Módulo de geração de relatórios com Reconhecimento de Padrões no Sistema Web SISGEP-COMIC

Pedro Cassenote Batista, Alexandre Zamberlan
Curso de Ciência da Computação
UFN - Universidade Franciscana
Santa Maria - RS
pedro.cassenote@ufn.edu.br, alexz@ufn.edu.br

***Resumo*—Este projeto propôs a modelagem e o desenvolvimento de um módulo para geração de relatórios baseado no reconhecimento de padrões presentes no banco de dados do SISGEP-COMIC, o Sistema de Informação Gerencial da Universidade Franciscana. O módulo foi projetado e implementado em Python, utilizando a biblioteca pandas. Os resultados incluem a refatoração da modelagem inicial, a implementação do aplicativo Relatório (responsável por integrar a base de dados ao pandas), adaptações nas interfaces para atender às necessidades do administrador e do professor pesquisador, e o desenvolvimento de funcionalidades específicas como *widgets* para monitoramento de submissões de projetos com inconsistências. Vale destacar que o módulo responsável pela integração com o pandas e pelo reconhecimento de padrões foi desenvolvido como um protótipo de testes e, portanto, não foi incorporado ao sistema em produção.**

***Palavras-chave*: Mineração de Dados; Reconhecimento de Padrões; SISGEP-COMIC; biblioteca pandas**

I. Introdução

Este trabalho está inserido no contexto de Sistemas de Informação Gerencial (SIG) e Reconhecimento de Padrões, áreas fundamentais para a gestão e análise de dados em organizações. Um Sistema de Informação Gerencial é definido como a integração estratégica de tecnologias, recursos humanos, processos e documentos com o objetivo de coletar, processar e disseminar informações relevantes para o processo decisório. Esses sistemas permitem que os gestores acessem *insights* baseados em dados, facilitando a identificação de problemas, avaliação de desempenho e planejamento de estratégias futuras [1]. Assim, os SIG desempenham um papel crucial ao fornecer informações confiáveis e tempestivas para o suporte à tomada de decisão.

Por outro lado, o Reconhecimento de Padrões destaca-se como uma disciplina interdisciplinar que utiliza técnicas de mineração de dados e aprendizado de máquina para extrair conhecimento útil a partir de grandes volumes de dados brutos. Essa abordagem é responsável por identificar padrões, tendências e correlações, possibilitando a criação de modelos preditivos e a descoberta de *insights* valiosos [2]. No contexto deste trabalho, a aplicação combinada dessas áreas permite explorar dados gerenciais para identificar padrões ocultos, apoiar decisões estratégicas e promover melhorias nos processos organizacionais.

A. Justificativa

O sistema web SISGEP-COMIC [3], desenvolvido pelo Laboratório de Práticas da Computação da Universidade Franciscana, foi projetado para gerenciar submissões e avaliações de projetos científicos apresentados à Comissão Científica dos Hospitais Casa de Saúde e São Francisco de Assis. Contudo, até o presente momento, o sistema não possui um módulo dedicado à análise avançada de dados, como o reconhecimento de padrões ou a descoberta de conhecimento. Em outras palavras, não há um processo estruturado de mineração de dados capaz de identificar, por exemplo, quais tipos de projetos são predominantemente executados nos hospitais-escola ou qual área da Saúde realiza mais submissões de intervenções nesses hospitais.

Diante disso, esta pesquisa justifica-se pela necessidade de projetar e implementar um módulo no SISGEP-COMIC que integre técnicas de reconhecimento de padrões ao banco de dados existente. Esse módulo tem como objetivo explorar os dados armazenados para identificar tendências e padrões relevantes, gerando informações valiosas para a gestão científica e contribuindo para o planejamento estratégico das atividades de pesquisa nos hospitais-escola.

B. Objetivos

Projetar, implementar e avaliar módulo de mineração de dados para reconhecimento de padrões no sistema SISGEP-COMIC.

Os objetivos específicos foram:

- Pesquisar sobre os temas Mineração de Dados e Reconhecimento de Padrões;
- Pesquisar e testar bibliotecas do universo Python que implementem soluções sobre banco de dados MySQL Server;
- Projetar módulo de mineração de dados;
- Implementar módulo de mineração de dados;
- Integrar o módulo de mineração no SISGEP-COMIC;
- Definir um ambiente de avaliação e teste;
- Aplicar testes no ambiente de avaliação e teste;

- Organizar e comparar resultados.

II. Revisão bibliográfica

Esta seção aborda conceitos em subáreas de Sistemas de Informações para descoberta de conhecimento e mineração de dados.

A. Mineração de Dados, Reconhecimento de Padrões e Descoberta de Conhecimento

O reconhecimento de padrões, a descoberta de conhecimento e a mineração de dados estão intimamente relacionados e são elementos fundamentais no processo de análise de dados.

Mineração de dados é o processo de descobrir padrões, tendências e informações úteis a partir de conjuntos de dados volumosos e complexos. Ela envolve desde a aplicação de técnicas estatísticas, passando por algoritmos de redes neurais para aprendizado de máquina (*machine learning*), até a análise de dados para identificar relações e visões-entendimentos (*insights*) ocultos nos dados [4, 5, 6].

De acordo com Passos e Goldschmidt [4], pode-se afirmar que o objetivo da mineração de dados é dar significado às informações ou dados presentes em bases, gerando qualidade no processo de tomada de decisão. As decisões referem-se a prever comportamentos futuros, otimizar processos e encontrar soluções para problemas complexos em áreas como Negócios, Ciência, Saúde e Finanças.

O reconhecimento de padrões é uma parte crucial da mineração de dados, como já mencionado. Nessa fase, os dados coletados são analisados e contextualizados para identificar padrões de comportamento das informações. Isso possibilita a identificação de eventos futuros e relações anteriormente ocultas entre áreas distintas [4, 5].

Na parte final do processo, surge a descoberta de conhecimento, onde os padrões descobertos ou identificados transformam-se em informação compreensível e contextualizada sobre a área analisada, ou seja, conhecimento útil. É neste momento que os *insights* são criados, fruto dos processos anteriores [7].

Finalmente, o reconhecimento de padrões é o processo de identificar regularidades nos dados, a descoberta de conhecimento envolve extrair entendimentos ou *insights* úteis desses padrões, e a mineração de dados é o recurso computacional para realizar essas tarefas em larga escala. Todos esses elementos estão interligados e são essenciais para o processo de análise de dados e obtenção de *insights* significativos.

B. Ferramentas para Mineração de Dados, Reconhecimento de Padrões e Descoberta de Conhecimento

O emprego de ferramentas do ecossistema Python e suas bibliotecas para descoberta de conhecimento, mineração de dados e reconhecimento de padrões oferece uma combinação única de facilidade de uso, poder computacional, uma variedade de funcionalidades e a referência de uma comunidade atuante. Essa combinação de fatores tem consolidado o Python como uma escolha não apenas popular, mas também eficaz para profissionais envolvidos em Ciência de Dados, ou seja, uso de técnicas computacionais para descoberta de conhecimento.

Dentre os recursos do ecossistema Python, é possível citar:

- MySQL Connector é uma biblioteca para facilitar a integração entre programas Python e bancos de dados MySQL. Ela fornece uma interface Python para se conectar a um servidor MySQL, executar consultas SQL, recuperar resultados e gerenciar transações [8];
- Pandas: é uma biblioteca que fornece estruturas de dados e ferramentas de análise de dados de alto desempenho e com facilidade de uso. O principal objeto em pandas é o *DataFrame*, que é uma estrutura de dados tabular semelhante a uma planilha do eletrônica, que permite manipular e analisar conjuntos de dados de forma eficiente dentro de um programa Python [9];
- *Scikit-learn* é uma biblioteca de aprendizado de máquina de código aberto que oferece uma variedade de algoritmos de aprendizado supervisionado e não supervisionado, bem como ferramentas para avaliação, pré-processamento de dados e seleção de modelos. Ele é projetado para ser simples e eficiente, e é amplamente utilizado em projetos de Ciência de Dados [10];
- NumPy é uma biblioteca fundamental para computação científica (por exemplo, operações algébricas), permitindo que o desenvolvedor realize operações de matrizes multidimensionais com uma grande quantidade de dados [11];
- Matplotlib é uma biblioteca voltada para a criação de visualizações estáticas ou interativas de dados, permitindo a criação de gráficos e sintetização das informações desejadas [12];
- Seaborn é uma biblioteca que se baseia na Matplotlib e é capaz de se integrar com as estruturas de dados da biblioteca pandas. Ela permite a criação de gráficos com base nos *DataFrames* de pandas. Além disso, realiza internamente o mapeamento de significados necessários e a agregação estatística para produzir gráficos [13];
- TensorFlow é uma biblioteca que permite a criação de modelos de aprendizado de máquina para computadores, dispositivos móveis e Web [14].

Percebe-se que há três bibliotecas que contemplam cenários parecidos, pandas, scikit-learn e TensorFlow, porém cada uma tem um propósito específico e pode ser mais adequada dependendo do contexto do projeto. pandas é ideal para pré-processamento de dados, limpeza, manipulação e transformação de conjuntos de dados. Tem como cenário típico o processo de carregar dados de diferentes fontes (CSV, Excel, SQL, etc.), limpar e preparar os dados para análise, manipulação de séries temporais, análise explora-

tória de dados, etc. Não é aconselhada utilizá-la quando se tem tarefas de modelagem preditiva ou para treinamento de modelos de aprendizado de máquina [9]. Já scikit-learn oferece uma ampla gama de algoritmos de classificação, regressão, *clustering*, redução de dimensionalidade, entre outros. Tem como cenário adequado o treinamento e avaliação de modelos de aprendizado de máquina supervisionado e não supervisionado, seleção de características e validação cruzada, por exemplo. Não deve ser utilizado para lidar com redes neurais profundas e modelos de aprendizado profundo [10]. Por fim, TensorFlow é utilizado para construir e treinar modelos de aprendizado profundo, incluindo redes neurais convolucionais (CNNs), redes neurais recorrentes (RNNs), modelos de sequência, entre outros. Seus principais cenários de aplicação são desenvolvimento de modelos de aprendizado profundo, incluindo visão computacional, processamento de linguagem natural e reconhecimento de voz [14]. Não deve ser utilizado em conjuntos de dados pequenos ou modelos simples. Neste caso, pode ser mais eficiente usar bibliotecas mais simples como pandas e scikit-learn.

Dessa forma, pandas é ótimo para manipulação e preparação de dados, scikit-learn é ideal para modelagem de aprendizado de máquina tradicional e TensorFlow é ideal para construir e treinar modelos de aprendizado profundo. A escolha da biblioteca depende das necessidades específicas do projeto e do tipo de análise ou modelagem que se está realizando.

Também há duas bibliotecas similares no contexto de geração de gráficos e visualização de dados: Matplotlib e Seaborn. Entretanto, elas têm diferentes propósitos e estilos, e a escolha entre elas depende das necessidades específicas do projeto. Matplotlib oferece controle granular sobre a criação de gráficos e visualizações, permitindo personalizações detalhadas. Matplotlib é ideal quando você precisa de flexibilidade para criar gráficos personalizados ou quando está trabalhando em um contexto em que precisa integrar visualizações em uma interface de usuário. Dessa forma, tem como cenário ideal a criação de gráficos personalizados, gráficos 3D, incorporação de gráficos em aplicações Web ou interfaces de usuário [12]. Seaborn simplifica a criação de gráficos estatísticos complexos e estilizados com apenas algumas linhas de código. Seaborn é ideal quando você precisa criar visualizações estatísticas comuns de forma rápida e eficiente, sem se preocupar muito com os detalhes de personalização. Os cenários típicos são visualização de relacionamentos entre variáveis, *plots* de distribuição, visualizações categóricas, visualizações estilizadas para apresentações e relatórios [13].

Em termos de qualidade do serviço, ambas as bibliotecas são excelentes e amplamente utilizadas pela comunidade de Ciência de Dados e visualização.

C. Algoritmos na mineração de dados

A mineração de dados permite a utilização de diversos algoritmos para a descoberta de conhecimento, divididos em dois tipos principais: associação e regressão [4, 15, 2].

1) Algoritmos de Associação: Esses algoritmos são usados para descobrir relações entre itens em grandes conjuntos de dados. O objetivo é identificar padrões frequentes, como "se A ocorre, então B também ocorre frequentemente", onde cada regra tem um antecedente (se) e um consequente (então). Eles são particularmente úteis para descobrir associações entre diferentes itens, como produtos comprados juntos em uma loja, padrões de navegação na Web ou sintomas de uma doença. No entanto, os algoritmos de associação não são tão eficazes quando os dados não apresentam muitas relações entre os itens ou quando os padrões de interesse são muito específicos e raros. Esses algoritmos ajudam a transformar grandes volumes de dados em *insights* acionáveis, facilitando a tomada de decisões em diversos setores [4, 15].

Os tipos de algoritmos de associação mais utilizados na mineração de dados incluem [4, 15]:

- Apriori: é uma técnica clássica para a mineração de regras de associação, usada para identificar conjuntos de itens frequentes em grandes bancos de dados. Ele baseia-se na propriedade *Apriori*, que afirma que todos os subconjuntos de um conjunto de itens frequente também devem ser frequentes. O algoritmo começa identificando os itens individuais (*single items*) que aparecem com frequência mínima (suporte mínimo) nas transações, gera conjuntos de itens de tamanho maior (pares, trios, etc.) e filtra aqueles que também atendem ao suporte mínimo. Esse processo é repetido até que não seja possível encontrar conjuntos de itens maiores que atendam ao critério de suporte;
- FP-Growth (*Frequent Pattern Growth*): é uma técnica eficiente para a mineração de regras de associação, usada para encontrar conjuntos de itens frequentes em grandes bases de dados sem a necessidade de gerar múltiplos candidatos como no algoritmo Apriori. Ele utiliza uma estrutura de dados compacta chamada FP-Tree (*Frequent Pattern Tree*) para armazenar a informação de forma eficiente. Inicialmente, o algoritmo faz uma única varredura no banco de dados para contar a frequência de cada item, os itens então são ordenados de acordo com sua frequência e, com base nessa ordenação, o algoritmo faz uma segunda varredura no banco de dados, inserindo as transações na FP-Tree. Cada caminho na árvore representa uma transação, e os nós compartilham prefixos comuns. A FP-Tree é explorada para extrair conjuntos de itens frequentes. Isso é feito dividindo a árvore em subárvores condicionais (subárvores para cada item) e repetindo o processo de mineração recursivamente. A partir dos conjuntos

de itens frequentes extraídos, o algoritmo gera regras de associação, calculando as medidas de suporte e confiança. FP-Growth é especialmente eficiente para conjuntos de dados grandes, onde a construção da FP-Tree permite uma representação compacta e evita múltiplas varreduras no banco de dados;
- Eclat (*Equivalence Class Transformation*): o algoritmo é uma técnica de mineração de regras de associação que usa uma abordagem baseada em interseção de conjuntos de transações, em vez de geração de candidatos como no Apriori ou uma estrutura de árvore como no FP-Growth. Ele é conhecido por sua eficiência em encontrar conjuntos de itens frequentes através da exploração de equivalência de classes. As transações são convertidas para uma representação vertical, onde cada item é associado a uma lista de IDs de transações (TID list) nas quais o item aparece. A partir da representação vertical, o algoritmo combina TID lists para identificar conjuntos de itens frequentes. Por exemplo, para encontrar a frequência de um par de itens A, B, o algoritmo realiza a interseção das TID lists de A e B. Este processo de interseção é aplicado recursivamente para conjuntos de itens maiores, permitindo a identificação de todos os conjuntos de itens frequentes. Uma vez identificados os conjuntos de itens frequentes, o algoritmo gera regras de associação baseadas nesses conjuntos, calculando as medidas de suporte e confiança. É especialmente útil quando há muitas transações, pois a interseção de TID lists pode ser mais rápida que a geração de candidatos.

2) Algoritmos de Regressão: Esses algoritmos são usados para prever valores contínuos com base em variáveis independentes. Eles estabelecem uma relação entre as variáveis de entrada e a variável de saída. Eles são úteis quando se precisa prever um valor numérico com base em outras variáveis. Por exemplo, prever o preço de uma casa com base em características como tamanho, número de quartos, etc. Eles não são adequados quando a relação entre as variáveis não é linear ou quando os dados não seguem um padrão previsível [4, 15].

Os principais algoritmos de regressão utilizados na mineração de dados incluem [4, 15]:

- Regressão Linear Simples e Múltipla: a regressão linear é um dos métodos mais simples e amplamente utilizados para modelar a relação entre uma variável dependente e uma ou mais variáveis independentes. Na regressão linear simples, há apenas uma variável independente, enquanto na regressão linear múltipla, há múltiplas variáveis independentes;
- Regressão Logística: embora o nome contenha regressão, a regressão logística é na verdade um modelo de classificação binária que estima a probabilidade de uma variável dependente categórica (ou binária) com base em uma ou mais variáveis independentes. É muito utilizado em problemas de classificação, como prever se um e-mail é *spam* ou não, ou se um paciente tem uma determinada doença;
- Árvores de Decisão e Florestas Aleatórias (*Random Forest*): embora geralmente sejam associadas à classificação, as árvores de decisão e as florestas aleatórias também podem ser usadas para problemas de regressão. Esses algoritmos dividem repetidamente os dados em subgrupos com base nas variáveis independentes até que cada subgrupo tenha uma resposta média que serve como previsão;
- Máquinas de Vetores de Suporte (SVM): são frequentemente utilizadas para classificação, mas também podem ser adaptadas para problemas de regressão. Elas buscam encontrar o hiperplano que melhor separa os pontos de dados de acordo com a variável de saída, sendo capazes de lidar com dados não lineares por meio do uso de núcleos;
- Redes Neurais Artificiais: são modelos computacionais inspirados na estrutura do cérebro humano. Elas podem ser usadas para problemas de regressão, onde os dados são alimentados em uma rede de neurônios que ajusta os pesos das conexões entre eles para prever a variável de saída.

Quando decidir entre esses algoritmos, é importante entender a natureza dos seus dados e os objetivos da análise. Portanto, pode-se afirmar que Associação é utilizada para descobrir padrões interessantes e relações entre variáveis em conjuntos de dados, útil para identificar associações frequentes entre itens ou atributos. Como exemplos incluem análise de cestas de compras, onde se deseja descobrir quais itens são frequentemente comprados juntos. Já Regressão é utilizada para prever o valor de uma variável dependente com base em uma ou mais variáveis independentes, útil para modelagem preditiva e compreensão de relações causais entre variáveis. Como exemplos, incluem previsão de vendas com base em publicidade e preços, previsão de preço de imóveis com base em características, etc.

D. Trabalhos relacionados

Esta seção apresenta trabalhos que usaram tecnologias Python, Django e Bootstrap, além de técnicas de mineração de dados no contexto de banco de dados, ou seja, base de dados estruturada.

O trabalho *SISGEP-COMIC Gestão de Avaliação de Projetos* [3] é a base desta pesquisa e foi projetado, desenvolvido, documentado e disponibilizado em https://comic.lapinf.ufn.edu.br. O sistema possibilita submeter, avaliar, acompanhar e monitorar projetos da área da Saúde, de professores de instituições de ensino superior, que desejam realizar atividades de pesquisa ou extensão nos hospitais escola da Universidade Franciscana. Por meio do sistema, o professor submete título, resumo, objetivos,

justificativa, plano de intervenção ou ação, cronograma, documentação de autorização do Comitê Ético em Pesquisa com Seres Humanos. O administrador do sistema delega avaliadores ao projeto, que geram pareceres favoráveis ou não. O professor pode corrigir e adequar as solicitações ou cancelar o projeto. Uma vez adequado e aprovado, o projeto entra em execução e o administrador consegue monitorá-lo, via documentos (comprobatórios de atividades) entregues mensalmente pelo professor responsável pelo projeto. O sistema foi construído com tecnologias do ecossistema Python, como os *frameworks* Django e Bootstrap [3].

As Figuras 1, 2 e 3 ilustram o ambiente SISGEP-COMIC. Destaca-se a Figura 3 em que o administrador acompanha e gera relatórios de projetos aprovados. É nesta funcionalidade que o módulo proposto foi aplicado.

Figura 1. Interface de Login do SISGEP-COMIC (com alterações realizadas por esta pesquisa).

Figura 2. Interface do Administrador do SISGEP-COMIC.

Referente ao trabalho *Algoritmos de Mineração de Dados em Sistema de Monitoramento de Diabetes* [16], foi discutido o contexto de descoberta de conhecimento em bases de dados referentes a pacientes diabéticos. A base continha dados de monitoramento de data, quantidade de insulina utilizada, quantidade de calorias e carboidratos ingeridos, qualidade do sono (escala Likert) e esforço subjetivo de atividades físicas (escala Borg). Dessa forma, a pesquisa buscou estudar, aplicar, avaliar e apontar métodos de mineração de dados adequados para o contexto da doença diabetes. Assim, foram elencados três algoritmos (RandomForest, Expectation Maximisation e Bagging), presentes na ferramenta

Figura 3. Interface com alguns exemplos de projetos aprovados e em acompanhamento pelo SISGEP-COMIC.

Weka, que retornaram respostas significativas.

Já o trabalho *Aplicação das Bibliotecas Python para tratamento de dados em tempo real: A análise dos dados de isolamento social em Santa Catarina* [17] aborda com mais detalhes o código utilizado nas ferramentas pandas, Matplotlib e Numpy, assim como as etapas necessárias para a mineração de dados e descoberta de conhecimento.

Outro trabalho, bastante importante, é o *PRENATAL MANAGEMENT* [18], desenvolvido pela Universidade Franciscana, sob a orientação de Sylvio André Garcia Vieira. O sistema gerencia dados do Sistema Único de Saúde (SUS), em que armazena dados de doenças como dengue, malária, covid, entre outras, nos bairros de cidades. Além disso, gera processos de mineração de dados (via algoritmos de Regressão e Associação), predizendo possíveis casos de doenças em bairros e meses futuros. O sistema é Web disponibilizado no endereço https://ppsusufn.lapinf.ufn.edu.br e implementado em Python, nos *frameworks* Django e Bootstrap, com as bibliotecas pandas, NumPy e SciKit Learning.

No contexto dos trabalhos relacionados, destaca-se o estudo conduzido por Müller e Zamberlan [3], que oferece documentação e explicação detalhadas sobre o funcionamento do SISGEP-COMIC. As pesquisas de Cezário e Zamberlan [16], assim como o trabalho de Backes e colaboradores [18], contribuem para esta pesquisa ao fornecerem uma revisão dos algoritmos de mineração de dados em uma estruturação abrangente. Adicionalmente, a investigação conduzida por Vincentainer, Mattedi e Mello [17] apresenta um roteiro detalhado para a utilização eficaz das bibliotecas pandas, Matplotlib e Numpy, oferecendo suporte essencial para a análise e visualização de dados.

Ressalta-se que os sistemas Web projetados, construídos e disponibilizados nos trabalhos [3, 16, 18], são pesquisas do Laboratório de Práticas da Computação UFN e são baseados no modelo arquitetura *Model-View-Template* (MVT), associado ao *framework* Django. De acordo com o site oficial Django [19], esse modelo arquitetural é uma variação do padrão *Model-View-Controller* (MVC) que apresenta várias vantagens, especialmente em aplicações Web. O MVT do Django organiza a estrutura do projeto de uma maneira

que facilita o desenvolvimento, manutenção e expansão da aplicação. Em geral, o Django organiza a aplicação em três componentes principais: *models* (define a estrutura do banco de dados e as regras de negócios associadas aos dados), *views* (controla a lógica da aplicação e decide quais dados devem ser exibidos ou processados) e *templates* (cuida da apresentação, definindo como os dados serão exibidos para o usuário). Além disso, possui o recurso Mapeamento Objeto-Relacional (ORM), que permite a manipulação do banco de dados usando código Python, simplificando operações complexas e aumentando a produtividade. Essa abordagem [19] permite que desenvolvedores criem rapidamente aplicações robustas e de alta qualidade.

III. Metodologia e proposta do trabalho

Em relação à realização da pesquisa, a metodologia *Scrum* [20] foi utilizada, bem como a técnica *Kanban* para a gestão das atividades assumidas no cronograma. As ferramentas ou materiais utilizados são Trello; GitHub; Ferramenta Astah; Linguagem Python; Biblioteca Pandas do universo Python; Editor Visual Studio Code.

No contexto da metodologia Scrum, destacam-se os elementos fundamentais: papéis, eventos, artefatos e as regras que os interligam. Entre os papéis, o *Product Owner* é responsável por gerenciar e priorizar os itens no *Product Backlog*, que consiste em uma lista organizada de funcionalidades e requisitos mapeados para o produto. No cenário descrito, o *Product Owner* pode ser representado pela professora Camila Franco (atual gestora do SISGEP-COMIC), encarregada de priorizar os benefícios em manutenção, garantindo que as entregas agreguem valor ao sistema. O *Scrum Master*, por sua vez, assume o papel de facilitador da metodologia, promovendo a adoção das boas práticas do Scrum e removendo impedimentos. No caso, esse papel é desempenhado pelo professor orientador, que também atua como mentor do processo. Já o *Development Team* (*Dev Team*) é formado pelo aluno e pelo professor orientador, que colaboram no desenvolvimento do sistema, concentrando-se na entrega dos itens definidos no S*print Backlog*.

Quanto aos eventos do Scrum, um dos mais importantes é o *Sprint*, ciclo de trabalho de curta duração que, neste caso, pode ter periodicidade semanal ou de no máximo 15 dias. Durante o *Sprint*, o time foca na entrega de incrementos funcionais e avaliáveis do produto. Ao final de cada ciclo, ocorre o *Sprint Review*, onde o item desenvolvido é apresentado ao cliente para validação. Esse *feedback* é essencial para ajustar os próximos passos. Antes de cada *Sprint*, realiza-se o *Sprint Planning*, que define o próximo conjunto de itens a serem abordados, sempre baseado nas prioridades do *Product Backlog* e nas autorizações do *Product Owner*. Adicionalmente, a equipe promove retrospectivas para identificar melhorias no processo e aumentar a eficiência nas entregas futuras.

A. Modelagem

A Figura 4 mostra a relação da proposta de trabalho com o sistema existente e em funcionamento. No contexto SISGEP-COMIC é possível perceber que o sistema é focado em computação em nuvem e com comportamento de arquitetura SaaS (*Software as a Service*), construído com tecnologias Python, Django, Bootstrap e MySQL. Já o conceito Sistema de Apoio à Decisão mostra a dependência com as tecnologias em Python e com a área de descoberta de conhecimento via reconhecimento de padrões pelo processo de mineração de dados.

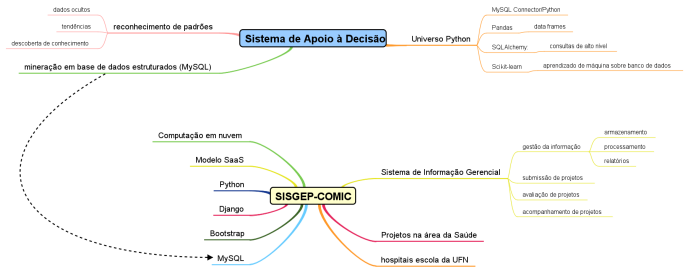

Figura 4. Mapa Mental ilustrando o contexto do sistema SISGEP-COMIC e a relação com o módulo mineração.

A Figura 5 apresenta o Diagrama de Casos de Uso, em que há dois pacotes: i) pacote do trabalho do Patrick Müller [3], como o projeto e a implementação do SISGEP-COMIC; ii) pacote do módulo de mineração para auxiliar na gestão de relatórios.

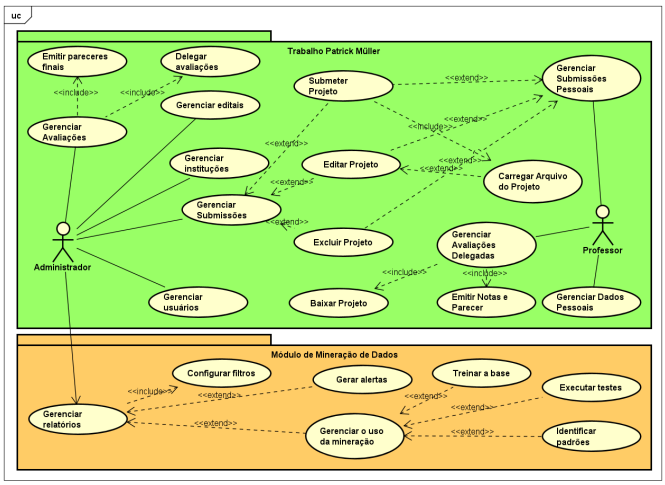

Figura 5. Diagrama de Casos de Uso contemplando o projeto do SISGEP-COMIC e o módulo de mineração.

Na Figura 6, apresenta-se a ideia de arquitetura do SISGEP-COMIC com o módulo de mineração proposto, bem como a relação das bibliotecas Python. Destaca-se que essa arquitetura tem como base o trabalho realizado por Sylvio Vieira e colaboradores [18].

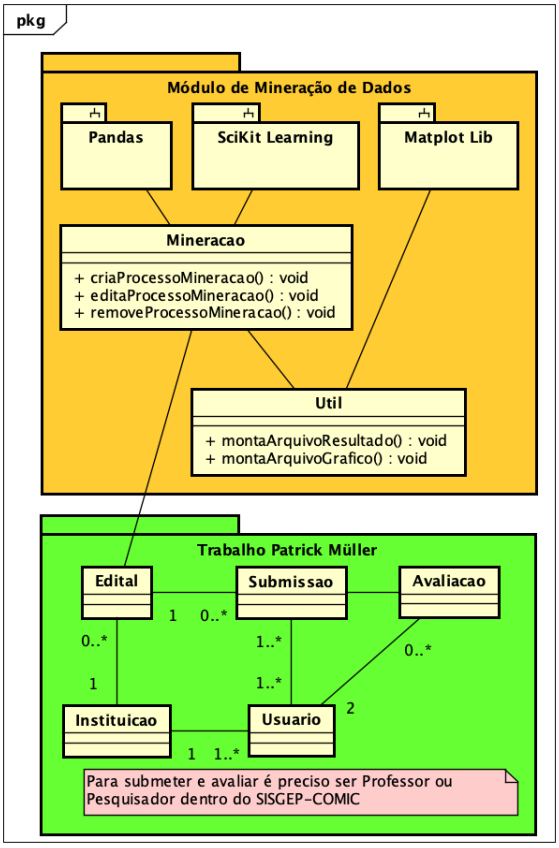

Figura 6. Diagrama de Classe contemplando o projeto do SISGEP-COMIC e o módulo de mineração.

IV. Resultados Alcançados

Nesta seção, os resultados são relacionados à prototipação de funcionalidades dentro do SISGEP-COMIC.

A. *Interfaces e códigos do módulo prototipado*

A Figura 7 ilustra, na visão do Administrador, o novo aplicativo Django (*app*) Relatório, em que o administrador pode gerar, listar e consultar relatórios a partir da integração do banco de dados do SISGEP-COMIC com o *framework* pandas. Registra-se, entretanto, que toda a implementação realizada é um módulo protótipo, ainda bastante distante do resultado desejado, com perguntas e respostas mais robustas e complexas. Aqui, decidiu-se mostrar e explicar a forma de integração do banco de dados *dbsqlite*, usado no protótipo para testes, com *dataframes* em pandas.

Na Figura 8, o administrador ou coordenador do SISGEP-COMIC pode visualizar os resultados do relatório gerado, tendo como visualização a exibição padrão do pandas. Vale ressaltar novamente que o objetivo foi ilustrar esse processo de integração, embora ainda distante de uma solução robusta.

Para criar a funcionalidade de integração do banco de dados com o pandas, foi criada a classe Conecta com um método estático *gera_dataframe*. Este método é invocado,

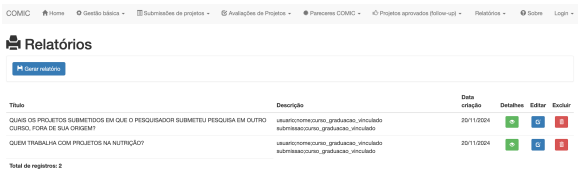

Figura 7. Visão do administrador, Gestão de Relatórios.

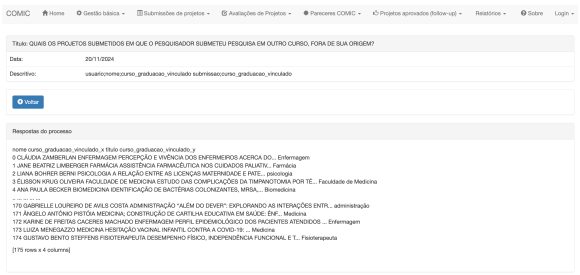

Figura 8. Após consulta gerada, detalhes de visualização (*DetailView*).

tanto na classe *views.RelatorioCreateView*, quanto na classe *views.RelatorioUpdateView*, que criam e editam um relatório, respectivamente. A Figura 9 faz a conexão com a base de dados (linha 91), monta duas consultas referentes as tabelas *Usuario* e *Submissao* (entre as linhas 93 e 107), cria os *dataframes* referentes a essas duas consultas (linhas 109 e 110), realiza o processo de unificação ou mesclagem dos *dataframes* gerados (linha 114) e gera um *dataframe* de resposta a partir das colunas selecionados no descritivo do relatório. Aqui, registra-se que este processo ainda não está ocorrendo de forma desejada, mas sim como um teste para avaliar tal integração.

```
87  class Conecta:
88      @staticmethod
89      def gera_dataframe():
90          # Conexão com o SQLite
91          conexao = sqlite3.connect('projeto/db.sqlite3')
92          # Query para tabela de usuário
93          consulta_usuario = """
94          SELECT
95              usuario.id AS usuario_id, usuario.nome, usuario.area_conhecimento_cnpq, usuario.curso_graduacao_vinculado,
96              usuario.curso_pos_graduacao, usuario.grupo_pesquisa
97          FROM
98              usuario_usuario as usuario
99          """
100         # Query para tabela de submissao
101         consulta_submissao = """
102         SELECT
103             submissao.edital_id, submissao.responsavel_id, submissao.area, submissao.curso_graduacao_vinculado,
104             submissao.curso_pos_graduacao, submissao.grupo_pesquisa, submissao.titulo, submissao.resumo
105         FROM
106             submissao_submissao AS submissao
107         """
108         # Carrega os dados no pandas DataFrame, para as duas consultas
109         data_frame_usuario = pandas.read_sql_query(consulta_usuario, conexao)
110         data_frame_submissao = pandas.read_sql_query(consulta_submissao, conexao)
111         # Feche a conexão
112         conexao.close()
113         # Mescla os dois DataFrames
114         data_frame_merge = data_frame_submissao.merge(data_frame_usuario, left_on='responsavel_id', right_on='usuario_id', how='left')
115
116         # Colunas do DataFrame mesclado
117         # ['edital_id', 'responsavel_id', 'area', 'curso_graduacao_vinculado_x', 'curso_pos_graduacao_x', 'grupo_pesquisa_x', 'titulo',
118         # 'resumo', 'usuario_id', 'nome', 'area_conhecimento_cnpq', 'curso_graduacao_vinculado_y', 'curso_pos_graduacao_y', 'grupo_pesquisa_y']
119
120         # Seleciona as colunas do descritivo
121         colunas_selecionadas = data_frame_merge[['nome', 'curso_graduacao_vinculado_x', 'titulo', 'curso_graduacao_vinculado_y']]
122         # Retorna o DataFrame
123         return colunas_selecionadas
```

Figura 9. Classe Conecta dentro da camada *views* do *app* Relatório.

Por fim, a Figura 10 com a Classe *views.RelatorioCreateView*, invocando o método *gera_dataframe* (linha 54). O resultado é armazenado na variável *resposta* da camada *models* de Relatório, que é exibida quando o administrador solicitar a visualização do detalhe do relatório conforme a Figura 8.

Outras melhorias foram realizadas no SISGEP-COMIC,

```python
class RelatorioCreateView(LoginRequiredMixin, CreateView):
    model = Relatorio
    fields = ['titulo','descricao']
    success_url = 'relatorio_list'

    def get_success_url(self):
        messages.success(self.request, 'Relatorio gerado com sucesso na plataforma!')
        return reverse(self.success_url)

    def form_valid(self, form):
        relatorio = form.save()

        # aplicar conexao com pandas
        relatorio.resposta = Conecta.gera_dataframe()

        relatorio.save()
        return super(RelatorioCreateView, self).form_valid(form)
```

Figura 10. Classe *RelatorioCreateView* da camada *view* do *app* Relatório.

que não envolveram o uso de pandas, mas incrementos em consultas Django, que geram *widgets*[1] com informações importantes ao administrador, coordenador do sistema e pesquisador responsável por projeto submetido ao SISGEP-COMIC. Na Figura 11, é possível visualizar indicadores para *Submissões*, *Avaliações*, *Pareceres e projetos aprovados com pendências*. Já na Figura 12, há os *widgets* como indicadores de atenção em projetos submetidos, em avaliação ou com pendências de documentação após aprovados (visão do professor pesquisador).

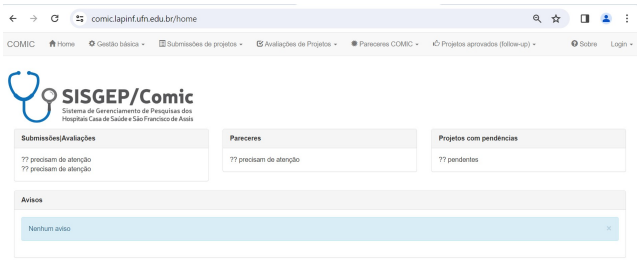

Figura 11. Nova interface visão administrador.

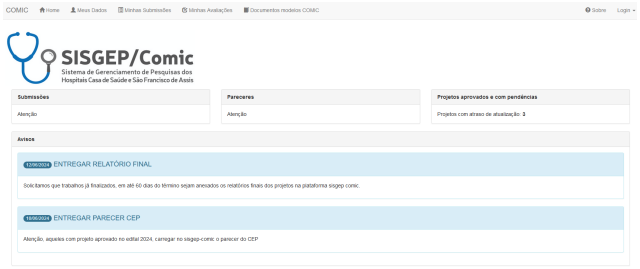

Figura 12. Nova interface visão professor.

Na Figura 13, há métodos dentro da classe *Usuario*, na camada *models* do modelo MVT, com notação *@property*, para contabilizar ou sinalizar informações para os *widgets*. Segundo a documentação oficial do Django [19], em Python, a notação *@property* é um *decorador* que permite definir métodos dentro de uma classe de forma que eles sejam acessados como se fossem atributos. No contexto do Django, quando se utiliza o *decorador @property* em um método dentro de uma classe, por exemplo *models.Usuario*, está criando-se uma propriedade que pode ser acessada diretamente, sem a necessidade de chamá-la/invocá-la como um método. Além disso, este *property* pode ser invocado dentro de um código html, como ilustra a Figura 14, que mostra os códigos das linhas 20 e 21, invocando *property* associados ao objeto usuário logado.

[1]São pequenos aplicativos que podem ser integrados em uma interface, como sites. Eles geralmente oferecem funcionalidades específicas e permitem que os usuários acessem informações ou realizem tarefas rapidamente [21]

```python
@property
def total_submissoes_atencao(self):
    return Submissao.objects.filter(Q(avaliacao__comissao = None) | Q(avaliacao__comissao__status ='EM EDIÇÃO') | Q(avaliacao__comissao__status='PÓS CORREÇÃO') | Q(avaliacao__comissao__status='PENDENTE')).count()

@property
def total_avaliacoes_atencao(self):
    return Avaliacao.objects.filter(Q(comissao = None) | Q(comissao__status ='EM EDIÇÃO') | Q(comissao__status ='EM ANÁLISE') | Q(comissao__status='PÓS CORREÇÃO') | Q(comissao__status='PENDENTE')).count()

@property
def total_em_comissao_atencao(self):
    return Comissao.objects.filter(Q(status='PÓS CORREÇÃO') | Q(status='PENDENTE')).count()

@property
def total_projetos_aprovados_sem_atualizacao(self):
    data_atual = timezone.now().date()
    return Submissao.objects.filter(
        Q(avaliacao__comissao__status ='APROVADO') &
        Q(dt_atualizacao_submissao__lte=data_atual - timedelta(days=60))
    ).count

@property
def pesquisador_total_projetos_aprovados_sem_atualizacao(self):
    data_atual = timezone.now().date()
    return Submissao.objects.filter(Q(responsavel=self) &
        Q(avaliacao__comissao__status ='APROVADO') &
        Q(dt_atualizacao_submissao__lte=data_atual - timedelta(days=60))
    ).count
```

Figura 13. Classe *models.Usuario* com métodos *@property* para os *widgets* construídos.

Ao observar os diferentes métodos, é possível notar que a linguagem SQL de Banco de Dados não é acionada diretamente, mas sim a linguagem de consulta (*queryset*) do *framework* Django, baseada em orientação a objetos e uso de estrutura de dados tipo lista. Por exemplo, o que ocorre na linha 124, em que a *queryset* está contabilizando todas as avaliações sem parecer da comissão de avaliação e que estão com status em edição, pós-correção e pendente.

```python
{% if user.is_authenticated %}
    <div class="container-fluid">
        <!-- Primeira linha -->
        <div class="row">
        <div class="col-md-4">
            <!-- Conteúdo da primeira coluna -->
            <div class="panel panel-default">
            <div class="panel-heading"><b>Submissões | Avaliações</b></div>
            <div class="panel-body">
                Atenção em submissões: <b>{{ request.user.total_submissoes_atencao }}</b> <br>
                Atenção em avaliações: <b>{{ request.user.total_avaliacoes_atencao }}</b>
            </div>
            </div>
        </div>
```

Figura 14. Código html da visão HOME do administrador.

Já do lado do professor-pesquisador, os projetos aprovados e sem atualizações aparecem conforme a Figura 15, onde a coluna *Status do Projeto* indica quantos dias o projeto não recebe atualizações obrigatórias a cada 60 dias.

Para que essa funcionalidade ocorresse, foi necessária a implementação de *templatetags* com a notação (*decorators*) *@register.filter*, conforme ilustrado na Figura 16. Na Figura 17, há o exemplo de como invocar um método na camada *template* (html), passando parâmetros. Isso evita o uso de *JavaScript*, mantendo todo o código em Python.

Por fim, na Figura 18, na coluna *Data última atualização*,

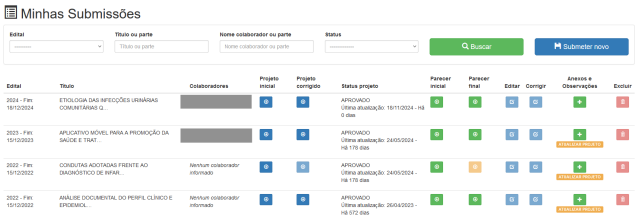

Figura 15. Visão professor com listagens de projetos aprovados e dias em atraso de atualização.

```python
from django import template

register = template.Library()

@register.filter
def diferenca_dias(data_atual, data_analise):
    """Calcula a diferença de dias entre data_atual e a data_analise"""
    data_analise = data_analise.date()
    return (data_atual - data_analise).days
```

Figura 16. Código Python com filtro a ser chamado na camada *template*.

```
{% if submissao.avaliacao.comissao %}
    {{ submissao.avaliacao.comissao.get_status_display|upper }}<br>
    {% if submissao.avaliacao.comissao.status == 'APROVADO' %}
        <p> Última atualização: {{ submissao.dt_atualizacao_submissao|date:'d/m/Y' }} -
            Há {{ submissao.get_data_atual|diferenca_dias:submissao.dt_atualizacao_submissao }} dias</p>
        {% comment %} <p>Diferença de dias: {{ submissao.get_data_atual|date:"U" | floatformat:0|add:"-"|
    {% endif %}
```

Figura 17. Código html invocando um filtro Python da *templatetags*.

há a indicação ou não de que o projeto precisa de atualização por parte do pesquisador responsável.

Figura 18. Visão do administrador: listagens de projetos aprovados e indicações de atrasos.

Todas essas funcionalidades proporcionam um monitoramento e controle aprimorados para o administrador ou coordenador do SISGEP-COMIC. Ressalta-se que as funcionalidades dos *widgets* já estão no sistema em produção e disponíveis ao administrador e aos professores-pesquisadores.

V. Conclusões, Limitações e Trabalhos Futuros

Neste trabalho, discutiu-se a área de Mineração de Dados, uma subárea da Ciência da Computação, destacando sua utilidade no auxílio à tomada de decisões e as ferramentas que possibilitam sua aplicação. Nesse contexto, foi proposto, modelado e implementado um módulo de gestão de relatórios baseado em reconhecimento de padrões, utilizando a biblioteca pandas, com o objetivo de auxiliar os administradores do sistema no seus processos de decisão.

O módulo foi integrado, ainda que em caráter de protótipo, ao SISGEP-COMIC. Através da utilização de pandas, foram criados *dataframes* para armazenamento e manipulação dos dados das tabelas *Usuario* e *Submissao* da base de dados, permitindo a análise das informações. A modelagem funcional e estrutural do módulo, baseada no sistema SISGEP-COMIC, garantiu uma definição clara para sua implementação, facilitando a integração das novas funcionalidades com a plataforma existente.

Os objetivos do trabalho focaram na construção de um módulo para geração de relatórios que utilizasse reconhecimento de padrões no SISGEP-COMIC. Contudo, alguns objetivos específicos não foram plenamente atingidos, principalmente devido a limitações relacionadas à integração com o sistema e à realização de avaliações mais detalhadas. Entre os objetivos não alcançados estão: a integração completa do módulo ao SISGEP-COMIC (parcialmente realizada) e a aplicação de testes em ambiente de avaliação e validação, também comprometida pela ausência de integração total.

Apesar de o módulo ter sido disponibilizado como protótipo em um ambiente de testes, e não no servidor oficial do sistema, ele estabelece uma base promissora para melhorias futuras. Entre as possibilidades de expansão, destaca-se a integração de técnicas avançadas de *machine learning* para aumentar a precisão na detecção de padrões, ampliando assim sua aplicabilidade e eficácia.

Referências

[1] Roger S. Pressman. *Software Engineering: a Practitioner's Approach*. Ed. por McGraw-Hill Education. 7th. New York: McGraw-Hill Education, 2010.

[2] Sylvio André Garcia Vieira. "Identificacáo de padrões de expressão em doencas genéticas usando uma rede de integracáo de vias de manutencáo do Genoma, Angiogênese, Hipóxia e Vigilância Imunológica". Tese de dout. Santa Maria: Centro Universitário Franciscano, 2016.

[3] Patrick Tolio Müller e Alexandre De Oliveira Zamberlan. *Módulo Sisgep-Comic Gestão De Avaliação De Projetos*. Santa Maria, RS, Brasil. Disponível em https://tfgonline.lapinf.ufn.edu.br: Trabalho de Conclusão de Curso Sistemas De Informação - Universidade Franciscana (UFN), 2020.

[4] Emmanuel Passos | Ronaldo Goldschmidt. *Data Mining: Um Guia Prático*. Editora Elsevier, 2005.

[5] Daniel T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. Editora John Wiley e Sons, 2005.

[6] Matheus Batista Furlan e Alex Sandro Romeo de Souza Poletto. "Algoritmos e Técnicas para Mineração de Dados". Em: *Fundação Educacional do Município de Assis/SP* (2018).

[7] Manuel Filipe Santos e Carla Sousa Azevedo. *Preâmbulo [a]"Data mining: descoberta de conhecimento em bases de dados"*. FCA-Editora de informática, Lda, 2005.

[8] Oracle. *MySQL Connectors*. Abr de 2024.

[9] The pandas development team. *pandas-dev/pandas: Pandas*. Mai de 2024.

[10] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". Em: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[11] NumPy Developers. *NumPy documentation*. Mai de 2024.

[12] J. D. Hunter. "Matplotlib: A 2D graphics environment". Em: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[13] Michael Waskom. *seaborn: statistical data visualization*. Mai de 2024.

[14] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[15] Pang-Ning Tan, Michael Steinbach e Vipin Kumar. *Introdução ao datamining: mineração de dados*. São Paulo: Ciência Moderna, 2009.

[16] Robson Cezário e Alexandre Zamberlan. *Algoritmos De Mineração De Dados Em Sistema De Monitoramento De Diabetes*. Santa Maria, RS, Brasil. Disponível em https://tfgonline.lapinf.ufn.edu.br: Trabalho de Conclusão de Curso Sistemas De Informação - Universidade Franciscana (UFN), 2021.

[17] Denis Vicentainer, Marcos Mattedi e Bruno Mello. "Aplicação das bibliotecas Python para tratamento de dados em tempo real: A análise dos dados de isolamento social em Santa Catarina". Em: *Metodologias e Aprendizado* 3 (out. de 2020), pp. 206–217. DOI: 10.21166/metapre.v3i0.1392. URL: https://publicacoes.ifc.edu.br/index.php/metapre/article/view/1392.

[18] Dirce Backes and Márian Oleques Pires and Vinicius Moura and Sylvio Garcia. *PRENATAL MANAGEMENT: Technological Product of the Masters Degree in Maternal and Child Health at UFN (LASMI)*. Abril de 2024.

[19] Django Software Foundation. *Django makes it easier to build better web apps more quickly and with less code.* Django. Nov. de 2024. URL: https://www.djangoproject.com/.

[20] Paulo Silveira et al. *Introdução à arquitetura e Design de Software*. Elsevier Editora, 2012.

[21] Porto. *Infopédia*. Agosto de 2020. URL: https://www.infopedia.pt/dicionarios/lingua-portuguesa/refatorar.