НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Курсова робота

з дисципліни «Програмування»

на тему: «Каталог бібліотеки»

Виконала:
студентка 1 курсу, групи ІА-33
Білик Дар'я Романівна _____
(підпис)

Керівник:
асистент кафедри ІСТ
Мягкий Михайло Юрійович _____
(підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2024 року

ЗМІСТ

ВСТУП

Бібліотеки сторіччями були найвагомішими й найціннішими сховищами знань для людства, проте в епоху інноваційних технологій їх віртуальні версії набули ще більшого значення для зберігання та передачі інформації й стали доступними шлюзами до величезних ресурсів найрізноманітнішої інформації всього світу. Завдяки цим змінам у суспільстві й світі традиційні методи створення каталогів та доступу до книг, які зазвичай були основані на ручних процесах, дедалі більше не відповідають вимогам сучасних користувачів. Перехід до систем цифрових каталогів є запорукою колосального підвищення ефективності користування.

Система бібліотечного каталогу є незамінним інструментом для організації, управління та полегшення доступу до великих колекцій, що зберігаються в бібліотеці. Це дає змогу користувачам швидко й точно знаходити ресурси, покращуючи таким чином їхній дослідницький досвід. Для бібліотекарів та адміністраторів надійна система каталогів спрощує керування інвентарем, забезпечуючи ефективне оновлення, відстеження та обслуговування бібліотечних ресурсів. Публічні бібліотеки та приватні колекції зростають з кожним днем, тому потреба в динамічній та зручній системі каталогів стає тільки актуальнішою.

Основна ціль розробки комплексної системи бібліотечних каталогів полягає у створенні ефективної, зручної платформи, яка відповідає різноманітним потребам як відвідувачів, так і адміністраторів бібліотек. Для користувачів система спрямована на спрощення процесу пошуку, оскільки дозволяє їм легко знаходити книги за автором, назвою або ключовими словами. Для адміністраторів у свою чергу система надає інструменти для керування каталогом, такі як створення, редагування та видалення інформації про книги. Впровадження ефективного пошуку та необхідних засобів для адміністративного менеджменту - основні задачі системи бібліотечного каталогу.

Створення сучасної системи бібліотечних каталогів має вирішальне значення для підвищення ефективності роботи бібліотек і покращення досвіду користувачів. Завдяки інтеграції розширених функцій пошуку та комплексних інструментів

адміністрування система задовольняє потреби як відвідувачів, так і бібліотекарів. Ця ініціатива не тільки підтримує ефективне управління бібліотечними ресурсами, але й сприяє створенню середовища, сприятливого для навчання та дослідження. Впровадження такої системи є значним кроком вперед у цифровій трансформації бібліотечних послуг, гарантуючи, що бібліотеки залишатимуться актуальними та цінними в епоху інновацій.

1 ВИМОГИ ДО СИСТЕМИ

1.1 Функціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

– незареєстрований користувач повинен мати можливість переглядати інформацію про книги;

– незареєстрований користувач повинен мати можливість здійснювати пошук книг;

– незареєстрований користувач повинен мати можливість дивитися деталі про авторів;

– зареєстрований користувач повинен мати усі можливості, що є у незареєстрованого користувача, а також він повинен мати можливість додавати авторів, книги до обраних;

– адміністратор повинен мати усі можливості, що є у зареєстрованого користувача, а також він повинен мати можливість створювати, редагувати, видаляти інформацію про книги.

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, а також адміністратор.

Зареєстрованому користувачу доступна вся функціональність, що і незареєстрованому, а також можливість додавати книги та авторів до обраних. Адміністратор крім того може створювати, редагувати, видаляти інформацію про книги.

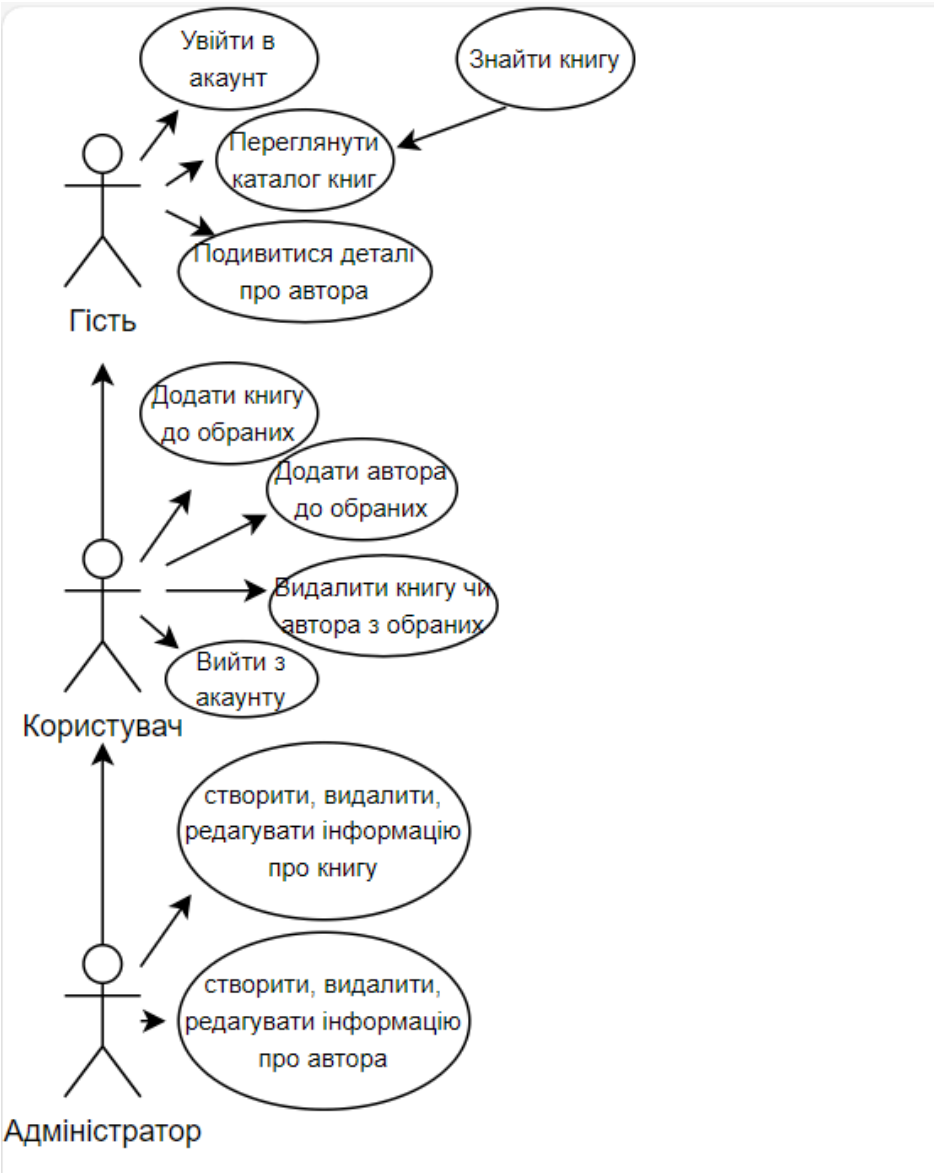

Рисунок 2.1 – Діаграма прецедентів

2.2 Опис сценаріїв використання системи

Таблиця 2.1 – Сценарій використання «Пошук по книзі, автору, ключовим словам»

Назва	Пошук по книзі, автору, ключовим словам
ID	1
Опис	Користувач, використовуючи поле для пошуку, шукає книгу, що відповідає критерію пошуку по автору, ключовим словам або назві
Актори	Користувач, зареєстрований користувач, адміністратор
Частота користування	Постійно
Тригери	Користувач вводить пошуковий запит у полі для пошуку
Передумови	Пошукове поле доступне на головній сторінці(каталог з книгами)
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	–

Таблиця 2.2 – Сценарій використання «Додавання книг чи авторів до обраного»

Назва	Додавання книг чи авторів до обраного
ID	2
Опис	Користувач додає книгу чи автора до обраного
Актори	Користувач, зареєстрований користувач, адміністратор
Частота користування	Часто
Тригери	Користувач натискає на відповідну кнопку
Передумови	Користувач знаходиться на сторінці із каталогом, результатом пошуку чи деталями про автора, натискає на відповідну кнопку
Постумови	Книгу чи автора додано до обраного, відбувається перехід користувача до сторінки акаунту, відкривається перелік обраних книг та авторів
Основний розвиток	Користувач натискає на кнопку “Додати до обраного” біля книги чи автора
Альтернативні розвитки	–
Виняткові ситуації	–

Таблиця 2.3 – Сценарій використання «Вхід до акаунту»

Назва	Вхід до акаунту
ID	3
Опис	Користувач входить у свій існуючий обліковий запис
Актори	Зареєстрований користувач, адміністратор
Частота користування	Часто
Тригери	Аби отримати доступ до розширеного функціоналу користувач виконує вхід до акаунту
Передумови	Відбувся вхід до акаунту з правами адміністратора
Постумови	Зареєстрований користувач чи адміністратор можуть перейти до сторінки акаунту а також мають доступ до усього передбаченого функціоналу
Основний розвиток	При коректному вводі логіна та пароля відбувається авторизація користувача
Альтернативні розвитки	Користувач не зареєстрований, для цього випадку є опція реєстрації після натискання на відповідну кнопку
Виняткові ситуації	Логін або пароль було введено неправильно, тоді вхід до акаунту не відбувається, з’являється повідомлення про помилку

Таблиця 2.4 – Сценарій використання «Додавання, редагування, видалення інформації про книгу чи автора»

Назва	Додавання, редагування, видалення інформації про книгу чи автора
ID	4
Опис	Адміністратор може керувати інформацією про книги та авторів
Актори	Адміністратор
Частота користування	Часто
Тригери	Адміністратор при перегляді книг й авторів обирає що саме йому потрібно змінити серед каталогу бібліотеки
Передумови	Відбувся вхід до акаунту з правами адміністратора
Постумови	Необхідна інформація буде додана, відредагована чи видалена
Основний розвиток	Адміністратор вибирає книгу чи автора, що потрібно змінити, натискає відповідну кнопку
Альтернативні розвитки	–
Виняткові ситуації	–

3 АРХІТЕКТУРА СИСТЕМИ

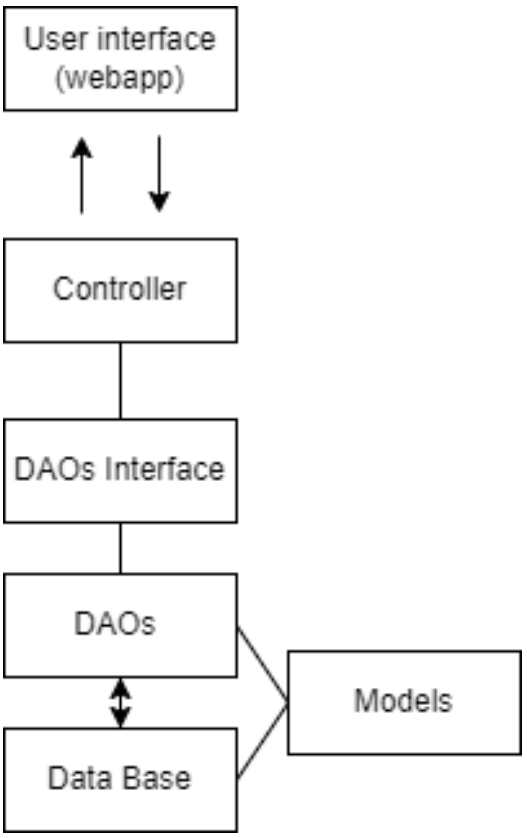

Рисунок 3.1 – Загальна архітектура системи

Система бібліотечного каталогу складається з декількох основних компонентів, кожен з яких виконує певні функції та взаємодіє з іншими компонентами для забезпечення повної функціональності системи. Архітектура системи включає такі компоненти:

- Клієнтська частина:

Відповідає за інтерфейс користувача та взаємодію з системою. Технології: HTML, CSS, JSP.

- Серверна частина:

Реалізує бізнес-логіку, обробляє запити від клієнтської частини та взаємодіє з передбаченою у більш складних та повноцінних реалізаціях базою даних. Основні компоненти: Контролери, Сервлети, DAO, Моделі, Сервіси.

- База даних(передбачена у більш складних та повноцінних реалізаціях):

Зберігає всі дані про книги, авторів, користувачів та ключові слова.

Компоненти системи та їх взаємодія

- Контролери (Controllers):

Використовують сервлети для обробки HTTP-запитів від клієнтів.

Розподіляють запити до відповідних сервісів для виконання бізнес-логіки.

- DAO (Data Access Objects):

Реалізують інтерфейси для взаємодії з базою даних.

DAO компоненти: AuthorsDAO, BooksDAO, UsersDAO.

- Інтерфейси (Interfaces):

Визначають контракт для DAO та сервісів.

Забезпечують абстракцію та полегшують заміну реалізацій.

- Моделі (Models):

Представляють дані системи та використовуються для передачі даних між різними компонентами.

-Сервіси (Services):

Виконують бізнес-логіку системи та взаємодіють з DAO для доступу до бази даних.

Взаємодія компонентів

Користувач (Гість або Адміністратор) надсилає HTTP-запит до контролера. Контролер обробляє запит і викликає відповідний метод сервісу. Сервіс виконує бізнес-логіку та звертається до DAO для доступу до бази даних. DAO виконує необхідні операції з базою даних і повертає дані сервісу. Сервіс передає дані контролеру. Контролер формує відповідь та надсилає її користувачу.

4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

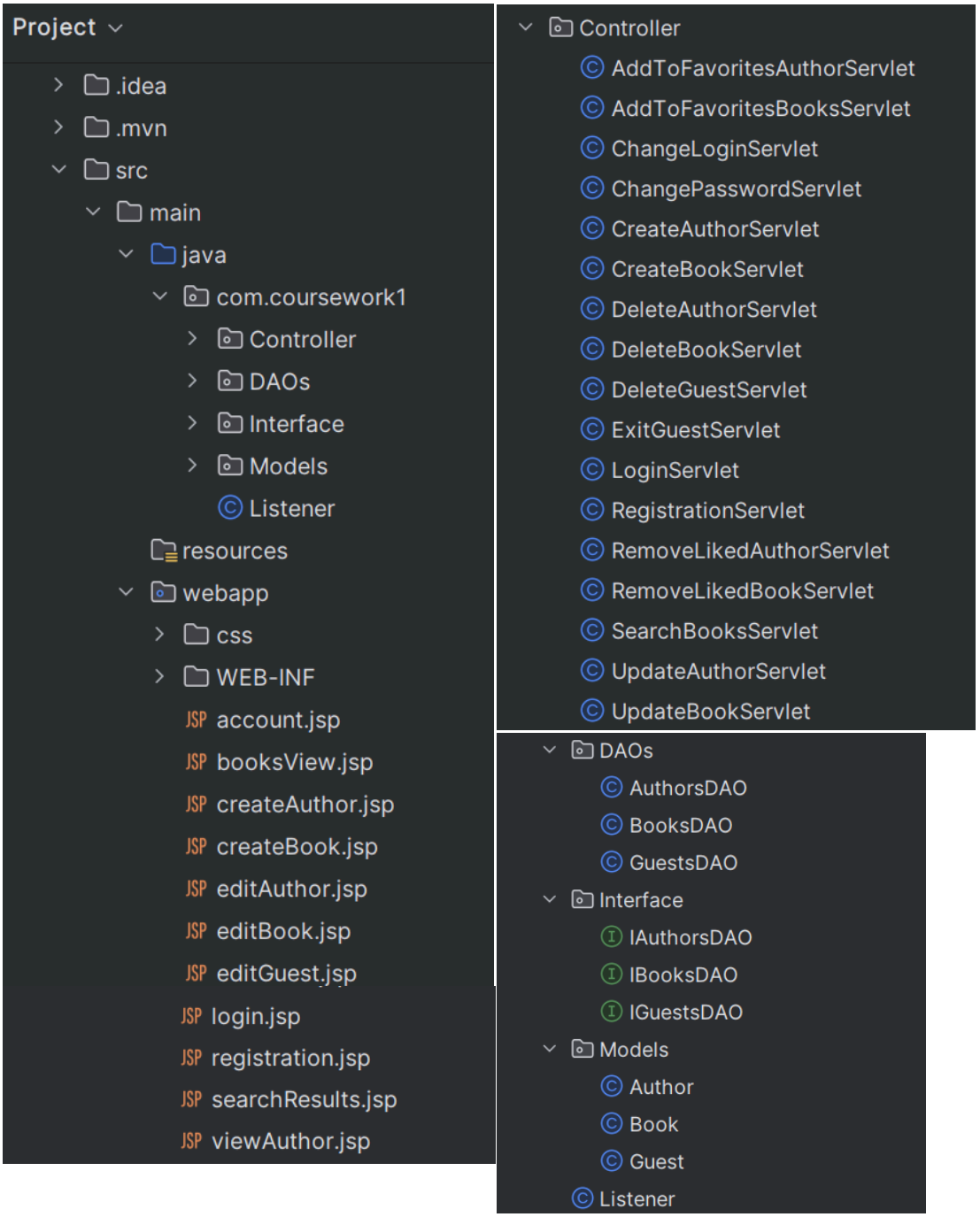

Рисунок 4.1 – Загальна структура проекту

Вихідний код проекту можна розділити на компоненти доступу до даних, бізнес-логіки та веб-компоненти.

4.2 Компоненти рівня доступу до даних

Data Access Layer - цей шар відповідає за взаємодію з даними, а саме за їх створення, читання, оновлення та видалення.

DAO Інтерфейси:

IAuthorsDAO.java

IBooksDAO.java

IGuestsDAO.java

DAO Реалізації:

AuthorsDAO

BooksDAO

GuatsDAO

Моделі (Models):

Author

Book

Guest

Сутність та зв’язки кожного з компонентів:

Моделі представляють дані, з якими працює програма. Вони визначають структуру даних, які зберігаються в базі даних, включаючи поля та їх типи. Моделі можуть включати логіку валідації для перевірки правильності даних перед їх збереженням у базу даних а також визначати зв'язки між різними об'єктами, наприклад, відношення один-до-одного, один-до-багатьох та багато-до-багатьох.

DAO є патерном проектування, який абстрагує та інкапсулює доступ до джерела даних. DAO надає інтерфейс та забезпечує методи для виконання CRUD операцій (створення, читання, оновлення, видалення даних). DAO працює з моделями, виконуючи операції з ними.

Інтерфейси DAO визначають набір методів, які повинні бути реалізовані для роботи з певними моделями. Вони використовуються як посередник між бізнес-логікою та конкретними реалізаціями DAO.

4.3 Компоненти рівня бізнес-логіки

Цей шар відповідає за обробку бізнес-логіки та координацію між DAO і веб-компонентами. У створеному проекті за логіку, яка обробляє дані, приймає рішення та виконує бізнес-правила додатка, відповідають сервлети:

AddToFavoritesAuthorServlet

AddToFavoritesBooksServlet

ChangeLoginServlet

ChangePasswordServlet

CreateAuthorServlet

CreateBookServlet

DeleteAuthorServlet

4.4 Компоненти рівня інтерфейсу користувача

JSP сторінки є веб-компонентами. Вони використовуються для створення динамічних веб-сторінок на серверній стороні, генеруючи HTML, який відправляється клієнту (веб-браузеру).

account.jsp

booksView.jsp

createAuthor.jsp

createBook.jsp

editAuthor.jsp

editBook.jsp

editGuest.jsp

тощо

ВИСНОВКИ

У процесі виконання роботи, на основі проаналізованих існуючих інтернет бібліотек, була розроблена система, яка дозволить користувачам зручно здійснювати пошук книг, а адміністраторам ефективно керувати інформацією про книги.

Згідно порядку виконання курсової роботи було визначено вимоги до системи: функціональні та нефункціональні. Після того, отримавши чітке уявлення про створюванну систему було продумано та відтворено різноманітні сценарії її використання з урахуванням прецедентів, тригерів, передумов, можливих варіантів розвитку та виняткових ситуацій. Для реалізації додатку було використано технології мови Java через їх надійність та універсальність. Завдяки таким складовим системи, як контролер з усіма сервлетами, децентралізовані системи організації, інтерфейс та певні моделі, створена система справно справляється з замисленим функціоналом. Окрім пошуку книг, система надає адміністраторам інструменти, необхідні для ефективного керування каталогом. Вони можуть додавати нові книги, оновлювати існуючі записи та видаляти застарілі або неправильні записи, забезпечуючи цілісність і актуальність каталогу. Для подальшого вдосконалення системи потужним внеском стане впровадження надійних систем баз даних по типу SQL та реновація пошуку по ключовим словам враховуючи синонімічний підбір та виправлення можливих помилок при введенні пошукового запиту користувачем.

Розробка системи бібліотечного каталогу успішно задовольнила критичні потреби сучасних бібліотек, забезпечивши зручну платформу як для відвідувачів, так і для адміністраторів. Разом з тим зберігаються такі переваги, як простота архітектури та налаштування, завдяки чому система є відкритою для покращень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Документації:

1. https://docs.oracle.com
2. https://jakarta.ee

Відомості про письменників та книжки:

3. https://uk.m.wikipedia.org
4. https://www.yakaboo.ua

Додаткові джерела щодо додатку:

5. https://www.geeksforgeeks.org
6. https://www.w3schools.com
7. https://developer.mozilla.org

ДОДАТОК А

Лістінг програми

https://github.com/BilykDaria/CourseworkBilyk.git

AddToFavoritesAuthorServlet.java

```
package com.coursework1.Controller;
import com.coursework1.DAOs.AuthorsDAO;
import com.coursework1.Models.Author;
import com.coursework1.Models.Guest;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.UUID;

@WebServlet("/addToFavoriteAuthors")
public class AddToFavoritesAuthorServlet extends HttpServlet {
   @Override
     protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException {



       String id = request.getParameter("authorId");
       UUID authorId = UUID.fromString(id);
                                  AuthorsDAO     authorsDataBase     =     (AuthorsDAO)
getServletContext().getAttribute("authorsDataBase");
       Author author = authorsDataBase.getAuthorById(authorId);

       Guest user = (Guest) request.getSession().getAttribute("user");

       if (user == null) {
           response.sendRedirect("login.jsp");
           return;
       }
       if (author != null) user.addLikedAuthor(author);
       else response.sendError(501, "Дані про автора відсутні");

       response.sendRedirect("account.jsp");
   }
}
```

AddToFavoritesBooksServlet.java

```
package com.coursework1.Controller;

import com.coursework1.DAOs.AuthorsDAO;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

@WebServlet("/addToFavorites")
public class AddToFavoritesBooksServlet extends HttpServlet {
   @Override
     protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException {

       String id = request.getParameter("bookId");
       UUID bookId = UUID.fromString(id);
                                       BooksDAO     booksDataBase     =     (BooksDAO)
getServletContext().getAttribute("booksDataBase");
       Book book = booksDataBase.getBookById(bookId);
       Guest user = (Guest) request.getSession().getAttribute("user");

       if (user == null) {
           response.sendRedirect("login.jsp");
           return;
       }

       if (book != null) {
           user.addLikedBook(book);
       } else response.sendError(501, "Дані про книгу відсутні");

       response.sendRedirect("account.jsp");
   }
}
```

ChangeLoginServlet.java

```
package com.coursework1.Controller;
```

```
import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Guest;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/changeLogin")
public class ChangeLoginServlet extends HttpServlet {

   private GuestsDAO guestsDataBase;

   @Override
   public void init() {
       guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
   }

   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
           throws ServletException, IOException {

       Guest user = (Guest) request.getSession().getAttribute("user");

       if (user == null) {
           System.out.println("user is null");
       }

       if (user != null) {

           String newUsername = request.getParameter("newUsername");

                           if (newUsername != null && !newUsername.isEmpty() &&
!guestsDataBase.isRegisteredUser(newUsername)) {
               user.setLogin(newUsername);
               request.getSession().setAttribute("user", user);
               request.setAttribute("usernameMessage", "Логін успішно змінений");

           } else {
               if (newUsername == null || newUsername.isEmpty()) {
                   request.setAttribute("usernameError", "Ви не ввели новий логін");
```

```
                } else {
                    request.setAttribute("usernameError", "Цей логін вже зайнятий");
                }
            }
            request.getRequestDispatcher("editGuest.jsp").forward(request, response);

        } else {
            response.sendError(500, "Об'єкт користувача нульовий");
        }
    }
}
```

ChangePasswordServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Guest;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;


@WebServlet("/changePassword")
public class ChangePasswordServlet extends HttpServlet {


    private GuestsDAO guestsDataBase;


    @Override
    public void init() {
        guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
    }


    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
        Guest user = (Guest) request.getSession().getAttribute("user");


        if (user == null) {
            System.out.println("user is null");
        }


        if (user != null) {


            String newPassword = request.getParameter("newPassword");


            if (newPassword != null && !newPassword.isEmpty() &&
!guestsDataBase.isRegisteredUser(user.getLogin())) {
                user.setPassword(newPassword);
                request.getSession().setAttribute("user", user);
                request.setAttribute("passwordMessage", "Пароль успішно змінений");


            } else {
                if (newPassword == null) {
                    request.setAttribute("passwordError", "Ви не ввели новий пароль");
                }
            }
            request.getRequestDispatcher("editGuest.jsp").forward(request, response);


        } else {
            response.sendError(500, "Об'єкт користувача нульовий");
        }
    }
}
```

CreateAuthorServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.AuthorsDAO;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
```

```
import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;


import java.io.IOException;


@WebServlet("/createAuthor")

public class CreateAuthorServlet extends HttpServlet {

   @Override

   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

       String name = request.getParameter("name");

       String biography = request.getParameter("biography");

       String country = request.getParameter("country");

       int age = Integer.parseInt(request.getParameter("age"));


       AuthorsDAO authorsDataBase = (AuthorsDAO)
getServletContext().getAttribute("authorsDataBase");

       if (authorsDataBase == null) {

           authorsDataBase = new AuthorsDAO();

           getServletContext().setAttribute("authorsDataBase", authorsDataBase);

       }


       if (authorsDataBase.isAuthorInDataBase(name)) {

           response.sendError(400, "Автор з таким іменем вже існує");

           return;

       }


       authorsDataBase.addAuthor(name, biography, country, age);


       response.sendRedirect("booksView.jsp");

   }

}
```

CreateBookServlet.java

```
package com.coursework1.Controller;
```

```
import com.coursework1.DAOs.AuthorsDAO;
import com.coursework1.DAOs.BooksDAO;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

@WebServlet("/createBook")
public class CreateBookServlet extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
       String name = request.getParameter("name");
       String authorName = request.getParameter("authorName");
       String description = request.getParameter("description");



       BooksDAO booksDataBase = (BooksDAO)
getServletContext().getAttribute("booksDataBase");

       if (booksDataBase.isBookInDataBase(name)) {
           response.sendError(400, "Книга з такою назвою вже існує");
           return;
       }

       AuthorsDAO authorsDataBase = (AuthorsDAO)
getServletContext().getAttribute("authorsDataBase");
       if (authorsDataBase.isAuthorInDataBase(authorName)) {
           booksDataBase.addBook(name, authorsDataBase.getAuthorByName(authorName),
description);
       } else response.sendError(500, "Не вийшло додати книгу");

       response.sendRedirect("booksView.jsp");
   }
```

```
}
```

DeleteAuthorServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.AuthorsDAO;


import com.coursework1.DAOs.BooksDAO;

import com.coursework1.Models.Book;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.Set;

import java.util.UUID;


@WebServlet("/deleteAuthor")

public class DeleteAuthorServlet extends HttpServlet {

   @Override

   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

       UUID authorId = UUID.fromString(request.getParameter("authorId"));


       AuthorsDAO authorsDAO = (AuthorsDAO)
getServletContext().getAttribute("authorsDataBase");

       Set<Book> books = authorsDAO.getAuthorById(authorId).getWrittenBooks();

       BooksDAO booksDataBase = (BooksDAO)
getServletContext().getAttribute("booksDataBase");


       for (Book book : books) {

           booksDataBase.deleteBookById(book.getBookId());

       }


       authorsDAO.deleteAuthor(authorId);
```

```
        response.sendRedirect("viewAuthor.jsp");
    }
}
```

DeleteBookServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.BooksDAO;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.UUID;


@WebServlet("/deleteBook")
public class DeleteBookServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        UUID bookId = UUID.fromString(request.getParameter("bookId"));
        BooksDAO booksDAO = (BooksDAO)
getServletContext().getAttribute("booksDataBase");


        booksDAO.getBooksDataBase().remove(bookId);


        response.sendRedirect("booksView.jsp");
    }
}
```

DeleteGuestServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Guest;
```

```
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/deleteGuest")
public class DeleteGuestServlet extends HttpServlet {

    private GuestsDAO guestsDataBase;

    @Override
    public void init() {
        guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {

        Guest user = (Guest) request.getSession().getAttribute("user");



        if (user != null && guestsDataBase.isRegisteredUser(user.getLogin())) {

            guestsDataBase.deleteUser(user);
            request.getSession().invalidate();
            response.sendRedirect("booksView.jsp");

        } else {
            response.sendError(500, "Помилка при видаленні акаунту");
        }
    }
}
```

ExitGuestServlet.java

```
package com.coursework1.Controller;
```

```
import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Guest;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/exitGuest")
public class ExitGuestServlet extends HttpServlet {

    private GuestsDAO guestsDataBase;

    @Override
    public void init() {
        guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {

        Guest user = (Guest) request.getSession().getAttribute("user");

        if (user != null && guestsDataBase.isRegisteredUser(user.getLogin())) {
            request.getSession().invalidate();
            response.sendRedirect("booksView.jsp");
        } else {
            response.sendError(500, "Помилка при виході з акаунту");
        }
    }
}
```

LoginServlet.java

```
package com.coursework1.Controller;
```

```
import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Guest;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    private GuestsDAO guestsDataBase;

    @Override
    public void init() {
        guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        if (guestsDataBase.isRegisteredUser(username)) {
            Guest user = guestsDataBase.findByLogin(username);

            if (user != null && user.getPassword().equals(password)) {
                request.getSession().setAttribute("user", user);
                response.sendRedirect("booksView.jsp");

            } else {
                request.setAttribute("errorMessage", "Неправильний логін або пароль");
                request.getRequestDispatcher("login.jsp").forward(request, response);
            }
```

```
        } else {
            request.setAttribute("errorMessage", "Користувача з таким логіном не
знайдено");
            request.getRequestDispatcher("login.jsp").forward(request, response);
        }
    }
}
```

RegistrationServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.GuestsDAO;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;


@WebServlet("/registration")
public class RegistrationServlet extends HttpServlet {


    private GuestsDAO guestsDataBase;


    @Override
    public void init() {
        guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
    }


    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {


        String username = request.getParameter("username");
        String password = request.getParameter("password");
```

```
        if (guestsDataBase.isRegisteredUser(username)) {
            request.setAttribute("errorMessage", "Користувач з таким логіном
зареєстрований");
            request.getRequestDispatcher("registration.jsp").forward(request, response);
        } else {
            guestsDataBase.createUser(username, password);
            request.setAttribute("successMessage", "Ви успішно зареєструвалися");
            request.getRequestDispatcher("registration.jsp").forward(request, response);
        }
    }
}
```

RemoveLikedAuthorServlet.java

```
package com.coursework1.Controller;

import com.coursework1.DAOs.GuestsDAO;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/registration")
public class RegistrationServlet extends HttpServlet {

    private GuestsDAO guestsDataBase;

    @Override
    public void init() {
        guestsDataBase = (GuestsDAO) getServletContext().getAttribute("guestsDataBase");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
```

```
        String username = request.getParameter("username");
        String password = request.getParameter("password");



        if (guestsDataBase.isRegisteredUser(username)) {
            request.setAttribute("errorMessage", "Користувач з таким логіном
зареєстрований");
            request.getRequestDispatcher("registration.jsp").forward(request, response);
        } else {
            guestsDataBase.createUser(username, password);
            request.setAttribute("successMessage", "Ви успішно зареєструвалися");
            request.getRequestDispatcher("registration.jsp").forward(request, response);
        }
    }
}
```

RemoveLikedBookServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Book;
import com.coursework1.Models.Guest;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.UUID;


@WebServlet("/removeFromFavorites")
public class RemoveLikedBookServlet extends HttpServlet {


    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException {
```

```
        Guest currentUser = (Guest) request.getSession().getAttribute("user");

        if (currentUser == null) {
            response.sendRedirect("login.jsp");
            return;
        }

        UUID bookId = UUID.fromString(request.getParameter("bookId"));

        Book bookToRemove = null;
        for (Book book : currentUser.getLikedBooks()) {
            if (book.getBookId().equals(bookId)) {
                bookToRemove = book;
                break;
            }
        }

        if (bookToRemove != null) {
            currentUser.deleteGuestLikedBook(bookToRemove);
        }

        GuestsDAO guestsDAO = (GuestsDAO)
getServletContext().getAttribute("guestsDataBase");
        guestsDAO.setUsersDataBase(guestsDAO.getUsersDataBase());

        response.sendRedirect("account.jsp");
    }
}
```

SearchBooksServlet.java

```
package com.coursework1.Controller;

import com.coursework1.DAOs.BooksDAO;
import com.coursework1.Models.Book;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
```

```
import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;


import java.io.IOException;

import java.util.List;


@WebServlet("/searchBooks")

public class SearchBooksServlet extends HttpServlet {

   @Override

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

       String keyword = request.getParameter("keyword");


       BooksDAO booksDataBase = (BooksDAO)
getServletContext().getAttribute("booksDataBase");

       if (booksDataBase == null) {

           booksDataBase = new BooksDAO();

           getServletContext().setAttribute("booksDataBase", booksDataBase);

       }


       List<Book> searchResults = booksDataBase.searchBooksByKeyword(keyword);

       request.setAttribute("searchResults", searchResults);

       request.getRequestDispatcher("searchResults.jsp").forward(request, response);

   }

}
```

UpdateAuthorServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.AuthorsDAO;

import com.coursework1.Models.Author;


import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;
```

```
import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.UUID;


@WebServlet("/updateAuthor")

public class UpdateAuthorServlet extends HttpServlet {

   @Override

   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

       UUID authorId = UUID.fromString(request.getParameter("authorId"));

       String name = request.getParameter("name");

       String biography = request.getParameter("biography");

       String country = request.getParameter("country");

       int age = Integer.parseInt(request.getParameter("age"));


       AuthorsDAO authorsDAO = (AuthorsDAO)
getServletContext().getAttribute("authorsDataBase");


       Author author = authorsDAO.getAuthorById(authorId);

       if (author != null) {

           author.setName(name);

           author.setBiography(biography);

           author.setCountry(country);

           author.setAge(age);


       } else response.sendError(500, "Автора не знайдено");


       response.sendRedirect("viewAuthor.jsp");

   }

}
```

UpdateBookServlet.java

```
package com.coursework1.Controller;


import com.coursework1.DAOs.AuthorsDAO;
```

```
import com.coursework1.DAOs.BooksDAO;

import com.coursework1.Models.Book;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.UUID;


@WebServlet("/updateBook")

public class UpdateBookServlet extends HttpServlet {

   @Override

   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException {

       String id = request.getParameter("bookId");

       UUID bookId = UUID.fromString(id);

       BooksDAO booksDataBase = (BooksDAO)
request.getServletContext().getAttribute("booksDataBase");

       Book book = booksDataBase.getBookById(bookId);

       String newName = request.getParameter("name");

       String newAuthorName = request.getParameter("author");

       String newDescription = request.getParameter("description");


       AuthorsDAO authorsDataBase = (AuthorsDAO)
getServletContext().getAttribute("authorsDataBase");


       if (newName.isEmpty() || newAuthorName.isEmpty() || newDescription.isEmpty()) {

           response.sendError(500, "Введіть значення в поля");

       }


       if (authorsDataBase.getAuthorByName(newAuthorName) == null) {

           response.sendError(400, "Автор не знайдений");

           return;

       }


       if (book != null) {

           book.setName(newName);
```

```
            book.setAuthor(authorsDataBase.getAuthorByName(newAuthorName));
            book.setDescription(newDescription);
        }

        response.sendRedirect("account.jsp");
    }
}
```

AutorsDAO.java

```
package com.coursework1.DAOs;

import com.coursework1.Interface.IAuthorsDAO;
import com.coursework1.Models.Author;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

public class AuthorsDAO implements IAuthorsDAO {
    private Map<UUID, Author> authorsDataBase = new HashMap<>();

    @Override
    public Map<UUID, Author> getAuthorsDataBase() {
        return authorsDataBase;
    }

    @Override
    public void setAuthorsDataBase(Map<UUID, Author> authorsDataBase) {
        this.authorsDataBase = authorsDataBase;
    }

    @Override
    public Author getAuthorByName(String name) {
        for (Author author : authorsDataBase.values()) {
            if (author.getName().equalsIgnoreCase(name)) {
                return author;
```

```
            }
        }
        return null;
    }

    @Override
    public Author getAuthorById(UUID id) {
        return authorsDataBase.get(id);
    }

    @Override
    public Author addAuthor(String name, String biography, String country, int age) {
        Author author = new Author(name, biography, country, age);
        authorsDataBase.put(author.getId(), author);
        return author;
    }

    @Override
    public boolean isAuthorInDataBase(String name) {
        for (Author author : authorsDataBase.values()) {
            if (author.getName().equalsIgnoreCase(name)) {
                return true;
            }
        }
        return false;
    }

    @Override
    public void deleteAuthor(UUID authorId) {
        authorsDataBase.remove(authorId);
    }
}
```

BooksDAO.java

```
package com.coursework1.DAOs;
```

```java
import com.coursework1.Interface.IBooksDAO;
import com.coursework1.Models.Author;
import com.coursework1.Models.Book;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.stream.Collectors;

public class BooksDAO implements IBooksDAO {
    private HashMap<UUID, Book> booksDataBase = new HashMap<>();

    @Override
    public Map<UUID, Book> getBooksDataBase() {
        return booksDataBase;
    }

    @Override
    public void setBooksDataBase(HashMap<UUID, Book> booksDataBase) {
        this.booksDataBase = booksDataBase;
    }

    @Override
    public Book getBookById(UUID id) {
        return booksDataBase.get(id);
    }

    @Override
    public void addBook(String name, Author author, String description) {
        Book book = new Book(name, author, description);
        author.addAuthorBook(book);
        booksDataBase.put(book.getBookId(), book);
    }
```

```
    @Override
    public Book getBookByName(String name) {
        for (Book book : booksDataBase.values()) {
            if (book.getName().equalsIgnoreCase(name)) {
                return book;
            }
        }
        return null;
    }

    @Override
    public boolean isBookInDataBase(String name) {
        for (Book book : booksDataBase.values()) {
            if (book.getName().equalsIgnoreCase(name)) {
                return true;
            }
        }
        return false;
    }

    @Override
    public void deleteBookById(UUID bookId) {
        booksDataBase.remove(bookId);
    }

    @Override
    public List<Book> searchBooksByKeyword(String keyword) {
        List<Book> filteredBooks = booksDataBase.values().stream()
                .filter(book ->
book.getName().toLowerCase().contains(keyword.toLowerCase()))
                .collect(Collectors.toList());

        if (filteredBooks.isEmpty()) {
            filteredBooks = booksDataBase.values().stream()
                    .filter(book ->
book.getAuthor().getName().toLowerCase().contains(keyword.toLowerCase()))
                    .collect(Collectors.toList());
```

```
        }


        return filteredBooks;
    }
}
```

GuestsDAO.java

```
package com.coursework1.DAOs;


import com.coursework1.Interface.IGuestsDAO;
import com.coursework1.Models.Guest;


import java.util.*;


public class GuestsDAO implements IGuestsDAO {
    private Set<Guest> guestsDataBase = new
TreeSet<>(Comparator.comparing(Guest::getLogin));


    @Override
    public void setUsersDataBase(Set<Guest> usersDataBase) {
        this.guestsDataBase = usersDataBase;
    }


    @Override
    public Set<Guest> getUsersDataBase() {
        return guestsDataBase;
    }


    @Override
    public boolean isRegisteredUser(String username) {
        for (Guest guest: guestsDataBase) {
            if (username.equals(guest.getLogin())) {
                return true;
            }
        }
        return false;
```

```
}

@Override
public Guest findByLogin(String username) {
    for (Guest guest : guestsDataBase) {
        if (guest.getLogin().equals(username)) {
            return guest;
        }
    }
    return null;
}

@Override
public void createUser(String username, String password) {
    if (!isRegisteredUser(username)) {
        guestsDataBase.add(new Guest(username, password));
    }
}

@Override
public void createUser(String username, String password, Guest.Role role) {
    if (!isRegisteredUser(username)) {
        guestsDataBase.add(new Guest(username, password, role));
    }
}

@Override
public void addUser(Guest guest) {
    if (!isRegisteredUser(guest.getLogin())) {
        guestsDataBase.add(guest);
    }
}

@Override
public void deleteUser(Guest guest) {
    if (isRegisteredUser(guest.getLogin())) {
```

```
            guestsDataBase.remove(guest);
        }
    }
}
```

IAutorsDAO.java

```
package com.coursework1.Interface;

import com.coursework1.Models.Author;

import java.util.Map;
import java.util.UUID;

public interface IAuthorsDAO {
    Map<UUID, Author> getAuthorsDataBase();
    void setAuthorsDataBase(Map<UUID, Author> authorsDataBase);
    Author getAuthorByName(String name);
    Author getAuthorById(UUID id);
    Author addAuthor(String name, String biography, String country, int age);
    boolean isAuthorInDataBase(String name);
    void deleteAuthor(UUID authorId);
}
```

IBooksDAO.java

```
package com.coursework1.Interface;

import com.coursework1.Models.Author;
import com.coursework1.Models.Book;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

public interface IBooksDAO {
    Map<UUID, Book> getBooksDataBase();
```

```
    void setBooksDataBase(HashMap<UUID, Book> booksDataBase);
    Book getBookById(UUID id);
    void addBook(String name, Author author, String description);
    Book getBookByName(String name);
    boolean isBookInDataBase(String name);
    void deleteBookById(UUID bookId);
    List<Book> searchBooksByKeyword(String keyword);
}
```

IGuestsDAO.java

```
package com.coursework1.Interface;

import com.coursework1.Models.Guest;
import java.util.Set;

public interface IGuestsDAO {
    void setUsersDataBase(Set<Guest> usersDataBase);
    Set<Guest> getUsersDataBase();
    boolean isRegisteredUser(String username);
    Guest findByLogin(String username);
    void createUser(String username, String password);
    void addUser(Guest guest);
    void deleteUser(Guest guest);
    void createUser(String username, String password, Guest.Role role);
}
```

Author.java

```
package com.coursework1.Models;

import java.util.*;

public class Author {
    private UUID id;
    private String name;
    private String biography;
```

```
    private String country;
    private int age;
    private TreeSet<Book> writtenBooks;

    public Author(String name, String biography, String country, int age) {
        this.id = UUID.randomUUID();
        this.name = name;
        this.biography = biography;
        this.country = country;
        this.age = age;
        this.writtenBooks = new TreeSet<>(Comparator.comparing(Book::getName));
    }

    public Set<Book> getWrittenBooks() {
        return writtenBooks;
    }

    public void setWrittenBooks(TreeSet<Book> writtenBooks) {
        this.writtenBooks = writtenBooks;
    }

    public void addAuthorBook(Book book) {
        writtenBooks.add(book);
    }

    public UUID getId() {
        return id;
    }

    public void setId(UUID id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getBiography() {
        return biography;
    }

    public void setBiography(String biography) {
        this.biography = biography;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

Book.java

```
package com.coursework1.Models;

import java.util.UUID;
```

```java
public class Book {
    private UUID bookId;
    private String name;
    private Author author;
    private String description;

    public Book(String name, Author author, String description) {
        this.bookId = UUID.randomUUID();
        this.name = name;
        this.author = author;
        this.description = description;
    }

    public UUID getBookId() {
        return bookId;
    }

    public void setBookId(UUID bookId) {
        this.bookId = bookId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Author getAuthor() {
        return author;
    }

    public void setAuthor(Author author) {
        this.author = author;
    }
```

```
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

Guest.java

```
package com.coursework1.Models;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

public class Guest {

    private UUID id;
    private String login;
    private String password;
    private ArrayList<Book> likedBooks;
    private ArrayList<Author> likedAuthors;
    private Role role;


    public Guest(String login, String password) {
        this.id = UUID.randomUUID();
        this.login = login;
        this.password = password;
        this.likedBooks = new ArrayList<>();
        this.likedAuthors = new ArrayList<>();
        this.role = Role.GUEST;
    }
```

```
public Guest(String login, String password, Role role) {
    this.id = UUID.randomUUID();
    this.login = login;
    this.password = password;
    this.likedBooks = new ArrayList<>();
    this.likedAuthors = new ArrayList<>();
    this.role = role;
}

public void setLogin(String login) {
    this.login = login;
}

public void setPassword(String password) {
    this.password = password;
}

public void setId(UUID id) {
    this.id = id;
}

public ArrayList<Author> getLikedAuthors() {
    return likedAuthors;
}

public void setLikedAuthors(ArrayList<Author> likedAuthors) {
    this.likedAuthors = likedAuthors;
}

public void addLikedBook(Book book) {
    this.likedBooks.add(book);
}

public void addLikedAuthor(Author author) {
    this.likedAuthors.add(author);
```

```
}

public String getLogin() {
    return login;
}

public String getPassword() {
    return password;
}

public UUID getId() {
    return id;
}

public List<Book> getLikedBooks() {
    return likedBooks;
}

public void deleteGuestLikedBook(Book book) {
    likedBooks.remove(book);
}

public void setLikedBooks(ArrayList<Book> likedBooks) {
    this.likedBooks = likedBooks;
}

public void setRole(Role role) {
    this.role = role;
}

public Role getRole() {
    return role;
}

public enum Role {
    ADMIN, GUEST
```

```
    }


}
```

Listener.java

```
package com.coursework1;


import com.coursework1.DAOs.AuthorsDAO;
import com.coursework1.DAOs.BooksDAO;
import com.coursework1.DAOs.GuestsDAO;
import com.coursework1.Models.Author;
import jakarta.servlet.ServletContextEvent;
import jakarta.servlet.ServletContextListener;
import jakarta.servlet.annotation.WebListener;

import static com.coursework1.Models.Guest.Role.ADMIN;


@WebListener
public class Listener implements ServletContextListener {


    private BooksDAO booksDataBase;
    private GuestsDAO guestsDataBase;
    private AuthorsDAO authorsDataBase;


    @Override
    public void contextInitialized(ServletContextEvent context) {


        booksDataBase = new BooksDAO();
        authorsDataBase = new AuthorsDAO();
        guestsDataBase = new GuestsDAO();


        guestsDataBase.createUser("dasha", "dasha");
        guestsDataBase.createUser("admin", "111", ADMIN);


        Author author1 = authorsDataBase.addAuthor("Тарас Шевченко", "Тара́с Григо́рович
Шевче́нко (25 лютого (9 березня) " +
```

```
                "1814, с. Моринці, Київська губернія, Російська імперія (нині
Звенигородський район, Черкаська область, Україна)" +

                " — 26 лютого (10 березня) 1861, Санкт-Петербург, Російська імперія) —
український поет, прозаїк, мислитель, " +

                "живописець, гравер, етнограф, громадський діяч. Національний герой і
символ України. Діяч українського " +

                "національного руху, член Кирило-Мефодіївського братства. Академік
Імператорської академії мистецтв (1860).",

                "Україна", 46);


        Author author2 = authorsDataBase.addAuthor("Оноре де Бальзак", "Народився 20
травня 1799 року в Турі," +

                        "Франція, помер 18 серпня 1850 року. Відомий французький
письменник, один з найвидатніших представників реалізму.",

                "Франція", 51);


        Author author3 = authorsDataBase.addAuthor("Марк Твен", "Народився 30 листопада
1835 року у Флориді, США, помер" +

                        " 21 квітня 1910 року. Відомий американський письменник,
гуморист і сатирик.",

                "США", 74);


        Author author4 = authorsDataBase.addAuthor("Чарльз Діккенс", "Народився 7 лютого
1812 року в Портсмуті," +

                        " помер 9 червня 1870 року. Він був одним з найвизначніших
англійських романістів вікторіанської епохи.",

                "Англія", 58);


        Author author5 = authorsDataBase.addAuthor("Джейн Остін", "Народилася 16 грудня
1775 року в Стівентоні," +

                        " Англія, померла 18 липня 1817 року. Вона була однією з
найвизначніших англійських письменниць," +

                        " відомих своїми романами про життя англійського дворянства.",

                "Англія", 41);


        Author author6 = authorsDataBase.addAuthor("Джордж Еліот", "Справжнє ім'я - Мері
Енн Еванс. Народилася" +

                        " 22 листопада 1819 року в Нейнітчі, Англія, померла 22 грудня
1880 року. Вона була англійською письменницею," +

                        " відомою своїми реалістичними романами про сільське життя.",

                "Англія", 61);
```

```
        Author author7 = authorsDataBase.addAuthor("Гюстав Флобер", "Народився 12 грудня
1821 року в Руані, Франція," +
                        "помер 8 травня 1880 року. Відомий французький письменник, один
з основоположників літературного реалізму.",
                "Франція", 58);


        booksDataBase.addBook("Кобзар", author1, "«Кобза́р» — назва першої книги-збірки
поетичних творів Тараса " +
                "Шевченка 1840 року. У наш час під назвою «Кобзар» видають повне
зібрання українських поезій Шевченка з додатком " +
                "двох російських поем «Тризна (Безталанний)» та «Сліпа»." );
        booksDataBase.addBook("Гайдамаки", author1, "Історична поема про повстання
гайдамаків проти польського " +
                "панування у XVIII столітті, яка підкреслює прагнення українців до
свободи." );
        booksDataBase.addBook("Батько Горіо", author2, "Роман про старого чоловіка, який
жертвує всім заради своїх невдячних дочок." );
        booksDataBase.addBook("Виховання почуттів", author7, "Роман про юнака Фредеріка
Моро, який переживає складні любовні і політичні перипетії у Франції XIX століття." );
        booksDataBase.addBook("Олівер Твіст", author4, "Роман про сироту Олівера Твіста,
який зіштовхується з жорстокістю і несправедливістю Лондона, але зрештою знаходить своє
місце в житті." );
        booksDataBase.addBook("Великі сподівання", author4, "Розповідь про юного Піпа,
який отримує можливість піднятися вгору по соціальній драбині завдяки таємничому
благодійнику." );
        booksDataBase.addBook("Девід Копперфільд", author4, "Автобіографічний роман про
життя юнака Девіда Копперфільда, який переживає багато труднощів і пригод, перш ніж
знайти своє місце в житті." );
        booksDataBase.addBook("Міддлмарч", author6, "Роман про життя і моральні дилеми
жителів вигаданого англійського містечка Міддлмарч." );
        booksDataBase.addBook("Сайлас Марнер", author6, "Роман про вигнанця Сайласа
Марнера, який знаходить нове життя і щастя, усиновивши маленьку дівчинку." );
        booksDataBase.addBook("Адам Бід", author6, "Роман про кохання і соціальні
конфлікти в англійському селі." );
        booksDataBase.addBook("Пригоди Тома Сойєра", author3, "Роман про пригоди
хлопчика Тома Сойєра на берегах річки Міссісіпі." );
        booksDataBase.addBook("Гордість і упередження", author5, "Роман про Елізабет
Беннет і її стосунки з містером Дарсі, що включає соціальні і моральні теми." );
        booksDataBase.addBook("Янкі з Коннектикуту при дворі короля Артура", author3,
"Сучасний американець потрапляє в середньовічну Англію і намагається використати свої
знання для покращення життя." );


        context.getServletContext().setAttribute("guestsDataBase", guestsDataBase);
```

```
        context.getServletContext().setAttribute("booksDataBase", booksDataBase);
        context.getServletContext().setAttribute("authorsDataBase", authorsDataBase);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        guestsDataBase = null;
        authorsDataBase = null;
        booksDataBase = null;
    }
}
```

styles.css

```
body {
    font-family: Arial, sans-serif;
    background: linear-gradient(45deg, #795548, #d7ccc8, #e4c0a8);
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
.container {
    width: 100%;
    max-width: 400px;
    padding: 20px;
    background-color: rgba(255, 255, 255, 0.9);
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    border-radius: 8px;
    position: relative;
}

h2 {
    text-align: center;
```

```
    margin-bottom: 20px;
    color: #4e342e;
    font-style: italic;
    font-family: "Times New Roman", Times, serif;
}
.form-group {
    margin-bottom: 15px;
}
.form-group label {
    display: block;
    margin-bottom: 5px;
    color: #6d4c41;
}
.form-group input {
    width: calc(100% - 20px);
    padding: 8px 10px;
    font-size: 16px;
    border: 1px solid #d7ccc8;
    border-radius: 15px;
    background-color: #efebe9;
    color: #4e342e;
}
.buttons {
    display: flex;
    justify-content: space-between;
}
.buttons button {
    width: 48%;
    padding: 10px;
    font-size: 16px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    align-items: center;
}
.buttons .login-btn {
```

```
    background-color: #8d6e63;
    color: #fff;
}
.buttons .register-btn {
    background-color: #795548;
    color: #fff;
}
.buttons_2type {
    display: flex;
    justify-content: center;
}
.buttons_2type button {
    width: 48%;
    padding: 10px;
    font-size: 16px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    align-items: center;
    background-color: #795548;
    color: #fff;
}
.back-link {
    display: block;
    text-align: center;
    margin-top: 20px;
    color: #795548;
    text-decoration: none;
}
.back-link:hover {
    text-decoration: underline;
}
```

stylesMain.css

```
body {
```

```
    background: linear-gradient(#d7ccc8, 88.2%, #8d6e63)no-repeat  fixed;
    font-family: Arial, sans-serif;
}
.buttons {
    padding: 10px;
    font-size: 16px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    align-items: center;
    background-color: #8d6e63;
    color: #fff;
}
h1 {
    padding-left: 10%;
    margin-bottom: 20px;
    color: #4e342e;
    font-style: italic;
    font-family: "Times New Roman", Times, serif;
}
.links {
    margin-top: 20px;
    padding-top: 10px;
    margin-bottom: 20px;
    padding-bottom: 10px;
    font-size: 20px;
    font-weight: bold;
    outline: none;
    text-decoration: none;
    color: #704170;
}

input {
    padding: 8px 10px;
    font-size: 16px;
    border: 1px solid #d7ccc8;
```

```
    border-radius: 15px;
    background-color: #efebe9;
    color: #4e342e;
}
```

account.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
    <title>Акаунт користувача</title>
    <style>
    table {
    width: 100%;
    border-collapse: collapse;
    }
    table th, table td {
    padding: 12px;
    border: 1px solid #000;
    text-align: left;
    }</style>
</head>
<body>

<h1>Акаунт користувача: ${user.login}</h1>

<a href="editGuest.jsp" class="links">Редагувати акаунт</a> | <a class="links"
href="booksView.jsp">Повернутись до каталогу</a>

<c:if test="${user.role == 'ADMIN'}">
    <form action="createBook.jsp" method="get">
        <button class="buttons" type="submit">Додати книгу</button>
    </form>
</c:if>
<c:if test="${user.role == 'ADMIN'}">
```

```
    <form action="createAuthor.jsp" method="get">
        <button class="buttons" type="submit">Додати автора</button>
    </form>
</c:if>
<h2>Вподобані книги</h2>
<table border="1">
    <tr>
        <th>Назва книги</th>
        <th>Автор</th>
        <th>Опис</th>
        <th>Дії</th>
    </tr>
    <c:forEach var="book" items="${user.likedBooks}">
        <tr>
            <td>${book.name}</td>
            <td>${book.author.name}</td>
            <td>${book.description}</td>
            <td>
                <c:if test="${user.role == 'ADMIN'}">
                    <form action="editBook.jsp" method="get" style="display:inline;">
                        <input type="hidden" name="bookId" value="${book.bookId}"/>
                        <button class="buttons" type="submit">Редагувати книгу</button>
                    </form>
                </c:if>
                <form action="removeFromFavorites" method="post"
style="display:inline;">
                    <input type="hidden" name="bookId" value="${book.bookId}"/>
                    <button class="buttons" type="submit">Видалити з обраних</button>
                </form>
            </td>
        </tr>
    </c:forEach>
</table>


<h2>Вподобані автори</h2>
<table border="1">
```

```
    <tr>
        <th>Ім'я автора</th>
        <th>Біографія</th>
        <th>Країна</th>
        <th>Дії</th>
    </tr>
    <c:forEach var="author" items="${user.likedAuthors}">
        <tr>
            <td>${author.name}</td>
            <td>${author.biography}</td>
            <td>${author.country}</td>
            <td>
                <form action="removeAuthorFromFavorites" method="post"
style="display:inline;">
                    <input type="hidden" name="authorId" value="${author.id}"/>
                    <button class="buttons" type="submit">Видалити з обраних</button>
                </form>
            </td>
        </tr>
    </c:forEach>
</table>


</body>
</html>
```

booksView.jsp

```
<%@ page import="com.coursework1.DAOs.BooksDAO" %>
<%@ page import="com.coursework1.Models.Book" %>
<%@ page import="java.util.Map" %>
<%@ page import="java.util.UUID" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
    <title>Каталог книг</title>
```

```
    <style>
    table {
        width: 100%;
        border-collapse: collapse;
    }
    table th, table td {
        padding: 12px;
        border: 1px solid #000;
        text-align: left;
    }
    input {
        padding: 8px 10px;
        font-size: 16px;
        border: 1px solid #d7ccc8;
        border-radius: 15px;
        background-color: #efebe9;
        color: #4e342e;
    }
</style>
</head>
<body>
<%
    BooksDAO booksDAO = (BooksDAO)
request.getServletContext().getAttribute("booksDataBase");
    Map<UUID, Book> booksDataBase = booksDAO.getBooksDataBase();
    request.setAttribute("books", booksDataBase);
%>


<c:choose>
    <c:when test="${user != null}">
        <a href="account.jsp" class="links">Перейти на сторінку користувача</a>
    </c:when>
    <c:otherwise>
        <a href="login.jsp" class="links">Логін</a>
    </c:otherwise>
</c:choose>
```

```
<form action="searchBooks" method="get">
    <label for="keyword">Ключове слово:</label>
    <input type="text" id="keyword" name="keyword" required>
    <button class="buttons" type="submit">Пошук</button>
</form>

<h1>Каталог книг</h1>
<table border="1">
    <c:choose>
        <c:when test="${empty books}">
            <h2>Каталог поки пустий...</h2>
        </c:when>
        <c:otherwise>
            <tr>
                <th>Назва книги</th>
                <th>Автор</th>
                <th>Опис</th>
                <th>Дії</th>
            </tr>
            <c:forEach var="book" items="${books}">
                <tr>
                    <td>${book.value.name}</td>
                    <td>${book.value.author.name}</td>
                    <td>${book.value.description}</td>
                    <td>
                        <form action="viewAuthor.jsp" method="post">
                            <input type="hidden" name="authorId"
value="${book.value.author.id}"/>
                            <button class="buttons" type="submit">Деталі про
автора</button>
                        </form>
                        <form action="addToFavorites" method="post">
                            <input type="hidden" name="bookId" value="${book.key}"/>
                            <button class="buttons" type="submit">Додати до
обраних</button>
                        </form>
```

```
                        <c:if test="${user.role == 'ADMIN'}">
                            <form action="editBook.jsp" method="get">
                                <input type="hidden" name="bookId" value="${book.key}"/>
                                <button class="buttons" type="submit">Редагувати
інформацію про книгу</button>
                            </form>
                        </c:if>
                    </td>
                </tr>
            </c:forEach>
        </c:otherwise>
    </c:choose>
</table>
</body>
</html>
```

createAuthor.jsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
 <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
 <title>Створення автора</title>
</head>
<body>
<h1>Створення автора</h1>
<form action="createAuthor" method="post">
 <label for="name">Ім'я автора:</label>
 <input type="text" id="name" name="name" required><br><br>

 <label for="biography">Біографія:</label>
 <textarea id="biography" name="biography" required></textarea><br><br>

 <label for="country">Країна:</label>
 <input type="text" id="country" name="country" required><br><br>

 <label for="age">Вік:</label>
```

```
 <input type="number" id="age" name="age" required><br><br>

 <button class="buttons" type="submit">Створити автора</button>
</form>
</body>
</html>
```

createBook.jsp

```
<%@ page import="com.coursework1.DAOs.BooksDAO" %>
<%@ page import="com.coursework1.Models.Author" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
 <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
 <title>Створення книги</title>
</head>
<body>
<h1>Створення книги</h1>
<form action="createBook" method="post">
 <label for="name">Назва книги:</label>
 <input type="text" id="name" name="name" required><br><br>

 <label for="authorName">Ім'я автора:</label>
 <input type="text" id="authorName" name="authorName" required><br><br>

 <label for="description">Опис книги:</label>
 <textarea id="description" name="description" required></textarea><br><br>

 <button class="buttons" type="submit">Створити книгу</button>
</form>
</body>
</html>
```

editAuthor.jsp

```
<%@ page import="com.coursework1.DAOs.AuthorsDAO" %>
<%@ page import="com.coursework1.Models.Author" %>
<%@ page import="java.util.UUID" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
    <title>Редагування автора</title>
    <style>
        textarea{
            width: 30%;
            height: 100px;
            background-color: #ecdde9;
        }
    </style>
</head>
<body>
<%
    UUID authorId = UUID.fromString(request.getParameter("authorId"));
    AuthorsDAO authorsDAO = (AuthorsDAO)
request.getServletContext().getAttribute("authorsDataBase");
    Author author = authorsDAO.getAuthorById(authorId);

    if (author == null) {
        response.sendError(500, "Автор не знайдений");
        return;
    }

    request.setAttribute("author", author);
%>

<h1>Редагування автора: ${author.name}</h1>

<form action="updateAuthor" method="post">
    <input type="hidden" name="authorId" value="${author.id}"/>
```

```
    <label for="name">Ім'я:</label>
    <input type="text" id="name" name="name" value="${author.name}" required/>
    <br/>
    <label for="biography">Біографія:</label>
    <textarea id="biography" name="biography" required>${author.biography}</textarea>
    <br/>
    <label for="country">Країна:</label>
    <input type="text" id="country" name="country" value="${author.country}" required/>
    <br/>
    <label for="age">Вік:</label>
    <input type="number" id="age" name="age" value="${author.age}" required/>
    <br/>
    <button class="buttons" type="submit">Оновити</button>
</form>

<form action="deleteAuthor" method="post">
    <input type="hidden" name="authorId" value="${author.id}"/>
    <button class="buttons" type="submit">Видалити автора</button>
</form>

<a class="links" href="booksView.jsp">Назад до каталогу</a>
<br>
<a class="links" href="account.jsp">Назад до акаунту</a>

</body>
</html>
```

editBook.jsp

```
<%@ page import="com.coursework1.Models.Book" %>
<%@ page import="java.util.UUID" %>
<%@ page import="com.coursework1.DAOs.BooksDAO" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
```

```
    <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
    <title>Редагування книги</title>
</head>
<style>
    textarea{
        width: 30%;
        height: 100px;
        background-color: #ecdde9;
    }
</style>
<body>
<%
    String id = request.getParameter("bookId");
    UUID bookId = UUID.fromString(id);
    BooksDAO booksDataBase = (BooksDAO)
request.getServletContext().getAttribute("booksDataBase");
    Book book = booksDataBase.getBookById(bookId);

    if (book == null) {
        response.sendError(500, "Книгу не знайдено");
        return;
    }

    request.setAttribute("book", book);
%>

<h1>Редагування книги: ${book.name}</h1>

<form action="updateBook" method="post">
    <input type="hidden" name="bookId" value="${book.bookId}"/>
    <label for="name">Назва:</label>
    <input type="text" id="name" name="name" value="${book.name}" required/>
    <br/>
    <label for="author">Автор:</label>
    <input type="text" id="author" name="author" value="${book.author.name}" required/>
    <br/>
```

```
    <label for="description">Опис:</label>
    <textarea id="description" name="description"
required>${book.description}</textarea>
    <br/>
    <button class="buttons" type="submit">Оновити</button>
</form>

<form action="deleteBook" method="post">
    <input type="hidden" name="bookId" value="${book.bookId}"/>
    <button class="buttons" type="submit">Видалити книгу</button>
</form>

<a class="links" href="account.jsp">Назад до акаунту</a>

</body>
</html>
```

editGuest.jsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
 <title>Редагування акаунту</title>
</head>
<body>
<div class="container">
 <h1>Редагування акаунту</h1>

 <form action="changeLogin" method="post">
    <h3>Редагування логіна</h3>
    <label for="newUsername">Новий логін:</label>
    <input type="text" id="newUsername" name="newUsername" required>
    <span class="message">${usernameMessage}</span>
    <span class="error">${usernameError}</span>
```

```
    <button class="buttons" type="submit">Оновити логін</button>
  </form>

  <form action="changePassword" method="post">
    <h3>Редагування пароля</h3>
    <label for="newPassword">Новий пароль:</label>
    <input type="text" id="newPassword" name="newPassword" required>
    <span class="message">${passwordMessage}</span>
    <span class="error">${passwordError}</span>
    <button class="buttons" type="submit">Оновити пароль</button>
  </form>

  <h3>Дії з акаунтом</h3>
  <form action="exitGuest" method="get">
    <button class="buttons" type="submit">Вийти з акаунту</button>
  </form>
  <br>
  <form action="deleteGuest" method="get">
    <button class="buttons" type="submit">Видалити акаунт</button>
  </form>

  <a href="account.jsp" class="links">Повернутися на сторінку користувача</a>

</div>
</body>
</html>
```

login.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="css/styles.css"/>
```

```
    <title>Вхід до акаунту</title>
</head>
<body>
<div class="container">
    <h2>Вхід до бібліотечного акаунту</h2>
    <form action="login" method="post">
        <div class="form-group">
            <label for="username">Логін:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div class="form-group">
            <label for="password">Пароль:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <span class="error">${errorMessage}</span><br>
        <div class="buttons">
            <button type="submit" class="login-btn"  id="login">Вхід</button>
            <button type="button" class="register-btn"
onclick="window.location.href='registration.jsp'">Реєстрація</button>
        </div>
    </form>
    <a href="booksView.jsp" class="back-link">Повернутися до каталогу</a>
</div>
</body>
</html>
```

registration.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="css/styles.css"/>
    <title>Реєстрація</title>
</head>
<body>
<div class="container">
```

```
    <h2>Реєстрація акаунту</h2>
    <form action="registration" method="post">
        <div class="form-group">
            <label for="username">Логін:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div class="form-group">
            <label for="password">Пароль:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <span class="error">${errorMessage}</span><br>
        <span class="message">${successMessage}</span><br>
        <div class="buttons_2type">
            <button type="submit" id="login">Зареєструватися</button>
        </div>
    </form>
    <a href="login.jsp" class="back-link">Вхід</a>
    <a href="booksView.jsp" class="back-link">Повернутися до каталогу</a>
</div>
</body>
</html>
```

searchResults.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
    <title>Результати пошуку</title>
    <style>
        table {
            width: 100%;
            border-collapse: collapse;
        }
        table th, table td {
```

```
            padding: 12px;
            border: 1px solid #000;
            text-align: left;
        }
</style>
</head>
<body>
<h1>Результати пошуку</h1>
<c:choose>
    <c:when test="${empty searchResults}">
        <h2>Жодна книга не знайдена</h2>
    </c:when>
    <c:otherwise>
        <table border="1">
            <tr>
                <th>Назва книги</th>
                <th>Автор</th>
                <th>Опис</th>
            </tr>
            <c:forEach var="book" items="${searchResults}">
                <tr>
                    <td>${book.name}</td>
                    <td>${book.author.name}</td>
                    <td>${book.description}</td>
                </tr>
            </c:forEach>
        </table>
    </c:otherwise>
</c:choose>

<a href="booksView.jsp" class="links">Повернутися до каталогу</a>
</body>
</html>
```

viewAuthor.jsp

```
<%@ page import="java.util.UUID" %>
<%@ page import="com.coursework1.Models.Author" %>
<%@ page import="com.coursework1.DAOs.AuthorsDAO" %>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="css/stylesMain.css"/>
    <title>Деталі про автора</title>
</head>
<body>
<%
    String authorIdStr = request.getParameter("authorId");
    UUID authorId = UUID.fromString(authorIdStr);
    AuthorsDAO authorsDataBase = (AuthorsDAO)
request.getServletContext().getAttribute("authorsDataBase");
    Author author = authorsDataBase.getAuthorById(authorId);
    if (author == null) {
        response.sendError(500, "Автор відсутній");
        return;
    }

    request.setAttribute("author", author);
%>
<h1>Деталі про автора</h1>
<c:if test="${not empty author}">
    <p><strong>Ім'я:</strong> ${author.name}</p>
    <p><strong>Біографія:</strong> ${author.biography}</p>
    <p><strong>Країна:</strong> ${author.country}</p>
    <p><strong>Вік:</strong> ${author.age}</p>
    <h2>Книги, написані автором:</h2>
    <ul>
        <c:forEach var="book" items="${author.writtenBooks}">
            <li>${book.name}</li>
        </c:forEach>
    </ul>
```

```
</c:if>
<c:if test="${empty author}">
    <p>Автор не знайдений</p>
</c:if>
<c:if test="${user.role == 'ADMIN'}">
    <form action="editAuthor.jsp" method="get">
        <input type="hidden" name="authorId" value="${author.id}"/>
        <button class="buttons" type="submit">Редагувати інформацію про автора</button>
    </form>
</c:if>
<form action="addToFavoriteAuthors" method="post">
    <input type="hidden" name="authorId" value="${author.id}"/>
    <button class="buttons" type="submit">Додати автора до обраних</button>
</form>
</body>
</html>
```