UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Engenharia de Software

Maria Luísa de Barros Costa Silva (2019.1906.007-3), Júlia Alves Corazza (2020.1906.021-0) e Caio Rosa Carvalho (2018.1906.038-1)

LABORATÓRIO DE BANCO DE DADOS

flask-admin-college

Trabalho prático utilizando PostgreSQL e Flask-Admin

Professora: Vanessa Araujo Borges

Data de entrega: 12/11/2021

SUMÁRIO

ESPECIFICAÇÃO DO PROBLEMA

1. Descrição Geral

O sistema acadêmico possui professores cadastrados que lecionam matérias e podem coordenar cursos. Os cursos são cursados por vários alunos. Os alunos podem cursar várias matérias.

2. Características das Entidades

Professor possui como atributos: id (identificador único de cada professor no banco, gerado automaticamente com a adição do registro no sistema), nome, cpf e e-mail. Os atributos e-mail e cpf são registros únicos, ou seja: um mesmo valor de cpf ou de email não pode se repetir no sistema. Professor se relaciona com Matéria (leciona) e com Curso (coordena).

Curso possui como atributos: id (identificador único de cada curso no banco, gerado automaticamente com a adição do registro no sistema), nome (registro único) e faculdade (instituição em que o curso é fornecido). Curso se relaciona com Professor (é coordenado por) e com Aluno (é cursado por).

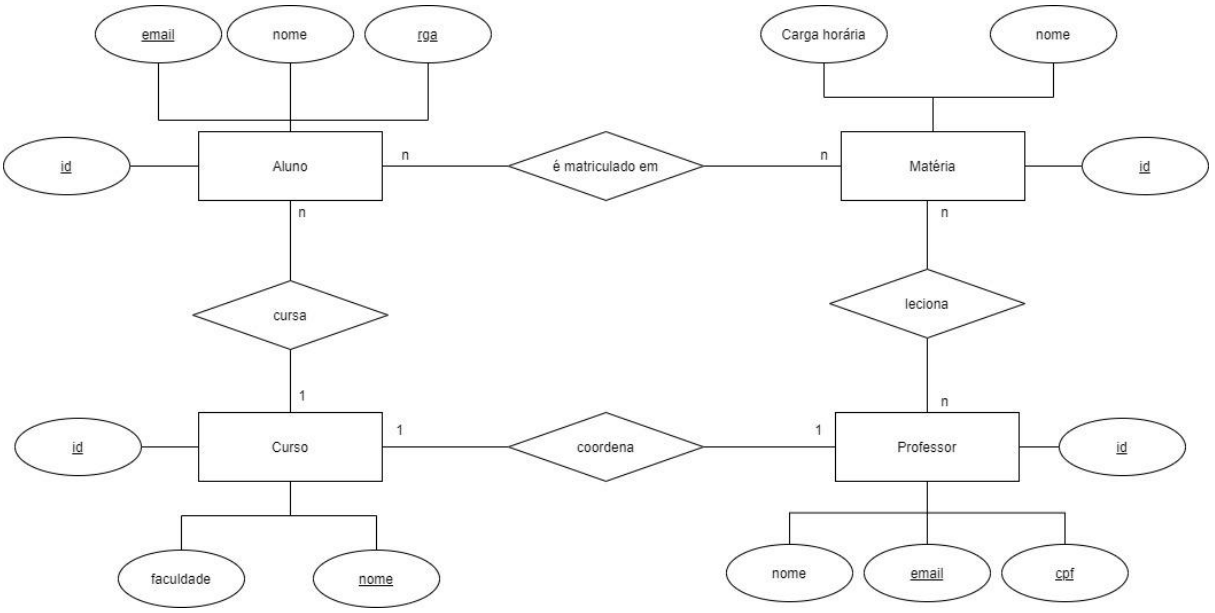
Aluno possui como atributos: id (identificador único de cada aluno no banco, gerado automaticamente com a adição do registro no sistema), nome, rga e e-mail. Os atributos e-mail e rga são registros únicos, ou seja: um mesmo valor de rga ou de email não pode se repetir no sistema. Aluno se relaciona com Matéria (é matriculado em) e com Curso (cursa).

Materia possui como atributos: id (identificador único de cada aluno no banco, gerado automaticamente com a adição do registro no sistema), nome e carga horária (o tempo estipulado como necessário para que matéria seja cumprida).Matéria se relaciona com Aluno (é matriculado em) e com Professor (é lecionada por).

3. Relacionamentos

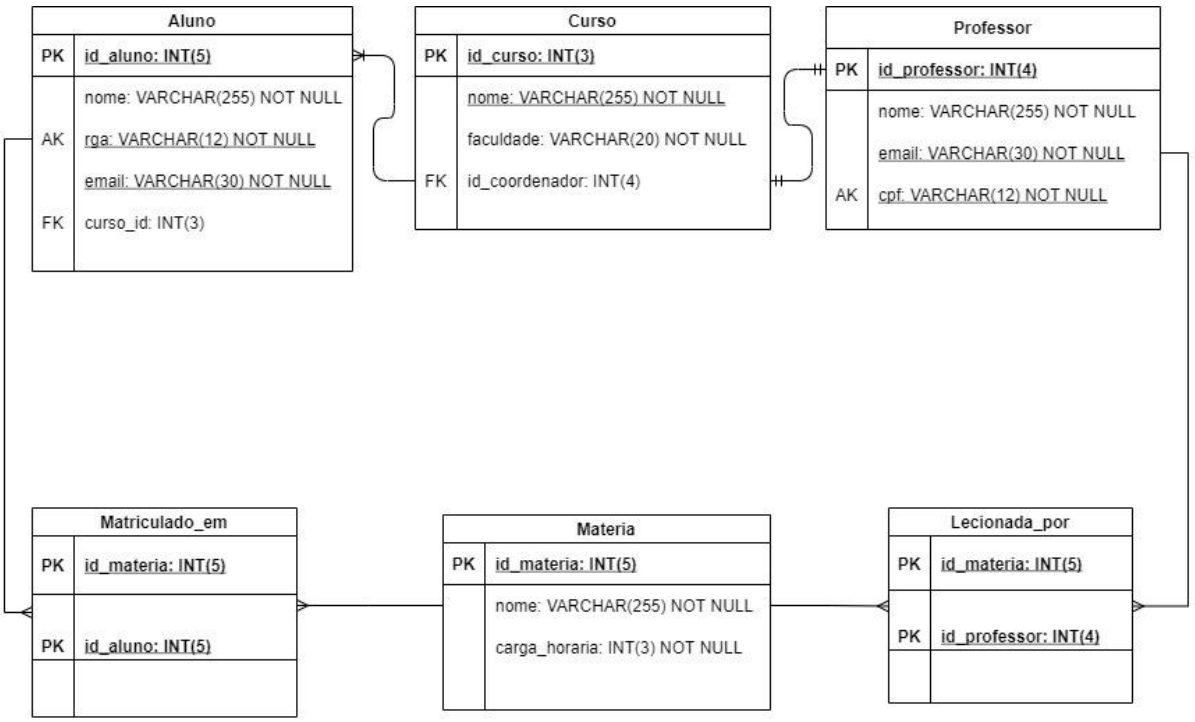
- **Professor** leciona **Materia** em um relacionamento muitos-para-muitos, ou seja, muitos professores lecionam muitas matérias.
- Um **Professor** pode coordenar no máximo um **Curso** e um **Curso** é coordenado por exatamente um **Professor**. (relacionamento um-para-um).
- Um **Curso** pode ser cursado por muitos **Alunos**, mas cada Aluno cursa exatamente um **Curso**. (relacionamento um-para-muitos).
- Vários **Alunos** podem estar matriculados em várias **Materias** (relacionamento muitos-para-muitos).

ESQUEMA CONCEITUAL (MER)



[Link para a imagem]

ESQUEMA RELACIONAL



[Link para a imagem]

Os relacionamentos muitos-para-muitos presentes no MER foram atualizados como tabelas próprias no Esquema Relacional:

- **“Lecionada_Por”** é a tabela referente ao relacionamento Leciona e sua chave primária é a junção das chaves primárias de Matéria e Professor.
- **“Matriculado_em”** é a tabela referente ao relacionamento é matriculado em e sua chave primária é a junção das chaves primárias de Matéria e Aluno.

TRIGGERS

Para a realização do trabalho, foram utilizados 4 *triggers* diferentes, os quais estão listados abaixo:

- **alunolog**: Ocorre depois de atualizações ou deleções na tabela **aluno**, utiliza a *trigger function* **alteracaolog** para inserir uma tupla na tabela **logs**;
- **cursolog**: Ocorre depois de atualizações ou deleções na tabela **curso**, utiliza a *trigger function* **alteracaolog** para inserir uma tupla na tabela **logs**;
- **professorlog**: Ocorre depois de atualizações e deleções na tabela **professor**, utiliza a *trigger function* **alteracaolog** para inserir uma tupla na tabela logs;
- **materialog**: Ocorre depois de atualizações e deleções na tabela **materia**, utiliza a *trigger function* **alteracaolog** para inserir uma tupla na tabela logs.

FUNCTION

A *trigger function* **alteracaolog** utilizada pelos *triggers*, insere uma tupla na tabela **logs** com o nome da tabela que foi alterada, o nome presente na tupla alterada, a data da alteração ou deleção e a operação realizada.

TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do presente trabalho, foram utilizadas as seguintes tecnologias:

NOME	DESCRIÇÃO
PostgreSQL	Sistema Gerenciador de Banco de Dados (SGBD) que manipula os dados inseridos no nosso sistema. (**Versão: 13.4**)
SQLAlchemy	Object-Relational Mapping (ORM) de código aberto desenvolvido para a linguagem Python distribuído sobre a licença MIT. (**Versão: 1.4.23**)
Flask	Microframework baseado em Python com o objetivo de criar páginas web. (**Versão: 2.0.1**)
Python	Linguagem de programação de alto nível. (**Versão: 3.9**)

TUTORIAL DE INSTALAÇÃO

1.1 Rodando na nuvem: Heroku

Para ter acesso a aplicação, basta clicar no link abaixo :

- **https://flask-admin-college.herokuapp.com**

Heroku é uma plataforma que permite rodar diversas aplicações na nuvem. Foi utilizada a versão gratuita da ferramenta e por isso ela pode apresentar lentidão, porém nada que impeça de utilizar a aplicação sem maiores problemas.

1.2 Rodando localmente

Para que seja possível rodar a aplicação localmente, basta seguir os passos abaixo:

1. A aplicação necessita que você tenha instalado Python 3.9 em sua máquina. Basta escolher de acordo com sua preferência entre os downloads disponíveis em https://www.python.org/downloads/ .
2. Após finalizado o download e a instalação do interpretador, a aplicação pode ser clonada na pasta de sua preferência. Pode-se utilizar o comando abaixo ou baixar o repositório zipado direto da interface do Github.

   ```
   $ git clone git@github.com:Marialuisabcs/flask-admin-college.git
   ```
3. Para rodar a aplicação, é altamente recomendável utilizar um ambiente virtual a fim de possibilitar a instalação de todas as bibliotecas necessárias. Para criar o ambiente, deve-se rodar o seguinte comando no terminal:

   ```
   $ py -3.9 -m venv nome_do_seu_ambiente
   ```
4. Assim que criado, o ambiente pode ser ativado com o comando:

   ```
   $ cd nome_do_seu_ambiente
   $ Scripts\activate
   ```
5. Para instalar as dependências:

   ```
   $ pip install -r requirements.txt
   ```

Obs: Note que você estará dentro do diretório do ambiente virtual criado.Para retornar a pasta raíz do projeto, rode o comando:

```
$ cd ..
```

6. Após instaladas corretamente todas as dependências, pode-se rodar a aplicação. Em função do uso do micro framework Flask, será usada a CLI disponível para executar o app.

```
$ flask run
```

Obs: Por padrão, o flask utiliza a porta http://127.0.0.1:5000/ .

LINKS ÚTEIS

Pasta do Google Drive contendo todo o conteúdo do trabalho:

- https://drive.google.com/drive/folders/1qbRwUv1M9sXJxlJcgzcJZCe1X4-Rr7hy?usp=sharing

Github do trabalho:

- https://github.com/Marialuisabcs/flask-admin-college.git