**Documentação da Biblioteca SecureIO**

**1. Tecnologias Utilizadas**

* **Java JDK 21**: A biblioteca foi desenvolvida utilizando a versão mais recente do Java, garantindo acesso a recursos e melhorias de desempenho.
* **IDE IntelliJ**: Utilizada como ambiente de desenvolvimento integrado, o IntelliJ oferece ferramentas avançadas para desenvolvimento em Java, incluindo suporte a Spring Boot e funcionalidades de refatoração.
* **Spring Boot**: Todo o projeto é baseado no ecossistema Spring Boot, que facilita a configuração e desenvolvimento de aplicações Java, oferecendo uma arquitetura modular e práticas recomendadas para construção de APIs.
* **Postman**: Utilizado para realizar testes de API, tanto com entradas em formato HTML quanto JSON, simulando ataques de SQL Injection e XSS. O Postman permitiu a validação do comportamento da biblioteca sob diferentes cenários de entrada.

**Necessidade da Biblioteca SecureIO**

**Ataque via SQL Injection**

Em fevereiro de 2024, o grupo de hackers chamado "ResumeLooters" realizou um ataque massivo usando SQL injection contra 65 sites de empregos e lojas de varejo. O ataque, focado principalmente na região da Ásia-Pacífico (APAC), resultou no roubo de dados pessoais de mais de 2 milhões de candidatos a emprego. Esses dados incluíam nomes, endereços de e-mail, números de telefone e históricos de emprego. O grupo utilizou ferramentas de código aberto, como SQLmap, para automatizar a exploração de vulnerabilidades de SQL injection, permitindo controle total sobre os servidores de banco de dados comprometidos. Os dados roubados foram então colocados à venda em canais do Telegram.

**Ataque via Cross-Site Scripting (XSS)**

Em julho de 2024, a Netgear identificou uma vulnerabilidade grave de cross-site scripting (XSS) em vários modelos de seus roteadores WiFi 6. A falha permitia que atacantes injetassem scripts maliciosos nas interfaces web dos dispositivos, o que poderia levar à execução remota de código, se explorada corretamente. A Netgear rapidamente lançou uma atualização de firmware para mitigar o risco. O ataque poderia ser explorado para roubar credenciais de login ou comprometer a rede doméstica de forma mais ampla

Ambos os ataques destacam a importância da correção rápida de vulnerabilidades e da implementação de boas práticas de segurança em sites e dispositivos conectados.

Em um cenário onde ataques cibernéticos estão se tornando cada vez mais sofisticados, a segurança das aplicações web é uma prioridade crucial. **SQL Injection** e **Cross-Site Scripting (XSS)** são algumas das vulnerabilidades mais comuns e perigosas que podem comprometer dados sensíveis e a integridade das aplicações.

A biblioteca **SecureIO** foi desenvolvida com a necessidade de proporcionar uma solução prática e eficaz para mitigar esses riscos. Com um foco em segurança, a biblioteca auxilia desenvolvedores a implementar medidas de proteção em suas aplicações Java de forma simples e eficiente.

**Objetivo**

O objetivo do SecureIO é fornecer ferramentas de validação, sanitização de entradas e geração de relatórios que permitam detectar e prevenir tentativas de ataque, contribuindo assim para a segurança das aplicações.

**Instalação do SecureIO**

Para utilizar a biblioteca SecureIO em um projeto Java web, siga estas etapas:

1. **Clone o repositório do GitHub**: A biblioteca está disponível para download no GitHub.

git clone <https://github.com/Ricardo200021/SecureIO.git>

**Adicione a dependência no seu pom.xml**: Para integrar o SecureIO ao seu projeto Maven, adicione a seguinte dependência ao arquivo pom.xml:

|  |
| --- |
| <dependency>  <groupId>br.com.darkscreen</groupId>  <artifactId>SecureIO</artifactId>  <version>1.0.0-SNAPSHOT</version>  </dependency> |

**Configure a biblioteca**: Certifique-se de que as classes de configuração e interceptores estão corretamente integrados no seu aplicativo Spring Boot. Isso pode incluir ajustes na configuração de segurança e nas rotas da API.

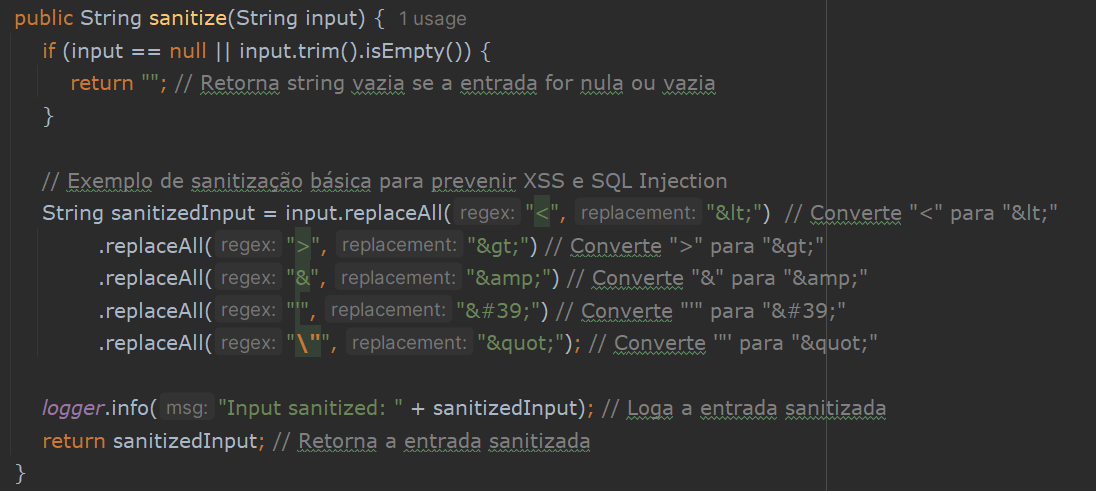
**Resumo das Classes e Funcionalidades**

A seguir, uma descrição detalhada de cada classe da biblioteca, suas funções principais e trechos de código relevantes:

**1. Sanitizer**

**Descrição**: Esta classe é responsável por sanitizar as entradas de dados, evitando ataques de XSS e SQL Injection através de um processo de limpeza de dados.

**Trecho principal:**

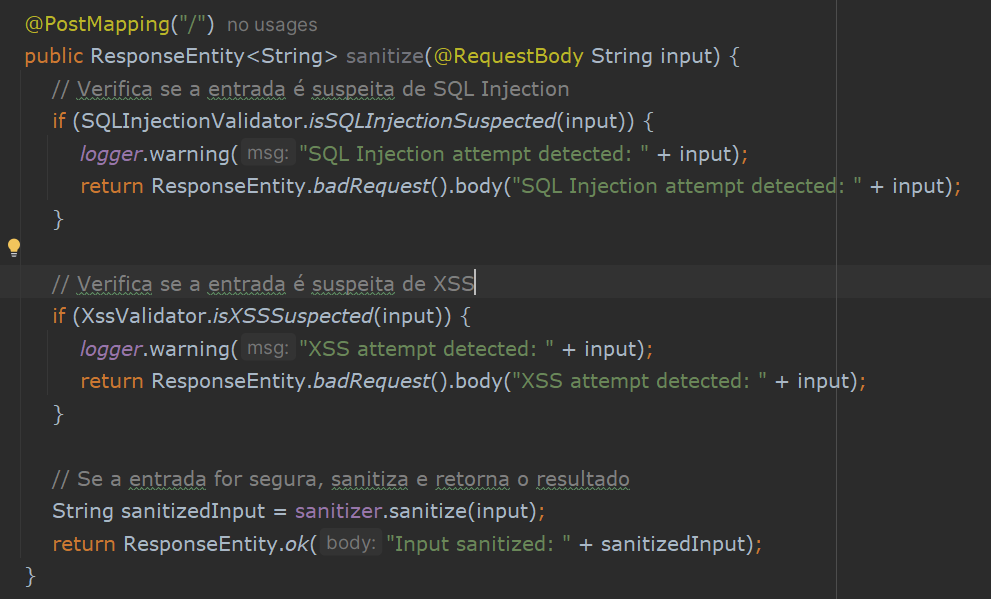


**Funcionalidade**: Converte caracteres especiais em entidades HTML, impedindo a execução de scripts maliciosos.

**SanitizerController**

**Descrição**: Controlador REST que gerencia as requisições para sanitização de entradas, utilizando a classe Sanitizer.

**Trecho principal**:

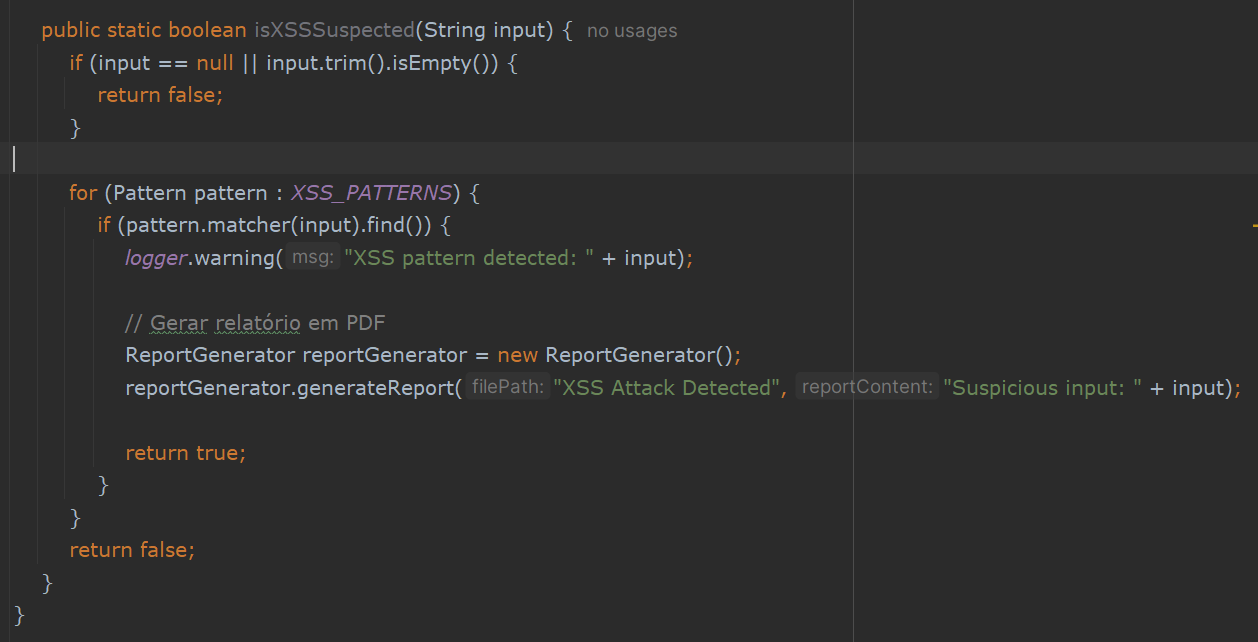


**Funcionalidade**: Recebe dados em formato JSON ou texto simples, aplica a sanitização e retorna o resultado ao usuário.

**SecureApplication**

**Descrição**: Contém métodos auxiliares para validar a entrada e verificar a suspeita de ataques XSS.

**Trecho principal**:

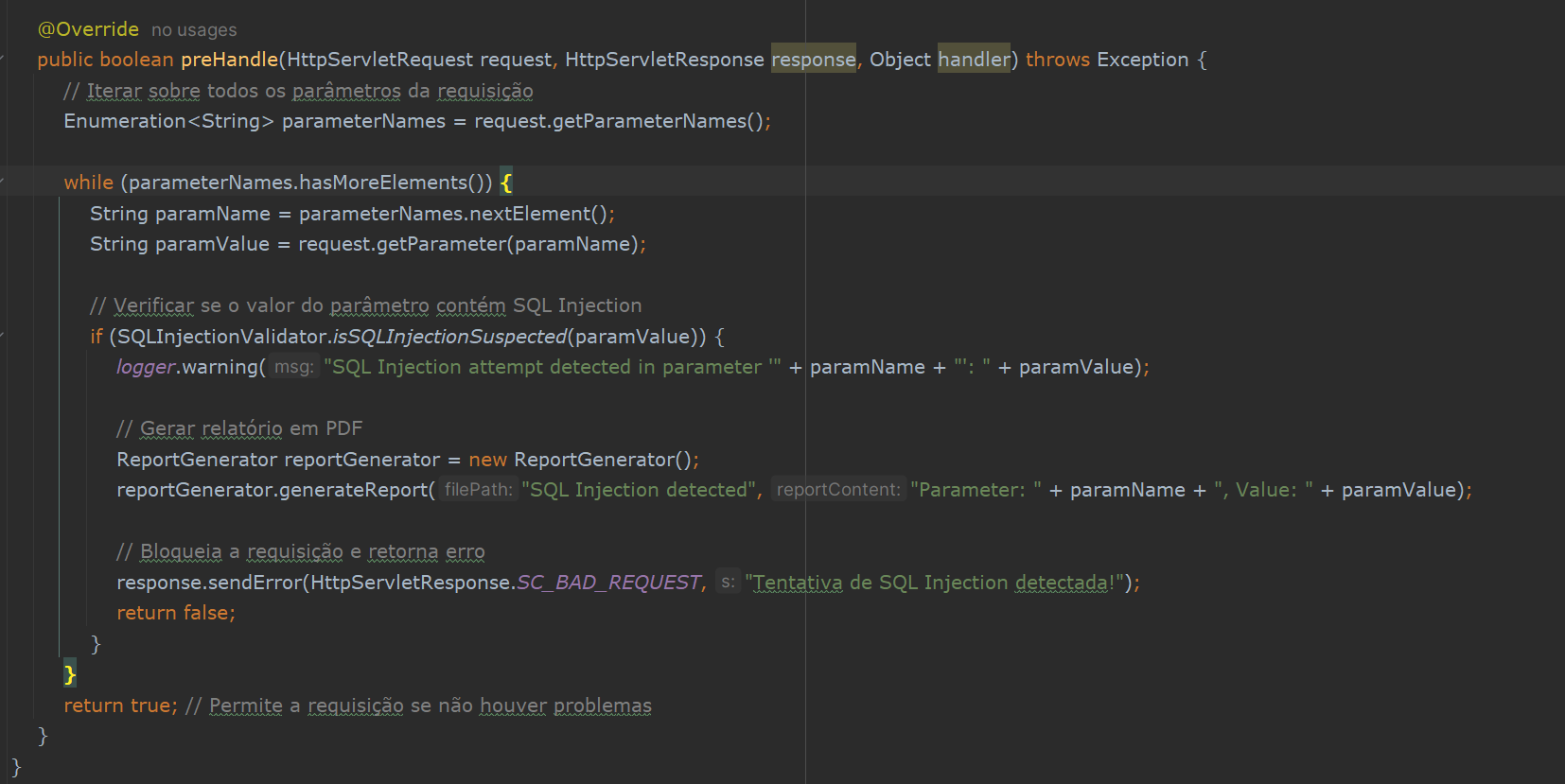


**Funcionalidade**: Avalia se a entrada possui padrões que indicam a presença de um ataque XSS.

**SQLInjectionInterceptor**

**Descrição**: Interceptor que analisa parâmetros de requisição HTTP em busca de possíveis tentativas de SQL Injection antes da execução.

**Trecho principal**:

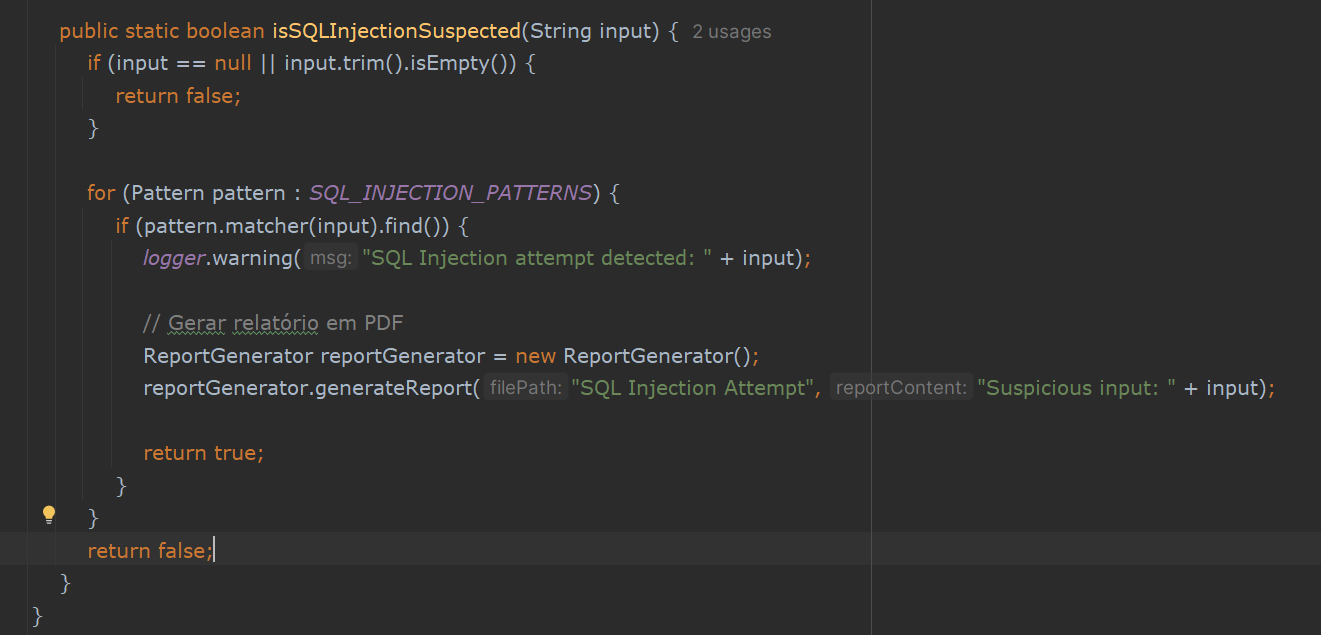


**Funcionalidade**: Interrompe a requisição caso um ataque seja detectado, enviando um erro 400 ao cliente.

**SQLInjectionValidator**

**Descrição**: Esta classe valida as entradas para identificar padrões de SQL Injection.

**Trecho principal**:

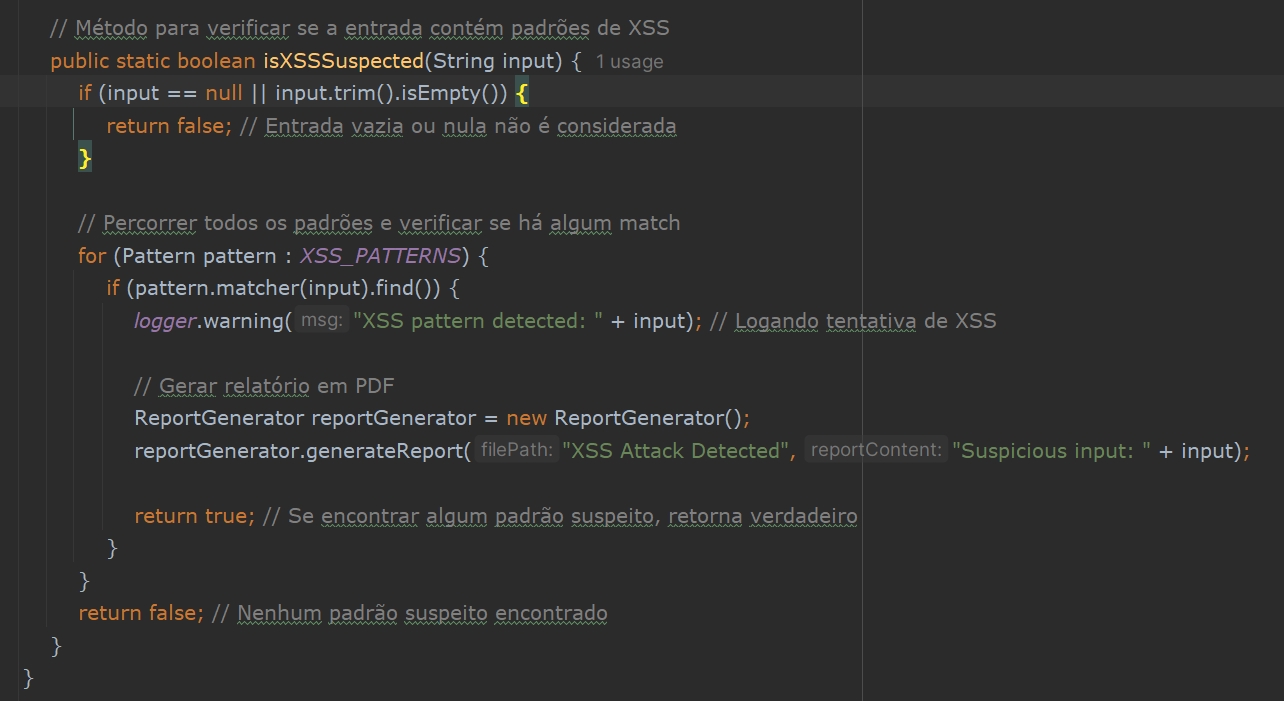


**Funcionalidade**: Verifica se a entrada contém comandos SQL suspeitos, retornando um booleano indicando a presença de um ataque.

**XssValidator**

**Descrição**: Similar ao SQLInjectionValidator, mas focada na detecção de XSS.

**Trecho principal**:

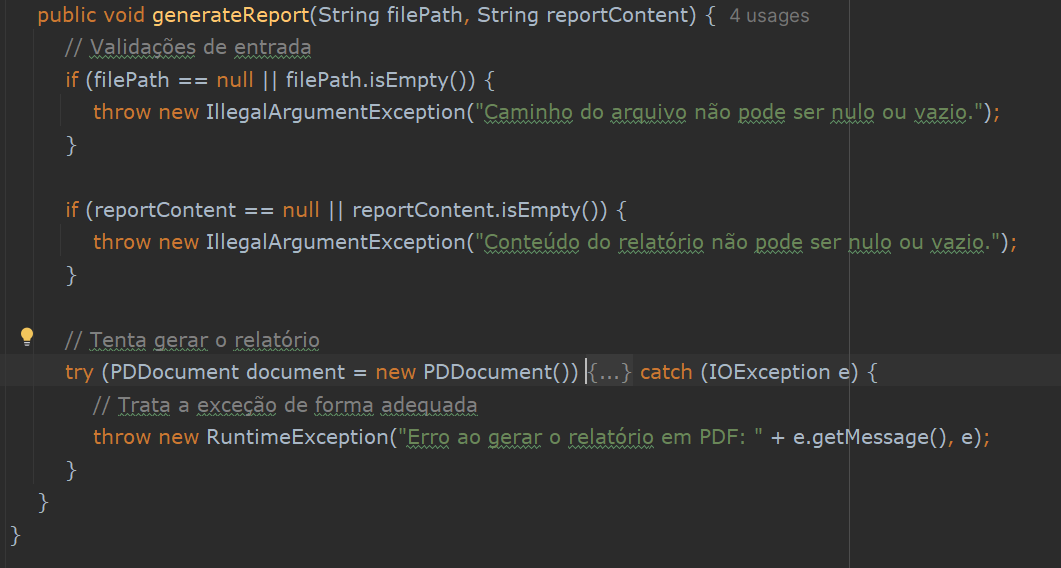


**Funcionalidade**: Identifica padrões que podem ser utilizados para injetar scripts maliciosos na aplicação.

**ReportGenerator**

**Descrição**: Gera relatórios em PDF sobre tentativas de ataques detectadas.

**Trecho principal**:



**Funcionalidade**: Documenta os incidentes de segurança, criando registros que podem ser usados para auditoria e análise.

**Conclusão**

A biblioteca **SecureIO** oferece uma solução prática e robusta para fortalecer a segurança de aplicações web em Java. Com funcionalidades que vão desde a sanitização de entradas até a geração de relatórios, o SecureIO ajuda desenvolvedores a implementar práticas de segurança eficazes. Recomendamos que você explore o repositório no GitHub, teste a biblioteca em seus projetos e contribua para o seu desenvolvimento contínuo.