| | | | |
|---|---|---|---|
| **FGA 0238 - Testes de Software – Turma:** | 02 | **Semestre:** | 2023.2 |
| **Equipe:** | Grupo 10 - Assertivos | | |
| **Nomes:** | Letícia Resende Da Silva | **Matrículas:** | 211031118 |
| | Artur Jackson Leal Fontinele | | 211030943 |
| | Mateus Vinícius Ferreira Franco | | 200024868 |
| | Luana Souza Silva Torres | | 190033011 |
| | Lucas Rodrigues Monteiro | | 180125974 |
| | Raquel Temóteo Eucaria Pereira da Costa | | 202045268 |
| | Ricardo Augusto Valle Maciel | | 180077899 |


# Atividade 5 – Teste de Segurança

## 1 Aplicação Analisada

### 1.1. Identificação da Aplicação:

- MEC Energia API
- MEC Energia Web

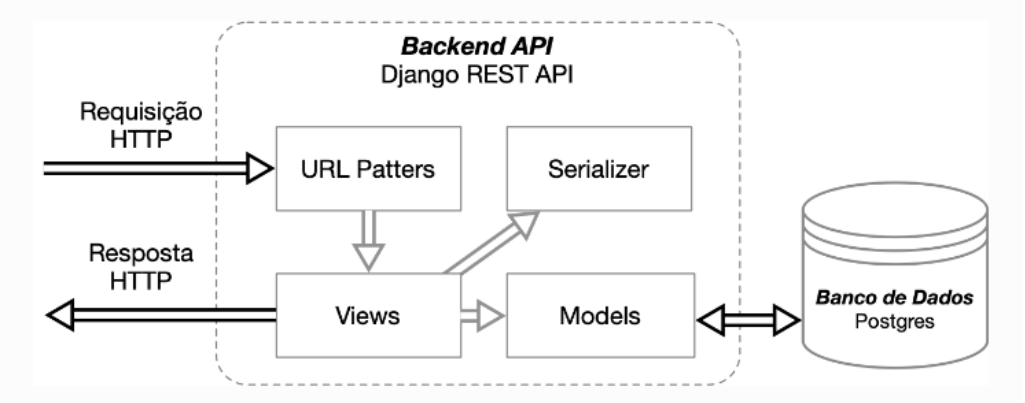
### 1.2. Descrição

O Sistema MEC-Energia foi desenvolvido com o propósito de oferecer suporte às Instituições de Ensino Superior (IES) no eficiente gerenciamento e na avaliação da adequação de contratos relacionados à conta de energia elétrica. Através do registro detalhado das faturas mensais de energia, o sistema proporciona a geração de relatórios especializados contendo recomendações precisas de ajustes nos contratos vigentes. O objetivo central dessas recomendações é otimizar a utilização de recursos, promovendo uma gestão mais econômica e sustentável da energia elétrica, alinhada às necessidades específicas e à realidade operacional das IES.

Este trabalho tem como objetivo realizar uma análise estática de código para identificar vulnerabilidades de segurança, utilizando o Teste de Segurança Estático (SAST). A ferramenta escolhida para essa análise é o SonarCloud, que examina o código fonte em busca de fragilidades, contribuindo para a identificação precoce de falhas e aprimoramento das práticas de programação. Essa integração reforça o compromisso do Sistema MEC-Energia com a segurança, garantindo eficiência na gestão energética e proteção dos dados, em conformidade com os mais altos padrões de segurança cibernética.

### 1.3. Linguagens

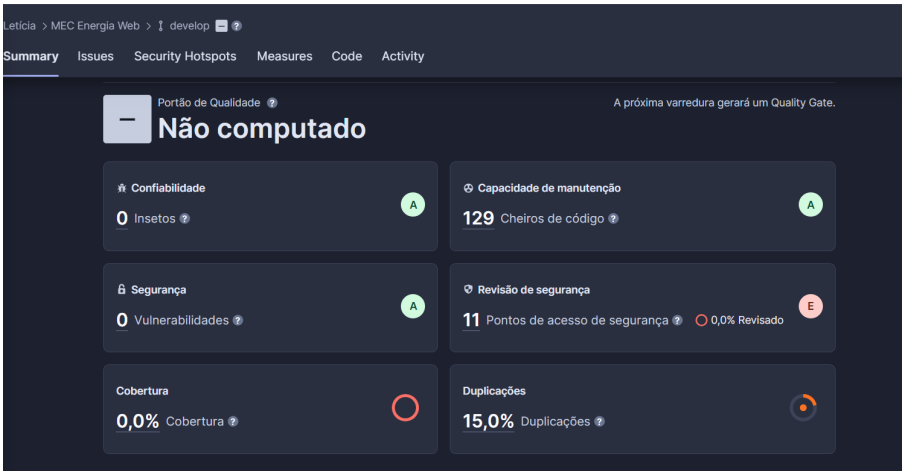
Python utilizando o Framework Django Rest




# 2 Visão Geral do Resultado

**MEC Energia API -** Não há vulnerabilidades registradas, porém foram verificados 18 hotspots e 13 bugs.

**MEC Energia Web -** Não há vulnerabilidades registradas, porém foram verificados 11 hotspots.



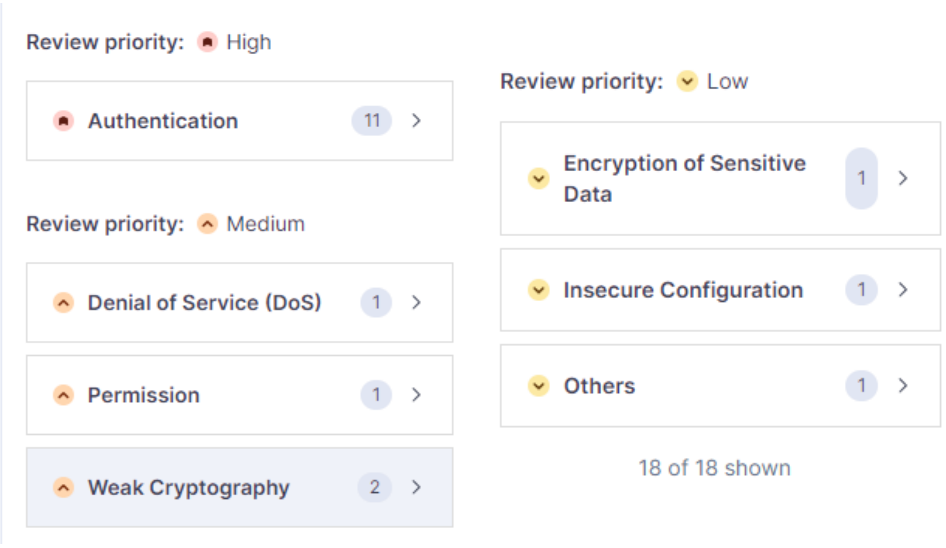

# 3 Vulnerabilidades

Não foram identificadas vulnerabilidades tanto no Back-End (MEC Energia API) quanto no Front-End (MEC Energia Web).

# 4 Hot Spots

**MEC Energia API -** Ao todo, foram verificados 18 Hot Spots, sendo:

- 11 de nível alto
- 4 de nível médio
- 3 de nível baixo



**MEC Energia Web:**

Ao todo, foram verificados 11 Hot Spots, sendo:

- 7 de nível médio
- 4 de nível baixo


# 5 Análise das Vulnerabilidades ou Hot Spots

**MEC Energia API**

## 5.1. Password Detected

### 5.1.1. Descrição

Credenciais codificadas são sensíveis à segurança

### 5.1.2. Solução

O código fornecido revela uma prática insegura ao lidar com senhas, expondo uma potencial vulnerabilidade de segurança. A senha é considerada fraca e previsível, o que pode comprometer a segurança do sistema. Para abordar essa questão, é recomendável adotar boas práticas de segurança ao lidar com credenciais sensíveis.

Uma abordagem mais segura envolve o uso de senhas fortes e a implementação de medidas que protejam as informações confidenciais. Armazenar senhas diretamente no código é uma prática de risco, pois expõe essas informações a possíveis ameaças. Em vez disso, é aconselhável utilizar técnicas de hash para armazenar apenas os hashes das senhas, contribuindo para a segurança do sistema.

**MEC Energia Web**

### 5.2. Copying using a glob pattern might inadvertently add sensitive data to the container.

#### 5.2.1. Descrição

A cópia de dados usando padrões globais no contexto do Docker pode ser arriscada, especialmente se isso envolver a cópia de dados confidenciais inadvertidamente para o contêiner. Os comandos COPY e ADD nos Dockerfiles podem aceitar padrões globais, o que pode resultar em cópias de arquivos que não eram originalmente destinados à imagem.

#### 5.2.2. Solução

Limitar o uso de globbing nos comandos COPY e ADD, preferindo listas explícitas de arquivos e diretórios necessários, em vez de cópias recursivas genéricas de todo o contexto. Utilizar o arquivo .dockerignore é crucial para excluir explicitamente dados sensíveis do contexto. Minimizar o tamanho do contexto, realizar revisões de segurança e conscientizar a equipe sobre a importância da revisão cuidadosa dos comandos de cópia também são medidas essenciais. Automatizar verificações de segurança no processo de construção contribui para identificar potenciais vazamentos de informações confidenciais, assegurando a integridade e segurança das imagens Docker resultantes.