



**Pós-Graduação Lato Sensu**  
Curso de Especialização em Tecnologia Java

# **APOSTILA – IMPLEMENTAR CONFIGURAÇÃO PROGRAMÁTICA**

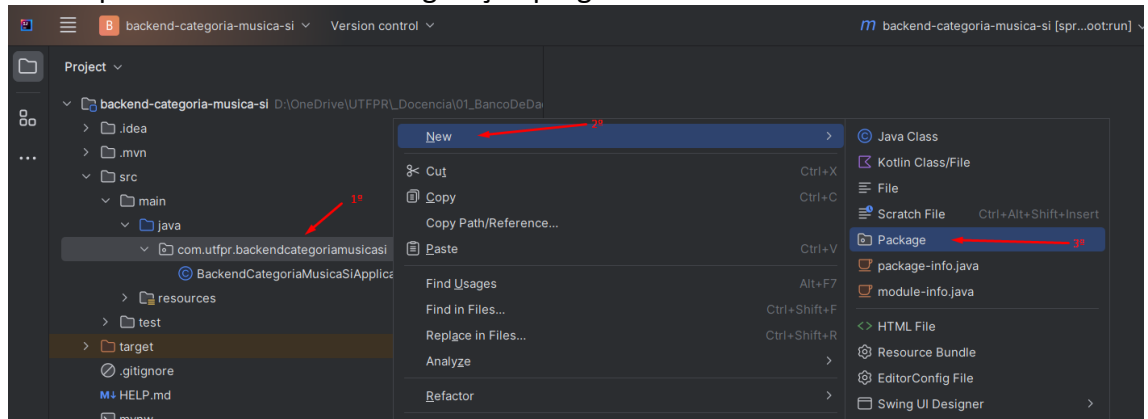
**Professor Hugo Baker Goveia**  
hbakergoveia@hotmail.com



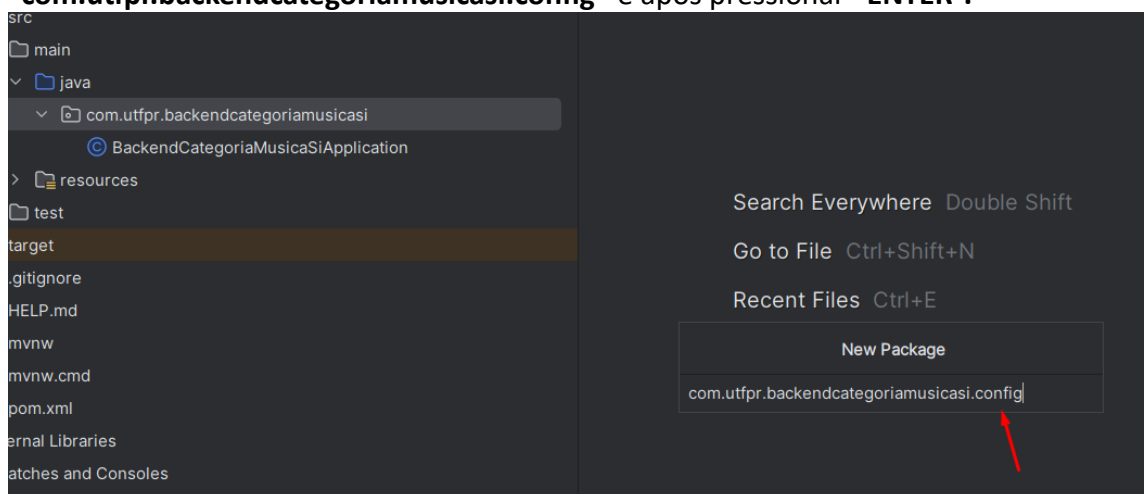
# 1. Implementar configuração programática

Seguiremos as etapas abaixo para implementar a configuração do Spring Data JPA de forma programática:

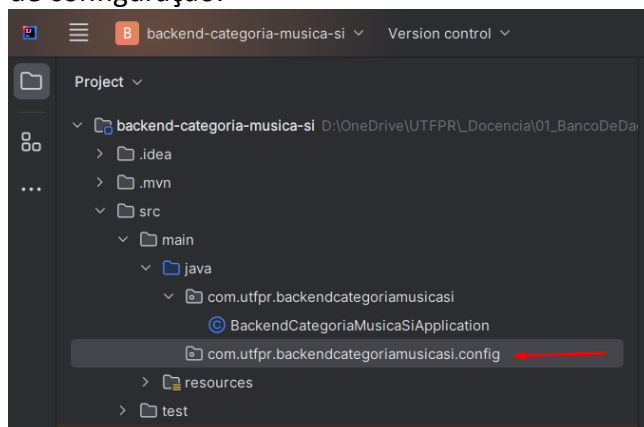
1. Clicar com o botão direito no **pacote raiz do seu projeto** -> **New** -> **Package** para criar o pacote da classe de configuração programática.



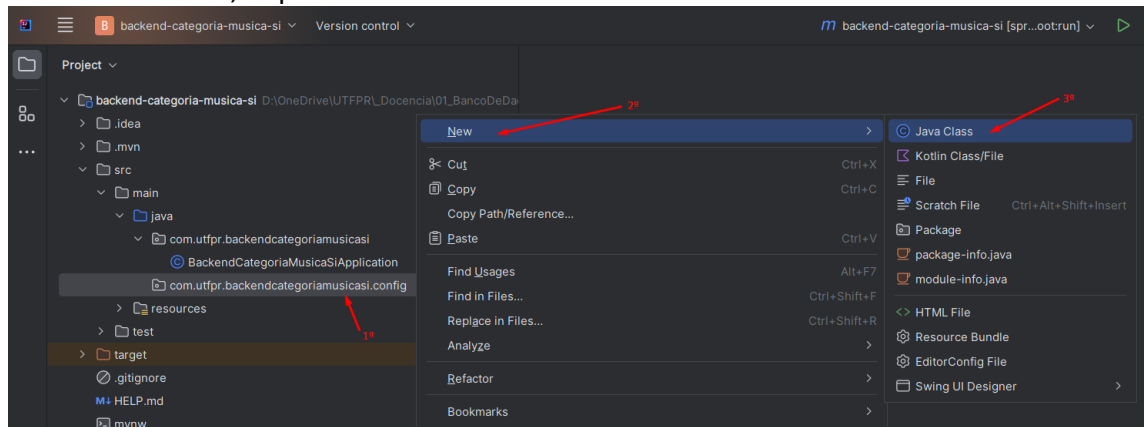
2. Na modal que abrir, informar o nome do pacote que será **"com.utfpr.backendcategoriamicasi.config"** e após pressionar **"ENTER"**.



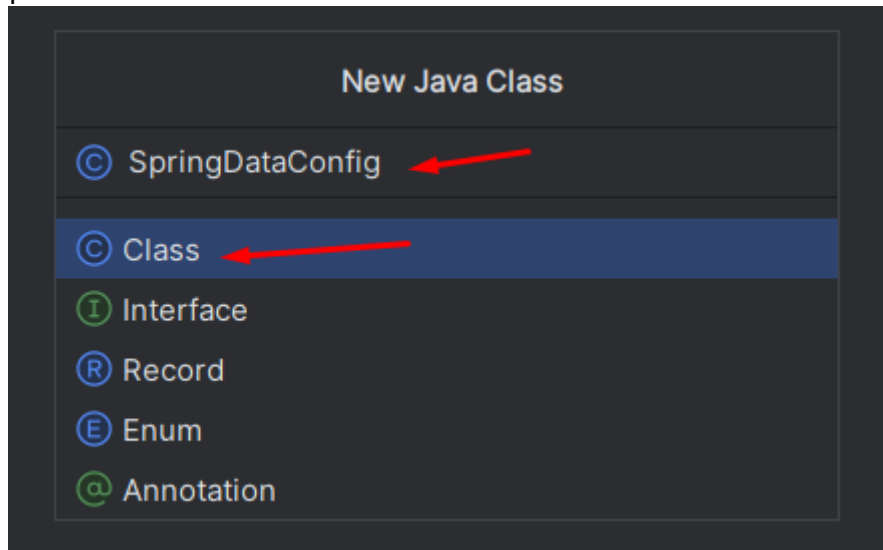
3. Após essa ação o novo pacote é criado e será nele que iremos criar nossa classe de configuração.



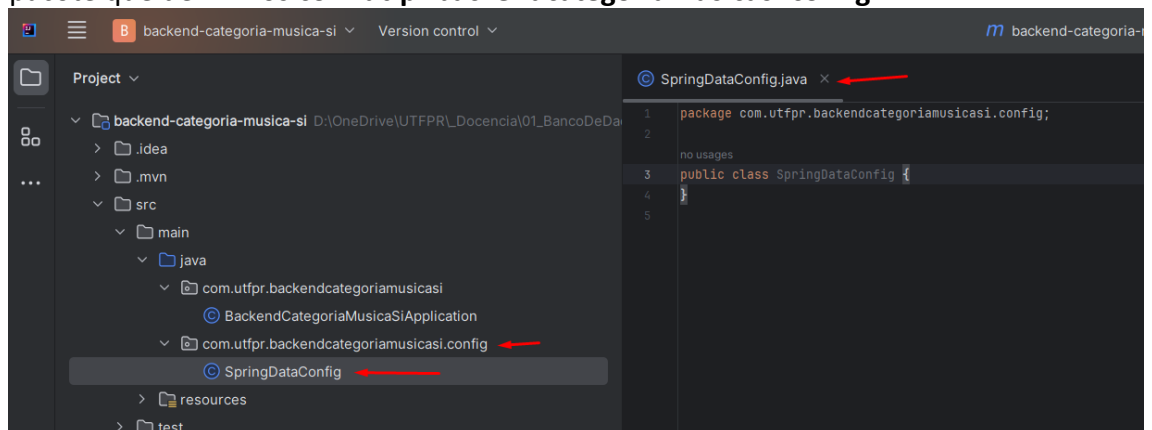
4. Para criar a classe, você deve clicar no pacote que acabamos de criar com o botão direito do mouse, depois **“New->Java Class”**.



5. Manter o “Class” selecionado, informar o nome da classe **“SpringDataConfig”** e pressionar **“Enter”**



6. Observe pela imagem abaixo que a classe **SpringDataConfig** foi criada dentro do pacote que definimos **com.utfpr.backendcategoriamicasi.config**



7. Seguindo o material apresentado no bloco anterior, a primeira parte da implementação será informar as anotações necessárias acima da assinatura da classe, essas anotações são usadas para indicar que a classe será observada pelo Spring como sendo de configuração e define suas características, que são:

- Digite “**@Configuration**” que é uma anotação do Spring Framework para marcar a classe como sendo uma classe de configuração. Este tipo de classe será uma das primeiras a ser inicializada pelo Spring para liberar os recursos que estão configurados nela;

- Digite “**@EnableJpaRepositories("Informar pacote do repository")**” e informe o parâmetro “**com.utfpr.backendcategoriamusica.repository**”, esta anotação tem dois objetivos.

*O primeiro* é habilitar, ou dizer ao Spring Framework, que os recursos do Spring Data JPA serão usados no projeto.

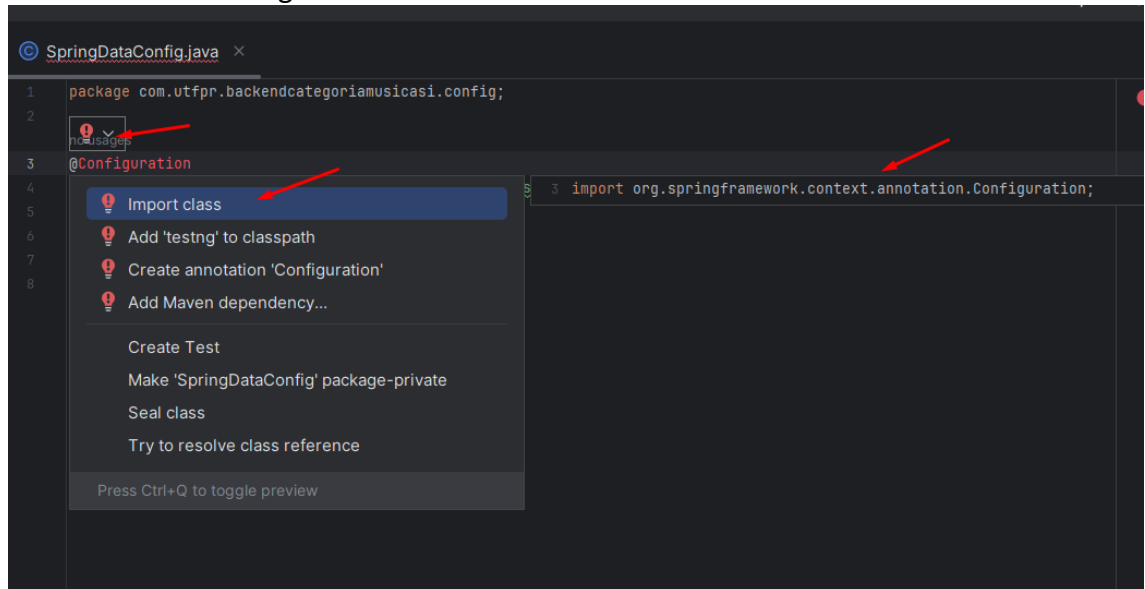
*O segundo* é informar em qual pacote da aplicação estarão as interfaces que vão representar os repositórios do SpringData JPA, ou seja, a camada de persistência de seu projeto e nesse exemplo ficará da seguinte forma:

**@EnableJpaRepositories("com.utfpr.backendcategoriamusica.repository ")**

- Digite “**@EnableTransactionManagement**”, a função desta anotação é habilitar no Spring o controle transacional com o banco de dados. Deste modo, o Spring é quem será o responsável em lidar com as transações;

```
© SpringDataConfig.java ×
1 package com.utfpr.backendcategoriamusicasi.config;
2
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
5 import org.springframework.transaction.annotation.EnableTransactionManagement;
6
7 no usages
8 @Configuration
9 @EnableJpaRepositories("com.utfpr.backendcategoriamusicasi.repository")
10 @EnableTransactionManagement
11 public class SpringDataConfig {
12 }
```

8. Na imagem anterior, podemos observar que a IDE do IntelliJ realiza os imports automaticamente, porém caso isso não aconteça e você precise fazer as importações devemos **clicar na lâmpada vermelha com o ponto de exclamação** -> **Import Class** -> **escolher a importação correta nas opções sugeridas** conforme demonstrado na imagem abaixo.



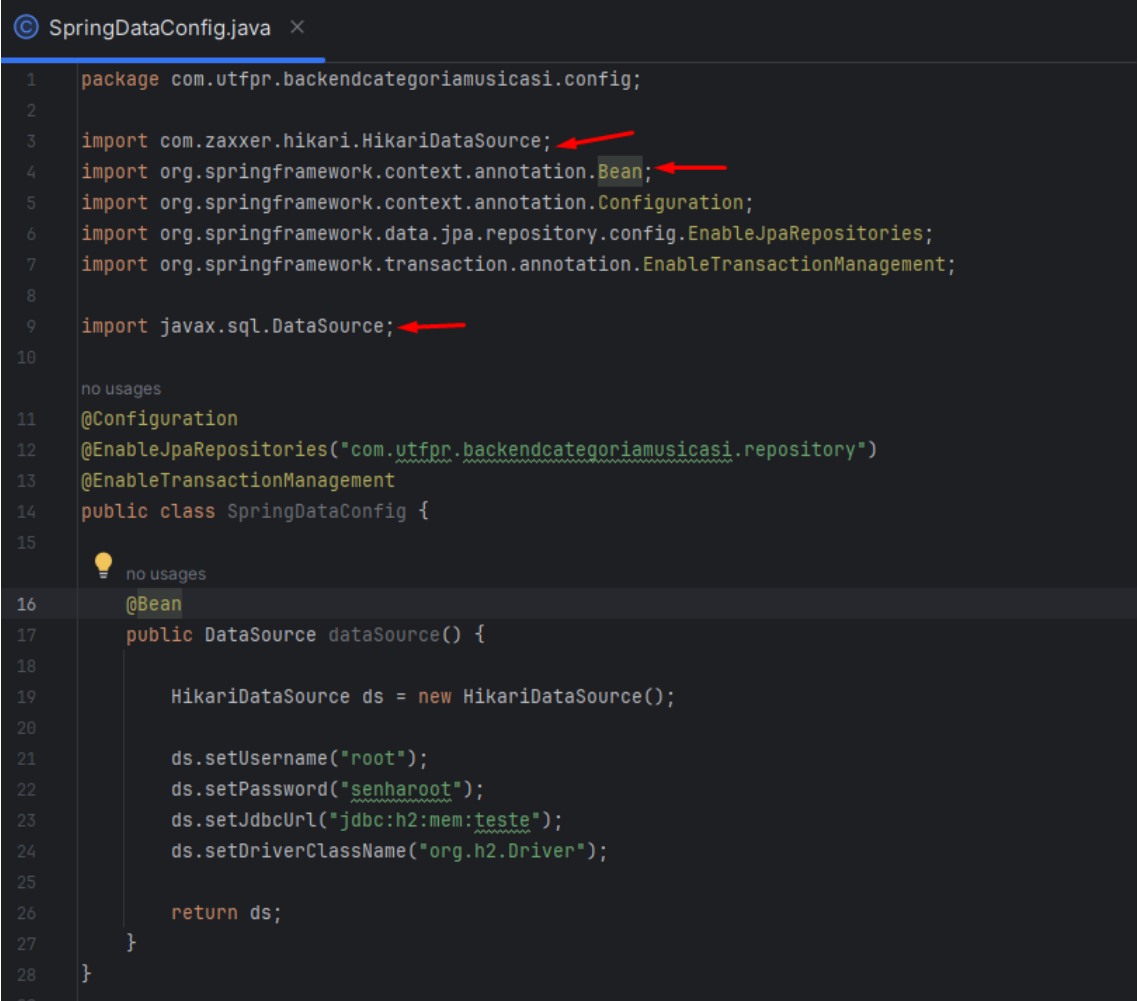
Realize os passos acima até todas as importações estarem corretas. Abaixo seguem as importações necessárias até esse momento.

```
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.transaction.annotation.EnableTransactionManagement;
```

9. Implementar o método `DataSource()` passando como parâmetro as configurações para conexão com o banco de dados e realizar as importações das áreas indicadas pelas setas na imagem abaixo.

No material introdutório apresentado no bloco anterior, foi exemplificado na **Listagem 2.2: MÉTODO DATASOURCE()** a conexão com o banco de dados MariaDB.

No exemplo dessa apostila, usaremos o banco de dados H2, conforme imagem abaixo.



```
1 package com.utfpr.backendcategoriamusicasi.config;
2
3 import com.zaxxer.hikari.HikariDataSource;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
7 import org.springframework.transaction.annotation.EnableTransactionManagement;
8
9 import javax.sql.DataSource;
10
11 no usages
12 @Configuration
13 @EnableJpaRepositories("com.utfpr.backendcategoriamusicasi.repository")
14 @EnableTransactionManagement
15 public class SpringDataConfig {
16
17     no usages
18     @Bean
19     public DataSource dataSource() {
20
21         HikariDataSource ds = new HikariDataSource();
22
23         ds.setUsername("root");
24         ds.setPassword("senharoot");
25         ds.setJdbcUrl("jdbc:h2:mem:teste");
26         ds.setDriverClassName("org.h2.Driver");
27
28         return ds;
29     }
30 }
```

Em alguns casos é necessário inserir a dependência do Hikari no pom.xml, se for o seu caso entre no pom.xml inclua a dependência abaixo e clique em salvar.

```
<dependency>
    <groupId>com.zaxxer</groupId>
    <artifactId>HikariCP</artifactId>
</dependency>
```

10. Implementar o método **entityManagerFactory()** que é o responsável por criar a fábrica de gerenciamento de entidades via JPA. Esse método foi apresentado no bloco anterior na **LISTAGEM 2.3: MÉTODO ENTITYMANAGERFACTORY()**

```
@Bean
public EntityManagerFactory entityManagerFactory() {
    LocalContainerEntityManagerFactoryBean factory =
        new LocalContainerEntityManagerFactoryBean();

    HibernateJpaVendorAdapter vendorAdapter =
        new HibernateJpaVendorAdapter();
    vendorAdapter.setGenerateDdl(true);
    vendorAdapter.setShowSql(false);

    factory.setDataSource(dataSource());
    factory.setJpaVendorAdapter(vendorAdapter);
    factory.setPackagesToScan("com.utfpr.backend.categoriamusicasi.entity");
    factory.afterPropertiesSet();

    return factory.getObject();
}
```

É importante lembrar que é preciso informar como parâmetro no **factory.setPackagesToScan()** o endereço completo do pacote onde serão implementadas nossas classes entidades do projeto futuramente, nesse exemplo usaremos o **entity**.

11. Realizar a importação do **HibernateJpaVendorAdapter**, do **EntityManagerFactory** e do **LocalContainerEntityManagerFactoryBean**

```
SpringDataConfig.java x
1 package com.utfpr.backend.categoriamusicasi.config;
2
3 import com.zaxxer.hikari.HikariDataSource;
4 import jakarta.persistence.EntityManagerFactory;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
8 import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
9 import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
10 import org.springframework.transaction.annotation.EnableTransactionManagement;
11
12 import javax.sql.DataSource;
13
```

12. Implementar o método **transactionManager()** que foi apresentado no bloco anterior na **LISTAGEM 2.4: MÉTODO TRANSACTIONMANAGER()**.

```
no usages
@Bean
public PlatformTransactionManager transactionManager() {

    JpaTransactionManager manager = new JpaTransactionManager();
    manager.setEntityManagerFactory(entityManagerFactory());
    manager.setJpaDialect(new HibernateJpaDialect());

    return manager;
}
```

13. Realizar a importação do **PlatformTransactionManager**, do **JpaTransactionManager** e do **HibernateJpaDialect**

```
© SpringDataConfig.java ×
1 package com.utfpr.backendcategoriamicasasi.config;
2
3 import com.zaxxer.hikari.HikariDataSource;
4 import jakarta.persistence.EntityManagerFactory;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
8 import org.springframework.orm.jpa.JpaTransactionManager; ←
9 import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
10 import org.springframework.orm.jpa.vendor.HibernateJpaDialect; ←
11 import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
12 import org.springframework.transaction.PlatformTransactionManager; ←
13 import org.springframework.transaction.annotation.EnableTransactionManagement;
14
15 import javax.sql.DataSource;
16
17 no usages
18 @Configuration
19 @EnableJpaRepositories("com.utfpr.backendcategoriamicasasi.repository")
```