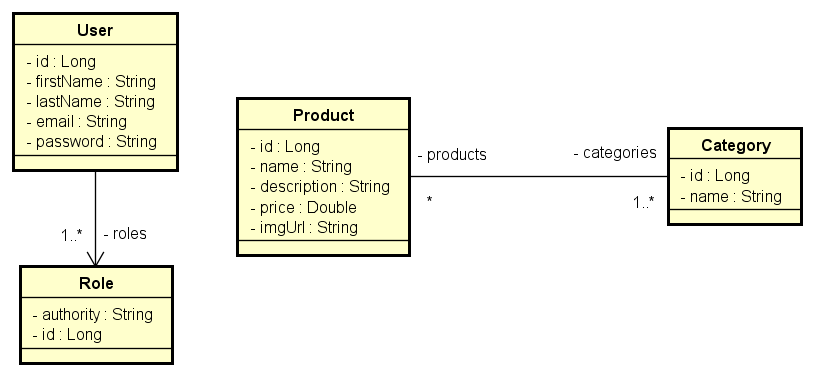
Java Spring Expert Capítulo 3

Validação e segurança

# Competências

* Modelo de dados de usuários e perfis
* Validação com Bean Validation
  + Annotations
  + Customizando a resposta HTTP
  + Validações personalizadas com acesso a banco
* Login e controle de acesso
  + Spring Security
  + OAuth 2.0
  + Token JWT
  + Autorização de rotas por perfil

# Modelo conceitual do DSCatalog



# 

# Referências sobre Bean Validation

<https://beanvalidation.org/>

<https://docs.jboss.org/hibernate/beanvalidation/spec/2.0/api/overview-summary.html>

<https://docs.jboss.org/hibernate/beanvalidation/spec/2.0/api/javax/validation/constraints/package-summary.html>

<https://www.baeldung.com/java-bean-validation-not-null-empty-blank>

<https://www.baeldung.com/spring-custom-validation-message-source>

<https://pt.stackoverflow.com/questions/133691/formatar-campo-cpf-ou-cnpj-usando-regex>

<https://regexlib.com/>

<https://regexr.com/>

# 

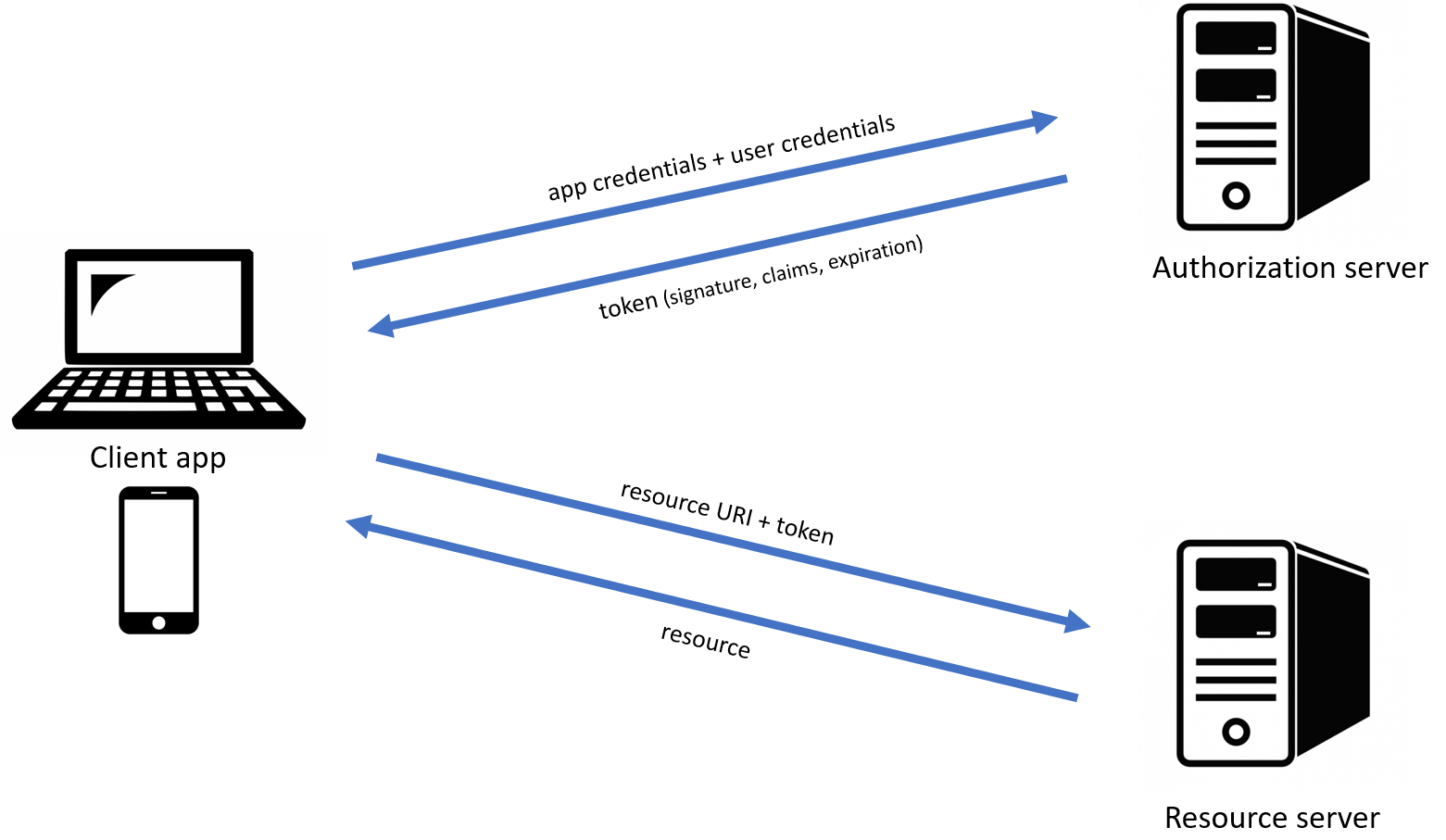
# Referências token JWT, autenticação e autorização

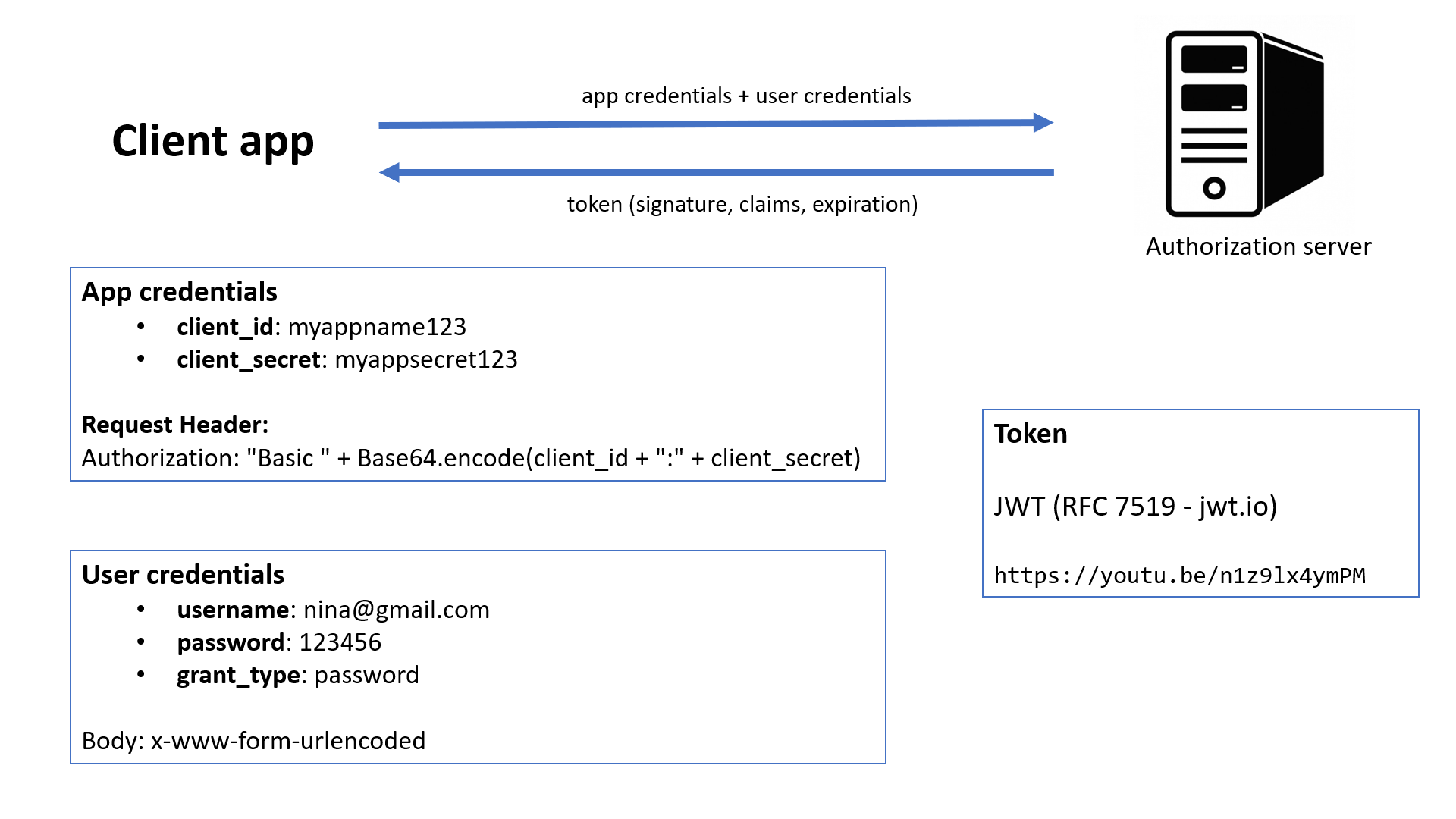
<https://jwt.io>

<https://www.youtube.com/watch?v=n1z9lx4ymPM>

# OAuth 2.0

<https://oauth.net/2/>





Projeto referência Spring OAuth2

<https://github.com/devsuperior/spring-boot-oauth2-jwt-demo>

**Recursos DSCatalog:**

<https://github.com/devsuperior/dscatalog-resources/tree/master/backend>

# Figma do DSCatalog

<https://www.figma.com/file/cNa2l3TqZXxbU6NBDPruNw/BDS-DSCatalog>

Spring Security

## Dependências

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

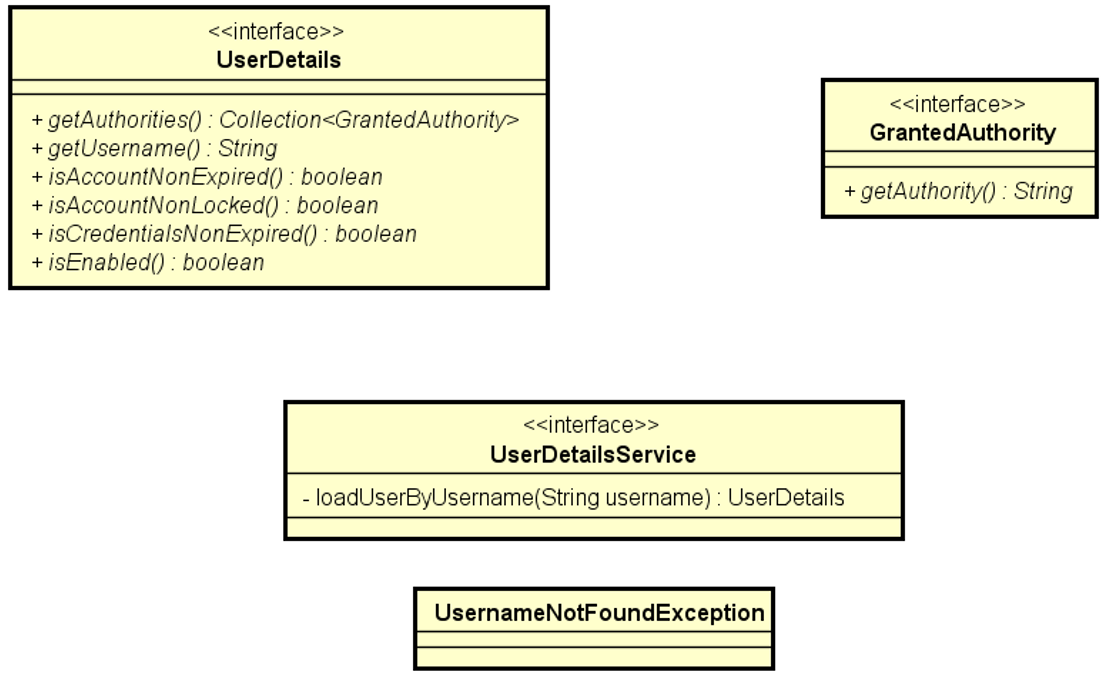
<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

## Checklist



# Spring OAuth2

## Dependências

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-oauth2-authorization-server</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-oauth2-resource-server</artifactId>

</dependency>

## Valores de configuração

security.client-id=${CLIENT\_ID:myclientid}

security.client-secret=${CLIENT\_SECRET:myclientsecret}

security.jwt.duration=${JWT\_DURATION:86400}

cors.origins=${CORS\_ORIGINS:http://localhost:3000,http://localhost:5173}

## Authorization Server

* Habilitar Authorization server
* Configurar token (codificação, formato, assinatura)
* Configurar autenticação / password encoder
* Registrar aplicação cliente

## Resource Server

* Configurar controle de acesso aos recursos
* Configurar CSRF, CORS
* Configurar token
* Liberar H2 Console no modo teste

## Requisição de login

Authorization:

Tipo: Basic

Username: client-id

Password: client-secret

Body:

Tipo: x-www-form-urlencoded

username: alex@gmail.com

password: 123456

grant\_type: password

## Controle de acesso por perfil e rota

@PreAuthorize("hasRole('ROLE\_ADMIN')")

@PreAuthorize("hasAnyRole('ROLE\_ADMIN', 'ROLE\_OPERATOR')")