

# BackEnd

# BackEnd - WebServer

Agora que sabemos como armazenar os dados no Banco de dados, precisamos também agora recuperar alguns dos dados para usá-los. Vamos criar uma função de nos permita carregar dados das turmas do professor.

Criar a função “carrega dados do professor”

# BackEnd - WebServer

Agora que sabemos como armazenar os dados no Banco de dados. Mas para garantir a consistência das informações, temos uma tabela de normalização, a “Turma Professor”, e devemos armazenar o relacionamento nele.  
Também precisamos listar as turmas que já estão para aquele professor

# BackEnd - WebServer

Agora que sabemos como armazenar os dados no Banco de dados. Mas para garantir a consistência das informações, temos uma tabela de normalização, a “Turma Professor”, e devemos armazenar o relacionamento nele.  
Também precisamos listar as turmas que já estão para aquele professor

# BackEnd - WebServer

## Login

Email:

nanda@gmail.com

Senha:

...

Enviar

Turmas do Professor:










nanda

Associar Turmas ao Professor:

## Cadastro de Turmas

Descrição:

Cadastrar

Turma B	
a	
Turma A	
Turma C	
Turma D	
Turma E	
Turma F	
Turma G	
Turma H	

# BackEnd - WebServer

Quando fizemos nossa modelagem de dados, colocamos a coluna de Id\_turma como auto incremento. O que significa que ele gera um código sozinho, sendo o último código mais 1.

Por isso não precisamos mais do código de turma.

Também é interessante fazemos a exibição das turmas já cadastradas

Turmas do Professor:










nanda

Associar Turmas ao Professor:

## Cadastro de Turmas

Descrição:

Cadastrar

Turma B	
a	
Turma A	
Turma C	
Turma D	
Turma E	
Turma F	
Turma G	
Turma H	

```
<div class="error-message">
  <!-- Mensagem de erro será inserida aqui -->

</div>

<div class="welcome-message">

</div>

<h4>Turmas do Professor: </h4>
<h3>{nome_professor}</h3>
<hr>
<h3>Associar Turmas ao Professor:</h3>
  <form action="/cad_turma" method="POST">
    <h1>Cadastro de Turmas</h1>
    <label>Descrição:</label>
    <input type="text" id="descTurma" name="descTurma" required>
    <input type="hidden" id="id_professor" name="id_professor" value="{id_professor}">
    <input type="hidden" id="login" name="login" value="{login}">

    <button type="submit">Cadastrar</button>
  </form>
  <hr/>
  <table>
    <!-- Tabela com linhas zebradas -->
  </table>
</div>
```

# Cad\_turma.html

# BackEnd - WebServer

```
def adicionar_turma_professor(self, descTurma, id_professor):  
    cursor = conexao.cursor()  
    cursor.execute("INSERT INTO turmas (descricao) VALUES (%s)", (descTurma,))  
    cursor.execute("SELECT id_turma FROM turmas WHERE descricao = %s", (descTurma,))  
    resultado = cursor.fetchone()  
    cursor.execute("INSERT INTO turmas_professor (id_turma, id_professor) VALUES (%s, %s)", (resultado[0], id_professor))  
    conexao.commit()  
    cursor.close()
```

Vamos acoplar no nosso cadastro da turma, a inclusão do professor turma também.

Abrindo uma conexão com o banco,

Pedindo a execução de uma instrução SQL, de inclusão na tabela TURMAS, no campo de descrição, uma vez que o campo ID já é incrementado automaticamente.

Porém esse mesmo ID é uma das Fks da tabela de turmas\_professor, então precisamos recuperá-la, fazendo um select.

Com todas as informações podemos fazer a inclusão em Turmas\_professor, seguido do commit e do close da chamada no banco.



```
def carrega_turmas_professor(self, login):
```

```
    # Carrega Turmas do Professor
```

```
    # -----
```

```
    cursor = conexao.cursor()
```

```
    cursor.execute("SELECT id_professor, nome FROM dados_login WHERE login = %s", (login,))
```

```
    resultado = cursor.fetchone()
```

```
    cursor.close()
```

```
    # resultado[0] trás o id_professor e resultado[1] trás o nome do professor
```

```
    id_professor = resultado[0]
```

```
    # Código para obter as turmas do professor
```

```
    cursor = conexao.cursor()
```

```
    cursor.execute(
```

```
        "SELECT turmas.id_turma, turmas.descricao FROM turmas_professor INNER JOIN turmas "
```

```
        "ON turmas_professor.id_turma = turmas.id_turma WHERE turmas_professor.id_professor = %s", (id_professor,))
```

```
    turmas = cursor.fetchall()
```

```
    cursor.close()
```

```
    # Construindo dinamicamente as linhas da tabela com as turmas do professor
```

```
    linhas_tabela = ""
```

```
    for turma in turmas:
```

```
        id_turma = turma[0]
```

Quebrei linha só para a visualização

Continua

# BackEnd - WebServer

```
linhas_tabela = ""
for turma in turmas:
    id_turma = turma[0]
    descricao_turma = turma[1]
    link_atividade = "<img src='icnatividade2.png' />"
    linha = "<tr><td style='text-align:center'>{}</td><td style='text-align:center'>{}</td></tr>".format(
        *args: descricao_turma, link_atividade)
    linhas_tabela += linha

with open(os.path.join(os.getcwd(), 'cad_turma.html'), 'r', encoding='utf-8') as cad_turma_file:
    content = cad_turma_file.read()

    content = content.replace(__old: '{nome_professor}', resultado[1])
    content = content.replace(__old: '{id_professor}', str(id_professor))
    content = content.replace(__old: '{login}', str(login))

    #Substituindo o marcador de posição pelas linhas da tabela
content = content.replace(__old: '<!-- Tabela com linhas zebradas -->', linhas_tabela)
self.send_response(200)
self.send_header(keyword: "Content-type", value: "text/html; charset=utf-8")
self.end_headers()

self.wfile.write(content.encode('utf-8'))
```

# BackEnd - WebServer

**def carrega\_turmas\_professor(self, login):** Define uma função que recebe o login como parâmetro

**cursor = conexao.cursor():** Inicializa um cursor para interagir com o banco de dados.

**cursor.execute("SELECT id\_professor, nome FROM dados\_login WHERE login = %s", (login,)):** Executa uma consulta SQL para buscar o ID do professor e o nome na tabela **dados\_login** com base no login fornecido como parâmetro.

**resultado = cursor.fetchone():** Obtém a primeira linha do resultado da consulta SQL anterior.

**cursor.close():** Fecha o cursor após a consulta ter sido realizada para liberar recursos do banco de dados.

**id\_professor = resultado[0]:** Extrai o ID do professor da primeira coluna e armazena na variável ( **id\_professor**)

**cursor = conexao.cursor():** Cria um novo cursor para interagir com o banco de dados.

**cursor.execute("SELECT turmas.id\_turma, turmas.descricao FROM turmas\_professor INNER JOIN turmas ON turmas\_professor.id\_turma = turmas.id\_turma WHERE turmas\_professor.id\_professor = %s",(id\_professor,)):** Executa uma segunda consulta SQL para obter os dados das turmas que estão vinculadas na tabela **turmas\_professor** de acordo com o **id\_professor** obtido na consulta anterior.

# BackEnd - WebServer

**Inicializo uma variável chamada linha\_tabelas vazia**

**For turma in turmas:** inicializo um laço de repetição, que grava em Id\_turmas o resultado da variável turmas na 1 posição, e a descrição recebe a descrição turma, que está armazenado em turma na segunda posição .

Crio uma variável para inserir uma imagem como botão que depois vamos colocar o link para o cadastro da atividade  
E uma variável linha que vai montar linha a linha o retorno das turmas correspondentes ao retorno da consulta.

Já faço a renderização da página carregando através dos contentes o nome do professor, o id\_professor e o login

# BackEnd - WebServer

```
# Se não for a rota "/login", continua com o comportamento padrão
super().do_GET()

def do_POST(self):
    # Verifica se a rota é "/enviar_login"
    if self.path == '/enviar_login':
        content_length = int(self.headers['Content-Length'])
        # Lê o corpo da requisição
        body = self.rfile.read(content_length).decode('utf-8')
        # Parseia os dados do formulário
        form_data = parse_qs(body, keep_blank_values=True)

        # Verifica se o usuário já existe
        login = form_data.get('email', [''])[0]
        senha = form_data.get('senha', [''])[0]

        if self.usuario_existente(login, senha):
            self.carrega_turmas_professor(login)
        else:
            cursor = conexao.cursor()
            cursor.execute("SELECT login FROM dados_login WHERE login = %s", (login,))
            resultado = cursor.fetchone()
```

# BackEnd - WebServer

Para que as turmas sejam exibidas logo em que o usuário acesse o sistema, devemos chamar a função que acabamos de escrever, conforme slide acima.

# BackEnd - WebServer

```
elif self.path == '/cad_turma':  
    content_length = int(self.headers['Content-Length'])  
    body = self.rfile.read(content_length).decode('utf-8')  
    form_data = parse_qs(body, keep_blank_values=True)  
    descTurma = form_data.get('descTurma', [''])[0]  
    id_professor = form_data.get('id_professor', [''])[0]  
    login = form_data.get('login', [''])[0]  
  
    print(f"Cad_turma, dados do formulário {descTurma}{id_professor}")  
  
    self.adicionar_turma_professor(descTurma, id_professor)  
    self.carrega_turmas_professor(login)
```

# BackEnd - WebServer

## Desafio



Agora vamos chamar cadastro de atividade quando selecionado o botão na frente de cada turma, e ao selecionar o gravar, gravaremos na tabela de atividade de no de Atividades\_Turma