

OPERAÇÕES CRUD COM PYTHON, CSS, JAVA E SQL

AVALIAÇÃO A3 – API RESTFULL

Sumário

OPERAÇÕES CRUD COM PYTHON, CSS, JAVA E SQL	1
AValiação A3 – API RESTFULL	1
Ambiente:.....	3
Módulos para instalar:	3
Outras variáveis do ambiente de homologação:	3
Caminho UNC na minha estação:.....	3
Caminho Serviço:.....	3
Evidência de pacotes Instalados:	3
Evidência de Pastas do Projeto Pelo Windows:	4
Visão geral do Aplicativo:	4
Visão das Pastas:	5
Pasta de Scripts para rodar no seu SQL Server;	5
Pasta de CSS para as páginas desenvolvidas do CRUD:	5
Banco de Dados:.....	6
Conexão com o Banco:	6
Automatizando Processos:.....	7
Conexão com o Banco:	7
Script:	8
Scripts Disponíveis:.....	10
Python e o código:.....	10
Código fonte:.....	11
Tela de logon:	11
Página Home:	12
Página Adicionar: "CREATE"	13
Página Listar: "READ"	13
Página de Edição / exclusão: "UPDATE & DELETE"	14
Código Python:	16

Ambiente:

SO: [Windows 11](#) – Professional de 64 Bits

Banco de Dados: [MS SQLSERVER 2022](#) (express edition)

IDE: [PyCharm 2023.1](#): Build #PY-231.9011.38, built on May 16, 2023 (versão pró trial – logado com o google)

Módulos para instalar:

```
pip install flask
```

```
python.exe -m pip install --upgrade pip
```

```
pip install pyodbc
```

Outras variáveis do ambiente de homologação:

Caminhos importantes para você identificar no seu projeto.

Caminho UNC na minha estação:

C:\Users\A58493\PycharmProjects\SociescA3

Caminho Serviço:

jetbrains://pycharm/navigate/reference?project=SociescA3&path=

Evidência de pacotes Instalados:

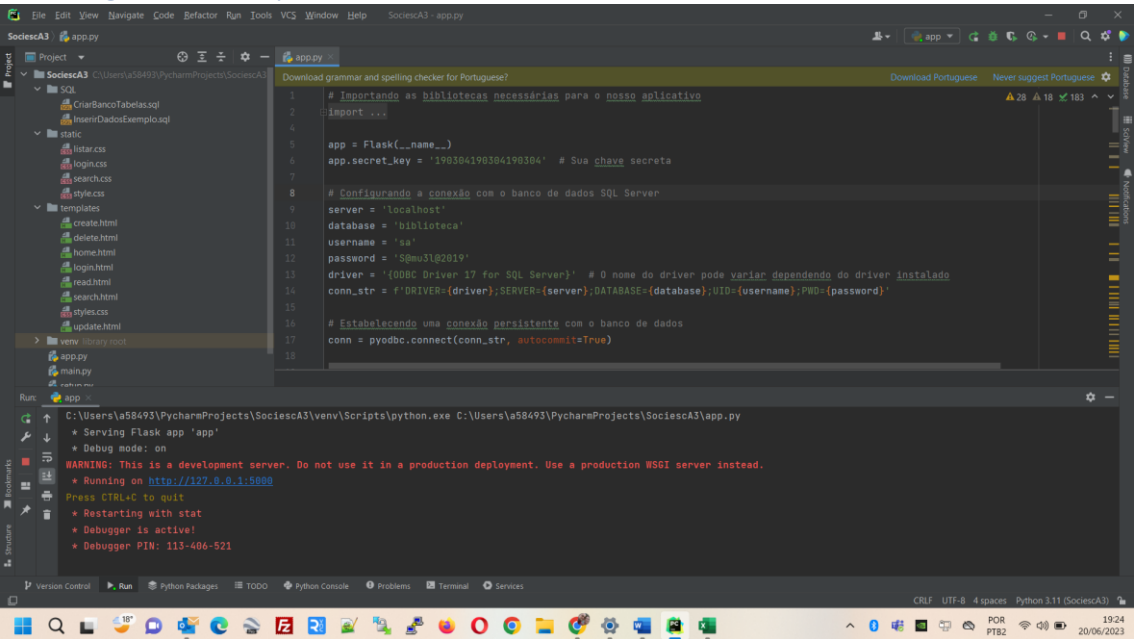
Python Packages	
🔍	🔄 ⚙️ Add Package ▾
▼ Installed	
Flask	2.3.2
Jinja2	3.1.2
MarkupSafe	2.1.3
Werkzeug	2.3.6
blinker	1.6.2
click	8.1.3
colorama	0.4.6
itsdangerous	2.1.2
pip	23.1.2
pyodbc	4.0.39
setuptools	65.5.1
wheel	0.38.4
▶ PyPI	

Evidência de Pastas do Projeto Pelo Windows:









📁 > a58493 > PycharmProjects > SociescA3 >

Nome	Data de modificação	Tipo	Tamanho
📁 .idea	20/06/2023 19:25	Pasta de arquivos	
📁 SQL	20/06/2023 17:04	Pasta de arquivos	
📁 static	20/06/2023 16:17	Pasta de arquivos	
📁 templates	20/06/2023 16:41	Pasta de arquivos	
📁 venv	20/06/2023 11:28	Pasta de arquivos	
📁 app	20/06/2023 19:19	JetBrains PyCharm	7 KB
📁 main	20/06/2023 10:17	JetBrains PyCharm	1 KB
📁 setup	20/06/2023 16:58	JetBrains PyCharm	1 KB



Visão geral do Aplicativo:







Visão das Pastas:

a58493 > PycharmProjects > SociescA3 > templates				
Nome	Data de modificação	Tipo	Tamanho	
 create	20/06/2023 16:41	Chrome HTML Do...	1 KB	
 delete	20/06/2023 15:58	Chrome HTML Do...	1 KB	
 home	20/06/2023 15:46	Chrome HTML Do...	1 KB	
 login	20/06/2023 16:06	Chrome HTML Do...	1 KB	
 read	20/06/2023 16:39	Chrome HTML Do...	1 KB	
 search	20/06/2023 16:36	Chrome HTML Do...	3 KB	
 styles	20/06/2023 07:59	Documento de fol...	1 KB	
 update	20/06/2023 13:55	Chrome HTML Do...	2 KB	

Pasta de Scripts para rodar no seu SQL Server;

a58493 > PycharmProjects > SociescA3 > SQL				
Nome	Data de modificação	Tipo	Tamanho	
 CriarBancoTabelas	20/06/2023 17:04	Microsoft SQL Ser...	2 KB	
 InserirDadosExemplo	20/06/2023 17:04	Microsoft SQL Ser...	6 KB	

Pasta de CSS para as páginas desenvolvidas do CRUD:

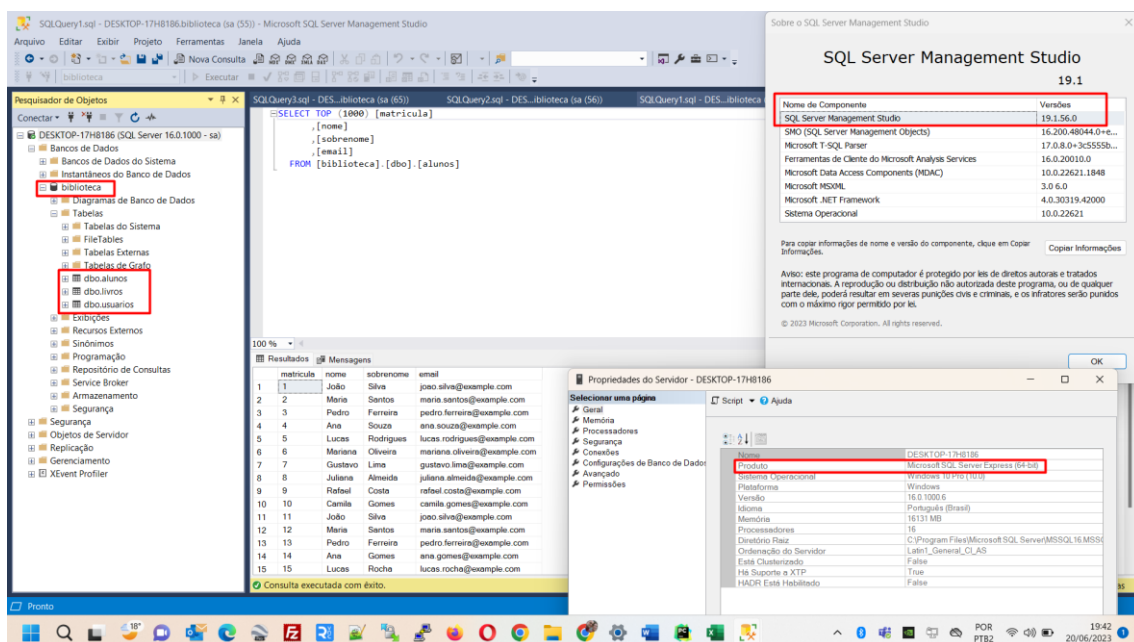
a58493 > PycharmProjects > SociescA3 > static				
Nome	Data de modificação	Tipo	Tamanho	
 listar	20/06/2023 16:11	Documento de fol...	1 KB	
 login	20/06/2023 16:05	Documento de fol...	1 KB	
 search	20/06/2023 16:17	Documento de fol...	1 KB	
 style	20/06/2023 14:12	Documento de fol...	3 KB	

Obs.: As evidências acima são a migração (validação) de um ambiente onde eu apenas copiei os arquivos do projeto, fiz novas instalações de SQL e PyCharm, instalei as extensões acima, **liberei as portas do firewall do Windows** e rodando o script para preparar o banco, funcionou perfeitamente, isso para validar o uso dele em seu ambiente simulando o uso do github. Atente para que as **credenciais de acesso ao seu banco** estejam no corpo da aplicação no arquivo app.py a conexão à persistência está definida lá.

```
# Configurando a conexão com o banco de dados SQL Server
server = 'localhost'
database = 'biblioteca'
username = 'sa'
password = 'S@mu3L@2019'
driver = '{ODBC Driver 17 for SQL Server}' # O nome do driver pode variar dependendo do driver instalado
conn_str = f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'
```

Banco de Dados:

Instalação padrão, nenhuma configuração específica para compatibilidade, visão geral pelo gerenciador, a intenção é saber que **o banco está no formato**: Latin1_General_CI_AS o restante são credenciais, drivers e liberações de firewall, eu gerei por IA dados para popular o banco dando uma carga de dados para utilizarmos, que também está na pasta Script do projeto.

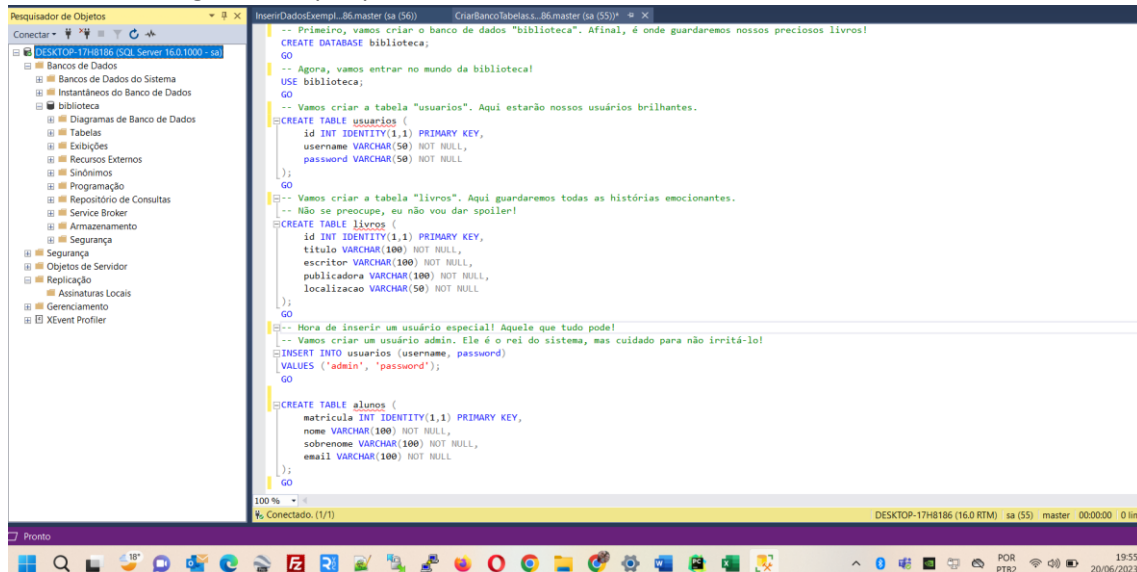


Conexão com o Banco:

Perceba que este é o meu ambiente, para seu ambiente pode ser uma máquina na rede, pode ser um MySQL ao invés de SQLEXPRESS, e as **credenciais** que você usar aqui, vai usar depois para criar a **conexão no arquivo app.py**, este ponto é importante, dependendo do **antivírus** que você tiver, desativar o **firewall do Windows** pode não permitir acesso ao seu banco, mesmo que localmente, atente para os passos até aqui para quando iniciar os testes, não ter limitações de extensões no seu framework pycharm, problemas de conectividade ou credenciais incorretas para acesso ao seu banco de dados, ainda no Windows Defender, a pasta do seu projeto precisa ser colocada como confiável, senão poderá ser bloqueada a execução das aplicações.

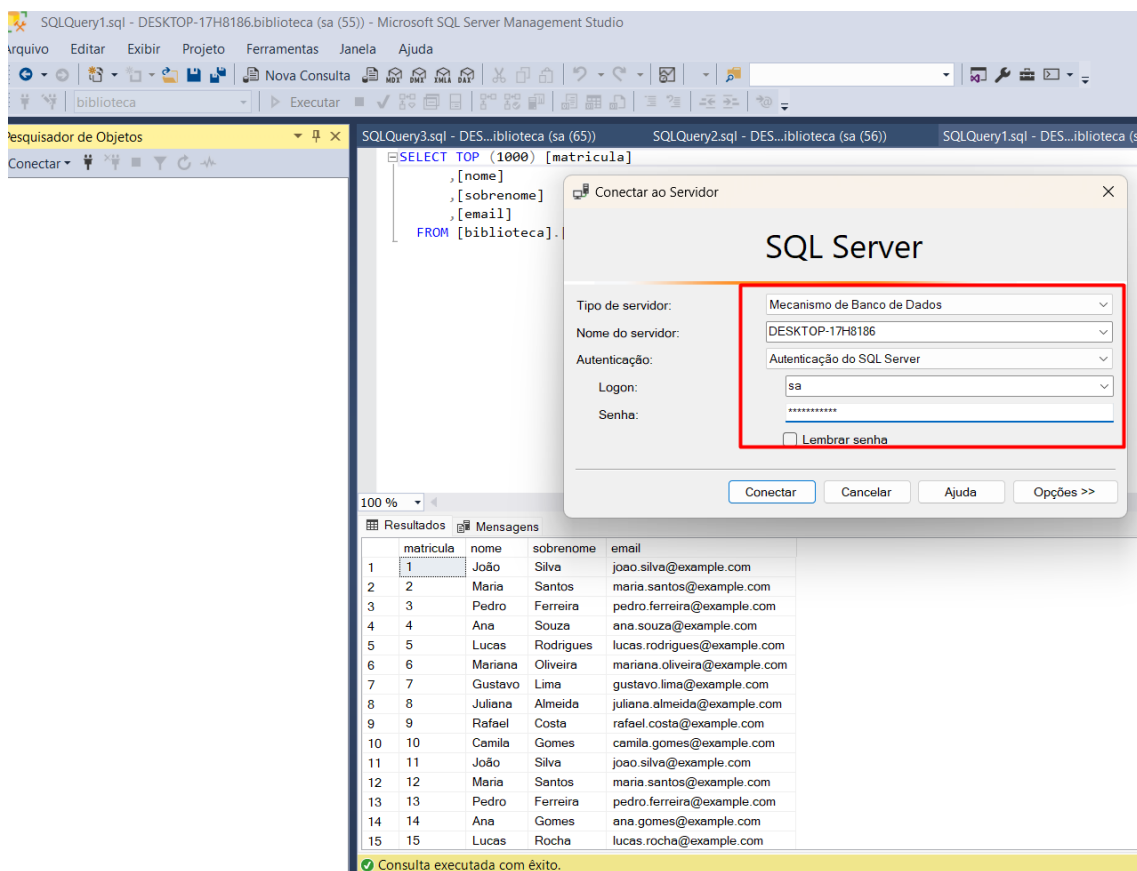
Automatizando Processos:

É possível fazer por código a criação de novas tabelas e seus dados, mas para separar as coisas desenvolvi o seguinte script, que cria o banco, e as tabelas com os índices de autoincremento



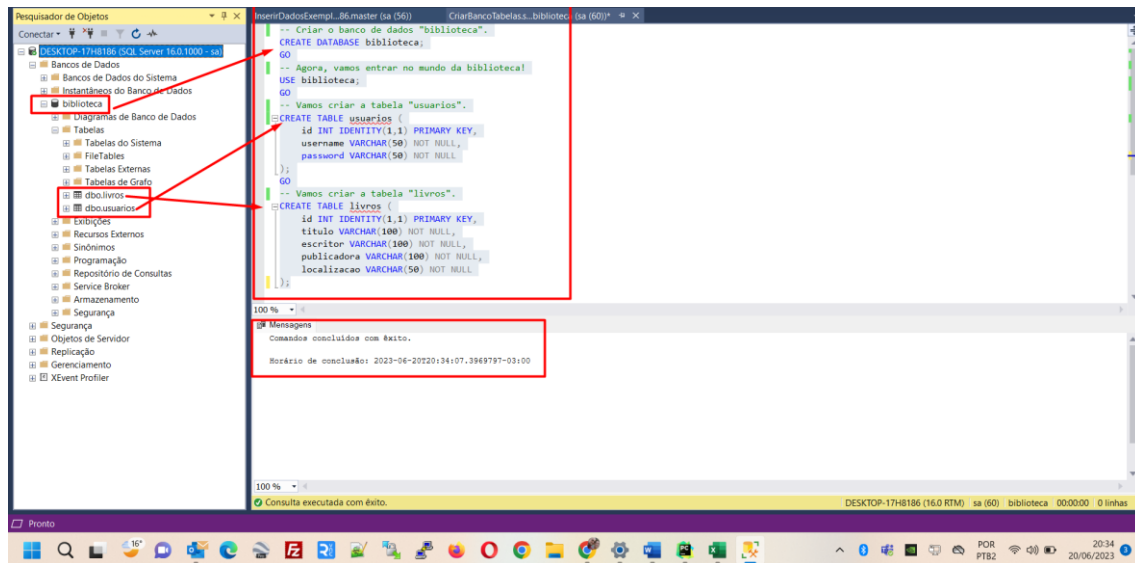
Conexão com o Banco:

Eu estou usando o Studio Management, mas poderia ser o [Dbeaver](#) ou outra solução como o [SQL Lite](#) de sua preferência:

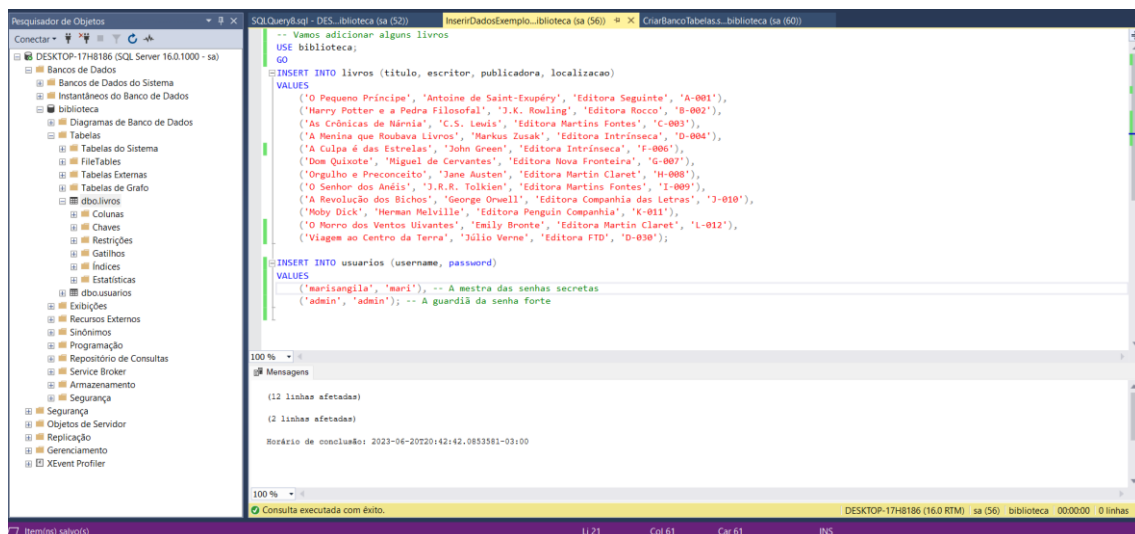


Script:

O Arquivo CriarBancoTabelas.sql cria a estrutura do projeto



O Arquivo InserirDadosExemplo.sql carrega dados para as tabelas criadas com o primeiro script, a ordem de execução dos scripts deve ser primeira **CriarBancoTabelas.sql** e depois **InserirDadosExemplo.sql**



Após isto você pode listar para validar os dados nas tabelas:

```
SELECT TOP (1000) [id]
      ,[titulo]
      ,[escritor]
      ,[publicadora]
      ,[localizacao]
```



```
FROM [biblioteca].[dbo].[livros]
```

SQLQuery9.sql - DES...iblioteca (sa (72)) SQLQuery8.sql - DES...iblioteca (sa (52)) InserirDadosExemplo...

```
SELECT TOP (1000) [id]
      ,[titulo]
      ,[escritor]
      ,[publicadora]
      ,[localizacao]
FROM [biblioteca].[dbo].[livros]
```

100 %

Resultados Mensagens

	id	titulo	escritor	publicadora	localizacao
1	1	O Pequeno Príncipe	Antoine de Saint-Exupéry	Editora Seguinte	A-001
2	2	Harry Potter e a Pedra Filosofal	J.K. Rowling	Editora Rocco	B-002
3	3	As Crônicas de Nárnia	C.S. Lewis	Editora Martins Fontes	C-003
4	4	A Menina que Roubava Livros	Markus Zusak	Editora Intrínseca	D-004
5	5	A Culpa é das Estrelas	John Green	Editora Intrínseca	F-006
6	6	Dom Quixote	Miguel de Cervantes	Editora Nova Fronteira	G-007
7	7	Orgulho e Preconceito	Jane Austen	Editora Martin Claret	H-008
8	8	O Senhor dos Anéis	J.R.R. Tolkien	Editora Martins Fontes	I-009
9	9	A Revolução dos Bichos	George Orwell	Editora Companhia das Letras	J-010
10	10	Moby Dick	Herman Melville	Editora Penguin Companhia	K-011
11	11	O Morro dos Ventos Uivantes	Emily Bronte	Editora Martin Claret	L-012
12	12	Viagem ao Centro da Terra	Júlio Verne	Editora FTD	D-030

```
SELECT TOP (1000) [id]
      ,[username]
      ,[password]
```

```
FROM [biblioteca].[dbo].[usuarios]
```

	id	username	password
1	1	marisangila	mari
2	2	admin	admin

Scripts Disponíveis:

Ok a respeito de bancos, estas são as observações os scripts estão na pasta SQL do projeto:

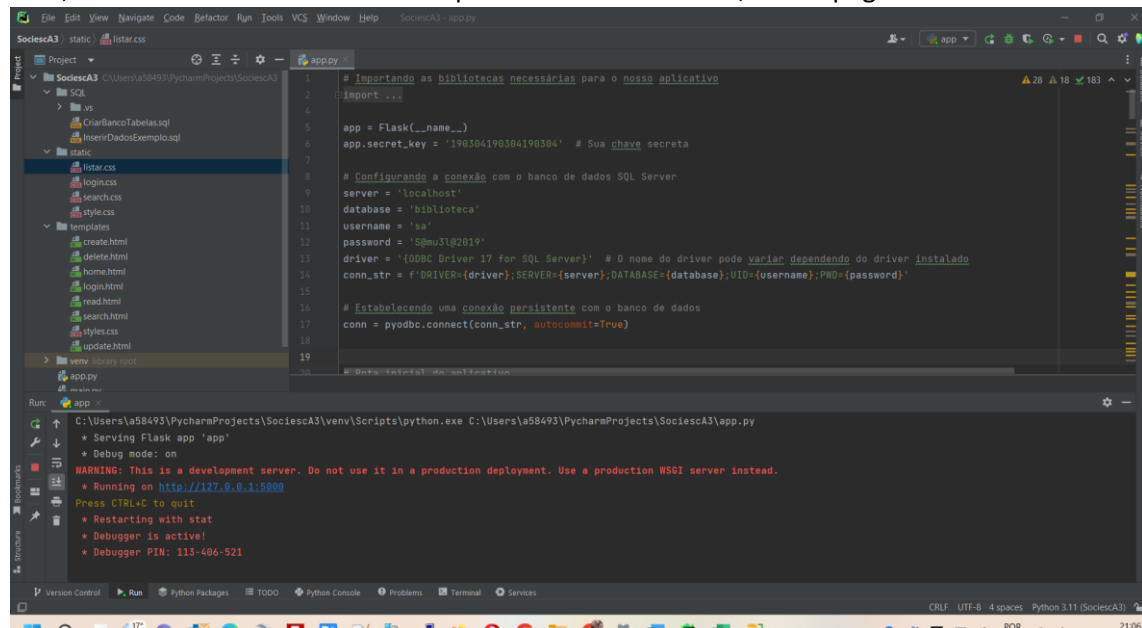
■ > a58493 > PycharmProjects > SociescA3 > SQL

Nome	Data de modificação	Tipo	Tamanho
.vs	20/06/2023 19:51	Pasta de arquivos	
CriarBancoTabelas	20/06/2023 20:35	Microsoft SQL Ser...	1 KB
InserirDadosExemplo	20/06/2023 20:42	Microsoft SQL Ser...	2 KB

Python e o código:

Ok, bora para o código, que aqui o filho chora e a mãe não vê, para atender a especificação inicial, a aplicação voltada para uma operação CRUD, faz apenas as operações básicas, tentei observar o uso de conceitos aprendidos nas aulas sobre usabilidade, o que se mostrou difícil a aplicação, pois o app.py do meu projeto, só pra funcionar o crud tinha cerca de 100 linhas, logava, exibia um menu com as opções de adição, pesquisa e para facilitar a minha vida, pensei a tela de alterar, já pode ter a função excluir, diminuí um css e um html, foi quase um êxtase quando consegui parar os erros de persistência pois o flask precisa de uma chave para fazer as operações de banco e isso me tomou uma tarde para descobrir, até que achei o IDE do PyCharm

e abandonei o Visual Studio, sim, para quem não tem experiência em desenvolvimento, uma IDE com o PyCharm facilita muito em atender os requisitos de ambiente e o modo debug dele, é muito simples e fácil, pega a visão:



Gosto não está em questão mas achar o terminal de console para adicionar as extensões aqui é intuitivo, o visual studio, é muito parrudo e com isso muito complexo, também abrangente, acho que no geral uma IDE dedicada, é sempre a melhor escolha.

Código fonte:

Vai estar disponível no GITHUB então vamos ao como eu fiz o uso dos CSS para formatar o texto puro que era feito no projeto inicial:

Tela de login:

Tentei fazer um tratamento para validar quando é user inexistente, e se o user for existente, validar a senha, se estiver errada avisar que é a senha, como parte das aulas de usabilidade onde deve ser permitido ao usuário auto recuperação, eu não fiz mecanismos para além de garantir que só acesse as outras páginas usuários autenticados, e dizer onde está o erro, mas pode ser uma melhoria aplicar conceitos de confirmação de pergunta secreta, cadastro positivo, entre outras formas

Login Page

Username

Password

Login

Oops! A senha não está batendo. Tenta de novo?

Uma vez logado, perceba que na tela de logon usei CSS para alinhar os objetos e fazer efeitos de texto. Na página home, onde é direcionado o usuário quando loga, se depara com estas opções e layout:

Página Home:



Pagina Adicionar: "CREATE"

127.0.0.1:5000/create

Adicionar Novo Livro

Formulário para adicionar um novo livro:

- Título:
- Escritor:
- Publicadora:
- Localização:

Adicionar Livro

[Voltar para a Home](#)

Pagina Listar: "READ"

127.0.0.1:5000/read

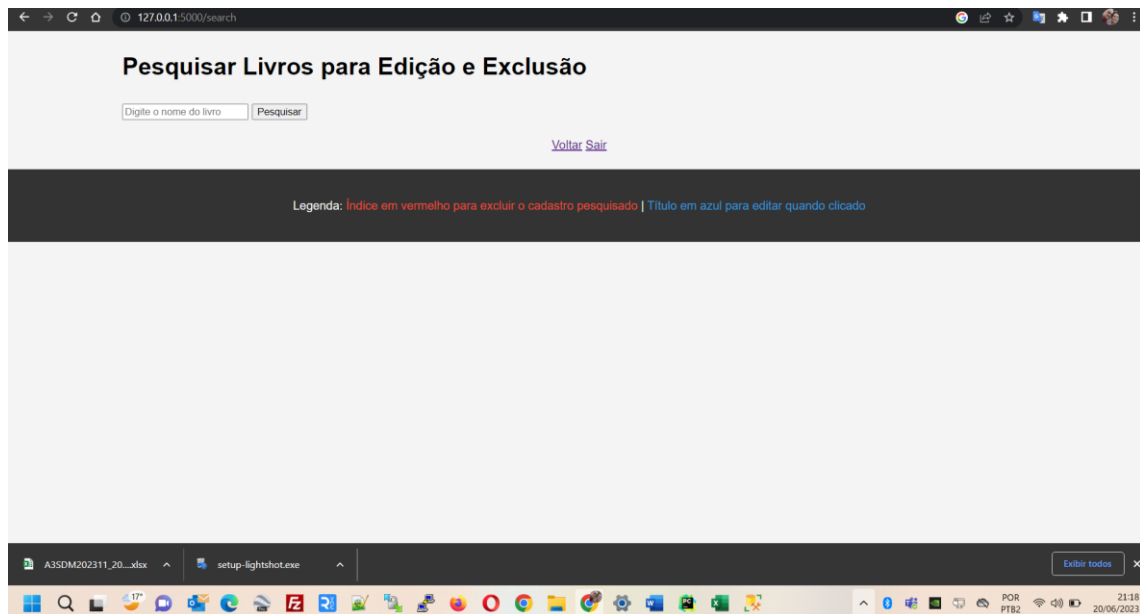
Aqui estão todos os nossos 12 livros:

Título	Escritor	Publicadora	Localização
O Pequeno Príncipe	Antoine de Saint-Exupéry	Editora Seguinte	A-001
Harry Potter e a Pedra Filosofal	J.K. Rowling	Editora Rocco	B-002
As Crônicas de Nárnia	C.S. Lewis	Editora Martins Fontes	C-003
A Menina que Roubava Livros	Markus Zusak	Editora Intrínseca	D-004
A Culpa é das Estrelas	John Green	Editora Intrínseca	F-006
Dom Quixote	Miguel de Cervantes	Editora Nova Fronteira	G-007
Orgulho e Preconceito	Jane Austen	Editora Martin Claret	H-008
O Senhor dos Anéis	J.R.R. Tolkien	Editora Martins Fontes	I-009
A Revolução dos Bichos	George Orwell	Editora Companhia das Letras	J-010
Moby Dick	Herman Melville	Editora Penguin Companhia	K-011
O Morro dos Ventos Uivantes	Emily Brontë	Editora Martin Claret	L-012
Viagem ao Centro da Terra	Júlio Verne	Editora FTD	D-030ss

[Voltar para a Home](#)

Exibir todos

Página de Edição / exclusão: “UPDATE & DELETE”



Permite que nesta tela eu chame 3 rotas, um get ou search, um alter e um delete
o get está no botão de pesquisa que lista os livros do seu acervo pessoal:



Clicando no nome do livro como mostra a legenda, vai para a tela de edição: “update”

127.0.0.1:5000/update/12

Atualizar Livro

Título:
Viagem ao Centro da Terra

Escritor:
Júlio Verne

Publicadora:
Editora FTD

Localização:
D-030ss

Atualizar Livro Home

Legenda: Você está editando o registro do livro selecionado na página de pesquisa

Por outro lado, se o usuário clicar em excluir, ele pede confirmação, lançando com Java, uma janela com o nome do livro que foi clicado a confirmação para a exclusão do registro:

127.0.0.1:5000/search

Pesquisar Livros para Edição

127.0.0.1:5000 diz
Tem certeza de que deseja excluir o livro "Viagem ao Centro da Terra"?

OK Cancelar

Dados Disponíveis para Edição e Exclusão:

- [A Culpa é das Estrelas](#) Escritor: John Green [\[Excluir\]](#)
- [A Menina que Roubava Livros](#) Escritor: Markus Zusak [\[Excluir\]](#)
- [A Revolução dos Bichos](#) Escritor: George Orwell [\[Excluir\]](#)
- [As Crônicas de Nárnia](#) Escritor: C.S. Lewis [\[Excluir\]](#)
- [Dom Quixote](#) Escritor: Miguel de Cervantes [\[Excluir\]](#)
- [Moby Dick](#) Escritor: Herman Melville [\[Excluir\]](#)
- [O Pequeno Príncipe](#) Escritor: Antoine de Saint-Exupéry [\[Excluir\]](#)
- [O Senhor dos Anéis](#) Escritor: J.R.R. Tolkien [\[Excluir\]](#)
- [Viagem ao Centro da Terra](#) Escritor: Júlio Verne [\[Excluir\]](#)

[Voltar](#) [Sair](#)

Se cancelar fica na tela sem ação, se confirmar em ok, ativa o "Delete"

← → ↺ 🏠 ⓘ 127.0.0.1:5000/delete/12

Livro Deletado

O livro "Viagem ao Centro da Terra" foi excluído com sucesso!

Você será redirecionado para a página inicial em 5 segundos...

O projeto possui o objetivo de cadastrar usuários e livros, permitindo identificação da localização, sem SSL no envio e retorno do texto puro, nem criptografia no armazenamento de banco, as credenciais e usuários estão salvos em texto puro, todavia foi feita uma persistência para validar e o programa só responde para seções devidamente autenticadas, para futuros trabalhos uma sugestão seria implantar SSL na comunicação com o banco e um algoritmo para embaralhar as senhas, aqui com exceção do CSS, o mais simples e funcional foi abordado de acordo com a minha percepção e conhecimento, obviamente passíveis de melhoria por mim mesmo se conseguisse mais tempo para dedicar-me ao projeto, mas dei foco para o CRUD. Alguns conceitos de usabilidade, não foram atendidos, pois dependiam de tecnologias que eu não tenho domínio e o tempo de pesquisa não foi suficiente para alterar novamente o projeto que teve 3 fases funcionais bem distintas, só texto, sem nenhuma formatação, para fazer funcionar a comunicação de dados e as telas, depois CSS no HTML, que não é uma prática correta, mas funcionou, e depois estruturando o CSS na pasta e só linkando a página ao CSS que é o correto, deste ponto fica a aprendizagem que atender os requisitos de usabilidade é possível, mas precisa envolver mais tecnologias, o uso de alguns frameworks, para mim não foi viável como o Bootstrap, mas o objetivo funcional da proposta da aplicação foi em boa parte atendida, além dos exemplos da Professora, as orientações dela permitiram evoluir, na tela de login foi observado parcialmente os conceitos de recuperação, mas faltou implementar algo como envio automático de uma notificação para o suporte, ou confirmar mais dados, do usuário para permitir que ele redefina a senha sozinho, confirmando mais dados. No geral a experiência foi excelente, embora o projeto seja bem simples, deu para no processo verificar a evolução do código, os recursos de formatação por CSS e como o Python diminui a curva de aprendizado de uma linguagem.

CódigoPython:

```
# Importando as bibliotecas necessárias para o nosso
aplicativo
from flask import Flask, render_template, request, session,
redirect, url_for
import pyodbc
app = Flask(__name__)
app.secret_key = '190304190304190304' # Sua chave secreta
# Configurando a conexão com o banco de dados SQL Server
```



```

server = 'localhost'
database = 'biblioteca'
username = 'sa'
password = 'S@mu3l@2019'
driver = '{ODBC Driver 17 for SQL Server}' # O nome do
driver pode variar dependendo do driver instalado
conn_str =
f'DRIVER={driver};SERVER={server};DATABASE={database};UID={
username};PWD={password}'
# Estabelecendo uma conexão persistente com o banco de
dados
conn = pyodbc.connect(conn_str, autocommit=True)
# Rota inicial do aplicativo
@app.route('/')
def index():
    return 'Hello, World!'
# Rota de login. Se os dados estiverem corretos, o usuário
é autenticado
# Aí, Professora Mari! Tô abrindo aqui a rota pra fazer
login. Se receber um GET, vou renderizar o formulário, se
for POST, vai checar as informações
@app.route('/login', methods=['GET', 'POST'])
def login():
    msg = ''
    # Checando se o método é POST e se os campos 'username'
e 'password' estão presentes
    if request.method == 'POST' and 'username' in
request.form and 'password' in request.form:
        # Capturando as informações digitadas pelo usuário
        username = request.form['username']
        password = request.form['password']

        # Criando o cursor, que vai ser tipo um
intermediário entre nosso código e o banco de dados
        cursor = conn.cursor()

        # Buscando o usuário no banco com o username
passado
        cursor.execute('SELECT * FROM usuarios WHERE
username = ?', username)
        # Pegando o resultado da busca
        account = cursor.fetchone()

        # Se a conta existir, vamos verificar a senha
        if account:
            # Se a senha bater, vamos logar o usuário e
redirecioná-lo para a rota home

```

```

        if password == account.password:
            session['loggedin'] = True
            session['id'] = account.id
            session['username'] = account.username
            return redirect(url_for("home"))
        else:
            # Senha não bateu, vamos retornar um aviso
            msg = 'Oops! A senha não está batendo.
Tenta de novo?'
            else:
                # Usuário não foi encontrado, vamos retornar
                outro aviso
                msg = 'Hum, não estamos encontrando você. Você
tem certeza que digitou o usuário certo?'
                return render_template('login.html', msg=msg)

# Aqui é a rota da página inicial. Se o usuário estiver
logado, mostra a página inicial. Se não, manda pra
# página de login
@app.route('/home')
def home():
    if 'loggedin' in session:
        cursor = conn.cursor()
        cursor.execute('SELECT COUNT(*) FROM livros')
        book_count = cursor.fetchone()[0]
        return render_template('home.html',
username=session['username'], book_count=book_count)
        return redirect(url_for('login'))

# Nessa rota será exibido todos os livros do banco de dados
@app.route('/read')
def read():
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM livros')
    livros = cursor.fetchall()
    return render_template('read.html', livros=livros)

# Rota para criar um novo livro. Se o método for GET,
renderiza o formulário. Se for POST, insere as informações
no banco de dados
@app.route('/create', methods=['GET', 'POST'])
def create():
    msg = ''
    if request.method == 'POST':
        titulo = request.form.get('titulo', '')
        escritor = request.form.get('escritor', '')

```

```

        publicadora = request.form.get('publicadora', '')
        localizacao = request.form.get('localizacao', '')

        # Se todas as informações foram preenchidas, insere
no banco
        if titulo and escritor and publicadora and
localizacao:
            cursor = conn.cursor()
            cursor.execute('INSERT INTO livros (titulo,
escritor, publicadora, localizacao) VALUES (?, ?, ?, ?)',
                            (titulo, escritor, publicadora,
localizacao))
            msg = 'Yeah! Seu novo livro foi adicionado!
Ponto Com a Professora Mari'
        else:
            # Alguma informação não foi preenchida, retorna
uma mensagem de erro
            msg = 'Opa! Parece que você esqueceu de
preencher algum campo.'
        return render_template('create.html', msg=msg)

# Rota para atualizar as informações de um livro
# Rota para atualizar as informações de um livro
@app.route('/update/<int:id>', methods=['GET', 'POST'])
def update(id):
    cursor = conn.cursor()
    msg = ''

    if request.method == 'POST':
        titulo = request.form['titulo']
        escritor = request.form['escritor']
        publicadora = request.form['publicadora']
        localizacao = request.form['localizacao']

        # Atualiza as informações no banco
        cursor.execute('''
            UPDATE livros SET titulo = ?, escritor = ?,
publicadora = ?, localizacao = ?
            WHERE id = ?
        ''', (titulo, escritor, publicadora, localizacao,
id))

        msg = 'Seu livro foi atualizado com sucesso!'

    # Se o método for GET, busca as informações do livro
para preencher o formulário
    cursor.execute('SELECT * FROM livros WHERE id = ?', id)

```

```

        livro = cursor.fetchone()

        return render_template('update.html', livro=livro,
msg=msg)

# Rota para excluir um livro
@app.route('/delete/<int:id>', methods=['GET', 'POST'])
def delete(id):
    cursor = conn.cursor()
    cursor.execute('SELECT titulo FROM livros WHERE id =
?', (id,))
    livro = cursor.fetchone()
    if livro:
        livro_nome = livro[0]
        # Exclui o livro e confirma a operação
        cursor.execute('DELETE FROM livros WHERE id = ?',
(id,))
        conn.commit()
        return render_template('delete.html',
livro=livro_nome, msg=f'O livro "{livro_nome}" foi excluído
com sucesso!')
    else:
        # Livro não foi encontrado, retorna uma mensagem de
erro
        msg = 'Livro não encontrado'
        return render_template('delete.html', msg=msg)

# Rota para buscar livros
@app.route('/search', methods=['GET', 'POST'])
def search():
    if request.method == 'POST':
        search = request.form.get('bookname', '')
        cursor = conn.cursor()
        if search:
            # Procura livros com base no nome
            cursor.execute('SELECT * FROM livros WHERE
titulo LIKE ?', f'%{search}%')
        else:
            # Mostra todos os livros se nenhum termo de
pesquisa for fornecido
            cursor.execute('SELECT * FROM livros')
            results = cursor.fetchall()
            return render_template('search.html',
results=results)
    else:
        return render_template('search.html', results=None)

```

```
# Aqui começa tudo! Vamos rodar a aplicação.  
if __name__ == '__main__':  
    app.run(debug=True)
```