Integração Segura entre API HMAC e API de Machine Learning com Java 17 e Spring Boot

Este artigo apresenta um guia passo a passo para integrar duas APIs em **Java 17 e Spring Boot**: uma API de autenticação com HMAC (que gera tokens JWT) e uma API de Machine Learning que consome esses tokens para realizar previsões.

O objetivo é implementar um sistema seguro, permitindo que apenas usuários autenticados possam acessar os recursos da API de Machine Learning.

Este artigo também traz exemplos práticos para testar cada etapa com o **Postman** e inclui códigos comentados para facilitar a compreensão.

Estrutura da Solução

1. **API HMAC**: Responsável por autenticar o usuário e gerar tokens JWT.
2. **API de Machine Learning**: Protegida por autenticação, acessível apenas com o token JWT fornecido pela API HMAC.

Arquitetura do Fluxo

1. A **API HMAC** valida as credenciais e gera um token JWT.
2. O cliente usa o **Postman** para enviar o token ao endpoint de previsão na **API de Machine Learning**.
3. A **API de Machine Learning** verifica a validade do token antes de processar a previsão.

1. Implementação da API HMAC

A **API HMAC** fornece um endpoint para autenticação e geração de tokens. Esse token é necessário para acessar a API de Machine Learning de maneira segura.

Endpoint de Autenticação e Geração de Token

Endpoint: /auth/token
Método: POST

O endpoint autentica o usuário com base em username e password e retorna um token JWT.

Código da API HMAC

```
@RestController
@RequestMapping("/auth")
public class AuthController {

    @PostMapping("/token")
    public ResponseEntity<String> authenticate(@RequestBody AuthRequest authRequest) {
        if ("meuUsuario".equals(authRequest.getUsername()) &&
"minhaSenha".equals(authRequest.getPassword())) {
            // Gera um token JWT se as credenciais estiverem corretas
            String token = Jwts.builder()
                    .setSubject(authRequest.getUsername())

.signWith(Keys.hmacShaKeyFor("minhaChaveSecretaParaTokenHMAC".getBytes()))
                    .compact();
            return ResponseEntity.ok(token);
        }
        // Retorna erro 401 caso as credenciais estejam incorretas
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Usuário ou senha
inválidos");
    }
}

class AuthRequest {
    private String username;
    private String password;

    // Getters e Setters
}
```

Explicação do Código:

- A classe AuthController contém o método authenticate() que verifica o username e password recebidos no corpo da requisição.
- Se as credenciais estiverem corretas, o método gera um token JWT com uma chave secreta (minhaChaveSecretaParaTokenHMAC).
- Retorna o token no corpo da resposta com 200 OK; caso contrário, retorna 401 Unauthorized.

Testando a API HMAC com Postman

Configuração do Postman

1. **URL**: http://localhost:8080/auth/token
2. **Método**: POST
3. **Body**: JSON (raw)

```
json
{
    "username": "meuUsuario",
    "password": "minhaSenha"
}
```

Resposta Esperada

Se as credenciais forem corretas:

```
json
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9"
```

Status: 200 OK

Se as credenciais forem inválidas:

```
json
"Usuário ou senha inválidos"
```

Status: 401 Unauthorized

2. Implementação da API de Machine Learning

A **API de Machine Learning** é protegida e só permite acesso a usuários com um token válido. O token, gerado pela API HMAC, deve ser incluído no cabeçalho da requisição como Bearer Token.

Endpoint para Previsão

Endpoint: /ml/predict
Método: POST

Código da API de Machine Learning

```
@RestController
@RequestMapping("/ml")
public class MachineLearningController {

    @PostMapping("/predict")
    public ResponseEntity<Map<String, String>> predict(@RequestHeader("Authorization")
String token, @RequestBody PredictionRequest predictionRequest) {
        try {
            // Valida o token JWT usando a mesma chave secreta
```

```
        Jwts.parserBuilder()

.setSigningKey(Keys.hmacShaKeyFor("minhaChaveSecretaParaTokenHMAC".getBytes()))
            .build()
            .parseClaimsJws(token.replace("Bearer ", ""));

        // Processa a previsão se o token for válido
        Map<String, String> response = new HashMap<>();
        response.put("result", "Previsão feita com sucesso para os dados: " +
predictionRequest.getData());
        return ResponseEntity.ok(response);
    } catch (Exception e) {
        // Retorna erro 401 se o token for inválido
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(Map.of("error", "Token
inválido"));
    }
  }
}

class PredictionRequest {
  private String data;

  // Getters e Setters
}
```

Explicação do Código:

- A classe MachineLearningController contém o método predict() que recebe um token JWT pelo cabeçalho Authorization e o valida.
- Caso o token seja válido, processa a requisição e retorna um resultado fictício de previsão.
- Se o token for inválido, o método retorna 401 Unauthorized.

Testando a API de Machine Learning com Postman

Configuração do Postman

1. **URL**: http://localhost:8081/ml/predict
2. **Método**: POST
3. **Headers**: Authorization: Bearer {token} (substitua {token} pelo token recebido da API HMAC)
4. **Body**: JSON (raw)

```
json
{
   "data": "dados para previsão"

}
```

Resposta Esperada

Se o token for válido:

json
```
{
   "result": "Previsão feita com sucesso para os dados: dados para previsão"
}
```

Status: 200 OK

Se o token for inválido:

json
```
{
   "error": "Token inválido"
}
```

Status: 401 Unauthorized

3. Resumo do Processo de Integração com Postman

Para integrar as duas APIs, siga os seguintes passos:

3.1. Obtenção do Token com a API HMAC

1. Enviar uma requisição POST para http://localhost:8080/auth/token.
2. No **Body**, incluir o username e password:

   json
   ```
   {
      "username": "meuUsuario",
      "password": "minhaSenha"
   }
   ```

Resposta: O token JWT gerado para autenticação.

json
```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9"
```

3.2. Consumo da API de Machine Learning com o Token

1. Enviar uma requisição POST para http://localhost:8081/ml/predict.
2. Incluir o **Header** Authorization: Bearer {token}, substituindo {token} pelo valor recebido da API HMAC.
3. No **Body**, enviar os dados para previsão:

```
json
{
   "data": "dados para previsão"
}
```

Resposta: O resultado da previsão ou erro, dependendo da validade do token:

```
json
{
   "result": "Previsão feita com sucesso para os dados: dados para previsão"
}
```

Conclusão

Este guia apresentou a criação e integração de duas APIs: uma para autenticação (HMAC) e outra para previsão (Machine Learning). A API HMAC fornece tokens JWT que permitem acesso seguro aos endpoints da API de Machine Learning.

Fluxo de Integração

1. **Obtenção do Token**: A API HMAC autentica o usuário e gera um token JWT.
2. **Consumo da API de Previsão**: A API de Machine Learning valida o token e processa a previsão.

Utilizando **Java 17** e **Spring Boot**, essas APIs oferecem uma estrutura segura e escalável para projetos que necessitam de autenticação robusta e previsões confiáveis.

EducaCiência FastCode para comunidade