Software Untouched

1st André Luiz D. Cassimiro
2024056266
andreldc@ufmg.br

2nd João Francisco M. Carreiro
2024055839
joaofranciscomc@ufmg.br

3rd Vitória Helena F. Vieira Angelino
2024055731
vh2021950810@ufmg.br

***Abstract*—Este artigo apresenta a Untouched, um sistema de vendas desenvolvido com o microframework Flask. A aplicação abrange funcionalidades como controle de estoque, gerenciamento de produtos e lógica procedural de vendas. A arquitetura segue o padrão MVC e integra bibliotecas como Flask-SQLAlchemy, Flask-Login, Flask-Mail, Jinja2 e SQLite, garantindo organização, segurança e persistência de dados. Conceitos de Programação Orientada a Objetos foram incorporados para favorecer a modularidade e clareza estrutural. A lógica empregada contempla diferentes perfis de usuário, com comportamentos distintos para clientes e funcionários, por meio de abordagens polimórficas.A interface é caracterizada por templates HTML, permitindo a operação por diferentes tipos de usuários. Testes demonstraram a robustez da aplicação e sua adequação a micro e pequenos negócios.**

***Index Terms*—Programação Orientada a Objetos, Flask, Sistema de Vendas, SQLite, MVC, Flask-SQLAlchemy, Flask-Login, Templates HTML.**

I. Introdução

Nos dias atuais, o comércio eletrônico representa um dos principais canais de compra e venda, impulsionado pela praticidade, alcance e pela crescente digitalização das atividades comerciais. Em um cenário cada vez mais competitivo, marcado por consumidores exigentes e transações em tempo real, a adoção de soluções digitais eficientes para gerenciar processos de venda tornou-se essencial. A experiência do usuário, aliada à fluidez das interações e à confiabilidade do sistema, exerce papel central no sucesso de plataformas online, influenciando diretamente os índices de conversão e fidelização de clientes.

Segundo Gameiro, no artigo Web Performance e as plataformas de venda online [1], a performance e a usabilidade de uma aplicação web são fatores críticos na jornada do consumidor, sendo que atrasos superiores a três segundos podem resultar na desistência de mais da metade dos acessos móveis. Essa realidade evidencia a necessidade de sistemas que aliem desempenho técnico, boa estrutura organizacional e suporte a múltiplos perfis de usuários.

Neste contexto, apresenta-se a Untouched, um sistema de vendas desenvolvido com o microframework Flask, que busca oferecer uma solução robusta, acessível e extensível para o gerenciamento de vendas. A aplicação contempla funcionalidades como controle de estoque, cadastro de produtos, carrinho de compras e finalização de vendas. Além disso, incorpora diferenciação de perfis de acesso — clientes, funcionários e gestores — com comportamentos distintos modelados a partir de princípios da Programação Orientada a Objetos, utilizando herança e polimorfismo para organizar a lógica de forma clara e modular.

A arquitetura do sistema segue o padrão Model-View-Controller (MVC), garantindo separação de responsabilidades e facilitando a manutenção e a escalabilidade do projeto. Para isso, são utilizadas bibliotecas como Flask-SQLAlchemy, Flask-Login, Flask-Mail, Jinja2 e SQLite, que oferecem suporte à persistência de dados, autenticação, envio de notificações por e-mail e renderização dinâmica de páginas. A interface é baseada em templates HTML, permitindo uma interação fluida entre os usuários e o sistema.

II. Materiais e metodologia

O desenvolvimento da Untouched foi orientado por princípios de modularidade e clareza estrutural, utilizando o microframework Flask como base da aplicação. O projeto foi elaborado para atender às necessidades de micro e pequenos negócios, permitindo o gerenciamento de produtos, controle de estoque e processamento de vendas em um ambiente web. A construção do sistema foi fortemente influenciada pela Programação Orientada a Objetos (POO), com foco em reutilização de código, organização hierárquica e diferenciação de comportamento por tipo de usuário. A arquitetura adotada foi o padrão Model-View-Controller (MVC), que estabelece uma divisão estrutural do sistema em três componentes distintos: o *Model*, responsável pela representação e persistência dos dados; a *View*, encarregada da apresentação da interface e interação com o usuário; e o *Controller*, que atua como intermediário, recebendo as entradas do usuário, aplicando a lógica de negócio e coordenando a comunicação entre o *Model* e a *View*. Essa separação promove coesão e baixo acoplamento, facilitando a manutenção, testes e escalabilidade do software , segundo Leon Forte no artigo BUILDING A MODERN WEB APPLICATION USING AN MVC FRAMEWORK [2] tal separação, inclusive, favorece a interoperabilidade por se basearem em padrões amplamente aceitos na indústria.

O Untouched é um sistema web para gestão de vendas, projetado para oferecer uma navegação acessível e adaptada aos diferentes perfis de usuário. Na Figura 1, pode-se visualizar o fluxograma geral de funcionamento da aplicação. O processo se inicia com a tela de login ou cadastro, onde o usuário pode autenticar-se ou registrar uma nova conta. A partir da autenticação, o sistema direciona o usuário de acordo com seu tipo — cliente ou funcionário —, habilitando funcionalidades específicas. Clientes têm acesso ao catálogo de produtos, ao carrinho de compras e à finalização da compra. Funcionários, por sua vez, acessam funcionalidades administrativas, como o cadastro de produtos e a realização de vendas em nome

de clientes. Todo o processo se comunica com o banco de dados SQLite, garantindo a persistência e consistência das informações. Esse fluxo geral abrange as etapas principais do sistema, destacando as bifurcações lógicas e decisões que garantem o comportamento esperado da aplicação.

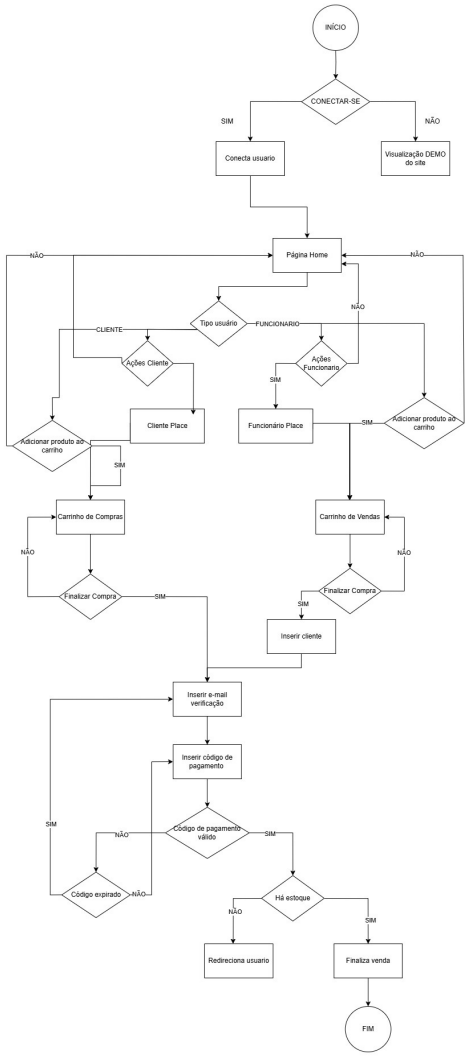

Fig. 1. Fluxograma geral do funcionamento do sistema Untouched.

A. Ferramentas e Técnicas utilizadas

O desenvolvimento do sistema contou com a aplicação de diversas técnicas e ferramentas fundamentais, cada uma desempenhando um papel essencial para garantir a modularidade, segurança, desempenho e clareza estrutural da aplicação. A seguir, são detalhadas as principais abordagens técnicas adotadas, ilustradas por meio dos fluxogramas que representam os processos centrais do sistema, evidenciando a integração entre programação orientada a objetos, frameworks Flask e bibliotecas auxiliares que suportam autenticação, persistência e apresentação dinâmica dos dados.

a) Controle de Fluxos Condicionais com Flask-Login e Flask-SQLAlchemy na Finalização de Venda: Para garantir a integridade do processo de venda, o sistema implementa controle rigoroso de fluxos condicionais durante a finalização, adaptando as ações conforme o perfil do usuário (cliente ou funcionário). Essa técnica utiliza o gerenciamento de sessões por meio do Flask-Login e a persistência segura dos dados via Flask-SQLAlchemy, usado como camada de mapeamento objeto-relacional (ORM) da aplicação. Essa ferramenta permite traduzir classes Python diretamente para tabelas do banco de dados, promovendo uma abordagem orientada a objetos para persistência de dados. Além disso, o SQLAlchemy oferece mecanismos robustos para controle de concorrência, como o suporte a colunas de versão e o uso do comando RETURNING para recuperar valores gerados pelo banco de forma eficiente [3]. O fluxograma da Figura 2 ilustra detalhadamente as decisões e etapas do processo, evidenciando como a lógica condicional assegura o comportamento esperado.

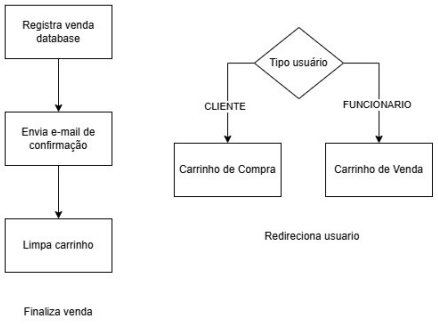

Fig. 2. Fluxograma da finalização da venda e redirecionamento por perfil de usuário.

b) Redirecionamento Condicional e Renderização Dinâmica com Jinja2 e Flask-Login na Navegação Inicial: A técnica de redirecionamento condicional após autenticação, combinada com a renderização dinâmica de templates pelo Jinja2, é aplicada para oferecer interfaces personalizadas de acordo com o tipo de usuário. O gerenciamento da autenticação é realizado pelo Flask-Login. Essa abordagem melhora a usabilidade e a experiência do usuário, adaptando o conteúdo conforme as permissões. O fluxo de navegação inicial é apresentado no fluxograma da Figura 3.

c) Especialização de Classes e Controle de Permissões Hierárquico para Modelagem das Ações do Funcionário : A organização das funcionalidades exclusivas para funcionários baseia-se no uso de herança e polimorfismo, técnicas fundamentais da programação orientada a objetos em Python. Isso permite a definição clara das operações, como cadastro de produtos e realização de vendas em nome de clientes, além da definição de uma hierarquia de usuários sendo possível manter o código reutilizável e modular. O fluxo das ações está representado no fluxograma da Figura 4, e no fluxograma da Figura 5 que demonstra o caminho lógico dessas operações.

d) Modelagem Relacional com Flask-SQLAlchemy para Gerenciamento de Produtos e Unidades: O sistema utiliza uma modelagem relacional robusta, facilitada pelo Flask-SQLAlchemy, para gerenciar produtos vinculados a unidades

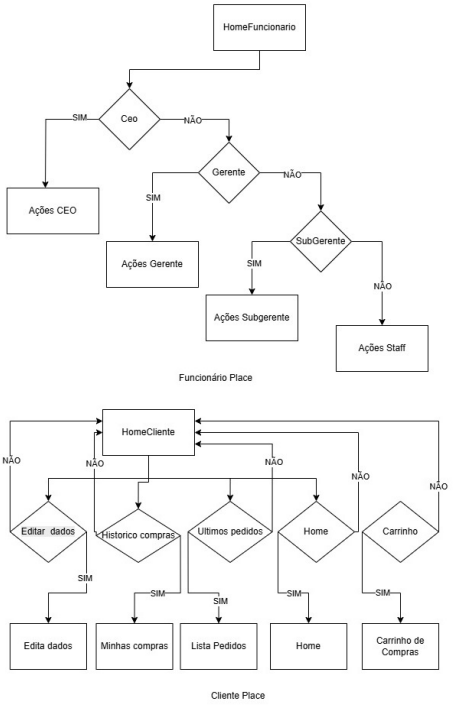

Fig. 3. Fluxograma das telas iniciais e ações para clientes e funcionários.

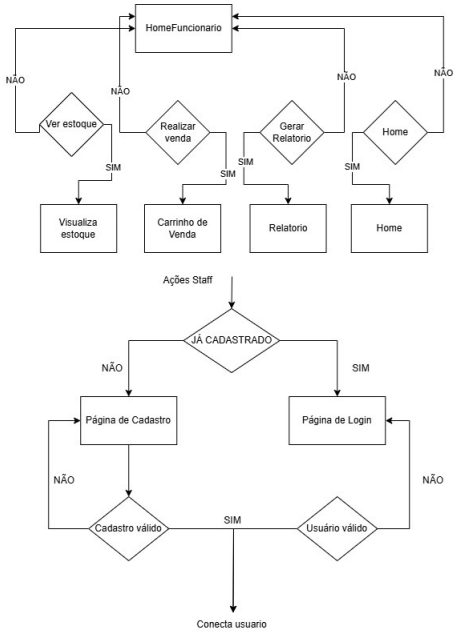

Fig. 4. Fluxograma das ações do funcionário e processos de cadastro.

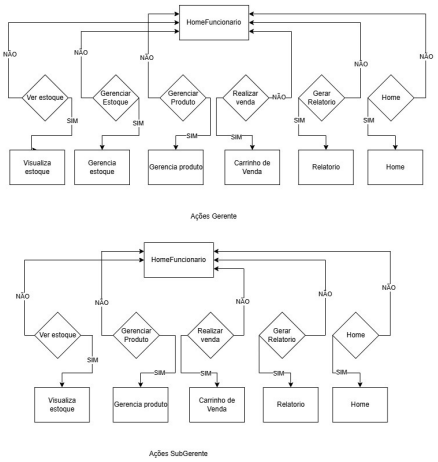

Fig. 5. Fluxograma das ações e permissões dos subgerentes e gerentes.

ou lojas específicas. Essa técnica possibilita a escalabilidade e o controle detalhado do inventário.

O fluxograma da Figura 6 ilustra o processo de cadastro e manutenção dos produtos e unidades.

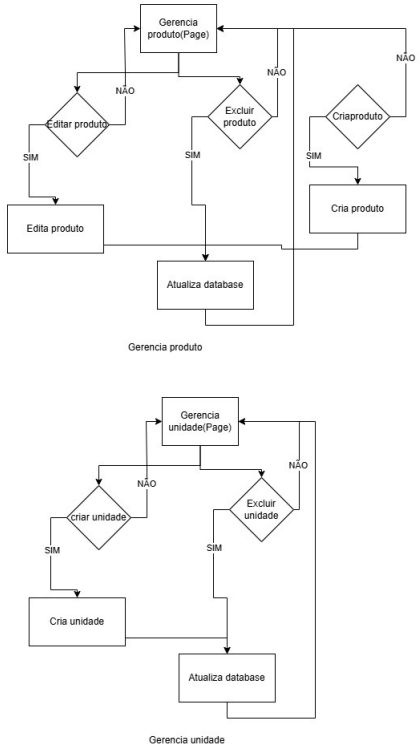

Fig. 6. Fluxograma do gerenciamento de produtos e unidades.

e) Consistência no Controle de Estoque com Flask-SQLAlchemy: Para manter a integridade das operações de estoque, o sistema utiliza transações controladas via Flask-

SQLAlchemy, garantindo atualizações seguras e atômicas em ambientes concorrentes. O fluxo detalhado está na Figura 25.

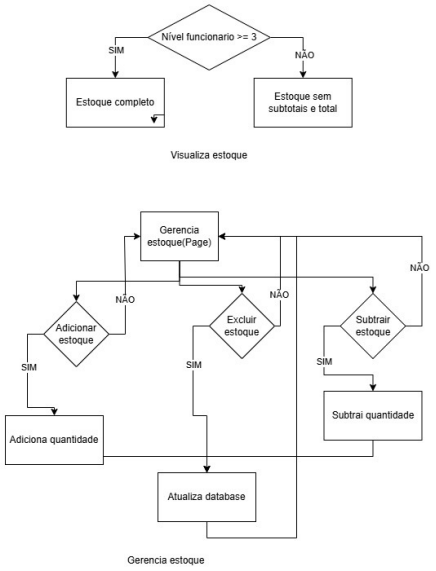

Fig. 7. Fluxograma do gerenciamento de estoque.

f) Modularização, Herança e Autenticação com Flask-Login no Gerenciamento Geral de Funcionários: Antes da execução das operações específicas, o sistema estabelece um gerenciamento geral de funcionários baseado em modularização e controle hierárquico com herança em POO. A autenticação e autorização são gerenciadas pelo Flask-Login, garantindo segurança e organização. O fluxograma da Figura 8 apresenta o panorama geral desse controle.

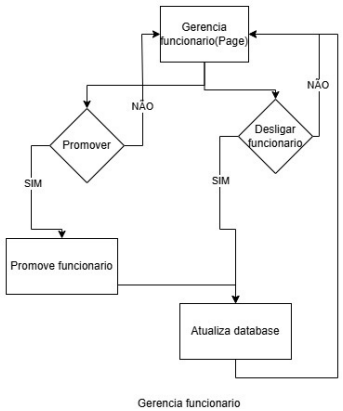

Fig. 8. Fluxograma geral do gerenciamento de funcionários.

O perfil CEO possui permissões amplas, com acesso total aos recursos do sistema. A segurança é garantida pela autenticação robusta e pelo controle de acesso provido pelo Flask-Login, aliado à modelagem orientada a objetos que assegura o isolamento e a proteção das funcionalidades sensíveis.

O fluxograma da Figura 9 exemplifica as ações exclusivas do CEO.

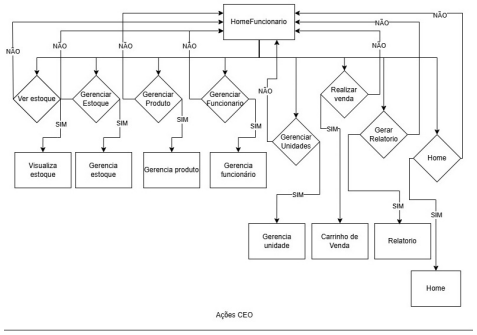

Fig. 9. Fluxograma das ações exclusivas do CEO.

g) Disponibilização do Código e Diagramas: Todo o código-fonte do sistema Untouched, assim como o diagrama UML representando a modelagem das classes e seus relacionamentos, está disponível publicamente no repositório do GitHub [4]. A disponibilização visa garantir a transparência do desenvolvimento e permitir que outros desenvolvedores possam estudar, adaptar ou contribuir com o sistema. Além disso, o diagrama de classes apresentado nas Figuras 10 e 11 foi construído com base nas diretrizes propostas por Ambler em [5], que recomenda práticas ágeis e simplificadas para a modelagem de classes em sistemas orientados a objetos.

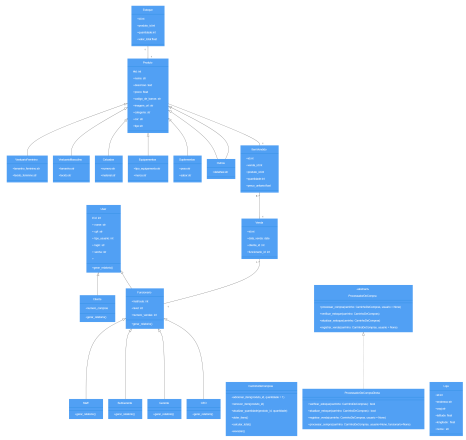

Fig. 10. Primeira parte do diagrama

Fig. 11. Segunda parte do diagrama

III. Resultados

A. Funcionalidades Implementadas

O desenvolvimento da aplicação Untouched resultou na implementação de diversas funcionalidades essenciais, que foram validadas com sucesso durante os testes manuais. Essas funcionalidades abrangem desde o acesso seguro à plataforma até o gerenciamento completo de vendas e estoque. Entre os principais recursos implementados, destacam-se:

- **Autenticação e Autorização:** Implementação de um sistema de login e gerenciamento de sessão com Flask-Login, garantindo acesso restrito às funcionalidades de acordo com o tipo de usuário (cliente, funcionário, sub-gerente, gerente ou CEO), como visto em 12, 13, 14 e 15.

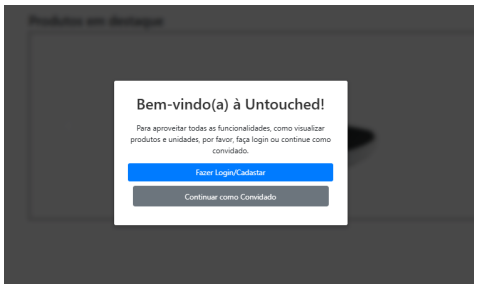

Fig. 12. Solicita autenticação usuário.

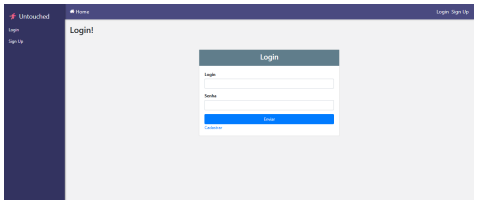

Fig. 13. Página de Login.

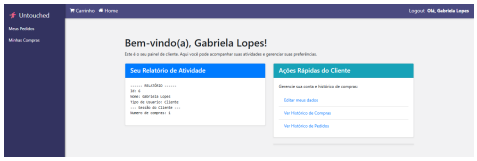

Fig. 14. Home Cliente.

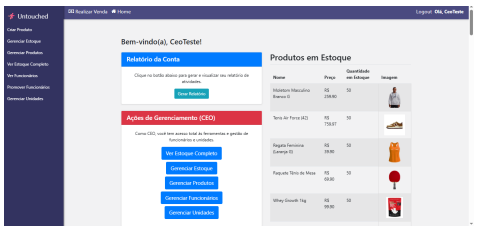

Fig. 15. Home Funcionário: CEO.

- **Carrinho de Compras:** Utilização da sessão Flask para armazenar os produtos temporariamente antes da finalização da compra, permitindo adicionar, remover e visualizar itens com total integração à lógica de estoque, como visto em 16, 17

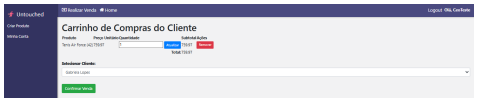

Fig. 16. Carrinho de vendas

Fig. 17. Carrinho de Compras

- **Finalização de Venda:** Processo adaptativo de finalização, onde clientes concluem suas próprias compras e funcionários podem registrar vendas em nome de clientes. Os dados são persistidos via Flask-SQLAlchemy e e-mails de confirmação são enviados com Flask-Mail. Para clientes, o processo é visto em 18, 20, 21 e 22. Enquanto que para funcionários é tido como 19, 20, 21 e 22.

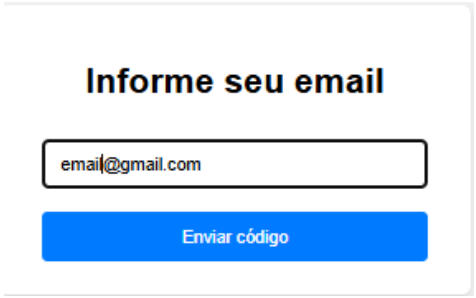

Fig. 18. Gera codigo cliente

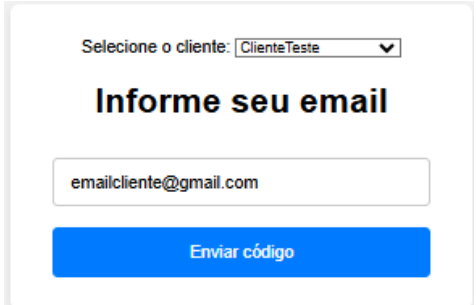

Fig. 19. Gera codigo funcionario

- **Gerenciamento de Produtos e Funcionarios:** Cadastro, edição e exclusão de produtos e ações de gestão de pessoas, com atualização automática do estoque e visibilidade segmentada por papel. Essa gerência pode ser visualizada na Figura 23 e 24.

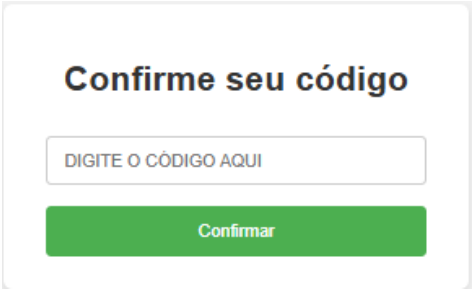

Fig. 20. Confirma código

Fig. 21. Email confirmado

- **Controle de Estoque:** Gerenciamento de entrada e saída de produtos em tempo real, com atualização automática ao finalizar uma venda e alertas sobre quantidade mínima, como visto na Figura 25.
- **Hierarquia de Perfis:** Estrutura orientada a objetos com herança para perfis distintos (Cliente, Funcionário, Subgerente, Gerente e CEO), com permissões e telas diferenciadas, essa dinâmica pode ser analisada pela difernça de ações disponíveis entre a Figura 15, correspondente ao CEO e a Figura 26, correspondente ao gerente.
- **Interface Dinâmica:** Templates HTML renderizados com Jinja2, adaptados ao perfil autenticado, promovendo uma navegação personalizada e fluida, a exemplo, cita-se a diferença entre as figuras 14 e 15, assim como a discrepância entre as figuras 15 e 26.

B. Fluxo de Dados

O fluxo de dados do sistema Untouched foi projetado para garantir a integridade das informações desde a entrada do usuário até a persistência no banco de dados. Abaixo, são descritos os principais processos:

- **Registro e Login de Usuário:**
  - O usuário preenche o formulário de cadastro ou login(Figura 13).
  - As informações são enviadas ao backend via requisição POST.
  - O backend valida os dados, autentica com Flask-Login e cria uma sessão.
  - O sistema redireciona o usuário de acordo com o perfil (cliente, funcionário, etc.)(Figuras 14 e 15).
- **Adição ao Carrinho:**
  - O cliente visualiza o catálogo de produtos e adiciona itens ao carrinho(Figuras 27 e [16 ou 17]).

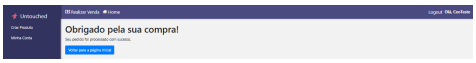

Fig. 22. Compra confirmada

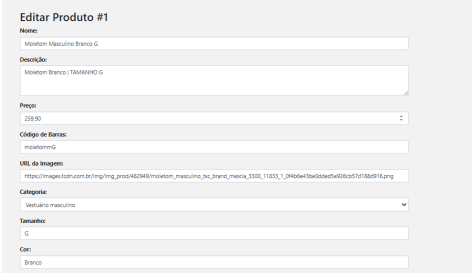

Fig. 23. Edição de produto

Fig. 24. Gerencia Funcionários

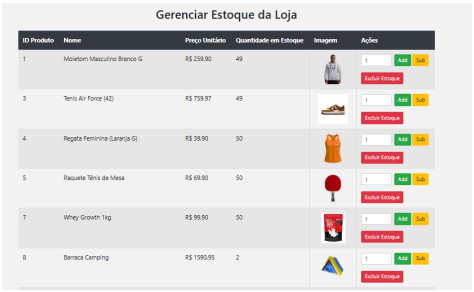

Fig. 25. Gerencia Estoque

Fig. 26. Home Funcionário: Gerente

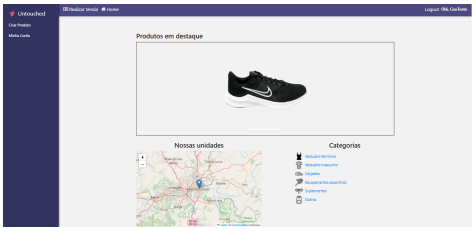

Fig. 27. Home

- Os dados dos produtos são armazenados na sessão do Flask.
- A interface exibe dinamicamente os itens adicionados com valores e quantidades atualizadas(Figuras 16 ou 17).

- **Finalização de Compra:**
  - O cliente (ou funcionário em nome de um cliente) confirma os produtos no carrinho.
  - Os dados são enviados ao backend, que cria um registro de venda e itens vendidos com Flask-SQLAlchemy.
  - A confirmação da compra é feita via inserção de código de validação, utilizando Flask-Mail (Figuras[18 ou 19], 20, 21 e 22).
  - O estoque é atualizado automaticamente com base nas quantidades vendidas.
  - Um e-mail de confirmação é enviado ao cliente com os dados da compra, utilizando Flask-Mail(Figura 28).

Fig. 28. Confirmação e Resumo da Compra

C. Arquitetura do Sistema

A arquitetura da aplicação segue o padrão Model-View-Controller (MVC), promovendo separação de responsabilidades e facilitando a manutenção. A estrutura é composta por três principais camadas:

- **Interface (View):**
  - Desenvolvida com HTML, CSS e Jinja2.
  - Possui templates dinâmicos para login, cadastro, listagem de produtos, carrinho, painel do funcionário e telas específicas para cada perfil.
- **Lógica de Negócio (Controller):**
  - Desenvolvida com Flask, organiza os fluxos de navegação e executa validações.
  - Cada operação está separada em controllers que controlam rotas, sessões, requisições POST e GET.
  - Aplica princípios de Programação Orientada a Objetos com herança e polimorfismo entre os perfis de usuário.
- **Persistência de Dados (Model):**
  - Utiliza Flask-SQLAlchemy para manipulação do banco de dados SQLite.
  - Tabelas incluem Usuário, Produto, Estoque, Venda, ItemVendido, Unidade e Funcionário.
  - Chaves estrangeiras são utilizadas para garantir integridade relacional.

D. Testes e Validação

Para garantir a robustez e funcionalidade do sistema Untouched, foram realizados testes manuais em diferentes cenários, com foco nos principais fluxos e nas permissões de acesso. Os testes cobriram:

- **Login e Autenticação:** Verificação de credenciais válidas e inválidas, tratamento de erros e redirecionamento adequado conforme o perfil(Figuras 29 e 30).

Fig. 29. Login Sucesso

Fig. 30. Login falhou

- **Carrinho e Estoque:** Adição e remoção de produtos, cálculo de totais, verificação de limites de estoque e bloqueio de finalização com estoque insuficiente(Figuras 31 e 32).

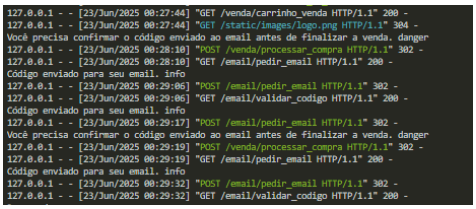

Fig. 31. Codigo verificado

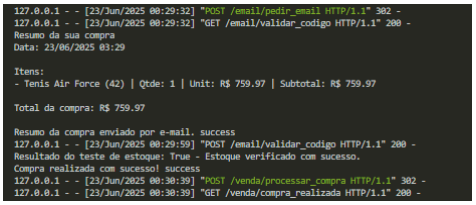

Fig. 32. Venda sucesso

- **Finalização de Venda:** Registro completo da venda com persistência de dados, atualização do estoque, identificação correta de cliente e funcionário e geração de registros no banco(Figuras 33 e 34).
- **Confirmação de Compra por E-mail:** Após a conclusão da venda, um e-mail de confirmação é enviado automaticamente para o cliente com os detalhes da compra. O teste verificou o envio correto, o conteúdo da mensagem e a entrega para diferentes provedores de e-mail(Figura 28).

E. Melhorias e Desenvolvimentos Futuros

Durante o desenvolvimento do sistema Untouched, foram identificadas diversas possibilidades de aprimoramento e expansão para versões futuras da aplicação:

Fig. 33. Verifica email

Fig. 34. Recebe Código

- **Aprimoramento do Modelo com SQLAlchemy:** Embora o sistema utilize os recursos fundamentais do SQLAlchemy para persistência de dados, futuras versões podem explorar mais profundamente suas funcionalidades avançadas, como relacionamentos complexos, callbacks, e mapeamentos customizados para tornar a estrutura do banco ainda mais robusta e escalável.
- **Integração com Sistemas de Pagamento:** Atualmente, a finalização de vendas não envolve transações financeiras reais. Uma melhoria futura prevê a conexão com gateways de pagamento (como MercadoPago, PagSeguro ou Stripe), permitindo que o sistema processe pagamentos online de forma segura e automatizada.
- **Criação de Aplicativo Móvel:** Como desdobramento natural, poderá ser desenvolvida uma versão mobile do Untouched, permitindo aos usuários (clientes e funcionários) realizarem compras, vendas e consultas diretamente de seus dispositivos móveis.

IV. Conclusão

Este artigo apresentou o Untouched, um sistema web para gestão de vendas desenvolvido com o microframework Flask, que integra diversas tecnologias com o intuito de atender às demandas específicas de micro e pequenos negócios. A aplicação engloba funcionalidades essenciais, tais como controle de estoque, gerenciamento de produtos e o processamento completo do ciclo de vendas.

O desenvolvimento do Untouched foi fundamentado em princípios rigorosos de Programação Orientada a Objetos, com ênfase em herança, encapsulamento e polimorfismo, os quais propiciaram uma estrutura hierárquica clara e promoveram a reutilização eficiente do código. A adoção da arquitetura Model-View-Controller (MVC) contribuiu para a segregação das responsabilidades entre interface, lógica de negócio e persistência de dados, favorecendo a manutenção, escalabilidade e testabilidade do sistema.

A seleção criteriosa das tecnologias, destacando-se o Flask para desenvolvimento web, SQLite para persistência local e bibliotecas como Flask-SQLAlchemy e Flask-Login para o gerenciamento de dados e autenticação, resultou em uma aplicação robusta, segura e com elevada usabilidade. O Untouched demonstrou-se eficaz na gestão das operações comerciais, facilitando o fluxo de trabalho tanto para clientes quanto para funcionários.

Em síntese, o sistema configura-se como uma solução acessível e flexível para o comércio eletrônico de pequeno porte, com perspectivas promissoras para futuras evoluções, que incluem a ampliação da integração com ferramentas externas e o suporte a pagamentos online, contribuindo para a modernização e automação dos processos comerciais.

Agradecimentos

Os autores agradecem à professora Gabriela Nunes Lopes pela disponibilidade, amplo conhecimento e pela ânsia em ensinar ao longo do desenvolvimento deste trabalho.Um agradecimento especial ao professor Marcelo Franco Porto pela disponibilidade e valiosas orientações sobre frameworks e suas vantagens.

References

[1] J. P. F. Gameiro, “Web Performance e as plataformas de venda online,” Dissertação de Mestrado em Engenharia Informática, Especialização em Engenharia de Software, Instituto Politécnico do Porto, Portugal, Outubro 2019.

[2] L. Forte, ”Building a Modern Web Application Using an MVC Framework,” Bachelor’s Thesis, Oulu University of Applied Sciences, Degree Programme in Business Information Technology, Spring 2016.

[3] M. Bayer, *SQLAlchemy Documentation*, Release 0.9.9, May 08, 2015. [Online]. Available: https://pdf-lib.org/images/pdf/sqlalchemy.pdf

[4] V. H. Frederico, “Untouched - Sistema de Vendas em Flask.” GitHub Repository. [Online]. Disponível em: https://github.com/Vitorelena/Untouched

[5] S. W. Ambler, ”UML Class Diagrams,” Agile Modeling, 2021. [Online]. Disponível em: https://agilemodeling.com/artifacts/classdiagram.htm. Acesso em: 23-jun-2025.