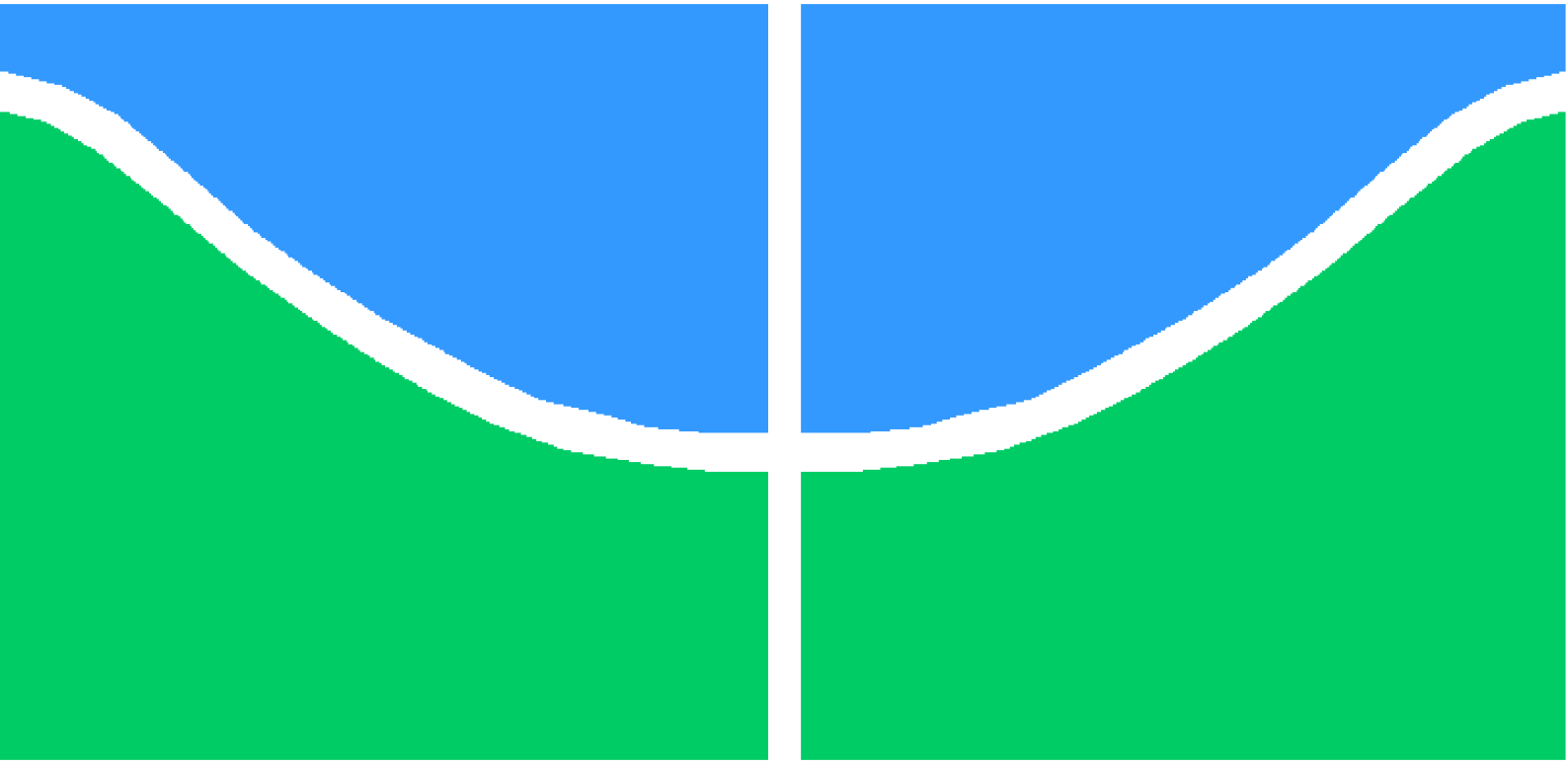

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

MLOpsTool: ferramenta de apoio a modelagem e deploy de soluções de IA

Autor: Victor Hugo Dias Coelho
Orientador: Prof. Dr. Nilton Correia da Silva

Brasília, DF
2021

Victor Hugo Dias Coelho

MLOpsTool: ferramenta de apoio a modelagem e deploy de soluções de IA

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Nilton Correia da Silva

Brasília, DF

2021

Victor Hugo Dias Coelho

MLOpsTool: ferramenta de apoio a modelagem e deploy de soluções de IA

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 05 de novembro de 2021:

Prof. Dr. Nilton Correia da Silva
Orientador

Prof. Dr. Fabrício Ataídes Braz
Convidado 1

Profa. Ma. Aline Dayany de Lemos
Convidado 2

Brasília, DF
2021

Este trabalho é dedicado a todos que desejam ter mais agilidade em processos de desenvolvimento de soluções de IA e que gostem de explorar maneiras de melhorar e automatizar processos de software. Esse projeto também é dedicado ao laboratório AILab que me formou como profissional e me deu ótimas oportunidades em trabalhar com tecnologias, dados, problemas e servidores que eu não teria acesso se não fosse por ele. E por fim, esse trabalho e dedicado a todos que participaram da minha formação acadêmica bem como aos meus familiares e amigos que me deram suporte de continuar com meu sonho apesar de todas as adversidades que tiveram durante período.

Agradecimentos

Agradeço a todos meus familiares e amigos que me deram o suporte necessário durante toda minha formação. Que me ajudaram não apenas em assuntos da faculdade como também da vida.

Agradeço aos ótimos professores que participaram da minha formação acadêmica e que me deixaram mais que preparado para o mercado de trabalho no qual eu atuo.

Agradeço, em especial, as duas mulheres da minha vida. À minha mãe, por sempre se preocupar, ajudar e incentivar a sempre querer mais. E à minha namorada, por estar ao meu lado o tempo todo, me motivando, mostrando meu potencial e me confortando nos momentos difíceis durante minha caminhada.

Agradecer ao Prof. Dr. Nilton Correia da Silva por acreditar em mim não apenas na confecção desse trabalho mas também ter aceitado e ensinado durante meu período de GPAM e depois no AILab. Sem as conversas, trabalhos e discussões que tivemos eu não teria tanto conhecimento na área como tenho hoje.

E por fim agradeço a todos que fizeram parte da minha formação de maneira direta ou indireta.

Resumo

No mercado contemporâneo, há diversas ferramentas para apoio ao ciclo de vida de um software e isso inclui os programas relacionados a aprendizado de máquina. Esta demanda de ferramentas para o auxilio de desenvolvimento e manutenção de softwares relacionados a inteligência artificial (IA), aumenta conforme as empresas usam de soluções com esse tipo de tecnologia. Entretanto, a maioria das soluções desenvolvidas para esta finalidade, abarcam apenas uma parte do ciclo completo de desenvolvimento e deploy de uma aplicação de IA. Dessa forma, as empresas não conseguem, de maneira eficiente ou fácil, garantir um vida útil de soluções com essas tecnologias, chamadas de lifelong learning (vida útil do aprendizado). Além disso, por resolverem apenas uma parte do problema proposto, as soluções do mercado precisam usar múltiplas plataformas e tecnologias distintas para compreender o ciclo completo de desenvolvimento, algo que atrasa o processo de criação e disponibilização de modelos de IA para o mercado. Portanto, a proposta geral desta monografia é a elaboração de uma ferramenta, aplicando metodologias com ideias ágeis, que englobe todo o ciclo de desenvolvimento de uma solução de IA que contenha a parte salvar dados, realizar processos de pré processamento, criar modelos, treinar e retreinar modelos, fazer classificações com suas avaliações corretas ou incorretas, em uma única plataforma focado em classificação de texto supervisionado, utilizando-se da linguagem python com o framework Django para realização da tarefa proposta. **Palavras-chaves**: MLOps. Lifelong Learning. IA. Django. Métodologia àgil.

Abstract

In the contemporary market, there are several tools to support the life cycle of a software and this includes programs related to machine learning. This demand for tools to aid in the development and maintenance of software related to artificial intelligence (AI), increases as companies use solutions with this type of technology. However, most solutions developed for this purpose only cover a part of the complete development and deployment cycle of an AI application. Thus, companies cannot efficiently or easily guarantee a lifetime of solutions with these technologies, called lifelong learning. Furthermore, as they only solve a part of the proposed problem, market solutions need to use multiple platforms and different technologies to cover the complete development cycle, something that delays the process of researching and making AI models available to the market. Therefore, the general proposal of this thesis is the elaboration of a tool, applying methodologies with agile ideas, which encompasses the entire development cycle of an AI solution on a single platform with a focus on supervised text classification, using the python language with framework Django to perform the proposed task.

Key-words: MLOps. LifeLong Learning. AI. Django. Agile Methodology.

Lista de ilustrações

Figura 1 – Consulta Google Trends desde 2004 dos termos citados. 19
Figura 2 – Investimento anual nas empresas de IA durante os anos. Retirado de (COLUMBUS, 2018) 20
Figura 3 – Ciclo de desenvolvimento de DevOps retirado de mandic no dia 12 de outubro de 2021 23
Figura 4 – Ciclo de desenvolvimento de MLOps retirado de nvidia no dia 12 de outubro de 2021 23
Figura 5 – Fluxo da aplicação MLOpsTool 34
Figura 6 – Diagrama de classes das modelos do projeto. Criado com django-extensions 02/05/2022. 45
Figura 7 – Tela inicial de criação de conta. 47
Figura 8 – Tela inicial de login. 48
Figura 9 – Tela inicial da aplicação com a o botão de criação do projeto. 48
Figura 10 – Tela de detalhes do projeto. 49
Figura 11 – Tela de detalhes do projeto com uma pipeline nova. 50
Figura 12 – Tela da edição de pipeline. 50
Figura 13 – Detalhes da pipeline. 51
Figura 14 – Criação da camada de pré processamento. 52
Figura 15 – Visualização de documentação. 52
Figura 16 – Detalhes da pipeline com pré processamento e base de dados. 53
Figura 17 – Criando Modelo. 53
Figura 18 – Detalhes da pipeline com modelo. 54
Figura 19 – Detalhes da pipeline já treinada com f1-score inicial. 55
Figura 20 – Detalhes da pipeline. 55
Figura 21 – Detalhes da pipeline. 56

Lista de tabelas

Tabela 1 – Comparativos das ferramentas com as funcionalidades propostas - parte 1. 24
Tabela 2 – Comparativos das ferramentas com as funcionalidades propostas - parte 2. 25
Tabela 3 – Tabela de requisitos funcionais 31
Tabela 4 – Tabela Backlog 32
Tabela 5 – Cronograma TCC 1 58
Tabela 6 – Tabela de associação épico e história de usuário. 58
Tabela 7 – Cronograma TCC 2 58

Lista de abreviaturas e siglas

IA	Inteligência Artificial
CD	Continuos Deployment
CI	Continuos Integration
API	Application Programming Interface
URL	Uniform Resource Locator
ELLA	Efficient Lifelong Learning Algorithm
UML	Unified Modeling Language
ORM	Object-Relational Mapping
TCC	Trabalho de Conclusão de Curso

Sumário

1 INTRODUÇÃO 19
1.1 Contextualização 19
1.1.1 Mercado de Inteligência Artificial 19
1.1.2 Processo de desenvolvimento de uma solução de IA 20
1.1.3 Processo de MLOps 22
1.1.4 Comparativo entre ferramentas do mercado 23
1.2 Justificativa 25
1.3 Objetivos 26
1.4 Metodologia do desenvolvimento 27
1.5 Estrutura do trabalho 28

2 REQUISITOS 29
2.1 Requisitos funcionais 30
2.2 Construção do Backlog 30
2.3 Fluxos 32

3 AMBIENTAÇÃO DO PROJETO 35
3.1 Tecnologias associadas 35
3.2 Utilizando CD/CI no Projeto 36

4 PONTOS PROPOSTOS 37
4.1 Camada de Dado ou Dataset 37
4.2 Funções de Manipulações Padrões 38
4.3 Funções de Manipulações Personalizadas 38
4.4 Modelagem Iterativa de Aprendizado de Máquina e Aprendizado Profundo 38
4.5 Métricas Padrões dos Modelos 39
4.6 Parecer para o Engenheiro de IA 40
4.7 Retreinamento e Deploy de Modelos 41
4.8 Requisições da API de classificação 41

5 RESULTADO 43
5.1 Ferramenta 43
5.2 Problemas enfrentados 46
5.3 Como utilizar a ferramenta e história de usuário que ela se associa . 47

6 CONCLUSÃO 57

6.1 Trabalhos futuros 57
6.1.1 Dívidas técnicas 57
6.1.2 Novas bibliotecas 57
6.1.3 Métricas 57
6.1.4 Retreinamento automático 57
6.2 Cronograma de execução 57
6.3 Considerações finais 58

REFERÊNCIAS 61

1 Introdução

1.1 Contextualização

1.1.1 Mercado de Inteligência Artificial

O mercado de inteligência artificial tem ganhado relevância nos últimos anos. Os termos como Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL) e suas traduções têm sua procura, em motores de busca, crescente desde o ano de 2004, no mundo todo, como mostrado na Figura 1, onde mostra uma constante crescente de interesse por esse tema.

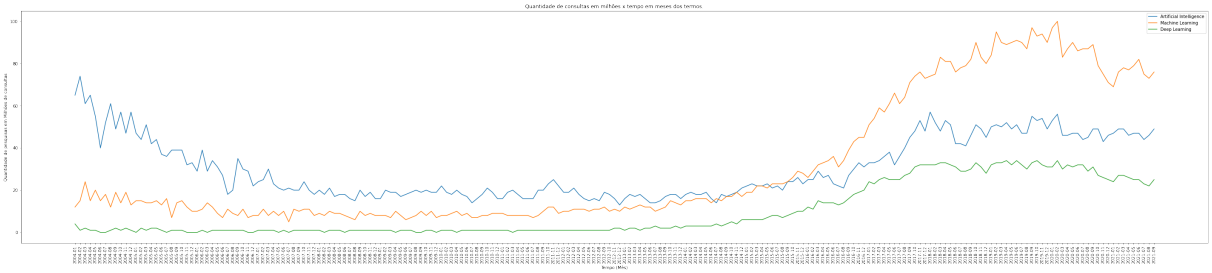

Figura 1 – Consulta Google Trends desde 2004 dos termos citados.

Com a procura crescente e as pesquisas feitas com relação ao tema de IA, houve o surgimento de cargos, como pode constatar na plataforma de buscador de vagas de emprego (GLASSDOOR, 2020) onde há uma clara enfase na área de ciência de dados sendo a terceira entre as melhores vagas de emprego. E também de ferramentas e bibliotecas nas mais diversas linguagens na área de IA como mostrado em (COLUMBUS, 2018). No mesmo período, a indústria começou a investir no setor e se utilizou desses recursos para melhorar e customizar seus serviços, tornando-os mais inteligentes e eficientes (DEPARTMENT,). Isso criou demandas cada vez maiores, de vagas de emprego para o setor, por profissionais, pesquisadores e desenvolvedores nesse nicho.

Como a indústria tem processos mais maduros do que os envolvidos no desenvolvimento de software, por conta de um mercado de desenvolvimento relativamente mais novo, começou-se um movimento para agilizar e automatizar estes processos, visando uma boa construção e utilização de soluções de IA. Isso gerou campos dos quais muitas empresas e startups começaram a atuar, como visto na Figura 2, que mostra o constante crescimento de investimento, ocasionando no surgimento de ferramentas cada vez mais robustas para dar suporte a nova área. Esta temática também pode ser percebida em (COLUMBUS, 2018), o qual mostra com mais de detalhes o crescimento da área ao longo dos anos.

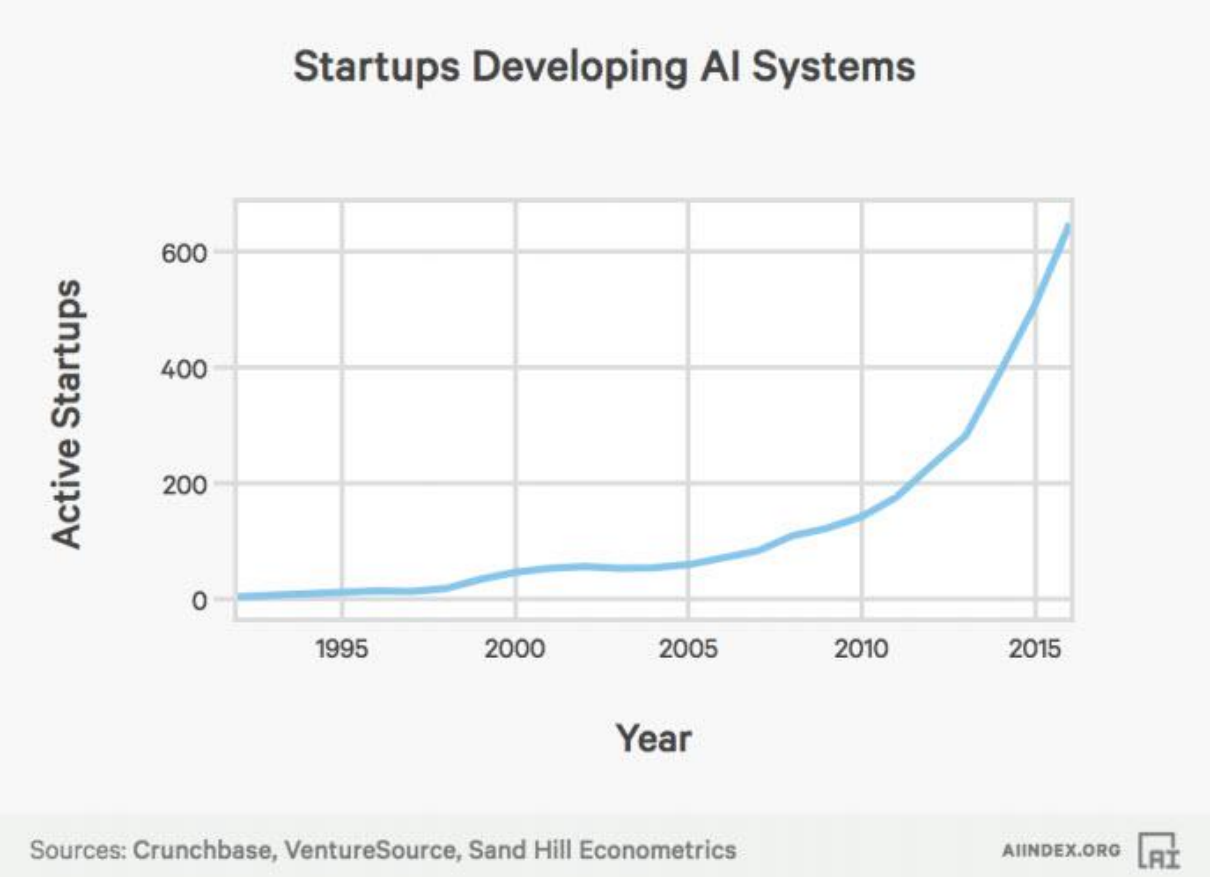

Figura 2 – Investimento anual nas empresas de IA durante os anos. Retirado de (COLUMBUS, 2018)

1.1.2 Processo de desenvolvimento de uma solução de IA

Um solução de inteligência artificial se divide em algumas partes. Elas são: a idealização, a definição de projeto, a curadoria dos dados, a prototipação de modelos, a disponibilização em produção e a continuação com o ciclo de vida IA (metrificação, retreinamento) (COVEYDUC; ANDERSON, 2020).

A idealização consiste em descobrir o que se quer resolver, de acordo com as ferramentas e modelos disponíveis atualmente, ou seja, saber qual problema quer ser mitigado com a sua solução. Logo após o objetivo ser estipulado, é necessário definir o que vai ser tratado no projeto.

A fase de definição de projeto é quando, uma vez decidido que a solução de IA pode ajudar no problema, estabelecer o objetivo que quer ser abordado com mais detalhes, como: onde os dados serão obtidos, como vai vir esse dado, como funcionaria, de forma básica, a implementação do projeto idealizado, definir quais requisitos serão utilizados para a realização do projeto, tanto os requisitos funcionais quanto os não funcionais, como as placas de vídeos e servidores. Nessa etapa, se possível, usar métodos de modelagem de requisitos, como o de histórias de usuário, para ter seus objetivos mais visíveis e mais

palpáveis, de forma a melhorar a produção de um produto de IA.

Para uma solução de IA que use aprendizado de máquina ou aprendizado profundo, uma das partes mais importante são os dados (JAIN et al., 2020). Isso faz com que a etapa de curadoria dos dados seja uma das mais importantes, se não for a mais importante dentro da produção de uma solução de IA. Nessa etapa é onde está definido como os dados vão ser extraídos, em que formato serão extraídos, como serão armazenados, como serão manipulados, quais possíveis problemas esse dado pode conter e que podem atrapalhar o sucesso do projeto e, por fim, como melhorar o dado para potencializar o treinamento do modelo de IA. Ainda nessa fase, como vários produtos de IA necessitam de muitos dados para que possam ter um bom desempenho, é normal usar de ferramentas que lidem com banco de dados e/ou dados descentralizados, como o Apache Hadoop.

A prototipação do modelo é a fase em que surgirão versões iniciais do que foi proposto na definição do projeto e serão utilizados os dados curados da etapa anterior. Esta é uma etapa iterativa, a qual constantemente ocorrem mudanças no modelo e no dado e, assim, é gerada outra versão do protótipo, até que se consiga uma versão base que será utilizada na solução. As mudanças nessa fase são constantes e sempre precisam ser mensuradas por métricas, como f1-score, recall e precision. Desta maneira é possível obter um progresso dos protótipos de forma objetiva. É na etapa de prototipação que as ferramentas nas quais trataremos adiante são usadas. Ferramentas como Knime, RStudio e IBM SPSS modeler são exemplos de alguma delas, que têm o foco em auxiliar a prototipação rápida dos modelos de IA.

A última fase desse processo é a disponibilização do modelo em produção. Nela, se obtém um protótipo com o desempenho desejado que irá trazer valor para o produto alvo dessa solução. É neste momento que deve acontecer uma revisita nas definições do projeto, deve ser feita uma re-priorização dos requisitos do sistema e serão adicionados os novos requisitos para a construção do sistema, o qual apontará a solução realizada. Nesse ponto são utilizadas metodologias ágeis de desenvolvimento de software, seguindo todas as etapas para a construção do sistema. Nessa etapa, ferramentas como MLFlow, AirFlow e MetaFlow são utilizadas para embarcar, de forma fácil, as soluções dentro do ciclo de vida da aplicação. Essas ferramentas versionam e implementam, na maioria dos casos, API's para realização de classificações.

A última fase também apresenta uma etapa a mais, que é a etapa de ciclo de vida de IA, chamada de lifelong learning, a qual verifica e indica a saúde do modelo em produção.

1.1.3 Processo de MLOps

Com a crescente aplicação de IA dentro do mercado, a necessidade de fazer uma forma de desenvolvimento com foco nessa área aumentou drasticamente. No desenvolvimento ágil de software, que não contém IA, já existia um termo parecido com MLOps, do qual ele herdou o significado para área de aprendizado de máquina. Esse termo é DevOps, que não apenas engloba a parte de desenvolvimento por si só, mas também entra nos aspectos operacionais do desenvolvimento de um produto, como deploy, testes automatizados, verificações de segurança, criação de ambientes de homologação de produção, configuração e monitoramento, assim como mostrado na Figura 3. Portanto, o MLOps vem com a mesma base de significado e funções, porém, ele substitui o termo de desenvolvimento por aprendizado de máquina, pois há mudanças na forma de pensar no desenvolvimento tradicional de um produto e no desenvolvimento de uma solução de IA (MÄKINEN et al., 2021), como pode ser visto na Figura 4.

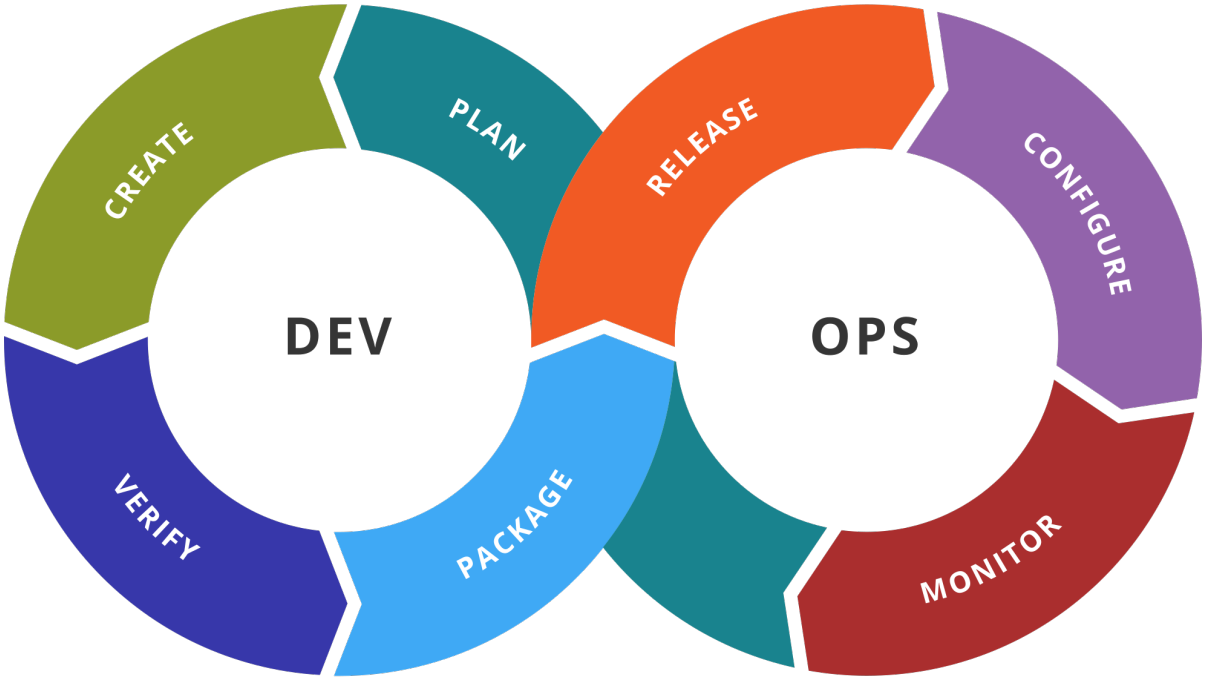

Figura 3 – Ciclo de desenvolvimento de DevOps retirado de mandic no dia 12 de outubro de 2021

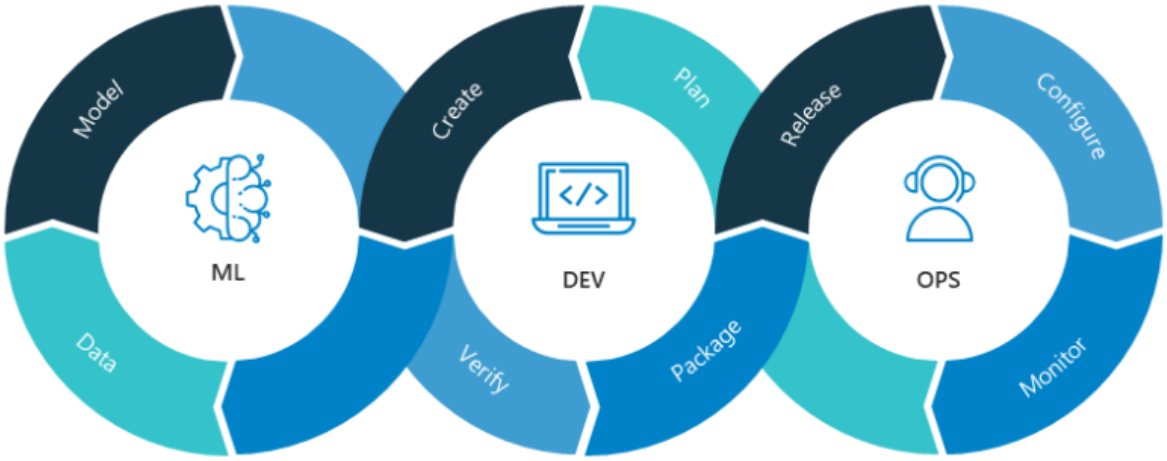

Figura 4 – Ciclo de desenvolvimento de MLOps retirado de nvidia no dia 12 de outubro de 2021

1.1.4 Comparativo entre ferramentas do mercado

Com base no estudo de projetos de IA(COVEYDUC; ANDERSON, 2020) foram retirados pontos nos quais uma ferramenta completa de desenvolvimentos de soluções tem que ter para dar suporte integro a esse tipo de objetivo. Essas funcionalidades serão explicadas a seguir junto com uma tabela comparativa de ferramentas de mercado com esse objetivo de suporte a área.

As ferramentas escolhidas foram Knime(KNIME,), Mlflow(MLFLOW,), Airflow(AIRFLOW,), RStudio(RSTUDIO,), Metaflow(METAFLOW,), IBM SSPS Modeler(MODELER,). Foram escolhidas por serem ferramentas listadas na plataforma aimultiple (AIMULTIPLE,), que tem como principal função listar e fazer relatórios de produtos destinados a IA.

Para melhor visualização dos comparativos entre as ferramentas, foram elaboradas duas tabelas. As colunas são relativas às funcionalidades e as linhas contém a ferramenta e, caso possua a funcionalidade, o campo da linha apresentará um x. Por conta do mesmo motivo citado anteriormente, as funcionalidades foram resumidas para que caibam na página. Dessa forma, os mapeamentos da coluna para a explicação da funcionalidade são:

- Armazenamento: Extração e armazenamento de datasets em diversos formatos;
- Manipulação de dados: Modificações e pré processamentos dos dados alvo da solução;
- Modelagem iterativa: Criação e estruturação de redes e algoritimos de inteligência artificial, de forma iterativa com o usuário;
- Metrificação: Metrificação e parametrização das soluções de IA;
- Retreinamento e versionamento: Retreinamento e versionamento das soluções realizadas;
- Deploy e monitoramento: Deploy contínuo dos modelos criados, para produção, monitoramento e atualização das métricas propostas.

Tabela 1 – Comparativos das ferramentas com as funcionalidades propostas - parte 1.

Ferramentas	Armazenamento	Manipulação de dados	Modelagem iterativa
MLflow	-	x	-
AirFlow	-	x	-
MetaFlow	-	x	-
Knime	-	-	x
RStudio	-	-	x
IBM SPSS Modeler	-	x	x

Tabela 2 – Comparativos das ferramentas com as funcionalidades propostas - parte 2.

Ferramentas	Metrificação	Retreinamento e versionamento	Deploy e monitoramento
MLflow	x	x	x
AirFlow	x	x	x
MetaFlow	-	x	-
Knime	x	-	x
RStudio	x	-	x
IBM SPSS Modeler	x	-	x

1.2 Justificativa

Por conta das ferramentas do mercado focadas em suporte, a área de IA não apresenta todas as características que somariam para que houvesse o desenvolvimento completo de uma solução com foco em durabilidade no tempo. Há a necessidade de uma ferramente que englobe todos os pontos anteriormente citados. Para isso, a ferramenta MLOpsTool vem como solução para essa lacuna, pois ela apresenta todos os pontos, como funcionalidades dentro da aplicação, englobando todo o ciclo de desenvolvimento de um software, desde a parte de guardar e estruturar dados, até a parte de monitoramento e melhoria dos modelos já existentes na ferramenta. Assim, além de ser uma ferramente para automatizar todo ciclo de uma aplicação de IA, também será focada em lifelong learning, para que os modelos da ferramenta realmente sejam usadas de forma desejável no mercado.

1.3 Objetivos

O Objetivo da ferramenta é atuar na área de modelagem e deploy de modelos de IA, nos pontos citados no comparativo. Além de, através de pareceres do usuário que irá utilizar a ferramenta, verificar a saúde do modelo em produção e, dessa forma, melhorar os dados de entrada do modelo criado. O projeto é destinado a projetos com enfoque em texto e classificação supervisionada se utilizando das bibliotecas sklearn e tensorflow, e não vai ser abranger outros tipos de entradas, como imagens e áudios ou classificações não supervisionadas.

1.4 Metodologia do desenvolvimento

Para a metodologia de desenvolvimento, será usada a metodologia com ideias ágeis (MANIFESTO. . . , 2001) e irá utilizar os frameworks kanban(JUNIOR; FILHO, 2010) e scrum (SCHWABER; SUTHERLAND, 2011). Do scrum será usado a parte planejamento das sprints com os backlogs de cada uma delas. Backlog é um acumulo de entrega ou afazeres que dentro da área de software se refere ao conjunto de tarefas que devem ser feitas para chegar em um determinado objetivo (IEEPEDUCACAO,).

O Scrum, que também é baseado no manifesto ágil, é um framework criado para lidar com equipes pequenas e experientes, para tratar de problemas de forma contínua e iterativa. Os processos envolvidos no Scrum podem ser utilizados em todos os tipos de equipe, não apenas para Tecnologia da Informação (TI) (SCHWABER, 2013). Porém como, no caso dessa monografia, não é destinada a uma equipe, os seus ritos não serão usados.

Outra metodologia adotada foi o kanban, que é um método para definir, gerenciar e melhorar serviços que entregam algum valor de conhecimento (ANDERSON; CARMICHAEL, 2016). O kanban tem como foco mostrar a forma de trabalhar dos indivíduos "começando com a aquilo que já se faz". O seu principal objetivo é deixar visível a quantidade de trabalho adequada para determinado espaço de tempo. Assim, esta é a ferramenta ideal para que se possa ter uma ideia de tempo de trabalho e uma agenda compatível com a data de conclusão dele.

Outro fator determinante para a escolha dessa metodologia dentro do trabalho foi a sua compatibilidade com o repositório git, que foi escolhido para a realização do trabalho. No caso, vai ser utilizado o gitlab(GITLAB,) para realizar todo gerenciamento, armazenamento e deploy do projeto. O gitlab tem seu próprio kanban, chamado de boards ou quadros, dentro da plataforma. Sendo assim, é possível fazer a associação de uma issue(cartão com problema a ser mitigado) no backlog, que é equivalente a um historia de usuário no nosso contexto, com seu momento e etapa no kanban.

1.5 Estrutura do trabalho

Esse trabalho está dividido em 3 partes principais: introdução, desenvolvimento e conclusão. A introdução é subdividida em Contextualização, onde será explicado sobre o mercado e sobre a área em que a ferramenta MLOpsTools deseja atuar; Justificativa, que é onde está a explicação sobre o desenvolvimento da ferramenta, dado o problema, o espaço de mercado e a oportunidade; Objetivos, onde estão explícitos os pontos que o projeto deseja atuar dentro do tema; Metodologia, que são os métodos usados para gerenciar o trabalho e o desenvolvimento da ferramenta; e, por último, está a Estrutura do trabalho, onde é explicado a divisão deste trabalho.

No desenvolvimento, composto pelos capítulos de requisitos, ambientação, pontos propostos, será atacado da mesma forma que a construção de um software moderno é feita. Nele, estão os requisitos do trabalho, que serão extraídos, priorizados e condensados em um backlog, a ambientação do projeto, onde irá acontecer a dockerização do ambiente e serão definidas as métricas e o deploy contínuo do projeto, e os pontos propostos, que são a explicação de como será abordado cada ponto extraído na análise de mercado e que serão abordados no projeto.

Na última parte do projeto é feita a conclusão que é onde está a análise total da ferramenta e se ela conseguiu cumprir com o que foi proposto.

2 Requisitos

Para que os objetivos do projeto sejam atendidos da maneira pretendida, é necessário um estudo sobre os requisitos que o sistema de IA precisa. Dessa forma, são abarcados nesse capítulo os requisitos funcionais que farão parte da definição do projeto. Assim, teremos a idealização, abordada na parte introdutória, e a definição do projeto que será abordada nas próximas secções.

Algo que deve ser feito antes de começar as especificações do projeto, é a importante tarefa de definir e priorizar os requisitos para cada área do software.

Um dos maiores problemas da indústria de desenvolvimento de software é não apenas entregar o produto combinado com o cliente, mas também entregar o produto da forma que o cliente desejava e que trás valor para o negócio em que o software vai dar suporte (BIRD, 2010). Para que isso seja mitigado, são necessários estudos e alinhamentos constantes com o cliente para extração, priorização e adição de novos requisitos, os quais precisam ser atendidos, de forma que o sistema consiga atender as expectativas do público alvo do produto.

Dentro da área de IA e das ferramentas de suporte a essa área não é diferente. Se precisa, para que se faça um bom projeto, que os requisitos sejam alinhados com o objetivo geral da solução inteligente. Ou seja, as definições do projeto devem estar alinhadas com a idealização do projeto.

Então, basicamente, o papel dos requisitos dentro do projeto é determinar o que vai ser produzido, quando vai ser produzido, quais as funcionalidades ou produtos virão de cada requisito e quando certo requisito foi ou não atendido (BRACKETT, 1990).

Portanto, após todos os requisitos serem atendidos, o produto estará completo. E, assim, se dá como encerrado o projeto e começa a etapa de evolução do produto.

As etapas seguidas, nesse capítulo, para construção do backlog do produto são:

- Requisitos funcionais: onde será definido o que, de fato, virará funcionalidade concreta no produto;

- Construção do backlog: em que todos os requisitos são priorizados e transformados em histórias de usuário para compor o backlog a ser desenvolvido nesse trabalho.

2.1 Requisitos funcionais

Os requisitos funcionais são aqueles que são concretos dentro do projeto. Normalmente, esses requisitos podem ser revertidos em funcionalidades para a aplicação. Dessa forma, esse requisito se torna o mais importante para guiar o desenvolvimento de um sistema.

Dentro dos requisitos existem algumas formas de elicitar os requisitos do sistema, como com os métodos de questionário, storytelling, observação, introspecção, entrevista e brainstorming. Por conta da facilidade e rapidez na elicitação dos requisitos, o método escolhido foi o de introspecção (ALFLEN; PRADO, 2020).

A introspecção é uma técnica simples, que pode ser realizada por qualquer pessoa. Porém, para que haja um identificação adequada dos requisitos necessários do sistema, é indicado que o indivíduo seja experiente no assunto em que deseja ser abordado na introspecção (ALFLEN; PRADO, 2020). No caso desse projeto, como há apenas um indivíduo os requisitos serão retirados com base na experiência dele na área.

A técnica consiste em uma busca pessoal, ao se colocar no lugar do usuário, de funcionalidades do sistema. Dessa forma, é possível criar uma tabela que lista tanto os pensamentos iniciais quanto as funcionalidades do sistema. Após a realização da tabela, tem de haver uma priorização das funcionalidades para que haja um desenvolvimento que foque no ponto principal da aplicação. Para a priorização, a técnica utilizada vai ser a MosCoW (MIRANDA,), que consiste em classificar o requisito elicitado em Must, Should, Could ou Won't, em que:

- Must: consiste em tudo que é imprescindível para o escopo do projeto, que sejam funcionalidades bases do seu projeto;
- Should: tudo que é importante para o escopo do projeto mas que, caso não tenha o produto, não perderá seu valor de negócio;
- Could: itens opcionais para a aplicação que são usadas, muitas das vezes, para agradar o cliente com funcionalidades extras;
- Won't: tudo aquilo que será descartado da aplicação, ou seja, não será desenvolvido.

Para melhor rastreabilidade dos requisitos, serão dados um identificador único (ID) para os requisitos.

2.2 Construção do Backlog

Para a construção do backlog do produto, será utilizado o método de histórias (COHN, 2004) de usuários, que consiste em uma forma em que o indivíduo, que executará

ID	Requisito	Prioridade
RF001	Usuário tem que conseguir fazer cadastro na aplicação	Must
RF002	Usuário tem que ser capaz de logar na ferramenta	Must
RF003	Usuário tem que ser capaz de visualizar os projetos	Must
RF004	Usuário tem que ser capaz de criar um projeto	Must
RF005	Usuário tem que ser capaz de criar um dataset	Must
RF006	Usuário tem que ser capaz de mudar o dataset	Must
RF007	Usuário tem que ser capaz de ver os detalhes do projeto	Must
RF008	Usuário tem que ser capaz de criar uma pipeline no projeto	Must
RF009	Usuário tem que ser capaz de escolher o dataset que deseja usar na pipeline	Must
RF010	Usuário tem que ser capaz de escolher as colunas x e y	Must
RF011	Usuário tem que ser capaz de escolher ou criar um método para pré processamento de informações e dados	Must
RF012	Usuário tem que ser capaz de escolher o pré processamento e a coluna que deseja fazer a operação	Must
RF013	Usuário tem que ser capaz de aplicar mais de um pré processamento no dado até que ele esteja satisfeito	Must
RF014	Usuário tem que ser capaz de testar o pré processamento	Could
RF015	Usuário tem que conseguir criar o modelo da redes de forma iterativa	Must
RF016	Usuário tem que conseguir adicionar quantas camadas quiser dentro da rede com os frameworks keras ou tensorflow ou sklearn	Must
RF017	Usuário tem que ser capaz de verificar se o modelo está funcionando	Could
RF018	Usuário tem que ser capaz de verificar as métricas do modelo	Could
RF019	Usuário tem que ser capaz de verificar a saúde do modelo	Must
RF020	Usuário tem que ser capaz de usar a API para classificar textos	Must
RF021	Usuário tem que ser capaz de enviar pareceres dos textos classificados	Must
RF022	Usuário tem como ser capaz de mudar de versão da API	Must
RF023	Usuário tem que ser capaz de ter acesso ao dado atualizado com os pareceres	Must
RF024	Usuário tem que ser capaz de fazer o retreinamento do modelo baseado em um novo dado	Must
RF025	Usuário tem que ser capaz de usar outros frameworks como pytorch e afins	Won't
RF026	Usuário deve poder criar tags	Could
RF027	Usuário tem que ser capaz de colocar tags no projeto	Could
RF028	Usuário administrador tem que ser capaz de aceitar outros usuários cadastrados	Could
RF029	Usuário tem que ser capaz de fazer o retreinamento do modelo	Must

Tabela 3 – Tabela de requisitos funcionais

ou usará a aplicação, descreve os seus desejos e os objetivos aos quais quer chegar com aquela ação realizada. Para visualização das historias, será usada uma tabela onde terá a identificação única, o individuo da ação, a ação a ser executada e o objetivo daquela ação.

ID	Eu, como	Desejo	Para que possa	Requisito relacionado
US01	Usuário	Cadastrar	Ter acesso ao sistema	RF001
US02	Administrador	Aceitar usuário	Para que ele possa ter acesso a ferramenta	RF028
US03	Usuário	Logar	Conseguir fazer ações na ferramenta	RF002
US04	Usuário	Criar Projeto	Conseguir diferenciar os projetos e criar projetos	RF004
US05	Usuário	Criar tags	Para diferenciar os projetos por tema	RF026
US06	Usuário	Criar pipeline	Para fazer os treinamentos	RF008
US07	Usuário	Criar dataset	Para utilizar nos treinos de modelos	RF005
US08	Usuário	Listar dataset	Para utilizar um dataset específico	RF006
US09	Usuário	Criar camada de dado na pipeline	Para utilizar na pipeline	RF009, RF010
US10	Usuário	Criar camada de pré processamento	Para manipular o dado da forma que quiser antes de treinar	RF011, RF013
US11	Usuário	Criar funções na camada de pré processamento	Para conseguir fazer múltiplas manipulações	RF012, RF013
US12	Usuário	Criar funções personalizadas de pré processamento	Para fazer as manipulações dos dados de forma personalizada	RF012
US13	Sistema	Listar funções de pré processamento	Para que a ferramenta consiga ser iterativa	RF013, RF014
US14	Usuário	Criar modelos de aprendizado de máquina	Para treinar com os dados	RF015, RF017, RF016
US15	Usuário	Criar modelo a partir de script externo	Para personalizar o treino	RF016
US16	Usuário	Criar modelo de aprendizado profundo	Para treinar os modelos com dados	RF015, RF017
US17	Sistema	Mostrar os modelos de forma iterativa	Para facilitar a prototipação para o usuário	RF016
US18	Usuário	Validar de a pipeline está correta e treinar	Para executar todo o processo mapeado.	RF029, RF024
US19	Usuário	Fazer deploy do modelo com sucesso	Para disponibilizar para classificação	RF023, RF019
US20	Sistema	Criar url e salvar modelo em disco	Para facilitar na hora de criar uma classificação	RF021
US21	Sistema	Mostrar status da pipeline	Para o engenheiro ter uma ideia das decisões para tomar	RF022, RF019
US22	Usuário	Ver status da pipeline	Tomar decisões	RF022, RF019
US23	Usuário	Retreinar modelo	para retreinar com novo dado ou alterações feitas	RF029, RF024, RF023
US24	Sistema	Disponibilizar url para classificação	para o cliente ter acesso a classificação	RF021, RF020
US25	Usuário	Compartilhar o link da url da API com cliente	Para ele ter acesso a classificação	RF021
US26	Cliente	classificar texto com o url da API	Classificar texto	RF020
US27	Cliente	Dar pareceres das classificações	Melhoria dos dados e para atualização das métricas	RF018, RF021, RF023

Tabela 4 – Tabela Backlog

2.3 Fluxos

Para melhor visualização das possibilidades dentro da ferramente, é importante ter a ideia de fluxos dentro da aplicação. Para isso, será utilizado um diagrama UML (LUCIDCHART, 2021) para que fiquem claras as decisões e suas implicações dentro do software. O diagrama que será utilizado é o diagrama de sequência (LUCIDCHART, 2021).

O diagrama de sequência é uma solução dinâmica de modelagem em UML bastante usada, porque incide especificamente sobre as linhas da vida, ou em processos e objetos que vivem simultaneamente, e sobre as mensagens trocadas entre eles, para desempenhar uma função antes do término da linha da vida (LUCIDCHART, 2021).

A Figura 5 mostra o funcionamento do fluxo com sucesso da aplicação. Tendo isso em vista, os primeiros passos são realizar o cadastro, para que a rastreabilidade dos processos envolvidos no treinamento sejam feitos. Graças ao login, será possível saber quem realizou cada ação dentro dos projetos na ferramenta. Logo após o usuário estar cadastrado e logado dentro da aplicação, é necessário criar um projeto assim, como o que foi citado nas etapas de um projeto de aprendizado de máquina (COVEYDUC; ANDERSON, 2020).

Após criar o projeto, é necessário criar as pipelines dele. As pipelines são as linhas de produção, ou seja, são as etapas em que os dados vão ter que passar para ser criada uma versão do modelo que se quer trabalhar.

Nessa etapa, são definidos o que será adicionado por camada. As camadas são dataset (camada de dados), pre processing (camada de pré processamento) e a camada de modelo (camada de prototipação de modelos de IA). Essas camadas são adicionadas de maneira customizada e manual pelo usuário que cria a pipeline.

Quando todas as camadas necessárias forem especificadas, o botão de treinamento será desbloqueado e, quando acionado, será feito o treino e as verificações se a pipeline está funcionando corretamente.

Caso tudo ocorra com sucesso, o botão de deploy daquela versão do modelo aparecerá no final do processo, de forma a deixar ele aberto para realizar as inferências em produção. E, em troca, o sistema retornará as métricas de saúde, neste caso o f1-score, do modelo e a url para requisições.

Depois dessa etapa, o fluxo do usuário que consome as API's de produção começa. Com a url em mãos, o indivíduo é capaz de fazer a classificação de um texto de forma a receber de volta o texto, a classificação do texto, a porcentagem de certeza da classificação e a url de parecer da classificação. Essa url serve para que ele (o usuário) julgue classificação do texto do dado enviado à API está correta ou não. E, caso esteja incorreta, terá o campo para que seja colocado a classificação real do texto.

Quando essa etapa de parecer é feita, um novo dataset versionado é feito com as mudanças dos dados que foram colocados como incorretos, e será feita a atualização das métricas de saúde do modelo. Assim finalizando o fluxo de inferência completo da aplicação.

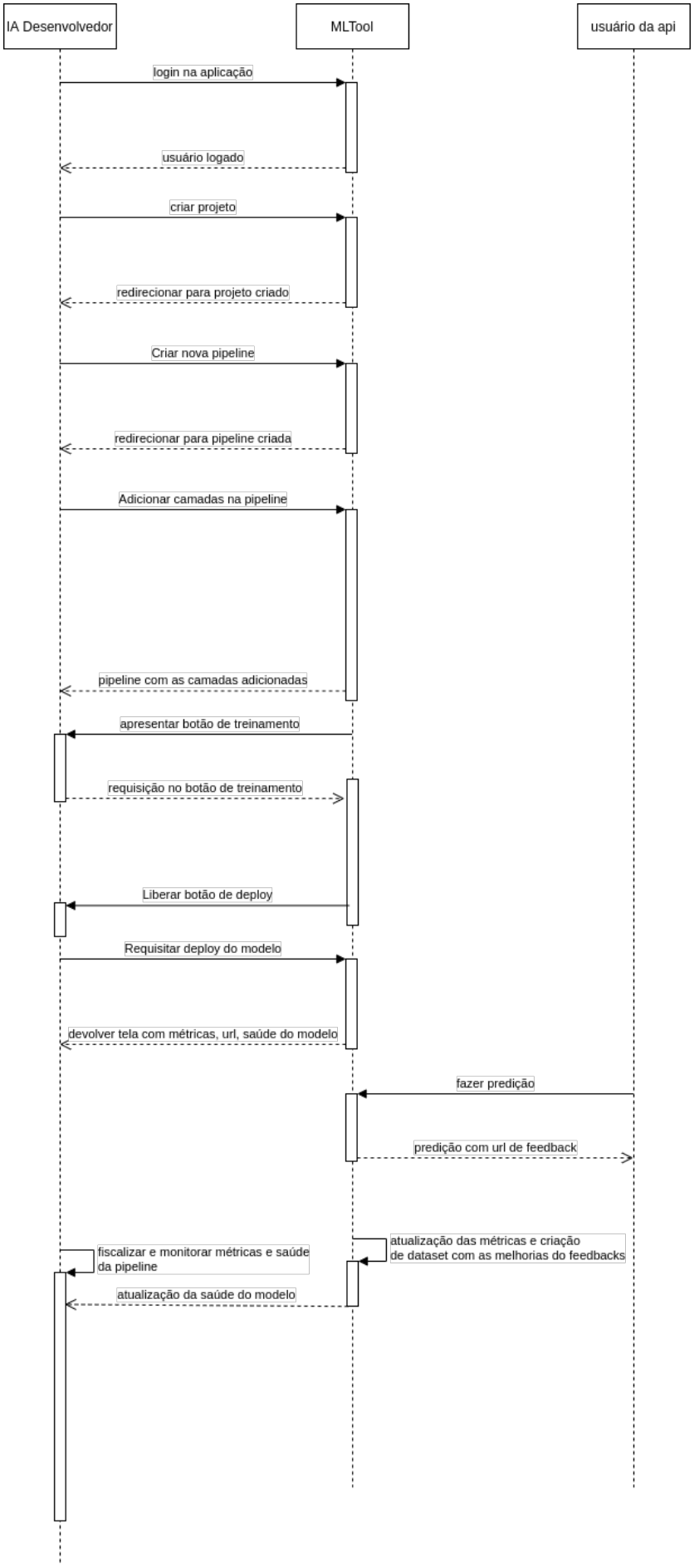

Figura 5 – Fluxo da aplicação MLOpsTool

3 Ambientação do projeto

3.1 Tecnologias associadas

Para a realização dessa ferramenta, é fundamental escolher uma linguagem e framework compatível com as necessidades do projeto. Essas necessidades são uma linguagem que suporta as bibliotecas de IA disponíveis no mercado, que seja de fácil acesso e com documentação vasta e rica, pois, uma vez que vamos trabalhar com probabilidade e estatísticas é importante ter várias bibliotecas de apoio (RASCHKA; MIRJALILI, 2017). Também é essencial uma linguagem que tenha frameworks de criação de API's e que seja possível fazer páginas web para a interface da ferramenta.

Por conta desses pontos, a linguagem escolhida foi o python, uma vez que é a linguagem que apresenta mais compatibilidade com as bibliotecas de IA, que apresenta uma vasta comunidade de código aberto, que apresenta grande variedades de bibliotecas de estatística e que possui frameworks de API's como o Django e Flask (RASCHKA; MIRJALILI, 2017).

Dentro dos frameworks web possíveis, foi escolhido o Django, tanto por conta da vasta compatibilidade de apps para realização de tarefas web, quanto pela ORM que ele possui para realização de consultas nos mais diversos bancos de dados (DJANGO,).

O banco de dados escolhido foi o postgres, pois é um banco open source que apresenta evoluções constantes, tem ótimo desempenho, é um banco relacional, mas que consegue lidar com tipos de dados complexos, como dicionários, imagens, vídeos e localização geográficas (GROUP,).

Para guardar os códigos da aplicação, foi utilizado a plataforma do gitlab, que utiliza do protocolo git para versionamento de código. Além disso, o gitlab possui o próprio kanban que vai ser utilizado durante a execução do projeto. E também por apresentar na própria plataforma servidores para realização de deploy e integrações contínuos de código. Algo que vai ajudar na criação do produto na web (GITLAB,).

Para que todo ambiente seja isolado e não tenha problemas com compatibilidade de hardware e sistemas operacionais, será adotado o docker e o docker-compose no projeto. O docker é uma ferramenta de virtualização de máquinas, que emula um sistema operacional de forma a isolar o ambiente de desenvolvimento do ambiente da máquina onde está rodando. Sendo assim, não importa onde irá rodar, o comportamento da aplicação é o mesmo (DOCKER,).

3.2 Utilizando CD/CI no Projeto

Para que a ferramenta tenha um comportamento adequado, tenha evoluções constantes e com qualidade, é necessário que haja etapas automáticas (FITZGERALD; STOL, 2014). Essas etapas têm o nome de deploy contínuo (Continuous Deploy) e integração contínua (Continuous Integration).

É na etapa de integração contínua que o novo código produzido é comparado com o atual e são verificados os testes automatizados, a folha de estilo, a cobertura de testes, a complexidade do código, a segurança e afins, com o intuito de garantir que a agregação do novo código ou funcionalidade esteja adequada aos padrões e que não apresente bugs (FITZGERALD; STOL, 2014).

Logo após a etapa de integração vem a etapa de deploy contínuo, em que o ambiente é geralmente separado em desenvolvimento e produção. Nessa etapa, o código que passou na etapa anterior é enviado para o(s) servidor(es) que está hospedada a aplicação, para que ela esteja disponível para ser utilizada. Essa etapa garante que as mudanças feitas e aprovadas na etapa anterior sejam colocadas em produção de maneira automática.

Para essas duas etapas no projeto, será utilizado o gitlab-ci (GITLAB,), para integração contínua e, para a realização do deploy contínuo, será utilizado o Heroku. No gitlab-ci serão automatizados os testes, a verificação da folha de estilo e a cobertura de testes da aplicação. Já no Heroku, será feita a alocação do servidor e do banco de dados de forma contínua na web.

4 Pontos Propostos

Neste capítulo serão desenvolvidos os pontos propostos dos capítulos anteriores, os quais foram usados para fazer o comparativo das ferramentas de apoio ao desenvolvimento de sistemas de IA. Dessa maneira, será tratado cada ponto crítico da aplicação de maneira separada em cada seção. A seção irá mostrar o problema que deve ser mitigado e como será feito a solução dentro da ferramenta proposta nesse trabalho.

4.1 Camada de Dado ou Dataset

No desenvolvimento das ferramentas e soluções de IA, um dos aspectos mais importantes, se não o mais importante, é o dado. Um bom dado, muitas vezes, é o que define se a solução é plausível ou não dentro do projeto, fazendo-o ser o ponto mais crítico dentro da aplicação. Sem o dado, as demais camadas da aplicação se tornam inúteis.

Dentro da camada de dado, existem dois principais pontos:

- Extração de dados: São necessárias fontes confiáveis e padrões bem definidos de dados para que se tenha confiabilidade. Além disso, muitas vezes, é fundamental também a abundância e a variabilidade de dados, para que os resultados da modelagem sejam mais próximos da vida real possível;

- Armazenamento de dados: Uma vez que possua o dado, há a necessidade de poder consultar e fazer manipulações sobre ele. Nesse ponto, um bom armazenamento com alta disponibilidade e performance é essencial para a área de IA. Essa área é a que dá mais suporte para a modelagem das soluções, uma vez que o dado seja de qualidade e que também seja confiável.

No projeto, a parte de extração de dados não será tratada dentro do escopo, como é evidenciado no requisitos e no backlog do produto a ser desenvolvido. Dessa forma, resta somente o armazenamento de dados e, para isso, será utilizado o banco de dados relacional postgres e a biblioteca pandas, para que possam ser lidos os mais diversos tipos de dados que a biblioteca dá suporte, como csv, parquet, json, xslx. Para isso, a ferramenta vai ler um arquivo, que será de responsabilidade do usuário, com o dado em que ele deseja trabalhar e irá transformar em uma tupla no banco de dados com as informações. Para isso, ele transforma o dado de qualquer formato em um dicionário e salva esse dicionário no banco de dados. Todos esses dados são associados juntos em um único pacote binário

e é enviado para o banco de dados, para que haja mais performance no commit dessas informações no banco. Para isso, vai ser utilizado a função bulk create[1] do Django.

4.2 Funções de Manipulações Padrões

Ainda sobre o dado, um aspecto muito importante são as manipulações que se pode fazer com ele, de forma que se tenha máxima otimização para a tarefa específica a ser feita com ele. Por conta disso, existem algumas manipulações padrões das bibliotecas que serão utilizadas dentro do projeto.

As bibliotecas que serão usadas do projeto são nltk, sklearn, keras e tensorflow. Nelas, é possível fazer algumas manipulações no dado alvo, como retirar stop words, tokenização dos dados e transformações vetoriais. Para facilitar o uso da ferramenta, todas essas funções de pré processamento de dados mais usados dentro da área de texto estarão disponíveis de maneira iterativa, apenas vai precisar parametrizar a função escolhida e escolher o dado ou coluna alvo para realizar o processamento.

Além disso, manipulações padrões como a de deletar dados duplicados, nulos e/ou com dada condição serão disponíveis de maneira iterativa também. Todas essas manipulações serão salvas no banco de dados da pipeline criada, para que possa ser refeito o experimento ou a modelagem de novo.

4.3 Funções de Manipulações Personalizadas

Mesmo as funções padrões de manipulação de dados serem bastante robustas, às vezes elas não são suficientes para o que se quer fazer com o dado. Para isso, será disponibilizado uma url dentro da ferramenta para adicionar mais funções de manipulações personalizadas pelo usuário. Essas funções terão um padrão para criação e esse padrão será mostrado ao usuário quando ele for adicionar essas funções na ferramenta. E serão armazenadas como arquivos com identificadores únicos que serão salvos no banco de dados.

4.4 Modelagem Iterativa de Aprendizado de Máquina e Aprendizado Profundo

Após todas as etapas da camadas anteriores, vem a camada de prototipação que será a que haverá o treinamento e que será responsável pela classificação dos textos. O mais importante nessa camada, é que seja fácil a prototipação de qualquer tipo de modelo dentro dos frameworks sklearn e tensorflow.

[1] Documentação da função bulk create disponível em: https://docs.djangoproject.com/en/3.2/ref/models/querysets/

No caso de modelos do sklearn, a iteração é fácil, pois não possui camadas personalizadas, apenas os parâmetros das funções dos modelos. Neste caso, a iteração terá todos os campos de parâmetros e o dado que é para ir no eixo x e no eixo y dos modelos. Dessa forma, será capaz de treinar os modelos dessa biblioteca.

O caso do tensorflow será tratado de duas formas. A primeira forma são modelos sequenciais, em que serão adicionadas camadas de forma iterativa até o final e, depois, haverá a validação se o modelo montado se encontra correto na compilação do modelo. E, a segunda forma, é utilizar funções externas de um arquivo a ser passado para a ferramenta através de uma url, em que terá um padrão de utilização e, desse modo, conseguir construir modelos não sequenciais dentro da parte de modelagem.

4.5 Métricas Padrões dos Modelos

Outro fator importante para qualquer projeto de IA, é saber o quanto seu protótipo está performando dentro do contexto escolhido. Para isso, as métricas são importantes, pois elas guiam o trabalho conforme o objetivo da métrica. No caso do projeto, será usado a métrica de f1 score. Essa métrica consiste em uma análise estatística de classificação binária, ou seja, caso o modelo devolva probabilidades, será estabelecido um limiar para binarizar suas saídas, visando conseguir utilizar essa métrica. Nela, é medida a assertividade do modelo com relação ao real e é calculado com base em outras duas métricas, que é a precision e a recall.

Antes de explicar como é calculado, é importante ressaltar o que é verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo.

- Verdadeiro positivo: é quando é classificado como pertencente daquela classe e realmente é da classe que foi classificada;
- Verdadeiro negativo: é quando determinado exemplar é classificado como pertencente da classe mas, na verdade, ele é de outra classe;
- Falso positivo: quando o exemplar é classificado como não pertencente daquela classe e, na verdade, realmente não pertence a ela.
- Falso negativo: quando o exemplar é classificado como não pertencente daquela classe mas, na verdade, ele pertence a ela.

A definição da precision é "Dos que eu classifiquei como corretos, quantos efetivamente são corretos?". E é dado quantidade de verdadeiros positivo, dividido pela quantidade de verdadeiros positivos mais a quantidade de falso positivos:

$$precision = \frac{verdadeiros\ positivos}{verdadeiros\ positivos + falsos\ positivos} \tag{4.1}$$

A definição de recall é "Se for realmente pertencente a essa classe, com que frequência você classifica como pertencente a ela?". Seu cálculo é feito pela quantidade de verdadeiros positivos dividido pela quantidade de verdadeiros positivos mais os falsos negativos.

$$recall = \frac{verdadeiros\ positivos}{verdadeiros\ positivos + falsos\ negativos} \tag{4.2}$$

O f1 score e a média harmônica da precision e do recall.

$$f1 = \frac{2 * precision * recall}{precision + recall} \tag{4.3}$$

Sendo assim, temos um número único que vai indicar o quão saudável nosso modelo está ou o quão assertivo ele é.

Essa métrica vai ser constantemente atualizada conforme o parecer do usuário, pois a quantidade de verdadeiro positivo e de falso negativo vai se atualizar durante a execução da API em produção.

4.6 Parecer para o Engenheiro de IA

Um dos pontos essências para que o projeto tenha uma durabilidade e assertividade é a saúde do modelo. Essa saúde vem do conceito de lifelong learning na área de IA, que tenta quantificar, através de métricas e pareceres, a assertividade e o quão eficiente o modelo está desempenhando.

Existem várias formas de medir a eficiência do modelo. Uma delas é a atualização constante de métricas baseadas no parecer que o usuário tem dos dados que ele fez a classificação. Por conseguinte, se novos dados tem sua classificação feito de forma errônea, a saúde do modelo será prejudicada, mas, caso contrário, a saúde permanece a mesma que estava anteriormente, ou seja, se mantém constante no tempo. Para isso, é necessário que seja definido a métrica de saúde do modelo que, no nosso caso, é o f1 score descrito na seção anterior, e os limiares no qual o modelo é considerado ruim, médio ou bom dentro do produto proposto. Esse limiar, no caso da aplicação, é um número de 0 a 1, em que 1 é 100 por cento de acerto e 0 é 100 por cento de erro.

Um algoritmo muito comum dentro de lifelong learning, é o de transferência de conhecimento e retreinamento do modelo baseado em novas entradas, como em (RUVOLO; EATON, 2013). Porém, diferente do algoritmo do ELLA, que é baseado na métrica, este já retreina de forma automática e transfere conhecimento para a próxima tarefa, isto é, para próxima geração do modelo. O algoritmo que será usado na ferramenta, será o de monitoramento da saúde, onde terá em cada pipeline do projeto um círculo com três cores: vermelha, amarela e verde, que significam ruim, médio e bom, nessa ordem. Dessa forma, cada modelo terá um status no qual o engenheiro responsável pelo projeto decidirá se

vai retreinar, alterar alguma camada da pipeline ou criar uma nova pipeline baseada na anterior com o novo dado corrigido, via parecer do usuário.

É importante ressaltar que o status inicial do modelo, antes dos pareceres, é baseado na métrica do conjunto de treino, teste e validação, que será extraído do dado de entrada do modelo.

4.7 Retreinamento e Deploy de Modelos

Uma vez que o modelo está treinado com sucesso, e tem seu monitoramento continuamente ligado, haverá a opção de fazer o deploy do modelo. Desta maneira, após a escolha de fazer o deploy daquela pipeline, será disponibilizada uma url com o modelo e a versão da pipeline, de forma a ficar disponível na rede em que a ferramenta está conectada. Essa rede pode ser online ou local.

Toda vez que uma pipeline nova for feita, o deploy terá uma url nova para ela, de forma que nunca se deve deixar de disponibilizar versões anteriores das pipelines dos projetos. O versionamento dessas pipelines vão ser feitos de forma automática, para que haja um padrão definido nas urls em deploy e evitar que existam mais de um padrão dentro da aplicação, o que poderia causar inconsistências. Sendo assim, para retreinar um modelo, você pode alterar as camadas e os dados e depois acionar a funcionalidade de treinar aquela pipeline ou, para uma nova versão, você tem que criar uma nova pipeline para aquele projeto.

4.8 Requisições da API de classificação

Uma vez feito o deploy do modelo e tendo a url, o usuário já pode realizar requisições para a ferramenta. Para realizar a classificação, o indivíduo terá de enviar o texto para a url destino e esperar a resposta da ferramenta. A ferramenta irá devolver o texto que foi enviado, um identificador único para recuperar aquela classificação, a assertividade da classe ou das classes que o texto foi classificado, a lista de classificações que aquele modelo pode haver e a url de parecer para criação de um dado atualizado com o parecer do usuário.

Quando um determinado dado é classificado, o usuário terá a opção de dar o parecer para a ferramenta. Caso deseje, ele pode enviar o identificador único da classificação e enviar ela para a ferramenta, indicando se aquela classificação está correta ou não. Dessa forma, será criado um novo dado com os dados de parecer daquela pipeline atualizados e as métricas serão atualizadas e refletidas dentro da saúde da pipeline.

5 Resultado

5.1 Ferramenta

A ferramenta MlOpsTool foi criada com sucesso, abarcando quase todas as funcionalidades pretendidas no projeto. Com exceção da funcionalidade da US12 e US15 da Tabela 4 de histórias de usuário. Que fala de confecção de funções de pré processamento e de criação de modelos personalizados.Porém, o restante dos objetivos do trabalho foi alcançado. Nesse caso, a ferramenta está operacional para todos os modelos do sklearn que tenham foco em classificação de texto, e todos os modelos de aprendizado profundo do keras que funcionem de forma sequencial.

A integração só foi possível graças a abstrações dos métodos das bibliotecas citadas. Foi utilizado o padrão de projeto chamado command dentro dos serviços de modelo e de pré processamento. Commmad (SHVETS,) é um padrão de projeto comportamental que tem como objetivo de transformar um método, em um objeto independente com o objetivo de ser parametrizado de forma individual. Dessa forma foi possível transformar cada tipo de modelagem e de pré processamento do sklearn e do tensorflow em objetos únicos e parametrizados. E ao transformar esses objetos em dicionários do python, foi possível salvar a estruturas dos métodos criados no banco de dados postgres (GROUP,). Pois, no postgres é possível salvar dados como json, e essa funcionalidade é abrangida pelo ORM do django que faz a integração com o banco. O que tornou possível realizar a criação de múltiplas camadas de pré processamento e de modelo dentro da aplicação.

Por conta do padrão estrutural criado, foi possível realizar o caminho de treinamento corretamente. Uma vez que, ao ter toda estrutura como dicionário contendo os objetos com seus parâmetros, foi possível fazer uma linha de produção na qual realizasse todos os passos de maneira sequencial conforme o usuário desejar. Assim não importando o número de camadas que o usuário queira em cada uma das etapas citadas.

O banco de dados da ferramenta nos dá um aspecto geral de seu funcionamento. Dessa forma, é de extrema importância que haja esse diagrama para explicitar as relações que as tabelas têm no projeto. Para isso, foi utilizada a ferramenta do django, chamada django-extensions, que consiste em um conjunto de bibliotecas que implementam ferramentas para análise e manipulações de dados dos projetos feitos pelo framework.

Uma breve explicação das relações do banco da dados. Dentro da aplicações podemos ter múltiplos projetos e base de dados. Dentro dos projetos tem múltiplas pipelines que são criadas apenas para aquele projeto não podendo ser compartilhada com outros projetos. Ao contrário das bases de dados que podem e devem ser utilizados em múltiplos

projetos. Dentro de cada pipeline pode conter a associação com uma base de dados da camada de dados, com múltiplos métodos na camada de pré processamento(que são específicos daquela pipeline não podendo ser compartilhada com outra pipeline) e com um modelo de rede neural ou com um modelo de aprendizado de máquina. Dessa forma se tem uma linha de produção completa e pronta para o treinamento dentro da plataforma.

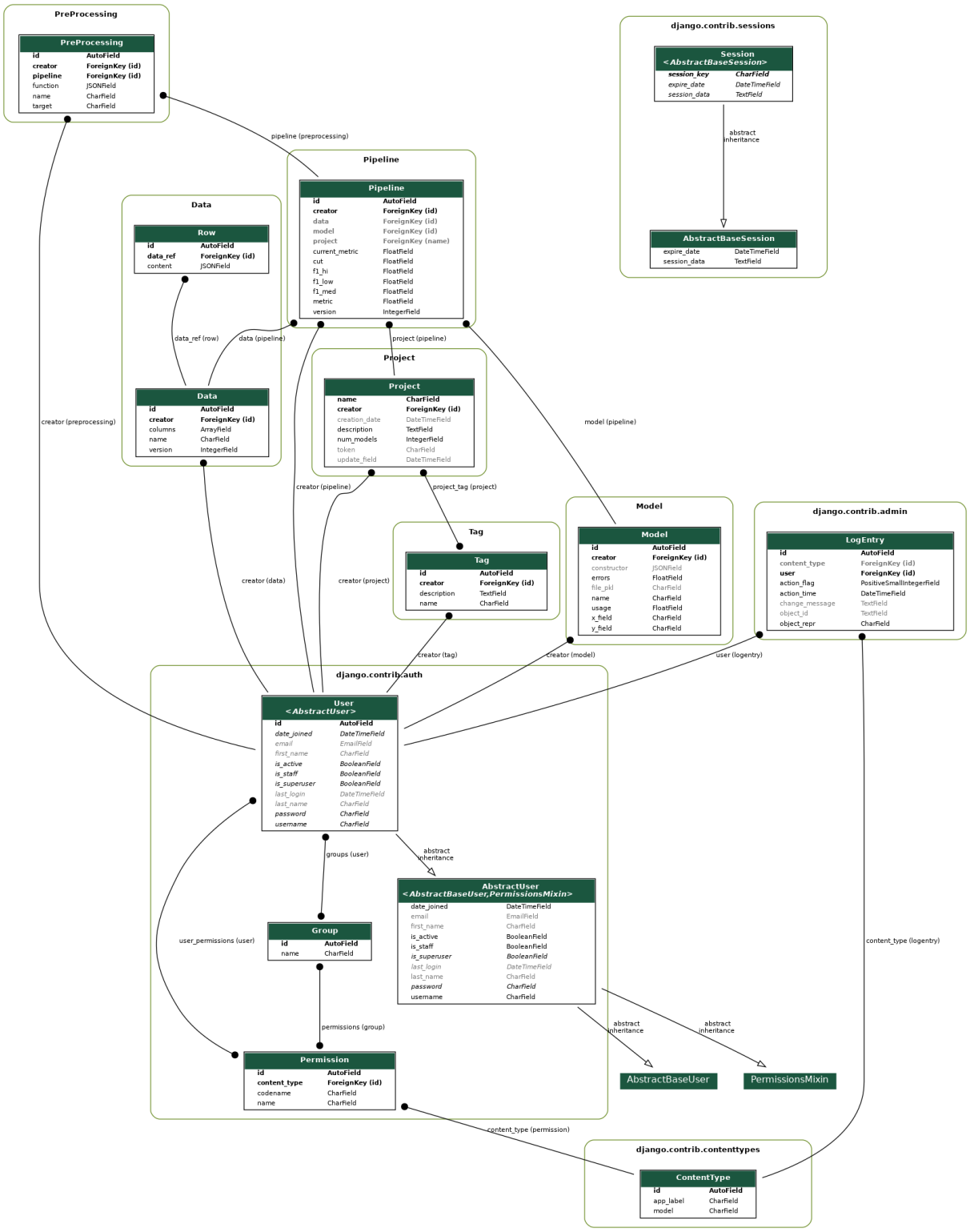

Figura 6 – Diagrama de classes das modelos do projeto. Criado com django-extensions 02/05/2022.

Todas as funcionalidades citadas dentro dos requisitos do projeto na tabela 4, foram atendidas. Porém, um dos requisitos de deploy contínuo do projeto, apesar de pronto, não pode ser levado até o servidor de testes do heroku. O motivo principal dessa escolha foi por conta da quantidade de dados que necessitam estar salvos. O heroku é

uma ferramenta que permite subir aplicações dentro de seus servidores e disponibilizar para todos de graça, porém, quando excedida certa quantidade de inserções diárias no banco de dados, este começa a cobrar do indivíduo em dólar, o que torna inviável subir o projeto, uma vez que só o dado simples de teste tem mais de dez mil tuplas a serem salvas em bancos de dados. Portanto, com a ferramenta disponível na internet a todos para teste, haveria um número muito grande de inserções no banco, o que acarretaria, por sua vez, em um alto prejuízo financeiro ao detentor da conta. Por conta dos motivos anteriormente citados, foi decidido que a apresentação do projeto acontecerá de maneira local, sem envolvimentos de servidores externos.

5.2 Problemas enfrentados

Os principais problemas enfrentados no decorrer do projeto foram relacionados a padronização dos métodos de treinamento. Por serem bibliotecas totalmente diferentes uma da outra (sklearn e keras), foi necessário bastante tempo para desenvolver uma abstração razoável, de forma que fosse possível a integração entre as duas bibliotecas, sem que a implementação estivesse muito específica. Desse modo, foi feita uma integração que permite a adição de futuras bibliotecas sem que haja alteração nas implementações anteriores.

Outro problema relevante encontrado no projeto, foi com relação à criação da aparência da ferramenta, de forma que permitisse fazer a modelagem e o pré processamento de um jeito iterativo e sem grandes problemas. Mas, por ser muito dinâmica, a renderização de componentes html dentro da página, fez com que fosse gasto muito tempo na adequação da lógica de criação de modelos e do pré processamento à parte visual do site.

Outra questão que foi adicionada para ajudar na criação dessas camadas, foram as documentações de todos os métodos listados das classes. Para isso, foi utilizada a biblioteca inspect do python, que lê todas as metainformações das classes e, ainda, podem ser extraídas as documentações das classes no formato PEP 275 e/ou o nome e métodos de cada classe que está presente na biblioteca em que se deseja inspecionar. Nessa parte, o principal desafio foi não apenas listar os métodos de cada biblioteca, mas, também, vincular ela à sua documentação da maneira correta. Além disso, fazer essa documentação mudar de maneira dinâmica, conforme os métodos fossem sendo mudados dentro do campo de escolhas do html, foi um trabalho que demandou bastante tempo e empenho. E, para solucionar esses casos, foram utilizados eventos no javascript com render dinâmico, conforme as decisões que foram feitas anteriormente, como a de biblioteca que se deseja utilizar os métodos.

No que diz respeito à utilização dos reportes que o usuário deu das classificações retornadas pela API, relacionada àquela pipeline desenvolvida no projeto, o problema

principal foi em como utilizar esse dado para acrescentar no já existente, com a finalidade de melhorar o dado. Para isso, foi feita uma nova base de dados com o nome da anterior, toda vez que ocorrer o primeiro reporte, mas com o sufixo "remake"após, para indicar o novo dado, a fim de copiar todo o dataset anterior e acrescentar os novos dados que vieram da classificação feita pelo usuário.

A métrica também se faz muito importante e presente dentro do trabalho, pois, através dela, é possível atestar a saúde do modelo. Sendo assim, existe a métrica inicial de f1-score, que é baseada no corte do dado designado a teste, e há a métrica de f1-atualizado, que é o dado com a classificação errada mais a classificação correta, acrecida no dado de teste e recalculada no f1-score. Assim, é possível ter uma ideia de como o modelo vai se degradando com o tempo. Os dados corretos também são acrescidos assim que são reportados, pois também impactam na métrica.

5.3 Como utilizar a ferramenta e história de usuário que ela se associa

Os identificadores da histórias de usuário se encontram na tabela 4. Sera usado os identificadores para associar a tela a sua história de usuário.

1. Crie uma conta e espere ser aceito pelo administrador da ferramenta conforme a Figura 7. Historias de usuário: US01;

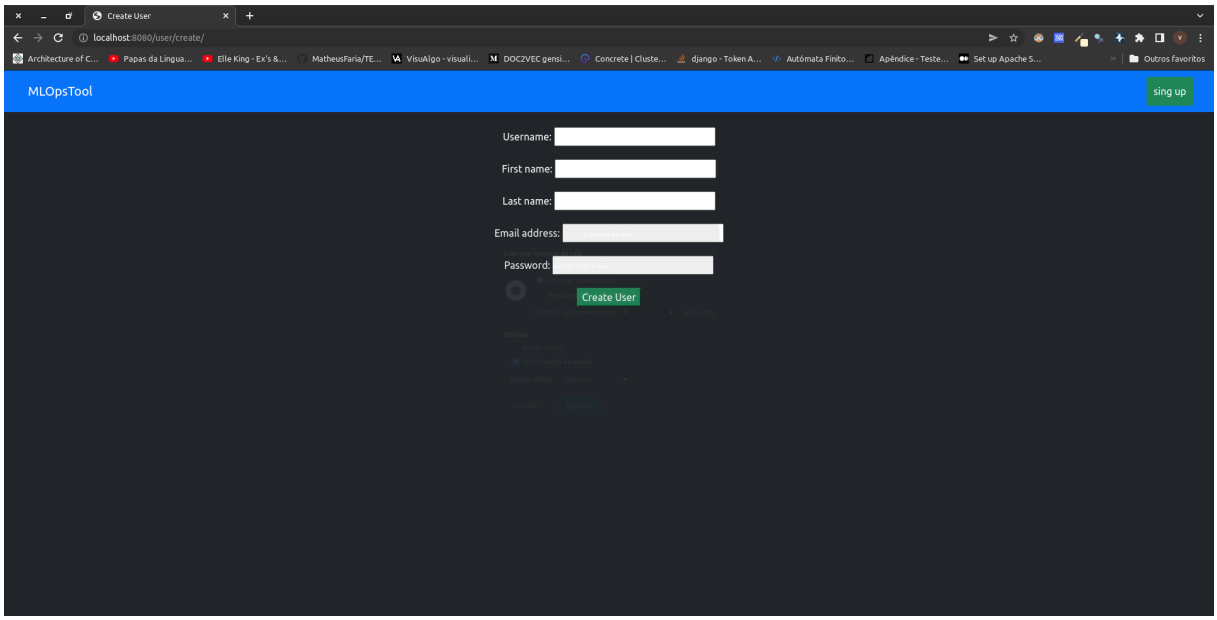

Figura 7 – Tela inicial de criação de conta.

2. Tela de login da aplicação. Onde se deve colocar as credenciais assim que for aceito pelo administrador Figura 8. Historias de usuário: US03;

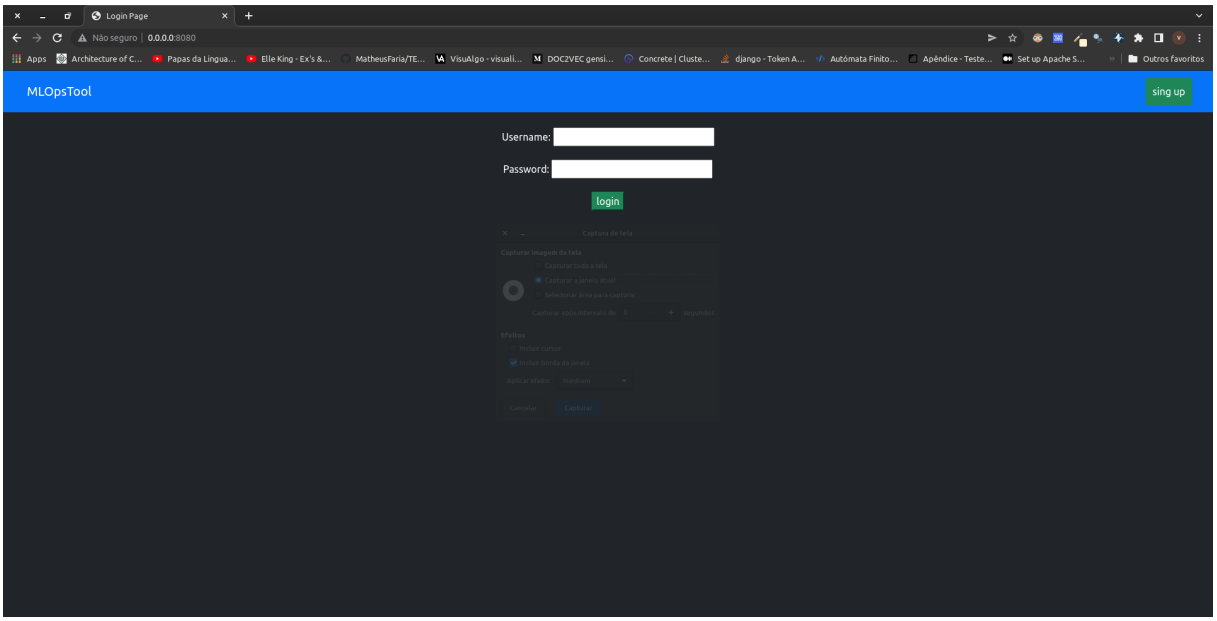

Figura 8 – Tela inicial de login.

3. Depois de logado, vá para página inicial da aplicação e, para criar um novo projeto, aperte no carácter "+"na parte superior da tela, conforme a Figura 9. Historias de usuário: US04;

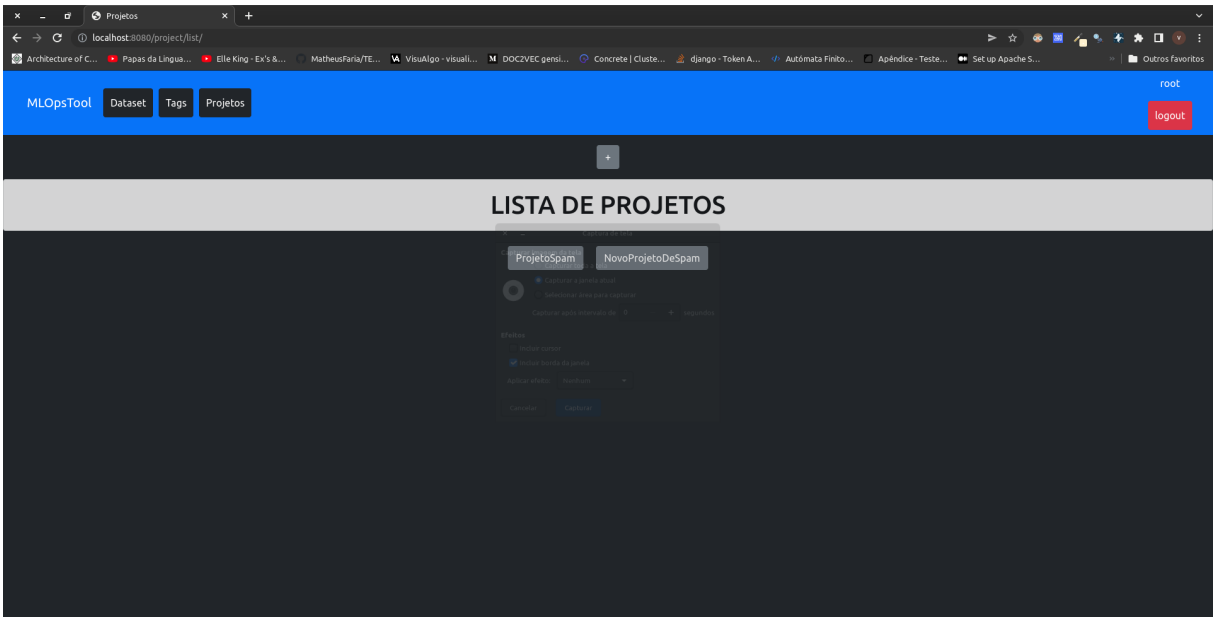

Figura 9 – Tela inicial da aplicação com a o botão de criação do projeto.

4. Crie o projeto com um nome único, ou seja, não pode haver um projeto com o mesmo nome que o seu. Adicione as tags que deseja e a descrição dele, com um breve resumo

do que se trata o objetivo do projeto. Após este passo, você vai ser redirecionado para a página inicial da aplicação novamente e, então, aperte no projeto que deseja trabalhar;

5. Um projeto vazio vai ser visualizado assim como na Figura 10;

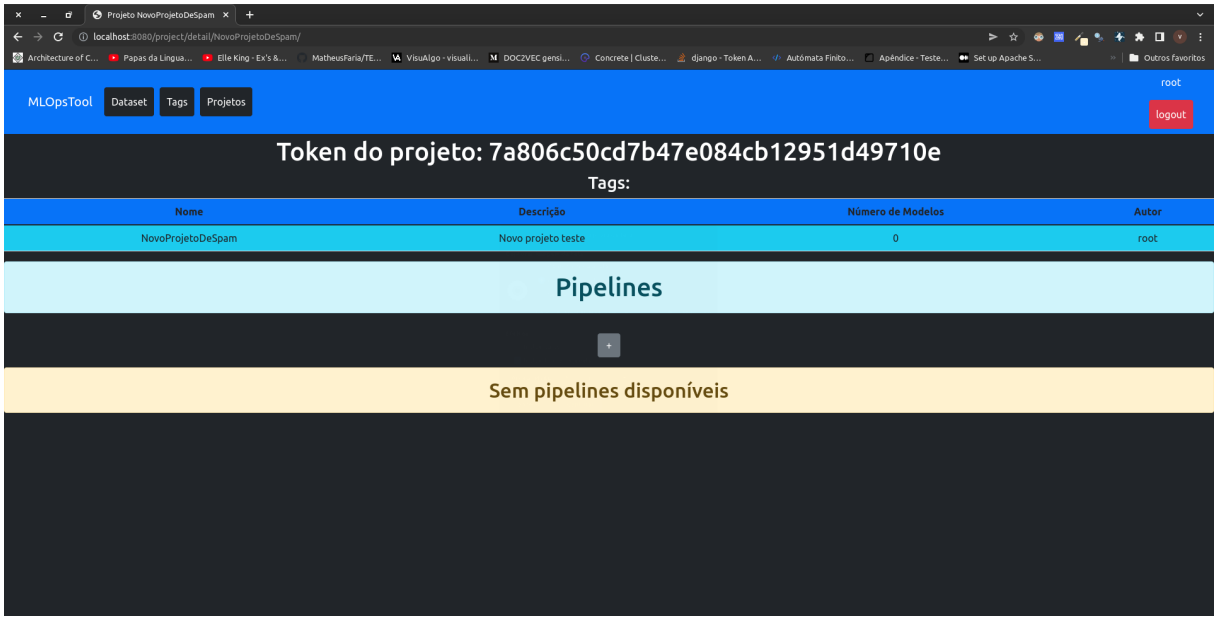

Figura 10 – Tela de detalhes do projeto.

6. Nesta tela, vai aparecer o token que será usado para conseguir fazer as classificações dos textos de forma segura e outros detalhes do projeto, como a listagem de todas as pipelines presentes;

7. Crie uma pipeline nova clicando no "+"dentro da página; Historias de usuário: US06;

8. Depois da criação, aparecerá uma pipeline nova na página com as informações de versão, com as métricas, com as opções de editar e excluir a pipeline e seus identificadores, assim com mostra na Figura 11;

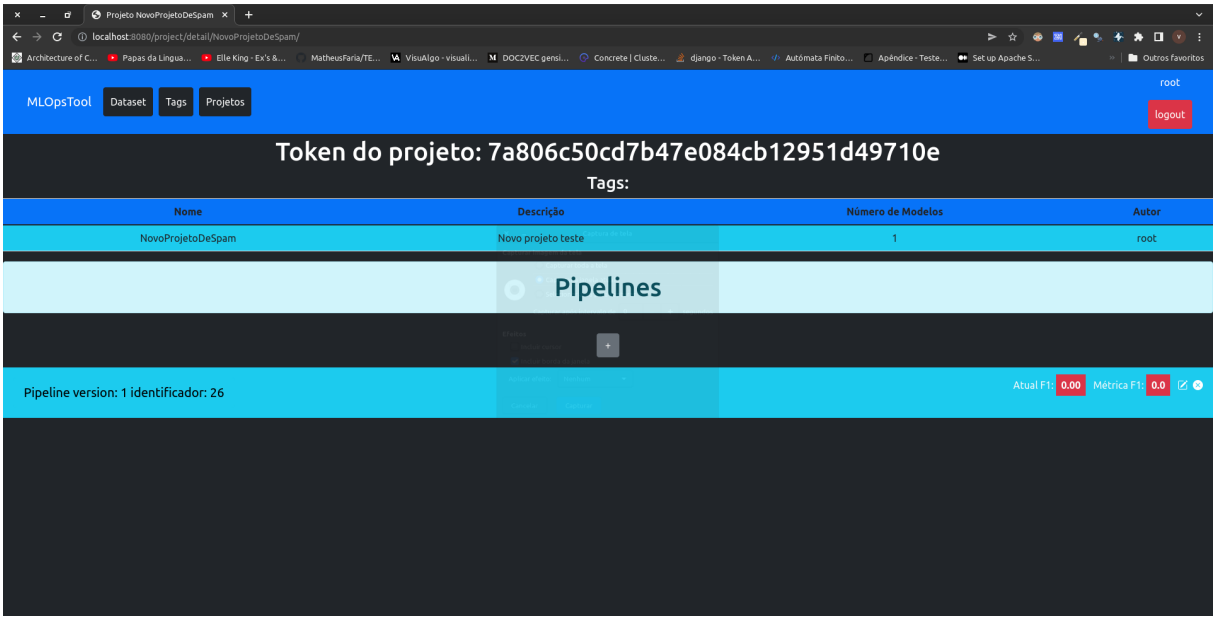

Figura 11 – Tela de detalhes do projeto com uma pipeline nova.

9. Clicando no item de editar, você pode editar tanto os limiares de saúde do modelo, quanto o corte de dado de treino e teste. Assim como na Figura 12;

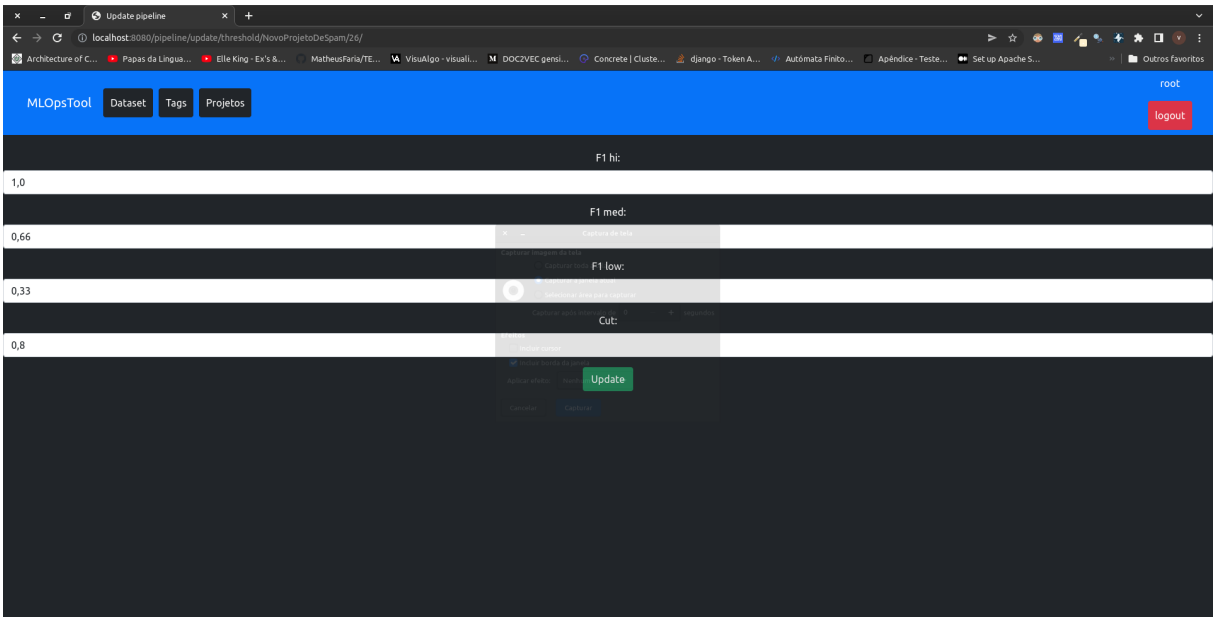

Figura 12 – Tela da edição de pipeline.

10. Clique na pipeline que deseja trabalhar, para criar toda linha de processos, assim como na Figura 13;

Figura 13 – Detalhes da pipeline.

11. Vincule o dado que foi criado anteriormente na aba de dados e prossiga para a criação do pré processamento. Historias de usuário: US07, US08, US09;

12. Na parte de criação do pré processamento, escolha a biblioteca que deseja e os métodos e, então, aparecerão todos os métodos no html logo em seguida. Ao clicar em qualquer um dos métodos, surgirá o botão de documentação que, quando clicado, vai mostrar a documentação do método escolhido. Assim como mostra na Figura 14 e na Figura 15. Historias de usuário: US10, US11, US13;

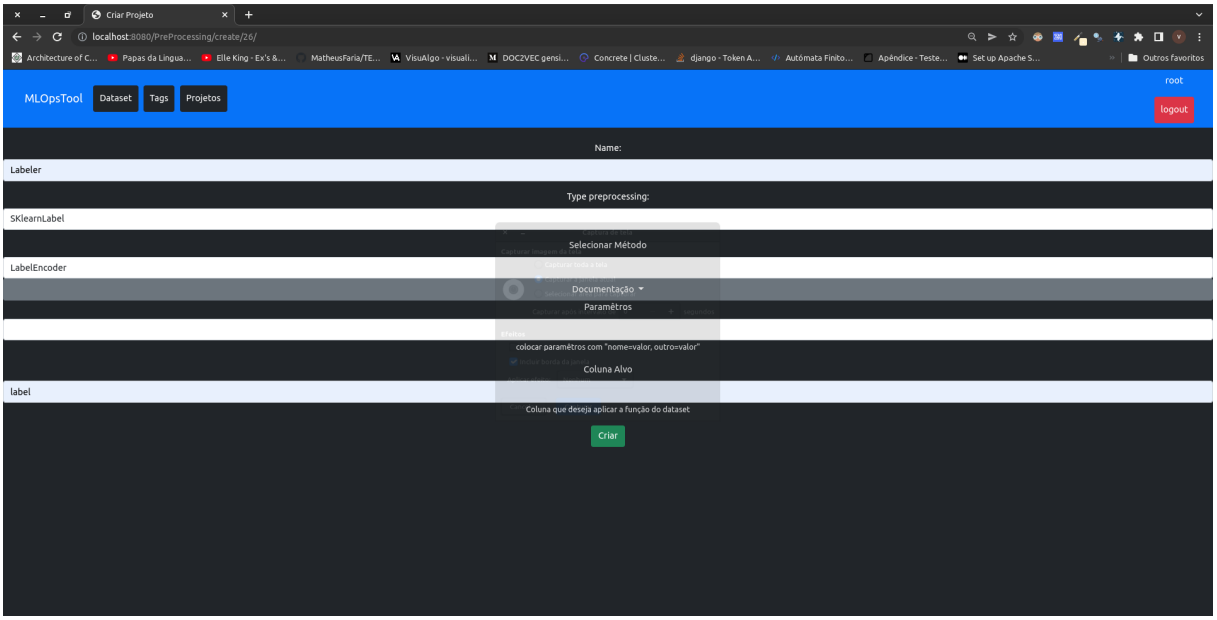

Figura 14 – Criação da camada de pré processamento.

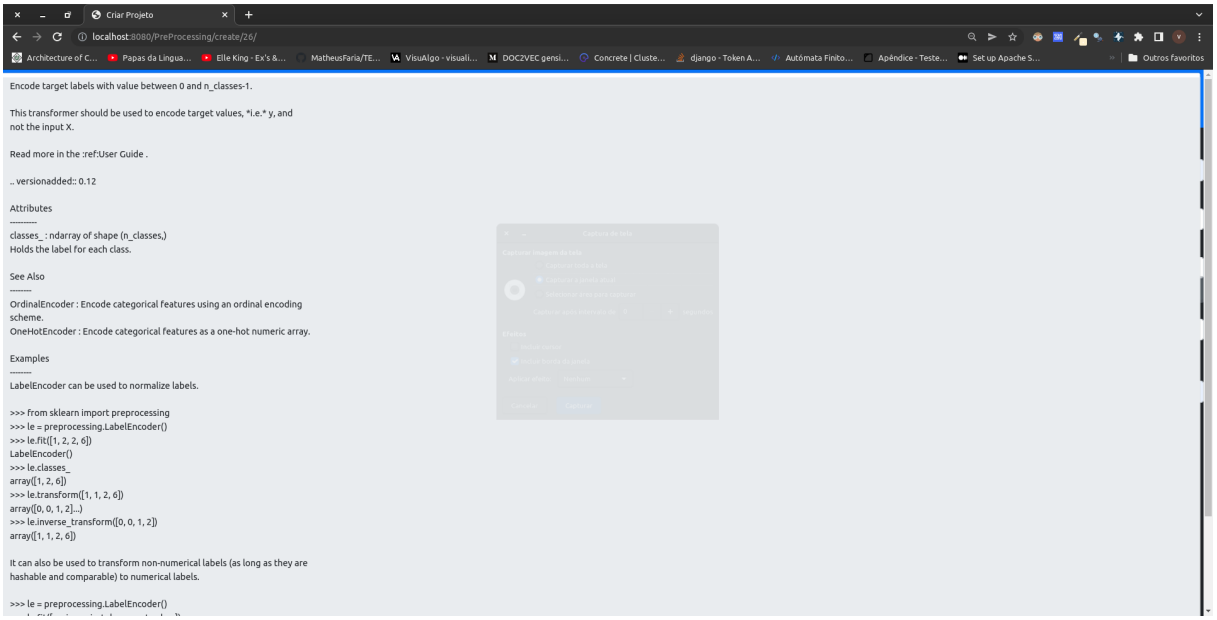

Figura 15 – Visualização de documentação.

13. Adicione, assim, todos os pré processamentos na ordem desejada, para que seja feito como na Figura 16;

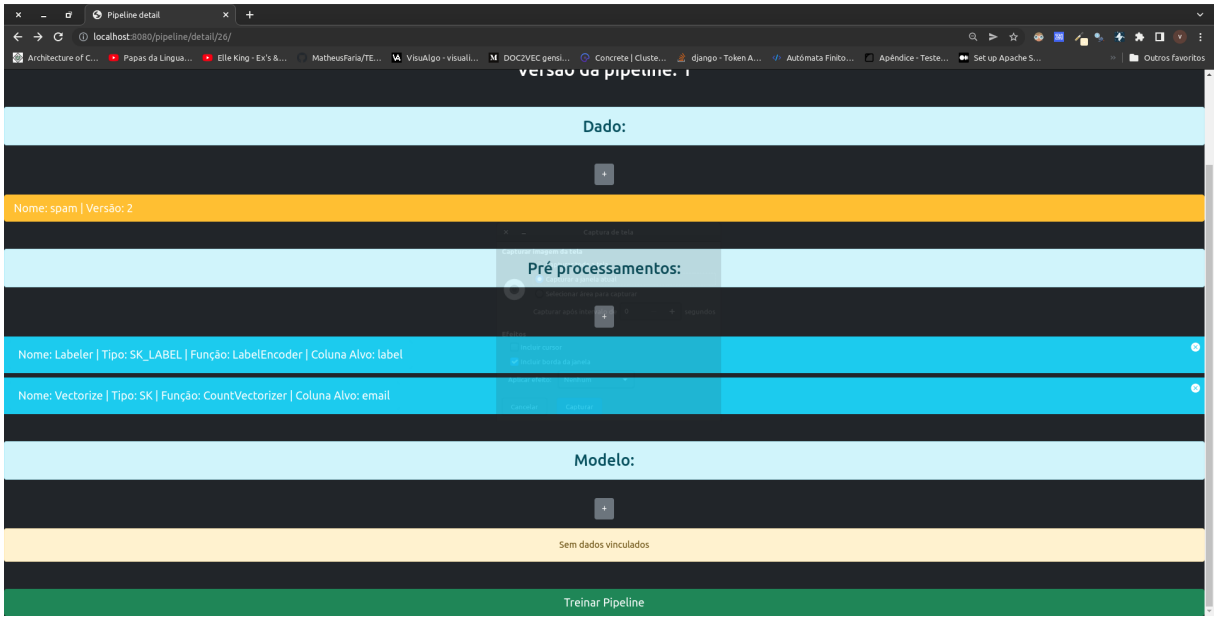

Figura 16 – Detalhes da pipeline com pré processamento e base de dados.

14. Clique em "adicionar modelo"para criar um modelo para ser treinado pela pipeline;

15. Na aba de modelos, escolha um nome e um tipo de modelo, para que saiba se é sklearn ou keras. Parametrize o modelo e escolha os dois eixos, x e y, para realizar o treinamento, assim como na Figura 17. Historias de usuário: US14, US16, US17;

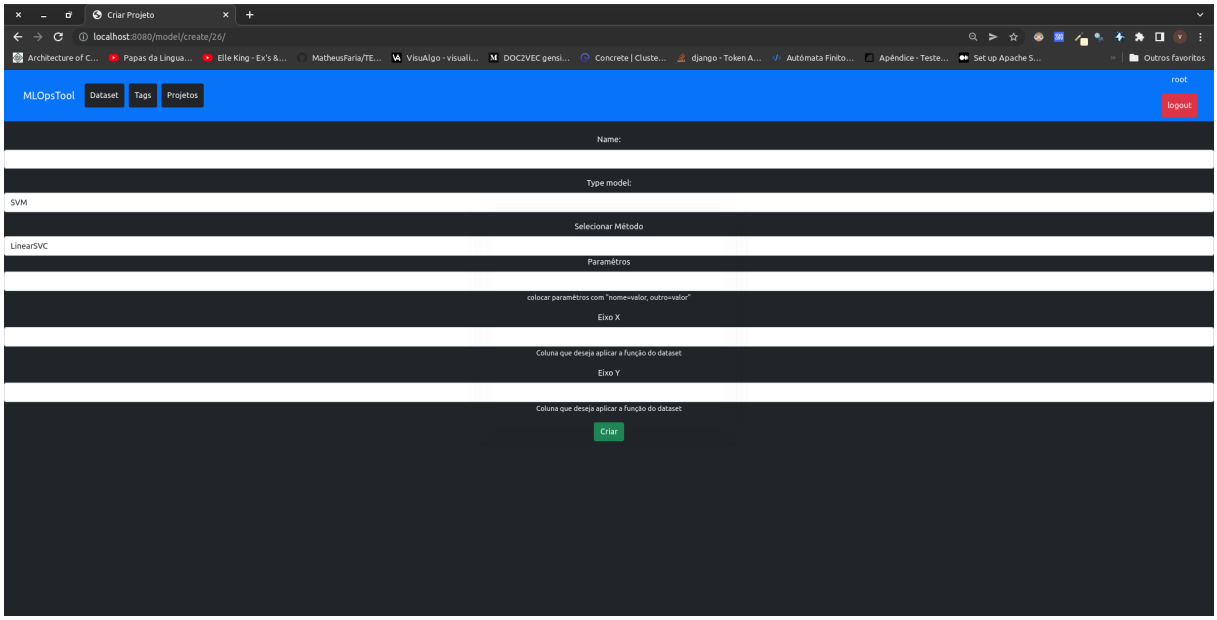

Figura 17 – Criando Modelo.

16. Volte para tela de pipeline e, nela, com todas as camadas preenchidas, pode treinar o modelo clicando no botão "treinar"no lado inferior da tela, como exposto na Figura 18. Historias de usuário: US18, US19, US20;

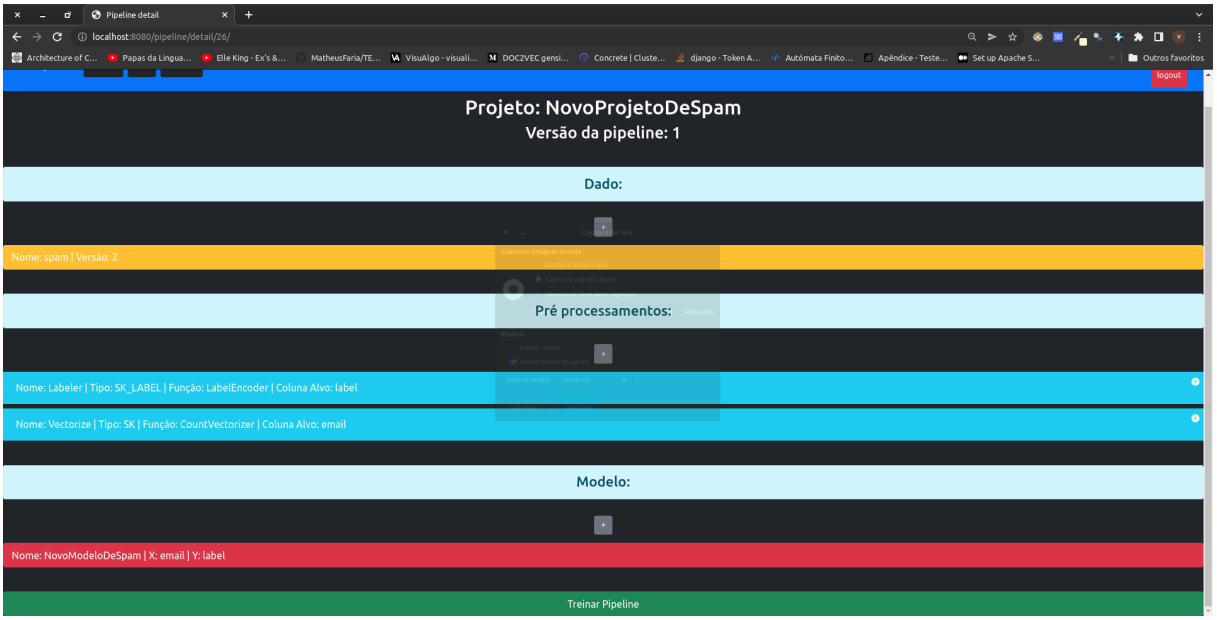

Figura 18 – Detalhes da pipeline com modelo.

17. Aperte em treinar e espere a página terminar de carregar, assim o modelo estará treinado.

18. Na tela do projeto, vai aparecer a métrica de f1-score e o f1-atual estará zerado, pois não houve nenhum reporte para o modelo, como mostra a Figura 19. Historias de usuário: US22, US21;

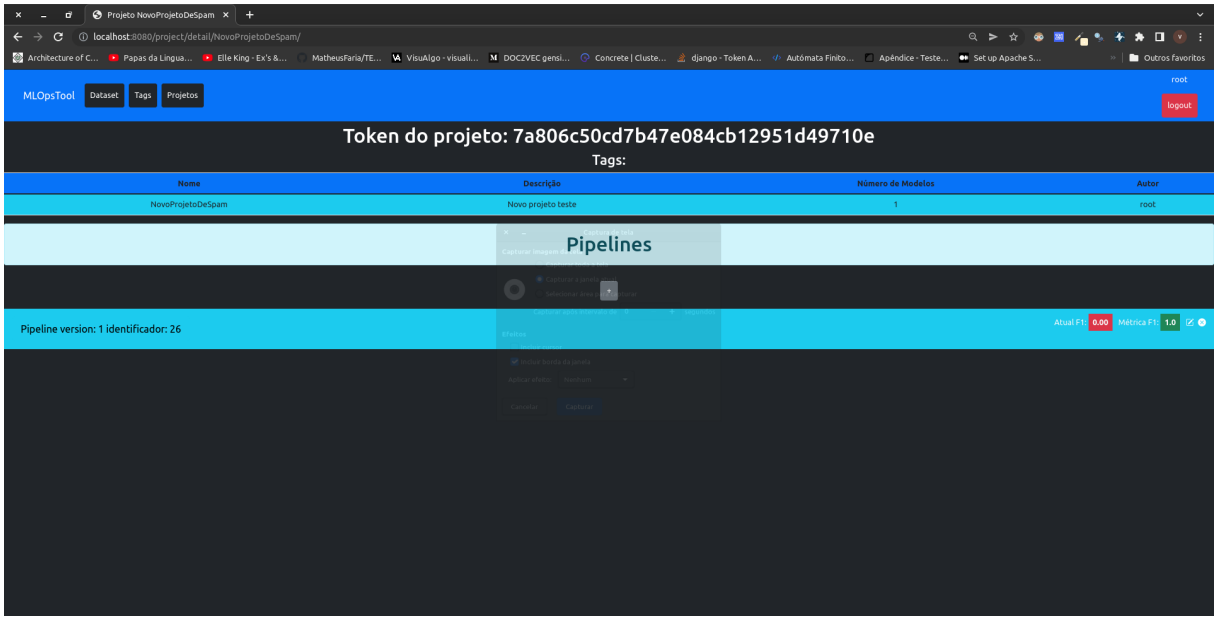

Figura 19 – Detalhes da pipeline já treinada com f1-score inicial.

19. Agora será possível fazer as classificações utilizando a API na url http://localhost:8080/model/p passando no json de requisição o token, a data e o identificador da pipeline. Assim como na Figura 20. Historias de usuário: US24, US25, US26;

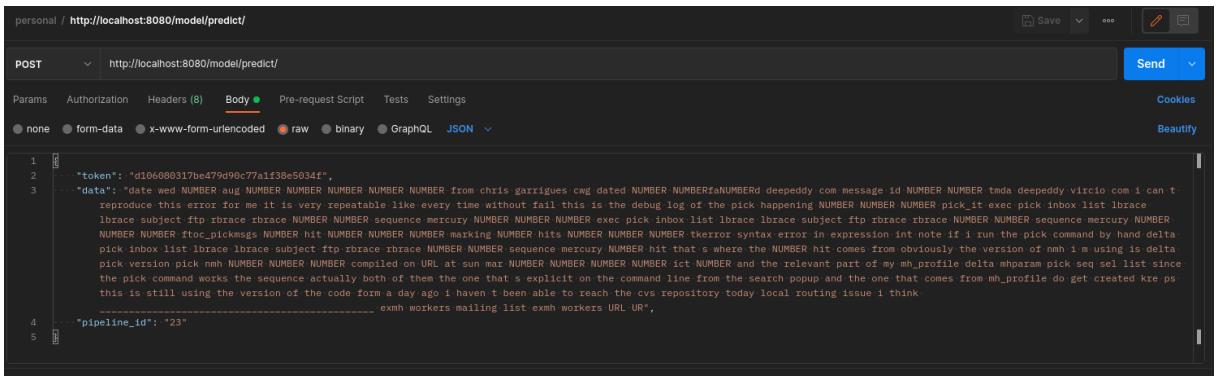

Figura 20 – Detalhes da pipeline.

20. Após fazer a requisição, será recebida a classificação e a url do local para fazer o reporte do modelo;

21. Faça o reporte, indicando na url anterior, com o campo "corret"verdadeiro, em caso de classificação certa, e errado, em classificação errada. Passe o dado que quer salvar para aquele reporte, com os campos que tem na base dados. Assim como na Figura 21. Historias de usuário: US27;

Figura 21 – Detalhes da pipeline.

6 Conclusão

6.1 Trabalhos futuros

6.1.1 Dívidas técnicas

Como dito anteriormente, as dívidas técnicas, como a utilização de scripts customizados para serem adicionados tanto na camada de pré processamento, quanto na camada de modelo, são exemplos de evoluções da ferramenta para maior flexibilidade dela de resolver problemas de classificação de texto.

6.1.2 Novas bibliotecas

Outro fator importante para se ressaltar, é que estamos lidando com apenas duas das bibliotecas principais quando se fala de aprendizado profundo e de máquina. Porém, há várias outras bibliotecas que não estão abarcadas e que podem ser evoluções para a ferramenta. Alguns exemplo são: pytorch, xgboost.

6.1.3 Métricas

A ferramenta está lidando apenas com a métrica de f1-score, o que é muito pouco comparado aos diversos problemas existentes na área. Porém, é possível realizar a evolução da ferramenta para funcionar com qualquer métrica que deseje para o problema.

6.1.4 Retreinamento automático

Atualmente, o retreinamento dos modelos são feitos de forma manual, tendo em vista que se tem a saúde do modelo em produção, entretanto, é interessante que se tenha esse retreinamento de maneira automática, sem que haja a intervenção de uma pessoa para fazer uma nova pipeline.

6.2 Cronograma de execução

Para facilitar a visualização dos próximos passos a serem seguidos, terá a seguir o cronograma do TCC 1 e TCC 2, com o objetivo de separar as histórias a serem executadas por épicos, baseados nos pontos propostos no trabalho. Ou seja, cada ponto proposto terá histórias associadas a ele. Dessa forma, tenha se o seguinte cronograma.

Tabela 5 – Cronograma TCC 1

Atividade	Descrição	Tempo	Concluído
Definir tema	Definir tema do trabalho e especifica-lo	15/07/2021 - 20/06/2021	Sim
Construção de um survey	Construção de um survey para especificação do trabalho	13/08/2021 - 27/08/2021	Sim
Escrita da introdução	Escrever a introdução do trabalho	28/08/2021 - 23/09/2021	Sim
Escrita dos requisitos	Escrever a parte de requisitos do desenvolvimento	24/09/2021 - 03/10/2021	Sim
Escrita da ambientação do projeto	Escrever e definir todas as tecnologias e justificativas para elas no trabalho	04/10/2021 - 14/10/2021	Sim
Escrita dos pontos propostos	Descrever todos os pontos propostos. Descrição do problema e execução da solução na ferramenta	15/10/2021 - 21/10/2021	Sim
Perda do trabalho	Overleaf corrompeu o projeto fazendo ter a perda de mais de 80 % do trabalho escrito	21/10/2021	Sim
Reescrita desenvolvimento do projeto	Reescrita de todas as partes desenvolvidas após a introdução que foram perdidas no dia 21/10/2021	22/10/2021 - 05/11/2021	Sim
Apresentar a banca	Apresentar TCC 1 a banca	11/11/2021	Sim

Para facilitar a leitura da execução das tarefas proposta por cada sprint do projeto. A seguinte tabela foi criada vinculando o épico com as histórias de usuário que foram desenvolvidas em cada um deles. Para facilitar será usado o identificador único da Tabela 4.

Tabela 6 – Tabela de associação épico e história de usuário.

Épico	História de Usuário associada
Camada de dados	US01, US02, US03, US07, US08
Desenvolver projetos	US04, US05, US06
Camada de Pré processamento	US10, US11, US13
Camada de Modelagem	US14, US16, US17
Camada de deploy	US18, US19, US20
Pipeline	US21, US22, US23
Sistema de parecer	US24, US25, US26, US27

Para Facilitar a leitura foi usado apenas os épicos relacionado as histórias de usuário.

Tabela 7 – Cronograma TCC 2

Atividade	Descrição	Tempo	Concluído
Camada de dados	Desenvolver parte da camada de dados	17/01/2022 - 31/01/2022	Sim
Desenvolver projetos	Desenvolver a funcionalidade de projetos	01/02/2022 - 08/02/2022	Sim
Camada de Pré processamento	Desenvolver camada de pré processamento	09/02/2022 - 23/02/2022	Sim
Camada de Modelagem	Desenvolver camada de modelagem com sklearn e tensorflow	24/02/2022 - 11/03/2022	Sim
Camada de deploy	Desenvolver deploy de API dos modelos	12/03/2022 - 26/03/2022	Sim
Pipeline	Integração de todas as outras camadas na pipeline	27/03/2022 - 13/04/2022	Sim
Sistema de parecer	Sistema de parecer do usuário cliente e atualização dos dados e métricas	14/04/2022 - 28/04/2022	Sim
Revisão do trabalho e testes do sistema	Revisão do trabalho e testes do sistema	29/04/2022 - 30/04/2022	Sim
Apresentação TCC 2	Apresentação do TCC 2	01/05/2022 - 09/05/2022	Sim

6.3 Considerações finais

Portanto, foi possível fazer ferramenta MLOpsTool abarcando todos os pontos que propostos e somando a questão do monitoramento da saúde do modelo em produção de forma automatizada. Assim, contribuindo para os desenvolvedores retreinarem ou refazerem as modelagem para que a solução consiga chegar no seu objetivo de assertividade.

Com o tempo médio de conclusão das tarefas do backlog sendo de 3 dias e havendo 27 tarefas propostas levou 81 dias para a conclusão do desenvolvimento do projeto. Assim, está dentro do prazo do calendário letivo de 90 dias. Contudo, há muito espaço para evoluções na ferramenta para torna-la cada vez mais robusta e competitiva frente as outras ferramentas citadas que tem mais tempo de maturidade. O projeto se encontra em ml-ops-tool [1]. E se encontra de forma pública para colaboração.

[1] ml-ops-tool: https://gitlab.com/victorhdcoelho/ml-ops-tool

Referências

AIMULTIPLE. *Compare Data Science / ML / AI Platforms*. Disponível em: <https://aimultiple.com/ai-platform>. Citado na página 24.

AIRFLOW. *Apache Airflow Documentation*. Disponível em: <https://airflow.apache.org/docs/apache-airflow/stable/index.html>. Citado na página 24.

ALFLEN, N. C.; PRADO, E. P. V. Técnicas de elicitação de requisitos no desenvolvimento de software: uma revisão sistemática da literatura. 2020. Disponível em: <https://brapci.inf.br/index.php/res/download/155722>. Citado na página 30.

ANDERSON, D. J.; CARMICHAEL, A. Essential kanban condensed. In: . [S.l.: s.n.], 2016. p. 5–20. Citado na página 27.

BIRD, M. S. *Utilizing agile software development as an effective and efficient process to reduce development time and maintain quality software delivery*. [S.l.]: Capella University, 2010. Citado na página 29.

BRACKETT, J. W. Software requirements. 1990. Disponível em: <https://apps.dtic.mil/sti/citations/ADA235642>. Citado na página 29.

COHN, M. *User stories applied: For agile software development*. [S.l.]: Addison-Wesley Professional, 2004. Citado na página 30.

COLUMBUS, L. 10 charts that will change your perspective on artificial intelligence's growth. 2018. Disponível em: <https://www.forbes.com/sites/louiscolumbus/2018/01/12/10-charts-that-will-change-your-perspective-on-artificial-intelligences-growth/?sh=46d99e4e4758>. Citado 3 vezes nas páginas 11, 19 e 20.

COVEYDUC, J. L.; ANDERSON, J. L. Artificial intelligence for business: A roadmap for getting started with ai. In: . [s.n.], 2020. p. 16–18. Disponível em: <https://play.google.com/books/reader?id=FSvfDwAAQBAJ&pg=GBS.PT16.w.4.0.0&hl=pt>. Citado 3 vezes nas páginas 20, 23 e 33.

DEPARTMENT, S. R. *Artificial intelligence software market revenue worldwide 2018-2025*. Disponível em: <https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/>. Citado na página 19.

DJANGO. *Django documentation contents*. Disponível em: <https://docs.djangoproject.com/en/4.0/contents/>. Citado na página 35.

DOCKER. *Docker Documentation*. Disponível em: <https://docs.docker.com/>. Citado na página 35.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering and beyond: Trends and challenges. In: *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2014. (RCoSE 2014), p. 1–9. ISBN 9781450328562. Disponível em: <https://doi.org/10.1145/2593812.2593813>. Citado na página 36.

GITLAB. *About us*. Disponível em: <https://about.gitlab.com/company/>. Citado na página 27.

GITLAB. *GitLab Docs*. Disponível em: <https://docs.gitlab.com/ee/>. Citado 2 vezes nas páginas 35 e 36.

GLASSDOOR. *50 Best Jobs in America for 2022*. 2020. Disponível em: <https://www.glassdoor.com/List/Best-Jobs-in-America-LST_KQ0,20.htm>. Citado na página 19.

GROUP, T. P. G. D. *PostgreSQL 14.2 Documentation*. Disponível em: <https://www.postgresql.org/docs/current/index.html>. Citado 2 vezes nas páginas 35 e 43.

IEEPEDUCACAO. *O que é backlog?* Disponível em: <https://www.ieepeducacao.com.br/o-que-e-backlog/>. Citado na página 27.

JAIN, A. et al. Overview and importance of data quality for machine learning tasks. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. New York, NY, USA: Association for Computing Machinery, 2020. (KDD '20), p. 3561–3562. ISBN 9781450379984. Disponível em: <https://doi.org/10.1145/3394486.3406477>. Citado na página 21.

JUNIOR, M. L.; FILHO, M. G. Variations of the kanban system: Literature review and classification. *International Journal of Production Economics*, Elsevier, v. 125, n. 1, p. 13–21, 2010. Citado na página 27.

KNIME. *Software Overview*. Disponível em: <https://www.knime.com/software-overview>. Citado na página 24.

LUCIDCHART. *O que é um diagrama de sequência UML?* 2021. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-sequencia-uml>. Citado na página 32.

MÄKINEN, S. et al. Who needs mlops: What data scientists seek to accomplish and how can mlops help? *CoRR*, abs/2103.08942, 2021. Disponível em: <https://arxiv.org/abs/2103.08942>. Citado na página 22.

MANIFESTO para Desenvolvimento Ágil de Software. [S.l.], 2001. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Citado na página 27.

METAFLOW. *What is metaflow ?* Disponível em: <https://docs.metaflow.org/introduction/what-is-metaflow>. Citado na página 24.

MIRANDA, E. Moscow rules: A quantitative exposé. Citado na página 30.

MLFLOW. *Docs*. Disponível em: <https://mlflow.org/docs/latest/index.html>. Citado na página 24.

MODELER, I. S. *IBM SPSS Modeler: Resource library*. Disponível em: <https://mlflow.org/docs/latest/index.html>. Citado na página 24.

RASCHKA, S.; MIRJALILI, V. Python machine learning: Machine learning and deep learning with python. *Scikit-Learn, and TensorFlow. Second edition ed*, v. 10, p. 3175783, 2017. Citado na página 35.

RSTUDIO. *About Rstudio*. Disponível em: <https://www.rstudio.com/about/>. Citado na página 24.

RUVOLO, P.; EATON, E. ELLA: An efficient lifelong learning algorithm. In: DASGUPTA, S.; MCALLESTER, D. (Ed.). *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, Georgia, USA: PMLR, 2013. (Proceedings of Machine Learning Research, 1), p. 507–515. Disponível em: <https://proceedings.mlr.press/v28/ruvolo13.html>. Citado na página 40.

SCHWABER, J. S. K. *Guia do Scrum*. [S.l.]: Scrum.Org and ScrumInc, 2013. Citado na página 27.

SCHWABER, K.; SUTHERLAND, J. The scrum guide. *Scrum Alliance*, v. 21, n. 1, 2011. Citado na página 27.

SHVETS, A. *Command*. Disponível em: <https://refactoring.guru/pt-br/design-patterns/command>. Citado na página 43.