**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: Сайт знайомств

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

(шифр групи)

Кручинін Максим Ігорович \_\_\_\_\_\_\_\_\_

(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

Кручинін.М.І. \_\_\_\_\_\_\_\_\_

(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 останні роки соціальні мережі міцно увійшли в наше життя. Поняття «соціальні мережі» багато категорій: блоги, чати, сайти знайомств і, звичайно, соціальні мережі. Однією з під-категорій соціальних мереж є сайти знайомств.

Найпопулярнішими серед молоді є такі соціальні мережі: Facebook, Twitter, Instagram, Telegram та багато інших. Деякі втрачають популярність, проте доволі часто на їх місце постають інші

З одного боку, соціальна мережа це дуже зручний інструмент щоб швидко обмінюватись повідомленнями, шукати друзів і нові знайомства, зберігати важливу інформацію та обговорювати актуальних проблем у групах, прямо не виходячи з дому. Через це, соціальні мережі дуже зручні для тих, кому необхідно підтримувати зв’язок з людьми, що живуть у різних містах або країнах.

Також невід’ємною частиною таких сайтів є пошук нових знайомств, нових друзів, та просто цікавих людей для спілкування.

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

* Створювати власний аккаунт.
* Надсилати запрошення іншим людям.
* Бачити приватну інформацію у людей які є в списку друзів.
* Приймати запрошення від інших користувачів.

# 1 ВИМОГИ ДО СИСТЕМИ

## 1.1 Функціональні вимоги:

* Редагування власної анкети.
* Перегляд анкет.
* Пошук анкет за ключовими словами.
* Відправка Запрошення.
* Перегляд Закритої інформації іншого користувача при отриманні від нього Запрошення чи підтвердження власного запрошення

## 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.

Актором виконання є користувач, взаємодії наведені нижче.

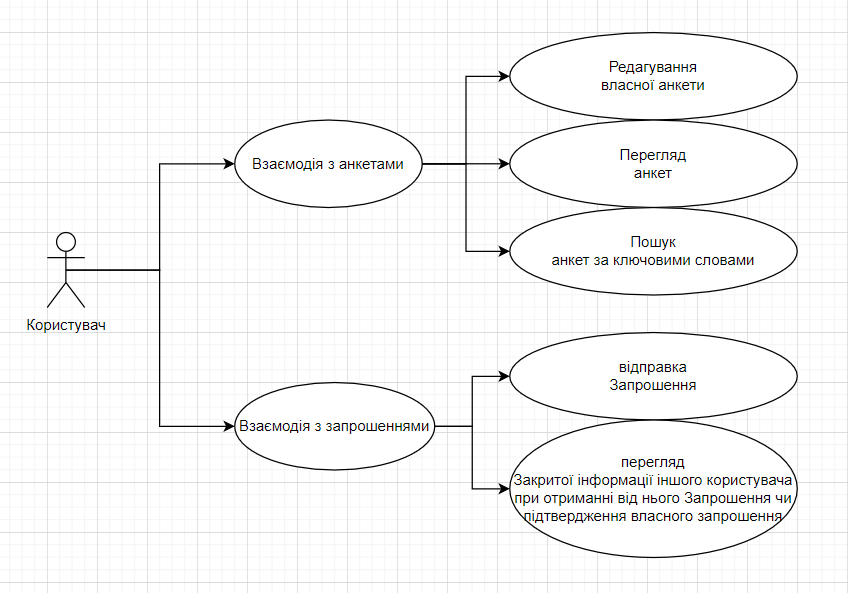


Рис 2.1. Діаграма прецендентів

## 2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.5

Таблиця 2.1 – Сценарій використання «Реєстрація аккаунту»

|  |  |
| --- | --- |
| Назва | Реєстрація |
| ID | 1 |
| Опис | Використовуючи форму, виконання реєстрації гостя, для створення аккаунту |
| Актори | Користувач/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 доступна при нажаті на кнопку реєстрації |
| Пост-умови | Користувач автоматично входить у свій аккаунт |
| Основний розвиток | Гість натискає реєстрація та вводить свої дані для створення аккаунту і потрапляє на сторінку для взаємодії |
| Альтернативні розвитки | – |
| Виняткові ситуації | якщо реєстрація пройшла не успішно користувач потрапить у вікно помилок з відповідним статусу помилки |

Таблиця 2.2 – Сценарій використання «Вхід у аккаунт»

|  |  |
| --- | --- |
| Назва | Вхід в аккаунт |
| ID | 2 |
| Опис | Використовуючи форму для входу у аккаунт взаємодіє з входом для входу відповідно. |
| 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 доступна при нажаті на кнопку входу |
| Пост-умови | Користувач потрапляє на сторінку де може взаємодіяти зі всім доступним функціоналом |
| Основний розвиток | Гість натискає вхід та вводить свої дані для входу у аккаунт |
| Альтернативні розвитки | – |
| Виняткові ситуації | Неправильно введені дані -> потраплення до сторінки помилок з відповідним статусом |

Таблиця 2.3 – Сценарій використання «Пошук користувачів»

|  |  |
| --- | --- |
| Назва | Пошук користувачів |
| ID | 3 |
| Опис | Користувач шукає інших користувачів |
| Актори | Користувач |
| Вигоди компанії | Взаємодія користувача з іншими користувачами, для створення зв’язку сайту знайомств |
| Частота користування | Постійно |
| Тригери | Користувач відкриває сторінку пошуку користувачів |
| Передумови | Сторінка доступна при нажатті кнопки пошуку на панелі, доступній на всіх сторінках |
| Пост-умови | Користувач потрапляє на сторінку для взаємодії з полем для введення назви щоб отримати користувачів що мають схоже ім’я користувача |
| Основний розвиток | Користувач натискає на пошук, вводить назву та отримує список користувачів по запиту |
| Альтернативні розвитки | – |
| Виняткові ситуації | Якщо не знайдено користувачів список не буде виведений |

Таблиця 2.4 – Сценарій використання «Список друзів»

|  |  |
| --- | --- |
| Назва | Список друзів |
| ID | 4 |
| Опис | Користувач отримує список друзів |
| Актори | Користувач |
| Вигоди компанії | Взаємодія користувача з друзями, для створення зв’язку сайту знайомств |
| Частота користування | Постійно |
| Тригери | Користувач відкриває сторінку друзі |
| Передумови | Сторінка доступна при нажатті кнопки друзі на панелі, доступній на всіх сторінках |
| Пост-умови | Користувач потрапляє на сторінку для взаємодії з друзями. |
| Основний розвиток | Користувач натискає на кнопку друзі, взаємодіє з іншими користувачами, та бачить їх приватну інформацію |
| Альтернативні розвитки | – |
| Виняткові ситуації | - |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на Рис 3.1

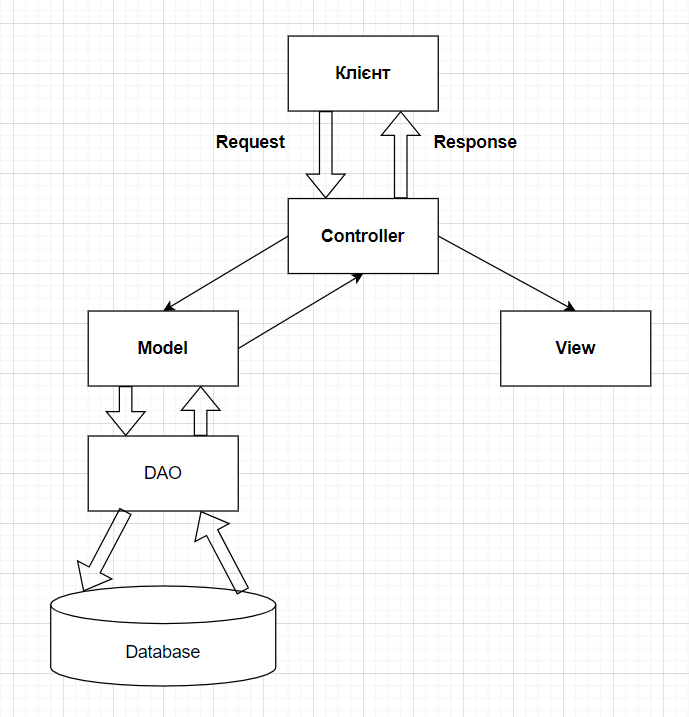


Рис 3.1 Загальна архітектура системи

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;

-Дао;

- БД.

На контролер надходять дані з графічного інтерфейсу. З контролеру, дані формуються в сервісі для запиту в репозиторій. З репозиторію дані надсилаються до бази даних і зберігаються. 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

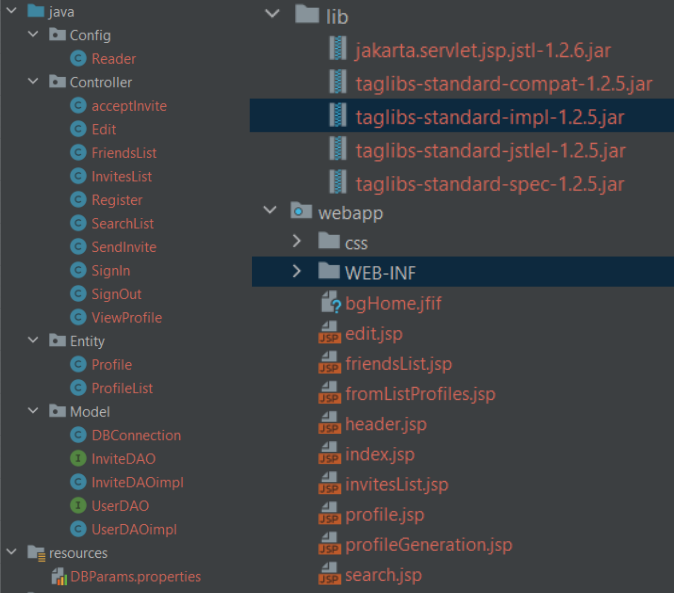


Рисунок 4.1 – Загальна структура проекту

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

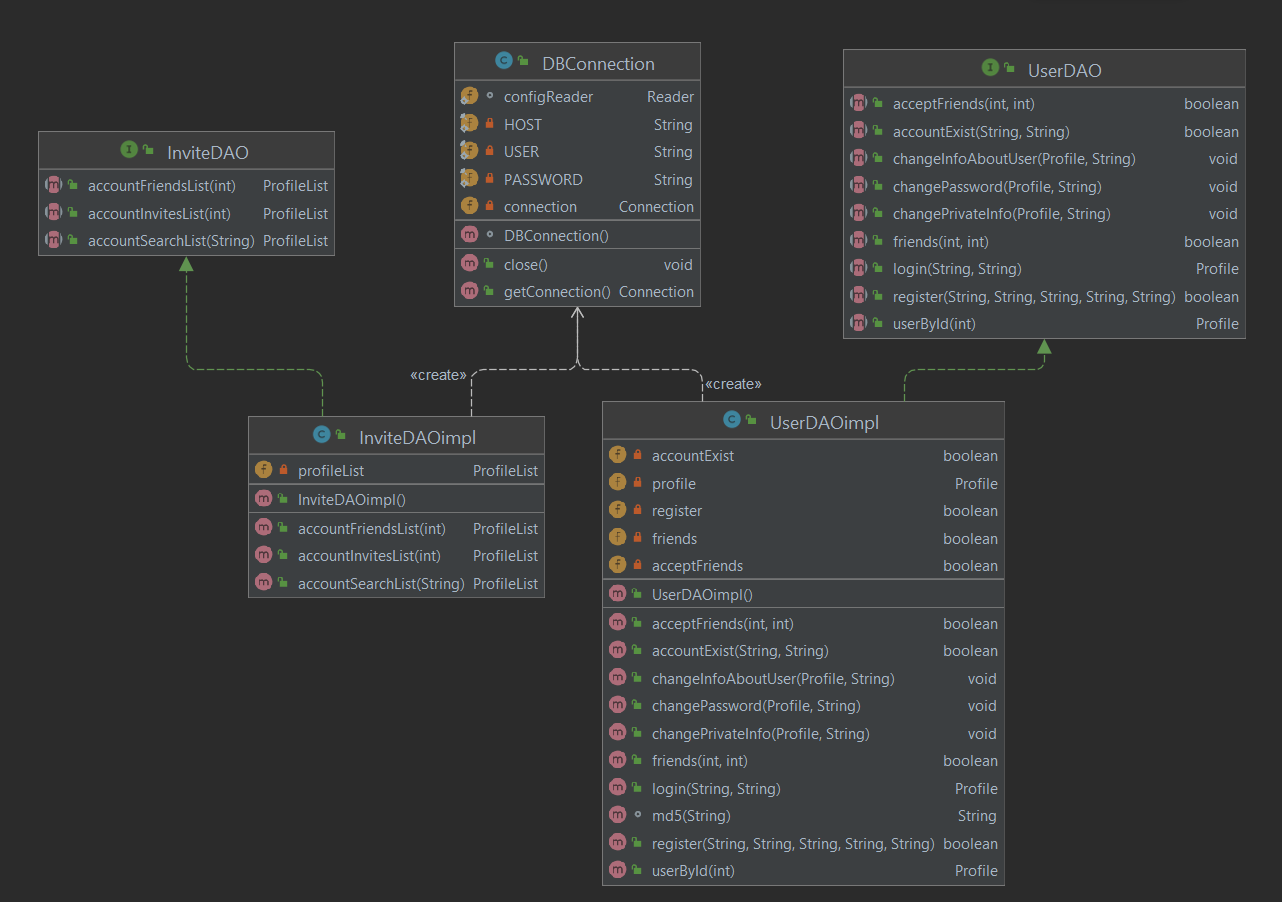
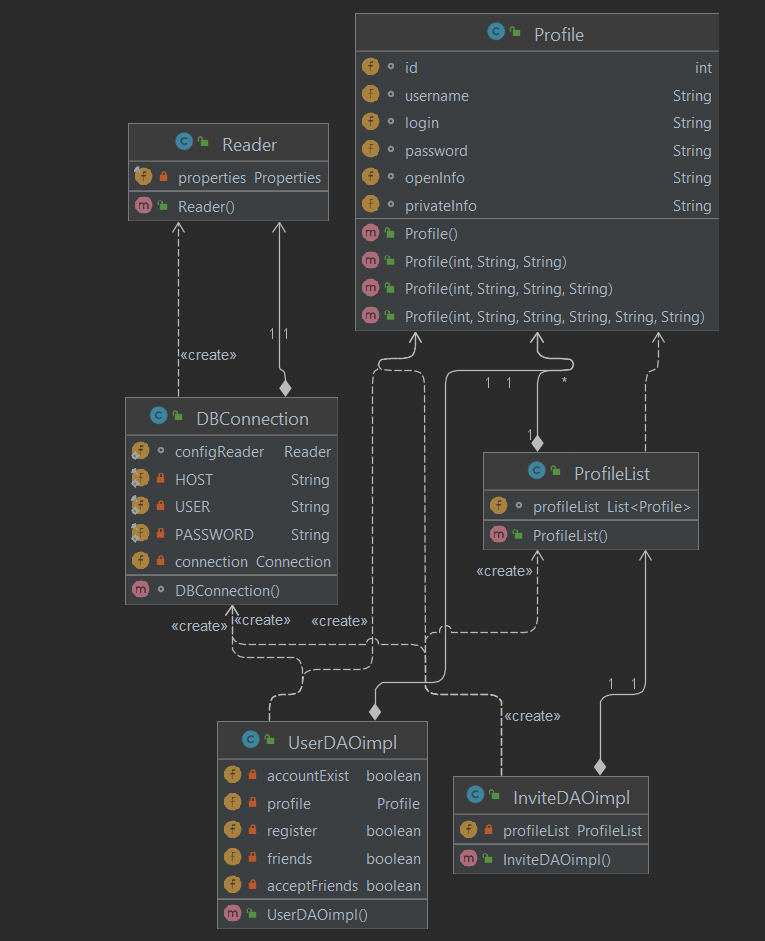
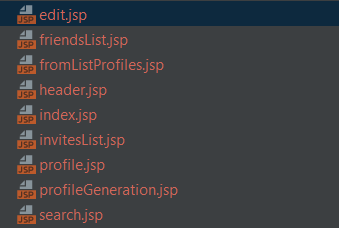


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

## 4.3 Компоненти рівня бізнес-логіки

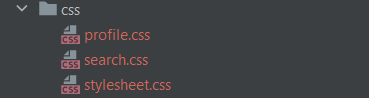


## 4.4 Компоненти рівня інтерфейсу користувача



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 Badoo, проаналізовано переваги та недоліки цієї ідеї та задумки, виконання основної роботи та проаналізований подальший розвиток проекту.

Було прийнято рішення щодо подальшого розвитку проекту:

1. Додавши функцію автоматичний підбір користувачів, яких немає у списку друзів, та користувачів заявки яких не були відкинуті користувачем.
2. Додавши функцію чату між користувачами, для підтримання зв’язку між користувачами що є друзями.
3. Додавши преміум статус, для підтримання серверів, та забезпечення прибутку компанії за рахунок сайту.
4. Додавши функцію відео, та аудіо дзвінків, для забезпечення безпечного зв’язку між користувачами що мають преміум статус.
5. Додавши фільтрування користувачів.
6. Додавши права доступу типу «Адміністратор».
7. Додавши функції «Бан, мут» для адміністраторів та «Блокування користувача(Чорний список)» для користувачів.
8. Додавши можливість створювати групи для спілкування користувачів.
9. Додавши статуси для групи та певного виду преміум можливості для груп.
10. Додавши можливість зміни теми сайту(Світла в темну, та інші)
11. Додавши можливість зміни теми чату з співрозмовником(Преміум)
12. Додавши Меню налаштувань для аккаунту, з метою забезпечення особистої безпеки користувачів, підтримки конфіденційності та інші налаштування.
13. Додавши можливість підтягувати дані з Google API, Facebook API, та інших можливих засобів, для швидкої реєстрації та входу.
14. Додавши зберігання cookie для забезпечення зручності роботи з поточним аккаунтом

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 ДАО, 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, та підбираючи ідеальні параметри у % для гарного відображення на різних девайсах, та використання max\_width для формування іншого стилю для девайсів що мають мале розширення, щоб користувач бачив нормальний розмір обєктів при малому екрані.

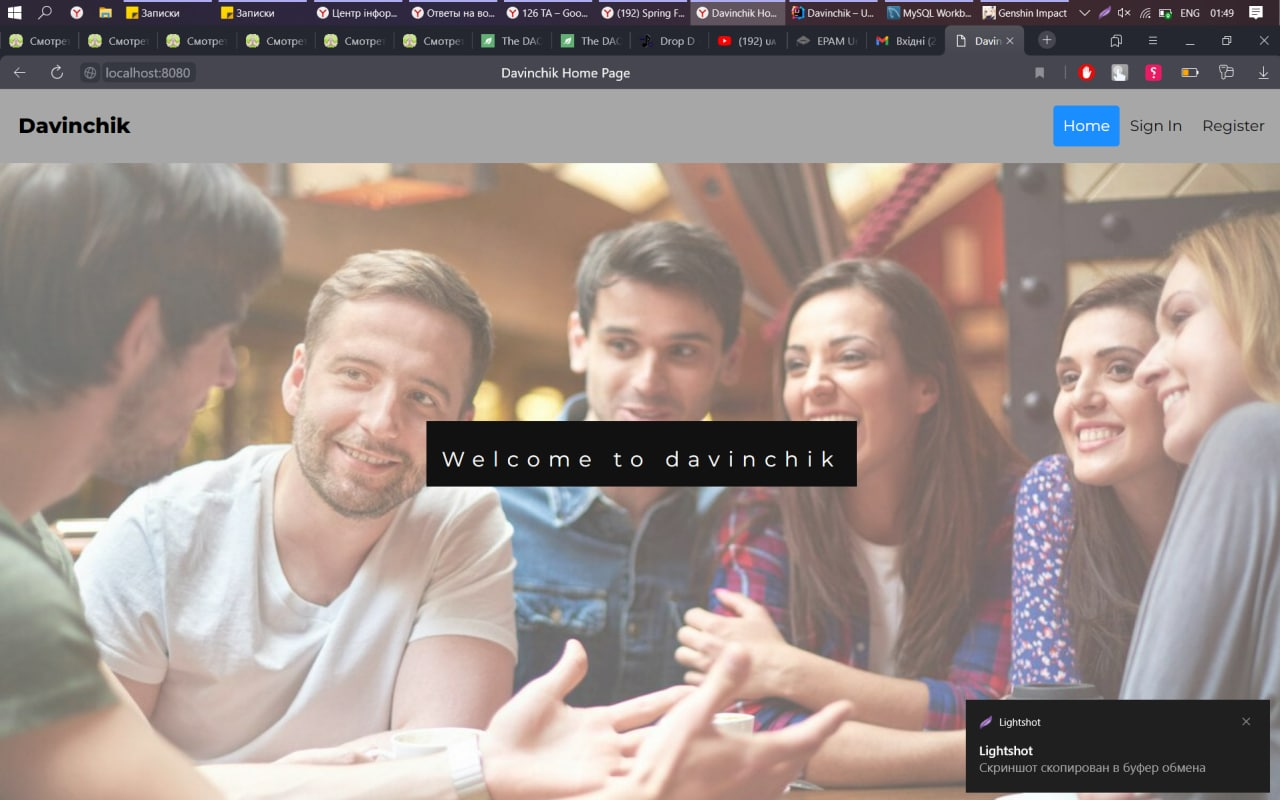
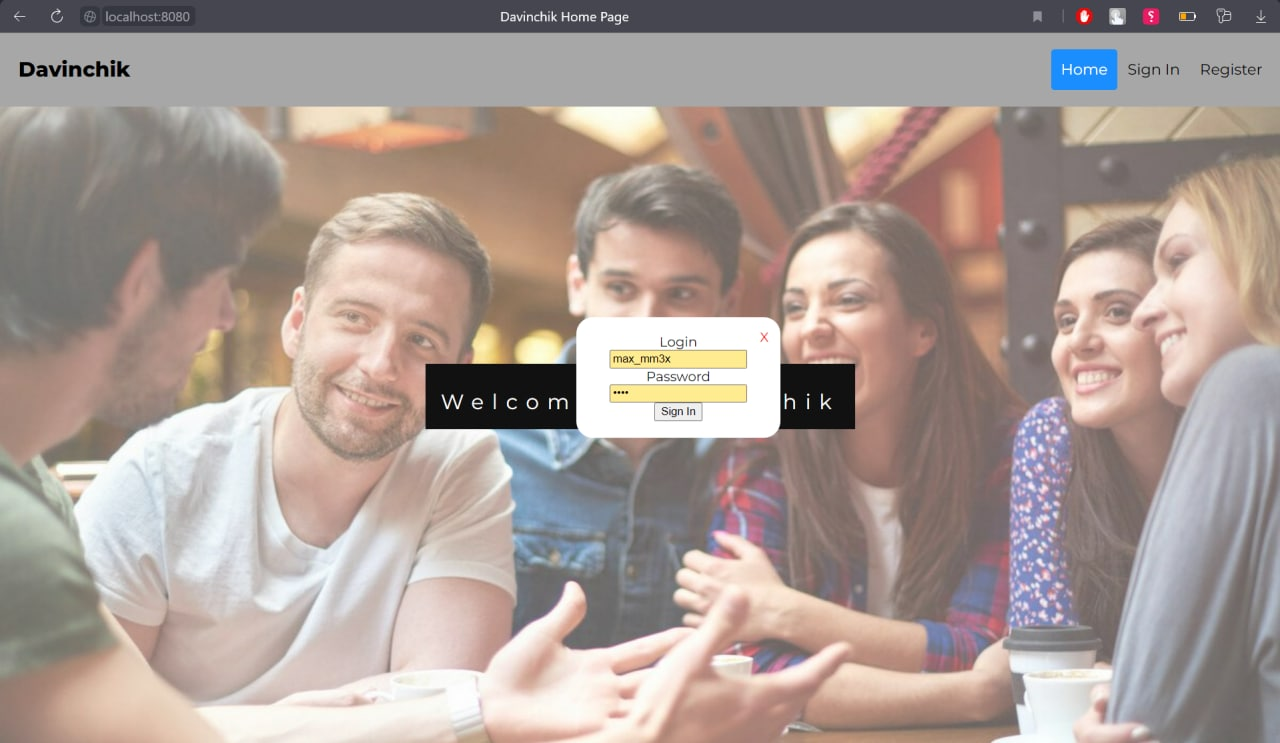
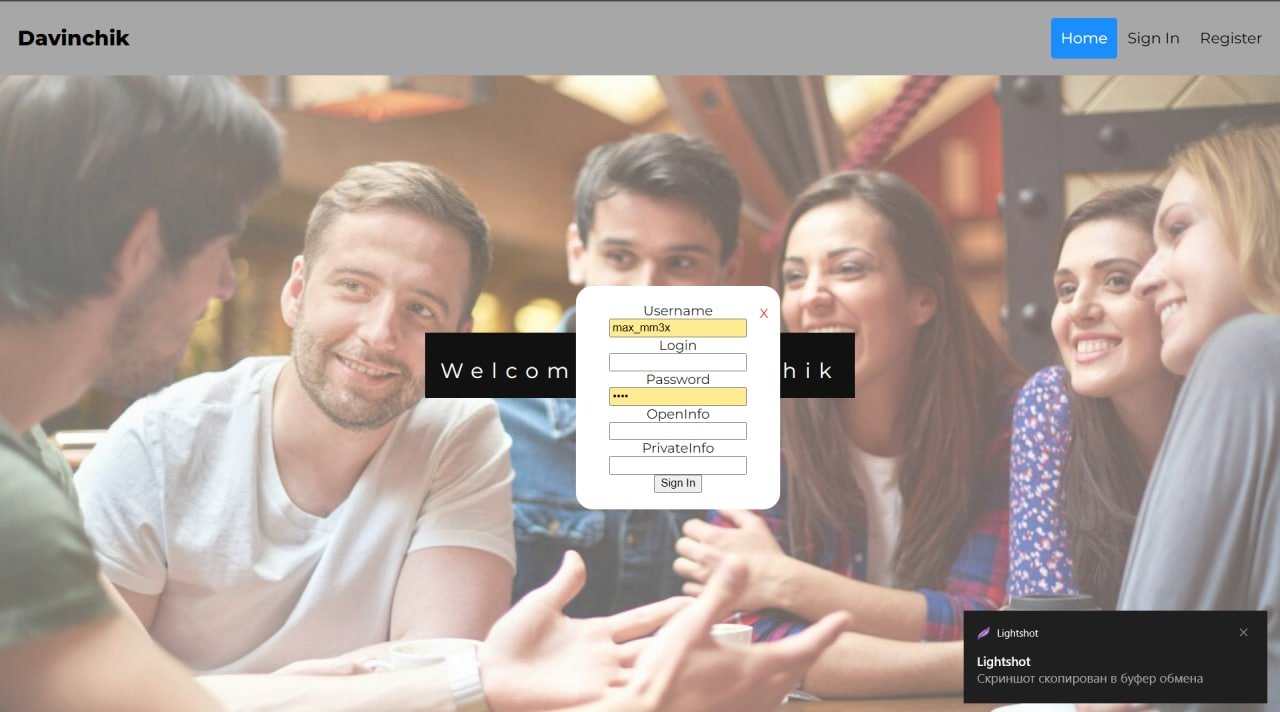
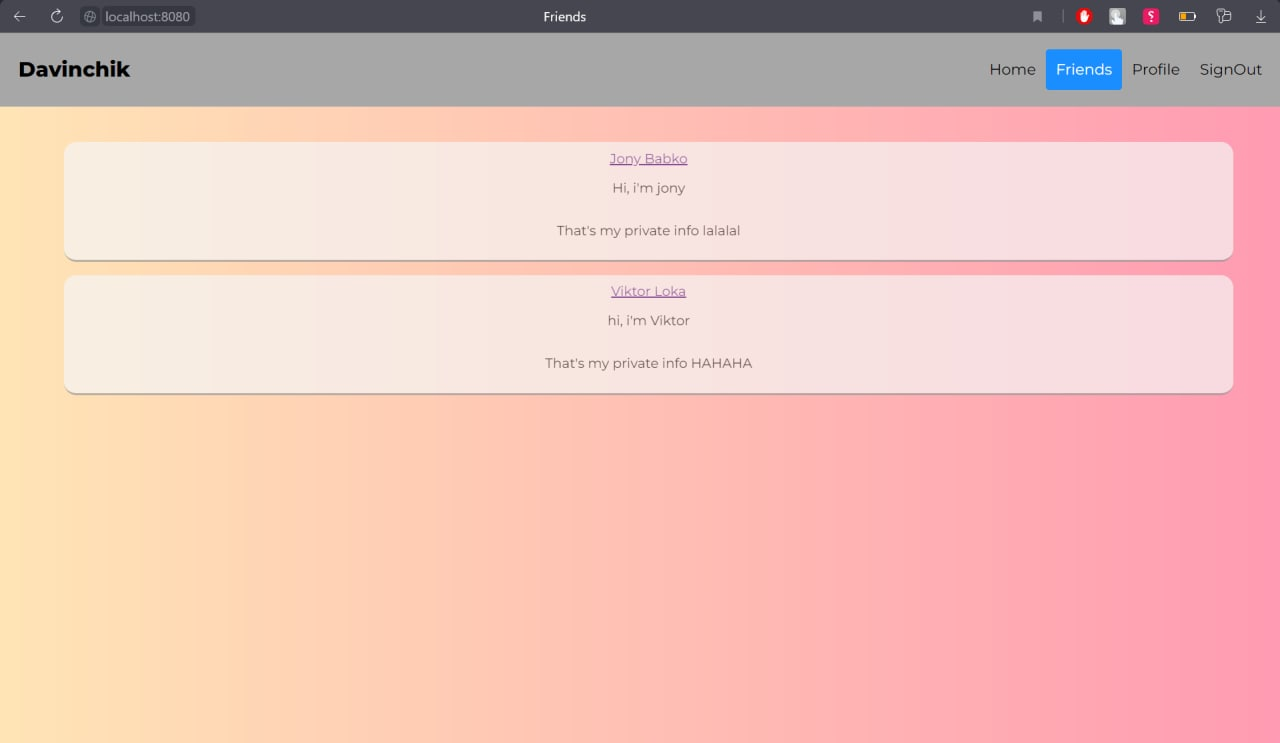
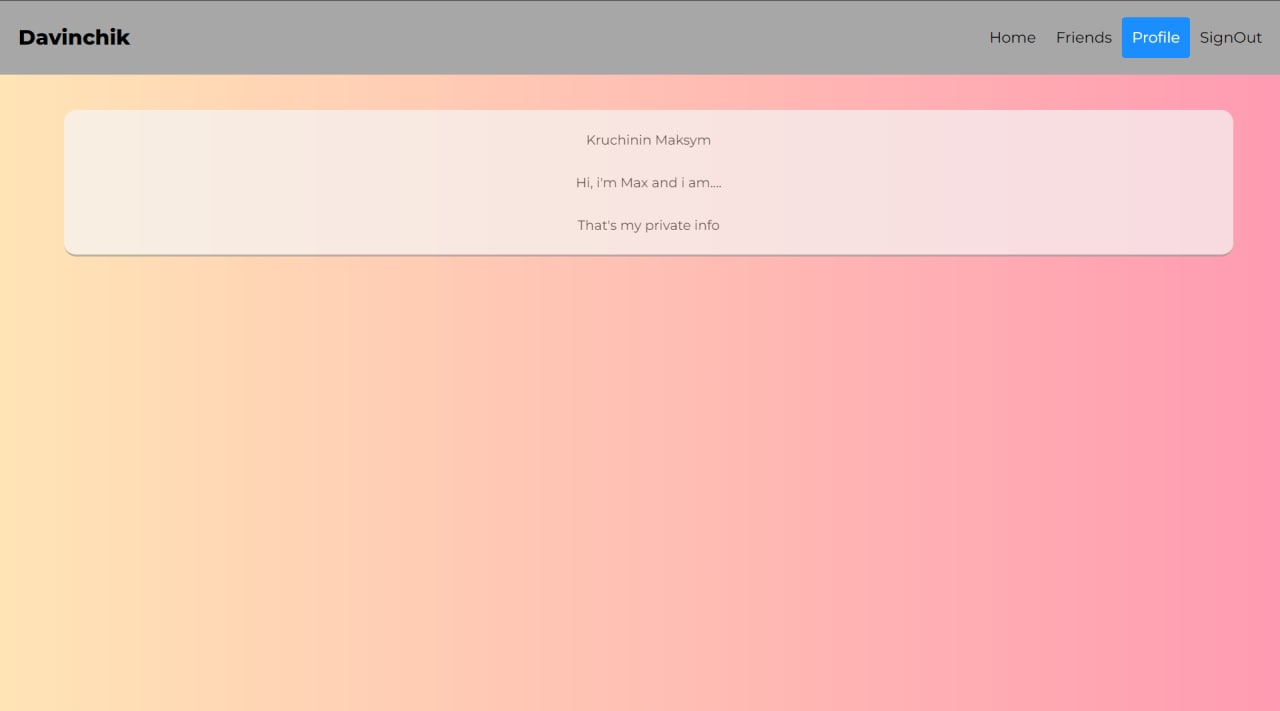
# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Вся інформація була використана з ресурсів <https://learn.epam.com/>

(В особливості з курсу Java Online Program )

Для аналізу ідеї роботи був використаний сайт <https://badoo.com/uk/>

# ДОДАТКИ

package Config;import java.io.IOException;import java.io.InputStream;import java.util.Properties;public class Reader { private final Properties properties = new Properties(); public Reader() { String filePath = "DBParams.properties"; try (InputStream out = getClass().getClassLoader().getResourceAsStream(filePath)) { properties.load(out); } catch (IOException e) { e.printStackTrace(); } } public String getProperty(String propertyName){ return properties.getProperty(propertyName); }}package Controller;import Entity.Profile;import Model.UserDAO;import Model.UserDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "acceptInvite", value = "/acceptInvite")public class acceptInvite extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { Profile currentProfile = (Profile) request.getSession().getAttribute("userData"); int id = currentProfile.getId(); int id1 = Integer.parseInt(request.getParameter("id")); UserDAO user = new UserDAOimpl(); if (user.acceptFriends(id, id1)){ request.getRequestDispatcher("/profileGeneration.jsp?id="+id1).forward(request, response); } else response.sendError(404); } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { }}package Controller;import Entity.Profile;import Model.UserDAO;import Model.UserDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "Edit", value = "/Edit")public class Edit extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { UserDAO user = new UserDAOimpl(); String password = request.getParameter("password"); String openInfo = request.getParameter("openInfo"); String privateInfo = request.getParameter("privateInfo"); Profile profile = (Profile) request.getSession().getAttribute("userData"); if (password != null) { user.changePassword(profile, password); } if (openInfo != null) { user.changeInfoAboutUser(profile, openInfo); } if (privateInfo != null) { user.changePrivateInfo(profile, privateInfo); } request.getRequestDispatcher("/edit.jsp").forward(request, response); }}package Controller;import Entity.Profile;import Entity.ProfileList;import Model.InviteDAO;import Model.InviteDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "FriendsList", value = "/FriendsList")public class FriendsList extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { Object logged = request.getSession().getAttribute("logged"); if (logged != null) { if ((boolean)logged == true) { Profile profile = (Profile) request.getSession().getAttribute("userData"); InviteDAO invite = new InviteDAOimpl(); ProfileList profileList = invite.accountFriendsList(profile.getId()); request.getSession().setAttribute("friendsList", profileList); request.getSession().setAttribute("active", 2); request.getRequestDispatcher("/friendsList.jsp").forward(request, response); } else response.sendError(404, "not logged"); } } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { }}package Controller;import Entity.Profile;import Entity.ProfileList;import Model.InviteDAO;import Model.InviteDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "InvitesList", value = "/InvitesList")public class InvitesList extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { Profile profile = (Profile) request.getSession().getAttribute("userData"); InviteDAO invite = new InviteDAOimpl(); ProfileList profileList = invite.accountInvitesList(profile.getId()); request.getSession().setAttribute("invitesList", profileList); request.getRequestDispatcher("/invitesList.jsp").forward(request, response); } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { }}package Controller;import Model.UserDAO;import Model.UserDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;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 username = request.getParameter("username"); String login = request.getParameter("login"); String password = request.getParameter("password"); String openInfo = request.getParameter("openInfo"); String privateInfo = request.getParameter("privateInfo"); UserDAO profile = new UserDAOimpl(); if (profile.accountExist(login, password)==false) { profile.register(username, login, password, openInfo, privateInfo); request.getSession().setAttribute("logged", true); request.getSession().setAttribute("active", 1); request.getSession().setAttribute("userData", profile.login(login, password)); } request.getRequestDispatcher("index.jsp").forward(request, response); }}package Controller;import Entity.ProfileList;import Model.InviteDAO;import Model.InviteDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "SearchList", value = "/SearchList")public class SearchList 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 search = request.getParameter("searchName"); InviteDAO invite = new InviteDAOimpl(); ProfileList profileList = invite.accountSearchList(search); request.getSession().setAttribute("searchList", profileList); request.getRequestDispatcher("/search.jsp").forward(request, response); }}package Controller;import Entity.Profile;import Model.UserDAO;import Model.UserDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "SendInvite", value = "/SendInvite")public class SendInvite extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { Profile currentProfile = (Profile) request.getSession().getAttribute("userData"); int id = currentProfile.getId(); int id1 = Integer.parseInt(request.getParameter("id")); UserDAO user = new UserDAOimpl(); if (user.acceptFriends(id1, id)){ request.getRequestDispatcher("/profileGeneration.jsp?id="+id1).forward(request, response); } else response.sendError(404); } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { }}package Controller;import Entity.Profile;import Model.UserDAO;import Model.UserDAOimpl;import javax.servlet.ServletException;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "SignIn", value = "/SignIn")public class SignIn extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) { } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { String login = request.getParameter("login"); String password = request.getParameter("password"); UserDAO user = new UserDAOimpl(); Profile profile = user.login(login, password); if (profile != null) { request.getSession().setAttribute("logged", true); request.getSession().setAttribute("userData", profile); }else response.sendError(404, "Wrong Username or Password"); request.getRequestDispatcher("index.jsp").forward(request, response); }}package Controller;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "SignOut", value = "/SignOut")public class SignOut extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { request.getSession().setAttribute("logged", false); request.getSession().setAttribute("active", 1); request.getSession().setAttribute("userData", false); request.getRequestDispatcher("index.jsp").forward(request, response); } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { }}package Controller;import Entity.Profile;import Model.UserDAO;import Model.UserDAOimpl;import javax.servlet.\*;import javax.servlet.http.\*;import javax.servlet.annotation.\*;import java.io.IOException;@WebServlet(name = "ViewProfile", value = "/ViewProfile")public class ViewProfile extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException { int id = Integer.parseInt(request.getParameter("id")); UserDAO user = new UserDAOimpl(); Profile profile = user.userById(id); Profile currentProfile = (Profile) request.getSession().getAttribute("userData"); if (profile != null) { if (currentProfile != null) { if (!user.friends(currentProfile.getId(), profile.getId())) { profile.setPrivateInfo(null); } } else { profile.setPrivateInfo(null); } request.getSession().setAttribute("profile", profile); request.getRequestDispatcher("/profileGeneration.jsp").forward(request, response); } else { response.sendError(404, "userNotExist"); } } @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) { }}package Entity;public class Profile { int id; String username; String login; String password; String openInfo; String privateInfo; public Profile(int id, String username, String login, String password, String openInfo, String privateInfo) { this.id = id; this.username = username; this.login = login; this.password = password; this.openInfo = openInfo; this.privateInfo = privateInfo; } public Profile(int id, String username, String openInfo) { this.id = id; this.username = username; this.openInfo = openInfo; } public Profile() { } public Profile(int id, String username, String openInfo, String privateInfo) { this.id = id; this.username = username; this.openInfo = openInfo; this.privateInfo = privateInfo; } public int getId() { return id; } public void setId(int id) { this.id = id; } public String getUsername() { return username; } public void setUsername(String username) { this.username = username; } public String getLogin() { return login; } public void setLogin(String login) { this.login = login; } public String getPassword() { return password; } public void setPassword(String password) { this.password = password; } public String getOpenInfo() { return openInfo; } public void setOpenInfo(String openInfo) { this.openInfo = openInfo; } public String getPrivateInfo() { return privateInfo; } public void setPrivateInfo(String privateInfo) { this.privateInfo = privateInfo; }}package Entity;import java.util.ArrayList;import java.util.List;public class ProfileList { public ProfileList() { this.profileList = new ArrayList<>(); } List<Profile> profileList; public List<Profile> getProfileList() { return profileList; } public void addProfileList(Profile profile) { profileList.add(profile); }}package Model;import Config.Reader;import java.sql.Connection;import java.sql.DriverManager;import java.sql.SQLException;public class DBConnection implements AutoCloseable{ static Reader configReader = new Reader(); 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; 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;import Entity.ProfileList;public interface InviteDAO { ProfileList accountInvitesList(int id); ProfileList accountFriendsList(int id); ProfileList accountSearchList(String likeAn);}package Model;import Entity.Profile;import Entity.ProfileList;import java.sql.Connection;import java.sql.PreparedStatement;import java.sql.ResultSet;public class InviteDAOimpl implements InviteDAO { private ProfileList profileList; @Override public ProfileList accountInvitesList(int idF) { String query = "SELECT idProfile, username, openInfo FROM invitelist e\n" + "INNER JOIN profile p ON p.idProfile = e.idProfileTo\n" + "where p.idProfile=(select table1.idProfileTo from " + "(SELECT idProfileFrom, idProfileTo FROM invitelist union " + "SELECT idProfileTo, idProfileFrom FROM invitelist) table1\n" + "left join (SELECT \* FROM invitelist) table2 On table1.idProfileTo=table2.idProfileTo " + "and table1.idProfileFrom=table2.idProfileFrom " + "where table2.idProfileTo is null and table1.idProfileFrom=?\n" + ")"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setInt(1, idF); ResultSet elements = preparedStatement.executeQuery(); profileList=new ProfileList(); while (elements.next()){ int id = elements.getInt(1); String username = elements.getString(2); String openInfo = elements.getString(3); profileList.addProfileList(new Profile(id, username, openInfo)); } } catch (Exception e) { e.printStackTrace(); } return profileList; } @Override public ProfileList accountFriendsList(int idF) { String query = "SELECT idProfile, username, openInfo, privateInfo FROM invitelist e\n" + "INNER JOIN profile p ON p.idProfile = e.idProfileTo\n" + "where e.idProfileFrom=? and \n" + "e.idProfileTo in (SELECT idProfileFrom FROM invitelist where idProfileTo=?);"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setInt(1, idF); preparedStatement.setInt(2, idF); ResultSet elements = preparedStatement.executeQuery(); profileList=new ProfileList(); while (elements.next()){ int id = elements.getInt(1); String username = elements.getString(2); String openInfo = elements.getString(3); String privateInfo = elements.getString(4); profileList.addProfileList(new Profile(id, username, openInfo, privateInfo)); } } catch (Exception e) { e.printStackTrace(); } return profileList; } @Override public ProfileList accountSearchList(String likeAn) { String query = "SELECT idProfile, username, openInfo FROM profile \n" + "where username LIKE ?;"; if(likeAn.isEmpty()){return null;} try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, likeAn); ResultSet elements = preparedStatement.executeQuery(); profileList=new ProfileList(); while (elements.next()){ int id = elements.getInt(1); String username = elements.getString(2); String openInfo = elements.getString(3); profileList.addProfileList(new Profile(id, username, openInfo)); } } catch (Exception e) { e.printStackTrace(); } return profileList; }}package Model;import Entity.Profile;public interface UserDAO { boolean accountExist(String login, String password); Profile login(String login, String password); boolean register(String userName, String login, String password, String infoAboutUser, String privateInfo); void changePassword(Profile profile, String passwordToChange); void changeInfoAboutUser(Profile profile, String infoAboutUser); void changePrivateInfo(Profile profile, String infoAboutUser); Profile userById(int id); boolean friends(int id, int id1); boolean acceptFriends(int id, int id1);}package Model;import Entity.Profile;import org.apache.commons.codec.digest.DigestUtils;import java.sql.Connection;import java.sql.PreparedStatement;import java.sql.ResultSet;public class UserDAOimpl implements UserDAO { private boolean accountExist; private Profile profile; private boolean register; private boolean friends; private boolean acceptFriends; @Override public boolean accountExist(String login, String password) { String query = "SELECT COUNT(\*) FROM profile WHERE login=? and password=?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, login); preparedStatement.setString(2, password); ResultSet resultSet = preparedStatement.executeQuery(); resultSet.next(); if (resultSet.getInt(1) > 0) { accountExist = true; } else { accountExist = false; } } catch (Exception e) { e.printStackTrace(); } return accountExist; } @Override public Profile login(String login, String password) { String query = "SELECT idProfile, username, openInfo, privateInfo FROM profile WHERE login=? and password=?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, login); preparedStatement.setString(2, md5(password)); ResultSet elements = preparedStatement.executeQuery(); while (elements.next()) { int id = elements.getInt(1); String username = elements.getString(2); String openInfo = elements.getString(3); String privateInfo = elements.getString(4); profile = new Profile(id, username, login, password, openInfo, privateInfo); } } catch (Exception e) { e.printStackTrace(); } return profile; } @Override public boolean register(String username, String login, String password, String infoAboutUser, String privateInfo) { register = false; String query = "insert profile(username, login, password, openInfo, privateInfo)\n" + " VALUES (?, ?, ?, ?, ?);"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, username); preparedStatement.setString(2, login); preparedStatement.setString(3, md5(password)); preparedStatement.setString(4, infoAboutUser); preparedStatement.setString(5, privateInfo); preparedStatement.executeUpdate(); register = true; } catch (Exception e) { e.printStackTrace(); } return register; } @Override public void changePassword(Profile profile, String passwordToChange) { String query = "UPDATE profile set password=? where login = ? and idProfile = ?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, passwordToChange); preparedStatement.setString(2, profile.getLogin()); preparedStatement.setInt(3, profile.getId()); } catch (Exception e) { e.printStackTrace(); } } @Override public void changeInfoAboutUser(Profile profile, String infoAboutUser) { String query = "UPDATE profile set openInfo=? where login = ? and idProfile = ?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, infoAboutUser); preparedStatement.setString(2, profile.getLogin()); preparedStatement.setInt(3, profile.getId()); } catch (Exception e) { e.printStackTrace(); } } @Override public void changePrivateInfo(Profile profile, String infoAboutUser) { String query = "UPDATE profile set privateInfo=? where login = ? and idProfile = ?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setString(1, infoAboutUser); preparedStatement.setString(2, profile.getLogin()); preparedStatement.setInt(3, profile.getId()); } catch (Exception e) { e.printStackTrace(); } } @Override public Profile userById(int id) { String query = "SELECT username, openInfo, privateInfo FROM profile WHERE idProfile=?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setInt(1, id); ResultSet elements = preparedStatement.executeQuery(); while (elements.next()) { String username = elements.getString(1); String openInfo = elements.getString(2); String privateInfo = elements.getString(3); profile = new Profile(id, username, openInfo, privateInfo); } } catch (Exception e) { e.printStackTrace(); } return profile; } @Override public boolean friends(int id, int id1) { String query = "SELECT COUNT(\*) FROM invitelist Where idProfileFrom=? and idProfileTo=?;"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setInt(1, id); preparedStatement.setInt(2, id1); ResultSet elements = preparedStatement.executeQuery(); elements.next(); if (elements.getInt(1) > 0) { friends = true; } else { friends = false; } } catch (Exception e) { e.printStackTrace(); } return friends; } @Override public boolean acceptFriends(int id, int id1) { acceptFriends = false; String query = "insert invitelist(idProfileFrom, idProfileTo) values (?, ?);"; try (DBConnection db = new DBConnection()) { Connection connection = db.getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(query); preparedStatement.setInt(1, id); preparedStatement.setInt(2, id1); preparedStatement.executeUpdate(); acceptFriends = true; } catch (Exception e) { e.printStackTrace(); } return acceptFriends; } String md5(String password) { return DigestUtils.md5Hex(password).toLowerCase(); }}body{ background: rgb(255,229,182); background: linear-gradient(90deg, rgba(255,229,182,1) 0%, rgba(255,155,178,1) 100%);}div.container{ background-color: whitesmoke; opacity: 70%; display: flex; flex-direction: column; align-items: center; position: relative; top: 5%; width: 90%; left: 5%; padding: 10px; border-radius: 15px; margin-bottom: 1%; border-bottom-style: outset;}input.search{ width: 50%; height: 5%; font-size: 200%; text-align: center;}button.search{ border: none; background-color: #d5d5d5; color: black; padding: 2% 7%; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; cursor: pointer;}div.search{ top: 1%; position: relative; text-align: center;}@import url('https://fonts.googleapis.com/css?family=Montserrat:400,800');body { margin: 0; overflow: hidden; height: 100%; font-family: 'Montserrat', sans-serif;}html{ margin: 0; height: 100%;}.bgImage{ margin: 0; height: 100%; background-image:url("/bgHome.jfif"); opacity: 0.65; background-position: center; background-repeat: no-repeat; background-size: cover;}.header { overflow: hidden; background-color: #a7a7a7; padding: 20px 10px;}button:active { transform: scale(0.95);}.header a { float: left; color: black; text-align: center; padding: 12px; text-decoration: none; font-size: 18px; line-height: 25px; border-radius: 4px; cursor: pointer;}.header a.logo { font-size: 25px; font-weight: bold;}.header a:hover { background-color: #d9d9d9; color: black;}.header a.active { background-color: #1a8dff; color: white;}.header a.active:hover { background-color: #676767; color: black;}.header-right { float: right;}.caption { position: absolute; left: 0; top: 50%; width: 100%; text-align: center; color: #5b5b5b; z-index: 1;}.caption span { background-color: #111; color: #fff; padding: 18px; padding-top: 30px; font-size: 25px; letter-spacing: 10px;}div.hidden{ display: none;}div.hidden.show{ display: block; position: absolute; top: 40%; z-index: 10; left: 45%; text-align: -webkit-center;}div.hidden \*{ z-index: 10; display: block;}form.openForm{ background-color: white; border-radius: 20px; height: 20%; padding: 20px; width: 100%;}#closeButtonReg{ color: red; position: relative; width: 10px; height: 10px; top: -265px; left: 60%; background: center;}#closeButtonSign{ color: red; position: relative; width: 10px; height: 10px; top: -150px; left: 60%; background: center;}@media screen and (max-width: 500px) { .header a { float: none; display: block; text-align: left; } .header-right { float: none; }}<%@ page contentType="text/html;charset=UTF-8" language="java" %><html><head> <title>Edit</title> <link href="css/profile.css" rel="stylesheet"></head><body><%request.getSession().setAttribute("active", 3);%><jsp:include page="header.jsp"></jsp:include><div class="container"> <form action="/Edit" method="post"> <label>Change Password</label> <input type="password" name="password"> <label>Change OpenInfo</label> <input type="text" name="openInfo"> <label>Change PrivateInfo</label> <input type="text" name="privateInfo"> <button type="submit">Sign In</button> </form></div></body></html><%@ page contentType="text/html;charset=UTF-8" language="java" %><html><head> <title>Friends</title></head><body><jsp:include page="header.jsp"></jsp:include><jsp:include page="fromListProfiles.jsp"></jsp:include></body></html><%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %><%@ page contentType="text/html;charset=UTF-8" language="java" %><link href="css/profile.css" rel="stylesheet"><c:choose> <c:when test="${friendsList!=null}"> <c:forEach var="list" items="${friendsList.profileList}" > <div class="container"> <a href="/ViewProfile?id=${list.id}">${list.username}</a> <p>${list.openInfo}</p> <c:if test="${list.privateInfo!=null}"> <p>${list.privateInfo}</p> </c:if> </div> </c:forEach> ${friendsList=null} </c:when> <c:when test="${invitesList!=null}"> <c:forEach items="${invitesList.profileList}" var="list"> <div class="container"> <a href="/ViewProfile?id=${list.id}">${list.username}</a> <br> <p>${list.openInfo}</p> <c:if test="${list.privateInfo!=null}"> <br> <p>${list.privateInfo}</p> </c:if> <a href="/acceptInvite?id=${list.id}">Accept</a> </div> </c:forEach> ${invitesList=null} </c:when> <c:when test="${searchList!=null}"> <c:forEach items="${searchList.profileList}" var="list"> <div class="container"> <a href="/ViewProfile?id=${list.id}">${list.username}</a> <br> <p>${list.openInfo}</p> <a href="/SendInvite?id=${list.id}">Send friends request</a> </div> </c:forEach> ${searchList=null} </c:when></c:choose><%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %><%@ page contentType="text/html;charset=UTF-8" language="java" %><link href="css/stylesheet.css" rel="stylesheet"><div class="header"> <a href="/" class="logo">Davinchik</a> <c:choose> <c:when test="${logged!=true}"> <div class="header-right"> <a class="active" href="/">Home</a> <a onclick="showForm('signIn')">Sign In</a> <a onclick="showForm('register')">Register</a> </div> </c:when> <c:otherwise> <div class="header-right"> <a <c:if test="${active==1}">class="active"</c:if> href="/index.jsp">Home</a> <a <c:if test="${active==2}">class="active"</c:if> href="/FriendsList">Friends</a> <a <c:if test="${active==3}">class="active"</c:if> href="/profile.jsp">Profile</a> <a <c:if test="${active==4}">class="active"</c:if> href="/search.jsp">Search</a> <a <c:if test="${active==5}">class="active"</c:if> href="/InvitesList">Invites</a> <a href="/SignOut">SignOut</a> </div> </c:otherwise> </c:choose></div><%@ 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>Davinchik Home Page</title> <link href="css/stylesheet.css" rel="stylesheet"> <script> function showForm(elementId) { document.getElementById('signIn').parentElement.classList.remove('show'); document.getElementById('register').parentElement.classList.remove('show'); document.getElementById(elementId).parentElement.classList.add('show'); } function closeForm(elementId) { document.getElementById(elementId).parentElement.classList.remove('show'); } </script></head><body><%request.getSession().setAttribute("active", 1);%><jsp:include page="header.jsp"></jsp:include><div class="caption"><span>Welcome to davinchik</span></div><div class="hidden"> <form action="/SignIn" method="post" class="openForm" id="signIn"> <label>Login</label> <input type="text" name="login"> <label>Password</label> <input type="password" name="password"> <button type="submit">Sign In</button> </form> <a id="closeButtonSign" class="header" onclick="closeForm('signIn')">X</a></div><div class="hidden"> <form action="/Register" method="post" class="openForm" id="register"> <label>Username</label> <input type="text" name="username"> <label>Login</label> <input type="text" name="login"> <label>Password</label> <input type="password" name="password"> <label>OpenInfo</label> <input type="text" name="openInfo"> <label>PrivateInfo</label> <input type="text" name="privateInfo"> <button type="submit">Sign In</button> </form> <a id="closeButtonReg" class="header" onclick="closeForm('register')">X</a></div><div class="bgImage" style="z-index: -1"></div></body></html><%@ page contentType="text/html;charset=UTF-8" language="java" %><html><head> <title>InviteList</title></head><body><%request.getSession().setAttribute("active", 5);%><jsp:include page="header.jsp"></jsp:include><jsp:include page="fromListProfiles.jsp"></jsp:include></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>Profile</title> <link href="css/profile.css" rel="stylesheet"></head><body><%request.getSession().setAttribute("active", 3);%><jsp:include page="header.jsp"></jsp:include><div class="container"> <p>${userData.username}</p> <p>${userData.openInfo}</p> <p>${userData.privateInfo}</p> <a href="/edit.jsp"></a></div></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>Profile</title> <link href="css/profile.css" rel="stylesheet"></head><body><jsp:include page="header.jsp"></jsp:include><div class="container"> <p>${profile.username}</p> <p>${profile.openInfo}</p> <p>${profile.privateInfo}</p></div></body></html><%@ page contentType="text/html;charset=UTF-8" language="java" %><html><head> <title>Search</title> <link href="css/search.css" rel="stylesheet"></head><body><%request.getSession().setAttribute("active", 4);%><jsp:include page="header.jsp"></jsp:include><div class="search"> <form action="SearchList" method="post"> <input class="search" type="text" name="searchName"> <button class="search" type="submit">SEARCH</button> </form></div><jsp:include page="fromListProfiles.jsp"></jsp:include></body></html>