FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS

FATEC PROFESSOR Jessen Vidal

LUAN MATHEUS SATIRO DE OLIVEIRA

NLEARN (PLATAFORMA DE APRENDIZADO)

Orientador: Fabricio Galende Marques de Carvalho

São José dos Campos

2020

SUMÁRIO

[1 Introdução 3](#_Toc49841023)

[1.1 Definição do problema 3](#_Toc49841024)

[1.2 Objetivo 4](#_Toc49841025)

[1.3 Escopo 4](#_Toc49841026)

1.4 Ferramental tecnológico 6

[2 Desenvolvimento 7](#_Toc49841027)

[2.1 Arquitetura 7](#_Toc49841028)

[2.2 Modelo de Dados 13](#_Toc49841029)

[2.3 Processo dinamico 17](#_Toc49841030)

[3 Resultados e Discussão 20](#_Toc49841031)

# Introdução

O problema abordado neste trabalho consiste em uma dificuldade dos novos ingressantes nos meios de maratonas de programação, assim como uma falta de espaço exclusivo para monitores. O objetivo consiste em fornecer uma maneira alternativa e direta de encontrar conteúdo para as resoluções dos exercícios, assim como disponibilizar um espaço exclusivo para os monitores adicionarem conteúdo. Com base nos tópicos apresentados a seguir, é possível visualizar uma contribuição para a comunidade de maratonas de programação, auxiliando e facilitando o trajeto de novos ingressantes.

Este projeto foi desenvolvido no escopo da disciplina laboratório de engenharia de software alinhado ao trabalho de graduação do curso de análise e desenvolvimento de sistemas da FATEC São José dos Campos.

## Definição do problema

As maratonas de programação são eventos realizados por instituições de ensino da área de TI. O intuito é prover uma competição onde o ganhador será aquele que acertar mais questões fornecidas pela banca.

Os monitores são alunos que obtiveram uma nota alta ao passar em determinada matéria e optam por auxiliar os novos alunos, fornecendo um tempo dedicado para monitoria na faculdade na qual estudam.

Um problema típico consiste na dificuldade de se localizar conteúdo específico para resoluções de determinados exercícios de programação. Assim como um espaço para monitores de matérias relacionadas ao curso de TI, conseguirem adicionar conteúdo para os alunos, pois não existe um local exclusivo para esta tarefa. Dessa forma, os monitores podem avaliar conteúdos pertinentes de determinada matéria e adicionar na plataforma para que os alunos possam visualizar aquelas informações e terem um melhor entendimento de determinado assunto.

As plataformas como URI e Ucoderfornecem exercícios de maratonas de programação e um sistema que recebe a resolução para verificar se a resposta é válida baseado no modelo de maratona. As plataformas também fornecem um fórum para conversar sobre dificuldades encontradas. No entanto, o foco do fórum é para esclarecer dúvidas especificas das resoluções e não de fato sobre o caminho necessário para chegar em determinada resolução.

Se situar no conteúdo de algoritmo de uma linguagem de programação pode ser extenso. Inicialmente, isso pode gerar uma dificuldade e criar um desencorajamento para o aluno em prosseguir neste caminho.

Um exemplo prático e simples: existe um algoritmo para resolver determinado problema. No entanto, o aluno não possui o conhecimento da existência do mesmo e não teve um direcionamento eficiente. Dessa forma, o aluno irá perder horas para realizar a resolução do exercício de uma forma ‘’alternativa’’ e isso tornará o exercício bem mais complexo. Dessa forma, o aluno se sentirá frustrado em não conseguir encontrar a resolução adequada.

Esse trabalho foca na solução dos problemas anteriormente citados.

## Objetivo

O objetivo desse trabalho é desenvolver um sistema web para aprendizado separado por tópicos, com guias provenientes da própria documentação da linguagem ou conteúdos criados pelos monitores de programação, orientando as etapas de aprendizado para conseguir resolver problemas de maratona de programação ou algo relacionado.

## Escopo

O sistema possuirá uma página administrativa na qual se encarregara de adicionar, alterar ou remover informações. Nessas funcionalidades haverá duas seções: adicionar linguagem e adicionar conteúdo para a linguagem. Para a seção linguagem os campos referentes são adicionar título, descrição, imagem e key. Os campos da seção conteúdo serão o título, subtítulo de um conteúdo, conteúdo, exercícios, key do tópico e key da linguagem.

Somente os monitores do sistema poderão realizar essas alterações e cada monitor só pode alterar o conteúdo que ele mesmo tenha postado. Também haverá um usuário administrador que além de poder também inserir, alterar ou deletar linguagens ou conteúdos, poderá deletar monitores do sistema. Ao deletar uma linguagem isso também deletará todo o conteúdo atribuído a ela.

Funcionalidades:

1. Exclusão de um monitor: o usuário administrador pode excluir monitores do sistema;
2. Cadastrar nome do monitor: O administrador poderá adicionar um nome para o monitor;
3. Cadastrar senha do monitor: O administrador poderá cadastrar uma senha para o monitor;
4. Cadastrar contato do monitor: O administrador poderá adicionar o contato do monitor;
5. Adicionar título da linguagem: O monitor pode adicionar o título de uma linguagem pela pagina administrativa;
6. Adicionar descrição da linguagem: O monitor pode adicionar a descrição de uma linguagem pela pagina administrativa;
7. Adicionar imagem da linguagem: O monitor pode adicionar uma imagem para linguagem pela pagina administrativa;
8. Adicionar a key da linguagem: O monitor pode adicionar a key de uma linguagem pela pagina administrativa;
9. Alterar título da linguagem: O monitor pode alterar o título de uma linguagem pela pagina administrativa;
10. Alterar descrição da linguagem: O monitor pode alterar a descrição de uma linguagem pela pagina administrativa;
11. Alterar imagem da linguagem: O monitor pode alterar uma imagem para linguagem pela pagina administrativa;
12. Alterar a key da linguagem: O monitor pode alterar a key de uma linguagem pela pagina administrativa;
13. Excluir linguagem: O monitor pode excluir a linguagem do sistema;
14. Adicionar título do conteúdo: O monitor pode adicionar o título de um post pela pagina administrativa;
15. Adicionar subtítulo do conteúdo: O monitor pode adicionar o subtítulo de um post pela pagina administrativa;
16. Adicionar conteúdo: O monitor pode adicionar o conteúdo de um post pela pagina administrativa;
17. Adicionar exercício: O monitor pode adicionar o exercício de um post pela pagina administrativa;
18. Adicionar key do tópico do conteúdo: O monitor pode adicionar a key do tópico de um post pela pagina administrativa;
19. Adicionar key da linguagem para atribuir ao conteúdo: O monitor pode adicionar a key da linguagem de um post pela pagina administrativa;
20. Alterar título do conteúdo: O monitor pode alterar o título de um post pela pagina administrativa;
21. Alterar subtítulo conteúdo: O monitor pode alterar o subtítulo de um post pela pagina administrativa;
22. Alterar conteúdo: O monitor pode alterar o conteúdo de um post pela pagina administrativa;
23. Alterar exercício: O monitor pode alterar o exercício de um post pela pagina administrativa;
24. Alterar key do tópico do conteúdo: O monitor pode adicionar a key do tópico de um post pela pagina administrativa;
25. Alterar key da linguagem para atribuir ao conteúdo: O monitor pode adicionar a key da linguagem de um post pela pagina administrativa;
26. Excluir conteúdo: O monitor pode excluir um conteúdo da página administrativa.

Ilustrar preferencialmente com um diagrama de caso de uso.

## Ferramental tecnológico

Nesse trabalho foram utilizas as seguintes tecnologias:

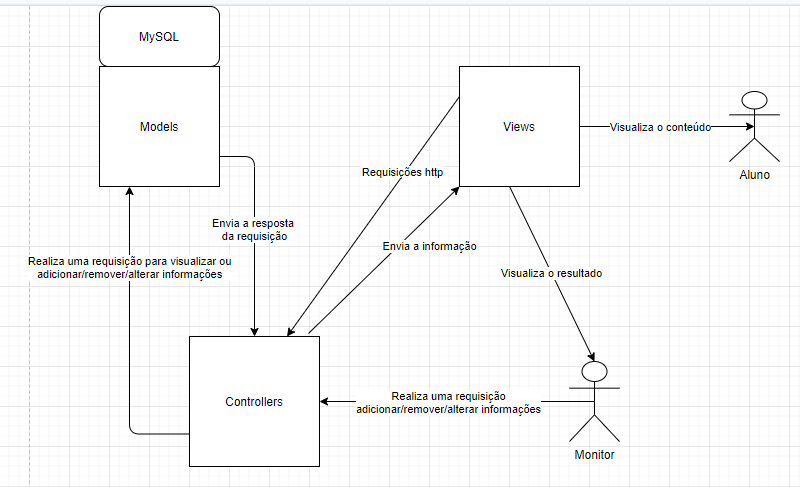
1. Python no Back-end;
2. Jinja2 para geração de páginas dinâmicas;
3. Gunicorn como servidor de implantação, em conjunto com o Flask;
4. SqlAlchemy para persistência de dados;
5. Flask : micro framework para desenvolvimento web;
6. Bootstrap framework para o front-end;
7. Html5 para estruturar a página;
8. Css3 para estilizar a página;
9. Virtualenv para isolamento de ambiente de desenvolvimento e obtenção de pacotes;
10. Javascript para funcionalidades voltadas ao front-end.

# MODELAGEM

Este Capítulo apresenta diferentes modelos e código-fonte representativos do sistema que foi desenvolvido para auxiliar os alunos da FATEC São José dos Campos, oferecendo uma plataforma para visualização de assuntos pertinentes aos cursos Análise e Desenvolvimento de Sistemas e Banco de Dados.

## Arquitetura

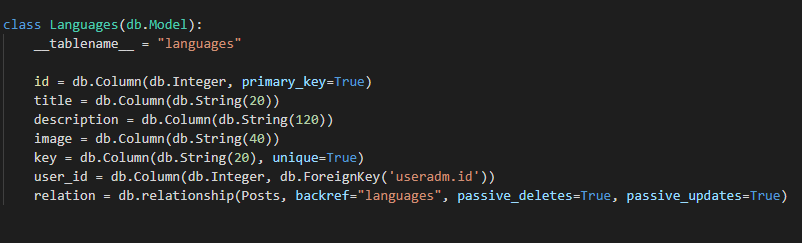
A figura seguinte ilustra a arquitetura do sistema:

**Figura 1** – Arquitetura do sistema

Neste projeto foi utilizado o formato de estrutura MVC (MVC é o acrônimo de Model-View-Controller é um padrão de projeto de software ou padrão de arquitetura de software) como apresentado na figura 1.

**Model:** É responsável pela leitura e escrita de dados e também de suas validações.

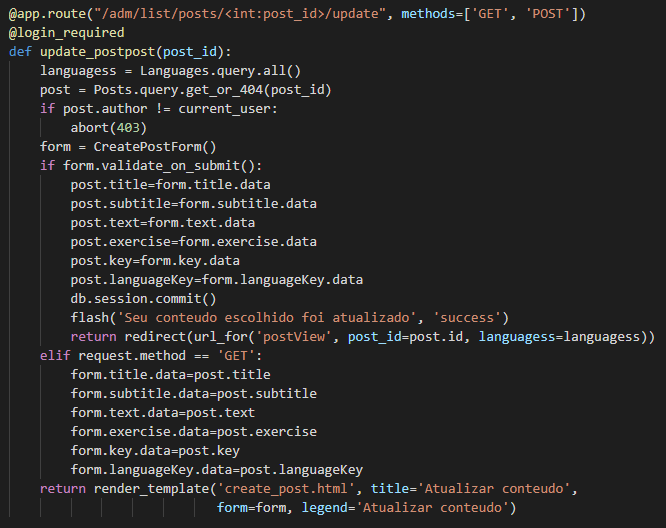
Um exemplo de modelo é ilustrado na figura 2.

**Figura 2** – Classe Languages do model

Na figura 2 é realizado a descrição da estrutura de uma das tabelas do banco. É necessário definir seus atributos, tipos e relações. Dessa forma, ao executar os comandos provenientes da tecnologia Flask, as tabelas serão criadas no banco de acordo com as especificações definidas no código.

No exemplo da figura 2 temos um id que é a chave primária e é constituído de um inteiro, após isso temos o título, descrição, imagem, key e todos esses campos são do tipo texto. Existe uma particularidade no campo key, é um atributo único, ou seja, não pode existir valores iguais em linguagens diferentes. A seguir temos uma chave estrangeira fazendo referência ao id da tabela usuário. E por fim, uma relação na qual implica que quando uma linguagem for excluída ou atualizada, todos os seus filhos também serão deletados ou atualizados.

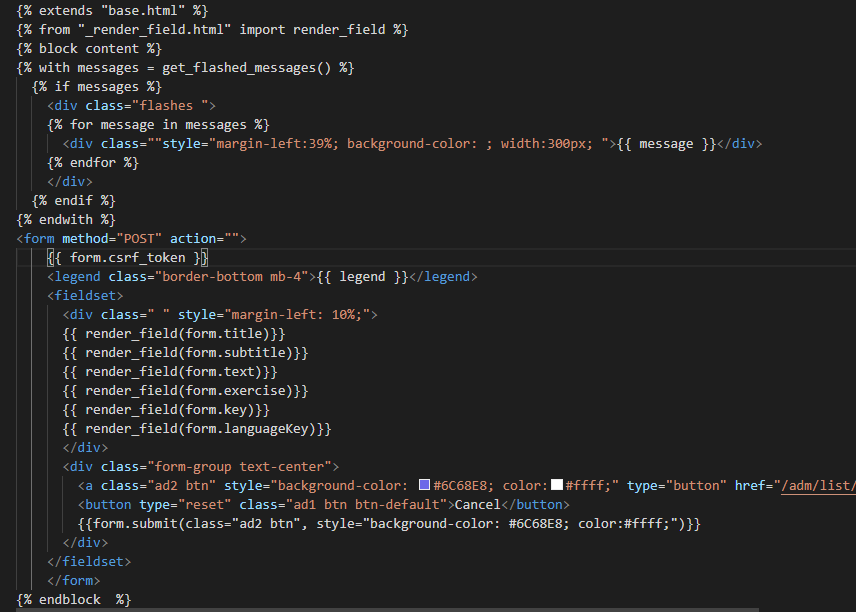
**Controller:** Um controlador é responsável por controlar a maneira como um monitor interage com uma aplicação MVC e possui o fluxo de controle lógico para uma aplicação.

**Figura 3** – Rota para atualizar um conteúdo

Na figura 3 existem dois tipos de requisições: get e post. Essas requisições são utilizadas para receber e enviar informações ao banco.

* Variável languagess: Essa variável é responsável por pegar todo o conteúdo existente na tabela Languages, para que possa ser exibido os títulos das linguagens no menu de navegação do site;
* Variável post: Essa variável é responsável por pegar o id do post que o usuário está no momento;
* Form: Essa variável é responsável por criar o formulário da classe CreatePostForm;
* Primeiro if: Esse if serve para validar se o monitor logado é o mesmo que criou aquele post, caso não seja, ele abortara a ação, pois somente quem criou o post pode o atualizar ou deletar;
* Segundo if: Esse if serve para realizar uma validação, ou seja, se o botão de finalizar for acionado, os dados abaixo serão inseridos numa variável que realizara o commit para o banco. Dessa forma, salvará as informações obtidas nos forms e se isso for feito retornara um flash. O flash exibira uma mensagem informando que a linguagem foi atualizada. O return desse if retornará para outra função na qual exibirá todos os posts existentes;
* Elif: Esse elif serve para pegar as informações existentes no banco para que o monitor visualize o conteúdo já existente para decidir o que de fato quer atualizar em cada tópico;
* Return: o return da função update\_postpost serve para que caso nenhuma condicional seja executada, retorne a página de criação de conteúdo.

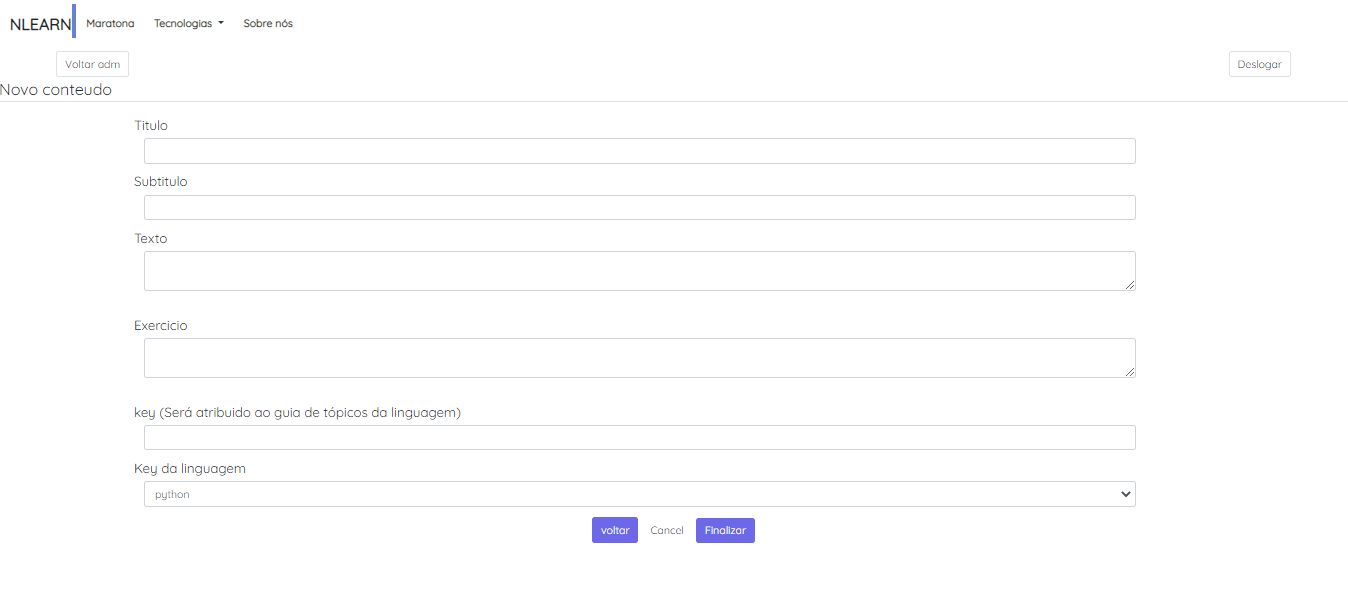
**View:** É responsável por tudo que o usuário final visualiza, toda a interface, informação.

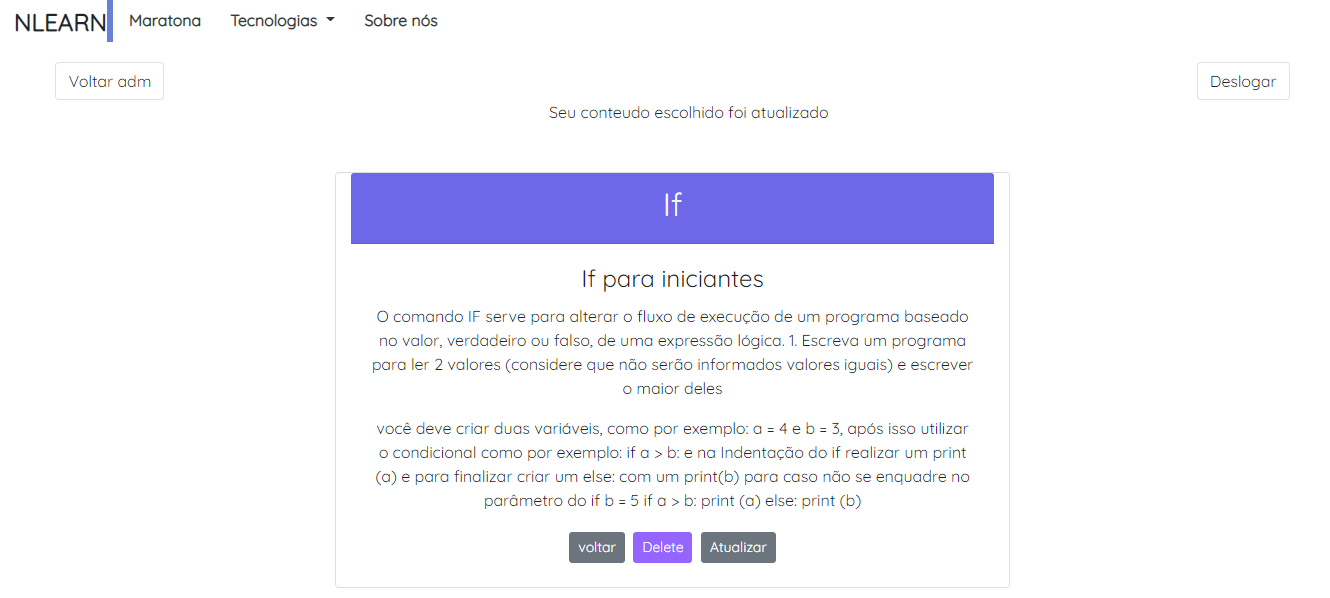
**Figura 4** – Template para criação de conteúdo

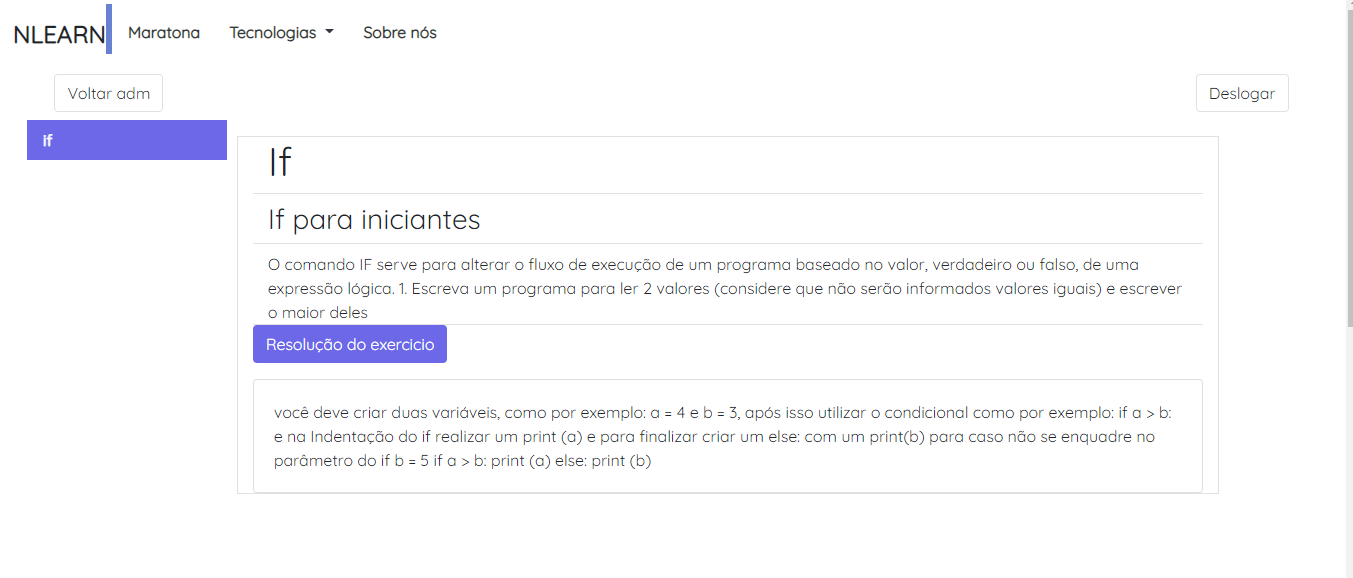
Na figura 4 é apresentado a utilização de jinja, html, css e boostrap. O parâmetro block é estendido do template base (template base é a estrutura geral do site). Após isso, existem parâmetros chamando a mensagem flash. A tag <div> (containers) com as configurações de estilo. A tag <form> serve para estabelecer as ações que possibilitaram inserir as variáveis de formulário. A tag <fieldset> é encarregada de receber as variáveis do formulário para que os monitores possam digitar e enviar essas informações aos controllers..

Nas figuras 4, 6 e 7 será apresentado o que o monitor/administrador visualizará.

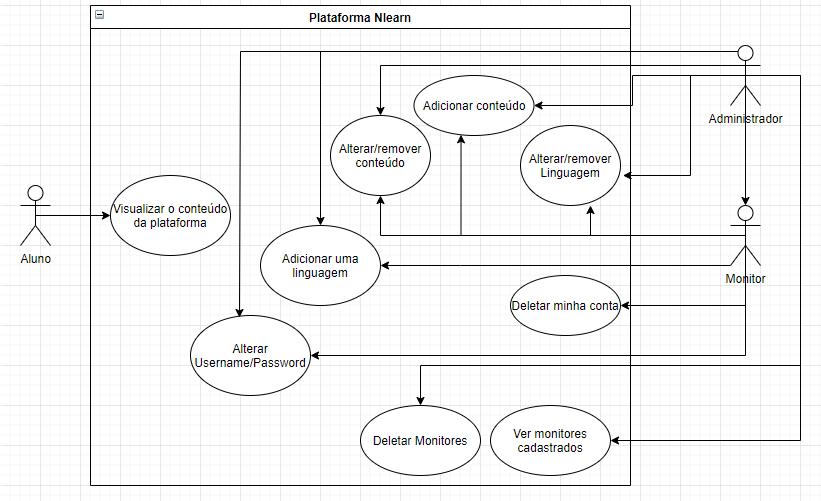
.

**Figura 5** – View para criação de conteúdo

 **Figura 6** – View de exibição de conteúdo para alterar/deletar

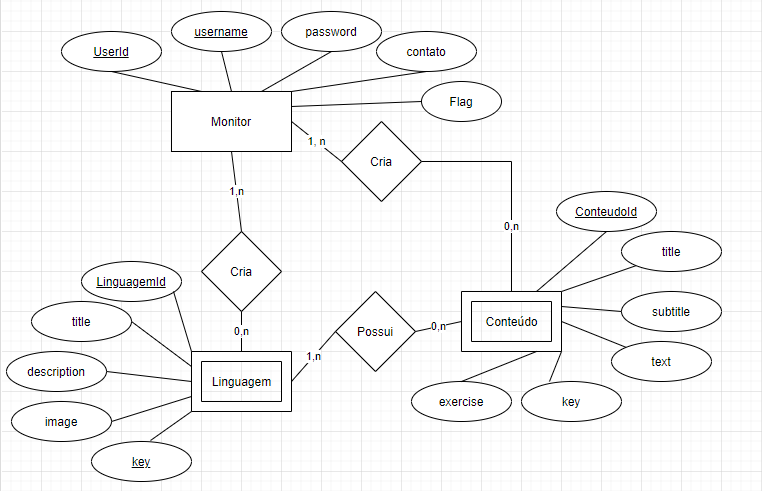
**Figura 7** – View de exibição de conteúdos cadastrados nas linguagens

## Modelo de Dados

**Figura 8** – Caso de uso

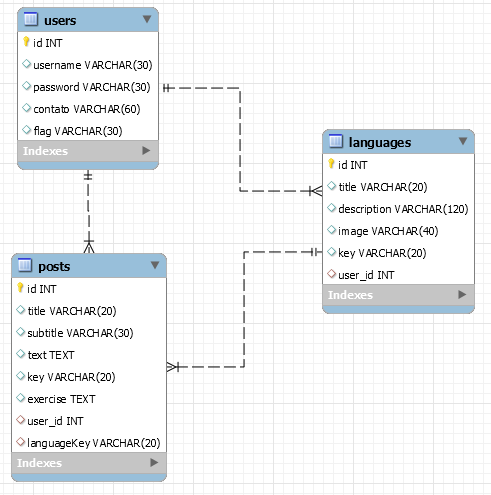
Na figura 8, existem 3 atores. O aluno poderá visualizar o conteúdo na plataforma. O monitor poderá adicionar/alterar/remover linguagens ou conteúdo para linguagens, sendo que só podem realizar ações sobre o conteúdo que criaram, assim como podem atualizar seus dados como usuário e senha ou deletar a própria conta. O administrador, tem os mesmos direitos dos monitores, com o acréscimo de poder visualizar todos os monitores cadastrados ou realizar a deleção deles. É importante salientar que o usuário administrador pertence a mesma tabela do monitor, com a diferença que o administrador tem uma flag no código onde o permite acessar uma rota para administrar monitores

Figura 9 – Diagrama entidade relacionamento (modelo conceitual)



Na figura 9 existem 3 entidades. O monitor possui 5 atributos e duas relações, sendo que UserId é sua determinante e username seu identificador único. O monitor pode criar nenhuma ou muitas linguagens, assim como pode criar nenhum ou muitos conteúdos. A seguir a linguagem possui 5 atributos e uma relação com o conteúdo, sendo que LinguagemId é sua determinante e key seu identificador único. A linguagem pode conter nenhum ou muitos conteúdos, sendo que a linguagem só pode existir se um monitor ou administrador a cadastrar. E por fim, o conteúdo com 6 atributos, sendo que ConteudoId é seu determinante. Um conteúdo só pode existir caso exista uma linguagem e um monitor/administrador. Um conteúdo pode existir em um ou nenhum monitor/administrador, assim como pode existir em uma ou nenhuma linguagem.

**Figura 10** – Modelo lógico



Na figura 10, existe três tabelas. A tabela de posts(conteúdo), languages(linguagens) e users(monitores/administrador).

Na tabela users existem quatro atributos:

* ID: É uma chave primária que serve para identificar qual é o usuário;
* Username: Para o nome;
* Password: Para fazer as validações de login na plataforma;
* Contato: Para inserir o contato do monitor;
* Flag: Para definir se o usuário poderá administrar ou não os monitores.

Na tabela posts existem 8 atributos:

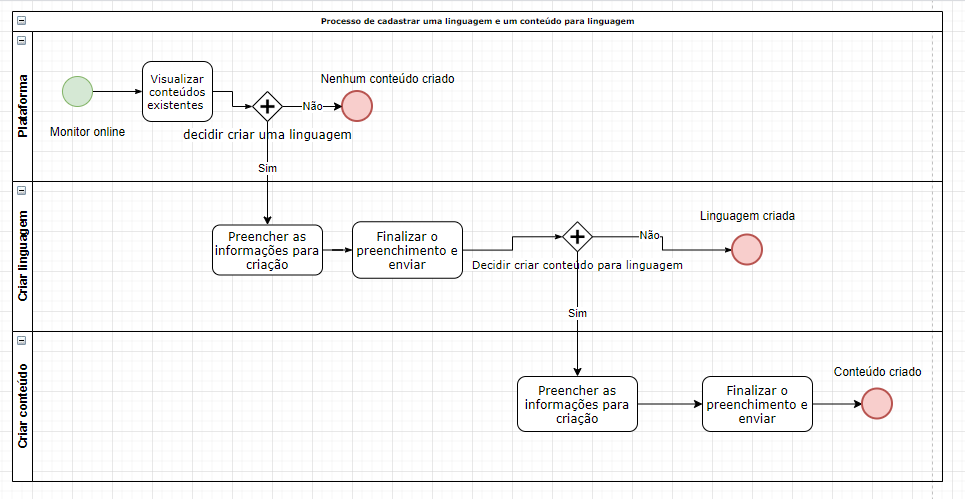
* ID: É uma chave primária que serve para identificar qual é o conteúdo;
* Title: Para inserir um título;
* Subtitle: Para inserir um subtítulo;
* Text: Para inserir explicações;
* Exercise: Para inserir a resolução do exercício;
* User\_id: É uma chave estrangeira para atribuir conteúdo para o usuário que o criou;
* LanguageKey: É uma chave estrangeira para atribuir cada post para cada linguagem escolhida.

Na tabela languages existem 6 atributos:

* ID: É uma chave primária que serve para identificar qual é o conteúdo;
* Title: Para inserir um título;
* Description: Para inserir uma descrição;
* Image: Para inserir imagens para os cards da página inicial;
* Key: Para definir qual linguagem será;
* User\_id: É uma chave estrangeira para atribuir a linguagem para o usuário que o criou.

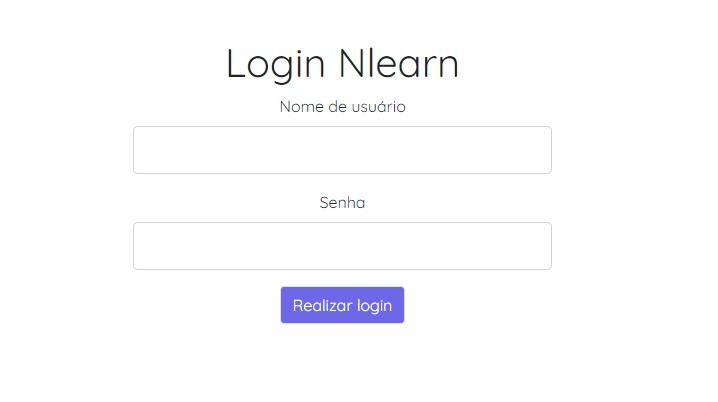
## Modelagem Dinâmica

A Figura ... ilustra (dizer qual processo do sistema é ilustrado).

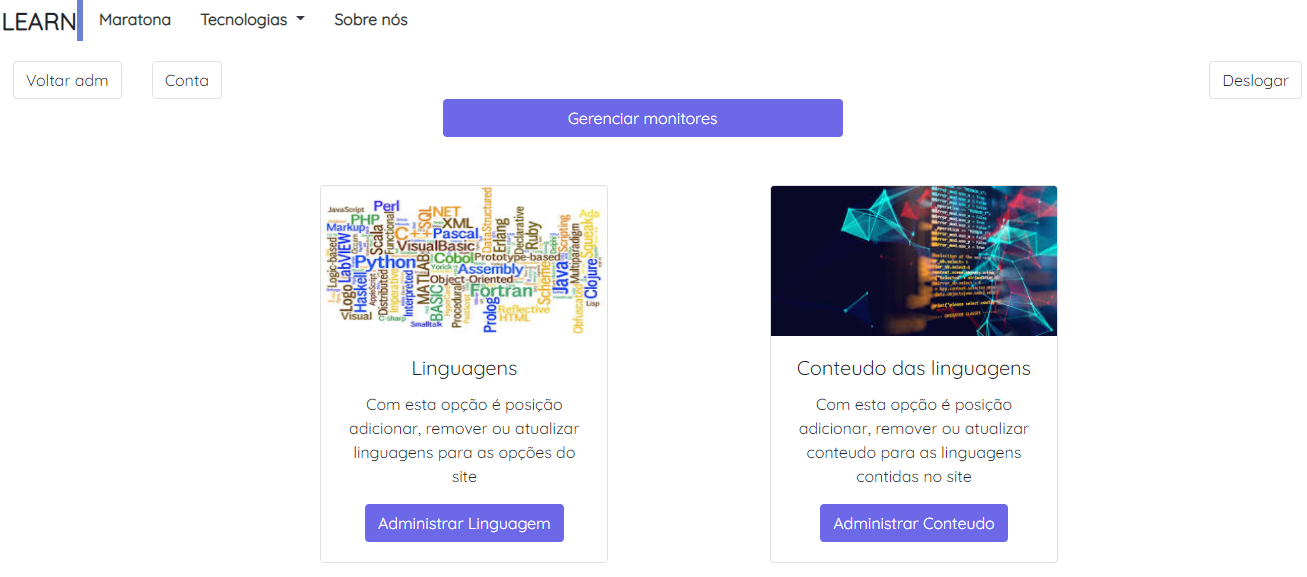
**Figura 11** – Diagrama de processo

Na figura 11 é demonstrado as ações que um monitor precisará fazer para criar uma linguagem ou um conteúdo.

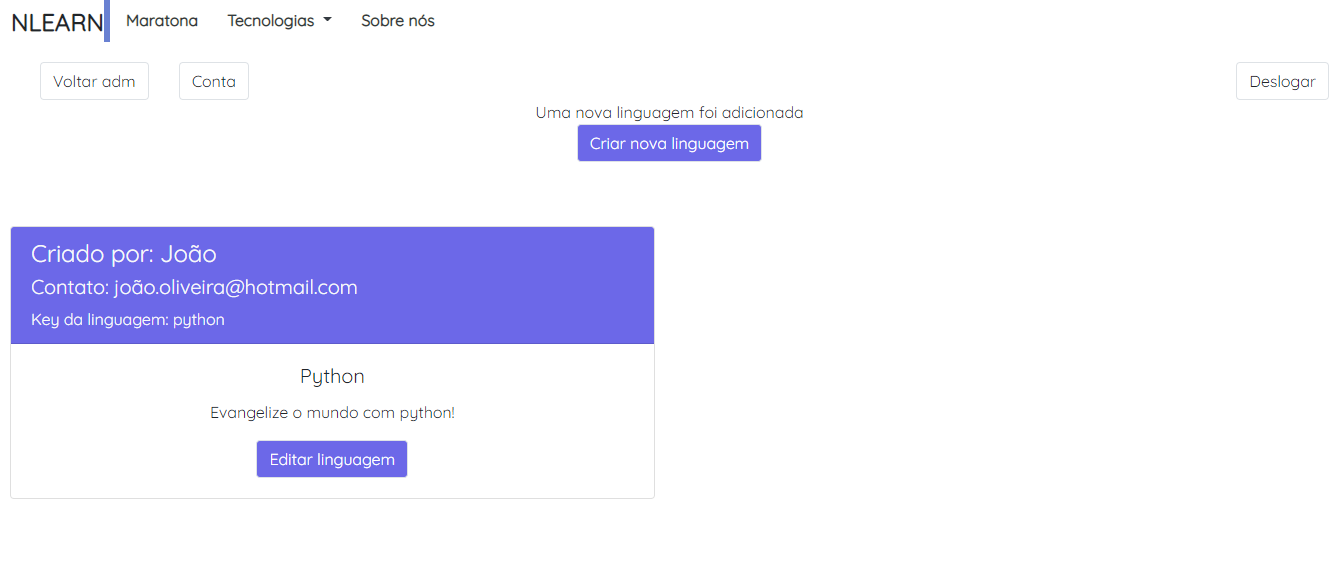
**Figura 12** – Tela de login



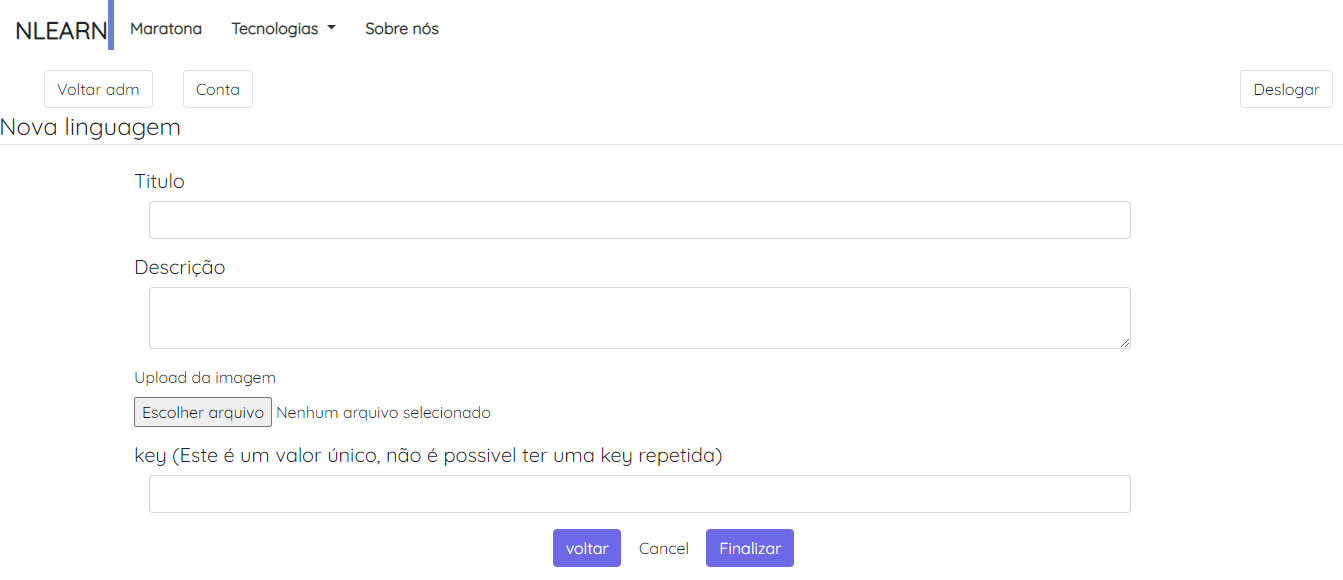
.

**Figura 13** – Tela para selecionar a seção linguagem ou conteúdo

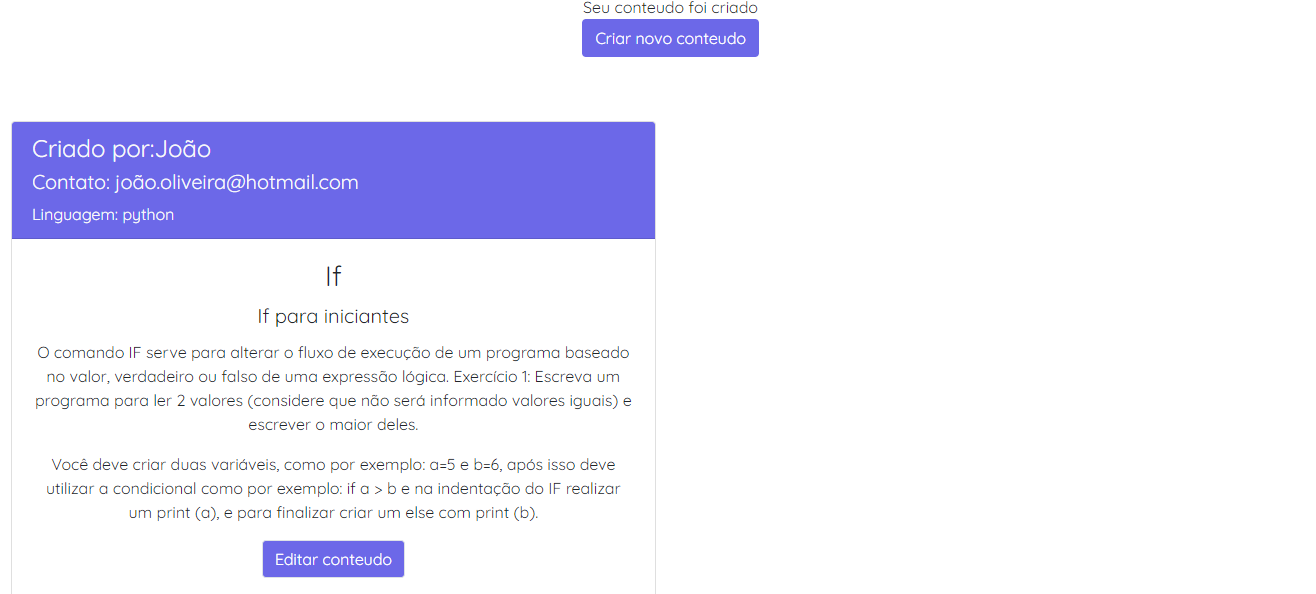
**Figura 14** – Visualizar linguagens existentes



**Figura 15** – Criar nova linguagem



**Figura 16** – Visualizar conteúdos existentes



Nas figuras 12, 13, 14, 15, 16 e 5 foi apresentado a visualização do monitor ao realizar os passos da figura 11.

* A figura 12: Consiste no primeiro passo onde o monitor irá realizar o login na plataforma;
* A figura 13: Consiste na seção onde o monitor irá escolher se quer cadastrar uma nova linguagem ou conteúdo;
* A figura 14: Consiste na seção onde o monitor consegue visualizar todas as linguagens cadastradas por ele ou por outros monitores;
* A figura 15: Consiste na seção onde o monitor encontra os campos para cadastrar uma nova linguagem;
* A figura 16: Consiste na seção onde o monitor consegue visualizar todos os conteúdos cadastrados por ele ou por outros monitores;
* A figura 5: Consiste na seção onde o monitor encontra os campos para cadastrar um novo conteúdo.

# Resultados OBTIDOS

COLOCAR AQUI AS FIGURAS E DESCREVER, UMA A UMA, E NÃO TODAS DE UMA VEZ, A MENOS QUE REPRESENTEM SEQUÊNCIA DE AÇÕES/PASSOS.

# CONSIDERAÇÕES FINAIS

O foco neste trabalho foi na utilização da linguagem Python e da tecnologia Flask. Além disso, esse trabalho foi desenvolvido com um editor de código-fonte Virtual Studio *Code*, SQLAlchemy (ORM) e o banco de dados MySQL.

Junto com o Flask, é possível utilizar o Jinja2, uma engine de templates para criação de HTML que pode ser retornado ao usuário através de requisições HTTP. O SQLAlchemy (Object Relational Mapper) é responsável por traduzir as classes em tabelas no banco de dados (MySQL).

O Flask, é um micro-framework, e por isso de certa forma simples, leve e rápido, recomendado para pequenos projetos e podendo atingir até um certo grau de complexidade, oferecendo maior flexibilidade e controle para o desenvolvedor. Escalável, mas por ser single threadedpode não apresentar uma boa performance em algumas circunstâncias.

Atualmente Python já é utilizado por grandes empresas, tais como, Netflix, Google, Spotify, Globo dentre outros. Escolhido por ser uma linguagem de programação de alto nível, imperativa e de scripts. Sem esquecer que pode ser orientado a objetos e de interpretação fácil, isso são fatores que fazem esta linguagem ser amplamente utilizada por muitas empresas.

As limitações da atual versão do sistema são ...

Como sugestões para trabalhos futuros tem-se:

* Sugestão 1;
* Sugestão N.