



PROJETO COM BÁSICO COM SPRING

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

HERCULANO DE BIASI

herculano.debiasi@unoesc.edu.br



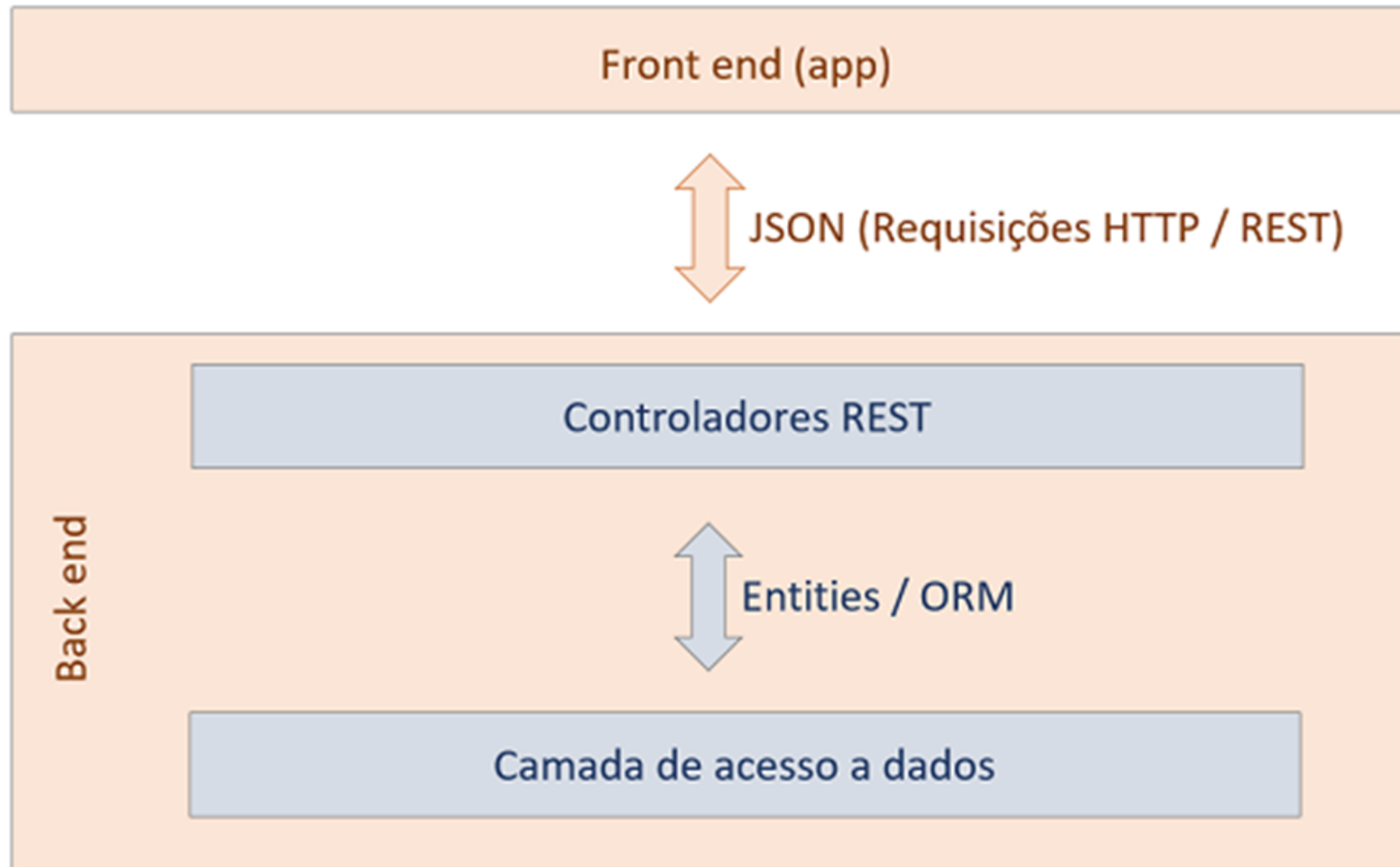
TÓPICOS

- Arquitetura do projeto
- Configuração do projeto
- *Plug-ins*
- SGBD H2
- Estrutura de pacotes
- Record
- Repositório
- *Command line runner*
- Refatoração para JPA
- Consultas personalizadas
- Controlador v.1
- *Front-end*
- Camada de serviço
- Controlador v.2
- Controlador v.3



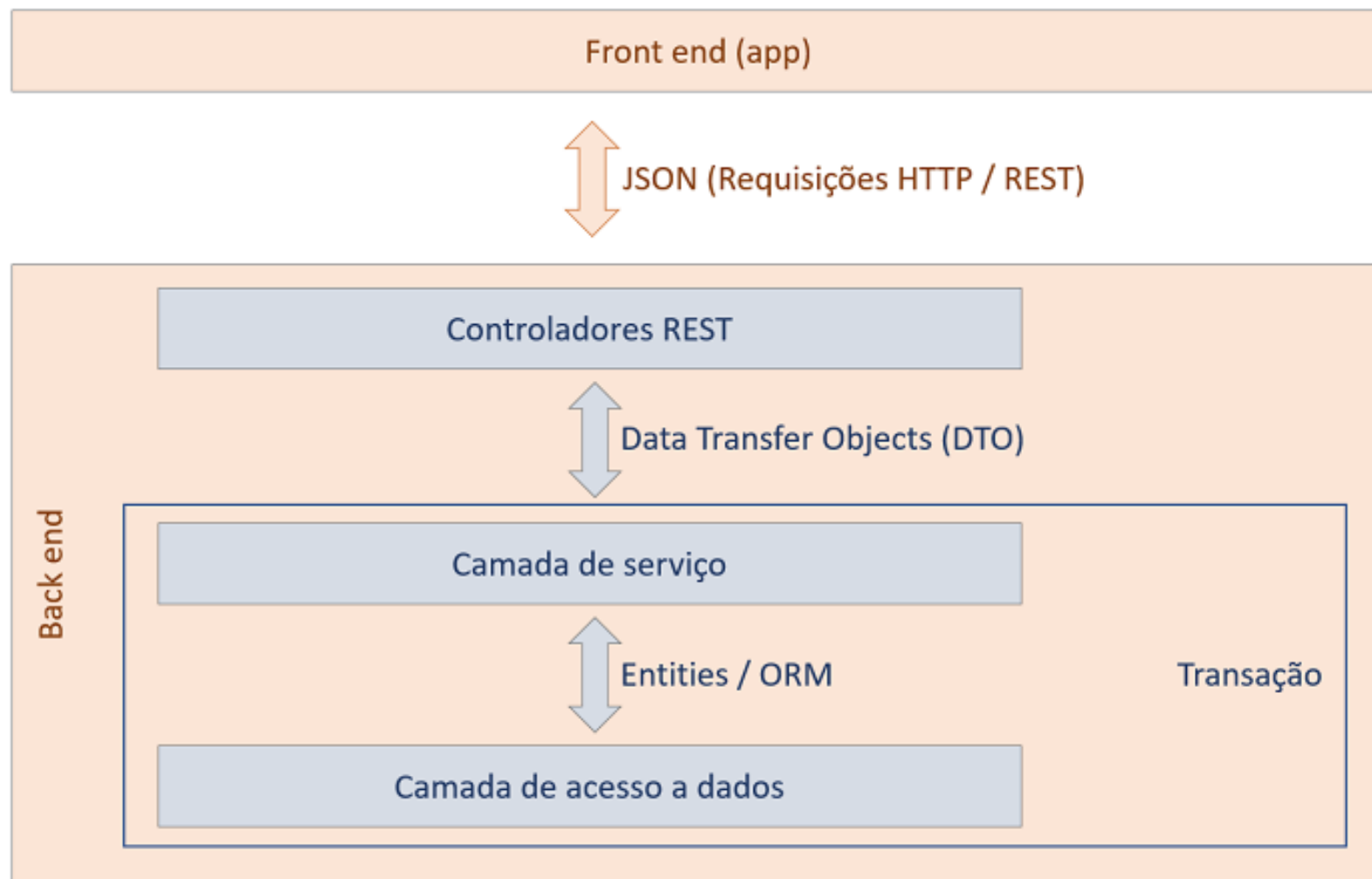
ARQUITETURA DO PROJETO

■ Arquitetura inicial do projeto



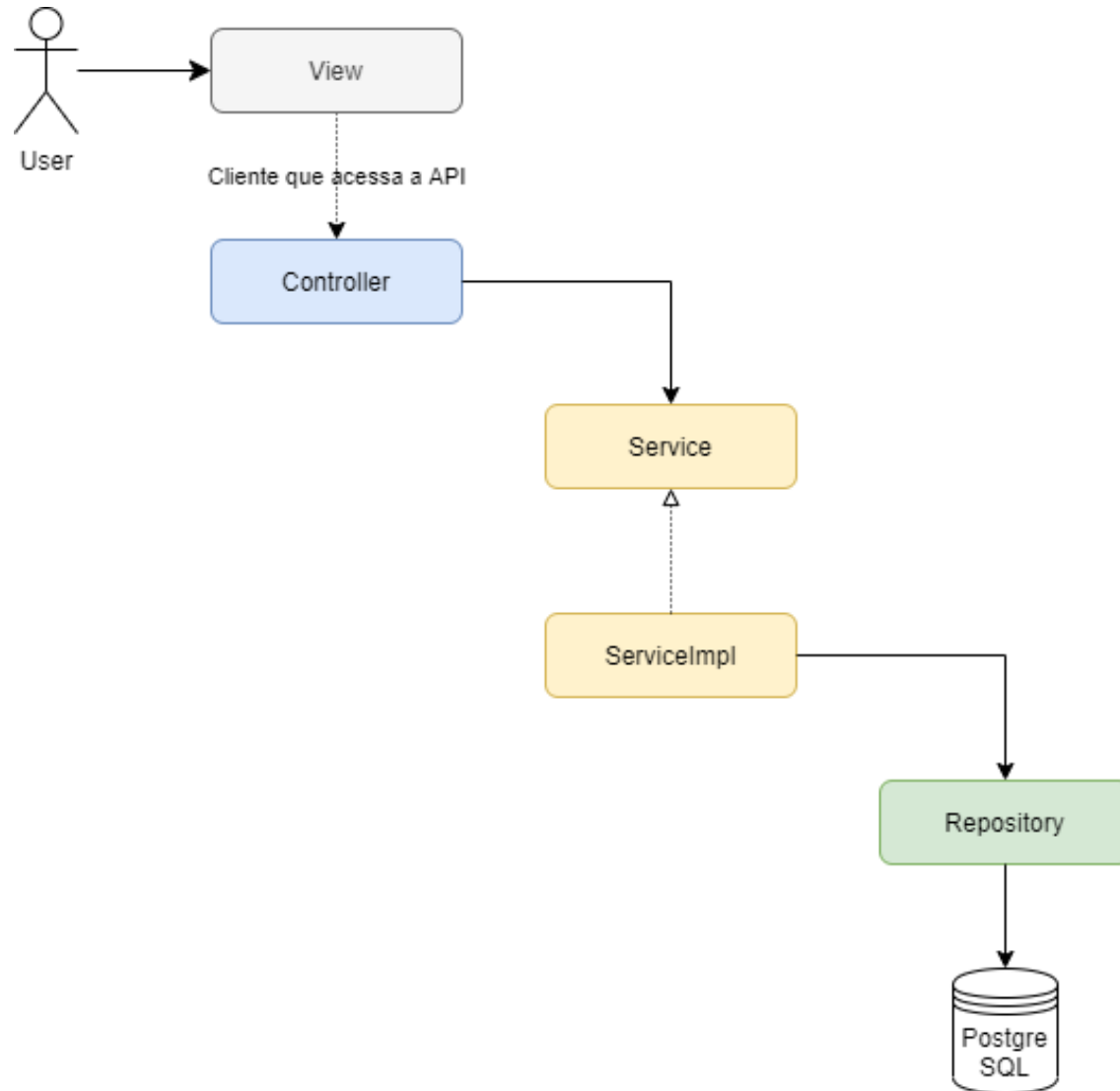
ARQUITETURA DO PROJETO

■ Arquitetura final do projeto



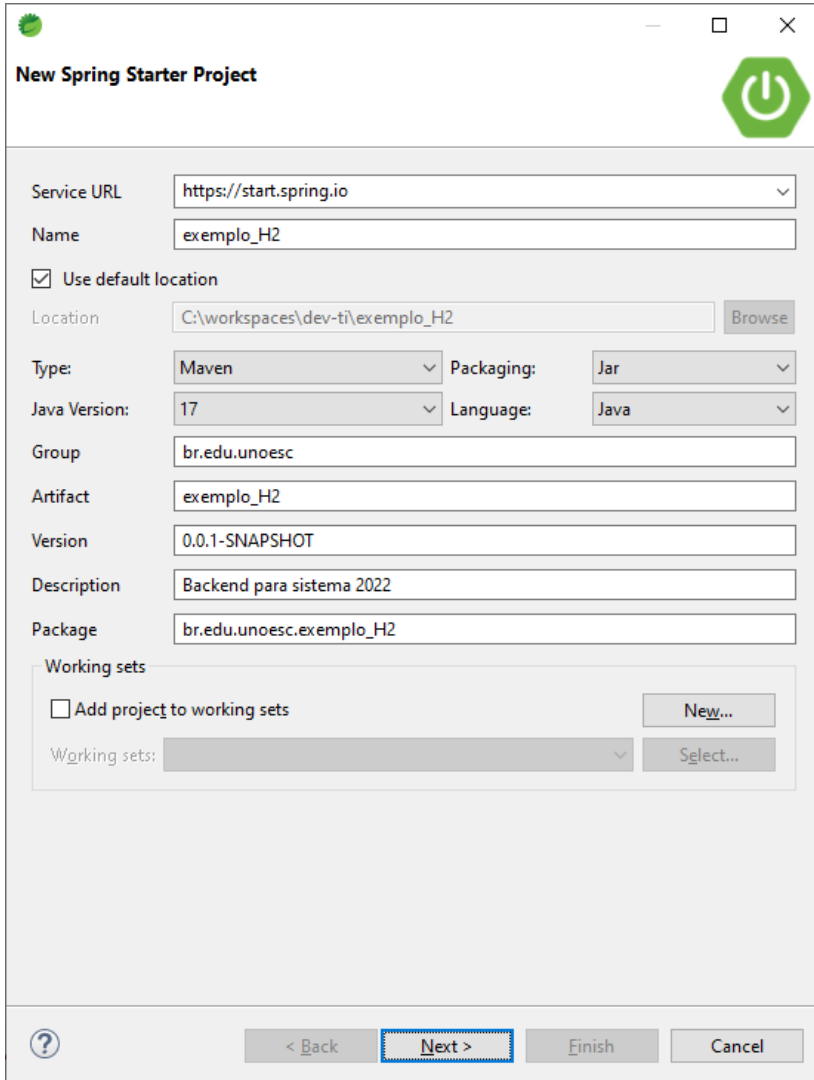
ARQUITETURA DO PROJETO

■ Arquitetura



CONFIGURAÇÃO DO PROJETO

■ Importe o arquivo como um projeto do Maven



New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

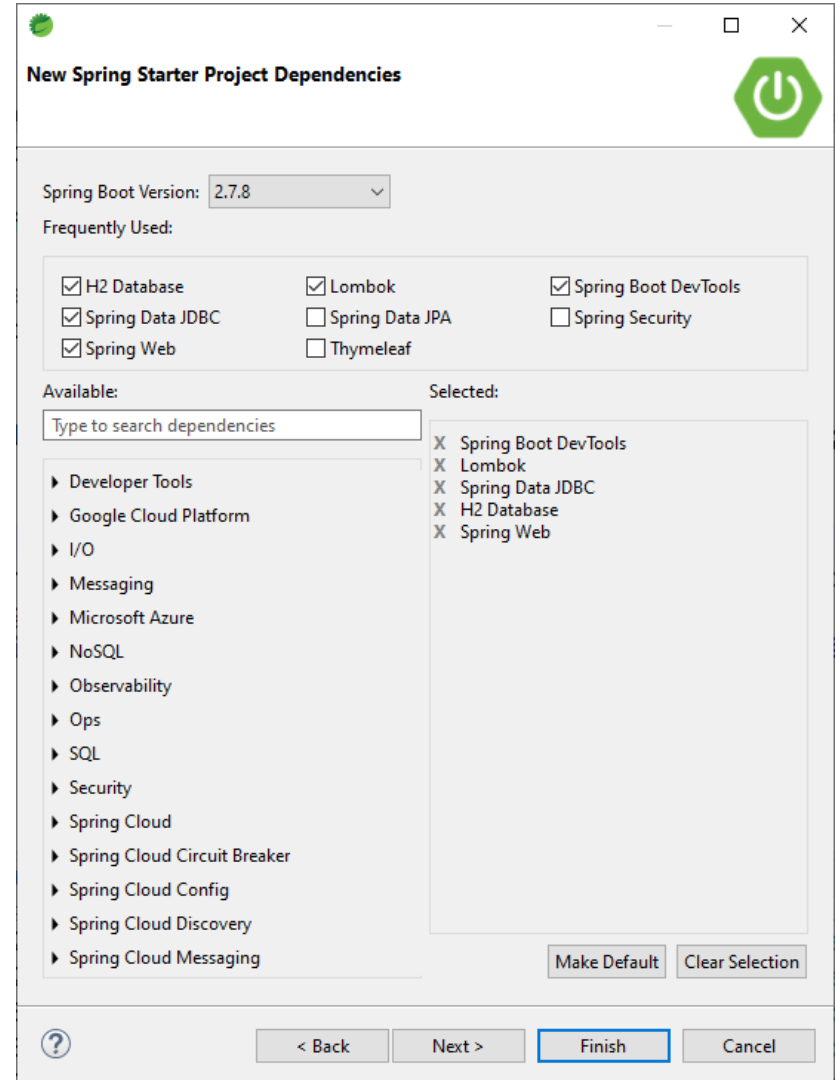
Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

<input checked="" type="checkbox"/> H2 Database	<input checked="" type="checkbox"/> Lombok	<input checked="" type="checkbox"/> Spring Boot DevTools
<input checked="" type="checkbox"/> Spring Data JDBC	<input type="checkbox"/> Spring Data JPA	<input type="checkbox"/> Spring Security
<input checked="" type="checkbox"/> Spring Web	<input type="checkbox"/> Thymeleaf	

Available:

Type to search dependencies

- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Messaging

Selected:

- X Spring Boot DevTools
- X Lombok
- X Spring Data JDBC
- X H2 Database
- X Spring Web

CONFIGURAÇÃO DO PROJETO

■ Dependências

- Webjars são bibliotecas web (como jQuery ou Bootstrap), empacotadas em arquivos *jar*
- São uma opção interessante para aplicações sem acesso à internet, pois não é preciso fazer nenhum *download*

The screenshot shows the WebJars website (https://www.webjars.org) in a browser. The page has a dark header with the WebJars logo and navigation links. The main content area explains that WebJars are client-side web libraries packaged into JAR files. It lists several benefits and mentions that they are deployed on Maven Central and provided by JSDELIVR. Below this, it states 'WebJars come in four flavors:' and lists four options: NPM WebJars, Bower GitHub WebJars, Classic WebJars, and Bower Original WebJars. Each flavor has a list of details like GroupId and ArtifactId. At the bottom, there is a 'Search Results' section with a search bar containing 'jquery'. The results show 'Alpaca - Easy Forms for jQuery' with version 1.5.24. The 'Build Tool' section shows various options like SBT, Play 2, Maven, Ivy, Grape, Gradle, Buildr, and Leiningen. The 'Files' column shows '6428 Files'.

WebJars - Web Libraries in Jars

https://www.webjars.org

WebJars Documentation Sponsored by Heroku

WebJars are client-side web libraries (e.g. jQuery & Bootstrap) packaged into JAR (Java Archive) files.

- Explicitly and easily manage the client-side dependencies in JVM-based web applications
- Use JVM-based build tools (e.g. Maven, Gradle, sbt, ...) to download your client-side dependencies
- Know which client-side dependencies you are using
- Transitive dependencies are automatically resolved and optionally loaded via RequireJS
- Deployed on [Maven Central](#)
- Public CDN, generously provided by: [JSDELIVR](#)

WebJars come in four flavors:

NPM WebJars

- Contents mirror NPM package
- GroupId: `org.webjars.npm`
- ArtifactId: NPM Package or URL-based Name

Bower GitHub WebJars

- Contents mirror Bower package
- GroupId: `org.webjars.bowergithub.[GITHUB_ORG]`
- ArtifactId: GitHub Repo Name

Classic WebJars

- Custom Built and Manually Deployed
- GroupId: `org.webjars`
- ArtifactId: Varies

Bower Original WebJars

- **Deprecated** Use Bower GitHub WebJars instead
- GroupId: `org.webjars.bower`
- ArtifactId: Bower Package or URL-based Name

Search Results

Add a WebJar

☐ NPM ☐ Bower GitHub ☐ Bower Original ☒ Classic

jquery

Name	Versions	Build Tool:	Files
Alpaca - Easy Forms for jQuery	1.5.24	SBT / Play 2 Maven Ivy Grape Gradle Buildr Leiningen	6428 Files

```
<dependency>
<groupId>org.webjars</groupId>
<artifactId>alpaca</artifactId>
<version>1.5.24</version>
</dependency>
```

CONFIGURAÇÃO DO PROJETO

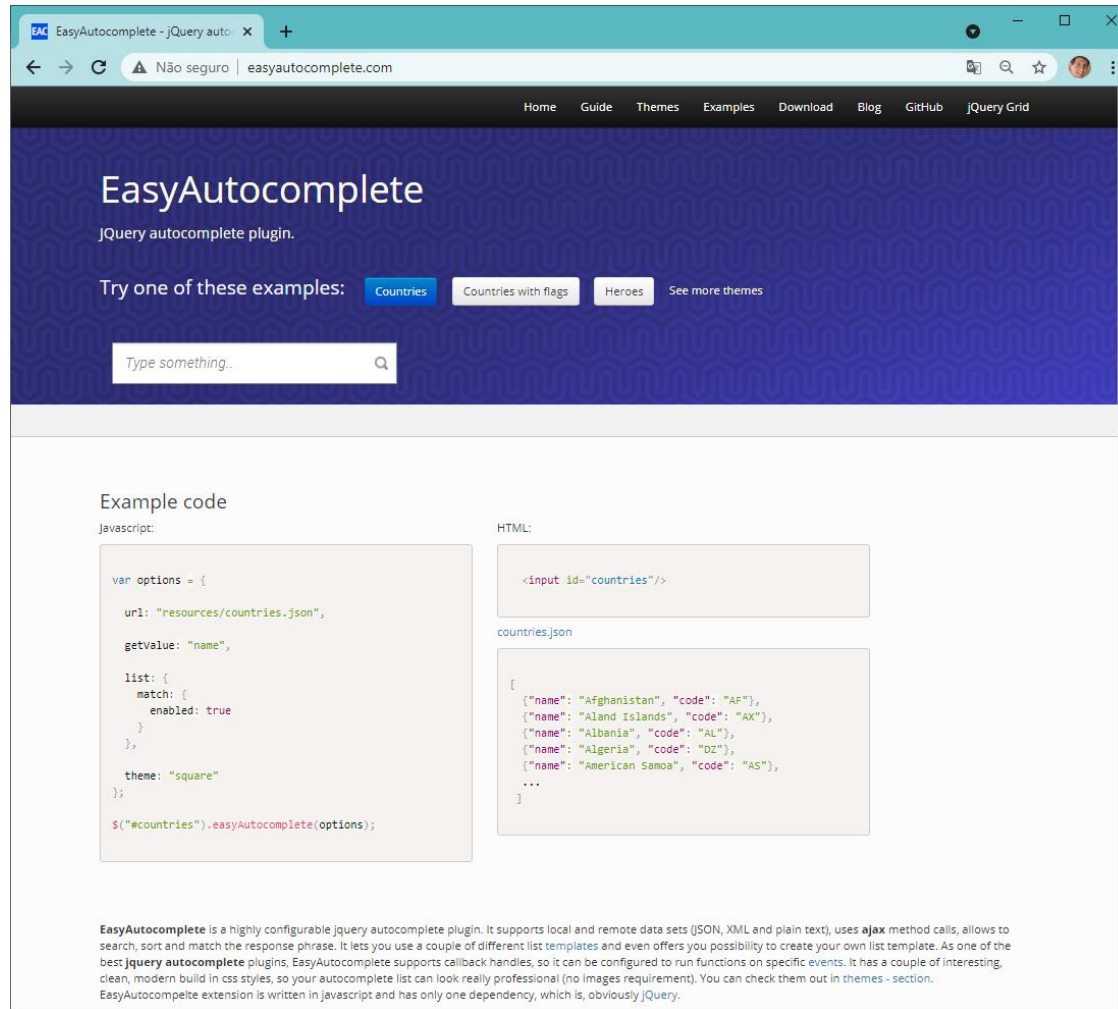
■ Configuração do arquivo *pom.xml*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://m
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.7.8</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>br.edu.unoesc</groupId>
12  <artifactId>exemplo_H2</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>exemplo_H2</name>
15  <description>Backend para sistema 2022</description>
16  <properties>
17    <java.version>17</java.version>
18  </properties>
19  <dependencies>
20    <dependency>
21      <groupId>org.springframework.boot</groupId>
22      <artifactId>spring-boot-devtools</artifactId>
23      <scope>runtime</scope>
24      <optional>true</optional>
25    </dependency>
26
27    <dependency>
28      <groupId>org.springframework.boot</groupId>
29      <artifactId>spring-boot-starter-data-jdbc</artifactId>
30    </dependency>
31    <dependency>
32      <groupId>com.h2database</groupId>
33      <artifactId>h2</artifactId>
34      <scope>runtime</scope>
35    </dependency>
36
37    <dependency>
38      <groupId>org.springframework.boot</groupId>
39      <artifactId>spring-boot-starter-web</artifactId>
40    </dependency>
```

```
41
42  <dependency>
43    <groupId>org.projectlombok</groupId>
44    <artifactId>lombok</artifactId>
45    <optional>true</optional>
46  </dependency>
47  <dependency>
48    <groupId>org.springframework.boot</groupId>
49    <artifactId>spring-boot-starter-test</artifactId>
50    <scope>test</scope>
51  </dependency>
52
53  <dependency>
54    <groupId>org.webjars</groupId>
55    <artifactId>bootstrap</artifactId>
56    <version>5.1.3</version>
57  </dependency>
58  <dependency>
59    <groupId>org.webjars</groupId>
60    <artifactId>jquery</artifactId>
61    <version>3.6.3</version>
62  </dependency>
63  <dependency>
64    <groupId>org.webjars.bower</groupId>
65    <artifactId>EasyAutocomplete</artifactId>
66    <version>1.3.3</version>
67  </dependency>
68 </dependencies>
69
70 <build>
71   <plugins>
72     <plugin>
73       <groupId>org.springframework.boot</groupId>
74       <artifactId>spring-boot-maven-plugin</artifactId>
75       <configuration>
76         <excludes>
77           <exclude>
78             <groupId>org.projectlombok</groupId>
79             <artifactId>lombok</artifactId>
80           </exclude>
81         </excludes>
82       </configuration>
83     </plugin>
84   </plugins>
85 </build>
86
87 </project>
```


PLUG-INS

- EasyAutocomplete é um *plugin autocomplete* para jQuery



The screenshot shows the EasyAutocomplete website. The header includes navigation links: Home, Guide, Themes, Examples, Download, Blog, GitHub, and jQuery Grid. The main content area features the title "EasyAutocomplete" and the subtitle "jQuery autocomplete plugin." Below this, there are buttons for "Countries", "Countries with flags", "Heroes", and a link "See more themes". A search input field with the placeholder "Type something.." is also present. The "Example code" section is divided into two parts: "Javascript:" and "HTML:". The Javascript code defines an options object for the plugin, including a URL to a JSON file, a match function, and a theme. The HTML code shows an input field with the ID "countries". Below the HTML code, the "countries.json" file content is displayed, showing a list of countries with their names and codes. At the bottom, a paragraph describes the plugin's capabilities, such as supporting local and remote data sets, using AJAX, and being highly configurable.

Example code

Javascript:

```
var options = {  
  url: "resources/countries.json",  
  getValue: "name",  
  list: {  
    match: {  
      enabled: true  
    }  
  },  
  theme: "square"  
};  
$("#countries").easyAutocomplete(options);
```

HTML:

```
<input id="countries"/>
```

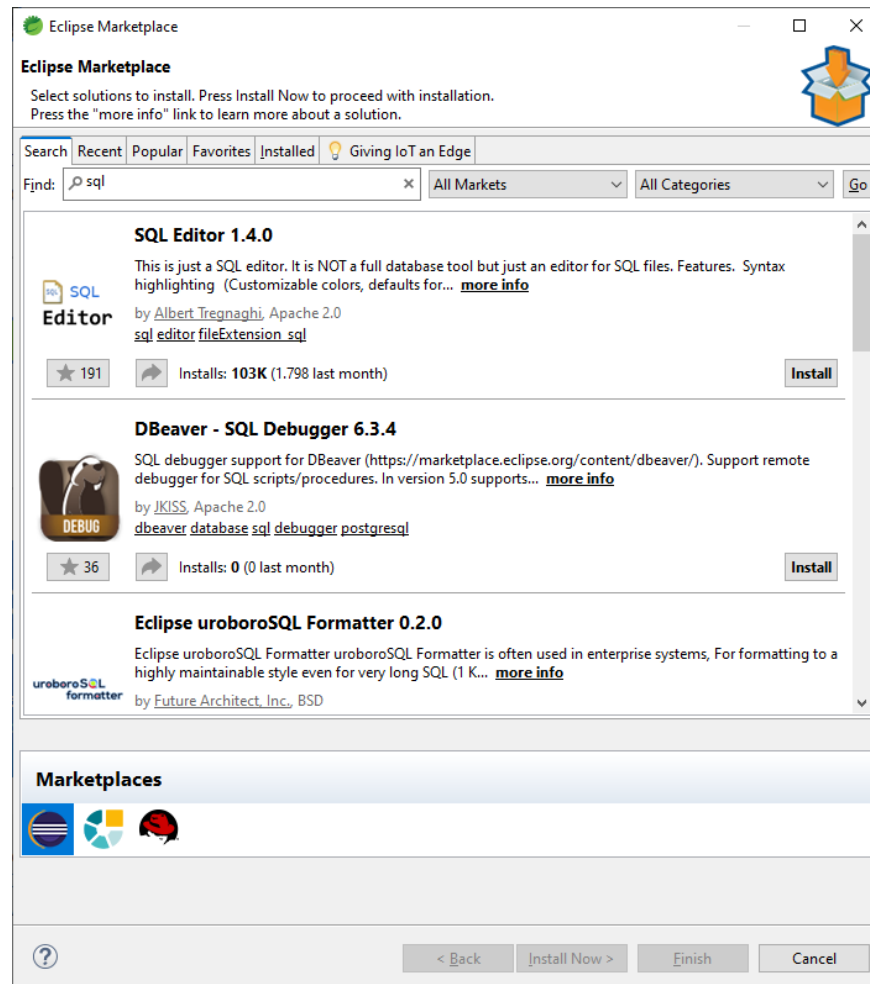
countries.json

```
[  
  {"name": "Afghanistan", "code": "AF"},  
  {"name": "Aland Islands", "code": "AX"},  
  {"name": "Albania", "code": "AL"},  
  {"name": "Algeria", "code": "DZ"},  
  {"name": "American Samoa", "code": "AS"},  
  ...  
]
```

EasyAutocomplete is a highly configurable jquery autocomplete plugin. It supports local and remote data sets (JSON, XML and plain text), uses **ajax** method calls, allows to search, sort and match the response phrase. It lets you use a couple of different list templates and even offers you possibility to create your own list template. As one of the best **jquery autocomplete** plugins. EasyAutocomplete supports callback handles, so it can be configured to run functions on specific events. It has a couple of interesting, clean, modern build in css styles, so your autocomplete list can look really professional (no images requirement). You can check them out in themes - section. EasyAutocomplete extension is written in javascript and has only one dependency, which is, obviously jQuery.

PLUG-INS

- Instale o plug-in SQL Editor (*Help → Eclipse Marketplace*)



SGBD H2

- Configurações do arquivo *application.properties*

```
application.properties ×  
1 logging.level.com.zaxxer.hikari.HikariConfig=DEBUG
```

SGBD H2

■ Propriedades no console

```
dev-ti - exemplo_H2/src/main/resources/application.properties - Spring Tool Suite 4
File Edit Navigate Search Project Git Run Window Help
Problems Javadoc Declaration Console Terminal Git Staging History
exemplo_H2-1 - ExemploH2Application [Spring Boot App] C:\sts4\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (8 de fev. de 2023 12:27:48) [pid: 37720]
ain] j.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
ain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
ain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.71]
ain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
ain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 316 ms
ain] com.zaxxer.hikari.HikariConfig : Driver class org.h2.Driver found in Thread context class loader jdk.internal.loader.ClassLoaders$AppClassLoader@2f333739
ain] com.zaxxer.hikari.HikariConfig : HikariPool-2 - configuration:
ain] com.zaxxer.hikari.HikariConfig : allowPoolSuspension.....false
ain] com.zaxxer.hikari.HikariConfig : autoCommit.....true
ain] com.zaxxer.hikari.HikariConfig : catalog.....none
ain] com.zaxxer.hikari.HikariConfig : connectionInitSql.....none
ain] com.zaxxer.hikari.HikariConfig : connectionTestQuery.....none
ain] com.zaxxer.hikari.HikariConfig : connectionTimeout.....30000
ain] com.zaxxer.hikari.HikariConfig : dataSource.....none
ain] com.zaxxer.hikari.HikariConfig : dataSourceClassName.....none
ain] com.zaxxer.hikari.HikariConfig : dataSourceJNDI.....none
ain] com.zaxxer.hikari.HikariConfig : dataSourceProperties.....{password=<masked>}
ain] com.zaxxer.hikari.HikariConfig : driverClassName....."org.h2.Driver"
ain] com.zaxxer.hikari.HikariConfig : exceptionOverrideClassName.....none
ain] com.zaxxer.hikari.HikariConfig : healthCheckProperties.....{}
ain] com.zaxxer.hikari.HikariConfig : healthCheckRegistry.....none
ain] com.zaxxer.hikari.HikariConfig : idleTimeout.....600000
ain] com.zaxxer.hikari.HikariConfig : initializationFailTimeout.....1
ain] com.zaxxer.hikari.HikariConfig : isolateInternalQueries.....false
ain] com.zaxxer.hikari.HikariConfig : jdbcUrl.....jdbc:h2:mem:d769fe3c-2049-46cd-a96f-d104f4e9cfe4;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
ain] com.zaxxer.hikari.HikariConfig : keepaliveTime.....0
ain] com.zaxxer.hikari.HikariConfig : leakDetectionThreshold.....0
ain] com.zaxxer.hikari.HikariConfig : maxLifetime.....1800000
ain] com.zaxxer.hikari.HikariConfig : maximumPoolSize.....10
ain] com.zaxxer.hikari.HikariConfig : metricRegistry.....none
ain] com.zaxxer.hikari.HikariConfig : metricsTrackerFactory.....none
ain] com.zaxxer.hikari.HikariConfig : minimumIdle.....10
ain] com.zaxxer.hikari.HikariConfig : password.....<masked>
ain] com.zaxxer.hikari.HikariConfig : poolName....."HikariPool-2"
ain] com.zaxxer.hikari.HikariConfig : readOnly.....false
ain] com.zaxxer.hikari.HikariConfig : registerMbeans.....false
ain] com.zaxxer.hikari.HikariConfig : scheduledExecutor.....none
ain] com.zaxxer.hikari.HikariConfig : schema.....none
ain] com.zaxxer.hikari.HikariConfig : threadFactory.....internal
ain] com.zaxxer.hikari.HikariConfig : transactionIsolation.....default
ain] com.zaxxer.hikari.HikariConfig : username....."sa"
ain] com.zaxxer.hikari.HikariConfig : validationTimeout.....5000
ain] com.zaxxer.hikari.HikariDataSource : HikariPool-2 - Starting...
ain] com.zaxxer.hikari.HikariDataSource : HikariPool-2 - Start completed.
ain] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:d769fe3c-2049-46cd-a96f-d104f4e9cfe4'
ain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
```

SGBD H2

■ Configurações do arquivo *application.properties*

■ Utilização de nome fixo em vez de GUID (*Globally Unique Identifier*) aleatório

```
1 #logging.level.com.zaxxer.hikari.HikariConfig=DEBUG
2
3 spring.h2.console.enabled=true
4 spring.datasource.generate-unique-name=false
```

```
com.zaxxer.hikari.HikariDataSource      : HikariPool-3 - Starting...
com.zaxxer.hikari.HikariDataSource      : HikariPool-3 - Start completed.
o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:testdb'
```

■ Configuração do nome do banco de dados

```
1 #logging.level.com.zaxxer.hikari.HikariConfig=DEBUG
2
3 spring.h2.console.enabled=true
4 spring.datasource.generate-unique-name=false
5 spring.datasource.name=meuDB
```

```
com.zaxxer.hikari.HikariDataSource      : meuDB - Starting...
com.zaxxer.hikari.HikariDataSource      : meuDB - Start completed.
o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:meuDB'
```

SGBD H2

- Console no endereço <http://localhost:8080/h2-console>

The screenshot shows a web browser window titled "H2 Terminal" with the address bar displaying "localhost:8080/h2-console/test.d...". The page has a language dropdown set to "Português (Brasil)" and links for "Preferências", "Tools", and "Ajuda".

The main content area is titled "Login" and contains the following fields and buttons:

- Configuração ativa:** A dropdown menu showing "Generic H2 (Embedded)".
- Nome da configuração:** A text input field containing "Generic H2 (Embedded)", with "Gravar" and "Remover" buttons to its right.
- Classe com o driver:** A text input field containing "org.h2.Driver".
- JDBC URL:** A text input field containing "jdbc:h2:mem:meuDB".
- Usuário:** A text input field containing "sa".
- Senha:** An empty password input field.
- Buttons:** "Conectar" and "Testar conexão" buttons are located below the password field.

At the bottom of the form, a green message box states "Teste bem sucedido" (Test successful).

SGBD H2

■ Console no endereço <http://localhost:8080/h2-console>

The screenshot shows the H2 Terminal web console in a browser. The address bar shows `localhost:8080/h2-console/login.do?sessionId=da9163877844dcb744e438a6bba7dda0`. The interface includes a sidebar with a tree view showing the database structure: `jdbc:h2:mem:meuDB`, `INFORMATION_SCHEMA`, `Usuários`, `SA`, and `Administrador`. The main area has tabs for `Executar comando`, `Executar selecionado`, `Auto complete`, `Limpar`, and `Comando SQL:`. Below the tabs is a large text area for entering SQL commands. At the bottom, there are sections for **Comandos importantes** (important commands) and **Scripts de exemplo** (example scripts).

Comandos importantes

Ícone	Descrição
?	Mostrar esta página de ajuda
📜	Mostrar o histórico de comandos
▶	Ctrl+Enter Executar o comando SQL corrente
👤	Shift+Enter Executes the SQL statement defined by the text selection
⌨	Ctrl+Space Auto complete
🔌	Fechar conexão à Base de Dados

Scripts de exemplo

Ação	SQL
Apagar a tabela, caso ela exista	DROP TABLE IF EXISTS TEST;
Criar uma tabela nova com as colunas ID e NAME	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Adicionar uma linha nova	INSERT INTO TEST VALUES(1, 'Hello');
Adicionar outra linha	INSERT INTO TEST VALUES(2, 'World');
Pesquisar uma tabela	SELECT * FROM TEST ORDER BY ID;
Alterar os dados de uma linha	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Apagar uma linha	DELETE FROM TEST WHERE ID=2;
Ajuda	HELP ...

Adicionar drivers de Base de Dados

É possível registrar outros drivers, adicionando o arquivo JAR respectivo na variável de ambiente H2DRIVERS ou CLASSPATH. Exemplo (Windows): Para adicionar o driver que está em `C:/Programs/hsqldb/lib/hsqldb.jar` altere o valor da variável de ambiente H2DRIVERS para `C:/Programs/hsqldb/lib/hsqldb.jar`.

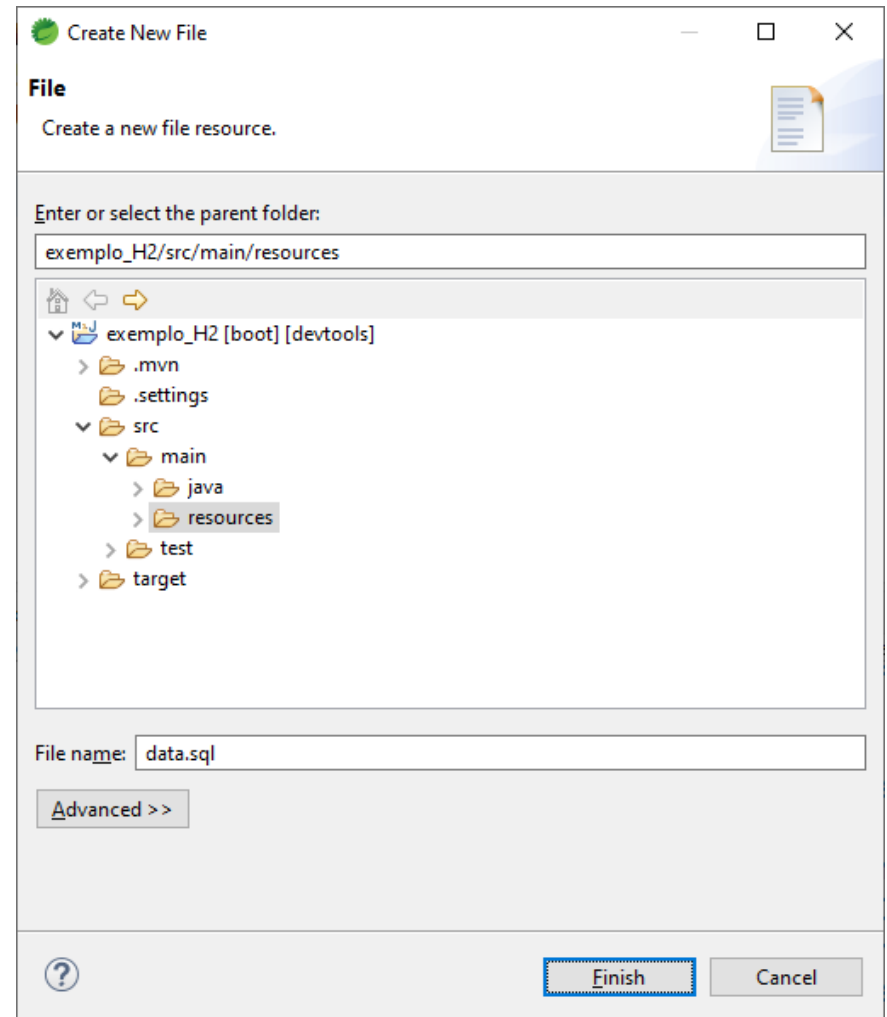
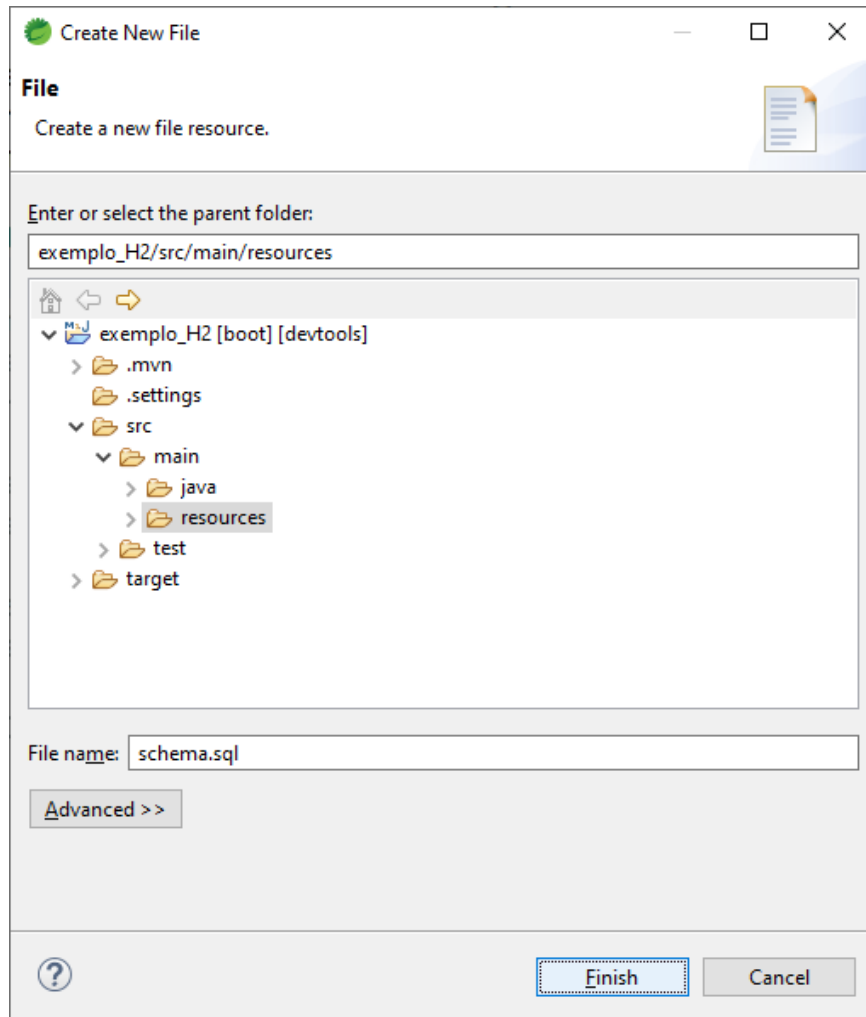
SGBD H2

■ Configurações do arquivo *application.properties*

```
1 #logging.level.com.zaxxer.hikari.HikariConfig=DEBUG
2
3 spring.h2.console.enabled=true
4 spring.datasource.generate-unique-name=false
5 spring.datasource.name=meuDB
6
7 spring.jpa.show-sql=true
8 spring.jpa.properties.hibernate.format_sql=true
9 logging.level.org.springframework.jdbc=DEBUG
```


SGBD H2

■ Criação de scripts de inicialização (*schema.sql* e *data.sql*)



SGBD H2

■ Scripts de inicialização

■ Arquivo *schema.sql*

```
1 CREATE TABLE livro (  
2     id INT AUTO_INCREMENT,  
3     titulo VARCHAR(255) NOT NULL,  
4     paginas INT NOT NULL,  
5     autor VARCHAR(255) NOT NULL  
6 );
```

■ Arquivo *data.sql*

```
1 INSERT INTO livro (titulo, paginas, autor) VALUES ('O senhor dos anéis', 1000, 'Tolkien');  
2 INSERT INTO livro (titulo, paginas, autor) VALUES ('Spring Boot', 328, 'Mark Heckler');
```

SGBD H2

■ Log de inicialização

dev-ti - exemplo_H2/src/main/resources/data.sql - Spring Tool Suite 4

File Edit Source Navigate Search Project Git Run Window Help

Problems @ Javadoc Declaration Console x Terminal Git Staging History

exemplo_H2 - ExemploH2Application [Spring Boot App] [pid: 32012]

```
: No active profile set, falling back to 1 default profile: "default"  
: Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable  
: For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'  
: Bootstrapping Spring Data JPA repositories in DEFAULT mode.  
: Finished Spring Data repository scanning in 9 ms. Found 0 JPA repository interfaces.  
: Tomcat initialized with port(s): 8080 (http)  
: Starting service [Tomcat]  
: Starting Servlet engine: [Apache Tomcat/9.0.71]  
: Initializing Spring embedded WebApplicationContext  
: Root WebApplicationContext: initialization completed in 1396 ms  
: meuDB - Starting...  
: meuDB - Start completed.  
: H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:meuDB'  
: Fetching JDBC Connection from DataSource  
: Fetching JDBC Connection from DataSource  
: Executing SQL script from URL [file:C:/workspaces/dev-ti/exemplo_H2/target/classes/schema.sql]  
: 0 returned as update count for SQL: CREATE TABLE livro ( id INT AUTO_INCREMENT, titulo VARCHAR(255) NOT NULL, paginas INT NOT NULL, autor VARCHAR(255) N  
: Executed SQL script from URL [file:C:/workspaces/dev-ti/exemplo_H2/target/classes/schema.sql] in 17 ms.  
: Fetching JDBC Connection from DataSource  
: Fetching JDBC Connection from DataSource  
: Executing SQL script from URL [file:C:/workspaces/dev-ti/exemplo_H2/target/classes/data.sql]  
: 1 returned as update count for SQL: INSERT INTO livro (titulo, paginas, autor) VALUES ('O senhor dos anéis', 1000, 'Tolkien')  
: 1 returned as update count for SQL: INSERT INTO livro (titulo, paginas, autor) VALUES ('Spring Boot', 328, 'Mark Heckler')  
: Executed SQL script from URL [file:C:/workspaces/dev-ti/exemplo_H2/target/classes/data.sql] in 7 ms.  
: Fetching JDBC Connection from DataSource  
: HHH000204: Processing PersistenceUnitInfo [name: default]  
: HHH000412: Hibernate ORM core version 5.6.14.Final  
: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}  
: HHH000400: Using dialect: org.hibernate.dialect.H2Dialect  
: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]  
: Initialized JPA EntityManagerFactory for persistence unit 'default'  
: spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-  
: LiveReload server is running on port 35729  
: Tomcat started on port(s): 8080 (http) with context path ''  
: Started ExemploH2Application in 3.699 seconds (JVM running for 4.591)
```

Search for TypeDeclara...in: none (78%)

SGBD H2

Console

The screenshot shows the H2 Console web interface in a browser. The address bar shows the URL `localhost:8080/h2-console/login.do?jsessionid=c6b0b9b20f4e9a7bcb440c28b14b3f2e`. The interface includes a toolbar with options like `Auto commit` (checked), `Max rows: 1000`, `Auto complete` (Off), and `Auto select` (On). On the left, a tree view shows the database structure: `jdbc:h2:mem:meuDB` containing a table `LIVRO` with columns `ID`, `TITULO`, `PAGINAS`, and `AUTOR`, and a schema `INFORMATION_SCHEMA`. The main area shows the SQL statement `SELECT * FROM livro;` and its results in a table format.

SQL statement:

```
SELECT * FROM livro;
```

ID	TITULO	PAGINAS	AUTOR
1	O senhor dos anéis	1000	Tolkien
2	Spring Boot	328	Mark Heckler

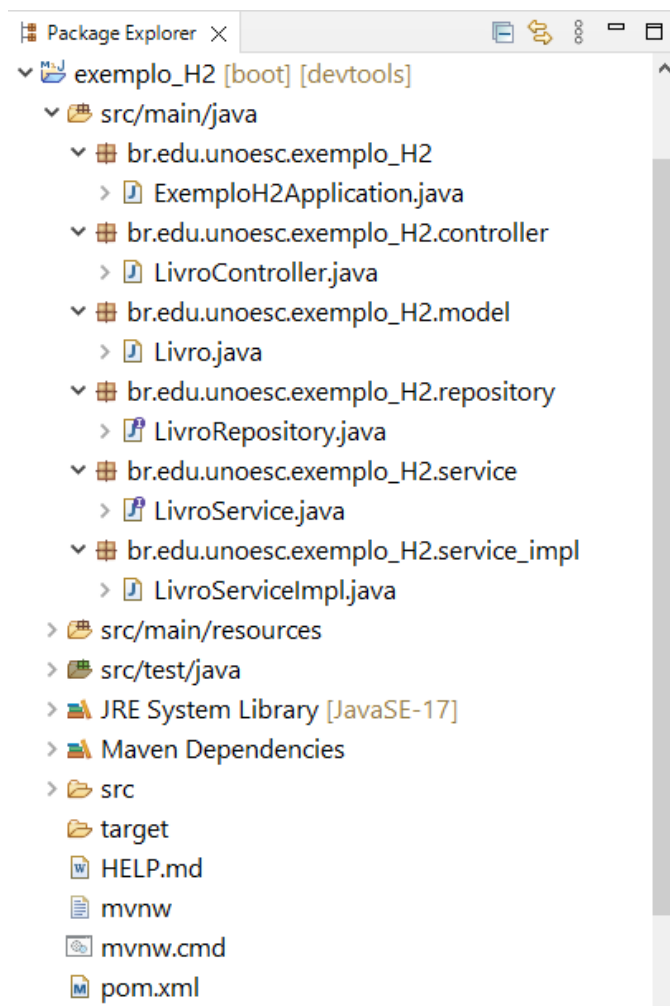
(2 rows, 3 ms)

ESTRUTURA DE PACOTES

■ Criação dos pacotes

- model
- controller
- repository
- service
- service_impl

■ Cópia dos recursos



EXEMPLO H2

■ Registro (record) Livro

```
1 package br.edu.unoesc.exemplo_H2.model;  
2  
3 import org.springframework.data.annotation.Id;  
4  
5 public record Livro(  
6     @Id  
7     Integer id,  
8     String titulo,  
9     Integer paginas,  
10    String autor  
11 ) {  
12  
13 }
```

RECORD

- Registro (`record`) é um recurso que apareceu pela primeira vez na versão 14 como experimental e foi liberado de forma definitiva no Java 16
 - Um registro é imutável, ou seja, após criado, um `record` não pode mais ser alterado
 - Record oferece uma sintaxe compacta para declarar classes que são portadores transparentes para dados imutáveis, reduzindo significativamente o detalhamento dessas classes e irá melhorando a capacidade de leitura e manutenção do código
- Características
 - Um `Record` não possui uma cláusula `extends`
 - Um `Record` não pode ser abstrato
 - Os atributos derivados da classe `Record` são todos finais
 - Não se pode declarar campos de instância e nem métodos nativos
 - Uma instância de `Record` é criada com a expressão `new`
 - Pode ser declarada como um tipo genérico
 - Pode declarar métodos, atributos e inicializadores estáticos
 - Pode declarar métodos de instância
 - Pode implementar *interfaces*
 - Pode utilizar *annotations*
 - Pode ser serializado e desserializado

REPOSITÓRIO

■ *Interface* LivroRepository

```
1 package br.edu.unoesc.exemplo_H2.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import br.edu.unoesc.exemplo_H2.model.Livro;
6
7 public interface LivroRepository extends CrudRepository<Livro, Integer> {
8
9 }
```


COMMAND LINE RUNNER

■ Programa principal

■ Utilização do método

`commandLineRunner()`
para testes

```
1 package br.edu.unoesc.exemplo_H2;
2
3 import java.util.Optional;
4
5 import org.springframework.boot.CommandLineRunner;
6 import org.springframework.boot.SpringApplication;
7 import org.springframework.boot.autoconfigure.SpringBootApplication;
8 import org.springframework.context.annotation.Bean;
9
10 import br.edu.unoesc.exemplo_H2.model.Livro;
11 import br.edu.unoesc.exemplo_H2.repository.LivroRepository;
12
13 @SpringBootApplication
14 public class ExemploH2Application {
15
16     public static void main(String[] args) {
17         SpringApplication.run(ExemploH2Application.class, args);
18     }
19
20     @Bean
21     CommandLineRunner commandLineRunner(LivroRepository repositorio) {
22         return args -> {
23             Livro l = new Livro(null, "O senhor dos pastéis", 666, "Tolkien");
24             l = repositorio.save(l);
25
26             Livro n = new Livro(l.id(), "The Lord of the Rings", l.paginas(), l.autor());
27             repositorio.save(n);
28             // repositorio.delete(l);
29
30             repositorio.save(new Livro(null, "O hobbit", 42, "J.R.R.Tolkien"));
31             Optional<Livro> p = repositorio.findById(4);
32             // repositorio.delete(p.get());
33
34             System.out.println(repositorio.findAll());
35             for (var livro : repositorio.findAll()) {
36                 System.out.println(livro);
37             }
38         };
39     }
40 }
```

REFATORAÇÃO PARA JPA

■ Refatorações

1. Remova o *starter* do JDBC e acrescente o do JPA
2. Acrescente a seguinte configuração no arquivo *application.properties*
`spring.jpa.hibernate.ddl-auto=update`
3. Modifique o registro (*record*) `Livro` para uma classe (entidade) JPA

REFATORAÇÃO PARA JPA

■ Classe Livro

```
1 package br.edu.unoesc.exemplo_H2.model;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 import lombok.AllArgsConstructor;
11 import lombok.Data;
12 import lombok.NoArgsConstructor;
13
14 @Data
15 @Entity
16 @AllArgsConstructor
17 @NoArgsConstructor
18 public class Livro implements Serializable {
19     private static final long serialVersionUID = 1L;
20
21     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
22     private Integer id;
23
24     private String titulo;
25     private Integer paginas;
26     private String autor;
27 }
```

REFATORAÇÃO PA

■ Classe principal

- Possibilidade de alterar os valores das entidades

- Teste com a classe Optional

```
1 package br.edu.unoesc.exemplo_H2;
2
3 import java.util.Optional;
4
5 import org.springframework.boot.CommandLineRunner;
6 import org.springframework.boot.SpringApplication;
7 import org.springframework.boot.autoconfigure.SpringBootApplication;
8 import org.springframework.context.annotation.Bean;
9
10 import br.edu.unoesc.exemplo_H2.model.Livro;
11 import br.edu.unoesc.exemplo_H2.repository.LivroRepository;
12
13 @SpringBootApplication
14 public class ExemploH2Application {
15
16     public static void main(String[] args) {
17         SpringApplication.run(ExemploH2Application.class, args);
18     }
19
20     @Bean
21     CommandLineRunner commandLineRunner(LivroRepository repositorio) {
22         return args -> {
23             Livro l = new Livro(null, "O senhor dos pastéis", 666, "Tolkien");
24             repositorio.save(l);
25
26             repositorio.save(new Livro(null, "O hobbit", 42, "J.R.R.Tolkien"));
27
28             Optional<Livro> p = repositorio.findById(6);
29             if (p.isPresent()) {
30                 repositorio.delete(p.get());
31             } else {
32                 System.out.println("Registro não encontrado!");
33             }
34
35             l = repositorio.findById(2).get();
36             l.setTitulo("O livro do Spring Boot");
37             l.setPaginas(100);
38             repositorio.save(l);
39         };
40     }
41 }
```

CONSULTAS PERSONALIZADAS

■ Novos métodos no repositório

```
1 package br.edu.unoesc.exemplo_H2.repository;
2
3 import java.util.List;
4
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.data.repository.CrudRepository;
7 import org.springframework.data.repository.query.Param;
8
9 import br.edu.unoesc.exemplo_H2.model.Livro;
10
11 public interface LivroRepository extends CrudRepository<Livro, Integer> {
12     public List<Livro> findByAutorContainingIgnoreCase(String autor);
13
14     @Query("Select l from Livro l where l.paginas >= :paginas")
15     public List<Livro> porQtdPaginas(@Param("paginas") Integer paginas);
16
17     @Query("Select l from Livro l where upper(l.titulo) like upper(concat('%', :filtro, '%')) order by titulo")
18     public List<Livro> findByFiltro(@Param("filtro") String filtro);
19 }
```

CONTROLADOR V.1

■ *Controller* (*LivroController.java*) utilizando diretamente o repositório

■ Injeção de dependência através da anotação `@Autowired`

```
1 package br.edu.unoesc.exemplo_H2.controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RequestParam;
9 import org.springframework.web.bind.annotation.RestController;
10
11 import br.edu.unoesc.exemplo_H2.model.Livro;
12 import br.edu.unoesc.exemplo_H2.repository.LivroRepository;
13
14 @RestController
15 @RequestMapping("/api/livros")
16 public class LivroController {
17     @Autowired
18     private LivroRepository repositorio;
19
20     @GetMapping("/find")
21     List<Livro> findByFiltro(@RequestParam("filtro") String filtro) {
22         return repositorio.findByFiltro(filtro);
23     }
24
25     @GetMapping
26     public Iterable<Livro> findAll() {
27         return repositorio.findAll();
28     }
29 }
```

CONTROLADOR V.1

■ Consultas

The screenshot displays a web browser window with two tabs. The active tab is titled 'localhost:8080/api/livros/find?filtro=senhor' and shows a JSON response from the H2 Console. The response is a list of books filtered by the author 'senhor' (Tolkien).

```
[{"id": 1, "titulo": "O senhor dos anéis", "paginas": 1000, "autor": "Tolkien"}, {"id": 3, "titulo": "O senhor dos pastéis", "paginas": 666, "autor": "Tolkien"}]
```

The browser's address bar shows the URL 'localhost:8080/api/livros/find?filtro=senhor'. The page includes a 'Raw' button and a 'Parsed' button, with the 'Parsed' button currently selected. The JSON response is displayed in a formatted, color-coded manner.

FRONT-END

■ Página *index.html* (início e fim)

```
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Spring Boot : Livros</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8   <link rel="stylesheet" type="text/css" href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" />
9   <link rel="stylesheet" type="text/css" href="webjars/EasyAutocomplete/1.3.3/dist/easy-autocomplete.themes.css" />
10 </head>
11 <body>
12   <div class="container">
13     <script type="text/javascript" src="webjars/jquery/3.6.3/jquery.min.js"></script>
14     <script type="text/javascript" src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>
15     <script type="text/javascript" src="webjars/EasyAutocomplete/1.3.3/dist/jquery.easy-autocomplete.js"></script>
16     .
17     .
18     .
19
20     <h1 class="text-center mt-2">
21       
22       Spring Boot : Livros
23     </h1>
24
25     <form class="mt-1">
26       Buscar:
27       <input type="text" id="filtro" autofocus="autofocus" class="form-control" />
28     </form>
29   </div>
30 </body>
31 </html>
```

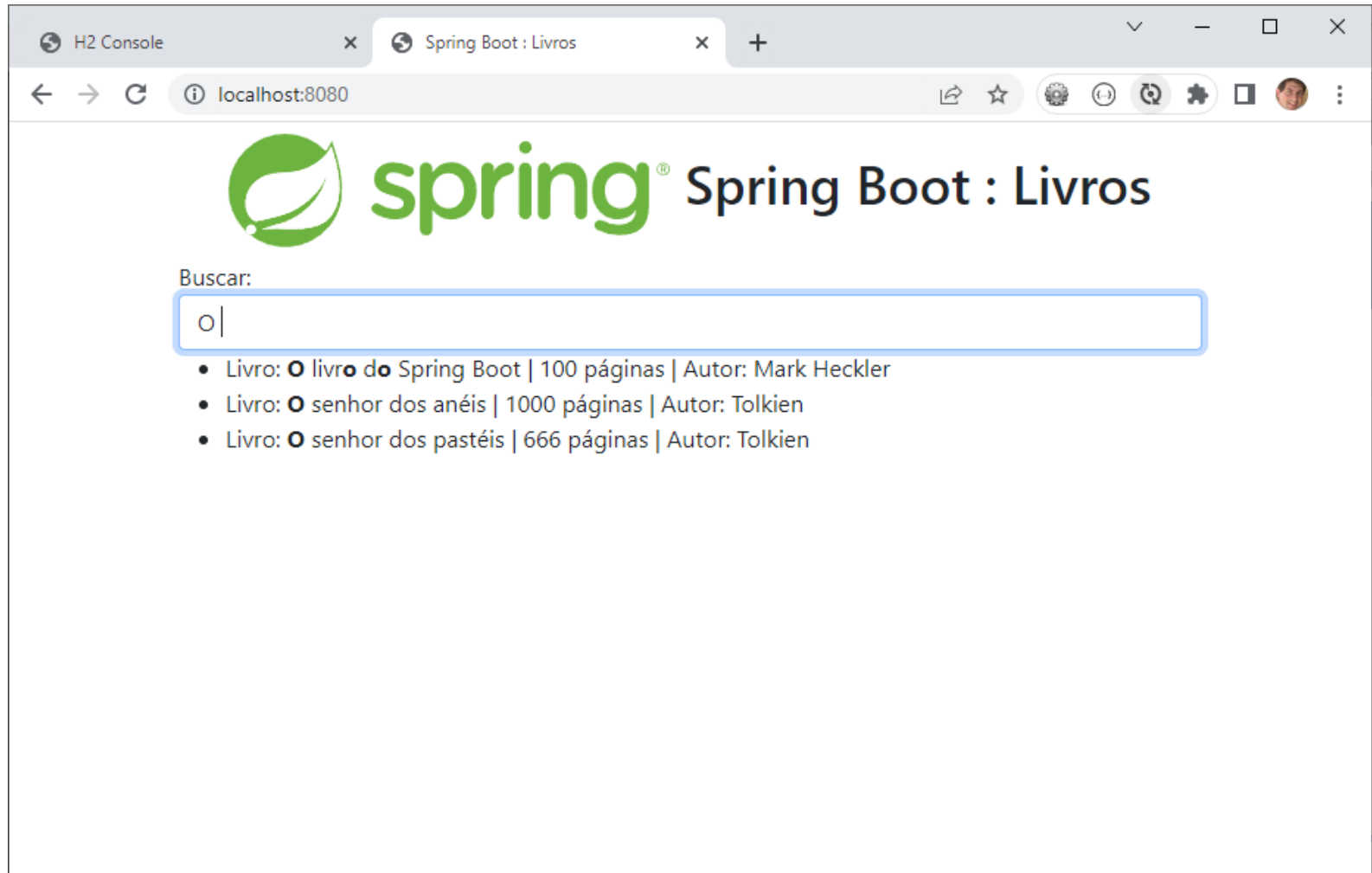

FRONT-END

■ Página *index.html*

```
17<script type="text/javascript">
18    $(document).ready(function () {
19        let opcoes = {
20            url: function (filtro) {
21                return '/api/livros/find/';
22            },
23            getValue: function (elemento) {
24                return elemento.titulo;
25            },
26            ajaxSettings: {
27                dataType: 'json',
28                method: 'GET',
29                data: { }
30            },
31            template: {
32                type: "custom",
33                method: function (value, item) {
34                    return "Livro: " + value + " | " + item.paginas + " páginas | Autor: " + item.autor;
35                }
36            },
37            preparePostData: function (data) {
38                data.filtro = $('#filtro').val();
39                return data;
40            },
41            theme: "dark",
42            list: {
43                showAnimation: {
44                    type: "slide"
45                },
46                hideAnimation: {
47                    type: "slide"
48                },
49                match: {
50                    enabled: true
51                },
52                maxNumberOfElements: 15
53            },
54            //requestDelay: 100
55        };
56
57        $('#filtro').easyAutocomplete(opcoes);
58    });
59</script>
```

FRONT-END

■ Exemplo



FRONT-END

■ Página *index_ajax_tabela.html*

```
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4     <meta charset="UTF-8">
5     <title>Spring Boot : Livros</title>
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8     <link rel="stylesheet" type="text/css" href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" />
9     <link rel="stylesheet" type="text/css" href="webjars/EasyAutocomplete/1.3.3/dist/easy-autocomplete.themes.css" />
10 </head>
11 <body>
12     <div class="container">
13         <script type="text/javascript" src="webjars/jquery/3.6.3/jquery.min.js"></script>
14         <script type="text/javascript" src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>
15         <script type="text/javascript" src="webjars/EasyAutocomplete/1.3.3/dist/jquery.easy-autocomplete.js"></script>
```

FRONT-END

■ Página *index_ajax_tabela.html*

```
17<script type="text/javascript">
18    $(document).ready(function () {
19        let filtroForm = document.getElementById('filtro');
20
21        filtroForm.addEventListener('input', (filtro) => {
22            $.ajax({
23                type : 'GET',
24                url : '/api/livros/find?filtro=' + filtroForm.value,
25                data : '$format=json',
26                dataType : 'json',
27                success : function(dados) {
28                    let total = 0;
29                    $('#livros tbody').empty();
30                    $.each(dados, function(d, resultado) {
31                        $('#livros tbody').append(
32                            '<tr>' +
33                                '<td class="text-nowrap">' + resultado.titulo + '</td>' +
34                                "&td class=\"text-nowrap\\>" + resultado.paginas + '</td>' +
35                                "&td class=\"text-nowrap\\>" + resultado.autor + '</td>' +
36                                '</tr>');
37                        total++;
38                    });
39
40                    if (total === 0) {
41                        $('#resultados').text('Nenhum registro encontrado!');
42                    } else if (total === 1) {
43                        $('#resultados').text('1 registro encontrado!');
44                    } else {
45                        $('#resultados').text(total + ' registros encontrados!');
46                    }
47                }
48            });
49        });
50    });
51</script>
```

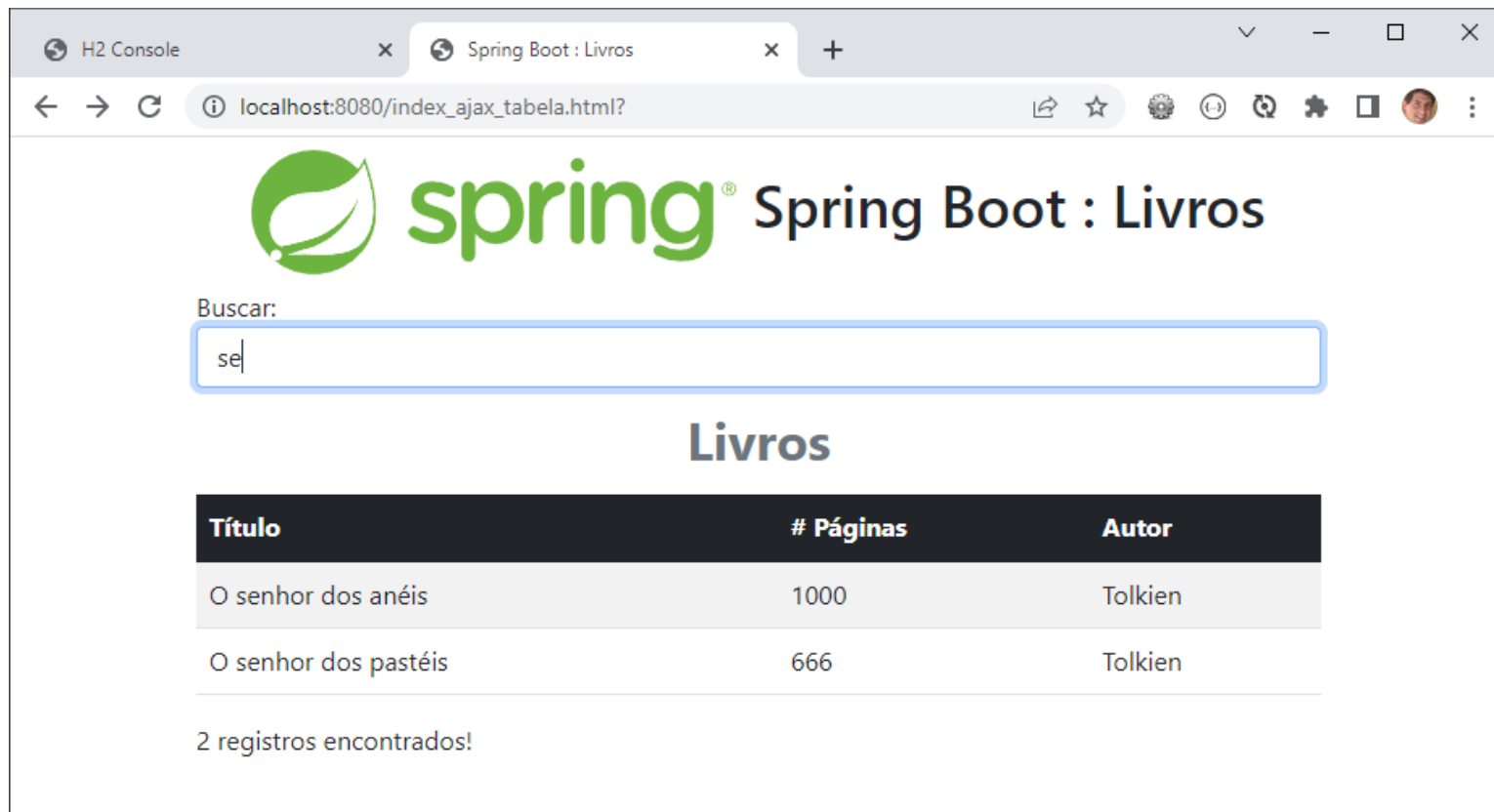
FRONT-END

■ Página `index_ajax_tabela.html`

```
53<h1 class="text-center mt-2">
54    
55    Spring Boot : Livros
56</h1>
57
58<form class="mt-1">
59    Buscar:
60    <input type="text" id="filtro" autofocus="autofocus" class="form-control" />
61</form>
62
63<table id="livros" class="table table-striped table-condensed table-hover table-responsive">
64    <caption style="font-size: 2em; font-weight: bold; text-align: center; caption-side: top">
65        Livros
66    </caption>
67
68    <thead class="table-dark">
69        <tr>
70            <th>Título</th>
71            <th># Páginas</th>
72            <th>Autor</th>
73        </tr>
74    </thead>
75
76    <tbody></tbody>
77</table>
78
79    <div id="resultados"></div>
80</div>
81</body>
82</html>
```

FRONT-END

■ Exemplo



Buscar:

se

Livros

Título	# Páginas	Autor
O senhor dos anéis	1000	Tolkien
O senhor dos pastéis	666	Tolkien

2 registros encontrados!

CAMADA DE SERVIÇO

- Uma boa prática de desenvolvimento é separar o sistema em camadas e dividir as responsabilidades nos componentes corretos
 - A camada *controller* tem a finalidade de apenas orientar a requisição para a lógica de negócio e devolver a resposta
 - A tarefa do componente controlador é receber as requisições, chamar os componentes de negócios e devolver as respostas, não devendo implementar regras de negócios
 - As regras de negócio podem ser colocadas dentro das entidades, dessa forma ela não fica sendo somente uma estrutura de dados mas passa a ter responsabilidades
 - Normalmente nela ficam regras de negócio mais básicas, que não envolvem interações com outras entidades
- Quando há interação com outras entidades ou com repositórios costuma-se criar uma camada ou componente de serviço
- Métodos de consulta podem ser implementados na classe controladora ou de serviço

CAMADA DE SERVIÇO

- Acrescentando uma camada de serviço: *Interface* LivroService

```
1 package br.edu.unoesc.exemplo_H2.service;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import br.edu.unoesc.exemplo_H2.model.Livro;
7
8 public interface LivroService {
9     Livro salvar(Livro livro);
10    void excluir(Integer id);
11
12    Livro buscar(Integer id);
13    Livro buscarPorCodigo(Integer id);
14    Optional<Livro> porCodigo(Integer id);
15
16    List<Livro> buscarPorTitulo(String titulo);
17    List<Livro> buscarPorAutor(String autor);
18    List<Livro> buscarPorQtdPaginas(Integer qtdPaginas);
19
20    Iterable<Livro> listar();
21 }
```


CAMADA DE SERVIÇO

■ Implementação concreta da interface: `LivroServiceImpl`

```
1 package br.edu.unoesc.exemplo_H2.service_impl;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.hibernate.ObjectNotFoundException;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9
10 import br.edu.unoesc.exemplo_H2.model.Livro;
11 import br.edu.unoesc.exemplo_H2.repository.LivroRepository;
12 import br.edu.unoesc.exemplo_H2.service.LivroService;
13
14 @Service
15 public class LivroServiceImpl implements LivroService {
16     @Autowired
17     private LivroRepository repositorio;
18
19     @Override
20     public Livro salvar(Livro livro) {
21         return repositorio.save(livro);
22     }
23
24     @Override
25     public void excluir(Integer id) {
26         if (repositorio.existsById(id)) {
27             repositorio.deleteById(id);
28         } else {
29             throw new ObjectNotFoundException("Objeto não encontrado! Id: " + id +
30                 ", Tipo: " + Livro.class.getName(),
31                 null);
32         }
33     }
34
35     @Override
36     public Iterable<Livro> listar() {
37         return repositorio.findAll();
38     }
39 }
```

CAMADA DE SERVIÇO

- Implementação concreta da interface: `LivroServiceImpl`

```
40 @Override
41 public Livro buscar(Integer id) {
42     Optional<Livro> obj = repositorio.findById(id);
43     return obj.orElseThrow(() ->
44         new ObjectNotFoundException("Objeto não encontrado! Id: " + id +
45                                     ", Tipo: " + Livro.class.getName(),
46                                     null
47     ));
48 }
49
50
51 @Override
52 public Optional<Livro> porCodigo(Integer id) {
53     return repositorio.findById(id);
54 }
55
56 @Override
57 // Retorna um novo objeto caso id não exista
58 public Livro buscarPorCodigo(Integer id) {
59     return repositorio.findById(id).orElse(new Livro());
60 }
61
62 @Override
63 public List<Livro> buscarPorTitulo(String titulo) {
64     return repositorio.findByFiltro(titulo);
65 }
66
67 @Override
68 public List<Livro> buscarPorAutor(String autor) {
69     return repositorio.findByAutorContainingIgnoreCase(autor);
70 }
71
72 @Override
73 public List<Livro> buscarPorQtdPaginas(Integer qtdPaginas) {
74     return repositorio.porQtdPaginas(qtdPaginas);
75 }
76 }
```

CAMADA DE SERVIÇO

- Refatorando a classe principal a fim de utilizar a classe service e testando os novos métodos

```
1 package br.edu.unoesc.exemplo_H2;
2
3 import java.util.Optional;
4
5 import org.hibernate.ObjectNotFoundException;
6 import org.springframework.boot.CommandLineRunner;
7 import org.springframework.boot.SpringApplication;
8 import org.springframework.boot.autoconfigure.SpringBootApplication;
9 import org.springframework.context.annotation.Bean;
10
11 import br.edu.unoesc.exemplo_H2.model.Livro;
12 import br.edu.unoesc.exemplo_H2.service.LivroService;
13
14 @SpringBootApplication
15 public class ExemploH2Application {
16     public static void main(String[] args) {
17         SpringApplication.run(ExemploH2Application.class, args);
18     }
19
20     @Bean
21     CommandLineRunner commandLineRunner(LivroService servico) {
22         return args -> {
23             Livro l = new Livro(null, "O senhor dos pastéis", 666, "Tolkien");
24             servico.salvar(l);
25
26             servico.salvar(new Livro(null, "O hobbit", 42, "J.R.R.Tolkien"));
27
28             try {
29                 servico.excluir(6);
30             } catch (ObjectNotFoundException e) {
31                 System.out.println(e);
32             }
33
34             // Teste com a classe Optional
35             Optional<Livro> o = servico.porCodigo(3);
36             if (!o.isPresent()) {
37                 System.out.println("Livro não existe!");
38             } else {
39                 System.out.println(o);
40             }
41
42             // Cria nova instância pois objeto com id 10 não existe
43             l = servico.buscarPorCodigo(10);
44             l.setTitulo("O livro do Spring Boot");
45             l.setPaginas(100);
46             l.setAutor("Zé das Couves");
47             servico.salvar(l);
48         };
49     }
50 }
```

CONTROLADOR

■ Refatorando a classe *controller*

```
1 package br.edu.unoesc.exemplo_H2.controller;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.web.bind.annotation.DeleteMapping;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.PostMapping;
11 import org.springframework.web.bind.annotation.PutMapping;
12 import org.springframework.web.bind.annotation.RequestBody;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestParam;
15 import org.springframework.web.bind.annotation.RestController;
16
17 import br.edu.unoesc.exemplo_H2.model.Livro;
18 import br.edu.unoesc.exemplo_H2.service.LivroService;
19
20 @RestController
21 @RequestMapping("/api/livros")
22 public class LivroController {
23     @Autowired
24     private LivroService servico;
25
26     @PostMapping()
27     public Livro incluir(@RequestBody Livro livro) {
28         return servico.salvar(livro);
29     }
30
31     @PutMapping("/{id}")
32     public Livro atualizar(@PathVariable("id") Integer id, @RequestBody Livro livro) {
33         livro.setId(id);
34
35         return servico.salvar(livro);
36     }
37
38     @DeleteMapping("/{id}")
39     public void excluir(@PathVariable("id") Integer id) {
40         servico.excluir(id);
41     }
42 }
```

CONTROLADOR V.2

■ Refatorando a classe *controller*

```
43= @GetMapping
44 public Iterable<Livro> listar() {
45     return servico.listar();
46 }
47
48= @GetMapping("/{id}")
49 public Livro buscarPorCodigo(@PathVariable("id") Integer id) {
50     return servico.porCodigo(id).get();
51 }
52
53= @GetMapping("/find")
54 public List<Livro> findByFiltro(@RequestParam("filtro") String filtro) {
55     return servico.buscarPorTitulo(filtro);
56 }
57
58= @GetMapping("/autor")
59 public List<Livro> buscarPorAutor(@RequestParam("filtro") String filtro) {
60     return servico.buscarPorAutor(filtro);
61 }
62
63= @GetMapping("/qtdpaginas")
64 public List<Livro> buscarPorQtdPaginas(@RequestParam("filtro") Integer filtro) {
65     return servico.buscarPorQtdPaginas(filtro);
66 }
67
68= @GetMapping(value={"/livroporqtdpgs", "/livroporqtdpgs/{paginas}"})
69 public List<Livro> listarPorQuantidade(@PathVariable Optional<Integer> paginas) {
70     return servico.buscarPorQtdPaginas(paginas.orElse(0));
71 }
72 }
```

CONTROLADOR V.3

- Refatorando: Acréscimo de código para tratamento de erros e códigos de retorno

- A classe `ResponseEntity` representa toda a resposta HTTP, incluindo o código de *status*, cabeçalho (*header*) e corpo (*body*)

```
1 package br.edu.unoesc.exemplo_H2.controller;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.hibernate.ObjectNotFoundException;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpStatus;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.DeleteMapping;
11 import org.springframework.web.bind.annotation.GetMapping;
12 import org.springframework.web.bind.annotation.PathVariable;
13 import org.springframework.web.bind.annotation.PostMapping;
14 import org.springframework.web.bind.annotation.PutMapping;
15 import org.springframework.web.bind.annotation.RequestBody;
16 import org.springframework.web.bind.annotation.RequestMapping;
17 import org.springframework.web.bind.annotation.RequestParam;
18 import org.springframework.web.bind.annotation.ResponseStatus;
19 import org.springframework.web.bind.annotation.RestController;
20
21 import br.edu.unoesc.exemplo_H2.model.Livro;
22 import br.edu.unoesc.exemplo_H2.service.LivroService;
```

```
24 @RestController
25 @RequestMapping("/api/livros")
26 public class LivroController {
27     @Autowired
28     private LivroService servico;
29
30     @GetMapping
31     public Iterable<Livro> listar() {
32         return servico.listar();
33     }
34
35     @GetMapping("/{id}")
36     public ResponseEntity<Livro> buscarPorCodigo(@PathVariable("id") Integer id) {
37         Optional<Livro> livro = servico.porCodigo(id);
38
39         if (livro.isPresent()) {
40             return ResponseEntity.ok(livro.get());
41         }
42
43         return ResponseEntity.notFound().build();
44     }
45
46     @GetMapping("/find")
47     public List<Livro> findByFiltro(@RequestParam("filtro") String filtro) {
48         return servico.buscarPorTitulo(filtro);
49     }
50
51     @GetMapping("/autor")
52     public List<Livro> buscarPorAutor(@RequestParam("filtro") String filtro) {
53         return servico.buscarPorAutor(filtro);
54     }
55
56     @GetMapping("/qtdpaginas")
57     public List<Livro> buscarPorQtdPaginas(@RequestParam("filtro") Integer filtro) {
58         return servico.buscarPorQtdPaginas(filtro);
59     }
}
```

CONTROLADOR V.3

- Refatorando: Acréscimo de código para tratamento de erros e códigos de retorno

- A classe `ResponseEntity` representa toda a resposta HTTP, incluindo o código de *status*, cabeçalho (*header*) e corpo (*body*)

```
61 @GetMapping(value={"/livroporqtdpgs", "/livroporqtdpgs/{paginas}"})
62 public List<Livro> listarPorQuantidade(@PathVariable Optional<Integer> paginas) {
63     return servico.buscarPorQtdPaginas(paginas.orElse(0));
64 }
65
66 @PostMapping()
67 @ResponseStatus(HttpStatus.CREATED)
68 public Livro incluir(@RequestBody Livro livro) {
69     return servico.salvar(livro);
70 }
71
72 @PutMapping("/{id}")
73 @ResponseStatus(HttpStatus.CREATED)
74 public ResponseEntity<Livro> atualizar(@PathVariable("id") Integer id,
75                                     @RequestBody Livro livro) {
76     if (servico.porCodigo(id).isEmpty()) {
77         return ResponseEntity.notFound().build();
78     }
79
80     livro.setId(id);
81     livro = servico.salvar(livro);
82
83     return ResponseEntity.ok(livro);
84 }
85
86 @DeleteMapping("/{id}")
87 public ResponseEntity<Void> excluir(@PathVariable("id") Integer id) {
88     try {
89         servico.excluir(id);
90     } catch (ObjectNotFoundException e) {
91         return ResponseEntity.notFound().build();
92     }
93
94     return ResponseEntity.noContent().build();
95 }
96 }
```