ESCOLA SENAI “HENRIQUE LUPO”
TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

ANA JULIA FIRMIANO DA SILVA
ANA LIVIA ROSA COUTO
ISABELLE ELOISE RUGNO FERREIRA
JHENIFFER RAYANNE DE ANDRADE
MARIA CECÍLIA ZACCARO THOMÉ

LOGCLASS

ARARAQUARA
2024

ANA JULIA FIRMIANO DA SILVA
ANA LIVIA ROSA COUTO
ISABELLE ELOISE RUGNO FERREIRA
JHENIFFER RAYANE DE ANDRADE
MARIA CECÍLIA ZACCARO THOMÉ

LOGCLASS

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do certificado de Técnico em Desenvolvimento de Sistemas, da Escola SENAI "Henrique Lupo".

Orientador(a): Prof. Alex Fernando Stocco e Prof. Ivo Conceição Neto.

ARARAQUARA
2024

Aos nossos pais,

Aos nossos avós,

Aos nossos irmãos.

AGRADECIMENTOS

Agradecemos, primeiramente, à Deus pelas bênçãos concebidas.

Agradecemos à nossa família por todo o suporte durante esses anos e por acompanhar de perto o nosso crescimento pessoal e intelectual, amparando-nos nos momentos difíceis e comemorando cada conquista.

Agradecemos aos nossos colegas e amigos que nos acompanharam durante o curso.

Agradecemos aos professores pelos anos de convívio, pelos conhecimentos trocados e conversas descontraídas.

RESUMO

O processo logístico é um conjunto de atividades essenciais que envolve a movimentação, armazenamento e gestão de materiais, desde a aquisição de matérias-primas até a entrega final dos produtos ao consumidor. Ele abrange diversas operações interligadas, como transporte, controle de estoques, processamento de pedidos e gestão da cadeia de suprimentos, e desempenha um papel estratégico no funcionamento das empresas. A eficiência logística é fundamental para que as organizações possam atender de forma ágil às demandas do mercado, reduzir custos operacionais e melhorar a competitividade. Uma gestão logística bem executada permite, por exemplo, a otimização dos recursos disponíveis, minimizando desperdícios e custos desnecessários, como o excesso de estoques ou rotas de transporte mal planejadas.

No contexto educacional, mais especificamente no curso de Aprendizagem Industrial de Auxiliar de Linha de Produção do SENAI, a importância de se preparar os alunos para a realidade do mercado de trabalho é ainda mais evidente. Uma lacuna foi identificada no ensino dessa área: a ausência de um sistema que simule o processo de cadastramento de produtos, fundamental para a formação prática dos alunos. A introdução de um sistema simulado de cadastramento de produtos nas aulas de logística permitirá que os alunos experimentem um ambiente de aprendizado automatizado, que se assemelha aos processos utilizados nas indústrias reais. Isso possibilitará uma vivência mais próxima das operações logísticas cotidianas, contribuindo para o desenvolvimento de habilidades práticas essenciais, como o controle de estoque, movimentação e armazenamento de produtos. Este sistema simulado visa não só complementar a teoria aprendida nas aulas, mas também aumentar a empregabilidade dos alunos ao prepará-los para os desafios e exigências do mercado de trabalho.

Palavras-chave: logística; sistema; otimização; aprendizagem; simulação;

ABSTRACT

Logistics is a set of essential activities that involves the movement, storage, and management of materials, from the acquisition of raw materials to the final delivery of products to the consumer. It encompasses various interconnected operations, such as transportation, inventory control, order processing, and supply chain management, playing a strategic role in the functioning of companies. Logistic efficiency is crucial for organizations to respond swiftly to market demands, reduce operational costs, and improve competitiveness. Well-executed logistics management allows, for example, the optimization of available resources, minimizing waste and unnecessary costs, such as excess inventory or poorly planned transportation routes.

In the educational context, specifically in the SENAI Industrial Learning Program for Production Line Assistants, the importance of preparing students for the reality of the labor market is even more evident. A gap was identified in the teaching of this area: the lack of a system that simulates the product registration process, which is essential for the students' practical training. The introduction of a simulated product registration system in logistics classes will allow students to experience an automated learning environment, similar to the processes used in real industries. This will enable a closer experience with daily logistics operations, contributing to the development of essential practical skills, such as inventory control, product movement, and storage. This simulated system aims not only to complement the theory learned in classes but also to enhance the employability of students by preparing them for the challenges and demands of the job market.

Keywords: logistics; system; optimization; learning; simulation.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
POP	Procedimento Operacional Padrão
RNC	Registro de Não Conformidade
SAP	Systems, Applications, and Products in Data Processing
SQL	Structured Query Language
SENAI	Serviço Nacional de Aprendizagem Industrial
SEO	Search Engine Optimization
TCC	Trabalho de Conclusão de Curso

SUMÁRIO

1 INTRODUÇÃO

O processo logístico abrange todas as atividades relacionadas à movimentação e armazenamento de materiais, desde a obtenção de matérias-primas até a entrega dos produtos finais ao consumidor. Ele envolve uma série de operações coordenadas, como transporte, controle de estoques, processamento de pedidos e gestão de cadeia de suprimentos. A importância da logística vai além de apenas mover produtos; ela influencia diretamente a capacidade das empresas de atender às demandas do mercado, reduzir custos operacionais e aumentar a competitividade. No contexto atual da indústria, a eficiência dos processos logísticos é fundamental para garantir a eficácia operacional das empresas.

A importância da logística vai além de apenas mover produtos; ela influencia diretamente a capacidade das empresas de atender às demandas do mercado, reduzir custos operacionais e aumentar a competitividade. Uma logística eficiente permite otimizar o uso de recursos, minimizando o desperdício e os custos desnecessários, como excesso de estoques ou rotas de transporte mal planejadas. Além disso, contribui para a melhoria dos prazos de entrega, o que impacta positivamente na satisfação do cliente e na fidelização. Em um ambiente globalizado, onde as cadeias de suprimentos se tornam mais complexas e interconectadas, a eficiência logística é crucial para enfrentar desafios como flutuações na demanda, mudanças nos custos de transporte e questões regulatórias.

A unidade curricular de Processos Logísticos do curso de Aprendizagem Industrial de Auxiliar de Linha de Produção do SENAI, é projetada para fornecer aos alunos conhecimentos e habilidades essenciais para o gerenciamento de recebimento, movimentação, armazenamento, embalagem de produtos e controle de estoque. Contudo, uma lacuna significativa foi identificada na formação dos alunos: a omissão de um sistema que simula o processo de cadastramento de produtos, oferecendo um ambiente de treinamento automatizado e eficiente nas aulas de logística. A introdução de um sistema simulado de cadastramento de produtos nas aulas de logística melhora a aprendizagem ao permitir que os alunos pratiquem em um ambiente automatizado, semelhante ao real. Essa experiência prática facilitará a adaptação dos alunos às

exigências do mercado de trabalho, resultando em uma maior empregabilidade e eficácia na execução de processos logísticos em ambientes industriais reais.

1.1 Hipótese

A introdução de um sistema simulado para o cadastramento e gerenciamento de produtos na unidade curricular de Processos Logísticos do SENAI pode contribuir significativamente para aprimorar a aprendizagem dos alunos, ao proporcionar uma compreensão prática das operações logísticas e desenvolver habilidades técnicas essenciais.

1.2 Objetivo Geral

Desenvolver um sistema para o cadastramento e gerenciamento de produtos e materiais destinados aos alunos do curso de Aprendizagem Industrial de Auxiliar de Linha de Produção no SENAI.

1.3 Objetivos Específicos

1.3.1 Entrevistar docentes e alunos do curso de Auxiliar de Linha de Produção para identificar as demandas e necessidades relacionadas ao gerenciamento de estoque.

1.3.2 Analisar os dados coletados para determinar quais funcionalidades são prioritárias para o sistema simulado.

1.3.3 Levantar e pesquisar bibliografia sobre conceitos relevantes, como processos logísticos, gerenciamento de estoque e linguagens de programação que possam ser utilizadas.

1.4 Justificativa

A relevância desta pesquisa reside na necessidade de criar uma solução educacional que simula o uso de um sistema de gerenciamento de estoque, permitindo que os alunos

desenvolvam habilidades práticas em um ambiente controlado. O desenvolvimento de um sistema simulado baseado em uma plataforma online permitirá aos alunos experimentar diretamente as funcionalidades de um sistema de gerenciamento de estoque similar ao SAP (Systems, Applications, and Products in Data Processing). Essa abordagem proporcionará uma experiência prática mais autêntica e relevante, preparando os alunos para o uso de ferramentas tecnológicas avançadas que eles encontrarão em suas futuras carreiras na indústria.

A escolha deste projeto está diretamente relacionada à necessidade de fornecer aos alunos do curso de Auxiliar de Linha de Produção uma ferramenta prática que complemente sua formação teórica. O desenvolvimento de um sistema simulado para o cadastramento e gerenciamento de produtos é justificado por vários aspectos: este projeto visa enriquecer o aprendizado dos alunos ao oferecer uma aplicação prática dos conceitos abordados na unidade curricular de Processos Logísticos. A simulação de um sistema de gerenciamento de estoque permitirá aos alunos vivenciar na prática os processos logísticos, promovendo um entendimento mais profundo e significativo. A falta de experiências práticas pode limitar a capacidade dos alunos de se adaptarem rapidamente às exigências do mercado de trabalho. Ao desenvolver este sistema, buscamos equipar os alunos com habilidades práticas essenciais, preparando-os melhor para enfrentar os desafios reais na indústria. Alinhado à missão do SENAI de formar profissionais altamente qualificados, este projeto não apenas beneficia os alunos, mas também contribui para o fortalecimento do setor industrial. Ao preparar melhor os alunos para suas carreiras, esperamos impactar positivamente as empresas que os contratarão, promovendo eficiência e inovação no setor.

2 DESENVOLVIMENTO

A história da logística remonta às primeiras civilizações humanas, quando se aperceberam da necessidade de transportar bens para subsistir. O império egípcio (entre 3300 a.C. e 332 a.C.) desenvolveu técnicas de transporte e armazenamento para manter um fornecimento constante de alimentos e bens essenciais. O império romano (entre 27 a.C. e 476 d.C.), por sua vez, estabeleceu uma extensa rede de vias terrestres, assim como marítimas, também conhecidas como "estradas romanas", para facilitar o transporte de bens e tropas pelo seu vasto território. Os romanos também foram pioneiros no uso de comboios de navios para fazer o transporte marítimo e estabeleceram depósitos de suprimentos ao longo de suas rotas.

Durante essa época, foram estabelecidas rotas comerciais que interligam a Europa e a Ásia, facilitando a troca de conhecimentos e de bens, tais como metais, tecidos, pedras preciosas e especiarias, entre diferentes regiões. Uma das mais conhecidas, a Rota da Seda, tornou-se, durante séculos, a principal ligação comercial entre Oriente e Ocidente.

A origem da logística remonta às civilizações antigas, como Mesopotâmia, Egito, China e Roma, que desenvolveram sistemas de transporte e armazenamento para abastecer os exércitos, as cidades e os comércios. Eram utilizados carroças, navios e animais de carga e estabelecidas rotas e armazéns estratégicos durante essa época, as guildas e as rotas comerciais foram essenciais para a troca de bens. Foram desenvolvidos métodos de armazenamento e distribuição nos mercados.

O século XVIII marcou um ponto de inflexão na logística. O desenvolvimento de maquinaria, ferrovias e navios a vapor transformaram os sistemas de transporte e permitiram um fluxo mais rápido e eficiente das mercadorias. A logística tornou-se uma área de estudo e pesquisa mais formal. A produção em massa, a globalização e a adoção de tecnologias como telefone, rádio e, posteriormente, internet, transformaram o gerenciamento logístico.

Com o avanço da tecnologia digital e o surgimento da era da informação, a logística foi objeto de mudanças radicais. Foram desenvolvidos sistemas de gerenciamento de armazém e transporte.

O curso de Logística do SENAI é projetado para formar profissionais capacitados para atuar na gestão da cadeia de suprimentos, transporte e armazenamento de produtos. Com um currículo que abrange desde os fundamentos de logística até o uso de softwares de gestão, os alunos aprendem a otimizar processos e reduzir custos. A duração do curso varia, geralmente sendo oferecido em formato técnico com carga horária de um a dois anos. Os egressos encontram diversas oportunidades de trabalho em setores como indústrias, comércio, transportadoras e empresas de logística, dada a crescente demanda por profissionais qualificados. O SENAI também se destaca pela sua infraestrutura, com laboratórios e equipamentos modernos que proporcionam uma experiência prática valiosa. Ao final do curso, os alunos recebem um certificado reconhecido, que agrega valor ao currículo e abre portas no mercado de trabalho.

O projeto entrou em execução a partir de uma reunião feita com a professora Naiara Marchetti de Paula do curso de logística. A elaboração de um documento preliminar do projeto e a organização das funções de cada membro do grupo foi iniciado.

Assim, o desenvolvimento da tabela de requisitos funcionais e não funcionais foi realizado, em conjunto com o estudo de cores, wireframe e protótipo. A criação do modelo de

requisitos serviu como base para a definição e análise das especificações do projeto, além da estrutura do banco de dados. Após a conclusão dos documentos necessários para a definição das tarefas, iniciou-se o desenvolvimento do código (HTML, CSS e banco de dados).

O programa tem como objetivo preencher a lacuna existente na formação prática dos alunos, oferecendo uma ferramenta que simula operações típicas de sistemas industriais, como o SAP. Através desse projeto, os alunos poderão aplicar conceitos da unidade curricular de Processos Logísticos, incluindo recebimento, movimentação, armazenamento, embalagem e controle de estoque, conforme as normas técnicas da produção industrial. O software visa proporcionar uma experiência prática que complementa a formação teórica, preparando os alunos para uma integração mais eficaz ao mercado de trabalho e aos processos produtivos da indústria.

2.1 Visão Geral

A arquitetura do Sistema de Processos Logísticos é baseada em uma estrutura de duas camadas: a camada de apresentação e a camada de dados. Essa abordagem facilita a separação de responsabilidades, tornando o sistema mais modular e fácil de manter. A Camada de Apresentação e a Camada de Dados são elementos fundamentais de um sistema bem estruturado, trabalhando em conjunto para oferecer uma experiência de uso eficiente e confiável. A Camada de Apresentação é responsável por fornecer a interface com o usuário, permitindo a visualização e interação com o sistema. Para isso, são empregadas tecnologias de Front End, como HTML, CSS e JavaScript, que garantem uma interface intuitiva e responsiva, proporcionando uma navegação agradável e funcional. Já a Camada de Dados cuida do armazenamento e da recuperação das informações. Utilizando um banco de dados relacional, como o MySQL, essa camada assegura a persistência e a integridade dos dados, garantindo que as informações estejam sempre disponíveis e corretas para o usuário. Dessa forma, as duas camadas se integram para criar uma solução completa e eficaz, conectando a experiência visual com a gestão dos dados de maneira eficiente.

2.2 Cadastro de Aluno e Professor

O cadastro do aluno é implementado por meio de um formulário na camada de apresentação, que coleta dados como nome completo, CPF, e-mail e senha. Já o cadastro de professor é semelhante ao cadastro de aluno, mas com campos adicionais para definir o perfil

de acesso. A autenticação e autorização são geridas para garantir que apenas usuários com permissões apropriadas possam acessar e modificar os dados.

2.3 Processo de Login

O Login do Aluno e do Professor são implementados com autenticação baseada em sessão. As credenciais do usuário são verificadas contra as informações armazenadas na camada de dados. Em caso de sucesso, uma sessão é iniciada e o usuário é redirecionado para a tela principal conforme seu perfil.

2.4 Cadastro e Controle de Produtos

O Cadastro de Produto é um formulário que permite a inserção de informações detalhadas sobre o produto. Dados são validados e armazenados no banco de dados, com a possibilidade de visualizar produtos cadastrados. O controle de Estoque. Permite o registro detalhado de entradas e saídas de produtos. A tela de controle de estoque exibe uma tabela com as informações mais recentes, atualizando em tempo real conforme novas entradas e saídas são registradas.

2.5 Registro de Não Conformidade (RNC)

O registro de não conformidades é um processo essencial para a gestão da qualidade e a melhoria contínua dentro da instituição. Esse registro é realizado por meio de um formulário específico que coleta informações detalhadas e estruturadas, garantindo que todos os aspectos relevantes sejam devidamente documentados. Entre as informações solicitadas, destacam-se a descrição minuciosa da não conformidade, que deve incluir o que ocorreu, as circunstâncias envolvidas e os impactos identificados no produto.

2.6 Picking e Expedição

O Cadastro de Picking permite registrar o processo de separação de produtos. Dados como código do produto e quantidade são armazenados, e uma tabela mostra os produtos e quantidades separadas. Do mesmo modo, o Cadastro de Expedição registra a expedição de

produtos, incluindo detalhes como data de embalagem e quantidade. Mensagens de confirmação são fornecidas após o cadastro.

2.7 POP

O POP, mais conhecido como Procedimento Operacional Padrão, é um documento que define como realizar tarefas específicas dentro de uma organização. Na área de logística, o POP é essencial para padronizar atividades como recebimento, armazenamento, movimentação e controle de estoque, assegurando que todas as operações sejam executadas de maneira eficiente, segura e dentro dos padrões de qualidade estabelecidos.

Sua principal função é criar uma sequência de etapas padronizadas que todos os colaboradores devem seguir, o que reduz erros, garante a consistência dos processos e facilita o treinamento de novos funcionários, permitindo que eles se adaptem mais rapidamente ao meio ambiente de trabalho. Ele também é um recurso valioso para auditorias e inspeções, pois registra de forma organizada os procedimentos realizados, garantindo que a empresa esteja em conformidade com normas e regulamentações da área.

Além disso, ele é um instrumento para a melhoria contínua, já que os processos documentados podem ser revisados e otimizados com o tempo, incorporando novas práticas e tecnologias que aumentam a eficiência e a competitividade na logística.

2.8 Processo de Cadastramento

O Processo de Cadastramento em logística é uma etapa fundamental para registrar e organizar informações essenciais sobre produtos, fornecedores, clientes, transportadoras e outros elementos envolvidos na cadeia de suprimentos. Esse processo consiste em coletar, validar e armazenar dados, garantindo que estejam atualizados, completos e padronizados para facilitar o acesso e a utilização em operações logísticas.

A função do cadastramento é assegurar que todas as informações relevantes estejam prontamente disponíveis para processos como controle de estoque, expedição, recebimento de mercadorias e planejamento de rotas. Ele também ajuda a evitar erros, como duplicidades ou inconsistências nos registros, permitindo uma gestão mais eficiente e precisa. Além disso, o cadastramento bem estruturado possibilita que as empresas cumpram com regulamentações e mantenham uma base de dados confiável para tomada de decisões estratégicas na logística.

2.9 Controle de Estoque

O Controle de Estoque é o processo de monitoramento, gerenciamento e organização das mercadorias armazenadas por uma empresa. Em logística, ele é essencial para garantir que os produtos estejam disponíveis na quantidade certa, no momento certo, evitando tanto faltas quanto excessos que podem comprometer o fluxo de operações e o atendimento ao cliente.

Esse controle envolve o registro de entradas e saídas de mercadorias, a atualização dos níveis de estoque, e a realização de inventários periódicos. Sua função é assegurar uma visão precisa dos produtos em estoque, facilitando a reposição eficiente, a previsão de demanda e a otimização do espaço de armazenagem. Além disso, o controle de estoque contribui para a redução de desperdícios, o aumento da eficiência operacional e a diminuição de custos, sendo uma ferramenta fundamental para o planejamento estratégico e a competitividade na logística.

2.10 Visual Studio

É uma ferramenta da Microsoft, conhecida como uma IDE — Integrated Development Environment. Ou seja, é basicamente um software editor de texto que possibilita aos usuários escreverem seus códigos em uma determinada linguagem para, então, serem traduzidos em comandos para os computadores.

Com o Visual Studio é possível desenvolver aplicativos da Web, aplicativos desktop e aplicativos móveis. O Visual Studio é um assistente para desenvolvimento. Independentemente da linguagem que você utiliza, ele faz o preenchimento automático dos comandos para agilizar a construção do seu código.

Quando você terminar ou desejar testar um código, é possível depurá-lo (verificar erros de sintaxe do código e da lógica de comandos do programa), além de dispor de uma facilidade em acompanhar seu código com marcações de início de código, filtros e funções.

2.11 HTML

A Linguagem de Marcação de Hipertexto (HTML) é uma linguagem de computador que compõe a maior parte das páginas da internet e dos aplicativos online. Um hipertexto é um texto usado para fazer referência a outros textos, enquanto uma linguagem de marcação é composta por uma série de marcações que dizem para os servidores da web qual é o estilo e a estrutura de um documento.

O HTML não é considerado uma linguagem de programação, já que ele não pode criar funcionalidades dinâmicas. Ao invés disso, com o HTML, os usuários podem criar e estruturar seções, parágrafos e links usando elementos, tags e atributos.

Os desenvolvedores usam códigos HTML para projetar como um navegador vai exibir os elementos das páginas, como textos, hiperlinks e arquivos de mídia. Os usuários podem navegar facilmente e inserir links entre as páginas e sites relacionados, já que o HTML é amplamente usado para incorporar hiperlinks.

O HTML torna possível a organização e a formatação de documentos, de maneira similar ao Microsoft Word.

Todas as páginas HTML possuem uma série de elementos, que consistem num conjunto de tags e atributos. Os elementos HTML são os tijolos de construção de uma página da internet. Uma tag para o navegador onde um elemento começa e termina, enquanto um atributo descreve as características de um elemento.

2.12 CSS

O CSS (Cascading Style Sheets) foi desenvolvido pelo W3C (World Wide Web Consortium) em 1996, com um objetivo crucial: resolver as limitações do HTML na formatação e estilização de páginas web. Desde o início, o HTML foi projetado como uma linguagem de marcação, focando principalmente na estruturação do conteúdo. No entanto, à medida que os sites começaram a evoluir e a se diversificar, tornou-se evidente que o HTML não possuía as ferramentas necessárias para controlar a apresentação visual de forma eficiente.

As primeiras versões do HTML, embora inovadoras, não incluíam tags específicas para a formatação visual de elementos. A introdução de tags como <font> na versão 3.2 do HTML, por exemplo, trouxe consigo uma série de problemas para os desenvolvedores. O uso excessivo de tags de formatação resultava em um código extenso e complexo, que se tornava cada vez mais difícil de manter. Cada vez que era necessário alterar uma cor, uma fonte ou um estilo, os desenvolvedores precisavam percorrer todo o código, o que tornava o processo demorado e custoso. Esse cenário gerou a necessidade urgente de uma solução mais prática e eficiente para gerenciar a estética das páginas web.

Foi então que o W3C criou o CSS, proporcionando uma maneira de separar o conteúdo da apresentação. Com o CSS, os desenvolvedores podem aplicar estilos a múltiplos elementos de um site por meio de um único arquivo, permitindo a manutenção e a realização de alterações de forma rápida e centralizada. Por exemplo, se um desenvolvedor deseja mudar a cor de fundo

de um elemento, ele pode simplesmente utilizar uma regra CSS como background-color: blue;. Para alterar a fonte de um texto, pode-se aplicar a regra font-family: Arial, sans-serif;, e para criar um espaçamento uniforme entre elementos, utiliza-se margin: 20px;. Essa capacidade de aplicar estilos de maneira eficiente transformou a abordagem de desenvolvimento web, reduzindo significativamente o tempo e o esforço necessários para criar e manter sites.

A relação entre HTML e CSS é uma parceria fundamental no desenvolvimento web. O HTML serve como a estrutura, ou o alicerce, do site, enquanto o CSS se concentra na estética e na apresentação visual. Essa divisão de responsabilidades permite que os desenvolvedores trabalhem de maneira mais organizada e produtiva, criando experiências de usuário mais agradáveis e funcionais. Por exemplo, ao usar CSS, um único arquivo pode controlar a aparência de várias páginas, o que facilita a implementação de mudanças de design sem a necessidade de reescrever o código HTML.

Embora o CSS não seja uma necessidade técnica absoluta para que um site funcione, a sua ausência resulta em uma experiência visualmente insatisfatória. Um site que utiliza apenas HTML pode parecer antiquado e pouco atraente, com uma apresentação que pode parecer "abandonada". Quando um site não carrega corretamente ou aparece com um fundo branco e texto azul, isso geralmente indica que a parte do CSS não foi carregada ou não existe. Essa situação não só afeta a estética, mas também pode impactar a usabilidade e a percepção do usuário em relação à qualidade do site.

Além disso, o CSS permite uma maior flexibilidade e criatividade no design web. Com a capacidade de aplicar estilos responsivos, os desenvolvedores podem criar layouts que se adaptam a diferentes tamanhos de tela e dispositivos, melhorando a experiência do usuário em dispositivos móveis, tablets e desktops. A utilização de media queries, por exemplo, permite que os desenvolvedores definam regras de estilo específicas para diferentes resoluções de tela, garantindo que o conteúdo seja apresentado de forma adequada, independentemente do dispositivo utilizado.

Em resumo, o CSS revolucionou a maneira como os desenvolvedores criam e mantêm websites, oferecendo uma solução prática e eficiente para gerenciar a apresentação visual. A combinação de HTML e CSS não apenas possibilitou uma web mais atraente e funcional, mas também estabeleceu um padrão que continua a ser a base do design web moderno. Essa separação entre estrutura e estilo é fundamental para a criação de experiências web dinâmicas e agradáveis, permitindo que os desenvolvedores se concentrem em produzir conteúdo de qualidade enquanto mantêm a estética de suas páginas. Essa evolução não só transformou o

desenvolvimento web, mas também impactou a forma como interagimos com a internet, elevando a qualidade da experiência do usuário a patamares antes inimagináveis.

2.13 Backend

Backend refere-se a toda a parte "invisível" do site ou aplicativo, mas que é essencial para o seu funcionamento. O Backend é responsável por processar as informações enviadas pelo usuário, gerenciar dados e realizar as operações necessárias para que o sistema funcione corretamente. Por exemplo, ao realizar uma compra em um site, o backend verifica a disponibilidade do produto, processa o pagamento, atualiza o estoque e gera a confirmação do pedido. Ele também gerencia o armazenamento e a segurança dos dados do usuário, como informações de login e histórico de transações.

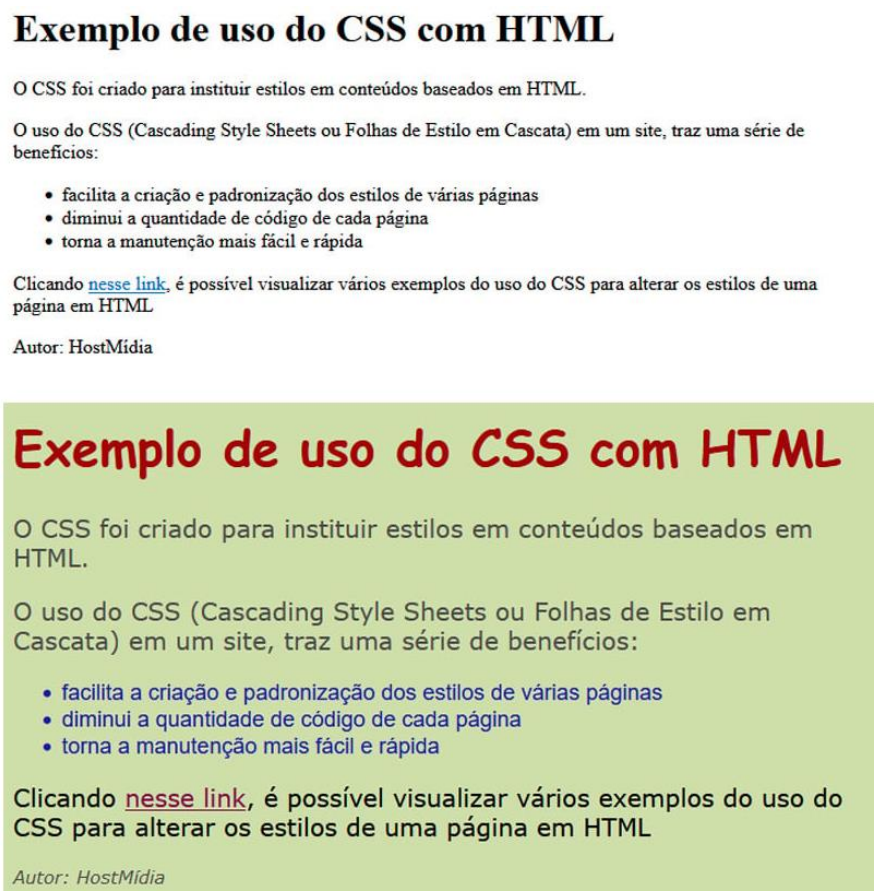
O backend é composto por servidores, bancos de dados e aplicações que interagem para garantir que o sistema funcione de maneira eficiente e segura. Embora o frontend seja o que o usuário vê, o backend faz o trabalho pesado nos bastidores, garantindo que a comunicação entre o servidor e o cliente aconteça de forma rápida e sem falhas. O backend também cuida de processos essenciais, como a autenticação de usuários, a validação de dados e a integração com outros sistemas.

2.14 Frontend

Frontend refere-se à parte de um site ou aplicativo com a qual o usuário interage diretamente. Pode ser comparado à fachada de um edifício, ou à vitrine de uma loja, onde estão expostos os produtos e serviços. No contexto digital, o frontend envolve todos os elementos visíveis em uma página da web ou aplicativo: o layout, os menus, os botões, as imagens, os textos, as cores e os efeitos de interação. Por exemplo, ao acessar uma loja online, o frontend é a área onde o usuário visualiza os produtos, navega entre as opções, adiciona itens ao carrinho e conclui a compra.

O objetivo do frontend é garantir que o usuário tenha uma experiência de navegação intuitiva e agradável. Para alcançar isso, é necessário um design bem planejado, que considere a usabilidade, a acessibilidade e a estética, além da funcionalidade. A programação de frontend é geralmente feita com as linguagens HTML (para estruturar o conteúdo), CSS (para definir o estilo e o layout) e JavaScript (para tornar a página interativa e dinâmica). Essas tecnologias garantem que a interface do usuário seja atraente e funcione de maneira eficiente, independentemente do dispositivo utilizado (computador, tablet ou celular).

Figura 1 - HTML e CSS



Fonte: https://www.hostmidia.com.br/blog/o-que-e-css/

2.15 Banco de Dados

Um banco de dados é um sistema organizado para armazenar, gerenciar e recuperar informações de forma eficiente. Ele permite que grandes volumes de dados sejam acessados, manipulados e atualizados de maneira rápida e segura. Os bancos de dados podem ser utilizados em diversas aplicações, desde sites e aplicativos até sistemas empresariais complexos.

Eles são fundamentais para manter a integridade e a organização da informação, permitindo que os usuários realizem consultas e operações de forma confiável. Eles são projetados para lidar com grandes quantidades de dados e garantir que esses dados possam ser acessados e modificados por vários usuários simultaneamente, sem comprometer a precisão e a segurança.

Existem diferentes tipos de bancos de dados, mas o mais comum é o banco de dados relacional.

2.16 Banco de Dados Relacional

Um banco de dados relacional é uma estrutura que organiza dados em tabelas separadas, ao invés de agrupá-los em um único repositório. Essa abordagem permite uma gestão mais eficiente e organizada das informações, facilitando a consulta, a atualização e a manutenção dos dados. Cada tabela pode conter um conjunto específico de dados relacionados, o que ajuda a evitar redundâncias e a manter a integridade das informações.

Um exemplo amplamente utilizado de banco de dados relacional é o MySQL. Este sistema de gerenciamento de banco de dados armazena dados em tabelas e utiliza a linguagem SQL (Structured Query Language) para permitir consultas complexas e específicas. A SQL é uma linguagem poderosa que permite realizar operações como selecionar, inserir, atualizar e excluir dados, tornando o MySQL altamente funcional para diversas aplicações.

O MySQL é reconhecido por sua velocidade, confiabilidade e facilidade de uso. Desde a sua criação, foi projetado para processar grandes volumes de dados de forma rápida e eficiente. Essa capacidade de lidar com grandes quantidades de informações fez dele uma escolha popular para aplicações em ambientes de produção que exigem desempenho robusto e confiável.

Além de seu uso em aplicações web e sistemas de gerenciamento de conteúdo, o MySQL é essencial para a gestão de dados em várias áreas, incluindo operações logísticas. No contexto logístico, o banco de dados relacional permite armazenar informações cruciais, como dados de inventário, informações de transporte e detalhes de clientes, proporcionando uma base sólida para a análise e a tomada de decisões.

Com a capacidade de realizar consultas complexas, o MySQL oferece suporte para análises aprofundadas dos dados, ajudando as organizações a otimizar suas operações e a melhorar a eficiência. Esse sistema não apenas facilita a organização e o armazenamento de dados, mas também contribui significativamente para a agilidade e a efetividade na gestão de informações em diversos setores.

2.17 SQL

O SQL, que é a sigla para Structured Query Language, ou Linguagem de Consulta Estruturada em português, é uma linguagem padrão projetada para trabalhar com bancos de dados relacionais. Essa linguagem é amplamente utilizada por profissionais de diversas áreas,

incluindo cientistas de dados, analistas de dados e até mesmo usuários de ferramentas como o Excel, que necessitam de manipulação e consulta de dados.

Uma das características mais relevantes do SQL é a sua semelhança entre os principais Sistemas Gerenciadores de Banco de Dados (SGBDs) disponíveis no mercado. Exemplos desses SGBDs incluem Oracle, MySQL, MariaDB, PostgreSQL e Microsoft SQL Server. Essa semelhança facilita a transferência de habilidades e conhecimentos entre diferentes plataformas, permitindo que os profissionais se adaptem a diferentes sistemas com relativa facilidade.

No entanto, é crucial destacar que, apesar das semelhanças, cada um desses SGBDs possui características e particularidades distintas. Cada sistema pode ter suas próprias implementações de SQL, com extensões e funcionalidades específicas que atendem a diferentes necessidades. Por exemplo, enquanto o MySQL é conhecido por sua simplicidade e eficiência em aplicações web, o PostgreSQL é frequentemente elogiado por seu suporte a operações complexas e por ser altamente extensível.

Uma característica que destaca o MySQL e o PostgreSQL é que ambos oferecem versões gratuitas, o que contribui para sua popularidade entre desenvolvedores e pequenas empresas. Essa acessibilidade permite que muitos usuários experimentem e implementem soluções sem custos significativos, tornando-os escolhas preferidas para projetos de todos os tamanhos. Essa vantagem financeira, aliada à robustez das funcionalidades oferecidas, ajuda a explicar por que esses bancos de dados são tão amplamente adotados no mercado.

Assim, o SQL se torna uma ferramenta essencial na gestão e manipulação de dados, oferecendo a flexibilidade e a capacidade necessárias para atender às demandas de diferentes usuários e contextos, enquanto os SGBDs que o utilizam oferecem uma variedade de características que podem ser aproveitadas de acordo com as necessidades específicas de cada projeto.

2.18 JavaScript

O JavaScript é uma linguagem de programação interpretada de alto nível que se destaca como a mais popular do mundo, de acordo com a Pesquisa de Desenvolvedores do Stack Overflow de 2022. Essa popularidade é atribuída, em grande parte, ao fato de que o JavaScript é a linguagem padrão que os navegadores da web interpretam, formando uma combinação essencial com o HTML, que é a base de toda a Web. Juntas, essas tecnologias permitem a criação de páginas dinâmicas e interativas, fundamentais para a experiência do usuário online.

Desde sua criação, o JavaScript passou por uma evolução significativa. Inicialmente concebido para adicionar interatividade a sites, ele se expandiu para abarcar uma variedade de aplicações e usos. Essa evolução é acompanhada por uma comunidade forte e consolidada de desenvolvedores, que contribuem ativamente para seu crescimento. Essa comunidade proporciona uma série de benefícios, como atualizações frequentes que introduzem novas funcionalidades e melhorias, além de garantir segurança em suas implementações.

Outra vantagem da comunidade é o suporte mútuo entre desenvolvedores, que se ajuda por meio de fóruns, grupos de discussão e conferências. Essa colaboração não só facilita a resolução de problemas, mas também fomenta a criação de novas bibliotecas e frameworks que tornam o desenvolvimento em JavaScript mais eficiente e acessível. Ferramentas como React, Angular e Vue.js são exemplos de bibliotecas populares que oferecem recursos avançados para a construção de interfaces ricas e responsivas.

A versatilidade do JavaScript é outro ponto forte que contribui para sua popularidade. Além de ser amplamente utilizado no desenvolvimento de aplicações web, ele também pode ser aplicado em uma variedade de outras áreas. O JavaScript é utilizado para criar aplicativos móveis através de frameworks como React Native, permitindo que desenvolvedores escrevam código para plataformas iOS e Android usando a mesma linguagem.

Essa capacidade de se adaptar a diferentes contextos e necessidades faz do JavaScript uma escolha atraente para desenvolvedores de todos os níveis. Sua presença em diversos setores e aplicações, juntamente com uma comunidade vibrante e um ecossistema em constante evolução, solidificam o JavaScript como uma linguagem essencial no cenário atual da tecnologia. Em suma, o JavaScript não é apenas uma linguagem de programação; é uma ferramenta poderosa que molda o futuro do desenvolvimento web e de muitas outras áreas da tecnologia.

2.19 Requisitos funcionais

Os requisitos funcionais são definições detalhadas que especificam as funções e comportamentos que um sistema deve apresentar para atender às necessidades dos usuários e às expectativas do negócio. Eles são fundamentais no processo de desenvolvimento de software, pois estabelecem um guia claro sobre o que o sistema deve realizar. Esses requisitos incluem descrições de ações que o sistema deve executar, como a entrada de dados, o processamento dessas informações e a saída de resultados esperados.

Essencialmente, os requisitos funcionais abordam perguntas como: "O que o sistema deve fazer?" e "Quais são as interações do usuário com o sistema?" Por exemplo, em um sistema de e-commerce, requisitos funcionais podem incluir a capacidade de um usuário criar uma conta, adicionar produtos ao carrinho, realizar o checkout e receber confirmações de pedido. Cada uma dessas ações representa uma função que o sistema deve suportar.

A importância dos requisitos funcionais se estende além da simples descrição de funções. Eles orientam a equipe de desenvolvimento sobre o que precisa ser construído, assegurando que todos os membros estejam alinhados em relação às expectativas do projeto. Essa clareza ajuda a evitar mal-entendidos que podem surgir durante o desenvolvimento, o que, por sua vez, pode economizar tempo e recursos.

Além disso, os requisitos funcionais servem como base para os testes e validação do sistema. Uma vez que o software é desenvolvido, é necessário verificar se ele atende a todos os requisitos funcionais especificados. Essa etapa é crucial para garantir que o produto final não apenas funcione, mas também atenda às necessidades dos usuários. A documentação clara dos requisitos funcionais permite que os testadores desenvolvam casos de teste que possam ser usados para validar cada função do sistema.

Outra razão pela qual a documentação dos requisitos funcionais é tão crítica é que ela ajuda a garantir que o produto final seja útil e alinhado com as metas do projeto. Quando os requisitos são bem definidos, a equipe pode focar em desenvolver soluções que realmente atendam às necessidades do negócio, evitando funcionalidades desnecessárias ou irrelevantes.

Em suma, os requisitos funcionais são uma parte essencial do desenvolvimento de software. Eles proporcionam uma visão clara do que o sistema deve fazer, orientam a equipe durante todo o processo de desenvolvimento e são fundamentais para garantir que o produto final atenda às expectativas dos usuários e aos objetivos do negócio. Essa documentação não só facilita a comunicação entre os membros da equipe, mas também estabelece uma base sólida para o sucesso do projeto.

2.20 Requisitos não funcionais

Os requisitos não funcionais são critérios que descrevem como um sistema deve operar, ao invés de focar em comportamentos ou funcionalidades específicas. Esses requisitos abrangem aspectos críticos que influenciam a qualidade e a eficiência do sistema, como desempenho, usabilidade, segurança, confiabilidade, escalabilidade e manutenção. Eles são fundamentais para assegurar que o sistema não apenas funcione corretamente, mas também

atenda a padrões de qualidade que são essenciais para a experiência do usuário e o sucesso do projeto.

Por exemplo, um requisito não funcional pode especificar que o sistema deve suportar 1.000 usuários simultâneos, o que se refere à escalabilidade. Essa característica é crucial para sistemas que esperam um grande volume de acessos, garantindo que o desempenho não seja comprometido quando o número de usuários aumenta. Outro exemplo pode ser um requisito de usabilidade que determina que a interface do usuário deve ser intuitiva e acessível a diferentes perfis de usuários, o que é vital para assegurar que todos possam utilizar o sistema com facilidade, independentemente de suas habilidades tecnológicas.

A definição clara dos requisitos não funcionais é vital para orientar decisões arquiteturais e de design. Eles ajudam os desenvolvedores e arquitetos de software a escolher as tecnologias e abordagens adequadas para atingir os padrões de qualidade esperados. Por exemplo, um foco em segurança pode levar à implementação de criptografia e autenticação robustas, enquanto um foco em desempenho pode levar à otimização de consultas de banco de dados e ao uso de cache.

Além disso, os requisitos não funcionais facilitam a avaliação do sucesso do sistema após sua implementação. Eles fornecem critérios que podem ser usados para medir se o sistema atinge os padrões desejados, como tempos de resposta em determinadas condições ou a taxa de erros em operações críticas. Essa avaliação é essencial para identificar áreas de melhoria e garantir que o sistema continue a atender às necessidades dos usuários ao longo do tempo.

Outro ponto importante é que os requisitos não funcionais são essenciais para garantir que o software possa ser mantido e evoluído. Um sistema que atende a critérios de manutenção será mais fácil de atualizar e corrigir, permitindo que a equipe responda rapidamente a mudanças nas necessidades do negócio ou a novas exigências de segurança. Isso é particularmente relevante em um cenário tecnológico em constante evolução, onde as demandas dos usuários e as condições de mercado podem mudar rapidamente.

Em suma, os requisitos não funcionais desempenham um papel crucial no desenvolvimento de software, assegurando que o sistema não só funcione corretamente, mas também ofereça uma experiência de alta qualidade aos usuários. Eles estabelecem diretrizes que orientam decisões de design e arquitetura, facilitam a avaliação do sucesso do sistema e garantem a sua manutenção ao longo do tempo. Ao serem bem definidos e documentados, esses requisitos contribuem significativamente para o sucesso geral do projeto.

2.21 Estudo das cores

O estudo de cores na tecnologia é um campo abrangente que integra diversas áreas, incluindo design gráfico, web design, fotografia e interfaces de usuário. As cores desempenham um papel fundamental na comunicação visual, pois influenciam não apenas a percepção estética, mas também as emoções e comportamentos dos usuários. Compreender como as cores impactam a experiência do usuário é essencial para criar projetos eficazes e envolventes.

A teoria das cores serve como a base desse estudo, utilizando o círculo cromático para ilustrar as relações entre cores primárias, secundárias e terciárias. Este círculo é uma ferramenta valiosa para designers, pois ajuda a visualizar como diferentes cores interagem entre si. A escolha de esquemas de cores, como combinações complementares (cores opostas no círculo cromático) e análogas (cores que estão próximas umas das outras), é fundamental para criar harmonia visual em projetos. Esses esquemas não apenas tornam as composições estéticamente agradáveis, mas também facilitam a comunicação da mensagem desejada.

Outro aspecto importante é a psicologia das cores, que explora como diferentes cores podem evocar reações emocionais e comportamentais. Por exemplo, o azul é frequentemente associado à calma e à serenidade, tornando-o uma escolha popular em ambientes que buscam transmitir segurança. Por outro lado, o vermelho é conhecido por evocar urgência e energia, frequentemente utilizado em contextos que requerem ação rápida. Compreender essas associações emocionais é vital para o design de marcas e campanhas publicitárias, pois permite que os designers escolham cores que ressoem com o público-alvo e transmitam a mensagem desejada de forma eficaz.

No contexto de um projeto específico, a escolha da cor azul foi feita devido à sua forte associação com a logística. O azul simboliza tecnologia, eficiência, organização e confiabilidade, características essenciais para a gestão de processos logísticos. Essa escolha não é apenas estética, mas também estratégica, uma vez que reforça valores que a marca deseja transmitir.

Além do azul, o roxo foi escolhido por sua capacidade de estimular a criatividade e a inovação. O roxo é frequentemente associado à originalidade e ao pensamento inovador, aspectos que a equipe pretende incorporar em sua abordagem. Essa combinação de cores visa não apenas criar uma identidade visual coesa, mas também inspirar a equipe e os usuários a pensar de maneira criativa e eficiente.

Em suma, o estudo de cores na tecnologia é um campo que vai além da estética; é uma disciplina que envolve a compreensão das interações emocionais e funcionais das cores. A escolha cuidadosa de esquemas de cores e a consideração de suas associações psicológicas são

fundamentais para o sucesso em design gráfico, web design e outros campos relacionados. Assim, as cores escolhidas para um projeto devem refletir não apenas a identidade visual desejada, mas também os valores e as emoções que a marca pretende transmitir aos seus usuários.

Figura 2 - Cores

Fonte: Canva

2.22 Python

"Python é uma linguagem de programação de alto nível que se destaca pela sua simplicidade. Criada por Guido Van Rossum e lançada em 1991, Python foi projetada para ser intuitiva, permitindo que os desenvolvedores escrevam código de forma clara e concisa. Essa característica torna a linguagem especialmente atraente tanto para iniciantes, que estão dando seus primeiros passos na programação, quanto para programadores experientes que valorizam a manutenção e a legibilidade do código ao longo do tempo."

A simplicidade de Python não se limita apenas à sua sintaxe. A linguagem é estruturada de maneira que as ações e intenções do código sejam facilmente compreendidas, o que facilita o aprendizado e a colaboração entre desenvolvedores. Essa clareza ajuda a reduzir a curva de aprendizado, permitindo que novos programadores se tornem produtivos rapidamente. Para programadores mais experientes, essa legibilidade significa que o código pode ser mantido e atualizado com mais eficiência, economizando tempo e esforço em projetos de longo prazo.

Outro aspecto que contribui para a popularidade de Python é a sua vasta comunidade de usuários. Essa comunidade ativa não apenas compartilha conhecimento e recursos, mas também contribui para um ecossistema rico em bibliotecas e frameworks. Essas ferramentas ajudam os desenvolvedores a resolver problemas complexos de forma mais acessível, permitindo que eles se concentrem nas soluções em vez de reinventar a roda. Por exemplo, bibliotecas como NumPy e pandas facilitam o trabalho com dados, enquanto frameworks como Django e Flask são amplamente utilizados para o desenvolvimento web.

A popularidade do Python continua a crescer, consolidando sua posição como uma das linguagens de programação mais exigidas no mercado de trabalho. Seu uso se estende a diversas áreas, como desenvolvimento web, análise de dados, inteligência artificial, automação e muito mais. A versatilidade da linguagem, combinada com sua simplicidade, faz dela uma escolha favorita entre empresas e desenvolvedores que buscam soluções rápidas e eficientes.

Em resumo, Python é mais do que uma linguagem de programação; é uma ferramenta poderosa que combina simplicidade e eficácia. Desde sua criação, evoluiu para se tornar uma das linguagens mais utilizadas no mundo, impulsionada por uma comunidade engajada e um ecossistema rico em recursos. Sua capacidade de facilitar a aprendizagem e a colaboração, juntamente com sua popularidade crescente, a tornam uma escolha ideal para desenvolvedores de todos os níveis.

2.23 Wireframe

Wireframe é uma representação visual básica de uma página da web ou de um aplicativo, que tem como principal objetivo esboçar a estrutura e o layout do conteúdo. Essa ferramenta é essencial no processo de design, pois oferece um guia visual que mostra a disposição de elementos cruciais, como menus, botões, imagens e textos, sem se preocupar com detalhes visuais, como cores ou tipografia. Ao focar na estrutura, os wireframes permitem que as equipes de design e desenvolvimento se concentrem na funcionalidade e na usabilidade do produto.

Uma das principais vantagens dos wireframes é que eles ajudam a alinhar ideias entre os membros da equipe, facilitando a comunicação e o entendimento mútuo sobre a direção do projeto. Com um wireframe, é mais fácil visualizar como os diferentes componentes interagem e como o usuário navegará pelo site ou aplicativo. Isso pode ser especialmente útil em fases iniciais do desenvolvimento, onde as ideias estão sendo formuladas e o feedback é essencial.

Além disso, os wireframes são uma ferramenta valiosa para identificar problemas de usabilidade antes que o projeto avance para o design final. Ao permitir que os stakeholders visualizem a estrutura e a funcionalidade propostas, eles podem fornecer feedback construtivo, ajudando a prevenir retrabalhos dispendiosos em fases posteriores do desenvolvimento. Isso contribui para um processo de design mais eficiente e eficaz, onde as necessidades do usuário são consideradas desde o início.

Os wireframes podem ser criados de várias maneiras, desde esboços à mão, que capturam rapidamente ideias iniciais, até protótipos mais elaborados feitos com ferramentas digitais especializadas, como Figma, Sketch ou Adobe XD. A escolha do método de criação pode variar dependendo do estágio do projeto e da complexidade necessária. Os níveis de detalhe também podem diferir, com alguns wireframes apresentando apenas a disposição básica dos elementos, enquanto outros podem incluir informações mais específicas, como anotações sobre interações e comportamentos.

Em resumo, os wireframes são uma ferramenta fundamental no processo de design de páginas da web e aplicativos, permitindo que equipes alinhem suas ideias, identifiquem problemas de usabilidade e construam uma base sólida para o desenvolvimento futuro. Ao focar na estrutura e na funcionalidade, eles ajudam a garantir que o produto final atenda às expectativas dos usuários e às metas do projeto.

2.24 Testes de Usabilidade

Os testes de usabilidade foram realizados por alguns alunos e pelos professores Lucas, Naiara, Mariana e Wilson, para avaliar a experiência dos usuários com o sistema, abordando organização dos elementos, navegação, design e funcionalidades. De modo geral, as avaliações foram muito positivas, destacando a clareza da interface, facilidade de navegação entre as páginas e a atratividade das cores e do layout.

As páginas de cadastro, como RNC, Picking, Estoque, POP e Expedição, bem como página de login, receberam boas avaliações, mostrando que as funcionalidades são práticas e de fácil uso.

Entre as sugestões, os usuários propuseram a adição de QR codes para substituir etiquetas tradicionais, além de ajustes visuais, como centralizar os temas na página inicial. Essas sugestões visam aprimorar a praticidade e modernizar o sistema.

O sistema foi bem aceito, com algumas sugestões pontuais que podem agregar valor e melhorar a experiência dos usuários.

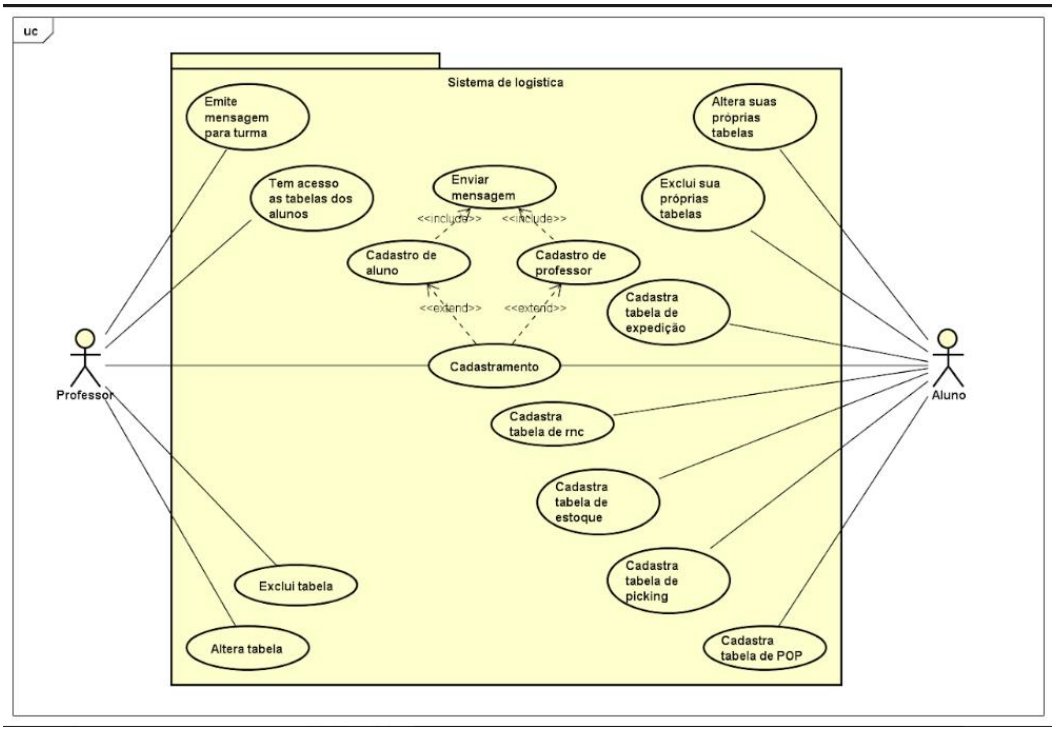
2.25 Diagrama de caso de uso

O diagrama de caso de uso é uma ferramenta da UML (Unified Modeling Language) que representa graficamente as interações entre os atores externos e as funcionalidades de um sistema. Ele destaca as ações que o sistema deve realizar e quais usuários ou sistemas externos interagem com essas ações.

O diagrama é composto por atores (que representam usuários ou sistemas externos), casos de uso (que são as funcionalidades oferecidas pelo sistema) e relações entre eles. Essa ferramenta é útil para compreender melhor o funcionamento do sistema, identificar requisitos funcionais e facilitar a comunicação entre a equipe de desenvolvimento e as partes interessadas.

Além disso, o diagrama de caso de uso é frequentemente usado como base para o planejamento e implementação do sistema, servindo como um ponto de partida para o desenvolvimento do projeto. Por exemplo, em um sistema de logística, ele pode ilustrar como um "Gerente de Estoque" acessa funcionalidades como "Gerenciar Inventário" po "Emitir Relatórios".

Figura 3 - Diagrama



Fonte: Astah, Software

2.26 GitHub

Em uma equipe, apenas poder acessar o código de outras pessoas colaboradoras não é suficiente. Mais do que isso: precisamos manter o histórico dos nossos arquivos e das nossas modificações. Afinal de contas, muitas vezes mudamos arquivos em grupo, num movimento único — que, no contexto do Git, é um commit. O que, em tradução literal para português, significa “compromisso” ou “comprometer-se” às alterações em um repositório.

Dessa forma, podemos voltar atrás e recuperar o estado do sistema: como ele era ontem, ou no ano passado, comparar as mudanças para encontrar bugs e estudar otimizações.

Figura 4 - Commits

Fonte: https://alura.com.br

Na imagem acima há destaque para 3 ***commits*:**

1 - “Adicionando título a página index”: Neste primeiro commit percebemos que o dev responsável criou a estrutura inicial de uma página de perfil de usuário.

2 - “Altera o título da index para Perfil do usuário e adiciona xyz.png”: Neste segundo commit notamos que alteraram o título da página e acrescentaram uma nova imagem.

3 - “Altera o layout da página”: Neste terceiro commit, observamos que alteraram o layout da página de usuário, cores e posições de elementos. Com base nestes commits, se porventura o

cliente não gostar da mudança de layout implementada no commit 3, o time desenvolvimento pode voltar o layout como era no commit 2.

Além disso, podemos identificar que cada um desses commits contam alguma história, a partir do seu motivo de criação.

Isso auxilia bastante quem está lendo a identificar o que está procurando.

Figura 5 - GitHub

.github/workflows	Add or update the Azure App Service build and deplo...	28 minutes ago
.vscode	.	2 weeks ago
static	atualizando o menu	13 minutes ago
templates	Merge branch 'main' of https://github.com/IsabelleElo...	last week
.deployment	tentativa de deploy	2 weeks ago
.gitignore	.	2 months ago
README.md	Update README.md	2 months ago
aluno.py	conclusão da página para redefinir a senha do aluno	2 weeks ago
app.py	ajuste na tabela do inventario - permissões	last week
cadastramento.py	ajuste na tabela do inventario - permissões	last week
conexao.py	conexao com o azure	2 weeks ago
estoque.py	filtros e alteração na estrutura do código	3 weeks ago
expedicao.py	VALIDACAO-V4-FINAL	last month
picking.py	VALIDACAO-V4-FINAL	last month
pop.py	VALIDACAO-V4-FINAL	last month
professor.py	ajuste na tabela do inventario - permissões	last week

Fonte: https://github.com.br

3 CONSIDERAÇÕES FINAIS

A introdução de um sistema simulado no curso de Aprendizagem Industrial de Auxiliar de Linha de Produção do SENAI tem o potencial de transformar a formação dos alunos, proporcionando uma experiência prática que complementa a teoria. Com a simulação de processos logísticos como cadastramento de produtos, controle de estoque e expedição, os alunos poderão vivenciar atividades industriais reais em um ambiente controlado. Isso contribui para o desenvolvimento de habilidades técnicas essenciais, preparando os alunos para os desafios do mercado de trabalho. Este projeto, alinhado à missão do SENAI de formar profissionais capacitados, não só beneficia os alunos, mas também fortalece o setor industrial, promovendo maior eficiência e inovação.

Existe também toda uma área de gestão que atua no mesmo ramo, não só na logística, mas em diversos outros setores, e o projeto tem o potencial de continuar evoluindo. Futuramente, é possível implementar novas funcionalidades, como a criptografia de senha, para aumentar a segurança do sistema e acompanhar as exigências do mercado e da tecnologia.

Os testes de usabilidade realizados mostraram que o sistema foi bem aceito pelos usuários, com algumas sugestões de melhorias, como a inclusão de QR codes e ajustes visuais na interface. Essas sugestões visam aprimorar a praticidade e modernizar ainda mais o sistema, melhorando a experiência do usuário e a eficiência do sistema.

Além das habilidades técnicas, o projeto também contribuiu para o desenvolvimento das competências socioemocionais do grupo. Ao longo da execução do projeto, os alunos aprimoraram suas habilidades de ouvir, regular melhor suas emoções, assumir responsabilidade e organização, e fortaleceram sua resiliência, aprendendo a crescer e se adaptar frente a desafios. Essas competências são essenciais tanto no ambiente acadêmico quanto no mercado de trabalho, e são fundamentais para o desenvolvimento profissional e pessoal dos alunos.

O objetivo principal deste projeto foi testar e desenvolver um sistema que proporcionasse aos alunos uma experiência prática e dinâmica, simulando processos logísticos reais. Através dos testes de usabilidade, foi possível avaliar a eficácia do sistema e identificar áreas para melhorias. O desenvolvimento do sistema, com base nas avaliações recebidas, permitiu não apenas ajustar suas funcionalidades, mas também planejar futuras implementações que garantam uma experiência mais segura, moderna e eficiente. Assim, o projeto alcançou seu propósito de integrar teoria e prática, preparando os alunos para os desafios do mercado de trabalho.

REFERÊNCIAS

https://www.totvs.com/blog/gestao-logistica/processos-logisticos/
https://www.fretefy.com.br/blog/fique-atento-processos-logisticos
https://nao-conformidade.portaliso.com/o-que-e-registro-de-nao-conformidade/
https://qualyteam.com/pb/blog/registro-de-nao-conformidade/
https://www.alura.com.br/artigos/html-css-e-js-definicoes?srsltid=AfmBOoqdYIN7EWDFfTrd0FcBydAORzus0_bcZRsE6wrJCQ0fLDly88kG
https://www.oracle.com/br/database/what-is-database/
https://rockcontent.com/br/blog/banco-de-dados/
https://www.oracle.com/br/database/what-is-a-relational-database/
https://www.alura.com.br/artigos/o-que-e-sql?srsltid=AfmBOopYagawOc8zapNb0Yc7renPFQIv0yxjrOD04kxpEaLuJGA0gXC7
https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript
https://www.mestresdaweb.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao

https://querobolsa.com.br/revista/requisitos-funcionais-e-nao-funcionais
https://rockcontent.com/br/blog/psicologia-das-cores/
https://ebaconline.com.br/blog/significa-a-cor-azul-seo
https://aws.amazon.com/pt/what-is/python/
https://www.alura.com.br/artigos/python?srsltid=AfmBOooYXARjYG0bi8fHorfBHlHosXzeBOz9hnTr4V6eZa0lxDWfGrQZ
https://miro.com/pt/wireframe/o-que-e-wireframe/
https://www.lucidchart.com/pages/pt/o-que-e-wireframe
https://www.totvs.com/blog/gestao-logistica/historia-da-logistica/
https://www.mecalux.com.br/blog/historia-logistica
https://www.alura.com.br/artigos/o-que-e-git-github?srsltid=AfmBOop6MjoqIl4HfPWBsLzquT_lkI_sM0xezSOEAN5fCazzEp-83Mmj
https://logsmartbrasil.com.br/2024/05/14/o-que-e-procedimento-operacional-padrao-pop/
https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408
https://www.impacta.com.br/blog/voce-sabe-o-que-e-visual-studio/

GLOSSÁRIO

Application Programming Interface (API): Conjunto de regras e protocolos que permitem que diferentes softwares se comuniquem entre si.

Abastecimento de Linha de Produção: Processo de garantir que a linha de produção tenha todos os materiais e produtos necessários para operar continuamente e sem interrupções.

Asynchronous JavaScript and XML (AJAX): Técnica de desenvolvimento web que permite a atualização de partes de uma página da internet sem a necessidade de carregá-la inteira.

Backend: A parte de um sistema que processa informações e interage com o banco de dados, garantindo o funcionamento e a segurança da aplicação.

Banco de dados: Um sistema organizado para armazenar, gerenciar e recuperar informações de forma eficiente. Permite que grandes volumes de dados sejam estruturados, categorizados e acessados rapidamente.

Cadastro de Lotes: Registro de informações sobre lotes específicos de produtos ou materiais, incluindo detalhes como número do lote, data fabricação, validade e outras características importantes para o controle de qualidade e rastreabilidade.

Cadastro de Produtos Embalados para Expedição: Registro de informações sobre produtos que foram embalados e estão prontos para serem enviados aos clientes.

Cascading Style Sheets (CSS): Linguagem de marcação usada para estilizar elementos escritos como o HTML.

Controle de Estoque: Sistema e processos usados para monitorar e gerenciar os níveis de inventário de produtos e materiais em um armazém ou de depósito.

Controle de Expedição: É o conjunto de processos e atividades envolvidos na preparação, verificação e envio de mercadorias para o destino final.

Controle de Picking: Processo de registro e gestão das atividades relacionadas à coleta de itens no armazém para atender a um pedido ou operação.

Desenvolvimento web: O processo de criar e manter sites e aplicações, envolvendo frontend e backend. Inclui SEO, acessibilidade e segurança , com alta demanda por desenvolvedores devido à importância da presença online.

Django: Um framework web para Python que acelera o desenvolvimento de aplicações, usando a arquitetura Model-View-Template(MTV) e oferecendo uma interface administrativa, com foco em segurança e escalabilidade.

Ficha de Controle de Estoque: Documento ou formulário utilizado para registrar e monitorar informações detalhadas sobre o estoque de produtos, como quantidades recebidas, atendidas e em estoque, bem como movimentação e ajustes.

Flask: É um microframework web para Python, leve e flexível, ideal para aplicações simples e escaláveis. Permite a escolha de bibliotecas, é simples de usar e suportar APIs RESTful, sendo uma opção para desenvolvimento ágil.

Frameworks: Uma estrutura de software que fornece ferramentas e bibliotecas para facilitar o desenvolvimento de aplicações, permitindo que os desenvolvedores se concentrem na lógica de negócios e aumentando a eficiência.

Frontend: É a parte de um site ou aplicativo na qual os usuários interagem. Ele envolve a criação de layouts, botões e textos usando HTML, CSS e JavaScript. O objetivo é oferecer uma experiência de usuários intuitiva e agradável.

Gestão de Estoque: Conjunto de práticas e ferramentas utilizadas para manter o equilíbrio adequado entre a oferta e a demanda de produtos.

Hypertext Markup Language (HTML): Linguagem de marcação usada para criar e estruturar páginas da web.

Informática Aplicada: Uso de sistemas e ferramentas de informática para resolver problemas e otimizar processos em diversas áreas.

JavaScript (JS): Linguagem de programação amplamente utilizada para criar interatividade e dinâmica em páginas web.

JavaScript Object Notation (JSON): Formato leve para troca de dados estruturados, usado principalmente para comunicação entre clientes e servidores na web.

Matplotlib: É uma biblioteca de Python para criar gráficos e visualizações de dados. É personalizável e é frequentemente usada com outras bibliotecas, como NumPy e Pandas, para análise de dados.

MySQL: Um sistema de gerenciamento de banco de dados relacional de código aberto, amplamente utilizado para armazenar e gerenciar grandes volumes de dados.

NoSQL: Uma categoria de banco de dados que não usa o modelo relacional, focada em escalabilidade e flexibilidade para grandes volumes de dados e aplicações com dados não estruturados.

NumPy: É uma biblioteca Python para trabalhar com arrays e realizar operações matemáticas de forma eficiente, essencial em computação científica e ciência de dados.

Pandas: É uma biblioteca Python para análise de dados, que facilita a manipulação e limpeza de conjuntos de dados com estruturados como DataFrames.

Planilha Dinâmica: Ferramenta de software que permite a criação e o gerenciamento de dados em tabelas interativas.

Procedimento Operacional Padrão (POP): É um documento que descreve detalhadamente os procedimentos e normas que devem ser seguidos para realizar tarefas ou processos específicos de forma consistente e eficiente.

Python: É uma linguagem de programação amplamente utilizada para desenvolvimento de aplicações web, software, ciência de dados e aprendizado de máquina (ML).

Registro de Não Conformidade (RNC): É um documento ou registro formal que identifica e descreve a ocorrência de um produto, serviço ou processo que não atende aos requisitos estabelecidos ou padrão de qualidade.

Scikit-learn: É uma biblioteca Python para aprendizado de máquina, que oferece ferramentas para criar e avaliar modelos de classificação e regressão.

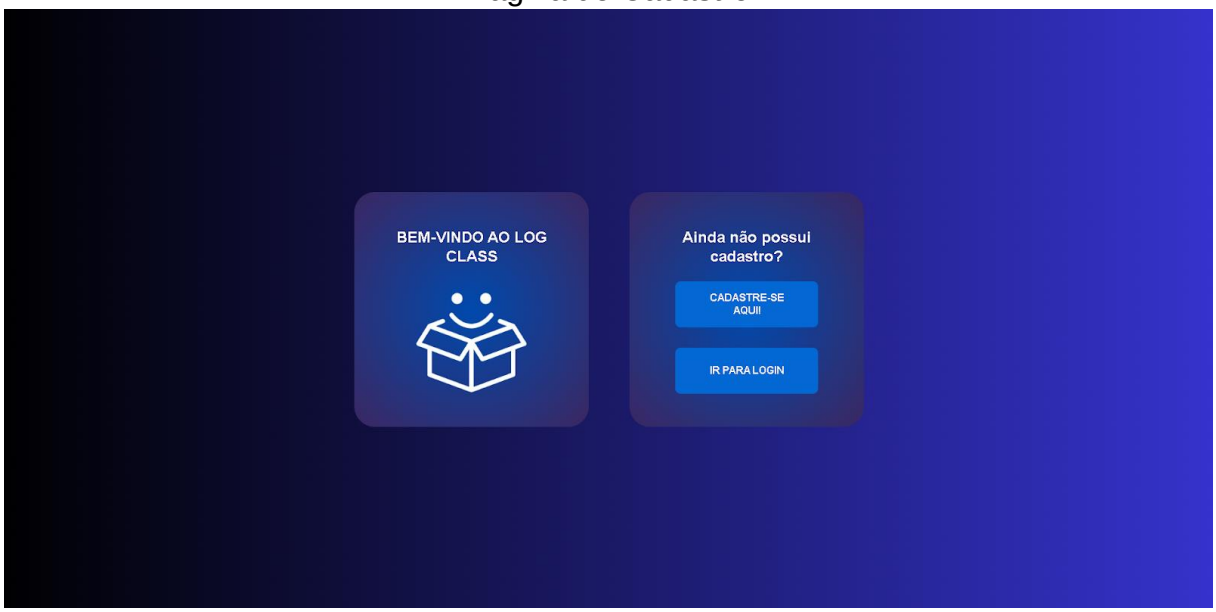
Search Engine Optimization (SEO): Um conjunto de práticas que melhora a visibilidade de um site nos resultados de busca orgânica, aumentando o tráfego e a relevância por meio de otimização de conteúdo e estrutura.

Structured Query Language (SQL): Linguagem padrão para gerenciar e manipular dados em sistemas de banco de dados relacionais.

Systems, Applications, and Products in Data Processing (SAP): É um sistema integrado de gestão empresarial amplamente utilizado para gerenciar e coordenar diferentes áreas como finanças, logística, produção, recursos humanos e vendas, fornecendo uma visão consolidada e em tempo real das operações e dados da empresa.

APÊNDICE

Página de Cadastro



HTML

```
<div class="container-bem-vindo" id="intro-section">
    <div class="inicio-bem-vindo">
        <h2 id="logo-h2">BEM-VINDO AO LOG CLASS</h2>
        <figure>
            <img src="/static/img/logo.png" alt="Logo" class="logo-image">
        </figure>
    </div>

    <div class="login-prompt">
        <h3 class="login-question">Ainda não possui cadastro?</h3>
        <button class="signup-btn" id="signup-btn"> CADASTRE-SE AQUI!</button>
        <button class="login-btn" id="login-btn"> IR PARA LOGIN </button>
    </div>
</div>
```

CSS

```
body{
    overflow-x: hidden;
    margin: 0;
    padding: 0;
    background: rgb(0,0,0);
    background: var( --cor-fundo-login);
    font-family: Arial, sans-serif;
    color: var(--cor-padrao-letras);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
```

```
.container-bem-vindo {
    align-items: baseline;
    justify-content: center;
    align-content: center;
    gap: 4rem;
    flex-wrap: wrap;
    display: flex;
    flex-direction: row;
}

.inicio-bem-vindo, .login-prompt{
    height: 20rem;
    padding: 2rem;
    border-radius: 2rem;
    width: 20rem;
    text-align: center;
    padding: 1.5rem;
    background: var(--cor-cards-pgLogin);
}

.inicio-bem-vindo h2 {
    margin-top: 2rem;
    font-size: 1.5rem;
    color:var(--cor-padrao-letras);
}

.login-prompt h3 {
    margin-top: 2rem;
    font-size: 1.5rem;
    color:var(--cor-padrao-letras);
}

.signup-btn, .login-btn {
    padding: 1rem 2rem;
    border: none;
    border-radius: 0.5rem;
    background-color: var(--cor-btns);
    color: var(--cor-padrao-letras);
    font-weight: bold;
    cursor: pointer;
    height: 4.5rem;
    width: 70%;
    font-size: 1rem;
}

.register-box, .connect-box {
    background: var(--cor-fundo-inputs);
    background: var(--cor-cards-pgLogin);
    border-radius: 1rem;
    padding: 4rem;
    text-align: center;
```

```
}

.register-box h2, .connect-box h2 {
    margin-top: 1.5rem;
    margin-bottom: 2rem;
    font-weight: bold;
    font-size: 1.7rem;
    color: var( --text-color-dark);
}

.connect-box{
    width: 18rem;
    height: 20rem;
}

.connect-box p{
    font-size: 1rem;
    color: var(--text-color-dark);
}

.register-box{
    width: 20rem;
    height: 28rem;
}

form {
    display: flex;
    flex-direction: column;
    gap: 1rem;
}

input {
    padding: 1rem;
    width: 97%;
    border: none;
    border-radius: 0.5rem;
    outline: none;
    background-color: var(--cor-fundo-inputs);
    color: var(--cor-padrao-letras);
}

button[type="submit"] {
    padding: 1rem;
    border: none;
    border-radius: 0.5rem;
    background-color: var(--cor-btns);
    color: var(--cor-padrao-letras);
    font-weight: bold;
    cursor: pointer;
    width: 97%;
    font-size: 1.2rem;
}
```

```
.signup-link {
    display: block;
    margin-top: 1rem;
    color: var(--cor-btns);
    text-decoration: none;
}

.account-type {
    padding: 1rem;
    border: none;
    border-radius: 0.5rem;
    background-color: var(--cor-btns);
    color: var(--cor-padrao-letras);
    font-weight: bold;
    width: 100%;
    margin-top: 1rem;
    cursor: pointer;
    font-size: 1.2rem;
}

.select-menu {
    position: relative;
    margin-bottom: 1.5rem;
}

.select-btn {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem;
    color: #686868;
    cursor: pointer;
    border-radius: 0.5rem;
    width: 90%;
}

.select-btn i {
    font-size: 1.5rem;
    transition: 0.3s;
}

.select-menu.active .select-btn i {
    transform: rotate(-180deg);
}

.options {
    position: absolute;
    top: 100%;
    left: 0;
    width: 80%;
    padding: 0;
```

```
    list-style: none;
    border-radius: 1.5rem;
    max-height: 5rem;
    overflow-y: auto;
    display: none;
}

.select-menu.active .options {
    display: block;
    z-index: 10;
}

.option {
    padding: 1rem;
    cursor: pointer;
}

.option:hover {
    background-color: #696969;
}

.login-box{
    width: 18rem;
    height: 21rem;
}

.logo-image {
    margin-top: 1.5rem;
    width: 12rem;
    display: block;
    margin-left: auto;
    margin-right: auto;
}

.login-btn {
    background-color: var(--cor-btns);
    color: var(--cor-padrao-letras);
    padding: 1rem;
    border: none;
    border-radius: 0.5rem;
    margin-top: 2rem;
    font-weight: bold;
    cursor: pointer;
}

#register-aluno, #register-professor {
    display: none;
}
.active {
    display: block;
}
```

```css
#login-existente-box {
    background: var(--cor-btns);
    background: var(--cor-cards-pgLogin);
    border-radius: 1rem;
    padding: 3rem;
    width: 3rem;
    text-align: center;
    margin: 0 auto;
}

#login-existente-box h2 {
    margin-bottom: 2rem;
    font-weight: bold;
}

#login-existente-box form {
    display: flex;
    flex-direction: column;
    gap: 1rem;
}

#email-login, #senha-login {
    padding: 1rem;
    width: 90%;
    border: none;
    border-radius: 1.5rem;
    outline: none;
    background-color: var(--cor-fundo-inputs);
    color: var(--cor-padrao-letras);
}

#login-existente-btn {
    padding: 1rem;
    border: none;
    border-radius: 1.5rem;
    background-color: var(--cor-btns);
    color: var(--cor-padrao-letras);
    font-weight: bold;
    cursor: pointer;
}

.signup-link {
    display: block;
    margin-top: 1rem;
    color: var(--cor-btns);
}

.register-box, .login-box, .connect-box {
    background: var(--cor-cards-pgLogin);
    border-radius: 1rem;
    padding: 1.5rem;
    text-align: center;
```

```
    display: none;
}


.voltar-btn {
    display: block;
    width: 97%;
    margin-top: 0.5rem;
    border: none;
    border-radius: 1.5rem;
    color: var(--cor-padrao-letras);
    font-weight: 1rem;
    cursor: pointer;
    text-align: center;
    text-decoration: none;
}
```

JavaScript

```
document.addEventListener("DOMContentLoaded", function() {
    // Botões e seções principais
    const signupBtn = document.getElementById("signup-btn");
    const loginBtn = document.getElementById("login-btn");
    const introSection = document.getElementById("intro-section");
    const registerSection = document.getElementById("register-section");
    const loginSection = document.getElementById("login-section");

    // Seções de cadastro
    const alunoRegisterBtn = document.getElementById("aluno-register-btn");
    const professorRegisterBtn = document.getElementById("professor-register-
btn");
    const registerAluno = document.getElementById("register-aluno");
    const registerProfessor = document.getElementById("register-professor");
    const connectBoxRegister = document.querySelector("#register-section
.connect-box");

    // Seções de login
    const alunoLoginBtn = document.getElementById("aluno-login-btn");
    const professorLoginBtn = document.getElementById("professor-login-btn");
    const loginAluno = document.getElementById("login-aluno");
    const loginProfessor = document.getElementById("login-professor");
    const connectBoxLogin = document.getElementById("connect-login");

    // Função para exibir a tela de cadastro
    signupBtn.addEventListener("click", function() {
        introSection.style.display = "none";
        registerSection.style.display = "flex";
        connectBoxRegister.style.display = "block";
        registerAluno.style.display = "none";
```

```
        registerProfessor.style.display = "none";
    });

    function exibirLogin(){
        introSection.style.display = "none";
        loginSection.style.display = "block";
        connectBoxLogin.style.display = "block";
        loginAluno.style.display = "none";
        loginProfessor.style.display = "none";
    }

    // Função para exibir a tela de login
    loginBtn.addEventListener("click", function() {
        exibirLogin()
    });

    // Função para exibir o cadastro do aluno
    alunoRegisterBtn.addEventListener("click", function() {
        connectBoxRegister.style.display = "none";
        registerAluno.style.display = "block";
        registerProfessor.style.display = "none";
    });

    // Função para exibir o cadastro do professor
    professorRegisterBtn.addEventListener("click", function() {
        connectBoxRegister.style.display = "none";
        registerProfessor.style.display = "block";
        registerAluno.style.display = "none";
    });

    function exibirLoginAluno(){
        connectBoxLogin.style.display = "none";
        loginAluno.style.display = "block";
        loginProfessor.style.display = "none";
    };

    // Função para exibir o login do aluno
    alunoLoginBtn.addEventListener("click", function() {
        exibirLoginAluno();
    });

    function exibirLoginProfessor(){
        connectBoxLogin.style.display = "none";
        loginProfessor.style.display = "block";
        loginAluno.style.display = "none";
    };

    // Função para exibir o login do professor
    professorLoginBtn.addEventListener("click", function() {
        exibirLoginProfessor();
    });
```

```
    // Função para voltar à tela de seleção no cadastro
    document.querySelectorAll("#voltar-register").forEach(button => {
        button.addEventListener("click", function() {
            registerSection.style.display = "flex";
            connectBoxRegister.style.display = "block";
            registerAluno.style.display = "none";
            registerProfessor.style.display = "none";
        });
    });

    // Função para voltar à tela de seleção no login
    document.querySelectorAll("#voltar-login").forEach(button => {
        button.addEventListener("click", function() {
            loginSection.style.display = "block";
            connectBoxLogin.style.display = "block";
            loginAluno.style.display = "none";
            loginProfessor.style.display = "none";
        });
    });

    // Função para enviar os dados de cadastro do aluno e verificar duplicatas
    document.querySelector('#register-aluno form').addEventListener('submit',
function(event) {
        event.preventDefault(); // Previne o envio automático do formulário

        // Coleta os dados do formulário
        const nome = document.querySelector("input[name='nome']").value;
        const email = document.querySelector("input[name='email']").value;
        const senha = document.querySelector("input[name='senha']").value;
        const turma = document.querySelector("input[name='turma']").value;

        // Cria um objeto com os dados
        const dados = {
            tipo: "Aluno",  // Adicione o tipo de usuário aqui
            nome: nome,
            email: email,
            senha: senha,
            turma: turma
        };

        // Envia os dados para o backend para verificar duplicatas
        fetch('/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(dados),
        })
        .then(response => {
            if (response.status === 409) {
                alert('Erro: Usuário já cadastrado com este email.');
            } else if (response.status === 201) {
```

```
                alert('Cadastro realizado com sucesso! Realize o login.');
                window.location.href = '/login'; // Redireciona para o login
            } else {
                alert('Erro ao tentar realizar o cadastro.');
            }
        })
        .catch(error => {
            console.error('Erro ao realizar o cadastro:', error);
        });
    });

    // Função para enviar os dados de cadastro do professor e verificar
duplicatas
    document.querySelector('#register-professor
form').addEventListener('submit', function(event) {
        event.preventDefault(); // Previne o envio automático do formulário

        // Coleta os dados do formulário
        const nome = document.querySelector("#register-professor
input[name='nome']").value;
        const email = document.querySelector("#register-professor
input[name='email']").value;
        const senha = document.querySelector("#register-professor
input[name='senha']").value;

        // Cria um objeto com os dados
        const dados = {
            tipo: "Professor", // Adicione o tipo de usuário aqui
            nome: nome,
            email: email,
            senha: senha
        };

        // Envia os dados para o backend para verificar duplicatas
        fetch('/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(dados),
        })
        .then(response => {
            if (response.status === 409) {
                alert('Erro: Usuário já cadastrado com este email.');
            } else if (response.status === 201) {
                alert('Cadastro realizado com sucesso! Realize o login.');
                window.location.href = '/login'; // Redireciona para o login
            } else {
                alert('Erro ao tentar realizar o cadastro.');
            }
        })
        .catch(error => {
```

```
            console.error('Erro ao realizar o cadastro:', error);
        });
    });

    // Função para enviar os dados de login do aluno
    document.querySelector('#login-aluno form').addEventListener('submit',
function(event) {
        event.preventDefault(); // Previne o envio automático do formulário

        // Coleta os dados do formulário
        const email = document.querySelector("#login-aluno
input[name='email']").value;
        const senha = document.querySelector("#login-aluno
input[name='senha']").value;
        const turma = document.querySelector("#login-aluno
input[name='turma']").value;

        // Cria um objeto com os dados
        const dados = {
            tipo: "LoginAluno",
            email: email,
            senha: senha,
            turma: turma
        };

        // Envia os dados para o backend para realizar o login
        fetch('/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(dados),
        })
        .then(response => {
            if (response.ok) {
                // Login bem-sucedido
                alert('Login realizado com sucesso! Seja bem vindo ao ambiente
educacional.');
                window.location.href = '/'; // Redireciona para a página
inicial
            } else {
                alert('Email ou senha incorretos.');
            }
        })
        .catch(error => {
            console.error('Erro ao tentar fazer login:', error);
        });
    });

    // Função para enviar os dados de login do professor
    document.querySelector('#login-professor form').addEventListener('submit',
function(event) {
```

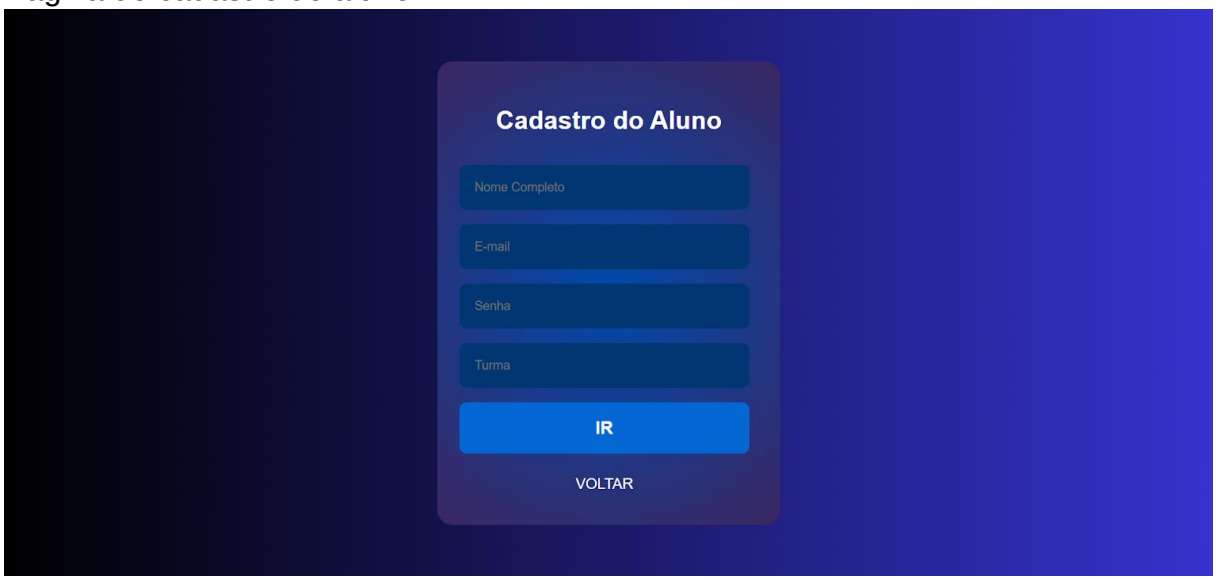
```
        event.preventDefault(); // Previne o envio automático do formulário

        // Coleta os dados do formulário
        const email = document.querySelector("#login-professor
input[name='email']").value;
        const senha = document.querySelector("#login-professor
input[name='senha']").value;

        // Cria um objeto com os dados
        const dados = {
            tipo: "LoginProfessor",
            email: email,
            senha: senha
        };

        // Envia os dados para o backend para realizar o login
        fetch('/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(dados),
        })
        .then(response => {
            if (response.ok) {
                // Login bem-sucedido
                alert('Login realizado com sucesso! Seja bem vindo ao ambiente
educacional.');
                window.location.href = '/'; // Redireciona para a página
inicial
            } else {
                alert('Email ou senha incorretos.');
            }
        })
        .catch(error => {
            console.error('Erro ao tentar fazer login:', error);
        });
    });



});
```

Python

```
@app.route("/login", methods=["GET", "POST"])
def pagina_cadastro():
    if request.method == "GET":
        # conectando com o banco de dados
        mydb = Conexao.conectar()
        # criando um objeto Aluno
```

```
        mycursor = mydb.cursor()
        # criando uma variável para armazenar a lista de turmas
        mycursor.execute("SELECT * FROM databaseprofessor.tb_database")
        resultado = mycursor.fetchall()
        mydb.close()

        # criando uma lista para armazenar todas as turmas que foram
"retirados"
        lista_nomes = [{"database": nomeBD[0]} for nomeBD in resultado]
        return render_template("login.html", lista_nomes=lista_nomes)
```

Página de cadastro do aluno

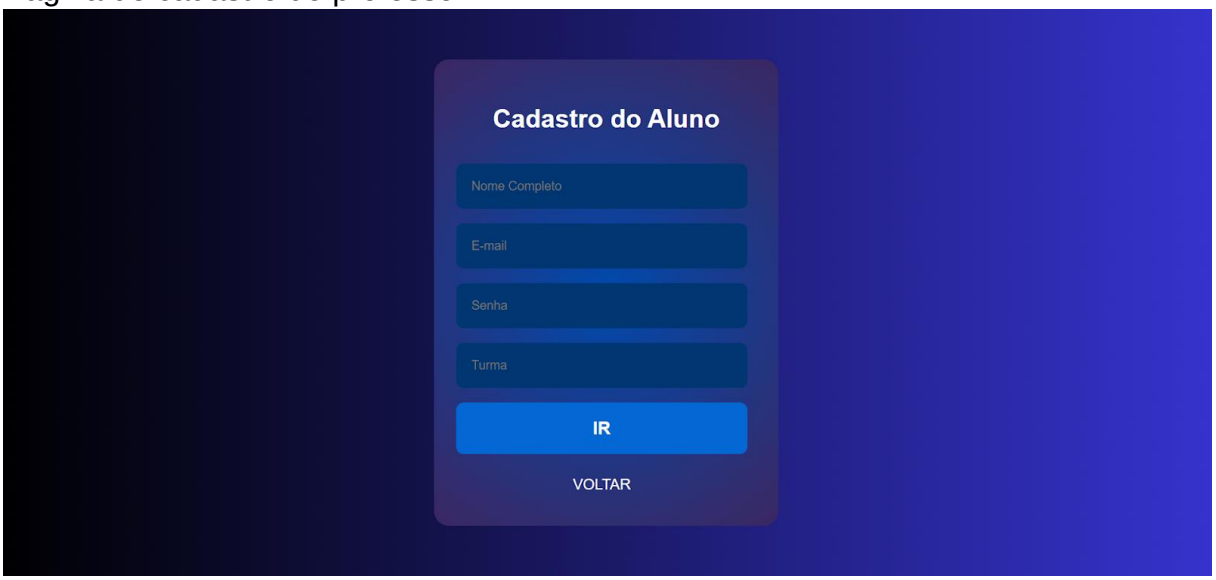


Python

```
# realizando o cadastro do aluno
        if formulario == "Aluno":
            # pegando os dados do formulário, mas em forma de json
            nome = request.json.get("nome")
            email = request.json.get("email")
            senha = request.json.get("senha")
            turma = request.json.get("turma")
            # criando um objeto Aluno
            aluno = Aluno()

            # verificando, por meio de uma função dentro do objeto aluno, se
existem duas pessoas com os mesmos cadastros no banco de dados
            if aluno.verificar_duplicata(email, turma):
                # retornando um arquivo json para caso haja usuários com esses
dados
                return jsonify({'mensagem': 'Usuário já cadastrado'}), 409

            # realizando o cadastro do usuário
            if aluno.cadastrar(nome, email, senha, turma):
                # retornando um arquivo json confirmando o cadastro realizado
com sucesso
```

```
                return jsonify({'mensagem': 'Cadastro realizado com
sucesso'}), 201
            else:
                # retornando um arquivo json caso o cadastro nao seja
concluído
                return jsonify({'mensagem': 'Erro ao cadastrar o aluno'}), 400
```

Página de cadastro do professor

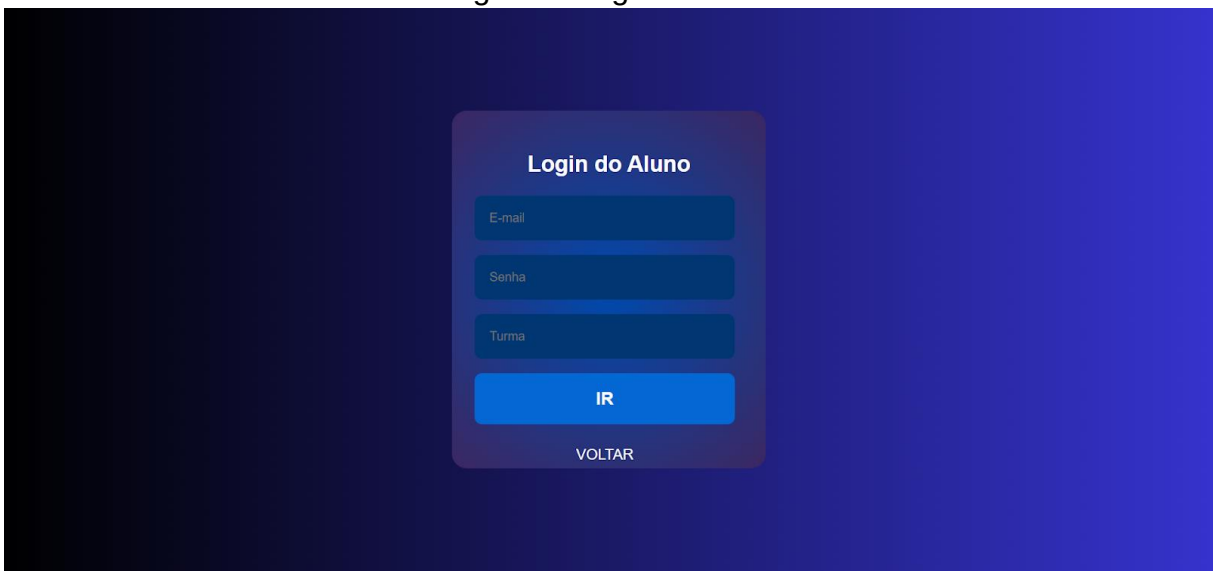


Python

```
# realizando o cadastro do professor
        if formulario == "Professor":
            # pegando os dados do formulário, mas em forma de json
            nome = request.json.get("nome")
            email = request.json.get("email")
            senha = request.json.get("senha")

            # criando um objeto para armazrnar a classe Professor
            professor = Professor()

            # verificando se já existe um usuário cadastrado esses dados no
banco de dados
            if professor.verificar_duplicata(email):
                # retornando um arquivo json caso haja usuários com esses
dados
                return jsonify({'mensagem': 'Usuário já cadastrado'}), 409

            # verificando se o usuário cadastrado inserio a senha correta de
acesso
            if senha == "logclass":
                # realizando o cadastro do professor através da função que
está dentro do objeto
                if professor.cadastrarProf(nome, email, senha):
                    # retornando um arquivo json confirmando o cadastro
realizado com sucesso
```

```
                    return jsonify({'mensagem': 'Cadastro realizado com
sucesso'}), 201
                else:
                    # retornando um arquivo json caso o cadastro nao seja
concluído
                    return jsonify({'mensagem': 'Erro ao cadastrar o
professor'}), 400

            else:
                # retornando um arquivo json caso a senha inserida seja
incorreta
                return jsonify({'mensagem': 'Senha incorreta'}), 401
```

Página de login do aluno

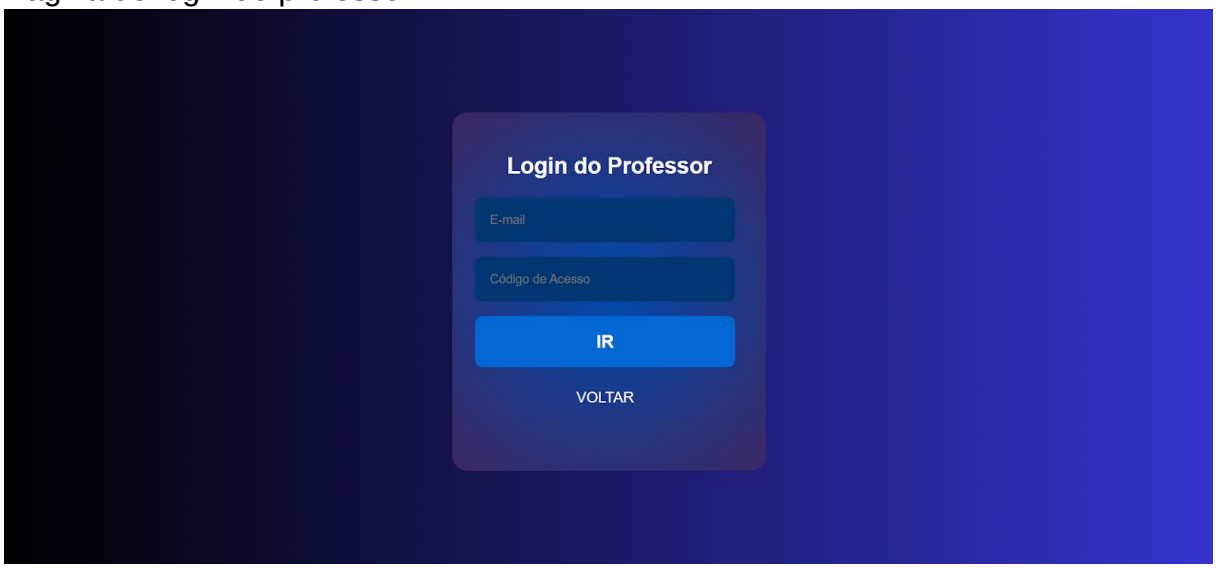


Python

```
formulario = request.json.get("tipo")
        if formulario == "LoginAluno":
            # pegando os dados do formulário, mas em forma de json
            email = request.json.get("email")
            senha = request.json.get("senha")
            turma = request.json.get("turma")

            # criando um objeto com a classe Aluno
            loginAluno = Aluno()

            # realizando o login do aluno por meio da função armazenada na
variável
            if loginAluno.logar(email, senha, turma):
                # armazenando os dados em uma session para poder consultar
posteriormente
                session['usuario_logado'] = {
                    'email': loginAluno.email,
                    'turma': loginAluno.turma,
```

```
                    'nome': loginAluno.nome,
                    'cod_aluno': loginAluno.cod_aluno
                }
                # validação por meio de um alert na tela do usuário, para
quando o login der certo
                flash("alert('Muito Bem Vindo ao seu ambiente
educacional!!')")
                return redirect('/')
            else:
                # limpando a session caso o login esteja errado
                session.clear()
                return 'Email ou senha incorretos.', 401
```

Página de login do professor



Python

```
if formulario == "LoginProfessor":
            # pegando os dados do formulário, mas em forma de json
            email = request.json.get("email")
            senha = request.json.get("senha")

            # criando um objeto com a classe Professor
            loginProfessor = Professor()

            # realizando o login do professor por meio da função armazenada na
variável
            if loginProfessor.logarProf(email, senha):
                # armazenando os dados em uma session para poder consultar
posteriormente
                session['professor_logado'] = {
                    'email': loginProfessor.email_prof,
                    'nome': loginProfessor.nome_prof,
                    'turma': "databaseProfessor",
                    'senha': loginProfessor.senha_espec,
                    'cod_aluno': loginProfessor.cod_aluno
```

```
                }
                # validação por meio de um alert na tela do usuário, para
quando o login der certo
                flash("alert('Muito Bem Vindo ao seu ambiente
educacional!!')")
                return redirect('/')
            else:
                # limpando a session caso o login esteja errado
                session.clear()
                            return 'Email ou senha incorretos.', 401
```

Página inicial - Layout

HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LogClass</title>
    <!-- importando o arquivo css da página -->
    <link rel="stylesheet" href="/static/styles/layout.css">
    <link rel="stylesheet" href="/static/styles/variaveis.css">

    <script src="https://kit.fontawesome.com/cbfeaec436.js"
crossorigin="anonymous"></script>

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap"
rel="stylesheet">

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
```

```
    <link
href="https://fonts.googleapis.com/css2?family=Julius+Sans+One&display=swap"
rel="stylesheet">

    <script src="/static/js/menu.js"></script>
    <script src="/static/js/light&dark.js" defer></script>


    {% block css %}
    {% endblock %}


    <!-- Importando os ícones do bootstrap -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css">
</head>
<body class="theme-container">
{% if "professor_logado" in session or "usuario_logado" in session %}
    <!-- Início do Cabeçalho -->
    <header class="sidebar">

        <!-- Início da Seção do cabeçalho -->
        <section class="header-container">

         <div class="container-logo">
            <!-- Imagem e logo da empresa -->
            <figure>

                <a href="/"><img src="/static/img/logclass_dark.svg"
class="dark" alt="logodark"></a>
                <a href="/"><img src="/static/img/LOGCLASS.svg" class="light"
alt="logo"></a>
                <!-- <figcaption>Logo da LogClass</figcaption> -->
            </figure>
        </div>



            <!-- Início da navegação principal -->
            <nav>

                <div class="dropdown">
                    <button class="dropbtn">Processos</button>
                    <div class="dropdown-content">
                        <div class="card">
                            <a href="/cadastramento">Cadastramento</a>
                        </div>

                        <div class="card">
                            <a href="/rnc">RNC</a>
                        </div>
```

```
                    <div class="card">
                        <a href="/picking">Cadastro de Picking</a>
                    </div>

                    <div class="card">
                        <a href="/estoque">Controle de Estoque</a>
                    </div>

                    <div class="card">
                        <a href="/pop">POP</a>
                    </div>

                    <div class="card">
                        <a href="/expedicao">Cadastro de Expedição</a>
                    </div>
                    <div class="sombreado"></div>
                </div>
            </div>


            {% if "professor_logado" in session %}
            <!-- Início do menu dropdown -->
            <div class="dropdown">
                <button onclick="myFunction()" class="dropbtn">Gerenciar
turmas</button>
                <div id="myDropdown" class="dropdown-content">
                    <div class="dropdown-menu">
                        {% if "professor_logado" in session %}

                            <div class="card">
                                <a href="/criarBD">Crie sua Turma</a>
                            </div>
                            <div class="card">
                                <a href="/professor/listarBD">Gerenciar
sua turma</a>
                            </div>
                            <div class="card">
                                <a href="/enviar_mensagem">Enviar
Mensagem</a>
                            </div>
                            <div class="card">
                                <a href="/redefinir_senha">Redefinir senha
dos alunos</a>
                            </div>
                        {% endif %}

                            <div class="sombreado"></div>
                    </div>
                </div>

            </div>
```

```
                {% endif %}

                <!-- Fim do menu dropdown -->
            </nav>

            <!-- Fim da navegação principal -->

            <div class="inventario">
                <a href="/inventario">Inventário</a>
            </div>


            <button class="button-container" id="toggle-theme"><i class="fa-
regular fa-lightbulb"></i></button>

            <figure class="sair">
                <a href="/logoff">
                    <img src="/static/img/box-arrow-right.svg" alt="ícone de
sair">
                </a>
            </figure>

        </section>
        <!-- Fim da Seção do cabeçalho -->

    </header>
{% endif %}
    <!-- Fim do Cabeçalho -->

    <!-- Bloco de conteúdo dinâmico -->
    {% block conteudo %}
    {% endblock %}

    <!-- Início do rodapé -->
    <footer>
        <p>&copy; 2024 LogClass Sistema Educacional</p>
    </footer>
    <!-- Fim do rodapé -->


</body>
</html>
```

CSS

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  font-family: Verdana, Geneva, Tahoma, sans-serif;
  font-weight: 700;
```

```
  list-style: none;
  font-size: 1rem;
  text-decoration: none;
}

/* ****************************** padronização/ estilização MENU
********************************** */

.theme-container {
  height: 100vh;
  width: 100vw;
}

/* Estilização do cabeçalho */
.header-container {
  width: 100%;
  height: 16vh;
  display: flex;
  align-items: center; /* Centraliza os itens verticalmente */
  flex-direction: row;
  position: fixed;
  left: 0;
  top: 0;
  margin: 0;
  transition: background-color 0.3s;
  z-index: 10;
}


.dropdown {
  flex-direction: row;
  align-items: center;
  gap: 1rem; /* Ajusta o espaço entre os itens do menu */
  position: relative;
  display: inline-block;
}

.container-logo{
  position: relative;
  height: 16vh;
  width: 29%;

}

/* Estilização da imagem do logo */
.container-logo img{
  left: 40px;
  margin-top: 1rem;
  margin-right: 16rem;
  margin-left: 8rem;
  position: absolute;
}
```

```
.light img{
  width: 5rem;
  height: 6rem;
}

.dark img{
  width: 5rem;
  height: 6rem;
  display: none;
}


/* Estilização do botão de dropdown */

.dropbtn{
  background-color: transparent;
  padding: 1rem 2rem;
  font-family: "Bebas Neue", sans-serif;
  font-weight: 400;
  font-style: normal;
  border: none;
  border-radius: 2rem;
  cursor: pointer;
  text-transform: uppercase;
  margin-bottom: 1.7rem;
  margin-top: 2rem;
  font-size: 1.8rem;
  margin-right: 4rem;

}

.dropdown {
  z-index: 1;
  height: 15vh;
  display: inline-block;

}

.dropdown-content {
  left: 0; /* Alinha à esquerda do pai */
  right: 0; /* Alinha à direita do pai, cobrindo 100% */
  background-color: var(--cor-drop);
  z-index: 15;
  border-radius: 1rem;
  padding: 1rem;
  width: 100%; /* Ocupa 100% do pai */
  height: 50vh;
  display: none;
  position: fixed; /* Independente do botão e do header */
  top: 5.5rem; /* Espaço abaixo do header */
  width: 100vw; /* Largura total da viewport */
```

```
  z-index: 15;
  padding: 1rem 0;
  min-width:8rem;
  margin-right: 2rem;

}

.dropdown-content a {
  color: var(--cor-drop-letras);
  padding: 0.8rem 1rem;
  text-decoration: none;
  color: #333;
  font-weight: bold;
  font-size: 1rem;
  display: flex;
  align-items: center;
  gap: 2rem;
}

.dropdown-content a i {
  font-size: 1rem;
  color: var(--cor-icone-cadastramento);
}

.dropdown a:hover {
  background-color: var(--cor-drop-hover);

}

.show {
  display: block;
}

.dropdown-menu .card a i {
  font-size: 1rem;
  color: var(--cor-icone-cadastramento);
}

.dropdown-menu .card a {
  text-decoration: none;
  color: #333;
  font-weight: bold;
  font-size: 1rem;
  display: flex;
  align-items: center;
  gap: 1rem;
}

.dropdown:hover .dropdown-content {
  display: block;
}
```

```
.dropdown-menu {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1rem;
}

.dropdown-menu .card {
  padding: 1rem;
  background-color: var(--cor-fundo-options);
  border-radius: 1rem;
  text-align: center;
  transition: transform 0.2s;
}

.dropdown-menu .card:hover {
  transform: scale(1.01);
  background-color: var(--cor-hover-options);
}

.dropdown-menu .card a {
  text-decoration: none;
  color: var(--cor-drop-letras);
  font-weight: bold;
  font-size: 1rem;
  display: flex;
  align-items: center;
  gap: 8rem;
}

.sombreado{
  height: 56vh;
  opacity: 0.7;
  width: 100vw;
  background-color: #333;
}

.inventario{
  text-decoration: none;
  margin-top: 1rem;
}

.inventario a{
  color: var(--primary-color-dark);
  font-family:"Bebas Neue", sans-serif ;
  font-size: 1.8rem;
  font-weight: 300;
  margin-right: 20rem;
}

.button-container {
  height: 3rem;
```

```
  width: 4rem;
  border-radius: 1.5rem;
  border: none;
  background-color: white;
  margin-left:2rem;
}


.conectar{
  padding: 3rem;
  margin-left: 10rem;
  margin-top: 2.5rem;
  margin-left: 20rem;
  margin-top: 2rem;
  margin-bottom: 2rem;
}

.conectar a{
  color: var(----cor-titulos);
}

.header-container .sair img{
  height: 3rem;
  width: 4rem;
  margin-top: 0;
}

/* ***************************** Padronização/ estilização RODAPÉ
********************************** */

/* Estilização do rodapé */
footer {
  width: 100vw;
  height: 4vh;
  background-color: var(--cor-footer);
  position: fixed;
  bottom: 0;
  left: 0;
  color: var(--cor-padrao-letras);
  display: flex;
  justify-content: center;
  align-items: center;
}

footer p {
  /* margin-left: 2rem; */

  font-family: "Nanum Gothic", sans-serif;
  font-weight: 700;
  font-style: normal;
  margin-top: 0.5rem;
}
```

```
/* ************************************ MODO CLARO E ESCURO
****************************************** */



.theme-container {
  margin: 0;
  transition: background-color 0.3s, color 0.3s;
}

.theme-container.light {
  color: var(--text-color-light);
}

.theme-container.dark {
  background-color: var(--primary-color-dark);
  color: var(--text-color-dark);
}

.header-container {
  transition: background-color 0.3s;
}

.theme-container.dark .header-container {
  background-color: var(--sidebar-bg-dark);
}

.theme-container.dark td {
  color: var( --text-color-light);
}

.theme-container.dark td {
  border: var(--sidebar-bg-dark);
}

.theme-container.dark .dropdown{
  background-color: var(--secondary-color-dark);
}

.theme-container.dark .sombreado{
  background-color: var( --secondary-color-light);
}

.theme-container.dark .container-grid a{
  color: var(--text-color-dark);
}

.theme-container.dark .inventario a{
  color: var(--text-color-dark);
}
```

```
.theme-container.light .login-box h2{
  color: var(--text-color-dark);
}

.theme-container.dark .dropbtn{
  background-color: var(--text-color-dark);
}

.theme-container.dark .expedicao1{
  color: var(--text-color-dark);
}

.theme-container.dark .dark{
  display: block;
}
```

JavaScript - Menu

```
document.addEventListener("DOMContentLoaded", function () {
    const menuToggle = document.getElementById("menu-toggle");
    const mainMenu = document.getElementById("main-menu");

    menuToggle.addEventListener("click", function () {
        mainMenu.classList.toggle("show");
    });
});
```

```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}

<link rel="stylesheet" href="/static/styles/pagina-inicial.css">
<link rel="stylesheet" href="/static/styles/media/pagina-inicial-media.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">

<script src="https://unpkg.com/scrollreveal" defer></script>
<script src="/static/js/pagina-inicial.js" defer></script>
<script src="/static/js/light&dark.js" defer></script>

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Unlock&display=swap"
rel="stylesheet">


{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}
```

```
<!-- Início da seção geral -->
<section class="container-geral">

    <!-- Início da seção principal -->
    <div class="container-principal">

        <!-- Título da página -->
        <div class="titulo-destaque">
            <span class="part1">L</span>
            <div class="container">
                <div class="shape"></div>
            </div>
            <span class="partG">G</span>
            <span class="part2">CLASS</span>

        </div>

    </div>
    <!-- Fim da seção principal -->
    {% if "usuario_logado" in session %}
    <div class="container-grid">
        <a href="#mensagem">Ver mensagem do Professor <i class="fa-solid fa-
message"></i></a>
    </div>
    {% endif %}

    <!-- Início da seção de cards -->
    <div class="container-cards">
        <div class="c-galeria">

            <!-- Card: Processo de cadastramento -->
            <div class="servicos" id="servicos">
                <figure><img src="/static/img/caixas.png" alt="Processo de
cadastramento"></figure>
                <a href="/cadastramento">
                <div class="descricao">
                    <img src="/static/img/caixa_cad.png" alt="">
                </div>
                </a>
            </div>

                <div class="servicos" id="servicos" >
                    <figure><img src="/static/img/caixas.png" alt="Não
conformidade"></figure>
                    <a href="/rnc">
                    <div class="descricao">
                        <img src="/static/img/rnc.png" alt="">
                    </div>
                    </a>
                </div>
```

```
                <div class="servicos" id="servicos">
                    <figure><img src="/static/img/caixas.png"
alt="Picking"></figure>
                    <a href="/picking">
                    <div class="descricao">
                        <img src="/static/img/picking.png" alt="">
                    </div>
                    </a>
                </div>

                <div class="servicos" id="servicos">
                    <figure><img src="/static/img/caixas.png"></figure>
                    <a href="/estoque">
                    <div class="descricao">
                        <img src="/static/img/controleDeEntradaeSaída.png"
alt="entrada e saída">
                    </div>
                    </a>
                </div>

                <div class="servicos" id="servicos">
                    <figure><img src="/static/img/caixas.png"
alt="pop"></figure>
```

```
  <a href="/pop">
                    <div class="descricao">
                        <img src="/static/img/pop.png" alt="">
                    </div>
                    </a>
                </div>

                <div class="servicos" id="servicos">
                    <figure><img src="/static/img/caixas.png"
alt="expedicao"></figure>
                    <a href="/expedicao">
                    <div class="descricao">
                        <img src="/static/img/expedicao.png" alt="">
                    </div>
                    </a>
                </div>
        </div>
    </div>
    <!-- Fim da seção de cards -->

    <!-- Início da seção de mensagem do professor -->

    {% if "usuario_logado" in session %}
    <div class="titulomensagem-prof" id="mensagem">
        <h2>Mensagem do Professor</h2>
    </div>
```

```
    <div class="container-mensagem_professor" id="mensagem">
        <div class="mensagem_prof">
            {% for mensagem in lista_mensagens %}
                <div class="mensagem">
                    <p>{{ mensagem['mensagem'] }}</p>
                </div>
            {% endfor %}
        </div>
    </div>
    {% endif %}
    <!-- Fim da seção de mensagem do professor -->

</section>

<!-- Fim da seção geral -->

<script >
    {% with messages = get_flashed_messages() %}
        {% for mensagem in messages %}
            {{ mensagem|safe }}
        {% endfor %}
    {% endwith %}
</script>
{% endblock %}
```

Página inicial - CSS

```
body, html {
  margin: 0;
  padding: 0;
  font-family: Arial, sans-serif;
  overflow-x: hidden;
  transition: background-color 0.5s ease, color 0.5s ease;
  z-index: -1;
  scroll-behavior: smooth;
}

.container-geral{
  height: 100vh;
  width: 100vw;
  padding: 0;
}

.container-principal{
  display: flex;
  align-items: center;
  justify-content: space-evenly;
  flex-direction: row;
  height: 70vh;
  text-align: center;
  z-index: -5;
}
```

```
.titulo-destaque {
  display: flex;
  align-items: center;
}

.titulo-destaque .part1 {
  font-size: 22rem;
  color: var(--cor-titulo1-pgInicial);
  opacity: 0.7;
  font-family: "Unlock", serif;
  font-weight: 400;
  font-style: normal;
  /* margin-left: 3rem; */
}

.titulo-destaque .part2 {
  font-size: 22rem;
  color: var(--cor-titulo2-pgInicial);
  font-family: "Unlock", serif;
  font-weight: 400;
  font-style: normal;
}

.titulo-destaque .partG {
  font-size: 22rem;
  color: var(--cor-titulo1-pgInicial);
  opacity: 0.7;
  font-family: "Unlock", serif;
  font-weight: 400;
  font-style: normal;
  margin-left: 0.5rem;
}

.container-grid {
  color: black;
  display: flex;
  margin-bottom: 10rem;
  justify-content: center;
  align-items: center;
}

.container-grid a{
  color: black;
  font-size: 2rem;
  font-family:'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida
Sans', Arial, sans-serif;
  text-transform: uppercase;
}

.container-cards{
  height: 100vh;
```

```
  width: 100vw;
  display: flex;
  align-items: center;
  justify-content: center;
  max-width: 100%;
  margin-top: 0.5rem;
  /*DEFINE OS ESPAÇOS ENTRE AS LINHAS E COLUNAS*/
  gap: 1;
}

/*AQUI COMEÇA A GALERIA*/
.c-galeria{

  width: 100%;
  display: grid;
  grid-template-columns: repeat(3, 2fr);
  grid-auto-rows: 200px;
  grid-column-gap: 10px;
  grid-row-gap: 6.5rem;
  align-items: center;
  margin-bottom: 25rem;
  margin-top: 10rem;
  text-align: center;
  /*DEFINE O TAMANHO DO EFEITO NA IMAGEM QUANDO O MOUSE É PASSADO*/
  flex: 2 5 10px;
  float: left;
  /*DEFINE A TRANSPARÊNCIA DO ELEMENTO*/
  /* opacity: 1; */
  transition: 0.5s;
}



.servicos{
  border-radius: 0.5rem;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  position: relative;
  /*NÃO DEIXA O CARD FICAR VISIVEL SE O MESMO FICAR MAIOR QUE O PAI*/
  overflow: hidden;
  display: flex;
  /*MARGEM EXTERNA*/
  margin: 8rem;
  margin-top: 10rem;
  /*FLUTUAR*/
  float: left;
  /*LARGURA DO CARD.*/
  width: 70%;
  height: 150%;
  /*CENTRALIZA O TEXTO*/
  /*SUAVIZAR EFEITO DA TRANSIÇÃO*/
  transition: transform 0.5s;
  flex-direction: column;
}
```

```css
.servicos img{
  /*AJUSTAR A IMAGEM DE ACORDO COM A LARGURA DO CARD*/
  width: 100%;
  /*ALTURA AUTOMATICA DA IMAGEM*/
  height: 20rem;
  object-fit: cover;
  filter: brightness(80%);/*DEIXA A IMAGEM MAIS ESCURA, MAIS CINZA*/
  padding: 0;
  transition: transform 0.5s ease;
}


.servicos img:hover{
  /*NOS PERMITE DAR O EFEITO DE ZOOM E OU PERSPECTIVA*/
  transform: scale(1.1);
  filter: brightness(100%);
}

.descricao{
  position: absolute;
  /*DEIXA A DESCRICAO ABAIXO DA IMAGEM*/
  top:100%;
  height: 100%;
  padding: 3px;
  box-sizing: border-box;
  transition: top 0.3s;
  font-size: 1rem;
  margin-top: 1rem;
}

.servicos:hover .descricao{
  /*FAZ IR PARA O TOPO QUANDO O MOUSE PASSAR EM CIMA*/
  top: 0;
  background-color: var(--cor-servicos-hover);
  /*AUMENTAR O TAMANHO DA IMAGEM*/
  transform: scale(1.1);
  width: 100%;
}

.container-mensagem_professor{
  height: 30rem;
  width: 40rem;
  border-radius: 1.5rem;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-left: 2rem;
}

.titulomensagem-prof h2{
  font-size: 3rem;
```

```
  text-align: center;
}

.mensagem p{
  color: var(--cor-msg-professor);
  font-size: 1.6rem;
  font-family:Georgia, 'Times New Roman', Times, serif;
  font-weight: 400;
  font-style: normal;
  text-align: center;
  margin-bottom: 1rem;
  padding: 2rem;
}

.mensagem{
  width:  40vw;
  height: 30vh;
  margin-top: 3rem;
  margin-bottom: 20rem;
  text-align: center;
  border-radius: 2rem;
  background-color:var(--cor-card-mensagem);
  color: var(--cor-msg-professor);
}

.btn-excluir {
  background-color: var(--cor-btn-excluir);
  color: var(--primary-color-light);
  border: none;
  cursor: pointer;
  transition: 0.3s;
  padding: 0.8rem;
  border-radius: 1rem;
}

.btn-excluir:hover {
  background-color:var(--cor-btn-excluir-hover);
}

/* ---------------ESTILIZAÇÃO DA BLOB----------------- */


.container {
  align-items: center;
  display: flex;
  height: 20rem;
  justify-content: center;
}

.shape {
  background: linear-gradient(45deg, var(--third) 0%, var(--secondary) 100%);
  animation: morph 8s ease-in-out infinite;
```

```
  border-radius: 60% 40% 30% 70% / 60% 30% 70% 40%;
  height: 200px;
  transition: all 1s ease-in-out;
  width: 200px;
  z-index: 5;
  margin-left: 0.5rem;
  z-index: 0;
  left: 10rem;
}

.fa-message {
  color: green;
  height: 3rem;
  width: 3rem;
}

@keyframes morph {
  0% {
    border-radius: 60% 40% 30% 70% / 60% 30% 70% 40%;
    background:  linear-gradient(45deg,  var(--primary)  0%,  var(--secondary)
100%);
  }

  50% {
    border-radius: 30% 60% 70% 40% / 50% 60% 30% 60%;
    background: linear-gradient(45deg, var(--third) 0%, var(--secondary) 100%);

  }

  100% {
    border-radius: 60% 40% 30% 70% / 60% 30% 70% 40%;
    background:  linear-gradient(45deg,  var(--primary)  0%,  var(--secondary)
100%);
  }
}
```

Página inicial - Media

```
/* Media de telas para smartphones */
@media (max-width: 576px){

    .titulo-destaque .part1{
        font-size: 3.5rem;
        margin-left: 1.3rem;
    }

    .titulo-destaque .part2{
        font-size: 3.5rem;
    }

    .shape {
        height: 2.5rem;
```

```
        width: 2.5rem;
        z-index: 0;
        margin-left: 0.2rem;
    }

    .titulo-destaque .partG {
        font-size: 3.5rem;
        opacity: 0.7;
        margin-left: 0.2rem;
        z-index: 0;
    }

    .container-grid{
        width: 8rem;
        height: 10rem;
    }

    .container-cards{
        height: 100vh;
        width: 100vw;
        gap: 1;
    }

    .c-galeria{
        grid-template-columns: repeat(2, 2fr);
        width: 100%;
        grid-auto-rows: 180px;
        grid-column-gap: 2rem;
        grid-row-gap: 4.5rem;
        margin-bottom: 6rem;
        margin-right: 12rem;
        margin-left: 2rem;
    }

    .servicos{
        /*LARGURA DO CARD.*/
        width: 75%;
        /*COR DE FUNDO DO CARD*/
        height: 100%;
    }

    .servicos img{
        width: 100%;
        /*ALTURA AUTOMATICA DA IMAGEM*/
        height: 11rem;
    }

    .container-mensagem_professor{
        height: 25rem;
        width: 20rem;
        justify-content: flex-start;
        margin-left: 2.5rem;
```

```
    }

    .titulomensagem-prof h2{
        margin-top: 13rem;
        font-size: 1.8rem;
        text-align: center;
    }

    .mensagem p{
        font-size: 1rem;
        text-align: center;
    }

    .mensagem{
        width: 78vw;
        height: 36vh;
        margin-top: 4rem;
        margin-bottom: 8rem;
        border-radius: 2rem;
    }

    .btn-excluir {
        padding: 0.5rem;
        border-radius: 1rem;
        width: 10rem;
        font-size: 0.7rem;
    }

}

/* Media de telas para Tablets */
@media (min-width: 768px) {

    .titulo-destaque .part1{
        font-size: 9rem;
        margin-left: 2rem;
    }

    .titulo-destaque .part2{
        font-size: 9rem;
    }

    .shape {
        height: 6.5rem;
        width: 6.5rem;
        margin-left: 0.2rem;
    }

    .titulo-destaque .partG {
        font-size: 9rem;
        opacity: 0.7;
```

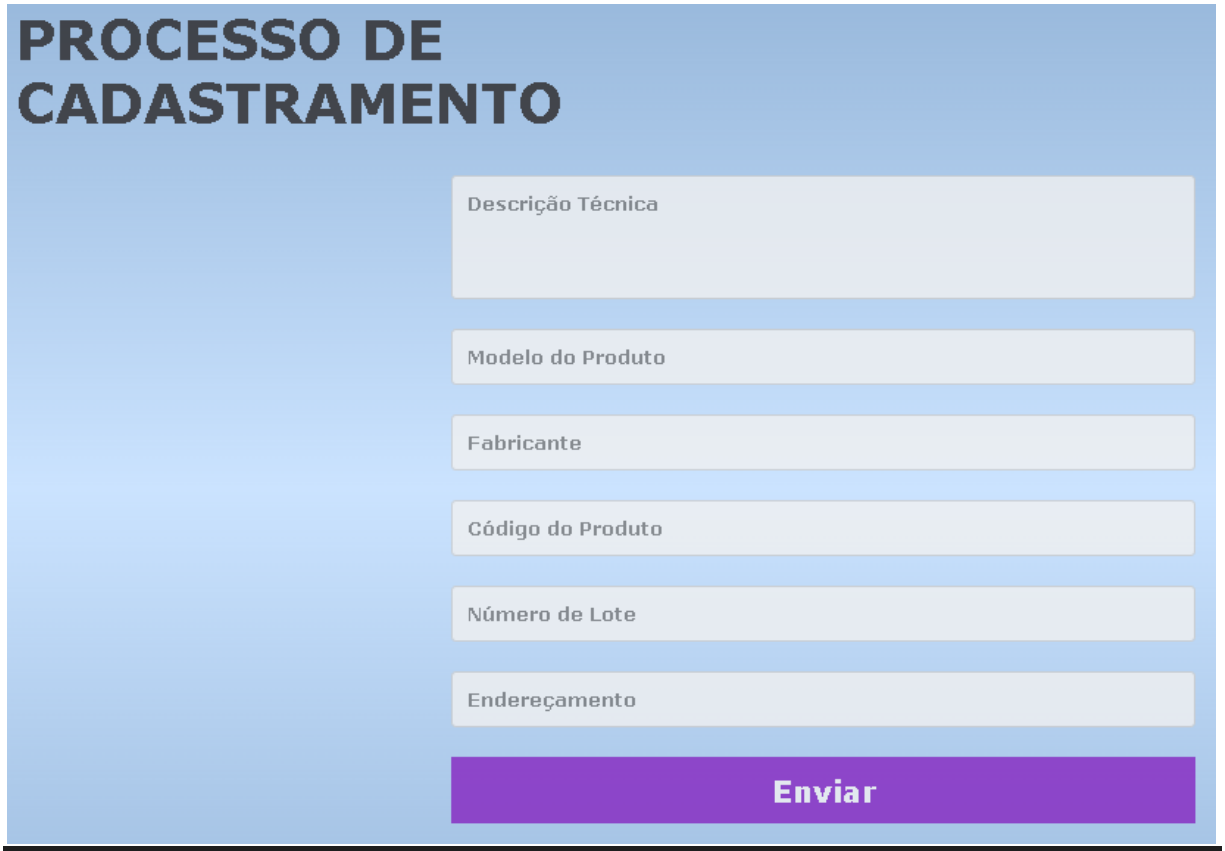
```
        margin-left: 0.2rem;
        z-index: 0;
    }

    .container-cadastramento{
        gap: 20rem;
    }

    .cadastramento-titulo span{
        font-size: 2.5rem;
        margin-left: 10rem;

    }

    .c-galeria{

        margin-top: 4rem;
        margin-right: 10rem;
        grid-row-gap: 7rem;
        grid-column-gap: 7rem;
        margin-right: 14rem;
        margin-left: 2rem;
    }

    .servicos{
        /*LARGURA DO CARD.*/
        width: 100%;
        /*COR DE FUNDO DO CARD*/
        height: 116.5%;
    }

    .servicos img{
        height: 14rem;
    }

    .container-img {
        width: 10rem;
        height: 20rem;
    }


    .titulomensagem-prof h2{
        margin-top: 1rem;
        font-size: rem;
        text-align: center;
    }

    .container-mensagem_professor{
        height: 25vh;
        width: 60vw;
        margin-left: 10rem;
```

```
    }

    .mensagem p{
        font-size: 1.7rem;
    }

    .mensagem{
        width: 60vw;
        height: 25vh;
        margin-top: 25rem;
        margin-bottom: 20rem;
        border-radius: 2rem;
    }

    .btn-excluir {
        padding: 0.5rem;
        border-radius: 1rem;
        width: 10rem;
        font-size: 0.7rem;
    }



}

@media (min-width: 1200px){

    .titulo-destaque .part1{
        font-size: 14rem;
        margin-left: 7rem;
    }

    .titulo-destaque .part2{
        font-size: 14rem;
    }

    .shape {
        height: 10rem;
        width: 10rem;
        margin-left: 0.2rem;
    }

    .titulo-destaque .partG {
        font-size: 14rem;
        opacity: 0.7;
        margin-left: 0.2rem;
        z-index: 0;
    }

    .container-cadastramento{
        gap: 20rem;
```

```
    }

    .cadastramento-titulo span{
        font-size: 2.5rem;
        margin-left: 10rem;

    }

    .servicos{
        /*LARGURA DO CARD.*/
        width: 100%;
        /*COR DE FUNDO DO CARD*/
        height: 100%;
    }

    .titulomensagem-prof h2{
        margin-top: 1rem;
        font-size: rem;
        text-align: center;
    }

    .container-mensagem_professor{
        margin-left: 16.5rem;
        margin-top: 5rem;
    }

    .mensagem p{
        font-size: 2rem;
    }

    .mensagem{
        width: 50vw;
        height: 30vh;
        margin-top: 5rem;
        margin-bottom: 5rem;
        border-radius: 2rem;
    }

    .btn-excluir {
        padding: 1rem;
        border-radius: 1rem;
        width: 14rem;
        font-size: 1rem;
    }

}

@media (min-width: 1440px){

    .titulo-destaque .part1{
        font-size: 19rem;
```

```
        margin-left: 1rem;
    }

    .titulo-destaque .part2{
        font-size: 19rem;
    }

    .shape {
        height: 13rem;
        width: 13rem;
        margin-left: 0.2rem;
    }

    .titulo-destaque .partG {
        font-size: 19rem;
        opacity: 0.7;
        margin-left: 0.2rem;
        z-index: 0;
    }

    .servicos{
        /*LARGURA DO CARD.*/
        width: 80%;
        height: 110%;
    }

    .container-mensagem_professor{
        height: 30rem;
        width: 40rem;
        justify-content: center;
        align-items: center;
        margin-left: 2rem;
    }

    .mensagem p{
        font-size: 1.6rem;
        padding: 2rem;
    }

    .mensagem{
        width:  40rem;
        height: 20rem;
        margin-top: 2rem;
        margin-bottom: 15rem;
        margin-left: 52rem;
    }
}
```

Página processo de cadastramento




```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/cadastramento.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}


    <!-- LINK PARA O JAVASCRIPT(FUNCIONALIDADE)    -->
    <script src="/static/js/cadastramento.js" defer></script>


    <div id="successMessage" style="display: none;" class="alert">
        <span class="alert-icon">✔</span>
        <span class="alert-text">Processo concluído com sucesso!</span>
    </div>

    <section class="geral-cadastramento">

            <div class="cadastramento-titulo">
                <span class="cadastramento1">PROCESSO DE</span><br>
                <span class="cadastramento2">CADASTRAMENTO</span>
            </div>

            <div class="formulario">
                <div class="form_container">
```

```
                    <form action="/cadastramento" method="POST">
                        <!-- Campo "Descrição Técnica" -->
                        <textarea     placeholder="Descrição     Técnica"
class="textarea-field" name="descricao"></textarea>

                        <input       list="browsers"       name="modelo"
placeholder="Modelo do Produto" class="input-field">

                        <!-- Campo "Fabricante" -->
                        <input    type="text"    placeholder="Fabricante"
class="input-field" name="fabricante">

                        <!-- Campo "Código do produto" -->
                        <input   type="number"   placeholder="Código   do
Produto" class="input-field" name="codigo">

                        <!-- Campo "Responsável pelo Recebimento" -->
                        <input type="number" placeholder="Número de Lote"
class="input-field" name="numeroLote">

                        <!-- Campo "Responsável pelo Recebimento" -->
                        <input  type="text"  placeholder="Endereçamento"
class="input-field" name="enderecamento">

                        <!-- Botão de Enviar -->
                        <button                          class="submit-btn"
id="confirmButton">Enviar</button>
                    </form>

                </div>
            </div>

        </section>

</section>

<script >
    {% with messages = get_flashed_messages() %}
        {% for mensagem in messages %}
            {{ mensagem|safe }}
        {% endfor %}
    {% endwith %}
</script>




{% endblock %}
```

```
document.querySelectorAll('.select-menu').forEach(menu => {
    const selectBtn = menu.querySelector('.select-btn');
    const options = menu.querySelectorAll('.option');
    const sBtnText = menu.querySelector('.sBtn-text');

    selectBtn.addEventListener('click', () => {
        menu.classList.toggle('active');
    });

    options.forEach(option => {
        option.addEventListener('click', () => {
            let    selectedOption    =    option.querySelector('.option-
text').innerText;
            sBtnText.innerText = selectedOption;
            menu.classList.remove('active');
        });
    });
});

// Alert de confirmação

document.getElementById('confirmButton').addEventListener('click',
function() {
    const steps = document.querySelectorAll('.step');

    steps.forEach(step => {
        step.classList.add('active');
    });

    const successMessage = document.getElementById('successMessage');
    successMessage.style.display = 'flex';

    sessionStorage.setItem('showSuccessMessage', 'true');

    setTimeout(() => {
        successMessage.style.display = 'none';
        sessionStorage.removeItem('showSuccessMessage');
    }, 3000);
});


window.addEventListener('load', () => {
    const successMessage = document.getElementById('successMessage');

    if (sessionStorage.getItem('showSuccessMessage') === 'true') {
        successMessage.style.display = 'flex';
        setTimeout(() => {
            successMessage.style.display = 'none';
            sessionStorage.removeItem('showSuccessMessage');
        }, 3000);
    }
});
```

```
window.addEventListener('beforeunload', () => {
    const successMessage = document.getElementById('successMessage');
    if (successMessage.style.display === 'flex') {
        sessionStorage.setItem('showSuccessMessage', 'true');
    }
})
```

```
document.getElementById('status-filter').addEventListener('change',
function() {
    const filterValue = this.value;
    const users = document.querySelectorAll('.user-item');

    users.forEach(user => {
        if (filterValue === 'all') {
            user.style.display = 'flex';
        } else {
            if (user.getAttribute('data-status') === filterValue) {
                user.style.display = 'flex';
            } else {
```

```
  user.style.display = 'none';
            }
        }
    });
});
```

```
# importando a classe necessária para poder realizar a conexão com o banco
de dados
from conexao import Conexao
#  classe  que  irá  armazenar  as  funções  referentes  ao  processo  de
cadastramento
class Cadastramento:
    def __init__(self):
        # inicializa os atributos da classe com o valor None
        self.cod_prod = None
        self.descricao_tecnica = None
        self.modelo = None
        self.fabricante = None
        self.num_lote = None
        self.enderecamento = None

    # função para cadastrar os produtos
    def  cadastramento  (self,  cod_prod,  descricao_tecnica,  modelo,
fabricante, num_lote, enderecamento, turma):
        # conectando com o banco de dados
        mydb = Conexao.conectarAluno(turma)

        # criando um cursor para executar comandos SQL na conexão com o
banco de dados
        mycursor = mydb.cursor()
```

```
        # armazenando as variáveis que contém os valores obtidos no form
(na página "cadastramento.html")
        valores = (cod_prod, descricao_tecnica, modelo, fabricante,
num_lote, enderecamento)
        # comando sql para inserir os dados no banco de dados
        dados = f"INSERT INTO tb_cadastramento (cod_prod,
descricao_tecnica, modelo, fabricante, num_lote, enderecamento) VALUES
(%s, %s, %s, %s, %s, %s)"

        #executando a variável a cima
        mycursor.execute(dados, valores)



        # realiza o commit da transação, garantindo que as alterações
feitas na base de dados sejam salvas
        mydb.commit()

        # fechando o banco de dados
        mydb.close()

        return True

    def cadastramentoProf (self, cod_prod, descricao_tecnica, modelo,
fabricante, num_lote, enderecamento):
        # conectando com o banco de dados
        mydb = Conexao.conectar()

        # criando um cursor para executar comandos SQL na conexão com o
banco de dados
        mycursor = mydb.cursor()

        dados = f"INSERT INTO databaseProfessor.tb_cadastramento
(cod_prod, descricao_tecnica, modelo, fabricante, num_lote,
enderecamento) VALUES ('{cod_prod}', '{descricao_tecnica}', '{modelo}',
'{fabricante}', '{num_lote}', '{enderecamento}')"

        #executando a variável a cima
        mycursor.execute(dados)



        # realiza o commit da transação, garantindo que as alterações
feitas na base de dados sejam salvas
        mydb.commit()

        mydb.close()

        return True
```

```
# roteamento da página de cadastramento
# RF005
@app.route("/cadastramento", methods=["GET", "POST"])
def pagina_cadastramento():
```

```
# as páginas são protegidas por autenticação de sessão para garantir que
apenas usuários autenticados possam acessá-las.
# if que determina o acesso às páginas apenas se o aluno estiver logado.
    if "usuario_logado" in session:
        if request.method == "GET":
            return render_template("cadastramento.html")
        if request.method == "POST":
            # pegando os valores dos inputs da página cadastramento
            descricao = request.form.get("descricao")
            modelo = request.form.get("modelo")
            fabricante = request.form.get("fabricante")
            codigo = request.form.get("codigo")
            numeroLote = request.form.get("numeroLote")
            enderecamento = request.form.get("enderecamento")

            # armazenando a classe da página de cadastramento em que estão
os comandos sql em uma várial
            tbCadastramento = Cadastramento()

            # chamando a função que está dentro da classe
            if tbCadastramento.cadastramento(codigo, descricao, modelo,
fabricante,                 numeroLote,                 enderecamento,
session['usuario_logado']['turma']):
                return render_template("cadastramento.html")
            else:
                return 'Erro ao realizar o processo de Cadastramento'
    # verificando se o usuário logado é o professor, para poder liberar
a vizualização das páginas
    elif "professor_logado" in session:
        if request.method == "GET":
            return render_template("cadastramento.html")
        if request.method == "POST":
            # pegando os valores dos inputs da página cadastramento
            descricao = request.form.get("descricao")
            modelo = request.form.get("modelo")
            fabricante = request.form.get("fabricante")
            codigo = request.form.get("codigo")
            numeroLote = request.form.get("numeroLote")
            enderecamento = request.form.get("enderecamento")
            # transformando a classe Cadastramento em um objeto
            tbCadastramento = Cadastramento()

            # pegando a função armazenada no objeto para realizar o
processo de cadastramento de um produto
            if   tbCadastramento.cadastramentoProf(codigo,   descricao,
modelo, fabricante, numeroLote, enderecamento):
                return render_template("cadastramento.html")
            else:
                return 'Erro ao realizar o processo de Cadastramento'
    else:
        return redirect("/login")
```

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;
500;600&display=swap');


body{
    position: relative;
    width: 100rem;
    height: 80rem;
    background-image: var( --cor-icone-cadastramento);
    border-radius: 2.5rem;
    margin-bottom: 1.5rem;
    opacity: 0.8;
    overflow-x: hidden;
}

.geral-cadastramento{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}


.cadastramento1{
    position: absolute;
    left: 30.8rem;
    font-size: 3rem;
    color: #292525;
    margin-top: 10rem;
    font-size: 2.75rem;
}

.cadastramento2{
    position: absolute;
    left: 30.8rem;
    font-size: 2.75rem;
    color: #292525;
    margin-top: 12rem;
}

.formulario{
    margin-top: 14rem;
}

.form_container {
    margin-top: 2rem;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 41rem;
    padding: 1.5rem;
```

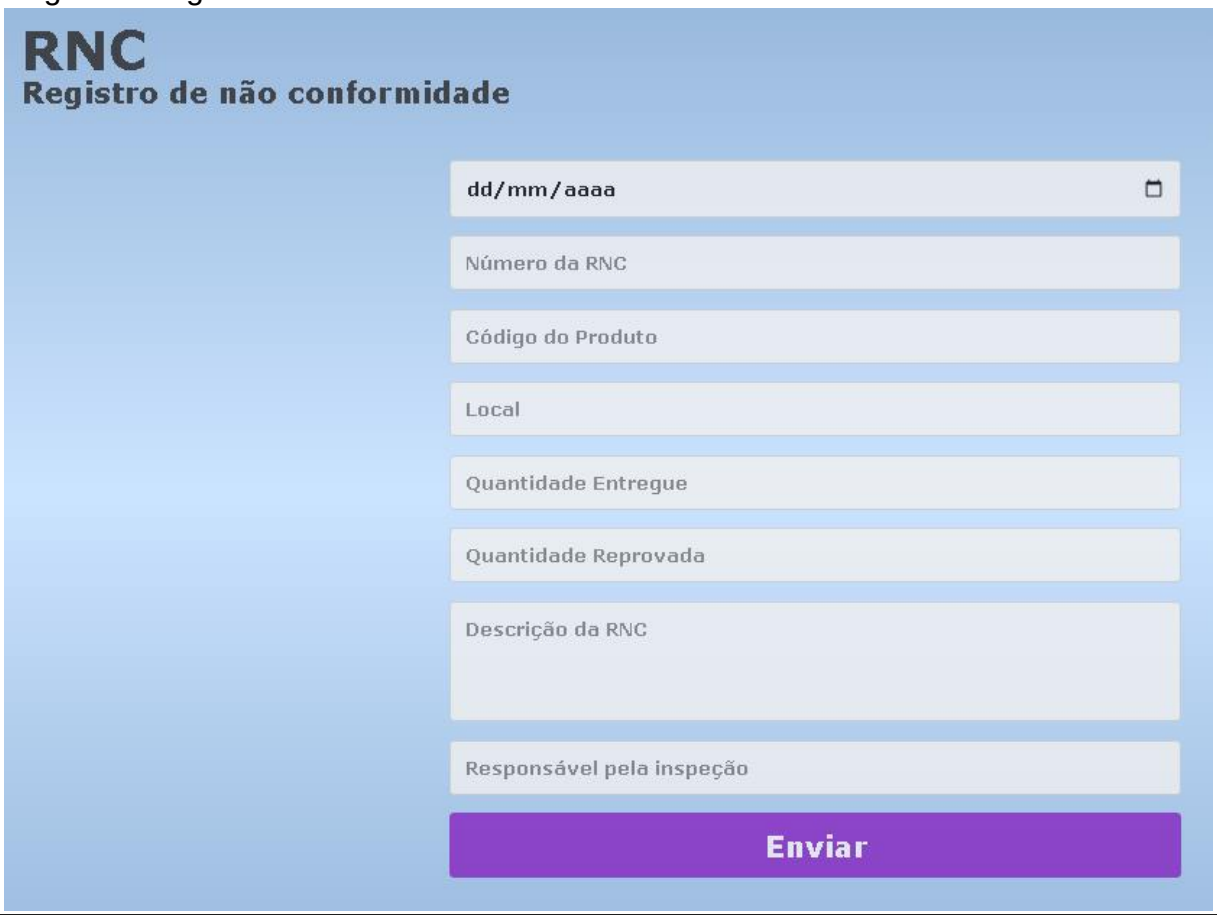
```
    border-radius: 8px;
    margin-bottom: 2.5rem;

}

.input-field, .textarea-field, .select-menu{
    width: 100%;
    margin-bottom: 1.5rem;
    padding: 12px;
    font-size: 16px;
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    background-color: var(--primary-color-light);
}

.textarea-field {
    height: 100px;
    resize: none;
}

.select-menu {
    position: relative;
}

.select-btn {
    display: flex;
    justify-content: space-between;
    align-items: center;
    cursor: pointer;
}

.select-btn i {
    font-size: 20px;
    transition: 0.3s;
}

.select-menu.active .select-btn i {
    transform: rotate(-180deg);
}

.options {
    position: absolute;
    top: 100%;
    left: 0;
    width: 100%;
    padding: 0;
    list-style: none;
    background-color: var(--primary-color-light);
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    max-height: 150px;
    overflow-y: auto;
    display: none;
```

```
}

.select-menu.active .options {
    display: block;
    z-index: 10;
}

.option {
    padding: 10px;
    cursor: pointer;
}

.option:hover {
    background-color: var(--cor-hover-options);
}

.submit-btn {
    background-color: var(--cor-btn-enviar);
    color: var(--primary-color-light);
    border: none;
    cursor: pointer;
    transition: background-color 0.3s;
    width: 100%;
    padding: 12px;
    font-size: 1.5rem;
}

.submit-btn:hover {
    background-color:var(--cor-hover-btn-enviar);
}



/* Estilização da verificação do processo */
.tracking-container{
    border: 1px solid #e0e0e0;
    border-radius: 8px;
    padding: 20px;
    max-width: 500px;
    text-align: center;
    font-family: Arial, sans-serif;
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-left: 4.5rem;
}

.status{
    font-weight: bold;
    color: #2f855a;
    margin-bottom: 20px;
    text-align: center;
}
```

```
.tracking-steps {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 2rem;
}

.step{
    text-align: center;
    color: #c0c0c0;
}

.step.active{
    color: #2f855a;
}

.icon-box{
    font-size: 30px;
    margin-bottom: 10px;
}

.step p{
    margin-top: 0;
    font-size: 14px;
}

.step.active .icon-box{
    color: #2f855a;
}

.step:not(.active) .icon-box{
    color: #c0c0c0;
}

.step.active + .step::before{
    background-color: #2f855a;
}

.alert{
    display: none;
    position: fixed;
    top: 5px;
    left: 50%;
    transform: translateX(-50%);
    align-items: center;
    justify-content: center;
    background-color: #4CAF50;
    color: white;
    padding: 15px;
    border-radius: 5px;
    position: relative;
    z-index: 1000;
```

```
    text-align: center;
    width: 90%;
    max-width: 500px;
}

.alert-icon{
    margin-right: 10px;
    font-size: 20px;
}

.alert-text{
    font-size: 16px;
}
```

Página de registro de não conformidade




```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/rnc.css">
<link rel="stylesheet" href="/static/styles/media/rnc-media.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}


    <!-- LINK PARA O JAVASCRIPT(FUNCIONALIDADE)    -->
    <script src="/static/js/rnc.js" defer></script>
```

```
    <section class="geral-cadastramento">

            <div class="cadastramento-titulo">
                <span class="cadastramento1"> RNC </span>
                <span     class="cadastramento2">Registro     de     não
conformidade</span>
            </div>


        <div class="formulario">

            <div class="form_container">

                <form action="/rnc" method="POST">

                    <!-- Campo "Recebido em" -->
                    <input    type="date"    placeholder="Recebido    em:"
class="input-field" name="date">

                    <!-- Campo "Número da RNC" -->
                    <input   type="text"   placeholder="Número   da   RNC"
class="input-field" name="numRNC">

                    <!-- Campo "Número do Código do Produto" -->
                    <input  type="text"  placeholder="Código  do  Produto"
class="input-field" id="codigoProduto" name="codProd">

                    <!-- Campo "Local" -->
                    <input type="text" placeholder="Local" class="input-
field" id="enderecamentoProduto" name="local">

                    <!-- Campo "Quantidade entregue" -->
                    <input type="text" placeholder= "Quantidade Entregue"
class="input-field"
                    name="qtdentregue">

                    <!-- Campo "Quantidade Reprovada" -->
                    <input      type="text"      placeholder="Quantidade
Reprovada" class="input-field" name="qtdrepro">

                    <!-- Campo "Descrição da RNC" -->
                    <textarea     placeholder="Descrição     da     RNC"
class="textarea-field" id="descricaoProduto" name="descRNC"></textarea>

                    <!-- Campo "Responsável pela inspeção" -->
                    <input   type="text"   placeholder="Responsável   pela
inspeção" class="input-field" name="respInsp">

                    <!-- Botão de Enviar -->
```

```
                    <button class="submit-btn">Enviar</button>

                </form>
            </div>

        </div>

    </section>

    <script>
        document.addEventListener("DOMContentLoaded", function() {
            const produtos = {{ lista_produtos | tojson }};

            const                       codigoInput                       =
document.getElementById("codigoProduto");

            codigoInput.addEventListener("input", function() {
                const codigoDigitado = codigoInput.value.trim();

                const   produto   =   produtos.find(p   =>   p.codigo   ===
codigoDigitado);

                if (produto) {
                    document.getElementById("descricaoProduto").value  =
produto.descricao;

document.getElementById("enderecamentoProduto").value                    =
produto.enderecamento;
                }
            });
        });
    </script>
{% endblock %}
```

```
# importando a classe necessária para poder realizar a conexão com o banco
de dados
from conexao import Conexao
# classe que armazena as funções referentes ao processo de registro de
rnc
class Rnc:
    def __init__(self):
        self.desc_rnc = None
        self.recebimento = None
        self.num_rnc = None
        self.local_rnc = None
        self.quant_entregue = None
        self.quant_reprovada = None
        self.resp_inspecao = None
        self.cod_prod = None

    # criando uma função para que seja possível a realização do processo
de rnc
```

```
    def rnc(self, desc_rnc, recebimento, num_rnc, local_rnc, quant_entregue, quant_reprovada, resp_inspecao, cod_prod, cod_aluno, turma):
        # conectando com o banco de dados
        mydb = Conexao.conectarAluno(turma)

        mycursor = mydb.cursor()

        valores = (desc_rnc, recebimento, num_rnc, local_rnc, quant_entregue, quant_reprovada, resp_inspecao, cod_prod, cod_aluno)

        dados = f"INSERT INTO tb_rnc(desc_rnc, recebimento, num_rnc, local_rnc, quant_entregue, quant_reprovada, resp_inspecao, cod_prod, cod_aluno) VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s)"

        #executar
        mycursor.execute(dados, valores)

        # realiza o commit da transação, garantindo que as alterações feitas na base de dados sejam salvas
        mydb.commit()

        mydb.close()

        # retorna True indicando que a operação foi realizada com sucesso
        return True
```

```
# roteamento da página dos processos de registro rnc
# RF006
@app.route("/rnc", methods=["GET", "POST"])
def pagina_rnc():
    # verificando se há alguém logado no sistema
    if "usuario_logado" in session or "professor_logado" in session:
        if request.method == "GET":
            #conectando com o banco de dados
            mydb = Conexao.conectar()

            mycursor = mydb.cursor()

            # armazenando os dados que estão na session em uma variável
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # comando sql para pegar todos os produtos que foram cadastrados no banco de dados
            produtos = (f"SELECT * FROM {turma}.tb_cadastramento")

            mycursor.execute(produtos)
```

```
            resultado = mycursor.fetchall()

            # criando uma lista para armazenar os produtos posteriormente
            lista_produtos = []

            # loop para ir adicioando cada produto na lista
            for produto in resultado:
                lista_produtos.append({
                    "codigo":produto[0],
                    "descricao":produto[1],
                    "modelo":produto[2],
                    "fabricante":produto[3],
                    "numero_lote":produto[4],
                    "enderecamento":produto[5],
                    "quantidade":produto[6]
                })

            return    render_template("rnc.html",    lista_produtos    =
lista_produtos)

        if request.method == "POST":
            # pegando os dados que foram enviados pelo formulário
            data = request.form.get("date")
            numRNC = request.form.get("numRNC")
            local = request.form.get("local")
            qtdentregue = request.form.get("qtdentregue")
            qtdrepro = request.form.get("qtdrepro")
            descRNC = request.form.get("descRNC")
            respInsp = request.form.get("respInsp")
            codProd = request.form.get("codProd")

            # criando um objeto para armazenar a classe
            tbrnc = Rnc()

            # armazendo os dados que estão guardados na session, em uma
variável
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # realizando o processo de RNC por meio de uma função que
está armazenada no objeto criado anteriormente
            if  tbrnc.rnc(descRNC,  data,  numRNC,  local,  qtdentregue,
qtdrepro, respInsp, codProd, cod_aluno, turma):
                # emitindo uma mensagem de confirmação da realização do
processo na interface do sistema
                flash("alert('Parabéns, você acaou de realizar o processo
de Registro de Não Conformidade!!🎉')")
                return redirect("/")
```

```
            else:
                return 'Erro ao realizar o processo de RNC'

    else:
        return redirect("/login")
```

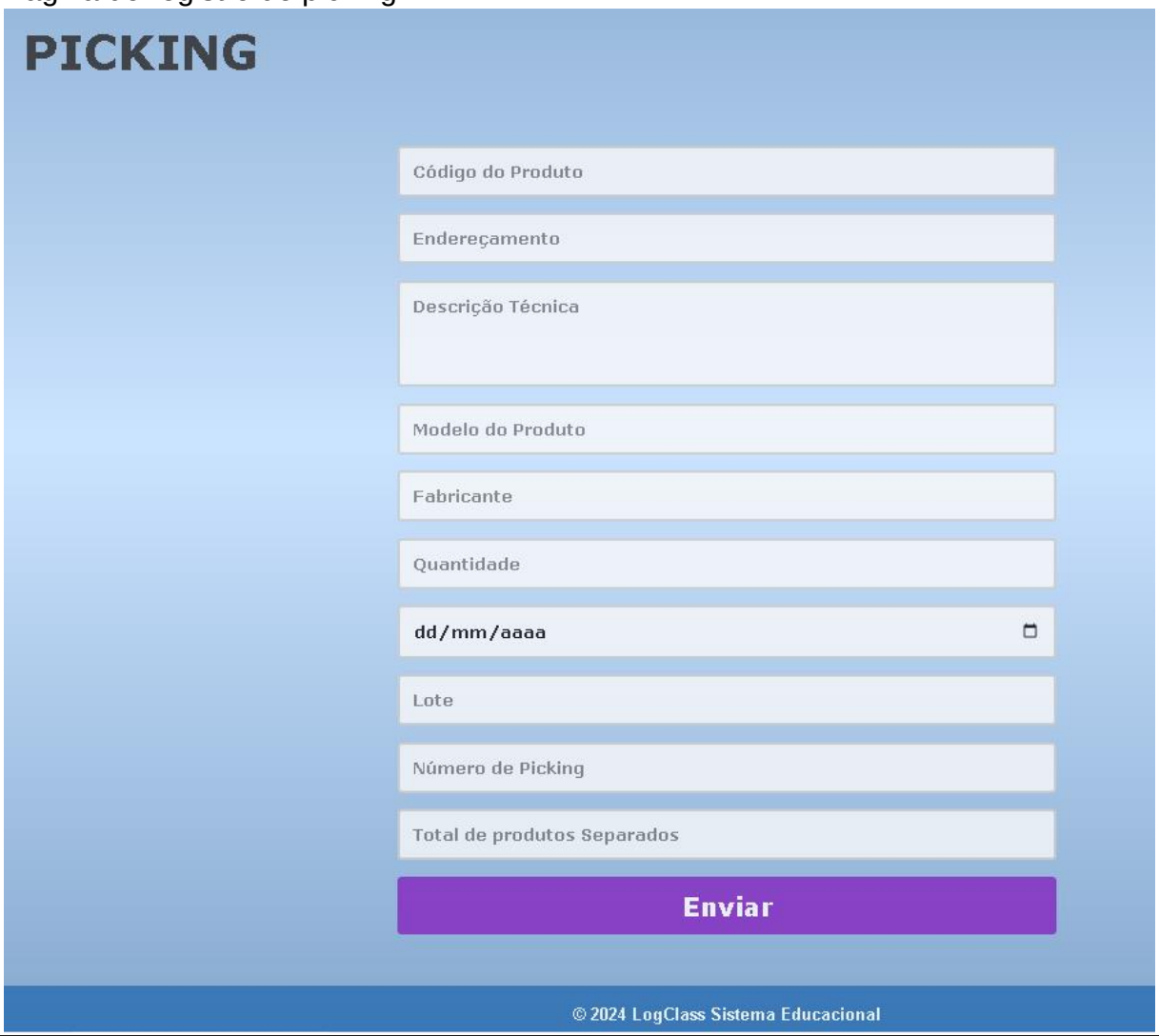
```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;
500;600&display=swap');


body{
    position: relative;
    width: 100rem;
    height: 80rem;
    background-image: var( --cor-icone-cadastramento);
    border-radius: 2.5rem;
    margin-bottom: 1.5rem;
    opacity: 0.8;
    overflow-x: hidden;
}


.geral-cadastramento {
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}

.cadastramento1{
    position: absolute;
    left: 30.8rem;
    font-size: 3rem;
    color: #292525;
    margin-top: 10rem;
    font-size: 2.75rem;
}

.cadastramento2{
    position: absolute;
    left: 30.8rem;
    font-size: 1.5rem;
    color: #292525;
    margin-top: 13rem;
}

.formulario{
    margin-top: 16rem;
}

.form_container {
```

```
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 41rem;
    padding: 1.5rem;
    border-radius: 8px;
    margin-bottom: 2rem;

}

.input-field, .textarea-field, .select-menu, .submit-btn {
    width: 100%;
    margin-bottom: 15px;
    padding: 12px;
    font-size: 16px;
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    background-color: var(--primary-color-light);

}

.textarea-field {
    height: 100px;
    resize: none;
}

.select-menu {
    position: relative;
}

.select-btn {
    display: flex;
    justify-content: space-between;
    align-items: center;
    cursor: pointer;
}

.select-btn i {
    font-size: 20px;
    transition: 0.3s;
}

.select-menu.active .select-btn i {
    transform: rotate(-180deg);
}

.options {
    position: absolute;
    top: 100%;
    left: 0;
    width: 100%;
    padding: 0;
```

```
    list-style: none;
    background-color: var(--primary-color-light);
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    max-height: 150px;
    overflow-y: auto;
    display: none;
}

.select-menu.active .options {
    display: block;
    z-index: 10;
}

.option {
    padding: 10px;
    cursor: pointer;

}

.option:hover {
    background-color: var(--cor-hover-options);
}

.submit-btn {
    background-color: var(--cor-btn-enviar);
    color: var(--primary-color-light);
    border: none;
    cursor: pointer;
    transition: 0.3s;
    font-size: 1.5rem;
}

.submit-btn:hover {
    background-color:var(--cor-hover-btn-enviar);
}

.container-tabela{
    margin-top: 5rem;
    width: 50%;

}

.container-tabela h2{
    font-size: rem;
}

table {
    width: 100%;
    border-collapse: collapse;
}

table, th, td {
```

```
    border: 1px solid;
}

th, td {
    padding: 1rem;
    text-align: left;
}

th {
    background-color: #f2f2f2;
}
```

Página do registro de picking




```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/picking.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
{% endblock %}

{% extends "layout.html" %}
```

```
{% block conteudo %}

<!-- LINK PARA O JAVASCRIPT(FUNCIONALIDADE) -->
<script src="/static/js/picking.js" defer></script>

<section class="geral-cadastramento">

        <div class="cadastramento-titulo">
            <span class="cadastramento1"> PICKING </span>
        </div>
```

```
<div class="formulario">
        <div class="form_container">
            <form action="/picking" method="POST">
                <!-- Campo "Código do Produto" -->
                <input  type="number"  placeholder="Código  do  Produto"
class="input-field" id="codigoProduto" name="codProd">

                <!-- Campo "Endereçamento" -->
                <input      type="text"      placeholder="Endereçamento"
class="input-field" id="enderecamentoProduto" name="enderecamento">

                <!-- Campo "Descrição Técnica" -->
                <textarea        placeholder="Descrição        Técnica"
class="textarea-field" id="descricaoProduto" name="descTec"></textarea>

                <input         list="browsers"         name="modeloPick"
placeholder="Modelo do Produto" id="modeloProduto" class="input-field">

                <!-- Campo "Fabricante" -->
                <input       type="text"       placeholder="Fabricante"
class="input-field" id="fabricanteProduto" name="fabri">

                <!-- Campo "Quantidade" -->
                <input       type="text"       placeholder="Quantidade"
class="input-field" id="quantidadeProduto" name="qtde">

                <!-- Campo "Data" -->
                <input  type="date"  placeholder="Data:"  class="input-
field" name="data">

                <!-- Campo "lote" -->
                <input  type="text"  placeholder="Lote"  class="input-
field" id="numeroLote" name="lote">

                <!-- Campo "número de Picking" -->
                <input  type="number"  placeholder="Número  de  Picking"
class="input-field" name="numPicking">

                <!-- Campo "Total de produtos Separados" -->
```

```
                <input type="number" placeholder="Total de produtos
Separados" class="input-field" name="totalProd">

                <!-- Botão de Enviar -->
                <button class="submit-btn">Enviar</button>
            </form>
        </div>

        <div class="botoes">
            <a href="/simulador" class="botao-simulador">Ir para
Simulador</a>
        </div>
    </div>
</section>


<script>
    document.addEventListener("DOMContentLoaded", function() {
        const produtos = {{ lista_produtos | tojson }};

        const codigoInput = document.getElementById("codigoProduto");

        codigoInput.addEventListener("input", function() {
            const codigoDigitado = codigoInput.value.trim();

            const produto = produtos.find(p => p.codigo ===
codigoDigitado);

            if (produto) {
                document.getElementById("descricaoProduto").value =
produto.descricao;
                document.getElementById("modeloProduto").value =
produto.modelo;
                document.getElementById("fabricanteProduto").value =
produto.fabricante;
                document.getElementById("numeroLote").value =
produto.numero_lote;
                document.getElementById("enderecamentoProduto").value =
produto.enderecamento;
                document.getElementById("quantidadeProduto").value =
produto.quantidade;
            }
        });
    });
</script>



{% endblock %}
```

```
document.querySelectorAll('.select-menu').forEach(menu => {
```

```
    const selectBtn = menu.querySelector('.select-btn');
    const options = menu.querySelectorAll('.option');
    const sBtnText = menu.querySelector('.sBtn-text');

    selectBtn.addEventListener('click', () => {
        menu.classList.toggle('active');
    });

    options.forEach(option => {
        option.addEventListener('click', () => {
            let    selectedOption    =    option.querySelector('.option-
text').innerText;
            sBtnText.innerText = selectedOption;
            menu.classList.remove('active');
        });
    });
});
```

```
# importando a classe necessária para poder realizar a conexão com o banco
de dados
from conexao import Conexao
# classe que irá armazenar as funções de registro de picking
class Picking:
    def __init__(self):
        self.num_picking = None
        self.endereco = None
        self.desc_tecnica = None
        self.modelo_pk = None
        self.fabricante_pk = None
        self.quant_pk = None
        self.data_pk = None
        self.lote_pk = None
        self.total_pk = None
        self.cod_prod = None

    # criando a função para que possa ser realizado o processo de picking
    def  picking(self,  num_picking,endereco,  desc_tecnica,  modelo_pk,
fabricante_pk, quant_pk, data_pk, lote_pk, total_pk, cod_prod, turma):
        # conectando com o banco de dados
        mydb = Conexao.conectarAluno(turma)

        mycursor = mydb.cursor()

        valores  =  (num_picking,  endereco,  desc_tecnica,  modelo_pk,
fabricante_pk, quant_pk, data_pk, lote_pk, total_pk, cod_prod)

        dados   =   f"INSERT   INTO   tb_picking(num_picking,   endereco,
desc_tecnica,  modelo_pk,  fabricante_pk,  quant_pk,  data_pk,  lote_pk,
total_pk, cod_prod) VALUES( %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

        #executar
        mycursor.execute(dados, valores)
```

```
        mydb.commit()

        mydb.close()

        return True
```

```
# roteamento da página dos processos de registro picking
# RF007
@app.route("/picking", methods=["GET", "POST"])
def pagina_picking():
    # verificando se há algum usuário cadastrado e logado para poder habilitar a visualização da página
    if "usuario_logado" in session or "professor_logado" in session:
        if request.method == "GET":
            #conectando com o banco de dados
            mydb = Conexao.conectar()

            mycursor = mydb.cursor()

            # verificando se os usuários estão logados para poder armazenar as informações que foram guardadas na sessão, em uma variável
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # comando sql para poder pegar todos os produtos que foram cadastrados no banco de dados
            produtos = (f"SELECT * FROM {turma}.tb_cadastramento")

            mycursor.execute(produtos)

            resultado = mycursor.fetchall()

            # criando uma lista para posterioirmente armazenar os produtos
            lista_produtos = []

            # loop para poder ir adicionando os produtos na lista
            for produto in resultado:
                lista_produtos.append({
                    "codigo":produto[0],
                    "descricao":produto[1],
                    "modelo":produto[2],
                    "fabricante":produto[3],
                    "numero_lote":produto[4],
                    "enderecamento":produto[5],
                    "quantidade":produto[6],
                })
```

```
            return                          render_template("picking.html",
lista_produtos=lista_produtos)

        if request.method == "POST":
            # pegando os dados que foram enviados pelo formulário
            numPicking = request.form.get("numPicking")
            enderecamento = request.form.get("enderecamento")
            descTec = request.form.get("descTec")
            modeloPick = request.form.get("modeloPick")
            fabri = request.form.get("fabri")
            qtde = request.form.get("qtde")
            data = request.form.get("data")
            lote = request.form.get("lote")
            totalProd = request.form.get("totalProd")
            codProd = request.form.get("codProd")

            # criando um objeto para aramzenar a classe Picking
            tbpicking = Picking()

            # armazenando as irformações que são guardadas na session, em
uma variável
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # realizando o registro de picking por meio da função que
pertence ao objeto criado
            if  tbpicking.picking(numPicking,  enderecamento,  descTec,
modeloPick, fabri, qtde, data, lote, totalProd, codProd, turma):
                # emitindo uma mensagem para quando o cadastro for
realizado com sucesso
                flash("alert('Parabéns, você acaou de realizar o processo
de picking!!🎉')")
                return redirect("/")
            else:
                # emitindo uma mensagem para quando o cadastro não for
realizado
                return 'Erro ao realizar o processo de Picking'

    else:
        return redirect("/login")
```

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600
&display=swap');

body{
    position: relative;
    width: 100rem;
```

```
    height: 80rem;
    background: var( --cor-icone-cadastramento);
    border-radius: 2.5rem;
    margin-bottom: 1.5rem;
    opacity: 0.8;
}

.geral-cadastramento{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}

.cadastramento1{
    position: absolute;
    left: 30.8rem;
    font-size: 2.75rem;
    color: #292525;
    margin-top: 10rem;
}


.formulario{
    margin-top: 16rem;
}

.form_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 41rem;
    padding: 20px;
    border-radius: 1.5rem;
    margin-bottom: 2rem;

}

.input-field, .textarea-field, .select-menu, .submit-btn {
    width: 100%;
    margin-bottom: 15px;
    padding: 12px;
    font-size: 16px;
    border: 0.2rem solid var(--cor-bordas);
    border-radius: 4px;
    background-color: var(--cor-fundo-formularios);
}

.textarea-field {
    height: 100px;
    resize: none;
```

```
}

.select-menu {
    position: relative;
}

.select-btn {
    display: flex;
    justify-content: space-between;
    align-items: center;
    cursor: pointer;
}

.select-btn i {
    font-size: 20px;
    transition: 0.3s;
}

.select-menu.active .select-btn i {
    transform: rotate(-180deg);
}

.options {
    position: absolute;
    top: 100%;
    left: 0;
    width: 100%;
    padding: 0;
    list-style: none;
    background-color: var(--cor-fundo-options);
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    max-height: 150px;
    overflow-y: auto;
    display: none;
}

.select-menu.active .options {
    display: block;
    z-index: 10;
}

.option {
    padding: 10px;
    cursor: pointer;
}

.option:hover {
    background-color: var(--cor-hover-options);
}

.submit-btn {
```

```
    background-color: var(--cor-btn-enviar);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition: background-color 0.3s;
    font-size: 1.5rem;
}

.submit-btn:hover {
    background-color: var(--cor-hover-btn-enviar);
}


/* Estilização da verificação do processo */
.tracking-container{
    border: 1px solid #e0e0e0;
    border-radius: 8px;
    padding: 20px;
    max-width: 100%;
    width: 600px;
    text-align: center;
    font-family: Arial, sans-serif;
    display: flex;
    flex-direction: column;
    align-items: center;
}


.status{
    font-weight: bold;
    color: #2f855a;
    margin-bottom: 20px;
    text-align: center;
}

.tracking-steps {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 2rem;
}

.step{
    text-align: center;
    color: #c0c0c0;
}

.step.active{
    color: #2f855a;
}

.icon-box{
```

```
    font-size: 30px;
    margin-bottom: 10px;
}

.step p{
    margin-top: 0;
    font-size: 14px;
}

.step.active .icon-box{
    color: #2f855a;
}

.step:not(.active) .icon-box{
    color: #c0c0c0;
}

.step.active + .step::before{
    background-color: #2f855a;
}

.alert{
    display: none;
    position: fixed;
    top: 5px;
    left: 50%;
    transform: translateX(-50%);
    align-items: center;
    justify-content: center;
    background-color: #4CAF50;
    color: white;
    padding: 15px;
    border-radius: 5px;
    position: relative;
    z-index: 1000;
    text-align: center;
    width: 90%;
    max-width: 500px;
}

.alert-icon{
    margin-right: 10px;
    font-size: 20px;
}

.alert-text{
    font-size: 16px;
}
```

Página de controle de estoque


```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/estoque.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">

{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}

    <!-- Section 01.
    Essa Section possui um espaço para imagem (cor solicitada pelo cliente) e
nome da matéria. -->
<section class="geral-cadastramento">

        <div class="cadastramento-titulo">
            <span class="controle1">CONTROLE DE</span><br>
            <span class="controle2">ESTOQUE</span>
        </div>



    <!-- Formulário para cadastro de estoque com todos os campos originais -->
    <div class="formulario">
        <div class="form_container">
            <form action="/estoque" method="POST">
```

```
                <span class="cadastramentoD"> Cadastrar dados: </span>

                <!-- Campo de código do produto para busca automática -->
                <input    type="number"    placeholder="Código    do    Produto"
class="input-field" name="cod_prod" id="codigoProduto">

                <!-- Campos já existentes para informações do produto -->
                <input type="number" placeholder="Número de Lote" class="input-
field" name="num_lt" id="numeroLote">
                <input   type="text"   placeholder="Localização"   class="input-
field" name="loc_" id="enderecamentoProduto">
                <textarea   placeholder="Descrição   Técnica"   class="textarea-
field" name="descricao" id="descricaoProduto"></textarea>
                <span class="cadastramento">Data de entrada:</span>
                <input type="date" placeholder="Data de Entrada:" class="input-
field" name="dt_enter">
                <input    type="number"    placeholder="Quantidade    de    itens"
class="input-field" name="qt_item">
                <span class="cadastramento">Data de saída:</span>
                <input type="date" placeholder="Data de Saída:" class="input-
field" name="dt_end">
                <input    type="number"    placeholder="Quantidade    da    Saída"
class="input-field" name="qt_saida">
                <input  type="number"  placeholder="Saldo"  class="input-field"
name="_saldo" id="saldoProduto">
                <input    type="text"    placeholder="Funcionário    Responsável"
class="input-field" name="funcionario">

                <!-- Botão de envio do formulário -->
                <button class="submit-btn">Enviar</button>
            </form>
        </div>
    </div>
</section>

<script>
    document.addEventListener("DOMContentLoaded", function() {
        const produtos = {{ lista_produtos | tojson }};

        const codigoInput = document.getElementById("codigoProduto");

        codigoInput.addEventListener("input", function() {
            const codigoDigitado = codigoInput.value.trim();

            const produto = produtos.find(p => p.codigo === codigoDigitado);

            if (produto) {
                document.getElementById("descricaoProduto").value            =
produto.descricao;
                document.getElementById("numeroLote").value                  =
produto.numero_lote;
```

```
                document.getElementById("enderecamentoProduto").value         =
produto.enderecamento;
                document.getElementById("saldoProduto").value                 =
produto.quantidade;
            }
        });
    });
</script>


{% endblock %}
```

```
# importando a classe necessária para poder realizar a conexão com o banco de
dados
from conexao import Conexao
# classe que irá armazenar as funções referentes ao processo de registro de
estoque
class Estoque:
    def __init__(self):
        # inicializa os atributos da classe com o valor None
        self.cod_prod_est = None
        self.num_lote_est = None
        self.loc_est = None
        self.desc_tec = None
        self.data_entrega = None
        self.quant_itens_entrada = None
        self.data_saida = None
        self.quant_saida = None
        self.saldo = None
        self.func_responsavel = None
        self.cod_aluno = None

    # função para o cadastro de estoque
    def  estoque  (self,  cod_prod_est,  num_lote_est,  loc_est,  desc_tec,
data_entrega,    quant_itens_entrada,    data_saida,    qt_saida,    saldo,
fuc_responsavel, cod_aluno, turma):
        # conectando com o banco de dados
        mydb = Conexao.conectarAluno(turma)

        # criando um cursor para executar comandos SQL na conexão com o banco
de dados
        mycursor = mydb.cursor()

        if data_entrega == '':
            data_entrega = None

        if quant_itens_entrada == '':
            quant_itens_entrada = None

        if data_saida == '':
            data_saida = None
```

```
        if qt_saida == '':
            qt_saida = None

        dados = f"INSERT INTO tb_estoque(cod_prod_est, num_lote_est, loc_est,
desc_tec, data_entrega, quant_itens_entrada, data_saida, quant_saida, saldo,
func_responsavel,  cod_aluno)  VALUES  ('{cod_prod_est}',  '{num_lote_est}',
'{loc_est}', '{desc_tec}', %s, %s, %s, %s, '{saldo}', '{fuc_responsavel}',
'{cod_aluno}')"

        valores = (data_entrega, quant_itens_entrada, data_saida, qt_saida)
        #executando o comando que foi determinado a cima
        mycursor.execute(dados, valores)

        # realiza o commit da transação, garantindo que as alterações feitas na
base de dados sejam salvas
        mydb.commit()

        mydb.close()

        # retorna True indicando que a operação foi realizada com sucesso
        return True
```

```
# Página de controle de estoque
@app.route("/estoque", methods=["GET", "POST"])
def pagina_estoque():
    # verificando se há algum usuário logado no sistema, seja ele professor ou
aluno
    if "usuario_logado" in session or "professor_logado" in session:
        if request.method == "GET":
            mydb = Conexao.conectar()

            mycursor = mydb.cursor()

            turma = session['usuario_logado']['turma'] if "usuario_logado" in
session else session['professor_logado']['turma']

            produtos = f"SELECT * FROM {turma}.tb_cadastramento"

            mycursor.execute(produtos)

            resultado = mycursor.fetchall()

            lista_produtos = [{"codigo": produto[0], "descricao": produto[1],
"modelo": produto[2], "fabricante": produto[3], "numero_lote": produto[4],
"enderecamento": produto[5], "quantidade": produto[6]} for produto in resultado]

            return                              render_template("estoque.html",
lista_produtos=lista_produtos)
```

```
        if request.method == "POST":
            cod_prod = request.form.get("cod_prod")
            num_lote = request.form.get("num_lt")
            loc_ = request.form.get("loc_")
            descricao = request.form.get("descricao")
            dt_enter = request.form.get("dt_enter")
            qt_item = int(request.form.get("qt_item") or 0)  # Define 0 se o
campo estiver vazio
            dt_end = request.form.get("dt_end")
            qt_saida = int(request.form.get("qt_saida") or 0)  # Define 0 se o
campo estiver vazio
            _saldo = int(request.form.get("_saldo") or 0)  # Define 0 se o campo
estiver vazio
            funcionario = request.form.get("funcionario")

            # armazenando o banco de dados de cada usuário logado e seus
respectivos códigos que são AUTO_INCREMENT
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # Conectar ao banco e buscar o saldo atual
            mydb = Conexao.conectar()

            mycursor = mydb.cursor()
            # adicionando o saldo na tabela tb_estoque
            mycursor.execute(f"SELECT saldo FROM {turma}.tb_estoque WHERE
cod_prod_est = %s", (cod_prod,))
            resultado = mycursor.fetchone()
            saldo_atual = resultado[0] if resultado else 0

            # Atualizar saldo
            saldo_novo = saldo_atual + qt_item - qt_saida

            # Inserir/Atualizar registro no estoque
            tbEstoque = Estoque()
            sucesso = tbEstoque.estoque(cod_prod, num_lote, loc_, descricao,
dt_enter, qt_item, dt_end, qt_saida, saldo_novo, funcionario, cod_aluno, turma)

            mycursor.execute(
                f"UPDATE {turma}.tb_cadastramento SET quantidade = quantidade +
%s WHERE cod_prod = %s",
                (qt_item - qt_saida, cod_prod)
            )
            mydb.commit()

            if sucesso:
```

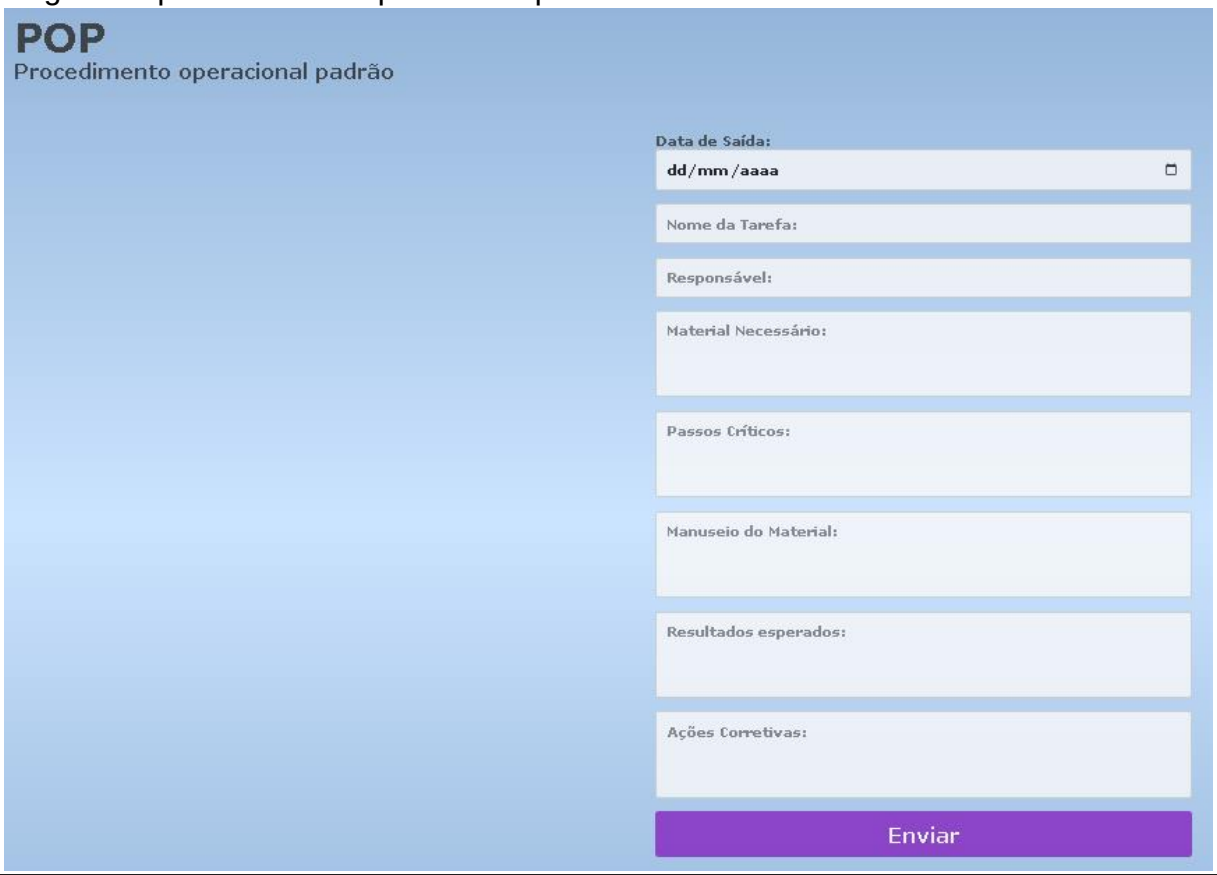
```
                flash("alert('Movimentação de estoque registrada com
sucesso!')")
                return redirect("/")
            else:
                return "Erro ao registrar movimentação de estoque"

    else:
        return redirect("/login")
```

```
body{
    position: relative;
    width: 100rem;
    height: 80rem;
    background: var( --cor-icone-cadastramento);
    border-radius: 2.5rem;
    margin-bottom: 1.5rem;
    opacity: 0.8;
}

/* Geral cadastramento: section que engloba toda interface dessa página. */
.geral-cadastramento{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}




.controle1{
    position: absolute;
    left: 30.8rem;
    color: #292525;
    margin-top: 10rem;
    font-size: 2.75rem;
}

.controle2{
    position: absolute;
    left: 30.8rem;
    font-size: 2.75rem;
    color: #292525;
    margin-top: 12rem;
}

.formulario{
    margin-top: 16rem;
}

/* Form Container: section que determina o formulário. */
```

```
.form_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 41rem;
    padding: 20px;
    border-radius: 8px;
    margin-bottom: 2rem;
    margin-top: 2rem;

}

/* Determina todas as regiões funcionais do formulário. Tudo o que o usuário
clica. */
.input-field, .textarea-field, .select-menu, .submit-btn {
    width: 100%;
    margin-bottom: 15px;
    padding: 12px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
    background-color: var(--cor-fundo-formularios);
}

/* Espaço da Descrição Técnica. */
.textarea-field {
    height: 100px;
    resize: none;
}

/* Estiliza o botão de enviar. */
.submit-btn {
    background-color: var(--cor-btn-enviar);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition: background-color 0.3s;
    font-size: 1.5rem;
}

/* Estiliza o botão de enviar quando passamos o mouse. */
.submit-btn:hover {
    background-color: var(--cor-hover-btn-enviar);
}

.cadastramentoD {
    font-size: medium;
    font-weight: 700;
}

table {
```

```
    width: 100%;
    border-collapse: collapse;
}

table, th, td {
    border: 1px solid black;
}

th, td {
    padding: 8px;
    text-align: left;
}

th {
    background-color: #f2f2f2;
}
```

Página de procedimento operacional padrão




```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/pop.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
```

```
{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}

    <!-- Section 01.
    Essa Section possui um espaço para imagem (cor solicitada pelo cliente) e
nome da matéria. -->
    <section class="geral-cadastramento">

            <div class="cadastramento-titulo">
                <span class="cadastramento1"> POP </span><br>
                <span      class="cadastramento2">Procedimento      operacional
padrão</span>
            </div>

    <!-- Section 02. Formulário de POP (procedimento operacional padrão)
    Section que contém o formulário composto por inputs. Foi utilizado "text"
para dados escritos e "date" para datas. Na descrição técnica, optamos por
textarea
    porque facilita a inserção de textos extensos. -->
        <section class="formulario">
            <div class="form_container">
                <form action="/pop" method="POST">

                    <span class="cadastramentoD"> Data de Saída: </span>
                    <input    type="date"    placeholder="Data    de    Saída:"
class="input-field" name="dt_end1">

                    <input    type="text"    placeholder="Nome    da    Tarefa:"
class="input-field" name="task_name">

                    <input type="text" placeholder="Responsável:" class="input-
field" name="resp_">

                    <textarea        placeholder="Material        Necessário:"
class="textarea-field" name="material"></textarea>
                    <textarea placeholder="Passos Críticos:" class="textarea-
field" name="passos"></textarea>
                    <textarea     placeholder="Manuseio     do     Material:"
class="textarea-field" name="manuseio"></textarea>
                    <textarea        placeholder="Resultados        esperados:"
class="textarea-field" name="resultados"></textarea>
                    <textarea placeholder="Ações Corretivas:" class="textarea-
field" name="acoes"></textarea>

                    <!-- Botão para enviar os dados para o Banco de Dados -->
                    <button class="submit-btn">Enviar</button>

                </form>
            </div>
```

```
        </section>

    </section>

{% endblock %}
# importando a classe necessária para poder realizar a conexão com o banco de dados
from conexao import Conexao
# classe que armazena as funções que correspondem ao registro de pop
class Pop:
    def __init__(self):
        self.data_pop = None
        self.tarefa_pop = None
        self.responsavel_pop = None
        self.material_pop = None
        self.passos_pop = None
        self.manuseio_pop = None
        self.resul_pop = None
        self.acao_pop = None
        self.cod_aluno = None

    # criando a função para realizar o processo de pop
    def pop(self, data_pop, tarefa_pop, responsavel_pop, material_pop, passos_pop, manuseio_pop, resul_pop, acao_pop, cod_aluno, turma):
        # conectando com o banco de dados
        mydb = Conexao.conectarAluno(turma)

        mycursor = mydb.cursor()

        valores = (data_pop, tarefa_pop, responsavel_pop, material_pop, passos_pop, manuseio_pop, resul_pop, acao_pop, cod_aluno)

        dados = f"INSERT INTO tb_pop(data_pop, tarefa_pop, responsavel_pop, material_pop, passos_pop, manuseio_pop, resul_pop, acao_pop, cod_aluno) VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s)"

        #executar
        mycursor.execute(dados, valores)

        mydb.commit()

        mydb.close()

        return True
```

```
# roteamento da página dos processos de registro pop
# RF009
@app.route("/pop", methods=["GET", "POST"])
def pagina_pop():
    # verificando se há algum perfil logado no sistema
    if "usuario_logado" in session or "professor_logado" in session:
```

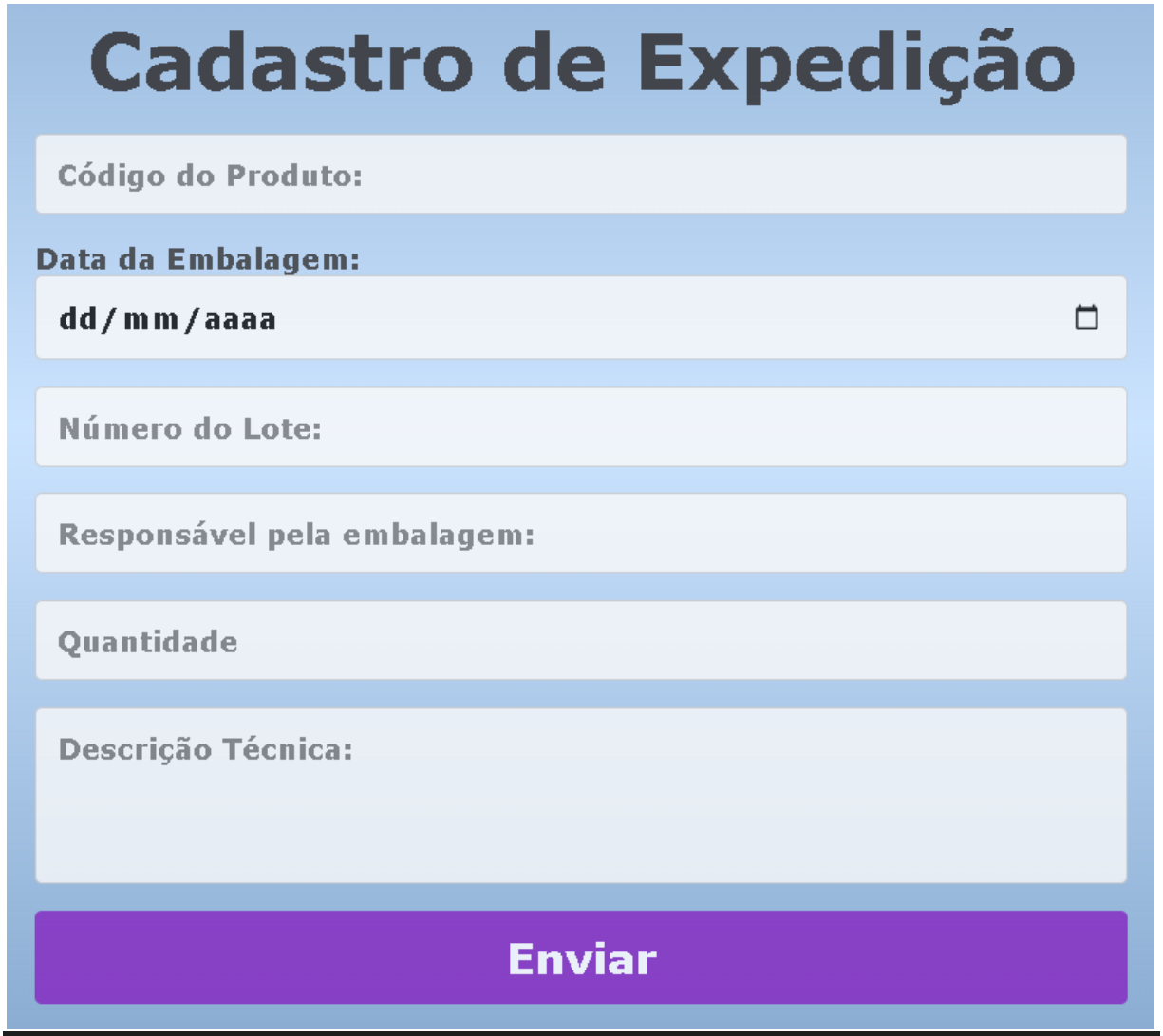
```
        if request.method == "GET":
            return render_template("pop.html")
        if request.method == "POST":
            # pegando os dados que foram enviados pelo formulário
            dt_end1 = request.form.get("dt_end1")
            task_name = request.form.get("task_name")
            resp_ = request.form.get("resp_")
            material = request.form.get("material")
            passos = request.form.get("passos")
            manuseio = request.form.get("manuseio")
            resultados = request.form.get("resultados")
            acoes = request.form.get("acoes")

            # criando um objeto com a classe Pop
            tbPop = Pop()

            # armazenando os dados da session em uma variável
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # realizandpo o processo de registro de POP por meio de uma função
que está armazenada no objeto criado anteriormente
            if tbPop.pop(dt_end1, task_name, resp_, material, passos, manuseio,
resultados, acoes, cod_aluno, turma):
                # emitindo uma mensagem para quando o processo for realizado com
sucesso
                flash("alert('Parabéns, você acaou de realizar o processo de
registro de POP!!🎉')")
                return redirect ('/')
            else:
                # emitindo uma mensagem para quando o processo não for realizado
                return 'Erro ao realizar o processo de POP'
    else:
        return redirect("/login")
```

```
body{
    position: relative;
    width: 100rem;
    height: 80rem;
    background-image: var( --cor-icone-cadastramento);
    border-radius: 2.5rem;
    margin-bottom: 1.5rem;
    opacity: 0.8;
    overflow-x: hidden;
}

/* Geral cadastramento: section que engloba toda interface dessa página. */
```

```
.geral-cadastramento{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}



/* Título do cadastramento: Controle de Estoque */
.cadastramento1{
    position: absolute;
    left: 30.8rem;
    font-size: 3rem;
    color: #292525;
    margin-top: 10rem;
    font-size: 2.75rem;
}

.cadastramento2{
    position: absolute;
    left: 30.8rem;
    font-size: 1.5rem;
    color: #292525;
    margin-top: 12rem;
}

.formulario{
    margin-top: 16rem;
}

/* Form Container: section que determina o formulário. */
.form_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 41rem;
    padding: 20px;
    border-radius: 8px;
    margin-bottom: 2rem;

}

/* Determina todas as regiões funcionais do formulário. Tudo o que o usuário
clica. */
.input-field, .textarea-field, .select-menu, .submit-btn {
    width: 100%;
    margin-bottom: 15px;
    padding: 12px;
    font-size: 16px;
    border: 1px solid #ccc;
```

```
    border-radius: 4px;
    background-color: var(--cor-fundo-formularios);
}

/* Espaço da Descrição Técnica. */
.textarea-field {
    height: 100px;
    resize: none;
}

/* Estiliza o botão de enviar. */
.submit-btn {
    background-color: var(--cor-btn-enviar);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition: 0.3s;
    font-size: 1.5rem;
}

/* Estiliza o botão de enviar quando passamos o mouse. */
.submit-btn:hover {
    background-color: var(--cor-hover-btn-enviar);
}

.cadastramentoD {
    font-size: medium;
    font-weight: 700;
}
```

Página de cadastro de expedição




```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/estoque.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">

{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}

    <!-- Section 01.
    Essa Section possui um espaço para imagem (cor solicitada pelo cliente) e
nome da matéria. -->
<section class="geral-cadastramento">

        <div class="cadastramento-titulo">
            <span class="controle1">CONTROLE DE</span><br>
            <span class="controle2">ESTOQUE</span>
        </div>
```

```
    <!-- Formulário para cadastro de estoque com todos os campos originais -->
    <div class="formulario">
        <div class="form_container">
            <form action="/estoque" method="POST">
                <span class="cadastramentoD"> Cadastrar dados: </span>

                <!-- Campo de código do produto para busca automática -->
                <input type="number" placeholder="Código do Produto"
class="input-field" name="cod_prod" id="codigoProduto">

                <!-- Campos já existentes para informações do produto -->
                <input type="number" placeholder="Número de Lote" class="input-
field" name="num_lt" id="numeroLote">
                <input type="text" placeholder="Localização" class="input-
field" name="loc_" id="enderecamentoProduto">
                <textarea placeholder="Descrição Técnica" class="textarea-
field" name="descricao" id="descricaoProduto"></textarea>
                <span class="cadastramento">Data de entrada:</span>
                <input type="date" placeholder="Data de Entrada:" class="input-
field" name="dt_enter">
                <input type="number" placeholder="Quantidade de itens"
class="input-field" name="qt_item">
                <span class="cadastramento">Data de saída:</span>
                <input type="date" placeholder="Data de Saída:" class="input-
field" name="dt_end">
                <input type="number" placeholder="Quantidade da Saída"
class="input-field" name="qt_saida">
                <input type="number" placeholder="Saldo" class="input-field"
name="_saldo" id="saldoProduto">
                <input type="text" placeholder="Funcionário Responsável"
class="input-field" name="funcionario">

                <!-- Botão de envio do formulário -->
                <button class="submit-btn">Enviar</button>
            </form>
        </div>
    </div>
</section>

<script>
    document.addEventListener("DOMContentLoaded", function() {
        const produtos = {{ lista_produtos | tojson }};

        const codigoInput = document.getElementById("codigoProduto");

        codigoInput.addEventListener("input", function() {
            const codigoDigitado = codigoInput.value.trim();

            const produto = produtos.find(p => p.codigo === codigoDigitado);

            if (produto) {
```

```
                document.getElementById("descricaoProduto").value             =
produto.descricao;
                document.getElementById("numeroLote").value                   =
produto.numero_lote;
                document.getElementById("enderecamentoProduto").value         =
produto.enderecamento;
                document.getElementById("saldoProduto").value                 =
produto.quantidade;
            }
        });
    });
</script>


{% endblock %}
```

```
# importando a classe necessária para poder realizar a conexão com o banco de
dados
from conexao import Conexao
# classe que irá armazenar as funções referentes ao processo de cadastro de
expedição
class Expedicao:
    def __init__(self):
        self.cod_prod_exp = None
        self.desc_exp = None
        self.num_lote_exp = None
        self.quant_exp = None
```

```
  self.data_emb_exp = None
        self.responsavel_exp = None
        self.cod_aluno = None

    # criando a função para cadastro da expedição dos produtos
    def  expedicao(self,  cod_prod_exp,  desc_exp,  num_lote_exp,  quant_exp,
data_emb_exp, responsavel_exp, cod_aluno, turma):
        # conectando com o banco de dados
        mydb = Conexao.conectarAluno(turma)

        mycursor = mydb.cursor()

        valores   =   (cod_prod_exp,   desc_exp,   num_lote_exp,   quant_exp,
data_emb_exp, responsavel_exp, cod_aluno)

        dados = f"INSERT INTO tb_expedicao(cod_prod_exp, desc_exp, num_lote_exp,
quant_exp, data_emb_exp, respnsavel_exp, cod_aluno) VALUES (%s, %s, %s, %s, %s,
%s, %s)"

        #executar
        mycursor.execute(dados, valores)
```

```
        mydb.commit()

        mydb.close()

        return True
```

```
# roteamento da página dos processos de registro expedição
# RF010
@app.route("/expedicao", methods=["GET", "POST"])
def pagina_expedicao():
    # verificando se um dos usuários estão conectados para habilitar a visualização da página
    if "usuario_logado" in session or "professor_logado" in session:
        if request.method == "GET":
            #conectando com o banco de dados
            mydb = Conexao.conectar()

            mycursor = mydb.cursor()

            # armazenando o banco de dados de cada usuário logado e seus respectivos códigos que são AUTO_INCREMENT
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # armazenando o banco de dados de cada usuário logado e seus respectivos códigos que são AUTO_INCREMENT
            produtos = (f"SELECT * FROM {turma}.tb_cadastramento")

            mycursor.execute(produtos)

            resultado = mycursor.fetchall()

            # criando uma lista para armazenar os produtos
            lista_produtos = []

            # usando um loop para ir adicionando os produtos na lista (variável) que foi criada anteriormente
            for produto in resultado:
                lista_produtos.append({
                    "codigo":produto[0],
                    "descricao":produto[1],
                    "modelo":produto[2],
                    "fabricante":produto[3],
                    "numero_lote":produto[4],
                    "enderecamento":produto[5],
                    "quantidade":produto[6]
                })
```

```
            return                              render_template("expedicao.html",
lista_produtos=lista_produtos)

        if request.method == "POST":
            # armazenando os dados do formulário
            cod_prod = request.form.get("cod_prod")
            data_saida = request.form.get("data_saida")
            num_lote = request.form.get("num_lote")
            responsavel = request.form.get("responsavel")
            quantidade = request.form.get("quantidade")
            descricao_tec = request.form.get("descricao_tec")

            # transformando a classe em um objeto
            tbExpedicao = Expedicao()

            # pegando dados que foram armazenados na session e "guardando" em
uma variável
            if "usuario_logado" in session:
                turma = session['usuario_logado']['turma']
                cod_aluno = session['usuario_logado']['cod_aluno']
            else:
                turma = session['professor_logado']['turma']
                cod_aluno = session['professor_logado']['cod_aluno']

            # realizando o processo de registro de expedição
            if    tbExpedicao.expedicao(cod_prod,    descricao_tec,    num_lote,
quantidade, data_saida, responsavel, cod_aluno, turma):
                # exibindo uma mensagem na interface do usuário para quando o
cadastro for realizado com sucesso
                flash("alert('Parabéns, você acabou de realizar o processo de
registro de expedição!!🎉')")
                return redirect ('/')
            else:
                # exibindo uma mensagem na interface do usuário para quando o
cadastro não for realizado
                return "Erro ao realizar o processo de cadastro de expedição."
    else:
        return redirect("/login")
```

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600
&display=swap');

body{
    position: relative;
    width: 100rem;
    height: 80rem;
    background: var( --cor-icone-cadastramento);
    border-radius: 2.5rem;
    margin-bottom: 1.5rem;
    opacity: 0.8;
```

```
}

.geral-cadastramento{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}

.container-cadastramento{
    margin-top: 10rem;
    display: flex;
    align-items: center;
    text-align: center;
    justify-content: space-around;
    gap: 20rem;
}

.expedicao1{
    position: absolute;
    left: 30.5rem;
    font-size: 2.75rem;
    color: #292525;
}


.formulario{
    margin-top: 2rem;

}

.form_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 40rem;
    padding: 10px;
    border-radius: 8px;
    margin-bottom: 2rem;

}

.input-field, .textarea-field, .select-menu, .submit-btn {
    width: 100%;
    margin-bottom: 15px;
    padding: 12px;
    font-size: 16px;
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    background-color: var(--cor-fundo-formularios);
}
```

```
.textarea-field {
    height: 100px;
    resize: none;
}

.select-menu {
    position: relative;
}

.select-btn {
    display: flex;
    justify-content: space-between;
    align-items: center;
    cursor: pointer;
}

.select-btn i {
    font-size: 20px;
    transition: 0.3s;
}

.select-menu.active .select-btn i {
    transform: rotate(-180deg);
}

.options {
    position: absolute;
    top: 100%;
    left: 0;
    width: 100%;
    padding: 0;
    list-style: none;
    background-color: var(--cor-fundo-options);
    border: 1px solid var(--cor-bordas);
    border-radius: 4px;
    max-height: 150px;
    overflow-y: auto;
    display: none;
}

.select-menu.active .options {
    display: block;
    z-index: 10;
}

.option {
    padding: 10px;
    cursor: pointer;
}

.option:hover {
```

```
    background-color: var(--cor-hover-options);
}

.submit-btn {
    background-color: var(--cor-btn-enviar);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition:  0.3s;
    font-size: 1.5rem;
}

.submit-btn:hover {
    background-color: var(--cor-hover-btn-enviar);
}

table {
    width: 100%;
    border-collapse: collapse;
}

table, th, td {
    border: 1px solid black;
}

th, td {
    padding: 8px;
    text-align: left;
}

th {
    background-color: #f2f2f2;
}
```

Página para gerenciar a turma

```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/listarBD.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}

<section class="gerenciador">
    <h2>Lista das turmas</h2>
    <table>
        <thead>
            <tr>
                <th>Nome da turma</th>
                <th>Ações</th>
            </tr>
        </thead>
        <tbody>
            {% for banco in lista_bancos %}
            <tr>
                <td>{{ banco.nome }}</td>
                <td>
                    <form          action="{{          url_for('excluir_banco',
nomeBD=banco.nome) }}" method="post">
                        <button    type="submit"    class="btn-submit">Encerrar
turma</button>
                    </form>
                </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</section>

<script >
    {% with messages = get_flashed_messages() %}
        {% for mensagem in messages %}
            {{ mensagem|safe }}
        {% endfor %}
    {% endwith %}
</script>

{% endblock %}
```

```
# roteamento para exibir todos os bancos de dados que foram criados pelo
professor
@app.route("/professor/listarBD")
def listar_bancos():
    # verificando se o usuário está logado para poder exibir a página
    if "professor_logado" in session:
```

```
        # Conectando ao banco de dados
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()

        # Buscando os nomes dos bancos de dados criados
        mycursor.execute("SELECT nomeBase FROM databaseprofessor.tb_database")
        bancos = mycursor.fetchall()

        # Convertendo o resultado em uma lista de dicionários
        lista_bancos = [{"nome": banco[0]} for banco in bancos]

        # Fechando a conexão
        mycursor.close()
        mydb.close()

        # Renderizando o template com a lista de bancos de dados
        return render_template("listar_bancos.html", lista_bancos=lista_bancos)
    else:
        return "Acesso negado", 403

# roteamento para excluir os bancos de dados criados pelo professor
@app.route("/professor/excluirBD/<nomeBD>", methods=["POST"])
def excluir_banco(nomeBD):
    # verificando se há algum usuário logado para poder liberar a visualização
da página
    if "professor_logado" in session:
        # Conectando ao banco de dados
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()

        # Comando para excluir o banco de dados
        mycursor.execute(f"DROP DATABASE IF EXISTS {nomeBD}")

        # Remover o banco de dados da tabela de referência
        mycursor.execute("DELETE   FROM   databaseprofessor.tb_database   WHERE
nomeBase = %s", (nomeBD,))

        mydb.commit()
        mycursor.close()
        mydb.close()

        # mensagem na interface do usuário para quando a turma for excluída
        flash("alert('Turma finalizada com sucesso!!🎉')")
        # retornando para a página em que estão sendo exibidos os bancos de
dados em forma de lista
        return redirect("/professor/listarBD")
    else:
        return "Acesso negado", 403
```

```
body{
    margin: 0;
```

```
    padding: 0;
    display: flex;
    background-color: var(--cor-fundo-pg-funcoes);
    justify-content: center;
    align-items: center;
    height: 100vh;
    flex-direction: column;
}

.gerenciador{
    display: flex;
    width: 100%;
    height: 100%;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}

.gerenciador h2{
    font-size: 3.5rem;
    margin-bottom: 4rem;
}

.gerenciador table{
    width: 80%;
    border: 0.2rem solid var(--cor-bordas);
    background: var(--cor-fundo-formularios);
}

table {
    width: 100%;
    border-collapse: collapse;
}

table, th, td {
    border: 0.2rem solid var(--cor-bordas);
    color: black;
}

td{
    padding: 8px;
    text-align: center;
    font-size: 1.3rem;
}

th {
    background-color: #f2f2f2;
    padding: 1rem;
    font-size: 1.3rem;
}

.btn-submit{
```

```
    background-color: var(--cor-btn-enviar-bd);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition: background-color 0.3s;
    padding: 1rem;
    border-radius: 2rem;
    font-size: 1.3rem;
}

.btn-submit:hover {
    background-color: var(--cor-fundo-input-bd  );
}
```

Página para criar turma

HTML

```
{% extends "layout.html" %}
{% block css %}
<link rel="stylesheet" href="/static/styles/professor.css">
<link rel="stylesheet" href="/static/styles/media/professor-media.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
{% endblock %}

{% block conteudo %}

<section class="cadastro_turma">

    <div class="titulomensagem">
        <h2>Cadastre sua turma</h2>
        <p>A turma deve ser cadastrada sem espaços e caracteres especiais!</p>
    </div>

    <form action="/criarBD" method="POST">
```

```
        <input type="text" placeholder="Nome da Turma" name="nomeTurma">
        <button type="submit">ENVIAR</button>
    </form>

</section>

{% endblock %}
```

Python

```
    def cadastrarProf (self, nome_prof, email_prof, senha_espec):
        # conectando com o banco de dados
        mydb = Conexao.conectar()

        mycursor = mydb.cursor()

        dados = f"INSERT INTO tb_aluno (nome, email, senha) VALUES
('{nome_prof}', '{email_prof}', '{senha_espec}')"

        #executar
        mycursor.execute(dados)

        mydb.commit()

        mydb.close()

        return True
```

CSS

```
body {
    margin: 0;
    padding: 0;
    color: var(--cor-padrao-letras);
    height: 100vh;
}

.cadastro_turma{
    width: 100vw;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}


.titulomensagem h2{
    margin-bottom: 2rem;
    font-size: 3rem;
}
```

```css
.titulomensagem p{
    font-size: 1.1rem;
    text-align: center;
    margin-bottom: 2rem;
}


input {
    padding: 1rem;
    width: 97%;
    border: none;
    border-radius: 0.5rem;
    outline: none;
    background-color: var(--cor-drop-btn);
    color: var(--cor-padrao-letras);
    margin-bottom: 2rem;
}

input::placeholder{
    color: var(--cor-padrao-letras);
}



button[type="submit"] {
    padding: 1rem;
    border: none;
    border-radius: 0.5rem;
    background-color: var(--cor-btn-enviar-bd);
    color: var(--cor-padrao-letras);
    font-weight: bold;
    cursor: pointer;
    width: 100%;
}
```

CSS Media

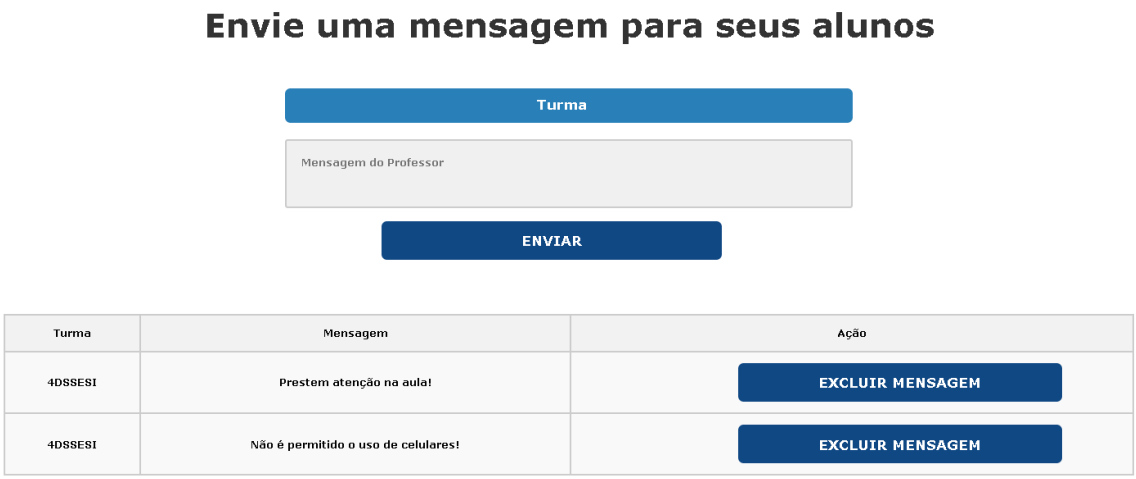
```css
/* Media de telas para smartphones */
@media (max-width: 576px) {

    .titulomensagem h2{
        margin-bottom: 1rem;
        font-size: 1.5rem;
        text-align: center;
    }

    .titulomensagem p{
        font-size: 1rem;
        margin-bottom: 1.2rem;
    }
```

```
    input {
        padding: 1rem;
        width: 70%;
        outline: none;
        margin-bottom: 1rem;
        margin-left: 3.5rem;
        text-align: center;
    }


    button[type="submit"] {
        width: 50%;
        margin-left: 5.5rem;
    }


}

/* Media de telas para Tablets */
@media (min-width: 768px){

    .titulomensagem h2{
        margin-bottom: 2rem;
        font-size: 2.5rem;
        text-align: center;
    }

    .titulomensagem p{
        font-size: 1.3rem;
        margin-bottom: 1.5rem;
    }



    input {
        padding: 1rem;
        width: 80%;
        margin-bottom: 1rem;
        margin-left: 3.5rem;
        text-align: center;
    }

    input::placeholder{
        font-size: 1.3rem;
    }


    button[type="submit"] {
        width: 50%;
        margin-left: 8.5rem;
    }
```

```
}

/* Media de telas para Desktop */
@media (min-width: 1200px){

    .titulomensagem h2{
        margin-bottom: 1.6rem;
        font-size: 3rem;
    }

    .titulomensagem p{
        font-size: 1.3rem;
        margin-bottom: 1.5rem;
    }


    input {
        width: 90%;
        margin-left: 2.5rem;
    }

    input::placeholder{
        font-size: 1.3rem;
    }

    button[type="submit"] {
        width: 60%;
        margin-left: 7.5rem;
        font-size: 1.2rem;
    }

}

/* Media de telas para Desktop */
@media (min-width: 1440px){

    .titulomensagem h2{
        font-size: 3.5rem;
    }

    .titulomensagem p{
        font-size: 1.4rem;
        margin-bottom: 1.5rem;
    }

    input {
        width: 100%;
        margin-left: 1rem;
    }
```

```
}
```

Página para enviar mensagem para a turma



HTML

```
{% extends "layout.html" %}
{% block css %}
<link rel="stylesheet" href="/static/styles/mensagem.css">

<link rel="stylesheet" href="/static/styles/variaveis.css">


{% block conteudo %}


{% endblock %}


<!-- Início da seção de mensagem do professor -->
<div class="container-mensagem_professor">
    <div class="titulomensagem">
        <h2>Envie uma mensagem para seus alunos</h2>
    </div>
    <div class="mensagem">
        <form action="/enviar_mensagem" method="POST">
            <input list="browsers" name="turma" placeholder="Turma">
            <datalist id="browsers">
            {% for nomeBD in lista_nomes %}
            <option
value="{{nomeBD['database']}}">{{nomeBD['database']}}</option>
            {% endfor %}
            </datalist>
            <!-- Campo para escrever a mensagem para os alunos -->
```

```
            <textarea placeholder="Mensagem do Professor" class="textarea-
mensagem" name="mensagem" type="text"></textarea>
            <button type="submit">ENVIAR</button>
        </form>
    </div>

    <!-- Exibição das mensagens com opção de exclusão para o professor -->
    <div class="container-mensagens-enviadas">

        <table>
            <thead>
                <tr>
                    <th>Turma</th>
                    <th>Mensagem</th>
                    <th>Ação</th>
                </tr>
            </thead>
            <tbody>
                {% for mensagem in lista_mensagens %}
                <tr>
                    <td><small>{{ mensagem['turma'] }}</small> <!-- Exibe a
turma da mensagem --></td>
                    <td>{{ mensagem['mensagem'] }}</td>
                    <td>
                        <form action="/excluir_mensagem" method="POST"
style="display:inline;">
                            <input type="hidden" name="mensagem_id" value="{{
mensagem['cod_mensagem'] }}">
                            <button type="submit" class="btn-excluir">EXCLUIR
MENSAGEM</button>
                        </form>
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</div>

<!-- Fim da seção de mensagem do professor -->
{% endblock %}
```

Python

```
# roteamento da página que o professor utiliza para enviar mensagens para os
alunos
@app.route("/enviar_mensagem", methods=["GET", "POST"])
def enviar_mensagens():
    # verificando se o usuário está logado para poder permitir a vizualização
da página
    if "professor_logado" in session:
        # Conectando ao banco de dados
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()
```

```
        if request.method == "GET":
            # Consulta para obter a lista de turmas
            mycursor.execute("SELECT * FROM databaseprofessor.tb_database")
            resultado = mycursor.fetchall()
            lista_nomes = [{"database": nomeBD[0]} for nomeBD in resultado]

            # Consulta para obter a lista de mensagens enviadas
            mycursor.execute("SELECT  cod_mensagem,  mensagens,  turma  FROM
tb_mensagens")
            mensagens = mycursor.fetchall()
            lista_mensagens  =  [{"cod_mensagem":  msg[0],  "mensagem":  msg[1],
"turma": msg[2]} for msg in mensagens]

            # Fechar a conexão com o banco de dados
            mydb.close()

            # Renderizar a página com as listas de turmas e mensagens
            return  render_template("mensagem.html",  lista_nomes=lista_nomes,
lista_mensagens=lista_mensagens)

        if request.method == "POST":
            # Conectando ao banco de dados para enviar mensagem
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Pega a mensagem do formulário
            mensagem = request.form.get("mensagem")
            bancoDados = request.form.get("turma")

            # Inserir a nova mensagem no banco de dados
            inserir_mensagem  =  f"INSERT  INTO  tb_mensagens  (mensagens,  turma)
VALUES (%s, %s)"
            mycursor.execute(inserir_mensagem, (mensagem, bancoDados))
            mydb.commit()
            mydb.close()

            # Emite uma mensagem de confirmação ao usuário
            flash("alert('Mensagem enviada para a turma com sucesso! 🥕')")
            return redirect("/enviar_mensagem")



# rota em que está a função que o professor usará para excluir uma mensagem do
banco de dados e da interface do usuário
@app.route("/excluir_mensagem", methods=["POST"])
def excluir_mensagem():
    if "professor_logado" in session:
        # peagndo o id da mensagem
        mensagem_id = request.form.get("mensagem_id")
```

```
        if mensagem_id:
            # Conectando ao banco de dados
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Query para deletar a mensagem com o ID fornecido
            delete_query = "DELETE FROM databaseProfessor.tb_mensagens WHERE
cod_mensagem = %s"

            # Executar a query passando o ID da mensagem
            mycursor.execute(delete_query, (mensagem_id,))
            mydb.commit()
            mydb.close()

            # mensagem na interface do usuário para quando der certo o processo
de excluir uma mensagem
            flash("alert('Mensagem excluída com sucesso!')")
        else:
            # mensagem na interface do usuário para quando não for possível
excluir uma mensagem
            flash("alert('Erro ao excluir a mensagem. ID inválido.')")
    # retornando para a p´gina inicial
    return redirect("/")
```

```
body {
    margin: 0;
    padding: 0;
    color: var(--cor-padrao-letras);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    flex-direction: column;
    overflow-x: hidden;
}

.container-mensagem_professor{
    width: 90%;
    padding-top: 8rem;
}

.titulomensagem{
    margin-top: 10rem;
}

.titulomensagem h2{
    margin-bottom: 4rem;
    font-size: 3rem;
    text-align: center;
```

```
}

.mensagem{
    display: flex;
    justify-content: space-around;
    align-items: center;

}


input {
    padding: 1rem;
    width: 100%;
    border: none;
    border-radius: 0.5rem;
    outline: none;
    background-color: var(--cor-drop-btn);
    color: var(--cor-padrao-letras);
    text-align: center;
    margin-bottom: 1.5rem;
}

input::placeholder{
    color: var(--cor-padrao-letras);
    font-size: 1.3rem;
}

.textarea-mensagem{
    width: 100%;
    margin-bottom: 1rem;
    padding:1.3rem;
    font-size: 1rem;
    border: 0.2rem solid var(--cor-bordas);
    border-radius: 4px;
    background-color: var(--primary-color-light);
}

.textarea-mensagem {
    height: 100px;
    resize: none;
}

button[type="submit"] {
    padding: 1rem;
    border: none;
    border-radius: 0.5rem;
    background-color: var(--cor-btn-enviar-bd);
    color: var(--cor-padrao-letras);
    font-weight: bold;
    cursor: pointer;
    width: 60%;
    margin-left: 8.8rem;
```

```
    font-size: 1.3rem;
}


.container-mensagens-enviadas{
    display: flex;
    margin-top: 5rem;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    margin-bottom: 4rem;
}

.container-mensagens-enviadas table{
    width: 80%;
    border: 0.2rem solid var(--cor-bordas);
    background-color: var(--cor-fundo-formularios);
}

table{
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 3rem;
}

table, th, td{
    border: 0.2rem solid var(--cor-bordas);
    color: black;
}

td{
    padding: 1rem;
    text-align: center;
}

th{
    background-color: #f2f2f2;
    padding: 1rem;
}

.btn-excluir{
    background-color: var(--cor-btn-enviar);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition: background-color 0.3s;
    padding: 1rem;
    border-radius: 2rem;
}

.btn-excluir:hover{
    background-color: var(--cor-hover-btn-enviar);
```

```
}
```

Página para redefinir a senha dos alunos

HTML

```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/redefinir_senha.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
{% endblock %}


{% extends "layout.html" %}
{% block conteudo %}
<section class="container-redefinir-senha">

        <h2>Redefinir a senha dos Alunos</h2>

    <table>
        <thead>
            <tr>
                <th>Nome</th>
                <th>Email</th>
                <th>Turma</th>
                <th>Nova Senha</th>
            </tr>
        </thead>
        <tbody>
            {% for aluno in alunos %}
            <tr>
                <td>{{ aluno[0] }}</td>
                <td>{{ aluno[1] }}</td>
                <td>{{ aluno[2] }}</td>
                <td>
                    <form     action="{{     url_for('redefinir_senha')     }}"
method="POST">
                        <input    type="hidden"    name="cod_aluno"    value="{{
aluno[4] }}">
```

```
                        <input  type="hidden"  name="turma"  value="{{  aluno[2]
}}">
                        <input        type="password"        name="nova_senha"
placeholder="Nova Senha" required>
                        <button           type="submit"           class="btn-
submit">Redefinir</button>
                    </form>
                </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</section>
{% endblock %}
```

Python

```
def redefinir_senha():
    # Verifica se o professor está logado
    if "professor_logado" not in session:
        return redirect("/login")

    if request.method == "GET":
        # Conectando ao banco de dados do professor para obter as turmas
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()
        mycursor.execute("SELECT nomeBase FROM databaseprofessor.tb_database")
        turmas = mycursor.fetchall()
        mydb.close()

        # Lista para armazenar dados de todos os alunos
        alunos = []

        # Itera sobre cada turma e busca os alunos
        for turma in turmas:
            nome_turma = turma[0]
            try:
                mydb_turma = Conexao.conectarAluno(nome_turma)  # Conectando ao
banco da turma
                cursor_turma = mydb_turma.cursor()
                cursor_turma.execute("SELECT nome, email, %s AS turma, senha,
cod_aluno FROM tb_aluno", (nome_turma,))
                alunos.extend(cursor_turma.fetchall())
            finally:
                mydb_turma.close()

        return render_template("redefinir_senha.html", alunos=alunos)

    elif request.method == "POST":
        # Coletando dados do formulário para atualização
        cod_aluno = request.form.get("cod_aluno")
        nova_senha = request.form.get("nova_senha")
```

```
        turma = request.form.get("turma")

        # Atualiza a senha no banco de dados da turma específica
        mydb_turma = Conexao.conectarAluno(turma)
        cursor_turma = mydb_turma.cursor()
        cursor_turma.execute("UPDATE tb_aluno SET senha = %s WHERE cod_aluno =
%s", (nova_senha, cod_aluno))
        mydb_turma.commit()
        mydb_turma.close()

        flash("Senha redefinida com sucesso!")
        return redirect("/redefinir_senha")
```

CSS

```
body{
    margin: 0;
    padding: 0;
    display: flex;
    background-color: var(--cor-fundo-pg-funcoes);
    justify-content: center;
    align-items: center;
    height: 100vh;
    flex-direction: column;
}

.container-redefinir-senha h2{
    font-size: 3rem;
    margin-bottom: 4rem;
}

.container-redefinir-senha{
    display: flex;
    width: 100%;
    height: 100%;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}



.container-redefinir-senha table{
    width: 60%;
    border: 0.2rem solid var(--cor-bordas);
    background: var(--cor-fundo-formularios);
}

table {
    width: 100%;
    border-collapse: collapse;
}
```

```
table, th, td {
    border: 0.2rem solid var(--cor-bordas);
    color: black;
}

td{
    padding: 8px;
    text-align: center;
    font-size: 1.3rem;
}

th {
    background-color: #f2f2f2;
    padding: 1rem;
    font-size: 1.3rem;
}

.btn-submit{
    background-color: var(--cor-btn-enviar-bd);
    color: var(--cor-fundo-options);
    border: none;
    cursor: pointer;
    transition: background-color 0.3s;
    padding: 1rem;
    border-radius: 2rem;
    font-size: 1.3rem;
}

.btn-submit:hover {
    background-color: var(--cor-fundo-input-bd);
}
```

Página do inventário


```
<!-- EXTENDENDO O CSS DA PAGINA PRINCIPAL (LAYOUT) -->
{% block css %}
<link rel="stylesheet" href="/static/styles/media/layout-media.css">
<link rel="stylesheet" href="/static/styles/inventario.css">
<link rel="stylesheet" href="/static/styles/variaveis.css">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Host+Grotesk:ital,wght@0,300..800;1,300..800&display=swap" rel="stylesheet">
{% endblock %}

{% extends "layout.html" %}
{% block conteudo %}
<script src="/static/js/inventario.js" defer></script>

<section class="container-geral-inventario">
```

```
    <section class="container-inventario">
        <div class="title">Inventário de Produtos</div>

        <div class="div-pesquisa">
            <input     type="text"     id="pesquisa"     oninput="pesquisar()"
placeholder="Pesquise aqui...">
            </div>

        <table class="tabela-inventario">
            <thead>
                <tr>
                    <th>Código</th>
                    <th>Descrição</th>
                    <th>Modelo</th>
                    <th>Fabricante</th>
                    <th>Número do Lote</th>
                    <th>Endereçamento</th>
                    <th>Saldo</th> <!-- Nova coluna para o saldo -->
                    {% if "professor_logado" in session %}
                    <th>Turma</th>
                    {% endif %}
                </tr>
            </thead>
            <tbody>
                {% for produto in lista_produtos %}
                <tr>
                    <td>{{ produto['codigo'] }}</td>
                    <td>{{ produto['descricao'] }}</td>
                    <td>{{ produto['modelo'] }}</td>
                    <td>{{ produto['fabricante'] }}</td>
                    <td>{{ produto['numero_lote'] }}</td>
                    <td>{{ produto['enderecamento'] }}</td>
                    <td>{{ produto['quantidade'] }}</td> <!-- Exibe o saldo do
produto -->

                    {% if "professor_logado" in session %}
                    <td>
                        <!-- Formulário com opção de escolha da turma -->
                        <form    action="{{    url_for('excluir_produto')    }}"
method="POST">
                            <input   type="hidden"   name="cod_prod"   value="{{
produto['codigo'] }}">

                            <!-- Dropdown para selecionar a turma -->
                            <select name="turma">
                                {% for nomeBD in lista_nomes %}
                                <option
value="{{nomeBD['database']}}">{{nomeBD['database']}}</option>
                                {% endfor %}
                            </select>
```

```
                            <button        type="submit"        onclick="return
confirm('Deseja  realmente  excluir  este  produto?')"><i  class="fa-solid  fa-
trash"></i>
                            </button>
                        </form>
                    </td>
                    {% endif %}
                </tr>
                {% endfor %}
            </tbody>
        </table>

    </section>

</section>




{% endblock %}
```

```
function pesquisar() {
    var input, filtro, tabela, linhas, textoLinha;

    input = document.getElementById('pesquisa');
    filtro = removerAcentos(input.value.toUpperCase());

    tabela = document.querySelector(".tabela-inventario tbody");
    linhas = tabela.querySelectorAll('tr');

    linhas.forEach(linha => {
        textoLinha = Array.from(linha.querySelectorAll('td'))
        .map(coluna => removerAcentos(coluna.textContent.toUpperCase()))
        .join(" ");

        if (textoLinha.indexOf(filtro) > -1) {
            linha.style.display = "";
        } else {
            linha.style.display = "none";
        }
    });
}

function removerAcentos(texto) {
    return texto.normalize("NFD").replace(/[\u0300-\u036f]/g, "");
}
```

```
# roteamento da página inventário
@app.route('/inventario')
def inventario():
```

```
    if "usuario_logado" in session or "professor_logado" in session:
        if request.method == "GET":
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            turma = session['usuario_logado']['turma'] if "usuario_logado" in
session else session['professor_logado']['turma']

            # Query para listar os produtos do inventário
            produtos = f"SELECT cod_prod, descricao_tecnica, modelo, fabricante,
num_lote, enderecamento, quantidade FROM {turma}.tb_cadastramento"
            mycursor.execute(produtos)
            resultado = mycursor.fetchall()

            lista_produtos = []
            for produto in resultado:
                lista_produtos.append({
                    "codigo": produto[0],
                    "descricao": produto[1],
                    "modelo": produto[2],
                    "fabricante": produto[3],
                    "numero_lote": produto[4],
                    "enderecamento": produto[5],
                    "quantidade": produto[6]
                })

            # Obtém a lista de bancos de dados, incluindo o do professor
            mycursor.execute("SELECT * FROM databaseprofessor.tb_database")
            resultado = mycursor.fetchall()

            # Adiciona a turma do professor à lista de opções se ele estiver
logado
            if "professor_logado" in session:
                resultado.append((session['professor_logado']['turma'],))

            mydb.close()

            # Cria a lista de opções para o select
            lista_nomes = [{"database": nomeBD[0]} for nomeBD in resultado]

            return                          render_template("inventario.html",
lista_produtos=lista_produtos, lista_nomes=lista_nomes)
```