**ПРИЛОЖЕНИЕ А**

(обязательное)

Листинг кода

com/epa/epadiplom/config/ApplicationConfig.java

package com.epa.epadiplom.config;

import com.epa.epadiplom.service.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.AuthenticationProvider;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

@RequiredArgsConstructor

public class ApplicationConfig {

private final UserService userService;

@Bean

public UserDetailsService userDetailsService(){

return userService::findUserByFirstName;

}

@Bean

public AuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();

authProvider.setUserDetailsService(userDetailsService());

authProvider.setPasswordEncoder(passwordEncoder());

return authProvider;

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config)

throws Exception {

return config.getAuthenticationManager();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

com/epa/epadiplom/config/JwtAuthenticationFilter.java

package com.epa.epadiplom.config;

import com.epa.epadiplom.service.JwtService;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import lombok.RequiredArgsConstructor;

import org.springframework.lang.NonNull;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component

@RequiredArgsConstructor

public class JwtAuthenticationFilter extends OncePerRequestFilter {

private final JwtService jwtService;

private final UserDetailsService userDetailsService;

@Override

protected void doFilterInternal(@NonNull HttpServletRequest request, @NonNull HttpServletResponse response, @NonNull FilterChain filterChain) throws ServletException, IOException {

final String authHeader = request.getHeader("Authorization");

final String jwToken;

final String login;

if (authHeader == null || !authHeader.startsWith("Bearer")) {

filterChain.doFilter(request, response);

return;

}

jwToken = authHeader.substring(7);

login = jwtService.extractUsername(jwToken);

if(login != null &&

SecurityContextHolder.getContext().getAuthentication() == null){

UserDetails userDetails = this.userDetailsService.loadUserByUsername(login);

if(jwtService.isTokenValid(jwToken, userDetails)){

UsernamePasswordAuthenticationToken authenticationToken =

new UsernamePasswordAuthenticationToken(

userDetails,

null,

userDetails.getAuthorities());

authenticationToken.setDetails(

new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authenticationToken);

}

}

filterChain.doFilter(request, response);

}

}

com/epa/epadiplom/config/SecurityConfig.java

package com.epa.epadiplom.config;

import lombok.RequiredArgsConstructor;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationProvider;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import org.springframework.security.web.session.SessionManagementFilter;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.CorsConfigurationSource;

import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import org.springframework.web.filter.CorsFilter;

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

import java.util.Arrays;

import java.util.List;

@Configuration

@EnableWebSecurity(debug = true)

@RequiredArgsConstructor

public class SecurityConfig implements WebMvcConfigurer {

private final JwtAuthenticationFilter jwtAuthenticationFilter;

private final AuthenticationProvider authenticationProvider;

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {

httpSecurity

.cors(Customizer.withDefaults())

.and()

.addFilterBefore(new CorsFilter(corsConfigurationSource()), SessionManagementFilter.class)

.httpBasic().disable()

.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/\*\*")

.permitAll()

.anyRequest()

.authenticated()

.and()

.sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS)

.and()

.authenticationProvider(authenticationProvider)

.addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

return httpSecurity.build();

}

@Bean

CorsConfigurationSource corsConfigurationSource() {

CorsConfiguration configuration = new CorsConfiguration();

configuration.setAllowedOrigins(List.of("http://localhost:3000"));

configuration.setAllowedMethods(Arrays.asList("GET","POST"));

UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

source.registerCorsConfiguration("/\*\*", configuration);

return source;

}

}

com/epa/epadiplom/controller/AuthenticationController.java

package com.epa.epadiplom.controller;

import com.epa.epadiplom.entity.authentication.AuthenticationRequest;

import com.epa.epadiplom.entity.authentication.AuthenticationResponse;

import com.epa.epadiplom.entity.authentication.RegisterRequest;

import com.epa.epadiplom.service.AuthenticationService;

import lombok.RequiredArgsConstructor;

import org.springframework.http.ResponseEntity;

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

@RestController

@RequestMapping("/auth")

@RequiredArgsConstructor

public class AuthenticationController {

private final AuthenticationService authenticationService;

@PostMapping("/register")

public ResponseEntity<AuthenticationResponse> register (

@RequestBody RegisterRequest request){

return ResponseEntity.ok(authenticationService.register(request));

}

@PostMapping("/authenticate")

public ResponseEntity<AuthenticationResponse> authenticate (

@RequestBody AuthenticationRequest request){

return ResponseEntity.ok(authenticationService.authenticate(request));

}

}

com/epa/epadiplom/controller/MainPageController.java

package com.epa.epadiplom.controller;

import com.epa.epadiplom.entity.\*;

import com.epa.epadiplom.entity.authentication.\*;

import com.epa.epadiplom.service.\*;

import lombok.RequiredArgsConstructor;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

import org.springframework.security.core.Authentication;

import org.springframework.security.crypto.password.PasswordEncoder;

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

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

import java.util.List;

@RestController

@EnableMethodSecurity

@EnableWebMvc

@RequiredArgsConstructor

public class MainPageController {

private final EmployeesViewService employeesViewService;

private final EmployeeFullViewService employeeFullViewService;

private final LogStatementsViewService logStatementsViewService;

private final EventsViewService eventsViewService;

private final TasksViewService tasksViewService;

private final UserService userService;

private final LogStatementService logStatementService;

private final EventService eventService;

private final NoticeEventService noticeEventService;

private final TaskService taskService;

private final EmployeeTaskService employeeTaskService;

private final PasswordEncoder passwordEncoder;

private final DocumentService documentService;

private final EmployeeService employeeService;

private final PersonalService personalService;

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Request to see the main info of authorized employee page

@GetMapping(path = "/mainUserInfo", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<List<EmployeeFullView>> getEmployeeInfo(Authentication authentication) {

return ResponseEntity.ok(employeeFullViewService.findAllByLoginUser(authentication.getName()));

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Request to see the statements that need to approve

@GetMapping(path = "/ls", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<List<LogStatementsView>> getLsRequests(Authentication authentication) {

return ResponseEntity.ok(this.logStatementsViewService.findAllByIdApproverAndStatus(

userService.findUserByFirstName(authentication.getName()).getIdLogin(), 3));

}

// Request to approve the statements

@PostMapping(path = "/ls/{id}", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> setLsApprove(

@PathVariable Long id,

Authentication authentication,

@RequestBody LogStatementRequest request

) {

LogStatement logStatement = logStatementService.findByIdAndIdApprover(

id,

userService.findUserByFirstName(authentication.getName()).getIdLogin()

).orElse(new LogStatement());

logStatement.setStatus(request.getStatus());

System.out.println(logStatement.getId() + " " +

logStatement.getStatus());

if(logStatementService.saveLogStatement(logStatement))

System.out.println("ok");

return ResponseEntity.ok("Done ");

}

// Request to see the events for authorized employee

@GetMapping(path = "/events", produces = MediaType.APPLICATION\_JSON\_VALUE)

public @ResponseBody List<EventsView> getEvents(Authentication authentication) {

return this.eventsViewService.findAllByIdRecipient(

userService.findUserByFirstName(authentication.getName()).getIdLogin());

}

// Request to see the tasks for authorized employee

@GetMapping(path = "/tasks", produces = MediaType.APPLICATION\_JSON\_VALUE)

public @ResponseBody List<TasksView> getTasks(Authentication authentication) {

System.out.println(authentication);

return this.tasksViewService.findAllByIdExecutor(

userService.findUserByFirstName(authentication.getName()).getIdLogin());

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Request for seeing list of all employees

@GetMapping(path = "/employees", produces = MediaType.APPLICATION\_JSON\_VALUE)

public @ResponseBody List<EmployeesView> getEmployees() {

return this.employeesViewService.findAll();

}

@PostMapping(path = "/event", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> createEvent(@RequestBody EventRequest request) {

Event event = Event.builder()

.dateOfEvent(request.getDateOfEvent())

.commentFe(request.getCommentFe())

.typeOfEvent(request.getTypeOfEvent())

.build();

if(eventService.saveEvent(event))

System.out.println("ok");

return ResponseEntity.ok("Done");

}

@PostMapping(path = "/noticeevent/{idevent}", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> createNoticeEvent(

@RequestBody NoticeEventRequest request,

Authentication authentication,

@PathVariable Long idevent

) {

NoticeEvent noticeEvent = NoticeEvent.builder()

.recipientId(request.getRecipientId())

.eventId(idevent)

.employeeId(userService.findUserByFirstName(authentication.getName()).getIdLogin())

.build();

System.out.println(noticeEvent.getRecipientId() + " " +

noticeEvent.getEventId() + " " +

noticeEvent.getEmployeeId());

if(noticeEventService.saveNoticeEvent(noticeEvent))

System.out.println("ok");

return ResponseEntity.ok("Done");

}

@PostMapping(path = "/task", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> createTask(@RequestBody TaskRequest request) {

Task task = Task.builder()

.dateTask(request.getDateTask())

.nameOfTask(request.getNameOfTask())

.idExecutor(request.getIdExecutor())

.commentTe(request.getCommentTe())

.build();

System.out.println(task.getId());

EmployeeTask employeeTask = EmployeeTask.builder()

.idTask(task.getId())

.idEmployee(request.getIdExecutor())

.build();

if (taskService.saveTask(task))

System.out.println("ok");

if (employeeTaskService.saveEmployeeTask(employeeTask))

System.out.println("ok");

return ResponseEntity.ok("Done");

}

@PostMapping(path = "/password", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> changePassword(@RequestBody LoginRequest request,

Authentication authentication) {

User user = userService.findUserByFirstName(authentication.getName());

user.setPassword(passwordEncoder.encode(request.getPassword()));

if (userService.saveUserPassword(user))

System.out.println("ok");

return ResponseEntity.ok("Done");

}

@PostMapping(path = "/lscreate", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> createLS( @RequestBody LogStatementCreateRequest request,

Authentication authentication

){

LogStatement logStatement = LogStatement.builder()

.id(0)

.idApprover(request.getIdApprover())

.status(3)

.idEmployee(userService.findUserByFirstName(authentication.getName()).getIdLogin())

.commentLs(request.getCommentLs())

.typeLeave(request.getTypeLeave())

.dateLeave(request.getDateLeave())

.dateOfLs(request.getDateOfLs())

.daysSum(request.getDaysSum())

.build();

if(logStatementService.saveLogStatementAll(logStatement))

System.out.println("ok");

if(request.getBodyDoc()!= null) {

Document document = Document.builder()

.bodyDoc(request.getBodyDoc())

.idLs(logStatement.getId())

.build();

if(documentService.saveDocument(document))

System.out.println("ok");

}

return ResponseEntity.ok("Done ");

}

@PostMapping(path = "/editemployee/{id}", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<String> edit( @RequestBody EditRequest request,

Authentication authentication,

@PathVariable Long id

){

Employee employee = Employee.builder()

.id(id)

.firstName(request.getFirstName())

.lastName(request.getLastName())

.middleName(request.getMiddleName())

.workNumber(request.getWorkNumber())

.locationStreet(request.getLocationStreet())

.cabinetOffice(request.getCabinetOffice())

.build();

if(employeeService.saveEmployee(Employee))

System.out.println("ok");

Personal personal = Personal.builder()

.birthD(request.getBirthD())

.idPersonal(id)

.personalNumber(request.getPersonalNumber())

.build();

if(personalService.savePersonal(Personal))

System.out.println("ok");

User user = User.builder()

.idLogin(id)

.role(request.getRole())

.build();

if(userService.saveUser(User))

System.out.println("ok");

return ResponseEntity.ok("Done ");

}

}

com/epa/epadiplom/entity/authentication/AuthenticationRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class AuthenticationRequest {

private String login;

String password;

}

com/epa/epadiplom/entity/authentication/AuthenticationResponse.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class AuthenticationResponse {

private String token;

}

com/epa/epadiplom/entity/authentication/EditRequest.java

package com.epa.epadiplom.entity.authentication;

import com.epa.epadiplom.entity.employeeAttributes.Role;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.sql.Date;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class EditRequest {

private String firstName;

private String middleName;

private String lastName;

private Date birthD;

private Role role;

private long workNumber;

private long personalNumber;

private String locationStreet;

private String cabinetOffice;

}

com/epa/epadiplom/entity/authentication/EventRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.\*;

import java.sql.Date;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class EventRequest {

private String typeOfEvent;

private String commentFe;

private Date dateOfEvent;

}

com/epa/epadiplom/entity/authentication/LoginRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class LoginRequest {

private String password;

}

com/epa/epadiplom/entity/authentication/LogStatementCreateRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.sql.Date;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class LogStatementCreateRequest {

private long idApprover;

private String commentLs;

private int daysSum;

private int typeLeave;

private Date dateLeave;

private Date dateOfLs;

private String bodyDoc;

}

com/epa/epadiplom/entity/authentication/LogStatementRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class LogStatementRequest {

private int status;

}

com/epa/epadiplom/entity/authentication/NoticeEventRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.\*;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class NoticeEventRequest {

private long recipientId;

}

com/epa/epadiplom/entity/authentication/RegisterRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class RegisterRequest {

private String firstName;

private String middleName;

private String lastName;

private String login;

private String mail;

private String password;

}

com/epa/epadiplom/entity/authentication/TaskRequest.java

package com.epa.epadiplom.entity.authentication;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.sql.Date;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class TaskRequest {

private Date dateTask;

private String nameOfTask;

private String commentTe;

private long idExecutor;

}

com/epa/epadiplom/entity/User.java

package com.epa.epadiplom.entity;

import com.epa.epadiplom.entity.employeeAttributes.Role;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.databind.annotation.JsonSerialize;

import jakarta.persistence.\*;

import lombok.\*;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;

import java.util.List;

@Builder

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

//To suppress serializing properties with null values

@JsonSerialize

//Ignoring new fields on JSON objects

@JsonIgnoreProperties(ignoreUnknown = true)

@Entity

@Table(name = "login", schema = "public", catalog = "EPA")

public class User implements UserDetails {

@Id

//@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id\_login")

private long idLogin;

@Column(name = "login\_user")

private String firstName;

@Column(name = "password\_user")

private String password;

@Column(name = "mail\_user")

private String mail;

@Enumerated(EnumType.STRING)

private Role role;

@OneToOne

@JoinColumn(name = "id\_login")

private Employee employee;

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

return List.of(new SimpleGrantedAuthority(role.name()));

}

@Override

public String getUsername() {

return firstName;

}

@Override

public boolean isAccountNonExpired() {

return true;

}

@Override

public boolean isAccountNonLocked() {

return true;

}

@Override

public boolean isCredentialsNonExpired() {

return true;

}

@Override

public boolean isEnabled() {

return true;

}

}

com/epa/epadiplom/service/AuthenticationService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.Employee;

import com.epa.epadiplom.entity.User;

import com.epa.epadiplom.entity.authentication.AuthenticationRequest;

import com.epa.epadiplom.entity.authentication.AuthenticationResponse;

import com.epa.epadiplom.entity.authentication.RegisterRequest;

import com.epa.epadiplom.entity.employeeAttributes.Role;

import com.epa.epadiplom.repository.EmployeeRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class AuthenticationService {

private final UserService userService;

private final EmployeeService employeeService;

private final PasswordEncoder passwordEncoder;

private final JwtService jwtService;

private final AuthenticationManager authenticationManager;

private final EmployeeRepository employeeRepository;

public AuthenticationResponse register(RegisterRequest request) {

Employee employee = employeeRepository

.findByFirstNameAndMiddleNameAndLastName(

request.getFirstName(),

request.getMiddleName(),

request.getLastName()).orElse(new Employee());

var user = User.builder()

.idLogin(employee.getId())

.firstName(request.getLogin())

.mail(request.getMail())

.password(passwordEncoder.encode(request.getPassword()))

.role(Role.USER)

.build();

userService.saveUser(user);

var jwtToken = jwtService.generateToken(user);

return AuthenticationResponse.builder()

.token(jwtToken)

.build();

}

public AuthenticationResponse authenticate(AuthenticationRequest request) {

authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(request.getLogin(), request.getPassword()));

var user = userService.findUserByFirstName(request.getLogin());

var jwtToken = jwtService.generateToken(user);

return AuthenticationResponse.builder()

.token(jwtToken)

.build();

}

}

com/epa/epadiplom/service/DepartmentService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.repository.DepartmentRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class DepartmentService {

private final DepartmentRepository departmentRepository;

}

com/epa/epadiplom/service/DocumentService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.Document;

import com.epa.epadiplom.repository.DocumentRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class DocumentService {

private final DocumentRepository documentRepository;

public boolean saveDocument(Document document){

if(documentRepository.findById(document.getId()).isPresent()) {

return false;

}

documentRepository.save(document);

return true;

}

}

com/epa/epadiplom/service/EmployeeFullViewService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.EmployeeFullView;

import com.epa.epadiplom.repository.EmployeeFullViewRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@RequiredArgsConstructor

public class EmployeeFullViewService {

private final EmployeeFullViewRepository employeeFullViewRepository;

public List<EmployeeFullView> findAllByLoginUser(String loginUser){

return employeeFullViewRepository.findAllByLoginUser(loginUser);

}

}

com/epa/epadiplom/service/EmployeeService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.Employee;

import com.epa.epadiplom.repository.EmployeeRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class EmployeeService {

private final EmployeeRepository employeeRepository;

public boolean saveEmployee(Employee employee) {

if(employeeRepository.findById(employee.getId()).isPresent()) {

employeeRepository.save(employee);

return true;

}

return false;

}

}

com/epa/epadiplom/service/EmployeesViewService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.EmployeesView;

import com.epa.epadiplom.repository.EmployeesViewRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@RequiredArgsConstructor

public class EmployeesViewService {

private final EmployeesViewRepository employeesViewRepository;

public List <EmployeesView> findAll(){

return employeesViewRepository.findAll();

}

}

com/epa/epadiplom/service/EmployeeTaskService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.EmployeeTask;

import com.epa.epadiplom.repository.EmployeeTaskRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class EmployeeTaskService {

private final EmployeeTaskRepository employeeTaskRepository;

public boolean saveEmployeeTask (EmployeeTask employeeTask){

if(employeeTaskRepository.findById(employeeTask.getId()).isPresent())

return false;

employeeTaskRepository.save(employeeTask);

return true;

}

}

com/epa/epadiplom/service/EventService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.Event;

import com.epa.epadiplom.repository.EventRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class EventService {

private final EventRepository eventRepository;

public boolean saveEvent (Event event){

if(eventRepository.findById(event.getId()).isPresent())

return false;

eventRepository.save(event);

return true;

}

}

com/epa/epadiplom/service/EventsViewService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.EventsView;

import com.epa.epadiplom.repository.EventsViewRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@RequiredArgsConstructor

public class EventsViewService {

private final EventsViewRepository eventsViewRepository;

public List<EventsView> findAllByIdRecipient(long idLogin) {

return eventsViewRepository.findAllByIdRecipient(idLogin);

}

}

com/epa/epadiplom/service/JobEmployeeService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.repository.JobEmployeeRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class JobEmployeeService {

private final JobEmployeeRepository jobEmployeeRepository;

}

com/epa/epadiplom/service/JobTitleService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.repository.JobTitleRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class JobTitleService {

private final JobTitleRepository jobTitleRepository;

}

com/epa/epadiplom/service/JobTitleViewService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.repository.JobTitleViewRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class JobTitleViewService {

private final JobTitleViewRepository jobTitleViewRepository;

}

com/epa/epadiplom/service/JwtService.java

package com.epa.epadiplom.service;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.io.Decoders;

import io.jsonwebtoken.security.Keys;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Service;

import java.security.Key;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

@Service

public class JwtService {

private static final String SECRET\_KEY = "77217A25432A462D4A614E645267556B58703273357538782F413F4428472B4B";

public String extractUsername(String jwToken) {

return extractClaim(jwToken, Claims::getSubject);

}

public <T> T extractClaim(String jwToken, Function<Claims, T> claimsResolver){

final Claims claims = extractAllClaims(jwToken);

return claimsResolver.apply(claims);

}

public String generateToken(

Map<String, Object> extraClaims,

UserDetails userDetails

){

return Jwts

.builder()

.setClaims(extraClaims)

.setSubject(userDetails.getUsername())

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1200 \* 60 \* 24))

.signWith(getSignInKey(), SignatureAlgorithm.HS256)

.compact();

}

public String generateToken(UserDetails userDetails){

return generateToken(new HashMap<>(), userDetails);

}

public boolean isTokenValid (String jwToken, UserDetails userDetails){

final String username = extractUsername(jwToken);

return (username.equals(userDetails.getUsername())) &&

!isTokenExpired(jwToken);

}

private boolean isTokenExpired(String jwToken) {

return extractExpiration(jwToken).before(new Date());

}

private Date extractExpiration(String jwToken) {

return extractClaim(jwToken, Claims::getExpiration);

}

private Claims extractAllClaims(String jwToken){

return Jwts

.parserBuilder()

.setSigningKey(getSignInKey())

.build()

.parseClaimsJws(jwToken)

.getBody();

}

private Key getSignInKey() {

byte[] keyBytes = Decoders.BASE64.decode(SECRET\_KEY);

return Keys.hmacShaKeyFor(keyBytes);

}

}

com/epa/epadiplom/service/LogStatementService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.LogStatement;

import com.epa.epadiplom.repository.LogStatementRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.Optional;

@Service

@RequiredArgsConstructor

public class LogStatementService {

private final LogStatementRepository logStatementRepository;

public boolean saveLogStatement(LogStatement logStatement) {

if(logStatementRepository.findById(logStatement.getId()).isPresent()) {

logStatementRepository.save(logStatement);

return true;

}

return false;

}

public boolean saveLogStatementAll(LogStatement logStatement){

if(logStatementRepository.findById(logStatement.getId()).isPresent()) {

return false;

}

logStatementRepository.save(logStatement);

return true;

}

public Optional<LogStatement> findByIdAndIdApprover (long id, long idApprover){

return logStatementRepository.findByIdAndIdApprover(id, idApprover);

}

}

com/epa/epadiplom/service/LogStatementsViewService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.LogStatementsView;

import com.epa.epadiplom.repository.LogStatementsViewRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@RequiredArgsConstructor

public class LogStatementsViewService {

private final LogStatementsViewRepository logStatementsViewRepository;

public List<LogStatementsView> findAllByIdApproverAndStatus(long idApprover, int status){

return logStatementsViewRepository.findAllByIdApproverAndStatus(idApprover, status);

}

}

com/epa/epadiplom/service/NoticeEventService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.NoticeEvent;

import com.epa.epadiplom.repository.NoticeEventRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class NoticeEventService {

private final NoticeEventRepository noticeEventRepository;

public boolean saveNoticeEvent (NoticeEvent noticeEvent){

if(noticeEventRepository.findById(noticeEvent.getId()).isPresent())

return false;

noticeEventRepository.save(noticeEvent);

return true;

}

}

com/epa/epadiplom/service/PersonalService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.Personal;

import com.epa.epadiplom.repository.PersonalRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class PersonalService {

private final PersonalRepository personalRepository;

public boolean savePersonal(Personal personal) {

if(personalRepository.findById(personal.getIdPersonal())) {

personalRepository.save(personal);

return true;

}

return false;

}

}

com/epa/epadiplom/service/TaskService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.Event;

import com.epa.epadiplom.entity.Task;

import com.epa.epadiplom.repository.TaskRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class TaskService {

private final TaskRepository taskRepository;

public boolean saveTask (Task task){

if(taskRepository.findById(task.getId()).isPresent())

return false;

taskRepository.save(task);

return true;

}

}

com/epa/epadiplom/service/TasksViewService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.TasksView;

import com.epa.epadiplom.repository.TasksViewRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@RequiredArgsConstructor

public class TasksViewService {

private final TasksViewRepository tasksViewRepository;

public List<TasksView> findAllByIdExecutor(long idLogin) {

return tasksViewRepository.findAllByIdExecutor(idLogin);

}

}

com/epa/epadiplom/service/UserService.java

package com.epa.epadiplom.service;

import com.epa.epadiplom.entity.User;

import com.epa.epadiplom.repository.UserRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

@RequiredArgsConstructor

public class UserService {

private final UserRepository userRepository;

public User findUserByFirstName(String login) {

return userRepository.findByFirstName(login).orElseThrow();

}

public List<User> allUsers() {

return userRepository.findAll();

}

public boolean saveUser(User user) {

if(userRepository.findByFirstName(user.getFirstName()).isPresent())

return false;

userRepository.save(user);

return true;

}

public boolean saveUserPassword(User user) {

if(userRepository.findByFirstName(user.getFirstName()).isPresent()) {

userRepository.save(user);

return true;

}

return false;

}

}