



Flask

Programação Web III

Prof. Diego Max

Flask

Aula 04: CRUD com o banco de dados MySQL

Exibindo, cadastrando, alterando e deletando dados

Flask

Tópico 04.1: Criação do banco e leitura de dados *CRUD com o banco de dados MySQL*

O **PyMySQL** é uma biblioteca em Python que fornece uma interface para se conectar e interagir com bancos de dados **MySQL** usando Python. Com o **PyMySQL**, você pode executar **consultas SQL**, inserir dados, atualizar registros e realizar outras operações de banco de dados diretamente de dentro do seu código Python.

Essa biblioteca facilita a comunicação entre o Python e o **MySQL**, permitindo que os desenvolvedores criem aplicativos e scripts que manipulem dados armazenados em bancos de dados **MySQL** de forma eficiente e conveniente.

O **PyMySQL** é uma escolha popular para quem trabalha com Python e **MySQL** devido à sua simplicidade de uso e ampla adoção pela comunidade.

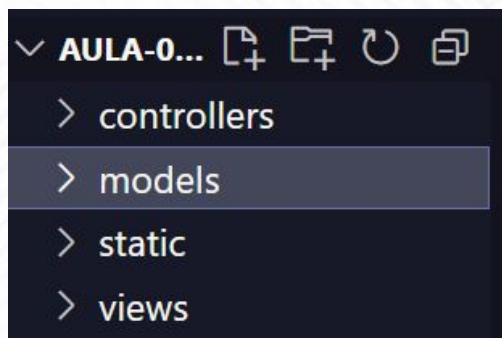


Criação dos Models

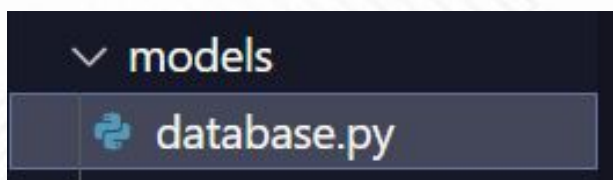


Para utilizarmos o MySQL em nossa aplicação o primeiro passo é instalar a biblioteca **SQLAlchemy**. Para instalar, execute o seguinte comando no terminal: **pip install -U Flask-SQLAlchemy**

Feito isso, seguindo a **arquitetura MVC** criaremos nossa pasta de **Models** que será a camada responsável pela **manipulação de dados**.



Dentro da pasta “**models**” crie um arquivo com o nome “**database.py**”.



Criação dos Models



Nesse arquivo, iremos importar a biblioteca **SQLAlchemy**, e faremos o seu carregamento na variável “**db**”, conforme código abaixo:

```
from flask_sqlalchemy import SQLAlchemy
db = SQLAlchemy()
```

Logo abaixo dessa linha, criaremos a classe “**Game**”, que será a classe responsável por criar a entidade “**Game**” no banco de dados com seus respectivos atributos:

```
class Game(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    titulo = db.Column(db.String(150))
    ano = db.Column(db.Integer)
    categoria = db.Column(db.String(150))
    plataforma = db.Column(db.String(150))
    preco = db.Column(db.Float)
    quantidade = db.Column(db.Integer)

    def __init__(self, titulo, ano, categoria, plataforma, preco, quantidade):
        self.titulo = titulo
        self.ano = ano
        self.categoria = categoria
        self.plataforma = plataforma
        self.preco = preco
        self.quantidade = quantidade
```


Conexão com o banco MySQL



O primeiro passo para realizar a conexão com o banco de dados **MySQL** é realizar a instalação da biblioteca **PyMySQL**, para isso, execute o seguinte comando no terminal:

```
pip install pymysql
```

Em seguida, no arquivo **“app.py”**, devemos importar a biblioteca **pymysql**:

```
import pymysql
```

Feito isso, logo abaixo, iremos definir o **nome do banco** e o seu **endereço**.

Caso seu banco não possua senha deixe em branco. Exemplo: **root@localhost...**

```
# Define o nome do banco de dados
DB_NAME = 'games'
app.config['DATABASE_NAME'] = DB_NAME
# Passando o endereço do banco ao SQLAlchemy
app.config['SQLALCHEMY_DATABASE_URI'] = f'mysql://root@localhost/{DB_NAME}'
```

Ainda no arquivo **“app.py”** faremos a importação do nosso **model**.

```
from flask import Flask, render_template
from controllers import routes
import pymysql
from models.database import db, Game
```

Conexão com o banco MySQL



Ainda no arquivo “**app.py**”, iremos incluir o código para criar o banco de dados caso ele ainda não exista, conforme a seguir:

```
if __name__ == '__main__':
    # Conecta ao MySQL para criar o banco de dados, se necessário
    connection = pymysql.connect(host='localhost',
                                user='root',
                                password='',
                                charset='utf8mb4',
                                cursorclass=pymysql.cursors.DictCursor)

    try:
        with connection.cursor() as cursor:
            # Cria o banco de dados se ele não existir
            cursor.execute(f"CREATE DATABASE IF NOT EXISTS {DB_NAME}")
            print(f"O banco de dados está criado!")
    except Exception as e:
        print(f"Erro ao criar o banco de dados: {e}")
    finally:
        connection.close()

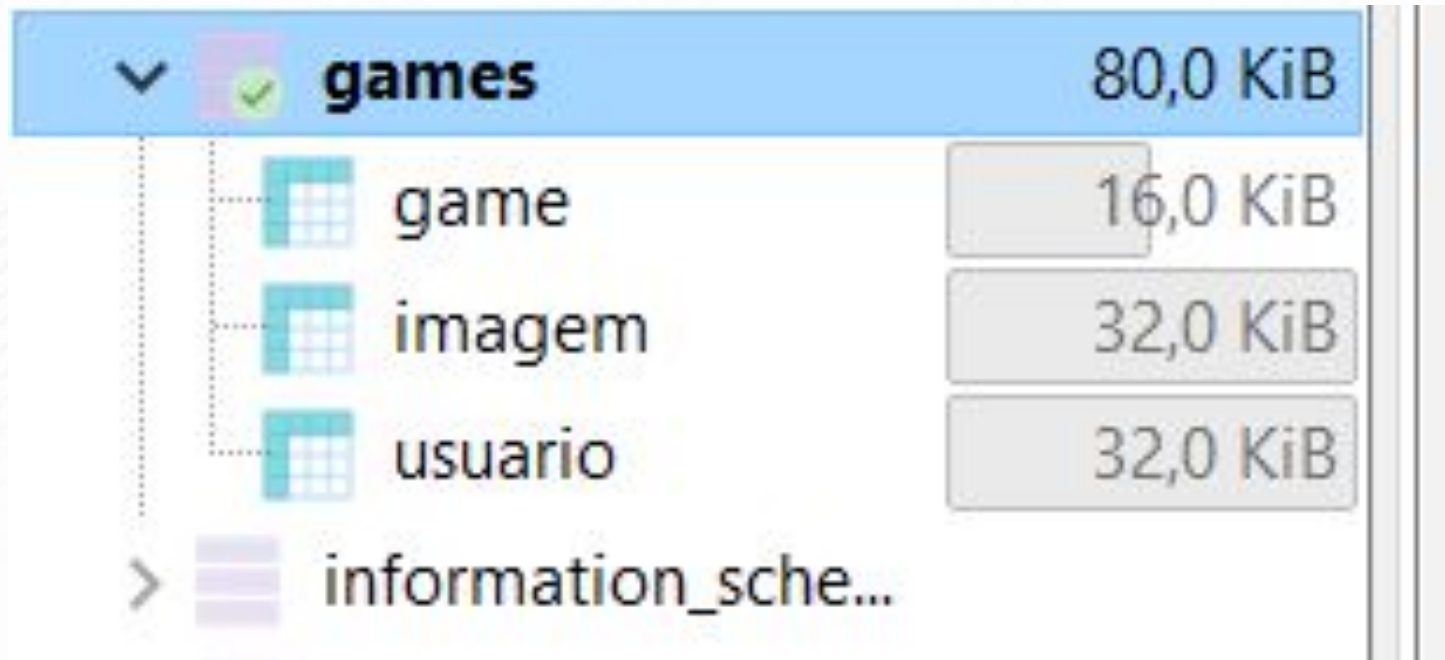
# Inicializa a aplicação Flask
db.init_app(app=app)
with app.test_request_context():
    # Cria as tabelas
    db.create_all()

# Inicia o aplicativo Flask
app.run(host='localhost', port=5000, debug=True)
```


Conexão com o banco MySQL



Com essas alterações feitas, basta rodar a aplicação e checar se o banco e as tabelas foram criadas corretamente:



A screenshot of a MySQL database management tool interface. The 'games' database is selected and expanded, showing three tables: 'game', 'imagem', and 'usuario'. The 'game' table has a size of 16,0 KiB, 'imagem' has 32,0 KiB, and 'usuario' has 32,0 KiB. Below these, the 'information_sche...' database is partially visible.

▼	✓	games	80,0 KiB
		game	16,0 KiB
		imagem	32,0 KiB
		usuario	32,0 KiB
>		information_sche...	

Exibindo dados do banco



Agora iremos inserir de forma manual os primeiros dados em nossa tabela, para isso vá até a guia “Dados”. Depois, insira alguns dados na tabela.

Estrutura

Dados

Constraints

Índices

Triggers

DDL

Exibição em grade

Visualização do formulário

1

id	titulo	ano	categoria	plataforma	preco	quantidade	
1	NULL	FIFA 2019	2019	Esporte	XBOX 360	39.90	15

Feito isso, iremos criar a rota “/estoque” que será responsável por renderizar a página de estoque e exibir os games cadastrados no banco de dados.

Nessa rota também faremos uma consulta no banco de dados para selecionar todos os games e passar para a página. Para isso, usaremos o método `query.all()` do SQL Alchemy:

```
@app.route('/estoque')
def estoque():
    gamesestoque = Game.query.all()
    return render_template('estoque.html', gamesestoque=gamesestoque)
```

Exibindo dados do banco



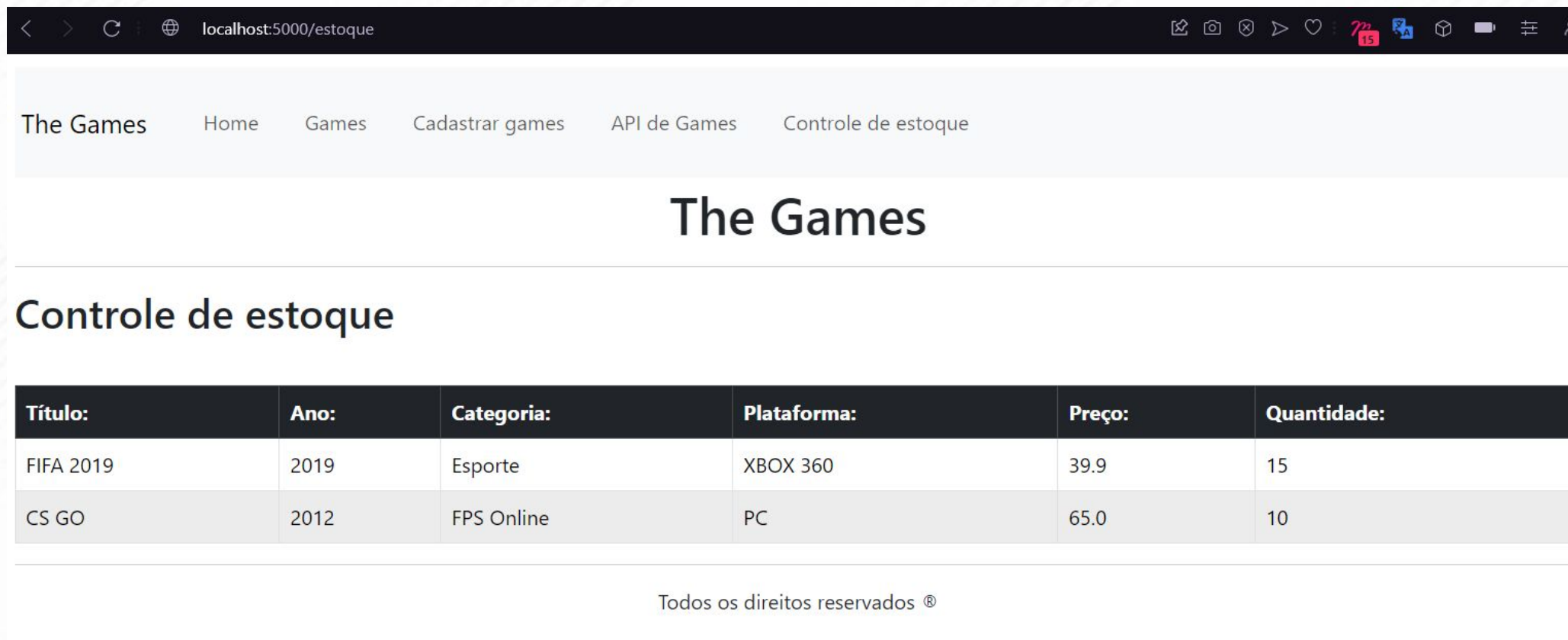
Agora iremos criar a view “**estoque.html**” que será a página responsável por gerir o nosso controle de estoque de games. Nela, futuramente poderemos **exibir**, **cadastrar**, **editar** e **deletar games**. Essa página terá a seguinte estrutura:

```
{% extends 'base.html' %}
{% block content %}
    <h2>Controle de estoque</h2><br>
    <table class="table table-bordered table-hover">
        <thead class="table-dark">
            <tr>
                <th>Título:</th>
                <th>Ano:</th>
                <th>Categoria:</th>
                <th>Plataforma:</th>
                <th>Preço:</th>
                <th>Quantidade:</th>
            </tr>
        </thead>
        <tbody>
            {% for g in gamesestoque %}
                <tr>
                    <td>{{g.titulo}}</td>
                    <td>{{g.ano}}</td>
                    <td>{{g.categoria}}</td>
                    <td>{{g.plataforma}}</td>
                    <td>{{g.preco}}</td>
                    <td>{{g.quantidade}}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock content %}
```


Exibindo dados do banco



Acessando agora a rota “**localhost:5000/estoque**” temos o seguinte resultado:



Título:	Ano:	Categoria:	Plataforma:	Preço:	Quantidade:
FIFA 2019	2019	Esporte	XBOX 360	39.9	15
CS GO	2012	FPS Online	PC	65.0	10

Todos os direitos reservados ®

Note que os jogos cadastrados no banco são exibidos dentro de uma tabela. Aprendemos assim como criar um banco de dados **SQLite** no Flask e fazer a consulta de dados. No próximo tópico aprenderemos como **cadastrar dados no banco**.



Flask

Programação Web III

Prof. Diego Max