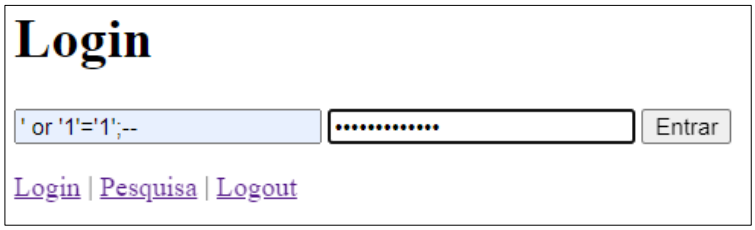
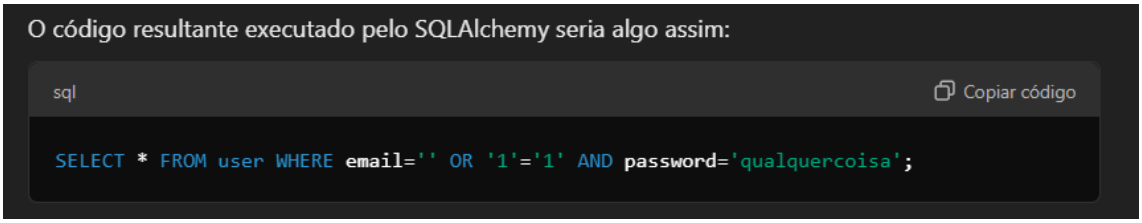
SQL-Injection

O SQL-Injection consiste em manipular os campos dos formulários que contêm os valores das variáveis que são usadas pelo backend para realizar a qery à BD.

Neste caso pretendemos fazer login na aplicação:

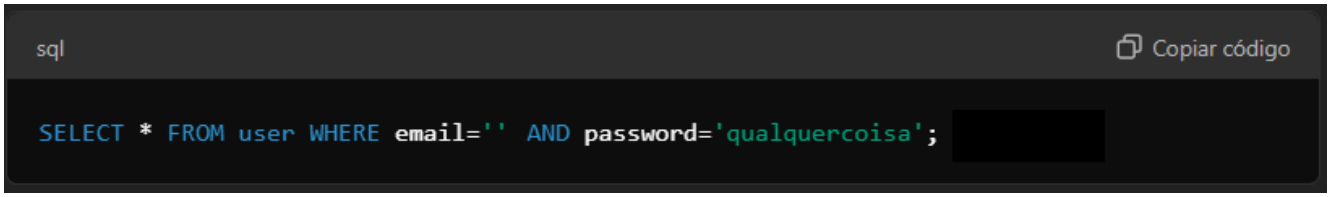


SQL-Injection: **' or '1'='1';--**



SELECT * FROM user WHERE email='' OR '1'='1' AND password='qualquercoisa';

NOTA: a query original **sem o SQL-Injection** seria:

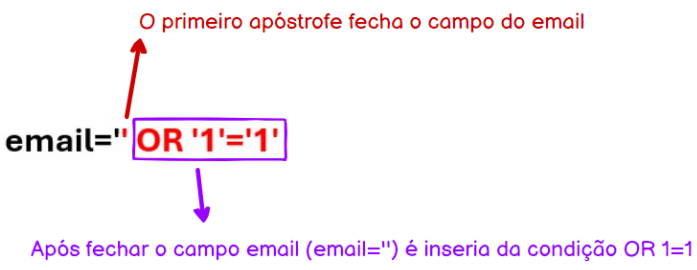


O que foi feito?

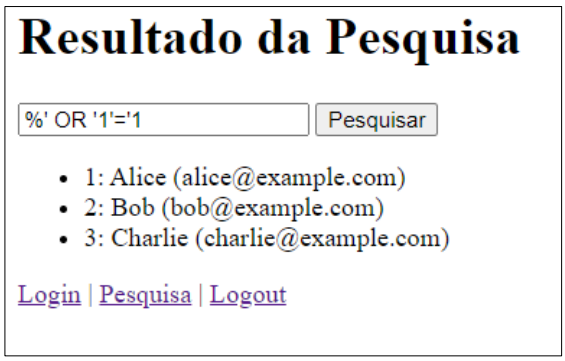
Conseguimos manipular e inserir no meio da query original a nossa condição **' or '1'='1';--**

Análise:

email='' OR '1'='1'



Após o Login com sucesso, obtemos acesso à página de Pesquisa, onde podemos pesquisar na base de dados por utilizadores inserindo o nome do utilizador.



SQL-Injection: **%' OR '1'='1**

A consulta resultante será:

```sql
SELECT * FROM user WHERE name LIKE '%%' OR '1'='1';
```

SELECT * FROM user WHERE name LIKE '%%' OR '1'='1';

NOTA: a query original **sem o SQL-Injection** seria:

```python
results = db.engine.execute(f"SELECT * FROM user WHERE name LIKE '%{query}%'")
```

O que foi feito?
Conseguimos manipular e inserir no meio da query original a nossa condição **%' OR '1'='1** ou também pode ser utilizado **%' OR '1'='1';--**

Aqui, o LIKE '%%' procura todos os utilizadores da tabela, pois a condição '1'='1' sempre será verdadeira, e o resultado será uma lista completa de todos os usuários na tabela.

Por Que Isso Funciona?

Em ambas as injeções, o código SQL foi alterado para incluir uma condição sempre verdadeira ('1'='1'), o que permite que todos os registos sejam retornados. Sem sanitização e parametrização adequada, esses valores são inseridos diretamente na consulta SQL.