UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

IGOR MACIEL DE SOUSA

DESENVOLVIMENTO DE UM SISTEMA DE APOIO AUTOMATIZADO PARA GESTÃO EFICIENTE DE AUXÍLIOS ESTUDANTIS NA UFC – CAMPUS SOBRAL

SOBRAL

2025

IGOR MACIEL DE SOUSA

DESENVOLVIMENTO DE UM SISTEMA DE APOIO AUTOMATIZADO PARA GESTÃO EFICIENTE DE AUXÍLIOS ESTUDANTIS NA UFC – CAMPUS SOBRAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientadora: Profa. Dra. Jermana Lopes de Moraes

Coorientadora: Me. Rayane Alves Lacerda

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

D32d de Sousa, Igor Maciel.
DESENVOLVIMENTO DE UM SISTEMA DE APOIO AUTOMATIZADO PARA GESTÃO EFICIENTE DE AUXÍLIOS ESTUDANTIS NA UFC – CAMPUS SOBRAL / Igor Maciel de Sousa. – 2025.
47 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Profa. Dra. Jermana Lopes de Moraes.
Coorientação: Profa. Ma. Rayane Alves Lacerda.

1. Assistência Estudantil. 2. Aplicação Desktop. 3. Automação de processos. I. Título.
CDD 621.39

IGOR MACIEL DE SOUSA

DESENVOLVIMENTO DE UM SISTEMA DE APOIO AUTOMATIZADO PARA GESTÃO EFICIENTE DE AUXÍLIOS ESTUDANTIS NA UFC – CAMPUS SOBRAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 06/03/2025.

BANCA EXAMINADORA

Profa. Dra. Jermana Lopes de Moraes (Orientadora)
Universidade Federal do Ceará (UFC)

Me. Rayane Alves Lacerda (Coorientadora)
Universidade Federal do Ceará (UFC)

Prof.Thiago Iachiley Araujo de Souza
Universidade Federal do Ceará (UFC)

Prof.Fernando Rodrigues de Almeida Junior
Universidade Federal do Ceará (UFC)

"Nada é tão poderoso no mundo como uma ideia
cuja oportunidade chegou."

(Victor Hugo)

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, aos meus pais, Francisco e Solange, por toda a dedicação e apoio, que foram essenciais para que eu pudesse seguir adiante e concluir o curso.

À Profa. Dra. Jermana Lopes de Moraes, expresso minha gratidão pela orientação no meu trabalho de conclusão de curso e por idealizar este projeto. Agradeço pela oportunidade, atenção e paciência durante todo o desenvolvimento. Também sou grato pela ponte estabelecida com a assistência estudantil, que permitiu alinhar as necessidades reais dos assistentes sociais ao propósito do projeto, tornando-o mais aplicável e impactante para a universidade.

À Universidade Federal do Ceará (UFC) e à Coordenadoria de Assistência Estudantil (CASE) pelo suporte e pela oportunidade de desenvolver um sistema que contribui para a assistência estudantil. Um reconhecimento aos assistentes sociais, especialmente à Rayane, cujo conhecimento e feedback foram fundamentais para garantir que o sistema atendesse às necessidades reais da instituição.

RESUMO

Os processos administrativos da assistência estudantil têm um impacto significativo na vida dos estudantes, uma vez que as decisões tomadas influenciam diretamente o acesso a auxílios, como os programas de bolsas. Atualmente, grande parte desses processos ainda é realizada manualmente, o que os torna demorados e suscetíveis a erros, além de sobrecarregar os assistentes responsáveis pela análise dos dados. Para otimizar essa gestão, este trabalho propõe o desenvolvimento de um sistema desktop para automatizar e facilitar a administração dos auxílios estudantis na Universidade Federal do Ceará, campus Sobral. A escolha por um sistema desktop se deve à necessidade de restringir o acesso apenas a computadores autorizados na rede local, garantindo maior controle e segurança sobre os dados. Diferentemente de um sistema web, que pode ser acessado a partir do IP da máquina servidora e permite a manipulação das páginas por meio de ferramentas como o inspetor de elementos dos navegadores, o sistema desktop oferece uma camada adicional de proteção contra acessos não autorizados e modificações indevidas na interface. O sistema permite a importação de dados socioeconômicos dos estudantes a partir de arquivos CSV, preenchendo automaticamente uma interface interativa que possibilita edição individual e em massa, aplicação de filtros, ocultação de colunas, pesquisa dinâmica, paginação e exportação de dados em diferentes formatos. Além disso, o sistema gera relatórios detalhados e aplica critérios objetivos para a seleção dos estudantes com base em uma fórmula de soma de pesos definido pela PRAE, garantindo transparência e agilidade no processo de decisão. O desenvolvimento do sistema foi realizado utilizando a linguagem de programação Python, com PyQt como biblioteca principal para a interface gráfica. A manipulação e análise dos dados são feitas com as bibliotecas Pandas e NumPy, enquanto o armazenamento das informações é gerenciado por um banco de dados MySQL. Com essa abordagem, o sistema visa reduzir o tempo gasto na análise dos dados, minimizar erros e proporcionar uma gestão mais eficiente e acessível da assistência estudantil. Além disso, a centralização dos dados no banco de dados garante maior integridade, segurança e consistência das informações, permitindo que análises futuras sejam mais precisas e completas, facilitando a tomada de decisões estratégicas com base em dados consolidados.

Palavras-chave: Aplicação Desktop; PyQt; Pandas; Assistência Estudantil, Automação de processos

ABSTRACT

The administrative processes of student assistance have a significant impact on students' lives, as the decisions made directly influence access to benefits such as scholarship programs. Currently, a large portion of these processes is still carried out manually, making them time-consuming and prone to errors, in addition to overburdening the assistants responsible for data analysis. To optimize this management, this work proposes the development of a desktop system to automate and facilitate the administration of student aid at the Federal University of Ceará, Sobral campus. The choice of a desktop system is due to the need to restrict access only to authorized computers within the local network, ensuring greater control and security over the data. Unlike a web system, which can be accessed from the server machine's IP address and allows page manipulation through browser tools such as the element inspector, the desktop system provides an additional layer of protection against unauthorized access and improper modifications to the interface. The system enables the importation of students' socioeconomic data from CSV files, automatically populating an interactive interface that allows individual and bulk editing, filtering, column hiding, dynamic search, pagination, and data export in different formats. Additionally, the system generates detailed reports and applies objective criteria for student selection based on a weighted sum formula defined by PRAE, ensuring transparency and agility in the decision-making process. The system was developed using the Python programming language, with PyQt as the main library for the graphical interface. Data manipulation and analysis are handled using the Pandas and NumPy libraries, while information storage is managed by a MySQL database. With this approach, the system aims to reduce the time spent on data analysis, minimize errors, and provide a more efficient and accessible management of student assistance. Furthermore, centralizing the data in the database ensures greater integrity, security, and consistency of the information, allowing future analyses to be more accurate and comprehensive, facilitating strategic decision-making based on consolidated data.

Palavras-chave: Desktop Application; PyQt; Pandas; Student Assistance; Process automation

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CASE	Coordenadoria de Assistência Estudantil
CEU	*Clube de Estudantes Universitário*
CSV	*Comma Separated Values*
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DQL	Data Query Language
DTL	Data Transaction Language
EER	*Diagrama Entidade-Relacionamento*
JSON	*JavaScript Object Notation*
ORM	*Object Relational Mapping*
PRAE	Pró-Reitoria de Assistência Estudantil
QSS	*Qt Style Sheets*
SGBD	Sistema Gerenciador de Banco de Dados
SIGAA	Sistema Integrado de Gestão de Atividades Acadêmicas
SQL	Structured Query Language
UFC	Universidade Federal do Ceará
XSS	*Cross-Site Scripting*

SUMÁRIO

1 INTRODUÇÃO

A Lei nº 14.914, de 3 de julho de 2024 (BRASIL, 2024), instituiu a Política Nacional de Assistência Estudantil (PNAES), visando ampliar e consolidar as ações de permanência de estudantes de graduação em instituições públicas de ensino superior no Brasil.

Anteriormente, o Decreto nº 7.234 (BRASIL, 2010), de 19 de julho de 2010, já havia estabelecido o PNAES, proporcionando suporte para que estudantes de graduação em universidades públicas brasileiras tivessem condições de concluir o ensino superior. Com a promulgação da Lei nº 14.914/2024, o PNAES foi elevado ao status de lei, reforçando o compromisso do governo federal com a assistência estudantil e a permanência dos alunos no ensino superior público.

Nesse contexto, a Universidade Federal do Ceará (UFC) instituiu a Pró-Reitoria de Assistência Estudantil (PRAE), uma unidade administrativa dedicada ao bem-estar e à permanência dos estudantes. A PRAE coordena uma série de programas e serviços voltados à assistência estudantil, entre os quais se destacam: Ajuda de Custo, Auxílio Creche, Auxílio Emergencial, Auxílio Concludente – Caminhando Juntos, Auxílio Ingressante e Auxílio Moradia. Além disso, a UFC oferece bolsas como a Bolsa de Incentivo ao Desporto, a Bolsa de Iniciação Acadêmica (BIA) e a Bolsa Permanência, ampliando o suporte aos alunos em situação de vulnerabilidade socioeconômica (UFC-PRAE, 2025).

Visando uma estruturação organizacional e contato direto dos estudantes com assistentes sociais e psicólogos, cada Campus da Universidade Federal do Ceará (UFC) possui a sua Coordenadoria de Assistência Estudantil (CASE), a qual precisa executar a política e administrar a permanência, acompanhamento e desligamento dos discentes, além de selecionar os discentes com condições socioeconômicas mais precárias para recebimento dos auxílios supracitados. A seleção dos discentes baseada em muitos critérios socioeconômicos bem estabelecidos ocasiona uma demanda significativa para a equipe da Assistência Estudantil, principalmente em alguns campi do interior do Ceará, como é o caso de Sobral, que as solicitações efetuadas pelos discentes (aproximadamente 400 por semestre) são analisadas manualmente. Tal fato pode ser agravado se considerarmos que as demandas pela bolsa de auxílio estudantil vêm crescendo de modo vertiginoso a cada ano, como em 2022 foram 372 e em 2023, 647 (UFC-SOBRAL, 2025), implicando em decisões cada vez mais difíceis quanto à definição dos discentes que serão contemplados.

Uma maneira eficaz de mitigar a problemática da análise manual na assistência

estudantil, é utilizar um sistema computacional para automatizar e otimizar o processo de gestão dos auxílios. Com a digitalização dos dados e a aplicação de critérios objetivos pré-definidos, é possível reduzir inconsistências nas decisões, diminuir o tempo de análise das solicitações dos discentes, reduzir um trabalho manual dos colaboradores da CASE Sobral que podem canalizar esse esforço na promoção de outras atividades, garantir maior transparência e padronizar a avaliação dos estudantes solicitantes. Dessa forma, a CASE Sobral poderia aprimorar sua análise e atendimento, agilizando a tomada de decisão e assegurando uma distribuição mais eficiente dos auxílios.

Uma abordagem viável para automatizar e otimizar esse processo é a criação de um sistema de gestão de auxílios estudantis que importe e processe automaticamente os dados socioeconômicos dos estudantes a partir de arquivos *Comma Separated Values* (CSV). Esse sistema permitirá a análise dos critérios de elegibilidade com base em uma fórmula de soma de pesos, considerando fatores como renda, despesas e bens familiares. Além disso, oferecerá ferramentas avançadas, como edição individual e em massa, filtros, pesquisa dinâmica, ocultação de colunas, paginação e exportação de relatórios, proporcionando maior agilidade e precisão no gerenciamento dos dados.

Para a implementação desse processo, será desenvolvido um software desktop utilizando a linguagem de programação Python. A escolha por um sistema desktop se deve à necessidade de restringir o acesso apenas a computadores autorizados na rede local, garantindo maior controle e segurança sobre os dados. Diferentemente de um sistema web, que pode ser acessado a partir do IP da máquina servidora e permite a manipulação das páginas por meio de ferramentas como o inspetor de elementos dos navegadores, o sistema desktop oferece uma camada adicional de proteção contra acessos não autorizados e modificações indevidas na interface. A interface gráfica será construída com a biblioteca PyQt, garantindo uma experiência interativa e intuitiva para as assistentes sociais da CASE Sobral. Os dados inseridos e processados pelo sistema serão armazenados em um banco de dados local, utilizando o Sistema Gerenciador de Banco de Dados (SGBD) MySQL para garantir a integridade e acessibilidade das informações.

1.1 Objetivos

1.1.1 Objetivos Gerais

Este trabalho tem como objetivo desenvolver um software desktop para otimizar e automatizar o processo de seletivo dos Auxílios e Bolsas vinculados a Assistência Estudantil do campus Sobral. A aplicação visa agilizar a tomada de decisão, garantindo maior precisão e transparência na análise das características socioeconômicas dos discentes, padronizando os critérios de seleção e facilitando a gestão dos auxílios estudantis.

1.1.2 Objetivos Específicos

- Analisar o ambiente da assistência estudantil, mapeando os principais processos de tomada de decisão, identificando as partes envolvidas e compreendendo os critérios utilizados na concessão de auxílios;
- Desenvolver uma interface gráfica intuitiva e eficiente, permitindo a importação de arquivos CSV para coletar e organizar os dados socioeconômicos dos estudantes de forma estruturada e acessível;
- Implementar e validar a lógica do sistema de avaliação, utilizando uma fórmula de soma de pesos baseada em critérios como renda, despesas e outros fatores socioeconômicos relevantes para a elegibilidade dos auxílios;
- Disponibilizar ferramentas de análise e manipulação de dados, como edição individual e em massa, filtros, pesquisa dinâmica, ocultação de colunas, paginação e exportação de relatórios, garantindo maior agilidade e precisão na gestão dos auxílios;

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica deste trabalho, abordando os principais conceitos e tecnologias que servem de base para o desenvolvimento do sistema proposto. Serão discutidos os subtemas relacionados à PRAE e CASE, ao desenvolvimento de interfaces gráficas, à utilização de banco de dados e persistência de dados, ao processamento e análise de dados, à segurança e autenticação, à comunicação assíncrona e mensageria, e, por fim, à containerização e implantação. Esses subtemas fornecerão a base necessária para a compreensão da solução desenvolvida.

2.1 PRAE e CASE

A assistência estudantil na Universidade Federal do Ceará (UFC) teve início no primeiro mandato do Reitor Martins Filho, com a criação do *Clube de Estudantes Universitário* (CEU), que centralizava atividades assistenciais, culturais e esportivas. O CEU abrigou o Restaurante Universitário, inaugurado em 1957, e a primeira Residência Universitária. Em 1966, foi criada a Vice-Reitoria de Assuntos Estudantis para administrar a assistência estudantil, sendo transformada, em 1969, na Pró-Reitoria de Assuntos Estudantis. Em 2023, a pró-reitoria foi renomeada para Pró-Reitoria de Assistência Estudantil (PRAE), reafirmando seu compromisso com a permanência e o bem-estar dos estudantes (PRAE, 2025).

Atualmente, a Pró-Reitoria de Assistência Estudantil (PRAE) é responsável pela gestão das políticas voltadas à promoção e apoio aos estudantes de graduação, fortalecendo a cidadania dentro dos diversos segmentos acadêmicos da comunidade universitária. Seu trabalho visa incentivar e acompanhar o desenvolvimento dos alunos ao longo de sua trajetória acadêmica, promovendo ações eficazes nas áreas social, técnico-científica, cultural, política e esportiva (PRAE, 2025).

A Coordenadoria de Assistência Estudantil (CASE) é um órgão vinculado ao Pró-Reitoria de Assistência Estudantil (PRAE), com o objetivo de oferecer apoio e assistência aos estudantes da universidade, principalmente aos que se encontram em situação de vulnerabilidade socioeconômica. Ela desempenha um papel importante na promoção da permanência e sucesso acadêmico dos alunos de graduação, oferecendo serviços e benefícios como bolsas, auxílios financeiros, moradia estudantil, alimentação e apoio psicológico (PRAE, 2025).

Atualmente, a equipe da CASE é bastante reduzida, composta por apenas duas

assistentes sociais, dois técnicos administrativos e um psicólogo, que, no momento, está afastado por questões de saúde. Esse quadro limitado impacta diretamente a capacidade de atendimento e a execução das políticas de assistência estudantil, tornando essencial a ampliação da equipe para melhor atender às demandas da comunidade acadêmica.

2.2 Desenvolvimento de Interfaces Gráficas

Nas subseções seguintes serão abordadas as definições das tecnologias utilizadas no desenvolvimento do sistema proposto, o qual foi projetado para oferecer uma interação intuitiva e eficiente, permitindo a visualização, edição e organização dos dados de forma acessível.

2.2.1 PyQt

O PyQt é um conjunto de bibliotecas Python que permite a criação de aplicações gráficas desktop com interfaces de usuário otimizadas. Ele combina a linguagem de programação Python com o framework Qt, oferecendo uma maneira poderosa e flexível de desenvolver aplicações multiplataforma (WILLMAN, 2020). Entre suas características, destaca-se a multiplataforma, com suporte a Windows, macOS, Linux, entre outros sistemas operacionais. Oferece também widgets personalizáveis, com uma ampla variedade de elementos pré-construídos que podem ser ajustados conforme as necessidades da aplicação. A conectividade é uma característica importante, permitindo a conexão de sinais e slots, o que facilita a interação entre os elementos da interface e a lógica da aplicação. Além disso, o PyQt suporta recursos avançados como formatação de texto, desenho 2D e gráficos. A integração com o Qt Designer também é uma vantagem, possibilitando a criação de interfaces gráficas de forma visual por meio dessa ferramenta e a posterior integração com o código Python.

2.2.2 Jinja2

O Jinja2 é um mecanismo de template para Python, amplamente utilizado para a geração dinâmica de páginas HTML em aplicações web. Ele permite a separação entre a lógica da aplicação e a apresentação, facilitando a criação de templates reutilizáveis. O Jinja2 é inspirado no mecanismo de templates do Django e fornece recursos como herança de templates, filtros, macros e controle de fluxo (como loops e condicionais) (RONACHER, 2008). Entre suas características, destaca-se a sintaxe semelhante ao Python, que facilita o aprendizado e uso.

Além disso, oferece suporte à herança de templates, permitindo o reuso de código, e expressões dinâmicas, possibilitando a manipulação avançada de dados dentro dos templates. O Jinja2 também conta com segurança integrada, protegendo contra ataques como *Cross-Site Scripting* (XSS), e apresenta rápida execução, pois é baseado na biblioteca C Markupsafe, que proporciona melhor desempenho.

2.3 Banco de Dados e Persistência de Dados

Essa etapa trata das tecnologias aplicadas no banco de dados e persistência de dados. O sistema utiliza um banco de dados relacional para armazenar de forma segura e organizada todas as informações, garantindo a integridade e a consistência dos dados. A persistência é assegurada por meio de técnicas que permitem a atualização, recuperação e armazenamento eficientes das informações, mesmo após o fechamento ou reinício do sistema.

2.3.1 Modelo Relacional

Apesar de banco de dados ser um termo técnico, a maioria das pessoas nos dias de hoje tem contato direto com ele. De fato, grande parte da população atualmente tem acesso a equipamentos, cuja função (principal ou secundária) é o armazenamento de informações. Por exemplo no caso do aparelho celular, possui uma agenda, na qual podemos gravar nomes e telefones para, em um segundo momento, acessá-los. Uma lista telefônica impressa também é um exemplo válido disso, pois nela são relatados todos os nomes, endereços e números de telefone das empresas e dos moradores da sua cidade e, eventualmente, dos arredores (CARVALHO, 2015).

Independentemente do aplicativo que se deseja usar para o armazenamento e manipulação das informações, todos os bancos de dados relacionais são constituídos por elementos básicos: campos, colunas, linhas ou tuplas e tabelas. Campos são os espaços reservados para inserção de um determinado dado; as colunas são os registros de um determinado campo; as tuplas são as linhas de registros de um conjunto de campos; e as tabelas são os conjuntos de linhas, campos e colunas. Para visualizar melhor essa definição, é mostrado na Figura 1 a estrutura de um banco de dados relacional (CARVALHO, 2015).

2.3.2 SQL

Structured Query Language (SQL) é a linguagem padrão utilizada pelos bancos de dados relacionais devido à sua simplicidade e versatilidade de uso. Ela é fundamental para interagir com bancos de dados relacionais em uma variedade de contextos de desenvolvimento de software, oferecendo um conjunto de comandos que permitem criar, modificar, consultar e gerenciar dados armazenados (CARVALHO, 2015). SQL é composta por cinco categorias principais de comandos.

A primeira categoria é a Data Manipulation Language (DML), que inclui comandos utilizados para recuperar, inserir e modificar registros no banco de dados. Exemplos desses comandos são INSERT, DELETE e UPDATE, que permitem inserir novos dados, excluir dados existentes e atualizar registros.

A segunda categoria é a Data Definition Language (DDL), responsável pela criação, alteração e exclusão de objetos no banco de dados, como tabelas, índices e visões. Comandos típicos dessa categoria incluem CREATE TABLE, CREATE INDEX, ALTER TABLE, DROP TABLE, DROP VIEW e DROP INDEX.

Em seguida, temos a Data Control Language (DCL), que gerencia o controle de segurança e acesso dos usuários, assim como o controle de sessões. Os comandos mais comuns dessa categoria são DENY, GRANT e REVOKE, que permitem conceder ou revogar permissões de acesso aos usuários.

A Data Query Language (DQL) é a quarta categoria, e seu principal objetivo é consultar dados no banco de dados. Comandos como SELECT, JOIN, GROUP BY e ORDER BY permitem recuperar informações específicas de uma ou mais tabelas, organizando e filtrando os resultados conforme necessário.

Por fim, temos a Data Transaction Language (DTL), que gerencia as transações no banco de dados. Comandos como BEGIN TRANSACTION, COMMIT e ROLLBACK são usados para iniciar, confirmar ou desfazer transações, garantindo a integridade dos dados durante a execução de múltiplas operações.

2.3.3 MySQL

O MySQL é um servidor e gerenciador de banco de dados SGBD relacional, de licença dupla(sendo uma delas de software livre), projetado inicialmente para trabalhar com

aplicações de pequeno e médio portes, mas hoje atendendo a aplicações de grande porte. Além de ser extremamente rápido, pelo fato de armazenar os dados em tabelas no modo ISAM (código de baixo nível), o MySQL é altamente confiável. Atualmente, tem sido o banco de dados open-source mais utilizado em aplicações desktop (MILANI, 2007). Entre suas características, destaca-se o fato de ser de código aberto e multiplataforma. O MySQL utiliza o modelo relacional, organizando os dados em tabelas com colunas e linhas, e suporta consultas rápidas, sendo capaz de lidar com grandes volumes de dados. Ele também permite escolher diferentes mecanismos de armazenamento e oferece conexões criptografadas com suporte a SSL/TLS para maior proteção.

2.3.4 PyMySQL

PyMySQL é uma biblioteca em Python utilizada para conectar aplicações ao banco de dados MySQL ou MariaDB através do protocolo MySQL Client/Server. Ele funciona como um conector que permite executar comandos SQL como SELECT, INSERT, UPDATE, DELETE diretamente a partir de um script Python, além de manipular tabelas e transações (HUNT, 2023).

2.3.5 SQLAlchemy

SQLAlchemy é uma biblioteca de Python que facilita o uso de bancos de dados relacionais. Ela fornece uma interface poderosa e flexível para interagir com bancos de dados, permitindo trabalhar com consultas SQL diretas. Com SQLAlchemy, é possível modelar tabelas e suas relações como classes Python, simplificando a manipulação de dados no código (COPELAND, 2008). A biblioteca também oferece um *Object Relational Mapping* (ORM), que mapeia tabelas e colunas de bancos de dados para classes e atributos Python. SQLAlchemy é amplamente compatível, com suporte a diversos bancos de dados, como MySQL, PostgreSQL, SQLite, Oracle, Microsoft SQL Server, entre outros. Além disso, é adequada para projetos de diferentes escalas, oferecendo escalabilidade e desempenho, com controle preciso sobre transações e conexões.

2.4 Processamento e Análise de Dados

Nas próximas subseções serão apresentados os conceitos das tecnologias empregadas no processamento e análise de dados. O sistema desenvolvido permite a importação, manipulação,

edição e exportação dos dados, assegurando uma gestão eficiente, organizada e precisa das informações.

2.4.1 Pandas

A biblioteca Pandas é uma das ferramentas mais populares em Python para manipulação e análise de dados. Ela fornece estruturas de dados de alto nível, como **DataFrame** e **Series**, e uma variedade de funções e métodos para facilitar a importação, limpeza, transformação e análise de dados (MCKINNEY *et al.*, 2011). Entre suas principais características, destaca-se o DataFrame, que é uma estrutura de dados tabular bidimensional com rótulos de linha e coluna, similar a uma tabela de banco de dados SQL ou uma planilha do Excel. A Series é um objeto unidimensional semelhante a uma matriz ou lista, capaz de armazenar dados de diferentes tipos, como números inteiros, strings e objetos Python. Pandas também facilita a importação e exportação de dados em diversos formatos, como CSV, Excel, SQL, entre outros. A biblioteca oferece funcionalidades para a limpeza e transformação de dados, incluindo o tratamento de valores ausentes e a manipulação de strings. Além disso, possibilita a filtragem, seleção e indexação de dados, assim como a agregação e o agrupamento de dados. Ela também permite o pivotamento e a remodelação de dados, além de fornecer estatísticas descritivas e cálculos matemáticos. A visualização de dados é simplificada por meio de integração com bibliotecas como matplotlib e seaborn.

2.4.2 NumPy

O NumPy (Numerical Python) é uma biblioteca fundamental para computação científica em Python. Ele fornece suporte para arrays multidimensionais, além de oferecer funções matemáticas avançadas, como operações matriciais, estatísticas e transformadas rápidas de Fourier (OLIPHANT *et al.*, 2006). Entre suas características, destaca-se os arrays multidimensionais, uma estrutura de dados otimizada (ndarray) que é mais eficiente que as listas tradicionais do Python. O NumPy também permite operações vetorizadas, facilitando cálculos matemáticos rápidos sem a necessidade de loops explícitos. A biblioteca possui integração com outras ferramentas, como Pandas, SciPy, Matplotlib e TensorFlow. Sua performance é otimizada, pois é escrita em C, o que a torna muito mais rápida que as listas comuns do Python.

2.4.3 Plotly Express

Plotly Express é uma biblioteca de visualização de dados em Python de alto nível, desenvolvida como um módulo wrapper para a biblioteca Plotly. Ela oferece uma interface simples e intuitiva para criar uma ampla variedade de visualizações, incluindo gráficos, mapas, objetos gráficos, layouts e figuras. Plotly Express se destaca por sua facilidade de uso e por gerar visualizações interativas, estilizáveis e informativas, que são ideais para explorar e comunicar dados de forma clara e atrativa. Além disso, os gráficos criados com Plotly Express possuem interatividade nativa, como a exibição de texto flutuante personalizável ao passar o mouse sobre os elementos, facilitando a análise e interpretação dos dados (DABBAS, 2021).

2.5 Segurança e Autenticação

As definições das tecnologias implementadas para garantir a segurança e autenticação serão conceituadas nesta Seção. O sistema adota mecanismos robustos para proteger os dados sensíveis, como senhas e informações pessoais, utilizando técnicas de criptografia e autenticação de usuários. A segurança é assegurada por meio de processos que garantem o acesso autorizado e a proteção contra possíveis ameaças, mantendo a confidencialidade e a integridade dos dados em todas as operações.

2.5.1 Algoritmo bcrypt

O bcrypt é uma biblioteca e um algoritmo amplamente usado para criptografar senhas de forma segura antes de armazená-las em um banco de dados. Ele é projetado especificamente para proteger senhas contra ataques de força bruta ou outras técnicas maliciosas, tornando-as difíceis de quebrar, mesmo com hardware moderno (SRIRAMYA; KARTHIKA, 2015). O algoritmo bcrypt é baseado no Blowfish, um cipher que utiliza o recurso de hashing adaptativo. Esse hashing adaptativo permite configurar a "dureza"(ou custo computacional) do hash, o que significa que o tempo de processamento pode ser aumentado à medida que o hardware se torna mais poderoso. O bcrypt também adiciona automaticamente um salt único (valor aleatório) ao hash da senha, o que impede ataques de tabela de hash, como as rainbow tables. Além disso, ele proporciona proteção contra ataques de força bruta, tornando o cálculo do hash intencionalmente mais lento e dificultando significativamente ataques de tentativa e erro. O bcrypt é seguro contra alterações nos hashes, pois permite a regeneração dos hashes se o custo (work factor) ou o

algoritmo precisarem ser alterados no futuro, sem comprometer as senhas originais.

2.5.2 Secrets

Secrets é uma biblioteca nativa do Python, introduzida no Python 3.6, destinada à geração de números aleatórios seguros para gerenciamento de senhas, tokens e outras chaves secretas que exigem um nível de segurança elevado.

Ela oferece funções que utilizam fontes de aleatoriedade criptograficamente seguras, ou seja, geradas de maneira que não possam ser facilmente previstas ou reproduzidas, o que a torna mais segura do que a função random para tarefas que envolvem segurança, como geração de senhas ou tokens de autenticação (FOUNDATION, 2025).

2.6 Comunicação Assíncrona e Mensageria

Aqui serão abordadas as tecnologias de comunicação assíncrona e mensageria. O sistema utiliza mensageria assíncrona para garantir a troca eficiente de informações entre diferentes componentes, sem bloquear o fluxo principal da aplicação. Isso permite o processamento de tarefas de forma independente e escalável, melhorando o desempenho e a resposta do sistema.

2.6.1 Apache Kafka

O Apache Kafka é uma plataforma distribuída de mensageria e streaming de eventos usada para construir sistemas de processamento de dados em tempo real e arquiteturas de microserviços. Seu principal objetivo é gerenciar fluxos de dados de alta escala e garantir a comunicação entre sistemas diferentes de maneira eficiente, escalável e tolerante a falhas (GARG, 2013).

Os principais componentes dessa plataforma distribuída são: o producer (produtor), os quais ão os componentes responsáveis por enviar mensagens (ou eventos) para o Kafka. Um produtor pode ser um aplicativo, um serviço ou um sistema que cria e envia dados. Essas mensagens são enviadas para tópicos (topics) no Kafka; o consumer (consumidor) que são os componentes responsáveis por consumir as mensagens enviadas para o Kafka. Eles podem processar esses dados de várias maneiras, como atualizando bancos de dados, enviando para outras filas de processamento, ou até mesmo para visualizações; o broker, o qual é o servidor que compõem um cluster do Kafka, sendo responsável por armazenar as mensagens de forma

persistente e fornecer mecanismos de leitura/escrita para os produtores e consumidores; o topic, o qual é uma "categoria"ou "canal"para onde as mensagens são enviadas pelos produtores e de onde os consumidores leem. Um tópico pode ser particionado para distribuir a carga entre vários brokers; e o zooKeeper que é utilizado para gerenciar sua configuração e fornecer alta disponibilidade e coordenação entre os brokers. O ZooKeeper mantém o estado e as informações de configuração do Kafka, como a qual broker contém cada partição, e cuida da manutenção do cluster.

2.7 Containerização e Implantação

As tecnologias utilizadas na containerização e implantação serão exploradas nesta seção. O sistema proposto é containerizado para garantir um ambiente isolado e consistente, facilitando sua execução em diferentes plataformas sem dependências conflitantes.

2.7.1 Docker

Docker é uma plataforma de código aberto para automação do desenvolvimento, empacotamento, distribuição e execução de aplicações dentro de containers. A principal proposta do Docker é garantir a portabilidade e isolamento das aplicações, permitindo que elas sejam executadas de maneira consistente em qualquer ambiente, desde o desenvolvimento até a produção (MERKEL *et al.*, 2014). Entre suas características, destaca-se a criação de imagens, onde, a partir de um Dockerfile, você pode criar uma imagem Docker utilizando o comando docker build. Com essa imagem, é possível executar um container usando o comando docker run, que instancia e executa a imagem, isolando o processo da aplicação. Para ambientes mais complexos, como os de microserviços, o Docker Compose pode ser utilizado para orquestrar múltiplos containers, facilitando o gerenciamento de instâncias e serviços. As imagens também podem ser distribuídas, enviando-as para um repositório como o Docker Hub ou outro registro privado, o que permite compartilhar e distribuir a aplicação de maneira simples e eficiente.

3 METODOLOGIA

Este capítulo apresenta a metodologia adotada para o desenvolvimento do sistema, detalhando as etapas seguidas desde o levantamento de requisitos até a implementação e validação da solução. Serão discutidos os subtemas relacionados ao levantamento de requisitos, à definição das funcionalidades do sistema e aos processos de desenvolvimento e implementação. Essa abordagem visa assegurar que o projeto atenda às necessidades da assistência estudantil de forma estruturada e sistemática.

3.1 Levantamento de Requisitos

O levantamento de requisitos desempenha um papel importante na construção de um sistema de informação, pois é o início para toda a atividade de desenvolvimento de software (MENDONÇA, 2014). O levantamento foi realizado com o objetivo de identificar as necessidades dos assistentes sociais da UFC Sobral no processo de administração dos auxílios estudantis. Esse processo envolveu a análise dos fluxos de trabalho atuais, a identificação das dificuldades enfrentadas no método manual e a definição das funcionalidades essenciais para o sistema proposto.

3.1.1 Entendimento do Processo Atual

Atualmente, o processo de avaliação socioeconômica dos estudantes para a concessão de auxílios na UFC Sobral é realizado de forma manual e envolve múltiplas etapas. Primeiramente, os assistentes recebem um arquivo CSV contendo 30 colunas com os dados preenchidos pelos estudantes em seus formulários. Em seguida, essas informações são transferidas para uma planilha do Google Planilhas, onde são aplicadas fórmulas para cálculo da pontuação de cada estudante com base em critérios socioeconômicos predefinidos. Após a realização dos cálculos e ajustes necessários, os assistentes geram manualmente um novo arquivo CSV, filtrando apenas as colunas essenciais para exibir os resultados finais, diferenciando os estudantes elegíveis e não elegíveis aos auxílios.

3.1.2 Requisitos Funcionais

Os requisitos funcionais descrevem as principais funcionalidades que o sistema deve implementar para atender às necessidades dos usuários. Para o desenvolvimento do sistema foram observados os seguintes requisitos funcionais:

- **Importação de Dados:** O sistema deve permitir a importação de arquivos CSV contendo informações dos estudantes;
- **Mapeamento Automático:** O sistema deve mapear automaticamente as colunas do CSV para a tableView, evitando ajustes manuais;
- **Edição de Dados:** O sistema deve permitir a edição de informações diretamente na tableView e na tela de edição individual;
- **Controle de Usuários** O sistema deve permitir a criação e edição de usuários, além da definição de permissões de acesso;
- **Cálculo Automático:** O sistema deve calcular automaticamente os valores dos campos Total e Total Moradia ao modificar os dados correspondentes;
- **Atualização em Tempo Real:** As edições realizadas na tableView devem ser sincronizadas entre os usuários conectados ao sistema;
- **Exportação de Dados:** O sistema deve permitir a exportação de dados em arquivos CSV organizados por ordem alfabética contendo informações sobre a pontuação.

3.1.3 Requisitos Não Funcionais

Os requisitos não funcionais especificam as restrições técnicas e critérios de qualidade do sistema. Para o desenvolvimento do sistema foram observados os seguintes requisitos não funcionais:

- **Segurança:** O sistema deve armazenar as senhas dos usuários de forma segura, utilizando técnicas de hash e salt;
- **Controle de Acesso:** O sistema deve restringir funcionalidades conforme o nível de permissão do usuário;
- **Registro de Atividades:** Todas as operações de edição e exclusão devem ser registradas em logs para auditoria;
- **Interface Amigável:** A interface deve ser intuitiva e organizada, destacando informações importantes e garantindo fácil navegação;

- **Compatibilidade:** O sistema deve ser compatível com os sistemas operacionais Windows e Linux;
- **Escalabilidade:** O sistema deve permitir a adição de novos usuários e registros sem perda significativa de desempenho.

3.2 Definição das Funcionalidades do Sistema

O sistema foi projetado para automatizar e otimizar a gestão dos auxílios estudantis, reduzindo a necessidade de manipulação manual de planilhas e aumentando a eficiência no processamento dos dados. As principais funcionalidades implementadas incluem:

3.2.1 Login e Autenticação

Ao iniciar o sistema, o usuário é direcionado para a tela de Login, onde deve inserir seu nome de usuário e senha previamente cadastrados. Caso o usuário esqueça a senha, o sistema oferece a opção de recuperação por meio de um código enviado ao e-mail cadastrado. O usuário também pode alterar a senha diretamente no sistema. A senha deve atender a requisitos de segurança, como conter letras, números, caracteres especiais e ter no mínimo 8 dígitos. As senhas são armazenadas no banco de dados de forma criptografada, garantindo a segurança das informações.

3.2.2 Importação e Mapeamento de Dados

Após o login, o usuário é direcionado para a tela de importação, que exibe os seguintes widgets e suas respectivas funções:

- **Configurações:** Apenas usuários com permissão de Administrador podem acessar as configurações do sistema.
- **Gestão de Usuários:** O sistema permite a criação e edição de usuários, com definição de permissões de acesso (ex: Administrador, Assistente).
- **Alteração do Salário Mínimo:** O sistema permite ajustar o valor do salário mínimo, que é utilizado na fórmula de pontuação dos estudantes. Essa funcionalidade facilita a atualização do cálculo sem a necessidade de modificar manualmente a fórmula em planilhas externas.
- **Configuração de Seletores Fixos:** O sistema permite alterar as opções das colunas que

são seletores fixos (ex: sexo, raça).

- **Relatórios:** Geração de relatórios e visualização de gráficos;
- **Deslogar:** Encerrar a sessão do usuário;
- **Nova Análise:** Permite criar uma nova análise e importar um novo arquivo CSV. O sistema permite a importação de um arquivo CSV contendo as informações dos estudantes. As colunas do CSV são mapeadas automaticamente para as colunas da TableView. Se o nome de uma coluna no CSV for diferente do nome correspondente na TableView, o sistema exibe uma marcação em vermelho para alertar o usuário, permitindo o mapeamento manual.
- **Abrir Análise Existente:** Se já existirem análises cadastradas, elas são exibidas em uma lista. Deve-se clicar duas vezes em uma análise para direcionar o usuário para a janela principal do sistema, carregando os dados na tableView correspondente. Usuários com permissão de Administrador podem acessar análises de semestres anteriores. Para outros usuários, as análises anteriores estão desativadas para abertura.

3.2.3 Tela Principal e Manipulação de Dados

Após importar o arquivo CSV ou abrir uma análise existente, o usuário é direcionado para a Tela Principal, onde é exibida a TableView com os dados dos estudantes. Essa tela é o núcleo do sistema, permitindo a visualização, edição e manipulação dos dados de forma eficiente. A TableView é composta por diversos widgets e funcionalidades que facilitam a interação do usuário com as informações. Abaixo estão as principais funcionalidades disponíveis:

- **Paginação:** botões para avançar e voltar páginas, melhorando a performance e a usabilidade com grandes volumes de dados.
- **Adição e Remoção de Linhas:** permissão para adicionar novas linhas e apagar linhas existentes.
- **Atualizar Tabela:** atualiza e Restaura os dados exibidos na TableView para a primeira página.
- **Ocultar Colunas:** ocultação de colunas para melhorar a visualização.
- **Filtrar Dados:** filtros com lógica "E"ou "OU", facilitando a busca por informações específicas.
- **Editar Linha:** Abre uma nova tela para edição detalhada dos dados do estudante, com campos e abas organizados. A tela inclui botões para aplicar ou cancelar as alterações e

uma label que exibe a pontuação do estudante de forma clara.

- **Pesquisa Dinâmica:** permite buscar estudantes por nome de forma rápida e eficiente.
- **Exportação de Dados:** exportação dos dados por tipo de auxílio, gerando um arquivo CSV em ordem alfabética, incluindo a pontuação dos estudantes.

3.2.4 Edição Sincronizada em Tempo Real

Uma das funcionalidades mais inovadoras do sistema é a edição sincronizada em tempo real. Quando um assistente edita uma célula na TableView, a alteração é imediatamente refletida em todos os terminais conectados ao sistema. Para facilitar a identificação de edições simultâneas, as células em edição são destacadas com cores diferentes. Além disso, ao selecionar uma célula, a seleção é espelhada nos outros terminais, permitindo que as assistentes visualizem as mesmas informações em tempo real. Essa funcionalidade elimina conflitos de edição e melhora a colaboração entre os usuários.

3.3 Processos de Desenvolvimento e Implementação

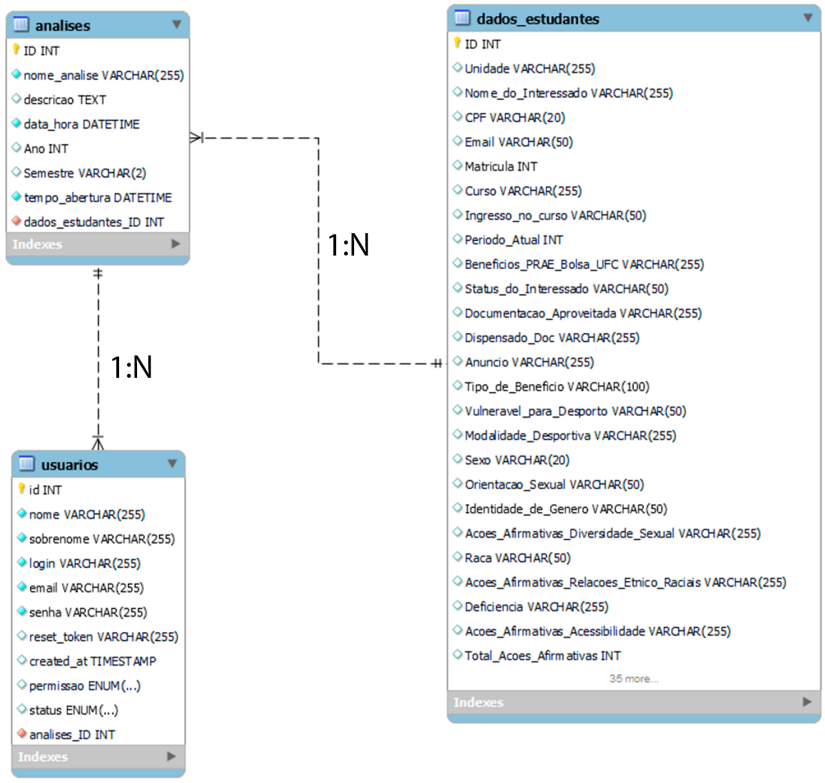
Nesta etapa, ocorre o trabalho de transformar todas as etapas anteriores em algo real e utilizável. Para isso, a implementação do sistema foi estruturada em três componentes principais, cada um com responsabilidades bem definidas: o Banco de Dados, o Front-end e o Back-end. O Banco de Dados é responsável pelo armazenamento e gerenciamento das informações dos estudantes, utilizando o MySQL como SGBD. O Front-end, desenvolvido com PyQt5, cuida da interface gráfica e da interação com o usuário, proporcionando uma experiência intuitiva e funcional. Por fim, o Back-end gerencia a lógica, o processamento de dados e a comunicação com o banco de dados, garantindo que as regras do sistema sejam aplicadas corretamente. O Front-end do sistema foi desenvolvido de forma paralela ao desenvolvimento do Back-end. Embora a divisão entre Front-end e Back-end seja mais comum em aplicações web, ela também foi aplicada neste sistema desktop para organizar o código e facilitar a manutenção.

3.3.1 Banco de Dados

Durante a criação do banco de dados, foi essencial planejar a forma como as diferentes partes do sistema se relacionariam. Após definir a estrutura das tabelas e seus vínculos, elaborou-se um *Diagrama Entidade-Relacionamento* (EER) para facilitar a compreensão das

conexões entre as entidades, conforme a Figura 1.

Figura 1 – Diagrama Entidade-Relacionamento(ER)



Fonte: Elaborado pelo Autor (2025)

O banco de dados nomeado como "caseleg"foi projetado para atender às necessidades do sistema de análise socioeconômica da UFC Sobral. Ele é composto por três tabelas principais: usuarios, analises e dados_estudantes.

A tabela usuarios armazena as informações dos usuários do sistema, incluindo credenciais de acesso, permissões e status. A tabela analises registra as análises socioeconômicas realizadas, com detalhes como nome, descrição e período de referência. Cada análise é vinculada a um usuário através da chave estrangeira usuario, garantindo que seja possível identificar o responsável pela sua criação. Assim, um usuário pode criar várias análises, estabelecendo um relacionamento 1:N entre usuarios e analises.

A tabela dados_estudantes contém os dados socioeconômicos dos estudantes, como informações pessoais, acadêmicas e de benefícios. Essa tabela está relacionada à tabela analises através da chave estrangeira ID_analise, garantindo que cada registro de estudante esteja vinculado a uma análise específica. Uma análise pode conter vários registros de estudantes, estabelecendo um relacionamento 1:N entre analises e dados_estudantes.

Essa estrutura permite uma gestão eficiente e organizada dos dados, além de facilitar a geração de relatórios e análises.

3.3.2 Back-end

O back-end do sistema é responsável por intermediar a comunicação entre a interface gráfica e o banco de dados, garantindo a integridade dos dados e a eficiência no processamento das operações. Sua estrutura foi planejada para permitir a manipulação de registros, a realização de cálculos automáticos e a atualização dinâmica da interface, assegurando um fluxo contínuo e confiável de informações.

A comunicação com o banco de dados foi realizada através do SQLAlchemy e PyMySQL, permitindo a execução de consultas SQL de forma mais abstrata e organizada. O SQLAlchemy foi utilizado para mapear as tabelas do banco de dados como classes Python, facilitando a manipulação dos registros e garantindo maior flexibilidade na implementação de novas funcionalidades. Essa abordagem também contribuiu para a manutenção do código, uma vez que as operações de banco de dados foram encapsuladas em métodos reutilizáveis.

A manipulação dos dados foi feita por meio de um DataFrame do Pandas, que serviu como um intermediário entre o banco de dados e a QTableView. Esse modelo permitiu a aplicação de filtros, cálculos e formatações personalizadas diretamente na memória, melhorando o desempenho da aplicação. Para manter os dados sempre sincronizados, o back-end foi configurado para que, ao realizar qualquer edição em um campo da tabela, os valores fossem automaticamente propagados para o banco de dados e recalculados conforme necessário.

A implementação dos cálculos automáticos foi um ponto essencial do back-end. Como algumas colunas da tabela dependem de valores de outros campos (como os totais de benefícios e cálculos de renda per capita), foram criados métodos que, ao detectar uma alteração, recalculavam os valores dependentes e atualizavam tanto a interface gráfica quanto o banco de dados. Esse processo foi realizado dentro do método setData da QAbstractTableModel, garantindo que qualquer modificação na QTableView acionasse automaticamente a atualização dos valores relacionados.

Para armazenar configurações do sistema de maneira flexível e persistente, foi utilizado o formato *JavaScript Object Notation* (JSON). Esse arquivo de configuração foi empregado para salvar informações como os delegates dos QComboBox, parâmetros de cálculo do salário mínimo, mapeamento das colunas e o estado de visualização das colunas dentro da

QTableView. Dessa forma, a personalização realizada pelo usuário na interface permanece mesmo após o fechamento do sistema, proporcionando maior usabilidade e controle.

A comunicação em tempo real do sistema foi implementada utilizando Kafka e Zookeeper, garantindo um fluxo eficiente de mensagens entre os componentes. O Kafka atua como um intermediário, permitindo a troca de informações entre a interface gráfica e o back-end por meio do modelo de publicação e assinatura (publish-subscribe). No sistema, a interface funciona como um produtor, enviando eventos, como alterações em registros ou solicitações de atualização, para tópicos específicos no Kafka. O back-end, por sua vez, opera como um consumidor, processando essas mensagens e executando as ações necessárias, como cálculos, persistência de dados e atualizações na interface. O Zookeeper gerencia e coordena os serviços do Kafka, garantindo a distribuição e sincronização das mensagens, tornando o sistema mais confiável e escalável. Em síntese, e sistematicamente explicando, todo esse processo foi organizado no Diagrama de Processos, conforme a Figura 2.

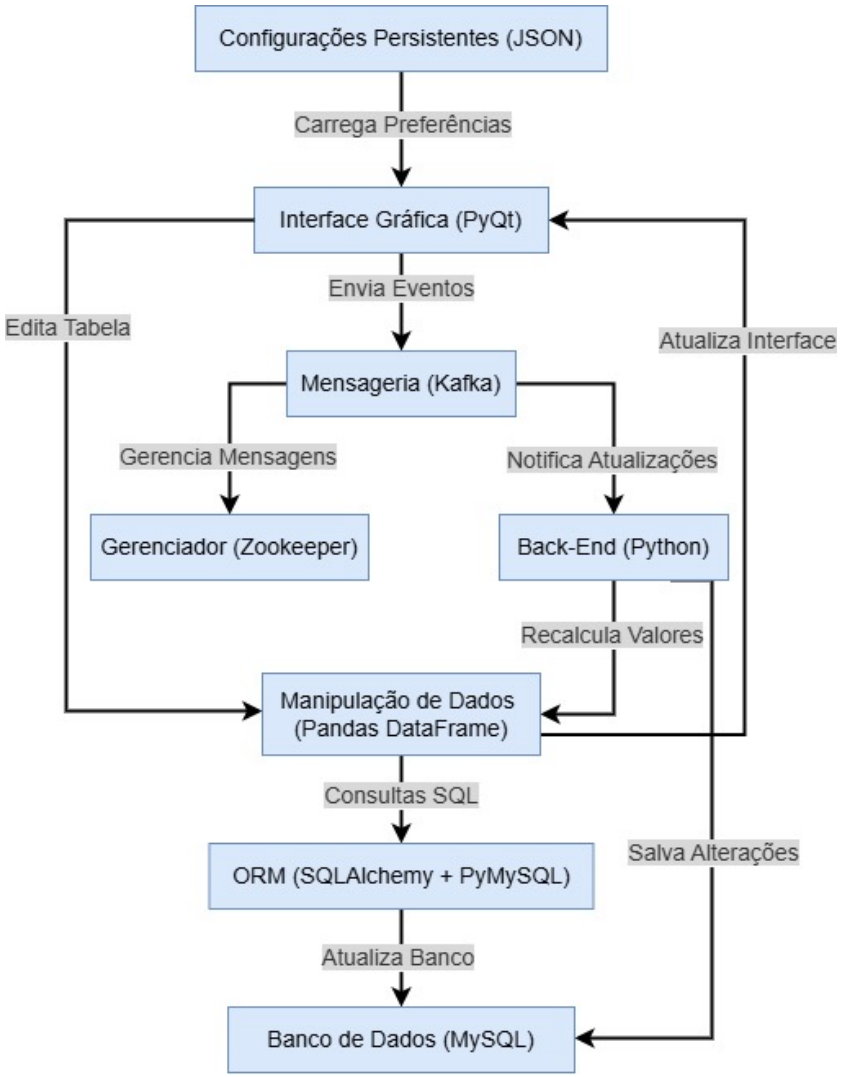
Além disso, o back-end gerencia a funcionalidade de filtros e pesquisas dinâmicas, permitindo que os usuários localizem rapidamente registros específicos sem comprometer a integridade dos dados. Inicialmente, a aplicação dos filtros foi feita diretamente no DataFrame, mas posteriormente passou a ser realizada no próprio banco de dados para otimizar o desempenho e evitar problemas ao salvar as edições. Essa mudança também reduziu o consumo de memória, especialmente ao trabalhar com grandes volumes de dados.

3.3.3 Front-end

A interface gráfica do sistema foi desenvolvida utilizando PyQt5, uma biblioteca poderosa para a criação de aplicações desktop com Python, baseada no framework Qt. Para facilitar a construção e organização dos componentes visuais, foi utilizado o Qt Designer, uma ferramenta que permite o design das telas de maneira intuitiva por meio de uma abordagem WYSIWYG (What You See Is What You Get). Esse ambiente gráfico possibilitou a criação das janelas do sistema de forma visual, gerando arquivos .ui, que posteriormente foram convertidos para código Python utilizando o comando pyuic5.

O uso do Qt Designer trouxe diversas vantagens para o desenvolvimento, como a disposição precisa de elementos na tela, a facilidade na configuração de layouts e estilos, além da possibilidade de reutilização de componentes. A interface foi estruturada utilizando QWidgets principais, como QMainWindow, QDialog e QWidget, organizados em layouts do tipo

Figura 2 – Diagrama de Processos backend



Fonte: Elaborado pelo Autor (2025)

QVBoxLayout e QGridLayout para garantir a responsividade e a organização dos elementos.

A exibição dos dados na aplicação foi realizada por meio do QTableView, que permitiu uma visualização estruturada e personalizável das informações. Para gerenciar os dados, utilizou-se um modelo baseado em QAbstractTableModel, onde foi implementado um PandasModel, responsável por integrar o DataFrame do pandas à tabela do PyQt, possibilitando operações como edição, ordenação e filtragem dinâmica.

Os campos de entrada da aplicação foram criados com QLineEdit para textos curtos, QTextEdit para campos maiores, QComboBox para seleção de opções predefinidas e QSpinBox para valores numéricos. Botões interativos, implementados com QPushButton, permitiram a execução das principais ações do sistema, como salvar alterações, aplicar filtros e exportar dados.

A aplicação também contou com a implementação de eventos e sinais do Qt para tornar a interface responsiva às interações do usuário. O mecanismo de signals e slots do QtCore foi amplamente utilizado para conectar ações aos elementos gráficos, permitindo, por exemplo, que a edição de um campo na tabela acionasse automaticamente a atualização dos cálculos nos campos relacionados. Além disso, foram implementados delegates personalizados para o QTableView, possibilitando o uso de QComboBox e formatações específicas dentro das células da tabela.

Para manter um padrão visual agradável e intuitivo, a interface foi estilizada com *Qt Style Sheets* (QSS), um sistema semelhante ao CSS utilizado para personalizar cores, fontes, espaçamentos e realces da aplicação. Essa personalização garantiu uma identidade visual mais organizada e facilitou a diferenciação de elementos importantes, como realce de linhas alteradas e botões de ação.

O desenvolvimento da interface ocorreu de maneira iterativa, sendo constantemente ajustado conforme os testes e feedbacks recebidos ao longo do projeto. Essa abordagem permitiu a otimização da usabilidade do sistema, garantindo que a interação do usuário fosse fluida e eficiente no gerenciamento dos auxílios estudantis.

3.3.4 Implantação

A implantação do sistema foi realizada utilizando Docker, garantindo um ambiente isolado e padronizado para a execução dos serviços essenciais. A arquitetura do sistema conta com três contêineres principais: Kafka, MySQL e Zookeeper, cada um desempenhando um papel fundamental na comunicação e armazenamento de dados. O Zookeeper é responsável por gerenciar e coordenar os nós do Kafka, garantindo a estabilidade e o gerenciamento de partições. O Kafka atua como middleware para a troca de mensagens assíncronas entre componentes do sistema, facilitando a comunicação eficiente entre os serviços. Já o MySQL armazena os dados da aplicação, garantindo persistência e integridade nas operações realizadas.

A configuração desses serviços foi definida no arquivo docker-compose.yml, conforme a Figura 3, permitindo a criação e gerenciamento dos contêineres de maneira simplificada. O Zookeeper é inicializado primeiro para garantir que o Kafka possa se conectar corretamente. O Kafka, por sua vez, é configurado para operar na rede interna do Docker, permitindo a comunicação eficiente com os demais serviços. O MySQL é configurado com um banco de dados inicial chamado caseleg e um volume persistente para evitar a perda de dados ao reiniciar os

contêineres.

Figura 3 – Código fonte do arquivo docker-compose.yml

```yaml
version: '3.8'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.5.0
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    networks:
      - kafka-net

  kafka:
    image: confluentinc/cp-kafka:7.5.0
    container_name: kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    depends_on:
      - zookeeper
    networks:
      - kafka-net

  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: caseleg
    volumes:
      - mysql_data:/var/lib/mysql
      - ./my.cnf:/etc/mysql/conf.d/my.cnf
    networks:
      - kafka-net


networks:
  kafka-net:
volumes:
  mysql_data:

```

Fonte: Elaborado pelo Autor (2025)

Essa abordagem proporciona diversos benefícios, como o isolamento do ambiente, onde cada serviço opera dentro de seu próprio contêiner, evitando conflitos de dependências, além da facilidade de replicação, permitindo que o mesmo ambiente seja reproduzido em diferentes máquinas. O gerenciamento também se torna mais simples, pois o docker-compose permite iniciar, parar e reiniciar todos os serviços com um único comando. Com essa configuração, o sistema pode ser implantado de forma eficiente e escalável, garantindo maior estabilidade e confiabilidade para a aplicação.

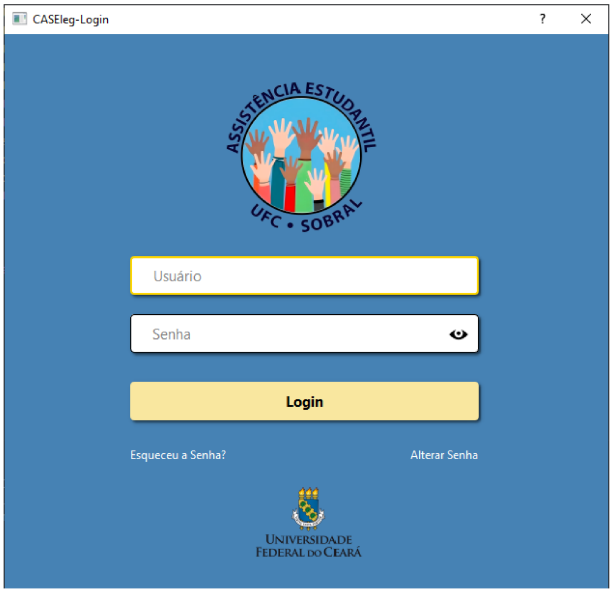
4 RESULTADOS ESPERADOS

Nesta seção, são apresentados alguns recortes de tela das principais partes do sistema desktop. A partir do desenvolvimento desse software, espera-se apresentar uma interface gráfica agradável e útil para o usuário da assistência estudantil. Proporcionar um aumento na eficiência do processo de avaliação de benefícios, destacando a redução do tempo necessário para avaliar e aprovar solicitações de assistência estudantil. Apresentar feedback positivo dos usuários, incluindo estudantes, pessoal administrativo e outros envolvidos no processo de assistência estudantil, indicando satisfação com a eficácia do software.

4.1 Tela de Login

Na Figura 4 é possível ver a Tela de Login. É a interface inicial do sistema, responsável por controlar o acesso dos usuários. Nela, são apresentados dois campos para entrada de credenciais: Usuário e Senha. Esses dados devem ser previamente cadastrados pelo Administrador, garantindo que apenas usuários autorizados possam acessar o sistema. Além do login convencional, a tela oferece duas opções adicionais: "Esqueceu a senha ?"e "Alterar Senha". A opção "Esqueceu a senha"permite que o usuário solicite a redefinição da senha, garantindo a recuperação do acesso em caso de esquecimento. Já a opção "Alterar Senha"possibilita a modificação da senha cadastrada

Figura 4 – Tela de Login

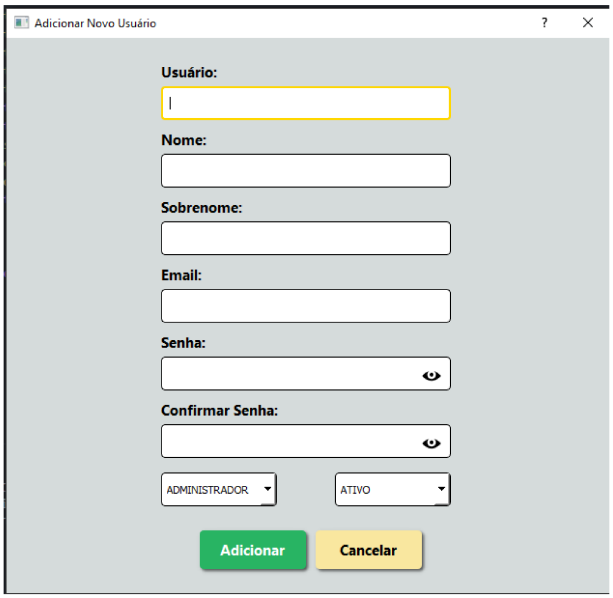


Fonte: Elaborado pelo Autor (2025)

4.2 Tela de Cadastro

Na Figura 5 é possível ver a Tela de Cadastro. Permite a criação de novos usuários no sistema. Nela, o usuário deve preencher os seguintes campos: usuário, nome, sobrenome, e-mail, senha, confirmação de senha, permissão e status. O e-mail é um dado essencial para recuperação de senha e comunicação com o sistema. A senha deve ser definida respeitando os critérios de segurança estabelecidos, sendo necessário digitá-la novamente no campo de confirmação de senha para evitar erros de digitação. O campo de permissão define o nível de acesso do usuário, podendo ser configurado, por exemplo, como Administrador ou Usuário Padrão, garantindo controle sobre as funcionalidades disponíveis para cada perfil. Já o status indica se a conta está ativa ou inativa, sendo que apenas usuários com status ativo podem acessar o sistema.

Figura 5 – Tela de Cadastro



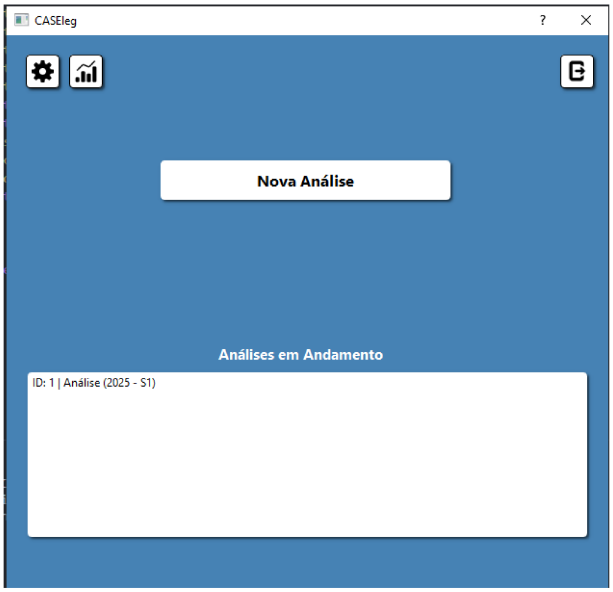
Fonte: Elaborado pelo Autor (2025)

4.3 Tela de Abrir / Criar Análise

Na Figura 6 é possível ver a tela de Abrir/Criar Análise. Essa tela é o ponto de entrada para a gestão das análises dentro do sistema. Nela, há um botão principal que permite a criação de uma nova análise. Além disso, a interface exibe uma lista com todas as análises existentes, permitindo que o usuário visualize, selecione e continue o trabalho em análises previamente criadas. Para garantir maior controle e personalização, a tela conta com um botão

de configuração, onde o usuário pode ajustar preferências do sistema. Também está disponível um botão de relatórios, que permite a geração de resultados das análises realizadas. Por fim, há um botão de logoff, possibilitando que o usuário encerre sua sessão.

Figura 6 – Tela de Abrir/ Criar Análise



Fonte: Elaborado pelo Autor (2025)

4.4 Tela Principal (TableView)

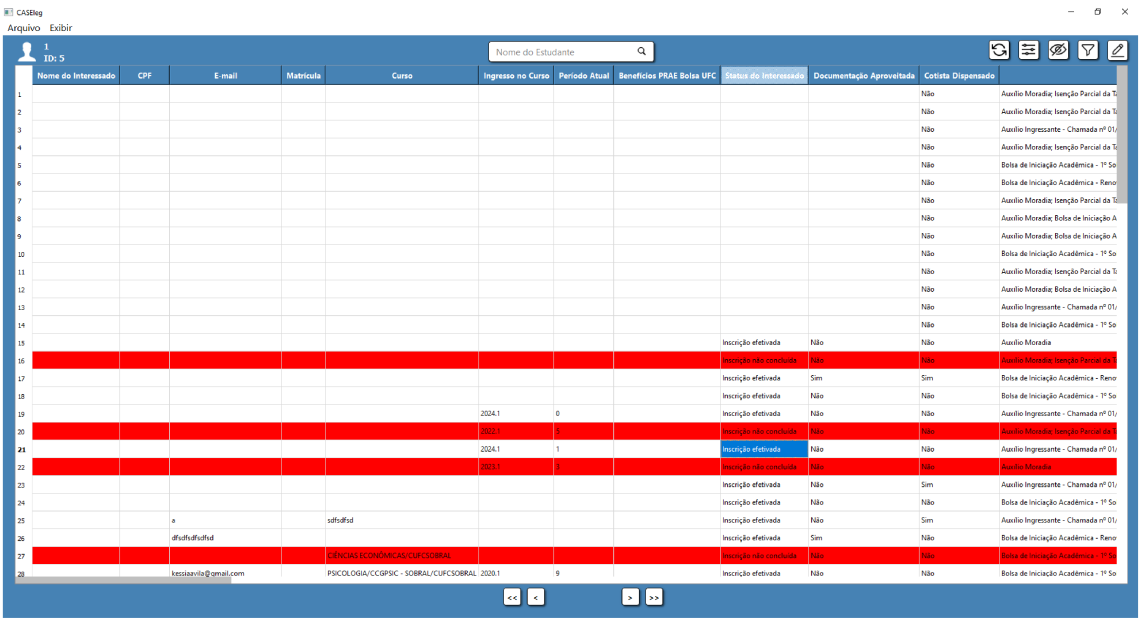
Na Figura 7 é possível ver a Tela Principal. Essa tela é o ambiente central do sistema, onde os usuários podem visualizar e gerenciar os dados dos estudantes de forma organizada e eficiente. No centro da interface, há uma tabela que exibe as informações dos estudantes, permitindo a navegação e interação direta com os registros. Acima da tabela, há uma caixa de pesquisa que possibilita a busca rápida pelo nome do estudante.

Para a navegação entre as páginas de registros, a tela conta com quatro botões: Primeiro, Anterior, Próximo e Último, permitindo que o usuário percorra os dados de forma fluida. Além disso, há cinco botões principais para ações específicas: o botão "Refresh", que atualiza e reorganiza os dados na tabela, o botão de "Atualização em Massa", que permite modificar múltiplos registros simultaneamente, o botão de "Ocultar Colunas", que possibilita personalizar a exibição das colunas da tabela; o botão de "Filtrar Dados", que aplica filtros personalizados para exibir apenas registros que atendam a determinados critérios; e o botão "Editar Informações do Estudante", que abre a tela de edição, onde é possível modificar os dados

individuais de um estudante.

No menu suspenso, há duas categorias principais. O menu "Arquivo"contém as opções de "Exportar", permitindo a exportação dos dados conforme critérios específicos, e "Sair", para encerrar a sessão. Já o menu "Exibir"oferece as opções de acesso à tela de "Configurações"e tela de "Relatórios".

Figura 7 – Tela Principal (TableView)



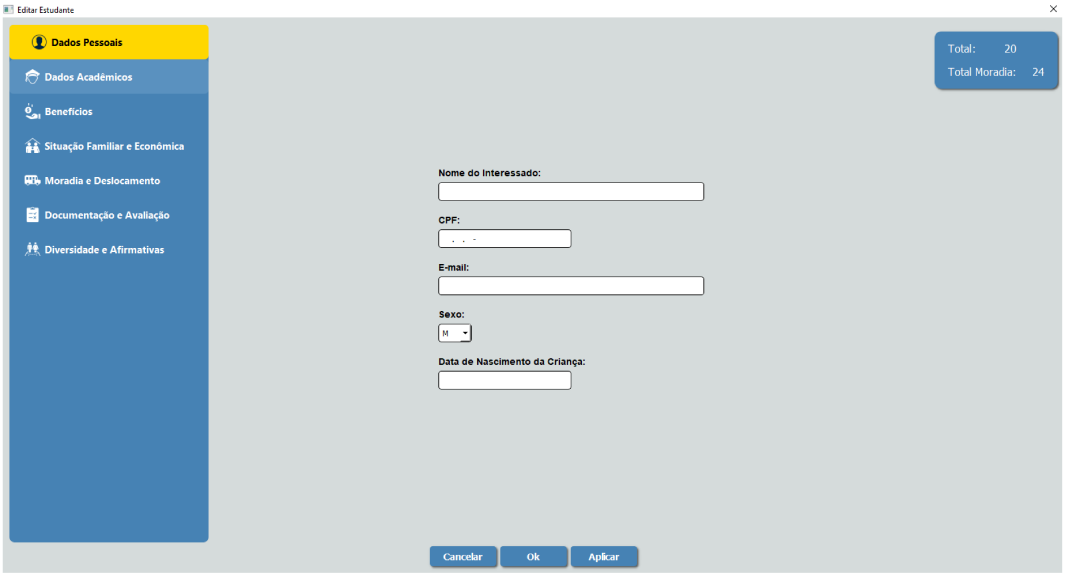
Fonte: Elaborado pelo Autor (2025)

4.5 Tela de Edição

Na Figura 8 é possível ver a Tela de Edição. Essa tela permite a modificação detalhada dos dados de um estudante, organizada em sete abas: Dados Pessoais, Dados Acadêmicos, Benefícios, Situação Familiar e Econômica, Moradia e Deslocamento, Documentação e Avaliação, e Diversidade e Afirmativas. Cada aba contém os campos necessários para atualizar as informações de forma estruturada.

Além disso, a tela exibe uma label informativa que apresenta a pontuação referente ao Total do Auxílio Moradia e ao Total dos demais auxílios. Para confirmar ou descartar alterações, há três botões principais: Aplicar, que salva as mudanças sem fechar a tela; Cancelar, que descarta modificações não confirmadas; e Ok, que confirma as edições e fecha a tela.

Figura 8 – Tela de Edição



Fonte: Elaborado pelo Autor (2025)

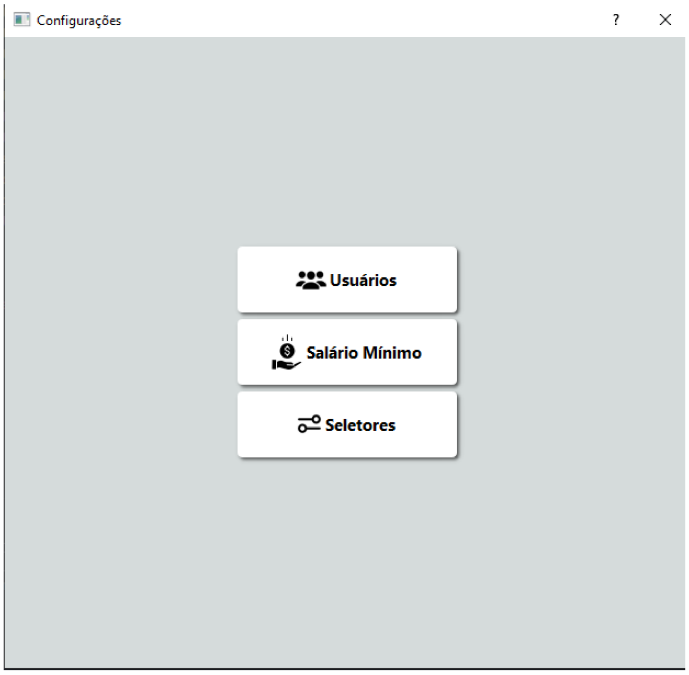
4.6 Tela de Configuração

Na Figura 9 é possível ver a Tela de Configuração. Essa tela permite ajustes essenciais do sistema por meio de três botões principais. O primeiro botão dá acesso à Tela de Gerenciamento de Usuários, onde é possível cadastrar e editar usuários do sistema. O segundo botão permite alterar o valor do salário mínimo, garantindo que os cálculos relacionados aos auxílios sejam atualizados conforme necessário. O terceiro botão possibilita a modificação dos seletores das opções das colunas (delegates), permitindo personalizar os valores disponíveis para seleção em determinadas colunas da tabela.

4.7 Tela de Relatórios

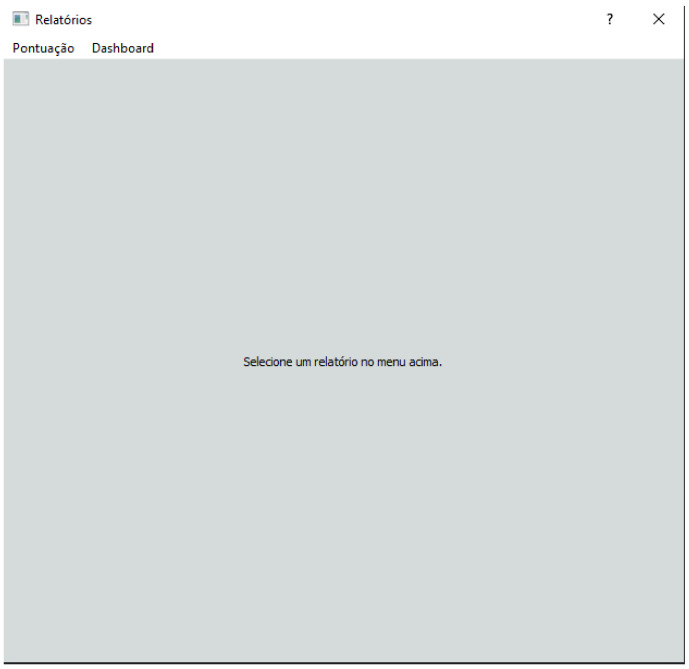
Na Figura 10 é possível ver a Tela de Relatórios. Essa tela permite a geração e visualização de diferentes tipos de análises sobre os dados do sistema. Nela, há um menu suspenso que organiza as opções disponíveis. O primeiro menu, denominado "Pontuação", contém duas opções: "Pontuação por Estudante", que exibe um relatório detalhado com a pontuação individual de cada estudante, e "Pontuação por Curso", que apresenta a pontuação agregada por curso, permitindo comparações entre diferentes turmas. O segundo menu, "Dashboard", fornece uma visão geral dos dados por meio de gráficos e indicadores, facilitando a análise de tendências e a tomada de decisões estratégicas.

Figura 9 – Tela de Configuração



Fonte: Elaborado pelo Autor (2025)

Figura 10 – Tela de Relatórios

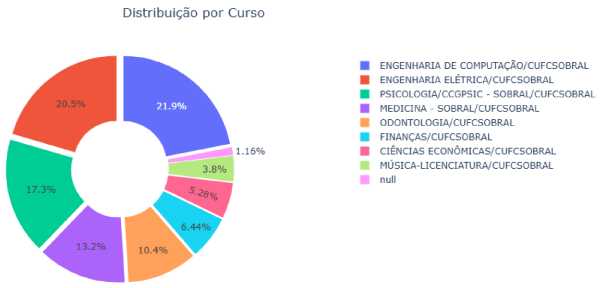


Fonte: Elaborado pelo Autor (2025)

4.8 Tela de Dashboard

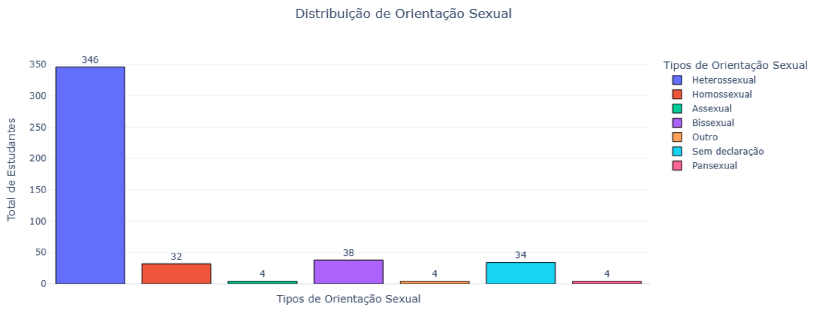
A Tela de Dashboard permite a geração e visualização de diferentes tipos de gráficos sobre a distribuição de alguns parâmetros. Na Figura 11 é possivel ver o gráfico de pizza para Distribuição dos Cursos. Na Figura 12 é possivel ver o gráfico de coluna para Distribuição de Orientação Sexual. Na Figura 13 é possivel ver o gráfico de coluna para Distribuição dos Raças. Na Figura 14 é possivel ver o gráfico de pizza para Distribuição de Sexos. Na Figura 15 é possivel ver o gráfico de coluna para Distribuição de Deficiências.

Figura 11 – Gráfico de Distribuição de Cursos



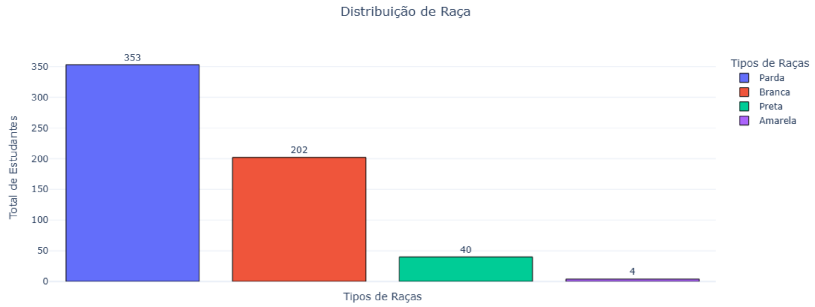
Fonte: Elaborado pelo Autor (2025)

Figura 12 – Gráfico de Distribuição de Orientação Sexual



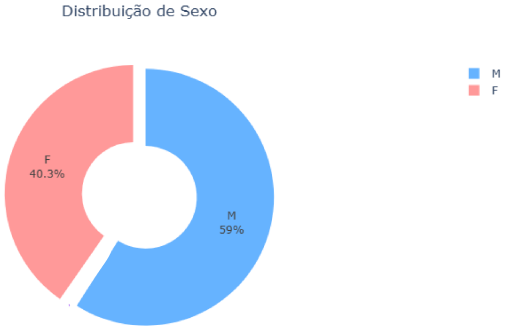
Fonte: Elaborado pelo Autor (2025)

Figura 13 – Gráfico de Distribuição de Raças



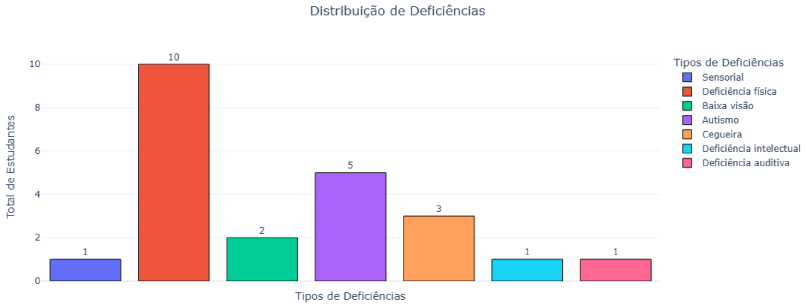
Fonte: Elaborado pelo Autor (2025)

Figura 14 – Gráfico de Distribuição de Sexos



Fonte: Elaborado pelo Autor (2025)

Figura 15 – Gráfico de Distribuição de Deficiências



Fonte: Elaborado pelo Autor (2025)

5 DISCUSSÃO

Inicialmente, o sistema foi concebido com a proposta de automatizar a elegibilidade dos estudantes por meio de técnicas de árvore de decisão e aprendizado de máquina. No entanto, após conversas com as assistentes sociais, constatou-se que a análise de elegibilidade é baseada em uma soma de pesos, estruturada a partir de critérios socioeconômicos bem definidos. Essa descoberta direcionou a implementação do sistema para um modelo mais alinhado à realidade da CASE Sobral, garantindo maior transparência e adequação às necessidades institucionais.

Atualmente, o mapeamento dos dados é realizado manualmente, com as assistentes sociais utilizando o Google Planilhas para organizar as informações. Esse processo exige que os dados sejam copiados manualmente de arquivos CSV para uma planilha principal, tornando a atividade demorada e sujeita a erros. Além disso, após a conclusão da análise, as assistentes precisam exportar os dados e montar um novo CSV manualmente para envio ao Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA). Com a introdução do novo sistema, tanto a importação quanto a exportação dos dados passarão a serem automatizadas, reduzindo significativamente o tempo de trabalho e eliminando a necessidade de intervenção manual, resultando em maior precisão, eficiência e agilidade.

O sistema também permite a alteração da variável do salário mínimo e dos pesos da fórmula de elegibilidade, oferece uma interface de edição mais organizada e intuitiva, centraliza as informações em um banco de dados e inclui relatórios e dashboards para análise dos dados.

Com a implementação do sistema, as assistentes sociais da CASE passam a dispor de mais tempo para se dedicar a outras atividades da assistência estudantil, como o acompanhamento individualizado dos estudantes e a formulação de novas políticas de apoio. Além disso, a otimização dos processos permitirá o desenvolvimento de uma assistência estudantil baseada em evidências, possibilitando a realização de pesquisas sociais e o mapeamento do perfil discente.

A redução do tempo gasto no processo seletivo permitirá a ampliação de espaços coletivos de escuta das demandas estudantis, favorecendo a criação de ações mais alinhadas às reais necessidades da comunidade acadêmica. Para a UFC, o desenvolvimento desse sistema representa um avanço na automação de processos administrativos e acadêmicos, melhorando a eficiência institucional e aprimorando o fluxo de trabalho em diferentes setores. A partir dos dados e resultados obtidos, será possível estruturar atividades e políticas mais eficazes, fortalecendo a permanência e o bem-estar dos estudantes.

Atualmente, o sistema está em ambiente de teste, onde está sendo validado e ajustado

com base no feedback dos usuários. Após essa fase, ele será instalado no servidor da UFC para utilização efetiva pela CASE, garantindo que todos as assistentes sociais possam utilizá-lo de forma integrada e eficiente.

6 CONCLUSÃO E TRABALHOS FUTUROS

Ao final do desenvolvimento deste projeto, alcançamos um resultado satisfatório: um software desktop funcional, que atende integralmente a todos os requisitos e objetivos estabelecidos. O processo foi desafiador, mas recompensador, especialmente ao observar o impacto positivo que a ferramenta trará para a gestão da assistência estudantil.

A implantação do sistema foi planejada com atenção para garantir uma transição segura, eficiente e intuitiva para os usuários. Inicialmente, ele será instalado no servidor da UFC, passando por uma fase de testes e ajustes finais com base no feedback das assistentes sociais da CASE. Após essa etapa, será disponibilizado para uso em produção, possibilitando uma análise socioeconômica mais ágil e precisa dos estudantes. Além disso, a interface intuitiva e as funcionalidades automatizadas, como o mapeamento de dados e a exportação de relatórios, facilitarão a comunicação e a colaboração entre os envolvidos no processo de concessão de auxílios estudantis. Dessa forma, o sistema estará sempre preparado para atender às demandas da CASE, promovendo um processo mais transparente, eficiente e justo.

Para aprimorar ainda mais a ferramenta, algumas melhorias futuras podem ser implementadas, tornando o sistema mais eficiente e adaptável às necessidades da CASE e da UFC como um todo. Uma das principais evoluções seria a integração com o SIGAA, permitindo a comunicação direta e eliminando a necessidade de exportação e importação de arquivos CSV. Além disso, a inclusão de um módulo de análise preditiva, utilizando técnicas de aprendizado de máquina, possibilitaria a identificação de padrões e a antecipação da demanda por auxílios estudantis, auxiliando na tomada de decisões estratégicas.

Outro avanço relevante seria a expansão do sistema para outras unidades da UFC que lidam com a concessão de auxílios estudantis, ampliando seu impacto e funcionalidade. Com esses aprimoramentos, o sistema poderá se tornar uma solução ainda mais robusta e abrangente, contribuindo para a modernização e eficiência da gestão da assistência estudantil.

Apesar dos avanços, reconhecemos que o software não é uma solução definitiva para todos os desafios da assistência estudantil. Podemos considerar como limitação, a estrutura da tabela dados_estudantes, que não foi dividida em múltiplas tabelas devido ao seu grande volume de informações, totalizando 63 colunas, o que pode impactar a escalabilidade e manutenção do banco de dados. Outra limitação é a ausência de métricas para avaliar os resultados da implantação do sistema, uma vez que ele ainda não foi posto em prática, impossibilitando uma análise concreta de seu impacto na gestão dos auxílios estudantis.

REFERÊNCIAS

BRASIL. **DECRETO Nº 7.234, DE 19 DE JULHO DE 2010**. 2010. <https://www.planalto.gov.br/ccivil_03/_ato2007-2010/2010/decreto/d7234.htm>. [Accessed 28-10-2023].

BRASIL. **LEI Nº 14.914, DE 3 DE JULHO DE 2024**. 2024. <https://www.planalto.gov.br/ccivil_03/_ato2023-2026/2024/lei/L14914.htm>. [Accessed 13-03-2025].

CARVALHO, V. **MySQL: Comece com o principal banco de dados open source do mercado**. Sao Paulo: Editora Casa do Código, 2015. 165 p. ISBN 978-85-5519-079-7.

COPELAND, R. **Essential SQLAlchemy**. 1. ed. Sebastopol: O'Reilly Media, 2008. ISBN 0596516142,9780596516147.

DABBAS, E. **Interactive Dashboards and Data Apps with Plotly and Dash: Harness the power of a fully fledged frontend web framework in Python–no JavaScript required**. Birmingham: Packt Publishing Ltd, 2021.

FOUNDATION, P. S. **secrets — Generate secure random numbers for managing secrets**. 2025. Acessado em 5 de fevereiro de 2025. Disponível em: <https://docs.python.org/3/library/secrets.html>.

GARG, N. **Apache kafka**. Birmingham: Packt Publishing Birmingham, UK, 2013.

HUNT, J. Pymysql module. In: **Advanced Guide to Python 3 Programming**. Heidelberg: Springer, 2023. p. 377–387.

MCKINNEY, W. *et al.* pandas: a foundational python library for data analysis and statistics. **Python for high performance and scientific computing**, Seattle, v. 14, n. 9, p. 1–9, 2011.

MENDONÇA, R. A. R. de. Levantamento de requisitos no desenvolvimento ágil de software. **Semana da Ciência e Tecnologia da PUC Goiás**, v. 12, 2014.

MERKEL, D. *et al.* Docker: lightweight linux containers for consistent development and deployment. **Linux j**, v. 239, n. 2, p. 2, 2014.

MILANI, A. **MySQL-guia do programador**. Sao Paulo: Novatec Editora, 2007.

OLIPHANT, T. E. *et al.* **Guide to numpy**. Charleston: Trelgol Publishing USA, 2006. v. 1.

PRAE. **PRÓ-REITORIA DE ASSISTENCIA ESTUDANTIL**. 2025. <https://prae.ufc.br/pt/>. [Accessed 28-10-2023].

RONACHER, A. Jinja2 documentation. **Welcome to Jinja2—Jinja2 Documentation (2.8-dev)**, 2008.

SRIRAMYA, P.; KARTHIKA, R. Providing password security by salted password hashing using bcrypt algorithm. **ARPN journal of engineering and applied sciences**, v. 10, n. 13, p. 5551–5556, 2015.

UFC-PRAE. **Auxílios e Bolsas Estudantis**. 2025. <https://prae.ufc.br/pt/auxilios-e-bolsas-estudantis/>. [Accessed 13-03-2025].

UFC-SOBRAL. **UFC Sobral — sobral.ufc.br**. 2025. <https://sobral.ufc.br/>. [Accessed 28-10-2023].

WILLMAN, J. M. Beginning pyqt: A hands-on approach to gui programming. Apress, 2020.