**Anotações!!!**

**Sobre Camadas**

*Exemplificação!*

É gerada uma requisão (Get para buscar.... Post para enviar) através do front... essa **requisição** é feita para a **Camada de Controle** (Rest), que por sua vez aciona a **camada de serviço**.

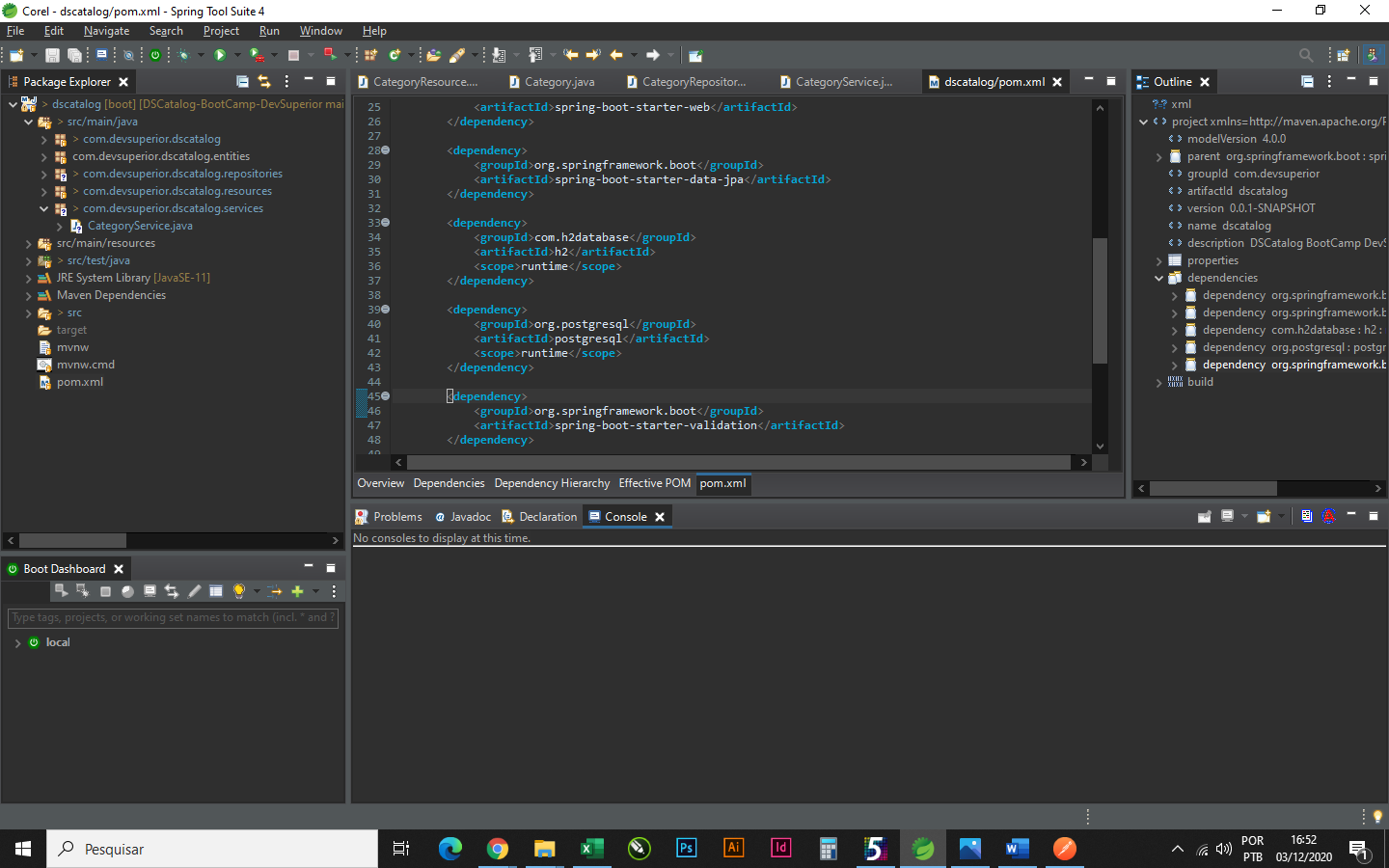
Na camada de serviços é acionado o a camada de acesso a dados, que irá retornar com o resultado da requisição para a camada de serviços, que por sua vez irá responder para a camada de Controle.

Requisição Controle(Rest/Api) Serviço Dados

**JPA**

É um recurso utilizado no java para fazer entre outras coisas o relacionamento da aplicação com o banco de dados, é responsável por “transformar” essas informações em objetos.

Necessário fazer a implementação das dependências no arquivo de configuração do maven, **pom.xml**



***Recursos a considerar do JPA:***

* Inserir as dependências no arquivo de configurações gerais do Spring Boot... POM.XML, buscar na web a versão mais atual do Maven JPA, e utilizar como dependência;
* Método: **ResponseEntity**; irá retornar um objeto com os dados.

**CLASSES RESOURCES**

**CAMADA DE CONTROLADORES**

TODA **CLASSE RESOURCE** SERÁ RESPONSÁVEL POR DISPONIBILIZAR **END POINT**!

**O Que é End-Point ?**

É o recurso que o front-web irá buscar na aplicação!

**Como preparar uma classe “End-Point”**

Acrescentar as annotatios na identificação da classe:

**@RestController**

*Essa annotation será responsável por fazer um pré-compilamento, informando que este será um recurso da classe.*

**@RequestMapping(value=””)**

*Esta annotation é responsável por indicar o caminho a qual o front-web irá chamar. Exemplo: (@RequestMappint(value=”/produtos”) no front-web será chamando pelo endereçoAplicação****/produtos****).*

***Portanto este será o end-point : /produtos***

**@GetMapping**

*Essa annotation informa ao STS que o método a qual ela está sobrescrita irá retornar um recurso para o end-point.*

**CLASSES CATEGORIES**

**CAMADA ENTIDADES**

É classe com os atributos referentes à Categoria (por exemplo).

No inicio da classe acrescenta-se a annotation:

***@Entity***

*Essa anotação vem do pacote* ***javax.persinstence.entity***, esse pacote vem da JPA.

Abaixo da annotation ***@Entity***, acrescente-se a annotation ***@Table***, ela será responsável por criar uma tabela no banco de dados, com o nome que for especificado.

***@Table (name = “tb\_category”)***

Serão utilizadas outras annotations para mostrar ao JPA qual será o campo referente a chave primária (***@Id***) e se ele será auto incrementável (**@GenerateValue(stategy=GenerationType.IDENTIFY**)

Exemplo:

*@Entity*

*@Table*(name= "tb\_category")

public class Category implements Serializable {

private static final long ***serialVersionUID*** = 1L;

*@Id*

*@GeneratedValue*(strategy=*GenerationType*.***IDENTITY***)

private Long id;

private String name;

**CLASSES REPOSITORIES**

**CAMADA ACESSO A DADOS**

Será a classe responsável por conectar os serviços ao banco de dados, irá utilizar recursos do JPA.

Uma classe/“**interface**” que irá **extender** a interface **JpaRepository**.

É um tipo genérico, necessitando identificar os atributos, nome da classe e o atributo do id.

***@Repository***

public interface CategoryRepository **extends JpaRepository<Category, Long**> {

}

É necessário identificar como um componente injetável, para isso utiliza a annotation

**@Repository**.(atenção para a semântica)

**@Repository**

*Irá especificar que a classe/interface é um* ***componente****.*

**CLASSES SERVICES**

**CAMADA DE SERVIÇOS**

A **Camada de serviço** é a responsável por fazer toda parte de serviços, e regras de negócio do sistema, ela faz a **ponte** entre os **Controladores**, e o **Acesso aos Dados**.

No ínicio da classe é adicio a annotatio:

**@Service**

*Ela vai registrar a classe como um componente que irá participar do sistema de injeção de dependências automatizadas do spring, quer dizer que quem vai gerenciar as dependecias será o maven.*

*Utiliza essa annotatio para manter a semântica em relação a camada da classe.*

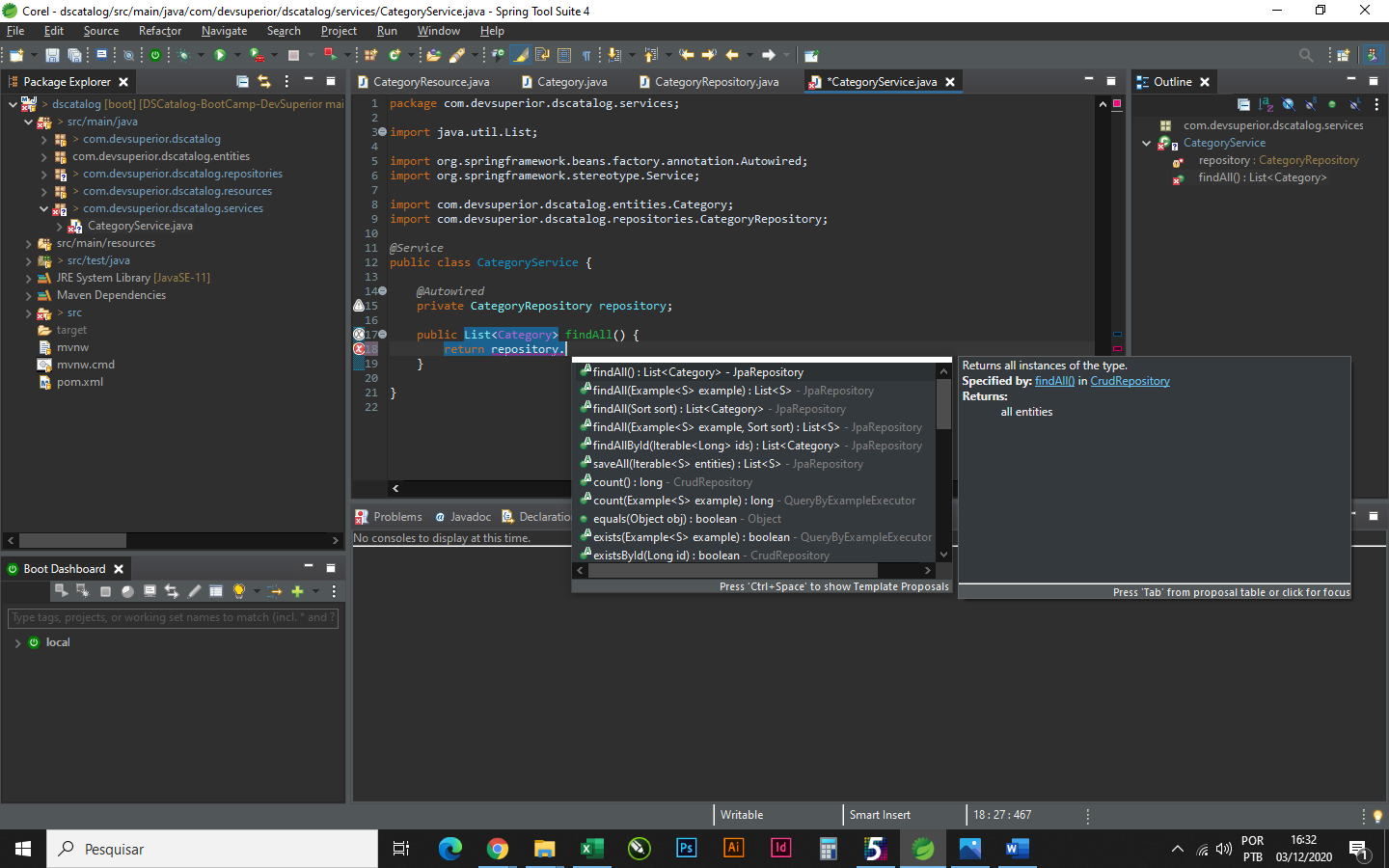
**@AutoWired**  
para que o servisse possa injetar uma instância gerenciada pelo Spring. Ele vai injetar uma dependência válida a qual ela foi instanciada.

Exemplo:

*@Autowired*

**private** CategoryRepository repository;

Com o uso da injeção de dependências, é possível utilizar vários métodos que já estão prontos, economizando escrita de códigos.



**BANCO DE DADOS H2**

O **H2**, é um banco de dados sql que roda **em memória**, é utilizado para testes no **spring**, no desenvolvimento de uma aplicação.

Deve-ser acrescentar a **dependência** no **pom.xml**.

<**dependency**>

<**groupId**>com.h2database</**groupId**>

<**artifactId**>h2</**artifactId**>

<**scope**>runtime</**scope**>

</**dependency**>

Importante, deve-se acrescentar um arquivo de configuração, a pasta **src/main/resources**

*Application.properties*

spring.profiles.active=test

spring.jpa.open-in-view=false

*Application-teste.properties*

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console