Projeto Orientado a Computação II
Estratégias de MLOps para o Flautim

Maria Luiza Leão Silva[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brasil

1. Introdução

Nos últimos anos, o Aprendizado de Máquina (ML) revolucionou a análise de dados e a tomada de decisões, tornando-se uma ferramenta essencial em diversas aplicações cotidianas. O pipeline de ML é um processo estruturado que abrange a coleta e pré-processamento de dados, o treinamento de modelos seguido por sua validação para avaliar a capacidade de generalização e, se aprovado, a implantação em produção. No entanto, essa etapa é apenas o começo. Modelos de aprendizado de máquina demandam atualização, monitoramento e manutenção contínuos, o que gera desafios operacionais que vão além do desenvolvimento inicial.

Para mitigar esses desafios, surgiu o conceito de Machine Learning Operations (MLOps)[3]. MLOps adapta os princípios do DevOps — tradicionalmente aplicados ao desenvolvimento de software — ao contexto do aprendizado de máquina, promovendo ciclos iterativos, rápidos e contínuos para a implantação e manutenção de modelos em produção[4].

Com o avanço dessas demandas, a plataforma Flautim foi concebida como uma solução para a experimentação e prototipagem de modelos de aprendizado de máquina, com foco no Aprendizado Federado (FL), mas também com suporte ao aprendizado centralizado tradicional. Seu principal objetivo é viabilizar a condução de experimentos distribuídos de forma segura, escalável e com respeito à privacidade dos dados. Dessa forma, o Flautim oferece suporte técnico e operacional para que cientistas de dados, engenheiros de machine learning e desenvolvedores possam criar, testar e acompanhar seus modelos em um ambiente controlado.

Nesse contexto, o projeto visa investigar e desenvolver soluções baseadas em práticas de MLOps que possam ser integradas à plataforma Flautim, a fim de aprimorar o gerenciamento do ciclo de vida de experimentos de aprendizado de máquina. A proposta busca estabelecer mecanismos que tornem os processos de versionamento de dados e modelos, execução e reprodutibilidade de experimentos, e automação de pipelines mais estruturados, escaláveis e seguros. Ao alinhar os princípios do MLOps às particularidades do ambiente distribuído e federado da plataforma, o projeto busca proporcionar uma base sólida para experimentações eficientes, confiáveis e sustentáveis dentro do ecossistema do Flautim.

Durante a primeira fase do projeto (POC1), o foco foi a criação de uma API para gerenciamento automatizado de pipelines de aprendizado de máquina, incorporando ferramentas como Git, DVC e GitHub Actions para controle de versionamento, execução de experimentos e integração contínua. Foram implementadas funcionalidades fundamentais como a criação de projetos, definição de etapas de pipeline, rastreamento de dados e experimentos, além da geração de métricas e relatórios de desempenho. Essa infraestrutura inicial estabeleceu uma base para a condução de experimentos centralizados, alinhando-se às diretrizes do MLOps

Com base nessa fundação, o POC2 teve como objetivo desenvolver uma interface gráfica que utilizasse a API implementada anteriormente, permitindo que os recursos de gerenciamento de experimentos, versionamento e execução de pipelines fossem acessados por meio de um ambiente visual

acessível e intuitivo. Tudo isso com o intuito de ampliar a usabilidade da solução construída, reduzindo barreiras técnicas e tornando a plataforma Flautim mais acessível para diferentes perfis de usuários. Este relatório documenta todas as etapas do projeto, abordando desde a fundamentação teórica até os detalhes técnicos da implementação, incluindo o desenvolvimento da interface e sua integração com a API anteriormente desenvolvida.

2. Referencial Teórico

Esta seção apresenta os fundamentos teóricos que sustentam o desenvolvimento do projeto, abordando conceitos essenciais para a compreensão do problema e das soluções propostas. Serão discutidos os princípios do MLOps, que guiam a organização e automação do ciclo de vida de modelos de aprendizado de máquina , bem como a estrutura e os objetivos da plataforma Flautim, que serve como ambiente de experimentação para a aplicação das práticas investigadas. Também será detalhada a API desenvolvida no âmbito do POC1, projetada para automatizar operações relacionadas ao versionamento de dados e modelos, à execução de pipelines e ao rastreamento de experimentos. Esses elementos fornecem o embasamento necessário para justificar as decisões técnicas e metodológicas adotadas ao longo do projeto.

2.1. Machine Learning Operations (MLOps)

O MLOps herda diretamente os princípios de CI/CD (integração e entrega contínuas) do DevOps, adaptando-os para lidar com a complexidade do aprendizado de máquina, onde não apenas o código, mas também os dados e os modelos precisam ser gerenciados e versionados. No contexto de ML, os modelos são altamente dependentes dos dados, e estes estão sujeitos a mudanças frequentes. Isso pode afetar diretamente a precisão e a eficácia do modelo, tornando necessário o re-treinamento para se adequar às novas informações; gerenciar todos os diferentes experimentos e modelos ao longo do tempo complica a manutenção e a escalabilidade do modelo como um produto. Portanto, a implementação do MLOps visa criar um pipeline de desenvolvimento de ML que seja reproduzível, escalável e de fácil manutenção, alinhando as práticas de CD com as particularidades do aprendizado de máquina.

2.1.1. Componentes Fundamentais do MLOps

Gerenciamento de Dados: Os dados são o núcleo dos sistemas de ML. Para garantir a reprodutibilidade dos experimentos, o MLOps requer práticas robustas de versionamento e gerenciamento de dados, permitindo que qualquer mudança nos dados seja rastreada e testada em relação ao seu impacto nos modelos. Ferramentas de versionamento de dados são frequentemente usadas para facilitar esse processo.
Treinamento e Validação Contínuos: O pipeline de MLOps inclui processos de re-treinamento e validação contínuas dos modelos, garantindo que estes permaneçam precisos e confiáveis à medida que os dados evoluem. Essa prática exige pipelines automatizados para realizar o treinamento e a validação de novos modelos, semelhantes aos pipelines de testes em DevOps, mas adaptados para incluir métricas de desempenho do modelo[2].
Versionamento e Implantação de Modelos: Assim como o versionamento de código em DevOps, o MLOps introduz o versionamento de modelos[1], permitindo que modelos antigos e novos coexistam para comparações e rollback em caso de falhas ou piora na performance. O processo de implantação contínua para modelos, conhecido como Continuous Delivery for Machine Learning (CD4ML), é implementado para permitir que novos modelos sejam testados e gradualmente promovidos para produção, garantindo segurança e confiabilidade.

Monitoramento e Feedback Contínuo: Um componente importante do MLOps é o monitoramento contínuo do desempenho dos modelos em produção. Assim como no DevOps, onde o monitoramento de logs e métricas de sistema é essencial, no MLOps é crucial monitorar métricas de acurácia e detectar desvios de conceito (drift), que ocorrem quando os dados de produção diferem significativamente dos dados de treinamento, comprometendo a eficácia do modelo.

2.1.2. Importância do MLOps para o Ciclo de Vida de ML

A prática de MLOps é essencial para transformar o aprendizado de máquina em uma prática sustentável e escalável dentro das organizações [5]. Ela automatiza e padroniza o desenvolvimento de modelos, permitindo que as equipes de ciência de dados e operações colaborem de forma eficiente e ágil. Além disso, o MLOps permite que modelos de ML sejam mantidos e monitorados com a mesma disciplina e controle que o software tradicional, minimizando riscos de falhas e garantindo que os modelos possam ser ajustados rapidamente para refletir novas informações. Sendo assim, o MLOps proporciona ao aprendizado de máquina a capacidade de evoluir de maneira contínua e controlada, atendendo tanto às necessidades de inovação quanto de estabilidade em ambientes de produção.

2.2. Flautim

O Flautim foi desenvolvido como uma solução de aprendizado federado voltada para o setor automotivo, permitindo a realização de experimentos de forma distribuída e segura. A plataforma viabiliza o treinamento distribuído de modelos descentralizados, o que é essencial para aplicações em veículos autônomos, manutenção preditiva, personalização da experiência do motorista e análise de dados de telemetria. Embora o sistema foque no Aprendizado Federado, ele também possui suporte para o desenvolvimento de modelos tradicionais de ML.

2.2.1. Arquitetura da plataforma

Camada de Aplicação: Uma biblioteca modularizada com três componentes principais. O componente Dataset que representa os dados de treinamento e permite seu encapsulamento conforme necessário. A classe Model representa qualquer conjunto de parâmetros do projeto. Por último, a classe Experiment que define o ciclo treinamento-validação. Existem dois tipos de experimentos: o centralizado, que segue um fluxo comum de ML, e o descentralizado, que segue um fluxo de FL.
Camada Física: Composta por servidores de alto desempenho, armazenamento e GPUs, possibilitando o processamento intensivo de dados necessário para o aprendizado federado.
Camada Lógica: Integra Kubernetes para a orquestração de contêineres, o framework Flower para comunicação Cliente/Servidor, e a interface de usuário. Assim, garante que a plataforma seja escalável, segura e eficiente.

2.2.2. Interface Web do Flautim

A interface web da plataforma Flautim é desenvolvida utilizando NodeJS e Typescript, oferecendo aos usuários uma experiência altamente intuitiva e amigável. Ela permite aos usuários realizarem a autenticação, gerenciarem projetos e executarem experimentos diretamente na plataforma. As principais funcionalidades da interface web incluem:

Principais funcionalidades da interface web incluem:

Gerenciamento de Projetos: Visualização e controle de múltiplos projetos com informações detalhadas.
Execução e Monitoramento de Experimentos: Inserção de dados experimentais, acompanhamento em tempo real dos experimentos e visualização gráfica dos resultados obtidos.
Visualização de Logs e Modelos: Acesso aos logs detalhados de execução e arquitetura dos modelos utilizados.
Integração Completa: Comunicação direta com serviços como Kubernetes, garantindo execução robusta e escalável dos experimentos federados.

2.3. API desenvolvida no POC1

A API desenvolvida no POC1 foi implementada utilizando FastAPI, um framework conhecido por sua alta performance e facilidade de uso, em conjunto com Uvicorn, um servidor ASGI de alto desempenho. A escolha dessas tecnologias proporcionou uma API eficiente, escalável e com rápida capacidade de resposta, ideal para atender às necessidades do protótipo desenvolvido. Ela desempenha um papel central no gerenciamento e automação das operações relacionadas ao versionamento de dados e código usando Git e DVC, além da integração contínua com GitHub Actions e o sistema de implantação contínua Argo CD. Projetada especificamente para o ambiente de experimentação da plataforma Flautim, a API oferece:

Inicialização e Configuração Automática de Projetos: Criação de novos projetos com ambiente configurado automaticamente para uso imediato, incluindo Git e DVC.
Gerenciamento Flexível de Pipelines: Definição de etapas específicas, como pré-processamento, treinamento, validação, com suporte a configuração dinâmica através de arquivos .yaml.
Versionamento Completo: Rastreabilidade detalhada de alterações nos dados e modelos, garantindo reprodutibilidade e transparência
Execução e Monitoramento de Experimentos: Facilitação da execução de experimentos com diferentes parâmetros, comparação dos resultados e visualização das métricas de desempenho.

3. Metodologia

O projeto seguiu uma metodologia estruturada em três etapas bem definidas: Design e prototipação, Desenvolvimento front-end e Integração com a API.

3.1. Design da Interface

O design foi realizado utilizando a ferramenta Figma, permitindo uma colaboração ágil e eficiente entre design e desenvolvimento.

3.2. Desenvolvimento Front-end

O front-end foi desenvolvido utilizando React.js, TypeScript, Tailwind CSS e DaisyUI. Essas tecnologias foram escolhidas por sua robustez, capacidade de modularização, tipagem estática e facilidade de manutenção. O gerenciamento de estado foi realizado com React Query, otimizando o desempenho geral e simplificando a comunicação com o backend.

3.3. Integração com a API

A comunicação com a API foi desenvolvida utilizando Axios, garantindo tratamento robusto de erros e gerenciamento eficiente das requisições assíncronas. Esta integração possibilitou uma interface ágil e dinâmica para gestão dos pipelines e experimentos.

4. Resultados Obtidos

4.1. Interface Web Implementada

A interface desenvolvida para a plataforma Flautim é composta por diversas páginas e funcionalidades, projetadas para proporcionar uma experiência fluida, intuitiva e produtiva. A seguir, descrevemos detalhadamente as principais páginas implementadas e suas funcionalidades.

4.1.1. Página Inicial

A página inicial, Figura 1, serve como o ponto de entrada principal da plataforma, funcionando como um dashboard central que fornece uma visão geral dos projetos e atividades do usuário. O objetivo principal desta página é oferecer uma visão consolidada do estado atual dos projetos de machine learning, permitindo aos usuários identificar rapidamente projetos ativos, experimentos em andamento e métricas importantes de performance.

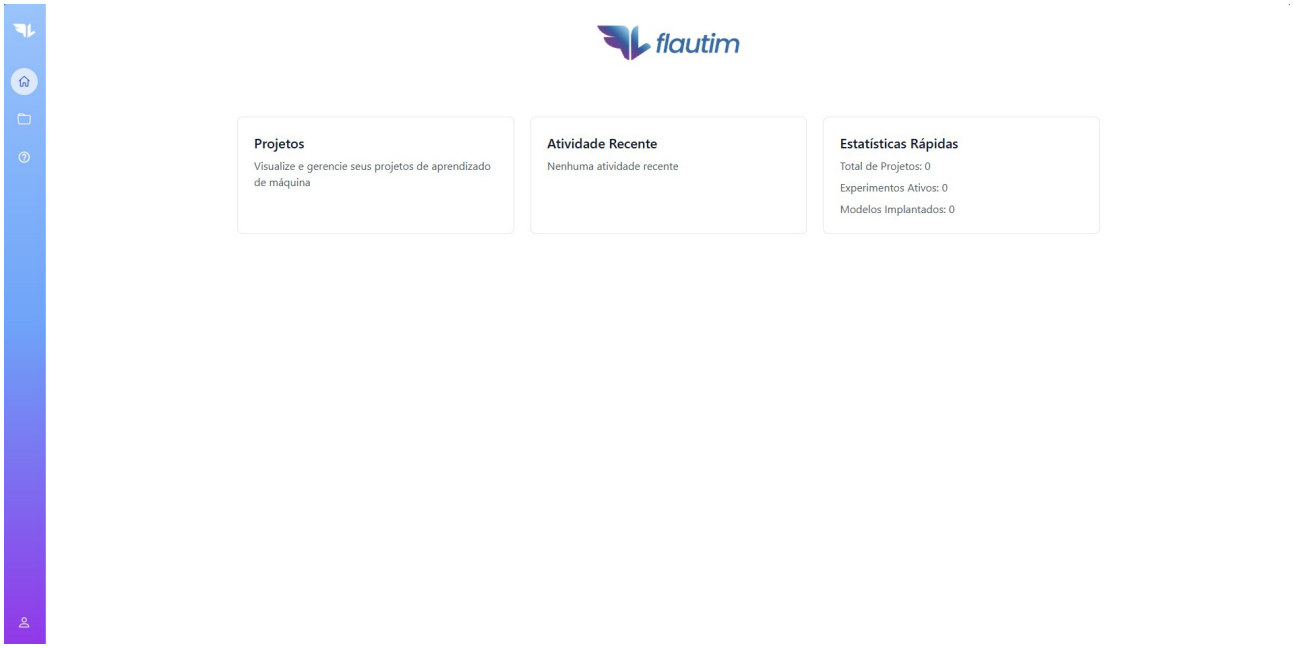

Figura 1.

4.1.2. Gerenciamento de Projetos

A página Projetos, Figura 2, representa o centro de gerenciamento de projetos da plataforma, permitindo aos usuários visualizar, criar e administrar todos os seus projetos de machine learning. O objetivo principal desta página é fornecer uma interface intuitiva para o ciclo de vida completo dos projetos, desde a criação inicial até o monitoramento contínuo de performance e status.

A funcionalidade de listagem de projetos implementa uma visualização em grid responsivo, onde cada projeto é representado por um card informativo contendo metadados essenciais como nome, descrição, tipo de projeto, framework utilizado e contadores de modelos e experimentos. Os cards de projeto são interativos, permitindo navegação direta para as páginas específicas de cada projeto através de cliques. O sistema implementa estados visuais para diferentes status de projeto, incluindo projetos ativos, arquivados e com problemas.

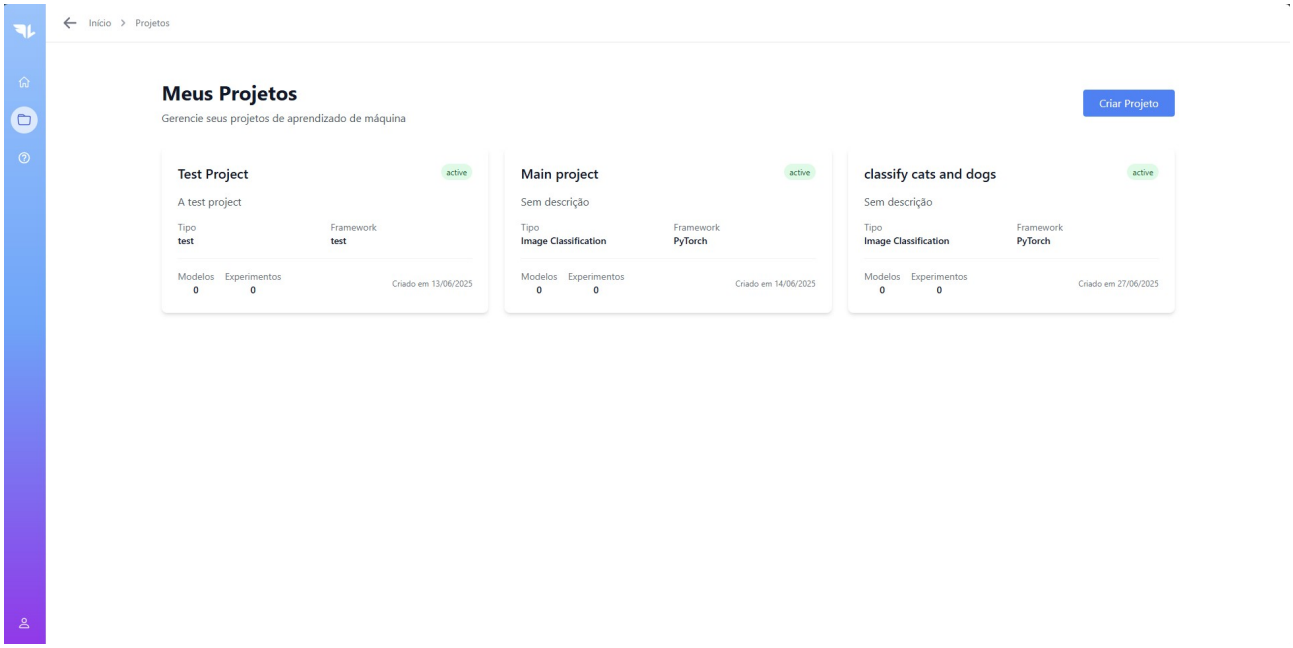

Figura 2.

A funcionalidade de criação de projetos é implementada através de um modal que permite aos usuários configurar todos os aspectos de um novo projeto. O formulário de criação inclui campos para nome do projeto, descrição detalhada, tipo de projeto (classificação, regressão, clustering, etc.), framework de machine learning (TensorFlow, PyTorch, Scikit-learn, etc.), versão do Python e lista de dependências.

4.1.3. Informações do Projeto

A página ”Informações do Projeto”, Figura 3, fornece uma visão detalhada e abrangente de um projeto específico, funcionando como um hub central para todas as operações relacionadas ao projeto selecionado. O objetivo principal desta página é consolidar informações críticas sobre o projeto e fornecer acesso rápido às funcionalidades mais importantes, permitindo aos usuários gerenciar eficientemente todos os aspectos de seus projetos de machine learning.

A funcionalidade de cabeçalho do projeto implementa uma seção informativa que exibe metadados essenciais como nome do projeto, descrição, data de criação e status atual. O cabeçalho inclui botões de ação rápida para compartilhamento do projeto e edição de configurações. A seção de estatísticas principais apresenta três cards informativos: o card de Modelos exibe o número total de modelos treinados, a melhor acurácia alcançada e a performance do modelo mais recente ; o card de Experimentos mostra o total de experimentos realizados e o tempo médio de treinamento ; e o card de Últimas Atividades lista as ações mais recentes realizadas no projeto.

A funcionalidade de configuração do projeto permite aos usuários visualizar e editar parâmetros técnicos como tipo de projeto, framework utilizado, versão do Python e dependências principais. O sistema implementa visualização hierárquica das dependências, com destaque para bibliotecas críticas e versões específicas. A seção de recursos fornece informações sobre utilização de armazenamento, horas de computação consumidas e uso de GPU, permitindo aos usuários monitorar o consumo de recursos e otimizar custos.

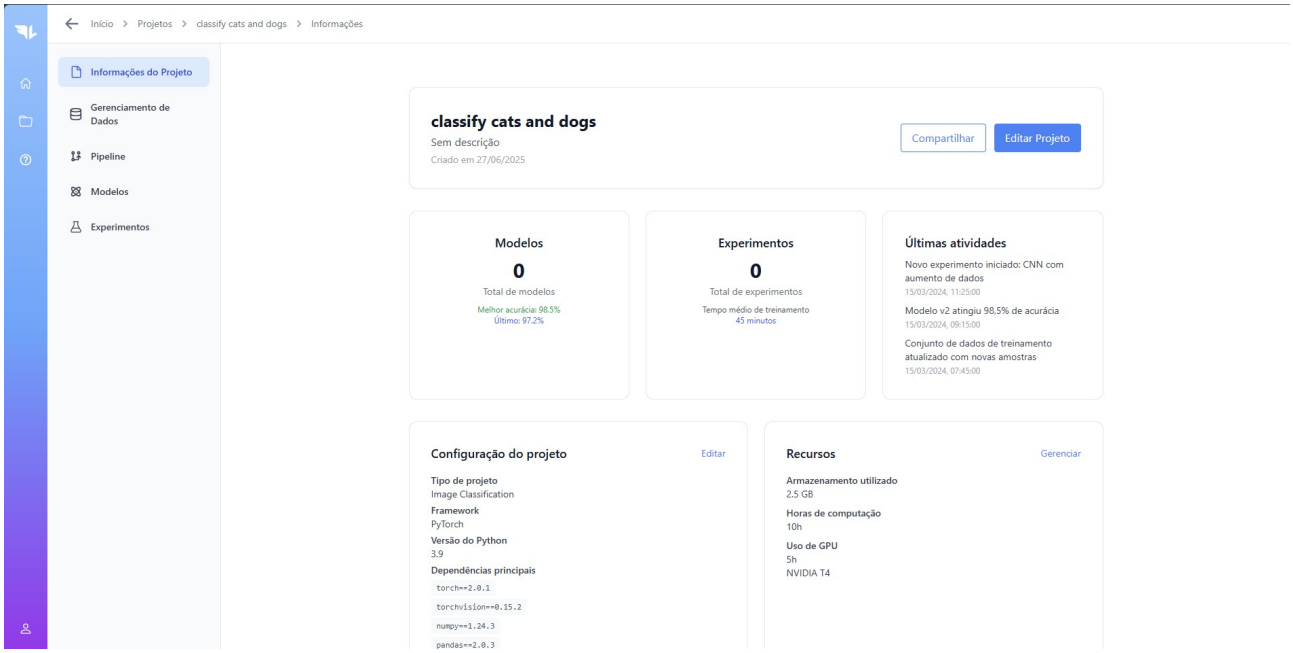

Figura 3. Página ”Informações do Projeto”de um projeto.

A página implementa navegação contextual através de uma barra lateral que fornece acesso rápido às principais funcionalidades do projeto, incluindo pipeline, gestão de dados, modelos e experimentos. O sistema de breadcrumbs mantém o contexto de navegação, permitindo aos usuários entender sua localização atual na hierarquia da aplicação. A funcionalidade de histórico de atividades fornece uma timeline detalhada de todas as ações realizadas no projeto, incluindo timestamps e descrições das operações. O tratamento de estados da página inclui loading states específicos para cada seção, permitindo carregamento progressivo do conteúdo. A página implementa tratamento robusto de erros, incluindo mensagens específicas para diferentes tipos de falha como projeto não encontrado, problemas de conectividade ou erros de autorização. A funcionalidade de atualização em tempo real permite que mudanças em outras partes da aplicação sejam refletidas automaticamente na página do projeto.

4.1.4. Sistema de Pipeline

A página ”Pipeline”, Figura 4, representa o centro de controle para gerenciamento de pipelines de machine learning, permitindo aos usuários definir, configurar, executar e monitorar pipelines complexos de processamento de dados e treinamento de modelos. O objetivo principal desta página é fornecer uma interface visual e intuitiva para o gerenciamento completo do ciclo de vida de pipelines, desde a definição de estágios até a análise de resultados de execução.

A funcionalidade de configuração de pipeline implementa um editor visual que permite a definição estágios de pipeline através de uma interface intuitiva. Cada estágio pode ser configurado com parâmetros específicos como dependências de entrada, arquivos de saída, parâmetros de configuração, métricas de monitoramento e comandos de execução. O sistema implementa validação automática de dependências entre estágios, garantindo que pipelines sejam executáveis e livres de ciclos. A funcionalidade de execução de pipeline permite aos usuários executar pipelines completos ou estágios individuais.

O sistema implementa monitoramento em tempo real da execução, incluindo progresso de cada

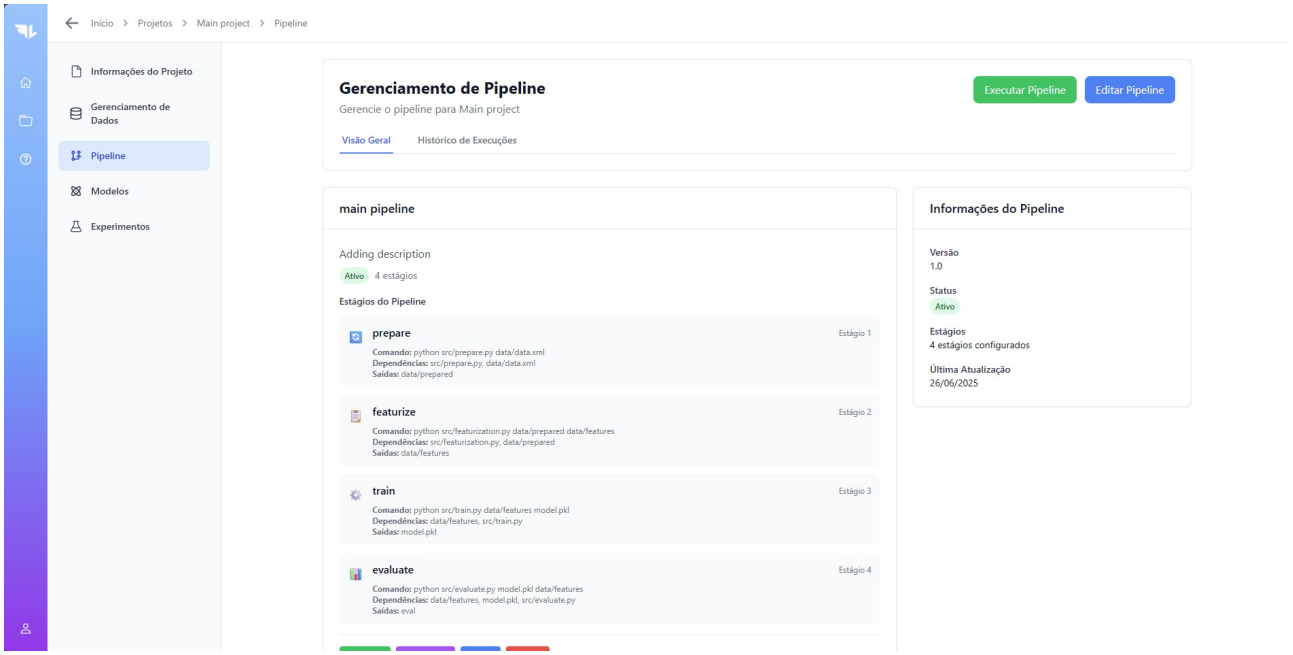

Figura 4. Página ”Pipeline”de um projeto.

estágio, logs de execução e métricas coletadas. A funcionalidade de recuperação de pipeline permite restaurar execuções interrompidas a partir do último ponto de sucesso. A página implementa duas abas principais: a aba ”Visão Geral”fornece uma visão geral do pipeline atual, incluindo diagrama visual dos estágios, status de cada componente e métricas de performance ; a aba ”Histórico de Execuções”exibe histórico completo de execuções, incluindo status, duração, parâmetros utilizados e resultados obtidos. O sistema implementa filtros avançados para busca de execuções específicas baseado em critérios como data, status ou parâmetros.

O tratamento de estados da página inclui indicadores visuais para diferentes status de pipeline (ativo, inativo, em execução, com erro) e feedback em tempo real sobre operações em andamento. A página implementa funcionalidades de exportação e importação de configurações de pipeline, facilitando o compartilhamento e backup de configurações complexas. O sistema também fornece análise de performance de pipeline, incluindo tempos de execução históricos e identificação de gargalos.

4.1.5. Gestão de Dados

A página ”Gerenciamento de Dados”, Figura 5, implementa funcionalidades avançadas para gerenciamento completo de dados em projetos de machine learning, incluindo upload de arquivos, configuração de fontes de dados remotas, versionamento de código e gerenciamento de parâmetros. O objetivo principal desta página é fornecer uma interface unificada para todas as operações relacionadas a dados, permitindo aos usuários gerenciar eficientemente datasets, código e configurações de projeto.

A funcionalidade de gerenciamento de fontes de dados permite aos usuários configurar e monitorar diferentes tipos de fontes de dados, incluindo URLs remotas, arquivos locais e conexões com sistemas de armazenamento em nuvem. O sistema suporta integração com Amazon S3, Google Cloud Storage, Azure Blob Storage e servidores SSH. Cada fonte de dados pode ser configurada com credenciais específicas, políticas de acesso e configurações de sincronização automática.

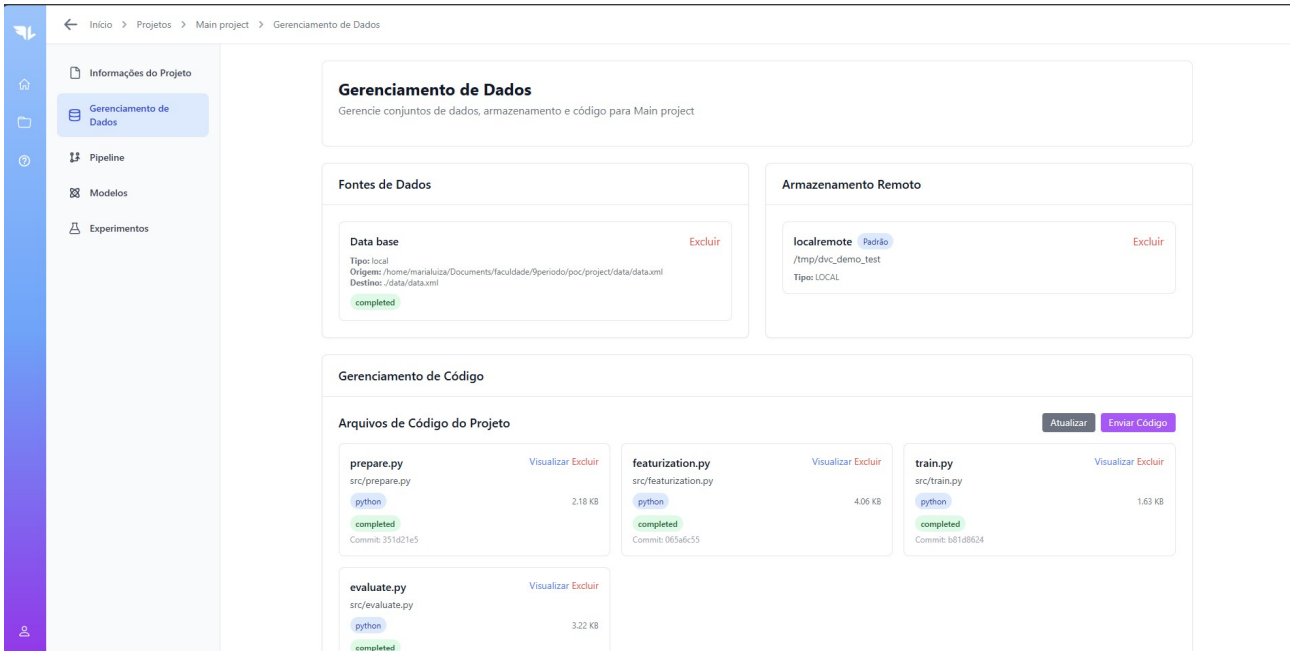

Figura 5. Página ”Gerenciamento de Dados”de um projeto.

A funcionalidade de upload de código implementa um sistema avançado para gerenciamento de arquivos de código de projeto, incluindo suporte a múltiplos formatos como Python, Jupyter notebooks, arquivos de configuração e documentação. O sistema implementa validação de código, extração automática de metadados e integração com controle de versão Git. A funcionalidade de visualização de código permite aos usuários visualizar conteúdo de arquivos diretamente na interface, com suporte a syntax highlighting e navegação por estrutura.

A funcionalidade de gerenciamento de parâmetros implementa um sistema completo para criação, edição e versionamento de conjuntos de parâmetros para experimentos e pipelines. O sistema suporta parâmetros de diferentes tipos (strings, números, booleanos, arrays, objetos) e implementa validação automática baseada em esquemas definidos. A funcionalidade de importação e exportação permite carregar parâmetros de arquivos externos (YAML, JSON, ENV) e exportar configurações para compartilhamento.

4.1.6. Sistema de Experimentos

A página ”Experimentos”, Figura 6, implementa funcionalidades completas para gerenciamento de experimentos de machine learning, permitindo aos usuários criar, executar, monitorar e comparar experimentos de forma sistemática e reproduzível. O objetivo principal desta página é fornecer uma interface intuitiva para o ciclo de vida completo de experimentos, desde a definição de parâmetros até a análise de resultados e comparação de performance.

A funcionalidade de criação de experimentos implementa um formulário avançado que permite aos usuários definir todos os aspectos de um novo experimento, incluindo nome, descrição e parâmetros de configuração. A funcionalidade de validação de parâmetros verifica automaticamente a consistência dos parâmetros definidos com o esquema do projeto. A funcionalidade de visualização de experimentos implementa uma interface rica para análise de resultados, incluindo listagem de experimentos com metadados detalhados, comparação de parâmetros entre experimentos e visualização

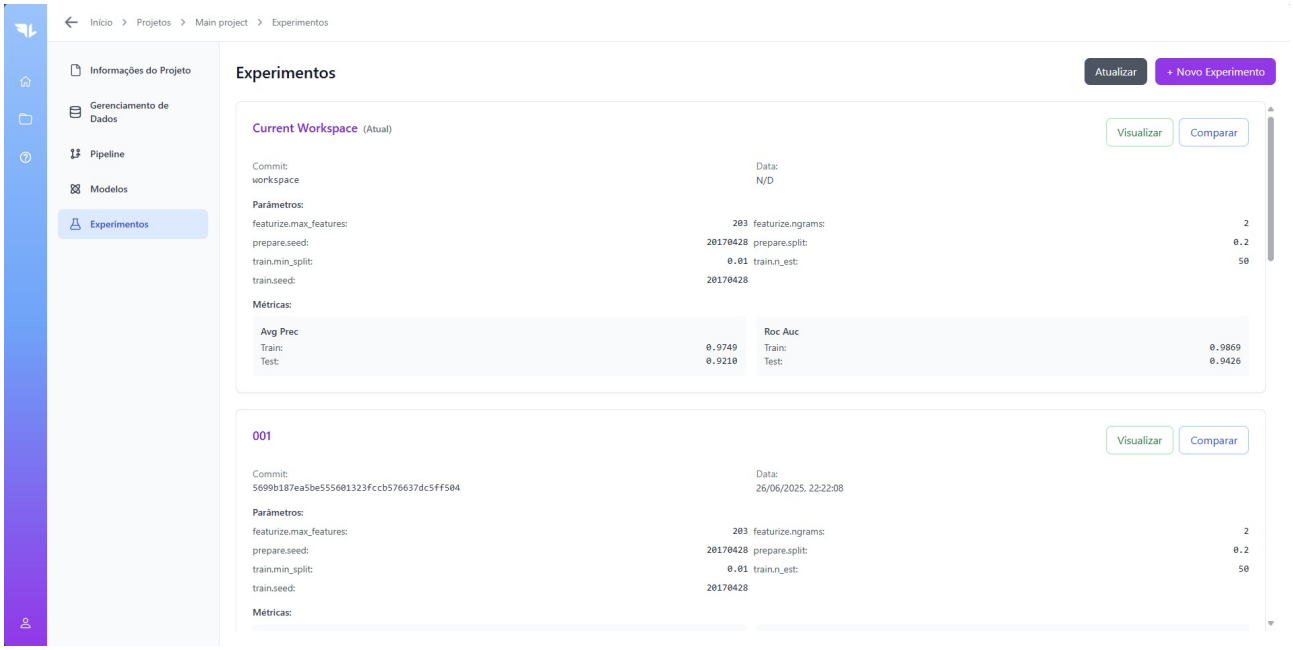

Figura 6. Página ”Experimentos”de um projeto.

de métricas de performance. O sistema suporta diferentes formatos de visualização, incluindo tabelas, gráficos e dashboards interativos.

A funcionalidade de comparação de experimentos implementa análise automática de diferenças entre experimentos, destacando variações em parâmetros e métricas de forma visual e intuitiva. O sistema fornece estatísticas comparativas incluindo melhorias de performance, mudanças significativas em parâmetros e identificação de tendências. A página implementa funcionalidades de versionamento de experimentos através de integração com DVC, permitindo controle completo de versão de parâmetros, código e dados utilizados em cada experimento. O sistema implementa operações de push e pull de experimentos para sincronização com repositórios remotos. A funcionalidade de tags e anotações permite aos usuários organizar e categorizar experimentos para facilitar busca e análise posterior.

4.1.7. Gerenciamento de Modelos

A página ”Modelos”, Figura 7, implementa funcionalidades completas para gerenciamento de modelos de machine learning, permitindo aos usuários versionar, avaliar, comparar e implantar modelos treinados de forma sistemática. O objetivo principal desta página é fornecer uma interface centralizada para o ciclo de vida completo de modelos.

A funcionalidade de gerenciamento de modelos implementa um sistema de versionamento robusto que permite aos usuários a organização e rastreamento de diferentes versões de modelos. O sistema suporta múltiplos formatos de modelo incluindo pickle, joblib, ONNX e modelos customizados de diferentes frameworks como TensorFlow, PyTorch, Scikit-learn e XGBoost. Cada modelo é associado com metadados estruturados incluindo tipo de modelo, framework, acurácia, data de criação e descrição detalhada. A funcionalidade de avaliação de modelos implementa um sistema automatizado para execução de métricas de performance em modelos treinados. O sistema suporta métricas padrão como acurácia, precisão, recall, F1-score e métricas específicas de domínio.

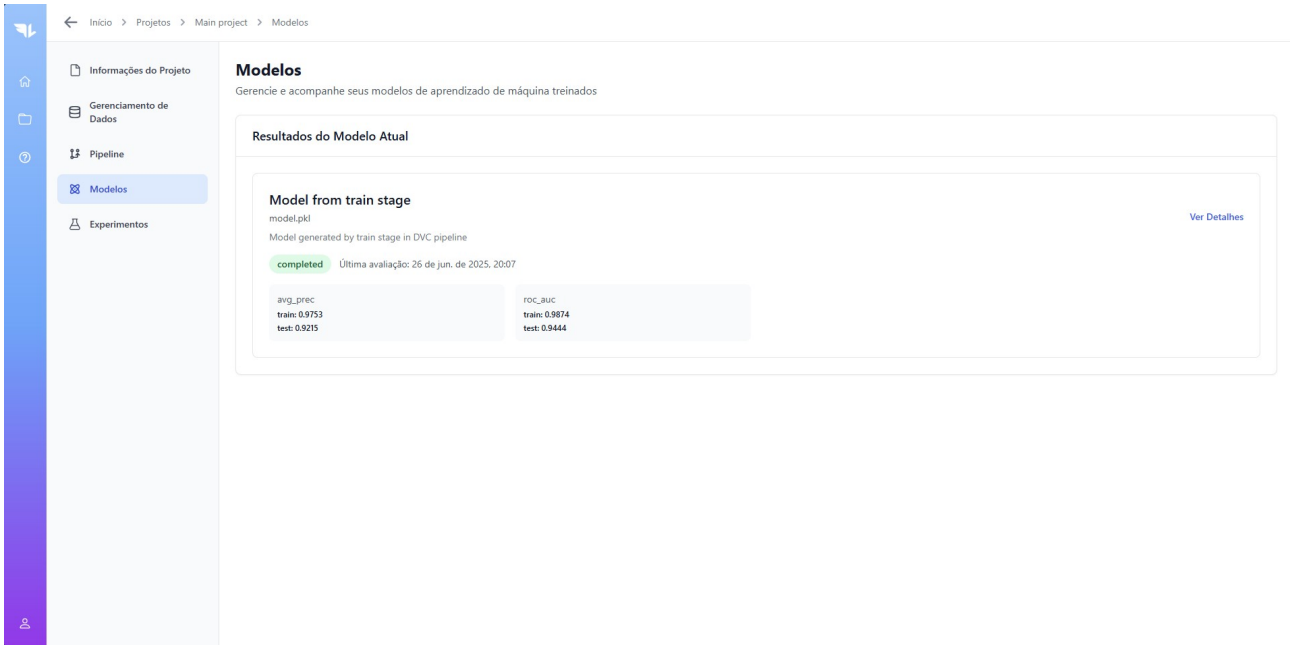

Figura 7. Página "Modelos"de um projeto.

A funcionalidade de avaliação customizada permite aos usuários definir e executar métricas específicas para seus casos de uso. O sistema implementa comparação automática de performance entre diferentes versões de modelos, fornecendo insights sobre evolução da qualidade. A funcionalidade de visualização de modelos implementa dashboards interativos para análise de performance, incluindo gráficos de evolução de métricas ao longo do tempo, comparação de performance entre modelos e análise de distribuição de erros. O sistema fornece visualizações específicas para diferentes tipos de modelo, incluindo matrizes de confusão para classificação, gráficos de regressão para modelos preditivos e análise de clusters para modelos não supervisionados.

4.2. Integração com API do POC I

A integração foi concluída com sucesso, garantindo comunicação eficiente com a API, possibilitando funcionalidades avançadas como execução e monitoramento de pipelines automáticos, geração de relatórios detalhados e comparação de resultados entre versões dos experimentos.

4.2.1. Visão Geral da Arquitetura de Comunicação

A plataforma implementa uma arquitetura de comunicação robusta entre o front-end React e o backend FastAPI, estabelecendo uma integração completa e eficiente para todas as operações de gerenciamento de projetos de machine learning. O sistema de comunicação é baseado em REST APIs com autenticação por usuário, permitindo operações seguras e controladas sobre projetos, pipelines, experimentos e modelos. A arquitetura de comunicação segue o padrão cliente-servidor com separação clara de responsabilidades. O front-end atua como cliente responsável pela interface do usuário e interações, enquanto o backend gerencia a lógica de negócio, persistência de dados e integração com ferramentas externas como DVC, Git e sistemas de armazenamento.

A comunicação é realizada através de requisições HTTP utilizando a biblioteca Axios, com tratamento robusto de erros e estados de carregamento. O sistema implementa um padrão de organização

modular para as APIs, onde cada área funcional possui seus próprios endpoints e serviços correspondentes no front-end. Essa organização facilita a manutenção, escalabilidade e compreensão do código. Cada módulo da aplicação (projetos, pipelines, dados, experimentos, modelos) possui sua própria API dedicada com endpoints específicos para operações CRUD e funcionalidades avançadas. A configuração da comunicação é centralizada através de constantes e configurações compartilhadas, permitindo fácil modificação de URLs base, timeouts e headers de autenticação. O sistema implementa interceptors para tratamento global de erros e transformação de respostas, garantindo consistência na comunicação entre front-end e backend.

5. Conclusão

Este trabalho detalhou o desenvolvimento e implementação de uma interface gráfica integrada à API MLOps para a plataforma Flautim, proporcionando uma solução completa e intuitiva para o gerenciamento do ciclo de vida de experimentos de aprendizado de máquina. As funcionalidades disponibilizadas pela interface web, desenvolvida com tecnologias robustas como React e TypeScript, tornaram acessível o gerenciamento de projetos, pipelines, experimentos e modelos para diferentes perfis de usuários, promovendo maior eficiência operacional e facilidade de uso.

Além disso, a integração bem-sucedida entre o frontend e a API backend demonstrou uma solução eficaz para automação, versionamento e monitoramento contínuo de dados e modelos. Os desafios técnicos enfrentados ao longo do projeto, como a garantia da responsividade e acessibilidade da interface, a otimização de performance nas interações com a API e a robustez na gestão de estados e tratamento de erros, foram superados através da adoção de boas práticas de desenvolvimento e ferramentas avançadas como React Query, Tailwind CSS e FastAPI. A arquitetura modular implementada facilitou a manutenção, a escalabilidade e a integração contínua das funcionalidades propostas, garantindo um ambiente consistente e confiável para execução e reprodução dos experimentos realizados na plataforma.

Por fim, este projeto reforça a importância de soluções MLOps integradas e bem estruturadas, evidenciando como práticas eficientes podem melhorar significativamente a produtividade e a experiência dos usuários envolvidos em processos complexos de aprendizado de máquina. Futuras melhorias podem contemplar a expansão das funcionalidades já implementadas, integração com sistemas adicionais de autenticação e autorização, e otimizações específicas para aprendizado federado, fortalecendo ainda mais o potencial da plataforma Flautim como ferramenta essencial para pesquisa e desenvolvimento em machine learning.

Referências

[1] Han BASIL. *Data Versioning for CD4ML - Part 2*. Accessed: 2024-11-05. 2021. URL: `https://ai4sme.aisingapore.org/2021/04/data-versioning-for-cd4ml-part-2/`.

[2] Satvik Garg et al. “On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps”. Em: *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. 2021, pp. 25–28.

[3] Dominik Kreuzberger, Niklas Kühl e Sebastian Hirschl. “Machine Learning Operations (MLOps): Overview, Definition, and Architecture”. Em: *IEEE Access* (2023).

[4] Sasu Makinen et al. “Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?” Em: *2021 IEEE First Workshop on AI Engineering - Software Engineering for AI (WAIN)*. Mai. de 2021, pp. 109–112.

[5] Danilo Sato, Arif Wider e Christoph Windheuser. *Continuous Delivery for Machine Learning*. Accessed: 2024-11-05. 2019. URL: https://martinfowler.com/articles/cd4ml.html.