**Estrutura do projeto:**

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Voltando para a camada de repositório do projeto:

/repositories/UsuarioRepository.java

* Adicionando métodos para:
  + Atualizar a senha do usuário
  + Consultar um usuário através do email e da senha

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 void update(Integer idUsuario, String novaSenha) throws Exception {**

**String query = "update usuario set senha = md5(?)**

**where idusuario = ?";**

**Object[] params = { novaSenha, idUsuario };**

**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

}

**public Usuario findByEmailAndSenha(String email, String senha)**

**throws Exception {**

**String query = "select \* from usuario where email = ?**

**and senha = md5(?)";**

**Object[] params = { email, senha };**

**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**

**}**

}

Próxima tarefa:

**Desenvolver a autenticação do usuário**

Segundo o padrão View, Model e Controller precisamos criar uma classe capaz de capturar os campos da página de autenticação e permitir que esses dados sejam enviados para o controlador.

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Neste caso, precisamos criar uma classe Model para capturar os campos do formulário da página de autenticação.

**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** AutenticarModel {

**private** String email;

**private** String senha;

}

Em seguida, precisamos definir que esta classe Model será utilizada para fazer a captura dos campos da página **autenticar.jsp**

Para isso, vamos até o controlador da página:

/controller/AutenticarController.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.AutenticarModel;

@Controller

public class AutenticarController {

@RequestMapping(value = "/") //Raiz do projeto

public ModelAndView autenticar() {

//WEB-INF/views/autenticar.jsp

ModelAndView modelAndView = new ModelAndView("autenticar");

**modelAndView.addObject("model", new AutenticarModel());**

return modelAndView;

}

}

Em seguida, vamos até a página **autenticar.jsp** e utilizar a biblioteca **spring/forms** para capturar os campos do formulário com o uso da classe de modelo criada no controlador:

/WEB-INF/views/**autenticar.jsp**

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

**<%@taglib**

**uri=**[***http://www.springframework.org/tags/form***](http://www.springframework.org/tags/form)

**prefix=*"form"***

**%>**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

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

pageEncoding=*"ISO-8859-1"*%>

<%@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-5"*>

<div class=*"card-body"*>

<div class=*"text-center"*>

<h2>Sistema de Contas</h2>

<h5>Autenticação de usuários</h5>

</div>

<hr/>

<form id=*"form\_autenticar"*

action=*"autenticar-usuario"* method=*"post"*>

<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"* class=*"form-control"*/>

<div class=*"text-end"*>

<a

href=*"/sistemacontas/recuperar-senha"*>

Esqueci minha senha?

</a>

</div>

</div>

<div class=*"mb-2 d-grid"*>

<input type=*"submit"*

value=*"Entrar"*

class=*"btn btn-primary"*/>

</div>

<div class=*"mb-2 d-grid"*>

<a href=*"/sistemacontas/criar-*

*usuario"* class=*"btn btn-light"*>

Não possui cadastro?

<strong>Clique 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\_autenticar").validate({

rules: {

'email' : { required: **true**, email : **true** },

'senha' : { required: **true**, minlength: 8,

maxlength: 20 }

}

});

})

</script>

</body>

</html>

Agora, vamos criar no controlador o método para

receber o **SUBMIT POST** do formulário:

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.AutenticarModel;

import br.com.cotiinformatica.repositories.UsuarioRepository;

@Controller

public class AutenticarController {

@Autowired

private UsuarioRepository usuarioRepository;

@RequestMapping(value = "/") //Raiz do projeto

public ModelAndView autenticar() {

//WEB-INF/views/autenticar.jsp

ModelAndView modelAndView = new ModelAndView("autenticar");

modelAndView.addObject("model", new AutenticarModel());

return modelAndView;

}

@RequestMapping(value = "/autenticar-usuario", method = RequestMethod.POST)

public ModelAndView autenticarUsuario(AutenticarModel model) {

ModelAndView modelAndView = new ModelAndView("autenticar");

try {

//consultar o usuário no banco de dados através do email e da senha

Usuario usuario = usuarioRepository.findByEmailAndSenha

(model.getEmail(), model.getSenha());

//verificar se o usuário foi encontrado

if(usuario != null) {

//TODO

}

else {

modelAndView.addObject("mensagem\_erro",

"Acesso negado. Usuário não encontrado.");

}

}

catch(Exception e) {

modelAndView.addObject("mensagem\_erro", e.getMessage());

}

modelAndView.addObject("model", model);

return modelAndView;

}

}

**Exibindo a mensagem na página:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**<p class=*"text-danger"*>**

**${mensagem\_erro}**

**</p>**

**Testando a mensagem de acesso negado:**

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Sessão**

Recurso utilizado para que aplicações web possam gravar dados no navegador do usuário. Estes dados são armazenados em uma sessão, que é apagada sempre que o navegador é fechado.

Para o nosso fluxo de autenticação funcionar, vamos gravar em sessão os dados do usuário autenticado no sistema.

Iremos gravar na sessão as seguintes informações:

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

/dtos/**UsuarioDTO.java**

**package** br.com.cotiinformatica.dtos;

**import** java.util.Date;

**import** lombok.AllArgsConstructor;

**import** lombok.Getter;

**import** lombok.NoArgsConstructor;

**import** lombok.Setter;

**import** lombok.ToString;

@Setter

@Getter

@NoArgsConstructor

@AllArgsConstructor

@ToString

**public** **class** UsuarioDTO {

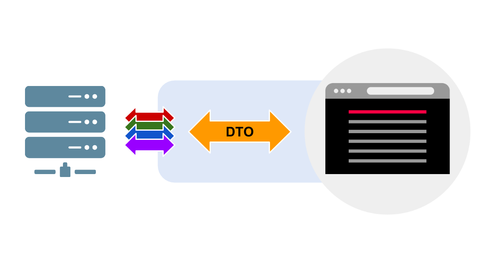
**private** Integer idUsuario;

**private** String nome;

**private** String email;

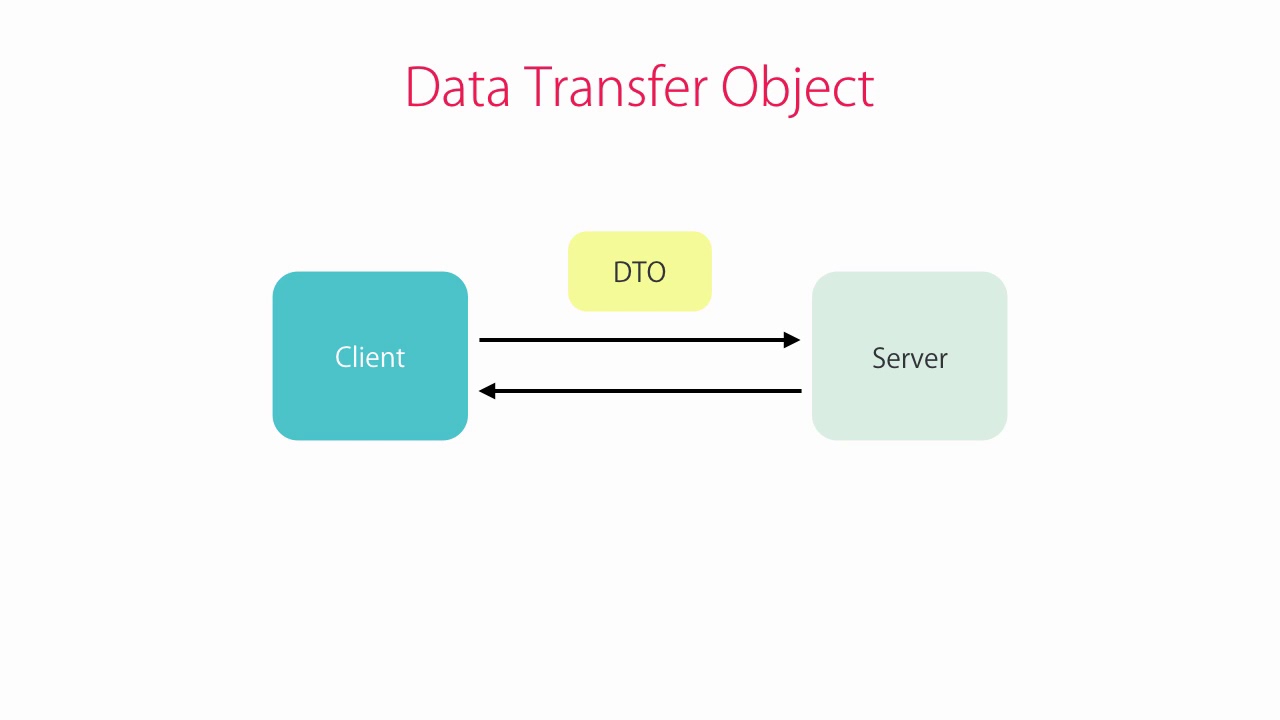
**private** Date dataHoraAcesso;

}



***Data Transfer Object*** (DTO) ou simplesmente ***Transfer Object*** é um padrão bastante usado em Java para o transporte de dados entre diferentes componentes de um sistema, diferentes instâncias ou processos de um sistema distribuído ou diferentes sistemas via serialização.  
  
A ideia consiste basicamente em agrupar um conjunto de atributos numa classe simples de forma a otimizar a comunicação.  
  
Numa chamada remota, seria ineficiente passar cada atributo individualmente. Da mesma forma seria ineficiente ou até causaria erros passar uma entidade mais complexa.

Além disso, muitas vezes os dados usados na comunicação não refletem exatamente os atributos do seu modelo. Então, um DTO seria uma classe que provê exatamente aquilo que é necessário para um determinado processo.



O padrão de projeto DTO é muito útil tanto para receber dados quanto para enviá-los, pois podemos manipular da forma que quisermos tais dados para facilitar a comunicação entre o servidor e o cliente.  
  
Quando se está criando uma API é de extrema importância não apenas pensar como um usuário regular irá interagir, mas também se defender contra usuários maliciosos para que eles não consigam causar nenhum prejuízo para a sua aplicação e para seus usuários.

Em Java, para podermos manipular dados em sessão precisamos   
usar um componente: **HttpServletRequest**

**package** br.com.cotiinformatica.controller;

**import** java.util.Date;

**import** javax.servlet.http.HttpServletRequest;

**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.dtos.UsuarioDTO;

**import** br.com.cotiinformatica.entities.Usuario;

**import** br.com.cotiinformatica.models.AutenticarModel;

**import** br.com.cotiinformatica.repositories.UsuarioRepository;

@Controller

**public** **class** AutenticarController {

@Autowired

**private** UsuarioRepository usuarioRepository;

@RequestMapping(value = "/") //Raiz do projeto

**public** ModelAndView autenticar() {

//WEB-INF/views/autenticar.jsp

ModelAndView modelAndView

= **new** ModelAndView("autenticar");

modelAndView.addObject("model", **new** AutenticarModel());

**return** modelAndView;

}

@RequestMapping(value = "/autenticar-usuario",

method = RequestMethod.***POST***)

**public** ModelAndView autenticarUsuario

(AutenticarModel model, HttpServletRequest request) {

ModelAndView modelAndView

= **new** ModelAndView("autenticar");

**try** {

//consultar o usuário no banco de dados

//através do email e da senha

Usuario usuario = usuarioRepository

.findByEmailAndSenha

(model.getEmail(), model.getSenha());

//verificar se o usuário foi encontrado

**if**(usuario != **null**) {

//criando um objeto da classe

//DTO (DATA TRANSFER OBJECT)

UsuarioDTO usuarioDTO = **new** UsuarioDTO();

usuarioDTO.setIdUsuario

(usuario.getIdUsuario());

usuarioDTO.setNome(usuario.getNome());

usuarioDTO.setEmail(usuario.getEmail());

usuarioDTO.setDataHoraAcesso(**new** Date());

//gravar os dados do usuário em sessão

request.getSession().setAttribute

("usuario", usuarioDTO);

//redirecionar para a página de

//boas vindas do sistema

modelAndView.setViewName

("redirect:/admin/dashboard");

}

**else** {

modelAndView.addObject("mensagem\_erro",

"Acesso negado. Usuário não encontrado.");

}

}

**catch**(Exception e) {

modelAndView.addObject

("mensagem\_erro", e.getMessage());

}

modelAndView.addObject("model", model);

**return** modelAndView;

}

}

Criando a página inicial que será exibida pelo sistema

assim que o usuário se autenticar.

/WEB-INF/views/admin/**dashboard.jsp**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Para que a página possa ser aberta no navegador, precisamos criar um controlador que mapeie a rota de navegação para esta página.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**package** br.com.cotiinformatica.controller;

**import** org.springframework.stereotype.Controller;

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

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

@Controller

**public** **class** DashboardController {

@RequestMapping(value = "/admin/dashboard")

**public** ModelAndView dashboard() {

// WEB-INF/views/dashboard.jsp

ModelAndView modelAndView

= **new** ModelAndView("admin/dashboard");

**return** modelAndView;

}

}

Desenhando a página **/dashboard**

/WEB-INF/views/admin/dashboard.jsp

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

pageEncoding=*"ISO-8859-1"*%>

<!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"* />

</head>

<body>

<nav class=*"navbar navbar-expand-lg bg-body-tertiary"*>

<div class=*"container-fluid"*>

<a class=*"navbar-brand"* href=*"#"*>Sistema Contas</a>

<button class=*"navbar-toggler"* type=*"button"* data-bs-toggle=*"collapse"*

data-bs-target=*"#navbarSupportedContent"* aria-controls=*"navbarSupportedContent"* aria-expanded=*"false"* aria-label=*"Toggle navigation"*>

<span class=*"navbar-toggler-icon"*></span>

</button>

<div class=*"collapse navbar-collapse"* id=*"navbarSupportedContent"*>

<ul class=*"navbar-nav me-auto mb-2 mb-lg-0"*>

<li class=*"nav-item"*>

<a class=*"nav-link"* aria-current=*"page"*

href=*"/sistemacontas/admin/dashboard"*>Dashboard</a>

</li>

<li class=*"nav-item"*>

<a class=*"nav-link"* aria-current=*"page"*

href=*"/sistemacontas/admin/dados-usuario"*>Dados do usuário</a>

</li>

<li class=*"nav-item dropdown"*>

<a class=*"nav-link dropdown-toggle"* href=*"#"* role=*"button"* data-bs-

toggle=*"dropdown"* aria-expanded=*"false"*>

Gerenciar contas

</a>

<ul class=*"dropdown-menu"*>

<li><a class=*"dropdown-item"*

href=*"/sistemacontas/admin/cadastrar-contas"*>

Cadastrar contas</a></li>

<li><a class=*"dropdown-item"*

href=*"/sistemacontas/admin/consultar-contas"*>

Consultar contas</a></li>

</ul>

</li>

</ul>

<div class=*"d-flex"*>

<div>

<div>

<strong>**${usuario.nome}**</strong>

</div>

<div style="margin-top: *-8px!important*;">

<small>**${usuario.email}**</small>

</div>

</div>

&nbsp;&nbsp;&nbsp;

<a href=*"/sistemacontas/logout"* class=*"btn btn-outline-secondary"*

onclick="return confirm('Deseja realmente sair do sistema?');">

Sair do sistema

</a>

</div>

</div>

</div>

</nav>

<!-- 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>

</body>

</html>

**Testando:**

<http://localhost:8081/sistemacontas/admin/dashboard>

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Criando o controlador para Logout do usuário

/controllers/LogoutController.java

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**package** br.com.cotiinformatica.controller;

**import** javax.servlet.http.HttpServletRequest;

**import** org.springframework.stereotype.Controller;

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

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

@Controller

**public** **class** LogoutController {

@RequestMapping(value = "/logout")

**public** ModelAndView logout(HttpServletRequest request) {

// apagando os dados do usuário gravado em sessão

request.getSession().removeAttribute("usuario");

// redirecionar para a página de autenticação

ModelAndView modelAndView

= **new** ModelAndView("redirect:/");

**return** modelAndView;

}

}

**Testando:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Publicando o trabalho no 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 'Autenticação de usuário'**

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**