Técnico em Desenvolvimento de Sistemas

Sistema de controle de reserva de laboratórios

Ariel Barbosa Santos

Orientador: Prof. Márcio Carmona Costa

Itumbiara – Goiás 2025

Resumo

Este trabalho apresenta o desenvolvimento de um sistema de controle de laboratórios, denominado LabFlow, utilizando a linguagem de programação Python e o framework Django. O sistema visa otimizar a gestão de equipamentos e recursos laboratoriais, proporcionando uma interface amigável e funcionalidades eficientes para os usuários. Sistema feito utilizando a linguagem Python e o framework Django, com banco de dados SQLite.

1. Introdução

A gestão de laboratórios em instituições de ensino é uma necessidade recorrente, frequentemente marcada por conflitos de horários e uso inadequado dos espaços. Para enfrentar esse problema, é fundamental compreender a realidade da instituição e propor soluções eficientes. Este projeto propõe o desenvolvimento de um sistema de controle de laboratórios, que facilite o agendamento e o uso dos ambientes por professores e alunos, promovendo organização e melhor aproveitamento dos recursos. A pesquisa foi realizada com base em um estudo de caso na Escola Senai de Itumbiara – GO, envolvendo laboratórios de mecânica, informática e química. Um questionário com 10 perguntas foi aplicado a usuários desses espaços, fornecendo dados essenciais para o desenvolvimento da aplicação. Embora focado em laboratórios, o conceito do sistema pode ser adaptado para outros ambientes, como salas de reuniões, bibliotecas e auditórios.

A gestão eficiente de laboratórios é essencial para o bom funcionamento de instituições educacionais e de pesquisa. Este trabalho propõe o desenvolvimento de um sistema web, o LabFlow, para auxiliar na administração de equipamentos e recursos laboratoriais.

2. Fundamentação Teórica

2.1 Desenvolvimento Web com Python e Django

Python é uma linguagem de programação de alto nível, conhecida por sua simplicidade e legibilidade. O Django é um framework web de alto nível que incentiva o desenvolvimento rápido e limpo de aplicações web, seguindo o padrão Model-View-Template (MVT). Ele oferece uma série de ferramentas integradas, como ORM (Object-Relational Mapping), sistema de autenticação e painel administrativo, facilitando o desenvolvimento de sistemas robustos.

Do ponto de vista computacional, sistemas de reserva de laboratórios visam uma maneira de solucionar o problema de conflitos e com isso, maximizar a utilização de espaços.

Existem exemplos práticos de sistemas de reserva que podem ser vistos como salas de conferência, hotéis ou até outros ambientes acadêmicos.

2.2 Sistemas de Gerenciamento de Laboratórios

Sistemas de gerenciamento de laboratórios (LIMS - Laboratory Information Management Systems) são utilizados para organizar e controlar as atividades dentro de laboratórios. Eles permitem o agendamento de equipamentos, controle de estoque, rastreamento de amostras e geração de relatórios.

A implementação de um LIMS pode melhorar significativamente a eficiência operacional e a conformidade com normas e regulamentos. Com isto em vista o sistema LabFlow será um exemplo simplificado de um LIMS.

3. Metodologia

3.1 Levantamento de Requisitos

O processo iniciou-se com a identificação das necessidades dos usuários, incluindo professores, técnicos e alunos. Foram realizadas entrevistas e questionários para entender os principais desafios enfrentados na gestão dos laboratórios.

3.2 Arquitetura do Sistema

O LabFlow foi desenvolvido seguindo a arquitetura MVT do Django. O modelo de dados foi definido utilizando o ORM do Django, permitindo a criação de tabelas no banco de dados de forma automatizada. As views foram responsáveis por processar as requisições e retornar as respostas apropriadas, enquanto os templates definiram a interface do usuário.

3.3 Tecnologias Utilizadas

Linguagem de Programação: Python

Framework Web: Django

Banco de Dados: SQLite

Front-end: HTML, CSS e JavaScript

Controle de Versão: Git e GitHub

3.4 Desenvolvimento e Testes

O desenvolvimento seguiu uma abordagem incremental, com testes realizados a cada nova funcionalidade implementada. Foram utilizados testes unitários para verificar o funcionamento correto das funções e testes de integração para garantir a comunicação adequada entre os componentes do sistema.

Além dos testes unitários e de integração, foram implementados testes automatizados utilizando o módulo unittest do Django, que permite verificar o comportamento das views e modelos de forma isolada. Esses testes garantem que as funcionalidades, como a criação de reservas e a validação de horários, funcionem conforme o esperado. A utilização do TestCase do Django facilitou a criação de cenários de teste com dados controlados, permitindo simular diferentes situações e verificar as respostas do sistema.

Para os testes de integração, foi utilizado o Client de testes do Django, que simula requisições HTTP ao sistema. Com ele, foi possível testar o fluxo completo de funcionalidades, como o processo de login de usuários, a navegação entre páginas e a realização de reservas. Esses testes asseguram que os componentes do sistema interagem corretamente entre si e que o usuário final terá uma experiência consistente e livre de erros. A abordagem de testes automatizados contribuiu significativamente para a robustez e confiabilidade do sistema desenvolvido.

3.5 Pesquisa de campo

Foi feita uma análise da problemática enfrentada por diversas instituições de ensino, em relação ao controle de uso de laboratórios e outros ambientes de estudo, como salas de reuniões e bibliotecas. O foco principal é apresentar uma solução para a otimização do gerenciamento de reservas desses espaços, com ênfase na redução de conflitos de horários e na melhoria da experiência de alunos e professores.

Após a coleta de dados sobre a realidade da Escola Senai de Itumbiara – GO, por meio de um questionário direcionado aos funcionários que utilizam esses ambientes foi possível compreender as dificuldades atuais e definir as bases para o desenvolvimento de um Sistema de Controle de Laboratórios, visando uma gestão eficiente desses espaços.

Os dados foram analisados de forma quantitativa e qualitativa ou seja, levando em conta a quantidade de dificuldades apresentadas e a natureza de cada demanda.

4. Modelagem do sistema

4.1 Diagrama de Caso de Uso (UML)

A elaboração do diagrama de Caso de Uso (UML) foi crucial para visualizar a interação entre os usuários e o sistema. Este artefato permitiu identificar os atores principais, como professores, alunos e administradores, e suas respectivas ações dentro do sistema. Através dos casos de uso, foi possível detalhar os processos de reserva, cancelamento, gerenciamento e consulta de disponibilidade dos laboratórios. Essa representação gráfica facilitou a compreensão dos requisitos funcionais e a definição das regras de negócio, garantindo que o sistema atendesse às necessidades específicas da Escola Senai de Itumbiara.

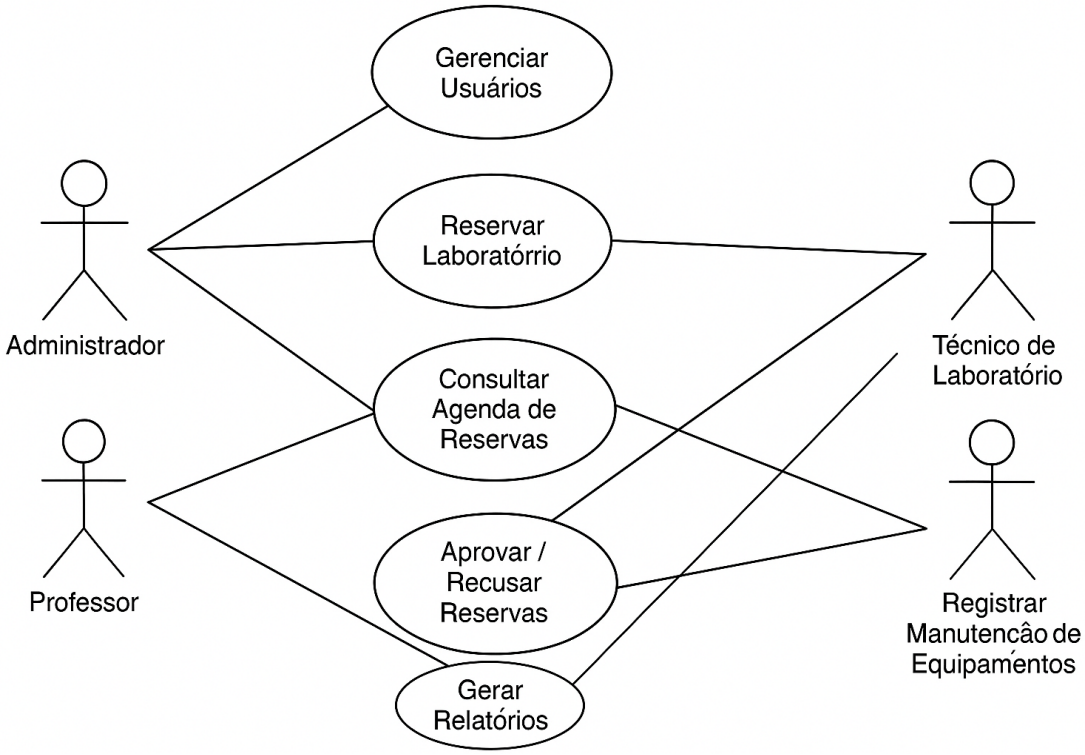

Figura 2 – Diagrama de Caso de Uso (UML)

Além do diagrama de Caso de Uso, outras ferramentas e técnicas foram utilizadas para aprofundar a análise e o design do sistema. A criação de protótipos de interface, por exemplo, permitiu simular a experiência do usuário e validar a usabilidade do sistema antes do desenvolvimento. A modelagem de dados foi essencial para definir a estrutura do banco de dados e garantir a integridade das informações. A aplicação de testes de software, tanto unitários quanto de integração, foi fundamental para assegurar a qualidade e a confiabilidade do sistema. Essas etapas complementares foram essenciais para garantir que o sistema de controle de laboratórios fosse robusto, eficiente e adequado às necessidades da instituição.

4.2 Modelo Relacional

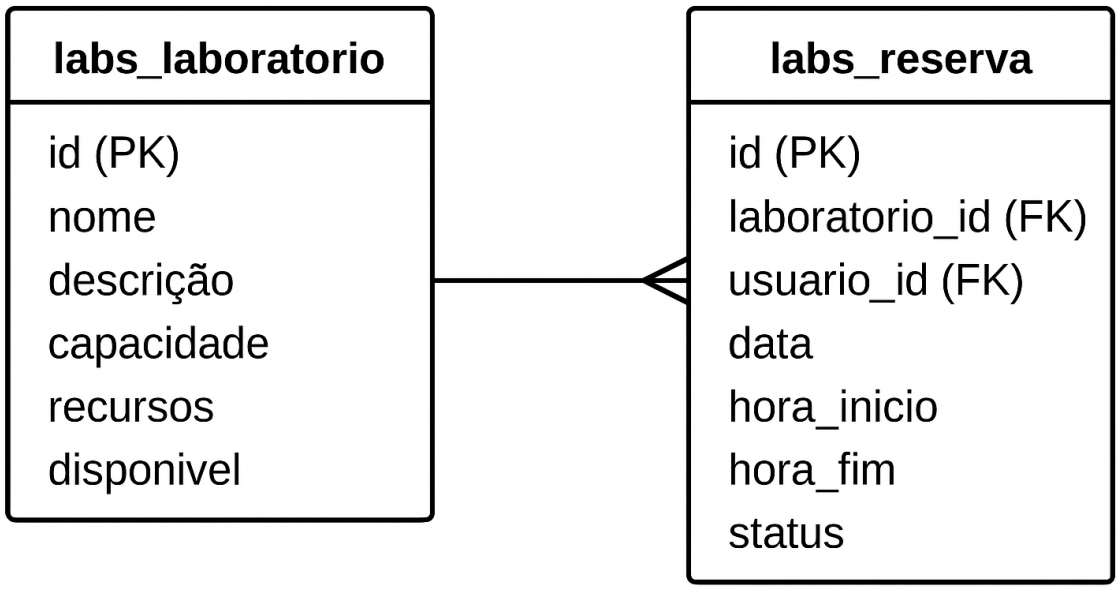

Figura 1 – Modelo de Entidade Relacional (MER)

4.3 Script SQL

```
-- Tabela de Laboratórios
CREATE TABLE "labs_laboratorio" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT,
    "nome" VARCHAR(100) NOT NULL,
    "descricao" TEXT NOT NULL,
    "capacidade" INTEGER NOT NULL CHECK("capacidade" >= 0),
    "recursos" TEXT,
    "disponivel" BOOLEAN NOT NULL DEFAULT 1
);

-- Tabela de Reservas
CREATE TABLE "labs_reserva" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT,
    "laboratorio_id" INTEGER NOT NULL,
    "usuario_id" INTEGER NOT NULL,
    "data" DATE NOT NULL,
    "hora_inicio" TIME NOT NULL,
    "hora_fim" TIME NOT NULL,
    "status" VARCHAR(20) NOT NULL DEFAULT 'indisponivel',
    FOREIGN KEY("laboratorio_id") REFERENCES "labs_laboratorio"("id") ON DELETE CASCADE,
    FOREIGN KEY("usuario_id") REFERENCES "auth_user"("id") ON DELETE CASCADE
);
-- Índices automáticos para chaves estrangeiras (opcional, dependendo do banco)
CREATE INDEX "labs_reserva_laboratorio_id_idx" ON "labs_reserva" ("laboratorio_id");
CREATE INDEX "labs_reserva_usuario_id_idx" ON "labs_reserva" ("usuario_id");
```

4.4 Observações importantes

Este script assume que você está usando o modelo de usuário padrão do Django (auth_user). Se estiver usando um modelo de usuário customizado, substitua "auth_user" pelo nome correto da tabela.

O campo status_atual não é persistido no banco de dados, pois é apenas uma propriedade Python.

Esse script é compatível com SQLite e PostgreSQL com pequenas alterações (ex: AUTOINCREMENT → SERIAL).

5. Requisitos

Os requisitos funcionais descrevem as funcionalidades que o sistema deve ter para atender às necessidades dos usuários.

ID	Descrição	Prioridade
RF01	Permitir que os usuários (professores e funcionários) reservem laboratórios online.	Alta
RF02	Exibir a disponibilidade dos laboratórios em um calendário.	Alta
RF03	Permitir a reserva de laboratórios por data, hora e tipo de laboratório.	Alta
RF04	Enviar notificações por e-mail ou SMS para confirmar as reservas.	Média
RF05	Permitir que os administradores gerenciem os horários e a disponibilidade dos laboratórios.	Alta
RF06	Gerar relatórios de uso dos laboratórios.	Média
RF07	Permitir o cancelamento de reservas.	Média
RF08	Exibir o histórico de reservas de cada usuário.	Baixa
RF09	Permitir a reserva de equipamentos específicos dentro dos laboratórios.	Média
RF10	Integrar o sistema com o sistema de autenticação da escola.	Alta

5.1 Requisitos Não Funcionais

Os requisitos não funcionais descrevem as características do sistema, como desempenho, segurança e usabilidade.

ID	Descrição	Prioridade
RNF01	O sistema deve ser responsivo e acessível em diferentes dispositivos (computadores, tabletes e smartphones).	Alta
RNF02	O sistema deve ter uma interface intuitiva e fácil de usar.	Alta
RNF03	O sistema deve ser seguro e proteger os dados dos usuários.	Alta
RNF04	O sistema deve ter um tempo de resposta rápido.	Média
RNF05	O sistema deve ser escalável para suportar um grande número de usuários e reservas.	Média
RNF06	O sistema deve ser compatível com os principais navegadores da web.	Alta
RNF07	O sistema deve ter um design moderno e agradável.	Baixa
RNF08	O sistema deve ser fácil de manter e atualizar.	Média
RNF09	O sistema deve estar disponível 24 horas por dia, 7 dias por semana.	Alta
RNF10	O sistema deve ter um manual do usuário e suporte técnico.	Média

6. Tecnologias Utilizadas

- **Linguagem de Programação:** Python 3
- **Framework Web:** Django 3
- **Banco de Dados:** SQLite
- **Frontend:** HTML5, CSS3
- **Controle de Versão:** Git
- **Hospedagem do Código:** GitHub

6.1 Estrutura de pastas do Projeto

A estrutura do projeto está organizada da seguinte forma:

```
├── labcontrol/        # Diretório principal do projeto Django
│   ├── __init__.py
│   ├── settings.py    # Configurações do projeto
│   ├── urls.py        # Rotas principais
│   └── wsgi.py
├── labs/              # Aplicativo Django responsável pelas funcionalidades
│   ├── migrations/
│   ├── templates/     # Templates HTML
│   ├── static/        # Arquivos estáticos (CSS, JS)
│   ├── admin.py       # Configurações do admin
│   ├── apps.py
│   ├── models.py      # Modelos de dados
│   ├── tests.py
│   └── views.py       # Lógicas das views
├── db.sqlite3         # Banco de dados SQLite
├── manage.py          # Script de gerenciamento do Django
└── README.md          # Documentação do projeto
```

7. Funcionalidades Implementadas

- **Cadastro de Laboratórios:** Permite o registro de novos laboratórios no sistema.
- **Cadastro de Equipamentos:** Permite o registro de equipamentos associados a um laboratório.
- **Reserva de Laboratórios:** Usuários podem realizar reservas para utilização dos laboratórios.
- **Listagem e Detalhamento:** Visualização de laboratórios, equipamentos e reservas.
- **Autenticação de Usuários:** Controle de acesso ao sistema mediante login e senha.

7.1 Interface do Usuário

A interface do sistema foi desenvolvida utilizando HTML5 e CSS3, proporcionando uma experiência amigável e responsiva para os usuários. As páginas principais incluem:

- **Página Inicial:** Apresenta uma visão geral dos laboratórios disponíveis.
- **Página de Cadastro:** Formulários para cadastro de laboratórios e equipamentos.
- **Página de Reserva:** Formulário para realização de reservas.
- **Página de Listagem:** Exibe listas de laboratórios, equipamentos e reservas.

Figura 3 – Tela de início ao iniciar o sistema

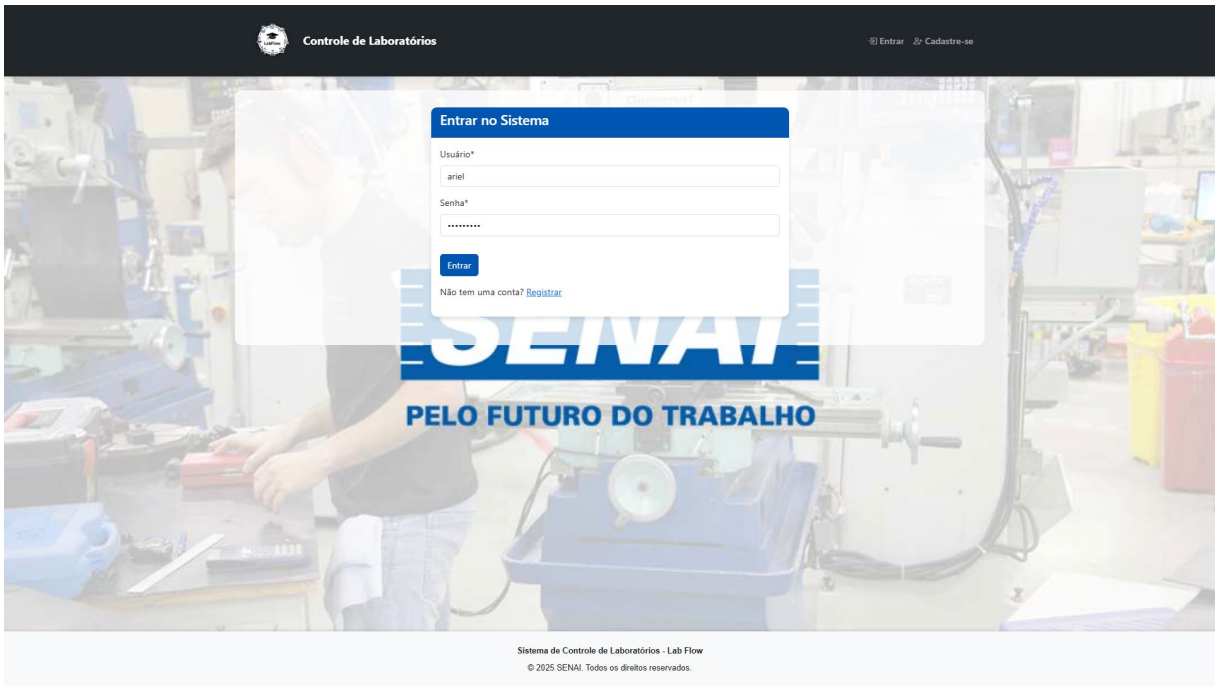

Figura 4 – Tela de Login

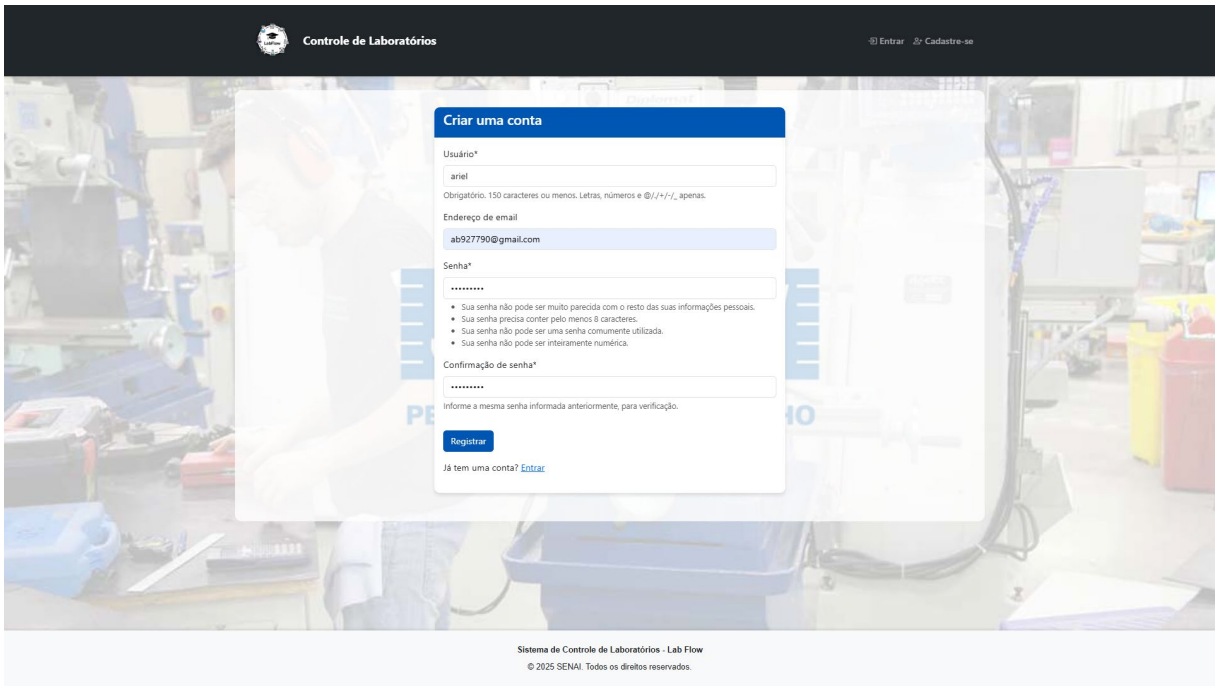

Figura 5 – Tela de cadastro de usuário

Figura 6 – Tela inicial após login efetuado

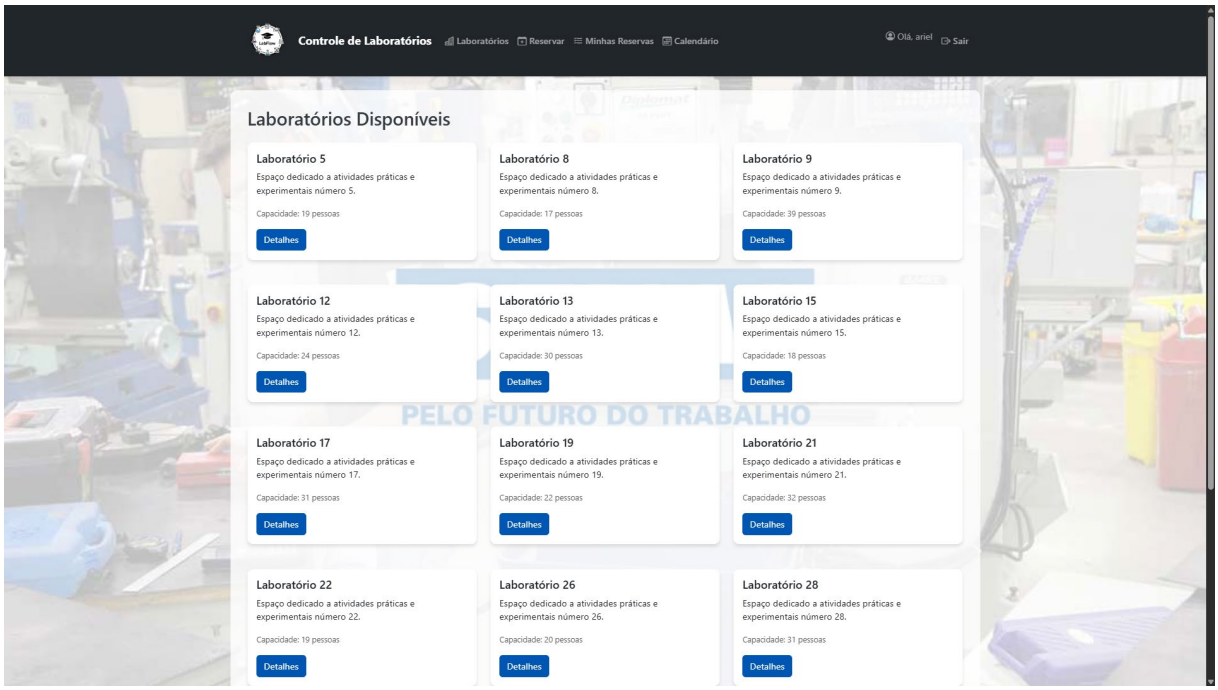

Figura 7 – Tela de exibição dos laboratórios

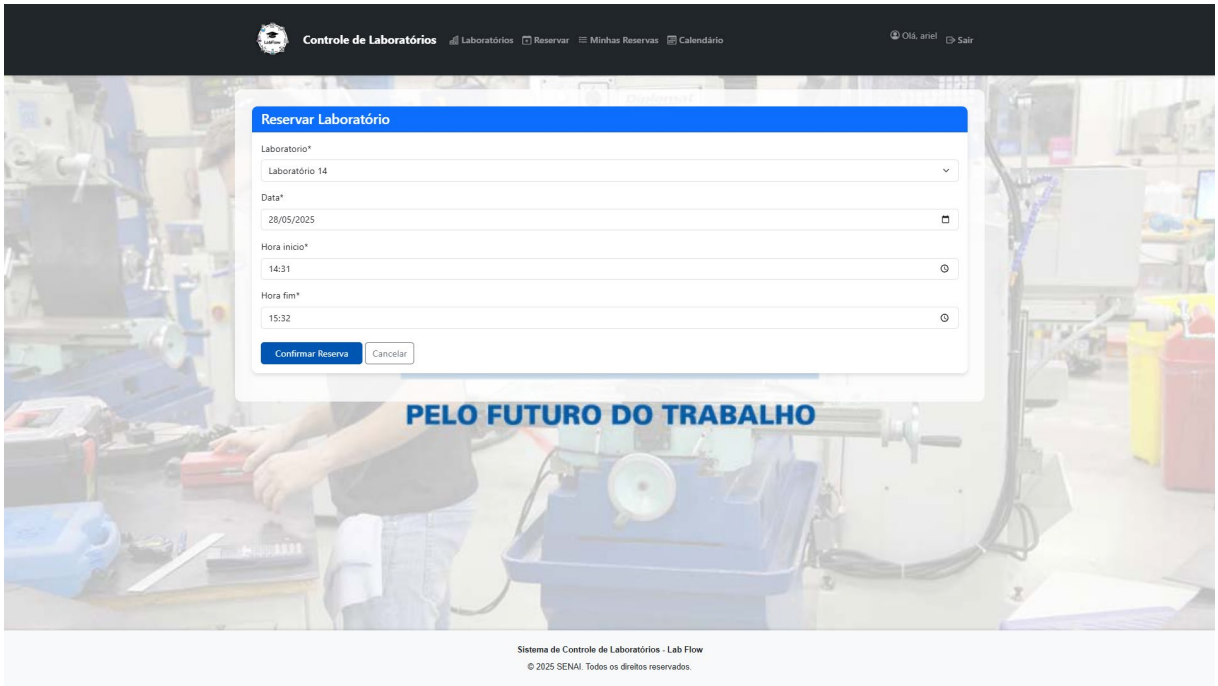

Figura 8 – Tela para efetuar a reserva do laboratório

Figura 9 – Tela de exibição das reservas feitas

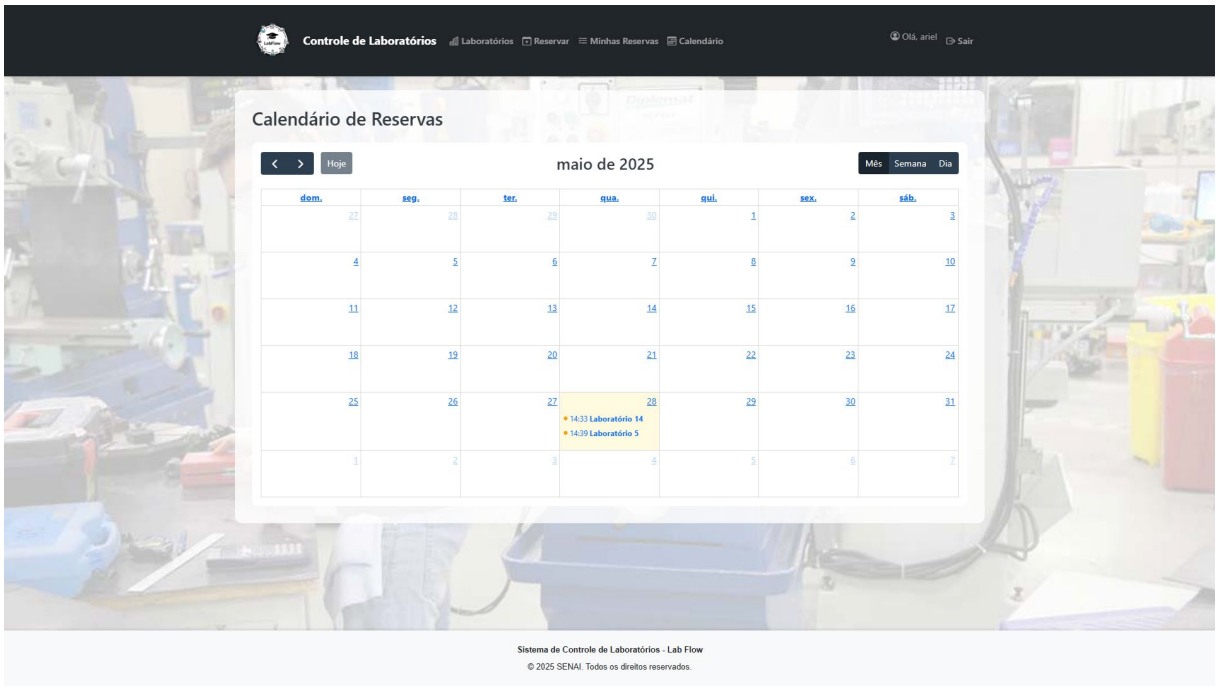

Figura 10 – Calendário que exibe quais dias há reservas feitas

7.2 Testes Realizados

Foram realizados testes manuais para verificar o correto funcionamento das funcionalidades implementadas, incluindo:

- Cadastro de laboratórios e equipamentos.
- Realização de reservas.
- Autenticação de usuários.
- Navegação entre as páginas do sistema.

8. Considerações Finais

O desenvolvimento do sistema LabFlow proporcionou uma experiência enriquecedora, tanto em termos técnicos quanto acadêmicos. A proposta de criar uma plataforma funcional e intuitiva para o controle de reservas de laboratórios foi plenamente atendida, com base nas necessidades observadas na Escola Senai de Itumbiara – GO. O sistema foi construído com tecnologias modernas, como Django, SQLite, HTML, CSS e JavaScript, e demonstrou-se eficaz na organização dos agendamentos, na prevenção de conflitos e na simplificação da gestão por parte dos responsáveis. Durante o processo de desenvolvimento, foi possível aplicar conceitos de modelagem de dados, autenticação, interface responsiva e persistência de informações. A estrutura de permissões implementada também assegura que apenas usuários autorizados possam efetuar ou gerenciar reservas, contribuindo para a segurança da aplicação.

Contudo, todo sistema está em constante evolução, e o **LabFlow** não é diferente. Algumas melhorias futuras são previstas para ampliar sua funcionalidade e robustez:

- **Integração com APIs externas**, como sistemas de autenticação da escola (Google Workspace ou Microsoft Azure AD) ou serviços de calendário como o Google Calendar, para sincronização dos eventos de reserva.
- **Melhorias no design responsivo**, garantindo acessibilidade total em dispositivos móveis e telas menores, o que facilitaria o uso por professores e alunos diretamente do celular.
- **Implementação de logs de auditoria**, possibilitando o registro de ações realizadas por cada usuário (criação, edição e exclusão de reservas), reforçando a transparência e a rastreabilidade do sistema.
- **Painel de estatísticas** para a equipe administrativa, com relatórios de uso dos laboratórios, horários mais utilizados, e frequência de reservas por curso.
- **Sistema de notificações por e-mail**, alertando os usuários sobre reservas futuras, alterações ou cancelamentos.
- **Internacionalização e localização da aplicação**, permitindo o uso do sistema por instituições que desejem adaptar o idioma e o formato de data/hora conforme sua localidade.

Essas perspectivas de aprimoramento demonstram que o **LabFlow** é um sistema escalável e preparado para futuras adaptações, podendo ser ampliado e replicado em outras unidades educacionais ou instituições com demandas semelhantes.

Referências Bibliográficas

1. **Python Software Foundation.** *The Python Language Reference, Release 3.13.3*. Disponível em: https://docs.python.org/pt-br/3/reference/index.html. Acesso em: 28 maio 2025.
2. **Django Software Foundation.** *The Django Project*. Disponível em: https://www.djangoproject.com/. Acesso em: 28 maio 2025.
3. **SQLite Consortium.** ***SQLite Documentation***. Disponível em: https://www.sqlite.org/docs.html Acesso em: 28 maio 2025.
4. **World Wide Web Consortium (W3C).** *HTML5 Specification*. Disponível em: https://www.w3.org/TR/html5/. Acesso em: 28 maio 2025.
5. **World Wide Web Consortium (W3C).** *Cascading Style Sheets (CSS) Specifications*. Disponível em: https://www.w3.org/Style/CSS/. Acesso em: 28 maio 2025.
6. **Normas ABNT.** Disponível em: https://www.normasabnt.org/referencias-de-sites-blogs-paginas-da-internet/?utm_source=chatgpt.com#google_vignette. https://www.w3.org/Style/CSS/. Acesso em: 28 maio 2025.