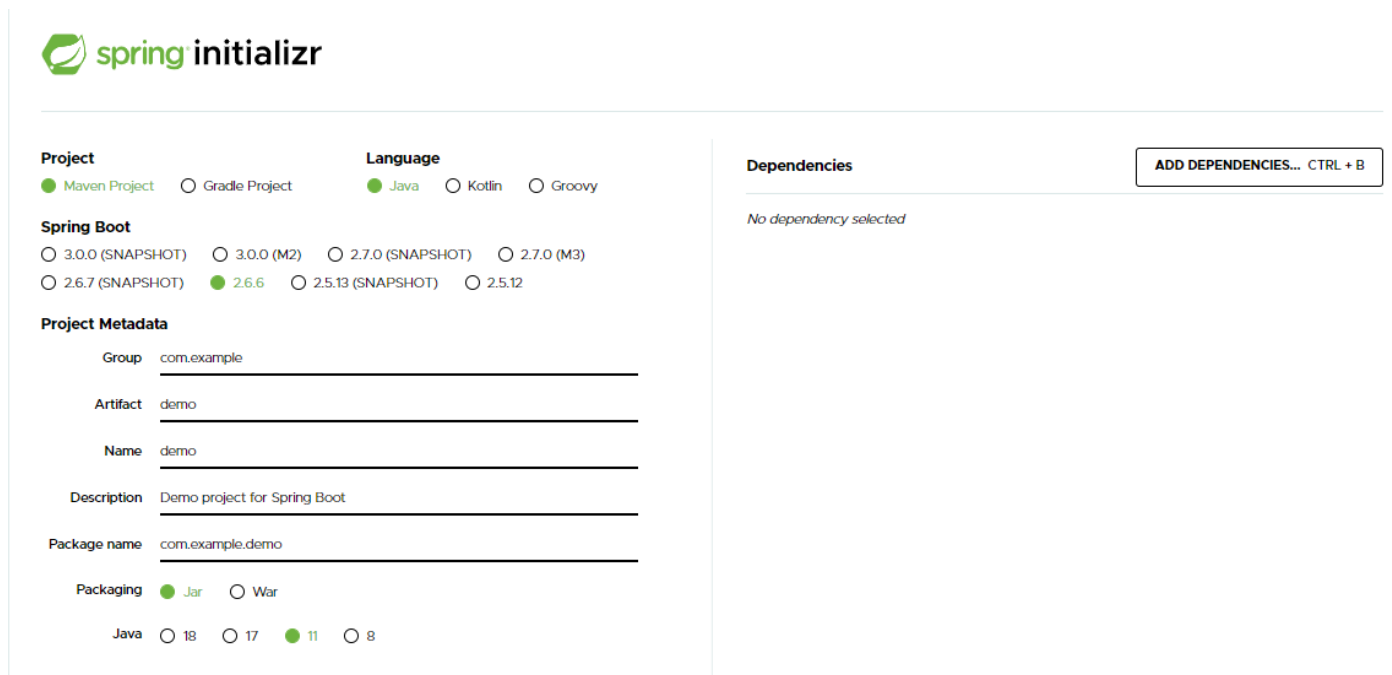


## Iniciando o Projeto:

Vamos criar uma aplicação web e analisar o seu resultado.

Para criar um novo projeto acessamos: [Spring Initializr](#).



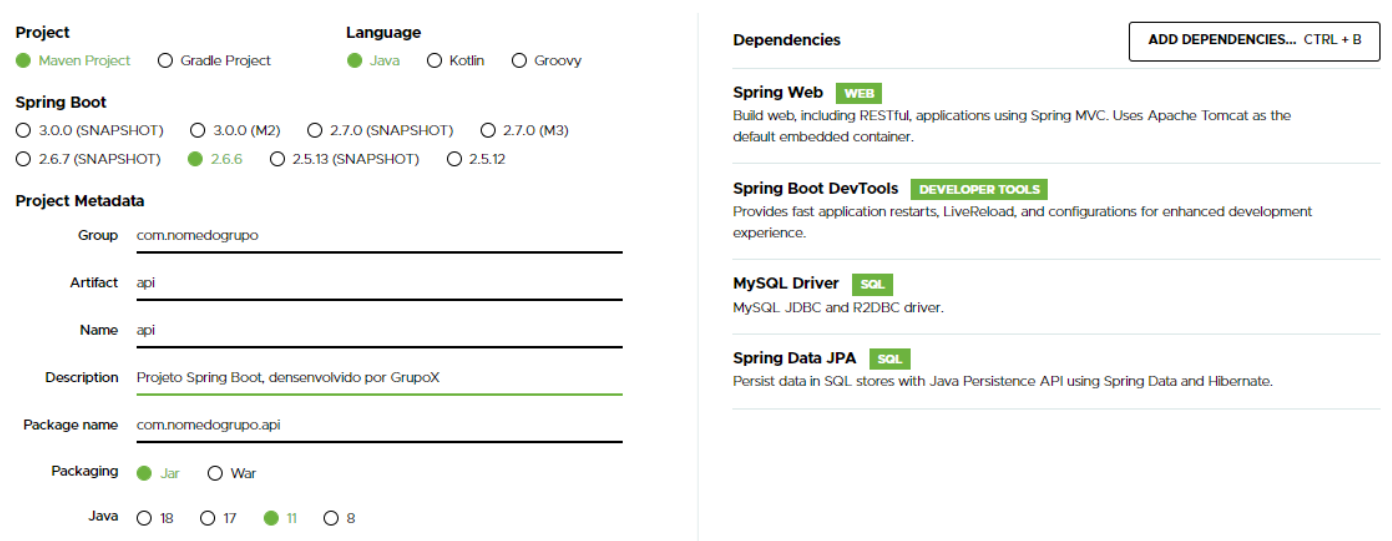
The image shows the Spring Initializr web form. On the left, under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', version '2.6.6' is selected. The 'Project Metadata' section has fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), and Package name (com.example.demo). Packaging is set to 'Jar' and Java version is '11'. On the right, the 'Dependencies' section is empty with the text 'No dependency selected' and a button 'ADD DEPENDENCIES... CTRL + B'.

A plataforma Spring Initializr serve para gerar um projeto Spring para IDEs que não configuram esse tipo de projeto por padrão, como é o exemplo as IDEs Eclipse e IntelliJ Community.

Ao lado esquerdo os campos: *Project*, *Language*, *Spring Boot* e *Project Metadata*, determinam o tipo de projeto, qual linguagem de programação e dados de conteúdo do projeto.

O lado direito temos a opção de inicialmente determinar Dependências, ou seja, bibliotecas que interagem com Spring Boot e que tem a principal característica de solucionar problemas comuns de desenvolvimento, nos fornecendo economia em tempo de desenvolvimento.

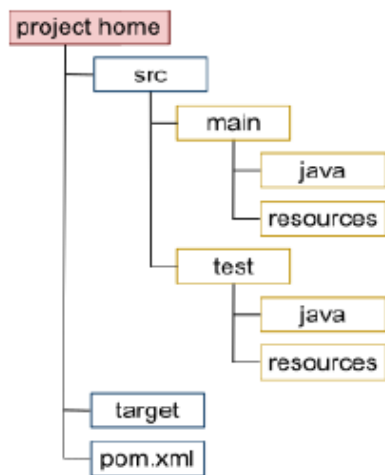
Para gerar nosso projeto deixaremos os campos conforme abaixo:



The image shows the Spring Initializr web form with the following configurations: Under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', version '2.6.6' is selected. The 'Project Metadata' section has fields for Group (com.nomedogrupo), Artifact (api), Name (api), Description (Projeto Spring Boot, desenvolvido por GrupoX), and Package name (com.nomedogrupo.api). Packaging is set to 'Jar' and Java version is '11'. On the right, the 'Dependencies' section contains three items: 'Spring Web' (WEB) with description 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.', 'Spring Boot DevTools' (DEVELOPER TOOLS) with description 'Provides fast application restarts, LiveReload, and configurations for enhanced development experience.', and 'MySQL Driver' (SQL) with description 'MySQL JDBC and R2DBC driver.'. There is also a 'Spring Data JPA' (SQL) dependency listed at the bottom.

## Maven

O Maven padroniza a criação de projetos. As estruturas dos projetos Maven sempre são iguais o que permite que independente da IDE que o programador esteja utilizando, ele pode importar um projeto Maven.



**src/main/java** - Arquivos .java do projeto (serão entregues ao cliente)

**src/main/resources** - Recursos do projeto (propriedades, xml, etc).

**src/test/java** - Arquivos .java dos testes (não vai para o cliente)

**src/test/resources** - Recursos de testes do projeto (propriedades, xml, etc).

**target** - Binários e documentações.

**pom.xml** - Project Object Model (pom.xml) (Veremos na sequência)

\*\*\*\*\* Caso exista necessidade de novas dependências basta acessar <https://mvnrepository.com/>

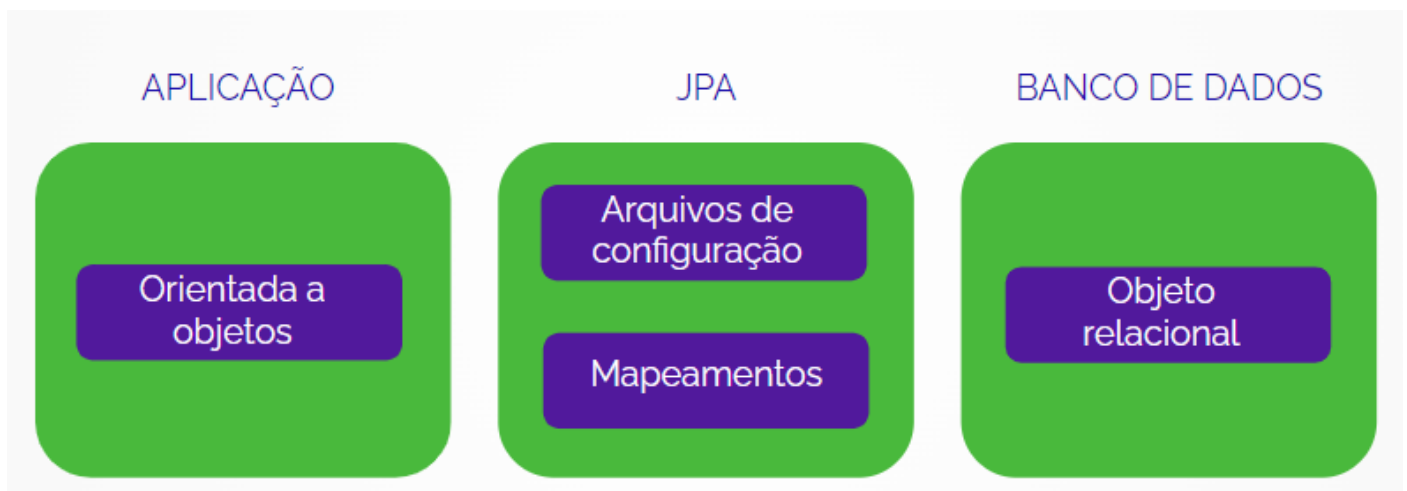
### Dependências:

**Spring Web:** Disponibiliza uma série de recursos para desenvolvimento de APIs Web, entre eles o servidor local embutido *Apache Tomcat* que nos permite testar realizar requisições em nossa API

**Spring Boot DevTools:** Disponibiliza uma atualização automática após salvar qualquer arquivo novo dentro da aplicação, sem a necessidade de reiniciar a aplicação.

**MySQL Driver:** Permite criar conexão com banco de dados MySQL.

**Spring Data JPA:** Usado no momento de persistência de dados em banco de dados relacional, fazendo a conversão de objetos e classes POO para objeto relacional.



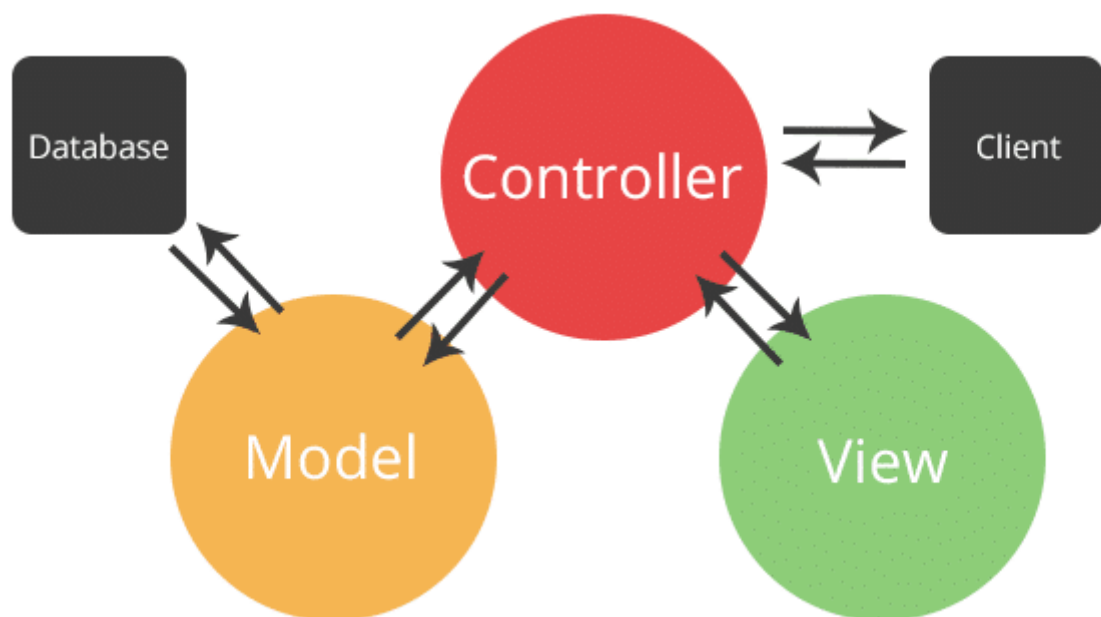
## Estrutura do projeto:

Podemos definir nosso projeto com a abordagem MVC:

**Model:** Camada Model ou Domain contém as informações de uma entidade, ou seja, tem a estrutura do que desejemos salvar nossos objetos em nosso banco de dados. Essa classe contém os construtores, atributos e métodos.

**Controller:** A camada responsável tanto por receber requisições e por enviar as respostas aos usuários, conforme cada tipo de requisição.

**View:** A camada de aplicação, onde implementamos as páginas de interação web.



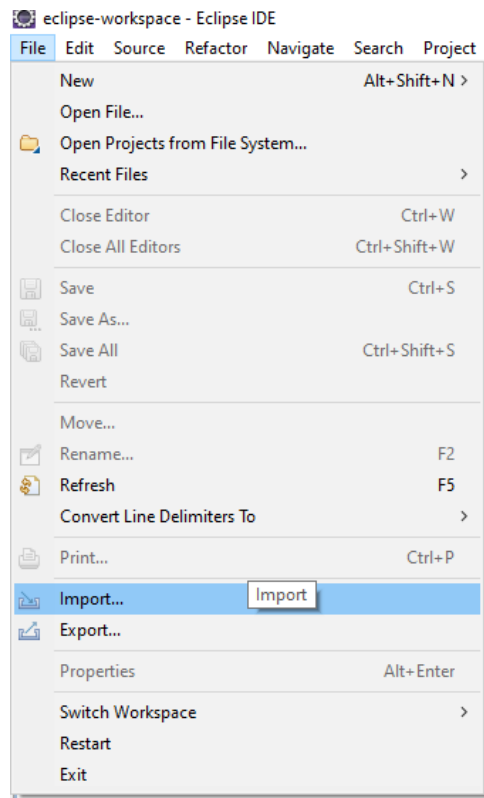
AtomsNetwork 

Disponível em [coremvc](https://coremvc.com)

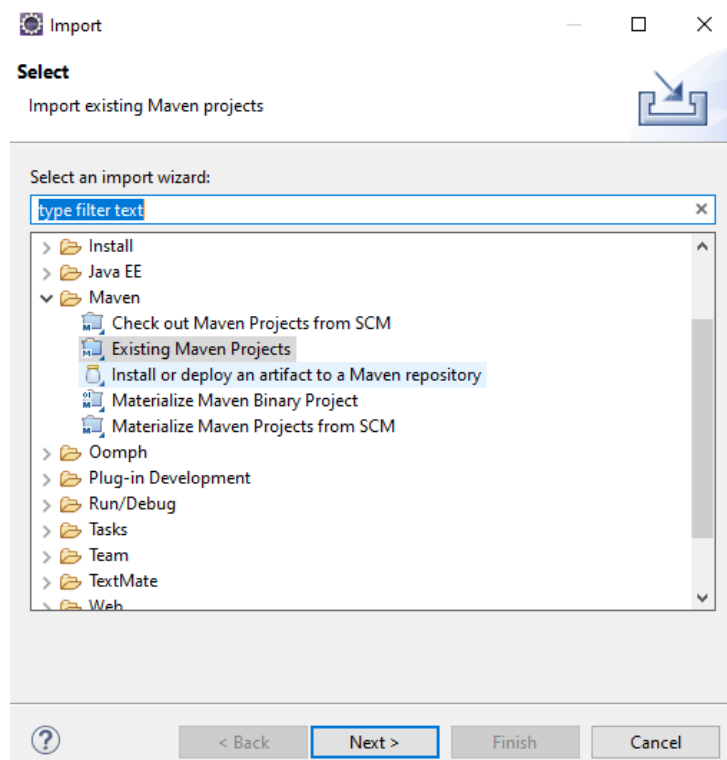
## Iniciando Projeto na IDE Eclipse

Após clicar no botão Gerar no site Sprint Initializr, um arquivo compactado deve ser gerado. Salve e descompacte o arquivo no local de sua preferência.

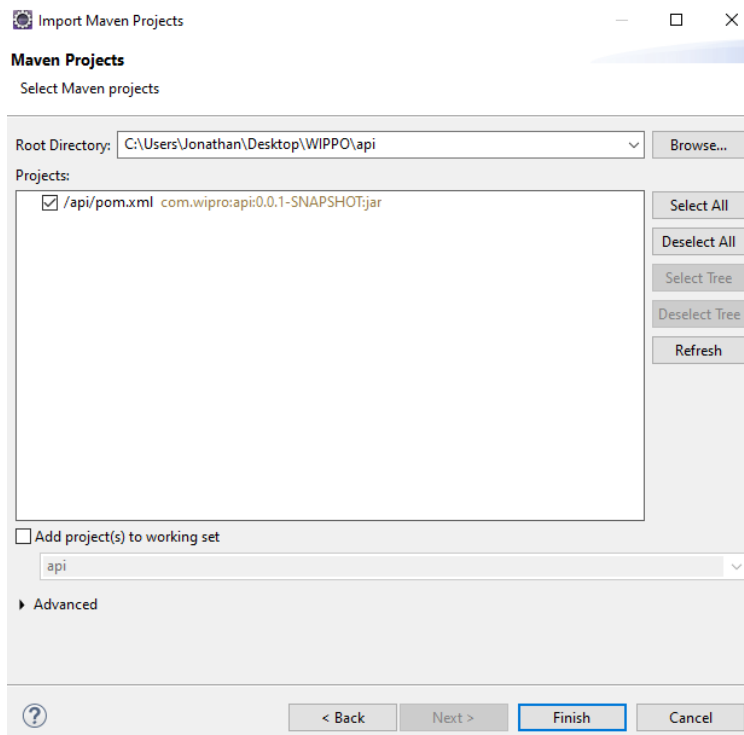
### No Eclipse File / Import



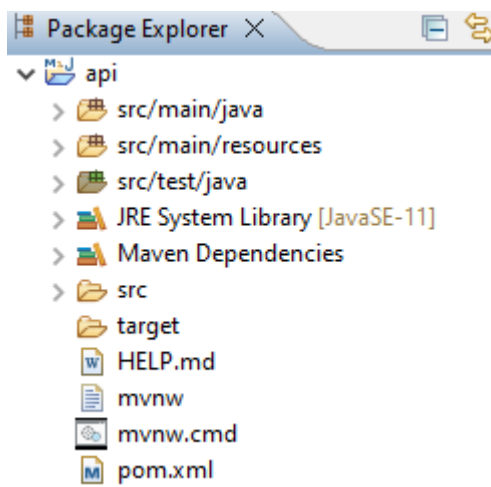
Na janela Import pesquise por Maven, em seguida selecione Existing Maven Projects e Next>



Por fim busque o local onde o arquivo está salvo e clique em Finish



Resultado após importação:



## pom.xml

Descreve todas as informações do projeto, como o group id, artifact id, version, que colocamos ao criar um projeto. Além disso, o pom.xml armazena informações das dependências (nossos jars), plugins, servidores e muitas coisas que podemos precisar no projeto.

## application.properties

Local: *application.properties* na pasta *src/main/resources*

Arquivo reservado para guardar “variáveis de ambiente” ou “propriedades do sistema”, porém efetivamente existe apenas propriedades (Properties). Uma propriedade é uma informação de configuração no formato chave/valor. Toda propriedade é portanto identificada por uma chave e possui um valor associado a esta.

```
server.port=8081
```

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:[NOME DO SEU BANCO DE DADOS]?createDatabaseIfNotExist=true
spring.datasource.username=[SEU USUÁRIO]
spring.datasource.password=[SUA SENHA]
spring.jpa.show-sql=true
```

*Links úteis:*

[Documentação Oficial](#)

Spring Boot — [Desmistificando o Application Properties](#)

Instalar MySQL, Workbench e POSTMAN :

[https://help.eset.com/esmc\\_install/71/pt-BR/mysql\\_windows.html](https://help.eset.com/esmc_install/71/pt-BR/mysql_windows.html)

<https://dev.mysql.com/downloads/installer/>

<https://www.postman.com/>

### MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »  
using my Oracle Web account

Sign Up »  
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account Login link. Otherwise, you can sign up for a free account by clicking the Sign Up link and follow instructions.

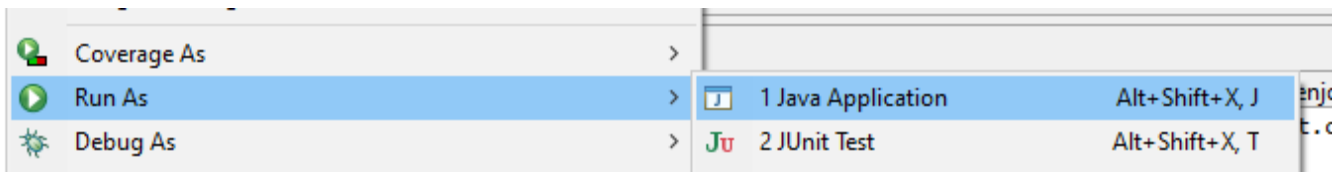
No thanks, just start my download.

Clicar em “no thanks, ...”

##### ATENÇÃO COM A SENHA \$\$\$\$\$\$\$\$\$\$

## Testar Aplicação

Basta clicar com botão direito/inverso do mouse sobre o pacote *com.nomelogrupo.api*, em seguida Run As / Java Application

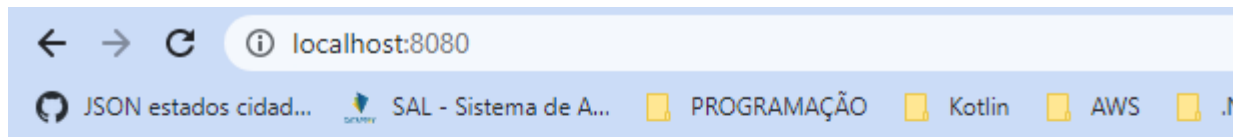


Console:

```
Spring Boot :: (v2.6.6)

2022-04-12 00:17:32.870 INFO 7952 --- [ restartedMain] com.nomelogrupo.api.ApiApplication : Starting ApiApplication using Java 11.0.14.1 on DESKTOP-3CQKGCT with PID 7952 (C:
2022-04-12 00:17:32.871 INFO 7952 --- [ restartedMain] com.nomelogrupo.api.ApiApplication : No active profile set, falling back to default profile: "default"
2022-04-12 00:17:32.918 INFO 7952 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false
2022-04-12 00:17:32.918 INFO 7952 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' proper
2022-04-12 00:17:33.427 INFO 7952 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2022-04-12 00:17:33.440 INFO 7952 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 4 ms. Found 0 JPA repository interfac
2022-04-12 00:17:34.122 INFO 7952 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-04-12 00:17:34.133 INFO 7952 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-04-12 00:17:34.133 INFO 7952 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.60]
2022-04-12 00:17:34.214 INFO 7952 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-04-12 00:17:34.378 INFO 7952 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1296 ms
2022-04-12 00:17:34.572 INFO 7952 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2022-04-12 00:17:34.655 INFO 7952 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.6.7.Final
2022-04-12 00:17:34.655 INFO 7952 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HCAN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-04-12 00:17:34.960 INFO 7952 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-04-12 00:17:34.988 INFO 7952 --- [ restartedMain] org.hibernate.dialect.Dialect : HikariPool-1 - Start completed.
2022-04-12 00:17:35.279 INFO 7952 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2022-04-12 00:17:35.287 INFO 7952 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jt
2022-04-12 00:17:35.326 WARN 7952 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-04-12 00:17:35.617 INFO 7952 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be
2022-04-12 00:17:35.650 INFO 7952 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : LiveReload server is running on port 35729
2022-04-12 00:17:35.660 INFO 7952 --- [ restartedMain] com.nomelogrupo.api.ApiApplication : Tomcat started on port(s): 8080 (http) with context path ''
2022-04-12 00:18:06.155 INFO 7952 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Started ApiApplication in 3.13 seconds (JVM running for 3.645)
2022-04-12 00:18:06.156 INFO 7952 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-04-12 00:18:06.156 INFO 7952 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-04-12 00:18:06.156 INFO 7952 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
```

Localhost:[PORTADEFINIDA]



## Whitelabel Error Page

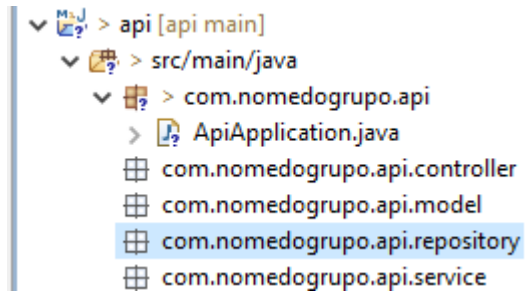
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Apr 12 00:18:06 BRT 2022

There was an unexpected error (type=Not Found, status=404).

No message available

## Criando pacotes



Criar os pacotes *controller*, *model*, *repository* e *service*.

Controller/ ou Resource

*Model* ou *Domain*

## Criar Classes

Dentro do pacote *model* criar a classe *Usuario*:

```
@Entity
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Integer id;

    String nome;

    String email;

    String senha;

    String sobrenome;

    String cartao;

    public Usuario() {
        super();
    }

    public Usuario(String nome, String email, String senha, String sobrenome, String cartao)
    {
        super();
        this.nome = nome;
        this.email = email;
        this.senha = senha;
        this.sobrenome = sobrenome;
        this.cartao = cartao;
    }

    // getters, setters, hashCode e equals ...
}
```



## Annotations Java

Basicamente é uma forma de transmitir metadados de maneira simples. As anotações fornecem dados sobre um programa que não faz parte do programa em execução.

Informações para o compilador — As anotações podem ser usadas pelo compilador para detectar erros ou suprimir avisos.

Processamento em tempo de compilação e implantação — As ferramentas de software podem processar informações de anotação para gerar código, arquivos XML e assim por diante.

Processamento em tempo de execução — Algumas anotações estão disponíveis para serem examinadas em tempo de execução.

Links Úteis:

[Entendendo-anotacoes-em-java](#)

<https://docs.oracle.com/javase/tutorial/java/annotations/>

## Repository

```
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Integer>{

}
```

## Service

```
@Service
public class UsuarioService {

    @Autowired
    private UsuarioRepository repository;

    public Usuario findById(Integer id) {
        Optional<Usuario> obj = repository.findById(id);
        return obj.orElse(null);
    }

    public List<Usuario> findAll() {
        return repository.findAll();
    }

    public Usuario update(Integer id, Usuario obj) {
        Usuario newObj = findById(id);
        newObj.setNome(obj.getNome());
        newObj.setSobrenome(obj.getSobrenome());
        newObj.setEmail(obj.getEmail());
        newObj.setSenha(obj.getSenha());
        return repository.save(newObj);
    }

    public Usuario create(Usuario obj) {
        return repository.save(obj);
    }

    public void delete(Integer id) {
        findById(id);
        repository.deleteById(id);
    }

}
```

## Controller

```
@RestController
@RequestMapping("/usuarios")
@CrossOrigin("*")
public class UsuarioController {

    @Autowired
    private UsuarioService service;

    @GetMapping("/{id}")
    public ResponseEntity<Usuario> GetById(@PathVariable Integer id) { // variavel presente
na uri
        Usuario obj = this.service.findById(id);
        return ResponseEntity.ok().body(obj);
    }

    @GetMapping
    public ResponseEntity<List<Usuario>> GetAll() {
        List<Usuario> list = service.findAll();
        return ResponseEntity.ok().body(list);
    }

    @PostMapping
    public ResponseEntity<Usuario> Post(@RequestBody Usuario usuario) {
        Usuario newObj = service.create(usuario);
        URI uri =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(newObj.getId()).toUri();
        return ResponseEntity.status(HttpStatus.GONE).body(service.create(usuario));
//
        return ResponseEntity.created(uri).build();
    }

    @PutMapping("/{id}")
    public ResponseEntity<Usuario> Put(@PathVariable Integer id, @RequestBody Usuario obj) {
        Usuario newUser = service.update(id, obj);
        return ResponseEntity.status(HttpStatus.ACCEPTED).body(newUser);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> Delete(@PathVariable Integer id) {
        service.delete(id);
        return ResponseEntity.noContent().build();
    }
}
```

## Testando POST

Requisição do tipo POST, End Point que passamos na classe Controller, clicar em Body, raw, JSON e descrever cada atributo conforme nosso Model

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:8080/usuarios
- Body Type:** JSON
- Body Content:**

```
1 {  
2   "nome": "Jonathan",  
3   "email": "j@j.com",  
4   "senha": "admin123",  
5   "sobrenome": "Ferreira",  
6   "cartao": "1111 2222 3333 444"  
7 }  
8
```

## Testando GETs

Requisição do tipo GET, End Point que passamos na classe Controller

GetByID:

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** localhost:8080/usuarios/1
- Response Type:** JSON
- Response Content:**

```
1 {  
2   "id": 1,  
3   "nome": "Jonathan",  
4   "email": "j@j.com",  
5   "senha": "admin123",  
6   "sobrenome": "Ferreira",  
7   "cartao": "1111 2222 3333 444"  
8 }
```

## GetAll:

GET

localhost:8080/usuarios

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1  [
2    {
3      "id": 1,
4      "nome": "Jonathan",
5      "email": "j@j.com",
6      "senha": "admin123",
7      "sobrenome": "Ferreira",
8      "cartao": "1111 2222 3333 444 "
9    }
10 ]
```

## Testando método Delete

Primeiro incluímos um novo usuário:

POST localhost:8080/usuarios

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▾

```
1 {
2   "nome": "Ana",
3   "email": "a@a.com",
4   "senha": "admin124",
5   "sobrenome": "Cruz",
6   "cartao": "2222 3333 4444 5555"
7 }
```

Body Cookies Headers (8) Test Results Status: 201 Created

Pretty Raw Preview Visualize **JSON** ▾

```
1 {
2   "id": 2,
3   "nome": "Ana",
4   "email": "a@a.com",
5   "senha": "admin124",
6   "sobrenome": "Cruz",
7   "cartao": "2222 3333 4444 5555"
8 }
```

Depois fazemos uma requisição tipo DELETE com o end point que determinamos em Controller + id que desejamos excluir

DELETE localhost:8080/usuarios/2

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCR
	Key	Value	Descrip

Body Cookies Headers (6) Test Results Status: 204 No Content

Pretty Raw Preview Visualize **Text** ▾

1

## Testando PUT

Assim como o método POST passamos via endpoint o id que desejamos alterar e no corpo da requisição os dados novos

The screenshot shows a REST client interface with a PUT request to `localhost:8080/usuarios/1`. The 'Body' tab is selected, showing a JSON payload. Below the request, the 'Test Results' tab shows the response in 'Pretty' format, which is a JSON object containing the updated user data.

**Request:**

```
PUT localhost:8080/usuarios/1
```

**Body:**

```
{  "nome": "Jonathan",  "email": "j2@52.com",  "senha": "senhasenha",  "sobrenome": "Ferreira",  "cartao": "2223 3334 4445 5556"}
```

**Response (Pretty):**

```
{  "id": 1,  "nome": "Jonathan",  "email": "j2@52.com",  "senha": "senhasenha",  "sobrenome": "Ferreira",  "cartao": "1111 2222 3333 444 "}
```

\*atenção ao número do cartão que permanece o inalterável, mesmo com a tentativa de mudança na linha 6