INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

JOÃO MARCOS DE SOUZA

DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE COMPRAS E ESTOQUE PARA O REFEITÓRIO DO CAMPUS MACAU DO IFRN

IPANGUAÇU

2026

JOÃO MARCOS DE SOUZA

DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE COMPRAS E ESTOQUE PARA O REFEITÓRIO DO CAMPUS MACAU DO IFRN

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Informática do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Licenciado em Informática.

Orientador: Dr. Paulo Orlando Vieira de Queiroz Sousa

IPANGUAÇU

2026

Instituto Federal de Educação, Ciência e Tecnologia do Estado do Rio Grande do Norte – IFRN Sistema Integrado de Bibliotecas – SIBi

Elaborada pela Bibliotecária
Taíze Nascimento – CRB-15/752

JOÃO MARCOS DE SOUZA

Desenvolvimento de um sistema de gerenciamento de compras e estoque para o refeitório do campus Macau do IFRN

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Informática do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Licenciado em Informática.

Trabalho de Conclusão de Curso aprovado em ___/___/____ pela seguinte Banca Examinadora:

_____

Paulo Orlando Vieira de Queiroz Sousa – Orientador

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Rafael Castro de Souza

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Samuray do Nascimento Fernandes

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Dedicatória

Dedico este trabalho a minha família e a minha esposa, que sempre estiveram ao meu lado.

RESUMO

A gestão de compras, contratos e estoque de gêneros alimentícios em refeitórios institucionais é fundamental para garantir a oferta contínua e adequada da merenda escolar, especialmente no contexto da administração pública, que exige controle, transparência e eficiência na aplicação dos recursos. Diante desse cenário, este trabalho tem como objetivo desenvolver um sistema web para o gerenciamento de compras, contratos e estoque dos insumos utilizados na merenda escolar do campus Macau do IFRN, atendendo às necessidades dos fiscais de contrato e dos colaboradores terceirizados do refeitório. A pesquisa caracteriza-se como aplicada, de abordagem qualitativa e procedimento de pesquisa-ação, sendo os requisitos do sistema levantados por meio de entrevistas presenciais com servidores diretamente envolvidos na fiscalização dos contratos de gêneros alimentícios. O sistema foi desenvolvido utilizando o framework Django e a linguagem Python no backend, além de HTML, CSS, JavaScript e Bootstrap na interface, com banco de dados SQLite. Os resultados obtidos demonstram que a ferramenta proposta contribui para a melhoria do controle de estoque, para a organização dos processos de compras e contratos e para a comunicação entre os envolvidos, possibilitando maior eficiência administrativa, redução de desperdícios e melhor suporte à tomada de decisão no âmbito da gestão da merenda escolar.

Palavras-chave: Sistema Web; Gestão de Compras; Gestão de Contratos; Merenda escolar.

ABSTRACT

The management of purchases, contracts, and inventory of food supplies in institutional cafeterias is fundamental to guaranteeing the continuous and adequate supply of school meals, especially in the context of public administration, which demands control, transparency, and efficiency in the application of resources. Given this scenario, this work aims to develop a web-based system for managing purchases, contracts, and inventory of supplies used in school meals at the Macau campus of IFRN, meeting the needs of contract supervisors and outsourced cafeteria staff. The research is characterized as applied, with a qualitative approach and action research procedure, with the system requirements gathered through face-to-face interviews with employees directly involved in the supervision of food supply contracts. The system was developed using the Django framework and the Python language in the backend, as well as HTML, CSS, JavaScript, and Bootstrap in the interface, with an SQLite database. The results obtained demonstrate that the proposed tool contributes to improved inventory control, organization of purchasing and contracting processes, and communication among those involved, enabling greater administrative efficiency, reduced waste, and better support for decision-making in the context of school meal management.

Keywords: Web-based system; Purchasing management; Contract management; School meals.

LISTA DE ILUSTRAÇÕES

Quadro 1 - Resumo dos trabalhos relacionados 27
Figura 1 - Diagrama de caso de uso Administrar 33
Figura 2 - Diagrama de caso de uso do sistema 33
Figura 3 - Diagrama de caso de uso do sistema - Recorte 34
Figura 4 - Recorte do arquivo models.py 35
Figura 5 - Diagrama Entidade Relacionamento 38
Figura 6 - Abas disponíveis para usuários anônimos 41
Figura 7 - Página de login 42
Figura 8 - Carrinho de itens 42
Figura 9 - Página de Pedidos para o funcionário da cozinha 43
Figura 10 - Página de Pedido para o fiscal de contrato 43
Figura 11 - Página de Compra para o fiscal de contrato 44
Figura 12 - Exemplo de PDF gerado para envio de solicitações aos fornecedores 45
Figura 13 - Mensagem de erro ao tentar pagar compra não entregue 45
Figura 14 - Erro ao tentar pagar a compra com saldo insuficiente no empenho 46
Figura 15 - Erro ao informar valor superior para pagamento da compra 47
Figura 16 - Pagamento de valor inferior ao valor total da compra 47
Figura 17 - Erro ao selecionar empenho com saldo insuficiente - Página de editar pagamento 48
Figura 18 - Erro ao informar valor superior para pagamento da compra - Página de editar pagamento 48
Figura 19 - Página de pagamentos com e sem o filtro de compras aplicado 49
Figura 20 - Página de histórico com filtro de data aplicado 49
Figura 21 - Página de contratos com mensagem de erro ao tentar excluir contrato 50

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade - Consistência - Isolamento - Durabilidade
API	Application Programming Interface
BSD	Berkeley Software Distribution
CSRF	Cross-Site Request Forgery
CWI	Centrum Wiskunde & Informatica (CWI)
CRUD	Create-Read-Update-Delete
CSS	Cascading Style Sheets
EVG	Escola Virtual do Governo
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IFRN	Instituto Federal do Rio Grande do Norte
MVC	Model-View-Controller
MVT	Model-View-Template
ORM	Object-Relational Mapping
PAA	Programa de Aquisição de Alimentos
PHP	Hypertext Preprocessor
RIA	Rich Internet Applications
SIG	Sistema de Informação Gerencial
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
SUAP	Sistema Unificado de Administração Pública
TLS	Transport Layer Security
XSS	Cross-Site Scripting

SUMÁRIO

1 INTRODUÇÃO 10
2 OBJETIVOS 12
2.1 OBJETIVO GERAL 12
2.2 OBJETIVOS ESPECÍFICOS 12
3 PROBLEMÁTICA 13
4 REFERENCIAL TEÓRICO 14
4.1 CONTRATOS PARA AQUISIÇÃO DE GÊNEROS ALIMENTÍCIOS 14
4.2 DESENVOLVIMENTO WEB 15
4.3 ARQUITETURA DE APLICAÇÕES WEB 16
4.4 REUSO, FRAMEWORKS E BOOTSTRAP 16
4.5 FRAMEWORK DJANGO 17
4.5.1 Conceitos e características 17
4.5.2 Arquitetura MVT(MODEL-VIEW-TEMPLATE) 18
4.5.3 Segurança das aplicações 19
4.6 PYTHON 19
4.6.1 Características da linguagem 20
4.7 HTML 20
4.8 CSS 21
4.9 JAVASCRIPT 22
4.10 SQLITE 23
5 TRABALHOS RELACIONADOS 25
6 METODOLOGIA 29
6.1 INTERESSADOS 29
6.2 REQUISITOS 30
6.3 DADOS 30
6.4 INSTRUMENTOS 31
6.4.1 Linguagens de Programação 31
6.4.2 Ferramentas Utilizadas 31
7 FERRAMENTA PROPOSTA 32
7.1 ARQUITETURA DA FERRAMENTA 32
7.2 DIAGRAMA DE CASO DE USO 32
7.3 MODELAGEM DO BANCO DE DADOS 34
8 RESULTADOS 41
9 CONSIDERAÇÕES FINAIS 51
REFERÊNCIAS 53
APÊNDICE A - DEPENDÊNCIAS INSTALADAS NO PROJETO 56
APÊNDICE B - CAPTURAS DE TELA DO SISTEMA 57

1 INTRODUÇÃO

Os refeitórios institucionais, presentes em escolas, universidades, empresas e hospitais, exercem papel fundamental na garantia de uma alimentação adequada a um grande número de pessoas, sendo parte essencial das políticas de bem-estar e permanência institucional. Para que esse serviço funcione de forma contínua e eficiente, torna-se indispensável uma gestão eficaz dos insumos alimentícios, abrangendo desde os processos de aquisição até o controle rigoroso do estoque disponível. Segundo Ballou (2006), a gestão inadequada de estoques pode resultar em desperdícios, escassez de materiais e impactos negativos no planejamento financeiro das organizações, especialmente em ambientes que lidam com recursos públicos.

No contexto da administração pública, a necessidade de controle, transparência e rastreabilidade dos gastos é ainda mais relevante, uma vez que os processos de compras e contratos devem atender aos princípios da legalidade, eficiência e economicidade (BRASIL, 2021). Apesar de existirem sistemas institucionais voltados à gestão administrativa, como o Sistema Unificado de Administração Pública (SUAP), observa-se que nem sempre tais ferramentas atendem de forma integral às demandas específicas relacionadas à gestão de contratos de alimentação escolar e ao controle de estoque dos insumos utilizados nos refeitórios. Dessa forma, muitos servidores ainda recorrem a métodos manuais, como planilhas eletrônicas, para registrar compras, controlar quantidades e acompanhar o consumo diário, prática que, embora comum, é suscetível a erros, retrabalho e inconsistências de informação (LAUDON; LAUDON, 2017).

A utilização de sistemas informatizados para o gerenciamento de compras e estoque tem se mostrado uma alternativa eficaz para reduzir falhas operacionais, otimizar processos e fornecer dados confiáveis que auxiliam a tomada de decisão gerencial (SANTOS; COSTA, 2019). No ambiente escolar, esses sistemas contribuem diretamente para o planejamento adequado da merenda, evitando tanto a falta quanto o excesso de alimentos, além de apoiar o controle financeiro e contratual dos recursos empregados.

Diante desse cenário, o presente trabalho tem como objetivo geral desenvolver e apresentar um sistema *web* para a gestão de compras, contratos e estoque dos insumos utilizados no preparo da merenda escolar do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), campus Macau. A solução proposta foi desenvolvida utilizando tecnologias amplamente adotadas no desenvolvimento de aplicações *web*, como o framework

Django e a linguagem Python, além de HTML, CSS, JavaScript, Bootstrap e o banco de dados SQLite, visando garantir segurança, escalabilidade e facilidade de manutenção da aplicação (SANTIAGO *et al*., 2020).

O desenvolvimento do sistema baseou-se na metodologia de *design thinking*, aliada a princípios da engenharia de software, com o intuito de compreender as necessidades dos usuários finais e projetar uma solução funcional, intuitiva e alinhada à realidade institucional. Por fim, este trabalho encontra-se estruturado em seções que abordam a fundamentação teórica, a metodologia empregada, o desenvolvimento do sistema e a análise dos resultados obtidos. Espera-se que a solução apresentada contribua para a melhoria dos processos administrativos relacionados à merenda escolar, promovendo maior organização, controle e eficiência no ambiente de trabalho dos servidores e colaboradores terceirizados.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver um sistema digital para gerenciar as compras, contratos e estoque de alimentos destinados à merenda escolar do campus Macau do IFRN, atendendo às necessidades de fiscais de contrato e funcionários responsáveis pelo preparo das refeições.

2.2 OBJETIVOS ESPECÍFICOS

- Identificar as necessidades e os desafios atuais enfrentados pelos fiscais de contrato e pelos funcionários terceirizados no gerenciamento do estoque de alimentação escolar.
- Mapear os processos existentes de solicitação, recebimento e controle de insumos alimentícios.
- Projetar a arquitetura e as funcionalidades essenciais do sistema de gerenciamento, incluindo módulos para entrada/saída de estoque e geração de relatórios.
- Modelar o banco de dados necessário para armazenar informações sobre fornecedores, produtos, quantidades e movimentações de estoque.
- Implementar as funcionalidades do sistema, garantindo uma interface intuitiva e acessível para os diferentes perfis de usuários (fiscais e funcionários terceirizados).
- Integrar as funcionalidades de registro de entradas (recebimento de mercadorias) e saídas (consumo na preparação das refeições) de forma eficiente.

3 PROBLEMÁTICA

A fiscalização de contratos é algo comum no serviço público em geral, tanto que, plataformas como a Escola Virtual do Governo (EVG), oferecem cursos online gratuitos para que os servidores possam realizá-la de forma adequada e em conformidade com a legislação. Dentro desse contexto, há contratos de menor complexidade, a exemplo do fornecimento de energia elétrica para um estabelecimento público, e contratos com grau de complexidade maior, como o de fornecimento de insumos para alimentação escolar.

Os principais problemas relacionados à fiscalização de contratos de insumos para alimentação escolar, que é objeto deste projeto, são a falta de uma ferramenta para controle de estoque do refeitório da instituição, a comunicação entre os funcionários que ficam no refeitório e os fiscais de contrato, e o controle dos pedidos enviados às empresas juntamente dos processos de pagamentos. Tais problemas acarretam na carência de organização e embasamento para as aquisições, a inexistência de relatórios de consumo de itens e de controle dos gastos, e desorganização nos pagamentos.

Para tentar contornar estes problemas, é comum que os funcionários façam uso de planilhas eletrônicas, contudo, além das limitações estruturais dessas planilhas, o uso delas exige muito conhecimento específico por parte dos fiscais, que precisam criar e modelar a planilha para tentar atender suas necessidades. Além disso, a atualização constante das planilhas exige que o fiscal se comunique recorrentemente com os profissionais da cozinha para monitorar o consumo dos itens.

O IFRN hoje utiliza uma ferramenta de administração chamada SUAP, que possui um módulo de contratos disponível para utilização dos fiscais. No módulo é possível registrar contratos e efetuar medições destes, que é o registro de abertura do processo de pagamento às empresas fornecedoras dos itens ou serviços. Porém, este módulo é genérico e não abrange os problemas abordados neste trabalho, como uma aba dedicada à solicitação de itens por parte da cozinha para a administração, ou ferramenta de controle de estoque.

Um sistema dedicado a essas tarefas facilitaria a comunicação entre os funcionários da cozinha e os fiscais dos contratos, que geralmente trabalham no setor de assistência estudantil, e otimizaria todo os processos de gestão de compras e estoques.

4 REFERENCIAL TEÓRICO

Nesta seção serão abordados os fundamentos teóricos que auxiliaram no desenvolvimento deste trabalho e que foram utilizados como base para desenvolvimento do sistema.

4.1 CONTRATOS PARA AQUISIÇÃO DE GÊNEROS ALIMENTÍCIOS

Os contratos administrativos constituem instrumentos jurídicos utilizados pela Administração Pública para formalizar ajustes com particulares, com a finalidade de atender ao interesse público e assegurar a prestação de bens ou serviços necessários ao funcionamento do estado. Embora a legislação brasileira vigente não apresente uma definição expressa e literal do conceito de contrato administrativo, a lei nº 14.133/2021 estabelece o regime jurídico aplicável a esses contratos, determinando que sejam regidos predominantemente por normas de direito público, admitindo-se, de forma supletiva, a aplicação de princípios e disposições do direito privado quando compatíveis (BRASIL, 2021)

A aquisição de gêneros alimentícios pela administração pública brasileira insere-se no conjunto das contratações públicas, sendo atualmente regulada pela lei nº 14.133/2021, que institui a nova lei de licitações e contratos administrativos. Essa legislação estabelece normas gerais para a contratação de bens, serviços, obras e compras pela administração pública, fundamentando-se em princípios como legalidade, impessoalidade, moralidade, publicidade, eficiência e economicidade. Tais princípios orientam tanto o processo licitatório quanto à execução contratual, assegurando a correta aplicação dos recursos públicos e a observância do interesse público nas aquisições de alimentos (BRASIL, 2021).

No que se refere especificamente à aquisição de gêneros alimentícios, a legislação brasileira passou a incorporar diretrizes que ultrapassam o aspecto meramente econômico das contratações, integrando objetivos sociais e de promoção da segurança alimentar. Nesse contexto, destaca-se a lei nº 14.628/2023, que institui o Programa de Aquisição de Alimentos (PAA), com a finalidade de incentivar a agricultura familiar, promover o acesso à alimentação adequada e fortalecer o desenvolvimento local. Essa norma reforça o papel das compras públicas como instrumento de política pública voltada à inclusão produtiva e à redução das desigualdades sociais (BRASIL, 2023).

A legislação do PAA estabelece que, sempre que possível, os órgãos e entidades da administração pública devem destinar no mínimo 30% dos recursos aplicados na aquisição de

gêneros alimentícios destinados à compra de produtos oriundos da agricultura familiar e de suas organizações, por meio da modalidade de compra institucional. Essa exigência busca garantir mercado aos pequenos produtores, ao mesmo tempo em que contribui para a diversificação da oferta de alimentos e para o fortalecimento de sistemas alimentares sustentáveis (BRASIL, 2023).

4.2 DESENVOLVIMENTO WEB

O desenvolvimento *web*, impulsionado pelo crescimento constante da Internet, visa criar soluções que não apenas entreguem as funcionalidades necessárias, mas que também ofereçam usabilidade, segurança, e escalabilidade, entre outros atributos (SANTIAGO *et al*., 2020). Podemos definir como a construção, criação e manutenção de sites na Internet ou em uma intranet, sendo geralmente dividido em duas principais áreas ou "subprodutos tecnológicos": front-end e back-end (GHIMIRE, 2020).

O *front-end*, também conhecido como desenvolvimento do lado do cliente, concentra-se na parte estética e de exibição de conteúdo, sendo o código que interage com o cliente através de um navegador (GHIMIRE, 2020). O desenvolvedor *front-end* é responsável por criar e manter tudo o que é visto e "navegado" pelo usuário, como a identidade visual do site, menus, textos e imagens, adicionando interatividade (SANTIAGO *et al*., 2020). O desenvolvedor transforma a ideia de *design* da aplicação em realidade, com foco no *layout*, *design* e interatividade da aplicação *web*.

As tecnologias de *back-end* estão ligadas à programação lógica dos sistemas *web*. Elas se preocupam com a forma como os dados são armazenados, acessados e servidos a partir dos servidores (GHIMIRE, 2020).

O *back-end* é o código executado no servidor *web* que encapsula a lógica de negócios da aplicação e a interação com o banco de dados. Essa parte da aplicação consiste no servidor *web* que hospeda o site, uma aplicação para executá-lo e um banco de dados para armazenar as informações (SANTIAGO *et al*., 2020). Linguagens de programação do tipo *server-side* utilizadas no *back-end* incluem PHP, Ruby, Java ou Python.

Dada a complexidade tecnológica do *back-end* e a necessidade de atender a requisitos de qualidade como desempenho, confiabilidade, segurança e disponibilidade do serviço, são frequentemente utilizados *frameworks* de *back-end* (SANTIAGO *et a*l., 2020).

4.3 ARQUITETURA DE APLICAÇÕES WEB

A arquitetura das aplicações *web* frequentemente se baseia em padrões como MVC (Model-View-Controller) ou suas variações, o que permite separar as representações internas dos dados do modo como são apresentados aos usuários (SANTIAGO *et al*., 2020).

O fluxo de interação em um padrão MVC começa quando o usuário faz uma requisição (*request*), o *Controller* a recebe e, em resposta, a aplicação consulta e/ou altera o *Model*, e fornece uma visualização atualizada com uma resposta (*response*) ao usuário.

As páginas dinâmicas são processadas no servidor antes de serem enviadas ao usuário. O processamento no servidor, utilizando linguagens como PHP (PHP Hypertext Preprocessor), altera o conteúdo antes de enviá-lo ao navegador (cliente), o que é essencial para que os sites recebam informações dos usuários e exibam conteúdos personalizados ou que mudam automaticamente (BENTO, 2013).

4.4 REUSO, FRAMEWORKS E BOOTSTRAP

O reuso de software, consolidado como um paradigma dominante na engenharia de software contemporânea, encontra nos frameworks de aplicações uma de suas formas mais eficazes de viabilização em larga escala. Segundo Schmidt *et al*. apud Sommerville (2011), um framework pode ser definido como um conjunto integrado de artefatos de software, tais como classes, objetos e componentes, que colaboram para fornecer uma arquitetura reusável voltada a uma família de aplicações relacionadas.

Os frameworks fornecem a estrutura genérica de um sistema ou subsistema, permitindo o reuso não apenas de código, mas do próprio projeto arquitetural. Essa arquitetura de esqueleto define as interações fundamentais entre os componentes, cabendo ao desenvolvedor apenas a tarefa de especializá-la para atender aos requisitos específicos da aplicação em questão (SOMMERVILLE, 2011).

Um exemplo notório e amplamente utilizado na indústria de desenvolvimento *web* é o Bootstrap, um *framework front-end* de código aberto. O Bootstrap é um *framework* que agrupa uma vasta coleção de trechos de código reutilizáveis escritos em HTML, CSS e JavaScript, funcionando como um kit de construção para interfaces (GARCIA, 2021).

Criado originalmente no Twitter para manter a consistência visual entre projetos internos, o *framework* baseia-se em uma filosofia "*mobile-first*", garantindo que as aplicações

sejam naturalmente responsivas e adaptáveis a diferentes tamanhos de tela (GARCIA, 2021). Ele facilita o reuso ao oferecer componentes e um sistema de *layout* prontos para uso, permitindo que desenvolvedores construam interfaces em uma fração do tempo que seria necessário se tivessem que escrever o CSS do zero.

4.5 FRAMEWORK DJANGO

O Django é um *framework web* de código aberto, escrito em Python, que surgiu com o objetivo de simplificar a criação de soluções *web* escaláveis, promovendo um desenvolvimento rápido e um *design* limpo (SANTIAGO *et al*., 2020).

O *framework* começou a ser desenvolvido em 2003 pela empresa Lawrence Journal World. Foi criado por um grupo de programadores com a intenção de acelerar o desenvolvimento de aplicações *web*. O projeto foi lançado publicamente em 2005 sob a licença BSD (*Berkeley Software Distribution*), tornando-se um projeto de código aberto (URANO, 2023). Atualmente, o Django é mantido pela Django Software Foundation.

A ascensão do Django teve um papel crucial no desenvolvimento *web*, pois influenciou a maior adoção da linguagem Python para a criação de aplicações *server-side* (*back-end*), tornando-a competitiva em relação a outras linguagens que já possuíam *frameworks* robustos (DEMCHENKO, 2019).

4.5.1 Conceitos e características

Django é um *framework* de desenvolvimento *web* que fornece um conjunto de bibliotecas e ferramentas padronizadas para a linguagem Python, visando reduzir o tempo e a complexidade inerentes à criação de sistemas (SANTIAGO *et al*., 2020).

A filosofia do Django é conhecida pelo conceito de “baterias incluídas”, o que significa que ele oferece suporte para a maior parte das necessidades do desenvolvimento *server-side* de forma pronta, sendo altamente personalizável, escalável e contando com uma documentação extensa e uma grande comunidade (URANO, 2023). Ele oferece soluções para grande parte dos problemas tradicionais do desenvolvimento *web*, como autenticação de usuário, administração de conteúdo e mapas de *site* (DJANGO, 2025).

O *framework* é reconhecido por auxiliar na prevenção de erros de segurança comuns e por proporcionar escalabilidade aos sistemas, sendo capaz de suportar picos de tráfego intensos, como 50 mil acessos por segundo, como ocorre em empresas que o utilizam (URANO, 2023).

O Django é amplamente utilizado na criação de:

• Sites e aplicações *web* complexas.

• Plataformas de redes sociais, como Instagram e Pinterest.

• Sistemas de gerenciamento interno (painéis de controle) e aplicações de *e-commerce*.

• *Backends* para aplicativos móveis e construção de APIs.

4.5.2 Arquitetura MVT(MODEL-VIEW-TEMPLATE)

Diferentemente de muitos *frameworks* que se baseiam no padrão MVC (*Model-View-Controller*), o Django adota a arquitetura MVT (*Model-Template-View*). Essa abordagem oferece benefícios como a modularização do código e a facilitação do reuso de componentes. O Django gerencia a maior parte da comunicação entre as requisições HTTP (DJANGO, 2025).

Os componentes centrais da arquitetura MVT são:

1. Model (Modelo): É a camada que representa a estrutura e a lógica dos dados da aplicação. O Django possui uma solução de mapeamento objeto-relacional (ORM) nativa, que permite aos desenvolvedores interagir com o banco de dados utilizando classes e objetos Python para a criação, consulta, atualização e exclusão de dados, sem a necessidade de escrever código SQL diretamente (URANO, 2023).

2. View (Visão): Esta camada é responsável por processar as requisições HTTP recebidas do cliente. A *View* encapsula a lógica de negócios, interage com o *Model* (para obter ou modificar dados) e, em seguida, formata os dados para o *Template* (DJANGO, 2025).

3. Template (*Template*): É a camada de apresentação, responsável pela exibição das informações aos usuários. Consiste em um documento (tipicamente HTML) com marcações especiais que definem como o conteúdo dinâmico será inserido e exibido. O *Template Engine* processa esses arquivos, permitindo a inclusão de lógica simples, como *loops* e condicionais, nos *templates* (URANO, 2023).

Além disso, o Django fornece uma Interface de Administração que é gerada automaticamente com base nos modelos definidos no projeto, facilitando as operações CRUD (*Create, Read, Update, Delete*) sem exigir o desenvolvimento de uma interface personalizada do zero (DJANGO, 2025).

4.5.3 Segurança das aplicações

O framework Django oferece um conjunto abrangente de mecanismos voltados à segurança de aplicações *web*, adotando práticas consolidadas para mitigar vulnerabilidades comuns. Entre essas proteções, destaca-se a prevenção contra *Cross-Site Scripting* (XSS), que é uma vulnerabilidade de segurança *web* onde um invasor injeta scripts maliciosos em sites legítimos, realizada por meio do escape automático de conteúdo dinâmico nos templates, evitando a execução de códigos maliciosos no navegador do usuário (DJANGO, 2025).

O Django também implementa proteção nativa contra *Cross-Site Request Forgery* (CSRF), utilizando *tokens* de verificação que garantem a autenticidade das requisições enviadas ao servidor, impedindo ações não autorizadas realizadas por terceiros (DJANGO, 2025). No acesso ao banco de dados, o uso do *Object-Relational Mapping* (ORM) reduz significativamente o risco de SQL *Injection*, uma vez que as consultas são parametrizadas e abstraídas da escrita direta de comandos SQL (OWENS, 2006).

No que se refere à transmissão de dados, o *framework* oferece suporte ao uso de HTTPS e TLS, assegurando a confidencialidade e a integridade das informações trocadas entre cliente e servidor. Em relação ao armazenamento de credenciais e senhas, o Django adota uma abordagem segura baseada em funções de derivação de chaves, utilizando o algoritmo PBKDF2 (*Password-Based Key Derivation Function* 2) em conjunto com o *hash* SHA-256.

Esse método consiste na aplicação repetida de uma função *hash* criptográfica sobre a senha original, combinada com um valor aleatório (*salt*), resultando em um valor derivado que dificulta ataques de força bruta e de dicionário. O uso de múltiplas iterações torna o processo computacionalmente custoso para atacantes, ao mesmo tempo em que mantém desempenho aceitável para o sistema, sendo amplamente recomendado para proteção de senhas em aplicações modernas (DJANGO, 2025).

4.6 PYTHON

A linguagem Python é uma linguagem de programação de alto nível, interpretada, de tipagem forte e dinâmica, desenvolvida por Guido van Rossum no final da década de 80 no Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI), na Holanda (MENEZES, 2014). O propósito geral do seu desenvolvimento era facilitar a programação sem que o programador precisasse se preocupar com o *hardware*.

4.6.1 Características da linguagem

Python se estabeleceu como uma linguagem poderosa, sendo amplamente reconhecida por sua simplicidade, clareza, robustez e versatilidade (URANO, 2023).

Os aspectos de *design* que contribuem para sua popularidade incluem:

1. Sintaxe enxuta e legibilidade: Python utiliza uma sintaxe simples e enxuta, recorrendo a poucas palavras, o que implica em menos código. Uma grande vantagem é a legibilidade dos programas, o que é reforçado pela sua exigência de indentação como parte da sintaxe, promovendo um código organizado e limpo (MENEZES, 2014). Essa característica facilita o entendimento e o aprendizado, visto que a linguagem utiliza palavras reservadas curtas da língua inglesa .

2. Multiparadigma e flexibilidade: Python possui um grande poder de processamento e suporta a interação com outras linguagens, como Java, C/C++, C#, .NET e PHP. A linguagem é multiplataforma, garantindo portabilidade e compatibilidade em diversos sistemas operacionais, incluindo Linux, Windows e Mac OS (MENEZES, 2014).

3. Código aberto e "baterias inclusas": Python é um projeto de código aberto (*open source*), o que permite que qualquer pessoa acesse e modifique seu código, contribuindo para suas bibliotecas (MENEZES, 2014). Essa abertura suporta a filosofia de “baterias incluídas” (*batteries included*), conceito que indica que o *core* da linguagem já oferece suporte e bibliotecas prontas para a maior parte das necessidades do desenvolvimento *server-side*. Estima-se que existam mais de 246.267 bibliotecas disponíveis para reúso de funcionalidades no desenvolvimento de *software* (SANTIAGO *et al*., 2020.).

4.7 HTML

A HTML (*HyperText Markup Language*) é a linguagem de marcação básica utilizada para o desenvolvimento *front-end* da *web*, tendo como função principal fornecer a estrutura para o conteúdo de um site (SANTIAGO *et al*., 2020). Ela é considerada uma das linguagens fundamentais dos navegadores.

No contexto do desenvolvimento *web*, o *front-end* (lado do cliente) é a parte do sistema que interage com o usuário através de um navegador. A HTML é a linguagem de marcação primária usada nessa camada.

A principal finalidade da HTML é definir elementos estruturais como palavras, títulos, parágrafos e imagens. Essencialmente, um documento HTML é um conjunto de linhas de texto reunidas para instruir um programa, o navegador (*browser*), sobre como apresentar o conteúdo (SOARES, 2008).

4.8 CSS

No início da década de 1990, a *web* apresentava limitações significativas quanto à apresentação visual dos documentos HTML, uma vez que os mecanismos de estilização disponíveis eram rígidos, pouco flexíveis e fortemente dependentes da plataforma utilizada pelo navegador. A ausência de um padrão consolidado dificultava a aplicação consistente de estilos, resultando em incompatibilidades e em uma experiência visual considerada limitada. Além disso, os autores dos documentos possuíam reduzido controle sobre a aparência de suas páginas, o que representava um obstáculo especialmente para profissionais acostumados ao rigor estético das publicações impressas (SILVA, 2015).

Diante desse cenário, surgiu a necessidade de uma solução que permitisse maior domínio sobre a diagramação das páginas, sem comprometer a possibilidade de personalização por parte dos usuários. A proposta do Cascading Style Sheets (CSS) buscou justamente esse equilíbrio, ao oferecer aos criadores de conteúdo maior controle estilístico, mantendo a capacidade de o usuário final influenciar determinados aspectos da apresentação (SILVA, 2015).

Segundo Bos (2016), a concepção do CSS teve origem em 1994, quando Håkon Wium Lie propôs um sistema de folhas de estilo para documentos HTML fundamentado em princípios inovadores. Entre eles, destaca-se o conceito de “cascata”, que permite a combinação de múltiplas folhas de estilo, considerando as preferências do navegador e do usuário como base, sobre as quais os estilos definidos pelo autor são aplicados de forma gradual e ponderada. Essa abordagem possibilitou enriquecer a apresentação visual da *web* sem a necessidade de linguagens complexas de descrição de página, contribuindo para a padronização e evolução do design de interfaces na Internet.

O CSS tem como finalidade definir a forma como os elementos estruturados em HTML são apresentados em uma página *web*, possibilitando o controle de propriedades visuais como cores, tipografia, espaçamentos, bordas e posicionamento dos componentes na interface (FLANAGAN, 2013). Uma de suas principais vantagens reside na separação entre a camada de

conteúdo e a camada de apresentação, o que contribui para a organização do código e facilita a manutenção e evolução dos documentos *web* (BENTO, 2021.).

Por meio do uso de folhas de estilo, torna-se viável modificar integralmente a aparência de um site a partir da alteração de um único arquivo, sem a necessidade de intervenções diretas na estrutura do HTML (BENTO, 2021). Além disso, o CSS possibilita a adaptação da apresentação de uma mesma página a diferentes meios de exibição, como monitores, dispositivos móveis e material impresso, ampliando a flexibilidade e a acessibilidade das aplicações *web* (DEITEL, 2008).

4.9 JAVASCRIPT

O JavaScript surgiu em 1995 como uma ferramenta para aprimorar as capacidades do navegador Netscape Navigator, visando oferecer ao usuário um ambiente de trabalho mais interativo e agradável (EBAC, 2023). Originalmente, a tecnologia foi concebida por Brendan Eich sob o nome de LiveScript. Contudo, em virtude de uma parceria estratégica com a Sun Microsystems, a linguagem foi rebatizada como JavaScript, embora guarde poucas semelhanças estruturais com a linguagem Java, sendo apenas uma semelhança sintática superficial (FLANAGAN, 2013).

A necessidade de padronização frente às diferentes implementações da época, como o JScript da Microsoft, levou a Netscape a submeter a linguagem à ECMA International. Esse esforço resultou, em junho de 1997, na publicação do padrão ECMA-262, formalmente conhecido como ECMAScript. Desde então, a evolução da linguagem foi marcada por versões significativas, como a ECMAScript 3 e a ECMAScript 5, que consolidaram recursos fundamentais para o desenvolvimento de software em larga escala e garantiram a interoperabilidade entre os diferentes navegadores do mercado (FLANAGAN, 2013).

No contexto contemporâneo do desenvolvimento de sistemas, o JavaScript é amplamente considerado a "linguagem da *web*", sendo utilizado em aproximadamente 98,3% de todos os sítios eletrônicos modernos (EBAC, 2023). Ele compõe, juntamente com o HTML e o CSS, a tríade essencial de tecnologias que todo desenvolvedor deve dominar para construir interfaces robustas. Enquanto o HTML foca na especificação do conteúdo e o CSS define a apresentação visual, o JavaScript atua como a camada responsável por gerenciar o comportamento e a interatividade programática através do acesso ao Document Object Model (DOM).

A aplicação dessa tecnologia transcendeu as fronteiras do navegador com o advento de ambientes de execução como o Node.js. Baseado no mecanismo V8 do Google, o Node.js permite que o código JavaScript seja executado no lado do servidor, possibilitando que os desenvolvedores utilizem uma linguagem unificada tanto no *front-end* quanto no *back-end*. Essa arquitetura é notadamente eficiente por ser orientada a eventos e possuir um modelo de entrada e saída não bloqueante, o que favorece o alto desempenho em aplicações com intenso tráfego de rede (BAGLIOTTI; GIBERTONI, 2020).

Outro marco fundamental na história do uso do JavaScript foi a popularização do Ajax, termo cunhado para descrever arquiteturas de aplicações *web* que utilizam scripts para realizar requisições HTTP assíncronas (DEITEL, 2008). Por meio de objetos como o XMLHttpRequest, as aplicações tornaram-se capazes de trocar dados com o servidor em segundo plano, permitindo atualizações parciais da página sem a necessidade de recarregamento completo do documento. Esse avanço foi o pilar para o desenvolvimento das *Rich Internet Applications* (RIAs), que buscam oferecer aos usuários uma sensibilidade e recursos de interface similares aos encontrados em softwares de *desktop* tradicionais (GASPAR *et al*, 2009).

4.10 SQLITE

O SQLite é um sistema gerenciador de banco de dados relacional (SGBD) leve, de código aberto, criado no ano de 2000 por D. Richard Hipp, com o objetivo de fornecer uma solução simples, eficiente e de fácil integração para aplicações que necessitam de persistência de dados sem a complexidade de um servidor de banco de dados tradicional (SQLITE, 2023). Diferentemente de outros SGBDs amplamente utilizados, como MySQL ou PostgreSQL, o SQLite opera de forma embarcada, ou seja, todo o mecanismo de armazenamento e manipulação de dados está contido em uma única biblioteca vinculada à aplicação, eliminando a necessidade de processos de servidor separados.

A principal motivação para o desenvolvimento do SQLite foi atender a cenários em que a simplicidade, a portabilidade e o baixo consumo de recursos são requisitos fundamentais. O SQLite foi projetado para ser confiável, rápido e autossuficiente, características que o tornaram amplamente adotado em aplicações desktop, dispositivos móveis, sistemas embarcados e softwares que demandam uma base de dados local. Sua arquitetura baseada em arquivos permite que todo o banco de dados seja armazenado em um único arquivo no sistema, facilitando a distribuição, o backup e a manutenção dos dados.

No que se refere à sua função, o SQLite é responsável por armazenar, organizar e recuperar dados de forma estruturada, utilizando a linguagem SQL como meio de definição e manipulação das informações. Apesar de sua leveza, o SQLite implementa a maior parte dos recursos essenciais do modelo relacional, como tabelas, índices, chaves primárias e consultas complexas, sendo adequado para aplicações de pequeno e médio porte (OWENS, 2006). Além disso, o sistema garante propriedades de atomicidade, consistência, isolamento e durabilidade (ACID), assegurando a integridade dos dados mesmo em situações de falhas ou interrupções inesperadas (SQLITE, 2023).

5 TRABALHOS RELACIONADOS

Com o intuito de consolidar a concepção do sistema de gerenciamento de compras e estoque destinado ao refeitório do IFRN Campus Macau, tornou-se necessária a realização de um levantamento prévio de soluções semelhantes. Essa etapa teve como finalidade identificar abordagens já adotadas, bem como levantar sugestões de funcionalidades que pudessem contribuir para o aprimoramento do sistema proposto. A partir da análise de artigos científicos e trabalhos de conclusão de curso, foi possível embasar teoricamente o desenvolvimento da proposta e estabelecer critérios relevantes para a definição de seus principais recursos.

Foram analisados cinco trabalhos acadêmicos publicados em repositórios institucionais públicos, que tratam do desenvolvimento de sistemas *web* para resolução de problemas, três deles para gestão de estoque e dois deles sobre gestão de contratos administrativos.

O trabalho acadêmico “Desenvolvimento de um Sistema de Gestão de Estoque para Plataforma Web” (SILVA, 2018) tratou do desenvolvimento de uma plataforma de gestão de estoque para uma empresa de papelaria, visando substituir o controle via planilhas eletrônicas por uma solução integrada. Ele contribuiu com uma base firme sobre a modelagem do banco de dados, além de possuir uma boa documentação a respeito da modelagem de requisitos do sistema. O sistema foi desenvolvido focado no controle de estoque, utilizando linguagens de programação diferentes das propostas no nosso trabalho, além de não conter detalhes sobre a arquitetura implementada, limitando as semelhanças entre os trabalhos.

O trabalho acadêmico “Sistema para controle de estoque”(QUEIROZ, 2018) apresenta um sistema dedicado a controle de estoque com foco na validade dos produtos, com objetivo de evitar perdas de mercadorias. O trabalho se destaca principalmente pela simplicidade e usabilidade, pois busca uma interface intuitiva para que o controle de estoque seja uma tarefa rápida, evitando a burocracia excessiva no registro de saídas.

Apesar dos alimentos serem perecíveis e ser muito importante a atenção com a validade dos mesmos, no nosso projeto optamos por não implementarmos tal funcionalidade, tendo em vista que o campus possui um cardápio semanal, com quantidades de itens pré-definidas, e isso por si só já evita desperdícios de alimentos.

O trabalho “Desenvolvimento de um Sistema para o Gerenciamento de Estoque e Pedidos de Artigos Religiosos” (OLIVEIRA FILHO, 2025) aborda o desenvolvimento de um sistema voltado à organização e monitoramento de produtos em um contexto específico de

comércio. O sistema é estruturado no padrão MVC o que facilita a manutenção e a escalabilidade da aplicação, e traz recursos muito interessantes como um *dashboard* com pedidos em abertos e dados financeiros, traduzindo dados brutos em informações gerenciais visuais. Além disso, o sistema também apresenta excelente gestão de fluxo de pedidos, essencial para a comunicação entre fornecedor e cliente.

O trabalho desenvolvido por Garcia e Queiroz (2020) intitulado “Desenvolvimento de um Sistema Web para Gerenciamento de Compras de Insumos de Manutenção em Contratos de Prestação de Serviços” descreve a criação de um sistema de gestão de compras de insumos de manutenção para a UFERSA, sendo a aquisição de insumos através de contratos de prestação de serviços. O desenvolvimento foi guiado pelos princípios de métodos ágeis Scrum e Lean, com entregas incrementais e prototipagem de baixa fidelidade e se destaca pelo controle de acesso rigoroso por perfis (Admin, Técnico, Assistente e Contratada), garantindo a segurança e a separação de responsabilidades.

O trabalho “Impacto da implantação de um sistema de informação gerencial na gestão de contratos públicos: o caso do Hospital Universitário de Dourados/MS” (CINTRA *et al*, 2012) analisa a implementação de um Sistema de Informação Gerencial (SIG) no Hospital Universitário de Dourados para otimizar a gestão de contratos públicos. O estudo fornece dados estatísticos comparativos entre o período "antes" e "durante" a implantação, demonstrando uma redução de 140,1% nas inconsistências administrativas.

A pesquisa destaca que a automação substituiu processos manuais e ineficientes que causavam perdas de prazos e desperdício de recursos e enfatiza como a tecnologia auxilia no cumprimento da Lei de Responsabilidade Fiscal e melhora o controle de aditivos, vigências e empenhos em instituições públicas.

Para fins comparativos referentes às tecnologias utilizadas nos trabalhos mencionados, foi elaborado o quadro 1 abaixo com um pequeno resumo sobre os pontos fortes de cada trabalho e as tecnologias utilizadas.

Quadro 1 - Resumo dos trabalhos relacionados

Autor / Ano	**Título do Trabalho**	**Pontos Fortes**	**Tecnologias utilizadas**
Silva (2018)	Desenvolvimento de um Sistema de Gestão de Estoque para Plataforma Web	Modelagem do banco de dados, além de possuir uma boa documentação a respeito da modelagem de requisitos do sistema.	Java, JSP, JavaScript, HTML, Bootstrap e banco de dados MySQL.
Queiroz (2018)	Sistema para controle de estoque	Simplicidade e usabilidade, com interface intuitiva.	C# (C Sharp), .NET Framework e banco de dados MySQL.
Oliveira Filho (2025)	Desenvolvimento de um Sistema para o Gerenciamento de Estoque e Pedidos de Artigos Religiosos	Presença de um *dashboard* com pedidos em abertos e dados financeiros; Excelente gestão de fluxo de pedidos	PHP, Laravel, JavaScript, HTML, Bootstrap e banco de dados MySQL.
Garcia & Queiroz (2020)	Desenvolvimento de um Sistema Web para Gerenciamento de Compras de Insumos de Manutenção em Contratos de Prestação de Serviços	Controle de acesso rigoroso por perfis, garantindo a segurança e a separação de responsabilidades;	JavaScript, Node.js, Express, VueJS, Bootstrap e banco de dados MongoDB.

Cintra *et al*. (2012)	Impacto da implantação de um sistema de informação gerencial na gestão de contratos públicos: o caso do Hospital Universitário de Dourados/MS	Análise de impacto real através de estudo comparativo entre dados de período antes e durante a implantação do sistema.	Sistema de informação gerencial(SIG)

Fonte: Autoria Própria

De forma geral, os trabalhos analisados reforçam a relevância do uso de sistemas informatizados para o controle de contratos, compras e estoque, tanto no setor público quanto no privado. Diferentemente das soluções estudadas, o presente trabalho se destaca por integrar, em um único sistema, o gerenciamento de contratos administrativos, compras e estoque de gêneros alimentícios, voltado especificamente ao contexto da merenda escolar de uma instituição pública de ensino, utilizando tecnologias modernas como Django, Python, HTML, CSS, JavaScript e Bootstrap.

6 METODOLOGIA

A configuração de pesquisa se dá da seguinte forma:

Trata-se de uma Pesquisa de natureza aplicada, ou seja, tem objetivo a utilização de informações disponíveis para criar novas tecnologias e resolver problemas específicos. Sua abordagem é qualitativa, tendo em vista que o que será avaliado é o atendimento integral das necessidades dos servidores e colaboradores do *campus* Macau do IFRN em relação às ferramentas de fiscalização de contratos de alimentação. E quanto aos procedimentos, o trabalho se configura como pesquisa-ação, pois o pesquisador, além de compreender, tem como objetivo intervir na situação, na intenção de sanar determinado problema (GERHARDT; SILVEIRA, 2009).

6.1 INTERESSADOS

Podemos classificar os beneficiados com esse trabalho em duas categorias, os beneficiados de forma direta e indireta. Os beneficiados de forma direta são:

I. Os servidores fiscais de contrato: Com a possibilidade de gerenciar os contratos, empenhos, compras e estoques na mesma ferramenta, será muito mais fácil planejar e executar suas demandas. A redução da dependência de planilhas manuais e da comunicação constante para obter dados operacionais liberará tempo para focar em outras atividades do setor;

II. Os colaboradores terceirizados: Com o sistema disponível para acesso pelos profissionais da cozinha, eles poderão alimentar os dados de consumo dos itens de forma mais organizada. Além disso, a visibilidade do estoque permitirá que planejem a utilização dos alimentos de forma mais eficaz, evitando a falta de ingredientes essenciais para o cardápio do dia.

Já os beneficiados de forma indireta são:

I. Os alunos: Os alunos são beneficiados de forma indireta visto que são público alvo do fornecimento da merenda escolar. A boa gestão dos contratos de alimentação escolar garante que não falte refeição escolar adequada para os alunos;

II. A instituição de ensino: Com o controle de estoque aprimorado, haverá uma otimização das compras, evitando excessos ou faltas, e uma consequente redução de desperdícios e

custos desnecessários. A maior organização e a disponibilidade de relatórios detalhados contribuirão para um melhor planejamento orçamentário e para o cumprimento das normas de controle interno e externo.

6.2 REQUISITOS

O objetivo do trabalho é construir uma ferramenta que auxilie o gerenciamento de contratos de fornecimento de insumos para merenda escolar de forma que:

I. Seja possível cadastrar, alterar e consultar informações sobre os contratos como: empresa, itens disponíveis, número do contrato e vigência;

II. Seja possível cadastrar, alterar e consultar informações sobre os empenhos disponíveis, como o número de identificação do empenho, saldo disponível, empresa vinculada e contrato vinculado.

III. Seja possível cadastrar, alterar e consultar itens disponíveis nos contratos, fazer pedidos a partir desses itens e salvar as informações de pedidos, que ficarão disponíveis para o servidor fiscal de contratos.

IV. Seja possível cadastrar, alterar e consultar informações sobre os pedidos, como data de criação, valor total do pedido e entrega.

V. Seja possível cadastrar, alterar e consultar informações sobre os processos de pagamento abertos pelo servidor fiscal de contrato, a fim de ter controle sobre quais pedidos já foram pagos para seus respectivos fornecedores.

VI. Seja possível exibir interfaces distintas, com restrições de acesso que variam conforme o perfil do usuário. Desse modo, um usuário anônimo, ou seja, sem autenticação, terá acesso apenas a consultas. Já os funcionários terceirizados poderão realizar consultas, cadastrar pedidos, confirmar recebimento de mercadoria e alterar o estoque, enquanto o servidor fiscal de contrato terá permissão livre no sistema.

6.3 DADOS

A etapa de coleta de dados se deu de forma presencial através de entrevistas com os servidores do campus Macau do Instituto Federal diretamente envolvidos com os contratos de gêneros alimentícios, que foram a coordenadora do setor de assistência estudantil (COAES) e o diretor de administração do campus.

As entrevistas tinham como objetivo principal fazer o levantamento de requisitos funcionais para o sistema, com perguntas a respeito dos problemas relacionados à fiscalização dos contratos de aquisição de merenda escolar, e como poderíamos resolver esses problemas através do sistema proposto. Enquanto a coordenadora da COAES apontou as principais necessidades do setor em relação aos contratos, o diretor de administração auxiliou retirando dúvidas a respeito da legislação e organização dos contratos administrativos.

6.4 INSTRUMENTOS

Vamos abordar quais tecnologias e métodos foram usados para construir nosso sistema. As próximas seções detalham todas as ferramentas e recursos utilizados.

6.4.1 Linguagens de Programação

Além dos frameworks Django e Bootstrap, a ferramenta foi criada utilizando algumas linguagens de programação trabalhando em conjunto. Para construção da interface *web* foram utilizadas as linguagens: HTML para definir a estrutura das páginas, CSS para descrever a aparência e JavaScript para adicionar funcionalidades. Já para o backend (lado servidor) foi utilizada a linguagem Python para controlar todo o funcionamento da estrutura cliente-servidor, como rotas e formulários e consultas ao banco de dados.

6.4.2 Ferramentas Utilizadas

Como ambiente de desenvolvimento para a aplicação, ou simplesmente IDE (*Integrated Development Environment*), foi utilizado Visual Studio Code, que é um editor de código aberto, e que suporta quase todas as principais linguagens de programação, além de possuir recursos de inteligência artificial integrados (MICROSOFT, 2025). E para o armazenamento do banco de dados, foi utilizado o SQLite.

7 FERRAMENTA PROPOSTA

7.1 ARQUITETURA DA FERRAMENTA

A ferramenta em desenvolvimento tem o propósito de informatizar integralmente os processos de organização, execução e registro das compras de insumos destinados à merenda escolar realizadas no IFRN Campus Macau. Trata-se de um sistema *web*, desenvolvido com o uso do framework Django, que visa oferecer maior eficiência, confiabilidade e controle sobre as atividades relacionadas à gestão de compras institucionais.

A solução adota a arquitetura cliente-servidor, possibilitando uma interação ágil e eficiente com o usuário por meio da camada de apresentação (*front-end*), enquanto o processamento das regras de negócio, bem como o armazenamento e gerenciamento dos dados, é realizado na camada de aplicação (*back-end*). Essa abordagem contribui para a organização do sistema e para a melhoria do desempenho geral da aplicação.

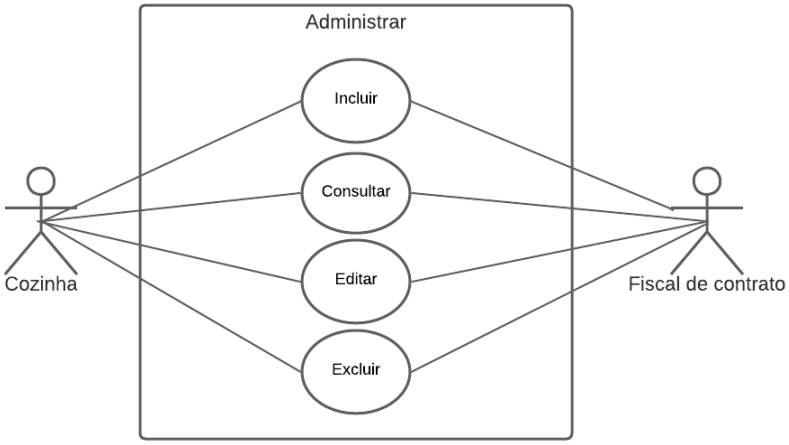
Embora se trate de um sistema de escopo específico, sua concepção segue os princípios da separação de responsabilidades, garantindo que as camadas de interface, lógica de negócio e acesso a dados sejam devidamente separadas. Essa característica favorece a manutenibilidade, a escalabilidade e a evolução futura da solução. As interações do usuário são tratadas de forma dinâmica, permitindo a utilização de JavaScript para validações imediatas e respostas no lado do cliente, enquanto o Django é responsável por assegurar a integridade, a consistência dos dados e os mecanismos de segurança da aplicação

7.2 DIAGRAMA DE CASO DE USO

Com base nas entrevistas realizadas para elicitação dos requisitos funcionais do sistema foi elaborado o diagrama de caso de uso presente na Figura 2. Esse diagrama documenta o que o sistema faz do ponto de vista do usuário. Em outras palavras, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema (DEVMEDIA, 2012). Neste diagrama não nos aprofundamos em detalhes técnicos que dizem como o sistema faz.

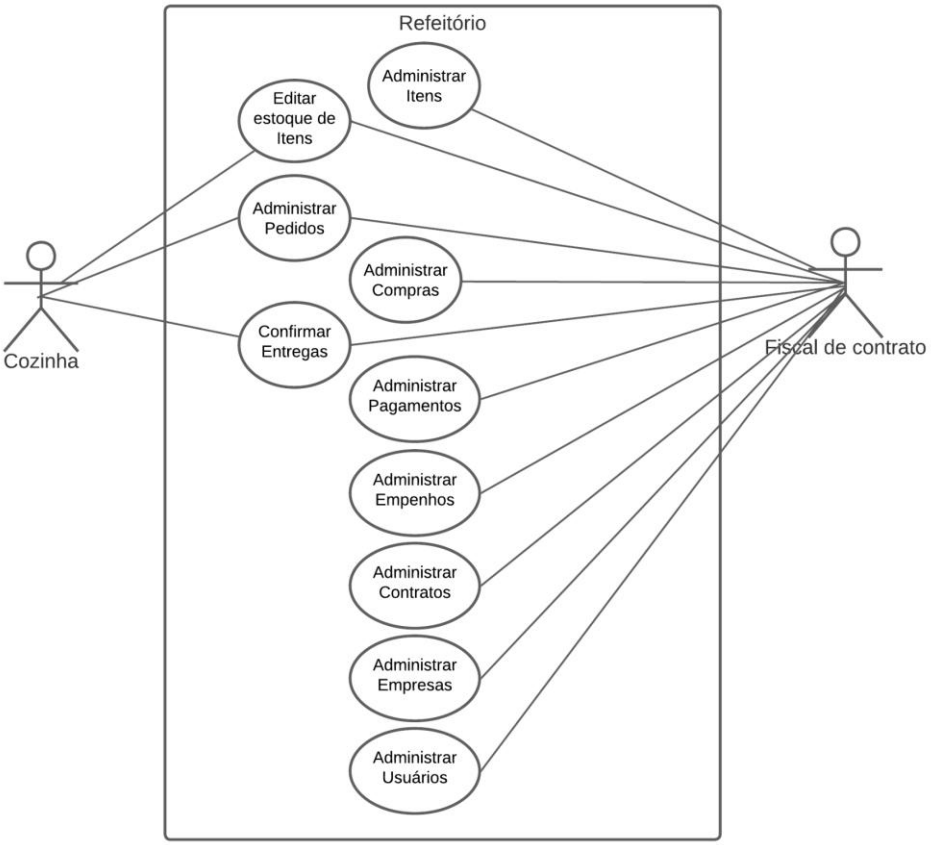
Complementar ao diagrama de caso de uso geral do sistema, foi elaborado também o diagrama de caso de uso Administrar, presente na Figura 1, que apresenta as ações possíveis dentro da opção Administrar, presente no diagrama geral.

Figura 1 - Diagrama de caso de uso Administrar



Fonte: Autoria Própria

Figura 2 - Diagrama de caso de uso do sistema

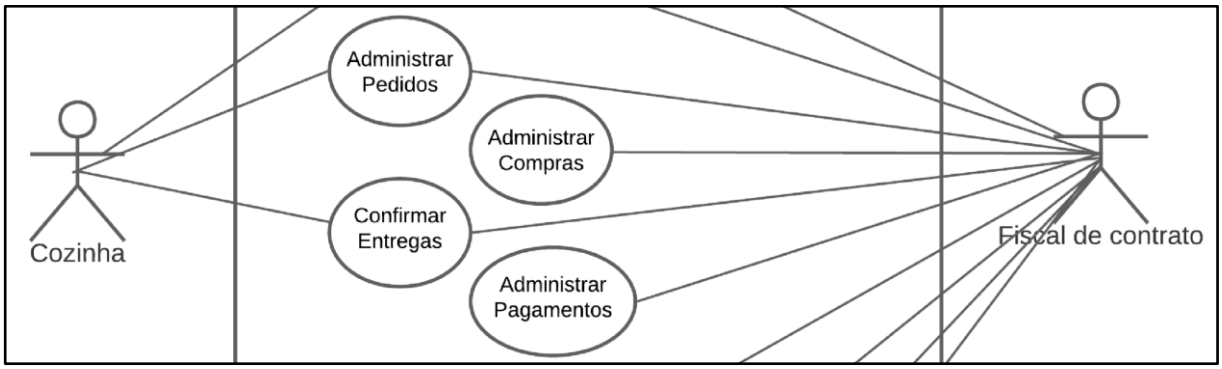


Fonte: Autoria Própria

As ações no diagrama de caso de uso acima foram organizadas de forma que, torna possível fazermos um recorte (Figura 3) das ações mais comuns do sistema, que acontecem de

forma sequencial. Essas ações são: Criação de pedidos de itens, o cadastro de compras através da confirmação do pedido, a confirmação de recebimento do material e o cadastro de pagamentos das compras.

Figura 3 - Diagrama de caso de uso do sistema - Recorte



Fonte: Autoria Própria

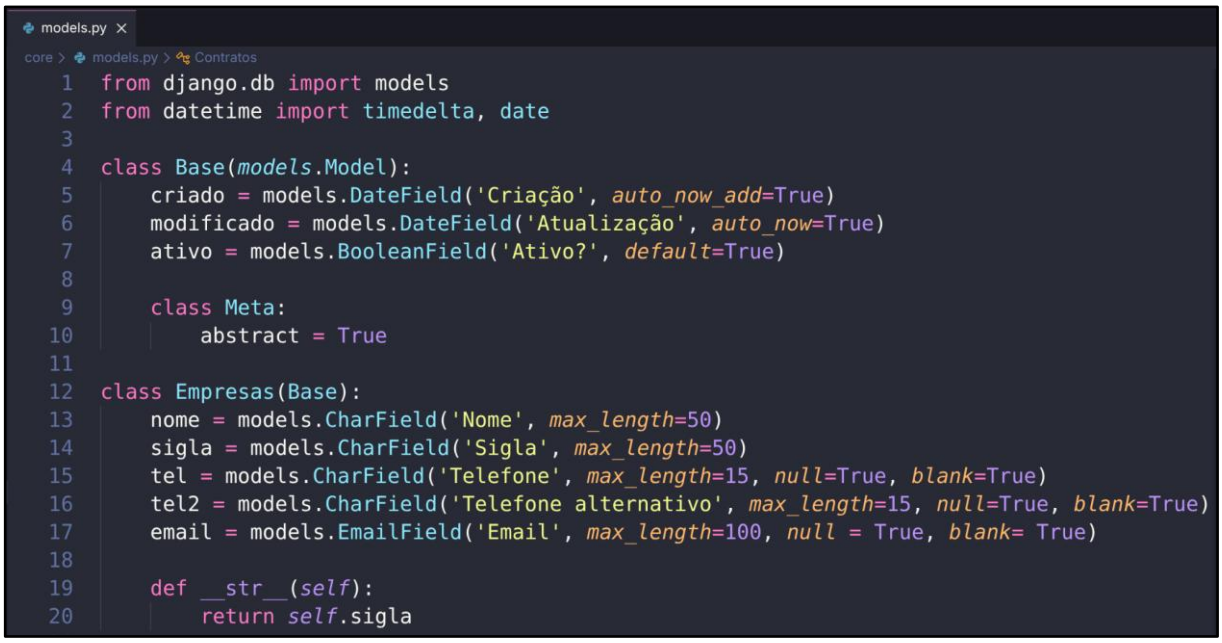
Dessa forma, para facilitar a compreensão do sistema a todos os usuários, temos o panorama geral de ações do sistema e um recorte das ações de rotina do sistema.

7.3 MODELAGEM DO BANCO DE DADOS

Como já dito anteriormente, o framework Django trabalha com Mapeamento Objeto-Relacional (ORM) nativa, com cada classe de objeto dando origem a uma tabela no banco de dados. Dessa forma, para criação do nosso banco de dados, criamos apenas nossas classes de objetos que devem existir no sistema e o Django trata de criar as tabelas de forma automática. A criação das classes é feita em um arquivo específico chamado models.py, e cada aplicação do nosso projeto tem seu próprio arquivo models.py.

Como é sabido, quando criamos classes de objetos em Python, definimos os atributos dessa classe e os métodos, e no framework Django isso funciona da mesma maneira. Na Figura 4 a seguir temos a captura de tela de uma parte do nosso arquivo de modelos.

Figura 4 - Recorte do arquivo models.py



Fonte: Autoria Própria

Na Figura 4 podemos observar, logo no início, as importações de bibliotecas que foram realizadas neste arquivo. Na sequência temos a classe abstrata que chamamos de Base, ela será herdada por todas as demais classes do arquivo. Dessa forma, todas as classes herdam seus atributos (criado, modificado e ativo), e herdam também os atributos e métodos de um *Model* do Django, como por exemplo os métodos .save() e .objects.

E por último, na classe Empresas, temos os atributos que o objeto deve possuir (nome, sigla, tel, tel2 e email) e temos também o conhecido método __str__(self). Podemos notar que nesta classe não há nenhum atributo com a indicação de chave primária. Quando isso acontece, o Django automáticamente adiciona um atributo na nossa tabela chamado id, este atributo será

preenchido com um número inteiro sempre que for criado um novo objeto do tipo Empresas, e esse atributo será a chave primária do objeto.

Na sequência estão todas as classes e atributos presentes no projeto.

-class Empresas(Base):

nome (CharField) - Nome da empresa

sigla (CharField) - Sigla da empresa

tel (CharField) - Número de telefone da empresa

tel2 (CharField) - Número de telefone da empresa

email (EmailField) - Endereço eletrônico

-class Contratos(Base):

numero (CharField) - Identificação do contrato

empresa (ForeignKey) - Chave estrangeira que faz referência a empresas

saldo (DecimalField) - Saldo total do contrato

data_inicio (DateField) - Data de início da vigência do contrato

data_fim (DateField) - Data de fim da vigência do contrato

arquivo (FileField) - Arquivo PDF do contrato

-class Empenhos(Base):

numero (CharField) - Identificação do empenho

saldo (DecimalField) - Saldo disponível no empenho

contrato (ForeignKey) - Chave estrangeira que faz referência a contratos

ano (IntegerField) - Ano de referência do empenho

-class Itens(Base):

nome (CharField) - Nome do item

valor (DecimalField) - Valor do item

qtd_estoque (IntegerField) - Quantidade em estoque do item

qtd_contrato (IntegerField) - Quantidade contratada no contrato

contrato (ForeignKey) - Chave estrangeira que faz referência a contratos

unidade (CharField) - Unidade de medida do item

-class Pedidos(Base):

valor_total (DecimalField) - Valor total do pedido

-class ItemPedido(Base):

pedido (ForeignKey) - Chave estrangeira que faz referência a pedidos

item (ForeignKey) - Chave estrangeira que faz referência a ites

quantidade (IntegerField) - Quantidade do item no pedido

valor_unitario (DecimalField) - Valor total do item no pedido

-class Compras(Base):

valor_total (DecimalField) - Valor total da compra

contrato (ForeignKey) - Chave estrangeira que faz referência a contratos

pedido (ForeignKey) - Chave estrangeira que faz referência a pedidos

status (CharField) - Situação da compra (pendente, entregue, cancelado, enviado, pago)

-class ItemCompra(Base):

compra (ForeignKey) - Chave estrangeira que faz referência a compras

item (ForeignKey) - Chave estrangeira que faz referência a itens

quantidade (IntegerField) - Quantidade do item na compra

valor_unitario (DecimalField) - Valor total do item na compra

-class Pagamentos(Base):

compra (ForeignKey) - Chave estrangeira que faz referência a compras

empenho (ForeignKey) - Chave estrangeira que faz referência a empenhos

num_processo (CharField) - Número do processo de pagamento no SUAP

saldo_utilizado (DecimalField) - Saldo utilizado do empenho

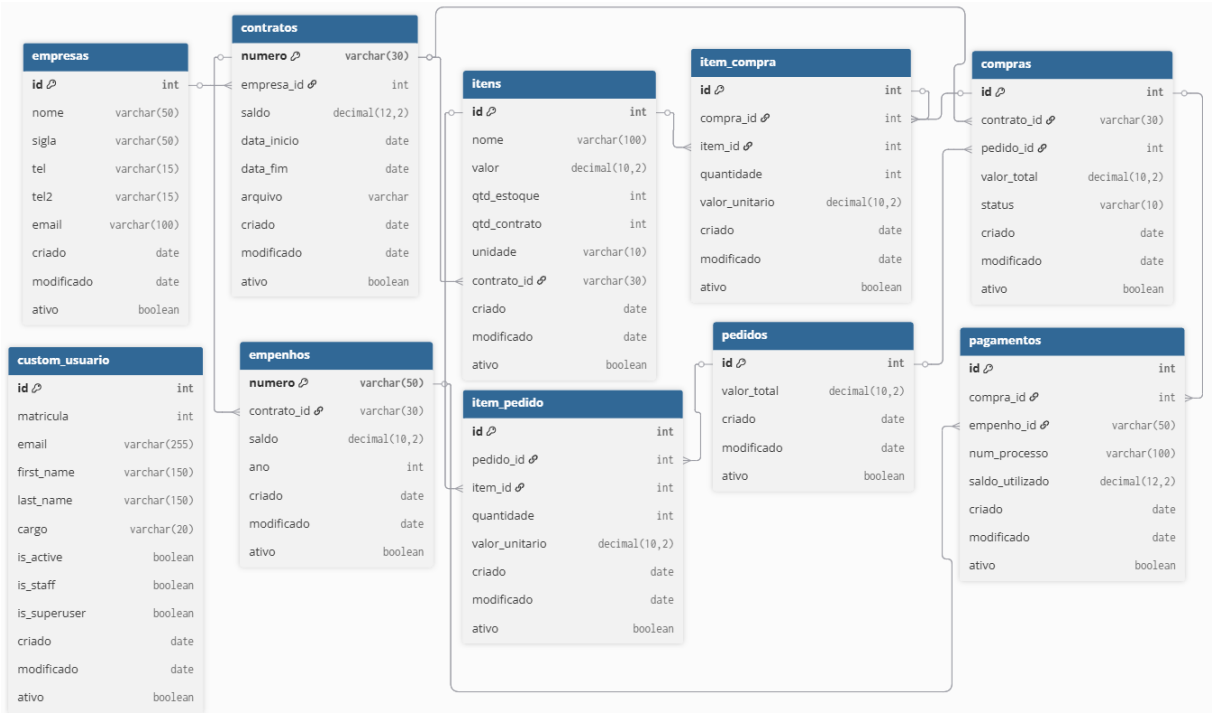
-class CustomUsuario(AbstractUser):

matricula(IntegerField) - Identificador do usuário

email(EmailField) - Endereço eletrônico

cargo(CharField) - Cargo do usuário dentro do sistema (cozinha, fiscal de contrato)

Figura 5 - Diagrama Entidade Relacionamento



Fonte: Autoria Própria

A classe Empresas é responsável por guardar as informações dos fornecedores. Ela tem como identificador o atributo id, que é gerado pelo Django, e seus demais atributos auxiliam na identificação e no contato com a referida.

A classe Contratos guarda as informações referentes aos contratos firmados entre o campus Macau do IFRN e os fornecedores. O atributo numero recebe o identificador, que é a chave primária do objeto, o atributo empresa é uma chave estrangeira para a empresa vinculada ao contrato. Além desses atributos ainda temos o saldo, que se refere ao valor total do contrato, as datas de início e fim da vigência do contrato que ficam nos atributos data_inicio e data_fim, e um atributo chamado arquivo tem a função de armazenar o contrato em formato pdf.

A classe Empenhos deve armazenar as informações referentes aos empenhos cadastrados para os contratos. Ela possui um atributo identificador chamado numero, que é a chave primária da classe, o atributo contrato é a chave estrangeira para o contrato ao qual o empenho está vinculado. O atributo saldo armazena o saldo disponível para uso no empenho, e o atributo ano guarda o ano de referência do empenho.

A classe Itens guarda informações sobre os itens pertencentes aos contratos. Ela possui um atributo de id como chave primária, um atributo contrato como chave estrangeira para o

contrato vinculado, e outros atributos com informações, como o atributo estoque, que guarda a quantidade do item que está disponível na cozinha do instituto.

A classe Pedidos tem a função de guardar as informações dos pedidos feitos pelo funcionário da cozinha ou pelo fiscal de contrato. Ela possui apenas os atributos de id como chave primária e valor_total que armazena o valor total do pedido.

A classe Compras armazena as informações das compras feitas a partir dos pedidos, porém possui alguns atributos adicionais. Além dos atributos id e valor_total, temos as chaves estrangeiras contrato e pedido para o contrato e pedido vinculados, e um atributo status para armazenar a situação da compra, que pode ser pendente, entregue, cancelado, enviado ou pago.

A classe Pagamentos salva as informações dos pagamentos cadastrados para as compras. A chave primária aqui também será o atributo id, gerado pelo Django, e temos duas chaves estrangeiras, uma para a compra de origem e uma para o empenho utilizado. Por fim os atributos num_processo responsável por salvar o número do processo de pagamento aberto no sistema SUAP e o saldo_utilizado que armazena o valor utilizado do empenho.

Observa-se que a entidade Contratos ocupa posição central no modelo, relacionando-se diretamente com Empresas, Empenhos, Itens e Compras, refletindo a realidade administrativa do refeitório institucional. As relações muitos-para-muitos entre Pedidos e Itens e entre Compras e Itens foram resolvidas por meio das entidades associativas ItemPedido e ItemCompra, permitindo o registro de quantidades e valores unitários.

A classe CustomUsuario armazena as informações dos usuários cadastrados no sistema. Além dos atributos visíveis listados acima, essa classe possui alguns atributos específicos herdados da classe AbstractUser. Os atributos de identificação first_name e last_name que guardam os nomes do usuário, e os atributos is_staff, is_superuser, is_active que tratam do acesso dos usuários ao sistema. Além desses, temos também o atributo password, que salva a senha do usuário. Não foi criada nenhuma relação entre a classe CustomUsuario e as demais pois não havia a intenção de registrar quais usuários realizaram as ações na plataforma.

8 RESULTADOS

O sistema desenvolvido para gerenciamento de compras e estoque para o refeitório do campus Macau do IFRN atende os objetivos gerais e específicos do trabalho, mais robusto do que foi pensado no início do projeto. Para mostrar um pouco sobre o funcionamento do sistema, foram feitas algumas capturas de tela do sistema em funcionamento na fase de desenvolvimento.

O site foi pensado para três tipos de usuários distintos, os usuários anônimos, servidores terceirizados que trabalham na cozinha do refeitório, e servidores efetivos que atuam na fiscalização dos contratos de alimentação escolar.

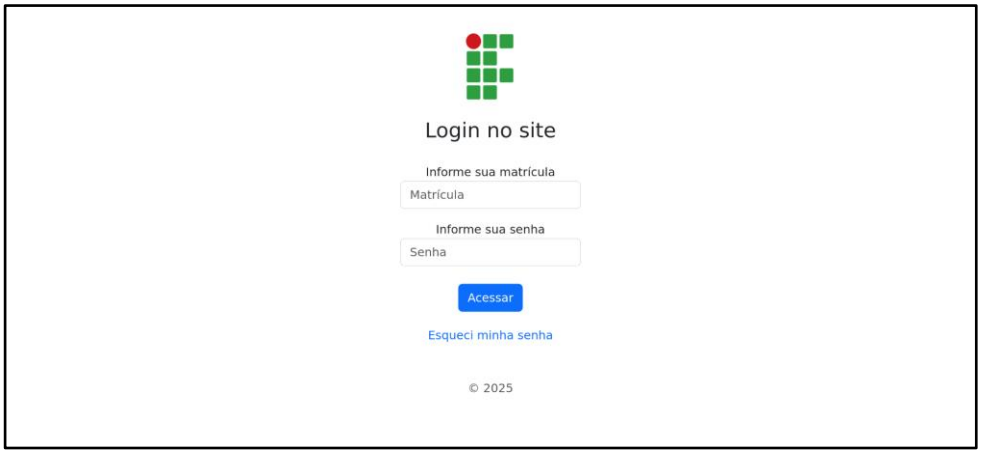
Para os usuários anônimos, o sistema permite que eles tenham acesso à páginas que deverão ser de conhecimento público. Como podemos observar na barra de navegação presente na Figura 6 os usuários podem ter acesso a lista de itens que fazem parte dos contratos da instituição, aos pedidos feitos pela cozinha para a administração, às compras que foram feitas pela administração para as empresas fornecedoras, e ao histórico de itens que foram comprados.

Figura 6 - Abas disponíveis para usuários anônimos

Fonte: Autoria Própria

Em todas as telas disponíveis para os usuários anônimos há apenas a possibilidade de consulta dos dados disponíveis. Para executar qualquer ação é necessário que o usuário faça *login* no sistema. Na Figura 7 a seguir está a página de *login* do site, que pode ser acessada por qualquer usuário que não esteja autenticado, através de um botão que fica no lado direito do cabeçalho das páginas.

Figura 7 - Página de login



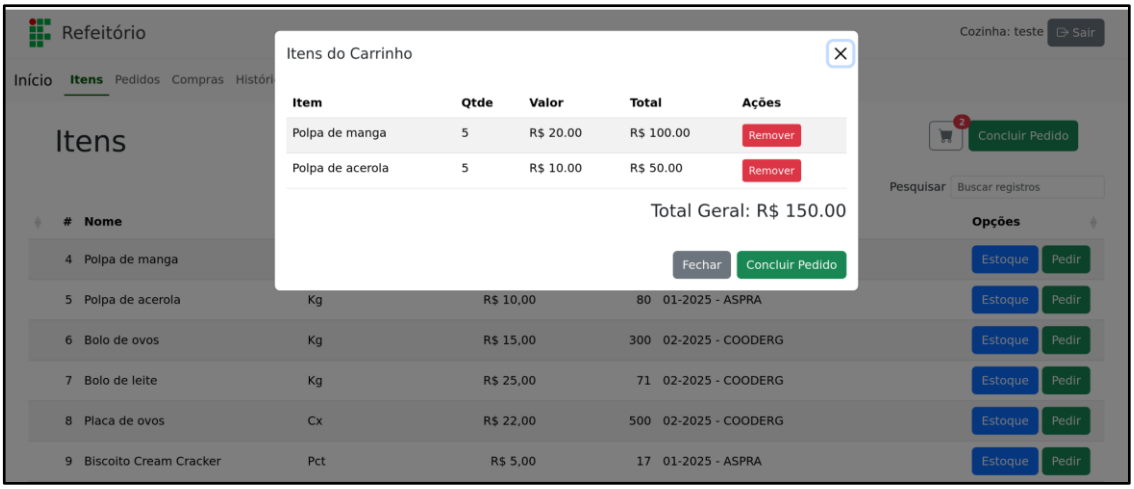
Fonte: Autoria Própria

Na página de login também é possível recuperar a senha através da opção “Esqueci minha senha”. Ao clicar na opção, o usuário é redirecionado para uma tela em que deve informar seu e-mail de cadastro e, caso o e-mail pertença a algum usuário cadastrado, será enviado um e-mail com link para que o usuário possa definir uma nova senha.

Para os funcionários terceirizados que trabalham na cozinha as telas disponíveis são exatamente as mesmas dos usuários anônimos, porém, agora as páginas possuem opções de interação com o sistema. Na página de itens é possível consultar o estoque de itens, registrar consumo e criar um pedido de itens. Os itens selecionados ficam armazenados em um carrinho, que aparece no início da página, junto de um botão para concluir o pedido. O carrinho é

armazenado na sessão do usuário, permitindo que o usuário possa alternar entre as abas do sistema sem que seu carrinho seja esvaziado, conforme exibido na Figura 8.

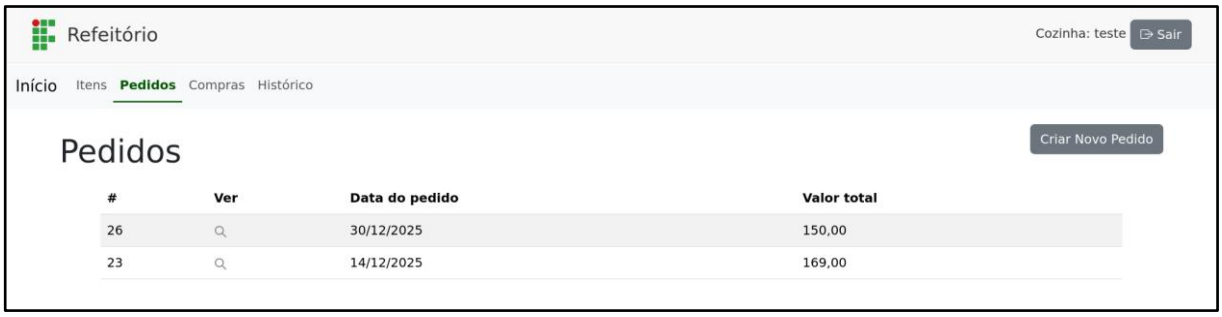
Figura 8 - Carrinho de itens



Fonte: Autoria Própria

Ao concluir o pedido é gerado um objeto do tipo pedido, que será avaliado pelo fiscal de contrato que, além de poder editar o pedido, se achar necessário, pode excluir o pedido ou prosseguir para gerar a compra. Caso o funcionário da cozinha necessite, ele também pode editar ou cancelar um pedido criado, abrindo o pedido que está presente na aba de Pedidos(Figura 9).

Figura 9 - Página de Pedidos para o funcionário da cozinha

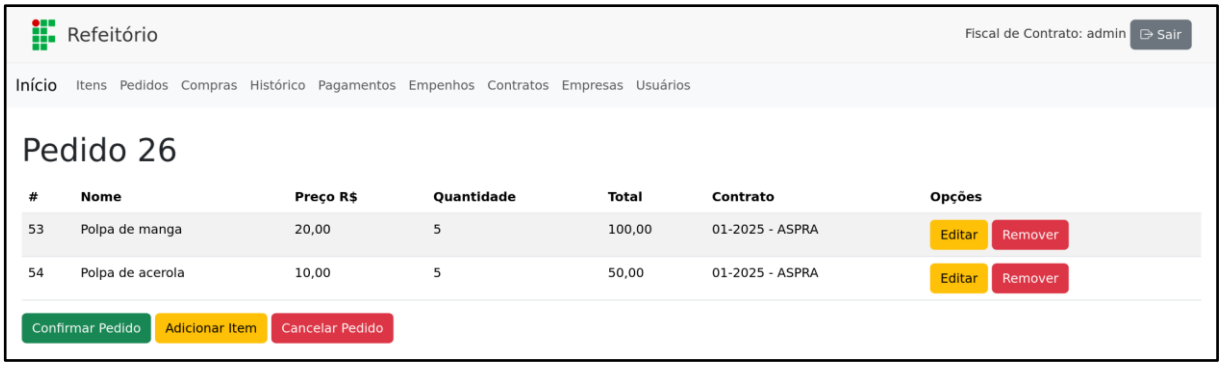


Fonte: Autoria Própria

O passo seguinte é a criação das compras. Clicando no botão “Confirmar Pedido”, que pode ser observado na Figura 10, o sistema automaticamente separa os itens presentes no pedido por contrato e cria compras distintas, de forma que, cada compra criada possua itens de apenas um contrato. A separação das compras por contrato é necessária pois, além de facilitar o envio dos pedidos para as empresas, facilita o controle dos saldos dos contratos e na organização dos

pagamentos. Caso não fosse feita, haveria problema na hora cadastrar os pagamentos, pois cada empenho é vinculado a um contrato, e só pode pagar pelos itens pertencentes a ele.

Figura 10 - Página de Pedido para o fiscal de contrato



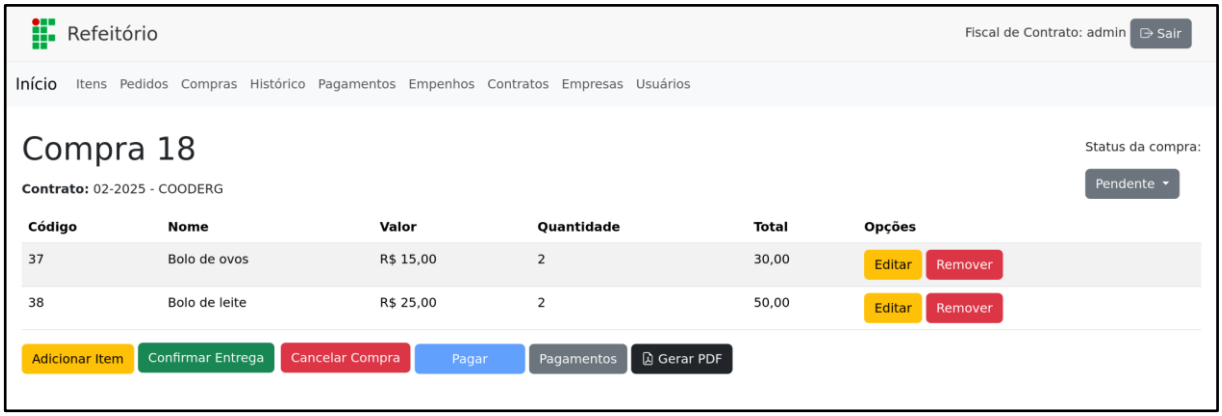
Fonte: Autoria Própria

A partir do pedido feito, e compra gerada pelo fiscal de contrato, existe mais uma função que pode ser executada pelo funcionário da cozinha, que é a conferência do material recebido. Na aba de compras (Figura 11), ao abrir uma compra, é possível tanto para o fiscal de contrato como para o funcionário que trabalha na cozinha, confirmar a entrega dos itens, corrigir as quantidades dos itens recebidos, adicionar itens que foram entregues sem terem sido solicitados e remover itens que não foram entregues.

Essa conferência e ajuste da compra ao que foi entregue é necessária, pois é a partir da compra que damos início ao registro de pagamento. A opção de adicionar item nesta página foi implementada para que permita somente a adição de novos itens que também façam parte do

mesmo contrato da compra, para que não seja possível juntar itens de contratos diferentes, que também ocasionaria um problema na hora do registro de pagamento.

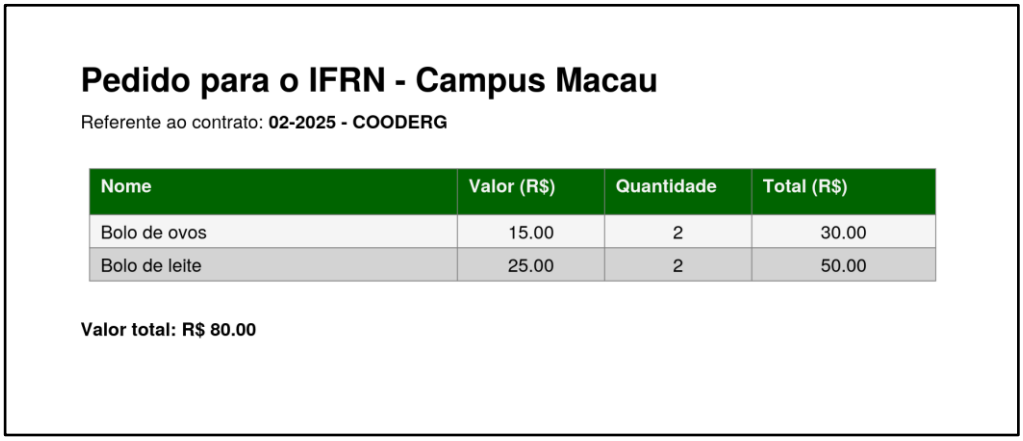
Figura 11 - Página de Compra para o fiscal de contrato



Fonte: Autoria Própria

Ainda na página de compra, o fiscal de contrato possui, além das opções citadas acima, as seguintes outras opções: cancelar compra, pagar, consultar os pagamentos que já foram feitos para a compra, gerar um arquivo PDF com os itens da compra para serem enviados para a empresa, e um botão para ver e alterar o *status* da compra. Na Figura 12 abaixo podemos conferir um exemplo de arquivo PDF gerado com a compra para ser encaminhado ao fornecedor.

Figura 12 - Exemplo de PDF gerado para envio de solicitações aos fornecedores



Pedido para o IFRN - Campus Macau

Referente ao contrato: **02-2025 - COODERG**

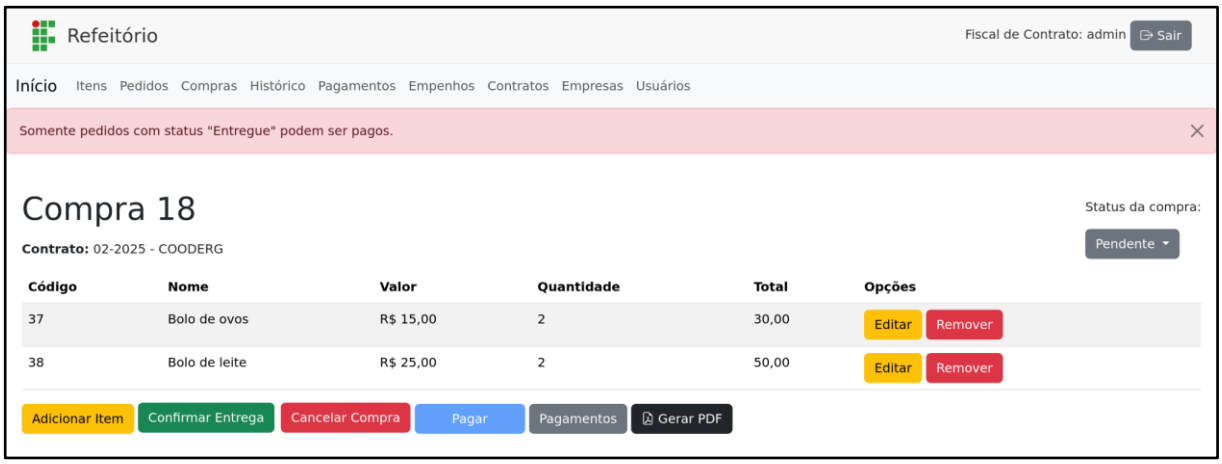
Nome	Valor (R$)	Quantidade	Total (R$)
Bolo de ovos	15.00	2	30.00
Bolo de leite	25.00	2	50.00

Valor total: R$ 80.00

Fonte: Autoria Própria

Quando uma compra é criada, por padrão o *status* dela é definido como pendente, este status pode ser alterado posteriormente pelo funcionário da cozinha, que pode confirmar a entrega, ou pelo servidor fiscal de contrato, que pode alterar o status para qualquer das demais opções (enviado, entregue, cancelado e pago). Para dar início ao processo de pagamento foi implementada uma uma restrição para que somente seja possível acessar a página de pagamento se a compra estiver com *status* de Entregue, essa condição foi incluída no sistema para evitar problemas de pagamentos indevidos ou acidentais. A Figura 13 abaixo mostra a resposta do sistema ao tentarmos pagar uma compra com *status* diferente de Entregue.

Figura 13 - Mensagem de erro ao tentar pagar compra não entregue



Fonte: Autoria Própria

Vale mencionar que, ao confirmar a entrega dos itens, o estoque dos itens da cozinha é automaticamente incrementado com os itens da compra. Após a confirmação da entrega,

seguimos para a etapa de pagamento clicando no botão pagar. Aqui podemos observar as seguintes características dos pagamentos:

1. Cada pagamento é referente a uma compra, mas cada compra pode ter mais de um pagamento;

2. Na página de pagamento (Figura 14), é possível selecionar mais de um empenho para pagar a compra, gerando mais de um objeto do tipo pagamento. Essa função é muito importante para quando se deseja dividir o pagamento entre empenhos, a exemplo da utilização de empenhos com pouco saldo disponível;

3. A página de pagamento exibe informações importantes sobre a compra como o contrato vinculado, o valor total da compra, o valor que já foi pago para a compra anteriormente, caso haja, e faz a soma dos valores informados nos campos em tempo real;

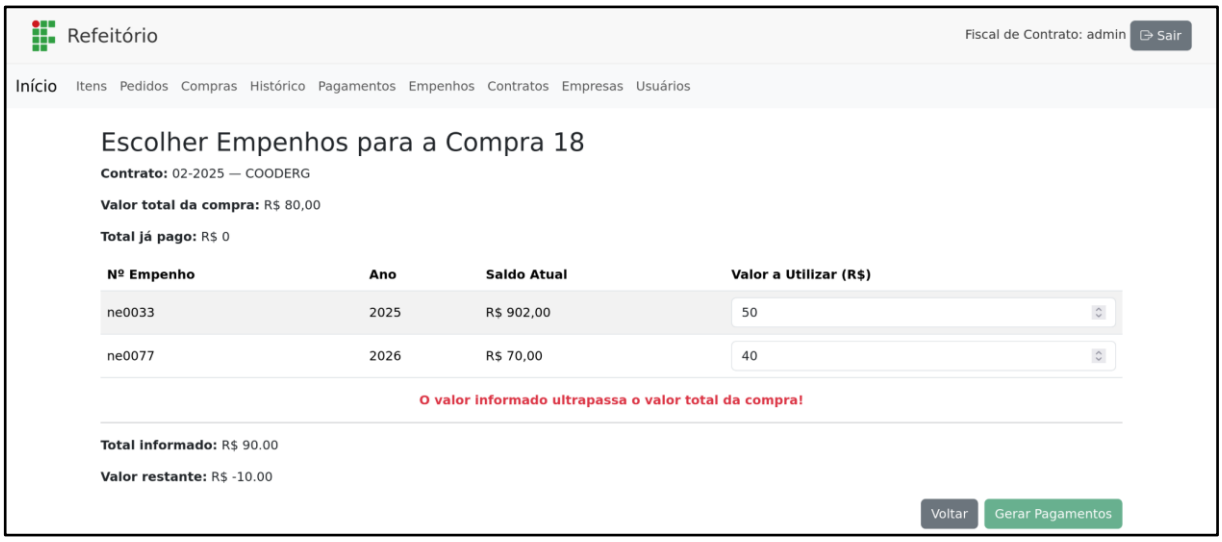
4. Foram implementadas validações de saldo no empenho, não permitindo que sejam cadastrados pagamentos utilizando empenhos com saldos insuficientes. O sistema entende que, caso o usuário digite um valor inferior ao total da compra, deve-se gerar um pagamento, e o valor restante será pago em momento posterior. E, caso o usuário digite um valor superior ao valor total da compra, ele deve impedir o prosseguimento. As capturas de telas referentes às validações citadas podem ser observadas nas Figuras 14, 15 e 16, com destaque nas informações importantes em vermelho.

Figura 14 - Erro ao tentar pagar a compra com saldo insuficiente no empenho



Fonte: Autoria Própria

Figura 15 - Erro ao informar valor superior para pagamento da compra



Fonte: Autoria Própria

Figura 16 - Pagamento de valor inferior ao valor total da compra



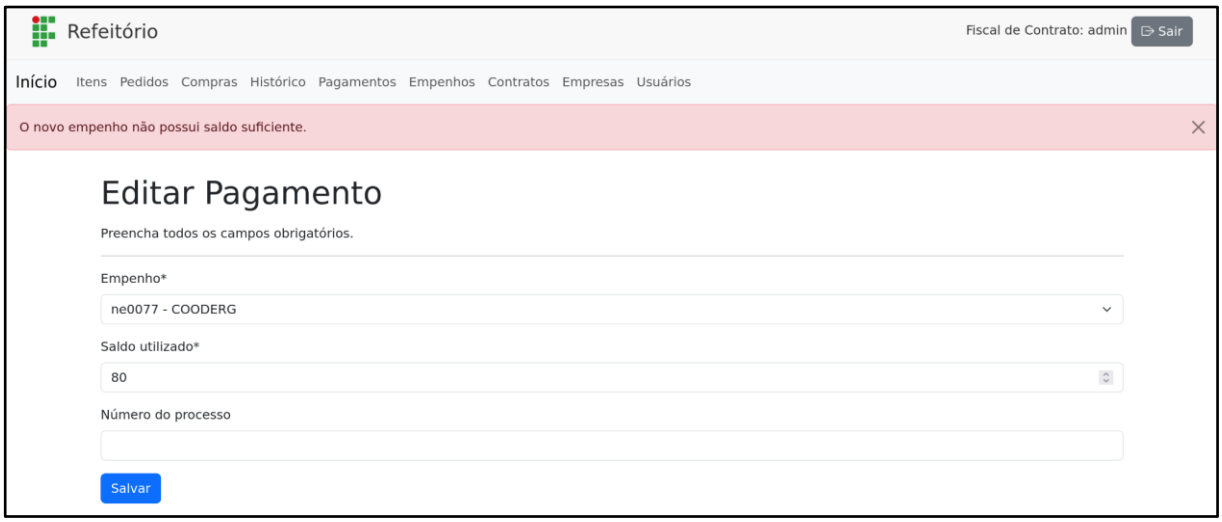
Fonte: Autoria Própria

Após a criação dos pagamentos, o saldo dos empenhos é atualizado de forma automática, e caso o pagamento seja editado ou excluído posteriormente, o valor do saldo é restituído novamente ao empenho.

É importante pontuar que também foram incluídas validações que evitam erros na hora de editar os pagamentos já criados. As validações impedem que sejam selecionados empenhos

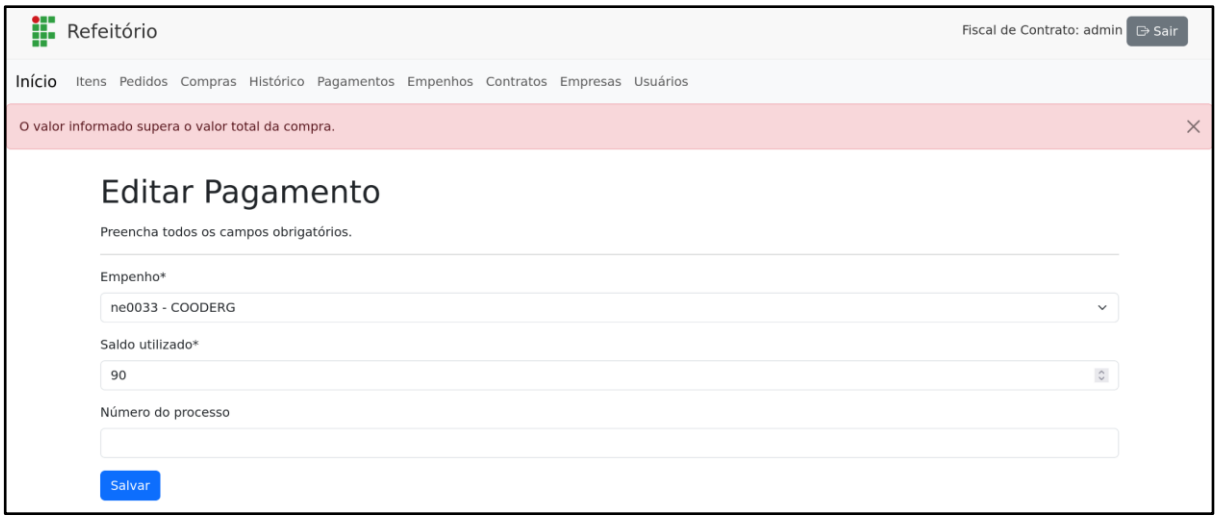
que tem saldo insuficiente para pagamento da compra, e que seja pago um valor superior ao da compra também, como podemos observar nas Figuras 17 e 18 abaixo.

Figura 17 - Erro ao selecionar empenho com saldo insuficiente - Página de editar pagamento



Fonte: Autoria Própria

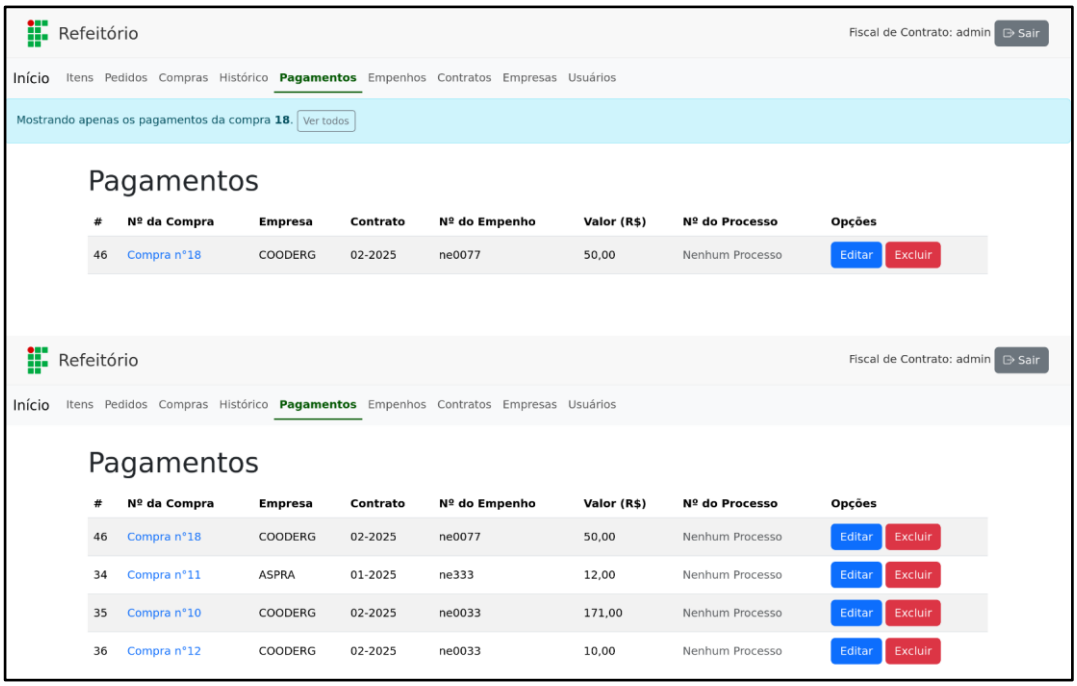
Figura 18 - Erro ao informar valor superior para pagamento da compra - Página de editar pagamento



Fonte: Autoria Própria

Quando o valor total da compra é pago, o status da compra é alterado automaticamente para Pago, enquanto isso não acontece ela permanece com *status* de entregue, mesmo com pagamentos parciais cadastrados. Para consultar se existe algum pagamento cadastrado para a compra basta clicar no botão de Pagamentos presente logo abaixo dos itens, na página da compra, presente na Figura 13. Podemos observar abaixo, na Figura 19, a página de pagamentos com e sem o filtro da compra aplicado.

Figura 19 - Página de pagamentos com e sem o filtro de compras aplicado



Fonte: Autoria Própria

Quando o *status* de uma compra é alterado para entregue ou para pago, os itens que fazem parte da compra são incluídos na página de histórico. Nessa página é possível fazer levantamento da quantidade de itens comprados com base no intervalo de tempo desejado, o que, em tese, auxilia no levantamento de relatórios e no planejamento de futuras compras. A página também permite que a busca seja exportada em arquivo de planilha .xls ou em formato pdf. Na Figura 20 abaixo podemos conferir a página de histórico de itens com a aplicação do

filtro de datas para encontrar nossa compra que está sendo usada como exemplo, que aconteceu no dia 07 de janeiro de 2026.

Figura 20 - Página de histórico com filtro de data aplicado



Fonte: Autoria Própria

As páginas de empenhos, contratos, empresas e usuários seguem o mesmo padrão. Todas possuem as funções que chamamos de CRUD, ou seja, criar, editar e excluir, e possuem alguma especificidade devido à natureza do objeto vinculado. Podemos destacar a tela de contratos, que possui o campo de arquivo com o endereço eletrônico dos arquivos de contratos no sistema de banco de dados. Além disso, possui também dois botões que exibem os empenhos e os itens cadastrados para o contrato (Figura 21).

Figura 21 - Página de contratos com mensagem de erro ao tentar excluir contrato



Fonte: Autoria Própria

Como também podemos observar na Figura 21, outra função que vale destacar é a restrição para exclusão de contratos, empenhos e de empresas que já tiverem pagamentos

cadastrados. Essa restrição preserva a integridade dos dados e previne erros por parte dos usuários.

É importante enfatizar que o projeto foi desenvolvido e testado apenas no ambiente de laboratório. O sistema não chegou a ser hospedado em um servidor real da instituição e testado pelo público alvo do projeto, o que limita a análise de sua efetividade real e fundamenta a necessidade de avaliações futuras com os usuários finais.

Em resumo, o sistema mostra que é possível integrar, organizar e informatizar todo o processo de compras de insumos para merenda escolar da instituição em uma única plataforma, dispensando o uso de planilhas para registros e de aplicativos de mensagens para comunicação entre a cozinha e a administração, e conferindo mais agilidade e padronização dos processos.

9 CONSIDERAÇÕES FINAIS

O sistema desenvolvido atingiu o objetivo geral do projeto, que era de desenvolver um sistema digital para gerenciar eficientemente as compras e o estoque de alimentos destinados à merenda escolar do campus Macau do IFRN. Esse resultado mostra que é possível planejar e desenvolver um sistema amplo para solução para problemas diferentes, que nesse caso são problemas de comunicação entre funcionários, gestão de estoque, gestão de compras e gestão de contratos administrativos.

Durante o desenvolvimento da aplicação, o framework Django mostrou-se uma ferramenta robusta e eficiente, atendendo de forma satisfatória aos requisitos de implementação, organização estrutural do código, facilidade de manutenção e segurança da aplicação. Seus recursos nativos, aliados à arquitetura, contribuíram significativamente para a construção de um sistema confiável, com separação clara de responsabilidades e potencial para evolução futura. A interface foi desenvolvida com foco na usabilidade e clareza das informações, buscando torná-la intuitiva e acessível a todos os perfis de usuários.

Destaca-se ainda que, após a realização de testes com os usuários finais e eventual validação institucional, o sistema desenvolvido poderá ser expandido para utilização em outros campi do IFRN, uma vez que sua estrutura foi pensada de forma flexível e adaptável, permitindo ajustes conforme as necessidades de cada unidade.

No entanto, o processo de desenvolvimento também apresentou desafios relevantes. As principais dificuldades encontradas estavam relacionadas à escassez de materiais didáticos sobre o framework, como aulas, cursos e conteúdos aprofundados, além do fato de sua documentação oficial estar disponível predominantemente em inglês. Essas limitações exigiram maior esforço de pesquisa e interpretação técnica, contribuindo, por outro lado, para o amadurecimento acadêmico e profissional do autor.

Como perspectivas de trabalhos futuros e melhorias na aplicação, destacam-se: a reestruturação do controle de acesso ao sistema, com a remoção do atributo “cargo” do modelo de usuário e a adoção de grupos e permissões para gerenciamento mais flexível dos níveis de acesso; a integração com a API do SUAP, possibilitando autenticação institucional e maior padronização no acesso; a implementação de ferramentas de acessibilidade e a criação de uma interface específica para o controle de estoque, visando aprimorar a visualização e o gerenciamento dos produtos disponíveis.

Por fim, embora o sistema tenha sido desenvolvido em um contexto acadêmico, observa-se que ele apresenta bases sólidas e condições técnicas adequadas para atuar como um importante aliado no cotidiano administrativo do campus. Sua adoção pode contribuir para a melhoria da organização, do controle e da transparência dos processos relacionados à aquisição e ao gerenciamento de insumos alimentícios, demonstrando o potencial da tecnologia da informação como ferramenta de apoio à gestão pública.

REFERÊNCIAS

BAGLIOTTI, Ian Rotondo; GIBERTONI, Daniela. **Reusabilidade no desenvolvimento de um sistema web utilizando o framework angular**. Interface Tecnológica, v. 17, n. 1, p. 192-204, 2020.

BALLOU, Ronald H. **Gerenciamento da cadeia de suprimentos: planejamento, organização e logística empresarial**. 5. ed. Porto Alegre: Bookman, 2006.

BENTO, Evaldo Junior. **Desenvolvimento web com PHP e MySQL**. [s.l.] Editora Casa do Código, 2021.

BOS, B. **Uma breve história do CSS até 2016**. Disponível em: <https://www.w3.org/Style/CSS20/history.html>. Acesso em: 2 dez. 2025.

BRASIL. **Lei nº 14.133, de 1º de abril de 2021.** Lei de Licitações e Contratos Administrativos. Diário Oficial da União: Brasília, DF, 1 abr. 2021.

BRASIL. **Lei nº 14.628, de 20 de julho de 2023.** Institui o Programa de Aquisição de Alimentos (PAA) e o Programa Cozinha Solidária e altera a Lei nº 14.133/2021. Diário Oficial da União: Brasília, DF, 21 jul. 2023.

CINTRA, Renato Fabiano et al. **Impacto da implantação de um sistema de informação gerencial na gestão de contratos públicos**: o caso do Hospital Universitário de Dourados/MS. Revista de Administração da UNIMEP, Piracicaba, v. 10, n. 2, p. 28-52, maio/ago. 2012. Disponível em: <http://www.redalyc.org/articulo.oa?id=273723607002>. Acesso em: 16 set. 2025.

DEITEL, Paul J.; DEITEL, Harvey M. **Ajax, rich internet applications e desenvolvimento web para programadores**. São Paulo: Pearson Prentice Hall, 2008.

DEMCHENKO, M. **Six huge tech companies that use python**: Does it fit your project? 2019.

DEVMEDIA. **O que é UML e Diagramas de Caso de Uso**: Introdução Prática à UML. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 15 dez. 2025

DJANGO. **Django documentation**. 2025. Disponível em: <https://www.djangoproject.com/>. Acesso em: 3 out. 2025.

EBAC, E. **O que é JavaScript e como funciona?** Disponível em: <https://ebaconline.com.br/blog/o-que-e-javascript-seo>. Acesso em: 20 dez. 2025.

FLANAGAN, David. **JavaScript: o guia definitivo**. 6. ed. Porto Alegre: Bookman, 2013.

GARCIA, Gelson Iezzi de Medeiros; QUEIROZ, Paulo Gabriel Gadelha. **Desenvolvimento de um sistema web para gerenciamento de compras de insumos de manutenção em contratos**

de prestação de serviços. Artigo (Graduação em Ciência da Computação) – Universidade Federal Rural do Semiárido (UFERSA), Mossoró, 2020.

GARCIA, Lorena. **Bootstrap:** o que é e como criar sites responsivos de forma rápida?. Disponível em: <https://www.alura.com.br/artigos/bootstrap?srsltid=AfmBOopr2-pki_lY_lzziLVL0kKj580sgObi120IoFoeEjqPFDSd51wk>. Acesso em: 6 dez. 2025

GASPAR, Tiago Caminha; YAGUINUMA, Cristiane Akemi; PRADO, Antonio Francisco do. **Desenvolvimento de aplicações colaborativas síncronas na web 2.0**. In: Anais do XV Simpósio Brasileiro de Sistemas Multimídia e Web. Fortaleza: SBC, 2009.

GERHARDT, Tatiana E.; SILVEIRA, Denise T.. **Métodos de Pesquisa**. 1.ed. Porto Alegre: Editora da UFRGS, 2009. ISBN: 978-85-386-0071-8

GHIMIRE, D. (2020). **Comparative study on Python web frameworks**: Flask and Django.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de informação gerenciais**. 15. ed. São Paulo: Pearson Education do Brasil, 2017.

MENEZES, N. N. C. **Introdução à Programação com Python:** Algoritmos e lógica de programação para iniciantes. 2. ed. 2014. ISBN: 978-85-7522-559-2.

MICROSOFT. **Visual Studio Code.** Disponível em: <https://code.visualstudio.com/>. Acesso em: 3 jan. 2026.

OLIVEIRA FILHO, Edison Cintra de. **Desenvolvimento de um sistema para o gerenciamento de estoque e pedidos de artigos religiosos**. 54 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG), Uruaçu, 2025.

OWENS, Mike. **The definitive guide to SQLite.** Berkeley: Apress, 2006.

QUEIROZ, Erick Fernando de. **Sistema para controle de estoque**. 43 f. Trabalho de Conclusão de Curso (Técnico em Manutenção e Suporte em Informática) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), Itapetininga, 2018.

RODRIGUES, William Costa. **Metodologia Científica.** FAETEC/IST. Paracambi, Rio de Janeiro. 2007.

SANTIAGO, C. et al. **Desenvolvimento de sistemas Web orientado a reuso com Python, Django e Bootstrap**. p. 97–120, 10 set. 2020. ISBN 978-65-87003-11-5.

SANTOS, J. A.; COSTA, M. R. **Sistemas de informação aplicados à gestão de estoques**. Revista Brasileira de Administração, v. 12, n. 2, p. 45–60, 2019.

SILVA, Leonardo Alves da. **Desenvolvimento de um sistema de gestão de estoque para plataforma web**. 83 f. Monografia (Graduação em Ciência da Computação) – Departamento de Informática, Universidade do Estado do Rio Grande do Norte (UERN), Mossoró, 2018.

SILVA, M. S. **Fundamentos de HTML5 e CSS3**. 1. ed. São Paulo: Novatec Editora. 2015. ISBN 978-85-7522-438-0.

SOARES, L. P. **Interfaces gráficas e os seus elementos lúdicos:** aproximações para um estudo comunicacional. Dissertação (Mestrado em Comunicação) – Universidade do Estado do Rio de Janeiro. Rio de Janeiro, 2008.

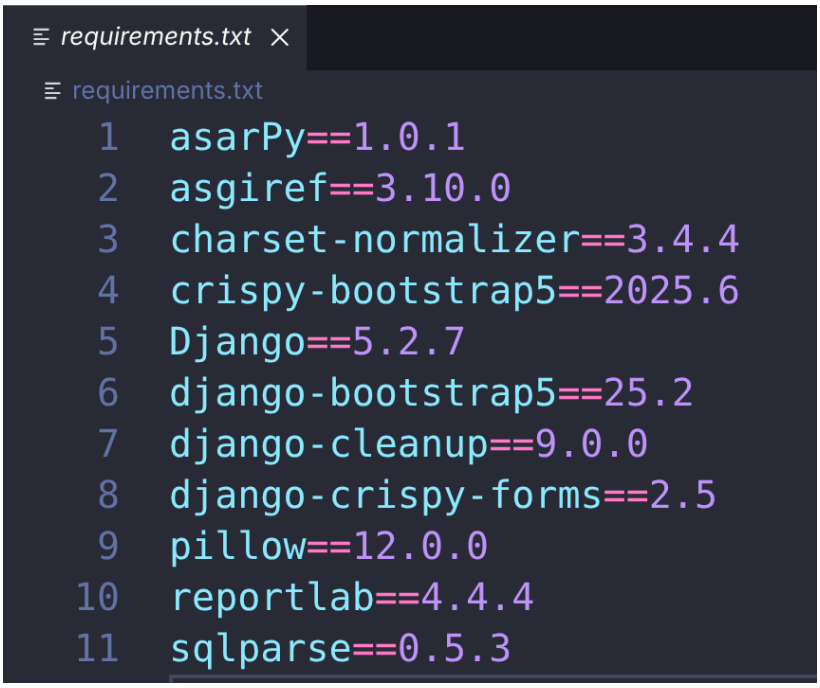
SOMMERVILLE, Ian. **Engenharia de Software** / Ian Sommerville ; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves ; revisão técnica Kechi Hirama. — 9. ed. — São Paulo : Pearson Prentice Hall, 2011.

SQLITE. **About SQLite.** 2023. Disponível em: <https://www.sqlite.org/about.html>. Acesso em: 13 nov. 2025.

URANO, L. **Django:** o que é, para que serve e um Guia desse framework Python. *Alura*, 06 nov. 2023. Disponível em: <https://www.alura.com.br/artigos/django-o-que-e-para-que-serve-guia-framework-python>. Acesso em: 3 out. 2025.

APÊNDICE A - DEPENDÊNCIAS INSTALADAS NO PROJETO

Figura A.1 - Arquivo requirements.txt



Fonte: Autoria Própria

APÊNDICE B - CAPTURAS DE TELA DO SISTEMA

Figura B.1 - Página de Início



Fonte: Autoria Própria

Figura B.2 - Página de Itens



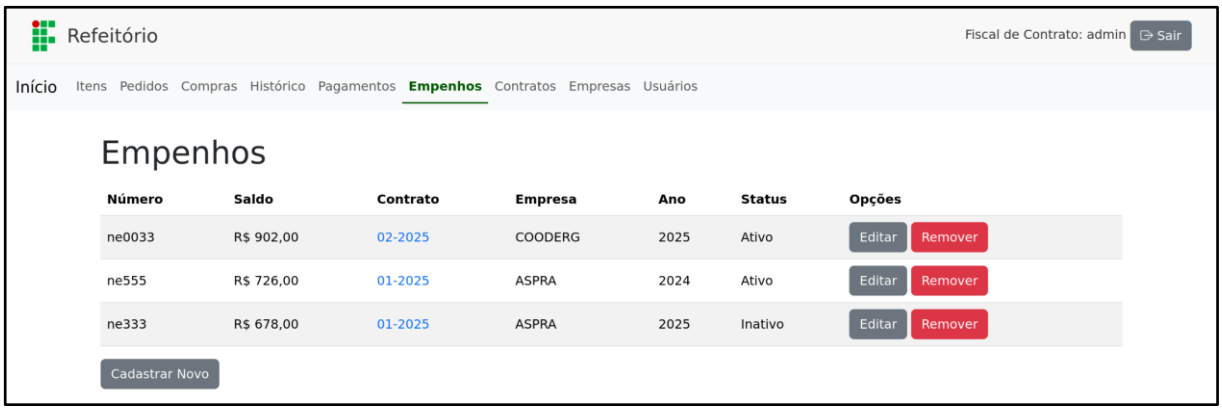
Fonte: Autoria Própria

Figura B.3 - Página de Compras



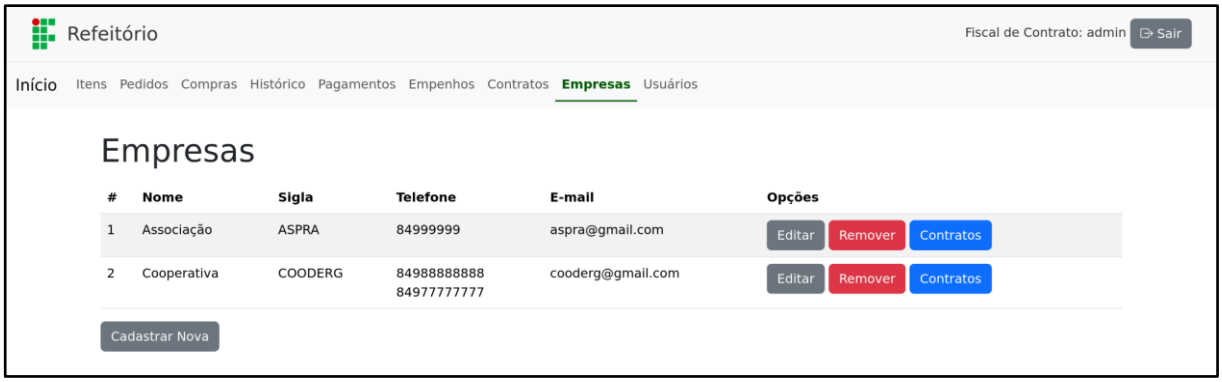
Fonte: Autoria Própria

Figura B.4 - Página de Empenhos



Fonte: Autoria Própria

Figura B.5 - Página de Empresas



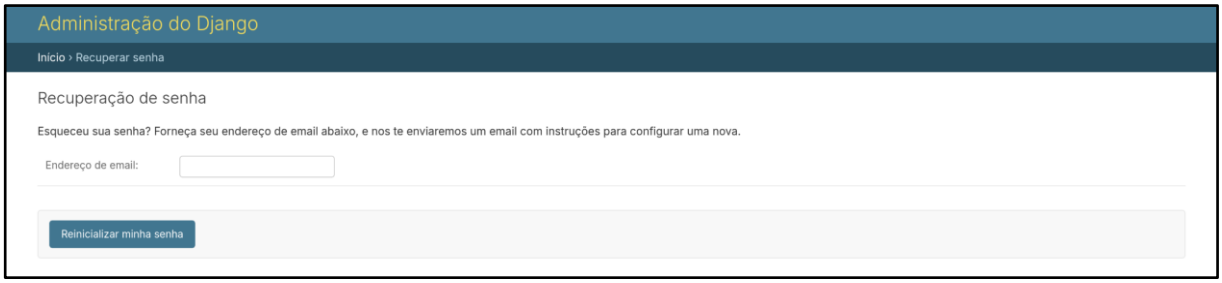
Fonte: Autoria Própria

Figura B.6 - Página de Usuários



Fonte: Autoria Própria

Figura B.7 - Página de recuperação de senha de usuário



Fonte: Autoria Própria