# CAMP 2024 - Desafio Security

```
// Endpoint de login (vulnerável a SQL Injection)
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = await User.findOne({ where: { username, password } });
  if (user) {
    res.json({ message: 'Login successful', user });
  } else {
    res.status(401).json({ message: 'Invalid credentials' });
  }
});
```

1- Primeira melhoria para o código seria a inserção do metodo Try ... Catch para possíveis erros durante consulta de login ao banco de dados.
2- A segunda melhoria consiste em parar de utilizar a senha diretamente no código pois desta forma está mais sucessivo a ataques de SQL INJECTION, realizando um melhor tratamento das senhas.
3- Outra possível melhoria seria a implementação de uma autentificação por token.

```
// Endpoint de listagem de usuários (expondo dados sensíveis)
app.get('/users', async (req, res) => {
  const users = await User.findAll({ attributes: ['id', 'username',
'password'] });
  res.json(users);
});
```

1- Primeira melhoria para o código seria a inserção do metodo Try ... Catch para possíveis erros durante consulta de login ao banco de dados.
2- A segunda melhoria consiste em parar de utilizar a senha diretamente no código pois desta forma garantimos que apenas os campos necessários dos usuários sejam retornados na resposta do servidor.

```
// Endpoint de detalhe do usuário logado (expondo senha)
app.get('/profile', async (req, res) => {
  const { username } = req.query;
  const user = await User.findOne({ where: { username: username ?? null } });
  if (user) {
    res.json(user);
  } else {
    res.status(404).json({ message: 'User not found' });
  }
});
```

1- Foi removida a seleção senha da consulta do banco de dados e também foi adicionado o método Try... Catch  para capturar e lidar com erros que possam ocorrer durante a consulta ao banco de dados.