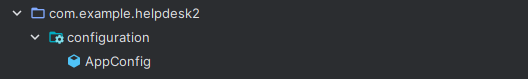


configuration



**configuration/AppConfig**

package com.example.helpdesk2.configuration;

import jakarta.servlet.MultipartConfigElement;

import org.springframework.boot.web.servlet.MultipartConfigFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.util.unit.DataSize;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

@ComponentScan("com")

public class AppConfig implements WebMvcConfigurer {

@Bean

public MultipartConfigElement multipartConfigElement() {

MultipartConfigFactory factory = new MultipartConfigFactory();

factory.setMaxFileSize(DataSize.parse("10MB"));

factory.setMaxRequestSize(DataSize.parse("10MB"));

return factory.createMultipartConfig();

}

@Bean

public WebMvcConfigurer corsConfigurer() {

return new WebMvcConfigurer() {

@Override

public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/api/\*\*")

.allowedOrigins("http://localhost:5173")

.allowedMethods("GET", "POST", "PUT", "DELETE")

.allowCredentials(true)

.maxAge(3600);

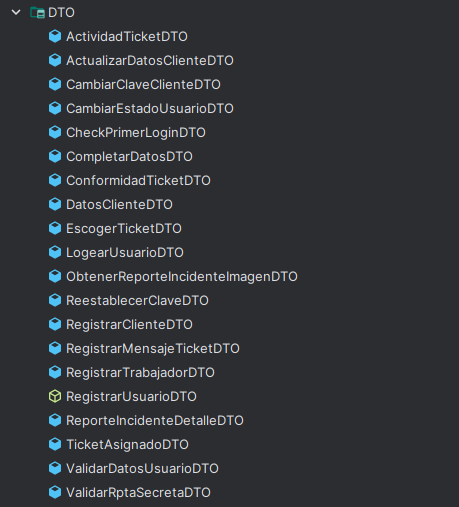
}

};

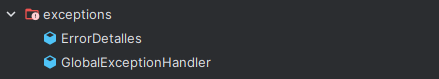
}

}

Data Transfer Object



Exceptions



**exceptions/ErrorDetalles**

package com.example.helpdesk2.exceptions;

public class ErrorDetalles {

private String mensaje;

public String getMensaje() {

return mensaje;

}

public void setMensaje(String mensaje) {

this.mensaje = mensaje;

}

}

**exceptions/GlobalExceptionHandler**

package com.example.helpdesk2.exceptions;

import org.springframework.http.HttpStatus;

import org.springframework.validation.FieldError;

import org.springframework.web.bind.MethodArgumentNotValidException;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

@RestControllerAdvice

public class GlobalExceptionHandler {

/\*@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public ResponseEntity<ErrorDetalles> exceptionEndPointNoEncontrado(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("El recurso no existe");

return ResponseEntity

.status(HttpStatus.NOT\_FOUND)

.body

(errorDetalles);

}\*/

@ExceptionHandler(MethodArgumentNotValidException.class)

@ResponseStatus(HttpStatus.BAD\_REQUEST)

public Map<String, Object> exceptionValidacionesInvalidas(MethodArgumentNotValidException e){

Map<String, String> errores = new HashMap<>();

e.getBindingResult().getAllErrors().forEach(error -> {

String campo = ((FieldError) error).getField();

String errorMensaje = error.getDefaultMessage();

errores.put(campo, errorMensaje);

});

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errores);

return respuesta;

}

@ExceptionHandler(IOException.class)

@ResponseStatus(HttpStatus.BAD\_REQUEST)

public Map<String, Object> exceptionIOException(IOException e){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("Hubo un error, intente nuevamente");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

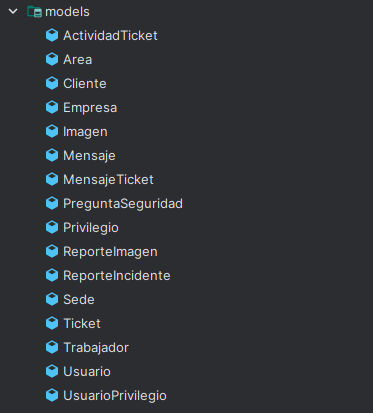
respuesta.put("error", errorDetalles);

return respuesta;

}

}

models



moduloCliente



**moduloCliente/controllers/ActualizarInformacionController**

package com.example.helpdesk2.moduloCliente.controllers;

import com.example.helpdesk2.DTO.ActualizarDatosClienteDTO;

import com.example.helpdesk2.DTO.CambiarClaveClienteDTO;

import com.example.helpdesk2.models.Cliente;

import com.example.helpdesk2.moduloCliente.services.ActualizarInformacionService;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import jakarta.validation.Valid;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.transaction.annotation.Transactional;

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

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/api/modulo-cliente")

public class ActualizarInformacionController {

private final ActualizarInformacionService actualizarInformacionService;

private final LoggedUserManagamentService loggedUserManagamentService;

public ActualizarInformacionController(ActualizarInformacionService actualizarInformacionService, LoggedUserManagamentService loggedUserManagamentService) {

this.actualizarInformacionService = actualizarInformacionService;

this.loggedUserManagamentService = loggedUserManagamentService;

}

@PostMapping("/actualizar-datos-cliente")

public ResponseEntity<Map<String, Object>> actualizarDatosCliente(@RequestBody ActualizarDatosClienteDTO actualizarDatosClienteDTO) {

int idUsuario = loggedUserManagamentService.getIdUsuario();

int idCliente = actualizarInformacionService.obtenerClientePorIdUsuario(idUsuario).getIdCliente();

actualizarInformacionService.cambiarDatosCliente(actualizarDatosClienteDTO, idCliente, idUsuario);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/datos-cliente")

public ResponseEntity<Map<String, Object>> obtenerDatosCliente() {

int idUsuario = loggedUserManagamentService.getIdUsuario();

Map<String, Object> respuesta = actualizarInformacionService.obtenerDatosCliente(idUsuario);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/cambiar-clave-cliente")

public ResponseEntity<Map<String, Object>> cambiarClaveCliente(@Valid @RequestBody CambiarClaveClienteDTO cambiarClaveClienteDTO) {

int idUsuario = loggedUserManagamentService.getIdUsuario();

actualizarInformacionService.cambiarClaveCliente(cambiarClaveClienteDTO, idUsuario);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

}

**moduloCliente/controllers/ReportarIncidenteController**

package com.example.helpdesk2.moduloCliente.controllers;

import com.example.helpdesk2.moduloCliente.services.ReportarIncidenteService;

import com.example.helpdesk2.services.FileStorageService;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import org.springframework.http.ResponseEntity;

import org.springframework.transaction.annotation.Transactional;

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

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.nio.file.Path;

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/api/modulo-cliente")

public class ReportarIncidenteController {

private final ReportarIncidenteService reportarIncidenteService;

private final FileStorageService fileStorageService;

private final LoggedUserManagamentService loggedUserManagamentService;

public ReportarIncidenteController(ReportarIncidenteService reportarIncidenteService, FileStorageService fileStorageService, LoggedUserManagamentService loggedUserManagamentService) {

this.reportarIncidenteService = reportarIncidenteService;

this.fileStorageService = fileStorageService;

this.loggedUserManagamentService = loggedUserManagamentService;

}

@Transactional

@PostMapping("/reportar-incidente")

public ResponseEntity<Map<String, Object>> reportarIncidente(

@RequestParam String nombreIncidente,

@RequestParam String descripcion,

@RequestPart(required = false) MultipartFile[] imagenes) throws IOException {

int idCliente = reportarIncidenteService.obtenerIdCliente(loggedUserManagamentService.getIdUsuario());

int idIncidente = reportarIncidenteService.registrarIncidente(idCliente, nombreIncidente, descripcion);

if (imagenes != null) {

Path reporteFolder = fileStorageService.crearDirectorio(idIncidente);

for (MultipartFile imagen : imagenes) {

String nombreImagen = fileStorageService.guardar(imagen, reporteFolder);

reportarIncidenteService.registrarImagen(idIncidente, nombreImagen);

}

}

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.ok()

.body(respuesta);

}

}

**moduloCliente/controllers/SeguimientoReporteIncidenteController**

package com.example.helpdesk2.moduloCliente.controllers;

import com.example.helpdesk2.DTO.ActividadTicketDTO;

import com.example.helpdesk2.DTO.ConformidadTicketDTO;

import com.example.helpdesk2.DTO.RegistrarMensajeTicketDTO;

import com.example.helpdesk2.DTO.ReporteIncidenteDetalleDTO;

import com.example.helpdesk2.models.Mensaje;

import com.example.helpdesk2.models.ReporteIncidente;

import com.example.helpdesk2.moduloCliente.services.SeguimientoReporteIncidenteService;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import org.springframework.http.ResponseEntity;

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

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

@RestController

@RequestMapping("/api/modulo-cliente")

public class SeguimientoReporteIncidenteController {

private final SeguimientoReporteIncidenteService seguimientoReporteIncidenteService;

private final LoggedUserManagamentService loggedUserManagamentService;

public SeguimientoReporteIncidenteController(SeguimientoReporteIncidenteService seguimientoReporteIncidenteService, LoggedUserManagamentService loggedUserManagamentService) {

this.seguimientoReporteIncidenteService = seguimientoReporteIncidenteService;

this.loggedUserManagamentService = loggedUserManagamentService;

}

@GetMapping("/reportes")

public ResponseEntity<List<ReporteIncidenteDetalleDTO>> obtenerReportesIncidentesCliente() {

int idUsuario = loggedUserManagamentService.getIdUsuario();

List<ReporteIncidenteDetalleDTO> reportesIncidentesDetalles = seguimientoReporteIncidenteService.obtenerReportesIncidentesCliente(idUsuario);

return ResponseEntity

.ok()

.body(reportesIncidentesDetalles);

}

@GetMapping("/reportes/{idReporteIncidente}")

public ResponseEntity<Map<String, Object>> obtenerReporteIncidentePorID(@PathVariable int idReporteIncidente) {

Map<String,Object> respuesta = seguimientoReporteIncidenteService.obtenerReporteIncidentePorID(idReporteIncidente);

return ResponseEntity

.ok()

.body(respuesta);

}

@GetMapping("/reportes/{idReporteIncidente}/mensajes")

public ResponseEntity<List<Mensaje>> obtenerMensajesTicket(@PathVariable int idReporteIncidente) {

List<Mensaje> mensajesTicket = seguimientoReporteIncidenteService.obtenerMensajesTicket(idReporteIncidente);

return ResponseEntity

.ok()

.body(mensajesTicket);

}

@PostMapping("/reportes/mensaje")

public ResponseEntity<Void> registrarMensajeTicket(@RequestBody RegistrarMensajeTicketDTO registrarMensajeTicketDTO) {

seguimientoReporteIncidenteService.registrarMensajeTicket(registrarMensajeTicketDTO);

return ResponseEntity

.ok()

.build();

}

@PostMapping("/reportes/cambiar-estado-ticket")

public ResponseEntity<Void> cambiarEstadoTicket(@RequestBody ConformidadTicketDTO conformidadTicketDTO) {

seguimientoReporteIncidenteService.confirmarAtencion(conformidadTicketDTO);

return ResponseEntity

.ok()

.build();

}

}

moduloCliente/exceptions

package com.example.helpdesk2.moduloCliente.exceptions;

import com.example.helpdesk2.exceptions.ErrorDetalles;

import org.apache.tomcat.util.http.fileupload.impl.FileSizeLimitExceededException;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;

import java.util.Map;

@RestControllerAdvice

public class ModuloClienteExceptionHandler {

@ExceptionHandler(FileSizeLimitExceededException.class)

public ResponseEntity<Map<String, Object>> handleFileSizeLimitExceed(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("Los archivos exceden el tamaño máximo permitido de 10MB");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.BAD\_REQUEST)

.body(respuesta);

}

}

**moduloCliente/services/ActualizarInformacionService**

package com.example.helpdesk2.moduloCliente.services;

import com.example.helpdesk2.DTO.ActualizarDatosClienteDTO;

import com.example.helpdesk2.DTO.CambiarClaveClienteDTO;

import com.example.helpdesk2.models.Cliente;

import com.example.helpdesk2.moduloSeguridad.exceptions.ClavesNoCoincidenException;

import com.example.helpdesk2.repositories.ClienteRepository;

import com.example.helpdesk2.repositories.UsuarioRepository;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.HashMap;

import java.util.Map;

@Service

public class ActualizarInformacionService {

private final ClienteRepository clienteRepository;

private final UsuarioRepository usuarioRepository;

public ActualizarInformacionService(ClienteRepository clienteRepository, UsuarioRepository usuarioRepository) {

this.clienteRepository = clienteRepository;

this.usuarioRepository = usuarioRepository;

}

public Cliente obtenerClientePorIdUsuario(int idUsuario){

return clienteRepository.buscarClientePorIdUsuario(idUsuario);

}

@Transactional

public void cambiarDatosCliente(ActualizarDatosClienteDTO actualizarDatosClienteDTO, int idCliente, int idUsuario){

clienteRepository.actualizarDatosCliente(actualizarDatosClienteDTO, idCliente);

usuarioRepository.actualizarCorreo(actualizarDatosClienteDTO.getCorreo(), idUsuario);

}

public Map<String, Object> obtenerDatosCliente(int idUsuario){

Cliente c = clienteRepository.buscarClientePorIdUsuario(idUsuario);

String correo = usuarioRepository.buscarCorreoUsuarioPorId(idUsuario);

Map<String, Object> datosCliente = new HashMap<>();

datosCliente.put("anydesk", c.getAnydesk());

datosCliente.put("teamviewer", c.getTeamviewer());

datosCliente.put("correo", correo);

return datosCliente;

}

public void cambiarClaveCliente(CambiarClaveClienteDTO cambiarClaveClienteDTO, int idUsuario){

if (cambiarClaveClienteDTO.getClave().equals(cambiarClaveClienteDTO.getReClave())) {

clienteRepository.actualizarClave(cambiarClaveClienteDTO, idUsuario);

} else {

throw new ClavesNoCoincidenException();

}

}

}

**moduloCliente/services/ReportarIncidenteService**

package com.example.helpdesk2.moduloCliente.services;

import com.example.helpdesk2.repositories.ClienteRepository;

import com.example.helpdesk2.repositories.ImagenRepository;

import com.example.helpdesk2.repositories.ReporteImagenRepository;

import com.example.helpdesk2.repositories.ReporteIncidenteRepository;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

@Service

public class ReportarIncidenteService {

private final ReporteIncidenteRepository reporteIncidenteRepository;

private final ImagenRepository imagenRepository;

private final ReporteImagenRepository reporteImagenRepository;

private final ClienteRepository clienteRepository;

public ReportarIncidenteService(ReporteIncidenteRepository reporteIncidenteRepository, ImagenRepository imagenRepository, ReporteImagenRepository reporteImagenRepository, ClienteRepository clienteRepository) {

this.reporteIncidenteRepository = reporteIncidenteRepository;

this.imagenRepository = imagenRepository;

this.reporteImagenRepository = reporteImagenRepository;

this.clienteRepository = clienteRepository;

}

public int registrarIncidente(int idCliente, String nombreIncidente, String descripcion) {

reporteIncidenteRepository.guardarReporteIncidente(idCliente, nombreIncidente, descripcion);

return reporteIncidenteRepository.obtenerUltimoIDReporteIncidente();

}

public void registrarImagen(int idReporte, String nombreImagen){

imagenRepository.guardarImagen(nombreImagen);

int idImagen = imagenRepository.obtenerUltimoIDImagen();

reporteImagenRepository.guardarReporteImagen(idReporte, idImagen);

}

public int obtenerIdCliente(int idUsuario){

return clienteRepository.buscarClientePorIdUsuario(idUsuario).getIdCliente();

}

}

**moduloCliente/services/SeguimientoReporteIncidenteService**

package com.example.helpdesk2.moduloCliente.services;

import com.example.helpdesk2.DTO.ActividadTicketDTO;

import com.example.helpdesk2.DTO.ConformidadTicketDTO;

import com.example.helpdesk2.DTO.RegistrarMensajeTicketDTO;

import com.example.helpdesk2.DTO.ReporteIncidenteDetalleDTO;

import com.example.helpdesk2.models.\*;

import com.example.helpdesk2.repositories.\*;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.\*;

@Service

public class SeguimientoReporteIncidenteService {

private final ReporteIncidenteRepository reporteIncidenteRepository;

private final TicketRepository ticketRepository;

private final ClienteRepository clienteRepository;

private final UsuarioRepository usuarioRepository;

private final MensajeRepository mensajeRepository;

private final MensajeTicketRepository mensajeTicketRepository;

private final TrabajadorRepository trabajadorRepository;

private final ActividadTicketRepository actividadTicketRepository;

public SeguimientoReporteIncidenteService(ReporteIncidenteRepository reporteIncidenteRepository, TicketRepository ticketRepository, ClienteRepository clienteRepository, UsuarioRepository usuarioRepository, MensajeRepository mensajeRepository, MensajeTicketRepository mensajeTicketRepository, TrabajadorRepository trabajadorRepository, ActividadTicketRepository actividadTicketRepository) {

this.reporteIncidenteRepository = reporteIncidenteRepository;

this.ticketRepository = ticketRepository;

this.clienteRepository = clienteRepository;

this.usuarioRepository = usuarioRepository;

this.mensajeRepository = mensajeRepository;

this.mensajeTicketRepository = mensajeTicketRepository;

this.trabajadorRepository = trabajadorRepository;

this.actividadTicketRepository = actividadTicketRepository;

}

public List<ReporteIncidenteDetalleDTO> obtenerReportesIncidentesCliente(int idUsuario) {

int idCliente = clienteRepository.buscarClientePorIdUsuario(idUsuario).getIdCliente();

List<ReporteIncidente> reportesIncidentes = reporteIncidenteRepository.buscarReportesIncidentesCliente(idCliente);

List<ReporteIncidenteDetalleDTO> reportesIncidentesDetalles = new ArrayList<>();

for (ReporteIncidente reporteIncidente : reportesIncidentes) {

Optional<Ticket> ticketOptional = ticketRepository.buscarTicketPorIDReporteIncidente(reporteIncidente.getIdReporteIncidente());

ReporteIncidenteDetalleDTO reporteIncidenteDetalleDTO = new ReporteIncidenteDetalleDTO();

reporteIncidenteDetalleDTO.setReporteIncidente(reporteIncidente);

if (ticketOptional.isPresent()) {

reporteIncidenteDetalleDTO.setTrabajador(usuarioRepository.buscarUsuarioPorId(idUsuario).getNombres());

reporteIncidenteDetalleDTO.setFechaCreacion(ticketOptional.get().getFecha());

} else {

reporteIncidenteDetalleDTO.setTrabajador("Por asignar");

reporteIncidenteDetalleDTO.setFechaCreacion(null);

}

reportesIncidentesDetalles.add(reporteIncidenteDetalleDTO);

}

return reportesIncidentesDetalles;

}

public Map<String, Object> obtenerReporteIncidentePorID(int idReporteIncidente) {

ReporteIncidente reporteIncidente = reporteIncidenteRepository.buscarReporteIncidente(idReporteIncidente);

Ticket ticket = ticketRepository.buscarTicketPorIDReporteIncidente(idReporteIncidente).get();

int idUsuario = trabajadorRepository.buscarTrabajadorPorIdTrabajador(ticket.getIdTrabajador()).getIdUsuario();

String nombres = usuarioRepository.buscarUsuarioPorId(idUsuario).getNombres();

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("reporteIncidente", reporteIncidente);

respuesta.put("ticket", ticket);

respuesta.put("nombreTrabajador", nombres);

return respuesta;

}

public List<Mensaje> obtenerMensajesTicket(int idTicket) {

return mensajeRepository.buscarMensajesTicket(idTicket);

}

@Transactional

public void registrarMensajeTicket(RegistrarMensajeTicketDTO registrarMensajeTicketDTO) {

mensajeRepository.guardarMensaje(registrarMensajeTicketDTO);

int idMensaje = mensajeRepository.obtenerUltimoIdInsertado();

mensajeTicketRepository.guardarMensajeTicket(registrarMensajeTicketDTO.getIdTicket(), idMensaje);

}

@Transactional

public void confirmarAtencion(ConformidadTicketDTO conformidadTicketDTO){

ticketRepository.actualizarEstadoTicket(conformidadTicketDTO.getEstadoTicket(), conformidadTicketDTO.getIdTicket());

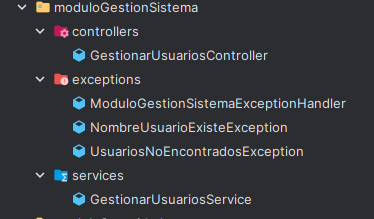
reporteIncidenteRepository.actualizarEstadoReporteIncidente(conformidadTicketDTO.getIdReporteIncidente(), conformidadTicketDTO.getEstadoReporteIncidente());

actividadTicketRepository.guardarActividadTicket(conformidadTicketDTO.getIdTicket(), conformidadTicketDTO.getMensaje(), conformidadTicketDTO.getEstadoTicket());

}

}

**moduloGestionSistema**



**moduloGestionSistema/controllers/GestionarUsuariosController**

package com.example.helpdesk2.moduloGestionSistema.controllers;

import com.example.helpdesk2.DTO.CambiarEstadoUsuarioDTO;

import com.example.helpdesk2.DTO.RegistrarClienteDTO;

import com.example.helpdesk2.DTO.RegistrarTrabajadorDTO;

import com.example.helpdesk2.DTO.RegistrarUsuarioDTO;

import com.example.helpdesk2.models.Privilegio;

import com.example.helpdesk2.models.Usuario;

import com.example.helpdesk2.moduloGestionSistema.services.GestionarUsuariosService;

import jakarta.validation.Valid;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.transaction.annotation.Transactional;

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

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@RestController

@RequestMapping("/api/gestion-sistema")

public class GestionarUsuariosController {

private final GestionarUsuariosService gestionarUsuariosService;

public GestionarUsuariosController(GestionarUsuariosService gestionarUsuariosService) {

this.gestionarUsuariosService = gestionarUsuariosService;

}

@GetMapping("/usuarios")

public ResponseEntity<Map<String, Object>> obtenerUsuarios(

@RequestParam(required = false) String filtro,

@RequestParam(required = false) String valor){

System.out.println("hola");

List<Usuario> usuarios = gestionarUsuariosService.obtenerUsuarios(filtro, valor);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("usuarios", usuarios);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/cambiar-estado-usuario")

public ResponseEntity<Map<String, Object>> cambiarEstadoUsuario(@RequestBody CambiarEstadoUsuarioDTO cambiarEstadoUsuarioDTO){

gestionarUsuariosService.cambiarEstadoUsuario(cambiarEstadoUsuarioDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/privilegios-todos")

public ResponseEntity<Map<String, Object>> privilegiosTodos(){

List<Privilegio> privilegios = gestionarUsuariosService.obtenerTodosPrivilegios();

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("privilegios", privilegios);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@Transactional

@PostMapping("/crear-trabajador")

public ResponseEntity<Map<String, Object>> crearTrabajador(@Valid @RequestBody RegistrarTrabajadorDTO registrarTrabajadorDTO){

int idUsuario = gestionarUsuariosService.crearUsuario(registrarTrabajadorDTO);

registrarTrabajadorDTO.setIdUsuario(idUsuario);

gestionarUsuariosService.crearTrabajador(registrarTrabajadorDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@Transactional

@PostMapping("/crear-cliente")

public ResponseEntity<Map<String, Object>> crearCliente(@Valid @RequestBody RegistrarClienteDTO registrarClienteDTO){

int idUsuario = gestionarUsuariosService.crearUsuario(registrarClienteDTO);

registrarClienteDTO.setIdUsuario(idUsuario);

gestionarUsuariosService.crearCliente(registrarClienteDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/empresas")

public ResponseEntity<Map<String, Object>> obtenerEmpresas(){

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("empresas", gestionarUsuariosService.obtenerEmpresas());

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/sedes")

public ResponseEntity<Map<String, Object>> obtenerSedes(@RequestParam int idEmpresa){

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("sedes", gestionarUsuariosService.obtenerSedesPorEmpresa(idEmpresa));

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/areas")

public ResponseEntity<Map<String, Object>> obtenerAreas(@RequestParam int idSede){

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("areas", gestionarUsuariosService.obtenerAreasPorSede(idSede));

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

}

**moduloGestionSistema/exceptions/ModuloGestionSistemaExceptionHandler**

package com.example.helpdesk2.moduloGestionSistema.exceptions;

import com.example.helpdesk2.exceptions.ErrorDetalles;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;

import java.util.Map;

@RestControllerAdvice

public class ModuloGestionSistemaExceptionHandler {

@ExceptionHandler(UsuariosNoEncontradosException.class)

public ResponseEntity<Map<String, Object>> handleUsuariosNoEncontrados(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("No se encontraron usuarios");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.BAD\_REQUEST)

.body(respuesta);

}

@ExceptionHandler(NombreUsuarioExisteException.class)

public ResponseEntity<Map<String, Object>> handleNombreUsuarioNoExiste(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("El nombre de usuario ya existe");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.BAD\_REQUEST)

.body(respuesta);

}

}

**moduloGestionSistema/exceptions/NombreUsuarioExisteException**

package com.example.helpdesk2.moduloGestionSistema.exceptions;

public class NombreUsuarioExisteException extends RuntimeException{

}

**moduloGestionSistema/exceptions/UsuariosNoEncontradosException**

package com.example.helpdesk2.moduloGestionSistema.exceptions;

public class UsuariosNoEncontradosException extends RuntimeException{

}

**moduloGestionSistema/services/GestionarUsuariosService**

package com.example.helpdesk2.moduloGestionSistema.services;

import com.example.helpdesk2.DTO.CambiarEstadoUsuarioDTO;

import com.example.helpdesk2.DTO.RegistrarClienteDTO;

import com.example.helpdesk2.DTO.RegistrarTrabajadorDTO;

import com.example.helpdesk2.DTO.RegistrarUsuarioDTO;

import com.example.helpdesk2.models.\*;

import com.example.helpdesk2.moduloGestionSistema.exceptions.NombreUsuarioExisteException;

import com.example.helpdesk2.moduloGestionSistema.exceptions.UsuariosNoEncontradosException;

import com.example.helpdesk2.repositories.\*;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class GestionarUsuariosService {

private final UsuarioRepository usuarioRepository;

private final PrivilegioRepository privilegioRepository;

private final EmpresaRepository empresaRepository;

private final SedeRepository sedeRepository;

private final AreaRepository areaRepository;

private final TrabajadorRepository trabajadorRepository;

private final ClienteRepository clienteRepository;

private final UsuarioPrivilegioRepository usuarioPrivilegioRepository;

public GestionarUsuariosService(UsuarioRepository usuarioRepository, PrivilegioRepository privilegioRepository, EmpresaRepository empresaRepository, SedeRepository sedeRepository, AreaRepository areaRepository, TrabajadorRepository trabajadorRepository, ClienteRepository clienteRepository, UsuarioPrivilegioRepository usuarioPrivilegioRepository) {

this.usuarioRepository = usuarioRepository;

this.privilegioRepository = privilegioRepository;

this.empresaRepository = empresaRepository;

this.sedeRepository = sedeRepository;

this.areaRepository = areaRepository;

this.trabajadorRepository = trabajadorRepository;

this.clienteRepository = clienteRepository;

this.usuarioPrivilegioRepository = usuarioPrivilegioRepository;

}

public List<Usuario> obtenerUsuarios(String filtro, String valor) {

List<Usuario> usuarios;

if (filtro != null && valor != null) {

usuarios = usuarioRepository.buscarUsuariosPorFiltro(filtro, valor)

.orElseThrow(UsuariosNoEncontradosException::new);

} else {

usuarios = usuarioRepository.buscarTodosLosUsuarios()

.orElseThrow(UsuariosNoEncontradosException::new);

}

return usuarios;

}

public void cambiarEstadoUsuario(CambiarEstadoUsuarioDTO cambiarEstadoUsuarioDTO) {

usuarioRepository.cambiarEstadoUsuario(cambiarEstadoUsuarioDTO);

}

public List<Privilegio> obtenerTodosPrivilegios() {

return privilegioRepository.buscarTodosPrivilegios();

}

public int crearUsuario(RegistrarUsuarioDTO registrarUsuarioDTO) {

if (usuarioRepository.buscarNombreUsuarioExistente(registrarUsuarioDTO).isPresent()) {

throw new NombreUsuarioExisteException();

}

usuarioRepository.registrarUsuario(registrarUsuarioDTO);

int idUsuario = usuarioRepository.obtenerUltimoIdUsuario();

registrarUsuarioDTO.getPrivilegios().forEach(

idPrivilegio -> usuarioPrivilegioRepository.registrarPrivilegioUsuario(idUsuario, idPrivilegio)

);

return idUsuario;

}

public void crearTrabajador(RegistrarTrabajadorDTO registrarTrabajadorDTO) {

trabajadorRepository.registrarTrabajador(registrarTrabajadorDTO);

}

public void crearCliente(RegistrarClienteDTO registrarClienteDTO){

clienteRepository.registrarCliente(registrarClienteDTO);

}

public List<Empresa> obtenerEmpresas(){

return empresaRepository.buscarEmpresas();

}

public List<Sede> obtenerSedesPorEmpresa(int idEmpresa){

return sedeRepository.buscarSedesPorEmpresa(idEmpresa);

}

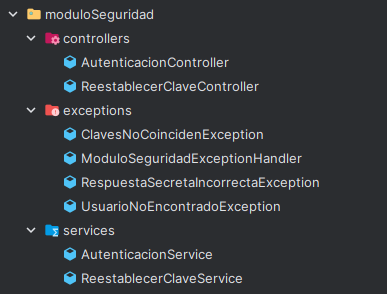
public List<Area> obtenerAreasPorSede(int idSede){

return areaRepository.buscarAreasPorSede(idSede);

}

}

**moduloSeguridad**



**moduloSeguridad/controllers/AuntenticacionController**

package com.example.helpdesk2.moduloSeguridad.controllers;

import com.example.helpdesk2.models.Usuario;

import com.example.helpdesk2.DTO.CompletarDatosDTO;

import com.example.helpdesk2.DTO.LogearUsuarioDTO;

import com.example.helpdesk2.DTO.CheckPrimerLoginDTO;

import com.example.helpdesk2.models.Privilegio;

import com.example.helpdesk2.moduloSeguridad.services.AutenticacionService;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import jakarta.servlet.http.Cookie;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpSession;

import jakarta.validation.Valid;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@RestController

@RequestMapping("/api/seguridad")

public class AutenticacionController {

private final AutenticacionService autenticacionService;

private final LoggedUserManagamentService loggedUserManagamentService;

public AutenticacionController(AutenticacionService autenticacionService, LoggedUserManagamentService loggedUserManagamentService) {

this.autenticacionService = autenticacionService;

this.loggedUserManagamentService = loggedUserManagamentService;

}

@PostMapping("/check-primer-login")

public ResponseEntity<Map<String, Object>> checkPrimerLogin(@Valid @RequestBody CheckPrimerLoginDTO checkPrimerLoginDTO) {

Usuario usuario = autenticacionService.checkUsuarioPrimerLogin(checkPrimerLoginDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("idUsuario", usuario.getIdUsuario());

respuesta.put("primerLogin", usuario.getPrimerLogin());

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/logear-usuario")

public ResponseEntity<Map<String, Object>> logearUsuario(@RequestBody LogearUsuarioDTO logearUsuarioDTO) {

autenticacionService.autenticarUsuario(logearUsuarioDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("idUsuario", loggedUserManagamentService.getIdUsuario());

respuesta.put("nombres", loggedUserManagamentService.getNombres());

respuesta.put("privilegios", loggedUserManagamentService.getPrivilegios());

respuesta.put("tipo", loggedUserManagamentService.getTipo());

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/logout-usuario")

public ResponseEntity<Map<String, Object>> logoutUsuario(HttpServletRequest request) {

HttpSession session = request.getSession(false);

if (session != null) {

session.invalidate();

}

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/preguntas-seguridad")

public ResponseEntity<Map<String, Object>> obtenerPreguntasSeguridad() {

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("preguntasSeguridad", autenticacionService.obtenerPreguntasSeguridad());

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/completar-datos")

public ResponseEntity<Map<String, Object>> completarDatos(@Valid @RequestBody CompletarDatosDTO completarDatosDTO) {

autenticacionService.completarDatosUsuario(completarDatosDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@GetMapping("/usuario-activo")

public ResponseEntity<Map<String, Object>> checkSesionActiva() {

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("idUsuario", loggedUserManagamentService.getIdUsuario());

respuesta.put("nombres", loggedUserManagamentService.getNombres());

respuesta.put("tipo", loggedUserManagamentService.getTipo());

respuesta.put("privilegios", loggedUserManagamentService.getPrivilegios());

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

}

**moduloSeguridad/controllers/ReestablecerClaveController**

package com.example.helpdesk2.moduloSeguridad.controllers;

import com.example.helpdesk2.models.Usuario;

import com.example.helpdesk2.DTO.ReestablecerClaveDTO;

import com.example.helpdesk2.DTO.ValidarDatosUsuarioDTO;

import com.example.helpdesk2.DTO.ValidarRptaSecretaDTO;

import com.example.helpdesk2.moduloSeguridad.services.ReestablecerClaveService;

import jakarta.validation.Valid;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

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

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/api/seguridad")

public class ReestablecerClaveController {

private final ReestablecerClaveService reestablecerClaveService;

public ReestablecerClaveController(ReestablecerClaveService reestablecerClaveService) {

this.reestablecerClaveService = reestablecerClaveService;

}

@PostMapping("/validar-datos-usuario")

public ResponseEntity<Map<String, Object>> validarDatosUsuario(@Valid @RequestBody ValidarDatosUsuarioDTO validarDatosUsuarioDTO){

Usuario usuario = reestablecerClaveService.validarUsuarioPorDatos(validarDatosUsuarioDTO);

String preguntaSeguridad = reestablecerClaveService.obtenerPreguntaSeguridad(usuario.getIdPreguntaSeguridad());

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("idUsuario", usuario.getIdUsuario());

respuesta.put("preguntaSeguridad", preguntaSeguridad);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/validar-rpta-secreta")

public ResponseEntity<Map<String, Object>> validarRptaSecreta(@Valid @RequestBody ValidarRptaSecretaDTO validarRptaSecretaDTO){

reestablecerClaveService.validarRespuestaSecreta(validarRptaSecretaDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

@PostMapping("/reestablecer-clave")

public ResponseEntity<Map<String, Object>> reestablecerClave(@Valid @RequestBody ReestablecerClaveDTO reestablecerClaveDTO){

reestablecerClaveService.reestablecerClave(reestablecerClaveDTO);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity

.status(HttpStatus.OK)

.body(respuesta);

}

}

**moduloSeguridad/exceptions/ClavesNoCoincidenException**

package com.example.helpdesk2.moduloSeguridad.exceptions;

public class ClavesNoCoincidenException extends RuntimeException{

}

**moduloSeguridad/exceptions/ModuloSeguridadExceptionHandler**

package com.example.helpdesk2.moduloSeguridad.exceptions;

import com.example.helpdesk2.exceptions.ErrorDetalles;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;

import java.util.Map;

@RestControllerAdvice

public class ModuloSeguridadExceptionHandler {

@ExceptionHandler(UsuarioNoEncontradoException.class)

public ResponseEntity<Map<String, Object>> handleUsuarioNoEncontrado(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("El usuario no existe o se encuentra deshabilitado");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.UNAUTHORIZED)

.body(respuesta);

}

@ExceptionHandler(ClavesNoCoincidenException.class)

public ResponseEntity<Map<String, Object>> handleClavesNoCoinciden(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("Las claves no coinciden");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.UNAUTHORIZED)

.body(respuesta);

}

@ExceptionHandler(RespuestaSecretaIncorrectaException.class)

public ResponseEntity<Map<String, Object>> handleRespuestaSecretaIncorrecta(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("La respuesta secreta es incorrecta");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.UNAUTHORIZED)

.body(respuesta);

}

}

**moduloSeguridad/exceptions/RespuestaSecretaIncorrectaException**

package com.example.helpdesk2.moduloSeguridad.exceptions;

public class RespuestaSecretaIncorrectaException extends RuntimeException{

}

**moduloSeguridad/exceptions/UsuarioNoEncontradoException**

package com.example.helpdesk2.moduloSeguridad.exceptions;

public class UsuarioNoEncontradoException extends RuntimeException{

}

**moduloSeguridad/services/AutenticacionService**

package com.example.helpdesk2.moduloSeguridad.services;

import com.example.helpdesk2.DTO.CheckPrimerLoginDTO;

import com.example.helpdesk2.DTO.CompletarDatosDTO;

import com.example.helpdesk2.DTO.LogearUsuarioDTO;

import com.example.helpdesk2.models.PreguntaSeguridad;

import com.example.helpdesk2.moduloSeguridad.exceptions.ClavesNoCoincidenException;

import com.example.helpdesk2.moduloSeguridad.exceptions.UsuarioNoEncontradoException;

import com.example.helpdesk2.models.Usuario;

import com.example.helpdesk2.repositories.PreguntaSeguridadRepository;

import com.example.helpdesk2.repositories.PrivilegioRepository;

import com.example.helpdesk2.repositories.UsuarioPrivilegioRepository;

import com.example.helpdesk2.repositories.UsuarioRepository;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class AutenticacionService {

private final UsuarioRepository usuarioRepository;

private final LoggedUserManagamentService loggedUserManagamentService;

private final PrivilegioRepository privilegioRepository;

private final PreguntaSeguridadRepository preguntaSeguridadRepository;

private final UsuarioPrivilegioRepository usuarioPrivilegioRepository;

public AutenticacionService(UsuarioRepository usuarioRepository, LoggedUserManagamentService loggedUserManagamentService, PrivilegioRepository privilegioRepository, PreguntaSeguridadRepository preguntaSeguridadRepository, UsuarioPrivilegioRepository usuarioPrivilegioRepository) {

this.usuarioRepository = usuarioRepository;

this.loggedUserManagamentService = loggedUserManagamentService;

this.privilegioRepository = privilegioRepository;

this.preguntaSeguridadRepository = preguntaSeguridadRepository;

this.usuarioPrivilegioRepository = usuarioPrivilegioRepository;

}

public Usuario checkUsuarioPrimerLogin(CheckPrimerLoginDTO checkPrimerLoginDTO){

return usuarioRepository.buscarUsuarioPorCredenciales(checkPrimerLoginDTO).orElseThrow(UsuarioNoEncontradoException::new);

}

public void autenticarUsuario(LogearUsuarioDTO logearUsuarioDTO){

Usuario u = usuarioRepository.buscarUsuarioPorId(logearUsuarioDTO.getIdUsuario());

loggedUserManagamentService.setIdUsuario(u.getIdUsuario());

loggedUserManagamentService.setNombres(u.getNombres());

loggedUserManagamentService.setPrivilegios(privilegioRepository.buscarPrivilegiosDeUsuario(u.getIdUsuario()));

loggedUserManagamentService.setTipo(u.getTipo());

}

public void completarDatosUsuario(CompletarDatosDTO completarDatosDTO) {

if(completarDatosDTO.getClave().equals(completarDatosDTO.getReClave())){

usuarioRepository.completarDatosUsuario(completarDatosDTO);

}

else{

throw new ClavesNoCoincidenException();

}

}

public List<PreguntaSeguridad> obtenerPreguntasSeguridad(){

return preguntaSeguridadRepository.buscarTodasLasPreguntasSeguridad();

}

}

**moduloSeguridad/services/ReestablecerClaveService**

package com.example.helpdesk2.moduloSeguridad.services;

import com.example.helpdesk2.models.Usuario;

import com.example.helpdesk2.DTO.ReestablecerClaveDTO;

import com.example.helpdesk2.DTO.ValidarDatosUsuarioDTO;

import com.example.helpdesk2.DTO.ValidarRptaSecretaDTO;

import com.example.helpdesk2.moduloSeguridad.exceptions.ClavesNoCoincidenException;

import com.example.helpdesk2.moduloSeguridad.exceptions.RespuestaSecretaIncorrectaException;

import com.example.helpdesk2.moduloSeguridad.exceptions.UsuarioNoEncontradoException;

import com.example.helpdesk2.repositories.PreguntaSeguridadRepository;

import com.example.helpdesk2.repositories.UsuarioRepository;

import org.springframework.stereotype.Service;

@Service

public class ReestablecerClaveService {

private final UsuarioRepository usuarioRepository;

private final PreguntaSeguridadRepository preguntaSeguridadRepository;

public ReestablecerClaveService(UsuarioRepository usuarioRepository, PreguntaSeguridadRepository preguntaSeguridadRepository) {

this.usuarioRepository = usuarioRepository;

this.preguntaSeguridadRepository = preguntaSeguridadRepository;

}

public Usuario validarUsuarioPorDatos(ValidarDatosUsuarioDTO validarDatosUsuarioDTO) {

return usuarioRepository.buscarUsuarioPorDatos(validarDatosUsuarioDTO)

.orElseThrow(UsuarioNoEncontradoException::new);

}

public String obtenerPreguntaSeguridad(int idPreguntaSeguridad) {

return preguntaSeguridadRepository.buscarPreguntaSeguridad(idPreguntaSeguridad);

}

public void validarRespuestaSecreta(ValidarRptaSecretaDTO validarRptaSecretaDTO) {

if (!usuarioRepository.buscarRptaSecreta(validarRptaSecretaDTO).equals(validarRptaSecretaDTO.getRptaSecreta())) {

throw new RespuestaSecretaIncorrectaException();

}

}

public void reestablecerClave(ReestablecerClaveDTO reestablecerClaveDTO) {

if (reestablecerClaveDTO.getClave().equals(reestablecerClaveDTO.getReClave())) {

usuarioRepository.actualizarClave(reestablecerClaveDTO);

} else {

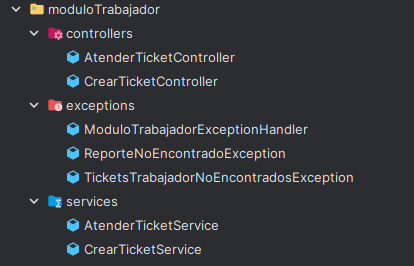
throw new ClavesNoCoincidenException();

}

}

}

moduloTrabajador



**moduloTrabajador/controllers/AtenderTicketController**

package com.example.helpdesk2.moduloTrabajador.controllers;

import com.example.helpdesk2.DTO.ActividadTicketDTO;

import com.example.helpdesk2.DTO.RegistrarMensajeTicketDTO;

import com.example.helpdesk2.DTO.TicketAsignadoDTO;

import com.example.helpdesk2.models.Mensaje;

import com.example.helpdesk2.moduloTrabajador.services.AtenderTicketService;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Map;

@RestController

@RequestMapping("/api/modulo-trabajador")

public class AtenderTicketController {

private final AtenderTicketService atenderTicketService;

private final LoggedUserManagamentService loggedUserManagamentService;

public AtenderTicketController(AtenderTicketService atenderTicketService, LoggedUserManagamentService loggedUserManagamentService) {

this.atenderTicketService = atenderTicketService;

this.loggedUserManagamentService = loggedUserManagamentService;

}

@GetMapping("/tickets")

public ResponseEntity<List<TicketAsignadoDTO>> obtenerTickets() {

int idUsuario = loggedUserManagamentService.getIdUsuario();

int idTrabajador = atenderTicketService.obtenerIdTrabajador(idUsuario);

List<TicketAsignadoDTO> ticketsTrabajador = atenderTicketService.obtenerTicketsTrabajador(idTrabajador);

return ResponseEntity

.ok()

.body(ticketsTrabajador);

}

@GetMapping("/tickets/{idTicket}")

public ResponseEntity<Map<String, Object>> obtenerTicketPorID(@PathVariable int idTicket) {

Map<String, Object> respuesta = atenderTicketService.obtenerDetallesTicketPorID(idTicket);

return ResponseEntity

.ok()

.body(respuesta);

}

@PostMapping("/tickets/mensaje")

public ResponseEntity<Void> registrarMensajeTicket(@RequestBody RegistrarMensajeTicketDTO registrarMensajeTicketDTO) {

atenderTicketService.registrarMensajeTicket(registrarMensajeTicketDTO);

return ResponseEntity

.ok()

.build();

}

@PostMapping("/tickets/cambiar-estado-ticket")

public ResponseEntity<Void> cambiarEstadoTicket(@RequestBody ActividadTicketDTO actividadTicketDTO) {

atenderTicketService.atenderTicket(actividadTicketDTO);

return ResponseEntity

.ok()

.build();

}

@GetMapping("/tickets/{idTicket}/mensajes")

public ResponseEntity<List<Mensaje>> obtenerMensajesTicket(@PathVariable int idTicket) {

List<Mensaje> mensajesTicket = atenderTicketService.obtenerMensajesTicket(idTicket);

return ResponseEntity

.ok()

.body(mensajesTicket);

}

}

**moduloTrabajador/controllers/CrearTicketController**

package com.example.helpdesk2.moduloTrabajador.controllers;

import com.example.helpdesk2.DTO.EscogerTicketDTO;

import com.example.helpdesk2.DTO.ObtenerReporteIncidenteImagenDTO;

import com.example.helpdesk2.models.ReporteIncidente;

import com.example.helpdesk2.moduloTrabajador.services.CrearTicketService;

import com.example.helpdesk2.repositories.TrabajadorRepository;

import com.example.helpdesk2.services.FileStorageService;

import com.example.helpdesk2.services.LoggedUserManagamentService;

import org.springframework.core.io.Resource;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

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

import java.io.IOException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@RestController

@RequestMapping("/api/modulo-trabajador")

public class CrearTicketController {

private final LoggedUserManagamentService loggedUserManagamentService;

private final CrearTicketService crearTicketService;

private final FileStorageService fileStorageService;

private final TrabajadorRepository trabajadorRepository;

public CrearTicketController(LoggedUserManagamentService loggedUserManagamentService, CrearTicketService crearTicketService, FileStorageService fileStorageService, TrabajadorRepository trabajadorRepository) {

this.loggedUserManagamentService = loggedUserManagamentService;

this.crearTicketService = crearTicketService;

this.fileStorageService = fileStorageService;

this.trabajadorRepository = trabajadorRepository;

}

@GetMapping("/reportes-incidentes")

public ResponseEntity<Map<String, Object>> obtenerReportesIncidentes() {

List<ReporteIncidente> reportesIncidentes = crearTicketService.obtenerReportesIncidentes();

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

respuesta.put("reportesIncidentes", reportesIncidentes);

return ResponseEntity.status(HttpStatus.OK).body(respuesta);

}

@GetMapping("/reportes-incidentes/{idReporteIncidente}")

public ResponseEntity<Map<String, Object>> obtenerReporteIncidente(@PathVariable int idReporteIncidente) {

Map<String, Object> respuesta = crearTicketService.obtenerReporteIncidente(idReporteIncidente);

return ResponseEntity.status(HttpStatus.OK).body(respuesta);

}

@PostMapping("/reporte-incidente-imagen")

public ResponseEntity<?> obtenerReporteIncidenteImagen(@RequestBody ObtenerReporteIncidenteImagenDTO obtenerReporteIncidenteImagenDTO) throws IOException {

Resource resource = fileStorageService.cargar(obtenerReporteIncidenteImagenDTO.getRutaImagen(), obtenerReporteIncidenteImagenDTO.getIdReporteIncidente());

String contentType = "application/octet-stream";

String headerValue = "attachment; filename=\"" + resource.getFilename() + "\"";

return ResponseEntity.ok()

.contentType(MediaType.parseMediaType(contentType))

.header(HttpHeaders.CONTENT\_DISPOSITION, headerValue)

.body(resource);

}

@PostMapping("/crear-ticket")

public ResponseEntity<Map<String, Object>> escogerReporteIncidente(@RequestBody EscogerTicketDTO escogerTicketDTO) {

int idUsuario = loggedUserManagamentService.getIdUsuario();

int idTrabajador = trabajadorRepository.buscarTrabajadorPorIdUsuario(idUsuario).getIdTrabajador();

crearTicketService.escogerReporteIncidente(escogerTicketDTO, idTrabajador);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", true);

return ResponseEntity.status(HttpStatus.OK).body(respuesta);

}

}

**moduloTrabajador/exceptions/ModuloTrabajadorExceptionHadler**

package com.example.helpdesk2.moduloTrabajador.exceptions;

import com.example.helpdesk2.exceptions.ErrorDetalles;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;

import java.util.Map;

@RestControllerAdvice

public class ModuloTrabajadorExceptionHandler {

@ExceptionHandler(ReporteNoEncontradoException.class)

public ResponseEntity<Map<String, Object>> handleReporteNoEncontrado(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("No se encontró reporte");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.BAD\_REQUEST)

.body(respuesta);

}

@ExceptionHandler(TicketsTrabajadorNoEncontradosException.class)

public ResponseEntity<Map<String, Object>> handleTicketsTrabajadorNoEncontrados(){

ErrorDetalles errorDetalles = new ErrorDetalles();

errorDetalles.setMensaje("No se encontraron tickets");

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ok", false);

respuesta.put("error", errorDetalles);

return ResponseEntity

.status(HttpStatus.BAD\_REQUEST)

.body(respuesta);

}

}

**moduloTrabajador/exceptions/ReporteNoEncontradoException**

package com.example.helpdesk2.moduloTrabajador.exceptions;

public class ReporteNoEncontradoException extends RuntimeException{

}

**moduloTrabajador/exceptions/TicketsTrabajadorNoEncontradosException**

package com.example.helpdesk2.moduloTrabajador.exceptions;

public class TicketsTrabajadorNoEncontradosException extends RuntimeException{

}

**moduloTrabajador/services/AtenderTicketService**

package com.example.helpdesk2.moduloTrabajador.services;

import com.example.helpdesk2.DTO.DatosClienteDTO;

import com.example.helpdesk2.DTO.ActividadTicketDTO;

import com.example.helpdesk2.DTO.RegistrarMensajeTicketDTO;

import com.example.helpdesk2.DTO.TicketAsignadoDTO;

import com.example.helpdesk2.models.Imagen;

import com.example.helpdesk2.models.Mensaje;

import com.example.helpdesk2.moduloTrabajador.exceptions.TicketsTrabajadorNoEncontradosException;

import com.example.helpdesk2.repositories.\*;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Service

public class AtenderTicketService {

private final TicketRepository ticketRepository;

private final TrabajadorRepository trabajadorRepository;

private final ClienteRepository clienteRepository;

private final ImagenRepository imagenRepository;

private final MensajeRepository mensajeRepository;

private final MensajeTicketRepository mensajeTicketRepository;

private final ActividadTicketRepository actividadTicketRepository;

private final ReporteIncidenteRepository reporteIncidenteRepository;

public AtenderTicketService(TicketRepository ticketRepository,

TrabajadorRepository trabajadorRepository,

ClienteRepository clienteRepository,

ImagenRepository imagenRepository,

MensajeRepository mensajeRepository,

MensajeTicketRepository mensajeTicketRepository,

ActividadTicketRepository actividadTicketRepository,

ReporteIncidenteRepository reporteIncidenteRepository) {

this.ticketRepository = ticketRepository;

this.trabajadorRepository = trabajadorRepository;

this.clienteRepository = clienteRepository;

this.imagenRepository = imagenRepository;

this.mensajeRepository = mensajeRepository;

this.mensajeTicketRepository = mensajeTicketRepository;

this.actividadTicketRepository = actividadTicketRepository;

this.reporteIncidenteRepository = reporteIncidenteRepository;

}

public List<TicketAsignadoDTO> obtenerTicketsTrabajador(int idTrabajador) {

return ticketRepository.buscarTicketsTrabajador(idTrabajador).orElseThrow(TicketsTrabajadorNoEncontradosException::new);

}

public int obtenerIdTrabajador(int idUsuario) {

return trabajadorRepository.buscarTrabajadorPorIdUsuario(idUsuario).getIdTrabajador();

}

@Transactional

public Map<String, Object> obtenerDetallesTicketPorID(int idTicket) {

TicketAsignadoDTO ticket = ticketRepository.buscarTicketPorID(idTicket);

DatosClienteDTO datosCliente = clienteRepository.buscarDatosCliente(ticket.getIdCliente());

List<Imagen> rutasImagenes = imagenRepository.buscarImagenesReporte(ticket.getIdReporteIncidente());

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("ticket", ticket);

respuesta.put("datosCliente", datosCliente);

respuesta.put("rutasImagenes", rutasImagenes);

return respuesta;

}

@Transactional

public void registrarMensajeTicket(RegistrarMensajeTicketDTO registrarMensajeTicketDTO) {

mensajeRepository.guardarMensaje(registrarMensajeTicketDTO);

int idMensaje = mensajeRepository.obtenerUltimoIdInsertado();

mensajeTicketRepository.guardarMensajeTicket(registrarMensajeTicketDTO.getIdTicket(), idMensaje);

}

@Transactional

public void atenderTicket(ActividadTicketDTO actividadTicketDTO) {

actividadTicketRepository.guardarActividadTicket(actividadTicketDTO.getIdTicket(), actividadTicketDTO.getMensaje(), actividadTicketDTO.getEstado());

ticketRepository.actualizarEstadoTicket(actividadTicketDTO.getEstado(), actividadTicketDTO.getIdTicket());

reporteIncidenteRepository.actualizarEstadoReporteIncidente(actividadTicketDTO.getIdReporteIncidente(), actividadTicketDTO.getEstado());

}

public List<Mensaje> obtenerMensajesTicket(int idTicket) {

return mensajeRepository.buscarMensajesTicket(idTicket);

}

}

**moduloTrabajador/services/CrearTicketService**

package com.example.helpdesk2.moduloTrabajador.services;

import com.example.helpdesk2.DTO.EscogerTicketDTO;

import com.example.helpdesk2.models.Imagen;

import com.example.helpdesk2.models.ReporteIncidente;

import com.example.helpdesk2.moduloTrabajador.exceptions.ReporteNoEncontradoException;

import com.example.helpdesk2.repositories.ImagenRepository;

import com.example.helpdesk2.repositories.ReporteIncidenteRepository;

import com.example.helpdesk2.repositories.TicketRepository;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Service

public class CrearTicketService {

private final ReporteIncidenteRepository reporteIncidenteRepository;

private final ImagenRepository imagenRepository;

private final TicketRepository ticketRepository;

public CrearTicketService(ReporteIncidenteRepository reporteIncidenteRepository, ImagenRepository imagenRepository, TicketRepository ticketRepository) {

this.reporteIncidenteRepository = reporteIncidenteRepository;

this.imagenRepository = imagenRepository;

this.ticketRepository = ticketRepository;

}

public List<ReporteIncidente> obtenerReportesIncidentes() {

return reporteIncidenteRepository.buscarReportesIncidentes().orElseThrow(ReporteNoEncontradoException::new);

}

public Map<String, Object> obtenerReporteIncidente(int idReporteIncidente) {

List<Imagen> rutasImagenes = imagenRepository.buscarImagenesReporte(idReporteIncidente);

ReporteIncidente reporteIncidente = reporteIncidenteRepository.buscarReporteIncidente(idReporteIncidente);

Map<String, Object> respuesta = new HashMap<>();

respuesta.put("reporteIncidente", reporteIncidente);

respuesta.put("rutasImagenes", rutasImagenes);

return respuesta;

}

@Transactional

public void escogerReporteIncidente(EscogerTicketDTO escogerTicketDTO, int idTrabajador) {

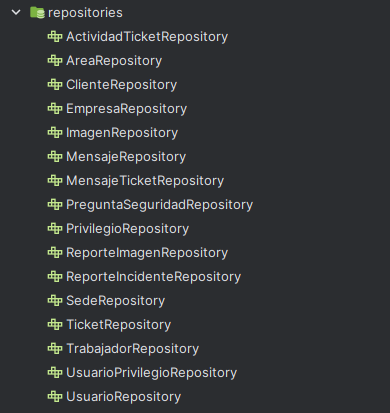
reporteIncidenteRepository.actualizarEstadoReporteIncidente(escogerTicketDTO.getIdReporteIncidente(), escogerTicketDTO.getEstado());

ticketRepository.guardarTicket(idTrabajador, escogerTicketDTO.getIdReporteIncidente());

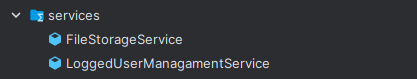
}

}

repositories



services



services/FileStorageService

package com.example.helpdesk2.services;

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

@Service

public class FileStorageService {

private final Path root = Paths.get("src/main/resources/static/reportes");

public Path crearDirectorio(int idReporteIncidente) throws IOException {

String nombreDirectorio = "reporte-" + idReporteIncidente;

Path reporteFolder = root.resolve(nombreDirectorio);

Files.createDirectories(reporteFolder);

return reporteFolder;

}

public String guardar(MultipartFile imagen, Path reporteFolder) throws IOException {

long timestamp = System.currentTimeMillis();

String nombreImagen = timestamp + "-" + imagen.getOriginalFilename();

Files.copy(imagen.getInputStream(), reporteFolder.resolve(nombreImagen), StandardCopyOption.REPLACE\_EXISTING);

return nombreImagen;

}

public Resource cargar(String archivo, int idReporte) throws IOException {

String nombreDirectorio = "reporte-" + idReporte;

Path rootReporte = root.resolve(nombreDirectorio);

Path file = rootReporte.resolve(archivo);

return new UrlResource(file.toUri());

}

}

HelpDesk2Application

