**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

факультет інформатики та обчислювальної техніки

(повна назва інституту/факультету)

кафедра інформаційних систем та технологій

(повна назва кафедри)

|  |  |
| --- | --- |
|  |  |

**Курсова робота**

з дисципліни «Програмування-2»

на тему: ……………………..

Виконав : студент \_\_1\_ курсу, групи \_\_\_\_ІА-14\_\_\_\_\_

(шифр групи)

……………… \_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові) (підпис)

Науковий керівник

асистент інформаційних систем та технологій В.О. Міщенко \_\_\_\_\_\_\_\_\_

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент

…………………... \_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2020

ЗМІСТ

[ВСТУП 3](#_Toc105442119)

[1 ВИМОГИ ДО СИСТЕМИ 4](#_Toc105442120)

[1.1 Функціональні вимоги: 4](#_Toc105442121)

[1.2 Нефункціональні вимоги до системи 4](#_Toc105442122)

[2.1 Діаграма прецедентів 5](#_Toc105442123)

[2.2 Опис сценаріїв використання системи 6](#_Toc105442124)

[3 АРХІТЕКТУРА СИСТЕМИ 10](#_Toc105442125)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 12](#_Toc105442126)

[4.1 Загальна структура проекту 12](#_Toc105442127)

[4.2 Компоненти рівня доступу до даних 13](#_Toc105442128)

[4.3 Компоненти рівня бізнес-логіки 14](#_Toc105442129)

[4.4 Компоненти рівня інтерфейсу користувача 15](#_Toc105442130)

[ВИСНОВКИ 16](#_Toc105442131)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 18](#_Toc105442132)

[ДОДАТКИ 19](#_Toc105442133)

# ВСТУП

В наші часи, з великою кількістю різних вірусів та інших хворіб, медицина стана дуже важливою галуззю для людства. Проте не всі лікарні йду вслід за технологіями і нажаль не всі лікарні мають доступність для дистанційної реєстрації, зручної для клієнтів. Багато з них наразі для реєстрації використовують реєстраційні пункти в самих лікарнях та дзвінки у ці самі пункти, щоб дізнатись вільні години, проте навіщо?

Сучасні технології дають нам великий поштовх у таких задачах, онлайн реєстратура – це одна з таких задач. Авжеж зручніше знайти розклад до певного лікаря по графіку, аніж дзвонити кожен раз коли потрібно зареєструватись чи відмінити реєстрацію щодо відвідувань лікаря.

Наразі в україні з кращих сайтів можна виділити одиниці, що зручні для пацієнта/хворого. З них можна виділити Helsi та pb4.

Метою роботи є створення сайту, що дозволяє людям виконувати такий перелік дій та вирішувати такі задачі:

* Створювати власний аккаунт.
* Реєструватись на відвідування до певного лікаря.
* Переглядати вільні та зайняті години на відвідування лікаря.
* Відміняти реєстрацію на відвідування.

# 1 ВИМОГИ ДО СИСТЕМИ

## Функціональні вимоги:

* Перегляд розкладу часів прийому вказаного лікаря.
* Запис на прийом.
* Відміна прийому

## 1.2 Нефункціональні вимоги до системи

* Система має відповідати наступним функціональним вимогам:
* система повинна мати відкриту архітектуру;
* система повинна мати веб-інтерфейс;
* інтерфейс користувача має бути зручним та інтуїтивно-зрозумілим;
* система повинна бути крос-платформною.

2 СЦЕНАРІЙ ВИКОРИСТАННЯ

## 2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами виконання є користувач та адміністрато, взаємодії наведені нижче на Рис 2.1.

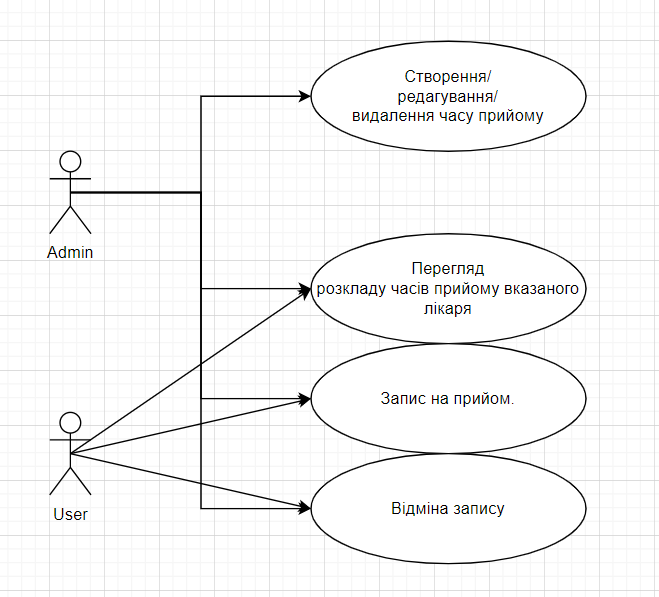


Рис 2.1. Діаграма прецендентів

## 2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.5

Таблиця 2.1 – Сценарій використання «Реєстраці»

|  |  |
| --- | --- |
| Назва | Реєстрація |
| ID | 1 |
| Опис | Використовуючи форму, виконання реєстрації гостя, для створення аккаунту |
| Актори | Гість |
| Вигоди компанії | Якщо Гість не може зареєструватись, вони не зможуть виконувати весь функціонал доступний користувачу. |
| Частота користування | Один раз для одного користувача |
| Тригери | Користувач відкриває форму реєстрації |
| Передумови | Форма відображається на домашній сторінці, якщо не виконаний вхід |
| Пост-умови |  |
| Основний розвиток | Гість вводить свої дані для створення аккаунту. |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

Таблиця 2.2 – Сценарій використання «Вхід у аккаунт»

|  |  |
| --- | --- |
| Назва | Вхід в аккаунт |
| ID | 2 |
| Опис | Використовуючи форму для входу, заходить у аккаунт. |
| Актори | Користувач |
| Вигоди компанії | Якщо користувач не зайшов у аккаунт він не зможе взаємодіяти з функціоналом сайту |
| Частота користування | Постійно |
| Тригери | Користувач відкриває форму входу |
| Передумови | Форма відображається на домашній сторінці, якщо не виконаний вхід |
| Пост-умови | Користувач потрапляє на сторінку де може взаємодіяти зі всім доступним функціоналом |
| Основний розвиток | Користувач натискає вхід та вводить свої дані для входу у аккаунт |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

Таблиця 2.3 – Сценарій використання «Додавання часу»

|  |  |
| --- | --- |
| Назва | Додавання часу |
| ID | 3 |
| Опис | Адміністратор додає час |
| Актори | Адміністратор |
| Вигоди компанії | Для зручного поповнення даних сайту |
| Частота користування | Постійно |
| Тригери | Адміністратор вводить дані у форму додавання часу |
| Передумови | Вхід в аккаунт адміністратора |
| Пост-умови | - |
| Основний розвиток | Адміністратор вводить дані лікаря, та вибирає час і натискає додати |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

Таблиця 2.4 – Сценарій використання «Список лікарів»

|  |  |
| --- | --- |
| Назва | Список лікарів |
| ID | 4 |
| Опис | Користувач отримує список лікарів |
| Актори | Користувач |
| Вигоди компанії | Зручна взаємодія користувача, для вибору лікаря для відвідування |
| Частота користування | Постійно |
| Тригери | Користувач вибирає лікаря |
| Передумови | Сторінка доступна при вході в аккаунт |
| Пост-умови | Користувач потрапляє на сторінку для взаємодії з певним лікарем. |
| Основний розвиток | Користувач натискає на кнопку біля певного лікаря, та отримує графік роботи та можливість записатись. |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на Рис 3.1

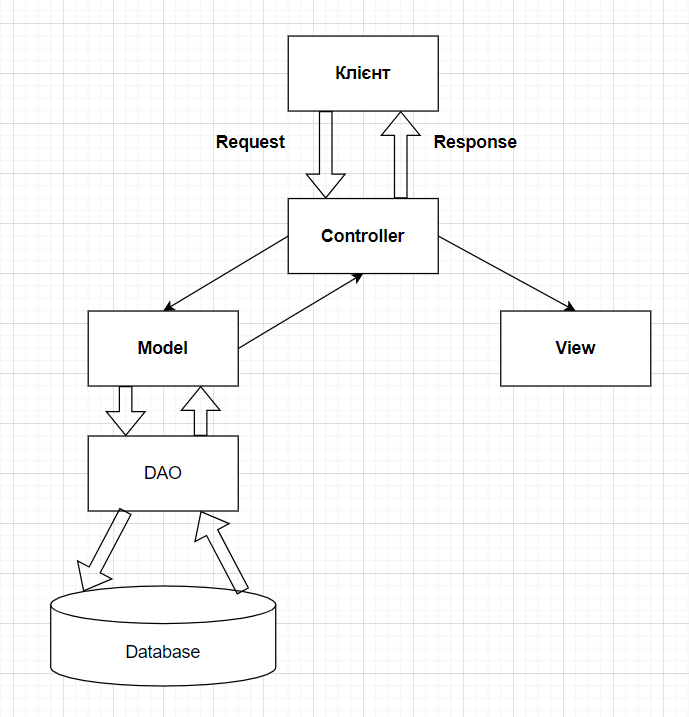


Рис 3.1 Загальна архітектура системи

Система складається з наступних елементів:

- графічний інтерфейс;

- серверна частина;

- база даних.

Графічний інтерфейс необхідний для взаємодії з користувачем. HTTP запит надходить до серверної частини, де оброблюється і повертається відповідь. На серверній частині виконується основна логіка системи. Дані, отриманні з графічного інтерфейсу конвертуються. Також, серверна частина формує запит до бази даних та оброблює відповідь і передає її до графічного інтерфейсу. База даних зберігає дані, які були сформовані на серверній частині та повертає їх у разі запиту. Схема зображена на Рис 3.1.

До серверної частини належать наступні елементи:

- Контролер;

- Модель та вигляд;

- DAO;

- База Даних.

На контролер надходять дані з графічного інтерфейсу. З контролеру, дані формуються в запит та взаємодіють через модель з базою даних і зберігаються, або витягуються. Також в контролері формується вид, тобто об’єкт і його ім’я для відображення на графічному інтерфейсі.

# 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

## 4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

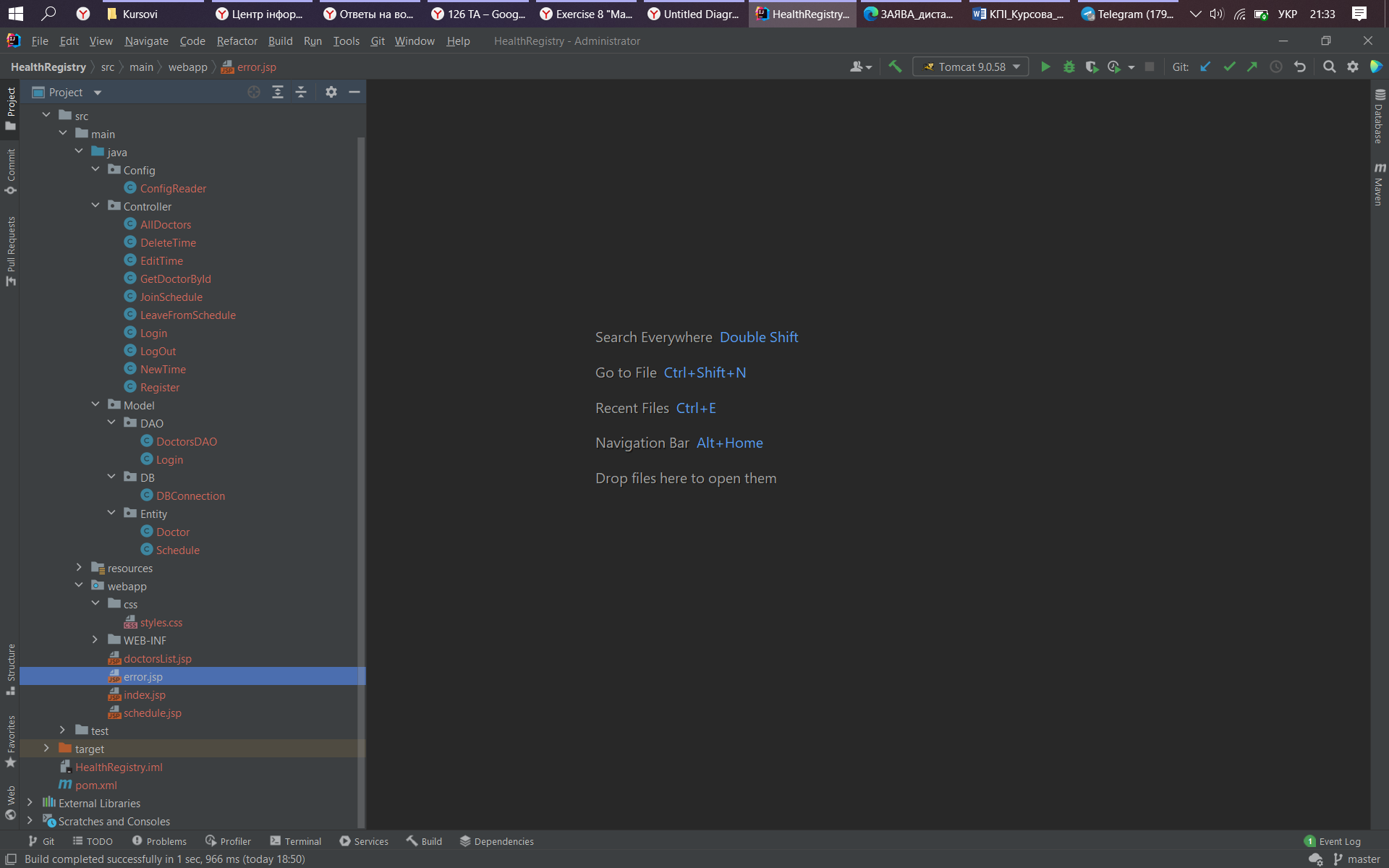


Рисунок 4.1 – Загальна структура проекту

Проект складається з веб-ресурсів, бібліотек, та вихідного коду, який в свою чергу можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти.

## 4.2 Компоненти рівня доступу до даних

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.2

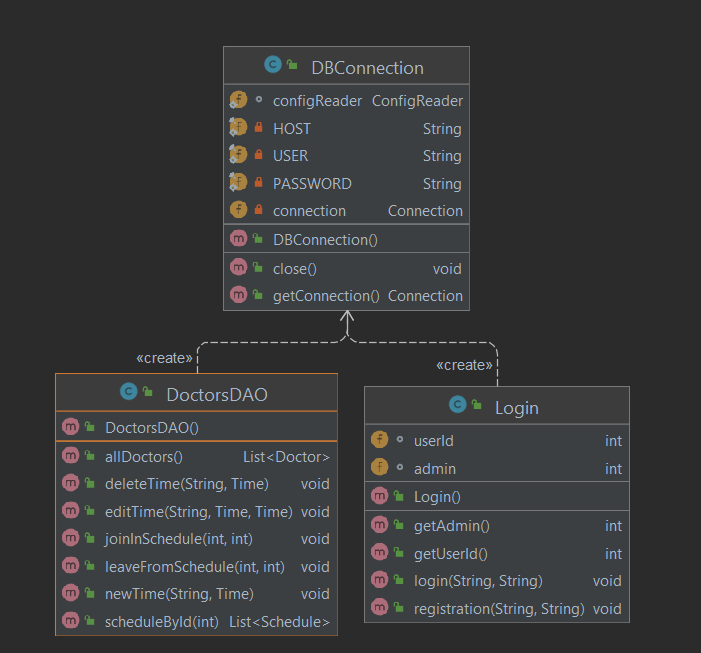
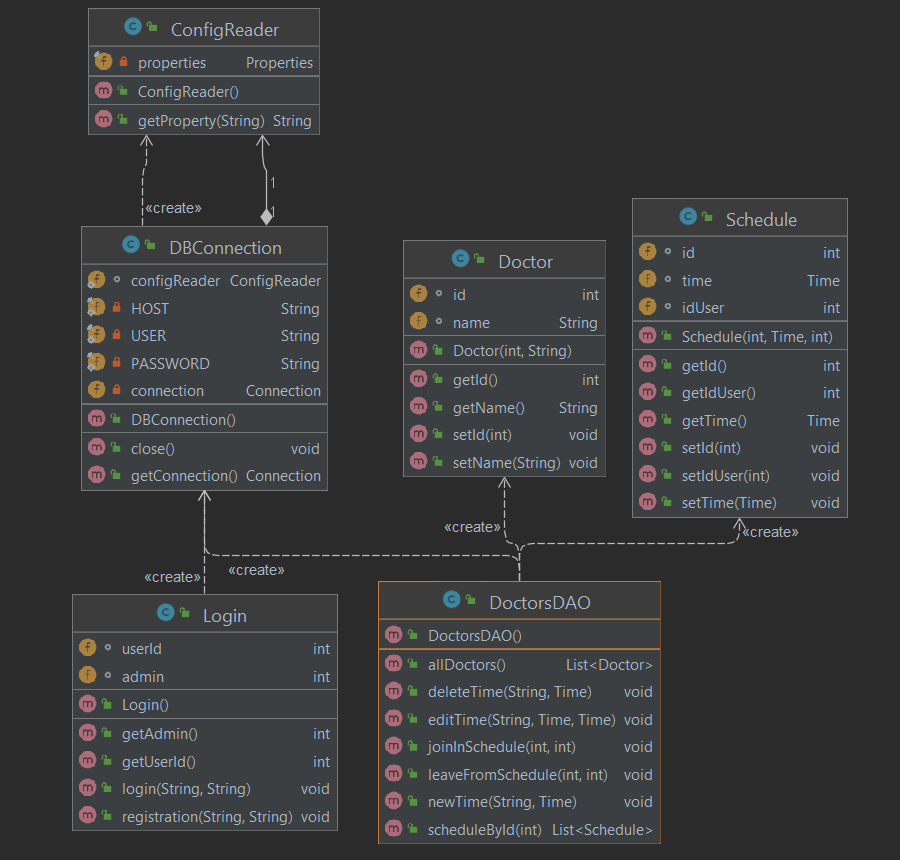
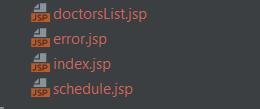


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

## 4.3 Компоненти рівня бізнес-логіки



## 4.4 Компоненти рівня інтерфейсу користувача



JSP + JSTL використовується для виведення результатів для користувача

Також прикріплений файл стилів, що дозволяє користувачу бачити приємніший інтерфейс



# ВИСНОВКИ

Під час виконання роботи було використано ідею сайту pb4, проаналізовано переваги та недоліки цієї ідеї та задумки, виконання основної роботи та проаналізований подальший розвиток проекту.

Є ще багато рішень для зручнішого користування сайтом, тому у подальшому розвитку проекта планується додати вибір певних симптомів, можливість онлайн зв’язатись з лікарем, та багато інших можливостей, також проект має ще багато способів для доповнення та покращення.

Сам розвиток був за такими пунктами:

1. Формулювння функціональні та нефункціональні вимоги до системи, що визначило очікувану поведінку системи.
2. Обрано технології на яких буде написана система. Java була обрана в якості мови програмування, середа розробки – IntelliJ IDEA за її можливості та потужність. В якості системи управління базами даних було вирішено використовувати MySQL за легкість її налаштування та підтримки.
3. Обдумані сценарії використання, та можливі нові функції для подальшого збільшення проекту.
4. Розраховуючи загальну архітектуру було вибранно використання шаблону проектування такого як – MVC pattern, через його зручність та безпечність за рахунок розділення логічних рівнів.В моєму випадку таких як Controller(Сервлети для взаємодії користувача з сервером), Model(Для взаємодії з DAO, що в свою чергу взаємодіє з базою даних), View(Для взаємодії з користувацьким інтерфейсом), та наявністю 2 інтерфейсів, для можливості зручної заміни модуля(DAO).
5. В кінці виконання всіх попередніх пунктів розвитку проекту стало підбирання стилів, для приємного відображення користувацького інтерфейсу. Використовуючи стилі CSS.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Что такое MVC [Електронний ресурс] – https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami

2. Руководство Java JSP для начинающих [Електронний ресурс] –https://betacode.net/10263/java-jsp 3. Metanit java.

# ДОДАТКИ

package Config;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.Properties;  
  
public class ConfigReader {  
  
 private final Properties properties = new Properties();  
  
 public ConfigReader() {  
 String filePath = "db.properties";  
 try (InputStream out = getClass().getClassLoader().getResourceAsStream(filePath)) {  
 properties.load(out);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 public String getProperty(String property){  
 return properties.getProperty(property);  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
import Model.Entity.Doctor;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.util.List;  
  
@WebServlet(name = "AllDoctors", value = "/AllDoctors")  
public class AllDoctors extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 if (userId > 0) {  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 List<Doctor> doctorList = doctorsDAO.allDoctors();  
 request.getSession().setAttribute("doctorsList", doctorList);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 }  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
import java.sql.Time;  
  
@WebServlet(name = "DeleteTime", value = "/DeleteTime")  
public class DeleteTime extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 int admin = (int) request.getSession().getAttribute("admin");  
 if (userId > 0 && admin == 1) {  
 String doctorName = request.getParameter("doctorName");  
 Time time = Time.valueOf(request.getParameter("time"));  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 doctorsDAO.deleteTime(doctorName,time);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 }  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
import java.sql.Time;  
  
@WebServlet(name = "EditTime", value = "/EditTime")  
public class EditTime extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 int admin = (int) request.getSession().getAttribute("admin");  
 if (userId > 0 && admin == 1) {  
 String doctorName = request.getParameter("doctorName");  
 Time time1 = Time.valueOf(request.getParameter("time1"));  
 Time time2 = Time.valueOf(request.getParameter("time2"));  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 doctorsDAO.editTime(doctorName,time1, time2);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 }  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
import Model.Entity.Doctor;  
import Model.Entity.Schedule;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
import java.util.List;  
  
@WebServlet(name = "GetDoctorById", value = "/GetDoctorById")  
public class GetDoctorById extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 if (userId > 0) {  
 int idDoctor = Integer.*parseInt*(request.getParameter("idDoctor"));  
 request.getSession().setAttribute("idDoctor", idDoctor);  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 List<Schedule> scheduleList = doctorsDAO.scheduleById(idDoctor);  
 request.getSession().setAttribute("scheduleList", scheduleList);  
 request.getRequestDispatcher("schedule.jsp").forward(request, response);  
 }  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
import Model.Entity.Schedule;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
import java.util.List;  
  
@WebServlet(name = "JoinSchedule", value = "/JoinSchedule")  
public class JoinSchedule extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 if (userId > 0) {  
 int idSchedule = Integer.*parseInt*(request.getParameter("id"));  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 doctorsDAO.joinInSchedule(userId,idSchedule);  
  
 request.getRequestDispatcher("GetDoctorById?idDoctor="+  
 request.getSession().getAttribute("idDoctor")).forward(request, response);  
 }  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
  
@WebServlet(name = "LeaveFromSchedule", value = "/LeaveFromSchedule")  
public class LeaveFromSchedule extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 if (userId > 0) {  
 int idSchedule = Integer.parseInt(request.getParameter("id"));  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 doctorsDAO.leaveFromSchedule(userId,idSchedule);  
 request.getRequestDispatcher("GetDoctorById?idDoctor="+  
 request.getSession().getAttribute("idDoctor")).forward(request, response);  
 }  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
}

package Controller;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "Login", value = "/Login")  
public class Login extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String login = request.getParameter("login");  
 String password = request.getParameter("password");  
 Model.DAO.Login loginDAO = new Model.DAO.Login();  
 loginDAO.login(login, password);  
 int userId = loginDAO.getUserId();  
 if (userId>0) {  
 int admin = loginDAO.getAdmin();  
 request.getSession().setAttribute("userId", userId);  
 request.getSession().setAttribute("admin", admin);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 } else {  
 request.setAttribute("error", "Wrong login or pass");  
 request.getRequestDispatcher("error.jsp").forward(request, response);  
 }  
 }  
}

package Controller;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "LogOut", value = "/LogOut")  
public class LogOut extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 request.getSession().setAttribute("userId", 0);  
 request.getSession().setAttribute("admin", 0);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
}

package Controller;  
  
import Model.DAO.DoctorsDAO;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.sql.Time;  
  
@WebServlet(name = "NewTime", value = "/NewTime")  
public class NewTime extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 int userId = (int) request.getSession().getAttribute("userId");  
 int admin = (int) request.getSession().getAttribute("admin");  
 if (userId > 0 && admin == 1) {  
 String doctorName = request.getParameter("doctorName");  
 Time time = Time.valueOf(request.getParameter("time"));  
 DoctorsDAO doctorsDAO = new DoctorsDAO();  
 doctorsDAO.newTime(doctorName,time);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 }  
 }  
}

package Controller;  
  
  
import Model.DAO.Login;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@WebServlet(name = "Register", value = "/Register")  
public class Register extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String login = request.getParameter("login");  
 String password = request.getParameter("password");  
 Login loginDAO = new Login();  
 loginDAO.registration(login, password);  
 request.getRequestDispatcher("index.jsp").forward(request, response);  
 }  
}

package Model.DAO;  
  
import Model.DB.DBConnection;  
import Model.Entity.Doctor;  
import Model.Entity.Schedule;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class DoctorsDAO {  
  
  
 public void newTime(String doctorName, Time time) {  
 String query1 = "INSERT IGNORE INTO time(time) values (?);";  
 String query = "INSERT IGNORE INTO doctortime(iddoctor, idtime) " +  
 "values((select iddoctors from doctors where doctorsName=?),(select idtime from time where time=?));";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query1);  
 preparedStatement1.setTime(1, time);  
 preparedStatement1.executeUpdate();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setString(1, doctorName);  
 preparedStatement.setTime(2, time);  
 preparedStatement.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void editTime(String doctorName, Time time1, Time time2) {  
 String query1 = "INSERT IGNORE INTO time(time) values (?);";  
 String query = "UPDATE doctortime set idTime=(select idtime from time where time.time=?) " +  
 "where iddoctor=(select iddoctors from doctors where doctorsName=?) \n" +  
 "and idtime=(select idtime from time where time.time=?);";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement1 = connection.prepareStatement(query1);  
 preparedStatement1.setTime(1, time2);  
 preparedStatement1.executeUpdate();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setTime(1, time2);  
 preparedStatement.setString(2, doctorName);  
 preparedStatement.setTime(3, time1);  
 preparedStatement.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void deleteTime(String doctorName, Time time) {  
 String query = "DELETE FROM doctortime \n" +  
 "where iddoctor=(select iddoctors from doctors where doctorsName=?) \n" +  
 "and idTime=(select idtime from time where time=?);";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setString(1, doctorName);  
 preparedStatement.setTime(2, time);  
 preparedStatement.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public List<Doctor> allDoctors() {  
 List<Doctor> doctorsList = new ArrayList<>();  
 String query = "SELECT iddoctors, doctorsName FROM health\_registry.doctors;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 while (resultSet.next()) {  
 int id = resultSet.getInt(1);  
 String name = resultSet.getString(2);  
 Doctor doctor = new Doctor(id, name);  
 doctorsList.add(doctor);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return doctorsList;  
 }  
  
 public void joinInSchedule(int idUser, int idSchedule) {  
 String query = "UPDATE doctortime set iduser =? where iddoctortime=?";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setInt(1, idUser);  
 preparedStatement.setInt(2, idSchedule);  
 preparedStatement.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public List<Schedule> scheduleById(int idDoctor) {  
 List<Schedule> scheduleList = new ArrayList<>();  
 String query = "SELECT iddoctortime, time, iduser FROM health\_registry.doctortime dt\n" +  
 "INNER JOIN time t on dt.idtime=t.idtime\n" +  
 "where dt.iddoctor=? order by time;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setInt(1, idDoctor);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 while (resultSet.next()) {  
 int id = resultSet.getInt(1);  
 Time time = resultSet.getTime(2);  
 int idUser = resultSet.getInt(3);  
 Schedule schedule = new Schedule(id, time, idUser);  
 scheduleList.add(schedule);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return scheduleList;  
 }  
  
 public void leaveFromSchedule(int userId, int idSchedule) {  
 String query = "UPDATE doctortime set iduser =0 where iddoctortime=? and iduser=?;";  
 try (DBConnection db = new DBConnection()) {  
 Connection connection = db.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setInt(1, idSchedule);  
 preparedStatement.setInt(2, userId);  
 preparedStatement.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

package Model.DAO;  
  
import Model.DB.DBConnection;  
  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
  
public class Login {  
 int userId = 0;  
 int admin = 0;  
  
 public int getUserId() {  
 return userId;  
 }  
  
 public int getAdmin() {  
 return admin;  
 }  
  
 public void login(String login, String password) {  
 try (DBConnection db = new DBConnection()) {  
 String query = "SELECT idusers, role FROM health\_registry.users where login = ? and password = ?;";  
 PreparedStatement preparedStatement = db.getConnection().prepareStatement(query);  
 preparedStatement.setString(1, login);  
 preparedStatement.setString(2, password);  
 ResultSet result = preparedStatement.executeQuery();  
 if (result.next()) {  
 userId = result.getInt(1);  
 admin = result.getInt(2);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void registration(String login, String password) {  
 try (DBConnection db = new DBConnection()) {  
 String query = "insert into health\_registry.users (login, password) VALUES (?,?);";  
 PreparedStatement preparedStatement = db.getConnection().prepareStatement(query);  
 preparedStatement.setString(1, login);  
 preparedStatement.setString(2, password);  
 preparedStatement.executeUpdate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
}

package Model.DB;  
  
import Config.ConfigReader;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DBConnection implements AutoCloseable{  
 static ConfigReader *configReader* = new ConfigReader();  
 private static final String *HOST* = *configReader*.getProperty("HOST");  
 private static final String *USER* = *configReader*.getProperty("USER");  
 private static final String *PASSWORD* = *configReader*.getProperty("PASSWORD");  
 private Connection connection = null;  
  
 public DBConnection() {  
 try {  
 connection = DriverManager.*getConnection*(*HOST*, *USER*, *PASSWORD*);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public Connection getConnection() {  
 return connection;  
 }  
  
  
 @Override  
 public void close() throws Exception {  
 connection.close();  
 }  
}

package Model.Entity;  
  
public class Doctor {  
 int id;  
 String name;  
  
 public Doctor(int id, String name) {  
 this.id = id;  
 this.name = name;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

package Model.Entity;  
  
import java.sql.Time;  
  
public class Schedule {  
 int id;  
 Time time;  
 int idUser;  
  
 public Schedule(int id, Time time, int idUser) {  
 this.id = id;  
 this.time = time;  
 this.idUser = idUser;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public Time getTime() {  
 return time;  
 }  
  
 public void setTime(Time time) {  
 this.time = time;  
 }  
  
 public int getIdUser() {  
 return idUser;  
 }  
  
 public void setIdUser(int idUser) {  
 this.idUser = idUser;  
 }  
}

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **page** contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  
<**c:if** test="${doctorsList!=null}">  
 <div class="doctorsList">  
 <**c:forEach** var="doctor" items="${doctorsList}">  
 <div class="doctor">  
 <p>${doctor.name}</p>  
 <a href="/GetDoctorById?idDoctor=${doctor.id}">Check schedule</a>  
 </div>  
 </**c:forEach**>  
 </div>  
 ${doctorsList=null}  
</**c:if**>

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **page** contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  
<!DOCTYPE html>  
<html>  
<head>  
 <title>Health Registry</title>  
 <link rel="stylesheet" href="css/styles.css">  
</head>  
<body>  
<**c:choose**>  
 <**c:when** test="${userId>0}">  
 <**c:if** test="${doctorsList==null}">  
 <**c:redirect** url="/AllDoctors"/>  
 </**c:if**>  
 <a href="/LogOut">Log Out</a>  
 <div class="adminFunctions">  
 <**c:if** test="${admin==1}">  
 <form method="post" action="/NewTime">  
 <p>Create Time</p>  
 <label>Name</label><br>  
 <input type="text" required name="doctorName"><br>  
 <label>Time</label><br>  
 <input type="time" step="1" required name="time"><br>  
 <button type="submit">Create</button>  
 </form>  
 <form method="post" action="/EditTime">  
 <p>Edit Time</p>  
 <label>Name</label><br>  
 <input type="text" required name="doctorName"><br>  
 <label>Time to edit</label><br>  
 <input type="time" step="1" required name="time1"><br>  
 <label>New time</label><br>  
 <input type="time" step="1" required name="time2"><br>  
 <button type="submit">Edit</button>  
 </form>  
 <form method="post" action="/DeleteTime">  
 <p>Delete Time</p>  
 <label>Name</label><br>  
 <input type="text" required name="doctorName"><br>  
 <label>Time</label><br>  
 <input type="time" step="1" required name="time"><br>  
 <button type="submit">Delete</button>  
 </form></div>  
 </**c:if**>  
 <**jsp:include** page="doctorsList.jsp"/>  
 </**c:when**>  
 <**c:otherwise**>  
 <div class="login">  
 <form action="/Login" method="post">  
 <label>Введіть логін</label><br>  
 <input name="login" required type="text"><br>  
 <label>Ввудіть пароль</label><br>  
 <input name="password" required type="password"><br>  
 <button type="submit">Вхід</button>  
 <button type="submit" formaction="/Register">Реєстрація</button>  
 </form>  
 </div>  
 </**c:otherwise**>  
</**c:choose**>  
</body>  
</html>

<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Schedule</title>  
</head>  
<body>  
<a href="index.jsp">Back</a>  
<**c:if** test="${scheduleList!=null}">  
 <div class="scheduleList">  
 <**c:forEach** var="schedule" items="${scheduleList}">  
 <div class="schedule">  
 <p>${schedule.time}</p>  
 <**c:choose**>  
 <**c:when** test="${schedule.idUser==0}">  
 <a href="/JoinSchedule?id=${schedule.id}">Join schedule</a></**c:when**>  
 <**c:when** test="${schedule.idUser==userId}">  
 <a href="/LeaveFromSchedule?id=${schedule.id}">Cancel</a></**c:when**>  
 <**c:otherwise**>  
 <p>No Slots</p>  
 </**c:otherwise**>  
 </**c:choose**>  
 </div>  
 </**c:forEach**>  
 </div>  
 ${scheduleList=null}  
</**c:if**>  
</body>  
</html>