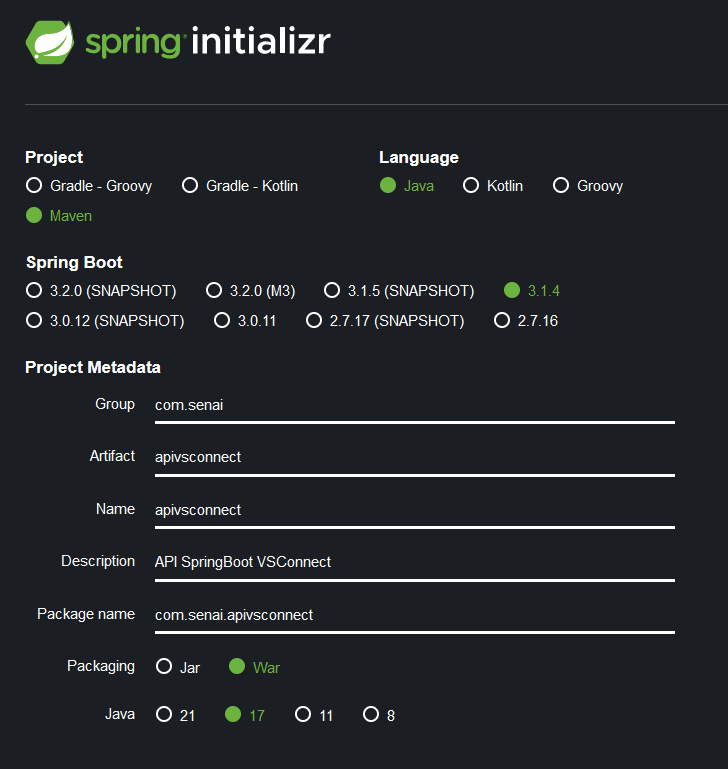
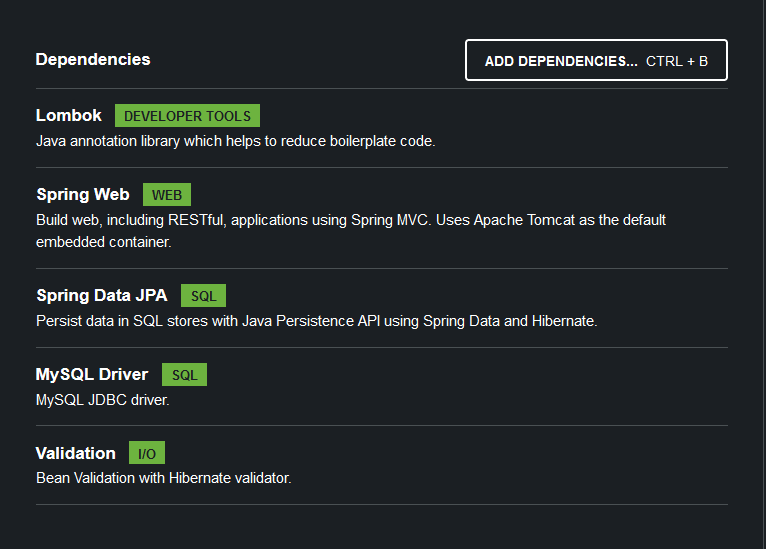
Spring Initializr

<https://start.spring.io/>  
  




## IntelliJ IDEA

Instalar Plugin JPA Buddy

File > Settings > Plugins – Instalar Plugin JPA Buddy

É um plugin que traz alguns templates de arquivos prontos comuns à um projeto que usa JPA

Instalar Plugin Lombok

File > Settings > Plugins – Instalar Plugin Lombok

É um plugin que faz associações de linting para as annotations do lombok

## Estrutura do Projeto

Adicionar configurações de banco de dados no application.properties

## Criar Models

Criar packages – models New JPA Entity, poderia ser um New Class, mas usando JPA Entity dá uma simplificada

## Criar Repositories

Criar package na raiz do projeto – repositories

New Spring Data Repository, poderia ser um New Interface, mas usando Spring Data Repository simplifica

## Criar Controllers

Criar package na raiz do projeto – controllers

New Java Class > Class

## Alterar o arquivo pom.xml

1. Tirar o <scope>provided</scope> da dependencia tomcat

## Criar Método GET

## Caso a porta 8080 esteja em uso:

Adicionar no arquivo application.properties (src/main/resources)

server.port=8090

## Criar Método GET por ID

## Método POST

## Criar DTO - Data Transfer Object

O DTO – Data Transfer Object, é um objeto que serve para “transferir” os dados entre locais diferentes.

Criar package na raiz do projeto – dtos

New Java Class > Record

Inserir como argumentos no construtor os dados que devem vir na request

Adicionar validações em cada argumento, Ex.: @NotBlank, @NotNull

## Criando o método POST de fato

Criar declaração do método findByEmail no UsuarioRepository

## Criar método no UsuarioController

@PostMapping(“/usuarios”) -> Indica o método HTTP e o recurso

## Criar Método PUT

## Criar Método DELETE

## Implementando Upload de Imagem

Criar diretório **static/img** na pasta **resources**

Criar package services na raiz do projeto para upload de imagem -> FileUploadService

Adicionar configuração application.propertie

## Adicionando o Spring Security

Adicionar dependência Spring Security no pom.xml e recarregar as dependências Maven

<https://docs.spring.io/spring-security/reference/getting-spring-security.html>

Ao adicionar o Spring Security, por padrão, todos os endpoints necessitam de autenticação  
  
Configurar UsuarioModel para autenticação

Implementar UserDetails na classe UsuarioModel

Configurar o getUsername e o getPassword

Retornar true nos outros métodos de validação (Indicam se o usuário está habilitado)

## Criar enum de TipoModel)

Em models, New Java Class > Enum

## No arquivo UsuarioModel trocar o tipo\_usuario para TipoModel e Configurar o getAuthorities

## Atualizar método UserDetails findByEmail(String email), que está no UsuarioRepository, pois será utilizado pelo Spring Security

Criar Service de autenticação (Java Class) -> AuthService

Implementar interface -> implements UserDetailsService

implementar método da interface UserDetailsService

## Criar classe de configuração do Spring Security (para remover a config padrão)

Criar package config -> criar classe SecurityConfig

Aqui você pode testar as requisições sem token

## Criar endpoint de Autenticação Login

Criar LoginDto – Java Class

Criar Controller de autenticação (Java Class)-> LoginController

## Alterar método de cadastro e update de Usuario para criptografar a senha (caso tenha feito sem)

## Implementar token JWT

Adicionar dependência do JWT no pom.xml e recarregar as dependências Maven

### Criar service do Token (TokenService)

Definir secret no application.properties

### Criar DTO para response do Token -> TokenDto – Java Class

### Retornar Token no método de Login

Testar Request de Login, deve retornar token

### Criar SecurityFilter (Java Class – pasta config)

Criar método recuperarToken()

Implementar herança OncePerRequestFilter

Implementar método doFilterInternal()

### Adicionar Filtro no SecurityConfig (caso não tenha feito)

Configurar os requestMatcher do securityConfig para aceitar requisições nos endpoints de login e de cadastro

Testar Requests

## Adicionando Swagger

Incluir dependência do swagger no pom.xml -> SpringDoc OpenAPI UI

Somente com isso já funciona no endereço:

http://localhost:8080/swagger-ui/index.html

Incluir annotation no arquivo ApiApplication.java

## Atualizando o Controller

Adicionar @Operation nos métodos

Adicionar @ApiResponses nos métodos

Repetir esses passos para cada método do Controller