Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределённые системы обработки информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры ЭИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Лыщик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Разработка системы автоматизации работы санатория»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Лянной Вадим Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**ВВЕДЕНИЕ**

В данной работе будет рассматриваться разработка системы автоматизации деятельности санаторно-оздоровительного учреждения. Данная тема является достаточно актуальной, т.к. санаторий охватывает множество аспектов экономической деятельности. Специфика работы санатория — совмещение отдыха с лечением. Его инфраструктура чрезвычайно развита, а сотрудники вынуждены иметь дело с огромными объемами типовой информации. Службы сервиса, медицинских услуг, питания в санатории испытывают в современных условиях острую необходимость в автоматизации. Поэтому является необходимым проводить комплексную автоматизацию.

Внедрение АСУ в данную область позволит значительно систематизировать обмен данными, регламентировать состав и формы представления данных, а также структуру информационных потоков в системе (информационных и командных связей между субъектами санатория, а также информационный обмен с внешними по отношению к санаторию организациями). Также значительно повысить точность и четкость ведения документации, гарантировать сохранность, предоставлять полную взаимоувязанную информацию по всем субъектам санатория. Все это должно привести к слаженной работе сотрудников данного учреждения и как следствие повысить эффективность функционирования предприятия в целом.

Данный курсовой проект состоит из 4 частей (глав). Первая глава описывает общее содержание предметной области, организационную структуру предприятия в целом, раскрывает особенности отдельных подразделений. Во второй главе представлена структура самой автоматизированной системы управления санаторным комплексом, архитектура подразделений и АРМ сотрудников, работающих на данном предприятии. Третья часть курсового проекта содержит конкретный пример разработанного программного обеспечения. Четвертая часть содержит инструкции по использованию программного средства.

# **Анализ и моделирование предметной области программного средства**

## **Описание предметной области**

В данном курсовом проекте в качестве предметной области рассматривается - санаторий. Санаторий представляет собой лечебно-профилактическое учреждение, в котором осуществляется деятельность оздоровительного характера. Данное учреждение характеризуется представлением разного рода услуг: медицинских, развлекательных, культурных и т.д. Возрастной диапазон пациентов включает все возрасты от детей 3-х лет до пенсионеров. В санатории отдыхающие могут принять лечебные процедуры и получить специальное диетическое питание, посещать спортивные залы, бассейн и другие развлекательные мероприятия.

Внутреннюю структуру рассматриваемого санаторного комплекса можно представить состоящей из следующих компонентов:

* Приемное отделение (регистратура) – осуществляет регистрацию прибывших клиентов и распределение их по комнатам;
* Жилой комплекс – совокупность жилых корпусов и отделов, которые обеспечивают проживание пациентов в санатории (ведение информации о свободных и занятых номерах, подбор номера и расселение пациентов);
* Лечебный комплекс – обеспечивает проведение лечебных процедур для пациентов (проведение обследования пациентов, назначение соответствующего поставленному диагнозу лечения, снабжение пациентов необходимыми им лекарственными препаратами);
* Комплекс питания – обеспечивает пациентов необходимым для каждого питанием (обследование пациента врачом-диетологом, выбор системы питания и диеты, назначение соответствующего рациона питания).
* Культурно-развлекательный комплекс – занимается культурно-развлекательными программами для пациентов санатория (кино, экскурсии и т.д.).

Главный врач

Отдел снабжения

Отдел кадров

Финансовый отдел

Отдел работы с клиентами

Отдел работы с клиентами

Культ. - развлекат.блок

Диет-блок

Лечебный блок

Приемное отделение

Врач-специалист

Лаборатория

Жилой комплекс

Лечащий врач

Медицинская сестра

Отдел кадров

Зав. отделом

Блок-учет работников

Финансовый блок

Отдел реализации путевок

Бухгалтерия

Зав. отделом

Главный бухгалтер

Рисунок 1.1 – схема структуры санатория

## **Разработка функциональной модели предметной области**

Построение модели информационной системы в системе моделирования начинается с описания функционирования системы в целом для того, чтобы конкретно определить входящую и исходящую информацию. На первом уровне представлена общая картина работы санатория. Здесь указаны входные и выходные данные.

## **Анализ требований к разрабатываемому программному средству.**

Основные требования к приложению:

* заселение в санаторий;
* запись на прием к врачу;
* заказ трансфера в санаторий;
* заказ еды из столовой санатория;
* возможность записаться на реабилитацию после операций;
* регистрация пользователей (администратора, посетителей, врачей);
* пользователь (Администратор) может вносить, изменять и удалять данные;
* обычный пользователь может работать с приложением только в режиме просмотра, может оставлять заявку на бронирование онлайн.

## **Разработка информационной модели предметной области**

Для хранения данных, используемых в приложении, применяется база данных PostgresSQL. Схема данных приведена на рисунке 1.8

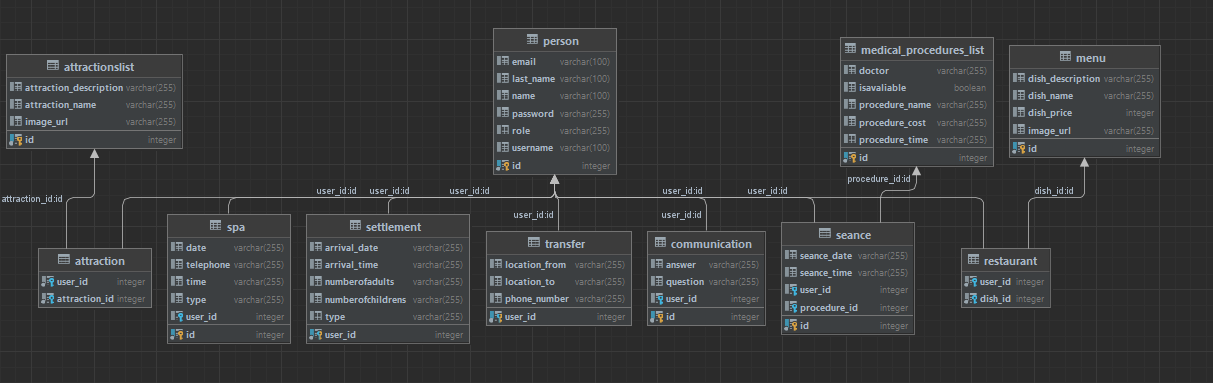
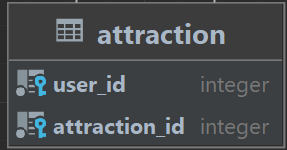


Рисунок. 1.8 – Схема данных

Для решения поставленной задачи созданы и использованы следующие таблицы:

* Достопримечательность (attraction);
* Список достопримечательностей (attractionslist);
* Обратная связь (communication);
* Список медицинских процедур(medical\_procedures\_list);
* Меню ресторана (menu);
* Пользователь (person);
* Ресторан санатория (restaurant);
* Сеанс (seance);
* Заселение в отель (settlement);
* СПА (SPA);
* Трансфер в отель (transfer)

Далее описаны структура и тип хранящихся данных каждой таблицы.

Рисунок. 1.9 – таблица «Достопримечательность»

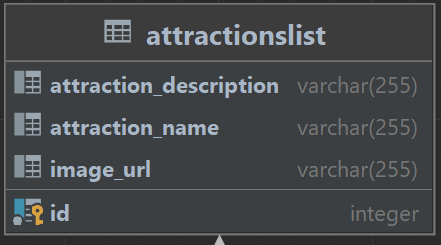
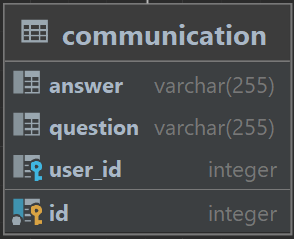


Рисунок. 1.10 – таблица «Список достопримечательностей»

Рисунок. 1.11 – таблица «Обратная связь»

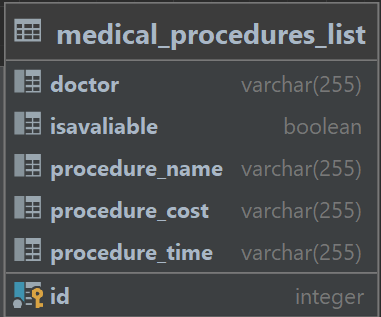


Рисунок. 1.12 – таблица «Список медицинских процедур»

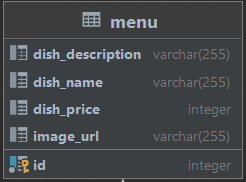


Рисунок. 1.13 – таблица «Меню ресторана»

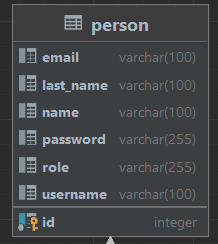


Рисунок. 1.14 – таблица «Пользователь»

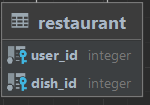
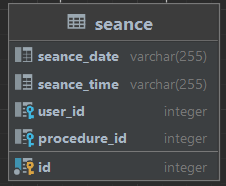


Рисунок. 1.15 – таблица «Ресторан санатория»

Рисунок. 1.16 – таблица «Сеанс»

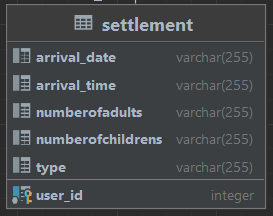


Рисунок. 1.17 – таблица «Заселение в отель»

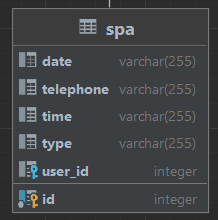


Рисунок. 1.18 – таблица «СПА»

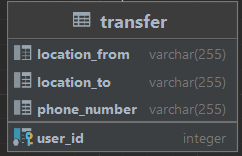


Рисунок. 1.19 – таблица «Трансфер в отель»

## **UML-модели представления программного средства и их описание**

# **Проектирование и конструирование программного средства**

## **Постановка задачи**

## **Обоснование выбора компонентов и технологий для реализации программного средства**

## **Архитектурные решения**

## **Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

## **Проектирование пользовательского интерфейса**

## **Методы и средства, используемые для обеспечения безопасности данных**

# **Тестирование и проверка работоспособности программного средства**

# **Руководство по развертыванию и использованию программного средства**

## **Руководство по установке (развертыванию) программного средства**

## **Руководство пользователя**

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

# **Приложение А (обязательное) - Отчет о проверке на заимствования в системе «Антиплагиат»**

# **Приложение Б (обязательное) - Листинг кода алгоритмов, реализующих бизнес-логику**

Класс Securityconfig:

package ru.vadim.spingapp.FirstSecurityApp.config;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.builders.WebSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import ru.vadim.spingapp.FirstSecurityApp.controllers.NewCustomAuthenticationSuccessHandler;  
import ru.vadim.spingapp.FirstSecurityApp.services.PersonDetailsService;  
  
  
@EnableWebSecurity  
@EnableGlobalMethodSecurity(prePostEnabled = true)  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
  
 private final PersonDetailsService personDetailsService;  
  
 @Autowired  
 public SecurityConfig(PersonDetailsService personDetailsService) {  
 this.personDetailsService = personDetailsService;  
 }  
  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/admin/admin","/admin/AdminRest").hasRole("ADMIN")  
 .antMatchers("/auth/login", "/auth/registration",  
 "/error","/auth/index","auth/homePage","/logout").permitAll()  
 .antMatchers("/auth/\*\*","/auth/sights/\*\*","/auth/medicine/\*\*","/user/\*\*").authenticated()  
 .anyRequest().hasAnyRole("USER", "ADMIN")  
 .and()  
 .formLogin().loginPage("/auth/index")  
 .loginProcessingUrl("/process\_login")  
 .defaultSuccessUrl("/auth/index", true)  
 .successHandler(new NewCustomAuthenticationSuccessHandler())  
 .failureUrl("/auth/login?error")  
 .and()  
 .logout()  
 .logoutUrl("/logout")  
 .logoutSuccessUrl("/auth/login");  
 }  
  
 // Настраиваем аутентификацию  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(personDetailsService)  
 .passwordEncoder(getPasswordEncoder());  
  
 }  
  
 @Override  
 public void configure(WebSecurity web) throws Exception {  
 web.ignoring().antMatchers("/css/\*\*","/images/\*\*");  
  
 }  
  
 @Bean  
 public PasswordEncoder getPasswordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
}

Класс AdminController:

package ru.vadim.spingapp.FirstSecurityApp.controllers;  
  
  
import org.apache.tomcat.jni.Proc;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
import ru.vadim.spingapp.FirstSecurityApp.models.\*;  
import ru.vadim.spingapp.FirstSecurityApp.repositories.\*;  
import ru.vadim.spingapp.FirstSecurityApp.services.\*;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.util.ArrayList;  
import java.util.List;  
  
@Controller  
public class AdminController {  
 List<Person> listPersons;  
 List<Menu> listMenu;  
  
  
 @Autowired  
 PeopleRepository peopleRepository;  
 @Autowired  
 PersonDetailsService personDetailsService;  
  
 @Autowired  
 CommunicationDetailService communicationDetailService;  
 @Autowired  
 CommunicationPepository communicationPepository;  
  
 @Autowired  
 AttractionDetailService attractionDetailService;  
 @Autowired  
 AttractionRepository attractionRepository;  
  
  
 @Autowired  
 MenuDetailsService menuDetailsService;  
  
 @Autowired  
 MenuRepository menuRepository;  
  
 @Autowired  
 ProcedureDetailService procedureDetailService;  
  
 @Autowired  
 ProcedureRepository procedureRepository;  
  
  
 @RequestMapping(value="/logout", method = RequestMethod.*GET*)  
 public String logoutPage (HttpServletRequest request, HttpServletResponse response) {  
 Authentication auth = SecurityContextHolder.*getContext*().getAuthentication();  
 if (auth != null){  
 new SecurityContextLogoutHandler().logout(request, response, auth);  
 }  
 return "redirect:/login?logout";  
 }  
  
 @GetMapping("/admin/admin")  
 public String userList(Model model)  
 {  
 listPersons = personDetailsService.listAll();  
 model.addAttribute("users",listPersons);  
 return "/admin/admin";  
 }  
  
 @PostMapping("/admin/admin/{id}")  
 public String userEdit(@PathVariable("id") int id,@ModelAttribute("person") Person newPerson)  
 {  
 Person person = personDetailsService.get(id);  
 person.setName(newPerson.getName());  
 person.setLast\_name(newPerson.getLast\_name());  
 person.setRole(newPerson.getRole());  
 person.setEmail(newPerson.getEmail());  
 person.setUsername(newPerson.getUsername());  
 peopleRepository.save(person);  
 return "redirect:/admin/admin";  
 }  
  
 /\* @GetMapping("/admin/menuOut/{id}")  
 public String menuUserList(@PathVariable("id") int id, Model model){  
 Person person = personDetailsService.get(id);  
 listMenu = person.getMenu();  
 model.addAttribute("listMenu",listMenu);  
 return "/admin/adminRest";  
 }\*/  
 @GetMapping("/admin/deleteUser/{id}")  
 public String deleteUser(@PathVariable int id) {  
 personDetailsService.deletePersonById(id);  
 return "redirect:/admin/admin";  
 }  
  
 @GetMapping("/admin/adminAnswer")  
 public String adminAnswer(Model model)  
 {  
 List<Communication> communicationList = communicationPepository.findAll();  
 model.addAttribute("communicationList",communicationList);  
 return "/admin/adminAnswer";  
 }  
  
 @PostMapping("/admin/adminAnswer/{id}")  
 public String adminPostAnswer(@PathVariable int id,@ModelAttribute("communication") Communication communication)  
 {  
 Communication communication1 = communicationDetailService.get(id);  
 communication1.setAnswer(communication.getAnswer());  
 communicationPepository.save(communication1);  
 return "redirect:/admin/adminAnswer";  
 }  
  
 @GetMapping("/admin/adminAddData")  
 public String adminAddData(Model model)  
 {  
 List<Attraction> attractionList = attractionDetailService.listAll();  
 Attraction newAttraction = new Attraction();  
 List<Procedure> procedureList = procedureDetailService.listAll();  
 Procedure newProcedure = new Procedure();  
 List<Menu> menuList = menuDetailsService.listAll();  
 Menu newMenu = new Menu();  
 model.addAttribute("menuList",menuList);  
 model.addAttribute("men",newMenu);  
 model.addAttribute("procedureList",procedureList);  
 model.addAttribute("proc",newProcedure);  
 model.addAttribute("attract",newAttraction);  
 model.addAttribute("attractionList",attractionList);  
 return "/admin/adminAddData";  
 }  
  
  
 @PostMapping("/admin/adminUpdateData/{id}")  
 public String userUpdateData(@PathVariable("id") int id,@ModelAttribute("attraction") Attraction newAttraction)  
 {  
 Attraction attraction = attractionDetailService.get(id);  
 attraction.setAttraction\_name(newAttraction.getAttraction\_name());  
 attraction.setAttraction\_description(newAttraction.getAttraction\_description());  
 attraction.setImage\_url(newAttraction.getImage\_url());  
 attractionRepository.save(attraction);  
 return "redirect:/admin/adminAddData";  
 }  
  
 @PostMapping("/admin/adminUpdateDataProcedure/{id}")  
 public String adminUpdateDataProcedure(@PathVariable("id") int id,@ModelAttribute("procedure") Procedure newProcedure)  
 {  
 Procedure procedure = procedureDetailService.get(id);  
 procedure.setLocation\_from(newProcedure.getLocation\_from());  
 procedure.setProcedure\_time(newProcedure.getProcedure\_time());  
 procedure.setProcedure\_cost(newProcedure.getProcedure\_cost());  
 procedure.setDoctor(newProcedure.getDoctor());  
 procedureRepository.save(procedure);  
 return "redirect:/admin/adminAddData";  
 }  
  
 @GetMapping("/admin/deleteEntity/{id}")  
 public String deleteEntity(@PathVariable int id) {  
 attractionDetailService.deleteAttractionByID(id);  
 return "redirect:/admin/adminAddData";  
 }  
  
  
 @GetMapping("/admin/deleteProcedure/{id}")  
 public String deleteProcedure(@PathVariable int id) {  
 procedureDetailService.deleteProcedureByID(id);  
 return "redirect:/admin/adminAddData";  
 }  
  
 @PostMapping("/admin/adminAddData")  
 public String userAddData(@ModelAttribute("attract") Attraction newAttraction)  
 {  
 attractionRepository.save(newAttraction);  
 return "redirect:/admin/admin";  
 }  
  
 @PostMapping("/admin/adminUpdateDataProcedure")  
 public String adminUpdateDataProcedure(@ModelAttribute("proc") Procedure newProcedure)  
 {  
 procedureRepository.save(newProcedure);  
 return "redirect:/admin/admin";  
 }  
  
 @PostMapping("/admin/adminUpdateDataMenu/{id}")  
 public String adminUpdateDataMenu(@PathVariable("id") int id,@ModelAttribute("menu") Menu newMenu)  
 {  
 Menu menu = menuDetailsService.get(id);  
 menu.setDish\_name(newMenu.getDish\_name());  
 menu.setDish\_description(newMenu.getDish\_description());  
 menu.setDish\_price(newMenu.getDish\_price());  
 menu.setImage\_url(newMenu.getImage\_url());  
 menuRepository.save(menu);  
 return "redirect:/admin/adminAddData";  
 }  
  
 @PostMapping("/admin/adminUpdateDataMenu")  
 public String adminUpdateMenu(@ModelAttribute("men") Menu newMenu)  
 {  
 menuRepository.save(newMenu);  
 return "redirect:/admin/adminAddData";  
 }  
 @GetMapping("/admin/deleteMenu/{id}")  
 public String deleteMenu(@PathVariable int id) {  
 menuDetailsService.deleteMenuById(id);  
 return "redirect:/admin/adminAddData";  
 }  
}

Класс AuthController:

package ru.vadim.spingapp.FirstSecurityApp.controllers;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.validation.BindingResult;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import ru.vadim.spingapp.FirstSecurityApp.models.Person;  
import ru.vadim.spingapp.FirstSecurityApp.repositories.PeopleRepository;  
import ru.vadim.spingapp.FirstSecurityApp.services.RegistrationService;  
import ru.vadim.spingapp.FirstSecurityApp.util.PersonValidator;  
  
import javax.validation.Valid;  
  
  
@Controller  
@RequestMapping("/auth")  
public class AuthController {  
  
 PeopleRepository peopleRepository;  
  
 private final RegistrationService registrationService;  
 private final PersonValidator personValidator;  
  
 @Autowired  
 public AuthController(RegistrationService registrationService, PersonValidator personValidator) {  
 this.registrationService = registrationService;  
 this.personValidator = personValidator;  
 }  
  
 @GetMapping("/login")  
 public String loginPage(){  
 return "auth/login";  
 }  
  
  
 @GetMapping("/index")  
 public String indexPage() {  
 return "auth/index";  
 }  
  
  
 @GetMapping("/registration")  
 public String registrationPage(@ModelAttribute("person") Person person) {  
 return "auth/registration";  
 }  
  
 @PostMapping("/registration")  
 public String performRegistration(@ModelAttribute("person") @Valid Person person,  
 BindingResult bindingResult) {  
 personValidator.validate(person, bindingResult);  
  
 if (bindingResult.hasErrors())  
 return "/auth/registration";  
  
 registrationService.register(person);  
  
 return "redirect:/auth/login";  
 }  
  
}

Класс UserController:

package ru.vadim.spingapp.FirstSecurityApp.controllers;  
  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
  
  
import ru.vadim.spingapp.FirstSecurityApp.services.AttractionDetailService;  
import ru.vadim.spingapp.FirstSecurityApp.services.MenuDetailsService;  
import ru.vadim.spingapp.FirstSecurityApp.services.PersonDetailsService;  
import ru.vadim.spingapp.FirstSecurityApp.services.ProcedureDetailService;  
import ru.vadim.spingapp.FirstSecurityApp.models.\*;  
import ru.vadim.spingapp.FirstSecurityApp.repositories.\*;  
  
import javax.persistence.EntityExistsException;  
import javax.validation.Valid;  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;  
import java.util.Set;  
  
@Controller  
public class UserController {  
  
 @Autowired  
 PersonDetailsService personDetailsService;  
  
 List<Seance> newSeance = new ArrayList<>();  
 List<Attraction> newAttractions = new ArrayList<>();  
 List<Menu> newMenu = new ArrayList<>();  
 List<Person> newPerson = new ArrayList<>();  
 @Autowired  
 MenuDetailsService menuDetailsService;  
  
 @Autowired  
 ProcedureDetailService procedureDetailService;  
 @Autowired  
 AttractionDetailService attractionDetailService;  
  
 @Autowired  
 AttractionRepository attractionRepository;  
 @Autowired  
 MenuRepository menuRepository;  
 @Autowired  
 PeopleRepository peopleRepository;  
 @Autowired  
 ProcedureRepository procedureRepository;  
  
 @Autowired  
 SeanceRepository seanceRepository;  
  
 @Autowired  
 CommunicationPepository communicationPepository;  
  
  
 @GetMapping("auth/homePage")  
 public String redirection(){  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
  
 if(person.getRole().equals("ROLE\_ADMIN")){  
 return "redirect:/admin/admin";  
 }  
 else  
 return "redirect:/user/userPage";  
  
 }  
  
  
 @GetMapping( "/auth/transfer" )  
 public String TransferPage(@ModelAttribute("transfer") Transfer transfer, Model model) {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Transfer tr = person.getTransfer();  
 model.addAttribute("Transfer",tr);  
 return "/auth/transfer";  
 }  
  
 @PostMapping("/auth/{id}")  
 public String addNewDish(@PathVariable("id") int id) {  
 Menu menu = menuDetailsService.get(id);  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 newMenu.add(menu);  
 newPerson.add(person);  
 person.setMenu(newMenu);  
 menu.setPersons(newPerson);  
 menuRepository.save(menu);  
 peopleRepository.save(person);  
 return "redirect:/auth/food";  
 }  
  
 @PostMapping("/auth/transfer")  
 public String postTransfer(@ModelAttribute("transfer") @Valid Transfer transfer) throws EntityExistsException{  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 if(person.getTransfer() == null) {  
 transfer.setPerson(person);  
 person.setTransfer(transfer);  
 peopleRepository.save(person);  
 }  
 return "/auth/transfer";  
 }  
  
  
 public String getCurrentUserName(){  
 Authentication authentication = SecurityContextHolder.*getContext*().getAuthentication();  
 return authentication.getName();  
 }  
  
  
 @GetMapping("/auth/food")  
 public String menuList(Model model)  
 {  
 List<Menu> menuList = menuDetailsService.listAll();  
 model.addAttribute("menuList",menuList);  
 return "/auth/food";  
 }  
  
  
 @GetMapping("/auth/sights")  
 public String sightsList(Model model)  
 {  
 List<Attraction> attractionList = attractionDetailService.listAll();  
 model.addAttribute("attractionList",attractionList);  
 return "/auth/sights";  
 }  
  
  
 @PostMapping("/auth/sights/{id}")  
 public String addNewAttractionToPerson(@PathVariable("id") int id) {  
 Attraction attraction= attractionDetailService.get(id);  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 newAttractions.add(attraction);  
 newPerson.add(person);  
 person.setAttractions(newAttractions);  
 attraction.setPersons(newPerson);  
 attractionRepository.save(attraction);  
  
 peopleRepository.save(person);  
 return "redirect:/auth/sights";  
 }  
  
  
 @GetMapping( "/auth/rooms" )  
 public String SettlementPage(@ModelAttribute("settlement") Settlement settlement, Model model) {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Settlement st = person.getSettlement();  
 model.addAttribute("Settlement",st);  
 return "/auth/rooms";  
 }  
  
 @PostMapping("/auth/settlement")  
 public String postSettlementPage(@ModelAttribute("settlement") @Valid Settlement settlement) throws EntityExistsException{  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 if(person.getSettlement() == null) {  
 settlement.setPerson(person);  
 person.setSettlement(settlement);  
 peopleRepository.save(person);  
 }  
 return "/auth/rooms";  
 }  
  
  
 @PostMapping("/auth/medicine")  
 public String postMedicinePage(@ModelAttribute("seance") @Valid Seance seance) {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 seance.setPerson(person);  
 Set<Seance> sc = new HashSet<>();  
 sc.add(seance);  
 newSeance.add(seance);  
 if(seance.getProcedure().isIsavaliable()) {  
 // seance.getProcedure().setIsavaliable(false);  
 person.setSeances(sc);  
 seance.getProcedure().setSeances(newSeance);  
 procedureRepository.save(seance.getProcedure());  
 peopleRepository.save(person);  
 seanceRepository.save(seance);  
 }  
 return "redirect:/auth/medicine";  
 }  
  
 @GetMapping("/auth/medicine")  
 public String doctorsList(@ModelAttribute("seance") Seance seance,Model model)  
 {  
 List<Procedure> procedureList = procedureDetailService.listAll();  
 model.addAttribute("procedureList",procedureList);  
 return "/auth/medicine";  
 }  
  
  
 public boolean findUsingEnhancedForLoop(String name, List<Seance> seance) {  
  
 for (Seance seance1: seance) {  
 if (seance1.getSeance\_time().equals(name)) {  
 return false;  
 }  
 }  
 return true;  
 }  
  
  
 @GetMapping("/auth/questions")  
 public String communication(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Communication communication = person.getCommunication();  
 model.addAttribute("communication",communication);  
 return "/auth/questions";  
 }  
  
 @PostMapping("/auth/questions")  
 public String postCommunication(@ModelAttribute("communication") @Valid Communication communication) throws EntityExistsException{  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 if(person.getCommunication() == null){  
 communication.setPerson(person);  
 person.setCommunication(communication);  
 communicationPepository.save(communication);  
 }  
 return "/auth/questions";  
 }  
  
 @GetMapping("/auth/spa")  
 public String spaGet(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Spa spa = person.getSpa();  
 model.addAttribute("spa",spa);  
 return "/auth/spa";  
 }  
  
  
 @PostMapping("/auth/spa")  
 public String spaPostPage(@ModelAttribute("spa") @Valid Spa spa) throws EntityExistsException{  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 if(person.getSpa() == null){  
 spa.setPerson(person);  
 person.setSpa(spa);  
 peopleRepository.save(person);  
 }  
 return "redirect:/auth/spa";  
 }  
  
  
 @GetMapping("/user/userPage")  
 public String userPageProfile(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 model.addAttribute("person",person);  
 return "/user/userPage";  
 }  
 @GetMapping("/user/userPageTransfer")  
 public String userPageTransfer(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Transfer transfer = person.getTransfer();  
 model.addAttribute("transfer",transfer);  
 return "/user/userPageTransfer";  
 }  
  
 @GetMapping("/user/userPageMedicine")  
 public String userPageMedicine(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Set<Seance> seances = person.getSeances();  
 model.addAttribute("seances",seances);  
 return "/user/userPageMedicine";  
 }  
  
 @GetMapping("/user/userPageSpa")  
 public String userPageSpa(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Spa spa = person.getSpa();  
 model.addAttribute("spa",spa);  
 return "/user/userPageSpa";  
 }  
  
 @GetMapping("/user/userPageRooms")  
 public String userPageRooms(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Settlement settlement = person.getSettlement();  
 model.addAttribute("settlement",settlement);  
 return "/user/userPageRooms";  
 }  
  
  
 @GetMapping("/user/userPageAttraction")  
 public String userPageAttraction(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 List<Attraction> attraction = person.getAttractions();  
 model.addAttribute("attractionList",attraction);  
 return "/user/userPageAttraction";  
 }  
  
 @GetMapping("/user/userPageRestourant")  
 public String userPageRestaurant(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 List<Menu> menu = person.getMenu();  
 model.addAttribute("menuList",menu);  
 return "/user/userPageRestourant";  
 }  
  
  
 @GetMapping("/user/userPageQuestion")  
 public String userPageQuestion(Model model)  
 {  
 Person person = peopleRepository.findByUsername(getCurrentUserName());  
 Communication communication = person.getCommunication();  
 model.addAttribute("communication",communication);  
 return "/user/userPageQuestion";  
 }  
}

# **Приложение В (обязательное) - Листинг скрипта генерации базы данных**

create table attraction (user\_id int4 not null, attraction\_id int4 not null)

create table attractionslist (id serial not null, attraction\_description varchar(255), attraction\_name varchar(255), image\_url varchar(255), primary key (id))

create table communication (id serial not null, answer varchar(255), question varchar(255), user\_id int4, primary key (id))

create table medical\_procedures\_list (id serial not null, doctor varchar(255), isavaliable boolean, procedure\_name varchar(255), procedure\_cost varchar(255), procedure\_time varchar(255), primary key (id))

create table menu (id serial not null, dish\_description varchar(255), dish\_name varchar(255), dish\_price int4, image\_url varchar(255), primary key (id))

create table person (id serial not null, email varchar(100), last\_name varchar(100), name varchar(100), password varchar(255), role varchar(255), username varchar(100), primary key (id))

create table restaurant (user\_id int4 not null, dish\_id int4 not null)

create table seance (id serial not null, seance\_date varchar(255), seance\_time varchar(255), user\_id int4, procedure\_id int4, primary key (id))

create table settlement (user\_id int4 not null, arrival\_date varchar(255), arrival\_time varchar(255), numberofadults varchar(255), numberofchildrens varchar(255), type varchar(255), primary key (user\_id))

create table spa (id serial not null, date varchar(255), telephone varchar(255), time varchar(255), type varchar(255), user\_id int4, primary key (id))

create table transfer (user\_id int4 not null, location\_from varchar(255), location\_to varchar(255), phone\_number varchar(255), primary key (user\_id))

alter table attraction add constraint FK935bi7n59kx4v5uj13hntq8f4 foreign key (attraction\_id) references attractionslist

alter table attraction add constraint FKqn7x4r41k3nf2e14wuw4jyv0j foreign key (user\_id) references person

alter table communication add constraint FK77y8wj38dhho0xsatcsw2rqip foreign key (user\_id) references person

alter table restaurant add constraint FK4ts6q9i59prn17m2k7969b0rl foreign key (dish\_id) references menu

alter table restaurant add constraint FKtljtoyc2kv34xvb2s8cl8g2ug foreign key (user\_id) references person

alter table seance add constraint FK4px9cuiko1yci1okg2oh8pjjo foreign key (user\_id) references person

alter table seance add constraint FKh1lbeup18718x1c3nu48eg8cs foreign key (procedure\_id) references medical\_procedures\_list

alter table settlement add constraint FKnp51xxouvds1kd7ujjvk0876w foreign key (user\_id) references person

alter table spa add constraint FKrxce4h7rjs0qj4aqa5a9jhhsa foreign key (user\_id) references person

alter table transfer add constraint FK748wmnjw1tfk36wuuot64fahg foreign key (user\_id) references person

# **Приложение Г (обязательное) - Ведомость документов курсового проекта**