**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

факультет інформатики та обчислювальної техніки

(повна назва інституту/факультету)

кафедра автоматика та управління в технічних системах

(повна назва кафедри)

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

**Курсова робота**

з дисципліни «Програмування-2»

на тему: Інформаційно-довідкова система «Бюро знахідок»

Виконав : студент \_\_1\_ курсу, групи \_\_\_\_ІА-02\_\_\_\_\_

(шифр групи)

Кульбаневич Микита Сергійович \_\_\_\_\_\_\_\_\_

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

Науковий керівник \_\_\_\_\_\_\_\_\_\_Міщенко В. О.\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

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

Члени комісії \_\_\_\_\_\_\_\_\_\_Бердник Ю. М. \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(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

Кульбаневич М. С. \_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2021

ЗМІСТ

[ВСТУП 3](#_Toc40064741)

[1 ВИМОГИ ДО СИСТЕМИ 4](#_Toc40064742)

[1.1 Функціональні вимоги до системи 4](#_Toc40064743)

[1.2 Нефункціональні вимоги до системи 4](#_Toc40064744)

[2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ 5](#_Toc40064745)

[2.1 Діаграма прецедентів 6](#_Toc40064746)

[2.2 Опис сценаріїв використання системи 7](#_Toc40064747)

[3 АРХІТЕКТУРА СИСТЕМИ 24](#_Toc40064748)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 26](#_Toc40064749)

[4.1 Загальна структура проекту 27](#_Toc40064750)

[4.2 Компоненти рівня доступу до даних 27](#_Toc40064751)

[4.3 Компоненти рівня бізнес-логіки 27](#_Toc40064752)

[4.4 Компоненти рівня 27](#_Toc40064753)

[ВИСНОВКИ 28](#_Toc40064754)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 30](#_Toc40064755)

# ВСТУП

З розвитком інформаційних технологій та зацікавленістю людей до нових гаджетів зросла і кількість втрат речей, а саме розумної електроніки: смартфонів, смарт-годинників, фітнес-трекерів, навушників (особливо бездротових, які простіше загубити) та ін. Разом з цим з’являється потреба у системах, що дозволяють власникам загублених речей повернути їх. Найвідоміші онлайн-системи бюро знахідок реалізовані у соціальних мережах, що змушує людей, що не зареєстровані у соцмережі – реєструватися там, реалізувати свої потреби, та видаляти свій акаунт. Звідси у користувачів випливає потреба у сервісі, що призначений тільки для пошуку загублених речей та публікуванні оголошень про знахідки.

Метою роботи є створення системи, що дозволяє людям, що знайшли знахідку, подавати оголошення, а людям, що втратили річ - знайти. Для цього система повинна мати певний перелік властивостей та вирішувати такі задачі:

* система повинна бути інтуїтивно зрозумілою для усіх користувачів, мати “user-friendly” інтерфейс, продуману навігацію;
* система повинна мати просту реєстрацію, що буде дозволяти легко створювати та видаляти свої оголошення;
* оголошення мають містити належну інформацію та бути доступними для перегляду незареєстрованими користувачами.

# 1 ВИМОГИ ДО СИСТЕМИ

## 1.1 Функціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

- незареєстрований користувач повинен мати можливість переглядати інформацію про знахідку, а саме заголовок, опис знахідки, ключові слова та контактну інформацію;

- незареєстрований користувач повинен мати можливість здійснювати пошук

оголошень про знахідки по ключовим словам;

- зареєстрований користувач повинен мати усі можливості, що є у незареєстрованого користувача, а також він повинен мати можливість створювати та видаляти власні оголошення про знахідки.

## 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 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

# 2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами є користувачі системи: незареєстрований (гість) та зареєстрований (користувач).

Зареєстрованому користувачу доступна уся функціональність, що і незареєстрованому, а також можливість створювати та видаляти оголошення про знахідки. Детально усі сценарії використання описані у наступному підрозділі.

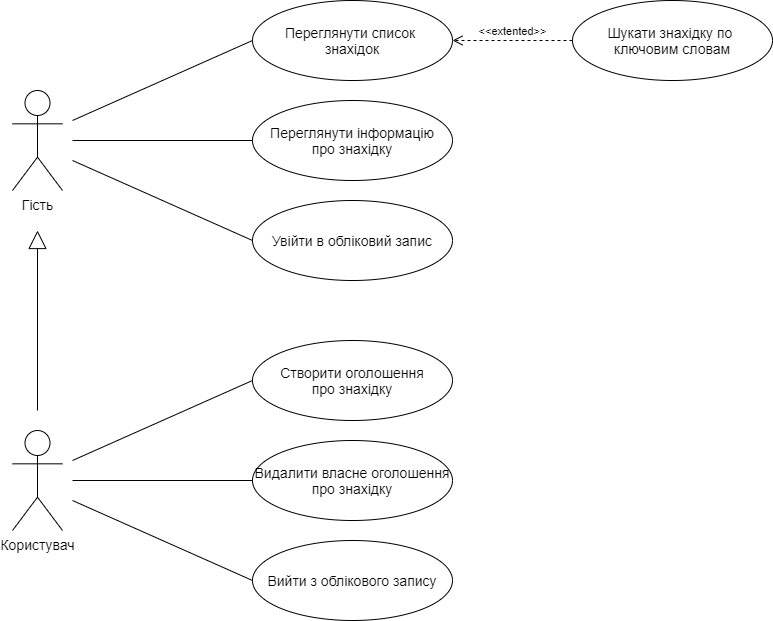


Рисунок 2.1 – Діаграма прецедентів

## 2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.17.

Таблиця 2.1 – Сценарій використання «Пошук оголошення про знахідку по ключовим словам»

|  |  |
| --- | --- |
| Назва | Пошук оголошення про знахідку по ключовим словам |
| ID | 1 |
| Опис | Користувач, використовуючи поле для пошуку, шукає оголошення про знахідку, що збігається з ключовими словами(хоча б одним, оголошення будуть відсортовані за кількістю співпадінь) |
| Актори | Гість, Зареєстрований користувач |
| Вигоди компанії | Якщо користувачі не можуть шукати оголошення про знахідку за відомими ключовими словами, то вони навряд чи продовжать користуватися сервісом |
| Частота користування | Постійно |
| Тригери | Користувач вводить пошуковий запит у полі для пошуку |
| Передумови | Пошукове поле доступне у будь–якому вікні |
| Постумови | Користувач потрапляє на вікно з результатами пошуку |
| Основний розвиток | Користувач вводить запит у пошукову строку, натискає на кнопку пошуку чи Enter |
| Альтернативні розвитки | – |
| Виняткові ситуації | – |

В таблиці 2.2 представлений сценарій використання «Перегляд обраного оголошення»

Таблиця 2.2 – Сценарій використання «Перегляд обраного оглошення»

|  |  |
| --- | --- |
| Назва | Перегляд обраного оглошення |
| ID | 2 |
| Опис | Користувач обирає оголошення, яке хоче переглянути і переглядає |
| Актори | Гість, Зареєстрований користувач |
| Вигоди компанії | Бюро знахідок без можливості перегляду інформації про знахідку не буде користуватися попитом |
| Частота користування | Постійно |
| Тригери | Користувач натискає на назву оголошення у списку оголошень про знахідки |
| Передумови | У вікні є список оглошень про знахідки, та назви, що є гіперпосиланнями. |
| Постумови | Відображення інформації про знахідку |
| Основний розвиток | Користувач натискає на назву оглошення  Оголошення завантажується з серверу  Інформації про знахідку відображається на сторінці |
| Альтернативні розвитки | – |
| Виняткові ситуації | За заданими ключовими словами не знайдено жодного оглошення. Тоді відображається відповідне повідомлення |

В таблиці 2.3 представлений сценарій використання «Перегляд головної сторінки із списком оголошень про знахідки»

Таблиця 2.3 – Сценарій використання «Перегляд головної сторінки із списком оголошень про знахідки»

|  |  |
| --- | --- |
| Назва | Перегляд головної сторінки із списком оголошень про знахідки |
| ID | 3 |
| Опис | Користувач переглядає головну сторінку із списком оголошень про знахідки |
| Актори | Гість, Зареєстрований користувач |
| Вигоди компанії | Головна сторінка важлива для зацікавлення нових користувачів, та актуальна інформація буде утримувати старих |
| Частота користування | Постійно |
| Тригери | Користувач переходить на сайт/запускає додаток |
| Передумови | Немає |
| Постумови | Користувач потрапляє на головну сторінку |
| Основний розвиток | Користувач запускає додаток/переходить на сайт |
| Альтернативні розвитки | – |
| Виняткові ситуації | Сервіс недоступний у країні користувача, тоді показується повідомлення про помилку |

В таблиці 2.4 представлений сценарій використання «Пошук композицій по наданому користувачем аудіо із мікрофону або файлу»

Таблиця 2.4 – Сценарій використання «Вхід в обліковий запис користувача»

|  |  |
| --- | --- |
| Назва | Вхід в обліковий запис користувача |
| ID | 4 |
| Опис | Користувач може увійти у його обліковий запис для додаткових можливостей(створення/видалення оголошень про знахідки) |
| Актори | Зереєстрований користувач |
| Вигоди компанії | Без можливості входу у власний обліковий запис користувачем доведеться реєструвати кожен раз новий, що є абсурдним. |
| Частота користування | Постійно |
| Тригери | Користувач на будь-якій сторінці вводить логін та пароль, натискає на кнопку «Увійти». |
| Передумови | Користувач має обліковий запис, знаходится на будь-якій сторінці. |
| Постумови | У користувача з’являється розділ мої оглошення |
| Основний розвиток | Користувач входить у свій обліковий запис, створює/видаляє власні оголошення. |
| Альтернативні розвитки | – |
| Виняткові ситуації | Користувач вводить неправильний логін та/або пароль. Тоді показується повідомлення про помилку. |

В таблиці 2.5 представлений сценарій використання «Формування персоналізованих плейлистів на основі результатів видачі»

Таблиця 2.5 – Сценарій використання «Реєстрація облікового запису»

|  |  |
| --- | --- |
| Назва | Реєстрація облікового запису |
| ID | 5 |
| Опис | Гість може зареєструвати обліковий запис |
| Актори | Гість |
| Вигоди компанії | Без власного облікового запису користувачі не зможуть видаляти оголошення про знахідки, адже система не знатиме хто ці оголошення створив |
| Частота користування | Часто |
| Тригери | Гість на будь-якій сторінці сайту натискає на посилання «Зареєструватися» |
| Передумови | Гість знаходиться на будь-якій сторінці сайту |
| Постумови | Гість створив обліковий запис та уввійшов в нього |
| Основний розвиток | Гість натискає на посилання «Зареєструватися», потрапляє на сторінку з відповідною формою, заповнює дані для реєстрації, натискає кнопку «Створити обліковий запис» |
| Альтернативні розвитки | – |
| Виняткові ситуації | Гість увів логін, що вже зареєстрований у системі. Гість не ввів усі дані у формі реєстрування. Тоді показуються відповідні повідомлення про помилки |

В таблиці 2.6 представлений сценарій використання «Створення оголошень про знахідки»

Таблиця 2.6 – Сценарій використання «Створення оголошень про знахідки»

|  |  |
| --- | --- |
| Назва | Створення оголошень про знахідки |
| ID | 6 |
| Опис | Зареєстрований користувач натискає кнопку «Додати знахідку», заповнює форму, натискає кнопку «Опублікувати» |
| Актори | Зареєстрований користувач |
| Вигоди компанії | Без можливості додавання оголошень про знахідки користувачами сайт буде нерентабельним |
| Частота користування | Часто |
| Тригери | Користувач натискає на відповідну кнопку |
| Передумови | Користувач увійшов у обліковий запис, знаходиться на будь-якій сторінці |
| Постумови | Оголошення про знахідку створене. |
| Основний розвиток | Користувач натискає на відповідну кнопку біля треку для додавання його до обраного |
| Альтернативні розвитки | – |
| Виняткові ситуації | Користувач не ввів усі дані у формі. Тоді показується повідомлення про помилку |

В таблиці 2.7 представлений сценарій використання «Видалення оголошень про знахідки користувача»

Таблиця 2.7 – Сценарій використання «Видалення оголошень про знахідки користувача»

|  |  |
| --- | --- |
| Назва | Видалення оголошень про знахідки користувача |
| ID | 7 |
| Опис | Зареєстрований користувач біля інформації про знахідку натискає кнопку «Видалити» |
| Актори | Зареєстрований користувач |
| Вигоди компанії | Без можливості видалення оголошень про знахідки, сайт буде засміченим – ним буде неприємно користуватись |
| Частота користування | Часто |
| Тригери | Користувач натискає відповідну кнопку |
| Передумови | Зареєстрований користувач знаходиться на сторінці своєї знахідки або у розділі «Мої оголошення» |
| Постумови | Оголошення видалене з системи |
| Основний розвиток | Зареєстрований користувач видаляє власне оголошення та потрапляє у розділ «Мої оголошення» |
| Альтернативні розвитки | – |
| Виняткові ситуації | – |

# 

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1

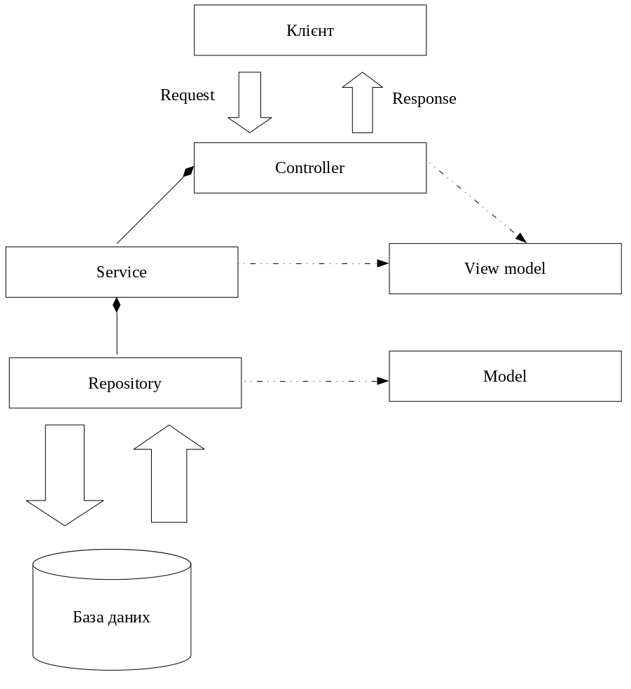


Рисунок 3.1 – Загальна архітектура системи

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 валідуються, 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. Структурна схема зображена в Додатоку Б

До серверної частини належать наступні елементи:

- контролер;

- модель та вигляд;

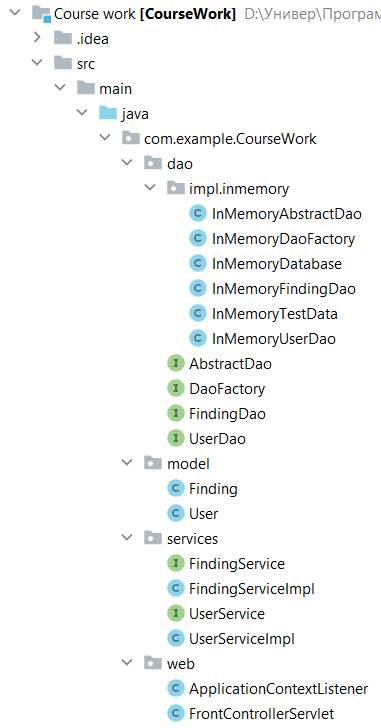
- сервіс;

- репозиторій.

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

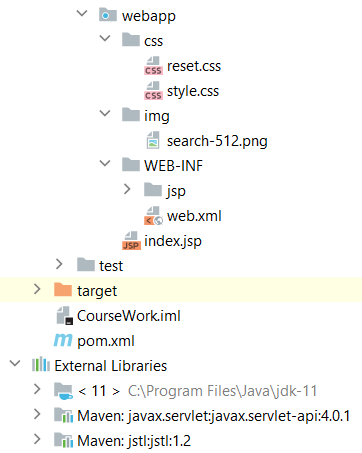


Рисунок 4.1 – Загальна структура проекту

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

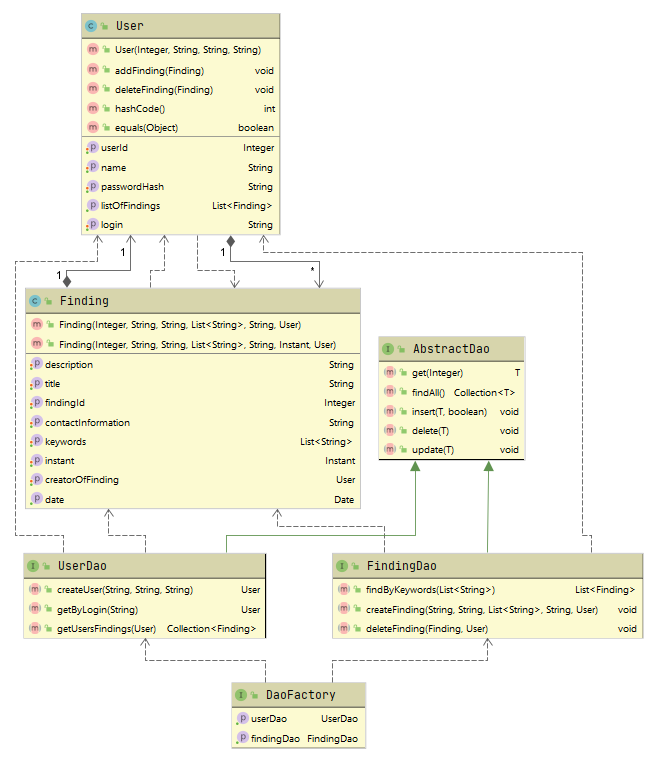


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

Клас “User” – сутність «Користувач», має наступні поля:

* Integer userId;
* String name;
* String login;
* String passwordHash;
* List<Finding> listOfFindings;

Та наступні методи взаємодії:

* addFinding(Finding) – метод, що додає користувачу оголошення;
* deleteFinding(Finding) – метод, що видаляє оголошення користувача;
* hashCode() – метод, що повертає хеш-код об’єкта;
* equals(Object) – метод, що порівнює на рівність з іншим об’єктом.

Клас Finding – сутність «Знахідка», має наступні поля:

* Integer findingId;
* String title;
* String keywords;
* String description;
* String contactInformation;
* User creatorOfFinding;
* Instant instant.

Інтерфейс UserDao має наступні методи взаємодії з даними:

* Метод, що створює новий об’єкт класу User: createUser(String, String, Stirng);
* Метод, що повертає об’єкт класу User за його полем login: getByLogin(String);
* Метод, що повертає список об’єктів класу Finding: getUsersFinding(User).

Інтерфейс FindingDao має наступні методи взаємодіх з даними:

* findByKeywords(List<String>) – здійснює пошук знахідок по ключовим словам та повертає результат у вигляді списка знахідок;
* createFinding(String, String, List<String>, String, User) – створює знахідку;
* deleteFinding(Finding, User) – видаляє знахідку.

## 4.3 Компоненти рівня бізнес-логіки

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.3

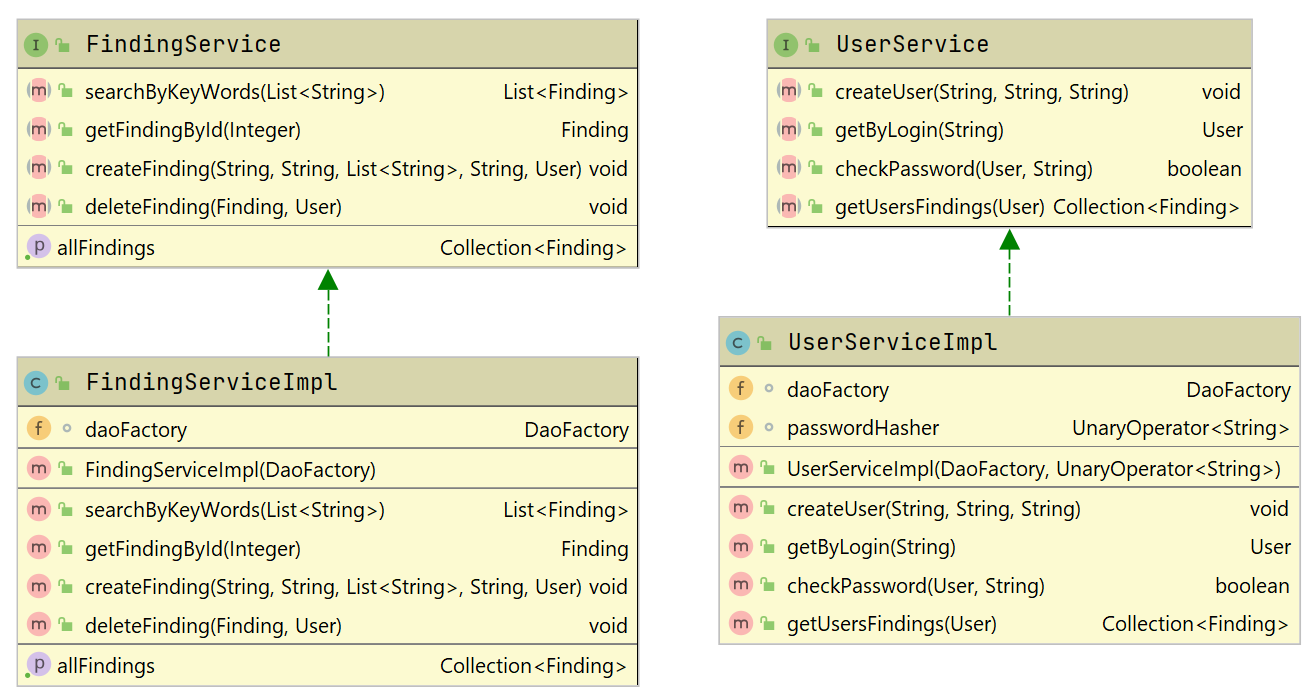


Рисунок 4.3 – Основні сутності та інтерфейси рівня бізнеслогіки

Інтерфейс UserService має наступні методи взаємодії:

* createUser(String, String, String) – метод, що створює об’єкт класу User;
* getByLogin(String) – метод, що повертає користувача із заданим логіном;
* checkPassword(User, String) – метод, що перевіряє пароль користувача;
* getUsersFindings(User) – метод, що повертає список знахідок користувача.

Інтерфейс FindingService має наступні методи взаємодії:

* getFindingById(Integer) – метод, що повертає знахідку за її індифікатором;
* searchByKeywords(List<String) – метод, що шукає знахідки по ключовим словам;
* createFinding(String, String, List<String>, String, User) – метод, що створює знахідку;
* deleteFinding(Finding, User) – метод, що видаляє знахідку.

## 4.4 Компоненти рівня інтерфейсу користувача

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.4

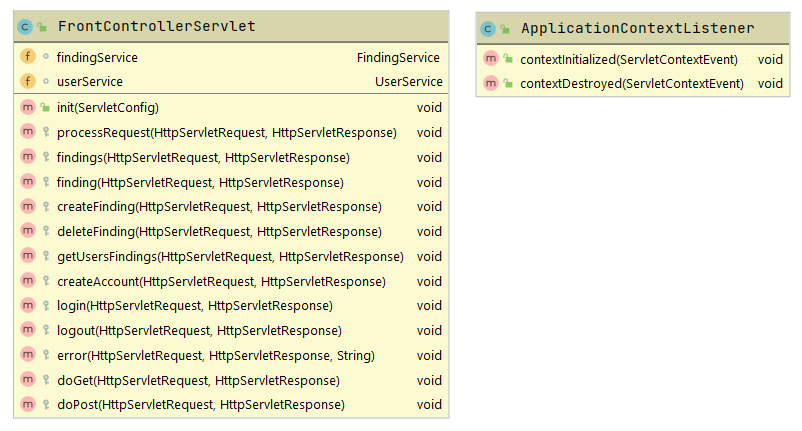


Рисунок 4.3 – Основні сутності та інтерфейси рівня бізнеслогіки

Клас-сервлет ApplicationContextListener має наступні методи взаємодії:

* contextInitialized(ServletContextEvent) – ініціалізує контекст сервлету;
* contextDestroyed(ServletContextEvent) – видаляє контекст сервлету.

Клас-сервлет FrontControllerServlet має наступні поля:

* FindingService findingService;
* UserService userService;

Та наступні методи взаємодії:

* doGet(HttpServletRequest, HttpServletResponse) – метод, що відповідає на запит “get”;
* doPost(HttpServletRequest, HttpServletResponse) – метод, що відповідає на запит “post”;
* processRequest(HttpServletRequest, HttpServletResponse) – метод, що відповідає на усі запити;
* findings(HttpServletRequest, HttpServletResponse) – метод, що витягує з БД оголошення(усі, або ті що відповідають пошуковим ключовим словам) та перенаправляє на відповідну jsp;
* finding(HttpServletRequest, HttpServletResponse) – метод, що витягує з БД певне оголошення та перенаправляє на відповідну jsp;
* createFinding(HttpServletRequest, HttpServletResponse) – метод, що створює оголошення та перенаправляє на відповідну jsp;
* deleteFinding(HttpServletRequest, HttpServletResponse) – метод, що видаляє оголошення та перенаправляє на відповідну jsp;
* getUsersFindings(HttpServletRequest, HttpServletResponse) – метод, що витягує з БД оголошення користувача та перенаправляє на відповідну jsp;
* createAccount(HttpServletRequest, HttpServletResponse) – метод, що обліковий запис користувача та перенаправляє на відповідну jsp;
* login(HttpServletRequest, HttpServletResponse) – метод, що здійснює перевірку на правильність логіну та паролю, і перенаправляє на відповідну jsp;
* logout(HttpServletRequest, HttpServletResponse) – метод, що видаляє поточну сесію та перенаправляє на відповідну jsp;
* error(HttpServletRequest, HttpServletResponse) – метод, що перенаправляє на сторінку з відповідною помилкою.

ВИСНОВКИ

Під час написання роботи, були проаналізовані системі бюро знахідок в Україні, переваги та недоліки існуючих систем. Було вирішено розробити систему, яка дозволить користувачам легко створювати оголошення та шукати їх на сайті.

Першим кроком було сформулювати функціональні та нефункціональні вимоги до системи, що визначило очікувану поведінку системи. Наступним, обрано технології на яких буде написана система. Java була обрана в якості мови програмування, середа розробки – IntelliJ IDEA за її можливості та потужність. Були використані бібліотеки JSTL 1.2 та javax.servlet-api:4.0.1 та локальний сервер Apache Tomcat 9.0.43.

Наступним кроком були описані сценарії використання. Система підтримує функціонал перегляду оголошень про знахідки, детальну інформацію знахідки, перегляду власних оголошень про знахіки. Також було впроваджено функціонал облікових записів – користувач може увійти та вийти з облікового запису. Облікові записи додають функціонал створення та видалення оголошень про знахідки. Для зручного пошуку знахідок система підтримує пошук по ключовим словам.

Далі, було розроблено загальну архітектуру системи. Завдяки шаблону проектування MVC, систему було розділено на логічні рівні. Також, окремо було виділені конвертери, помічники та моделі, які роблять систему більш гнучкою та зрозумілою. Для оброблення http-запитів було реалізовано патер “FrontController”. Даний шаблон проектування дозволяє уникнути дублювання поведінки контролерів, та робить легшим подальшу підтримку застосунку. Задачею фронт-контролера є надання єдиної точки входу для обробки усіх запитів та виклик відповідної поведінки в залежності від запиту.

Отже, сумуючи наведене вище, можна сказати, що основі переваги створеної системи в простоті її налаштування та оновлення функціоналу. Завдяки тому, що система має відкриту архітектуру, її розширення не є проблемою. Також графічний інтерфейс системи є легким в налаштуванні. Система є конкурентоспроможною та має потенціал до розвинення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Приклад – MovieDataBase” [Електронний ресурс] – Режим доступу до ресурсу: https://drive.google.com/file/d/1Pa1y9IaFn-YIjCXII31HUrq9Gzqwlwqg/view?usp=sharing
2. “Шаблон оформлення - Курсова робота з програмування.docx” - [Електронний ресурс] – Режим доступу до ресурсу: <https://drive.google.com/file/d/1QUPUcV9ir4M0P9PlLYRYMLG9A8EHbEqN/view?usp=sharing>
3. “Front Controller” - [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Front_controller>
4. “Model–view–controller” - [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>
5. “Data access object” - [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Data_access_object>

ДОДАТОК А

Рівень доступу до даних та моделі – код

package com.example.CourseWork.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class User {  
  
 private Integer userId;  
 private String name;  
 private String login;  
 private String passwordHash;  
 private final List<Finding> listOfFindings;  
  
 public User(Integer userId, String name, String login, String passwordHash) {  
 this.userId = userId;  
 this.name = name;  
 this.login = login;  
 this.passwordHash = passwordHash;  
 listOfFindings = new ArrayList<>();  
 }  
  
 public Integer getUserId() {  
 return userId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getLogin() {  
 return login;  
 }  
  
 public String getPasswordHash() {  
 return passwordHash;  
 }  
  
 public List<Finding> getListOfFindings() {  
 return listOfFindings;  
 }  
  
 public void setUserId(Integer userId) {  
 this.userId = userId;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setLogin(String login) {  
 this.login = login;  
 }  
  
 public void setPasswordHash(String passwordHash) {  
 this.passwordHash = passwordHash;  
 }  
  
 public void addFinding(Finding finding) {  
 listOfFindings.add(finding);  
  
 }  
  
 public void deleteFinding(Finding finding) {  
 listOfFindings.remove(finding);  
 }  
  
 @Override  
 public int hashCode() {  
 int hash = 5;  
 hash = 37 \* hash + Objects.*hashCode*(this.userId);  
 return hash;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) {  
 return true;  
 }  
 if (obj == null) {  
 return false;  
 }  
 if (getClass() != obj.getClass()) {  
 return false;  
 }  
 final User other = (User) obj;  
 return Objects.*equals*(this.userId, other.userId);  
 }  
  
}

package com.example.CourseWork.model;  
  
import java.time.Instant;  
import java.util.Date;  
import java.util.List;  
  
public class Finding {  
  
 private Integer findingId;  
 private String title;  
 private String description;  
 private List<String> keywords;  
 private String contactInformation;  
 private Instant instant;  
 private User creatorOfFinding;  
  
 public Finding(Integer findingId, String title, String description, List<String> keywords, String contactInformation, User creatorOfFinding) {  
 this(findingId, title, description, keywords, contactInformation, Instant.*now*(), creatorOfFinding);  
 }  
  
 public Finding(Integer findingId, String title, String description, List<String> keywords,  
 String contactInformation, Instant instant, User creatorOfFinding) {  
 this.findingId = findingId;  
 this.title = title;  
 this.description = description;  
 this.keywords = keywords;  
 this.contactInformation = contactInformation;  
 this.instant = instant;  
 this.creatorOfFinding = creatorOfFinding;  
 }  
  
 public Integer getFindingId() {  
 return findingId;  
 }  
  
 public void setFindingId(Integer findingId) {  
 this.findingId = findingId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public List<String> getKeywords() {  
 return keywords;  
 }  
  
 public void setKeywords(List<String> keywords) {  
 this.keywords = keywords;  
 }  
  
 public String getContactInformation() {  
 return contactInformation;  
 }  
  
 public void setContactInformation(String contactInformation) {  
 this.contactInformation = contactInformation;  
 }  
  
 public Instant getInstant() {  
 return instant;  
 }  
  
 public void setInstant(Instant instant) {  
 this.instant = instant;  
 }  
  
 public User getCreatorOfFinding() {  
 return creatorOfFinding;  
 }  
  
 public void setCreatorOfFinding(User creatorOfFinding) {  
 this.creatorOfFinding = creatorOfFinding;  
 }  
  
 public Date getDate() {  
 return Date.*from*(instant);  
 }  
}

package com.example.CourseWork.dao;  
  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Collection;  
  
public interface UserDao extends AbstractDao<User> {  
  
 User createUser(String name, String login, String password);  
  
 User getByLogin(String login);  
  
 Collection<Finding> getUsersFindings(User user);  
  
}

package com.example.CourseWork.dao;  
  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.List;  
  
public interface FindingDao extends AbstractDao<Finding> {  
  
 List<Finding> findByKeywords(List<String> keywords);  
  
 void createFinding(String title, String description, List<String> keywords, String contactInformation, User user);  
  
 void deleteFinding(Finding finding, User user);  
  
}

package com.example.CourseWork.dao;  
  
public interface DaoFactory {  
  
 FindingDao getFindingDao();  
  
 UserDao getUserDao();  
  
}

package com.example.CourseWork.dao;  
  
import java.util.Collection;  
  
public interface AbstractDao<T> {  
  
 T get(Integer id);  
  
 Collection<T> findAll();  
  
 void insert(T entity, boolean generateId);  
  
 void delete(T entity);  
  
 void update(T entity);  
}

package com.example.CourseWork.dao.impl.inmemory;  
  
import com.example.CourseWork.dao.AbstractDao;  
  
import java.util.Collection;  
import java.util.Map;  
import java.util.function.BiConsumer;  
import java.util.function.Function;  
  
class InMemoryAbstractDao<T> implements AbstractDao<T> {  
  
 protected Map<Integer, T> entities;  
 protected Function<T, Integer> idGetter;  
 protected BiConsumer<T, Integer> idSetter;  
 protected InMemoryDatabase database;  
  
 public InMemoryAbstractDao(Map<Integer, T> entities, Function<T, Integer> idGetter,  
 BiConsumer<T, Integer> idSetter, InMemoryDatabase database) {  
 this.entities = entities;  
 this.idGetter = idGetter;  
 this.idSetter = idSetter;  
 this.database = database;  
 }  
  
 @Override  
 public T get(Integer id) {  
 return entities.get(id);  
 }  
  
 @Override  
 public Collection<T> findAll() {  
 return entities.values();  
 }  
  
 @Override  
 public void insert(T entity, boolean generateId) {  
 if (generateId) {  
 int maxId = entities.keySet().stream()  
 .mapToInt(Integer::intValue)  
 .max()  
 .orElse(0);  
 idSetter.accept(entity, maxId + 1);  
 }  
 entities.put(idGetter.apply(entity), entity);  
 }  
  
 @Override  
 public void delete(T entity) {  
 entities.remove(idGetter.apply(entity));  
 }  
  
 @Override  
 public void update(T entity) {  
 entities.put(idGetter.apply(entity), entity);  
 }  
}

package com.example.CourseWork.dao.impl.inmemory;  
  
import com.example.CourseWork.dao.DaoFactory;  
import com.example.CourseWork.dao.FindingDao;  
import com.example.CourseWork.dao.UserDao;  
  
  
public class InMemoryDaoFactory implements DaoFactory {  
  
 InMemoryDatabase database;  
  
 FindingDao findingDao;  
 UserDao userDao;  
  
 InMemoryDaoFactory(InMemoryDatabase database) {  
 this.database = database;  
  
 findingDao = new InMemoryFindingDao(database);;  
 userDao = new InMemoryUserDao(database);  
 }  
  
 public FindingDao getFindingDao() {  
 return findingDao;  
 }  
  
 @Override  
 public UserDao getUserDao() {  
 return userDao;  
 }  
}

package com.example.CourseWork.dao.impl.inmemory;  
  
import com.example.CourseWork.dao.DaoFactory;  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Map;  
import java.util.TreeMap;  
  
public class InMemoryDatabase {  
 Map<Integer, Finding> findings;  
 Map<Integer, User> users;  
  
 public InMemoryDatabase() {  
 findings = new TreeMap<>();  
 users = new TreeMap<>();  
 }  
  
 public DaoFactory getDaoFactory() {  
 return new InMemoryDaoFactory(this);  
 }  
}

package com.example.CourseWork.dao.impl.inmemory;  
  
import com.example.CourseWork.dao.FindingDao;  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.List;  
import java.util.Locale;  
import java.util.Map;  
import java.util.stream.Collectors;  
  
public class InMemoryFindingDao extends InMemoryAbstractDao<Finding> implements FindingDao {  
  
 public InMemoryFindingDao(InMemoryDatabase database) {  
 super(database.findings, Finding::getFindingId, Finding::setFindingId, database);  
 }  
  
 @Override  
 public List<Finding> findByKeywords(List<String> keywords) {  
 return database.findings.values().stream()  
 .collect(Collectors.*toMap*(f -> f, f -> checkKeywordsOfFinding(f, keywords)))  
 .entrySet().stream()  
 .sorted(Map.Entry.comparingByValue())  
 .filter(e -> e.getValue() != 0)  
 .map(Map.Entry::getKey)  
 .collect(Collectors.toList());  
 }  
  
 @Override  
 public void createFinding(String title, String description, List<String> keywords, String contactInformation, User user) {  
 Finding finding = new Finding(-1, title, description, keywords, contactInformation, user);  
 insert(finding, true);  
 user.addFinding(finding);  
 }  
  
 @Override  
 public void deleteFinding(Finding finding, User user) {  
 user.deleteFinding(finding);  
 delete(finding);  
 }  
  
 private Integer checkKeywordsOfFinding(Finding finding, List<String> keywordsOfSearching) {  
 int numberOfCoincidences = 0;  
 for (String keywordOfFinding : finding.getKeywords()) {  
 for (String keywordOfSearching : keywordsOfSearching) {  
 if (keywordOfFinding.equalsIgnoreCase(keywordOfSearching))  
 numberOfCoincidences++;  
 }  
 }  
 return -numberOfCoincidences;  
 }  
  
}

package com.example.CourseWork.dao.impl.inmemory;  
  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Arrays;  
  
public class InMemoryTestData {  
  
 public static void generateTo(InMemoryDatabase database) {  
 database.findings.clear();  
 database.users.clear();  
  
 User nastya = new User(1, "Настя", "nastya@example.com", "passwordhash");  
 User borys = new User(2, "Борис", "boryska777@example.com", "passwordhash");  
 User andriy = new User(3, "Андрій", "andriy\_chered@example.com", "passwordhash");  
 User diana = new User(4, "Діана", "dianka120401@example.com", "passwordhash");  
 User max = new User(5, "Максим", "max@example.com", "passwordhash");  
 Arrays.asList(nastya, borys, andriy, diana, max).forEach(user -> database.users.put(user.getUserId(), user));  
  
 Finding backpack = new Finding(1, "Рюкзак Nike сірий", "Знайшла на вул.Парниковій 22, район Крошня. Всередині спортивна форма, пляшка з водою.",  
 Arrays.asList("рюкзак", "сірий", "крошня", "nike"), "0971234567", nastya);  
 Finding samsung = new Finding(2, "Samsung galaxy A10 чорний", "Був знайдений 28.05 біля магазину \"АТБ\" на Богунії. На екрані поклеєне розби",  
 Arrays.asList("Samsung", "самсунг", "Богунія", "чорний"), "borys@example.com", borys);  
 Finding airpods = new Finding(3, "Apple AirPods 2020", "Знайдені бездротові навушники airpods у червоному чохлі. Хтось залишив на лавці на Малікова",  
 Arrays.asList("airpods", "apple", "tws", "навушники", "малікова", "червоний"), "0987654321", nastya);  
 Finding bag = new Finding(4, "Сумка \"Guess\" рожева з ланцюжком", "Сумка залишена на задньому сидінні автобуса маршруту 110. Всередині гаманець, косметичка, дзеркальце та інші речі.",  
 Arrays.asList("сумка", "guess", "рожевий", "ланцюжок", "автобус"), "0738473294", andriy);  
 Finding passport = new Finding(5, "Паспорт Іванова Івана Івановича", "Знайшов біля Богунського районного суда на землі. Віддам при підтвердженні особистості іншими документами.",  
 Arrays.asList("паспорт", "документи", "центр"), "diana@example.com", diana);  
 Finding bankCard = new Finding(6, "Банківська картка monobank", "Хтось забув вийняти з банкомату на вул.Грушевського. Віддам власнику який зможе назвати останні чотири цифри 4441 1144 4123 \*\*\*\* та CVC код",  
 Arrays.asList("monobank", "банк", "картка", "гроші", "центр"), "max@example.com", max);  
 Finding autoKey = new Finding(7, "Ключі від автомобіля Ford Fiesta", "Знайшов в гідропарку біля кіоску с морозивом",  
 Arrays.asList("ключі", "авто", "ford", "гідропарк", "корбутівка"), "0682736501", borys);  
 Arrays.asList(backpack, samsung, airpods, bag, passport, bankCard, autoKey).  
 forEach(finding -> database.findings.put(finding.getFindingId(), finding));  
  
 nastya.getListOfFindings().addAll(Arrays.asList(backpack, airpods));  
 borys.getListOfFindings().addAll(Arrays.asList(samsung, autoKey));  
 andriy.getListOfFindings().addAll(Arrays.asList(bag));  
 diana.getListOfFindings().addAll(Arrays.asList(passport));  
 max.getListOfFindings().addAll(Arrays.asList(bankCard));  
 }  
}

package com.example.CourseWork.dao.impl.inmemory;  
  
import com.example.CourseWork.dao.UserDao;  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Collection;  
  
class InMemoryUserDao extends InMemoryAbstractDao<User> implements UserDao {  
  
 InMemoryUserDao(InMemoryDatabase database) {  
 super(database.users, User::getUserId, User::setUserId, database);  
 }  
  
  
 @Override  
 public User createUser(String name, String login, String password) {  
 User user = new User(-1 , name, login, password);  
 this.insert(user, true);  
 return null;  
 }  
  
 @Override  
 public User getByLogin(String login) {  
 return database.users.values()  
 .stream()  
 .filter(user -> user.getLogin().equals(login))  
 .findFirst()  
 .orElse(null);  
 }  
  
 @Override  
 public Collection<Finding> getUsersFindings(User user) {  
 return user.getListOfFindings();  
 }  
  
}

ДОДАТОК Б

Рівень бізнес-логіки – код

package com.example.CourseWork.services;  
  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Collection;  
  
public interface UserService {  
  
 void createUser(String name, String login, String password);  
  
 User getByLogin(String login);  
  
 boolean checkPassword(User user, String password);  
  
 Collection<Finding> getUsersFindings(User user);  
  
}

package com.example.CourseWork.services;  
  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Collection;  
import java.util.List;  
  
public interface FindingService {  
  
 Collection<Finding> getAllFindings();  
  
 List<Finding> searchByKeyWords(List<String> keywords);  
  
 Finding getFindingById(Integer findingId);  
  
 void createFinding(String title, String description, List<String> keywords, String contactInformation, User user);  
  
 void deleteFinding(Finding finding, User user);  
  
}

package com.example.CourseWork.services;  
  
import com.example.CourseWork.dao.DaoFactory;  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.List;  
  
public class FindingServiceImpl implements FindingService {  
 DaoFactory daoFactory;  
  
 public FindingServiceImpl(DaoFactory daoFactory) {  
 this.daoFactory = daoFactory;  
 }  
  
 @Override  
 public Collection<Finding> getAllFindings() {  
 return daoFactory.getFindingDao().findAll();  
 }  
  
 @Override  
 public List<Finding> searchByKeyWords(List<String> keywords) {  
 if (keywords == null || keywords.size() == 0) {  
 return new ArrayList<>(getAllFindings());  
 }  
 return daoFactory.getFindingDao().findByKeywords(keywords);  
 }  
  
 @Override  
 public Finding getFindingById(Integer findingId) {  
 return daoFactory.getFindingDao().get(findingId);  
 }  
  
 @Override  
 public void createFinding(String title, String description, List<String> keywords, String contactInformation, User user) {  
 daoFactory.getFindingDao().createFinding(title, description, keywords, contactInformation, user);  
 }  
  
 @Override  
 public void deleteFinding(Finding finding, User user) {  
 daoFactory.getFindingDao().deleteFinding(finding, user);  
 }  
}

package com.example.CourseWork.services;  
  
import com.example.CourseWork.dao.DaoFactory;  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
  
import java.util.Collection;  
import java.util.function.UnaryOperator;  
  
public class UserServiceImpl implements UserService {  
 DaoFactory daoFactory;  
 UnaryOperator<String> passwordHasher;  
  
 public UserServiceImpl(DaoFactory daoFactory, UnaryOperator<String> passwordHasher) {  
 this.daoFactory = daoFactory;  
 this.passwordHasher = passwordHasher;  
 }  
  
 @Override  
 public void createUser(String name, String login, String password) {  
 daoFactory.getUserDao().createUser(name, login, password);  
 }  
  
 @Override  
 public User getByLogin(String login) {  
 return daoFactory.getUserDao().getByLogin(login);  
 }  
  
 @Override  
 public boolean checkPassword(User user, String password) {  
 return user.getPasswordHash().equals(passwordHasher.apply(password));  
 }  
  
 @Override  
 public Collection<Finding> getUsersFindings(User user) {  
 return daoFactory.getUserDao().getUsersFindings(user);  
 }  
}

ДОДАТОК В

Рівень користувацького інтерфейсу – код

package com.example.CourseWork.web;  
  
import com.example.CourseWork.dao.DaoFactory;  
import com.example.CourseWork.dao.impl.inmemory.InMemoryDatabase;  
import com.example.CourseWork.dao.impl.inmemory.InMemoryTestData;  
import com.example.CourseWork.services.FindingService;  
import com.example.CourseWork.services.FindingServiceImpl;  
import com.example.CourseWork.services.UserService;  
import com.example.CourseWork.services.UserServiceImpl;  
  
import javax.servlet.ServletContextEvent;  
import javax.servlet.ServletContextListener;  
import javax.servlet.annotation.WebListener;  
import java.util.function.UnaryOperator;  
  
@WebListener  
public class ApplicationContextListener implements ServletContextListener {  
  
 @Override  
 public void contextInitialized(ServletContextEvent sce) {  
  
 *// Must be changed to real database in real project* InMemoryDatabase database = new InMemoryDatabase();  
 InMemoryTestData.*generateTo*(database);  
  
 DaoFactory daoFactory = database.getDaoFactory();  
  
 FindingService movieService = new FindingServiceImpl(daoFactory);  
 sce.getServletContext().setAttribute("movieService", movieService);  
  
 UserService userService = new UserServiceImpl(daoFactory, UnaryOperator.*identity*());  
 sce.getServletContext().setAttribute("userService", userService);  
 }  
  
 @Override  
 public void contextDestroyed(ServletContextEvent sce) {  
  
 }  
}

package com.example.CourseWork.web;  
  
import com.example.CourseWork.model.Finding;  
import com.example.CourseWork.model.User;  
import com.example.CourseWork.services.FindingService;  
import com.example.CourseWork.services.UserService;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
import java.io.IOException;  
import java.nio.charset.StandardCharsets;  
import java.util.Arrays;  
import java.util.Collection;  
import java.util.List;  
  
@WebServlet(name = "FrontControllerServlet", urlPatterns = {"/do/\*"})  
public class FrontControllerServlet extends HttpServlet {  
  
 FindingService findingService;  
 UserService userService;  
  
 @Override  
 public void init(ServletConfig config) throws ServletException {  
 findingService = (FindingService) config.getServletContext().getAttribute("movieService");  
 userService = (UserService) config.getServletContext().getAttribute("userService");  
 }  
  
 protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
  
 String pathInfo = request.getPathInfo();  
 if (pathInfo == null) {  
 pathInfo = "/";  
 }  
  
 try {  
 switch (pathInfo) {  
 case "/login":  
 login(request, response);  
 break;  
 case "/logout":  
 logout(request, response);  
 break;  
 case "/finding":  
 finding(request, response);  
 break;  
 case "/addFinding":  
 request.getRequestDispatcher("/WEB-INF/jsp/addFinding.jsp").forward(request, response);  
 break;  
 case "/createFinding":  
 createFinding(request, response);  
 break;  
 case "/deleteFinding":  
 deleteFinding(request, response);  
 break;  
 case "/createAccount":  
 createAccount(request, response);  
 break;  
 case "/usersFindings":  
 getUsersFindings(request, response);  
 break;  
 case "/register":  
 request.getRequestDispatcher("/WEB-INF/jsp/register.jsp").forward(request, response);  
 break;  
 case "/":  
 case "/search":  
 default:  
 findings(request, response);  
 break;  
 }  
 } catch (RuntimeException ex) {  
 ex.printStackTrace();  
 error(request, response, "Oops, " + ex.getMessage());  
 }  
  
 }  
  
 protected void findings(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String keywordsString = request.getParameter("keywords");  
 List<String> keywords = null;  
 boolean search = false;  
 if (keywordsString != null && keywordsString.length() != 0) {  
 keywords = Arrays.asList(keywordsString.split(" "));  
 request.setAttribute("keywords", keywords);  
 search = true;  
  
 }  
 request.setAttribute("search", search);  
 List<Finding> findings = findingService.searchByKeyWords(keywords);  
 request.setAttribute("findings", findings);  
 request.getRequestDispatcher("/WEB-INF/jsp/findings.jsp").forward(request, response);  
 }  
  
 protected void finding(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 User user = (User) request.getSession().getAttribute("user");  
 Integer findingId = Integer.parseInt(request.getParameter("findingId"));  
 Finding finding = findingService.getFindingById(findingId);  
 boolean viewByOwner = user != null && user.getListOfFindings().contains(finding);  
  
 request.setAttribute("viewByOwner", viewByOwner);  
 request.setAttribute("finding", finding);  
 request.getRequestDispatcher("/WEB-INF/jsp/finding.jsp").forward(request, response);  
 }  
  
 protected void createFinding(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 User user = (User) request.getSession().getAttribute("user");  
 String title = new String(request.getParameter("title").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
 String keywordsParam = new String(request.getParameter("keywords").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
 List<String> keywords = Arrays.asList(keywordsParam.split(" "));  
 String description = new String(request.getParameter("description").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
 String contactInformation = new String(request.getParameter("contactInformation").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
 if (title.length() == 0 || keywords.size() == 0 || description.length() == 0 || contactInformation.length() == 0) {  
 error(request, response, "Будь ласка, заповніть усі поля!");  
 }  
 findingService.createFinding(title, description, keywords, contactInformation, user);  
 response.sendRedirect("./usersFindings");  
 }  
  
 protected void deleteFinding(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 User user = (User) request.getSession().getAttribute("user");  
 Integer findingId = Integer.parseInt(request.getParameter("findingId"));  
 findingService.deleteFinding(findingService.getFindingById(findingId), user);  
 response.sendRedirect("./usersFindings");  
 }  
  
 protected void getUsersFindings(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 User user = (User) request.getSession().getAttribute("user");  
  
 Collection<Finding> findings = userService.getUsersFindings(user);  
 request.setAttribute("usersFindings", findings);  
 request.getRequestDispatcher("/WEB-INF/jsp/usersFindings.jsp").forward(request, response);  
 }  
  
 protected void createAccount(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String name = new String(request.getParameter("name").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
 String login = new String(request.getParameter("login").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
 String password = new String(request.getParameter("password").getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8);  
  
 if (name.length() == 0 || login.length() == 0 || password.length() == 0) {  
 error(request, response, "Будь ласка, заповніть усі поля!");  
 }  
 if (userService.getByLogin(login) != null) {  
 error(request, response, "Користувач з таким логіном вже існує.");  
 return;  
 }  
 userService.createUser(name, login, password);  
 request.getSession().setAttribute("user", userService.getByLogin(login));  
 response.sendRedirect(".");  
 }  
  
 protected void login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 request.getSession().invalidate();  
  
 String login = request.getParameter("login");  
 User user = userService.getByLogin(login);  
 if (user == null) {  
 error(request, response, "Вибачте, користувача з логіном '" + login + "' не існує.");  
 return;  
 }  
 String password = request.getParameter("password");  
  
 if (!userService.checkPassword(user, password)) {  
 error(request, response, "Невірний пароль!");  
 return;  
 }  
  
 request.getSession().setAttribute("user", user);  
 response.sendRedirect(".");  
 }  
  
 protected void logout(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 request.setAttribute("movies", findingService.getAllFindings());  
 request.getSession().invalidate();  
 response.sendRedirect(".");  
 }  
  
  
 protected void error(HttpServletRequest request, HttpServletResponse response, String message) throws ServletException, IOException {  
 request.setAttribute("message", message);  
 request.getRequestDispatcher("/WEB-INF/jsp/error.jsp").forward(request, response);  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 processRequest(req, resp);  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 processRequest(req, resp);  
 }  
}

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Бюро знахідок | Житомир</title>  
 <link rel="stylesheet" type="text/css" href="../css/reset.css">  
 <link rel="stylesheet" type="text/css" href="../css/style.css">  
 <meta charset="UTF-8">  
</head>  
<body>  
<%@include file="header.jspf" %>  
<c:if test="${!empty user}">  
 <div class="notification"><p><c:out value="${user.name}"/>, опишіть свою знахідку як найкраще, додайте ключові слова  
 (район знаходження, бренд/марка/ колір речі і т.п.). Це дозволить власнику швидше відшукати свою втрату.</p><br>  
 </div>  
 <form id="creatingFinding" class="creatingForm" action="createFinding" method="post">  
 Назва(заголовок оголошення): <input type="text" name="title"><br>  
 Ключові слова: <input type="text" name="keywords"><br>  
 Опис знахідки:<br><textarea name="description" rows="10" cols="65" form="creatingFinding"></textarea><br>  
 Контактна інформація: <input type="text" name="contactInformation"><br>  
 <input class="button" type="submit" value="Опублікувати"><br>  
 </form>  
</c:if>  
<c:if test="${empty user}">  
 <div class="notification"><p>Щоб створити оголошення про знахідку, потрібно увійти у свій обліковий запис!</p></div>  
</c:if>  
<a href="." class="toMainPage">На головну</a>  
<%@include file="footer.jspf" %>  
</body>  
</html>

<%@page contentType="text/html" pageEncoding="UTF-8" %>  
<!DOCTYPE html>  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
 <title>Бюро знахідок | Житомир</title>  
 <link rel="stylesheet" type="text/css" href="../css/reset.css">  
 <link rel="stylesheet" type="text/css" href="../css/style.css">  
</head>  
<body>  
<%@include file="header.jspf" %>  
<section>  
 <div class="notification"><p>${message}</p></div>  
 <a href="." class="toMainPage">На головну</a>  
</section>  
<br>  
<%@include file="footer.jspf" %>  
</body>  
</html>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Бюро знахідок | Житомир</title>  
 <link rel="stylesheet" type="text/css" href="../css/reset.css">  
 <link rel="stylesheet" type="text/css" href="../css/style.css">  
</head>  
<body>  
<%@include file="header.jspf" %>  
<section>  
 <table class="findings-table">  
 <tr>  
 <th>Знахідка</th>  
 <td><c:out value="${finding.title}"/></td>  
 </tr>  
 <tr>  
 <th>Ключові слова:</th>  
 <td>  
 <c:forEach items="${finding.keywords}" var="keyword">  
 <c:out value="${keyword}"/>  
 </c:forEach>  
 </td>  
 </tr>  
 <tr>  
 <th>Опис знахідки:</th>  
 <td><c:out value="${finding.description}"/></td>  
 </tr>  
 <tr>  
 <th>Контактна інформація</th>  
 <td><c:out value="${finding.contactInformation}"/></td>  
 </tr>  
 <c:if test="${viewByOwner}">  
 <tfoot>  
 <tr>  
 <td colspan="2">  
 <form action="deleteFinding" method="POST">  
 <input type="hidden" name="findingId" value="${finding.findingId}" />  
 <input class="deleteButton" type="submit" value="Видалити" />  
 </form>  
 </td>  
 </tr>  
 </tfoot>  
 </c:if>  
 </table>  
</section>  
<a href="." class="toMainPage">На головну</a>  
<%@include file="footer.jspf" %>  
</body>  
</html>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Бюро знахідок | Житомир</title>  
 <link rel="stylesheet" type="text/css" href="../css/reset.css">  
 <link rel="stylesheet" type="text/css" href="../css/style.css">  
</head>  
<body>  
<%@include file="header.jspf" %>  
<section>  
 <c:if test="${keywords != null}">  
 <div class="notification"><p> Результати пошуку по ключаовим словам:  
 <c:forEach items="${keywords}" var="keyword">  
 <c:out value="${keyword}"/>  
 </c:forEach></p></div>  
 </c:if>  
 <c:if test="${findings.size() != 0}">  
 <table class="findings-table">  
 <thead>  
 <tr>  
 <th>Знахідка</th>  
 <th>  
 Ключові слова  
 </th>  
 <th>  
 Час публікації  
 </th>  
 </tr>  
 </thead>  
 <tbody>  
 <c:forEach var="finding" items="${findings}">  
 <tr>  
 <td title="${finding.description}">  
 <a href="finding?findingId=${finding.findingId}">${finding.title}</a>  
 </td>  
 <td>  
 <c:forEach items="${finding.keywords}" var="keyword">  
 <c:out value="${keyword}"/>  
 </c:forEach>  
 </td>  
 <td><fmt:formatDate pattern="dd.MM.yyyy HH:mm" value="${finding.date}"/></td>  
 </tr>  
 </c:forEach>  
 </tbody>  
 </table>  
 </c:if>  
 <c:if test="${findings.size() == 0 && keywords == null}">  
 <div class="notification"><p> На сайті ще немає оголошень про знахідки. Будьте першим, хто опублікує  
 оголошення.</p></div>  
 </c:if>  
 <c:if test="${search}">  
 <a href="." class="toMainPage">Усі оголошення</a>  
 </c:if>  
</section>  
<br>  
<%@include file="footer.jspf" %>  
</body>  
</html>

<%@ page pageEncoding="UTF-8" %>  
<footer class="title">  
 (c) Copyright 2021  
</footer>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ page pageEncoding="UTF-8" %>  
  
<header class="title">  
 <h1>Бюро знахідок | Житомир</h1>  
 <form class="addFinding" action="addFinding">  
 <button class="button">Додати знахідку</button>  
 </form>  
 <form class="search" action="search">  
 <label for="search">Пошук знахідок за ключовими словами</label><br>  
 <input id="search" type="text" name="keywords" placeholder=" рюкзак nike сірий крошня" />  
 <input class="button" type="submit" value="Знайти" formaction="search"/>  
 </form>  
 <c:if test="${empty user}">  
 <form class="log-in" action="login" method="post">  
 Логін:<input type="text" name="login"/><br>  
 Пароль:<input type="password" name="password"/><br>  
 <input class="button" type="submit" value="Увійти" />  
 або <a href="register">Зареєструватися</a>  
 </form>  
 </c:if>  
 <c:if test="${!empty user}">  
 <form class="log-out" method="post">  
 <label>Вітаємо, ${user.name}</label><br>  
 <label><a href="usersFindings">Мої оголошення</a></label><br>  
 <button formaction="logout" class="button">Вийти</button>  
 </form>  
 </c:if>  
</header>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Бюро знахідок | Житомир</title>  
 <link rel="stylesheet" type="text/css" href="../css/reset.css">  
 <link rel="stylesheet" type="text/css" href="../css/style.css">  
 <meta charset="UTF-8">  
</head>  
<body>  
<%@include file="header.jspf"%>  
<section>  
 <div class="notification"><p>Після створення облікового запису Ви зможете публікувати власні оголошення про знахідки</p></div><br>  
 <form class="creatingForm" action="createAccount" method="post">  
 Введіть ім'я: <input type="text" name="name"><br>  
 Введіть логін: <input type="text" name="login"><br>  
 Введіть пароль: <input type="password" name="password"><br>  
 <input class="button" type="submit" value="Створити обліковий запис"><br>  
 </form>  
</section>  
<a href="." class="toMainPage">На головну</a>  
<%@include file="footer.jspf"%>  
</body>  
</html>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Бюро знахідок | Житомир</title>  
 <link rel="stylesheet" type="text/css" href="../css/reset.css">  
 <link rel="stylesheet" type="text/css" href="../css/style.css">  
 <meta charset="UTF-8">  
</head>  
<body>  
<%@include file="header.jspf" %>  
<section>  
 <c:if test="${usersFindings.size() != 0}">  
 <h2 class="textWithLines"><span>Ваші оголошення про знахідки</span></h2>  
 <table class="findings-table">  
 <thead>  
 <tr>  
 <th>Знахідка</th>  
 <th>  
 Ключові слова  
 </th>  
 <th colspan="2">  
 Час публікації  
 </th>  
 </tr>  
 </thead>  
 <tbody>  
 <c:forEach var="finding" items="${usersFindings}">  
 <tr>  
 <td title="${finding.description}">  
 <a href="finding?findingId=${finding.findingId}">${finding.title}</a>  
 </td>  
 <td>  
 <c:forEach items="${finding.keywords}" var="keyword">  
 <c:out value="${keyword}"/>  
 </c:forEach>  
 </td>  
 <td><fmt:formatDate pattern="dd.MM.yyyy HH:mm:ss" value="${finding.date}"/></td>  
 <td>  
 <form action="deleteFinding" method="POST">  
 <input type="hidden" name="findingId" value="${finding.findingId}"/>  
 <input class="deleteButton" type="submit" value="Видалити"/>  
 </form>  
 </td>  
 </tr>  
 </c:forEach>  
 </tbody>  
 </table>  
 </c:if>  
 <c:if test="${usersFindings.size() == 0}">  
 <h2 class="textWithLines"><span>У вас не має оголошень.</span></h2>  
 </c:if>  
</section>  
<a href="." class="toMainPage">На головну</a>  
<%@include file="footer.jspf" %>  
</body>  
</html>