscos de transporte que podem resultar do clima (ou seja, neblina, neve, gelo, tempestades, nuvens no que se refere a dirigir e voar, por exemplo);
- Auxilia as pessoas com problemas de saúde a planejar o dia (por exemplo, alergias, asma, estresse térmico);
- Ajuda os agricultores e jardineiros a planejar a irrigação e proteção de culturas (programação de irrigação, proteção contra congelamento);
- Assiste pessoas envolvidas com certas atividades a saber se as condições serão boas (por exemplo, esqui, passeios de barco, balonismo etc.).

A partir da listagem de alguns dos benefícios fica visível a importância da previsão do tempo inclusive no dia a dia dos indivíduos, isto é, nas tarefas cotidianas.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma ferramenta automatizada capaz de realizar previsões de chuva.

1.4.2 Objetivos Específicos

- Estruturar uma base de dados composta com variáveis atmosféricas relevantes ao processo de análise de precipitação.

- Desenvolver um modelo de aprendizado de máquina que seja capaz de "aprender" a partir de uma base confiável de dados e contribuir com predições com um grau de segurança adequado.

- Prover ao indivíduo uma percepção sobre a situação climática (ou do tempo) cotidiana de um local ou região onde ele vive ou estará, a fim de contribuir com as decisões que poderá vir a tomar.

1.5 Metodologias

A elaboração deste tipo de trabalho emprega duas categorias de metodologias a serem abordadas mais detalhadamente nesta dissertação, sendo elas a metodologia de pesquisa e a metodologia de desenvolvimento do sistema computacional (software).

As metodologias de pesquisa, segundo Gerhardt e Silveira (2009), podem ser classificadas de várias formas, sendo elas quanto à forma de abordagem do problema, sua natureza, seus objetivos e seus procedimentos. Quanto à abordagem, neste trabalho será utilizada uma abordagem quantitativa. Quanto à natureza, será uma pesquisa aplicada, quanto ao objetivo, será do tipo exploratória e finalmente, quanto aos procedimentos, será do tipo pesquisa-ação e bibliográfica.

Já quanto à metodologia de desenvolvimento, serão utilizadas algumas técnicas do SCRUM, como *sprints* iterativos na construção do software e elaboração de histórias de usuário para compor o *backlog* do produto. Aliado aos métodos ágeis, também será adotada a metodologia Kanban. Todas as etapas dessa metodologia serão abordadas com maior profundidade no Capítulo 3.

1.6 Organização do trabalho

O trabalho foi divido em quatro capítulos, sendo apresentado uma síntese de cada um deles a seguir:

1. Capítulo 1 - Introdução: refere-se à contextualização, levantamento da questão de pesquisa, justificativa, definição dos objetivos principais do trabalho e a indicação das metodologias;

2. Capítulo 2 - Referencial Teórico: abrange os conhecimentos relevantes para a compreensão dos principais conceitos e recursos relacionados a pesquisa proposta e sua contextualização. Este capítulo é subdividido nas seções de meteorologia e previsão do tempo, técnicas de aprendizado de máquina (ML) e metodologias de desenvolvimento de software;

3. Capítulo 3 - Metodologia: este capítulo apresenta as metodologias que foram empregadas na elaboração deste trabalho, além de relacionar as ferramentas e tecnologias usadas para o auxílio no desenvolvimento da pesquisa e na implementação do trabalho proposto;

4. Capítulo 4 - Desenvolvimento: retrata a execução planejada no capítulo anterior (3), em que o processo de desenvolvimento é detalhado, junto com as dificuldades e alterações que ocorreram.

5. Capítulo 5 - Considerações Finais: apresenta a conclusão deste trabalho, a partir do planejamento, desenvolvimento e análise de resultados, juntamente com possíveis trabalhos futuros que poderão ser originados a partir de novas abstrações e requisitos, além do desenvolvimento evolutivo sobre o que foi elaborado por este TCC.

2 Referencial Teórico

Este capítulo tem o objetivo de apresentar os principais conceitos e as informações dos quais este trabalho e sua proposta de solução foram fundamentados.

2.1 Meteorologia

A Meteorologia é a ciência que estuda os fenômenos da atmosfera e suas constantes mudanças, e esse estudo é de essencial importância para a humanidade de hoje (AHRENS, 2001). Como bem afirma Rigutti (2002), a Meteorologia é a ciência que prevê o tempo e o clima através dos estudos de diversos aspectos da atmosfera e seus fenômenos.

Como assegura Ynoue et al. (2017, p. 6), a "Meteorologia é a ciência que estuda os processos físicos, químicos e dinâmicos da atmosfera e as interações desses processos com os sistemas litosfera, hidrosfera, criosfera e biosfera".

A maneira com que a meteorologia influencia a vida do ser humano atualmente é fundamental para a tomada de decisões da humanidade, desde coisas simples como decidir se vai ou não levar um guarda-chuvas para o trabalho, ou decisões de segurança pública como voos comerciais ou até mesmo a previsão de desastres causados por fenômenos atmosféricos, como tempestades, nevascas e até furacões.

2.1.1 Previsão do Tempo

Para compreender melhor a Previsão do Tempo é preciso compreender a diferença entre tempo atmosférico e clima. O tempo é a situação atmosférica em um dado momento em uma área determinada, sendo essa situação a combinação de algumas variáveis observadas. Já o clima é o comportamento atmosférico do tempo de uma determinada região em um longo período (MOREIRA; SENE, 2016).

A previsão do tempo é produzida no mundo inteiro, com o propósito de uso geral, militar, governamental, produção de notícias, uso comercial e industrial com uso nacional e internacional. Como assegura Hakim e Patoux (2017), o tempo afeta o dia a dia da humanidade, tanto pessoalmente, como em sociedade, desde as roupas e o transporte a ser utilizado, ou mesmo as atividades durante o dia que não estão diretamente ligadas a tomadas de decisões das pessoas, como engarrafamentos, alagamentos, quedas de energia e até mesmo grandes desastres.

Com sua grande importância na sociedade, novos desenvolvimentos na área de previsão do tempo acontecem o tempo todo, isso pode ser verificado nos artigos e jornais

meteorológicos que possuem constantes publicações e inovações como contam Innes e Beasley (2013).

De acordo com Innes e Dorling (2013), para iniciar os estudos da atmosfera e conseguir realizar previsões do tempo, primeiramente tem-se que realizar observações, levantar perguntas e depois tentar respondê-las. Nessas observações são coletados dados chamados de variáveis, e com essas variáveis são feitos os estudos capazes de realizar a previsão do tempo.

De acordo com Moreira e Sene (2016), o tempo corresponde a combinação de algumas variáveis, como temperatura, umidade, pressão do ar, ventos e nebulosidade, essas que podem mudar constantemente.

Como assegura Innes e Dorling (2013), as quatro principais variáveis são temperatura, vento, precipitação e a pressão atmosférica. Essas quatro variáveis são constantemente observadas e de acordo com as suas mudanças, e de outras variáveis, seguindo as leis da física é possível realizar estudos para predição do tempo.

2.1.1.1 Temperatura

A temperatura é uma das variáveis mais importantes do tempo, de acordo com Moreira e Sene (2016). Ela corresponde a intensidade de calor presente na atmosfera, estando relacionada com a intensidade de radiação solar que chega nas superfícies e que são irradiadas. Já Hakim e Patoux (2017) ressalta que a temperatura é aferida de maneira padrão, para que possam ser feitos mapeamentos e comparações. As medições são sempre realizadas na sombra, em um contêiner, elevado a 2 metros de altura acima de uma área com vegetação. Este contêiner é ventilado, pintado de branco, para que haja a menor interferência possível de radiação solar direta e o calor da superfície do solo.

2.1.1.2 Pressão Atmosférica e Vento

De acordo com Moreira e Sene (2016), a pressão atmosférica é o peso que a massa de ar exerce sobre uma superfície. Esse ar, de acordo com a temperatura, é mais denso quando está frio, fazendo com que o ar frio deixe a pressão da atmosfera mais alta, sendo inverso quando está quente. Mas estas parcelas de ar quente e frio estão sempre em constante troca de calor entre elas, fazendo com que suas moléculas de ar se movimentem à medida que mudam de temperatura, fazendo assim com que o vento seja formado.

Como assegura Ynoue et al. (2017, p. 56-68), a pressão atmosférica equivale à pressão exercida pelo peso da coluna de ar sobre uma dada superfície, ou seja, representa o peso que a atmosfera exerce por unidade de área. Hakim e Patoux (2017) dizem que, para quantificar essa pressão, é utilizado o peso acumulado de uma parcela de ar em determinada área e verifica-se o peso aplicado a superfície através de um barômetro. Como

a pressão está diretamente ligada a temperatura das moléculas de ar, e sua constante troca de calor, provoca a movimentação das moléculas e consequentemente resultam no fenômeno chamado vento.

O vento é medido de acordo com sua velocidade. Para realizar tal medição é preciso analisar dois fatores, sua direção e intensidade, sendo utilizado um catavento para sua direção e um anemômetro para sua intensidade (HAKIM; PATOUX, 2017).

2.1.1.3 Precipitação e Chuva

De acordo com Ahrens (2001), a precipitação consiste em qualquer forma de água, seja ela líquida ou sólida, que cai das nuvens em direção à superfície terrestre. Como complementa Hakim e Patoux (2017), quando diz que precipitação é o termo genérico para chuva, neve, granizo etc.

Conforme Moreira e Sene (2016), a chuva está relacionada diretamente com a umidade relativa do ar. Então Ynoue et al. (2017, p. 39) define que "umidade do ar está relacionada à quantidade de vapor d'água na atmosfera".

Como confirma Hakim e Patoux (2017), quando diz que a umidade relativa do ar é a quantidade de vapor de água presente na atmosfera em uma determinada área. Moreira e Sene (2016) esclarecem que a atmosfera possui um limite de quantidade de vapor de água (umidade) que ela consegue comportar. Quando esse limite é atingido ocorre o fenômeno chamado ponto de saturação ou ponto de orvalho, ou seja, a atmosfera atinge o marco de umidade relativa do ar em 100%, ocorrendo assim a chuva.

2.1.2 Tecnologias de previsão do tempo

A previsão de chuvas existe há anos usando métodos que empregam técnicas estatísticas para avaliar a correlação entre precipitação, coordenadas geográficas (como latitude e longitude) e outros fatores atmosféricos (como pressão, temperatura, velocidade do vento e umidade) (BARRERA-ANIMAS et al., 2022).

De acordo com Hu et al. (2018), as Redes Neurais (RN), técnicas orientadas a dados, têm sido amplamente utilizadas na hidrologia como alternativa aos modelos físicos e conceituais. Essas técnicas de RN são baseadas em inteligência artificial (IA), que está entre as tecnologias mais destacadas dos últimos anos. Essas tecnologias podem capturar a não linearidade e a inconstância relacionada às aplicações hidrológicas.

Outro ponto que adiciona mais complexidade à tarefa de previsão de chuvas é que ela é prejudicada pela variação espacial e temporal das chuvas regionais (HOSSAIN et al., 2020). Nesse sentido, uma variante de redes neurais conhecida como Redes Neurais Recorrentes (RNN) é a mais adequada para enfrentar esses tipos de desafios, uma vez que RNNs são capazes de lidar com dinâmica temporal adotando conexões de *feed-*

back que a permitem lembrar informações anteriormente alimentadas em sua arquitetura (BARRERA-ANIMAS et al., 2022).

2.2 Aprendizado de Máquina

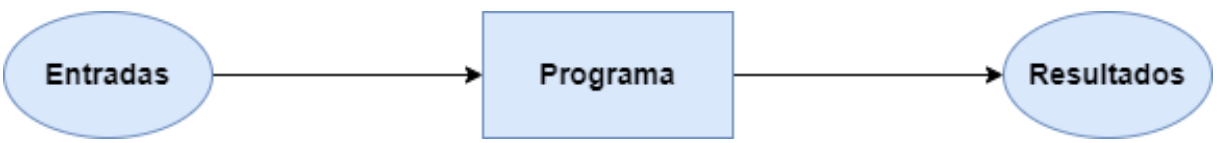
Como abordado anteriormente, *Deep Learning* ou Aprendizado Profundo é um tipo específico de aprendizado de máquina. Para entender *Deep Learning*, deve-se ter uma sólida compreensão dos princípios básicos do aprendizado de máquina.

A quantidade de dados que é gerado no mundo cresceu exponencialmente, segundo Sintef (2013). Em 2013 90% dos dados existentes no mundo tinham sido gerados nos últimos 2 anos. Apesar desse processo de transformação digital da sociedade ter facilitado os processos em diversas áreas, essa explosão de informação provoca uma complexidade maior em sua gestão. Esse dilúvio de dados exige métodos automatizados para o adequado armazenamento e análise dos dados, que é o que o aprendizado de máquina oferece.

Murphy (2013, p. 1) define aprendizado de máquina como "conjunto de métodos que podem detectar automaticamente padrões nos dados e, em seguida, usar os padrões descobertos para prever futuros dados, ou para realizar outros tipos de tomada de decisões sob incerteza". Normalmente, modelos de aprendizado de máquina são construídos com base em estatísticas, teoria da probabilidade, álgebra linear, e modelos matemáticos multivalorados e de otimização.

O aprendizado de máquina é, como a programação regular, uma maneira de fazer com que os computadores concluam uma tarefa específica. Normalmente, é fácil para os humanos executar etapas para concluir uma tarefa quando estão escrevendo um programa. Os passos a se fazer são pensados como que para fazer a tarefa à mão e, em seguida, aquela abstração é traduzida em código adequado ao processamento computacional (HOWARD; GUGGER, 2020). A Figura 1 representa a maneira tradicional da construção de uma solução de software.

Figura 1 – Um programa tradicional.

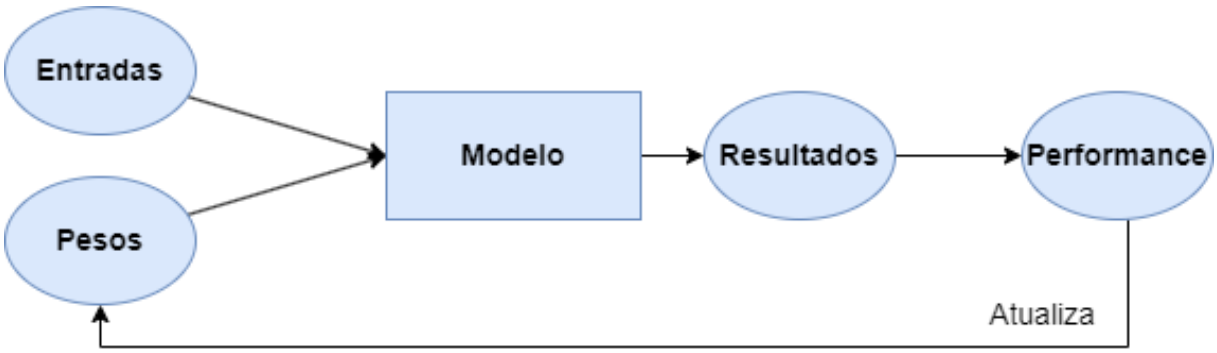


Fonte: Autoria própria.

Já quanto ao processo de treinamento do ML ocorre uma mudança de paradigma, primeiramente uma nova entrada é adicionada ao processo: os pesos. Os pesos são apenas

variáveis, e uma atribuição de peso é uma escolha particular de valores para essas variáveis. A ideia do processo de treinamento é que à medida que as previsões/resultados são executadas, a performance é avaliada a partir de uma métrica e um mecanismo que atualiza a atribuição de peso para maximizar a performance. Ao resultado dessas operações que buscam o melhor peso é chamado de modelo. A Figura 2 ilustra esse processo.

Figura 2 – Treinamento de um modelo de ML.



Fonte: Autoria própria.

O aprendizado de máquina é normalmente dividido em duas abordagens principais, o aprendizado supervisionado ou preditivo e o aprendizado não supervisionado ou descritivo.

2.2.1 Generalização

Quando um modelo finaliza seu treinamento, uma preocupação fundamental é garantir que ele generalize, isto é, aprenda padrões gerais dos dados de treinamento que também se aplicam a novos dados que encontrarão, a fim de que o modelo possa fazer previsões corretas sobre os novos dados.

Entretanto, existe um risco chamado *overfitting*, que acontece quando o treinamento do modelo é feito de modo errado, em que ao invés de generalizar e aprender características gerais dos dados, o modelo memoriza efetivamente os dados do conjunto de treinamento, ficando viciado no mesmo, e fazendo previsões inadequadas sobre novos dados (HOWARD; GUGGER, 2020). Já quando o modelo falha em reconhecer padrões entre os dados de teste, ocasionando por consequência que o modelo não consiga generalizar para novos dados, é dado o nome de *underfitting*.

2.2.2 Tipos de aprendizado

Os algoritmos de aprendizado supervisionado experimentam um conjunto de dados contendo recursos, mas cada exemplo também está associado a um rótulo ou destino (GOODFELLOW; BENGIO; COURVILLER, 2016). O termo aprendizado supervisionado

se origina da visão de o alvo sendo fornecido por um instrutor ou docente que mostra para máquina do sistema de aprendizagem o que fazer.

Os rótulos geralmente são fornecidos por bases de registro de eventos e especialistas. Além disso, se for viável, também podem ser produzidos por membros do público, como *crowdsourcing*, por exemplo. O aprendizado supervisionado é comumente utilizado em aplicações diárias, como reconhecimento facial e de fala, produtos ou recomendações de filmes e previsão de vendas.

Liu (2019, p. 14) relata que:

> quando os dados de aprendizagem contêm apenas sinais indicativos sem nenhuma descrição anexada, cabe a nós encontrarmos a estrutura dos dados por baixo, para descobrir informações ocultas ou para determinar como descrever os dados. Esse tipo de dado de aprendizado é chamado de dado não rotulado.

Na aprendizagem não supervisionada, não há instrutor ou docente, e o algoritmo deve aprender a entender os dados sem um guia.

O aprendizado não supervisionado é, em comparação ao aprendizado supervisionado, indiscutivelmente mais semelhante ao modo de aprendizado humano e animal (MURPHY, 2013). Esse tipo de aprendizado também é mais fácil de se aplicar em relação à aquisição de dados, uma vez que não requer um especialista humano para rotular manualmente os dados.

O aprendizado não supervisionado pode ser usado para detectar anomalias, como fraude ou equipamento defeituoso, ou para agrupar clientes com comportamentos *online* semelhantes para uma campanha de marketing, por exemplo.

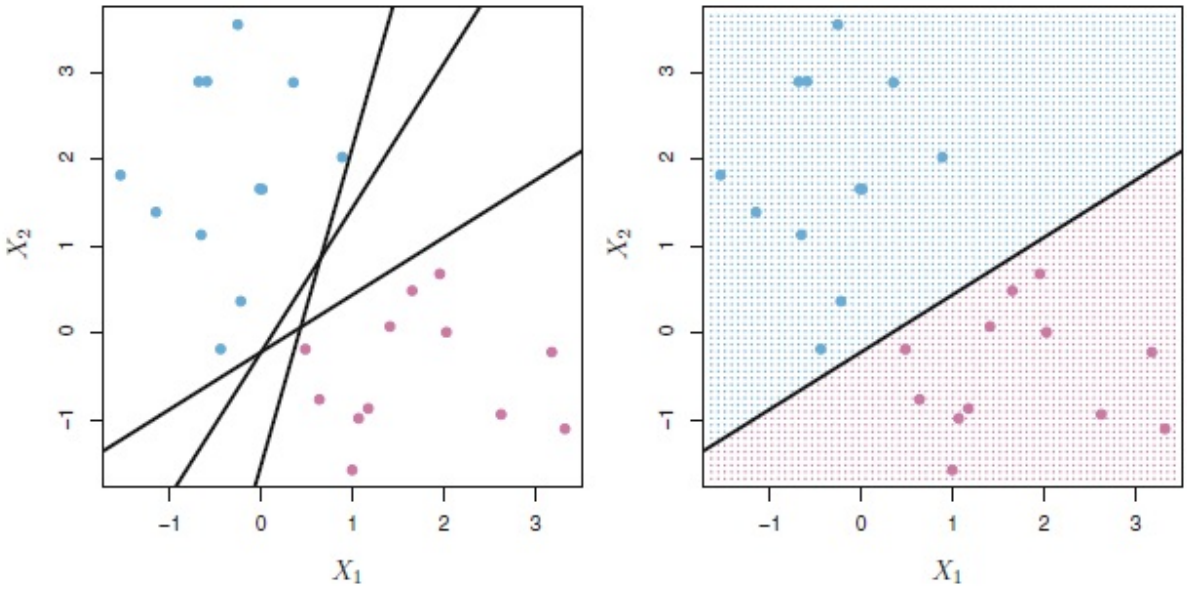
2.2.3 Máquina de vetores de suporte

O *Maximal Margin Classifier* (ou Classificador da Margem Máxima) é uma técnica na qual utiliza-se um hiperplano para separar as observações baseadas no aspecto desejado. No caso de uma problema com 2 variáveis, o hiperplano será uma reta.

Como existe uma infinidade de posições possíveis para o hiperplano caso os dados sejam perfeitamente separáveis, esse classificador busca determinar essa posição a partir da maximização da margem (ELOYR; NUNES, 2020).

A Figura 3 demonstra a representação gráfica do hiperplano Classificador da Margem Máxima.

Figura 3 – Representação gráfica de um hiperplano classificador.



Fonte: (ELOYR; NUNES, 2020).

Á esquerda, existem duas classes de observações, rosa e azul, cada uma delas possuem medidas de duas variáveis. Três hiperplanos possíveis, dentre vários, são mostrados em preto. À direita, um hiperplano é utilizado como um classificador.

Porém, o classificador da margem máxima perde eficiência consideravelmente quando os grupos não são separáveis, não sendo possível traçar um hiperplano.

Caso exista algum *outlier* no conjunto de dados ou os grupos não sejam completamente separáveis, esse método perde muito em eficiência ou até a aplicação, pois ou a margem fica pequena demais ou não é possível traçar o hiperplano.

O *Support Vector Classifier*, ou SVC, resolve este problema ao permitir a invasão da margem por certos pontos, permitindo assim ao classificador fazer algumas observações que sejam classificadas de maneira errada para que a classificação seja melhor no longo prazo. Porém, como denotado por Eloyr e Nunes (2020), o SVC só consegue desempenhar adequadamente se o delimitador entre as classes é linear, caso não seja linear o método vai apresentar um desempenho inadequado.

Para resolver este problema é possível transformar os dados de entrada não lineares para um espaço de maior dimensão chamado *feature space*, que por sua vez é linear. Isso é possível por meio de um conjunto de funções matemáticas chamadas *kernels*. Essa transformação dos dados em *feature space* permite definir uma medida de similaridade com base no produto escalar. Segundo Jakkula (2011), se o espaço de recurso for escolhido adequadamente, reconhecimento de padrões pode ser fácil. Uma vez que o problema fica linear, é possível utilizar o SVC.

Portanto, quando o SVC é combinado com uma função *kernel*, como a polinomial, o classificador resultante é conhecido como Máquina de vetores de suporte (ou SVM no inglês) é um algoritmo de *machine learning* supervisionado e principalmente utilizado para classificação.

2.2.4 *LightBGM*

O LightGBM é um framework de aumento de gradiente que usa algoritmos de aprendizado baseados em árvore. Esse *framework* oferece vários métodos de aumento do gradiente, sendo usado para os testes. O que obteve melhor performance foi o GBDT (*Gradient Boosting Decision Tree* ou Aumento de Gradiente em Árvore de Decisão), que foi sugerido no artigo de Friedman (2001).

Kuhn e Johnson (2013, p. 204) explica como funcionam algoritmos de aumento de gradiente:

> O algoritmo é tipicamente inicializado com a melhor estimativa de resposta. O gradiente é calculado e um modelo é então ajustado aos resíduos para minimizar a função de perda. O modelo atual é adicionado ao modelo anterior e o procedimento continua por um número de iterações especificado pelo usuário.

Assim, o aumento de gradiente envolve três elementos principais: Uma função de perda a ser otimizada, um aprendiz fraco (no inglês, *weak learner*) e um modelo aditivo que adiciona os aprendizes fracos para minimizar a função de perda. A função de perda escolhida depende do tipo de problema sendo resolvido, no caso da precipitação, é um problema de classificação, sendo escolhida a perda logarítmica que é indicativa de quão perto a probabilidade de previsão está do valor real/verdadeiro correspondente.

O motivo de ser usado um modelo de aprendizado fraco é se aproveitar da ideia de que várias hipóteses "fracas"diferentes combinadas resultam em uma hipótese final melhor (SCHAPIRE, 1990). Portanto, se o aprendizado "fraco"pode ser tão bom quanto o "forte", é possível aprender de forma mais rápida barata, mantendo resultados adequados.

O aprendiz fraco, ou *weak learner* no caso do GBDT são as árvores de decisão. Uma árvore de decisão é um modelo de aprendizado de máquina que se baseia em perguntas iterativas para particionar dados e chegar a uma solução. É a maneira mais intuitiva de se concentrar em uma classificação ou rótulo para um objeto. Visualmente também parece uma árvore de cabeça para baixo com galhos. É comum restringir os aprendizes fracos de maneiras específicas para que a árvore possua número máximo de camadas, nós, divisões ou nós folha.

O modelo aditivo vai adicionando uma árvore de cada vez, e as árvores existentes não são alteradas, sendo que um procedimento do gradiente descendente é utilizado para

minimizar a perda ao adicionar novas árvores. Seguindo o gradiente, são escolhidos novos parâmetros para a árvore na direção certa, reduzindo a perda residual. A saída para a nova árvore é então adicionada à saída da sequência de árvores existente em um esforço para corrigir ou melhorar a saída final do modelo.

2.3 Deep Learning

2.3.1 Redes Neurais Artificiais

Humanos e computadores são inerentemente adequados para diferentes tipos de tarefas. Por exemplo, calcular a raiz cúbica de um grande número é muito fácil para um computador, mas é difícil para humanos. Por outro lado, uma tarefa como reconhecer objetos em uma imagem é uma questão simples para um humano, mas difícil para um computador (AGGARWAL, 2019).

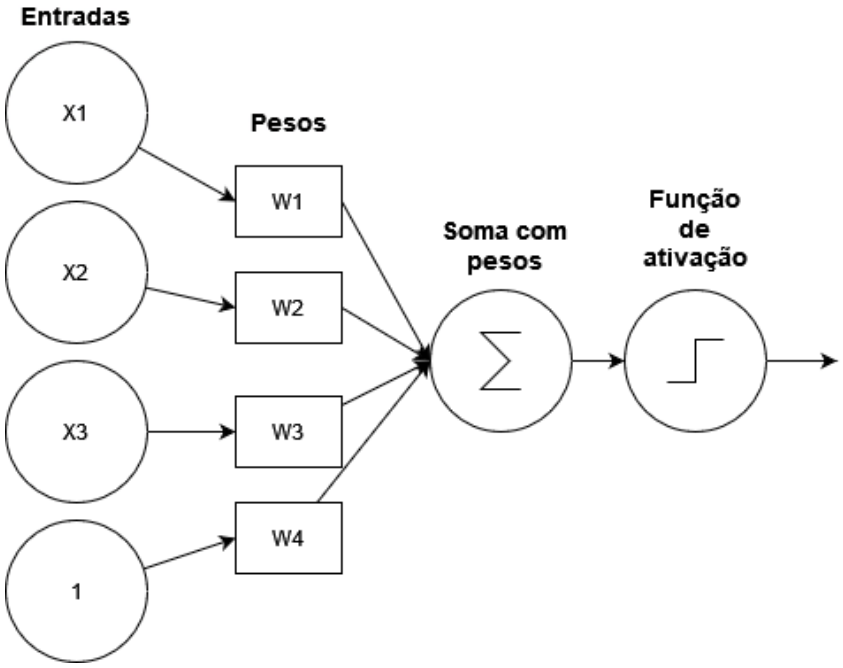
Para realizar esses tipos de tarefas que envolvem reconhecer padrões, cada ser humano contém uma rede neural biológica da vida real que está conectada ao sistema nervoso.

> A palavra neural é a forma adjetiva de "neurônio", e "rede" denota uma estrutura semelhante a um gráfico; portanto, uma rede neural artificial é um sistema de computação que tenta imitar (ou pelo menos é inspirado) as conexões neurais em nosso sistema nervoso. (ROSEBROCK, 2017, p. 122)

Voltando para a Figura 3, percebe-se que seria ideal a existência de algum tipo de mecanismo que fosse tão flexível que pudesse ser usado para resolver qualquer problema apenas variando seus pesos. Esse mecanismo existe e é a rede neural.

A Figura 4 representa uma rede neural básica, também referida como *Perceptron* (AGGARWAL, 2019). Os valores *x1*, *x2* e *x3* são as entradas para a rede, o valor constante 1 é o *bias*, que é a parte invariante da previsão.

Figura 4 – Esquema geral de uma rede neural básica.



Fonte: Autoria própria.

Uma rede neural básica recebe a soma ponderada do *input* x e pesos w. Essa soma é passada para função de ativação para determinar se o neurônio é disparado.

Cada x da entrada está conectado a um neurônio por meio de um vetor de peso W composto por *w1*, *w2*, ... *wn*, o que significa que para cada entrada x também tem-se um peso associado w.

Finalmente, no nó de saída à direita da Figura 4 x obtém a soma ponderada, aplica uma função de ativação f (usada para determinar se o neurônio "dispara" ou não) e gera um valor. Como denota Howard e Gugger (2020), isso pode ser matematicamente descrito na Equação 2.1;

$$f(net), net = \sum_{i=1}^{n} x_i + w_i + b \tag{2.1}$$

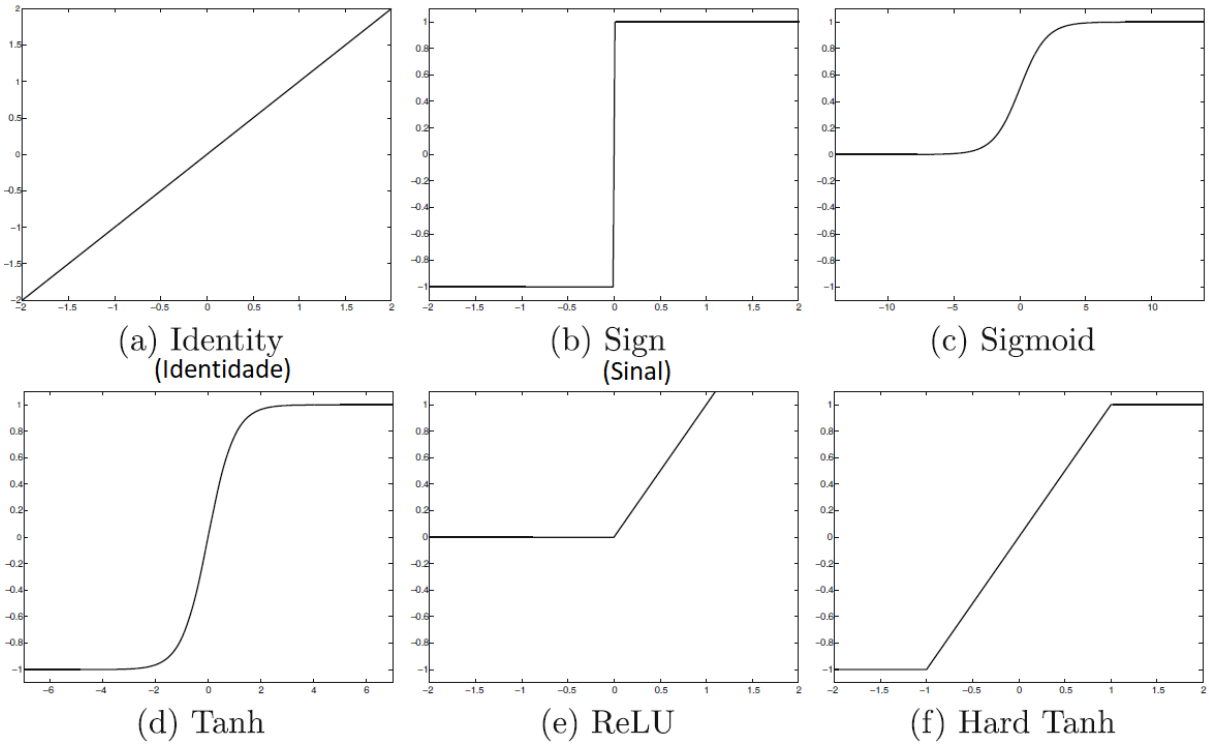
A função de ativação mais simples é a chamada *step*, usada pelo algoritmo *Perceptron*.

$$f(net) = \begin{cases} 1, & se\ net > 0 \\ 0, & caso\ contrario \end{cases} \tag{2.2}$$

Como mostra a Equação 2.2, esta é uma função simples. Se a soma ponderada *net* for maior que 0, produz resultado 1, caso contrário, produz 0. Plotando os valores de entrada ao longo do eixo x e a saída de $f(net)$ ao longo do eixo y, é observável por que

essa função de ativação recebeu o nome *step* (degrau) ou *sign* (Figura 5, centro superior). A saída de f é sempre zero quando *net* é menor ou igual a 0. Se *net* for maior que zero, então f retornará 1.

Figura 5 – Exemplos de funções de ativação.



Fonte: (AGGARWAL, 2019).

No entanto, apesar de intuitiva e fácil de usar, a função *step* (Equação 2.2) não é diferenciável, o que pode levar a problemas ao aplicar gradiente descendente e treinar a rede. (ROSEBROCK, 2017). Uma função mais utilizada na história das redes neurais é a função *sigmoid*, que pode ser visualizada na Equação 2.3:

$$t = \sum_{i=1}^{n} x_i + w_i + b, \qquad sigmoid(t) = 1/(1 + e^{-t}) \tag{2.3}$$

Rosebrock (2017) apresenta 3 motivos pelos quais a função *sigmoid* ou sigmoide é uma melhor escolha para o aprendizado em relação à função *step*. O primeiro é que a função é contínua e diferenciável em todos os pontos, é simétrica no eixo y e aproxima-se assintoticamente de seus valores de saturação.

A tangente hiperbólica, ou *tanh* (com uma forma semelhante ao sigmoide) também foi muito usada como uma função de ativação até o final da década de 1990 (ROSEBROCK, 2017) (Figura 5 (f)). A equação para *tanh* é apresentada na Equação 2.4:

$$t = \sum_{i=1}^{n} x_i + w_i + b, \qquad tanh(t) = (e^{t} - e^{-t})/(e^{t} + e^{-t}) \tag{2.4}$$

Como observa Ekman (2021), à medida que x se aproxima do infinito negativo, a função *tanh* se aproxima de -1, assim como a função *step*. Se o mesmo exercício for feito para a função sigmoide, será visível que ela também se aproxima de +1 quando x se aproxima do infinito, ao passo que se aproxima de zero quando x se aproxima do infinito negativo.

2.3.2 Otimização

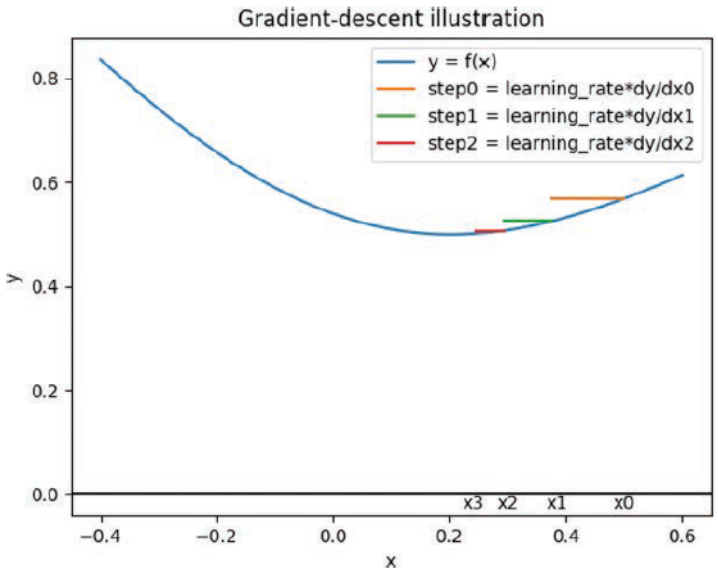
Algoritmos de otimização são aspectos essenciais do aprendizado de máquina e do *deep learning*. A acurácia de um modelo depende diretamente de encontrar um conjunto de pesos (W) e *bias* (b) tais que os dados são corretamente classificados (ROSEBROCK, 2017). Ao invés de escolher os valores de W e b aleatoriamente, pode-se definir um algoritmo de otimização que permite literalmente melhorar W e b. Um dos algoritmos mais usados para treinar modelos de redes neurais é o gradiente descendente.

O algoritmo do gradiente descendente tem muitas variantes, mas, em cada caso, a ideia é a mesma: avalie iterativamente seus parâmetros, calcule sua perda e dê um pequeno passo na direção que minimizará sua perda. A Figura 6 representa o algoritmo em uma dimensão.

Inicia-se com uma estimativa inicial x0. Esse valor é inserido na função f(x) e calcula-se o y correspondente, bem como sua derivada. Assumindo que ainda não é o valor mínimo de y, agora pode-se chegar a uma estimativa aprimorada x1 aumentando ou diminuindo x0 ligeiramente. O sinal da derivada indica se deve-se aumentar ou diminuir x0. Uma inclinação positiva (como na Figura 6) indica que y diminuirá se x diminuir. Pode-se então refinar iterativamente a solução fazendo repetidamente pequenos ajustes em x.

O algoritmo de *Back propagation* ou retro propagação consiste em um passo direto no qual exemplos de treinamento são apresentados à rede. É seguido por uma passagem para trás na qual os pesos são ajustados usando o gradiente descendente. O gradiente é calculado usando o algoritmo de *Back Propagation* (EKMAN, 2021).

Figura 6 – Gradiente descendente em uma dimensão.



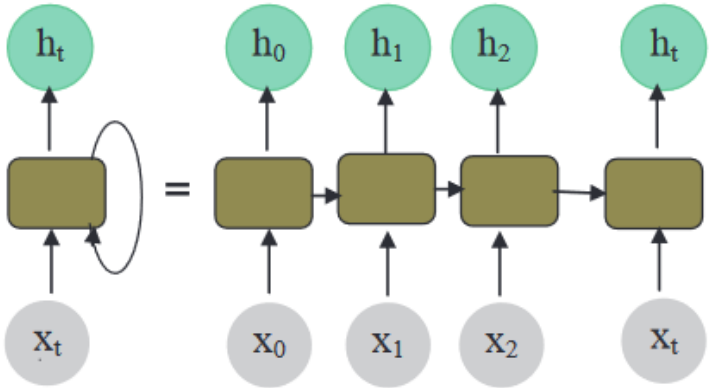
Fonte: (EKMAN, 2021).

2.3.3 Redes Neurais Recorrentes

As redes neurais recorrentes (RNNs no inglês) são amplamente reconhecidas como ferramentas eficazes que podem ser empregadas por uma ampla gama de aplicações que armazenam e processam sequências temporais. (SHAO, 2008)

A RNN é um sistema dinâmico com uma sequência de módulos repetidos formando uma estrutura em cadeia. Essa estrutura evoluiu com a ideia de que para cada módulo subsequente, a saída da etapa t não depende apenas na observação de entrada em t, mas também na etapa anterior a sequência de saída, ou seja, sequência de saída da etapa *t-1*. Nesse caminho, esta arquitetura faz uso das informações armazenadas das etapas de processamento anteriores (SAMAD et al., 2020). Este tipo de arquitetura é mostrado na Figura 7.

Figura 7 – Redes Neurais Recorrentes.



Fonte: (SAMAD et al., 2020).

O lado direito da Figura 7 mostra como uma camada recorrente pode ser desenrolada no tempo. Ao criar uma cópia da camada recorrente para cada passo de tempo, converte-se a camada recorrente em várias camadas de arquitetura direta (*feedfoward*). Segundo Ekman (2021) esse desenrolamento pode ser útil tanto para raciocinar sobre a rede, quanto para estender o funcionamento do algoritmo de *Back Propagation* em redes recorrentes.

> A RNN utiliza o algoritmo *Back Propagation Through Time* (BPTT) para cálculo de gradientes e ajustes de matrizes de peso da rede durante o processo de treinamento. Como vários *loops* estão sendo executados durante o treinamento, grandes atualizações estão acontecendo em paralelo com ajustes de pesos para cada unidade, dependendo do erro total de saída, todos esses resultados são agregados em gradientes de erro acumulados que tornam a rede instável (SAMAD et al., 2020, p. 192).

Essa imprecisão significa que muitas vezes os gradientes calculados para atualizar os pesos terminam como zero ou infinito para redes profundas. Isso é comumente referido como o problema dos gradientes de fuga ou gradientes explosivos (HOWARD; GUGGER, 2020).

2.3.4 Memória de Longo Curto Prazo (*Long short-term memory*)

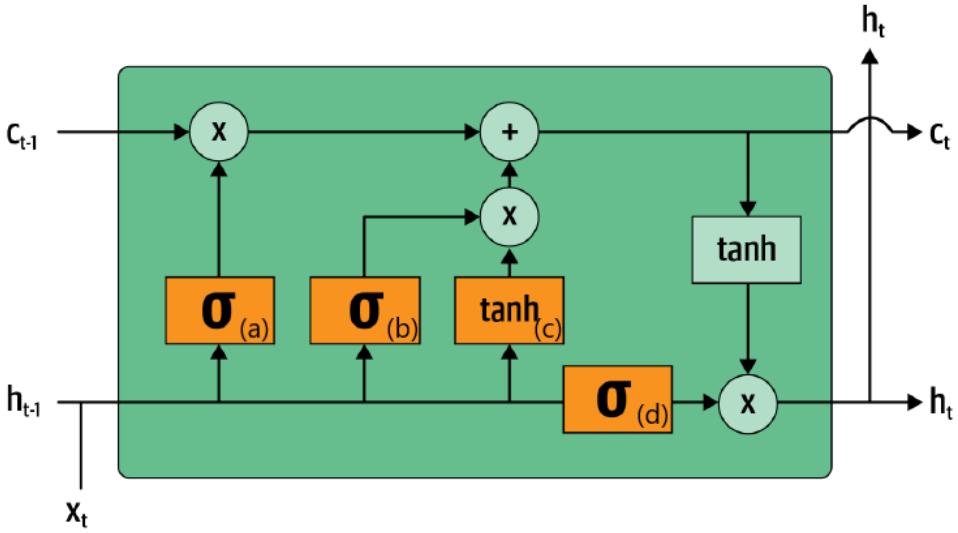
A *Long short-term memory* (LSTM), introduzida por Hochreiter e Schmidhuber (1997), é uma unidade mais complexa que atua como um substituto para um único neurônio em uma rede neural recorrente (EKMAN, 2021).

Nesta arquitetura, não há um, mas dois estados ocultos. Na prática, os RNNs não conseguem reter a memória do que aconteceu muito antes, e essa é a motivação para ter outro estado oculto (chamado *cell state*) no LSTM. O estado da célula será responsável por manter a memória de curto prazo longa, enquanto o estado oculto se concentrará no próximo *token* a ser previsto (HOWARD; GUGGER, 2020). A arquitetura do LSTM é representada na Figura 8.

Como a Figura 8 ilustra, a entrada xt entra à esquerda com o estado oculto anterior (ht-1) e o estado da célula (ct-1). As quatro caixas laranja representam quatro camadas (as redes neurais) ou portões, com a ativação sendo funções sigmoide ou *tanh*.

Os círculos verdes na Figura 8 são operações elementares. O que sai à direita é o novo estado oculto (ht) e o novo estado da célula (ct), prontos para a próxima entrada. O novo estado oculto também é usado como saída, e é por isso que a seta se divide para subir.

Figura 8 – A arquitetura de uma LSTM.



Fonte: (HOWARD; GUGGER, 2020).

O primeiro portão (letra "a" da Figura 8) é chamado de portão do esquecimento. Quando o resultado passa pela função sigmoide, sua saída será entre 0 e 1, logo após multiplica-se o resultado pelo *cell state*. Se o valor for mais próximo de 1 o resultado será mantido, se for de 0, será descartado, por isso o nome de portão do esquecimento.

Isso fornece ao LSTM a capacidade de esquecer as coisas sobre seu estado de longo prazo. (HOWARD; GUGGER, 2020).

O segundo e terceiro portões (respectivamente letras "b" e "c" da Figura 8) são utilizados para alterar o estado da célula, caso o portão de esquecimento tenha removido. O segundo portão, também chamado de portão de entrada, decide quais elementos do estado da célula devem ser atualizados ou não e o terceiro portão determina quais são esses valores atualizados.

O último portão (letra "d" da Figura 8) é o de saída, ele determina quais informações do estado da célula devem ser usadas para gerar a saída. O estado da célula passa por um *tanh* antes de ser combinado com a saída sigmoide da porta de saída, e o resultado é o novo estado oculto.

2.4 Aplicação Web

A finalidade desta seção é apresentar a perspectiva histórica do acesso à Internet no Brasil, prover detalhes específicos de desenvolvimento, a fim de fornecer embasamento para o desenvolvimento do presente trabalho.

Uma aplicação *web* é uma aplicação de software desenvolvida para ser executada em navegadores de Internet, que pode ser acessado como um site (Made In Web, 2020).

Essas aplicações não precisam de instalações na máquina do usuário e são geralmente acessadas em formato de um sítio eletrônico (*web site*).

2.4.1 Internet no Brasil

Como foi visto anteriormente, o acesso à informação relacionado a previsão do tempo é de extrema importância para as sociedades humanas nos dias atuais. Por isso a Engenharia de Software possui um papel importante em desenvolver aplicações que facilitem o acesso a essas informações.

De acordo com Spínola e Araújo (2007, p. 8), a "Engenharia de Software visa à criação de produtos de software que atendam às necessidades de pessoas e instituições [...]". Já Sommerville (2011) diz que o mundo moderno não poderia existir sem o desenvolvimento do software, e que as sociedades dependem da Engenharia de Software para seu pleno funcionamento.

Um indicativo desta informação pode ser visto no estudo do Armstrong (2021), que mostra que em 2021 já existiam 1,88 bilhões de *web sites* ativos. Por isso as aplicações *web* fazem parte da vida da maioria da população. Uma pesquisa realizada em 2020 pelo Comitê Gestor da Internet no Brasil (2021) aponta que o número de usuários de Internet no Brasil chegou a 152 milhões de pessoas, provavelmente tendo crescido neste período em que ainda se mantém a pandemia da Covid-19.

2.4.2 Aplicativo *Web* Progressivo (*Progressive Web App*)

Os *Progressive Web Apps* (PWA) são aplicações *web* que se beneficiam das novas funcionalidades que os navegadores de Internet possuem, como acesso a geolocalização, notificações *push* e diversos outros recursos que beneficiam o uso desta tecnologia (TANDEL; JAMADAR, 2018).

Uma aplicação PWA possui uma série de vantagens em comparação com aplicações móveis nativas, como a possibilidade de funcionar em qualquer dispositivo com navegador de Internet, não há necessidade de instalação e, por fim, consegue passar ao usuário a experiência de uma aplicação móvel nativa, e ainda permite a usabilidade de um site para dispositivos não móveis, como computadores *Desktop* (SHEPPARD, 2017).

2.4.3 Desenvolvimento *Web*

De acordo com a IBM (2012), as aplicações *web* progressivas são mais simples, mais baratas e mais fáceis de atualizar que outras soluções como aplicações nativas. Além de serem mais acessíveis, por não estarem em lojas de aplicativos ou necessitarem de instalação.

As aplicações *web* progressivas são desenvolvidas da mesma maneira e com as mesmas tecnologias utilizadas nas aplicações *web* mais simples utilizando tecnologias como *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript* (HEITKöTTER; HANSCHKE; MAJCHRZAK, 2012).

Segundo Heitkötter, Hanschke e Majchrzak (2012), as aplicações *web* surgem de uma necessidade de facilitar o acesso a sistemas *online* e que ao mesmo tempo promovem o desenvolvimento mais rápido, mais barato e precisam utilizar somente uma base de código fonte e ainda conseguem ser executadas em diversas plataformas.

2.5 Bases de dados

A escolha dos dados é essencial para o sucesso de um modelo de redes neurais e a grande variedade de fontes de dados meteorológicos associada aos diferentes resultados obtidos com a utilização de cada um deles levanta questionamentos em relação à melhor fonte de dados a ser utilizada. Após uma averiguação dos autores, três fontes de dados se destacaram para os dados procurados (como precipitação, temperatura, umidade e pressão atmosférica), sendos elas: INMET, HidroWeb e Rede Sonda. A Tabela 1 relaciona as três bases de dados com as variáveis de interesse estipuladas para o uso no modelo preditivo.

O Instituto Nacional de Meteorologia (INMET) coleta dados através de 194 estações automáticas. Essas estações coletam, de minuto em minuto, as informações meteorológicas representativas da área em questão. São coletadas temperatura, umidade, pressão atmosférica, precipitação, direção e velocidade do vento, irradiação solar. Estes dados são integrados e disponibilizados para serem transmitidos, via satélite ou telefonia celular, para a sede do INMET, em Brasília (FRASCISO et al., 2020). Um ponto interessante dessa base é que granularidade mínima encontrada *online* é por hora e a ela possui o maior número de variáveis de interesse entre seus atributos disponíveis.

O Portal HidroWeb possui menor detalhamento e granularidade, com coleta de dados semanais. HidroWeb (2005, p. 1) informa que:

> O Portal HidroWeb é uma ferramenta integrante do Sistema Nacional de Informações sobre Recursos Hídricos e oferece o acesso ao banco de dados que contém todas as informações coletadas pela Rede Hidrometeorológica Nacional (RHN), reunindo dados de níveis fluviais, vazões, chuvas, climatologia, qualidade da água e sedimentos.

O Instituto Nacional de Pesquisas Espaciais (INPE) possui a Rede Sonda. Essa rede entrou em operação em 2004 e atualmente a rede dispõe de 20 estações de observação distribuídas estrategicamente para representar as diferentes características climáticas

do Brasil. Os dados são coletados a cada 1 minuto e disponibilizados *online*. As estações da Rede Sonda coletam os dados de minuto em minuto. Esses dados são verificados e disponibilizados gratuitamente após alguns meses. Contudo, a abrangência espacial é extremamente baixa, com apenas 20 estações em todo o país (FRASCISO et al., 2020). Entretanto, a taxa com que os dados são salvos são por minuto, fazendo dessa base uma opção com o maior número de tuplas.

Tabela 1 – Bases de dados e variáveis de interesse.

	INMET	**HidroWeb**	**Rede Sonda**
Precipitação Total	sim	sim	sim
Temperatura Máxima	sim	sim	não
Temperatura Mínima	sim	sim	não
Temperatura do ar	sim	sim	sim
Temperatura do ponto de orvalho	sim	não	não
Umidade Relativa	sim	sim	sim
Pressão Atmosférica	sim	sim	sim
Velocidade do vento	sim	não	sim
Direção do vento	sim	não	sim

Fonte: Autoria própria.

3 Metodologia

Este capítulo tem o objetivo de apresentar a metodologia deste trabalho, de acordo com os objetivos definidos no Capítulo 1. Ele divide-se entre metodologia de pesquisa, que define os aspectos dos tipos de pesquisas adotados no mesmo, a metodologia de desenvolvimento do modelo de *Machine Learning* e a metodologia a ser implementada para o desenvolvimento da aplicação (software).

Após a realização de pesquisas e o melhor conhecimento obtido sobre as principais características no contexto desta proposta foi possível compreendê-la para se elaborar um projeto coerente aos objetivos almejados.

Assim, foram definidas as metodologias que seriam adequadas para resolvê-lo, sendo nesta seção abordados os conceitos acerca desses tipos de pesquisas e como o projeto será desenvolvido. Gerhardt e Silveira (2009) afirmam que existem quatro maneiras de classificar os diferentes tipos de pesquisa, que são quanto a sua abordagem, natureza, objetivos e procedimentos.

Quanto à abordagem, o problema será tratado de maneira quantitativa, com números e unidades de medida. Segundo Fonseca (2002), a pesquisa quantitativa é aquela que tem foco na objetividade, isto é, a realidade só pode ser compreendida com base na análise de dados brutos, recolhidos com instrumentos padronizados e neutros.

Este trabalho tem como objetivo apresentar uma solução para um problema único do mundo real, e por este motivo, classifica-se como uma pesquisa aplicada quanto a sua natureza. "Uma pesquisa classifica-se como aplicada quando busca resultados concretos. Desse modo, a pesquisa aplicada desenvolve-se com objetivos específicos para entregar aplicabilidades ou problemas práticos para resolver" (ZUCATTO; FREITAS; MARZZONI, 2020, p. 8).

Quanto aos objetivos, este projeto situa-se como uma pesquisa exploratória, pois busca proporcionar maior familiaridade com o problema, para torná-lo mais explícito e construir hipóteses de solução.

Outra forma de caracterização do trabalho, segundo Fonseca (2002), é aquela feita acerca dos procedimentos de pesquisa que serão executados. Para este trabalho foram escolhidos a pesquisa-ação, que é aquela que propõe uma intervenção participativa na realidade e a pesquisa bibliográfica, que é o estudo sistematizado desenvolvido com base em material científico publicado em livros, revistas, jornais e redes eletrônicas.

Para a realização dos objetivos propostos neste trabalho foi necessário realizar, primeiramente, o estudo de livros e artigos científicos que apresentam conteúdo relevante

para o trabalho. Esta primeira etapa do projeto busca construir a base teórica do mesmo, além de conhecer os recursos e projetos interessantes no mesmo escopo que estejam disponíveis. A técnica empregada foi a pesquisa bibliográfica, e seus principais achados fazem parte das referências bibliográficas deste trabalho.

Diante destas compreensões, as características metodológicas empregadas neste trabalho estão resumidas na Tabela 2.

Tabela 2 – Escolhas metodológicas.

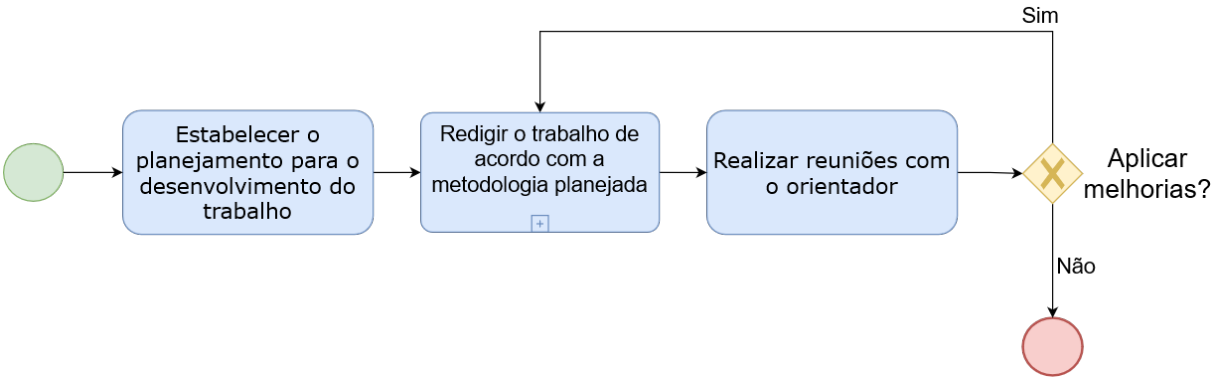
Abordagem	**Natureza**	**Objetivo**	**Procedimentos**
Pesquisa Quantitativa	Pesquisa Aplicada	Pesquisa Exploratória	Pesquisa-ação Pesquisa Bibliográfica

Fonte: Autoria própria.

3.1 Fluxo de Atividades

Em uma visão macro, o fluxo de atividades que seria realizado neste trabalho, está representado na Figura 9.

Figura 9 – Diagrama da visão macro de execução do trabalho.



Fonte: Autoria própria.

Portanto, no fluxo de execução do trabalho foram estabelecidas as seguintes atividades principais:

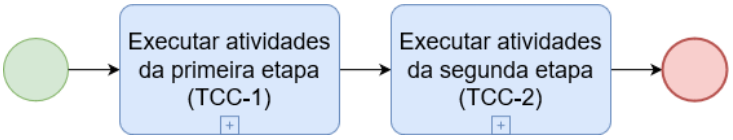
- **Estabelecer o planejamento para o desenvolvimento do trabalho** - Atividade que consiste em estabelecer a sequência de atividades a serem adotadas durante as duas diferentes etapas intituladas TCC-1 e TCC-2.
- **Redigir o trabalho de acordo com a metodologia planejada** - Esta atividade corresponde à execução do processo planejado na primeira atividade, que consiste em toda a execução da metodologia proposta, sendo dividida em vários subprocessos a serem executados. Os processos da metodologia serão abordados na Seção 3.1.1.

- **Realizar reuniões com o orientador** - Atividade que visa a realização de reuniões semanais com o professor orientador, com o objetivo de alinhar todas as partes envolvidas e avaliar, quando estiverem prontas, as entregas incrementais das etapas deste trabalho, direcionando para uma posição mais pertinente ao atingimento dos objetivos deste trabalho.

3.1.1 Metodologia de Execução do Trabalho

A Figura 10 representa o processo que foi definido para a execução deste trabalho. Tal processo corresponde ao subprocesso "Redigir o trabalho de acordo com a metodologia planejada", exibido na Figura 9.

Figura 10 – Diagrama do processo de execução do trabalho.



Fonte: Autoria própria.

Sendo que:

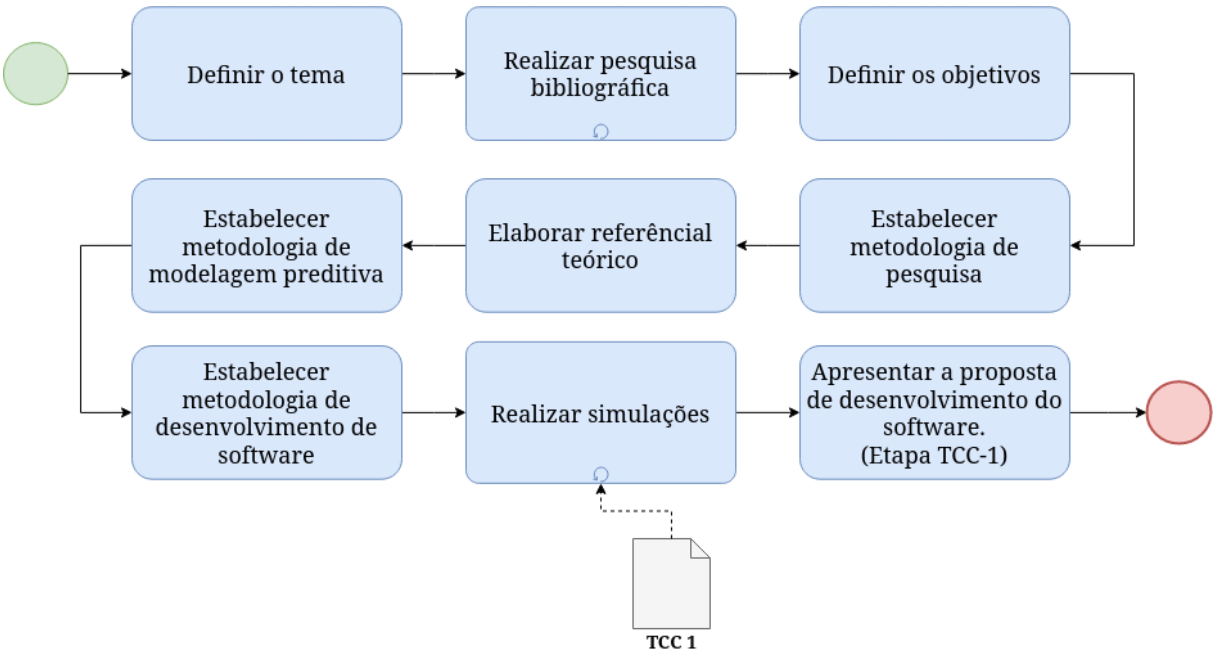
- **Executar atividades da primeira etapa (TCC-1)** - Essa atividade é o primeiro processo apresentado pelo subprocesso "Redigir o trabalho de acordo com a metodologia planejada", apresentado na Figura 9 e corresponde às atividades definidas para serem executadas durante a etapa TCC-1. Esta etapa compreende a criação do arcabouço teórico e planejamento das atividades que serão executadas para a elaboração da aplicação, além de uma simulação da arquitetura. Uma descrição detalhada do processo é apresentada na Seção 3.1.1.1.

- **Executar atividades da segunda etapa (TCC-2)** - Essa atividade é o segundo processo apresentado pelo subprocesso "Redigir o trabalho de acordo com a metodologia planejada", apresentado na Figura 9 e corresponde às atividades definidas para serem executadas durante a etapa TCC-2, em que a aplicação será elaborada seguindo o planejamento da etapa anterior. Uma descrição detalhada do processo é apresentada na Seção 3.1.1.2.

3.1.1.1 Atividades da Primeira Etapa

A Figura 11 representa o processo que foi estabelecido na primeira etapa de realização do TCC-1, que corresponde ao subprocesso "Executar atividades da primeira etapa (TCC-1)" indicado na Figura 10.

Figura 11 – Diagrama do processo geral para as atividades da primeira etapa (TCC-1).



Fonte: Autoria própria.

Na Figura 11 é ilustrada a seguinte sequência de atividades:

- **Definir o tema** - Atividade que consiste em definir a grande área a ser pesquisada no contexto de realização deste trabalho.
- **Realizar pesquisa bibliográfica** - Atividade que consiste em pesquisar por fontes de conhecimento legítimas que contribuam para o desenvolvimento do trabalho.
- **Definir os objetivos** - Atividade que consiste em delinear o escopo de desenvolvimento do trabalho valendo-se da definição de resumo da proposta de trabalho, objetivo geral e objetivos específicos.
- **Estabelecer metodologia de pesquisa** - Define como serão realizados os procedimentos relativos à formação da base teórica referente às atividades de pesquisa.
- **Elaborar referencial teórico** - Atividade em que é efetivamente criada uma linha de raciocínio de forma a construir um relacionamento entre as fontes de conhecimento (bibliografia) e o conteúdo a ser usado no trabalho.
- **Estabelecer metodologia de modelagem preditiva** - Visa estabelecer a arquitetura que será utilizada no modelo preditivo, os dados que serão usados e outras propriedades relevantes.
- **Estabelecer metodologia de desenvolvimento de software** - Objetiva definir o processo de desenvolvimento de software da aplicação proposta.

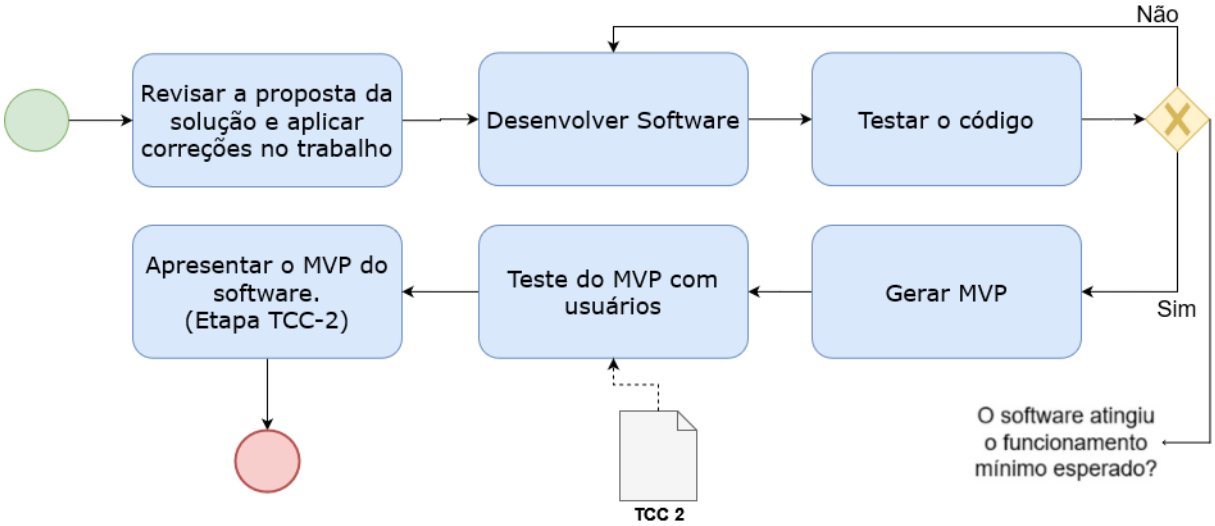
- **Realizar simulações** - É a atividade em que tem como objetivo verificar a viabilidade das metodologias previamente estabelecidas, por meio de uma prototipação de funcionalidades do software.

- **Apresentar a proposta de desenvolvimento do software (Etapa TCC-1)** - Atividade que finaliza o primeiro ciclo maior de desenvolvimento deste trabalho, com a exposição dos resultados obtidos até o momento para a banca avaliadora.

3.1.1.2 Atividades da Segunda Etapa

A Figura 12 representa o processo que será estabelecido na segunda etapa de realização deste trabalho, que corresponde ao subprocesso "Executar atividades da segunda etapa (TCC-2)" indicada na Figura 10.

Figura 12 – Diagrama do processo geral para as atividades da segunda etapa (TCC-2).



Fonte: Autoria própria.

- **Revisar a proposta da solução e aplicar correções no trabalho** - Atividade que consiste em realizar adições ou mudanças de conteúdo do trabalho, conforme o *feedback* dos professores da banca avaliadora após a etapa TCC-1.

- **Desenvolver Software** - Realização das *sprints* de desenvolvimento do aplicativo, com base nas adaptações discutidas na Seção 3.3.1.2 e posteriormente abordadas na Figura 17.

- **Testar o código** - Teste e validação das funcionalidades desenvolvidas na etapa anterior de desenvolvimento, a fim de manter o bom funcionamento do software;

- **Gerar o MVP** - Após iterações do ciclo de desenvolvimento e teste, a geração do MVP, ou seja, o Produto Mínimo Viável, com as funcionalidades essenciais devidamente implementadas;

- **Teste do MVP com usuários** - Testar junto à possíveis usuários se as funcionalidades do aplicativo conseguem corresponder às suas expectativas;

- **Apresentar o MVP do software (Etapa TCC-2)** - Atividade que fecha todo o ciclo de desenvolvimento deste trabalho, com a exposição e discussão dos principais resultados obtidos para a banca avaliadora.

3.1.1.3 Cronograma da Segunda Etapa (TCC-2)

Tabela 3 – Cronograma da Segunda Etapa (TCC-2).

Atividade	**Mês**			
	Junho	**Julho**	**Agosto**	**Setembro**
Revisar a proposta	x			
Desenvolver Software	x	x	x	x
Testar o código		x	x	x
Gerar MVP			x	x
Teste do MVP com usuários				x
Apresentar o TCC-2 para a banca				x

Fonte: Autoria própria.

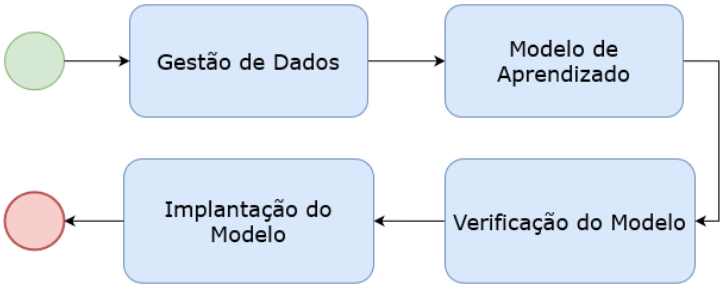
A Tabela 3 representa o cronograma para a realização das atividades da segunda etapa do projeto (TCC-2), contemplando os meses de execução de cada atividade.

3.2 Metodologia de *Machine Learning*

Assim como na Engenharia de Software, em que o processo de desenvolvimento de um software tradicional passa por diversas etapas até a sua conclusão é chamado de ciclo de vida do software (SOMMERVILLE, 2011). Na área de ML o desenvolvimento, treinamento e veiculação de modelos de ML são o resultado de um processo que é chamado de ciclo de vida do *Machine Learning* (GARCIA et al., 2018).

Dentre os vários modelos de ciclo de vida propostos na literatura de pesquisa, Ashmore, Calinescu e Paterson (2021) definem um ciclo de vida baseado em quatro fases inter-relacionadas. A Figura 13 ilustra esse modelo:

Figura 13 – O ciclo de vida do aprendizado de máquina compreendendo quatro estágios.



Fonte: Autoria Própria.

O primeiro estágio desse ciclo é a fase de Gestão de Dados. Essa fase concentra-se na obtenção dos conjuntos de dados necessários para o treinamento e para a verificação dos componentes do ML. Este estágio inclui atividades que vão desde a coleta de dados até o pré-processamento de dados (ASHMORE; CALINESCU; PATERSON, 2021).

A próxima etapa é intitulada de Modelo de Aprendizado, nela são executadas as atividades associadas à síntese do modelo de ML a partir do conjunto de dados de treinamento. O aprendizado de máquina real acontece nesta etapa, que também inclui atividades como seleção do algoritmo de ML, a função de perda (*loss*) e seleção de hiper parâmetros. (ASHMORE; CALINESCU; PATERSON, 2021).

A terceira etapa do ciclo de vida do ML é a Verificação do Modelo. O desafio central desta etapa é garantir que o modelo treinado está em conformidade com seus requisitos e consiga generalizar bem para dados não vistos anteriormente. Essa verificação se dá a partir do conjunto de dados de verificação que o estágio de Gestão de Dados produziu independentemente do conjunto de dados de treinamento. (ASHMORE; CALINESCU; PATERSON, 2021).

Finalmente a etapa de Implantação do Modelo compreende atividades relacionadas à integração do(s) modelo(s) de ML verificado(s) com o sistema de componentes desenvolvidos e apurados usando métodos de Engenharia de Software, como o acompanhamento do seu funcionamento e atualização. O resultado do estágio de implantação do modelo é o sistema em funcionamento. (ASHMORE; CALINESCU; PATERSON, 2021).

Era intenção utilizar deste ciclo de vida para a criação do modelo neste trabalho, mas algumas adaptações podem ser necessárias, como será explorado no Capítulo 4.

3.3 Metodologia de Desenvolvimento de Software

De acordo com Sommerville (2011) um processo de desenvolvimento de software é um conjunto de atividades que possuem o objetivo de construir um produto de software. Essas atividades podem tanto desenvolver sistemas que não existam ainda, como aprimorar ou modificar sistemas já existentes. Já Aléssio, Sabadin e Zanchett (2017) afirmam que

o processo de software pode ser entendido como um conjunto de atividades relacionadas ao entendimento, esboço e implementação de uma solução de software.

3.3.1 Metodologia Ágil

Em 2001, um grupo de dezessete pessoas, incluindo representantes do *Extreme Programming* (XP), *Scrum*, DSDM, *Adaptative Software Development* (ASD), *Crystal*, *Feature-Driven Development* (FDD), entre outros, se reuniram para discutir novas metodologias de desenvolvimento de software, e acabaram criando o "Manifesto para Desenvolvimento Ágil de software" (COHEN; LINDVALL; COSTA, 2012).

O desenvolvimento ágil de software procura descobrir melhores maneiras de gerir e executar o desenvolvimento, visando a colaboração em equipe e o auto aperfeiçoamento (BECK et al., 2001). O manifesto em si valoriza indivíduos e interações mais que processos e ferramentas, software em funcionamento mais que documentação extensiva, colaboração com o cliente mais que negociação de contratos e responder a mudanças mais que seguir um plano.

No desenvolvimento da solução de software, foi decidido a utilização de práticas e ferramentas baseadas nas metodologias ágeis de *Kanban* e *Scrum*, que foram adotadas na elaboração deste trabalho.

3.3.1.1 Kanban

O *Kanban* é um sistema baseado no sistema de produção da Toyota, criado para controlar estoque e demanda de componentes implementado na fábrica da Toyota na década de 1940 (JUNIOR; FILHO, 2010).

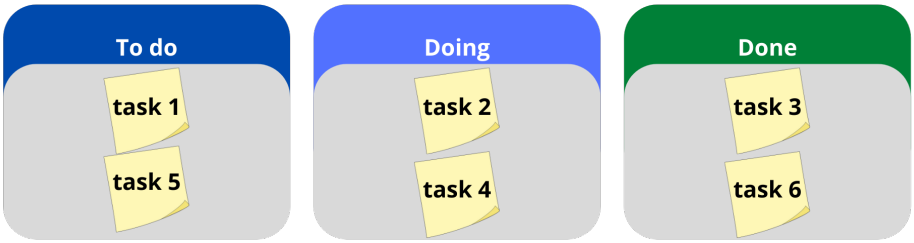
No desenvolvimento de software, o *Kanban* é uma metodologia ágil que foca no gerenciamento do desenvolvimento através de representações gráficas de um quadro com cartões e suas atividades (REHKOPF, 2020).

De acordo com Kniberg e Skarin (2010) os princípios do *Kanban* são visualização do fluxo de trabalho, limitação do fluxo de trabalho e acompanhamento do gerenciamento do fluxo de trabalho.

Segundo Silva, Santos e Neto (2012) o *Kanban* é uma metodologia altamente adaptativa, e permite mudanças e melhorias para que o processo de desenvolvimento possa fluir de maneira satisfatória.

Os fluxos de trabalho *Kanban* podem possuir diversos cartões e podem ser alterados ao longo de suas aplicações. Os fluxos clássicos são geralmente mais simplificados e começam com três cartões denominados: **A fazer** (*To do*), **Fazendo** (*Doing*), **Feito** (*Done*). A Figura 14 ilustra esse fluxo.

Figura 14 – Exemplo de quadro *Kanban*.



Fonte: Autoria própria.

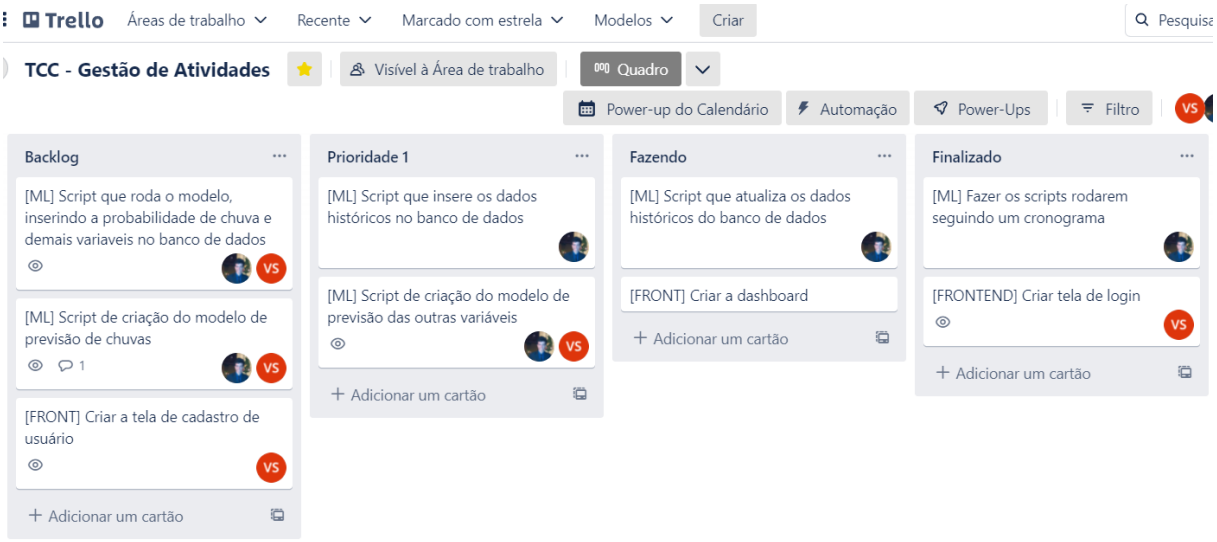
A Figura 14 ilustra um esquema de fluxo de trabalho *Kanban* clássico com três cartões, sendo eles, tarefas (*tasks*) a serem realizadas (*to do*), tarefas sendo feitas (*Doing*) e tarefas que já foram realizadas (*Done*).

No presente trabalho, a metodologia *Kanban* será utilizada por meio da ferramenta Trello, descrita na Seção 3.7.1, sendo eles:

- *Backlog*: Onde as tarefas já elicitadas em planejamentos próprios ficam até serem priorizados;
- Prioridade: Uma vez que as tarefas mais importantes de um ciclo de desenvolvimento são definidas, é nesse cartão que elas são colocadas;
- Fazendo: Uma vez que uma atividade do projeto é selecionada para execução, ela é colocada neste cartão;
- Validando: Cartão em que tarefas já realizadas vão para aguardar validação, seja pelo orientador ou integrante da equipe não envolvido na sua execução;
- Finalizado: Onde as tarefas já totalmente efetuadas são armazenadas.

Observa-se que, diferentemente no fluxo clássico descrito anteriormente, serão utilizados seis cartões sob medida para as particularidades do projeto. A Figura 15 demonstra o quadro *Kanban* utilizado no processo de desenvolvimento do presente trabalho, demonstrando suas colunas e fluxo de trabalho no desenvolvimento.

Figura 15 – Quadro *Kanban* utilizado no desenvolvimento do presente trabalho.



Fonte: Autoria Própria.

3.3.1.2 Scrum

Schwaber e Sutherland (2010) definem o *Scrum* como um *framework* leve que ajuda pessoas, equipes e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.

O *Scrum* é construído em volta das pessoas que o utilizam. Não é o objetivo do *scrum* proporcionar regras minuciosas, por isso um processo mais prático é adotado, em que o foco é garantir a transparência, inspeção e adaptação.

Como observa Gouvea et al. (2016), apesar de poder ser usado para projetos em várias áreas de conhecimento, o *scrum* é utilizado amplamente por equipes de desenvolvimento de software em todo o mundo com objetivo de maximizar as tarefas e a produtividade em projetos com foco na qualidade do produto final.

De acordo com Schwaber e Sutherland (2010), o *scrum team* é composto por três papéis principais, sendo eles:

- ***Scrum Master***: Não é o gerente da equipe, mas serve a equipe, ajudando a remover as barreiras para seu sucesso, facilitando reuniões, apoiando e estimulando a prática do *scrum*.
- ***Product Owner***: Responsável por representar as necessidades dos *stakeholders* pelo gerenciamento eficaz do *Product Backlog*, que é a lista de requisitos e funcionalidades a serem desenvolvidas.
- **Equipe de Desenvolvimento**: São as pessoas do *scrum team* que estão comprometidas em criar qualquer aspecto de um incremento utilizável a cada *sprint*. Além

do desenvolvimento são responsáveis por criar o plano da *sprint* e o *Sprint Backlog*.

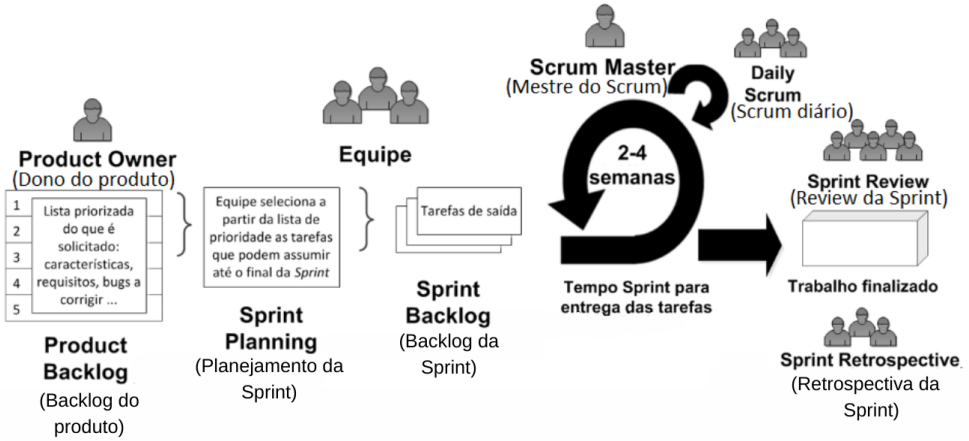
Esses papéis descrevem as principais responsabilidades dos participantes da equipe *Scrum*. Os papéis são flexíveis, podendo ser desempenhados por diferentes cargos já existentes na empresa.

O ciclo de desenvolvimento e validação do *scrum* é baseado em iterações, que são realizadas por todo o desenvolvimento do projeto. A Figura 16 ilustra esse ciclo de desenvolvimento com seus principais artefatos, etapas e papéis.

As *Sprints* são ciclos de desenvolvimento em que cada iteração resulta em incremento à aplicação. Com tempo fixo de um mês ou menos, pois quando o horizonte de uma *sprint* é muito longo, a meta da *sprint* pode se tornar inválida, a complexidade pode aumentar e o risco crescer. De acordo com Schwaber e Sutherland (2010) a *sprint* deve seguir alguns princípios:

- Nenhuma mudança é feita que coloque em risco a meta da *sprint*;
- A qualidade não diminui;
- O *product backlog* é refinado conforme necessário;
- O escopo pode ser esclarecido e renegociado com o *product owner* conforme mais é aprendido/compreendido.

Figura 16 – Ciclo de desenvolvimento do *scrum*.



Fonte: (GOUVEA et al., 2016) (tradução livre).

A *Sprint Planning* ou planejamento da *sprint* inicia a *sprint* ao definir o trabalho a ser realizado na *sprint*. O *Product Owner* garante que os participantes estejam preparados para discutir os itens mais importantes do *Product Backlog* e como eles são mapeados para a meta do produto. O *Product Owner* e os desenvolvedores discutem e selecionam quais

itens do *Product Backlog* serão incluídos na *sprint* atual (SCHWABER; SUTHERLAND, 2010).

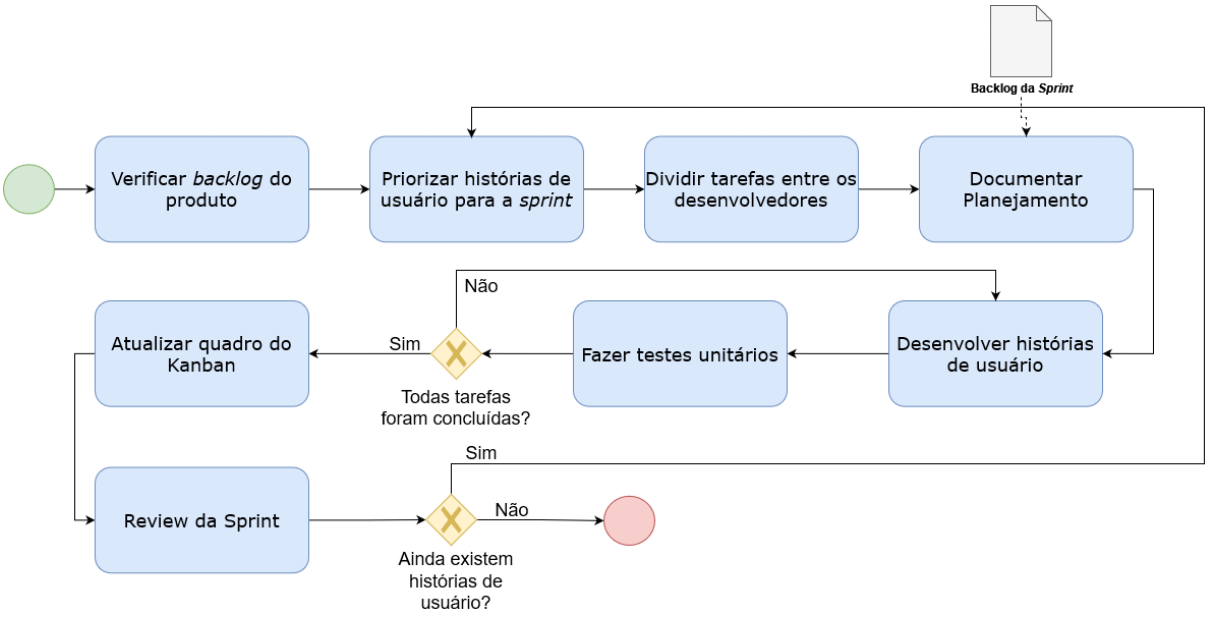
Durante o desenvolvimento, nas *sprints* são realizadas reuniões diárias chamadas *Daily Scrum*. Essas reuniões possuem horários fixos e geralmente não podem exceder 15 minutos de duração. O objetivo desses encontros é manter a equipe atualizada acerca do progresso das tarefas e produzir um plano de ação para o próximo dia de trabalho (GOUVEA et al., 2016).

Os artefatos do *scrum* representam trabalho ou valor. Um importante artefato é o *product backlog* ou *backlog* do produto. Schwaber e Sutherland (2010) definem *product backlog* como o uma lista ordenada e emergente do que é necessário para melhorar o produto. Os itens do *product backlog* que podem ser realizados pela equipe em uma *sprint* são considerados e preparados para seleção durante o planejamento da *sprint*.

Depois de finalizado o desenvolvimento, é realizado o *Sprint Review* ou Revisão da *Sprint*. Nela, a equipe entrega o produto testado. Neste momento o *Product Owner* verifica se os itens foram desenvolvidos de acordo com a especificação (GOUVEA et al., 2016). Também são analisados, a partir do *feedback* de todos os envolvidos, os pontos positivos e negativos que ocorreram durante a *sprint*, para que possivelmente possam ocorrer melhoras no processo.

Para execução da metodologia ágil no projeto, o fluxo ilustrado na Figura 17, representando as principais atividades realizadas durante a implementação desse trabalho, direcionará esta fase do desenvolvimento.

Figura 17 – Fluxo de desenvolvimento do software.



Fonte: Autoria Própria.

Para atender as necessidades e limitações do projeto, os artefatos e eventos do *scrum* foram adaptados. Dentre as práticas utilizadas estão as *sprints* de uma semana, reuniões diárias, o *backlog* do produto, da *sprint* e revisão da *sprint*.

Devido a limitação do tamanho da equipe, se fez necessária uma adaptação em relação aos papéis da equipe. O papel de *scrum master* será desempenhado pelo professor orientador, por ser quem está orientando o trabalho. Os papéis de *product owner* e equipe de desenvolvimento serão realizados de maneira conjunta pelos autores do trabalho, uma vez que se trata de um projeto sem um cliente específico.

3.4 Requisitos

Os requisitos de um sistema de software são as descrições do que o sistema deverá realizar, tanto os serviços que esse sistema oferece quanto suas restrições de funcionamento (SOMMERVILLE, 2011).

Os requisitos da aplicação foram elicitados pelos autores do trabalho. As técnicas de elicitação de requisitos utilizadas para obter o máximo de informação sobre a solução em questão foram o *brainstorming*, introspecção e método de personas.

3.4.1 Personas

O público alvo da aplicação (usuário) é o brasileiro de classe média, que vive em grandes cidades e trabalha longe de casa, o brasileiro usa transporte público, tem uma vida corrida e nem sempre está a par das condições climáticas futuras. É uma pessoa que sabe como usar um computador ou celular para navegar na internet. É uma pessoa que pode ter suas atividades cotidianas afetadas por uma forte chuva em um determinado dia.

Uma das personas é Marta Silva, mãe de família de 37 anos que trabalha em período integral como enfermeira e pode ter seu cotidiano prejudicado por condições climáticas adversas. Seu principal interesse é conseguir entender as condições climáticas no futuro de forma simples e de receber um alerta pelo celular em caso de chuvas.

A outra persona é o Marcos Raposo, que é um jovem adulto que trabalha como administrador na aplicação de previsão de chuvas, seus principais interesses são cuidar da gestão de usuários na plataforma e se assegurar que os dados meteorológicos das estações estão sendo atualizados.

3.4.2 Histórias de usuário

As histórias de usuário são criadas por meio de histórias narradas pelo usuário, que podem ser utilizadas para o planejamento do desenvolvimento do software e trazer detalhes não descritos nos requisitos (COHN, 2009). As histórias de usuário tem o objetivo

de realçar as necessidades e facilitar o entendimento dos desenvolvedores software durante sua fase de concepção e desenvolvimento (MINUZZI, 2007).

Quanto a elicitação de requisitos do software a ser executado nesse trabalho, foi identificado dois perfis de usuário, que seria o usuário da aplicação, que tem acesso de leitura às previsões e dados históricos, além de suas definições personalizadas de alarme e região padrão, e o usuário administrador, que faz a gestão de usuários da aplicação.

Foram dividias as principais funcionalidades do projeto em grupos chamados de *features*, onde cada *feature* contém um ou mais histórias de usuário associadas a ela. Zacharias, Cunha e Costa (2017) afirmam que as histórias de usuário elicitadas conseguem demonstrar uma funcionalidade (característica) que o sistema deve atender. Essas funcionalidades conseguem trazer aspectos ao desenvolvimento que vão além de conhecimentos técnicos, elas focam em transmitir o valor da aplicação desenvolvida para o usuário.

Para um levantamento inicial do projeto, foram identificadas as seguintes *features* e as principais histórias de usuário:

***Features*:**

- **Autenticação - F01**: Essa *feature* irá permitir ao usuário efetuar a conexão (*login*) e a desconexão (*logout*) na aplicação;
- ***Dashboard* de tempo - F02**: *Feature* que irá permitir ao usuário visualizar as condições atmosféricas atuais, se vai haver chuva e sua probabilidade de ocorrer nas próximas horas e nos próximos dias;
- **Pesquisa de tempo - F03**: Permitirá aos usuários consultar um dia específico (até um certo limite) para saber se irá ocorrer chuva, em qual probabilidade (%) por horário;
- **Alerta de tempo - F04**: Responsável por mostrar um alerta por notificação ao usuário quando o mesmo estiver em lugares em que vai chover;
- **Alimentação do Banco de Dados - F05**: Responsável por definir e implementar os processos pelos quais o banco de dados será alimentado com informações de previsões e dados históricos.

Histórias de usuário:

- **F01 - US01**: Eu como usuário, desejo me cadastrar na plataforma, para que eu possa acessar suas funcionalidades;
- **F01 - US02**: Eu como usuário, desejo alterar a minha senha, para que eu possa cadastrar uma senha mais fácil de ser memorizada;

- **F01 - US03**: Eu como usuário, desejo recuperar a minha senha, para que eu possa voltar a acessar a plataforma;
- **F01 - US04**: Eu como usuário, desejo um processo de autenticação do login, para que eu possa me sentir seguro de incluir as minhas informações na plataforma;
- **F02 - US05**: Eu como usuário, desejo visualizar as condições climáticas atuais, para que eu possa me informar acerca do estado meteorológico presentes e tomar decisões com embasamento;
- **F02 - US06**: Eu como usuário, desejo visualizar a probabilidade de precipitação nas próximas horas do dia e nos próximos dias, para ter informações acerca da chuva em um futuro próximo;
- **F03 - US07**: Eu como usuário, desejo buscar um dia específico no passado, para que eu possa acessar informações climáticas deste dia;
- **F03 - US08**: Eu como usuário, desejo buscar um dia específico no futuro, para que eu possa acessar informações climáticas deste dia;
- **F04 - US09**: Eu como usuário, desejo receber notificações baseadas em GPS quando for ocorrer uma chuva, para que eu possa estar alerta e me planejar adequadamente;
- **F05 - US10**: Eu como administrador, desejo inserir periodicamente no meu banco de dados informações temporais históricas das estações de clima;
- **F05 - US11**: Eu como administrador, desejo inserir periodicamente no meu banco de dados informações temporais sobre as previsões do modelo de *Machine Learning*.
- **F05 - US12**: Eu como administrador, desejo fazer a gestão e controle dos usuários que estão cadastrados na aplicação.

No contexto iterativo da metodologia ágil, buscando a limitação de quantidade de trabalho inicial em itens que não serão construídos ou podem sofrer mudanças, opta-se pelo uso de histórias de usuário inicialmente. Mas por meio das histórias de usuários serão realizadas discussões que terão como resultado um detalhamento mais técnico das funcionalidades de cada história, inclusive a criação de critérios de aceitação.

3.5 Arquitetura

De acordo com Martin (2019), uma boa arquitetura de software torna o sistema mais fácil de entender, desenvolver, de manter e de implantar. Dessa forma, torna-se essencial a escolha de um padrão arquitetural que seja adequado para a aplicação. Os princípios arquiteturais propostos estão descritos no decorrer desta seção.

O padrão mais tradicional de designs usados para conectar a interface do usuário com as camadas de aplicação e domínio é o *model view controler* (MVC). Esse padrão foi pioneiro na década de 70 e inspirou muitas das arquiteturas de interface de usuário que se seguiram (EVANS, 2003).

O princípio da inversão de dependência diz que módulos de alto nível não devem depender de módulos de baixo nível, mas ambos devem depender de abstrações (MARTIN, 2019). Deste modo, uma vez que no MVC muitas vezes geram um código altamente acoplado entre as camadas de *view* e *controller* (ALJAMEA; ALKANDARI, 2018), para a melhor qualidade da implementação proposto nesse trabalho, as adições de dois padrões arquiteturais serão incluídas para manter os princípios de um código mais limpo e adequado.

O padrão *repository* é uma abstração sobre armazenamento persistente, permitindo realizar o isolamento entre a camada *model* e a camada de *service*. O repositório não se importa com qual componente o está invocando, pois ele faz o que lhe é pedido (PERCIVAL; GREGORY, 2020). Assim é possível obter uma interface simples que pode ser controlada, obter a diminuição do acoplamento entre classes e maior facilidade na implementação de testes unitários.

O padrão da camada *service* reúne todas as funcionalidades de orquestração, como invocar o repositório se precisar gravar ou buscar dados, validar a entrada em relação ao estado do banco de dados, lidar com erros e executar o caminho feliz do código (PERCIVAL; GREGORY, 2020). O acesso à lógica de negócio ocorre por meio da *controller*, ou seja, a *controller* só está passando o trabalho para a camada de *service*, para que ela possa ficar simples e limpa.

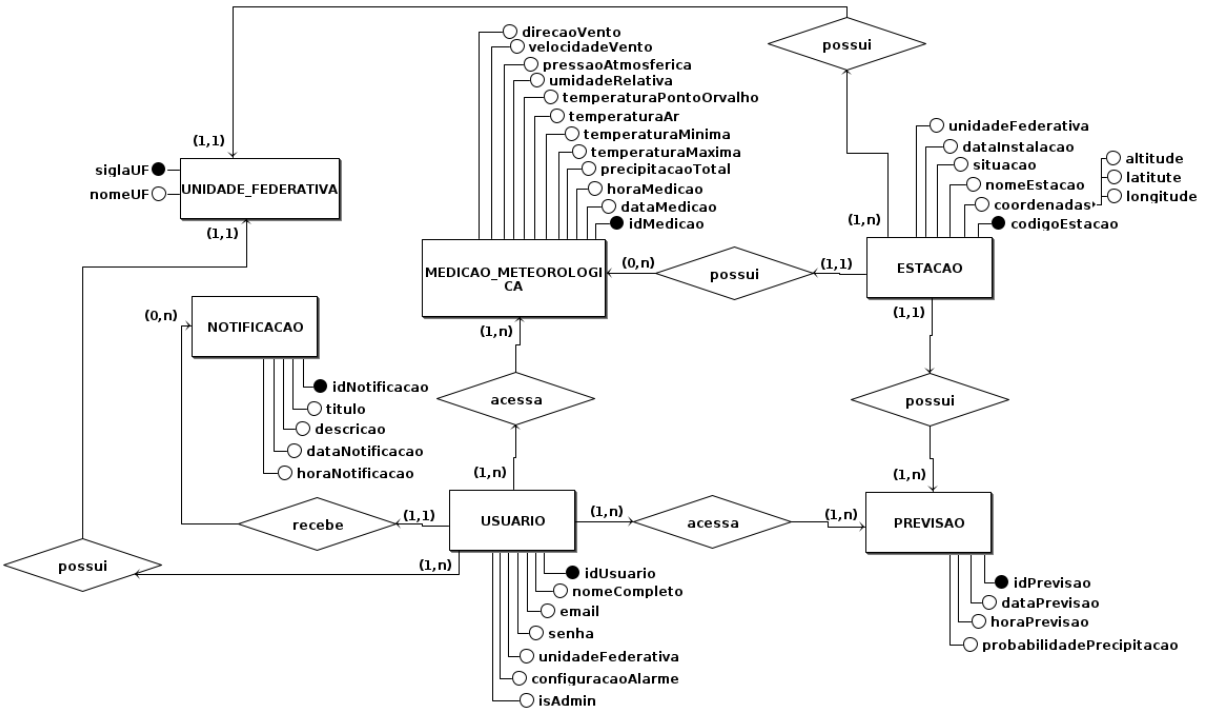
3.6 Base de Dados

3.6.1 Diagrama Entidade-Relacionamento

Para a representação da base de dados desse projeto, foi utilizado o Diagrama de Entidade-Relacionamento (DE-R) para a representação no nível conceitual.

Essa abordagem permite capturar e preservar algumas das semânticas importantes do mundo real e possibilita atingir um alto grau de independência de dados, tornando-se um artefato atraente para o projeto lógico de banco de dados (LING, 1985). A Figura 18 apresenta o DE-R do projeto

Figura 18 – Diagrama de Entidade-Relacionamento do projeto.



Fonte: Autoria Própria.

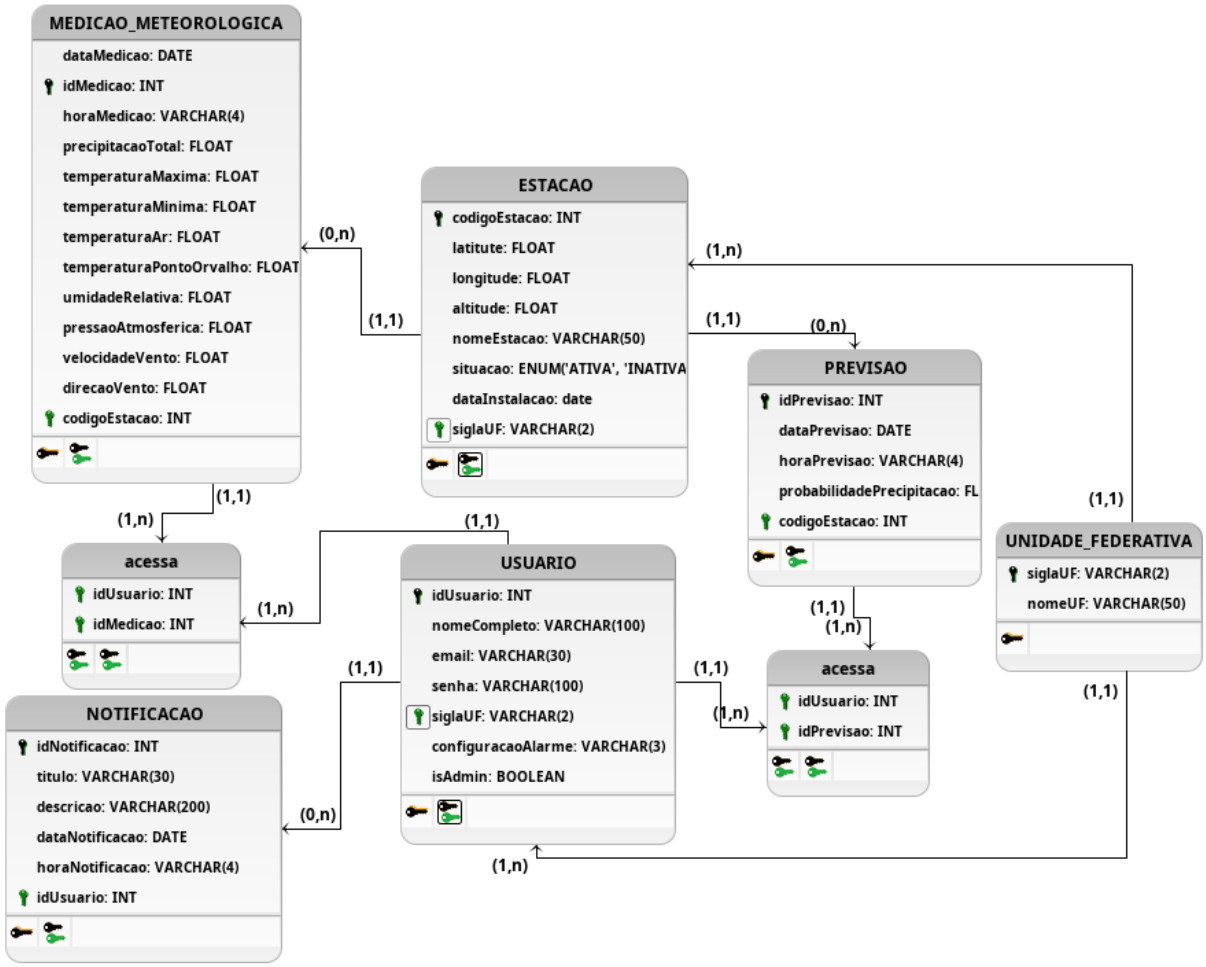
Nota-se na Figura 18 que a entidade central da aplicação é a de USUÁRIO devido a quantidade de seus relacionamentos e a importância de cada um deles. Esta entidade possui relacionamentos diretos com as entidades NOTIFICAÇÃO, MEDIÇÃO METEOROLÓGICA e PREVISÃO, sendo essas duas últimas relacionadas com a entidade ESTAÇÃO.

É possível observar neste nível da modelagem que um usuário que pode receber uma notificação, pode acessar tanto a base de dados com informações meteorológicas históricas, como a base de previsões. Cada registro nessas bases está associado a uma estação meteorológica.

3.6.2 Diagrama Lógico de Dados

Diferentemente do DE-R, o nível lógico de dados leva em conta algumas limitações e implementa recursos da modelagem com adequação de padrão e nomenclatura, definição de chaves primárias e estrangeiras, normalização, integridade referencial, entre outras (LEHMKUHL; EGER, 2013). A Figura 19 apresenta o Diagrama Lógico de Dados (DLD) elaborado a patir do DE-R deste projeto.

Figura 19 – Diagrama Lógico de Dados (DLD) do projeto.



Fonte: Autoria Própria.

A Figura 19 especifica as principais características da base de dados proposta a serem implementadas, correspondendo a um nível mais próximo da implementação (nível Lógico), em que tipos e precisão dos dados são definidos, formas coerentes da implementação dos relacionamentos, respeito as regras de negócio necessárias ao êxito do projeto, a normalização, etc. Além disso, o DLD também mostra como as estruturas de armazenamento de dados serão constituídas, realmente, em suas características físicas implementadas no SGBD.

3.7 Suporte Tecnológico

Nesta seção serão apresentados os principais recursos tecnológicos usados no desenvolvimento deste trabalho, e como eles foram utilizados.

3.7.1 Trello

O Trello é uma ferramenta de organização e gerenciamento de projetos baseado na metodologia *Kanban* (TRELLO, 2022). Através de seus cartões, o Trello permite o gerenciamento das tarefas descritas.

A ferramenta foi utilizada no projeto para dar apoio, principalmente a aplicação da metodologia *Kanban* na elaboração, desenvolvimento e implementação do projeto proposto.

3.7.2 Diagrams.net

O Diagrams.net é uma ferramenta *online* que permite a criação de diagramas. Esta é a ferramenta de diagramação baseada em navegadores da Internet mais utilizada do mundo (DIAGRAMS.NET, 2022).

Todos os processos de diagramação deste trabalho foram realizadas através da plataforma Diagrams.net.

3.7.3 Git

O Git é uma ferramenta de controle de versionamento de projetos, que pode ser utilizado para lidar com projetos pequenos até projetos muito grandes com rapidez e eficiência (GIT, 2022).

Foi utilizado Git para versionamento de todo o código fonte escrito no desenvolvimento da aplicação (software).

3.7.4 GitHub

O GitHub é uma plataforma de hospedagem de código-fonte e arquivos que utiliza a tecnologia do Git para controle de versão, além de possuir também ferramentas sociais, de colaboração e outras que possibilitam a automação de testes e integração contínua (GITHUB, 2022).

Neste projeto, o GitHub foi utilizado devido as suas funcionalidades de armazenamento, versionamento e compartilhamento de código juntamente com sua integração com Git.

3.7.5 React

O React é uma biblioteca JavaScript utilizada para criar interfaces de usuário de maneira declarativa baseado em componentes e multiplataformas (REACT, 2022). Será utilizado neste projeto para o desenvolvimento da interface dos usuários.

3.7.6 Node.js

O Node.js é um ambiente de execução de código JavaScript de maneira assíncrona, feito para executar aplicações *online* e escaláveis (NODE.JS, 2022).

Neste projeto o Node.js foi usado para executar todo o código JavaScript do *Backend*.

3.7.7 Python

O Python é uma linguagem de programação de alto nível (PYTHON, 2022). Neste projeto o Python será utilizado em todo o código fonte relacionado com o *Machine Learning*.

3.7.8 brModelo

O brModelo é uma ferramenta CASE (do inglês *Computer-Aided Software Engineering*) de código aberto que auxilia no processo de modelagem de bancos de dados (BRMODELO, 2022).

Foi utilizado o brModelo neste projeto como ferramenta de apoio da modelagem de dados com o desenvolvimento do Diagrama de Entidade-Relacionamento (DE-R) apresentado na Seção 3.6.1 e o Diagrama Lógico de Dados (DLD), mostrado na Seção 3.6.2.

3.7.9 PostgreSQL

O PostgreSQL é um SGBD relacional de código aberto que utiliza a Linguagem de Consulta Estruturada, do inglês *Structured Query Language* (SQL). Amplamente utilizado em aplicações *web* (POSTGRESQL, 2022).

Neste projeto o PostgreSQL foi utilizado para o gerenciamento de todos os dados armazenados e produzidos de maneira relacional, correspondendo ao recurso empregado na camada de persistência do projeto.

4 Desenvolvimento

Este capítulo refere-se a execução da proposta apresentada no Capítulo 3, correspondendo às metodologias de desenvolvimento apresentadas nas Seções 3.2 e 3.3.1. Serão abordados como foram os processos de desenvolvimento do modelo de ML e a construção da aplicação interativa (software) para que os usuários façam uso do modelo.

4.1 Modelagem da *previsão*

Como abordado na seção 3.2, na criação do modelo de *machine learning* foi utilizada uma metodologia composta de quatro fases. A seguir será apresentada a documentação em cada uma delas, assim como as decisões e suposições feitas durante o período de desenvolvimento e implementação do projeto proposto.

4.1.1 Gestão de Dados

Quanto a aquisição dos dados, foram elicitadas 3 fontes de dados possíveis que poderiam nos fornecer as informações de clima possivelmente relevantes, a descrição dessas fontes pode ser encontrada na Seção 2.5. Após verificar o estado das bases de dados possíveis, foi escolhida a base do INMET, tendo em vista que ela é que possuia melhor relação de campos não nulos (vazios ou sem dados), variáveis de interesse e granularidade dos dados.

A plataforma do INMET oferece duas maneiras de aquisição dos dados, sendo a primeira por meio de um formulário em página web, em que podem ser selecionadas opções como granularidade, variáveis, estações, entre outras. Os dados são enviados para o *e-mail* previamente indicado no formulário, sendo necessário confirmar a requisição por *e-mail* e após a confirmação os dados são colocados na fila para processamento e depois de um tempo (cerca de 1 a 10 minutos) eles são recebidos no *e-mail* informado no formato CSV (Valores Separados por Vírgula, do inglês *Comma Separated Values*), que é um arquivo de texto onde os valores são separados por vírgulas.

A segunda maneira acontece por meio do uso de uma API, que retorna dados para uma determinada estação e intervalo de datas, baseados nos parâmetros da *query* (consulta) da requisição. Na API a granularidade sempre é horária. O maior intervalo de extração possível é anual e os dados são retornados em questão de poucos segundos no formato JSON, que é um formato da linguagem de programação *javascript* para troca de dados de forma simples e rápida entre sistemas.

Apesar de possuir um número de opções de customização dos dados limitado, a aquisição dos dados pela API foi escolhida, principalmente por facilitar a automatização do processo.

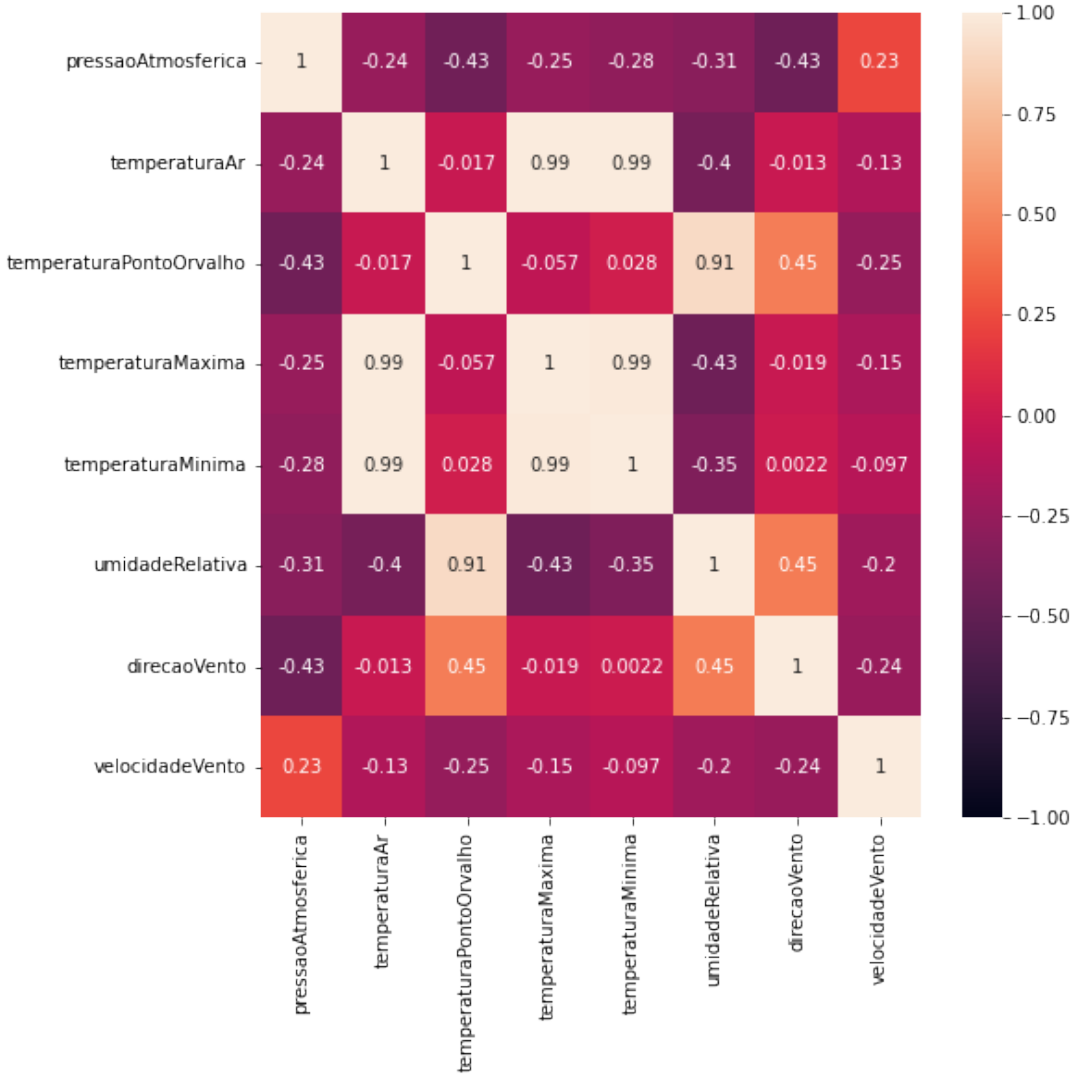
Foi decidido que a aplicação seria focada em pessoas que vivem em grandes cidades. O INMET possui dezenas de estações espalhadas por todo o Brasil, após selecionar algumas capitais do Brasil buscando aquela com maior quantidade de tuplas e menos informações nulas, nesse processo a cidade escolhida foi Brasília. Essa estação conta com cerca de 195 mil tuplas, a partir do ano 2000.

Os dados foram adquiridos através de um *script* em *Python* que realiza a requisição e insere os dados no banco de dados após pré-processamento. Após uma análise das variáveis de interesse, percebeu-se que muitas apresentam um alto grau de correlação com outra variável, sendo assim, foram removidas algumas das variáveis, sobrando as seguintes: pressão atmosférica, velocidade do vento, temperatura do ar, umidade relativa e direção do vento. As variáveis de temperatura do ponto de orvalho, temperatura máxima e temperatura mínima foram retiradas por possuírem alto grau de correlação com outras variáveis, que podem ser vistas na Figura 20. Também foram criadas colunas para proporcionar ao modelo um entendimento do tempo, sendo essas colunas integradas a base de dados: dia do ano, mês e hora.

Outro processamento realizado diz respeito a criação de outras colunas indicando se choveu no dia atual, que são necessárias para enquadrar a base de dados em um problema de classificação. É interessante esclarecer que na definição do projeto para tal coluna foi decidido que o valor de precipitação da hora (em milímetros) sendo maior que zero indica que choveu.

A partir dos dados escolhidos da base do INMET, e a integração de algumas novas colunas, seria possível seguir dois diferentes caminhos para a previsão de chuva: a univariada e a multivariada. Na univariada os atributos obtidos seriam examinados um de cada vez, cada um tendo sua própria previsão sendo gerada separadamente. Na previsão multivariada diversos atributos seriam utilizados para realizar a previsão de uma variável que interfere na possível precipitação. Assim, o trabalho realizou testes sobre as duas estratégias com diferentes métodos de previsão, que serão apresentados com maiores detalhes a partir da Seção 4.1.2.

Figura 20 – Matriz de correlação das variáveis.



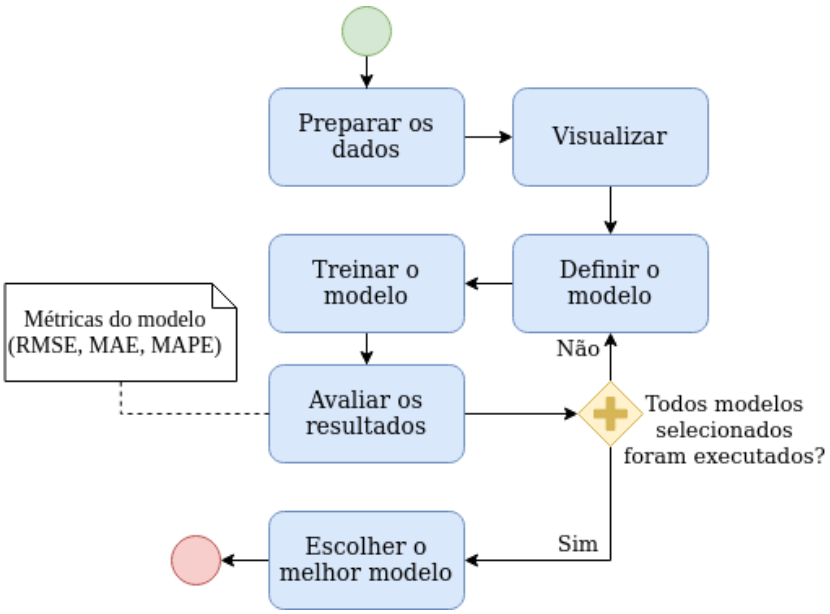
Fonte: Autoria Própria.

4.1.2 Modelo de aprendizado

Como apresentado anteriormente na Seção 4.1.1, os dados foram transformados para serem encaixados em um problema de classificação com aprendizado supervisionado. Nesse caso a classe que seria prevista indica se choveu (identificada pelo número 1) ou não (número 0).

O fluxo de atividades estabelecido para avaliar as opções de técnicas e modelos está ilustrado na Figura 21.

Figura 21 – Fluxo de atividades para avaliação de modelos/técnicas de previsão.



Fonte: Autoria Própria.

- **Preparar os dados**: O primeiro passo na previsão é preparar os dados no formato correto. Isso inclui realizar o tratamento dos dados e separar o *dataset* (fração da base de dados utilizada no treinamento, teste e validação de ML) entre conjuntos de treinamento, teste e validação.

- **Visualizar**: Plotar os dados é um passo essencial na compreensão dos dados. Observar os dados permite identificar padrões comuns e, posteriormente, especificar um modelo apropriado.

- **Definir o modelo**: Definir o modelo de previsão de acordo com as especificidades das *features* de entrada e saída escolhidas, podendo ser univariadas ou multivariadas.

- **Treinar o modelo**: Aplicar o modelo no conjunto de dados de treinamento.

- **Avaliar os resultados**: Avaliar os resultados nos *dataset* de testes, de acordo com métricas como a raiz quadrada do erro médio (RMSE).

- **Escolher o melhor modelo**: Escolher o melhor modelo e partir para a próxima fase do ciclo de vida da modelagem da previsão.

Os modelos selecionados para aplicação da previsão são apresentados na Tabela 4, sendo que eles foram categorizados em três tipos: clássicos, de *machine learning*, *deep learning* e quanto à natureza, isto é, o número de variáveis de entrada para prever a chance de precipitação (univariados ou multivariados).

Tabela 4 – Divisão do *dataset*.

Nome	**Tipo**	**Natureza**	**Granularidade**
Média	Clássico	Univariado	Por Dia
Ingênuo	Clássico	Univariado	Por Dia
Ingênuo Sazonal	Clássico	Univariado	Por Dia
LightBGM	*Machine Learning*	Univariado	Por Dia
LightBGM	*Machine Learning*	Multivariado	Por Dia
XGBoost	*Machine Learning*	Multivariado	Por Hora
Random Forest	*Machine Learning*	Multivariado	Por Hora
SVM	*Machine Learning*	Multivariado	Por Hora
LSTM	*Deep Learning*	Univariado	Por Hora

Fonte: Autoria própria.

Os modelos clássicos "ingênuos"(Média, Ingênuo e Ingênuo Sazonal) são essenciais para estabelecer o ponto de comparação inicial das previsões. Porém, os modelos clássicos têm a limitação de serem necessariamente univariados. Outra dificuldade é tratar de dados de granularidade horária, pois o *dataset* de precipitação é desbalanceado e técnicas como o *resampling* não se aplicam nesse caso.

Para a estratégia de *machine learning* foram escolhidos os modelos LightGBM, XGBoost, Random Forest e SVM. Já na área *deep learning* foi escolhido o LSTM.

Os dados foram divididos em treinamento, validação e teste. Sendo destinados o equivalente a três anos (1095 dias ou 23280 horas) para os dados de teste, o restante foi usado para o treinamento.

Para a comparação dos modelos optou-se pela métrica RMSE, que é a raiz quadrada do erro quadrático médio entre os valores previstos e reais. Um benefício de usar o RMSE é que a métrica que ele produz é em termos da unidade que está sendo prevista, nesse caso, de 0 (não ocorre chuva) até 1 (ocorre chuva). A Tabela 5 apresenta a comparação das técnicas quanto ao RMSE, com os dados de treinamento da estação meteorológica de Brasília.

Tabela 5 – Comparação da performance dos modelos.

Nome	**Natureza**	**RMSE**
Média	Univariado	0.536
Ingênuo	Univariado	0.357
Ingênuo Sazonal	Univariado	0.500
LightBGM	Multivariado	0.336
XGBoost	Multivariado	0.320
Random Forest	Multivariado	0.325
SVM	Multivariado	0.336
LSTM	Univariado	0.369

Fonte: Autoria própria.

Baseando-se nessa comparação, observa-se que muitos dos modelos não conseguem desempenhar acima do método Ingênuo, sendo que os 3 modelos de *machine learning*, o Random Forest (RF), XGBoost (XGB) e SVM foram os que obtiveram melhor performance.

Os modelos RF e XGB são baseados em árvore, modelos baseados em árvore consistem em uma ou mais instruções condicionais *if-then* aninhadas para os preditores que particionam os dados. Dentro dessas partições, um modelo é usado para prever o resultado. A Figura 22 apresenta um exemplo árvore simples.

Figura 22 – Árvore de decisão simples.

```
if Preditor A >= 1.7 and Preditor B >= 202.1 then Resultado = 1.3
if Preditor A >= 1.7 and Preditor B < 202.1 then Resultado = 5.6
if Preditor A < 1.7 then Resultado = 2.5
```

Fonte: Autoria Própria.

As instruções *if-then* geradas por uma árvore definem uma rota exclusiva para um nó terminal para qualquer amostra. Uma regra é um conjunto de condições *if-then* criadas por uma árvore. No exemplo da Figura 22, são três regras.

Essa forma de organizar uma base de conhecimento e executar predições baseadas em regras é semelhante aos Sistemas Baseados em Conhecimento (SBC). Um SBC é um dos principais membros da família de tecnologias provenientes da Inteligência Artificial. Este tipo de sistema de computador (software), usa e gera conhecimento a partir de dados, informações e conhecimento (SAJJA; AKERKAR, 2010). Dessa forma, um SBC pode atuar como um especialista de determinada área, sem perder tempo, a qualquer hora e em qualquer lugar colaborando com o que seja necessário.

Apesar de essencialmente utilizarem técnicas baseadas em árvores, os modelos RF e XGB são diferentes. As florestas aleatórias são uma coleção de árvores de decisão

com um único resultado agregado, resultando em menos variância para a predição. Já o XGBoost, apesar de também funcionar com um conjunto de árvores de decisão, constrói uma árvore por vez, enquanto a RF constrói cada árvore independentemente. Para mais informações sobre técnicas de aumento do gradiente, a Seção 2.2.4 do referencial teórico apresenta uma explicação detalhada.

Para comparar os 3 melhores algoritmos da Tabela 5, uma análise com outras métricas foi executada. As métricas de interesse foram o *recall* e a taxa de falsos positivos, no inglês *false positive rate* (FPR). O *recall* é uma métrica que quantifica o número de previsões positivas corretas feitas de todas as previsões positivas que poderiam ter sido feitas, seu cálculo está representado na Equação 4.1. Já a FPR é a probabilidade que um resultado positivo vai receber quando o valor verdadeiro é negativo, e sua formula está na Equação 4.2. Ambas métricas são utilizadas em diversos trabalhos que buscam realizar previsão de precipitação orientada a dados (HUI et al., 2018; MANANDHAR et al., 2019; SHILPA et al., 2018).

$$recall = TP/(TP + FN) \tag{4.1}$$

$$FPR = FP/(TN + FP) \tag{4.2}$$

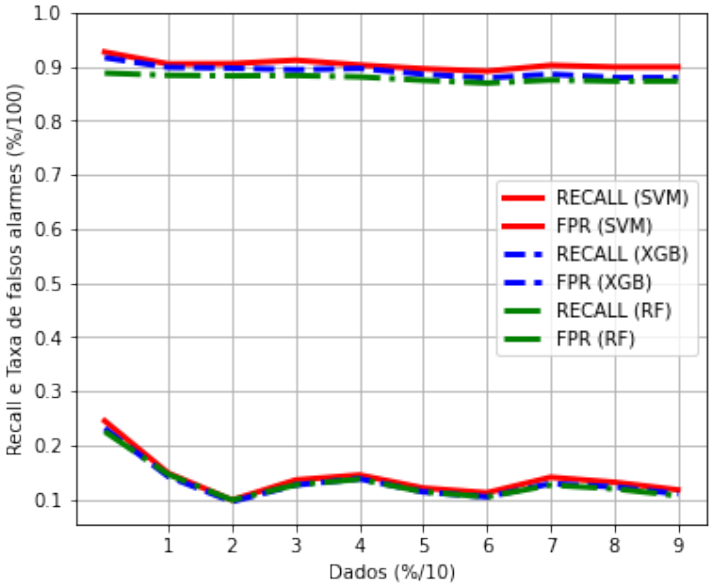
Onde TP representa uma previsão "verdadeiro positiva", isto é, o número de vezes que o modelo acerta em dizer quando vai chover. A variável FN ou falso negativo representa o número de vezes em que choveu e o modelo indicou que não iria chover. FP ou falso positivo indica o número de vezes em que não choveu e o modelo afirmou que iria chover e finalmente, TN ou verdadeiro negativo que aponta a quantidade de vezes que não choveu e o modelo acertou que não iria chover. Os resultados da aplicação das métricas são apresentados na Tabela 6. Uma comparação entre elas à medida que o volume de dados vai aumentando é apresentada na Figura 23.

Tabela 6 – Avaliação das métricas.

Nome	***Recall***	**FPR**
SVM	0.883	0.113
Random Forest	0.865	0.101
XGBoost	0.864	0.104

Fonte: Autoria própria.

Figura 23 – *Recall* e Taxa de Falso Positivo das três métricas conforme mais dados são testados.



Fonte: Autoria Própria.

Observa-se na Figura 23 como o *recall* e FPR aumenta durante o treinamento de cada modelo conforme mais dados são alimentados. Apesar da similaridade, o SVM demonstra maior desempenho.

Pelos testes feitos e exibidos da Tabela 6, de forma geral o modelo SVM teve o melhor desempenho, também vale a pena notar que o SVM é uma técnica comumente usada em problemas de aprendizado supervisionado que lidam com problemas de geociência (LARY et al., 2016). Por isso ele será utilizado nas próximas etapas do desenvolvimento do ciclo de vida. Para uma descrição em alto nível do funcionamento do algoritmo, foi elaborada a Seção 2.2.3.

Entretanto, como o método é multivariado, que usa variáveis como umidade e temperatura do dia para realizar a previsão da chuva, se faz necessário também realizar a previsão destas outras variáveis previamente. Para isso, foram utilizadas as técnicas listadas na Tabela 4, mas empregando apenas métodos univariados e o LightGBM, que alcançou a melhor previsão de 4 horas nas variáveis. Tendo em vista este fato, será adotado o LightGBM para a previsão de todas as variáveis que se usou para prever se vai ou não chover. A Seção 2.2.4 apresenta esse algoritmo.

4.1.3 Verificação do Modelo

A partir da escolha do modelo de *machine learning* para as variáveis listadas na Tabela 4, foi feita uma previsão delas de 4 em 4 horas, até chegar ao limite do *dataset*, indo de 2020 até agosto de 2022. Essas previsões de 4 em 4 horas são concatenadas juntas

e o modelo SVM faz a predição se vai chover ou não baseando-se nas variáveis previstas anteriormente. A criação dos dados concatenados e a previsão de chuva levou 24 horas e 56 minutos e os resultados estão representados na Tabela 7.

Tabela 7 – Resultado do SVM.

Nome	*Recall*	*FPR*
SVM	0.852	0.122

Fonte: Autoria própria.

O resultado apresentado acima na Tabela 7, se mostrou relativamente positivo em relação ao resultado anterior da Tabela 6, uma vez já existia a expectativa de piora das métricas devido a utilização de variáveis originárias da predição univariada.

Comparações com outros trabalhos que também utilizaram abordagens orientadas a dados para a previsão de chuva podem ser feitas, apesar destes trabalhos tratarem de dados de origem, quantidade e variáveis de entrada diferentes. Hui et al. (2018), utilizando um algoritmo próprio, fez a previsão para duas estações, na qual a estação de Singapura, prevendo os dados de 2016, conseguiu uma média de 90.7 e 50.2 para *recall* e falsos positivos respectivamente, na segunda estação, feita acerca de dados de São Luis em 2014 e 2015, obteve uma média de 84.7 e 37. Já Shilpa et al. (2018), utilizando o SVM, alcançou um resultado mais semelhante ao deste trabalho, com 85.8 e 28.5 para *recall* e falsos positivos para a previsão do ano de 2012 em Singapura.

4.1.4 Implantação do Modelo

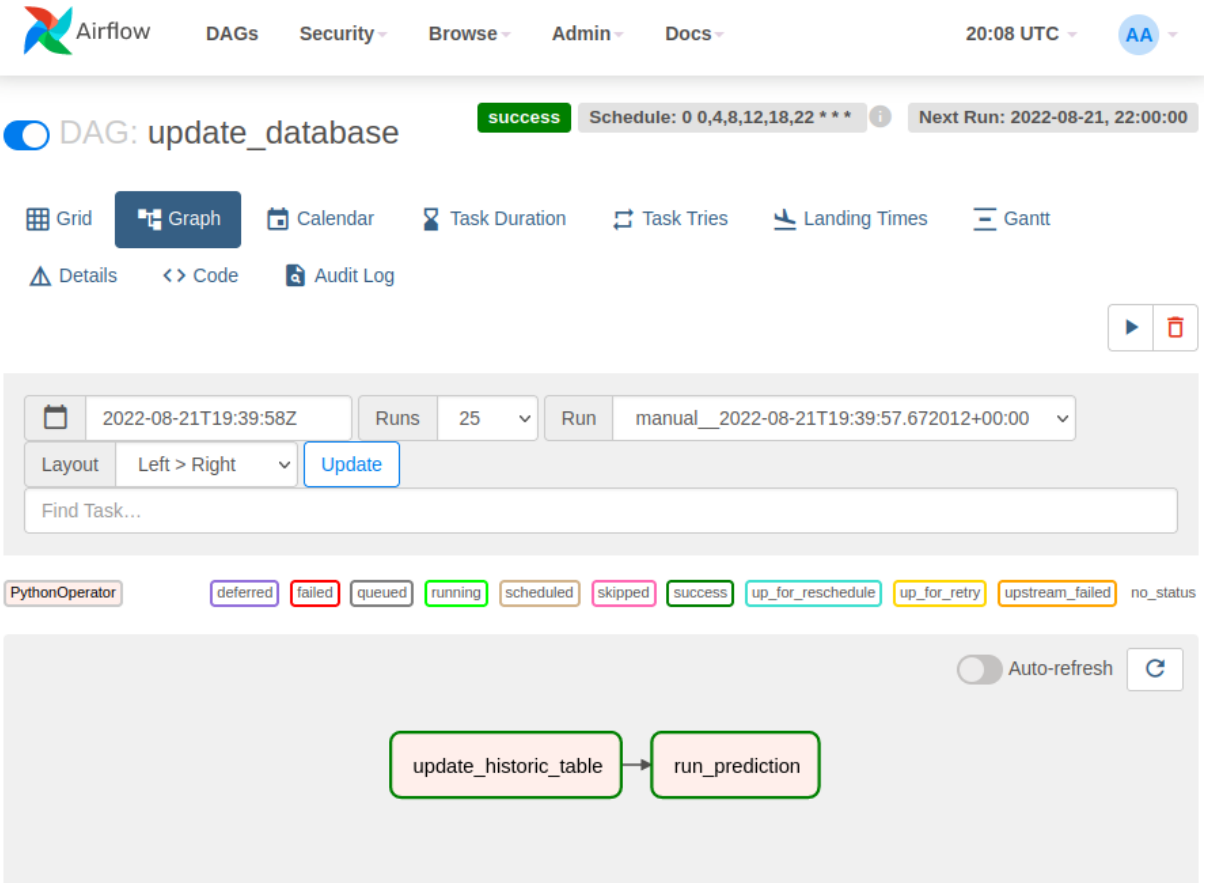
Para a implantação do modelo existem três *scripts* principais, sendo que o segundo e o terceiro rodam de hora em hora, um depois do outro. O primeiro é usado para a aquisição inicial dos dados da base do INMET, faz o processamento dos dados, como alteração dos nomes das colunas e geração de novas colunas e encerra salvando os dados no banco de dados *Postgres*. O segundo tem o intuito de atualizar essa base histórica a partir do último registro no banco de dados do projeto. O terceiro *script* primeiramente roda o modelo do LightBGM para a previsão univariada das variáveis que são entradas do modelo principal, depois o modelo principal SVM é executado e os resultados são salvos na tabela adequada. Este terceiro *script* foi adicionado no Apêndice B.3 deste trabalho.

Anteriormente na proposta, na Seção 3.5 foi apresentado que seria utilizada uma API no *framework* Flask para a raspagem dos dados meteorológicos e execução do modelo. No entanto, durante o desenvolvimento, decidiu-se que essas tarefas não precisariam ser realizadas por uma API, apenas os *scripts* sendo executados entre intervalos horários resolveriam o problema, diminuindo assim a complexidade da arquitetura.

Portanto, para orquestração dos *scripts* foi utilizado o Apache Airflow, nele é possível criar fluxos de trabalho na forma de gráficos acíclicos direcionados (DAGs). Também

possui uma interface, que facilita a visualização de pipelines em execução, o monitoramento do progresso e a solução de problemas quando necessário. A Figura 24 apresenta o DAG programado para rodar uma vez por hora e composta por duas tarefas, de atualizar os dados históricos e de executar os modelos de *machine learning*. Esse DAG leva cerca de 5 minutos para finalizar a execução.

Figura 24 – Captura da tela de um DAG na interface de usuário do Apache Airflow.



Fonte: Autoria Própria.

4.2 Fluxo de Desenvolvimento

Esta seção se refere à execução do fluxo de desenvolvimento apresentado na Seção 3.3.1.2. Será abordado como foi o processo de desenvolvimento do sistema de previsão proposto, assim como as dificuldades e alterações realizadas sobre a proposta inicial.

Também foi feito uma priorização das histórias de usuário seguindo a técnica *MoSCoW*, em que cada história do *backlog* deve-se atribuir uma das seguintes palavras ou expressões: *MUST* ("Deve", as histórias atribuídas nesta expressão devem ser realizadas senão o trabalho é considerado um fracasso), *SHOULD* ("Deve", aqui os itens são muito importantes mas não essenciais), *COULD* ("Poderia", aqui os itens são desejáveis, mas não

necessários e podem ser excluídos caso haja restrições de tempo ou recursos) e *WOULD* ("Seria", nesta atribuição os itens menos críticos para o funcionamento do projeto que não estão incluídos no cronograma de desenvolvimento). Além disso, foram estabelecidos critérios de aceitação para cada história. O *backlog* é apresentado na Tabela 8.

Tabela 8 – *Backlog* do produto com critérios de aceitação e priorização.

História	**Critérios de aceitação**	**Priorização**
US01 - Cadastro de usuário	- O Usuário deve ser capaz de se cadastrar com e-mail e senha. - Deve haver um campo de confirmação de senha.	SHOULD
US02 - Alteração de senha	- O usuário deve ser capaz de alterar sua senha uma vez que esteja logado na aplicação. - O usuário deve colocar sua senha antiga antes de alterar a senha.	COULD
US03 - Recuperação de senha	- O usuário deve receber um e-mail com um link para a página de recuperação da senha. - Deve haver um campo de confirmação de senha.	COULD
US04 - Realizar login	- O usuário deve conseguir fazer login com as credenciais cadastradas.	SHOULD
US05 - Visualizar condições climáticas atuais	- O usuário deve conseguir ver as informações climáticas do dia atual na página principal.	MUST
US06 - Visualizar previsões climáticas futuras	- O usuário deve conseguir ver as informações climáticas das próximas horas na página principal.	MUST
US07 - Buscar condição de um dia específico no passado	- O usuário deve ser capaz de visualizar informações climáticas de um intervalo de datas no passado. - Essa visualização deve ser apresentada por meio de uma tabela em uma página diferente da principal.	COULD
US08 - Buscar previsão de um momento no futuro	- O usuário deve ser capaz de visualizar informações climáticas de um intervalo de datas no futuro. - Essa visualização deve ser apresentada por meio de uma tabela em uma página diferente da principal.	COULD
US09 - Receber notificações de chuva baseada em GPS	- O usuário deve ser capaz de configurar alertas de chuva a partir da probabilidade escolhida. - O usuário deve receber o alerta assim que a previsão seja realizada.	SHOULD
US10 - Inserir no banco de dados informações climáticas históricas	- Os dados devem ser extraídos de hora em hora	MUST
US11 - Inserir no banco de dados previsões realizadas	- O modelo deve ser executado logo após a alimentação dos dados históricos. - As previsões devem ser feitas de hora em hora.	MUST
US12 - Gestão de usuários cadastrados	- O gestor deve conseguir visualizar, alterar, adicionar e excluir contas de usuário. - O gestor deve conseguir alterar a estação de um usuário.	WOULD
TS01 - Criação do modelo de machine learning	- O modelo deve conseguir performar positivamente quando comparada com trabalhos acadêmicos similares. - Devem ser feitas comparações entre vários tipos de técnicas de previsão.	MUST

Fonte: Autoria Própria.

Para melhor rendimento das atividades dos autores optou-se por realizar *sprints* de duas semanas, adaptando o planejamento feito anteriormente na proposta. A tarefa

que envolvia o *machine learning* foi sendo desenvolvida no decorrer de várias *sprints*.

Quanto aos testes, foram executados testes unitários em todos os *services* do *Backend* (NodeJS), onde fica a lógica de negócio em si. Também foram feitos testes de integração testando chamadas à API no *Backend*.

Tendo em vista o cronograma apresentado na Tabela 3, a atividade de "Teste do MVP com usuários"não foi realizada, tendo em vista a ênfase do trabalho estar na criação do modelo de *machine learning*. Tendo isso em mente, as funcionalidades chaves do *Frontend* foram implementadas, que são aquelas que se relacionam ao modelo, como será apresentado nas *sprints*.

4.3 *Sprints* de desenvolvimento

Esta seção foi dividida por *sprints* presentes no *backlog* do produto apresentado na Seção 4.2, sendo apresentados os artefatos gerados e uma breve discussão.

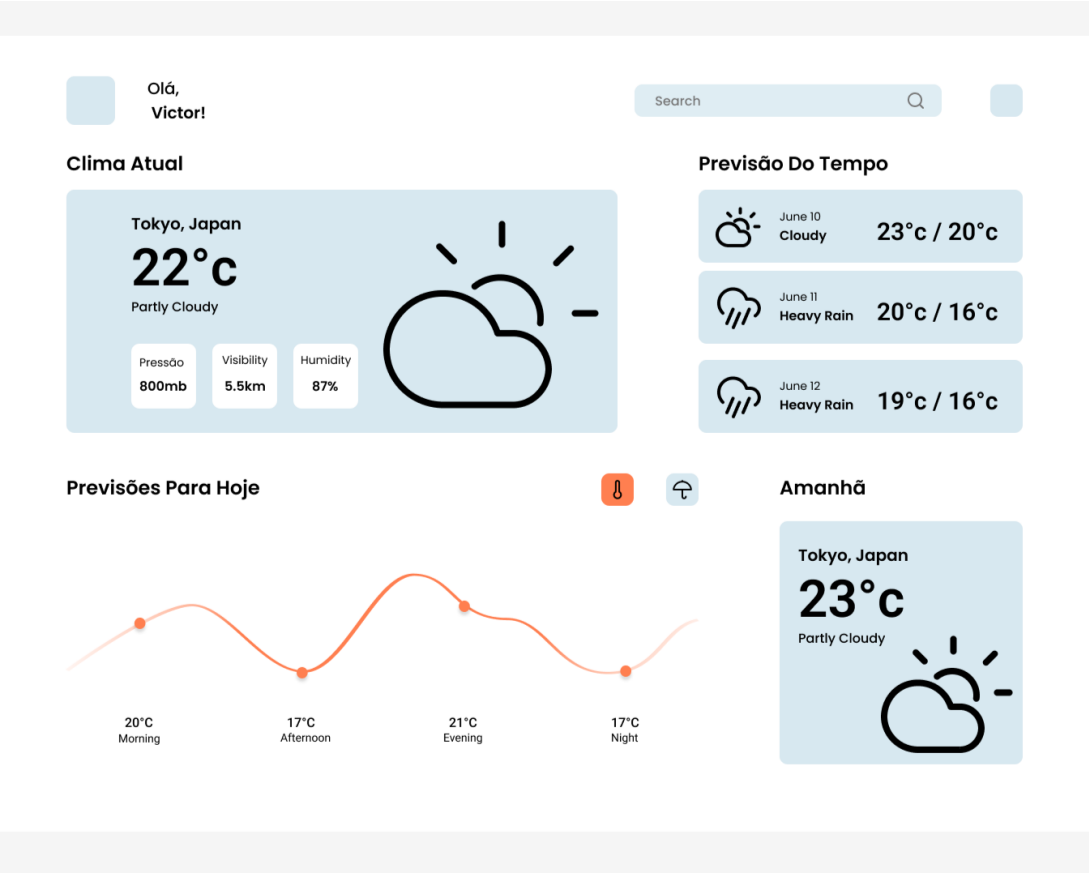
4.3.1 1ª *Sprint*

Na primeira *Sprint*, inicialmente foi realizada a avaliação das bases de dados disponíveis e escolha da base de dados do IMNET. Em seguida, foram desenvolvidos os *scripts* em Python que realizam a inserção e atualização dos dados meteorológicos históricos para o SGBD PostgreSQL. Essas atividades estão relacionadas com as histórias de usuários US07 e US10.

Na parte do *Frontend* foi feito um protótipo de alta fidelidade de um *dashboard* que apresenta informações climáticas na ferramenta Figma. E, essa atividade está associada às histórias de usuário US05 e US06. A Figura 25 apresenta o seu protótipo.

A Tabela 9 mostra o *backlog* da primeira *sprint*, com as histórias de usuário escritas de forma condensada, e o estado atual de conclusão para o *Backend* e *Frontend*. Além disso, foi adicionada uma história técnica referente a criação do modelo de *machine learning* já que esta não pode ser visualizada diretamente pelo usuário mas somente pelos desenvolvedores.

Figura 25 – Protótipo de alta fidelidade da *dashboard* da aplicação.



Fonte: Autoria Própria.

Tabela 9 – *Backlog* da 1ª *sprint*.

História de Usuário	**Backend Concluído?**	*Frontend Concluído?*
US01 - Cadastro de usuário	Sim	Sim
US04 - Realizar login	Sim	Sim
US05 - Visualizar condições climáticas atuais	Não	Em andamento
US06 - Visualizar previsões climáticas futuras	Não	Em andamento
US07 - Buscar condição de um dia específico no passado	Sim	Não
US10 - Inserir no banco de dados informações climáticas históricas	Sim	Não se aplica

Fonte: Autoria própria.

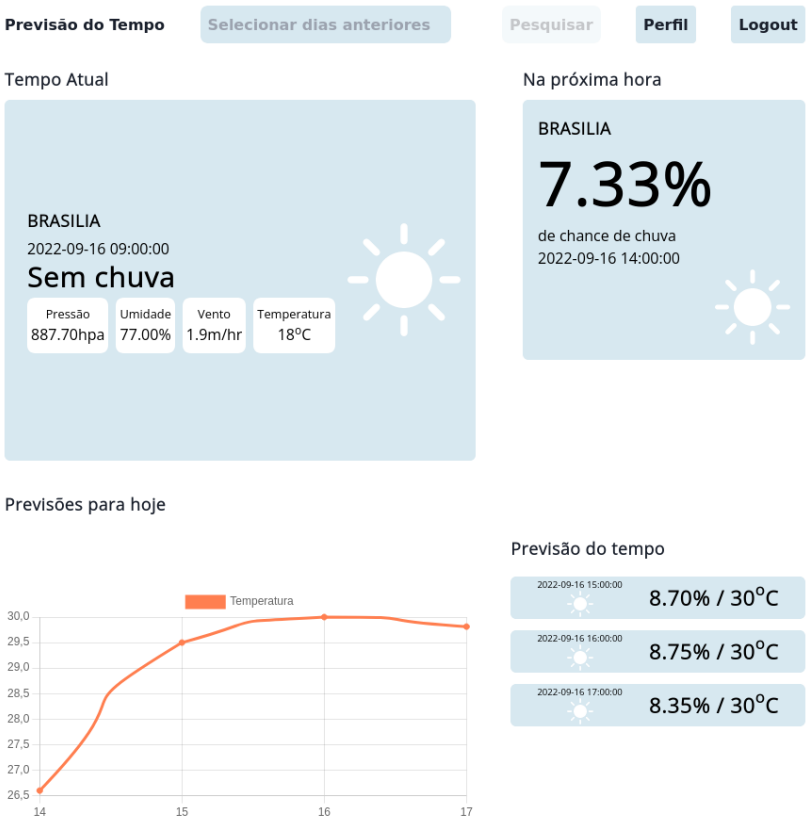
4.3.2 2ª *Sprint*

Após a aquisição dos dados e uma rotina de alimentação do banco de dados foi estabelecida, aconteceu o momento de começar o desenvolvimento do modelo de *machine learning*. Esse desenvolvimento é o núcleo deste trabalho e corresponde a tarefa que de-

manda mais esforço para o sucesso do projeto. Por isso, ela foi desenvolvida com a duração de várias *sprints*.

Nessa *sprint* também foi desenvolvida a tela de *dashboard* da aplicação, baseada no protótipo de alta fidelidade elaborado anteriormente (Figura 25), sendo que algumas mudanças no *layout*, além de como as variáveis foram tratadas para dar destaque à informação da precipitação, o que pode ser visto na Figura 26. Desta forma, as *sprints* US05 e US06 foram concluídas.

Figura 26 – Tela final do *Dashboard* da aplicação.



Fonte: Autoria Própria.

O *backlog* após a 2ª *sprint* está representado na Tabela 10.

Tabela 10 – *Backlog* da 2ª *sprint*.

História de Usuário	Backend Concluído?	Frontend Concluído?
US05 - Visualizar condições climáticas atuais	Sim	Sim
US06 - Visualizar previsões climáticas futuras	Não	Sim

Fonte: Autoria própria.

4.3.3 3ª a 6ª *Sprints*

Foi decidido realizar o registro das 3 *sprints* seguintes sob o mesmo tópico porque o foco foi integralmente voltado para o desenvolvimento do modelo de *machine learning*. Durante essas *sprints* diferentes granularidades e quantidades de dados foram testadas, com métodos univariados e multivariados. Foram testados e comparados modelos clássicos de previsão, de *machine learning* e *deep learning* sob diferentes métricas utilizadas em outros trabalhos científicos, tudo isso mantendo as reuniões com o orientador. Além disso, foi estabelecida a rotina que realiza a previsão de hora em hora e grava os resultados no banco de dados de forma persistente. A descrição detalhada deste desenvolvimento pode ser encontrada na Seção 4.1.

Portanto, nessas *sprints* foram concluídas as histórias US11 e TS01. O *backlog* até este momento está representado na Tabela 11.

Tabela 11 – *Backlog* após a 6ª *sprint*.

História de Usuário	**Backend Concluído?**	**Frontend Concluído?**
US11 - Inserir no banco de dados previsões realizadas	Sim	Não se aplica
TS01 - Criação do modelo de *machine learning*	Sim	Não se aplica

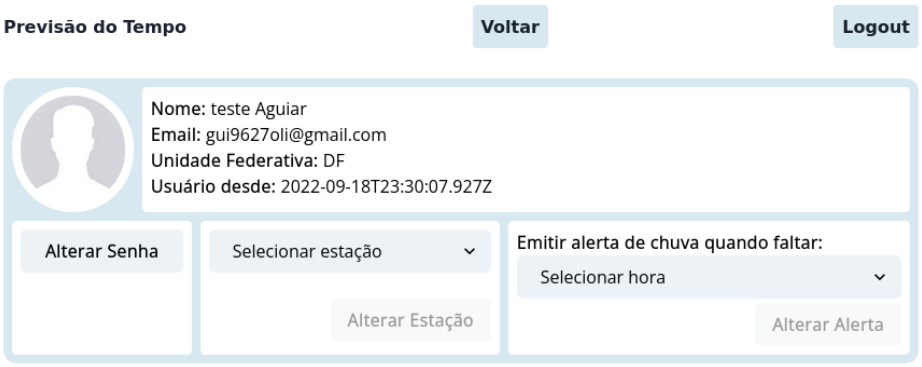
Fonte: Autoria própria.

4.3.4 7ª *Sprint*

Com a finalização do modelo de *machine learning*, o esforço foi alocado para as questões referentes às funcionalidades do *web site*, em relação ao login foram feitas as histórias de usuário US02 e US03, referentes a alteração e recuperação de senha. A alteração da senha é feita pela página virtual do perfil, como pode ser visto na Figura 27.

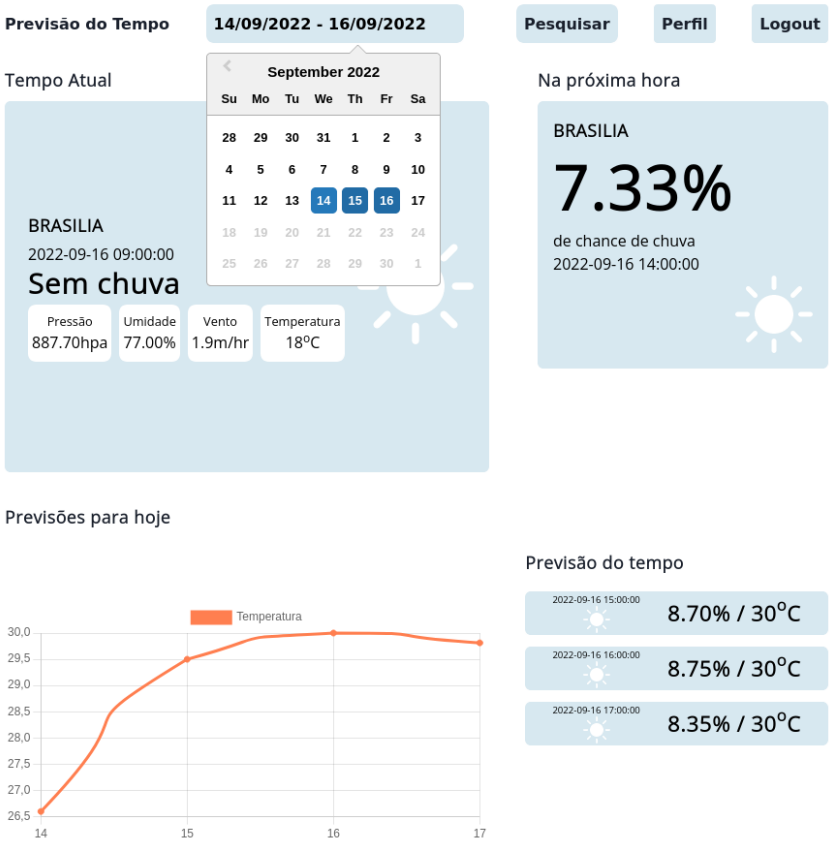
Também foi elaborada a história que trata da visualização do histórico das condições climáticas (US07). Na tela principal do *dashboard* é possível acessar um seletor, em que pode ser selecionado um intervalo de datas, como apresentado na Figura 28. Selecionando um intervalo, será só clicar no botão para pesquisar para ser possível visualizar os dados do período escolhido. E, essa visualização pode ser observada na Figura 29.

Figura 27 – Página da aplicação do perfil do usuário.



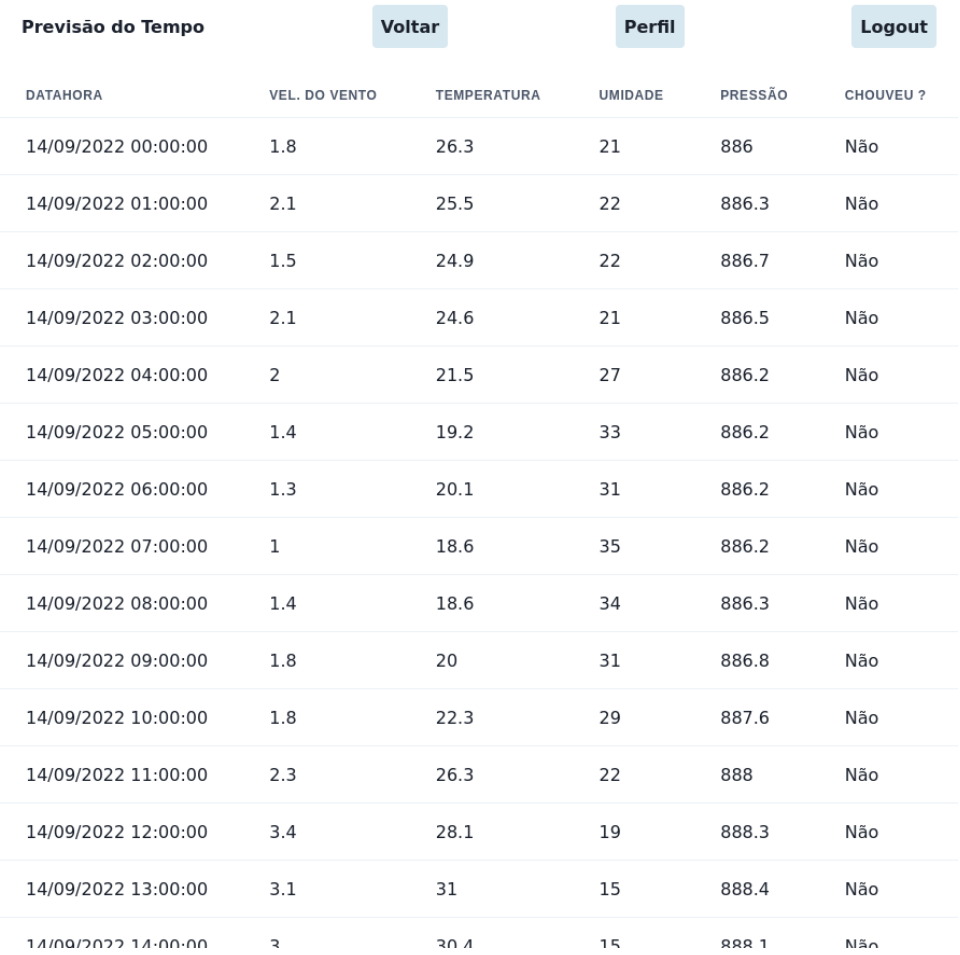
Fonte: Autoria Própria.

Figura 28 – Seleção do intervalo de datas no software desenvolvido.



Fonte: Autoria Própria.

Figura 29 – Tela com tabela mostrando o histórico meteorológico.



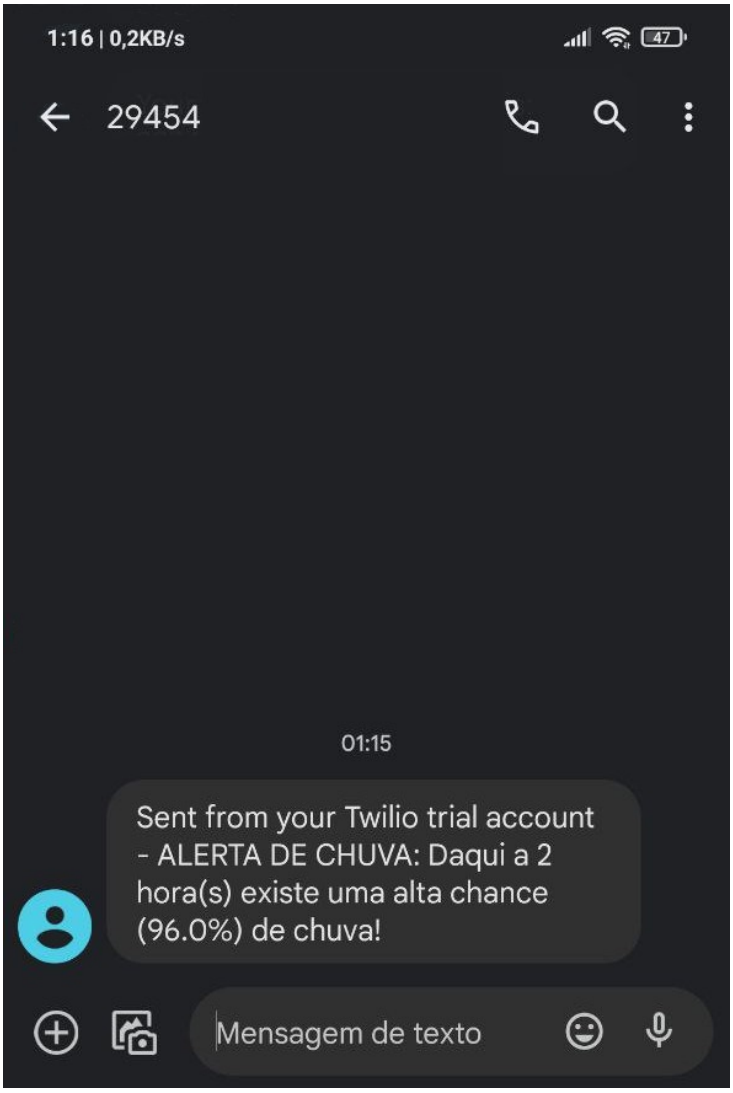
DATAHORA	VEL. DO VENTO	TEMPERATURA	UMIDADE	PRESSÃO	CHOUVEU ?
14/09/2022 00:00:00	1.8	26.3	21	886	Não
14/09/2022 01:00:00	2.1	25.5	22	886.3	Não
14/09/2022 02:00:00	1.5	24.9	22	886.7	Não
14/09/2022 03:00:00	2.1	24.6	21	886.5	Não
14/09/2022 04:00:00	2	21.5	27	886.2	Não
14/09/2022 05:00:00	1.4	19.2	33	886.2	Não
14/09/2022 06:00:00	1.3	20.1	31	886.2	Não
14/09/2022 07:00:00	1	18.6	35	886.2	Não
14/09/2022 08:00:00	1.4	18.6	34	886.3	Não
14/09/2022 09:00:00	1.8	20	31	886.8	Não
14/09/2022 10:00:00	1.8	22.3	29	887.6	Não
14/09/2022 11:00:00	2.3	26.3	22	888	Não
14/09/2022 12:00:00	3.4	28.1	19	888.3	Não
14/09/2022 13:00:00	3.1	31	15	888.4	Não
14/09/2022 14:00:00	3	30.4	15	888.1	Não

Fonte: Autoria Própria.

Já quanto a funcionalidade de alerta de chuva, foi realizada uma alteração, pois o alerta não foi feito baseando-se em informações de GPS, mas na estação que o usuário selecionar no perfil, dando assim mais autonomia para o usuário escolher a partir de qual estação ele quer receber alertas.

Assim como a escolha da estação, também é possível configurar o alerta na página de perfil de usuário, como pode ser observado na Figura 27. Essa configuração permite o usuário definir se ele quer receber a mensagem de alarme quando faltar uma, duas, três ou quatro horas para a possível chuva. O alerta é enviado por meio de uma mensagem SMS, assim como demonstra a Figura 30, que apresenta uma captura de tela da mensagem recebida no aparelho celular.

Figura 30 – Mensagem de alerta recebida em um aparelho celular via SMS.



Fonte: Autoria Própria.

Após a elaboração destas três histórias de usuário, ainda existem outras duas histórias no *backlog* de atividades. Uma delas trata da previsão de uma data no futuro escolhida pelo usuário, entretanto, durante a criação do modelo de ML percebeu-se que a previsão de dias no futuro, ou até em horas, resultaria em valores sem veracidade, ou seja, o objetivo da história é inviável com os dados meteorológicos utilizados. Sendo assim, optou-se por não prosseguir com o desenvolvimento dessa história de usuário.

Quanto a história de gestão de usuários, durante o desenvolvimento da aplicação foi percebido que não faria sentido existir um usuário administrador para cuidar das configurações do usuário padrão, primeiramente, porque não foram identificadas necessidades especiais para um usuário necessitar de um administrador, sendo que as configurações que existem, como a escolha da estação meteorológica e as notificações, por exemplo, podem ser feitas diretamente pelo próprio usuário padrão.

Portanto, nessas *sprints* foram concluídas as histórias US02, US03, US07 e US09.

O *backlog* até este momento do trabalho pode ser observado na Tabela 12.

Tabela 12 – *Backlog* da 7ª *sprint*.

História de Usuário	Backend Concluído?	Frontend Concluído?
US01 - Cadastro de usuário	Sim	Sim
US02 - Alteração de senha	Sim	Sim
US03 - Recuperação de senha	Sim	Sim
US04 - Realizar login	Sim	Sim
US05 - Visualizar condições climáticas atuais	Sim	Sim
US06 - Visualizar previsões climáticas futuras	Sim	Sim
US07 - Buscar condição de um dia específico no passado	Sim	Sim
US08 - Buscar previsão de um momento no futuro	Não	Não
US09 - Receber notificações de chuva baseado na estação atual	Sim	Sim
US10 - Inserir no banco de dados informações climáticas históricas	Sim	Não se aplica
US11 - Inserir no banco de dados previsões realizadas	Sim	Não se aplica
TS01 - Criação do modelo de *machine learning*	Sim	Não se aplica

Fonte: Autoria própria.

4.4 Testes de usabilidade

Uma vez acabado o desenvolvimento das funcionalidades da aplicação, foram executados testes de usabilidade com alguns possíveis usuários. São dois os objetivos dos testes que foram executados, o primeiro é avaliar se os usuários entendem sobre o que é o *site* (sítio virtual) e quais funcionalidades ele possui. O segundo é entender se a experiência de usuário está satisfatória, se a interface está fácil de entender e como os usuários avaliam a interface atual.

Para o teste foram selecionados 10 participantes de 19 à 60 anos, com os perfis em sintonia com aqueles levantados pelo método de personas, que foi apresentado na Seção 3.4.1 . Para os testes com os usuários foram separados dois cenários principais, o primeiro engloba o fluxo de visualização de dados históricos e o segundo analisa a configuração do alerta de chuvas. Ambos fluxos se iniciam na página da *dashboard* da aplicação.

A primeira parte dos testes é esperar que o usuário consiga realizar os cenários do fluxo de maneira adequada ou não. Para realizar os testes de usabilidade, foi utilizada a ferramenta *UsabilityHub*, em que é possível executar os testes de fluxo remotamente.

Após o fim da navegação, são feitas perguntas para o usuário, no qual será avaliado o quão compreensível é percorrer pelos cenários e como o usuário avalia as páginas virtuais apresentadas para ele. Essa avaliação é feita por meio de uma escala de 1 à 5, sendo 1 a nota mínima e 5 a nota máxima (melhor satisfação do usuário testador). As perguntas do questionário foram:

- **Q1 - Quão seguro você está que encontrou o lugar correto?**
- **Q2 - Quão difícil foi encontrar o lugar correto?**
- **Q3 - Como você descreveria sua experiência geral com o a aplicação?**
- **Q4 - Como é a linguagem utilizada na aplicação?**

A Tabela 13 e a Tabela 14 apresentam os resultados dos testes de usabilidade feitos com os usuários, nos quais para cada questão, foi atribuída a porcentagem de cada opinião fornecida, sendo que os 10 usuários participaram dos testes.

Tabela 13 – Resultado do teste do primeiro fluxo.

	1	2	3	4	5
Q1	0%	0%	0%	0%	100%
Q2	0%	0%	0%	20%	80%
Q3	0%	0%	20%	40%	40%
Q4	0%	0%	10%	40%	50%

Fonte: Autoria própria.

É possível verificar na Tabela 13 os resultados do teste do fluxo de pesquisa de dados históricos, no qual respostas obtidas foram predominantemente positivas, mantendo-se entre as notas 3 e 5. Em relação às notas 3 no Q3 e Q4, percebeu-se que algumas das pessoas não conseguiram associar imediatamente a barra de pesquisa com o objetivo de buscar por dados históricos, para remediar essa situação e deixar a experiência de usuário mais clara, foi feita um alteração que é apresentado na barra de pesquisa antes que o usuário clique nela, de "Selecionar Data"para "Selecionar dias anteriores", também foi alterado para uma fonte e cor mais visível.

Já na tabela que apresenta os dados históricos foi colorido o plano de fundo da cor de cada linha por dia, assim 24 linhas são brancas e as próximas 24 são cinzas, intercalando até o fim do intervalo selecionado. Essa mudança faz a leitura da tabela ficar mais compreensível, principalmente quando é selecionado um intervalo de muitos dias.

Tabela 14 – Resultado do teste do segundo fluxo.

	1	2	3	4	5
Q1	0%	0%	0%	20%	80%
Q2	0%	0%	20%	40%	40%
Q3	0%	0%	10%	50%	40%
Q4	0%	0%	10%	20%	70%

Fonte: Autoria própria.

A Tabela 14 apresenta o teste do fluxo de criação de um alarme. Percebeu-se algumas dificuldades para encontrar o local de definição do alarme, como mostra as notas 3 na questão Q2, Q3 e Q4. A página de perfil é acessada através do botão de "Perfil"na página inicial (*dashboard*). Para que componente de alteração de alarme ficasse mais compreensível, foi alterado o texto do botão, de "Alterar Alarme"para "Confirmar", seguindo a sugestão de um usuário.

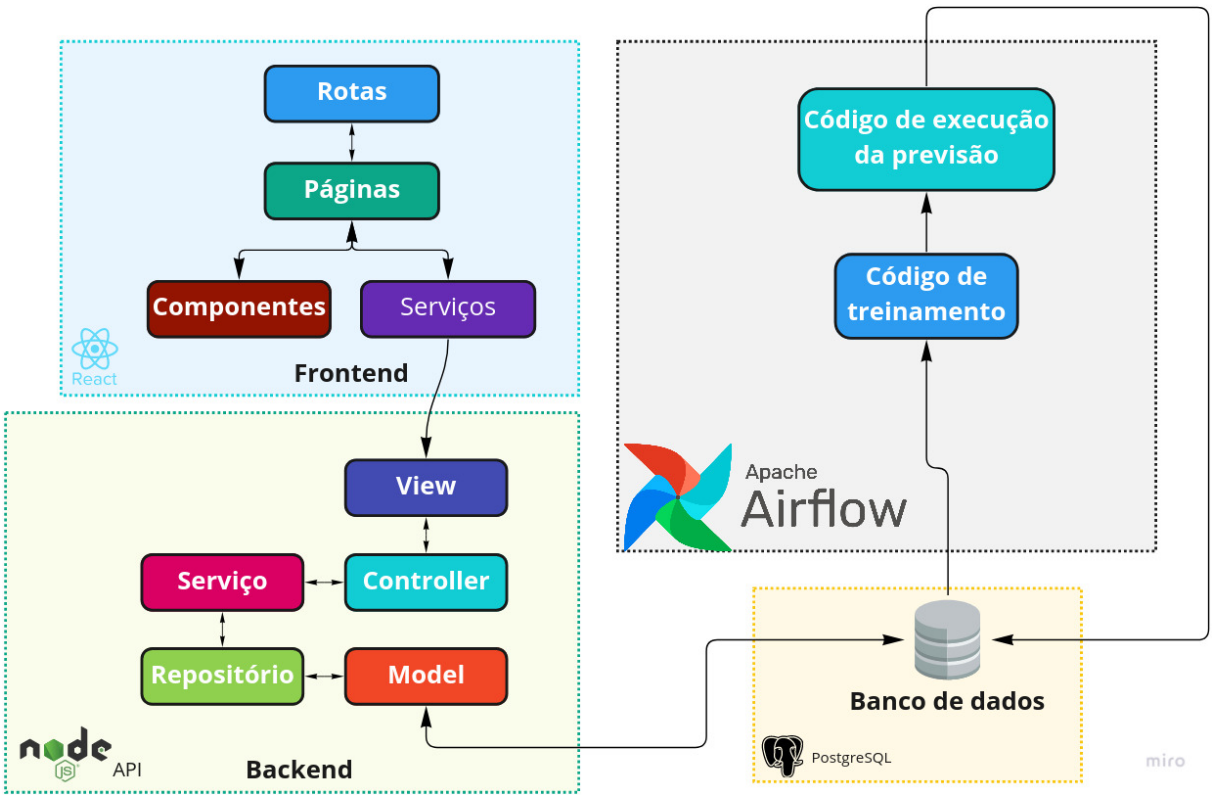
Com base no *feedback* obtido, verificou-se que os usuários conseguiram navegar e concluir os fluxos escolhidos e as funcionalidades foram compreendidas. Tendo em vista as respostas das questões, a experiência do usuário majoritariamente positiva.

4.5 Arquitetura

A representação arquitetural implementada para este trabalho, utilizando os padrões expostos anteriormente pode ser vista na Figura 31.

É observado na Figura 31 que a arquitetura da aplicação é dividida em duas partes, o *Frontend* e o *Backend*. O *Backend* será dividido em três partes. A primeira diz respeito a API (Interface de Programação de Aplicação, do inglês *Application Programming Interface*) feita no *framework* Node.JS e nela será aplicada os padrões discutidos anteriormente nesta seção. Além de ser um *framework* apropriado para a implementação dos padrões de projeto, o Node.JS também conta com um conjunto de bibliotecas que facilitam a implementação de uma API, como interação com os dados de forma dinâmica por meio de ORM (Mapeamento Objeto-Relacional, do inglês *Object-Relational Mapping*) e segurança nos *endpoints*.

Figura 31 – Representação arquitetural da aplicação.



Fonte: Autoria Própria.

Para o treinamento e armazenamento do modelo de *Machine Learning* e execução das predições, foi escolhido o *Flask*, que é um *framework* leve e flexível para Python. O Sistema Gerenciador de Banco de Dados (SGBD) escolhido foi o PostgreSQL, por ser relacional, aberto e amplamente utilizado.

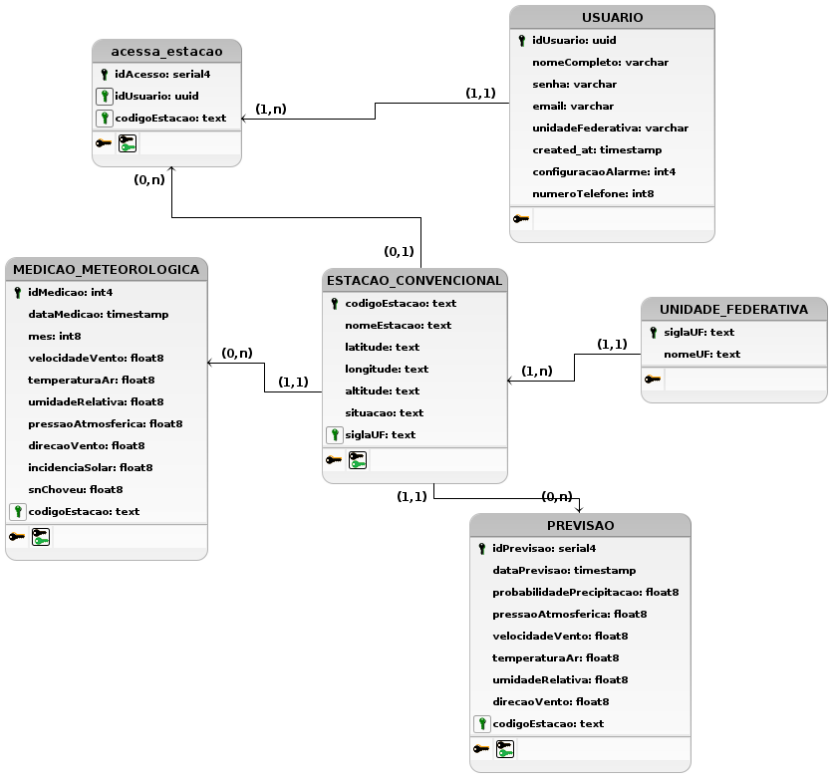
Já no *Frontend*, a estrutura a ser utilizada será *Components*, *Pages*, *Routes* e *Services*, utilizando a biblioteca JavaScript React, em que os *components* serão elementos individuais que serão reutilizados em mais de uma página, enquanto *page* será uma tela da aplicação, *route* será responsável por toda a configuração de rotas da aplicação e *service* toda a configuração da conexão e métodos da API.

Quanto a interação das partes, a API em Node.JS irá interagir com o banco de dados da aplicação consumindo e registrando informações do banco de dados (SGBD), como informações cadastrais de usuários e outras preferências de usuários. O banco de dados também receberá periodicamente informações históricas temporais e de previsão, que serão abastecidos tanto por um *script* de raspagem e um modelo de dados administrado pelo Apache Airflow. O *Frontend* por sua vez terá acesso às informações por meio de *endpoints* na API Node.JS.

4.6 Banco de dados

Quanto ao projeto da base de dados, foram feitas algumas alterações em relação ao apresentado na Seção 3.6 como proposta. A primeira alteração corresponde as tabelas provenientes de relacionamentos com cardinalidade (n, m) que associavam os usuários aos registros meteorológicos e de previsão, ao qual foi alterada para apenas uma, intitulada acessa *estacao*, que se relaciona com a tabela *USUARIO* e *ESTACAO CONVENCIONAL*, como pode ser visto na Figura 32.

Figura 32 – Diagrama Lógico de Dados (DLD) atualizado.



Fonte: Autoria Própria.

Houve ainda a retirada de algumas colunas refletindo a análise correlacional feita na Seção 4.1.1 e o atributo *precipitacaoTotal* teve o nome alterado para *choveuHoje*. Finalmente, a tabela e os atributos relacionados ao sistema de notificação foram removidos e a discussão sobre a história que trata das notificações está na Seção 4.3.4, que aborda a sétima *sprint*.

5 Considerações finais

5.1 Conclusão

O objetivo deste trabalho foi o desenvolvimento de uma aplicação de consulta, previsão meteorológica e alerta de chuvas. Por meio da aplicação elaborada será possível que um indivíduo tenha uma percepção mais segura sobre a situação climática (ou do tempo) cotidiana de um local ou região em que ele viva ou esteja, a fim de contribuir com as decisões que poderá vir a tomar considerando a possibilidade de chuva. Para a execução desta predição, conforme explorado no Capítulo 3, foi criado um modelo de *Machine Learning* com as práticas modernas, tendo como fonte uma base de dados meteorológica coerente as demandas abordadas no discorrer deste trabalho.

A pesquisa bibliográfica realizada até o momento proporcionou o entendimento contextual do trabalho, enquanto a pesquisa exploratória foi responsável pela estruturação de suas necessidades. Mediante a definição da metodologia de pesquisa do projeto foi possível delimitar o escopo e o processo metodológico coerente, que foi utilizado para o desenvolvimento e o gerenciamento das atividades realizadas até a conclusão do projeto proposto. Estas metodologias permitiram adaptações às mudanças do trabalho em sua segunda etapa e contribuíram para manter o foco na elaboração de um software (aplicação) funcional.

Após a fase de desenvolvimento (Capítulo 4) foi possível verificar, por meio da literatura de apoio, sendo as pesquisas continuamente efetuadas pelos autores durante todo o período de vigência do projeto, que o modelo de *Machine Learning* conseguiu performar em nível de acerto similar a alguns outros trabalhos que fazem uso de dados históricos numéricos, isto é, sem imagens de radar ou simulações atmosféricas.

Também foi elaborado o *Backend* da aplicação, que tem como principal objetivo atuar como interface entre os dados e o *Frontend*, sendo ainda responsável pela lógica de autenticação e outras configurações de usuário como a alteração de estação de obtenção dos dados para análise. No *Frontend* construído o usuário pode visualizar as condições climáticas por meio da API elaborada no *Backend*, assim como visualizar os dados históricos da estação. Para as rotinas de atualização do banco de dados, a integração, transformação e carregamento dos dados analisados foram feitas em Python e rodam a partir da plataforma Apache Airflow.

Por meio da execução do teste de usabilidade feito com possíveis usuários, foi possível validar que o que a aplicação faz e os principais fluxos de funcionalidade da aplicação estão compreensíveis. Também serviu para verificar que a interface e a experiência

de usuário foram majoritariamente positivas.

Portanto, foi constatado que os objetivos indicados na Seção 1.4 (Objetivos) foram atingidos no final de todo o processo, como a construção do sistema empregando *Machine Learning*, estruturação de uma base de dados atmosférica mais coerente aos objetivos almejados e a aplicação que proporciona ao usuário uma percepção sobre a situação climática atual e nas próximas horas.

5.2 Trabalhos Futuros

O presente projeto elaborou uma aplicação de previsão meteorológica e alertas de chuva para proporcionar o apoio na tomada de decisão do usuário, ajudando-o assim a remediar situações de dificuldade relacionadas ao clima.

A proposta inicial precisou de mudanças até alcançar seu êxito relacionado ao objetivo principal, sendo necessária uma busca exaustiva por técnicas existentes que pudessem realizar as previsões tendo como entrada apenas os dados numéricos. Da mesma forma as funcionalidades e interface da aplicação foram sendo refatoradas para oferecer cada vez mais ao usuário a experiência mais simples e benéfica. No entanto, pode-se afirmar que o presente trabalho necessita de continuidade evolutiva, e possivelmente corretiva, diante dos desafios que a previsão de chuva, apoiada por tecnologias, vem trilhando com o objetivo de tornar o processo mais eficiente e de melhor qualidade.

- **Estudar a aquisição de novas entradas mais relevantes para o modelo**: Durante o desenvolvimento do modelo de ML percebeu-se que os dados disponíveis na base do INMET são limitados para a tarefa de previsão de chuvas. Apesar de que algumas vezes as chuvas podem ser previstas observando a interação entre as variáveis utilizadas, como pressão atmosférica, temperatura e umidade, as interações atmosféricas acontecem em níveis moleculares e tem grande complexidade. Foi verificado em alguns trabalhos a quantificação da água guardada em uma seção transversal da atmosfera, também chamada de vapor de água precipitável, sendo usada como entrada para um modelo de ML.
- **Analisar a utilização do modelo de *Machine Learning* aliado à outras técnicas**: Esse ponto consiste na utilização de outros métodos de inteligência artificial que poderiam ser utilizados juntos do modelo de ML. Por exemplo, é possível atrelar um sistema baseado em conhecimento para agregar mais exatidão ao resultado do modelo. Também seria valido utilizar uma técnica alternativa, como de NWP e realizar a média dos resultados.

Outras opções ainda podem proporcionar a evolução desse sistema com a expec-

tativa de vir a contribuir com a vida e organização daqueles que possam utilizá-lo como uma ferramenta de apoio à decisão.

Referências

AGGARWAL, C. *Neural Networks and Deep Learning: A Textbook*. [S.l.]: Springer, 2019. Citado 2 vezes nas páginas 37 e 39.

AHRENS, C. D. *Essenstials of Meteorology: An Invitation to the Atmosphere*. California: Brooks/Cole, 2001. ISBN 978-0534572044. Citado 2 vezes nas páginas 29 e 31.

ALéSSIO, S. C.; SABADIN, N. M.; ZANCHETT, P. S. *Processos de Software*. 9th. ed. Indaial: Centro Universitário Leonardo da Vinci, 2017. ISBN 978-85-515-0046-0. Citado na página 53.

ALJAMEA, M.; ALKANDARI, M. Mmvmi: A validation model for mvc and mvvm design patterns in ios applications. *IAENG Int. J. Comput. Sci*, v. 45, n. 3, p. 377–389, 2018. Citado na página 62.

ARMSTRONG, M. How Many Websites Are There? 2021. Disponível em: <https://www.statista.com/chart/19058/number-of-websites-online/>. Acesso em: 12 de Março de 2022. Citado na página 44.

ASHMORE, R.; CALINESCU, R.; PATERSON, C. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 54, n. 5, p. 1–39, 2021. Citado 2 vezes nas páginas 52 e 53.

BARRERA-ANIMAS, A. Y. et al. Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting. *Machine Learning with Applications*, v. 7, p. 100204, 2022. ISSN 2666-8270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S266682702100102X>. Citado 2 vezes nas páginas 31 e 32.

BECK, K. et al. *Manifesto para Desenvolvimento Ágil de* Sofware. 2001. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 19 de Março de 2022. Citado na página 54.

BJERKNES, V. The problem of weather prediction, considered from the viewpoints of mechanics and physics. *Meteorologische Zeitschrift*, Schweizerbart Science Publishers, Stuttgart, Germany, v. 18, n. 6, p. 663–667, 12 2009. Disponível em: <http://dx.doi.org/10.1127/0941-2948/2009/416>. Citado na página 23.

BRMODELO. *Projeto brModelo 3.0 (atual v3.31)*. 2022. Disponível em: <https://www.sis4.com/brModelo/>. Acesso em: 03 de Abril de 2022. Citado na página 66.

COHEN, D.; LINDVALL, M.; COSTA, P. *Agile Software Development*. Fraunhofer: DACS State-of-the-Art/Practice Report, 2012. Citado na página 54.

COHN, M. *User Stories Applied for Agile Software Development*. Mexico City: Addison-Wesley, 2009. ISBN 0-321-20568-5. Citado na página 59.

Comitê Gestor da Internet no Brasil. *PAINEL TIC: COVID-19 - Pesquisa web sobre o uso da Internet no Brasil durante a pandemia do novo coronavírus*. São Paulo: Núcleo

de Informação e Coordenação do Ponto BR, 2021. ISBN 978-65-86949-33-9. Citado na página 44.

DIAGRAMS.NET. *About diagrams.net*. 2022. Disponível em: <https://www.diagrams.net/about>. Acesso em: 20 de Março de 2022. Citado na página 65.

EKMAN, M. *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Languagem Processing and Transformers Using Tensorflow*. [S.l.]: Nvdia Deep Learning Institute, 2021. Citado 3 vezes nas páginas 40, 41 e 42.

ELOYR, A.; NUNES, A. *Support Vector Machines*. 2020. Disponível em: <https://lamfo-unb.github.io/2020/07/04/SVM/>. Acesso em: 21 de Agosto de 2022. Citado 2 vezes nas páginas 34 e 35.

EVANS, E. *Domain-Driven Design - Tackling Complexity in the Heart of Software*. Addison Wesley, 2003. ISBN 0321125215; 9780321125217. Disponível em: <libgen.li/file.php?md5=11edcb29974052333fcf77d15eb084c5>. Citado na página 62.

FONSECA, J. da. *Apostila de metodologia da pesquisa científica*. João José Saraiva da Fonseca, 2002. Disponível em: <https://books.google.com.br/books?id=oB5x2SChpSEC>. Citado na página 47.

FRADKOV, A. L. Early history of machine learning. *IFAC-PapersOnLine*, v. 53, n. 2, p. 1385–1390, 2020. ISSN 2405-8963. 21st IFAC World Congress. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405896320325027>. Citado na página 22.

FRASCISO, H. et al. Que fonte de dados meteorolÓgicos utilizar no brasil? que incerteza esperar? uma comparaÇÃo entre diferentes abordagens e variadas fontes de dados. *VIII Congresso Brasileiro de Energia Solar*, v. 41, 06 2020. Citado 2 vezes nas páginas 45 e 46.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189 – 1232, 2001. Disponível em: <https://doi.org/10.1214/aos/1013203451>. Citado na página 36.

FRISINGER, H. H. Aristotle and his “meteorologica”. *Bulletin of the American Meteorological Society*, American Meteorological Society, Boston MA, USA, v. 53, n. 7, p. 634 – 638, 1972. Disponível em: <https://journals.ametsoc.org/view/journals/bams/53/7/1520-0477_1972_053_0634_aah_2_0_co_2.xml>. Citado na página 21.

GALLIZZI, B. *The countries most affected by global warming, based on natural disasters*. 2022. Disponível em: <https://www.uswitch.com/gas-electricity/global-warming-and-natural-disasters/>. Acesso em: 14 de Março de 2022. Citado na página 23.

GARCIA, R. et al. Context: The missing piece in the machine learning lifecycle. In: *KDD CMI Workshop*. [S.l.: s.n.], 2018. v. 114. Citado na página 52.

GERHARDT, T.; SILVEIRA, D. Métodos de pesquisa. 2009. Citado 2 vezes nas páginas 26 e 47.

GIT. *About*. 2022. Disponível em: <https://git-scm.com/about>. Acesso em: 20 de Março de 2022. Citado na página 65.

GITHUB. *Where the world builds software*. 2022. Disponível em: <https://github.com/about>. Acesso em: 20 de Março de 2022. Citado na página 65.

GOODFELLOW, I.; BENGIO, Y.; COURVILLER, A. *Deep Learning Book*. [S.l.]: O'reilly, 2016. Citado na página 33.

GOUVEA, C. et al. A utilização do scrum como recurso educacional no processo de aprendizagem em engenharia de software. v. 3, 12 2016. Citado 3 vezes nas páginas 56, 57 e 58.

GRAHAM, S.; PARKINSON, C.; CHAHINE, M. *Weather Forecasting Through the Ages*. 2002. Disponível em: <https://earthobservatory.nasa.gov/features/WxForecasting/wx2.php>. Acesso em: 12 de Janeiro de 2022. Citado 2 vezes nas páginas 21 e 22.

HABY, J. *Reasons for a weather forecast*. 2015. Disponível em: <https://www.theweatherprediction.com/habyhints3/985/>. Acesso em: 18 de Janeiro de 2022. Citado na página 25.

HAKIM, G.; PATOUX, J. *Weather: A Concise Introduction*. Cambridge University Press, 2017. ISBN 9781108404655. Disponível em: <https://books.google.com.br/books?id=pqXoAQAACAAJ>. Citado 5 vezes nas páginas 21, 23, 29, 30 e 31.

HEITKöTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. *International Conference on Web Information Systems and Technologies*, Department of Information Systems, University of Munster, Munster, Germany, v. 54, n. 5, p. 120–128, 2012. Citado na página 45.

HIDROWEB. *HIDROWEB: Rede Hidrometeorológica Nacional*. 2005. Disponível em: <https://www.snirh.gov.br/hidroweb/apresentacao>. Acesso em: 07 de Março de 2022. Citado na página 45.

HOLMSTROM, M.; LIU, D.; VO, C. Machine learning applied to weather forecasting. *Meteorol. Appl*, p. 1–5, 2016. Citado na página 24.

HOSSAIN, I. et al. Long-term seasonal rainfall forecasting using linear and non-linear modelling approaches: a case study for western australia. *Meteorology and Atmospheric Physics*, v. 132, 02 2020. Citado na página 31.

HOWARD, J.; GUGGER, S. *Deep Learning for Coders with fastai PyTorch AI Applications Without a PhD*. [S.l.]: O'reilly, 2020. Citado 5 vezes nas páginas 32, 33, 38, 42 e 43.

HU, C. et al. Deep learning with a long short-term memory networks approach for rainfall-runoff simulation. *Water*, v. 10, n. 11, 2018. ISSN 2073-4441. Disponível em: <https://www.mdpi.com/2073-4441/10/11/1543>. Citado na página 31.

HUI, Y. et al. Gps-derived pwv for rainfall nowcasting in tropical region. *IEEE Transactions on Geoscience and Remote Sensing*, v. 56, p. 4835–4844, 08 2018. Citado 2 vezes nas páginas 73 e 75.

IBM. *Native, web or hybrid mobile-app development*. 2012. Disponível em: <ftp://public.dhe.ibm.com/software/fr/pdf/whitepapers/Worklight-HTML5_Hybrid_Native_Mobile_App_Development.pdf>. Acesso em: 12 de Março de 2022. Citado na página 44.

INNES, P.; BEASLEY, W. *Operational Weather Forecasting. Advancing Weather and Climate Science*. Nova Jersey: Wiley-Blackwell, 2013. ISBN 978-0-470-71159-0. Citado na página 30.

INNES, P.; DORLING, S. *Operational Weather Forecasting*. Nova Jersey: Wiley-Blackwell, 2013. ISBN 978-0-470-71159-0. Citado na página 30.

INNESS, P.; DORLING, S. *Operational Weather Forecasting*. [S.l.: s.n.], 2012. ISBN 9780470711590. Citado 2 vezes nas páginas 22 e 23.

JAKKULA, V. R. Tutorial on support vector machine (svm). In: . [S.l.: s.n.], 2011. Citado na página 35.

JUNIOR, M. L.; FILHO, M. G. Variations of the kanban system: Literature review and classification. *Int. J. Production Economics*, v. 125, p. 13–21, 01 2010. Citado na página 54.

KNIBERG, H.; SKARIN, M. *Kanban and Scrum: Making the most of both pdf*. Toronto: InfoQueue, 2010. ISBN 978-0-557-13832-6. Citado na página 54.

KUHN, M.; JOHNSON, K. *Applied predictive modeling*. Springer, 2013. ISBN 9781461468493 1461468493 1461468485 9781461468486. Disponível em: <http://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485/>. Citado na página 36.

LARY, D. J. et al. Machine learning in geosciences and remote sensing. *Geoscience Frontiers*, v. 7, n. 1, p. 3–10, 2016. ISSN 1674-9871. Special Issue: Progress of Machine Learning in Geosciences. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1674987115000821>. Citado na página 74.

LEHMKUHL, D.; EGER, D. R. *PRINCÍPIOS DE BANCO DE DADOS*. Indaial: UNIASSELVI, 2013. Disponível em: <https://livrodigital.uniasselvi.com.br/GTI11_principios_de_banco_de_dados/>. Acesso em: 28 de Março de 2022. Citado na página 63.

LING, T. W. A normal form for entity-relationship diagrams. *ER*, v. 1985, p. 24–35, 1985. Citado na página 62.

LIU, Y. *Python Machine Learning By Example*. [S.l.]: Packt, 2019. Citado na página 34.

Made In Web. *O que é Web App?* 2020. Disponível em: <https://www.madeinweb.com.br/o-que-e-web-app/>. Acesso em: 12 de Março de 2022. Citado na página 43.

MANANDHAR, S. et al. A data-driven approach for accurate rainfall prediction. *IEEE Transactions on Geoscience and Remote Sensing*, PP, p. 1–9, 08 2019. Citado na página 73.

MARTIN, R. *Arquitetura Limpa: O guia do artesão para estrutura e design de software*. Alta Books, 2019. (Robert C. Martin). ISBN 9788550808161. Disponível em: <https://books.google.com.br/books?id=BOaPDwAAQBAJ>. Citado 2 vezes nas páginas 61 e 62.

MINUZZI, T. d. S. *Ustory-Refactory: ferramenta de refatoração de requisitos aplicada em cartões user stories (CRC Cards)*. São Leopoldo, RS: Universidade do Vale do Rio dos Sinos, 2007. Disponível em: <http://www.repositorio.jesuita.org.br/handle/UNISINOS/2242>. Acesso em: 28 de Março de 2022. Citado na página 60.

MOREIRA, J. C.; SENE, E. d. *Geografia Geral e Do Brasil*. São Paulo: Scipione, 2016. ISBN 978-85-262-9913-9. Citado 3 vezes nas páginas 29, 30 e 31.

MURPHY, K. P. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN 9780262018029 0262018020. Disponível em: <https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2>. Citado 2 vezes nas páginas 32 e 34.

NODE.JS. *About Node.js*. 2022. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 05 de Abril de 2022. Citado na página 66.

PERCIVAL, H.; GREGORY, B. *Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices*. O'Reilly Media, 2020. ISBN 9781492052173. Disponível em: <https://books.google.com.br/books?id=P5DUDwAAQBAJ>. Citado na página 62.

POSTGRESQL. *About PostgreSQL*. 2022. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 05 de Abril de 2022. Citado na página 66.

PYTHON. *What is Python? Executive Summary*. 2022. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em: 05 de Abril de 2022. Citado na página 66.

REACT. *React*. 2022. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 04 de Abril de 2022. Citado na página 65.

REHKOPF, M. *Kanban vs. Scrum: que tipo de ágil é você?* 2020. Disponível em: <https://www.atlassian.com/br/agile/kanban/kanban-vs-scrum>. Acesso em: 20 de Março de 2022. Citado na página 54.

RIGUTTI, A. *Meteorologia*. Milão: Giunti, 2002. ISBN 88-09-02823-6. Citado na página 29.

ROSEBROCK, A. *Deep Learning for computer vision with python*. [S.l.]: PYIMAGESEARCH, 2017. Citado 4 vezes nas páginas 22, 37, 39 e 40.

SAJJA, P. S.; AKERKAR, R. Knowledge-based systems for development. *Advanced Knowledge Based Systems: Model, Applications & Research*, Jones & Bartlett Publishers, v. 1, p. 1–11, 2010. Citado na página 72.

SAMAD, A. et al. An approach for rainfall prediction using long short term memory neural network. In: *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*. [S.l.: s.n.], 2020. p. 190–195. Citado 2 vezes nas páginas 41 e 42.

SCHAPIRE, R. E. The strength of weak learnability. *Machine Learning*, v. 5, n. 2, p. 197–227, 1990. ISSN 0885-6125. Disponível em: <http://www.cs.princeton.edu/~schapire/papers/strengthofweak.pdf>. Citado na página 36.

SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide*. [S.l.]: Scrum.org, 2010. Citado 3 vezes nas páginas 56, 57 e 58.

SHAO, H. Delay-dependent stability for recurrent neural networks with time-varying delays. *IEEE Transactions on Neural Networks*, v. 19, n. 9, p. 1647–1651, 2008. Citado na página 41.

SHEPPARD, D. *Beginning Progressive Web App Development*. first. Illinois: APRESS, 2017. ISBN 978-1-4842-3089-3. Citado na página 44.

SHILPA et al. A data-driven approach to detect precipitation from meteorological sensor data. In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. [S.l.: s.n.], 2018. p. 3872–3875. Citado 2 vezes nas páginas 73 e 75.

SILVA, D.; SANTOS, F.; NETO, P. S. Os benefícios do uso de kanban na gerência de projetos de manutenção de software. In: *Anais do VIII Simpósio Brasileiro de Sistemas de Informação*. Porto Alegre, RS, Brasil: SBC, 2012. p. 715–725. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/14454>. Citado na página 54.

SINTEF. *Big Data, for better or worse: 90% of world's data generated over last two years*. 2013. Disponível em: <www.sciencedaily.com/releases/2013/05/130522085217.htm>. Acesso em: 02 de Fevereiro de 2022. Citado na página 32.

SOMMERVILLE, I. *Engenharia de Software*. 9th. ed. São Paulo: Pearson, 2011. ISBN 978-85-7936-108-1. Citado 4 vezes nas páginas 44, 52, 53 e 59.

SPíNOLA, R. O.; ARAúJO, M. A. P. *Qualidade de Software*. Rio de Janeiro: DevMedia, 2007. Citado na página 44.

TANDEL, S. S.; JAMADAR, A. Impact of progressive web apps on web app development. *International Journal of Innovative Research in Science, Engineering and Technology*, v. 7, p. 9439–9444, 09 2018. ISSN 2347-6710. Citado na página 44.

TEAGUE, K. A.; GALLICCHIO, N. *The Evolution of Meteorology: A Look into the Past, Present, and Future of Weather Forecasting*. [S.l.: s.n.], 2017. Citado na página 21.

TRELLO. *About Trello*. 2022. Disponível em: <https://trello.com/about>. Acesso em: 20 de Março de 2022. Citado na página 65.

WANG, B. et al. Deep uncertainty learning: A machine learning approach for weather forecasting. *CoRR*, abs/1812.09467, 2018. Disponível em: <http://arxiv.org/abs/1812.09467>. Citado na página 24.

WATANABE, P. *Cerca de 116 milhões de brasileiros foram afetados por desastres naturais desde 1902*. 2019. Disponível em: <https://www1.folha.uol.com.br/ambiente/2022/02/cerca-116-milhoes-de-brasileiros-foram-afetados-por-desastres-naturais-desde-1902.shtml>. Acesso em: 14 de Março de 2022. Citado 2 vezes nas páginas 22 e 23.

WMO. *Desastres naturais foram responsáveis por 45% de todas as mortes nos últimos 50 anos, mostra OMM*. 2019. Disponível em: <shorturl.at/auAC2>. Acesso em: 18 de Janeiro de 2022. Citado na página 25.

YNOUE, R. Y. et al. *Meteorologia: Noções Básicas*. São Paulo: Oficina de Textos, 2017. ISBN 978-85-7975-263-6. Citado 3 vezes nas páginas 29, 30 e 31.

ZACHARIAS, I. S.; CUNHA, L.; COSTA, J. User stories: Quem, quando e como deve ser usado? In: . [S.l.: s.n.], 2017. p. 810–818. Citado na página 60.

ZUCATTO, L. C.; FREITAS, R. U. C. d.; MARZZONI, D. N. S. Basic research and applied research: an analysis from scientific production on covid-19. *Research, Society and Development*, v. 9, n. 11, p. e63791110179, Nov. 2020. Disponível em: <https://www.rsdjournal.org/index.php/rsd/article/view/10179>. Citado na página 47.

Apêndices

APÊNDICE A – Base de Dados

Este anexo tem o objetivo de apresentar os *scripts* desenvolvidos para a proposta de solução deste projeto.

A.1 *Script* de criação da base de dados

A Listagem A.1 apresenta o *script* físico em linguagem SQL desenvolvido de acordo com a implementação do Diagrama Lógico de Dados apresentado na Seção 3.6.2.

```
1  -- --------------      Aplicacao TCC-1      ----------------
2  --
3  --                    SCRIPT DE CRIACAO (DDL)
4  --
5  -- Data Criacao .......: 21/04/2022
6  -- Autores ............: Guilherme de Oliveira Aguiar e Victor Rodrigues
       Silva
7  -- Banco de Dados .....: PostgresSQL 11
8  -- Base de Dados (nome): tempesta
9  --
10 --
11 -- PROJETO => 1 Base de Dados
12 --         => 7 Tabelas
13 --
14 -- -----------------------------------------------------------
15
16 -- BASE DE DADOS
17
18 CREATE DATABASE IF NOT EXISTS tempesta;
19
20 USE tempesta;
21
22 -- TABELAS
23
24 CREATE TABLE USUARIO (
25     idUsuario SERIAL NOT NULL,
26     nomeCompleto VARCHAR(100) NOT NULL,
27     email VARCHAR(30) NOT NULL,
28     senha VARCHAR(100) NOT NULL,
29     unidadeFederativa ENUM('AL', 'AP', 'AM', 'BA', 'CE', 'DF', 'ES', 'GO
       ', 'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR', 'PE', 'PI', 'RJ', 'RN',
       'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO') NOT NULL,
30     configuracaoAlarme VARCHAR(50) NOT NULL,
```

```
31      CONSTRAINT USUARIO_PK PRIMARY KEY (idUsuario)
32  );
33
34  CREATE TABLE NOTIFICACAO (
35      idNotificacao SERIAL NOT NULL,
36      titulo VARCHAR(30) NOT NULL,
37      dataNotificacao DATE NOT NULL,
38      horaNotificacao VARCHAR(100) NOT NULL,
39      idUsuario INT NOT NULL,
40      CONSTRAINT NOTIFICACAO_PK PRIMARY KEY (idNotificacao),
41      CONSTRAINT NOTIFICACAO_USUARIO_FK FOREIGN KEY (idUsuario) REFERENCES
        USUARIO(idUsuario)
42          ON UPDATE CASCADE
43          ON DELETE CASCADE
44  );
45
46  CREATE TABLE ESTACAO (
47      codigoEstacao INT,
48      latitute FLOAT,
49      longitude FLOAT,
50      altitude FLOAT,
51      nomeEstacao VARCHAR(50),
52      situacao ENUM('ATIVA', 'INATIVA'),
53      dataInstalacao DATE,
54      unidadeFederativa ENUM('AL', 'AP', 'AM', 'BA', 'CE', 'DF', 'ES', 'GO
        ', 'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR', 'PE', 'PI', 'RJ', 'RN',
        'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO') NOT NULL,
55      CONSTRAINT ESTACAO_PK PRIMARY KEY (codigoEstacao)
56  )
57
58  CREATE TABLE MEDICAO_METEOROLOGICA (
59      idMedicao SERIAL NOT NULL,
60      dataMedicao DATE NOT NULL,
61      horaMedicao VARCHAR(4) NOT NULL,
62      precipitacaoTotal FLOAT NOT NULL,
63      temperaturaMaxima FLOAT NOT NULL,
64      temperaturaMinima FLOAT NOT NULL,
65      temperaturaAr FLOAT NOT NULL,
66      temperaturaPontoOrvalho FLOAT NOT NULL,
67      umidadeRelativa FLOAT NOT NULL,
68      pressaoAtmosferica FLOAT NOT NULL,
69      velocidadeVento FLOAT NOT NULL,
70      direcaoVento FLOAT NOT NULL,
71      codigoEstacao INT NOT NULL,
72      CONSTRAINT MEDICAO_METEOROLOGICA_PK PRIMARY KEY (idMedicao),
73      CONSTRAINT MEDICAO_ESTACAO_FK FOREIGN KEY (codigoEstacao) REFERENCES
        ESTACAO(codigoEstacao)
```

```
74          ON UPDATE CASCADE
75          ON DELETE CASCADE
76  );
77
78  CREATE TABLE PREVISAO (
79      idPrevisao SERIAL NOT NULL,
80      dataPrevisao DATE NOT NULL,
81      horaPrevisao VARCHAR(4) NOT NULL,
82      probabilidadePrecipitacao FLOAT NOT NULL,
83      codigoEstacao INT NOT NULL,
84      CONSTRAINT PREVISAO_PK PRIMARY KEY (idPrevisao),
85      CONSTRAINT PREVISAO_ESTACAO_FK FOREIGN KEY (codigoEstacao)
        REFERENCES ESTACAO(codigoEstacao)
86          ON UPDATE CASCADE
87          ON DELETE CASCADE
88  );
89
90  CREATE TABLE USUARIO_acessa_MEDICAO_METEOROLOGICA (
91      idUsuario INT,
92      idMedicao INT,
93      CONSTRAINT UAMM_USUARIO_FK FOREIGN KEY (idUsuario) REFERENCES
        USUARIO(idUsuario)
94          ON UPDATE CASCADE
95          ON DELETE CASCADE,
96      CONSTRAINT UAMM_MEDICAO_METEOROLOGICA_FK FOREIGN KEY (idMedicao)
        REFERENCES MEDICAO_METEOROLOGICA(idMedicao)
97          ON UPDATE CASCADE
98          ON DELETE CASCADE
99  )
100
101 CREATE TABLE USUARIO_acessa_PREVISAO (
102     idUsuario INT,
103     idPrevisao INT,
104     CONSTRAINT UAP_USUARIO_FK FOREIGN KEY (idUsuario) REFERENCES USUARIO
        (idUsuario)
105         ON UPDATE CASCADE
106         ON DELETE CASCADE,
107     CONSTRAINT UAP_PREVISAO_FK FOREIGN KEY (idPrevisao) REFERENCES
        PREVISAO(idPrevisao)
108         ON UPDATE CASCADE
109         ON DELETE CASCADE
110 )
```

Listagem A.1 – *Script* físico do banco de dados. Autoria Própria.

O A.1 mostrado na Listagem A.1 foi utilizado para a implementação da simulação, foi necessária somente a criação física da tabela USUARIO, uma vez que ela atende todas

as necessidades da história de usuário US04 no escopo da simulação realizada.

APÊNDICE B – Código Fonte

Este anexo apresenta alguns dos códigos fontes desenvolvidos para realização da simulação ilustrada na Figura 33 visando atender a história de usuário US04 (cadastro de usuários).

B.1 *Backend*

Este anexo tem o objetivo de apresentar os códigos fonte desenvolvidos no *frontend* do projeto com o intuito de testar a arquitetura do sistema proposta na Seção 3.5.

```
1  import { inject, injectable } from "tsyringe";
2  import { ICreateUserDTO } from "../../dtos/ICreateUserDTO";
3  import { IUserRepository } from "../../repositories/IUserRepository";
4  import { hash } from "bcrypt";
5  import { AppError } from "../../../../errors/AppError";
6
7  @injectable()
8  class CreateUserService {
9    constructor(
10     @inject("UserRepository")
11     private usersRepository: IUserRepository
12   ) {}
13
14   async execute({
15     nomeCompleto,
16     email,
17     unidadeFederativa,
18     senha,
19   }: ICreateUserDTO): Promise<void> {
20     const userAlreadyExists = await this.usersRepository.findByEmail(
       email);
21
22     if (userAlreadyExists) {
23       throw new AppError(`User ${nomeCompleto} already exists`);
24     }
25
26     const passwordHash = await hash(senha, 8);
27
28     await this.usersRepository.create({
29       nomeCompleto,
30       email,
```

```
31        unidadeFederativa,
32        senha: passwordHash,
33      });
34    }
35  }
36  export { CreateUserService };
```

Listagem B.1 – Código fonte do processo de registro de usuários (*backend*). Autoria Própria.

A Listagem B.1 apresenta o código fonte desenvolvido para o *backend* do projeto. Esta implementação é responsável pela comunicação do software com a base de dados modelada para o cadastro de usuários ilustrado na Figura 33.

B.2 *Frontend*

Este anexo tem o objetivo de apresentar os códigos fonte desenvolvidos no *frontend* do projeto com o intuito de testar a arquitetura do sistema proposta na Seção 3.5.

```
1  import { useEffect, useState } from "react";
2  import { api } from "../../services/api";
3  import { CustomForm, Container, CustomParagraph } from "./styles";
4
5  export function SignUp() {
6    const [signUpMessage, setSignUpMessage] = useState("");
7    const [isSignUpSuccess, setIsSignUpSuccess] = useState(true);
8    const [name, setName] = useState("");
9    const [email, setEmail] = useState("");
10   const [senha, setSenha] = useState("");
11   const [confimarSenha, setConfirmarSenha] = useState("");
12
13   function signUp() {
14     if (senha !== confimarSenha) {
15       setSignUpMessage("As senhas nao sao correspondentes.");
16       setIsSignUpSuccess(false);
17       return null;
18     }
19
20     const form = {
21       nomeCompleto: name,
22       email: email,
23       senha: senha,
24       unidadeFederativa: "DF",
25     };
26
```

```
27      api.post("users", form).then((res) => {
28        setSignUpMessage(res.data.message);
29        setIsSignUpSuccess(true);
30        console.log(res);
31      }).catch((error) => {
32        setSignUpMessage(error.response.data.error.message);
33        setIsSignUpSuccess(false);
34        console.log(error.response.data);
35      });
36    }
37
38    return (
39      <Container>
40        <CustomForm>
41          <h1>Registrar-se</h1>
42          <input placeholder="Nome" onChange={(e) => setName(e.target.
      value)} />
43          <input
44            type="email"
45            placeholder="Email"
46            onChange={(e) => setEmail(e.target.value)}
47          />
48          <input
49            type="password"
50            placeholder="Senha"
51            onChange={(e) => setSenha(e.target.value)}
52          />
53          <input
54            type="password"
55            placeholder="Confirmar senha"
56            onChange={(e) => setConfirmarSenha(e.target.value)}
57          />
58          <button type="button" onClick={signUp}>
59            Registrar
60          </button>
61          <CustomParagraph isLoginSuccess={isSignUpSuccess}>
62            {signUpMessage}
63          </CustomParagraph>
64        </CustomForm>
65      </Container>
66    );
67  }
```

Listagem B.2 – Código fonte da tela de registro de usuários (*frontend*). Autoria Própria.

A Listagem B.2 apresenta o código fonte desenvolvido para renderizar a tela de cadastro de usuários. O resultado visual da execução deste código pode ser visto na Figura

33. A tecnologia utilizada para o desenvolvimento deste código foi a biblioteca JavaScript chamada React, descrito na Seção 3.7.5.

B.3 *Script* que executa o modelo de *machine learning*

Este anexo tem o objetivo de apresentar os códigos fonte desenvolvido no para a execução do modelo de *machine learning* desenvolvido no Capítulo 4.

```
1  import pandas as pd
2  import datetime
3
4  from sklearn.pipeline import make_pipeline
5  from sklearn.preprocessing import StandardScaler
6  from sklearn.svm import SVC
7  from sklearn.utils import resample
8  from utils.create_univariate_lp_model import get_univariate_predictions
9  from utils.sql.get_historic_data_station_all import
       get_historic_data_station_all
10 from utils.helpers import fill_na, reorder_columns
11
12
13 def run_prediction():
14     connection = 'postgresql://postgres:docker@172.17.0.1:5432/postgres'
15
16     df = pd.read_sql(
17         get_historic_data_station_all
18         .format(codigoEstacao='A001'), connection)
19
20     cleaned_data = fill_na(df)
21
22     cleaned_data['hora'] = cleaned_data['dataMedicao'].dt.hour
23     cleaned_data = reorder_columns(cleaned_data)
24
25     cleaned_data['dia_ano'] = cleaned_data['dataMedicao'].dt.day_of_year
26     cleaned_data = reorder_columns(cleaned_data)
27
28     cleaned_data['choveuHoje'] = cleaned_data['choveuHoje'].apply(
29         lambda x: 1 if x == 'S' else 0)
30
31     cleaned_data = cleaned_data.drop(columns=['choveAmanha'])
32
33     df_future = cleaned_data[-3:].copy()
34     df_future = df_future.reset_index(drop=True)
35     date = df_future.at[2, 'dataMedicao'].to_pydatetime()
36
```

```
37      sr_dates = df_future['dataMedicao']
38      sr_dates[len(sr_dates)] = date + datetime.timedelta(hours=1)
39
40      cleaned_data = cleaned_data[:-3]
41
42      cleaned_data = cleaned_data.drop(columns=['dataMedicao'])
43      df_future = df_future.drop(columns=['dataMedicao'])
44
45      X_future = df_future.drop(columns=['choveuHoje'])
46      X_future = X_future.reset_index(drop=True)
47      y_future = df_future['choveuHoje'].astype(float)
48
49      df_chuva = cleaned_data[cleaned_data['choveuHoje'] == 1].copy()
50      df_seco = cleaned_data[cleaned_data['choveuHoje'] == 0].copy()
51
52      df_seco_downsampled = resample(df_seco,
53                                     replace=True,
54                                     n_samples=len(df_chuva)*2,
55                                     random_state=42)
56
57      cleaned_data = pd.concat([df_seco_downsampled, df_chuva]).sample(
    frac=1)
58
59      y = cleaned_data['choveuHoje'].astype(float)
60      X = cleaned_data.drop(columns=['choveuHoje'])
61
62      X_train = X.values
63      y_train = y.values
64
65      clf = make_pipeline(StandardScaler(), SVC(gamma='auto', probability=
    True))
66
67      clf.fit(X_train, y_train)
68
69      df_new_future = get_univariate_predictions(1, date)
70
71      df_new_future = pd.concat([X_future, df_new_future])
72
73      prediction = clf.predict_proba(df_new_future)
74
75      df_new_future['probabilidadePrecipitacao'] = prediction[:, 1].tolist
    ()
76      df_new_future['codigoEstacao'] = 'A001'
77      df_new_future = df_new_future.reset_index(drop=True)
78      df_new_future = df_new_future.join(sr_dates)
79      df_new_future = df_new_future.rename(columns={'dataMedicao':'
    dataPrevisao'})
```

```
80      df_new_future = df_new_future.drop(columns=['dia_ano' , 'hora', 'mes
        '])
81
82
83      df_new_future.to_sql('PREVISAO', connection, if_exists='append',
        index=False)
```

Listagem B.3 – Código fonte do *script* de ML. Autoria Própria.

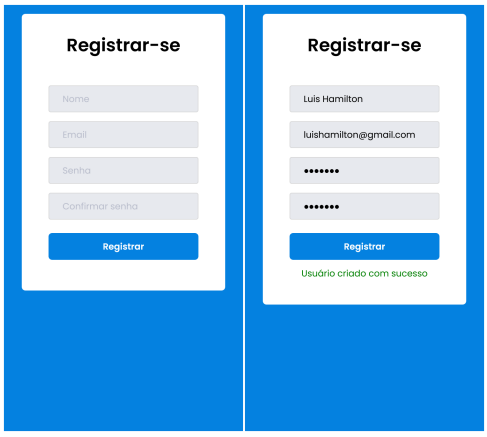
APÊNDICE C – Simulações

Este anexo tem o objetivo de apresentar as simulações desenvolvidas durante o TCC1 deste projeto.

Com o objetivo de verificação da viabilidade da arquitetura proposta, foi implementada a *feature* F01 - *Autenticação* contemplando as histórias de usuário US01 (Cadastro de usuários) e US04 (Autenticação de *Login*), disponíveis na Seção 3.4. Esta implementação busca testar a arquitetura e as tecnologias de desenvolvimento propostas, além de demonstrar o funcionamento e a comunicação do *Backend* e o *Frontend* planejados.

Na Figura 33 é apresentada a interface de usuário em que é realizado o cadastro de usuário.

Figura 33 – Interface de registro de usuários.

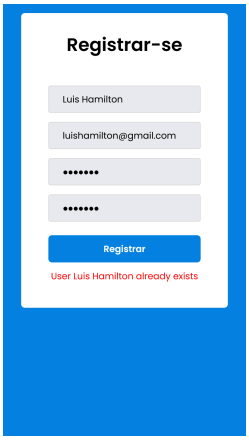


Fonte: Autoria Própria.

A Figura 33 apresenta a interface de usuário, em que é possível ver os campos destinados ao preenchimento dos dados do usuário a ser registrado na aplicação. Também é possível ver os dados preenchidos na interface de cadastro de usuário, que foram enviados do *Frontend* para o *Backend* e após a validação destes dados. O registro seria salvo no banco de dados e este gerará um identificador único aleatório para o cadastro daquele usuário. Após este processo, o *Backend* retorna o sucesso da operação, como pode ser observado em verde na Figura 33.

Na Figura 34 é apresentada a interface de registro de usuário quando ocorre uma falha no processo.

Figura 34 – Interface de falha no registro de usuário.

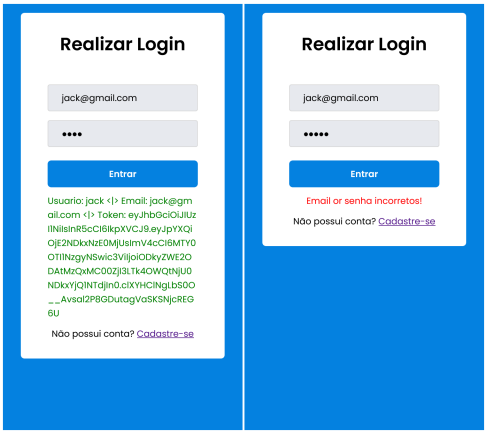


Fonte: Autoria Própria.

Ao preencher os dados de maneira incorreta, como demonstrado na Figura 34, o usuário será informado pela interface da aplicação com uma mensagem de erro, proveniente do *backend* da aplicação.

Na Figura 35 é apresenta dois possíveis resultados quando ocorre uma tentativa de login.

Figura 35 – Interface de sucesso (esquerda) e falha(direita) na realização da conexão de um usuário.



Fonte: Autoria Própria.

A interface de conexão (*login*) demonstrada na parte esquerda da Figura 35 exibe que o usuário se conectou a aplicação. Com os dados preenchidos corretamente, foi possível localizar o cadastro do usuário na base de dados cadastrais na própria tela de conexão. Na parte da direita da Figura 35 os dados de usuário foram preenchidos incorretamente, portanto o *login* não foi realizado e o registro deste usuário não foi localizado na base de dados. Assim, o *frontend* da aplicação informará sobre a falha na tentativa de conexão.

Com a implementação parcial da *feature* F01 e os casos de uso US01 e US04 torna-se possível perceber que a arquitetura proposta para a aplicação consegue realizar

a integração entre *Frontend* e *Backend* de maneira satisfatória, além de demonstrar a capacidade de realizar consultas e registros na camada de persistência na base de dados modelada. Assim, a arquitetura atende as expectativas propostas para o desenvolvimento deste projeto, viabilizando a continuidade do projeto e de sua implementação.