**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: «Електронний каталог товарів»**

Виконав:

студент 1 курсу, групи ІА-32

Варивода Кирил Сергійович \_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник:

асистент кафедри ІСТ

Степанов Андрій Сергійович \_\_\_\_\_\_\_\_\_\_

(підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань. Студент Варивода Кирил Сергійович

(підпис)

Київ – 2024 рок

ЗМІСТ

[ВСТУП 3](#_Toc158383261)

[1 ВИМОГИ ДО СИСТЕМИ 5](#_Toc158383262)

[1.1 Функціональні вимоги до системи 5](#_Toc158383263)

[1.2 Нефункціональні вимоги до системи 5](#_Toc158383264)

[2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ 6](#_Toc158383265)

[2.1 Діаграма прецедентів 7](#_Toc158383266)

[2.2 Опис сценаріїв використання системи 8](#_Toc158383267)

[3 АРХІТЕКТУРА СИСТЕМИ 23](#_Toc158383268)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 25](#_Toc158383269)

[4.1 Загальна структура проекту 26](#_Toc158383270)

[4.2 Компоненти рівня доступу до даних 27](#_Toc158383271)

[4.3 Компоненти рівня бізнес-логіки 30](#_Toc158383272)

[4.4 Компоненти рівня інтерфейсу користувача 32](#_Toc158383273)

[ВИСНОВКИ 34](#_Toc158383274)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 36](#_Toc158383275)

[ДОДАТОК А Лістінг програми 37](#_Toc158383276)

# ВСТУП

В еру інформаційних технологій та розвитку технологій стає дедалі більша потреба у сучасних «гаджетах», які будуть корисні для дому. Наприклад, як уявити рутину сьогодення без пральної машини, ноутбуку, вай-фай роутеру, тощо. Вони все далі заполонили нашу буденність і тому стає потреба в місці, де це все можна придбати і тут в нагоді стають сайти з електронним каталогом товарів, які можна придбати не виходячи з дому. Сьогодні, сайти з електронним каталогом товарів є невід’ємною частиною майже кожної людини, яка тісно пов’язана з побутом. Вони допомагають вибирати нові товари для дому, зберігати їх, проглядати та вибирати поміж безлічі варіантів від різних компаній з різних куточків світу. Але задля того, щоб каталог товарів користувався популярністю у споживачів, він повинен мати інтуїтивно зрозумілий інтерфейс, достатній каталог товарів, сортування по категоріям задля легшого пошуку товарів та системи профілів, щоб у користувача була можливість зберігати товари, які йому припали до душі. Також система профілів дозволяє зберігати дані про сесію, такі як: перегляд товарів, пошукові запити та обрані категорії, що в свою чергу дозволяє робити «таргетовану» рекламу, яка в більшості випадків є ефективнішою, оскільки рекомендує користувачу товари з категорій, які він переглядав до цього.

Головна мета роботи є створення застосунку, який дозволить переглядати категорії та товари електронного каталогу, як гість, користувач та адміністратор. Додатково адміністратор може редагувати, видаляти та додавати товари та категорії. Головні вимоги до застосунку є:

* ефективність серверної частини застосунку, щоб користувачу не доводилося чекати на відповідь від «бек енду»;
* додані адміністратором товари мають зберігатися у незалежному від користувача сховищі. Як «незалежне сховище» в даній роботі було використано базу даних яка базується на SQL, в моєму випадку – PostgreSQL;
* інтерфейс повинен мати функціонал для редагування, видалення та створення нових товарів та категорій лише адміністратором.

# 1 ВИМОГИ ДО СИСТЕМИ

## 1.1 Функціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

– незареєстрований користувач повинен мати можливість переглядати товари в каталозі;

– незареєстрований користувач повинен мати можливість здійснювати сортування товарів по категоріям;

– незареєстрований користувач повинен мати можливість зареєструватися;

– незареєстрований користувач повинен мати можливість здійснити вхід в існуючий акаунт;

– зареєстрований користувач без прав адміністратора повинен мати усі можливості, що є у незареєстрованого користувача, а також він повинен мати можливість переглядати свій профіль, здійснювати вихід з профілю, додавати та видаляти товари з «бажаного»;

– зареєстрований користувач з правами адміністратора повинен мати усі можливості, що є у зареєстрованого користувача без прав адміністратора, а також він повинен мати можливість додавати, редагувати та видаляти категорії та товари.

## 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 та адаптуватися під пристрій.

# 2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

## 2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами є користувачі системи: незареєстрований (гість), зареєстрований без прав адміністратора (користувач) та зареєстрований з правами адміністратора (адміністратор).

Зареєстрованому користувачу без прав адміністратора доступна уся функціональність, що і незареєстрованому, а також можливість вийти з облікового запису, переглянути профіль, додати товар до «збереженого», переглянути список «збережених» товарів та видалення товарів з цього списку, а зареєстрованому користувачу з правами адміністратора додається можливість редагувати та видаляти категорії та товари. 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 – 2.13.

Таблиця 2.1 – Сценарій використання «Створення облікового запису»

|  |  |
| --- | --- |
| Назва | Створення облікового запису |
| ID | 1 |
| Опис | Користувач, використовуючи два поля для вводу, вписує унікальний логін та довільний пароль |
| Актори | Гість |
| Вигоди компанії | Якщо гості зможуть зареєструватися на сайті, то у компанії, яка тримає цей сайт, буде можливість обробляти дані користувача та його сесії |
| Частота користування | При першому відвідуванні сайту |
| Тригери | Користувач вводить логін та пароль і натискає кнопку «Register» |
| Передумови | Поле реєстрації доступне у будь–якому вікні |
| Постумови | Користувач потрапляє на вікно каталогу маючи профіль |
| Основний розвиток | Користувач вводить логін і пароль у поле, натискає на кнопку «Register» |
| Альтернативні розвитки | – |
| Виняткові ситуації | У базі даних існує користувач з заданим логіном |

В таблиці 2.2 представлений сценарій використання «Вхід в обліковий запис»

Таблиця 2.2 – Сценарій використання «Вхід в обліковий запис»

|  |  |
| --- | --- |
| Назва | Вхід в обліковий запис |
| ID | 2 |
| Опис | Користувач, використовуючи два поля для вводу, вписує свій логін і пароль |
| Актори | Гість |
| Вигоди компанії | Якщо є можливість зареєструвати користувача та зберігати його акаунт десь у базі даних, то необхідно надати можливість здійснити вхід в існуючий акаунт |
| Частота користування | При повторних відвідуваннях сайту |
| Тригери | Користувач вводить логін та пароль і натискає кнопку «Log in» |
| Передумови | Поле реєстрації доступне у будь–якому вікні |
| Постумови | Користувач потрапляє на вікно каталогу маючи профіль |
| Основний розвиток | Користувач вводить логін і пароль у поле, натискає на кнопку «Log in» |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | * Не існує користувача з заданим логіном * Невірний пароль для заданого логіну |

В таблиці 2.3 представлений сценарій використання «Вихід з облікового запису»

Таблиця 2.3 – Сценарій використання «Вихід з облікового запису»

|  |  |
| --- | --- |
| Назва | Вихід з облікового запису |
| ID | 3 |
| Опис | Користувач, використовуючи кнопку «Log out» здійснює вихід з облікового запису |
| Актори | Користувач, Адміністратор |
| Вигоди компанії | Користувач має можливість вийти зі свого облікового запису, наприклад, для збереження особистих даних, якщо він здійснив вхід в особистий акаунт не на своєму пристрої |
| Частота користування | Відносно часто |
| Тригери | Користувач натискає кнопку «Log out» |
| Передумови | Користувач ввійшов у свій профіль та знаходиться на будь-якій сторінці сайту |
| Постумови | Користувач потрапляє на вікно каталогу без профілю |
| Основний розвиток | Користувач натискає на кнопку «Log out» |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | \_ |

В таблиці 2.4 представлений сценарій використання «Перегляд головної сторінки каталогу»

Таблиця 2.4 – Сценарій використання «Перегляд головної сторінки каталогу»

|  |  |
| --- | --- |
| Назва | Перегляд головної сторінки каталогу |
| ID | 4 |
| Опис | Користувач переглядає головну сторінку каталогу, де показані товари та категорії |
| Актори | Гість, Користувач, Адміністратор |
| Вигоди компанії | Основна інформація та пропозиції компанії знаходяться на головній сторінці, тому важливо зацікавити клієнта з перший секунд на сайті |
| 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 | * Якщо користувач не зареєстрований він не буде мати можливості додати товар до «збережених» * Якщо користувач не має прав адміністратора, він не буде мати можливість редагувати, додавати та видаляти товари та категорії |
| Виняткові ситуації | – |

В таблиці 2.5 представлений сценарій використання «Перехід на сторінку товару»

Таблиця 2.5 – Сценарій використання «Перехід на сторінку товару»

|  |  |
| --- | --- |
| Назва | Перехід на сторінку товару |
| ID | 5 |
| Опис | Користувач зможе докладно побачити інформацію про товар - його назву, ціну, опис та категорію, а також додати у «збережене» прямо на сторінці товару |
| Актори | Гість, Користувач, Адміністратор |
| Вигоди компанії | Якщо користувача зацікавив якийсь товар, то він захоче дізнатися про нього більше інформації та, можливо, придбати його |
| Частота користування | Постійно |
| Тригери | Користувач натискає на товар в каталозі |
| Передумови | Користувач знаходиться на головній сторінці |
| Постумови | Користувач потрапляє на сторінку із пошуковими видачами |
| Основний розвиток | Користувач натискає товар та отримує інформацію про нього. Можливість додати до збережених. Можливість перейти на головну сторінку |
| Альтернативні розвитки | * Якщо у користувача немає акаунту у нього не буде можливості додати товар до «збережених» * Якщо адміністратор натиснув кнопку «Edit», то його перекине на сторінку товару з додатковими полями вводу для зміни даних товару |
| Виняткові ситуації | * Дані, введені адміністратором при зміні товару, не є валідними |

В таблиці 2.6 представлений сценарій використання «Редагування товарів та категорій»

Таблиця 2.6 – Сценарій використання «Редагування товарів та категорій»

|  |  |
| --- | --- |
| Назва | Редагування товарів та категорій |
| ID | 6 |
| Опис | Редагування товарів та категорій |
| Актори | Адміністратор |
| Вигоди компанії | Редагування товарів та категорій є чудовим засобом для адміністрування сайту, бо не стоїть потреба видаляти та створювати нові товари та категорії при зміні актуальності даних |
| Частота користування | Рідко |
| Тригери | Адміністратор натиснув на кнопку «Edit» навпроти товару |
| Передумови | Адміністратора перекидує на сторінку товару, де у нього з’являються нові поля для вводу даних |
| Постумови | Товар чи категорія відредаговані |
| Основний розвиток | Адміністратор вводить дані для редагування та натискає на кнопку «Edit» |
| Альтернативні розвитки | – |
| Виняткові ситуації | - Дані, введені адміністратором при зміні товару чи категорії, не є валідними |

В таблиці 2.7 представлений сценарій використання «Видалення товарів чи категорій»

Таблиця 2.7 – Сценарій використання «Видалення товарів чи категорій»

|  |  |
| --- | --- |
| Назва | Видалення товарів чи категорій |
| ID | 7 |
| Опис | Видалення товарів чи категорій |
| Актори | Адміністратор |
| Вигоди компанії | Для легкого адміністрування адміністратор повинен мати можливість видаляти товари та категорії, щоб не робити постійні запити до бази даних для цього |
| Частота користування | Рідко |
| Тригери | Адміністратор натискає на кнопку «Delete» навпроти товару |
| Передумови | Адміністратор знаходиться на головній сторінці |
| Постумови | Товар чи категорія видалені |
| Основний розвиток | Адміністратор натискає на відповідну кнопку біля товару |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | \_ |

В таблиці 2.8 представлений сценарій використання «Створення товарів та категорій»

Таблиця 2.8 – Сценарій використання «Створення товарів та категорій»

|  |  |
| --- | --- |
| Назва | Створення товарів та категорій |
| ID | 8 |
| Опис | Адміністратор створює товар чи категорію |
| Актори | Адміністратор |
| Вигоди компанії | Функціонал для адміністрування є дуже корисним, наприклад створення категорій та товарів не потребує конкретної взаємодії з базою даних, якщо воно створено на сайті |
| Частота користування | Рідко |
| Тригери | Адміністратор натискає кнопку «Create» |
| Передумови | Адміністратор знаходиться на головній сторінці |
| Постумови | Адміністратора перекидує на сторінку з полями, де він може створити категорію або товар |
| Основний розвиток | Адміністратор створює категорію або товар |
| Альтернативні розвитки | – |
| Виняткові ситуації | * Адміністратор ввів невалідні дані |

В таблиці 2.9 представлений сценарій використання «Перегляд профілю користувача»

Таблиця 2.9 – Сценарій використання «Перегляд профілю користувача»

|  |  |
| --- | --- |
| Назва | Перегляд профілю користувача |
| ID | 9 |
| Опис | Перегляд профілю користувача |
| Актори | Користувач, Адміністратор |
| Вигоди компанії | Користувач повинен мати можливість переглядати дані про свій обліковий запис, також на цій сторінці користувач має можливість подивитися, чи є у нього права адміністратора |
| Частота користування | Відносно часто |
| Тригери | Користувач натискає на кнопку «Профіль» |
| Передумови | Користувач знаходиться на будь-якій сторінці сайту |
| Постумови | Користувача перекидує на сторінку з інформацією про профіль та його «збереженими» товарами |
| Основний розвиток | Користувач натискає на кнопку «Профіль» та отримує інформацію про свій профіль |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | \_ |

В таблиці 2.10 представлений сценарій використання «Зберегти товар»

Таблиця 2.10 – Сценарій використання «Зберегти товар»

|  |  |
| --- | --- |
| Назва | Зберегти товар |
| ID | 10 |
| Опис | Зберегти товар до списку збережених товарів |
| Актори | Користувач, Адміністратор |
| Вигоди компанії | Якщо користувачі зможуть додавати товари до збережених це набагато полегшить їх користування сайтом, а отже збільшить шанси на те, що користувач витратить гроші на цьому сайті |
| Частота користування | Часто |
| Тригери | Користувач натискає на кнопку «\*» біля товару |
| Передумови | Користувач знаходиться на головній сторінці |
| Постумови | Товар додається до списку «збережених». Кнопка «\*» біля товару зникає. Біля кнопки «Профіль» з’являється лічильник товарів, доданих до «збереженого» |
| Основний розвиток | Користувач натискає меню поряд із строкою програвання, відкривається меню із слайдерами налаштування, користувач може їх змінювати |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | \_ |

В таблиці 2.11 представлений сценарій використання «Перегляд «збережених» товарів»

Таблиця 2.11 – Сценарій використання «Перегляд «збережених» товарів»

|  |  |
| --- | --- |
| Назва | Перегляд «збережених» товарів |
| ID | 11 |
| Опис | Користувач переглядає «збережені» товари |
| Актори | Користувач, Адміністратор |
| Вигоди компанії | Якщо є можливість додати товар до «збережених», то має бути можливість переглядати їх |
| Частота користування | Часто |
| Тригери | Користувач натискає на кнопку «Профіль» коли біля неї є лічильник |
| Передумови | Користувач знаходиться на будь-якій сторінці сайту |
| Постумови | Користувача перекидує на сторінку профілю, де видно товари, які він додав до «збережених» |
| Основний розвиток | Користувач натискає на кнопку «Профіль» та отримує список «збережених» товарів |
| Альтернативні розвитки | – |
| Виняткові ситуації | * Користувач не додав жодного товару до «збережених» |

В таблиці 2.12 представлений сценарій використання «Сортування по категоріям»

Таблиця 2.12 – Сценарій використання «Сортування по категоріям»

|  |  |
| --- | --- |
| Назва | Сортування по категоріям |
| ID | 12 |
| Опис | Сортування товарів по категоріям |
| Актори | Гість, Користувач, Адміністратор |
| Вигоди компанії | Якщо користувачі зможуть швидко та зручно сортувати товари по своїм потребам, то це позитивно відобразиться на продажах сайту |
| Частота користування | Часто |
| Тригери | Користувач натискає на категорію, яка його цікавить |
| Передумови | Користувач знаходиться на головній сторінці |
| Постумови | Каталог товарів сортується відповідно заданої категорії |
| Основний розвиток | Користувач натискає на потрібну категорію та сортує товари |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | \_ |

В таблиці 2.13 представлений сценарій використання «Перехід на головну сторінку»

Таблиця 2.13 – Сценарій використання «Перехід на головну сторінку»

|  |  |
| --- | --- |
| Назва | Перехід на головну сторінку |
| ID | 13 |
| Опис | Користувач може перейти на головну сторінку |
| Актори | Гість, Користувач, Адміністратор |
| Вигоди компанії | Користувач повинен мати швидкий доступ до головної сторінки в разі, якщо він захоче продовжити вибирати товари |
| Частота користування | Часто |
| Тригери | Користувач натискає на кнопку «До каталогу» на будь-якій сторінці сайту |
| Передумови | Користувач знаходиться на будь-якій сторінці сайту |
| Постумови | Користувача перекидує на головну сторінку |
| Основний розвиток | Користувач натискає на кнопку «До каталогу» та повертається до перегляду товарів |
| Альтернативні розвитки | \_ |
| Виняткові ситуації | \_ |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1.

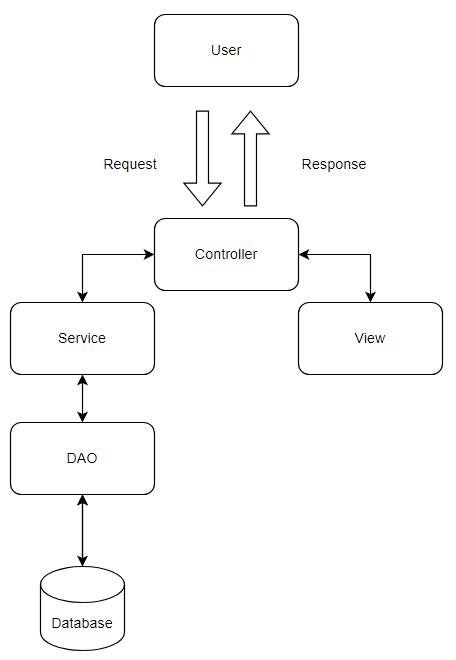


Рисунок 3.1 – Загальна архітектура системи

Система складається з наступних елементів:

– графічний інтерфейс;

– серверна частина;

– база даних.

Для взаємодії з користувачем зазвичай використовується графічний інтерфейс. HTTP запит надходить до серверної частини, яка складається з контролеру, сервісу та DAO, оброблюється та повертається до користувача через графічний інтерфейс.Основна логіка системи виконується в серверній частині. Дані, які були отриманні з графічного інтерфейсу приходять до контролеру, де валідуються та конвертуються. Оброблені дані з контролеру передаються до сервісу, де викликається запит до DAO, який отримує, записує, редагує чи вадаляє дані з бази даних. База даних потрібна для зберігання даних, які були створені на серверній стороні застосунку та повертає їх у разі запиту від сервісу.

До серверної частини належать наступні елементи:

– контролер;

– вигляд;

– сервіс;

– DAO.

Контролер отримує дані, які користувач вніс в графічному інтерфейсі. З контролеру дані обробляються та дається запит на сервіс, звідки формується запит до DAO. З DAO до бази даних надходять передані дані та зберігаються. В кінці все повертається до контролеру, де зі всіма готовими даними створюються потрібні об’єкти з якими йде запит на графічну частину застосунку для відображення користувачу.

# 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

Загальна реалізація всіх компонентів системи зображена на рис. 4.0

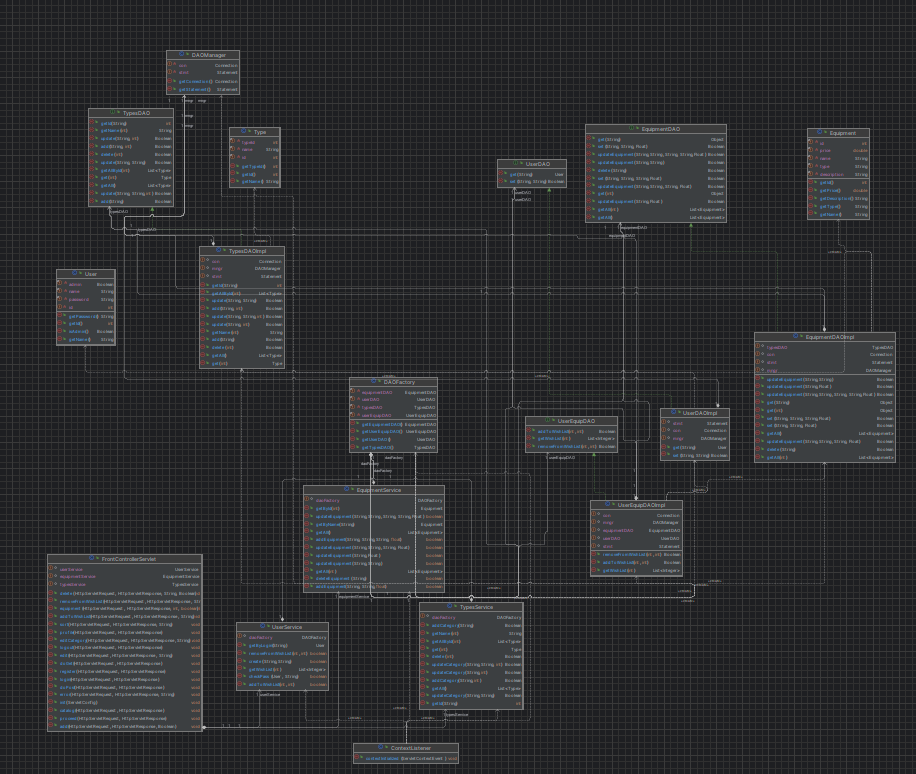


Рис 4.0 - Загальна реалізація всіх компонентів системи

## 4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

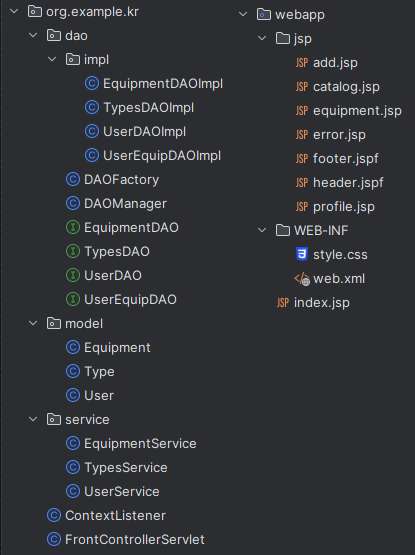


Рисунок 4.1 – Загальна структура проекту

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. 4.2.1

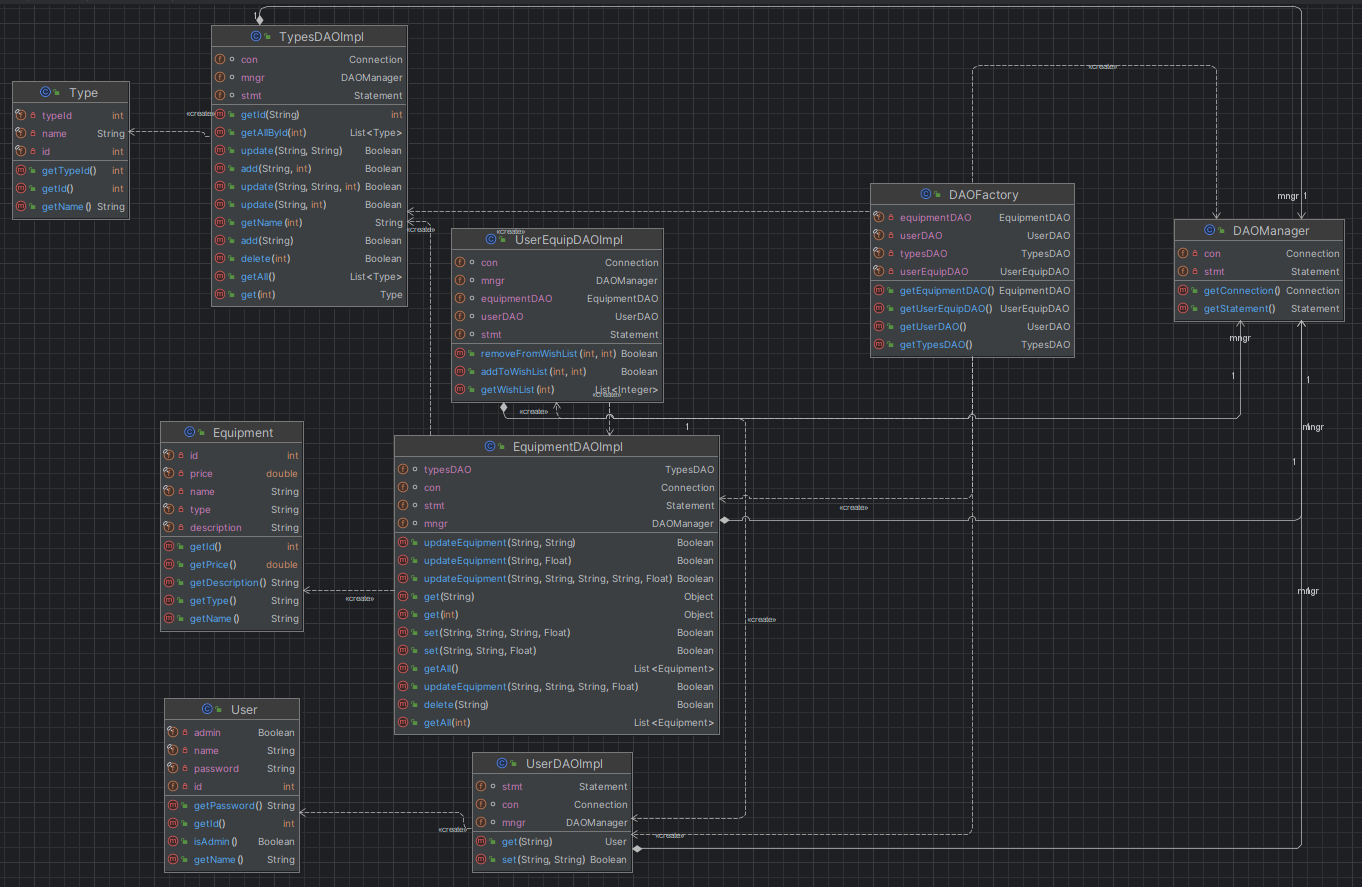


Рисунок 4.2.1 – Основні сутності та інтерфейси рівня доступу до даних

В нашому проекті є 3 головні сутності, які можна виділити – Користувач (User), Товар(Equipment) та Категорія(Type). Кожна сутність має в собі інформацію та методи, потрібні для роботи з сутностями. Користувач має такі поля як: логін(name), пароль(password), булеве значення, яке задає роль адміністратора(admin) та унікальний ідентифікатор(id). Товар має спільні з Користувачем поля name та id та особисті, такі як: ціна(price), категорія якій належить товар(type) та опис(description). Та остання сутність Категорія, яка не має унікальних полей, а має ті, які вже згадувалися: унікальний ідентифікатор(id), назва категорії(name) та категорія якій належить дана категорія(type). Для кожної з сутностей ми маємо особистий DAO, який зв’язується з базою даних та отримує, записує, редагує та видаляє дані пов’язані з потрібною сутністю та формує сутності в разі потреби. Також ми маємо UserEquipDAO – це клас, який працює з many-to-many таблицею в базі даних, яку ми можемо побачити на рисунку 4.2.2 «loginequip», яка зберігає унікальний ідентифікатор користувача та товару. Ця таблиця потрібна для зберігання «побажань» користувача, які він додав в свій список бажаних. Головними елементами рівня доступу до даних є DAOManager та DAOFactory. DAOManager має лише 1 функцію – налаштовувати зв’язок з базою даних завдяки JDBC драйвера. DAOFactory генерує інші DAO для зручності використання, бо при потребі звернутися до якогось DAO можна створити 1 об’єкт DAOFactory та звертатися до будь яких DAO, які нам потрібні.

Всі таблиці та колонки в базі даних наведені на рис. 4.2.2

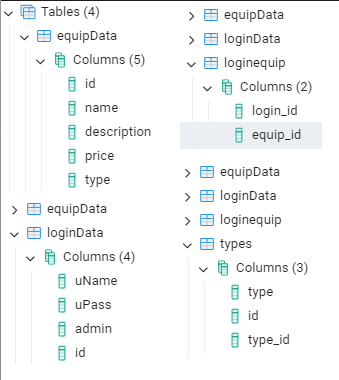


Рис 4.2.2 - Всі таблиці та колонки в базі даних

Загалом в базі даних ми маємо 4 таблиці 3 з яких зберігають інформацію сутностей проекту. equipData – таблиця для зберігання інформації про товар має 4 звичайних поля – ідентифікатор(id), назва товару(name), опис товару(description), ціна(price) та поле type, яке є foreign key типу one-to-many, бо одна й та сама категорія може бути у декількох товарів одночасно. loginData - таблиця для зберігання інформації про користувача має 4 звичайних поля – логін користувача(uName), пароль(uPass), чи є користувач адміністратором(admin) та ідентифікатор(id). loginequip – many-to-many таблиця, яка має 2 поля та 2 foreign key – login\_id та equip\_id, які є ідентифікатором користувача та ідентифікатором товару, відповідно. types - таблиця для зберігання інформації про категорію має 2 звичайних поля – назва категорії(type), ідентифікатор (id) та 1 foreign key, який є ідентифікатором категорії, якій належить ця категорія. Всі перелічені колонки в базі даних є notNull, тобто не можуть бути пусті, окрім колонки description в equipData та колонка type\_id в types. В кожній таблиці головним елементом є ідентифікатор(id), за його наявності, це значення за яким йде користувачі, товари чи категорії можуть використовуватися у many-to-many та one-to-many зв’язках.

## 4.3 Компоненти рівня бізнес-логіки

Основні компоненти рівня бізнес-логіки наведені на рис. 4.3

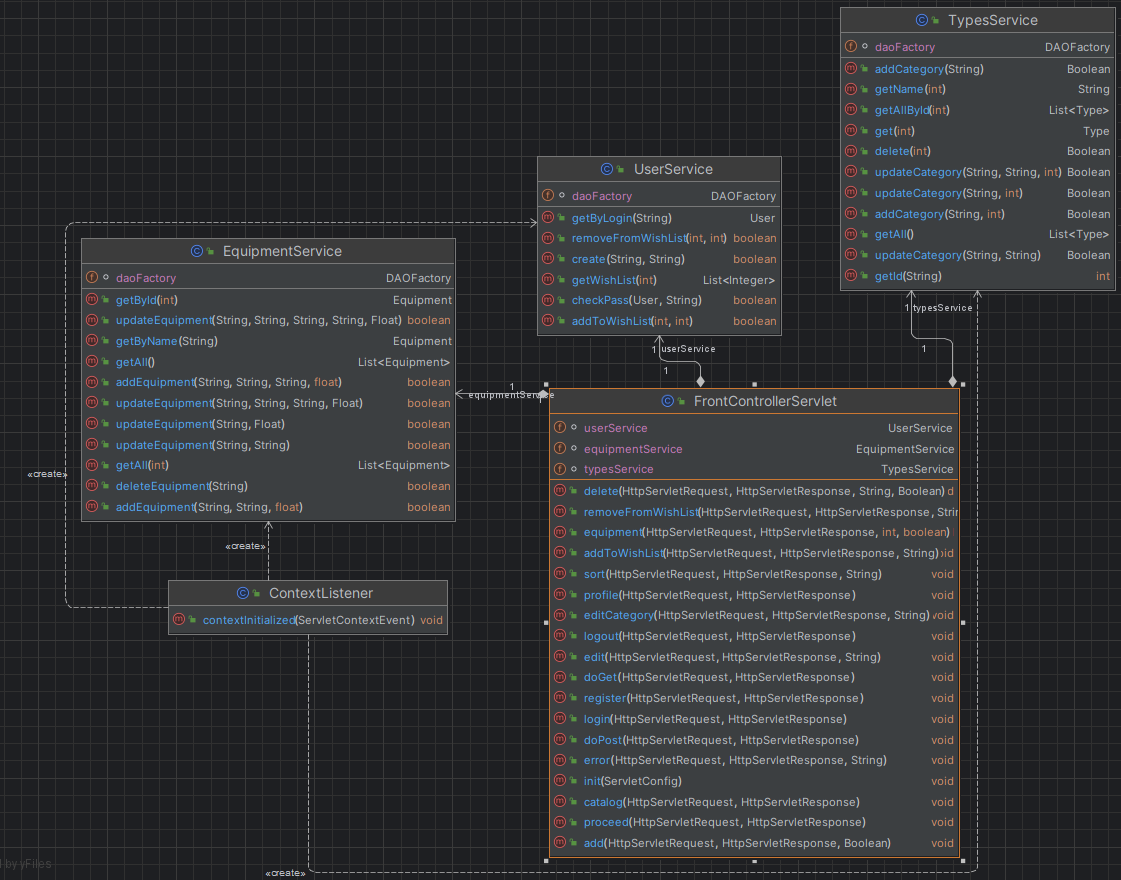


Рис 4.3 – Компоненти рівня бізнес-логіки

Основна бізнес-логіка зібрана у FrontControllerSevlet. Але перед тим як почати роботу FrontControllerSevlet отримує об’єкти сервісів та DAOFactory через контекст, який передає ContextListener. Всередині FrontControllerSevlet-у знаходяться ключові методи, завдяки яким view модель може взаємодіяти з даними, які лежать в базі даних. Головним механізмом у FrontControllerSevlet-і є конструкція switch-case, яка перевіряє шлях в url та при знаходженні такого шляху у case викликається потрібний метод, який отримує дані з реквесту або сесії, а при невідповідності url-у буде перекидувати на головну сторінку, яка в нашому випадку є каталогом. Всередині цих методів, зазвичай, робляться операції над переданими даними та відправляються в потрібні сервіси для подальшої обробки або передачі далі по ланцюжку до DAO. Назад з DAO значення отримуються до FrontControllerSevlet-у, де передаються до view моделі використовуючи реквест чи сесію. Три сервіси мають ті самі методи, що і DAO, для легкості звернень до них через основний сервлет. Всередині кожного методу проводиться або переробка даних, або одразу передача до DAO, де проводиться запит до бази даних з вже готовими даними.

## 4.4 Компоненти рівня інтерфейсу користувача

Основні компоненти серверної частини на рівні інтерфейсу користувача

наведені на рис. 4.4

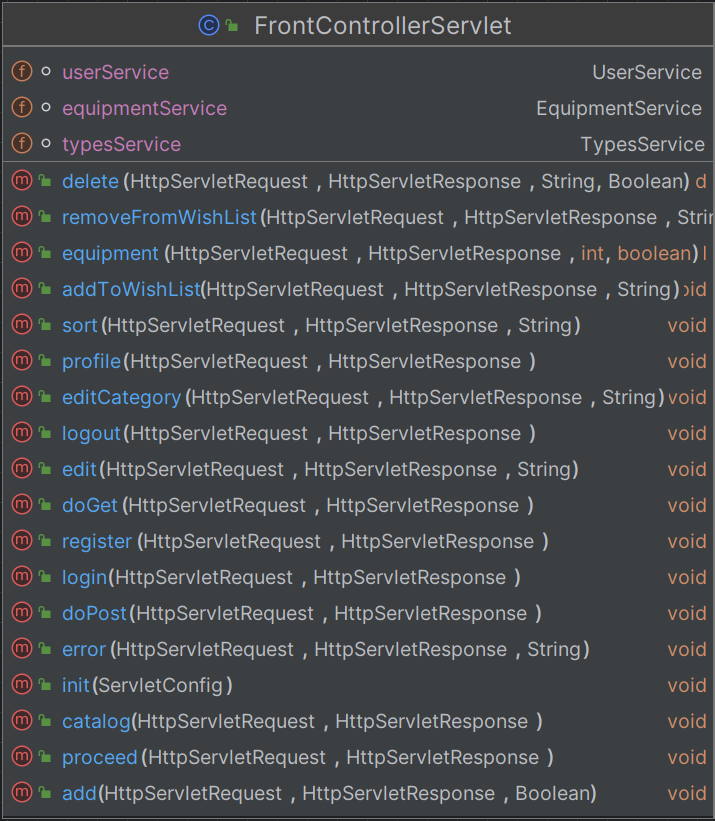


Рис 4.4 – Компоненти серверної частини на рівні інтерфейсу

Основні дії для отримання даних, які виведуться на інтерфейс відбувається у класі FrontConrollerServlet. Виділю всі методи, які відповідають саме за інтерфейс частину, а не пов’язані з даними та діями над ними. Метод equipment - отримує значення товару, у вигляді його ідентифікатору, вибраного користувачем, отримує дані про товар та відправляє на jsp equipment. Метод addToWishList прибирає значок «\*» біля товару, який користувач додав до «збережених», та додає його до профілю користувача. Метод sort передає до jsp catalog дані про те, яку категорію товарів треба показати, передача виконується завдяки збиранню потрібних товарів та категорій до списку і передачею на jsp catalog. Метод profile віддає на jsp profile дані про користувача, який є зареєстрованим на сайті на даний момент. Метод error отримує текст помилки та передає цей текст на jsp error при помилках в застосунку. Інші методи є допоміжними, які роблять дії над даними та ніяк не взаємодіють з візуальним інтерфейсом напряму, а тільки отримують дані з нього. Всі вище перелічені jsp можна побачити на рис. 4.1

# ВИСНОВКИ

Перед початком виконання роботи було проаналізовані характеристики та ключові ознаки хорошого інтернет магазину для планування та написання свого власного проекту. Був розроблений план та продумані основні моменти для подальшої розробки застосунку. Інтернет магазини є важливою частиною рутини людини в наші часи – це можна побачити по тому, яка кількість інтернет магазинів зараз є «на плаву», як мінімум, в Україні, тому розробка гарно спланованого інтернет магазину, який зможе дати користувачу зручний функціонал та великий вибір товарів, є запорукою вдалого проекту. З цих міркувань було вирішено розробити систему, яка зможе надавати користувачу можливість зручно переглядати товари та забезпечувати швидкодію сайту, щоб у користувача не було проблем з очікуванням операцій на сервері.

По-перше, були сформовані функціональні та нефункціональні вимоги до систем, що надає нам змогу передбачити поведінку нашої системи. Після ми обрали технології, на основі яких буде розроблена наша система. Було обрано Java, як мова програмування через її швидкодію та стабільність, що допоможе нам в розробці. В якості середи розробки була обрана - IntelliJ IDEA за функціонал та можливості, які надає ця IDE. В якості системи управління даними було обрано PostgreSQL через гнучкість та легкість у налаштовуванні. Для з’єднання між системою та базою даних була обрана стандартна JDBC бібліотека для PostgreSQL, бо це допоможе зберегти нативність та звільняє нас від додаткових прошарків системи. Для зв’язку між серверною частину та графічною було використано фреймворк JSTL за його зручність у передачі даних між графічною частиною та серверною. Як графічний інтерфейс було обрано JSP, бо вони можуть зв’язуватися з серверною частиною завдяки обраному фреймворку та CSS для стилізації.

По-друге, був розроблений сценарій використання системи. Система здатна на перехід на сторінку товару та перегляд інформації про нього, сортування товарів по категоріях, додавання товарів у «збережене» при наявності акаунту, реєстрація акаунту, вхід в особистий акаунт при його наявності, перехід на сторінку профілю для перегляду інформації про акаунт та збережені товари, редагування, створення та видалення як товарів, так і категорій при наявності прав адміністратора. Також доступні контактні дані розробника системи в нижньому колонтитулі.

По-третє, було розроблено основну структуру проекту. Завдяки шаблону проектування MVC за патерну DAO вдалося розбити проект на логічні рівні. Рівень Model відповідав за подачу отриманих з DAO, яка в свою чергу робила запити до бази даних. Controller робив запити до сутностей DAO та отримував дані з бази даних, які в подальшому обробляв та передавав на View. Рівень View відповідав за візуальну частину системи, передача даних потрібним чином на JSP файли та отримання та вивід цих даних у самих JSP файлах. Завдяки впровадженню шаблону MVC наш проект став задовольняти трьом принципам ООП, а саме: інкапсуляція, абстрагування та принцип єдиної відповідальності. Завдяки цьому стало набагато легше читати та розуміти код та структуру системи, що в майбутньому може допомогти в розширені системи. На стороні бази даних були продумані та створені таблиці, які зберігають потрібну нам інформацію. Були використані різні зв’язки між таблицями в базі даних, такі як: one-to-many та many-to-many, що допомогло зберігати дані більш ефективно.

Отже, дивлячись на сказане вище основними перевагами системи є можливість розширення та простоті роботи з нею, бо адміністратору не обов’язково мати доступ до бази даних, щоб вносити зміни у каталозі. Завдяки тому, що система має відкриту та розбиту на логічні рівні архітектуру не є проблемою подальше розширення. Графічний інтерфейс системи є легким та інтуїтивно зрозумілим для користувачів, що надає великої переваги системі. Система є конкурентоспроможною на наявному ринку інтернет магазинів в Україні та має потенціал до розвинення.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stack overflow [Електронний ресурс] – Режим доступу до ресурсу: <https://stackoverflow.com/>
2. Intellij IDEA documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/help/idea/getting-started.html>
3. W3Schools [Електронний ресурс] – Режим доступу до ресурсу: [https://www.w3schools.com/java/](https://www.w3schools.com/java/%20)
4. Програмування2-2023-2024-2 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.youtube.com/playlist?list=PLNM5bd0h-oz-mYjU0fez42je-zA5moKbR>
5. HTML documentation [Електронний ресурс] – Режим доступу до ресурсу: [https://developer.mozilla.org/en-US/docs/Web/HTML](https://developer.mozilla.org/en-US/docs/Web/HTML%20)
6. Java documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.oracle.com/en/java/>
7. PostgreSQL JDBC driver documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://jdbc.postgresql.org/documentation/>
8. PostgreSQL documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/>
9. Code Ranch forum [Електронний ресурс] – Режим доступу до ресурсу: <https://coderanch.com/forums/c/1/java>
10. Geeks for geeks [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/>

ДОДАТОК А  
Лістінг програми

<https://github.com/mavr1iq/VaryvodaKyrylKR>

EquipmentDAOImpl.java

package org.example.kr.dao.impl;  
  
import org.example.kr.dao.DAOManager;  
import org.example.kr.dao.EquipmentDAO;  
import org.example.kr.dao.TypesDAO;  
import org.example.kr.model.Equipment;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class EquipmentDAOImpl implements EquipmentDAO {  
 TypesDAO typesDAO;  
 DAOManager mngr;  
 Connection con;  
 Statement stmt;  
 public EquipmentDAOImpl(DAOManager mngr) {  
 this.mngr = mngr;  
 this.stmt = mngr.getStatement();  
 this.con = mngr.getConnection();  
 typesDAO = new TypesDAOImpl(mngr);  
 }  
  
 @Override  
 public Object get(String key){  
 String sql = "select \* from \"equipData\" where \"name\" = ? ";  
  
 try {  
 PreparedStatement pstmt = this.con.prepareStatement(sql);  
 pstmt.setString(1, key);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }  
 if (rs.next()) {  
 int id = rs.getInt("id");  
 String description = rs.getString("description");  
 int type\_id = rs.getInt("type\_id");  
 float price = rs.getFloat("price");  
  
 return new Equipment(id, key, description, typesDAO.getName(type\_id), price);  
 }  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public Object get(int id) {  
 String sql = "select \* from \"equipData\" where \"id\" = ? ";  
  
 try {  
 PreparedStatement pstmt = this.con.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }  
 if (rs.next()) {  
 String name = rs.getString("name");  
 String description = rs.getString("description");  
 int type\_id = rs.getInt("type\_id");  
 float price = rs.getFloat("price");  
  
 return new Equipment(id, name, description, typesDAO.getName(type\_id), price);  
 }  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public List<Equipment> getAll() {  
 String sql = "select \* from \"equipData\" order by id";  
 List<Equipment> list = new ArrayList<Equipment>();  
  
 try {  
 PreparedStatement pstmt = this.con.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String name = rs.getString("name");  
 String description = rs.getString("description");  
 int type\_id = rs.getInt("type\_id");  
 float price = rs.getFloat("price");  
  
 list.add(new Equipment(id, name, description, typesDAO.getName(type\_id), price));  
 }  
 return list;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public List<Equipment> getAll(int typeId) {  
 String sql = "select \* from \"equipData\" where \"type\_id\" = ? order by id";  
 List<Equipment> list = new ArrayList<Equipment>();  
  
 try {  
 PreparedStatement pstmt = this.con.prepareStatement(sql);  
 pstmt.setInt(1, typeId);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String name = rs.getString("name");  
 String description = rs.getString("description");  
 float price = rs.getFloat("price");  
  
 list.add(new Equipment(id, name, description, typesDAO.getName(typeId), price));  
 }  
 return list;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public Boolean set(String name, String description, String type, Float price) {  
 String sql = "insert into \"equipData\"(\"name\", \"description\", \"type\_id\", \"price\") values (?,?,?,?)";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setString(2, description);  
 pstmt.setInt(3, typesDAO.getId(type));  
 pstmt.setFloat(4, price);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean set(String name, String type, Float price) {  
 String sql = "insert into \"equipData\"(\"name\", \"type\_id\", \"price\") values (?,?,?)";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setInt(2, typesDAO.getId(type));  
 pstmt.setFloat(3, price);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean updateEquipment(String keyName, String name, String description, String type, Float price) {  
 String sql = "UPDATE \"equipData\" SET name=?, description=?, price=?, type\_id=? WHERE name=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setString(2, description);  
 pstmt.setFloat(3, price);  
 pstmt.setInt(4, typesDAO.getId(type));  
 pstmt.setString(5, keyName);  
 return pstmt.executeUpdate() > 0;  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean updateEquipment(String keyName, String name, String type, Float price) {  
 String sql = "UPDATE \"equipData\" SET name=?, price=?, type\_id=? WHERE name=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setFloat(2, price);  
 pstmt.setInt(3, typesDAO.getId(type));  
 pstmt.setString(4, keyName);  
 return pstmt.executeUpdate() > 0;  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean updateEquipment(String keyName, String type) {  
 String sql = "UPDATE \"equipData\" SET type\_id=? WHERE name=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, typesDAO.getId(type));  
 pstmt.setString(2, keyName);  
 return pstmt.executeUpdate() > 0;  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean updateEquipment(String keyName, Float price) {  
 String sql = "UPDATE \"equipData\" SET price=? WHERE name=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setFloat(1, price);  
 pstmt.setString(2, keyName);  
 return pstmt.executeUpdate() > 0;  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean delete(String name) {  
 String sql0 = "delete from \"equipData\" where name=?";  
 String sql1 = "delete from \"loginequip\" where equip\_id=?";  
 try {  
 PreparedStatement pstmt0 = con.prepareStatement(sql0);  
 pstmt0.setString(1, name);  
  
 PreparedStatement pstmt1 = con.prepareStatement(sql1);  
 pstmt1.setInt(1, ((Equipment)get(name)).getId());  
  
 pstmt1.executeUpdate();  
 return pstmt0.executeUpdate() > 0;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
}

TypesDAOImpl.java

package org.example.kr.dao.impl;  
  
import org.example.kr.dao.DAOManager;  
import org.example.kr.dao.TypesDAO;  
import org.example.kr.model.Type;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class TypesDAOImpl implements TypesDAO {  
 DAOManager mngr;  
 Connection con;  
 Statement stmt;  
 public TypesDAOImpl(DAOManager mngr) {  
 this.mngr = mngr;  
 this.stmt = mngr.getStatement();  
 this.con = mngr.getConnection();  
 }  
  
 @Override  
 public List<Type> getAll() {  
 String sql = "select \* from types where type\_id IS NULL ORDER BY id";  
 List<Type> list = new ArrayList<Type>();  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String name = rs.getString("type");  
 int typeId = rs.getInt("type\_id");  
 Type t = new Type(id, name, typeId);  
 list.add(t);  
 }  
 return list;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public List<Type> getAllById(int typeId) {  
 String sql = "select \* from types where type\_id = ? ORDER BY id";  
 List<Type> list = new ArrayList<Type>();  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, typeId);  
 ResultSet rs = pstmt.executeQuery();  
  
 if (rs == null) {  
 return null;  
 }  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String name = rs.getString("type");  
 Type t = new Type(id, name, typeId);  
 list.add(t);  
 }  
 return list;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public Type get(int id) {  
 String sql = "select \* from types where id = ?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 ResultSet rs = pstmt.executeQuery();  
  
 if (rs == null) {  
 return null;  
 }  
 if (rs.next()) {  
 int typeId = rs.getInt("type\_id");  
 String name = rs.getString("type");  
 return new Type(id, name, typeId);  
 }  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 return null;  
 }  
  
 @Override  
 public Boolean add(String name, int typeId) {  
 String sql = "insert into types(type, type\_id) values(?,?)";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setInt(2, typeId);  
 return pstmt.executeUpdate() > 0;  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean add(String name) {  
 String sql = "insert into types(type) values(?)";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 return pstmt.executeUpdate() > 0;  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean update(String key, String name, int typeId) {  
 String sql = "UPDATE types SET type=?, type\_id=? WHERE type=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setInt(2, typeId);  
 pstmt.setString(3, key);  
 return pstmt.executeUpdate() > 0;  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean update(String key, String name) {  
 String sql = "UPDATE types SET type=? WHERE type=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setString(2, key);  
 return pstmt.executeUpdate() > 0;  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean update(String key, int typeId) {  
 String sql = "UPDATE types SET type\_id=? WHERE type=?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, typeId);  
 pstmt.setString(2, key);  
 return pstmt.executeUpdate() > 0;  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public Boolean delete(int id) {  
 String sql = "delete from types where id = ?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 return pstmt.executeUpdate() > 0;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
  
 @Override  
 public int getId(String name) {  
 String sql = "select \* from types where type = ?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return -1;  
 }  
 if (rs.next()) {  
 return rs.getInt("id");  
 }  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return -1;  
 }  
  
 @Override  
 public String getName(int id) {  
 String sql = "select \* from types where id = ?";  
  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }  
 if (rs.next()) {  
 return rs.getString("type");  
 }  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
}

UserDAOImpl.java

package org.example.kr.dao.impl;  
  
import org.example.kr.dao.DAOManager;  
import org.example.kr.dao.UserDAO;  
import org.example.kr.model.User;  
  
import java.sql.\*;  
  
public class UserDAOImpl implements UserDAO {  
 DAOManager mngr;  
 Connection con;  
 Statement stmt;  
 public UserDAOImpl(DAOManager mngr) {  
 this.mngr = mngr;  
 this.stmt = mngr.getStatement();  
 this.con = mngr.getConnection();  
 }  
  
 @Override  
 public User get(String key) {  
 String sql = "select \* from \"loginData\" where \"uName\" = ?";  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, key);  
 ResultSet rs = pstmt.executeQuery();  
 if (rs == null) {  
 return null;  
 }else if (rs.next()) {  
 int id = rs.getInt("id");  
 String name = rs.getString("uName");  
 String password = rs.getString("uPass");  
 Boolean admin = rs.getBoolean("admin");  
 return new User(id, name, password, admin);  
 }  
  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 return null;  
 }  
 return null;  
 }  
  
 @Override  
 public Boolean set(String name, String pass) {  
 String sql = "insert into \"loginData\" (\"uName\", \"uPass\") values (?,?)";  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setString(1, name);  
 pstmt.setString(2, pass);  
 int count = pstmt.executeUpdate();  
 return count > 0;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return false;  
 }  
}

UserEquipDAOImpl.java

package org.example.kr.dao.impl;  
  
import org.example.kr.dao.\*;  
import org.example.kr.model.Equipment;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class UserEquipDAOImpl implements UserEquipDAO {  
 DAOManager mngr;  
 UserDAO userDAO;  
 EquipmentDAO equipmentDAO;  
 Connection con;  
 Statement stmt;  
 public UserEquipDAOImpl(DAOManager mngr) {  
 this.mngr = mngr;  
 this.stmt = mngr.getStatement();  
 this.con = mngr.getConnection();  
 userDAO = new UserDAOImpl(mngr);  
 equipmentDAO = new EquipmentDAOImpl(mngr);  
 }  
 @Override  
 public List<Integer> getWishList(int id) {  
 String sql = "select \* from loginequip where login\_id = ?";  
 List<Integer> list = new ArrayList<>();  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 ResultSet rs = pstmt.executeQuery();  
 while (rs.next()) {  
 String equipId = rs.getString("equip\_id");  
 list.add(Integer.*parseInt*(equipId));  
 }  
 return list;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public Boolean addToWishList(int equipID, int loginID) {  
 String sql = "INSERT INTO loginequip(login\_id, equip\_id) VALUES(?, ?)";  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, loginID);  
 pstmt.setInt(2, equipID);  
 return pstmt.executeUpdate() > 0;  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public Boolean removeFromWishList(int equipId, int userId) {  
 String sql = "DELETE FROM loginequip WHERE login\_id = ? AND equip\_id = ?";  
 try {  
 PreparedStatement pstmt = con.prepareStatement(sql);  
 pstmt.setInt(1, userId);  
 pstmt.setInt(2, equipId);  
 return pstmt.executeUpdate() > 0;  
 }catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
}

DAOFactory.java

package org.example.kr.dao;  
  
import org.example.kr.dao.impl.EquipmentDAOImpl;  
import org.example.kr.dao.impl.TypesDAOImpl;  
import org.example.kr.dao.impl.UserDAOImpl;  
import org.example.kr.dao.impl.UserEquipDAOImpl;  
  
public class DAOFactory {  
 private final UserDAO userDAO;  
 private final EquipmentDAO equipmentDAO;  
 private final UserEquipDAO userEquipDAO;  
 private final TypesDAO typesDAO;  
  
 public DAOFactory() {  
 DAOManager mngr = new DAOManager();  
 userDAO = new UserDAOImpl(mngr);  
 equipmentDAO = new EquipmentDAOImpl(mngr);  
 userEquipDAO = new UserEquipDAOImpl(mngr);  
 typesDAO = new TypesDAOImpl(mngr);  
 }  
  
 public UserDAO getUserDAO() { return userDAO; }  
 public EquipmentDAO getEquipmentDAO() { return equipmentDAO; }  
 public UserEquipDAO getUserEquipDAO() { return userEquipDAO; }  
 public TypesDAO getTypesDAO() { return typesDAO; }  
}

DAOManager.java

package org.example.kr.dao;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.Statement;  
  
public class DAOManager {  
 private Connection con;  
 private Statement stmt;  
  
 public DAOManager() {  
 try {  
 String url = "jdbc:postgresql://localhost/KR?user=postgres&password=kivar"; // Пароль, який ви робили при скачуванні postgres  
 Class.*forName*("org.postgresql.Driver");  
 System.*out*.println("Connecting to database...");  
 this.con = DriverManager.*getConnection*(url);  
 System.*out*.println("Connected to database");  
 this.stmt = this.con.createStatement();  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public Connection getConnection() { return con; }  
  
 public Statement getStatement() { return stmt; }  
}

EquipmentDAO.java

package org.example.kr.dao;  
  
import org.example.kr.model.Equipment;  
  
import java.sql.SQLException;  
import java.util.List;  
  
public interface EquipmentDAO {  
 Object get(int id);  
 Object get(String name);  
 List<Equipment> getAll();  
 List<Equipment> getAll(int typeId);  
  
 Boolean set(String name, String description, String type, Float price);  
 Boolean set(String name, String type, Float price);  
  
 Boolean updateEquipment(String keyName, String name, String description, String type, Float price);  
 Boolean updateEquipment(String keyName, String name, String type, Float price);  
 Boolean updateEquipment(String keyName, String type);  
 Boolean updateEquipment(String keyName, Float price);  
 Boolean delete(String name);  
}

TypesDAO.java

package org.example.kr.dao;  
  
import org.example.kr.model.Type;  
  
import java.util.List;  
  
public interface TypesDAO {  
 List<Type> getAll();  
 List<Type> getAllById(int id);  
 Type get(int id);  
 Boolean add(String name, int typeId);  
 Boolean add(String name);  
 Boolean update(String key, String name, int typeId);  
 Boolean update(String key, String name);  
 Boolean update(String key, int typeId);  
 Boolean delete(int id);  
 int getId(String name);  
 String getName(int id);  
}

UserDAO.java

package org.example.kr.dao;  
  
import org.example.kr.model.User;  
  
public interface UserDAO {  
 User get(String key);  
 Boolean set(String name, String pass);  
}

UserEquipDAO.java

package org.example.kr.dao;  
  
import java.util.List;  
  
public interface UserEquipDAO {  
 List<Integer> getWishList(int id);  
 Boolean addToWishList(int equipId, int userId);  
 Boolean removeFromWishList(int equipId, int userId);  
}

Equipment.java

package org.example.kr.model;  
  
public class Equipment {  
 private final int id;  
 private final String name;  
 private final String description;  
 private final double price;  
 private final String type;  
 public Equipment(int id, String name, String description,String type, double price) {  
 this.id = id;  
 this.name = name;  
 this.description = description;  
 this.price = price;  
 this.type = type;  
 }  
  
 public int getId() { return id; }  
 public String getName() { return name; }  
 public String getDescription() { return description; }  
 public double getPrice() { return price; }  
 public String getType() { return type; }  
}

Type.java

package org.example.kr.model;  
  
public class Type {  
 private final int id;  
 private final String name;  
 private final int typeId;  
 public Type(int id, String name, int typeId) {  
 this.id = id;  
 this.name = name;  
 this.typeId = typeId;  
 }  
 public int getId() { return id; }  
 public String getName() { return name; }  
 public int getTypeId() { return typeId; }  
}

User.java

package org.example.kr.model;  
  
public class User {  
 private int id;  
 private final String name;  
 private final String password;  
 private final Boolean admin;  
  
 public User(int id, String name, String password, Boolean admin) {  
 this.id = id;  
 this.name = name;  
 this.password = password;  
 this.admin = admin;  
 }  
  
 public String getName() {  
 return name;  
 }  
 public String getPassword() {  
 return password;  
 }  
 public int getId() {  
 return id;  
 }  
 public Boolean isAdmin() {  
 return admin;  
 }  
}

EquipmentService.java

package org.example.kr.service;  
  
import org.example.kr.dao.DAOFactory;  
import org.example.kr.model.Equipment;  
  
import java.sql.SQLException;  
import java.util.List;  
  
public class EquipmentService {  
 DAOFactory daoFactory;  
 public EquipmentService(DAOFactory daoFactory) {  
 this.daoFactory = daoFactory;  
 }  
  
 public Equipment getByName(String name) throws SQLException {  
 return (Equipment) daoFactory.getEquipmentDAO().get(name);  
 }  
  
 public Equipment getById(int id) throws SQLException {  
 return (Equipment) daoFactory.getEquipmentDAO().get(id);  
 }  
  
 public List<Equipment> getAll() throws SQLException {  
 return daoFactory.getEquipmentDAO().getAll();  
 }  
  
 public List<Equipment> getAll(int typeId) throws SQLException {  
 return daoFactory.getEquipmentDAO().getAll(typeId);  
 }  
  
 public boolean addEquipment(String name, String description, String type, float price) throws SQLException {  
 return daoFactory.getEquipmentDAO().set(name, description, type, price);  
 }  
  
 public boolean addEquipment(String name, String type, float price) throws SQLException {  
 return daoFactory.getEquipmentDAO().set(name, type, price);  
 }  
  
 public boolean updateEquipment(String keyName, String name, String description, String type, Float price) throws SQLException {  
 return daoFactory.getEquipmentDAO().updateEquipment(keyName, name, description, type, price);  
 }  
  
 public boolean updateEquipment(String keyName, String name, String type, Float price) throws SQLException {  
 return daoFactory.getEquipmentDAO().updateEquipment(keyName, name, type, price);  
 }  
 public boolean updateEquipment(String keyName, Float price) throws SQLException {  
 return daoFactory.getEquipmentDAO().updateEquipment(keyName, price);  
 }  
 public boolean updateEquipment(String keyName, String type) throws SQLException {  
 return daoFactory.getEquipmentDAO().updateEquipment(keyName, type);  
 }  
 public boolean deleteEquipment(String keyName) throws SQLException {  
 return daoFactory.getEquipmentDAO().delete(keyName);  
 }  
}

TypesService.java

package org.example.kr.service;  
  
import org.example.kr.dao.DAOFactory;  
import org.example.kr.model.Type;  
  
import java.util.List;  
  
  
public class TypesService {  
 DAOFactory daoFactory;  
 public TypesService(DAOFactory daoFactory) {  
 this.daoFactory = daoFactory;  
 }  
  
 public List<Type> getAll() {  
 return daoFactory.getTypesDAO().getAll();  
 }  
 public List<Type> getAllById(int id) {  
 return daoFactory.getTypesDAO().getAllById(id);  
 }  
  
 public Type get(int id) {  
 return daoFactory.getTypesDAO().get(id);  
 }  
  
 public Boolean addCategory(String name, int typeId) {  
 return daoFactory.getTypesDAO().add(name, typeId);  
 }  
  
 public Boolean addCategory(String name) {  
 return daoFactory.getTypesDAO().add(name);  
 }  
  
 public Boolean updateCategory(String key, String name, int typeId) {  
 return daoFactory.getTypesDAO().update(key, name, typeId);  
 }  
  
 public Boolean updateCategory(String key, String name) {  
 return daoFactory.getTypesDAO().update(key, name);  
 }  
  
 public Boolean updateCategory(String key, int typeId) {  
 return daoFactory.getTypesDAO().update(key, typeId);  
 }  
  
 public Boolean delete(int id) {  
 return daoFactory.getTypesDAO().delete(id);  
 }  
  
 public int getId(String name) {  
 return daoFactory.getTypesDAO().getId(name);  
 }  
  
 public String getName(int id) {  
 return daoFactory.getTypesDAO().getName(id);  
 }  
}

UserService.java

package org.example.kr.service;  
  
import org.example.kr.dao.DAOFactory;  
import org.example.kr.model.User;  
  
import java.util.List;  
  
public class UserService {  
 DAOFactory daoFactory;  
  
 public UserService(DAOFactory daoFactory) {  
 this.daoFactory = daoFactory;  
 }  
  
 public User getByLogin(String login) {  
 return daoFactory.getUserDAO().get(login);  
 }  
  
 public List<Integer> getWishList(int id) {  
 return daoFactory.getUserEquipDAO().getWishList(id);  
 }  
  
 public boolean addToWishList(int equipID, int loginID) {  
 return daoFactory.getUserEquipDAO().addToWishList(equipID, loginID);  
 }  
  
 public boolean removeFromWishList(int equipID, int loginID) {  
 return daoFactory.getUserEquipDAO().removeFromWishList(equipID, loginID);  
 }  
  
 public boolean checkPass(User user, String pass) {  
 return user.getPassword().equals(pass);  
 }  
  
 public boolean create(String login, String password) {  
 return daoFactory.getUserDAO().set(login, password);  
 }  
}

ContextListener.java

package org.example.kr;  
  
import jakarta.servlet.ServletContextEvent;  
import jakarta.servlet.ServletContextListener;  
import org.example.kr.dao.DAOFactory;  
import org.example.kr.service.EquipmentService;  
import org.example.kr.service.TypesService;  
import org.example.kr.service.UserService;  
  
public class ContextListener implements ServletContextListener {  
 @Override  
 public void contextInitialized(ServletContextEvent sce) {  
 DAOFactory factory = new DAOFactory();  
  
 UserService userService = new UserService(factory);  
 EquipmentService equipmentService = new EquipmentService(factory);  
 TypesService typesService = new TypesService(factory);  
  
 sce.getServletContext().setAttribute("userService", userService);  
 sce.getServletContext().setAttribute("equipmentService", equipmentService);  
 sce.getServletContext().setAttribute("typesService", typesService);  
 }  
}

FrontControllerServlet.java

package org.example.kr;  
  
import jakarta.servlet.ServletConfig;  
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 org.example.kr.model.\*;  
import org.example.kr.service.\*;  
  
import java.io.IOException;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;  
  
@WebServlet(name = "FrontControllerServlet", urlPatterns = "/do/\*")  
public class FrontControllerServlet extends HttpServlet {  
 UserService userService;  
 EquipmentService equipmentService;  
 TypesService typesService;  
 @Override  
 public void init(ServletConfig config) throws ServletException {  
 userService = (UserService) config.getServletContext().getAttribute("userService");  
 equipmentService = (EquipmentService) config.getServletContext().getAttribute("equipmentService");  
 typesService = (TypesService) config.getServletContext().getAttribute("typesService");  
 }  
  
 public void proceed(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException, SQLException {  
 String check;  
 String query;  
 String equipId;  
 String type;  
 int result;  
 String path = req.getPathInfo();  
  
 if (path == null) {  
 path = "/";  
 }  
 switch (path) {  
 case "/logout":  
 logout(req, resp);  
 break;  
 case "/login":  
 check = req.getParameter("submit");  
 if (check.equals("Log in")) {  
 login(req, resp);  
 }else if (check.equals("Register")) {  
 register(req, resp);  
 }  
 break;  
 case "/profile":  
 profile(req, resp);  
 break;  
 case "/equipment":  
 query = req.getQueryString();  
 result = Integer.*parseInt*(query);  
 equipment(req, resp, result, false);  
 break;  
 case "/action":  
 check = req.getParameter("action");  
 query = req.getParameter("hidden");  
 type = req.getParameter("type");  
  
 // Перевірка Редагування чи Видалення  
 // Всередині першої перевірки - перевірка чи категорія чи товар  
 if (check.equals("Edit")) {  
 if (type.equals("equip")) {  
 result = Integer.*parseInt*(query);  
 equipment(req, resp, result, true);  
 }else if (type.equals("category")) {  
 req.setAttribute("edit", true);  
 req.setAttribute("equip", false);  
 req.setAttribute("query", query);  
 req.getRequestDispatcher("/jsp/add.jsp").forward(req, resp);  
 }  
 }else if (check.equals("Delete")) {  
 if (type.equals("equip")) {  
 delete(req, resp, query, true);  
 }else if (type.equals("category")) {  
 delete(req, resp, query, false);  
 }  
 }  
 break;  
 case "/edit":  
 query = req.getParameter("hidden");  
 edit(req, resp, query);  
 break;  
 case "/editCategory":  
 query = req.getParameter("hidden");  
 editCategory(req, resp, query);  
 break;  
 case "/proceedAdd":  
 check = req.getParameter("add");  
  
 if (check.equals("Add Equipment")) {  
 req.setAttribute("equip", true);  
 req.getRequestDispatcher("/jsp/add.jsp").forward(req, resp);  
 } else if (check.equals("Add Category")) {  
 req.setAttribute("equip", false);  
 req.getRequestDispatcher("/jsp/add.jsp").forward(req, resp);  
 }  
 break;  
 case "/add":  
 check = req.getParameter("hidden");  
  
 if (check.equals("equip")) {  
 add(req, resp, true);  
 } else if (check.equals("category")) {  
 add(req, resp, false);  
 }  
 break;  
 case "/addToWishList":  
 equipId = req.getParameter("id");  
 addToWishList(req, resp,equipId);  
 break;  
 case "/removeFromWishList":  
 equipId = req.getParameter("id");  
 removeFromWishList(req, resp, equipId);  
 break;  
 case "/sort":  
 query = req.getQueryString();  
 sort(req, resp, query);  
 default:  
 catalog(req, resp);  
 break;  
 }  
 }  
  
 public void login(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 req.getSession().invalidate();  
 String username = req.getParameter("login");  
 String password = req.getParameter("pass");  
  
 User user = userService.getByLogin(username);  
  
  
 if (user == null) {  
 error(req, resp, "Invalid username");  
 return;  
 }  
  
 if (!userService.checkPass(user, password)) {  
 error(req, resp, "Invalid password");  
 return;  
 }  
  
 List<Integer> wish = userService.getWishList(user.getId());  
 req.getSession().setAttribute("user", user);  
 req.getSession().setAttribute("wish", wish);  
 resp.sendRedirect(".");  
 }  
  
 public void logout(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 req.getSession().invalidate();  
 resp.sendRedirect(".");  
 }  
  
 public void register(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 req.getSession().invalidate();  
 String username = req.getParameter("login");  
 String password = req.getParameter("pass");  
  
 User user = userService.getByLogin(username);  
 if (user != null) {  
 error(req, resp, "User already exists");  
 return;  
 }  
  
 boolean check = userService.create(username, password);  
 if (check) {  
 req.getSession().setAttribute("user", userService.getByLogin(username));  
 resp.sendRedirect(".");  
 }else {  
 error(req, resp, "Unable to register");  
 }  
 }  
  
 public void profile(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException, SQLException {  
 List<Integer> wishes = (List<Integer>) req.getSession().getAttribute("wish");  
 List<Equipment> list = new ArrayList<>();  
 for (int id: wishes) {  
 list.add(equipmentService.getById(id));  
 }  
 req.setAttribute("list", list);  
 req.getRequestDispatcher("/jsp/profile.jsp").forward(req, resp);  
 }  
  
 public void catalog(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException, SQLException {  
 if (req.getAttribute("types") == null) {  
 req.setAttribute("types", typesService.getAll());  
 req.setAttribute("list", equipmentService.getAll());  
 }else {  
 String type\_id = String.*valueOf*(req.getAttribute("type\_id"));  
 List<Equipment> list;  
 if (type\_id != null) {  
 list = equipmentService.getAll(Integer.*parseInt*(type\_id));  
 }  
 else {  
 list = new ArrayList<>();  
 }  
 req.setAttribute("list", list);  
 }  
  
 req.getRequestDispatcher("/jsp/catalog.jsp").forward(req, resp);  
 }  
  
 public void equipment(HttpServletRequest req, HttpServletResponse resp, int query, boolean edit) throws ServletException, IOException, SQLException {  
 Equipment equip = equipmentService.getById(query);  
 req.setAttribute("equip", equip);  
  
 if (edit) {  
 req.setAttribute("edit", true);  
 }  
  
 req.getRequestDispatcher("/jsp/equipment.jsp").forward(req, resp);  
 }  
  
 public void edit(HttpServletRequest req, HttpServletResponse resp, String key) throws ServletException, IOException, SQLException {  
 String name = req.getParameter("name");  
 String description = req.getParameter("description");  
 Float price = 0.0F;  
 String type = req.getParameter("type");  
  
 if (!req.getParameter("price").isEmpty()) {  
 try {  
 price = Float.*parseFloat*(req.getParameter("price"));  
 }catch (NumberFormatException e) {  
 error(req, resp, "Invalid price");  
 return;  
 }  
 }  
  
 if (description.isEmpty() && !name.isEmpty() && price != 0.0 && !type.isEmpty()) {  
 // Назва + Тип + Ціна  
 if (!equipmentService.updateEquipment(key, name, type, price)) {  
 error(req, resp, "Unable to update equipment");  
 return;  
 }  
 }else if (description.isEmpty() && name.isEmpty() && price == 0.0 && !type.isEmpty()) {  
 // Тип  
 if (!equipmentService.updateEquipment(key, type)) {  
 error(req, resp, "Unable to update equipment");  
 return;  
 }  
 }else if (description.isEmpty() && name.isEmpty() && price != 0.0 && type.isEmpty()) {  
 // Ціна  
 if (!equipmentService.updateEquipment(key, price)) {  
 error(req, resp, "Unable to update equipment");  
 return;  
 }  
 }else {  
 // Назва + Опис + Тип + Ціна  
 if (!equipmentService.updateEquipment(key, name, description, type, price)) {  
 error(req, resp, "Unable to update equipment");  
 return;  
 }  
 }  
 req.getRequestDispatcher("/do").forward(req, resp);  
 }  
  
 public void editCategory(HttpServletRequest req, HttpServletResponse resp, String key) throws ServletException, IOException {  
 String name = req.getParameter("name");  
 String category = req.getParameter("category");  
  
 if (category.isEmpty()) {  
 if (!typesService.updateCategory(key, name)) {  
 error(req, resp, "Unable to update category");  
 }  
 }  
 else if (name.isEmpty()) {  
 int typeId = typesService.getId(category);  
 if (!typesService.updateCategory(key, typeId)) {  
 error(req, resp, "Unable to update category");  
 }  
 }  
 else {  
 int typeId = typesService.getId(category);  
 if (!typesService.updateCategory(key, name, typeId)) {  
 error(req, resp, "Unable to update category");  
 }  
 }  
 req.getRequestDispatcher("/do").forward(req, resp);  
 }  
  
 public void delete(HttpServletRequest req, HttpServletResponse resp, String key, Boolean equip) throws ServletException, IOException, SQLException {  
 if (equip) {  
 if (!equipmentService.deleteEquipment(key)) {  
 error(req, resp, "Unable to delete equipment");  
 return;  
 }  
 }else {  
 int id = typesService.getId(key);  
 if (!typesService.delete(id)) {  
 error(req, resp, "Unable to delete category. Try to delete all elements from this category");  
 }  
 }  
  
 req.getRequestDispatcher("/do").forward(req, resp);  
 }  
  
 public void addToWishList(HttpServletRequest req, HttpServletResponse resp, String equipId) throws ServletException, IOException, SQLException {  
 User user = (User) req.getSession().getAttribute("user");  
 if (!userService.addToWishList(Integer.*parseInt*(equipId), user.getId())) {  
 error(req, resp, "Unable to add to wish list");  
 return;  
 }  
 List<Integer> wish = userService.getWishList(user.getId());  
 req.getSession().setAttribute("wish", wish);  
 req.getRequestDispatcher("/do").forward(req, resp);  
 }  
  
 public void removeFromWishList(HttpServletRequest req, HttpServletResponse resp, String equipId) throws ServletException, IOException, SQLException {  
 User user = (User) req.getSession().getAttribute("user");  
 if (!userService.removeFromWishList(Integer.*parseInt*(equipId), user.getId())) {  
 error(req, resp, "Unable to remove from wish list");  
 return;  
 }  
 List<Integer> wish = userService.getWishList(user.getId());  
 req.getSession().setAttribute("wish", wish);  
 profile(req, resp);  
 }  
  
 public void sort(HttpServletRequest req, HttpServletResponse resp, String query) throws ServletException, IOException {  
 int id = Integer.*parseInt*(query);  
 List<Type> types = typesService.getAllById(id);  
  
 req.setAttribute("types", types);  
 req.setAttribute("type\_id", id);  
 req.getRequestDispatcher(".").forward(req, resp);  
 }  
  
 public void add(HttpServletRequest req, HttpServletResponse resp, Boolean equipment) throws ServletException, IOException, SQLException {  
 if (equipment) {  
 String name = req.getParameter("name");  
 String description = req.getParameter("description");  
 Float price = null;  
 String type = req.getParameter("type");  
  
 if (!req.getParameter("price").isEmpty()) {  
 try {  
 price = Float.*parseFloat*(req.getParameter("price"));  
 }catch (NumberFormatException e) {  
 error(req, resp, "Invalid price");  
 return;  
 }  
 }else {  
 error(req, resp, "Price cannot be empty");  
 return;  
 }  
  
 if (!equipmentService.addEquipment(name, description, type, price)) {  
 error(req, resp, "Unable to add equipment");  
 }  
 }  
 else {  
 String name = req.getParameter("name");  
 String category = req.getParameter("category");  
  
 if (category.isEmpty()) {  
 if (!typesService.addCategory(name)) {  
 error(req, resp, "Unable to add category");  
 }  
 }else {  
 int id = typesService.getId(category);  
 if (!typesService.addCategory(name, id)) {  
 error(req, resp, "Unable to add category");  
 }  
 }  
 }  
 req.getRequestDispatcher("/do").forward(req, resp);  
 }  
  
 public void error(HttpServletRequest req, HttpServletResponse resp, String txt) throws ServletException, IOException {  
 req.setAttribute("error", txt);  
 req.getRequestDispatcher("/jsp/error.jsp").forward(req, resp);  
 }  
  
 @Override  
 public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 try {  
 proceed(req, resp);  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 @Override  
 public void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 try {  
 proceed(req, resp);  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

add.jsp

<%@ **taglib** prefix="c" uri="jakarta.tags.core"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <%@**include** file="header.jspf"%>  
 <title>Add Element</title>  
</head>  
<body>  
<form action="." method="post">  
 <input type="submit" value="До каталогу">  
</form>  
 <**c:choose**>  
 <**c:when** test="${requestScope.get(\"equip\")}">  
 <form action="${pageContext.request.contextPath}/do/add" method="post">  
 <input type="text" name="name" placeholder="Назва"><br />  
 <input type="text" name="description" placeholder="Опис"><br />  
 <input type="text" name="price" placeholder="Ціна"><br />  
 <input type="text" name="type" placeholder="Категорія"><br />  
 <input type="submit" value="Додати">  
 <input type="hidden" value="equip" name="hidden">  
 </form>  
 </**c:when**>  
 <**c:when** test="${!requestScope.get(\"equip\")}">  
 <**c:if** test="${!requestScope.get(\"edit\")}">  
 <form action="${pageContext.request.contextPath}/do/add" method="post">  
 <input type="text" name="name" placeholder="Назва"><br />  
 <input type="text" name="category" placeholder="В якій категорії"><br />  
 <input type="submit" value="Додати">  
 <input type="hidden" value="category" name="hidden">  
 </form>  
 </**c:if**>  
 <**c:if** test="${requestScope.get(\"edit\")}">  
 <form action="${pageContext.request.contextPath}/do/editCategory" method="post">  
 <input type="text" name="name" placeholder="Назва"><br />  
 <input type="text" name="category" placeholder="В якій категорії"><br />  
 <input type="submit" value="Змінити">  
 <input type="hidden" value="${requestScope.get("query")}" name="hidden">  
 </form>  
 </**c:if**>  
 </**c:when**>  
 </**c:choose**>  
</body>  
<div class="footer">  
 <%@**include** file="footer.jspf"%>  
</div>  
</html>

catalog.jsp

<%@ **taglib** prefix="c" uri="jakarta.tags.core"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<style>  
 <%@**include** file="/WEB-INF/style.css"%>  
</style>  
<head>  
 <title>Catalog</title>  
 <%@**include** file="header.jspf"%>  
</head>  
<body>  
<h1><a href="${pageContext.request.contextPath}/do">Catalog</a></h1>  
<span class="types">  
 <**c:forEach** var="type" items="${requestScope.get(\"types\")}">  
 <div>  
 <a href="${pageContext.request.contextPath}/do/sort?${type.getId()}">${type.getName()}</a>  
 <**c:if** test="${sessionScope.get(\"user\").isAdmin()}">  
 <form action="${pageContext.request.contextPath}/do/action" method="post" >  
 <input type="submit" value="Edit" name="action">  
 <input type="submit" value="Delete" name="action">  
 <input type="hidden" value="${type.getName()}" name="hidden">  
 <input type="hidden" value="category" name="type">  
 </form>  
 </**c:if**>  
 </div>  
 </**c:forEach**>  
</span>  
<br>  
<**c:forEach** var="equip" items="${requestScope.get(\"list\")}">  
 <div class = field>  
 <**c:if** test="${!empty sessionScope.get(\"user\")}">  
 <**c:if** test="${!sessionScope.get(\"wish\").contains(equip.getId())}">  
 <form action="${pageContext.request.contextPath}/do/addToWishList" method="post">  
 <input type="submit" value="\*">  
 <input type="hidden" value="${equip.getId()}" name="id">  
 </form>&nbsp&nbsp  
 </**c:if**>  
 </**c:if**>  
 <p class="catalog">  
 ${equip.getType()} <a href="${pageContext.request.contextPath}/do/equipment?${equip.getId()}">${equip.getName()}</a> ${equip.getPrice()}&nbsp грн &nbsp&nbsp  
 <**c:if** test="${sessionScope.get(\"user\").isAdmin()}">  
 <form action="${pageContext.request.contextPath}/do/action" method="post" >  
 <input type="submit" value="Edit" name="action">  
 <input type="submit" value="Delete" name="action">  
 <input type="hidden" value="${equip.getId()}" name="hidden">  
 <input type="hidden" value="equip" name="type">  
 </form>  
 </**c:if**>  
  
 </p>  
 </div>  
 <br>  
</**c:forEach**>  
<**c:if** test="${sessionScope.get(\"user\").isAdmin()}">  
 <form action="${pageContext.request.contextPath}/do/proceedAdd" method="post">  
 <input type="submit" value="Add Equipment" name="add">  
 <input type="submit" value="Add Category" name="add">  
 </form>  
</**c:if**>  
</body>  
<div class="footer">  
 <%@**include** file="footer.jspf"%>  
</div>  
</html>

equipment.jsp

<%@ **taglib** prefix="c" uri="jakarta.tags.core"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<style>  
 <%@**include** file="/WEB-INF/style.css"%>  
</style>  
<head>  
 <%@**include** file="header.jspf"%>  
 <title>${requestScope.get("equip").getName()}</title>  
</head>  
<body>  
<form action="." method="post">  
 <input type="submit" value="До каталогу">  
</form>  
<span class="nowrap">  
<**c:if** test="${!empty sessionScope.get(\"user\")}">  
 <**c:if** test="${!sessionScope.get(\"wish\").contains(requestScope.get(\"equip\").getId())}">  
 <br>  
 <form action="addToWishList" method="post">  
 <input type="submit" value="\*">  
 <input type="hidden" value="${requestScope.get("equip").getId()}" name="id">  
 </form>  
 </**c:if**>  
</**c:if**>  
 </span>  
<h1>Назва товару: ${requestScope.get("equip").getName()}</h1>  
<h2>Ціна: ${requestScope.get("equip").getPrice()} грн</h2>  
<h2>Категорія: ${requestScope.get("equip").getType()}</h2>  
<div class="text-area">  
<h3>Опис: ${requestScope.get("equip").getDescription()}</h3>  
</div>  
<**c:if** test="${requestScope.get(\"edit\")}">  
<h2>Зміна товару</h2>  
<form action="${pageContext.request.contextPath}/do/edit" method="post">  
 <input type="text" name="name" placeholder="Назва"><br />  
 <input type="text" name="description" placeholder="Опис"><br />  
 <input type="text" name="price" placeholder="Ціна"><br />  
 <input type="text" name="type" placeholder="Категорія"><br />  
 <input type="submit" value="Змінити">  
 <input type="hidden" name="hidden" value="${requestScope.get("equip").getName()}">  
 </**c:if**>  
</form>  
</body>  
<div class="footer">  
 <%@**include** file="footer.jspf"%>  
</div>  
</html>

error.jsp

<%@ **taglib** prefix="c" uri="jakarta.tags.core"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Error!</title>  
</head>  
<body>  
 <%@**include** file="header.jspf"%>  
 <form action="." method="post">  
 <input type="submit" value="До каталогу">  
 </form>  
 <br>  
 Помилка '<**c:out** value="${requestScope.get(\"error\")}"/>'  
</body>  
<div class="footer">  
 <%@**include** file="footer.jspf"%>  
</div>  
</html>

footer.jspf

<%@ **page** contentType="text/html;charset=UTF-8"%>  
<footer>  
<p style="text-align: center"> Copyright 2024  
<span style="float:right"><a href="https://t.me/zoro\_supremacy">Telegram</a></span>  
</p>  
</footer>

header.jspf

<%@ **taglib** prefix="c" uri="jakarta.tags.core"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<style>  
 <%@**include** file="/WEB-INF/style.css"%>  
</style>  
<header>  
 <span style="float:right" class="nowrap">  
 <**c:if** test="${empty sessionScope.get(\"user\")}">  
 <form action="${pageContext.request.contextPath}/do/login" method="post">  
 <input type="text" name="login" placeholder="Логін"/>  
 <input type="text" name="pass" placeholder="Пароль"/>  
 <input type="submit" value="Log in" name="submit">  
 <input type="submit" value="Register" name="submit">  
 </form>  
 </**c:if**>  
  
  
  
 <**c:if** test="${!empty sessionScope.get(\"user\")}">  
 <div class="field">  
 <a href="profile">Профіль</a><**c:if** test="${!empty sessionScope.get(\"wish\") && sessionScope.get(\"wish\").size() > 0}">  
 (${sessionScope.get("wish").size()})  
 </**c:if**>&nbsp&nbsp  
 <form action="${pageContext.request.contextPath}/do/logout" method="post">  
 <input type="submit" value="Log out">  
 </form>  
 </div>  
 </**c:if**>  
 </span>  
</header>  
</html>

profile.jsp

<%@ **taglib** prefix="c" uri="jakarta.tags.core"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<style>  
 <%@**include** file="/WEB-INF/style.css"%>  
</style>  
<head>  
 <%@**include** file="header.jspf"%>  
 <title>Profile</title>  
</head>  
<body>  
<form action="." method="post">  
 <input type="submit" value="До каталогу">  
</form>  
 <br>  
 <h2>  
 Ім'я: ${sessionScope.get("user").getName()}  
 <br>  
 Ви  
 <**c:if** test="${!sessionScope.get(\"user\").isAdmin()}">  
 не  
 </**c:if**>  
 є адміністратором  
 </h2>  
<h2>  
 Список збереженого:  
</h2>  
<h3>  
 <**c:forEach** var="wish" items="${requestScope.get(\"list\")}">  
 ${wish.getType()} <a href="${pageContext.request.contextPath}/do/equipment?${wish.getId()}">${wish.getName()}</a> ${wish.getPrice()}&nbsp&nbsp  
 <form action="removeFromWishList" method="post">  
 <input type="submit" value="Delete">  
 <input type="hidden" value="${wish.getId()}" name="id">  
 </form>  
 </**c:forEach**>  
</h3>  
</body>  
<div class="footer">  
 <%@**include** file="footer.jspf"%>  
</div>  
</html>

style.css

.text-area {  
 width: 600px; /\* Ширина области \*/  
 height: 200px; /\* Высота области \*/  
}  
.catalog {  
 margin: 0;  
 font-size: 20px;  
}  
.nowrap {  
 white-space: nowrap;  
}  
.types {  
 display: grid;  
 grid-template-columns: repeat(10,10%);  
 white-space: nowrap;  
 word-spacing: 10px;  
 font-size: 25px;  
 margin: 20px;  
}  
.wish {  
 display: grid;  
 grid-template-rows: repeat(10,10%);  
}  
.field {  
 display: flex;  
 margin-left: 20px;  
}  
label {  
 display: inline-block;  
 width: 80px;  
 text-align: left;  
}  
.footer {  
 position: fixed;  
 left: 0;  
 bottom: 0;  
 width: 100%;  
 background-color: #808080;  
 color: black;  
 text-align: center;  
}  
a:link {  
 text-decoration: none;  
}  
a:visited {  
 text-decoration: none;  
 color: blue;  
}  
a:hover {  
 text-decoration: underline;  
}  
  
a:active {  
 text-decoration: underline;  
}

index.jsp

<% response.sendRedirect("do/");%>