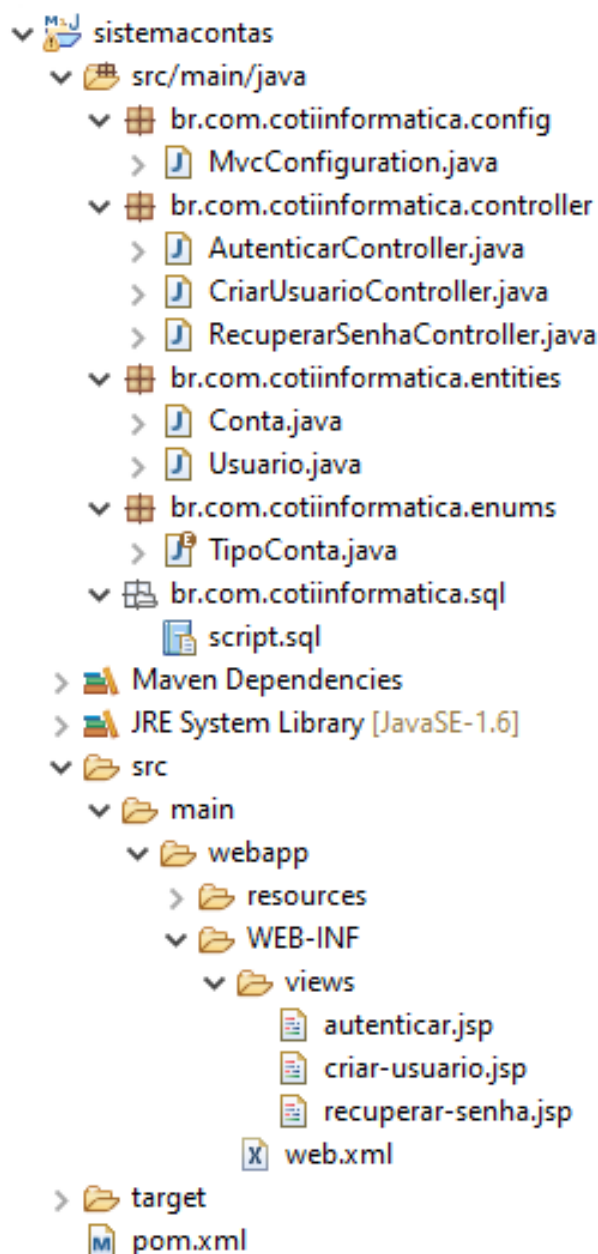


Spring Framework

Conjunto de tecnologias para desenvolvimento de aplicações web em Java. É composto por diversos módulos que incluem desde bibliotecas para acesso a banco de dados até recursos para desenvolvimento de aplicações para web como por exemplo projetos baseados no padrão **MVC** ou **API**.

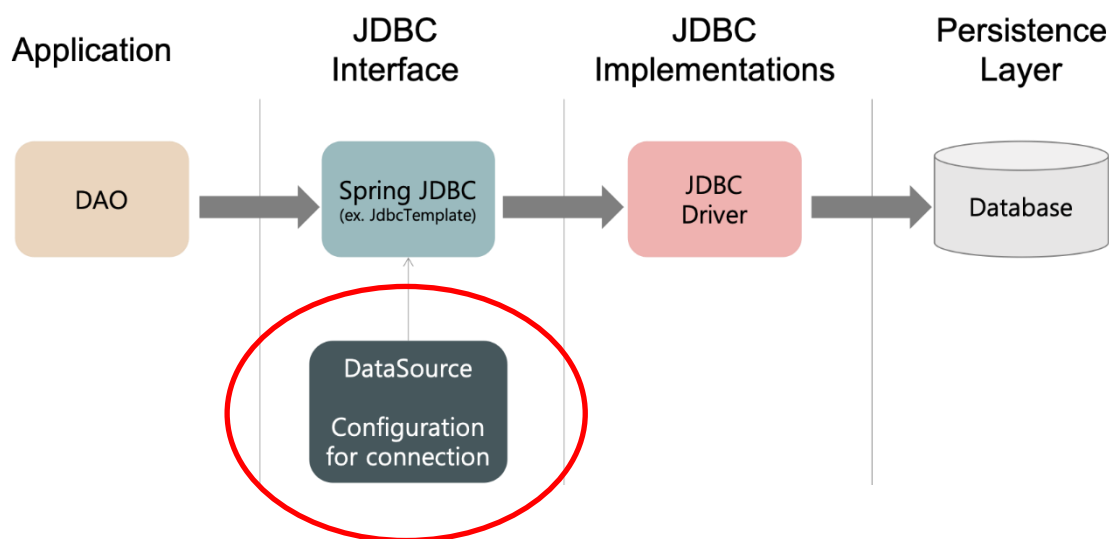


Configurando o Framework Spring para acessar uma base de dados

Precisamos mapear no projeto os parâmetros necessários para conectar nossa aplicação em um banco de dados, neste caso do PostGreSQL.

Data Source

Nome dado a configuração feita no projeto Spring para definir qual o banco de dados será acessado pela aplicação. Todo projeto criado em Spring irá configurar a conexão do seu banco de dados através de um **Data Source**



Onde configuramos o Data Source do projeto?

Todas as configurações do projeto Spring MVC ficam dentro da classe:

/config/**MvcConfiguration.java**

Esta é a principal classe de configuração do projeto Spring MVC.

@Bean

Annotation do Spring Framework utilizada para criarmos um método de configuração do projeto.

```
package br.com.cotiinformatica.config;
```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
```

```
@Configuration
```

```
@ComponentScan(basePackages="br.com.cotiinformatica")
```

```
@EnableWebMvc
```

```
public class MvcConfiguration extends WebMvcConfigurerAdapter{

    @Bean
    public ViewResolver getViewResolver(){
        InternalResourceViewResolver resolver
            = new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/views/");
        resolver.setSuffix(".jsp");
        return resolver;
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

        registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
    }
}
```

Precisamos adicionar uma configuração nesta classe para definir o **Data Source do projeto (conexão com banco de dados do projeto):**

```
package br.com.cotiinformatica.config;

import javax.sql.DataSource;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration
@ComponentScan(basePackages = "br.com.cotiinformatica")
@EnableWebMvc
public class MvcConfiguration extends WebMvcConfigurerAdapter {

    @Bean
    public ViewResolver getViewResolver() {
        InternalResourceViewResolver resolver
            = new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/views/");
        resolver.setSuffix(".jsp");
        return resolver;
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

        registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
    }
}
```

```

/*
 * Método para configurar o DATA SOURCE do projeto,
 * ou seja, a conexão com o
 * banco de dados utilizado pelo projeto
 */
@Bean
public DataSource getDataSource() {

    DriverManagerDataSource dataSource
        = new DriverManagerDataSource();

    dataSource.setDriverClassName("org.postgresql.Driver");

    dataSource.setUrl
        ("jdbc:postgresql://localhost:5432/bd_sistemacontas");

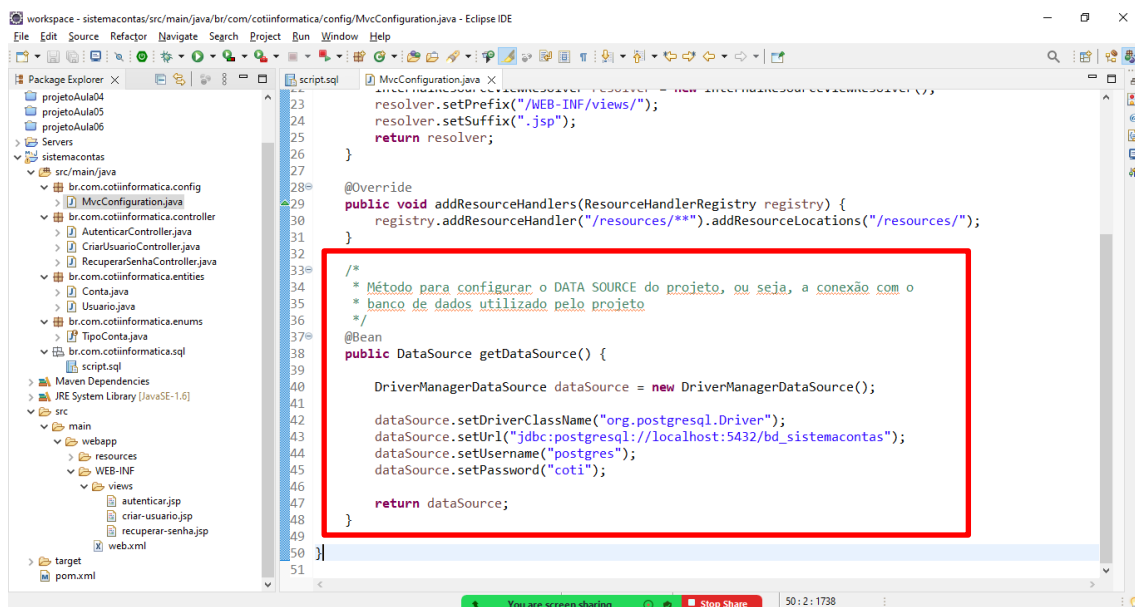
    dataSource.setUsername("postgres");

    dataSource.setPassword("coti");

    return dataSource;




}

```



Padrão Repository

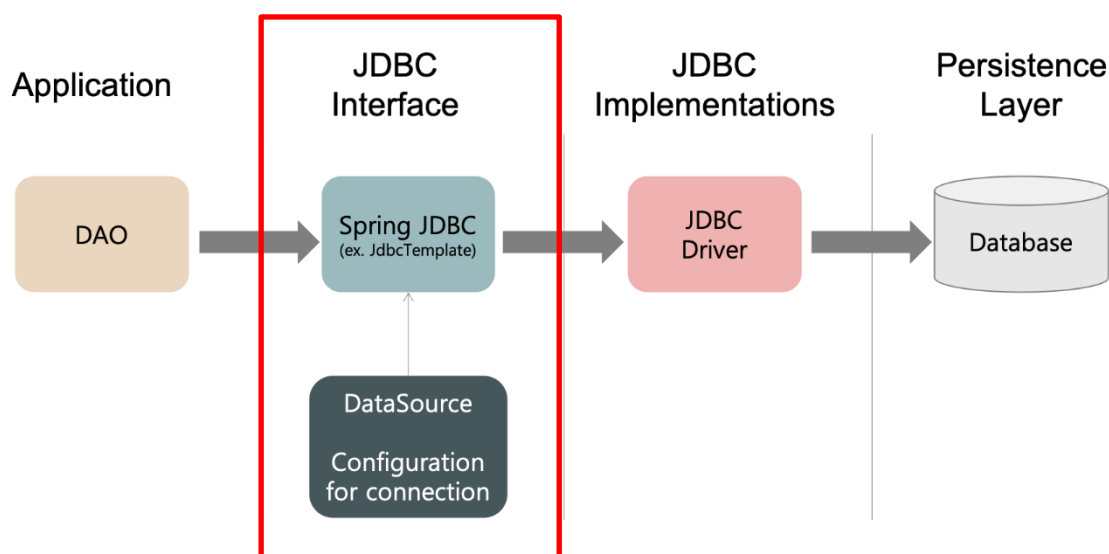
Consiste em criarmos classes chamadas de repositórios para realizar as operações necessárias em banco de dados para cada tabela/entidade do sistema (CRUD).

-  br.com.cotiinformatica.repositories
 -  ContaRepository.java
 -  UsuarioRepository.java

JDBC Template

É uma biblioteca do Spring Framework utilizada para simplificar a escrita de rotinas de banco de dados no projeto por meio do JDBC.

Para usarmos o JDBC Template precisamos configurar o **Data Source** (configuração do banco de dados) no projeto e fazer com que as classes de repositório possam receber esta configuração diretamente do Spring.



Criando as classes de repositório utilizando o **JdbcTemplate** do Spring e a conexão definida pelo **DataSource**:

```
package br.com.cotiinformatica.repositories;

import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

public class UsuarioRepository {

    // atributo
    private JdbcTemplate jdbcTemplate;

    // método construtor
    public UsuarioRepository(DataSource dataSource) {
        jdbcTemplate = new JdbcTemplate(dataSource);
    }
}

package br.com.cotiinformatica.repositories;

import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;
```

```
public class ContaRepository {  
  
    // atributo  
    private JdbcTemplate jdbcTemplate;  
  
    // construtor  
    public ContaRepository(DataSource dataSource) {  
        jdbcTemplate = new JdbcTemplate(dataSource);  
    }  
}
```

Se cada classe **Repository** do projeto precisa receber um **Data Source**, então precisamos configurar como este DataSource seja enviado para essas classes.

Voltando na classe de configuração do projeto:

/config/**MvcConfiguration.java**

```
package br.com.cotiinformatica.config;  
  
import javax.sql.DataSource;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import org.springframework.web.servlet.ViewResolver;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import  
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import  
org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;  
import org.springframework.web.servlet.view.InternalResourceViewResolver;  
  
import br.com.cotiinformatica.repositories.ContaRepository;  
import br.com.cotiinformatica.repositories.UsuarioRepository;  
  
@Configuration  
@ComponentScan(basePackages = "br.com.cotiinformatica")  
@EnableWebMvc  
public class MvcConfiguration extends WebMvcConfigurerAdapter {  
  
    @Bean  
    public ViewResolver getViewResolver() {  
        InternalResourceViewResolver resolver  
            = new InternalResourceViewResolver();  
        resolver.setPrefix("/WEB-INF/views/");  
        resolver.setSuffix(".jsp");  
        return resolver;  
    }  
}
```

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/resources/**")
        .addResourceLocations("/resources/");
}

/*
 * Método para configurar o DATA SOURCE do projeto,
 * ou seja, a conexão com o
 * banco de dados utilizado pelo projeto
 */
@Bean
public DataSource getDataSource() {

    DriverManagerDataSource dataSource
        = new DriverManagerDataSource();

    dataSource.setDriverClassName("org.postgresql.Driver");
    dataSource.setUrl
        ("jdbc:postgresql://localhost:5432/bd_sistemacontas");
    dataSource.setUsername("postgres");
    dataSource.setPassword("coti");

    return dataSource;
}

/*
 * Configuração para que a classe UsuarioRepository
 * possa receber o DataSource do projeto
 * (conexão com o banco de dados)
 */
@Bean
public UsuarioRepository getUsuarioRepository() {
    return new UsuarioRepository(getDataSource());
}

/*
 * Configuração para que a classe ContaRepository
 * possa receber o DataSource do projeto
 * (conexão com o banco de dados)
 */
@Bean
public ContaRepository getContaRepository() {
    return new ContaRepository(getDataSource());
}
}
```

Desenvolvendo o repositório de usuários:

/repositories/**UsuarioRepository.java**

```
package br.com.cotiinformatica.repositories;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

import javax.sql.DataSource;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import br.com.cotiinformatica.entities.Usuario;

public class UsuarioRepository {

    // atributo
    private JdbcTemplate jdbcTemplate;

    // método construtor
    public UsuarioRepository(DataSource dataSource) {
        jdbcTemplate = new JdbcTemplate(dataSource);
    }

    public void create(Usuario usuario) throws Exception {

        String query = "insert into usuario(nome, email, senha)
                        values(?,?, md5(?))";
        Object[] params = { usuario.getNome(), usuario.getEmail(),
                            usuario.getSenha() };

        jdbcTemplate.update(query, params);
    }

    public Usuario findByEmail(String email) throws Exception {

        String query = "select * from usuario where email = ?";
        Object[] params = { email };

        List<Usuario> lista = jdbcTemplate.query
            (query, params, new RowMapper<Usuario>() {

                @Override
                public Usuario mapRow(ResultSet rs, int rowNum)
                    throws SQLException {

                    Usuario usuario = new Usuario();

                    usuario.setIdUsuario(rs.getInt("idusuario"));
                    usuario.setNome(rs.getString("nome"));
                }
            });
    }
}
```



```

        usuario.setEmail(rs.getString("email"));
        usuario.setSenha(rs.getString("senha"));

        return usuario;
    }
});

if(lista.size() == 1)
//verificando se 1 usuário foi encontrado
    return lista.get(0);
//retornando os dados do usuário encontrado
else
    return null; //retornando vazio
}
}

```

Voltando para o MVC – Model View e Controller

Revisando os conceitos do padrão MVC.

O padrão MVC é utilizado para desenvolvimento de aplicações web em Java (através do Spring MVC) mas também em outras plataformas de desenvolvimento.

Resumindo...

Views

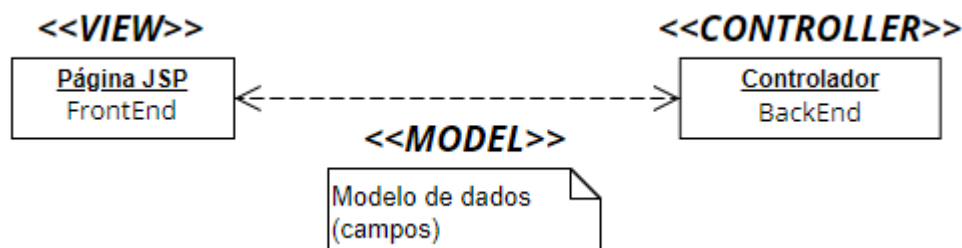
São as páginas utilizadas para definir a interface web do sistema (FrontEnd). No Spring MVC, estas páginas são arquivos de extensão **.jsp**

Controllers

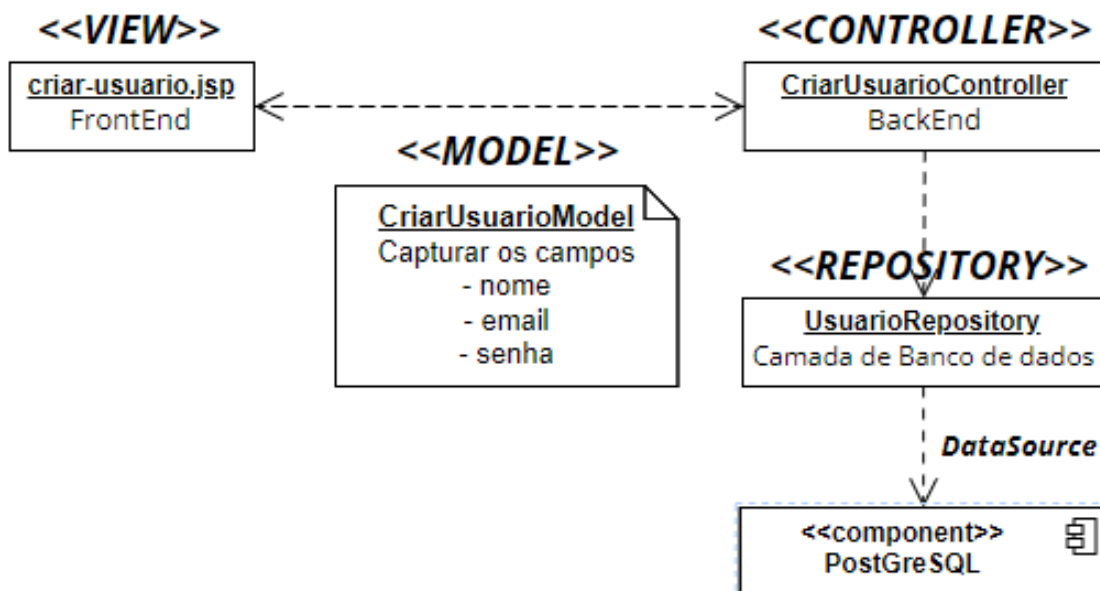
São as classes utilizadas para mapear as rotas de navegação para as Views e capturar as requisições enviadas das páginas para o controlador.

Models

São classes simples utilizadas para capturar os dados de uma página e enviar estes dados para um controlador e vice versa.



Criando uma classe de modelo de dados para capturar os campos do formulário de cadastro de usuário:



Vamos criar a classe que será utilizada como modelo de dados para capturar os campos da página de cadastro de usuário:

/models/**CriarUsuarioModel.java**

```
package br.com.cotiinformatica.models;
```

```
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```
@Setter
```

```
@Getter
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@ToString
```

```
public class CriarUsuarioModel {
```

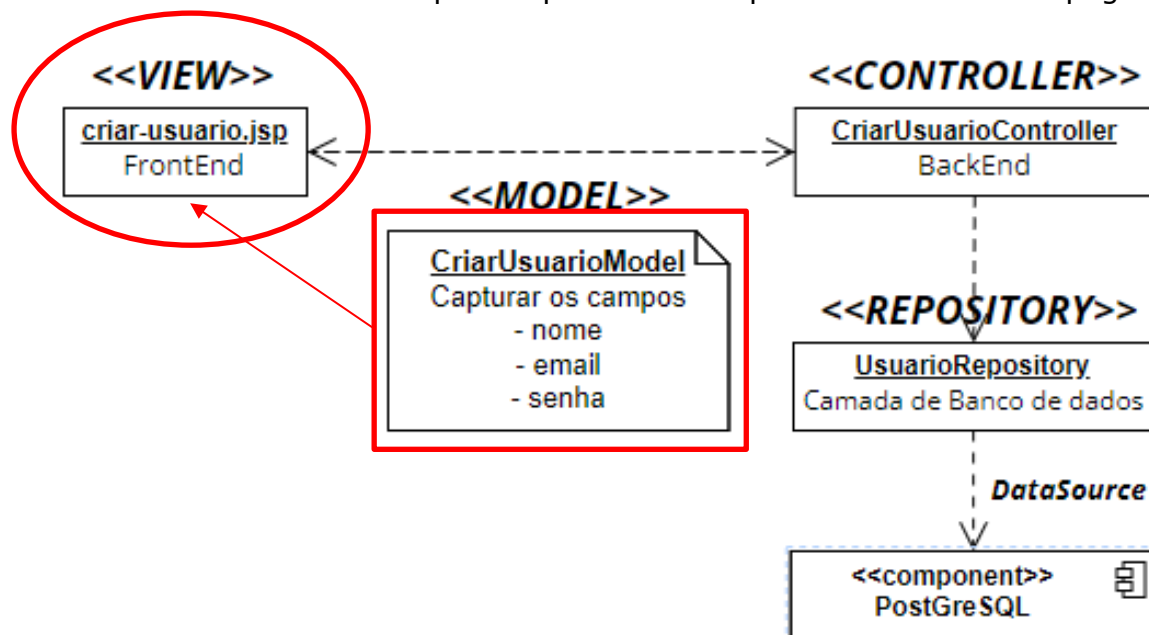
```
    private String nome;
```

```
    private String email;
```

```
    private String senha;
```

```
}
```

Precisamos associar esta classe Model à página de criação de usuário, pois esta classe será utilizada para capturar os campos do formulário da página:



Voltando no controlador:

/controllers/**CriarUsuarioController.java**

```
package br.com.cotiinformatica.controller;
```

```
import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.servlet.ModelAndView;
```

```
import br.com.cotiinformatica.models.CriarUsuarioModel;
```

```
@Controller
```

```
public class CriarUsuarioController {
```

```
    @RequestMapping(value = "/criar-usuario") // ROTA (navegação)
```

```
    public ModelAndView criarUsuario() {
```

```
        // WEB-INF/views/criar-usuario.jsp
```

```
        ModelAndView modelAndView = new ModelAndView("criar-usuario");
```

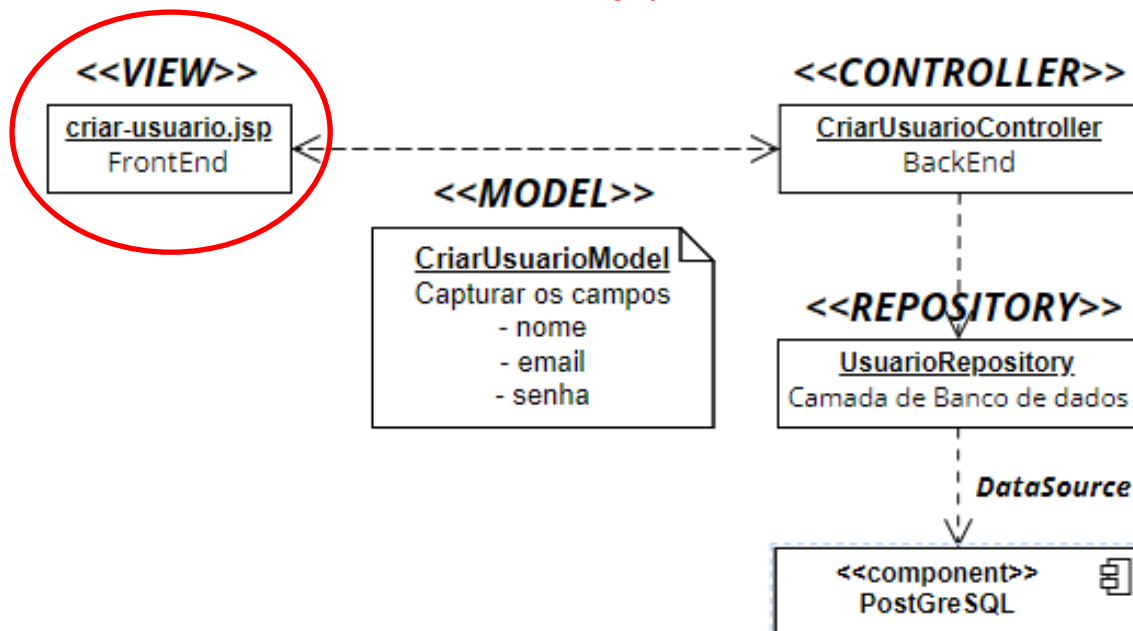
```
        modelAndView.addObject("model", new CriarUsuarioModel());
```

```
        return modelAndView;
```

```
    }
```

```
}
```

Vamos para a página JSP e fazer a captura dos campos:
/WEB-INF/views/criar-usuario.jsp



```
workspace - sistemacontas/src/main/webapp/WEB-INF/views/criar-usuario.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
criar-usuario.jsp
37<=
38
39<=
40<div class="mb-2">
41<label>Entre com o seu nome:</label>
42<form:input path="model.nome" type="text" name="nome" class="form-control"/>
43</div>
44<=
45<div class="mb-2">
46<label>Entre com o seu email:</label>
47<form:input path="model.email" type="text" name="email" class="form-control"/>
48</div>
49<=
50<div class="mb-2">
51<label>Entre com a sua senha:</label>
52<form:input path="model.senha" type="password" name="senha" id="senha" class="form-control"/>
53</div>
54<=
55<div class="mb-2">
56<label>Confirme a sua senha:</label>
57<input type="password" name="senhaConfirmacao" class="form-control"/>
58</div>
59<=
60<div class="mb-2 d-grid">
61<input type="submit" value="Cadastrar" class="btn btn-primary"/>
62</div>
63<=
64<div class="mb-2 d-grid">
65<a href="/sistemacontas/" class="btn btn-light">
    Já possui cadastro? <strong>Acesse aqui</strong>
66</div>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
```

```
<!-- Biblioteca de Tags do Spring MVC -->
<%@ taglib uri="http://www.springframework.org/tags/form"
    prefix="form" %>
```

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="ISO-8859-1">
<title>Insert title here</title>

<!-- CDN da folha de estilos CSS do bootstrap -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap
@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" />

<style>
    label.error { color: #df4759; }
    input.error { border: 2px solid #df4759; }
</style>

</head>
<body>

<div class="row">
    <div class="col-md-4 offset-md-4">
        <div class="card mt-3">
            <div class="card-body">

                <div class="text-center">
                    <h2>Sistema de Contas</h2>
                    <h5>Cadastro de usuários</h5>
                </div>

                <hr/>

                <form id="form_criarusuario"
                    action="cadastrar-usuario"
                    method="post">

                    <div class="mb-2">
                        <label>Entre com
                        o seu nome:</label>
                        <form:input path="model.nome"
                            type="text" name="nome"
                            class="form-control"/>
                    </div>

                    <div class="mb-2">
                        <label>Entre com o seu
                        email:</label>
                        <form:input
                            path="model.email"
                            type="text" name="email"
                            class="form-control"/>
                    </div>

                    <div class="mb-2">
                        <label>
                        Entre com a sua senha</label>
                        <form:input
```

```

        path="model.senha"
        type="password" name="senha"
        id="senha" class="form-
        control"/>
    </div>

    <div class="mb-2">
        <label>Confirme a sua
        senha</label>
        <input type="password"
        name="senhaConfirmacao"
        class="form-control"/>
    </div>

    <div class="mb-2 d-grid">
        <input type="submit"
        value="Cadastrar"
        class="btn btn-
        primary"/>
    </div>

    <div class="mb-2 d-grid">
        <a href="/sistemacontas/"
        class="btn btn-light">
        Já possui cadastro?
        <strong>Acesse aqui!</strong>
        </a>
    </div>
</form>

</div>
</div>
</div>
</div>

<!-- CDN do arquivo javascript do bootstrap -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap
@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

<!-- CDN do arquivo javascript do JQuery -->
<script src="https://code.jquery.com
/jquery-3.6.3.min.js"></script>

<!-- CDN dos arquivos da biblioteca JQuery Validation -->
<script src="https://cdnjs.cloudflare.com/ajax/libs
/jquery-validate/1.19.5/jquery.validate.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs
/jquery-validate/1.19.5
/additional-methods.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs
/jquery-validate/1.19.5/localization
/messages_pt_BR.min.js"></script>

```

```

<script>
    $(document).ready(function() {

        $("#form_criarusuario").validate({
            rules: {
                'nome' : { required: true, minlength: 8,
                           maxlength: 150 },
                'email' : { required: true,
                           email : true },
                'senha' : { required: true,
                           minlength: 8,
                           maxlength: 20 },
                'senhaConfirmacao' : { required: true,
                                       equalTo: '#senha' }
            }
        });

    })
</script>

</body>
</html>

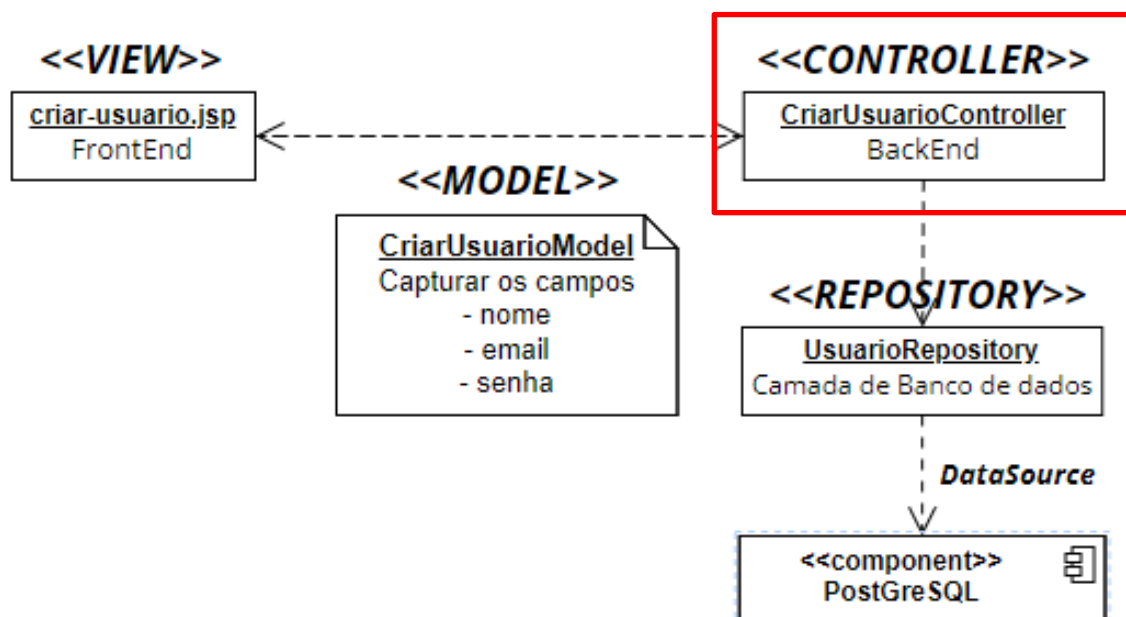
```

No controlador, vamos criar o método para capturar o envio do formulário (SUBMIT) e então cadastrar o usuário no banco de dados.

```

<form id="form_criarusuario"
      action="cadastrar-usuario" ←
      method="post">

```



/controllers/**CriarUsuarioController.java**

```
package br.com.cotiinformatica.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import br.com.cotiinformatica.models.CriarUsuarioModel;

@Controller
public class CriarUsuarioController {

    /**
     * Método para abrir a página
     */
    @RequestMapping(value = "/criar-usuario") // ROTA (navegação)
    public ModelAndView criarUsuario() {

        // WEB-INF/views/criar-usuario.jsp
        ModelAndView modelAndView = new ModelAndView
            ("criar-usuario");
        modelAndView.addObject("model", new CriarUsuarioModel());

        return modelAndView;
    }

    /**
     * Método para receber o SUBMIT POST do formulário
     */
    @RequestMapping(value = "/cadastrar-usuario",
        method = RequestMethod.POST)
    public ModelAndView cadastrarUsuario(CriarUsuarioModel model) {

        ModelAndView modelAndView = new ModelAndView
            ("criar-usuario");

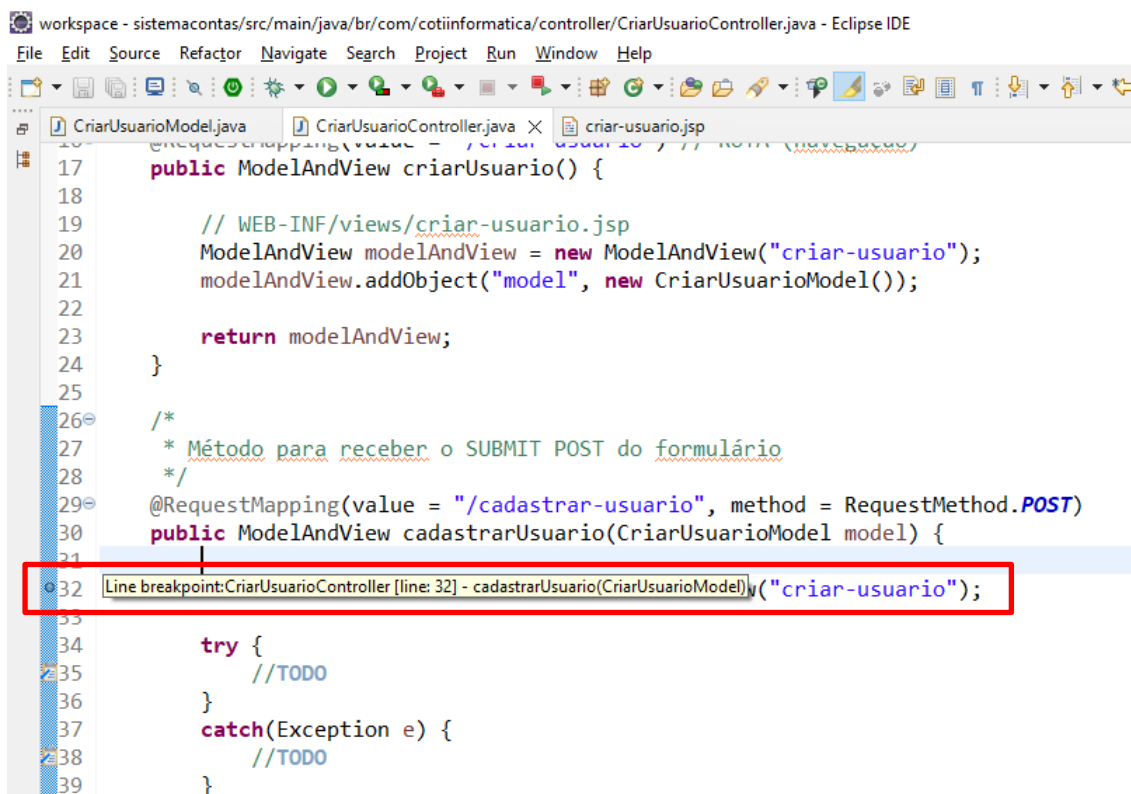
        try {
            //TODO
        }
        catch(Exception e) {
            //TODO
        }

        modelAndView.addObject("model", model);
        return modelAndView;
    }
}
```


Debugando o código fonte do projeto:

Debugando no Eclipse:

**** Primeiro, precisamos criar um BREAKPOINT**
(Local onde irá começar o Debug)

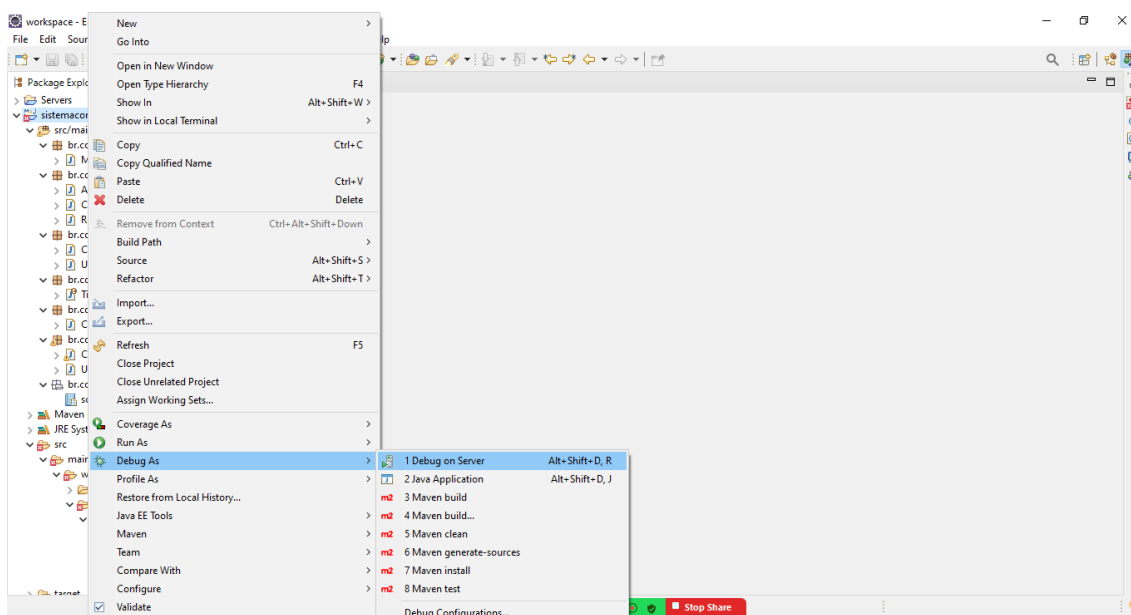


```
workspace - sistemacontas/src/main/java/br/com/cotiinformatica/controller/CriarUsuarioController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

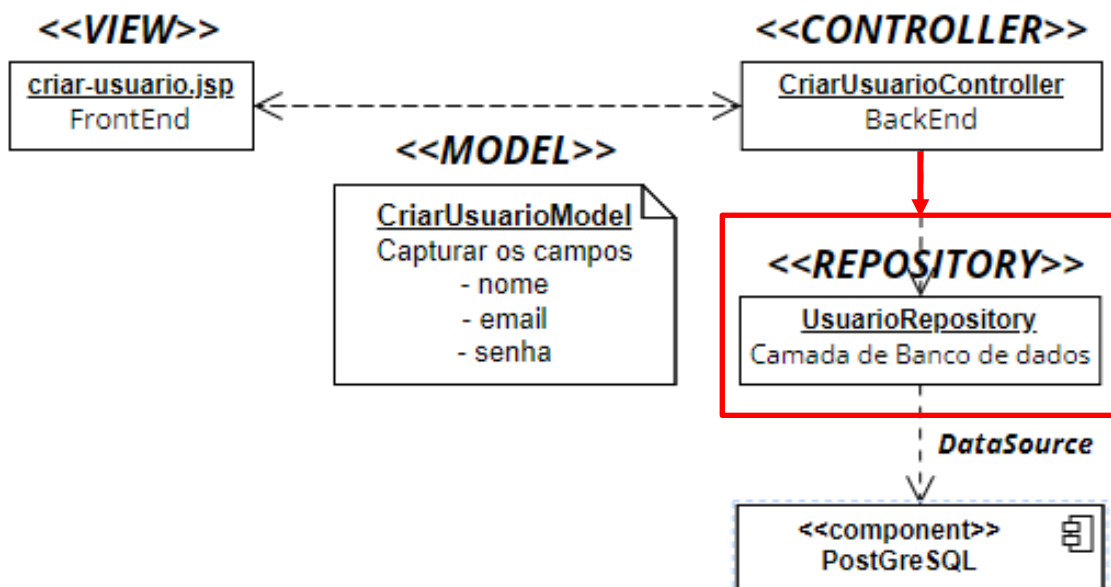
CriarUsuarioModel.java CriarUsuarioController.java X criar-usuario.jsp

17 public ModelAndView criarUsuario() {
18
19     // WEB-INF/views/criar-usuario.jsp
20     ModelAndView modelAndView = new ModelAndView("criar-usuario");
21     modelAndView.addObject("model", new CriarUsuarioModel());
22
23     return modelAndView;
24 }
25
26 /*
27  * Método para receber o SUBMIT POST do formulário
28  */
29 @RequestMapping(value = "/cadastrar-usuario", method = RequestMethod.POST)
30 public ModelAndView cadastrarUsuario(CriarUsuarioModel model) {
31
32 Line breakpoint: CriarUsuarioController [line: 32] - cadastrarUsuario(CriarUsuarioModel) ("criar-usuario");
33
34     try {
35         //TODO
36     }
37     catch (Exception e) {
38         //TODO
39     }
}
```

Executando o projeto em modo Debug
/DEBUG AS / DEBUG ON SERVER



Para que possamos cadastrar o usuário, o controlador vai precisar acessar o repositório:



@Autowired

Annotation utilizada pelo Spring para inicializar objetos de forma automática, utilizando um recurso nativo do framework chamado de **injeção de dependência**.

```
package br.com.cotiinformatica.controller;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
```

```
import br.com.cotiinformatica.models.CriarUsuarioModel;
import br.com.cotiinformatica.repositories.UsuarioRepository;
```

```
@Controller
```

```
public class CriarUsuarioController {
```

```
@Autowired
```

```
private UsuarioRepository usuarioRepository;
```

```
(...)
```

```
}
```

Implementando o fluxo para cadastro do usuário:

```
package br.com.cotiinformatica.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import br.com.cotiinformatica.entities.Usuario;
import br.com.cotiinformatica.models.CriarUsuarioModel;
import br.com.cotiinformatica.repositories.UsuarioRepository;

@Controller
public class CriarUsuarioController {

    @Autowired //injeção de dependência (inicialização automática)
    private UsuarioRepository usuarioRepository;

    /**
     * Método para abrir a página
     */
    @RequestMapping(value = "/criar-usuario") // ROTA (navegação)
    public ModelAndView criarUsuario() {

        // WEB-INF/views/criar-usuario.jsp
        ModelAndView modelAndView = new ModelAndView("criar-usuario");
        modelAndView.addObject("model", new CriarUsuarioModel());

        return modelAndView;
    }

    /**
     * Método para receber o SUBMIT POST do formulário
     */
    @RequestMapping(value = "/cadastrar-usuario", method = RequestMethod.POST)
    public ModelAndView cadastrarUsuario(CriarUsuarioModel model) {

        ModelAndView modelAndView = new ModelAndView("criar-usuario");

        try {

            //verificar se já existe um usuário cadastrado no banco de dados
            //com o email informado na página
            if(usuarioRepository.findByEmail(model.getEmail()) != null) {
                modelAndView.addObject("erro_email",
                    "O email informado já está cadastrado, tente outro.");
            }
            else {

                Usuario usuario = new Usuario();

                usuario.setNome(model.getNome());
                usuario.setEmail(model.getEmail());
                usuario.setSenha(model.getSenha());

                //gravando o usuário no banco de dados
                usuarioRepository.create(usuario);
            }
        }
    }
}
```

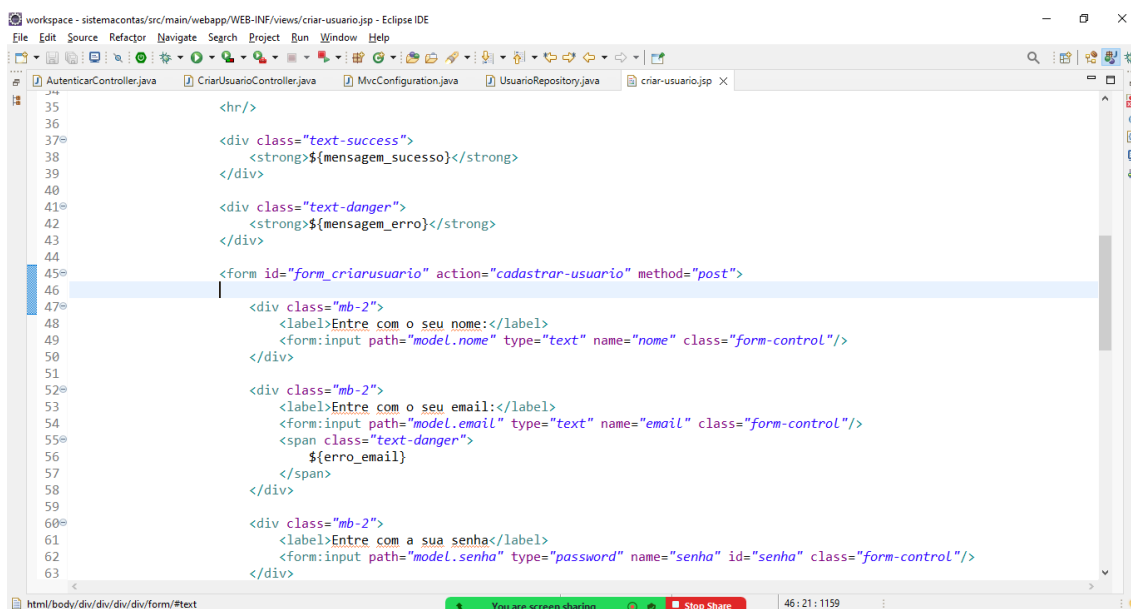
```

        modelAndView.addObject("mensagem_sucesso",
            "Parabéns! Sua Conta de usuário foi criada com sucesso.");
        model = new CriarUsuarioModel();
    }
}
catch(Exception e) {
    modelAndView.addObject("mensagem_erro", e.getMessage());
}

modelAndView.addObject("model", model);
return modelAndView;
}
}

```

Exibindo as mensagens na página:



```

35 <hr/>
36
37 <div class="text-success">
38 <strong>${mensagem_sucesso}</strong>
39 </div>
40
41 <div class="text-danger">
42 <strong>${mensagem_erro}</strong>
43 </div>
44
45 <form id="form_criarusuario" action="cadastrar-usuario" method="post">
46
47 <div class="mb-2">
48 <label>Entre com o seu nome:</label>
49 <form:input path="model.nome" type="text" name="nome" class="form-control"/>
50 </div>
51
52 <div class="mb-2">
53 <label>Entre com o seu email:</label>
54 <form:input path="model.email" type="text" name="email" class="form-control"/>
55 <span class="text-danger">
56 <strong>${erro_email}</strong>
57 </span>
58 </div>
59
60 <div class="mb-2">
61 <label>Entre com a sua senha:</label>
62 <form:input path="model.senha" type="password" name="senha" id="senha" class="form-control"/>
63 </div>

```

```

<div class="text-success">
    <strong>${mensagem_sucesso}</strong>
</div>

<div class="text-danger">
    <strong>${mensagem_erro}</strong>
</div>

<form id="form_criarusuario" action="cadastrar-usuario" method="post">

    <div class="mb-2">
        <label>Entre com o seu nome:</label>
        <form:input path="model.nome" type="text" name="nome"
            class="form-control"/>
    </div>

    <div class="mb-2">
        <label>Entre com o seu email:</label>
        <form:input path="model.email" type="text"

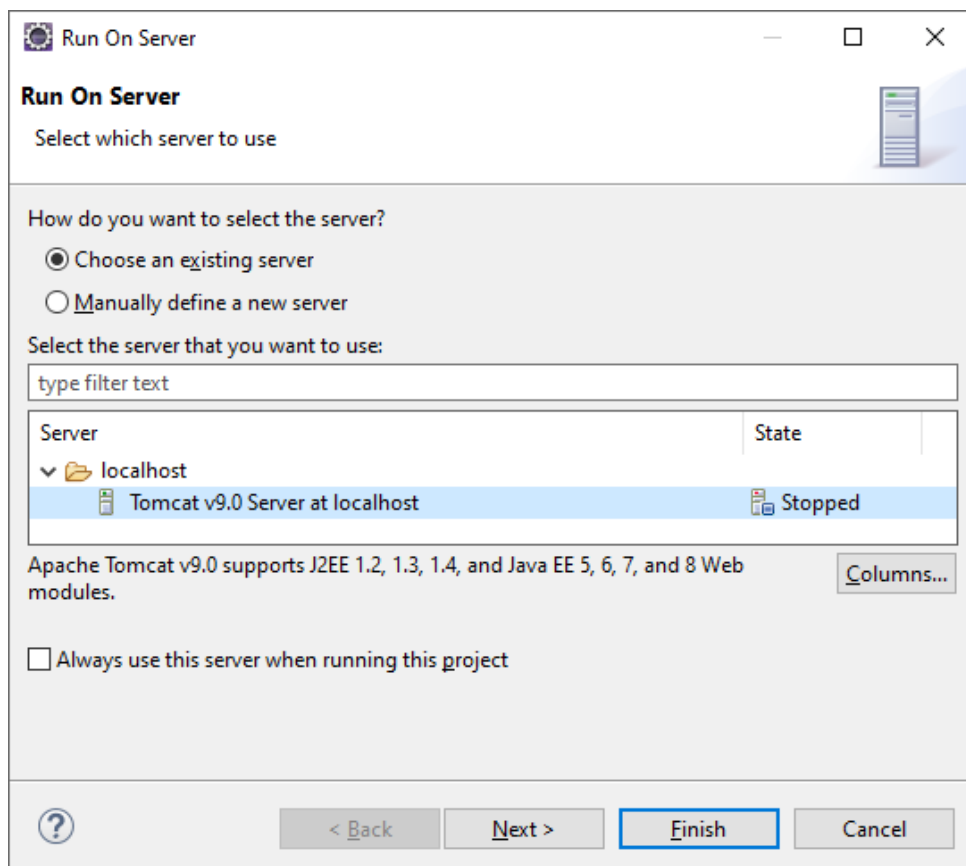
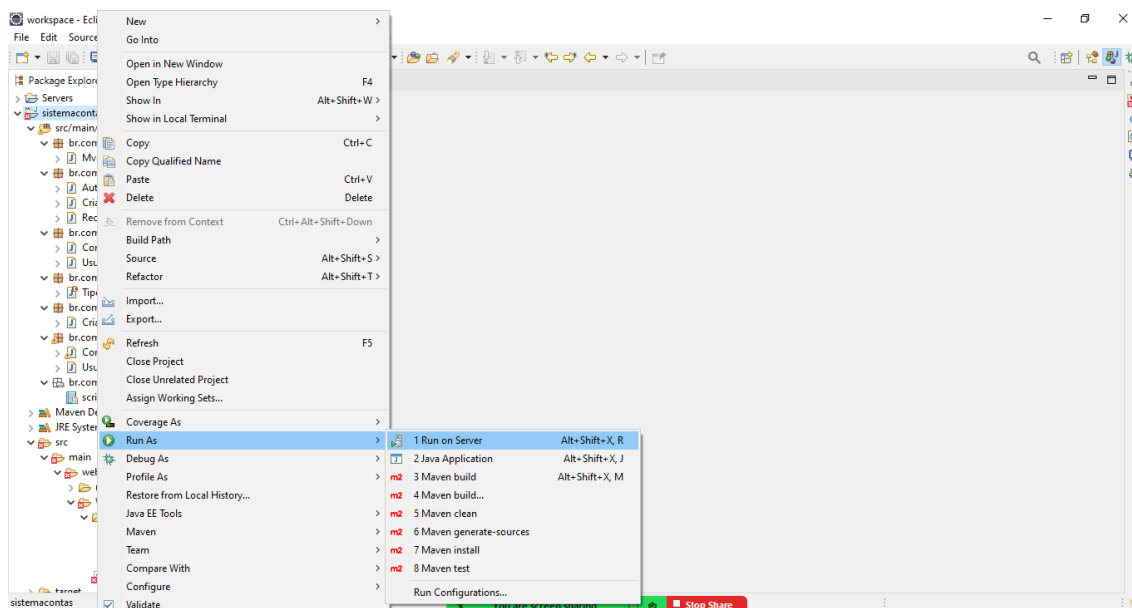
```

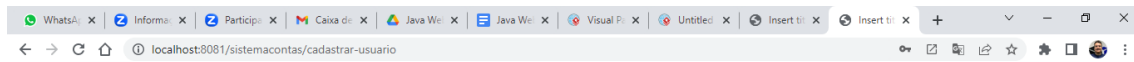
```

        name="email" class="form-control"/>
    <span class="text-danger">
        ${erro_email}
    </span>
</div>

```

Executando:





Sistema de Contas

Cadastro de usuários

Parabéns! Sua Conta de usuário foi criada com sucesso.

Entre com o seu nome:

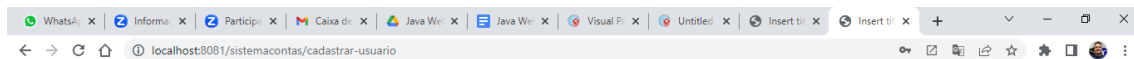
Entre com o seu email:
sergio.coti@gmail.com

Entre com a sua senha

Confirme a sua senha

Cadastrar

Já possui cadastro? [Acesse aqui!](#)



Sistema de Contas

Cadastro de usuários

Entre com o seu nome:
Sergio da Silva Mendes

Entre com o seu email:
sergio.coti@gmail.com

O email informado já está cadastrado, tente outro.

Entre com a sua senha

Confirme a sua senha

Cadastrar

Já possui cadastro? [Acesse aqui!](#)



No banco de dados:

pgAdmin 4

pgAdmin File Object Tools Help

Browser

- Servers (2)
 - PostgreSQL 14
 - Databases (3)
 - bd_aula03
 - bd_sistemacontas
 - casts
 - catalogs
 - event triggers
 - extensions
 - foreign data wrappers
 - languages
 - publications
 - schemas
 - subscriptions
 - postgres
 - login/group roles
 - tablespaces
 - PostgreSQL 15

Dashboard Properties SQL Statistics Dependencies bd_sistemacontas/postgres@PostgreSQL 14*

bd_sistemacontas/postgres@PostgreSQL 14

No limit

Query Query History

```
1 select * from usuario;
```

Data output Messages Notifications

| | idusuario [PK] integer | nome character varying (150) | email character varying (100) | senha character varying (50) |
|---|------------------------|------------------------------|-------------------------------|----------------------------------|
| 1 | 1 | Sergio da Silva Mendes | sergio.coti@gmail.com | 4c79273eed3d095e55d1224f6524ae92 |

Publicando o trabalho da aula para o GITHUB:

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
```

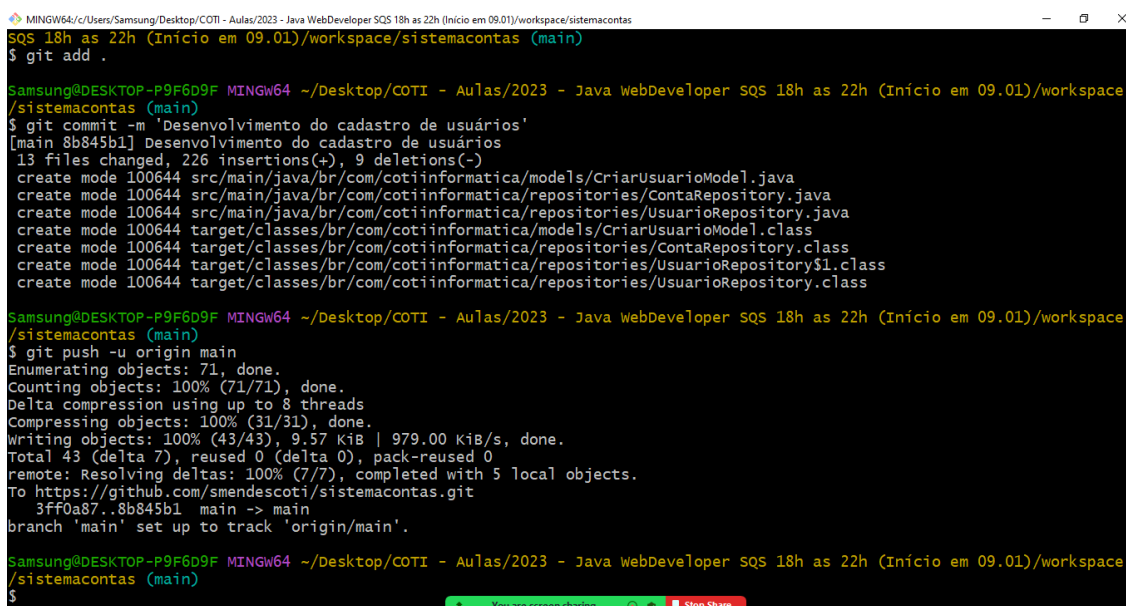
```
$ git add .
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
```

```
$ git commit -m 'Desenvolvimento do cadastro de usuários'
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
```

```
$ git push -u origin main
```



```
MINGW64/c/Users/Samsung/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
$ git add .

Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
$ git commit -m 'Desenvolvimento do cadastro de usuários'
[main 8b845b1] Desenvolvimento do cadastro de usuários
13 files changed, 226 insertions(+), 9 deletions(-)
create mode 100644 src/main/java/br/com/cotiinformatica/models/CriarUsuarioModel.java
create mode 100644 src/main/java/br/com/cotiinformatica/repositories/ContaRepository.java
create mode 100644 src/main/java/br/com/cotiinformatica/repositories/UsuarioRepository.java
create mode 100644 target/classes/br/com/cotiinformatica/models/CriarUsuarioModel.class
create mode 100644 target/classes/br/com/cotiinformatica/repositories/ContaRepository.class
create mode 100644 target/classes/br/com/cotiinformatica/repositories/UsuarioRepository.class

Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
$ git push -u origin main
Enumerating objects: 71, done.
Counting objects: 100% (71/71), done.
Delta compression using up to 8 threads
Compressing objects: 100% (31/31), done.
Writing objects: 100% (43/43), 9.57 KiB | 979.00 KiB/s, done.
Total 43 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 5 local objects.
To https://github.com/smendescoti/sistemacontas.git
3ff0a87..8b845b1 main -> main
branch 'main' set up to track 'origin/main'.

Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/sistemacontas (main)
$
```