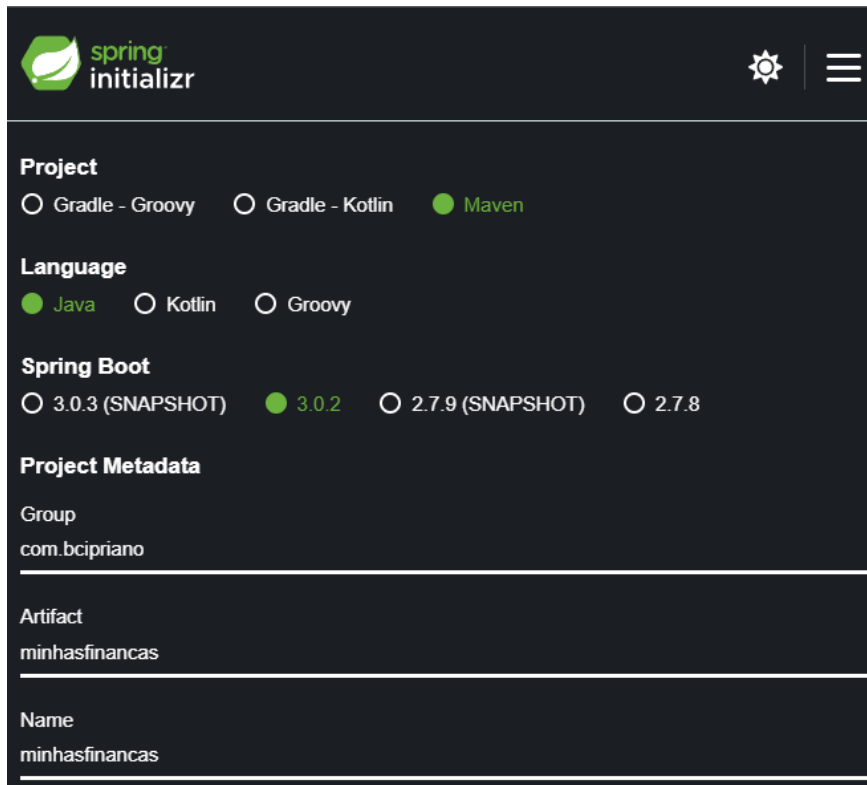


Documentação completa do curso Spring Boot com React

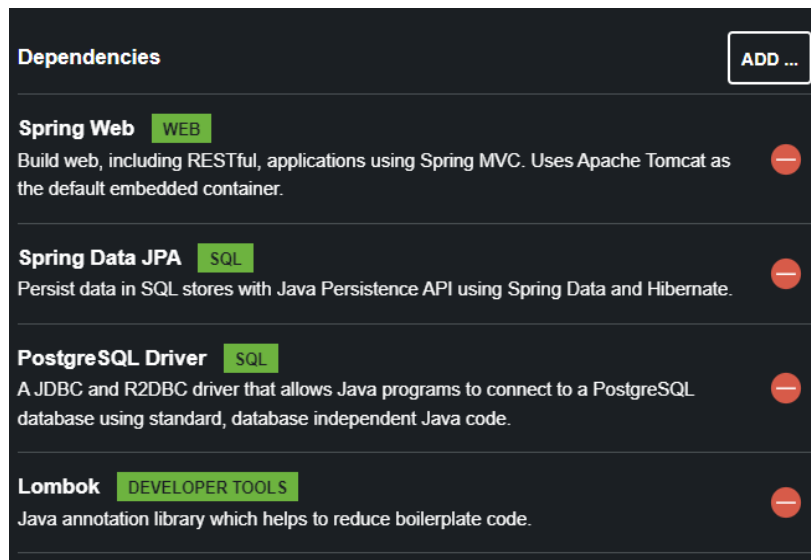
Inicialização do projeto Spring Boot

Para criar um novo projeto, vá em <https://start.spring.io/> e adicione as seguintes configurações:



The image shows the Spring Initializr web interface. At the top, there's a header with the Spring logo and 'initializr' text, a settings gear icon, and a hamburger menu icon. Below the header, the configuration is organized into sections: 'Project' with radio buttons for 'Gradle - Groovy', 'Gradle - Kotlin', and 'Maven' (selected); 'Language' with radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for '3.0.3 (SNAPSHOT)', '3.0.2' (selected), '2.7.9 (SNAPSHOT)', and '2.7.8'; and 'Project Metadata' with text input fields for 'Group' (com.bcipriano), 'Artifact' (minhasfinancas), and 'Name' (minhasfinancas).

Feito isso, podemos adicionar as seguintes dependências:



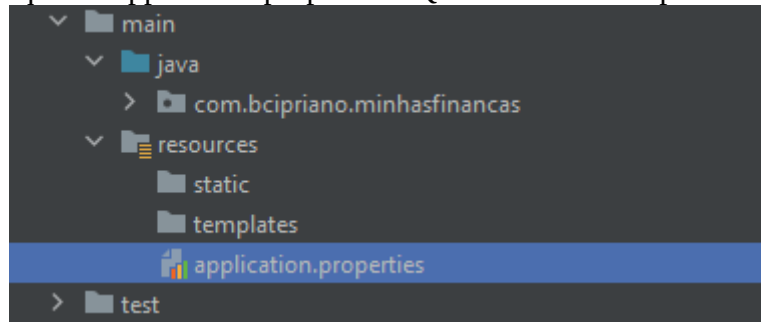
The image shows the 'Dependencies' section of the Spring Initializr interface. It has a title 'Dependencies' and an 'ADD ...' button. Below the title, there are four dependency entries, each with a category tag, a description, and a red minus button to remove it: 'Spring Web' (WEB) with description 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'; 'Spring Data JPA' (SQL) with description 'Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.'; 'PostgreSQL Driver' (SQL) with description 'A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.'; and 'Lombok' (DEVELOPER TOOLS) with description 'Java annotation library which helps to reduce boilerplate code.'

Feito isso, clique em “GENERATE” e extraia o arquivo baixado na pasta que desejar.

Após extrair o arquivo, abra a IDE que irá usar e importe o arquivo extraído. Isso pode demorar algum tempo até que todas as dependências do Maven sejam baixadas.

Configuração da String de conexão com banco de dados

Para realizar a conexão com o banco de dados PostgreSQL é necessário passar uma String de conexão dentro do arquivo “application.properties” Que fica dentro da pasta resources. Veja:



Dentro dele, adicione as seguintes linhas:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/minhasfinancas
spring.datasource.username=postgres
spring.datasource.password=?
spring.datasource.driver-class-name=org.postgresql.Driver
```

Configuração do banco de dados PostgreSQL

Abra seu gerenciador de banco de dados PostgreSQL e crie uma nova base de dados com nome “minhasfinancas”. Feito isso, adicione as seguintes linhas SQL: (Execute sequencialmente)

```
CREATE DATABASE minhasfinancas;
```

```
CREATE SCHEMA financas;
```

```
CREATE TABLE financas.usuario
(
  id bigserial NOT NULL PRIMARY KEY,
  nome character varying(150),
  email character varying(100),
  senha character varying(20),
  data_cadastro date default now()
);
```

```
CREATE TABLE financas.lancamento
(
  id bigserial NOT NULL PRIMARY KEY ,
  descricao character varying(100) NOT NULL,
  mes integer NOT NULL,
  ano integer NOT NULL,
  valor numeric(16,2) NOT NULL,
  tipo character varying(20) CHECK(tipo in('RECEITA', 'DESPESA')) NOT NULL,
  status character varying(20) CHECK(status in('PENDENTE', 'CANCELADO', 'EFETIVADO'))
  NOT NULL,
  id_usuario bigint REFERENCES financas.usuario (id) NOT NULL,
  data_cadastro date default now()
);
```

Teste de conexão com banco de dados

Depois de inserir a String de conexão e criar a base de dados, execute o arquivo “MinhasfinancasApplicationTests” no pacote “/test/java/com/bcipriano/minhasfinancas/”, crie um novo pacote de nome “databaseConnectionTest”. Dentro dele, adicione uma nova classe java com as seguintes linhas de código:

```
package com.bcipriano.minhasfinancas.databaseConnectionTest;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.test.context.SpringBootTest;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import static org.junit.jupiter.api.Assertions.assertTrue;
@SpringBootTest
public class DatabaseConnectionTest {
    @Value("${spring.datasource.url}")
    private String url;

    @Value("${spring.datasource.username}")
    private String username;

    @Value("${spring.datasource.password}")
    private String password;
    @Test
    public void testConnection() {

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            System.out.println("Successfully connected to the database.");

            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT 1");
            assertTrue(resultSet.next());
            System.out.println("Successfully executed SELECT 1 statement.");
        } catch (SQLException e) {
            System.out.println("Failed to connect to the database: " + e.getMessage());
            assertTrue(false);
        }
    }
}
```

Feito isso, inicie o teste da classe DatabaseConnectionTest e verifique tudo está ok com a conexão com o banco de dados.

