

Estrutura da Aplicação

- **Front-end:** HTML para formulários e tabela, CSS para estilização básica.
- **Back-end:** Python com Flask (servidor web leve e fácil de configurar).
- **Banco de dados:** MySQL, com uma tabela simples **contatos** (id, nome, telefone).

A estrutura de arquivos será a seguinte (crie uma pasta no VS Code, ex: **agenda-telefonica**, e organize assim):

```
agenda-telefonica/
├── app.py                  # Seu arquivo principal
├── db_setup.py              # Já corrigido anteriormente
├── requirements.txt         # Dependências
├── templates/               # Pasta obrigatória para templates
│   └── index.html           # Cole o conteúdo HTML aqui
└── static/                  # Pasta para CSS/JS
    └── style.css             # Cole o conteúdo CSS aqui
```

Instruções de Configuração e Execução (no VS Code, Windows 11 Pro)

1. Instale o Python 3.12+ (se não tiver: baixe de [python.org](https://www.python.org)).
2. Instale o MySQL Server (baixe de [mysql.com](https://dev.mysql.com), configure com usuário **root** e senha **senha123** – ajuste no código se necessário).
3. Abra o terminal no VS Code (**Ctrl + ``**) e navegue para a pasta do projeto.
4. Crie um ambiente virtual: **python -m venv venv** e ative: **venv\Scripts\activate**.
5. Instale dependências: **pip install -r requirements.txt**.
6. Execute o setup do banco: **python db_setup.py** (isso cria o DB **agenda_db** e a tabela).
7. Inicie o servidor: **python app.py**.
8. Abra no navegador: <http://127.0.0.1:5000/>.

Notas:

- O Flask usa Jinja2 para renderizar o HTML dinamicamente.
- CRUD:
 - **Create:** Formulário para adicionar.
 - **Read:** Lista todos os contatos.
 - **Update:** Botão editar (abre form com dados pré-preenchidos).
 - **Delete:** Botão excluir.
- Para editar, usei um parâmetro na URL; para simplicidade, não há autenticação.

Agora, os arquivos:

requirements.txt

```
Flask==3.0.3
mysql-connector-python==9.0.0
```

db_setup.py

```
import mysql.connector
from mysql.connector import Error

# Configuração do MySQL (sem 'database' inicial para criar o banco)
config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'senha123', # Mude para sua senha
    # 'database' removido aqui para conexão inicial
}

try:
    # Conectar sem banco específico
    connection = mysql.connector.connect(**config)
    if connection.is_connected():
        cursor = connection.cursor()

        # Criar banco se não existir
        cursor.execute("CREATE DATABASE IF NOT EXISTS agenda_db")
        print("Banco 'agenda_db' criado ou já existe.")

        # Usar o banco recém-criado
        cursor.execute("USE agenda_db")

        # Criar tabela contatos
        create_table_query = """
            CREATE TABLE IF NOT EXISTS contatos (
                id INT AUTO_INCREMENT PRIMARY KEY,
                nome VARCHAR(100) NOT NULL,
                telefone VARCHAR(20) NOT NULL
            )
        """
        cursor.execute(create_table_query)
        connection.commit()
        print("Tabela 'contatos' criada com sucesso!")

        cursor.close()
        connection.close()
        print("Configuração do banco concluída!")

except Error as e:
    print(f"Erro ao conectar ao MySQL: {e}")
```

```
finally:  
    # O finally só executa se connection foi criada (mesmo que falhe depois)  
    if 'connection' in locals() and connection.is_connected():  
        cursor.close()  
        connection.close()
```

app.py

```
from flask import Flask, render_template, request, redirect, url_for, flash  
import mysql.connector  
from mysql.connector import Error  
  
app = Flask(__name__)  
app.secret_key = 'chave_secreta_simples' # Para flash messages  
  
# Configuração do MySQL  
DB_CONFIG = {  
    'host': 'localhost',  
    'user': 'root',  
    'password': 'senha123', # Ajuste para sua senha  
    'database': 'agenda_db'  
}  
  
def get_db_connection():  
    try:  
        connection = mysql.connector.connect(**DB_CONFIG)  
        return connection  
    except Error as e:  
        print(f"Erro ao conectar ao banco: {e}")  
        return None  
  
@app.route('/', methods=['GET', 'POST'])  
def index():  
    connection = get_db_connection()  
    if not connection:  
        flash('Erro ao conectar ao banco de dados.')  
        return render_template('index.html', contatos=[])  
  
    cursor = connection.cursor()  
  
    if request.method == 'POST':  
        action = request.form.get('action')  
        nome = request.form.get('nome')  
        telefone = request.form.get('telefone')  
        contact_id = request.form.get('id', 0, type=int)  
  
        if action == 'adicionar' and nome and telefone:  
            insert_query = "INSERT INTO contatos (nome, telefone) VALUES (%s,  
%s)"
```

```

        cursor.execute(insert_query, (nome, telefone))
        connection.commit()
        flash('Contato adicionado com sucesso!')

    elif action == 'editar' and contact_id > 0 and nome and telefone:
        update_query = "UPDATE contatos SET nome = %s, telefone = %s WHERE
id = %s"
        cursor.execute(update_query, (nome, telefone, contact_id))
        connection.commit()
        flash('Contato atualizado com sucesso!')

    elif action == 'excluir' and contact_id > 0:
        delete_query = "DELETE FROM contatos WHERE id = %s"
        cursor.execute(delete_query, (contact_id,))
        connection.commit()
        flash('Contato excluído com sucesso!')

# Listar todos os contatos
cursor.execute("SELECT id, nome, telefone FROM contatos ORDER BY nome")
contatos = cursor.fetchall()

# Buscar contato para edição (se ID na URL)
edit_id = request.args.get('edit', 0, type=int)
contato_edit = None
if edit_id:
    cursor.execute("SELECT id, nome, telefone FROM contatos WHERE id = %s",
(edit_id,))
    contato_edit = cursor.fetchone()

cursor.close()
connection.close()
return render_template('index.html', contatos=contatos,
edit_contato=contato_edit)

if __name__ == '__main__':
    app.run(debug=True)

```

templates/index.html

```

<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Agenda Telefônica - CRUD</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}>
</head>
<body>

```

```

<div class="container">
    <h1>Agenda Telefônica</h1>

    <!-- Formulário para Adicionar/Editar -->
    <form method="POST" class="form">
        <input type="hidden" name="action" value="{{ 'editar' if
edit_contato else 'adicionar' }}>
        {% if edit_contato %}
            <input type="hidden" name="id" value="{{ edit_contato[0] }}>
            <h2>Editar Contato</h2>
        {% else %}
            <h2>Adicionar Contato</h2>
        {% endif %}

        <label for="nome">Nome:</label>
        <input type="text" id="nome" name="nome" value="{{ edit_contato[1]
if edit_contato else '' }}" required>

        <label for="telefone">Telefone:</label>
        <input type="text" id="telefone" name="telefone" value="{{
edit_contato[2] if edit_contato else '' }}" required>

        <button type="submit">{{ 'Atualizar' if edit_contato else
'Adicionar' }}</button>
        {% if edit_contato %}
            <a href="/" class="btn-cancel">Cancelar</a>
        {% endif %}
    </form>

    <!-- Lista de Contatos -->
    <h2>Contatos</h2>
    {% with messages = get_flashed_messages() %}
        {% if messages %}
            <ul class="messages">
                {% for message in messages %}
                    <li>{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}
    {% endwith %}

    {% if contatos %}
        <table>
            <thead>
                <tr>
                    <th>Nome</th>
                    <th>Telefone</th>
                    <th>Ações</th>
                </tr>
            </thead>
            <tbody>
                {% for contato in contatos %}
                    <tr>

```

```

        <td>{{ contato[1] }}</td>
        <td>{{ contato[2] }}</td>
        <td>
            <a href="?edit={{ contato[0] }}" class="btn-edit">Editar</a>
            <form method="POST" style="display: inline;" onsubmit="return confirm('Excluir?');">
                <input type="hidden" name="action" value="excluir">
                <input type="hidden" name="id" value="{{ contato[0] }}">
                <button type="submit" class="btn-delete">Excluir</button>
            </form>
        </td>
    </tr>
    {% endfor %}
</tbody>
</table>
{% else %}
    <p>Nenhum contato cadastrado.</p>
{% endif %}
</div>
</body>
</html>

```

static/style.css

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

h1, h2 {
    color: #333;
}

.form {

```

```
background: #f9f9f9;
padding: 20px;
margin-bottom: 20px;
border-radius: 5px;
}

label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}

input[type="text"] {
    width: 100%;
    padding: 8px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
}

button {
    background: #007bff;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background: #0056b3;
}

.btn-cancel, .btn-edit {
    background: #6c757d;
    color: white;
    padding: 5px 10px;
    text-decoration: none;
    border-radius: 4px;
    margin-left: 10px;
}

.btn-edit:hover, .btn-cancel:hover {
    background: #545b62;
}

.btn-delete {
    background: #dc3545;
    color: white;
    padding: 5px 10px;
    border: none;
    border-radius: 4px;
```

```
    cursor: pointer;
    margin-left: 10px;
}

.btn-delete:hover {
    background: #c82333;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 10px;
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

th {
    background: #007bff;
    color: white;
}

.messages {
    list-style: none;
    padding: 0;
}

.messages li {
    background: #d4edda;
    color: #155724;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #c3e6cb;
    border-radius: 4px;
}

p {
    color: #666;
}
```

Essa aplicação está pronta para rodar! Teste adicionando contatos, editando e excluindo. Se precisar de ajustes (ex: senha do MySQL), edite os arquivos no VS Code. Para depuração, use o modo debug do Flask (já ativado). Se houver erros, verifique o terminal para logs.