Cento Universitário UNA

Sistemas de Informação

Tecnologias Emergentes

Práticas de Laboratório

Wesley Dias Maciel

2019/02

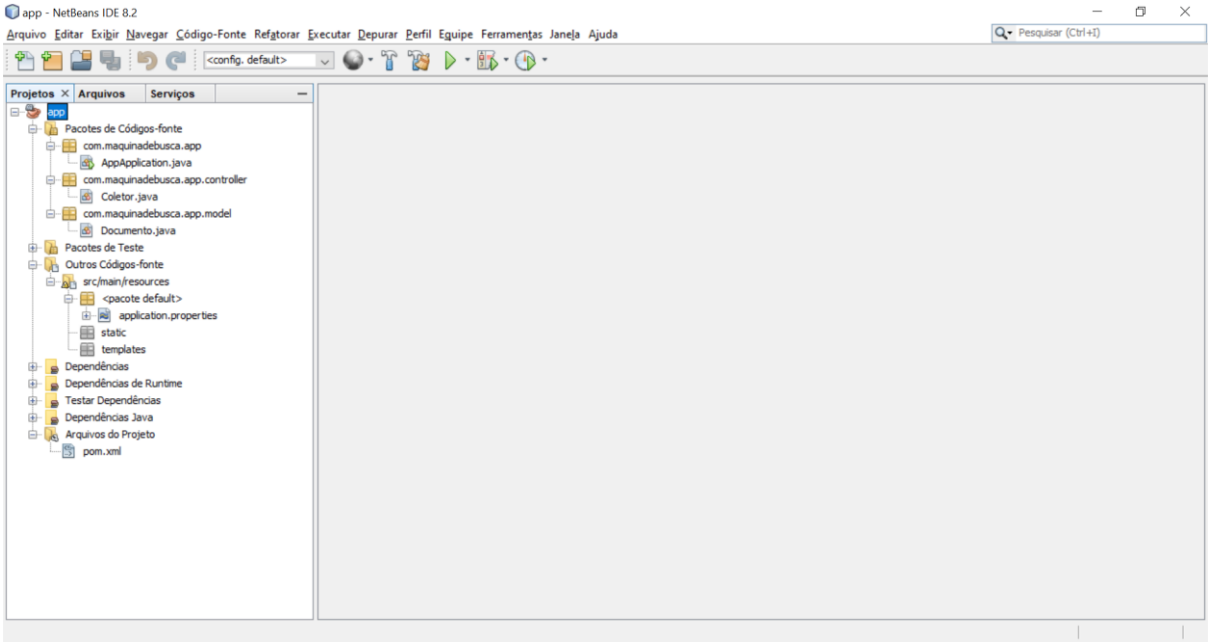
Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

Spring Boot, Hibernate e JPA

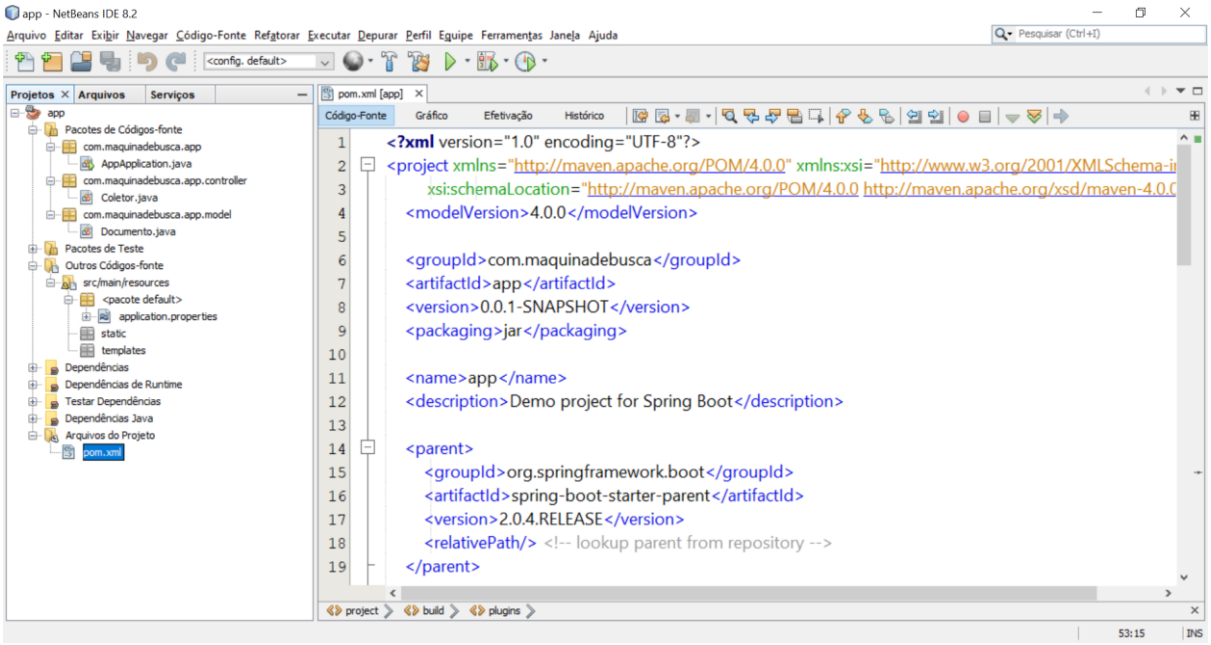
Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

Prática 03

1) Observe abaixo a estrutura atual do projeto.



Abra o arquivo **pom.xml** (Project Object Model) na pasta **Arquivos do Projeto**.

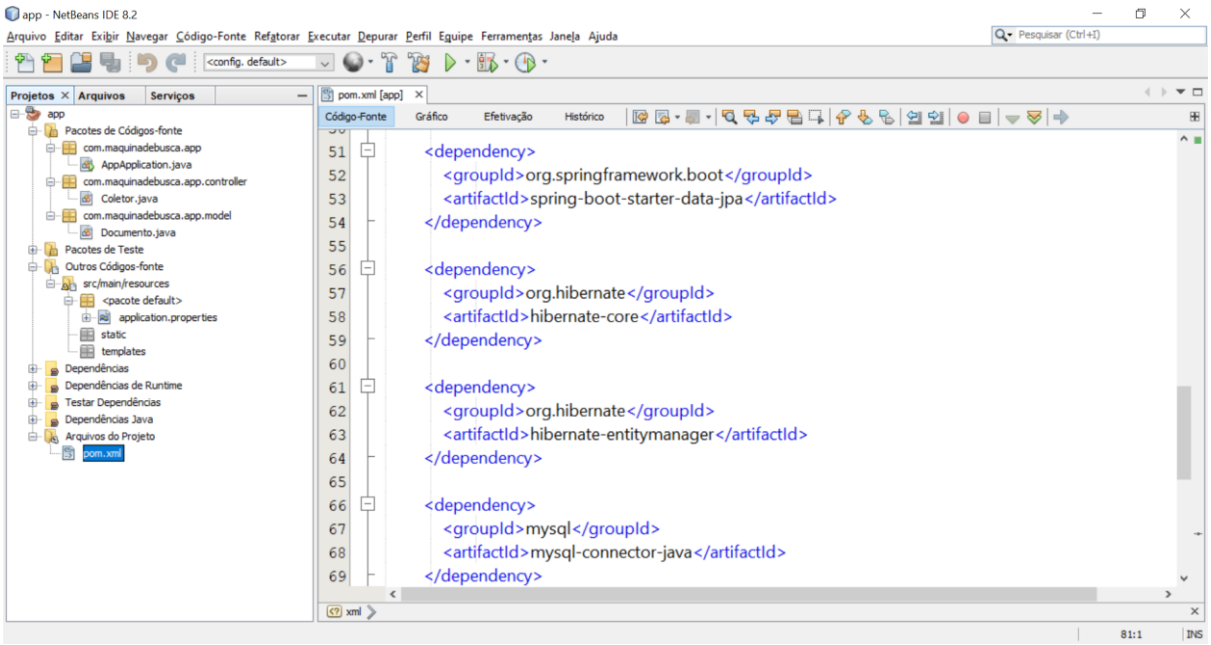


Vá até o elemento XML **<dependencies>** e insira as dependências para JPA, Hibernate e MySQL, conforme apresentado abaixo. Em seguida, salve as alterações realizadas no projeto.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.maquinadebusca</groupId>
  <artifactId>app</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>app</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.jsoup</groupId>
      <artifactId>jsoup</artifactId>
      <version>1.10.2</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
    </dependency>

    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-entitymanager</artifactId>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

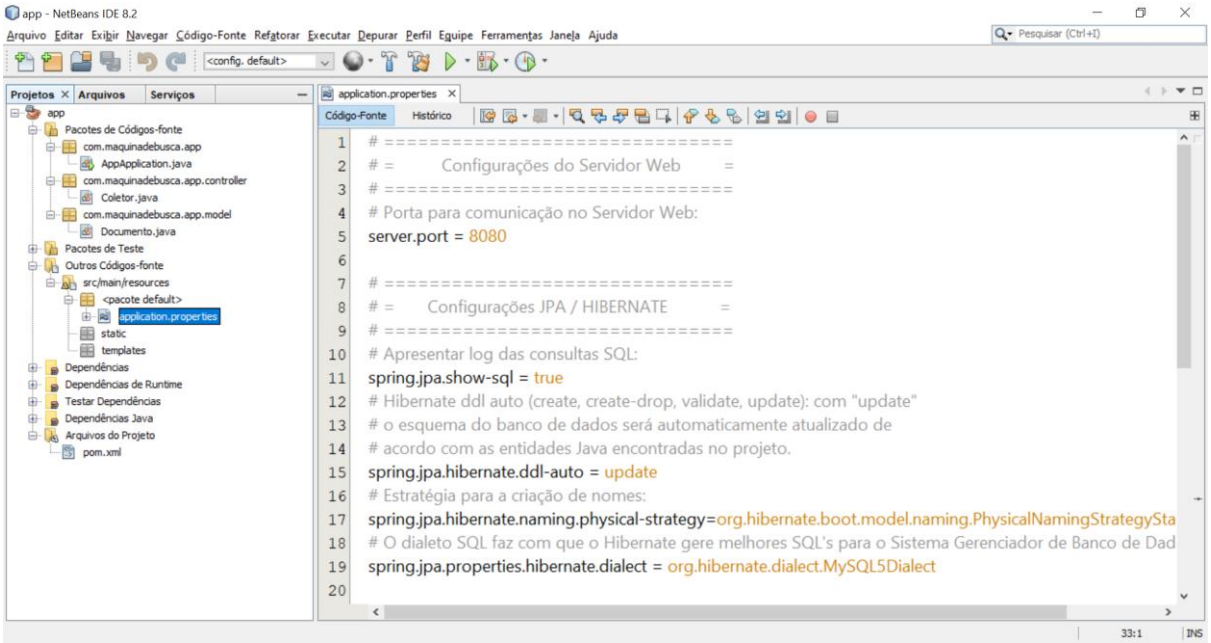


Em seguida, altere o conteúdo do arquivo **application.properties** no "**pacote default**" da pasta "**src/main/resources**", conforme apresentado abaixo:

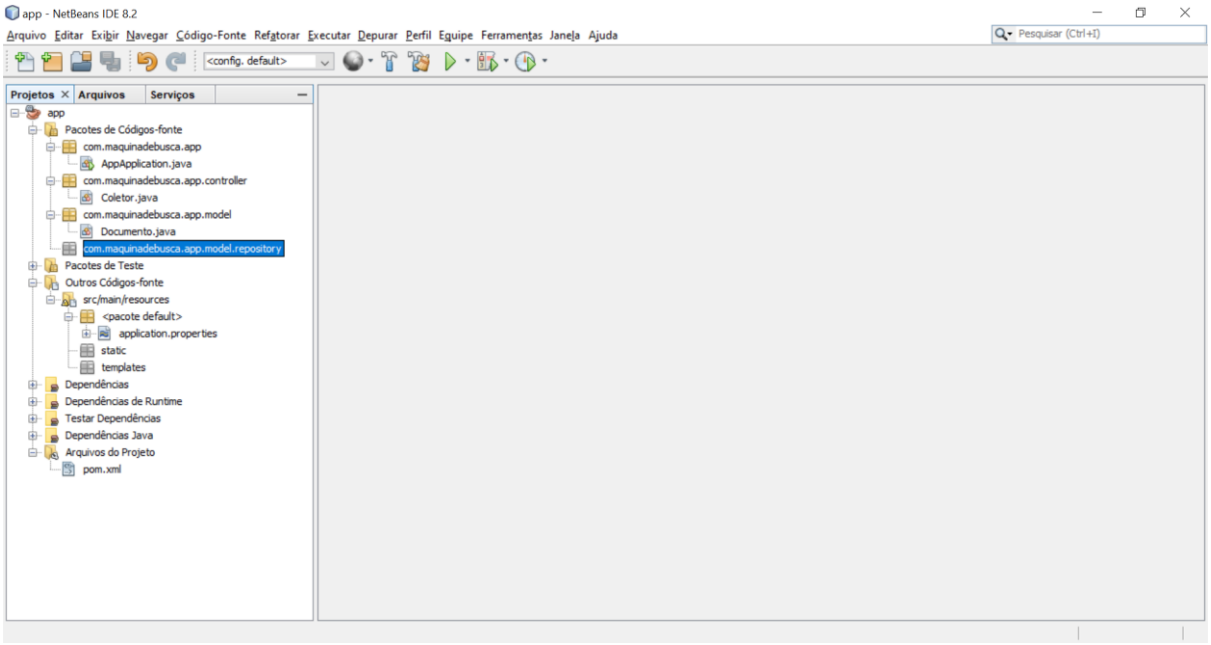
```
# ================================
# =        Configurações do Servidor Web       =
# ================================
# Porta para comunicação no Servidor Web:
server.port = 8080

# ================================
# =      Configurações JPA / HIBERNATE         =
# ================================
# Apresentar log das consultas SQL:
spring.jpa.show-sql = true
# Hibernate ddl auto (create, create-drop, validate, update): com "update"
# o esquema do banco de dados será automaticamente atualizado de
# acordo com as entidades Java encontradas no projeto.
spring.jpa.hibernate.ddl-auto = update
# Estratégia para a criação de nomes:
spring.jpa.hibernate.naming.physical-
strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
# O dialeto SQL faz com que o Hibernate gere melhores SQL's para o Sistema Gerenciador
de Banco de Dados escolhido:
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
```

```
# ====================================
# =                DATA SOURCE       =
# = Configurações para conexão com o banco de dados. =
# ====================================
# URL de conexão com o banco "maquinadebusca":
spring.datasource.url = jdbc:mysql://localhost:3306/maquinadebusca
# Username and password:
spring.datasource.username = root
spring.datasource.password = root
# Mantém a conecção aberta se o sistema ficar inativo por muito tempo (necessário em produção):
spring.datasource.testWhileIdle = true
spring.datasource.validationQuery = SELECT 1
```

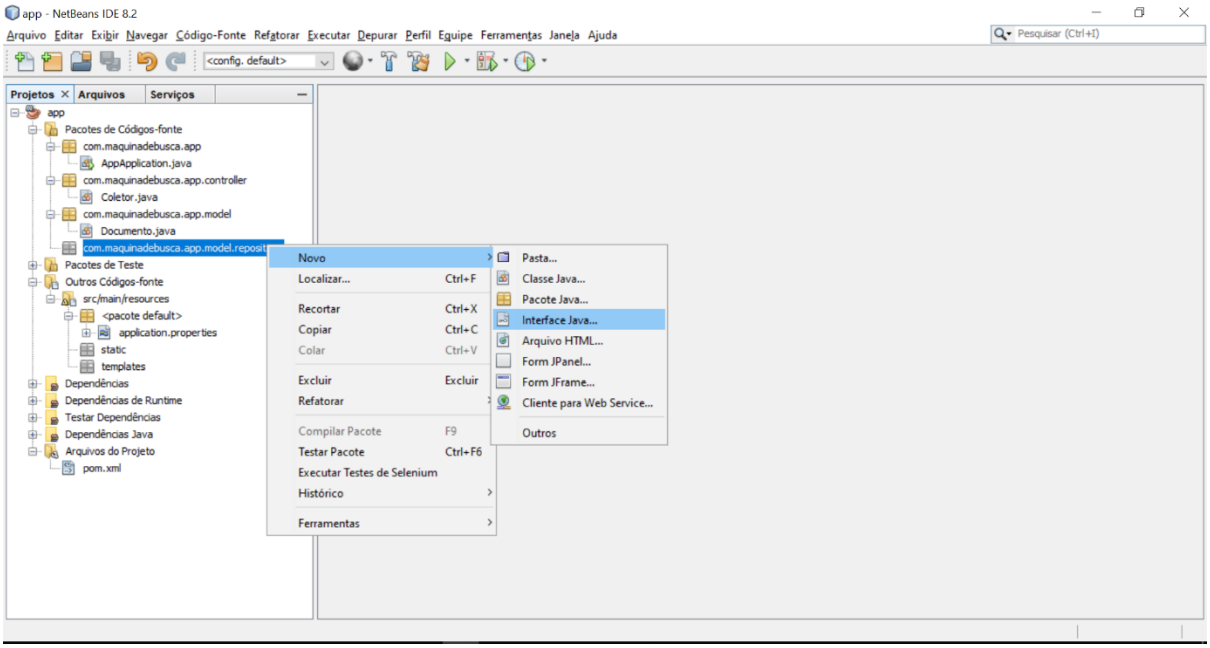
OBS: altere os valores de spring.datasource.username e de spring.datasource.password de acordo com as configurações de sua instalação do MySQL.



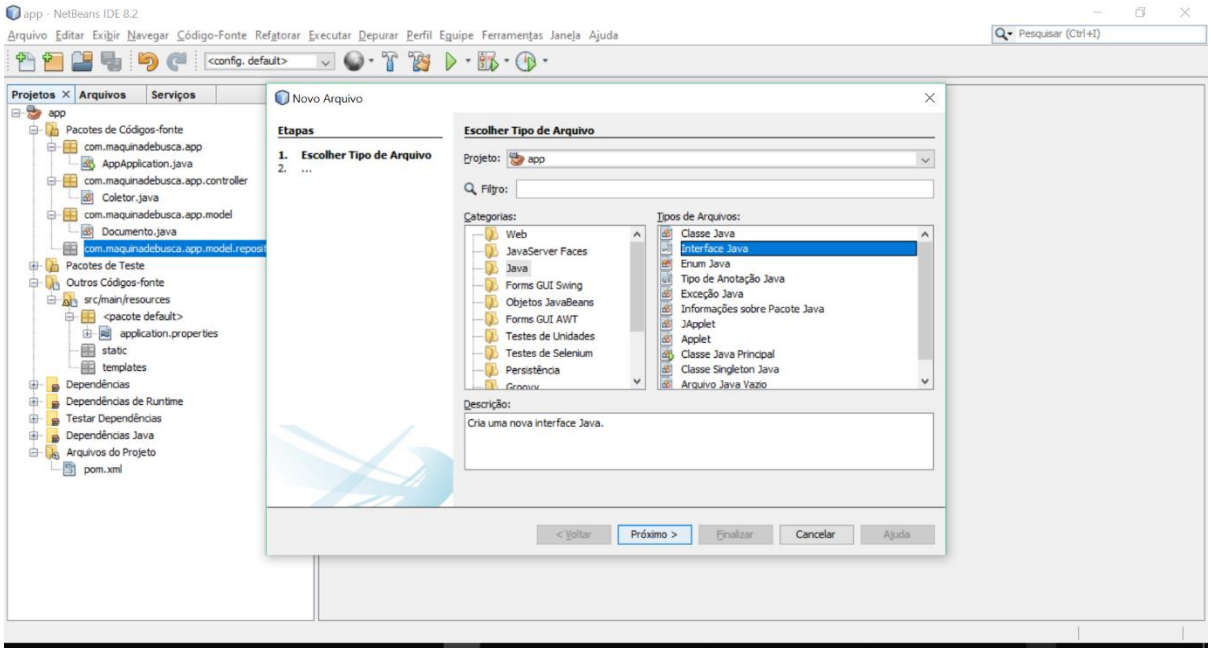
Crie o pacote “**com.maquinadebusca.app.model.repository**”.



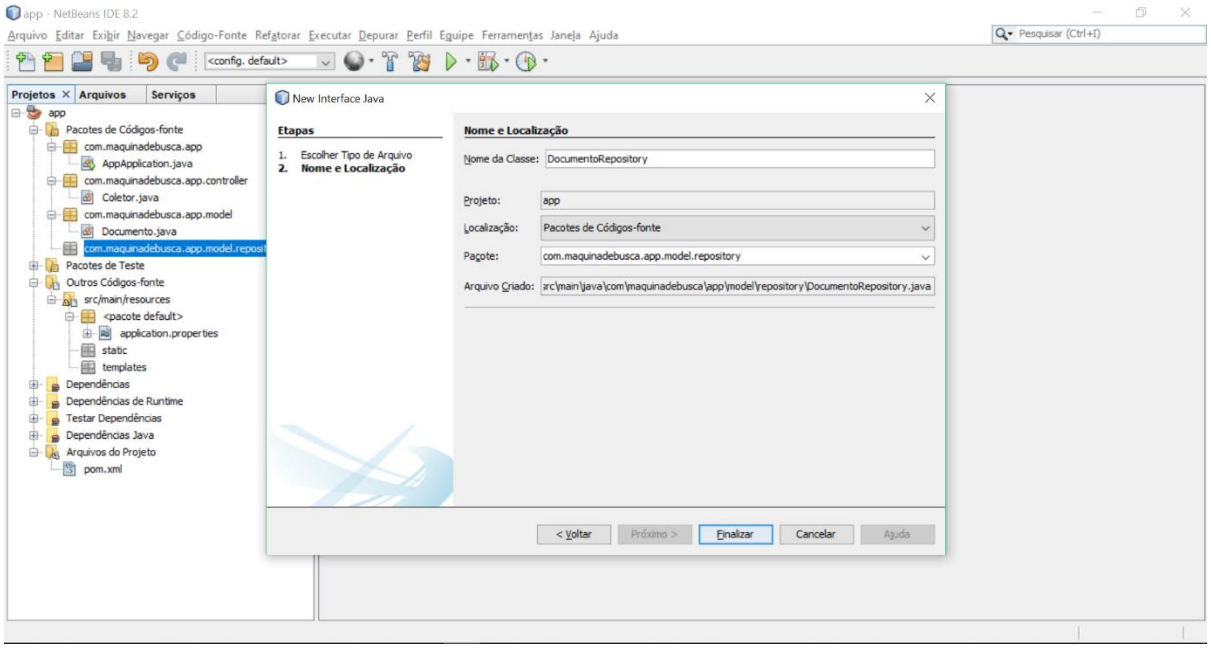
Clique com o botão direito do mouse sobre o pacote "**com.maquinadebusca.app.model.repository**". Em seguida, clique em **Novo**. Depois clique em **Interface Java**.



Caso o menu não apresente a opção **Interface Java**, clique em **Outros** e procure por **Interface Java** na janela que se abre. Em seguida, clique em Próximo.



Na janela que se abre, informe **DocumentoRepository** no campo **Nome da Classe**. Depois, clique em **Finalizar**.

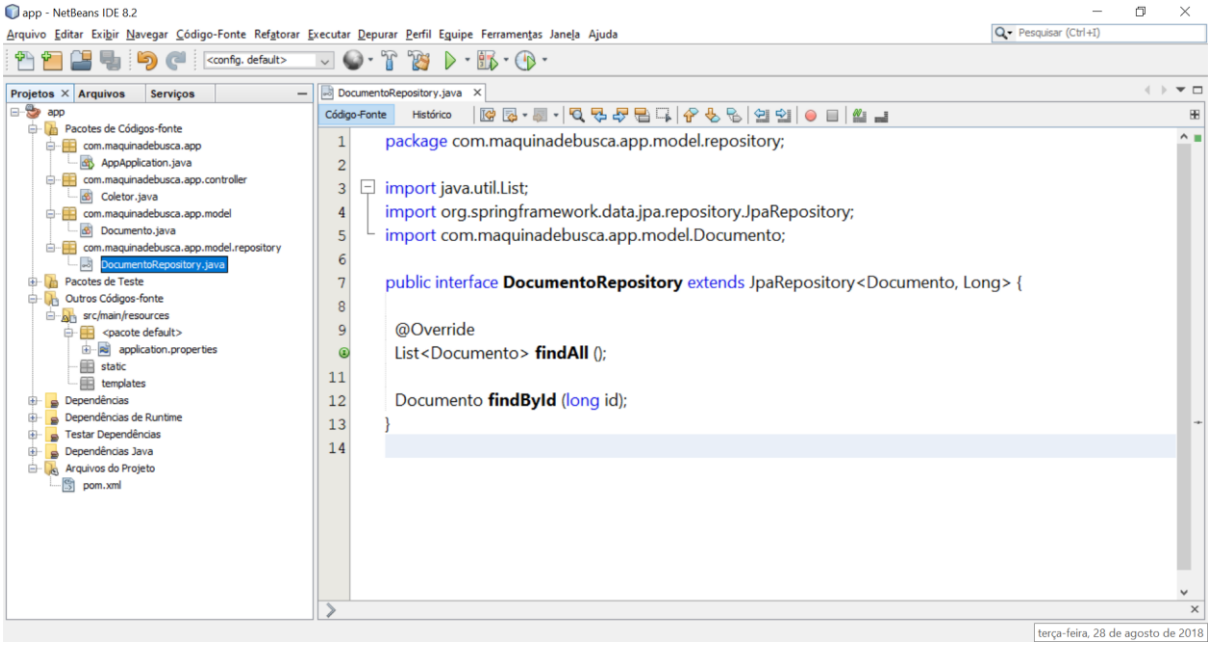


Crie a interface DocumentoRepository, como apresentado abaixo.

```
package com.maquinadebusca.app.model.repository;

import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.maquinadebusca.app.model.Documento;
```

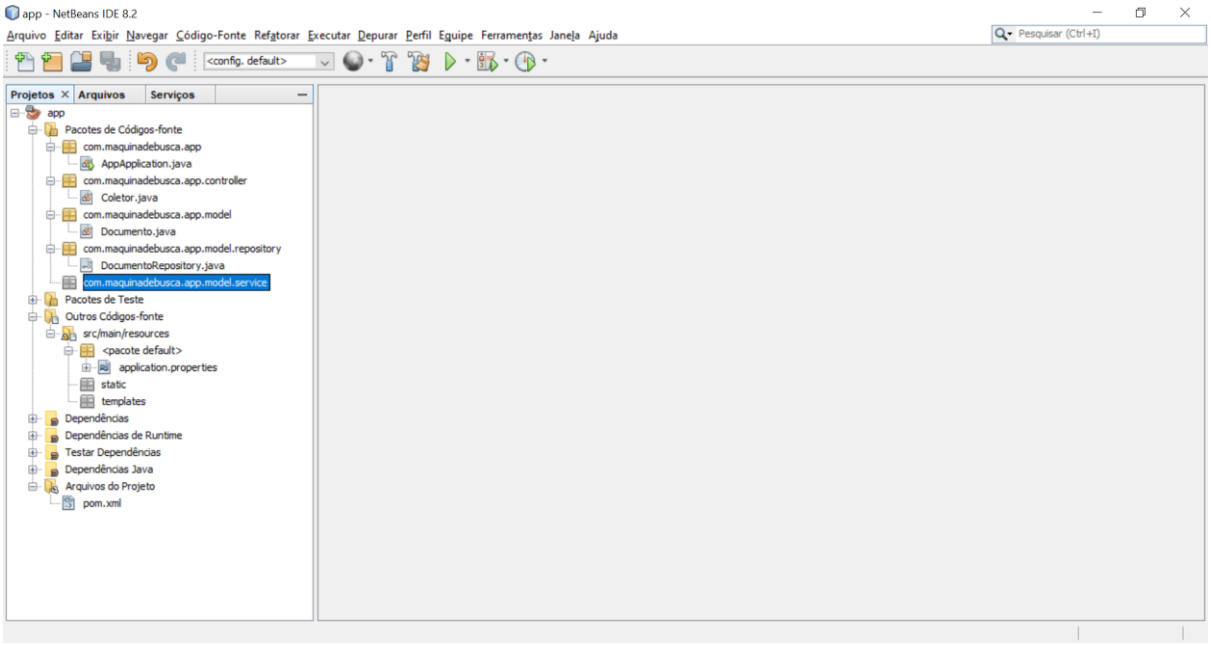
Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

```
public interface DocumentoRepository extends JpaRepository<Documento, Long> {

 @Override
 List<Documento> findAll ();

 Documento findById (long id);
}
```



Crie o pacote "**com.maquinadebusca.app.model.service**".

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

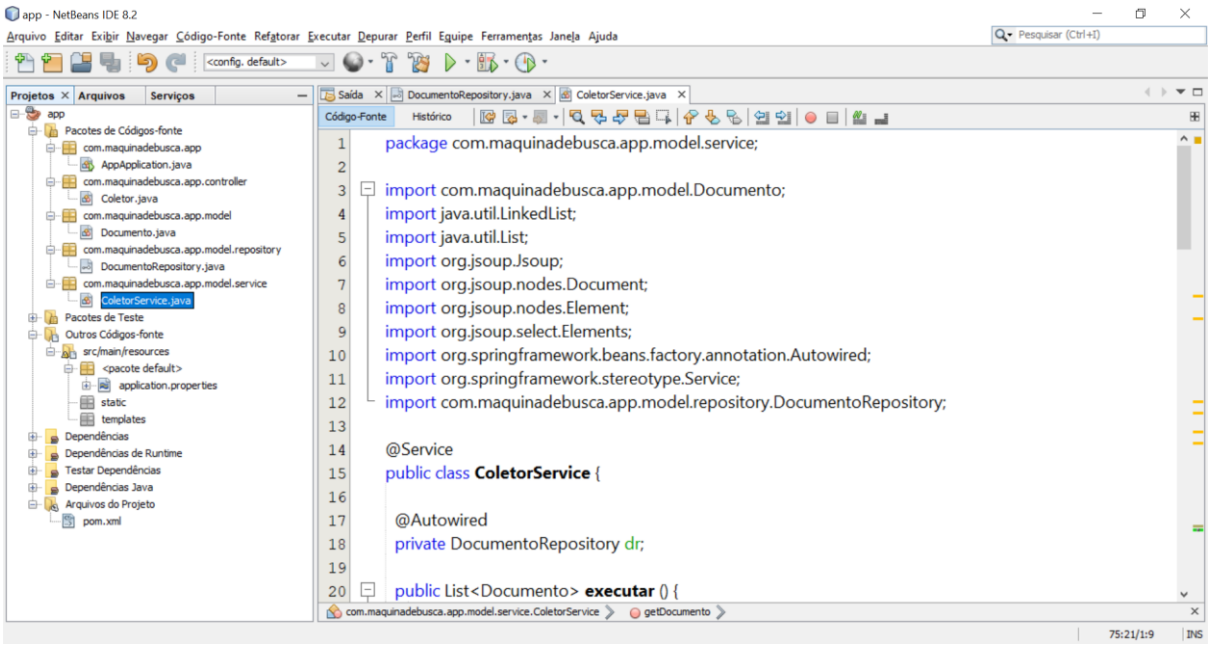


No pacote "**com.maquinadebusca.app.model.service**", crie a classe ColetorService, como apresentado abaixo.

```
package com.maquinadebusca.app.model.service;

import com.maquinadebusca.app.model.Documento;
import java.util.LinkedList;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.maquinadebusca.app.model.repository.DocumentoRepository;

@Service
public class ColetorService {

 @Autowired
 private DocumentoRepository dr;

 public List<Documento> executar () {
  List<Documento> documentos = new LinkedList ();
  List<String> sementes = new LinkedList ();

  try {
```

```
    sementes.add ("https://www.youtube.com/");
    sementes.add ("https://www.facebook.com/");
    sementes.add ("https://www.twitter.com/");

    for (String url : sementes) {
      documentos.add (this.coletar (url));
    }
   } catch (Exception e) {
    System.out.println ("Erro ao executar o serviço de coleta!");
    e.printStackTrace ();
   }
   return documentos;
}

public Documento coletar (String urlDocumento) {
  Documento documento = new Documento ();

  try {
    Document d = Jsoup.connect (urlDocumento).get ();
    Elements urls = d.select ("a[href]");

    documento.setUrl (urlDocumento);
    documento.setTexto (d.html ());
    documento.setVisao (d.text ());

    for (Element url : urls) {
      String u = url.attr ("abs:href");
      if ((!u.equals ("")) && (u != null)) {
        System.out.println (u);
      }
    }
  } catch (Exception e) {
    System.out.println ("Erro ao coletar a página.");
    e.printStackTrace ();
  }
  documento = dr.save (documento);
  return documento;
}

public List<Documento> getDocumentos () {
  Iterable<Documento> documentos = dr.findAll ();
  List<Documento> resposta = new LinkedList ();
  for (Documento documento : documentos) {
```

```
    resposta.add (documento);
   }
   return resposta;
  }

  public Documento getDocumento (long id) {
   Documento documento = dr.findById (id);
   return documento;
  }
}
```

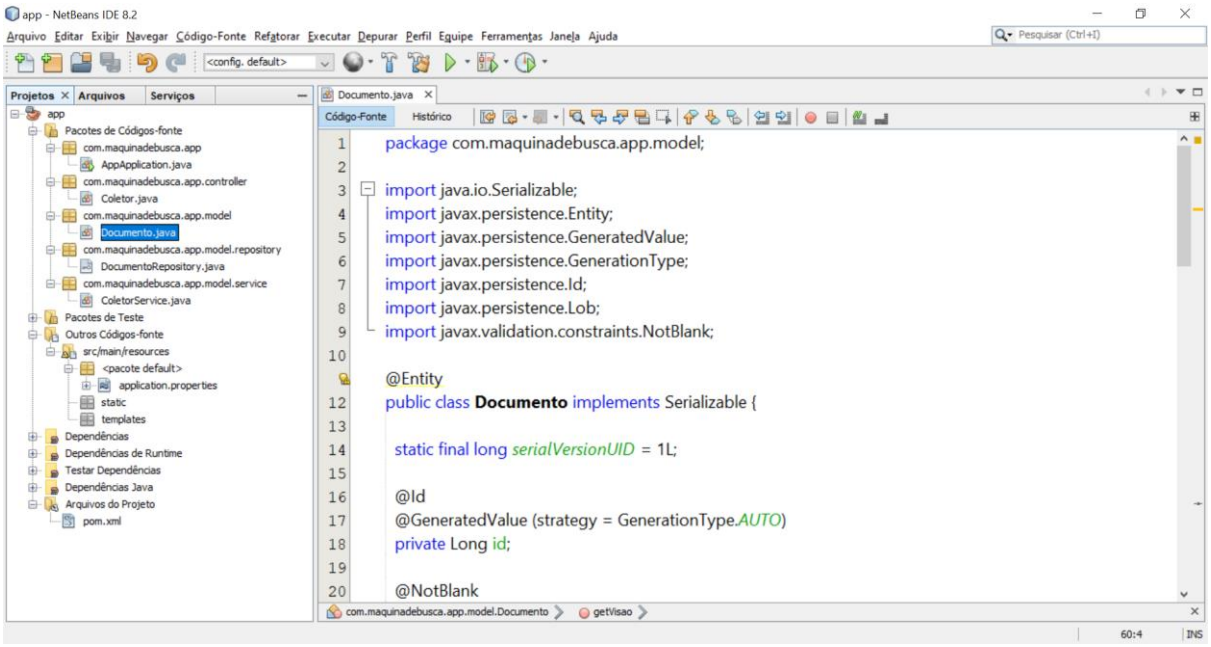


Altere a classe **Documento** no pacote "**com.maquinadebusca.app.model**", conforme apresentado abaixo.

```
package com.maquinadebusca.app.model;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.validation.constraints.NotBlank;

@Entity
public class Documento implements Serializable {
```

```
static final long serialVersionUID = 1L;

@Id
@GeneratedValue (strategy = GenerationType.AUTO)
private Long id;

@NotBlank
private String url;

@Lob
@NotBlank
private String texto;

@Lob
@NotBlank
private String visao;

public Documento () {
}

public Long getId () {
  return id;
}

public void setId (Long id) {
  this.id = id;
}

public String getUrl () {
  return url;
}

public void setUrl (String url) {
  this.url = url;
}

public String getTexto () {
  return texto;
}

public void setTexto (String texto) {
  this.texto = texto;
```

```
 }

 public String getVisao () {
  return visao;
 }

 public void setVisao (String visao) {
  this.visao = visao;
 }
}
```



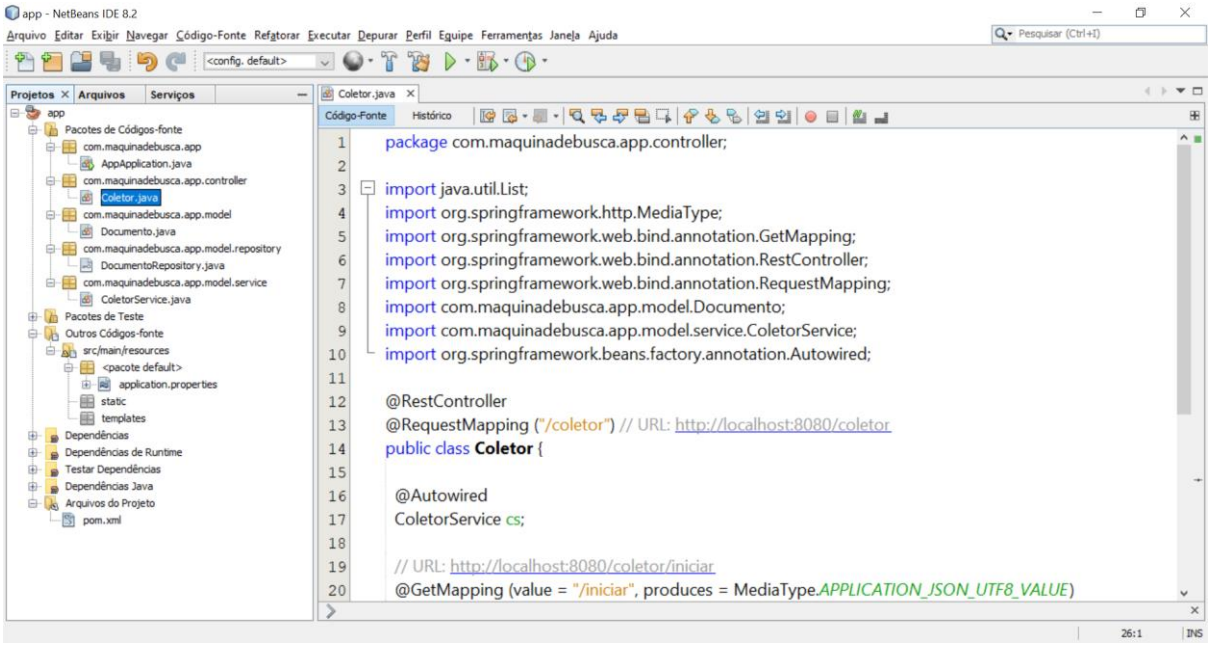
Altere a classe **Coletor** no pacote "**com.maquinadebusca.app.controller**", conforme apresentado abaixo.

```
package com.maquinadebusca.app.controller;

import java.util.List;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMapping;
import com.maquinadebusca.app.model.Documento;
import com.maquinadebusca.app.model.service.ColetorService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
```

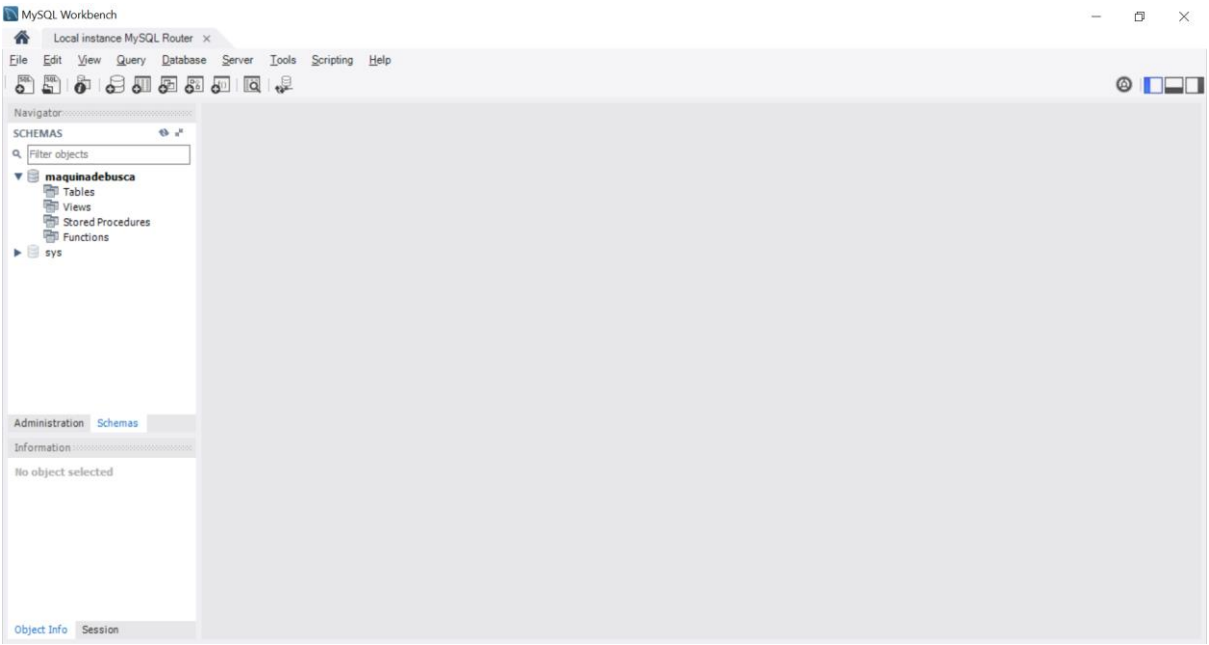
Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

```
@RestController
@RequestMapping ("/coletor") // URL: http://localhost:8080/coletor
public class Coletor {

 @Autowired
 ColetorService cs;

 // URL: http://localhost:8080/coletor/iniciar
 @GetMapping (value = "/iniciar", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
 public List<Documento> iniciar () {
  List<Documento> documentos = cs.executar ();
  return documentos;
 }

 // URL: http://localhost:8080/coletor/listar
 @GetMapping (value = "/listar", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
 public List<Documento> listar () {
  return cs.getDocumentos ();
 }

 // Request for: http://localhost:8080/coletor/listar/{id}
 @GetMapping (value = "/listar/{id}", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
 public Documento listar (@PathVariable (value = "id") long id) {
  return cs.getDocumento (id);
 }
}
```

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

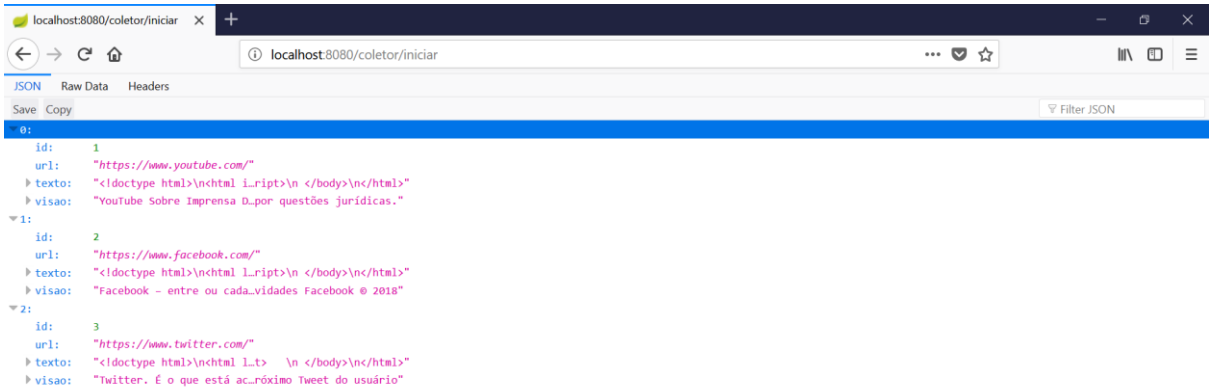


Inicie o SGBD MySQL através do MySQL Workbench. No MySQL Workbench, crie um esquema, banco, com nome **maquinadebusca**.

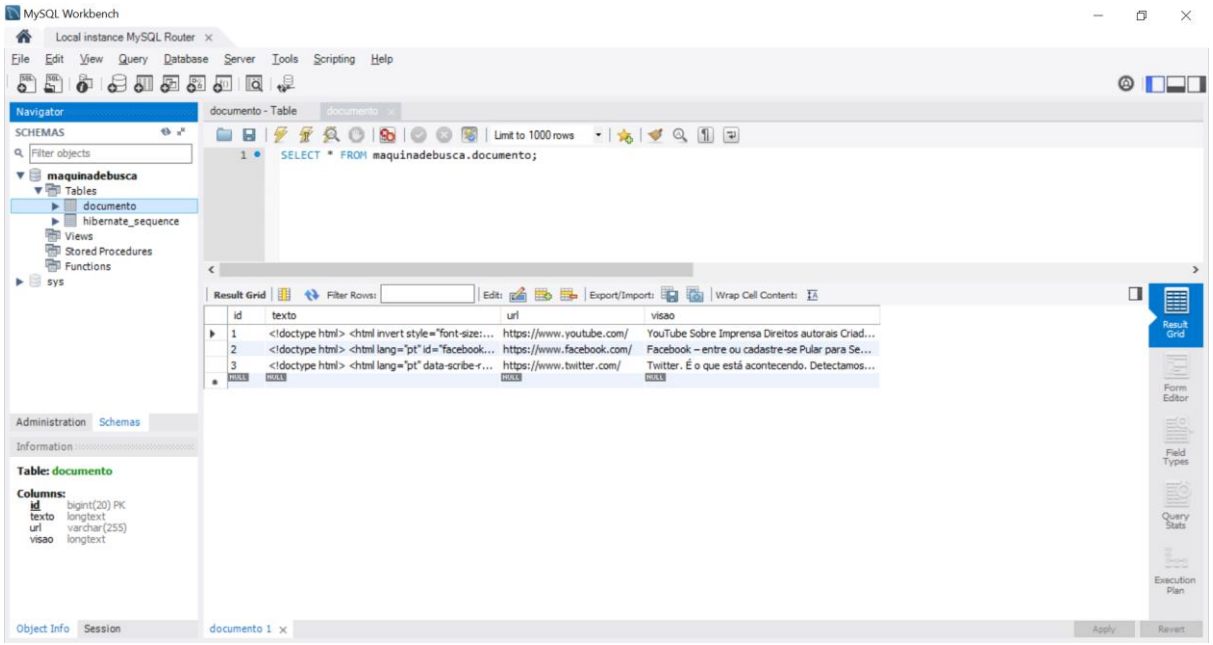


Execute o projeto. No navegador FireFox, informe a URL http://localhost:8080/coletor/iniciar. Observe o resultado da saída do servidor no navegador.

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

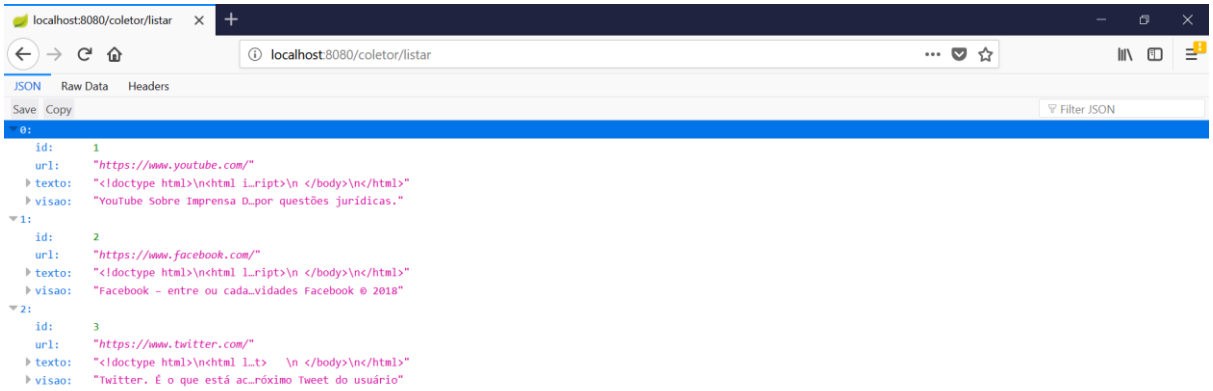


No MySQL Workbench, observe que a tabela Documento foi criada automaticamente. Além disso, as inserções na tabela Documento também foram realizadas automaticamente.

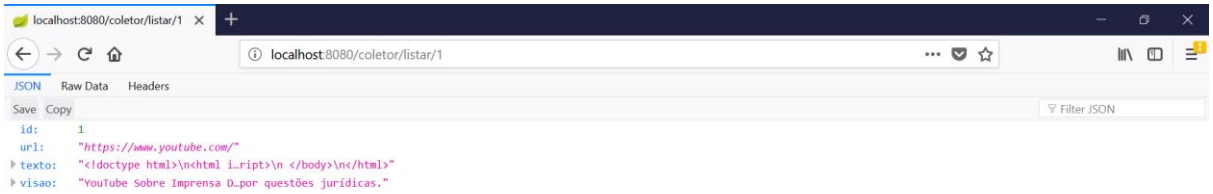


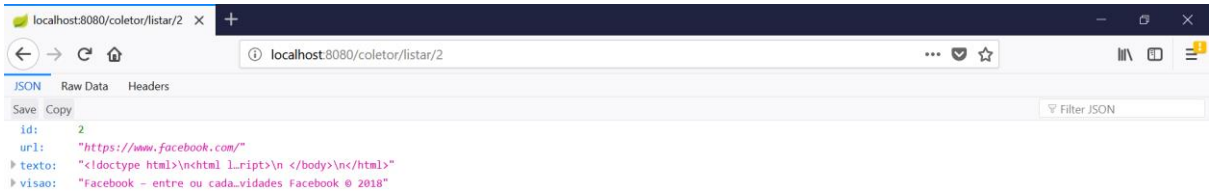
No navegador FireFox, informe a URL http://localhost:8080/coletor/listar. Observe o resultado da saída do servidor no navegador. Note que as consultas ao banco de dados foram realizadas automaticamente.

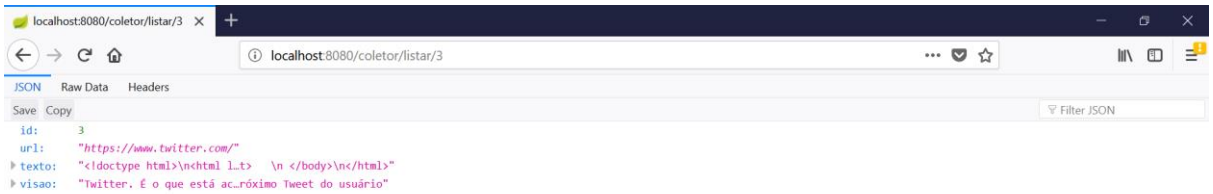
Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02



No navegador FireFox, informe a URL http://localhost:8080/coletor/listar/1. Observe o resultado da saída do servidor no navegador. Depois, altere o identificador “1” na URL por “2” e, depois, por “3”. Novamente, note que as consultas ao banco de dados foram realizadas automaticamente.

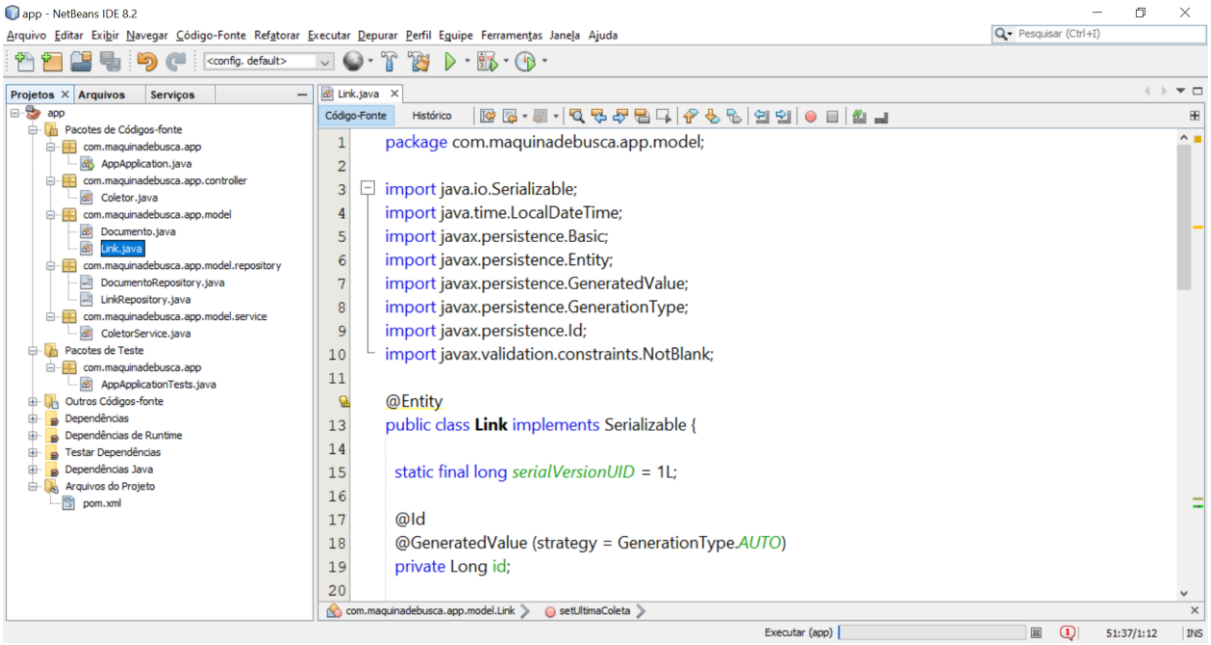






2) No pacote "**com.maquinadebusca.app.model**" crie a classe Link, conforme apresentado abaixo.

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

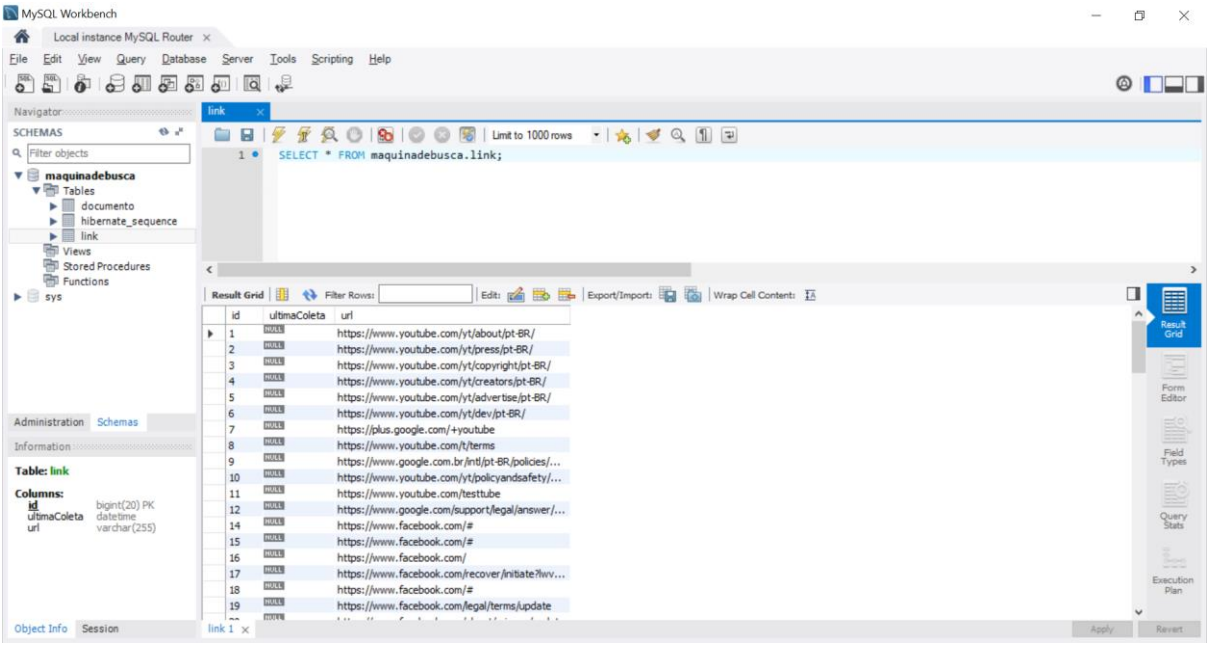


A classe Link deve ter os atributos:

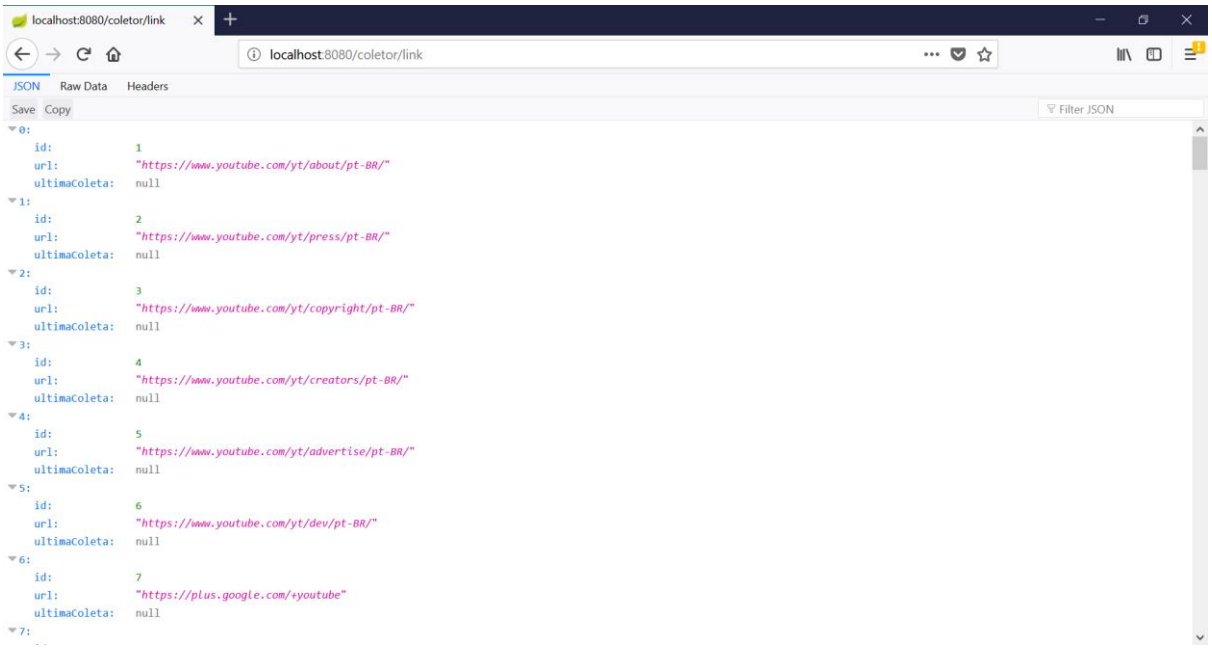
```
static final long serialVersionUID = 1L;

@Id
@GeneratedValue (strategy = GenerationType.AUTO)
private Long id;

@NotBlank
private String url;

@Basic
private LocalDateTime ultimaColeta;
```

Ao coletar um documento, as URLs presentes no documento devem ser armazenados em instâncias da classe Link e gravadas no banco de dados, como apresentado abaixo.

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02



O projeto deve permitir consultar todas as URLs coletadas e armazenadas no banco de dados, apresentando-as no navegador Web, como apresentado abaixo.



O projeto também deve permitir consultar uma URL específica do banco de dados, apresentando-a no navegador Web, como apresentado abaixo.

Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02