# Software sem segurança não serve

Aluno: Thomaz Alves Da Costa

Instituição: Estácio de Sá

# Introdução

Neste relatório serão feitos alguns comentários trazendo o objetivo da prática, das ações realizadas na prática por mim quanto ao código quanto a estrutura de pastas, e sobre como poderá ser feita a execução do projeto para fins de análise.

# Objetivos da prática:

- Descrever o controle básico de acesso a uma API Rest;
- Descrever o tratamento de dados sensíveis e log de erros com foco em segurança;
- Descrever a prevenção de ataques de acesso não autorizado com base em tokens desprotegidos/desatualizados;
- Descrever o tratamento de SQL Injection em códigos-fonte; descrever o tratamento de CRLF Injection em códigos-fonte;
- Descrever a prevenção a ataques do tipo CSRF em sistemas web;

Principais pontos da prática:

- O código original foi refatorado, substituindo o método de criptografia anterior pela implementação do padrão JWT (JSON Web Token). Essa alteração proporciona uma abordagem moderna e segura para autenticação, utilizando tokens assinados digitalmente e compatíveis com padrões amplamente adotados em APIs RESTful.
- A arquitetura da API foi ajustada para remover o envio do token por parâmetro de URL. A partir da refatoração, o token JWT é transmitido exclusivamente via header HTTP (Authorization), utilizando o formato padrão Bearer Token.
- Foi incluído um middleware de autenticação responsável por validar o token JWT em cada requisição recebida. Essa validação compreende:
    - Verificação da existência do usuário associado ao token;
    - Verificação da validade e da integridade do token (assinatura);

    - Verificação do tempo de expiração do token, que está configurado com validade de 3 minutos, reforçando a proteção contra sessões inativas e uso indevido de tokens expirados.
    - Verificação se o token não é um token inválido.
- Adicionalmente, foi implementado um middleware de controle de acesso com o objetivo de verificar permissões específicas dos usuários. Esse componente analisa o perfil ou papel do usuário autenticado e determina se ele está autorizado a realizar determinadas requisições ou alterações na API, garantindo o princípio de mínimo privilégio.
- O método de busca de contratos foi reformulado para se alinhar à nova arquitetura de segurança:
    - O token JWT é enviado via Bearer Token, não sendo mais necessário repassá-lo como parâmetro.
    - Foi implementada validação contra SQL Injection nos parâmetros utilizados na query, utilizando expressões regulares.
    - A busca retorna dados simulados.

# Observações adicionais:

- A arquitetura do projeto foi reorganizada para refletir uma separação clara de responsabilidades:
    - mock/: Armazena arquivos simulados de usuários (usuários fictícios para testes).
    - middlewares/: Contém os middlewares de autenticação (JWT) e controle de acesso.
    - services/: Agrupa as regras de negócio utilizadas nos endpoints.
    - validators/: Inclui validadores específicos, como o validador de contratos.
    - .env: Utilizado para armazenar variáveis de ambiente, como a chave secreta JWT, substituindo o uso de variáveis hardcoded no código.

# Executando o projeto:

- Clone o repositório
- Acesse a pasta do projeto pelo terminal no VS Code.

- Instale as dependências do projeto: utilize o comando npm install, para instalar as dependências do package.json.
- Após a instalação das dependências, inicie o projeto com: npm start, que o projeto será executado na porta 3000.