МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток організації замовлення товарів із-за кордону»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ-81 Думанський Тимур Валерійович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____«__» _____2022 р.

Науковий керівник _____ к.т.н., доц., Антипенко В.П.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

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

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____В. В. Шендрик
«__» _____ 2022 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Думанський Тимур Валерійович

1 Тема роботи *Web-додаток організації замовлення товарів із-за кордону*

керівник роботи *Антипенко Вікторія Петрівна, к.т.н., доцент* ,

затверджені наказом по університету від « 27 » квітня 2022 р. №0301-VI

2 Строк подання студентом роботи «10» червня 2022 р.

3 Вхідні дані до роботи технічне завдання на розробку web-додатку організації замовлення товарів із-за кордону

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування web-додатку, розробка web-додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____

_____

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 10.10.2021

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Дослідження предметної області	20.10.2021-30.10.2021	
	Розробка макету web-додатку	10.11.2021-30.11.2021	
	Розробка web-додатку	01.12.2021-24.02.2022	
	Тестування	25.02.2022-14.03.2022	
	Розміщення на хостинг	15.03.2022-16.03.2022	
	Перевірка працездатності	17.03.2022-23.03.2022	
	Оформлення пояснювальної записки	24.03.2022-31.05.2022	

Студент _____ Думаньский Т.В.

(підпис)

Керівник роботи _____ к.т.н., доц. Антипенко В.П.

(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток організації замовлення товарів із-за кордону».

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел із 18 найменувань та трьох додатків. Загальний обсяг пояснювальної записки складає 97 сторінки, у тому числі 37 сторінок основного тексту, 2 сторінки списку використаних джерел, 58 сторінки додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку організації замовлення товарів із-за кордону.

У першому розділі проведено аналіз предметної області. Виконано огляд останніх досліджень та публікації. Сформульовано мету та задачі даного проекту.

У другому розділі проведено структурно-функціональне моделювання. Визначено варіанти використання web-додатку та спроектовано базу даних. У результаті були створені контекстна діаграма IDEF0, її декомпозиція, діаграма варіантів використання та логічна модель бази даних.

У третьому розділі описано архітектуру та програмну реалізацію web-додатку. Також було продемонстровано використання даного програмного продукту. Проведено успішне тестування його роботи.

Ключові слова: WEB-ДОДАТОК, ЗАМОВЛЕННЯ, ТОВАР, КРАЇНА, КОРДОН, БАЗА ДАНИХ, JAVA, SPRING BOOT, JSP, JAVASCRIPT, CSS, РОЗРОБКА.

ЗМІСТ

ВСТУП

На сьогоднішній день неможливо уявити деяку сферу діяльності людини, де б не використовувалися або не стали б в нагоді інтернет-технології. Сучасні web-додатки вражають швидкістю їх функціонування, доступністю з будь-якої точки планети та водночас безпекою даних. Кожного дня ми використовуємо веб-ресурси задля навчання, роботи і, звісно, відпочинку. Особливим попитом користується онлайн-шопінг, представлений на багатьох сайтах Всесвітньої мережі. Користувачам надається великий спектр та розмаїття товарів, який навряд чи можна зустріти у фізичних магазинах. Такий вид шопінгу гарантує значну економію часу покупців та зручність. Значною перевагою онлайн-шопінгу є те, що користувач має необмежений час для обміркування покупки, аналізу аналогів та пошуку нижчої ціни. Тому, він дійсно досить популярний сьогодні.

Під час онлайн-шопінгу користувачам достатньо лише знайти бажаний інтернет-магазин, обрати товар та оплатити замовлення. Більшість таких представництв мережі Інтернет надають чіткий опис товару, оптимальну кількість його фото, детальну інформацію про доставку тощо.

Проте, доволі часто виникає ситуація, коли закордонний онлайн-магазин не виконує доставку в деякі країни. Це відразу зменшує бажання користувача здійснювати покупки, використовуючи той чи інший сервіс. Нерідко наявними є й такі випадки, що вартість доставки може коштувати 50% від вартості самого замовлення. Ці фактори викликають незручності користувачу.

Для вирішення представленої проблеми актуальним буде створення web-додатку організації замовлення товарів із-за кордону.

Метою даного проекту є автоматизація процесів формування спільного онлайн-замовлення з країн, де доставка не виконується самим інтернет-магазином, за рахунок

створення відповідного web-додатку. Використання останнього дозволить заощадити кошти користувачів, котрі бажають замовити товари із-за кордону.

Для вирішення поставленої мети необхідно вирішити наступні задачі:

- визначити актуальність проблеми;
- проаналізувати предметну область;
- провести дослідження існуючих аналогів web-додатків;
- визначити структуру web-додатку організації замовлення товарів із-за кордону;
- визначити функціонал створеного web-додатку;
- спроектувати структуру бази даних;
- реалізувати web-додаток та розмістити його на хостингу;
- провести тестування створеного web-додатку.

Цінність даного web-додатку полягає в тому, що користувачі не будуть обмеженими у виборі товару через неможливість доставки в країну проживання або через високу її вартість.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

У наш час особливої популярності набули інтернет-магазини [1], котрі надають можливість користувачам обрати та замовити необхідні товари за лічені хвилини. Вони вражають розмаїттям товарів, починаючи від одягу та закінчуючи електронними девайсами. Особливою популярністю користуються закордонні інтернет-ресурси. Останні надають змогу придбати товари за останніми світовими трендами.

Купівля речей в інтернет-магазині базується на їх перегляді та виборі, а також на оформленні замовлення та подальшій оплаті. Більшість закордонних таких представництв мають гнучкий підхід. Вони також надають користувачу змогу замовити товар у будь-яку точку планети, використовуючи міжнародну доставку. Проте трапляються різні випадки. Наприклад, коли доставка до країни замовника неможлива. Саме тому поширюються веб-додатки, котрі дозволяють замовити товар, використовуючи послуги посередників. Такий вид сервісу став досить популярним останнім часом.

Посередники допомагають клієнтам отримати бажану річ у разі неможливості доставки до бажаної країни. Або ж при наявності мінімальної вартості покупки для виконання замовлення. У такому разі, користувач надсилає інформацію про обраний товар посереднику, який розраховує його вартість. При цьому сюди вже включені додаткові витрати на доставку та податки. Замовник оплачує вказану суму посереднику й отримує товар у своїй країні призначення. Але такий спосіб має певні недоліки. Основним є те, що відповідальність за отримання свого товару замовником залежить від посередника, а не від офіційного магазину. Співпраця з такими сервісами вимагає більше часу. Він затрачається не лише на вибір товару, але й на

його очікування. Основним інструментом для отримання замовлення та подальшої обробки зазвичай є web-додаток, який забезпечує діалог між користувачем та посередником. Для розробки не складних додатків зазвичай використовується WordPress. Через великий інструментарій та велику кількість доступних плагінів [2], його також можна використовувати для створення інтернет-магазинів, але для web-додатків котрі мають складну логіку маніпулювання даними він зазвичай не підходить. Для таких задач необхідне більш гнучке налагодження бізнес-логіки, саме тому для розробки web-додатку було вибрано мову програмування Java.

Тому, розробка web-додатку для організації замовлень товарів із-за кордону є актуальною задачею.

1.2 Аналіз програмних продуктів-аналогів

На сьогоднішній день web-додатки організації замовлення товарів за кордоном набули особливою популярності.

Першим прикладом є web-додаток AliExpress [3]. Його функціонал включає в себе наступне:

- пошук та замовлення товарів різних продавців;
- введення та зберігання особистої інформації замовника;
- оплату замовлення та доставки;
- формування великого замовлення для замовників з однієї країни;
- консолідацію замовлень із різних магазинів на платформі AliExpress в одне замовлення;
- трекінг замовлення.

Основними перевагами цього web-додатку є безкоштовна доставка або її низька вартість. Додаток підтримує формування одного великого замовлення з декількох. Проте є певна умова. Користувач повинен оформити замовлення для цього на певну мінімальну суму. На рисунку 1.1 представлено головну сторінку web-додатку AliExpress.

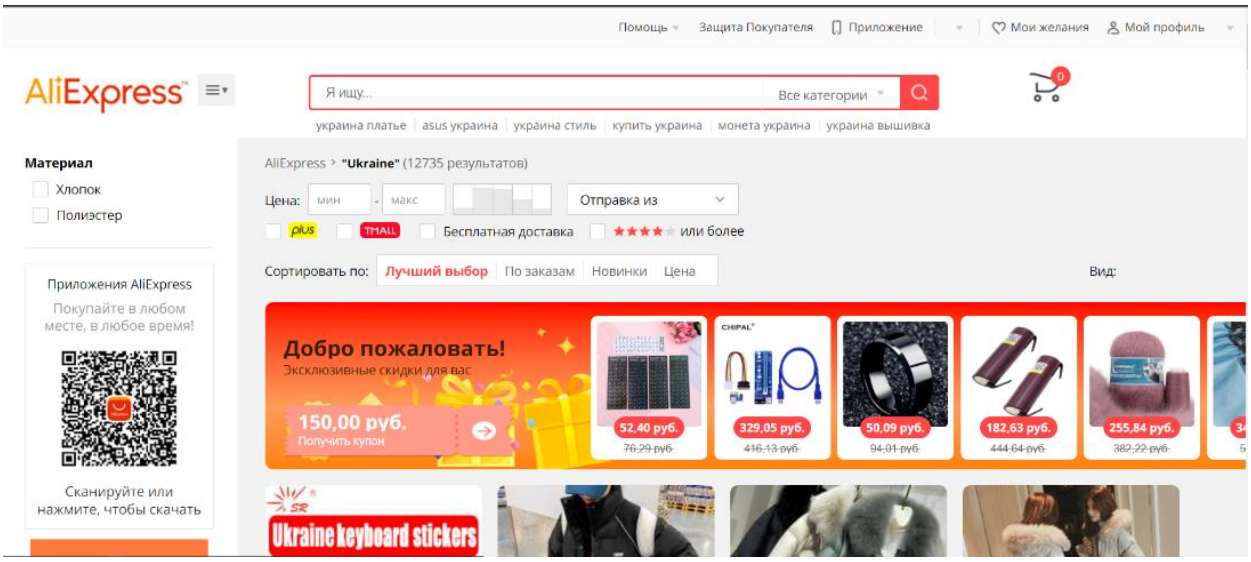


Рисунок 1.1 – Головна сторінка web-додатку AliExpress

Другий приклад – це web-додаток Bunddler [4]. Він надає можливість замовити речі за допомогою посередника. Користувач обирає товар. А оплачує його вартість посереднику. Bunddler формує усі покупки користувача до одного замовлення. Функціонал включає в себе наступне:

– вибір магазину з бажаними товарами та посередника;
– введення та зберігання особистої інформації замовника;
– оплату замовлення та доставки;
– консолідацію замовлень із різних інтернет-;
– трекінг замовлення.

Основними перевагами цього web-додатку є можливість замовити товари з найпопулярніших закордонних магазинів.

На рисунку 1.2 зображено головну сторінку web-додатку Bunddler.

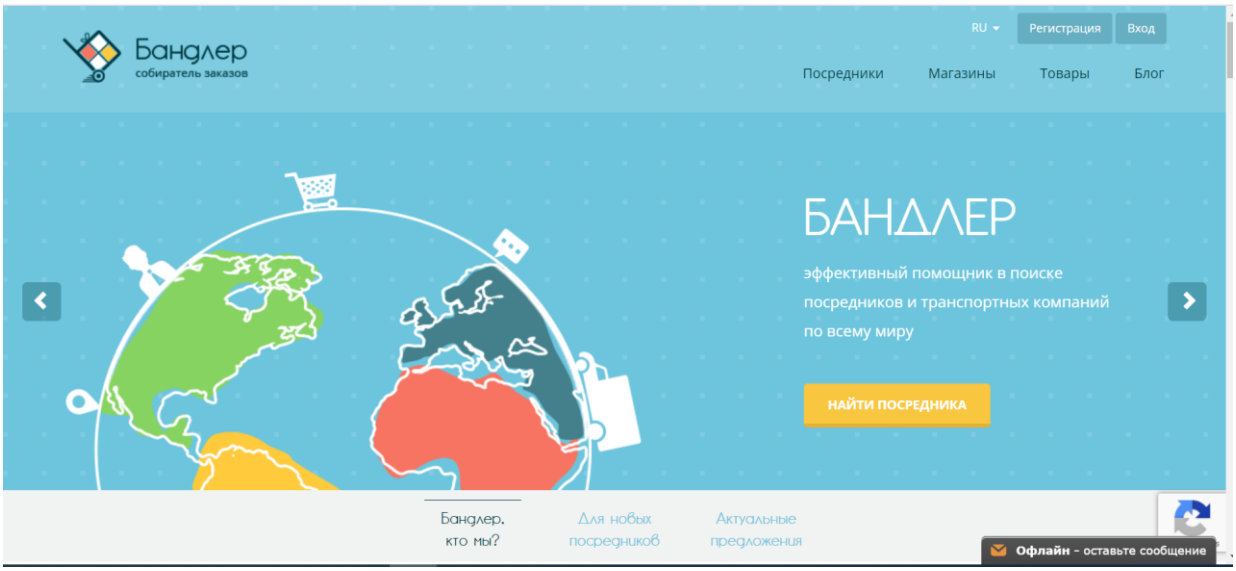


Рисунок 1.2 – Головна сторінка web-додатку Bunddler

Третій приклад – це web-додаток Meest [5]. За його допомогою можна проводити міжнародні поштово-логістичні рішення. Meets надає можливість зробити замовлення із будь-якого інтернет-магазину та оформити його на адресу власного складу. Далі сформована посилка доставляється до замовника на вказану адресу. До основного функціоналу входить наступне:

– введення та зберігання особистої інформації замовника;
– оформлення замовлення на адресу складу;
– оплату доставки;
– консолідацію замовлень із різних інтернет-магазинів в одне замовлення;
– трекінг замовлення.

Основними перевагами цього web-додатку є те, що Meest є одним з найбільш популярних сервісів, має велику кількість складів та пропонує користувачам додаткові послуги такі, як викуп товарів, їх упаковку та переупаковку у відправленні задля більшої надійності.

На рисунку 1.3 представлено головну сторінку web-додатку Meest

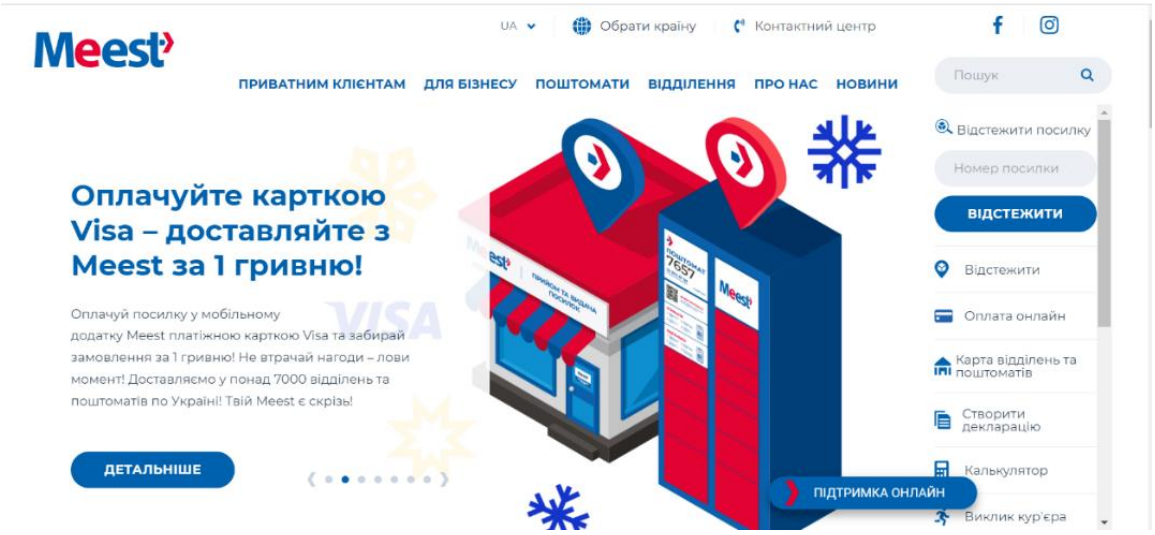


Рисунок 1.3 – Головна сторінка web-додатку Meest

Результати порівняльної характеристики програмних продуктів описаних вище представлено в таблиці 1.3.

Таблиця 1.3 – Порівняльна характеристика програмних продуктів

Web-додаток	AliExpress	Bunddler	Meest
Вартість користування	Безкоштовно	Безкоштовно	Безкоштовно
Можливість зробити замовлення із будь-якого закордонного магазину	-, лише магазини представлені на платформі AliExpress	-, лише магазини котрі мають посередників Bunddler	-, лише із магазинів в країнах котрих присутні склади компанії
Консолідація замовлень із різних інтернет-магазинів в одне замовлення	+	+	+
Відсутність посередників	+	-	+
Платформи	Web-app, мобільний додаток	Web-app	Web-app, мобільний додаток
Зрозумілість інтерфейсу	Інтуїтивно не зрозумілий	Інтуїтивно зрозумілий	Інтуїтивно зрозумілий
Наявність трекінгової системи	Деякі замовлення не відстежуються	Усі замовлення відстежуються	Усі замовлення відстежуються
Можливість спільного замовлення	-	+	-

За результатами проведеного аналізу існуючих аналогів усі зазначені web-додатки мають недоліки. Наприклад, Bunddler вимагає співпраці з посередниками. AliExpress доставляє лише ті товари, які представлені на своїй платформі. А в Meest немає можливості зробити спільне замовлення з іншими користувачами. Через це виникає потреба створити власний web-додаток організації замовлення товарів із-за кордону, який би подолав виявлені недоліки.

1.3 Постановка задачі

Основна мета проекту полягає в автоматизації процесів формування спільного онлайн-замовлення з країн, де доставка не виконується самим інтернет-магазином, за рахунок створення відповідного web-додатку.

Для досягнення мети проекту необхідно виконати наступні задачі:

- визначити актуальність проблеми;
- проаналізувати предметну область;
- провести дослідження існуючих аналогів web-додатків;
- визначити структуру web-додатку організації замовлення товарів із-за кордону;
- визначити функціонал створеного web-додатку;
- спроектувати структуру бази даних;
- реалізувати web-додаток та розмістити його на хостингу;
- провести тестування створеного web-додатку.

У цільовій аудиторії web-додатку можна виділити такі групи, як користувач та адміністратор. Розробка призначена для мешканців України, оскільки доставка товарів буде надаватися саме до цієї країни.

Основні вимоги до створюваного програмного продукту є наступними:

- інтуїтивно зрозумілий інтерфейс;
- підтримка web-додатку основними браузерами;
- зручне маніпулювання даними (введення, видалення, редагування).

Технічне завдання на розробку продукту у повному обсязі наведено у Додатку А.

Для реалізації даного web-додатку було обрано такі технології, як HTML [6], CSS [7]. Це стандартні інструменти для web-розробки. Більшість сучасних браузерів і web-додатків використовують мову програмування Javascript[8]. Вона дозволяє зробити інтерфейс для користувача більш динамічним, зручним та зрозумілим. Тому було прийняти рішення використовувати саме Javascript. Для роботи з базою даних було обрано систему управління базами даних (СУБД) MySQL [9]. Це реляційна СУБД, яка є одним з найкращих рішень для малих та середніх проектів. Для обробки запитів користувачів було обрано мову програмування Java [10].

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

2.1 Функціональне моделювання web-додатку в IDEF0

IDEF0 [11] представляє собою графічну нотацію, основне призначення якої полягає в формалізації та опису бізнес-процесів. Об’єкти демонструються ієрархічно, ставиться акцент на логічних відносинах між роботами.

Функціональне моделювання процесу замовлення товарів із-за кордону в нотації IDEF0 представлене на рисунку 2.1.

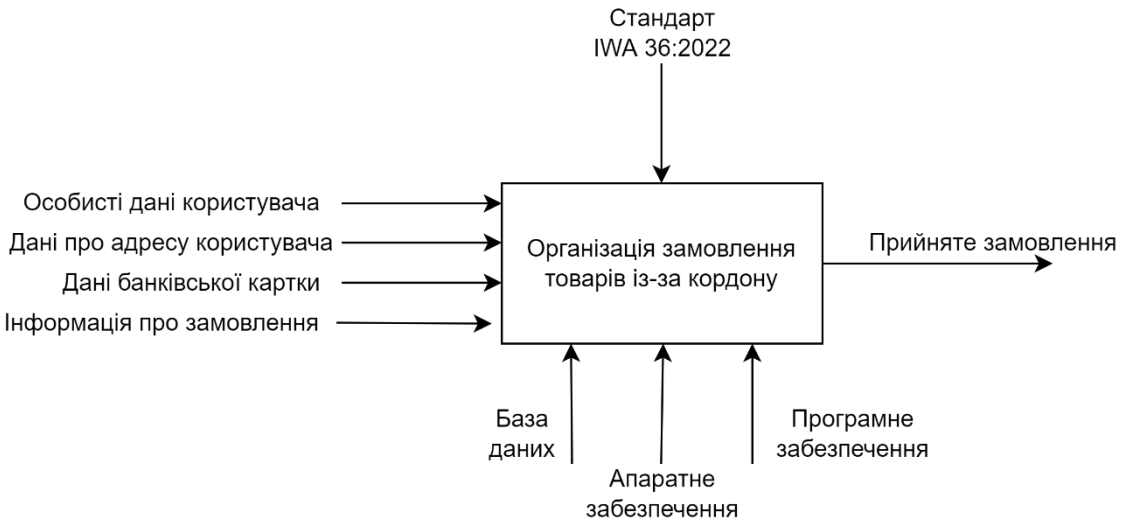


Рисунок 2.1 – Функціональна діаграма IDEF0

На вхід поступають особисті дані користувача, дані про адресу користувача, дані банківської карти та інформація про замовлення. Відбувається процес замовлення товарів із-за кордону за участі бази даних, апаратного забезпечення та програмного забезпечення. Обмежує даних процес стандарт IWA 36:2022. На виході маємо прийняте замовлення.

Декомпозиція ґрунтується на тому, що комплексний процес ділять на складові. Сприйняття даної моделі спрощується за рахунок представлення її меншими частинами у вигляді ієрархічної структури окремих діаграм. Декомпозиція 1-го рівня моделювання процесу замовлення товарів у нотації IDEF0 представлена на рисунку 2.2.

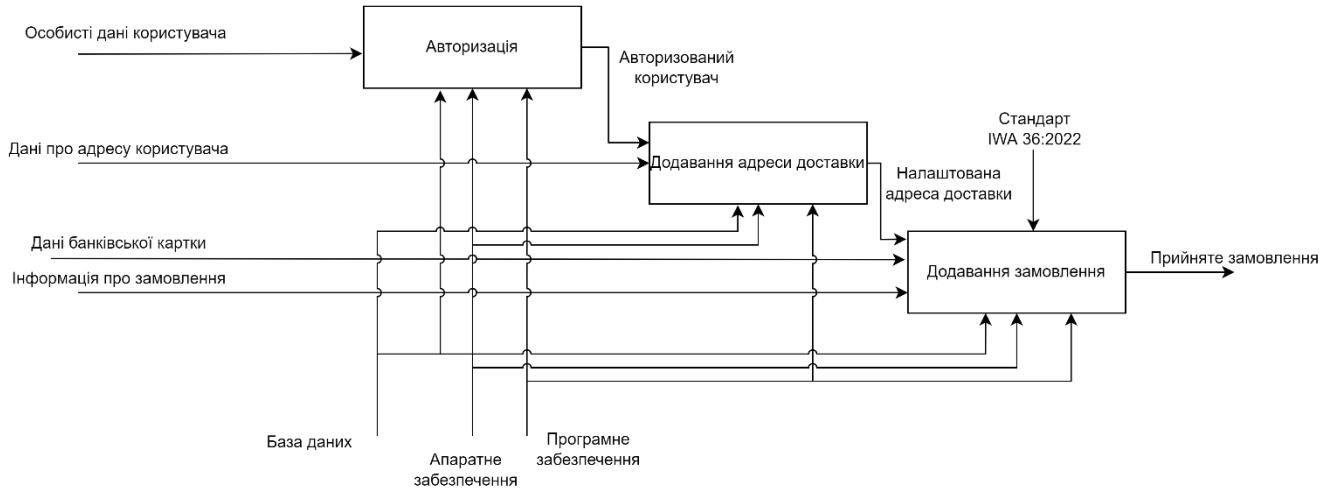


Рисунок 2.2 – Декомпозиція функціональної моделі

Процес замовлення товарів із-за кордону складається з таких етапів, як авторизація, додавання активного замовлення та додавання банківської картки.

2.2 Моделювання варіантів використання

Use-case [12] діаграма або діаграма варіантів використання визначає головний функціонал тієї чи іншої системи. Зазначаються актори, які мають певні варіанти застосування розробки. Варіанти використання представляють набір дій, які актор може виконувати уданому випадку над створюваним web-додатком. Графічне

представлення надає можливість краще зрозуміти як розробка може застосовуватися, які є варіанти взаємодії у користувачів із нею тощо.

Діаграма варіантів використання в нотації UML представлена на рисунку 2.3.

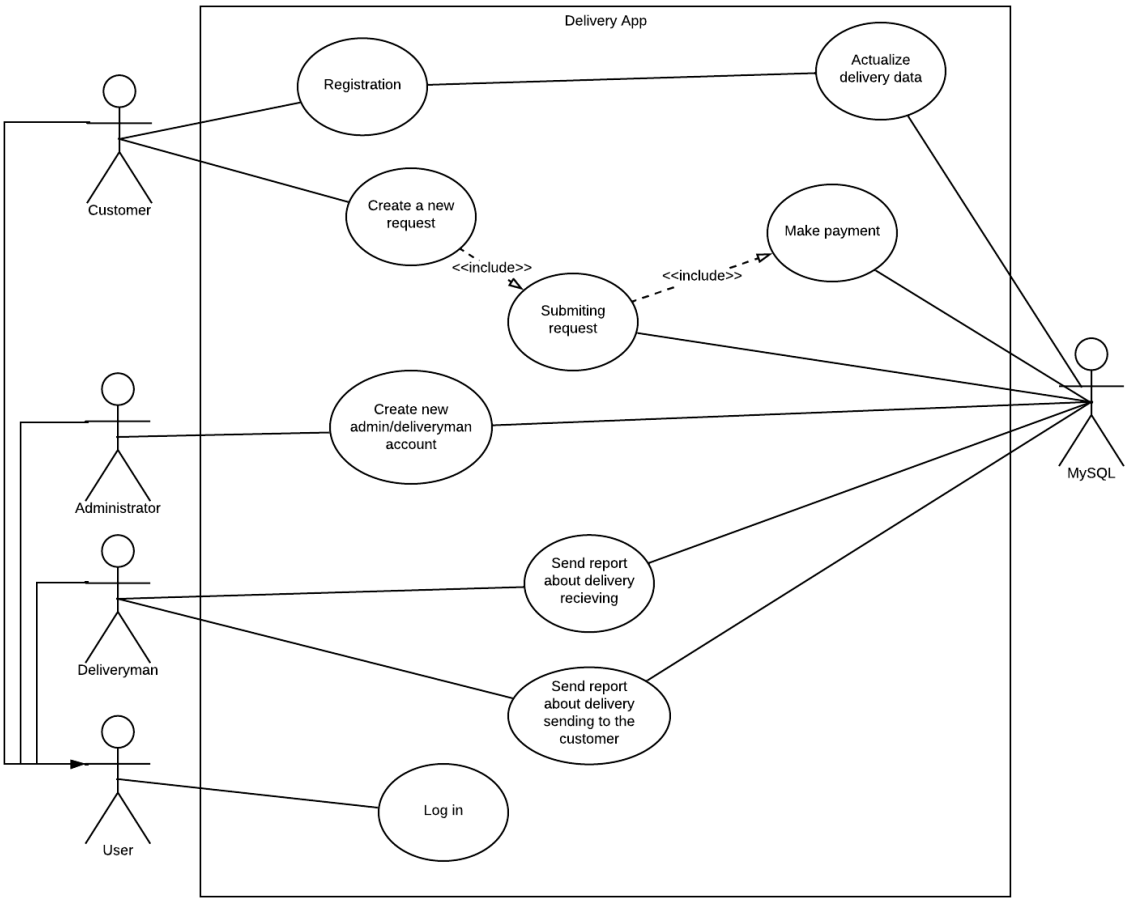


Рисунок 2.3 – Діаграма варіантів використання

У даному web-додатку існує 5 типів акторів:

– Клієнт; зареєстрований користувач, який має можливість зареєструватись, актуалізувати дані доставки, створити новий запит на доставку та оплатити замовлення;

– Адміністратор; підтримує роботу даного web-додатку та створює додаткові акаунти для адміністраторів та персоналу доставки;

– Персонал доставки; відповідає за отримання замовлення з магазину та відправлення його користувачу;

– Користувач; має змогу увійти в акаунт, скориставшись даними для авторизації;

– База даних; реагує на запити та зміни даних.

2.4 Проектування бази даних

Після створення шаблонів сторінок web-додатку, необхідно не тільки наповнити їх коректною інформацією й налагодити роботу з нею, але й забезпечити її правильне зберігання. Для описання фізичної структури БД використовуються фізичні моделі даних [13] (рис. 2.4).

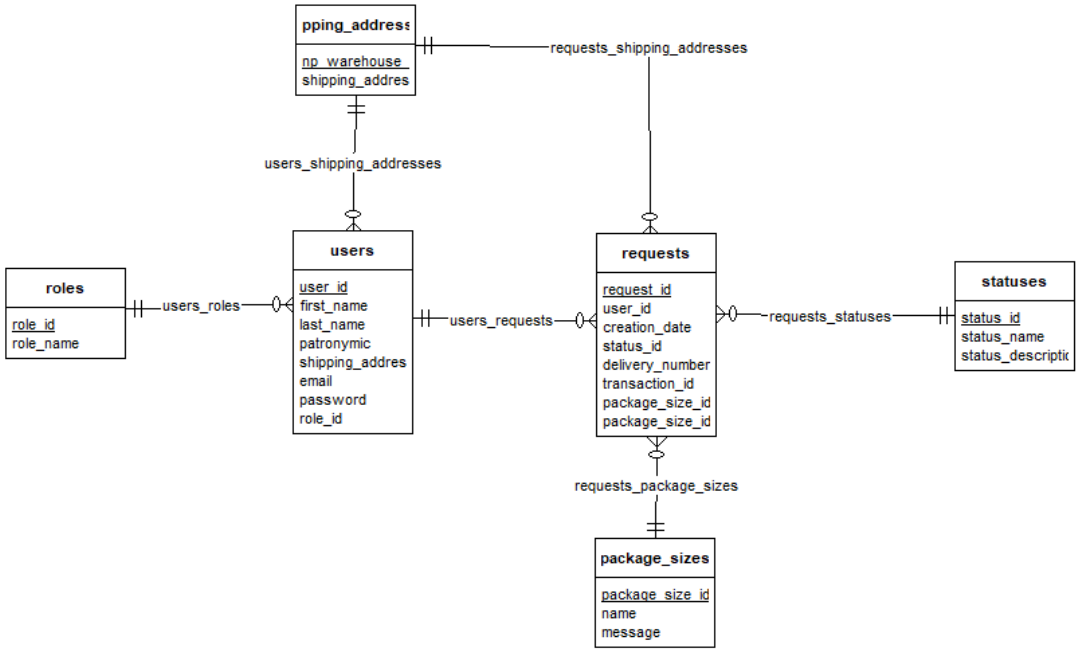


Рисунок 2.4 – Фізична модель даних

3 РОЗРОБКА WEB-ДОДАТКУ ОРГАНІЗАЦІЇ ЗАМОВЛЕННЯ ТОВАРІВ ІЗ-ЗА КОРДОНУ

3.1 Архітектура web-додатку

Для розробки архітектури даного програмного продукту перш за все необхідно визначити взаємодії між користувачем та його основними компонентами за допомогою High Level Design (HLD) діаграми [14].

Архітектура web-додатку організації замовлення товарів із-за кордону складається з наступних елементів:

- обробник відображення (Handler Mapping), який на основі запиту користувача обирає необхідний контролер для подальшої обробки;
- контролер (Controller), який взаємодіє з сервісами для коректної генерації вигляду та відправлення його користувачу;
- вигляд (View), який отримує дані від контролера для генерації відповіді користувачу;
- сервіси (Service), який виконує маніпуляції з даними та відправляє дані до контролеру;
- сутності (Entity), яка являється описом сутностей бази даних для використання їх в сервісі;
- репозиторії (Repository), який забезпечує доступ до бази даних;
- база даних (Database), яка забезпечує web-додаток даними.

HLD діаграма представлена на рисунку 3.1.

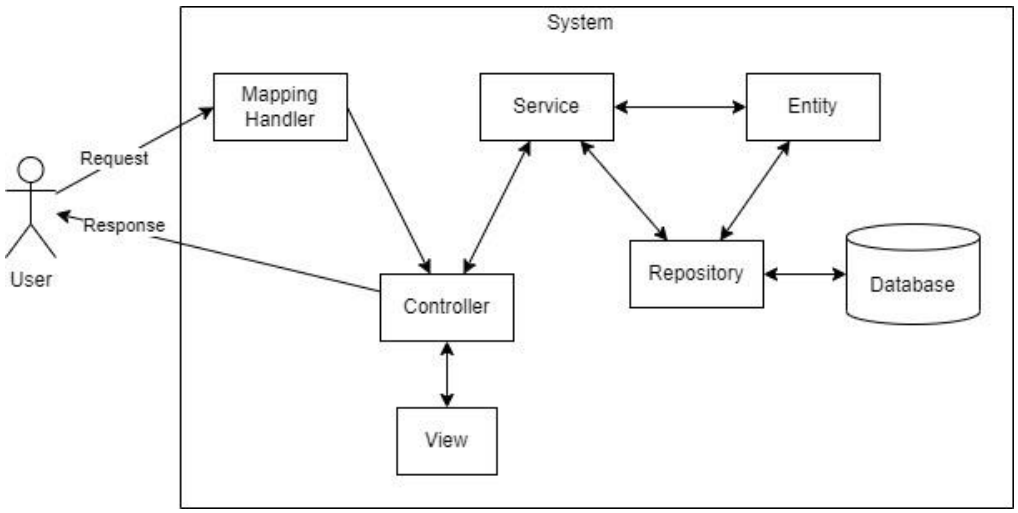


Рисунок 3.1 – HDL діаграма

3.2 Програмна реалізація web-додатку

Для запобігання виникнення помилок при завантаженні розроблюваного програмного продукту на хостинг та його автоматичного розгортання було вирішено скористатись контейнеризатором додатків Docker [15]. Отже, першим кроком необхідно створити файл-інструкцію Dockerfile [16] (рис. 3.2). Це виконується в кореневій папці проекту для генерації зображення web-додатку, яке потім буде використовуватись для створення контейнера.

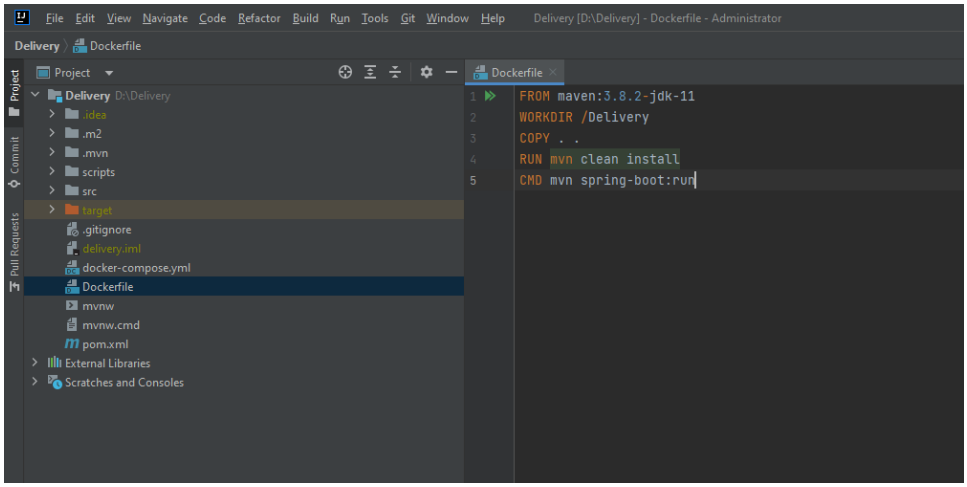


Рисунок 3.2 – Файл Dockerfile

Наступним кроком буде створення файлу docker-compose.yml [17] (рис. 3.3), котрий відповідає за розгортання бази даних MySQL та самого web-додатку.

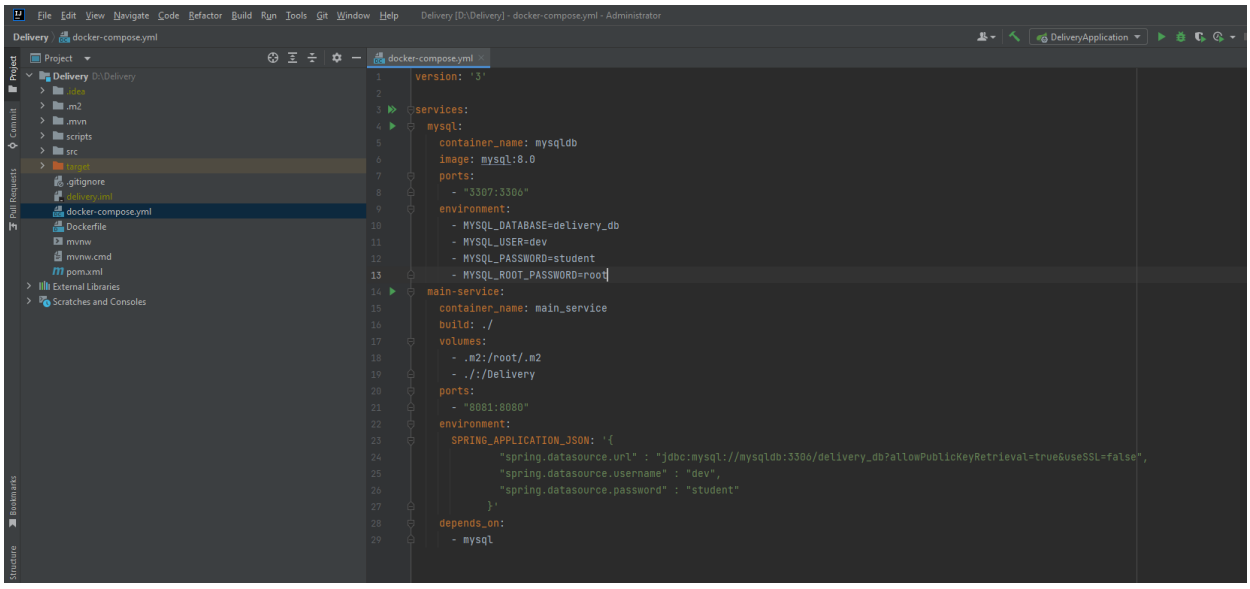


Рисунок 3.3 – Файл docker-compose.yml

Наступним етапом є створення бази даних, а саме таблиць та зв’язків між ними (рис. 3.4).

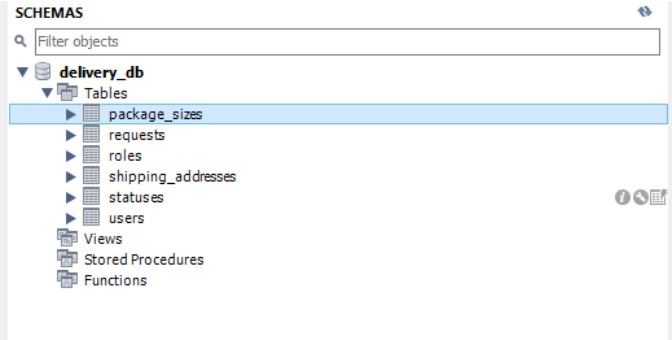


Рисунок 3.4 – Список таблиць бази даних

Для налагодження обробки запитів був створений клас MappingConfig (рис. 3.5), який відповідає за коректну обробку запитів у контролерах та вказування місця знаходження jsp-файлів для генерування відповіді користувачу.

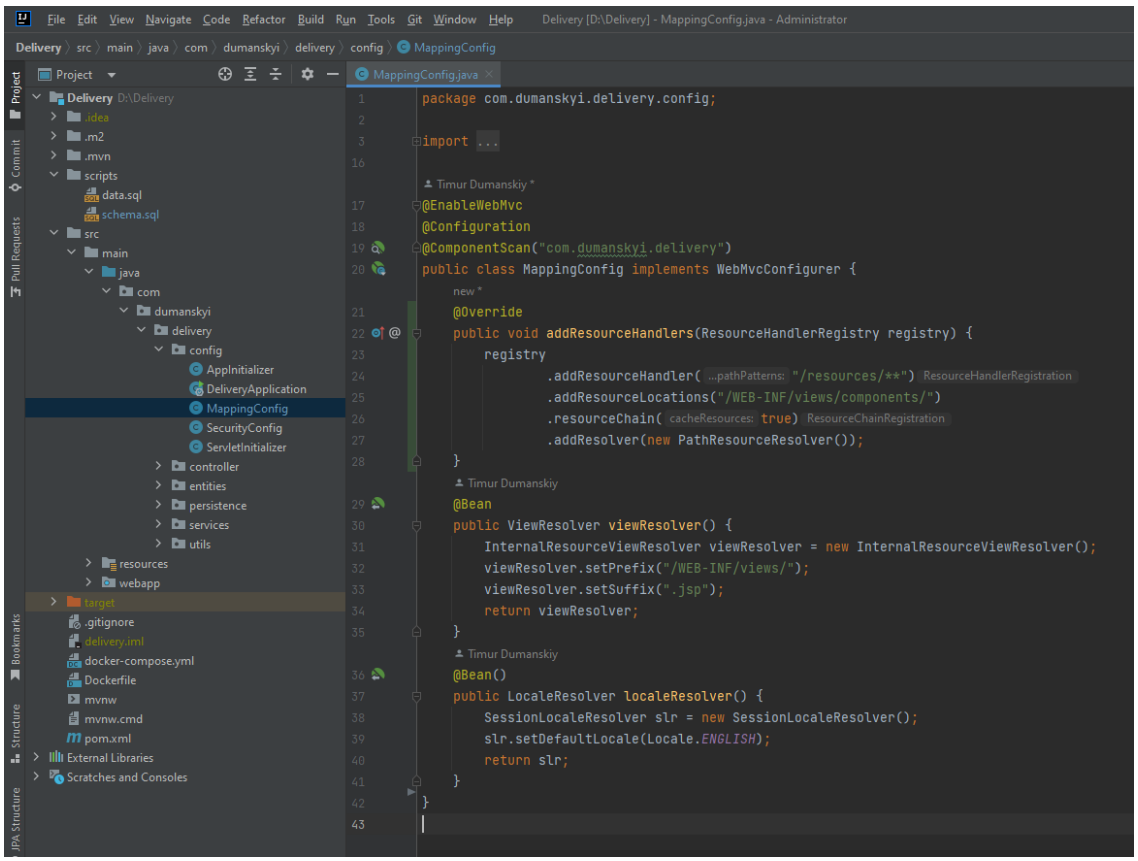


Рисунок 3.5 – Файл класу MappingConfig

Для реалізації аутентифікації користувачів було використано фреймворк Spring Security [18]. Для його конфігурації створено клас SecurityConfig представлений на рисунку 3.6.

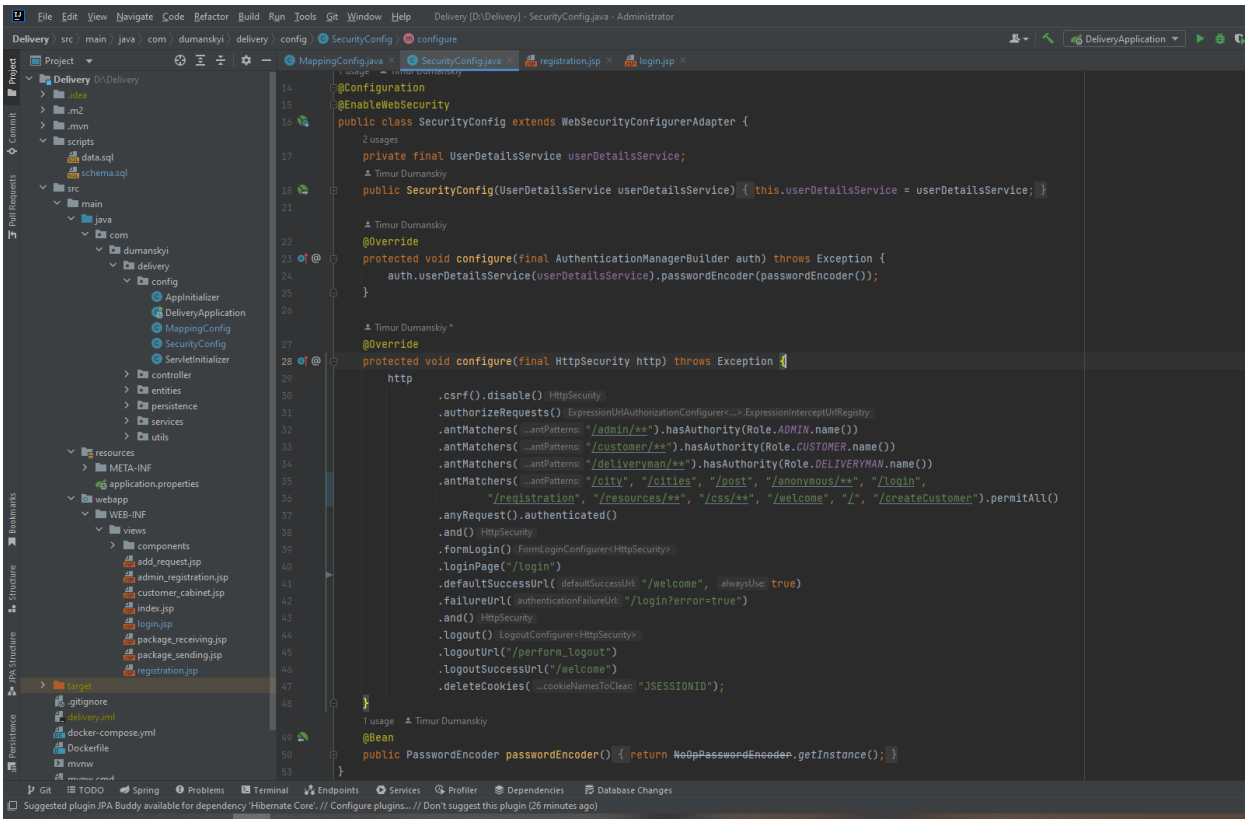


Рисунок 3.6 – Файл класу SecurityConfig

Для обробки запитів користувача було створено контролер представлений на рисунку 3.7.

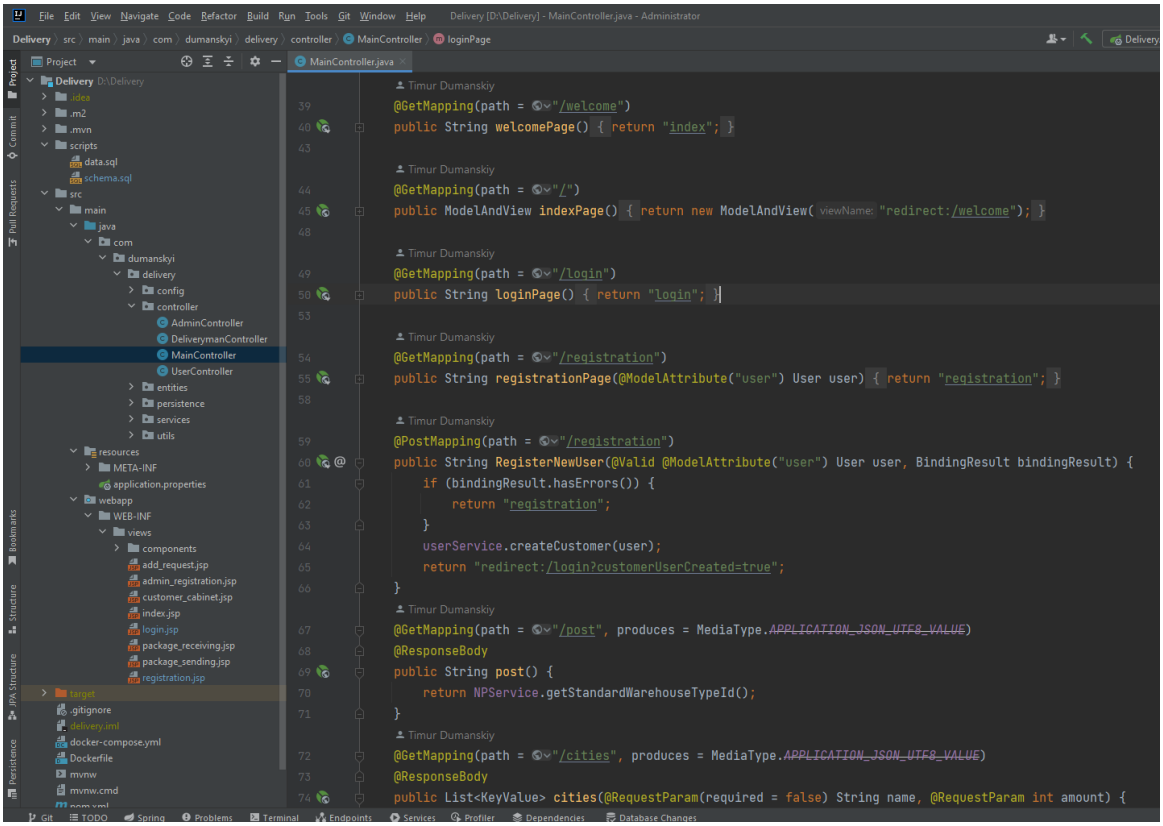


Рисунок 3.7 – Файл класу MainController

Для роботи з базою даних було використано фреймворк Spring JPA. На рисунку 3.8 представлено сутність User, поля якої відповідають полям в таблиці Users в базі даних.

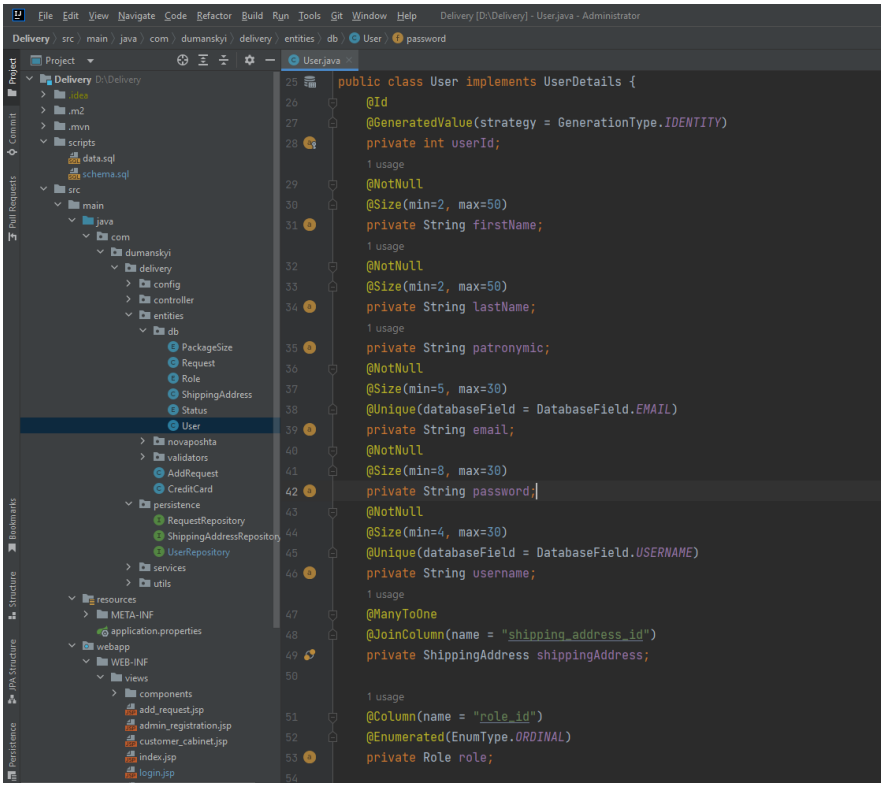


Рисунок 3.8 – Файл класу User

Для доступу до таблиці User було створено клас UserRepository, представлений на рисунку 3.9.

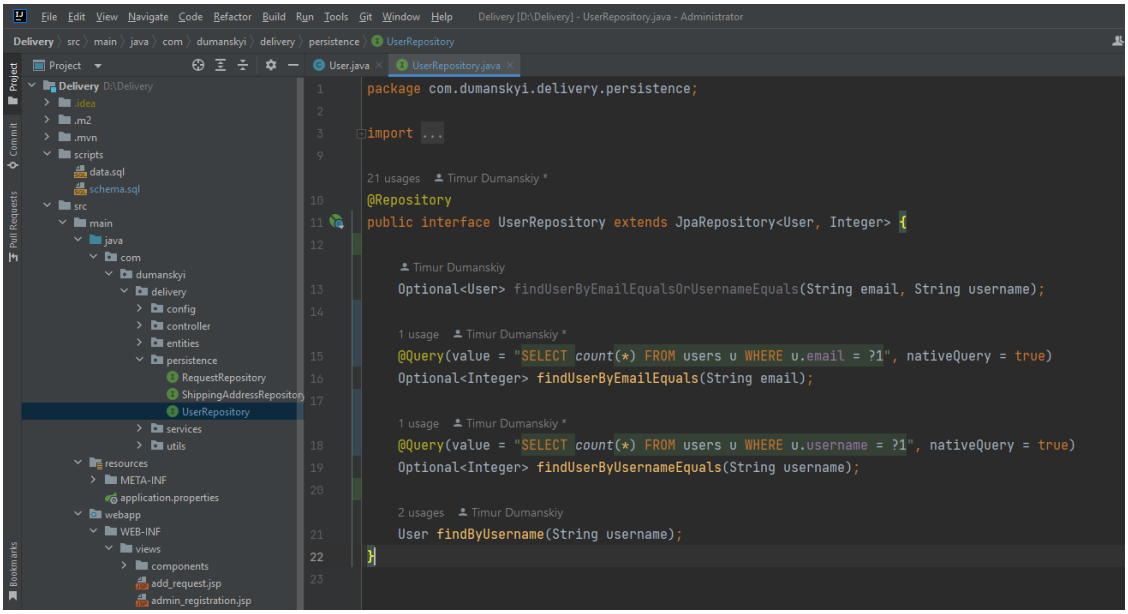


Рисунок 3.9 – Файл класу UserRepository

Наступним етапом є створення класу UserService (рис. 3.10), який відповідає за роботу з даними про користувачів.

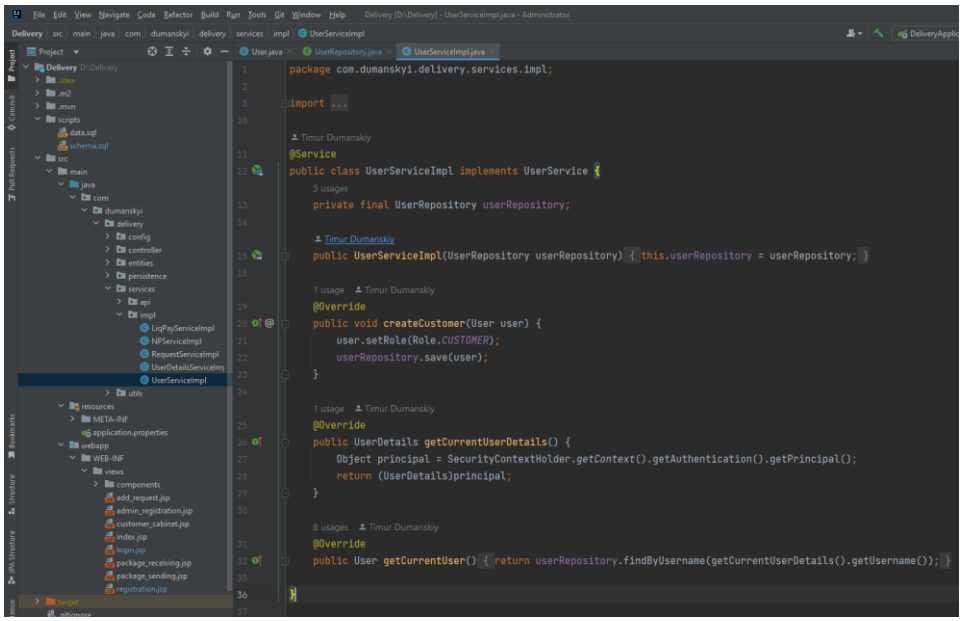


Рисунок 3.10 – Файл класу UserService

Сторінка для входу в даний web-додаток представлена файлом login.jsp (рис. 3.11).

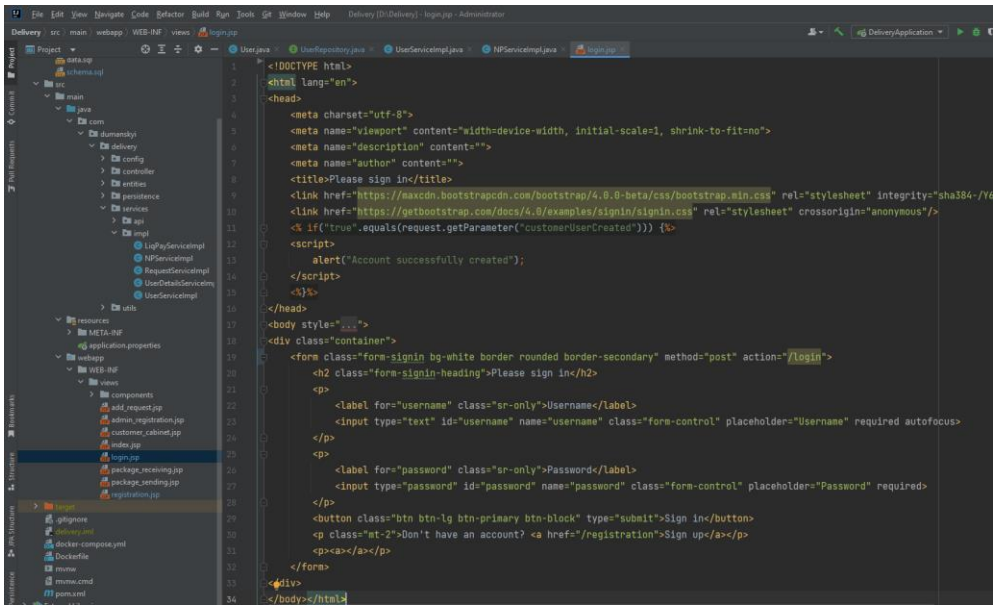


Рисунок 3.11 – Файл login.jsp

3.3 Використання web-додатку

Головна сторінка створеного програмного продукту для неавторизованого юзера містить інструкції користування ним. Також наявним є меню з назвою web-додатка та кнопками для входу та реєстрації (рис. 3.11).

Рисунок 3.11 – Головна сторінка для неавторизованого користувача

При натисканні на кнопку входу в систему, користувач переходить на сторінку для введення відповідних даних (рис. 3.12)

Рисунок 3.12 – Сторінка авторизації

При натисканні на іншу кнопку, відвідувач переходить на сторінку з формою для ведення даних для реєстрації (рис. 3.12).

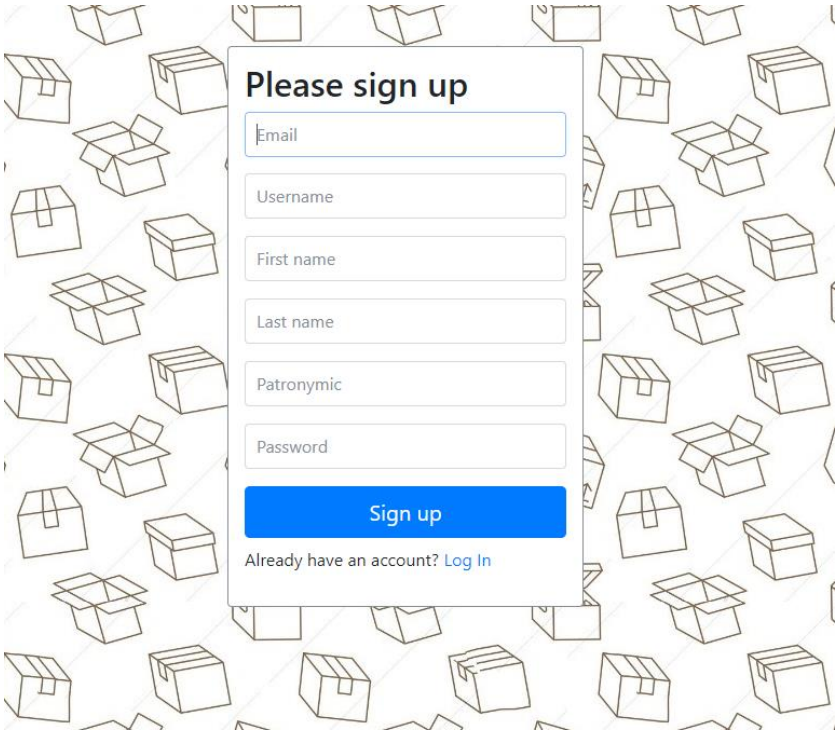


Рисунок 3.12 – Форма для реєстрації користувача

Якщо юзер авторизується під роллю «Адміністратор», він отримує можливість перейти на форму створення нового користувача з правами «Адміністратор» або «Персонал доставки» (рис. 3.13).

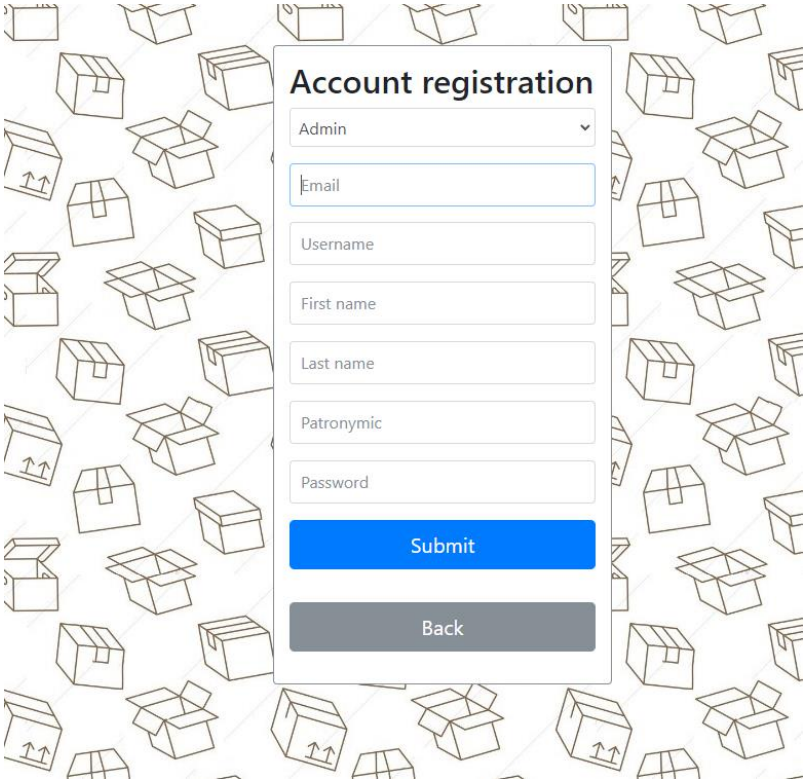


Рисунок 3.13 – Форма створення користувача для адміністратора

Якщо користувач авторизується під роллю «Клієнт», він має змогу перейти в особистий кабінет (рис. 3.14). Там надано можливості оновлення адреси доставки (рис. 3.15) та перегляду поточних запитів.

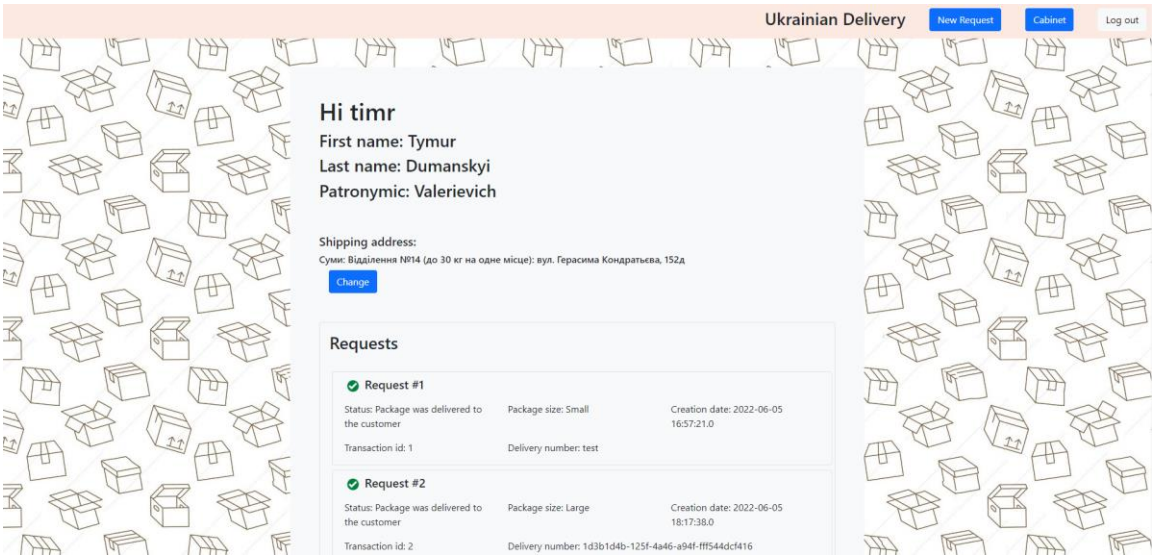


Рисунок 3.14 – Особистий кабінет клієнта

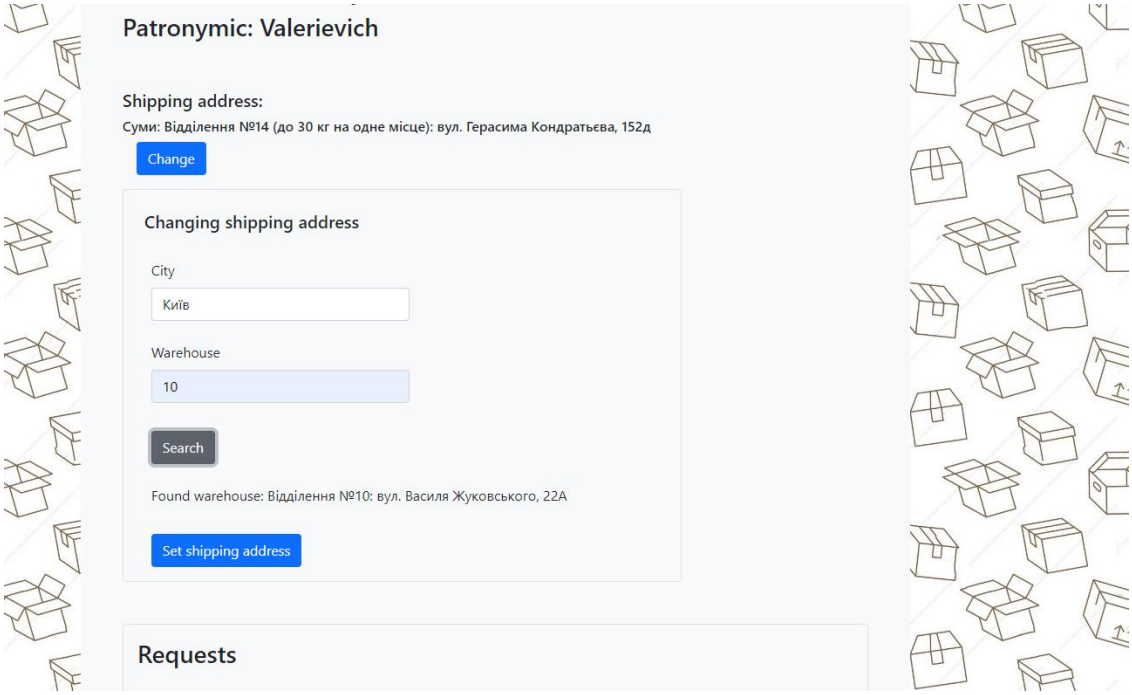


Рисунок 3.15 – Форма для оновлення адреси доставки

Також користувач має можливість перейти на сторінку додавання замовлення, яка представлена на рисунку 3.16.

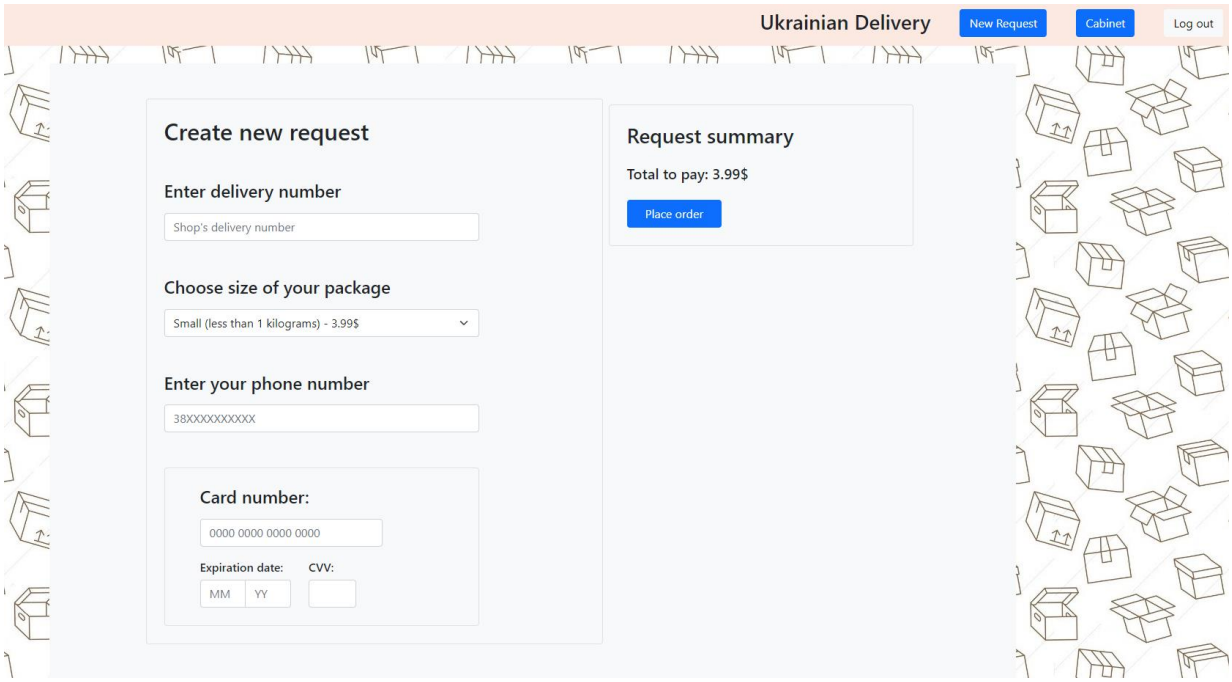


Рисунок 3.16 – Сторінка додавання замовлення

Після того, як клієнт розмістить замовлення, воно буде відображатися в його особистому кабінеті (рис. 3.17).

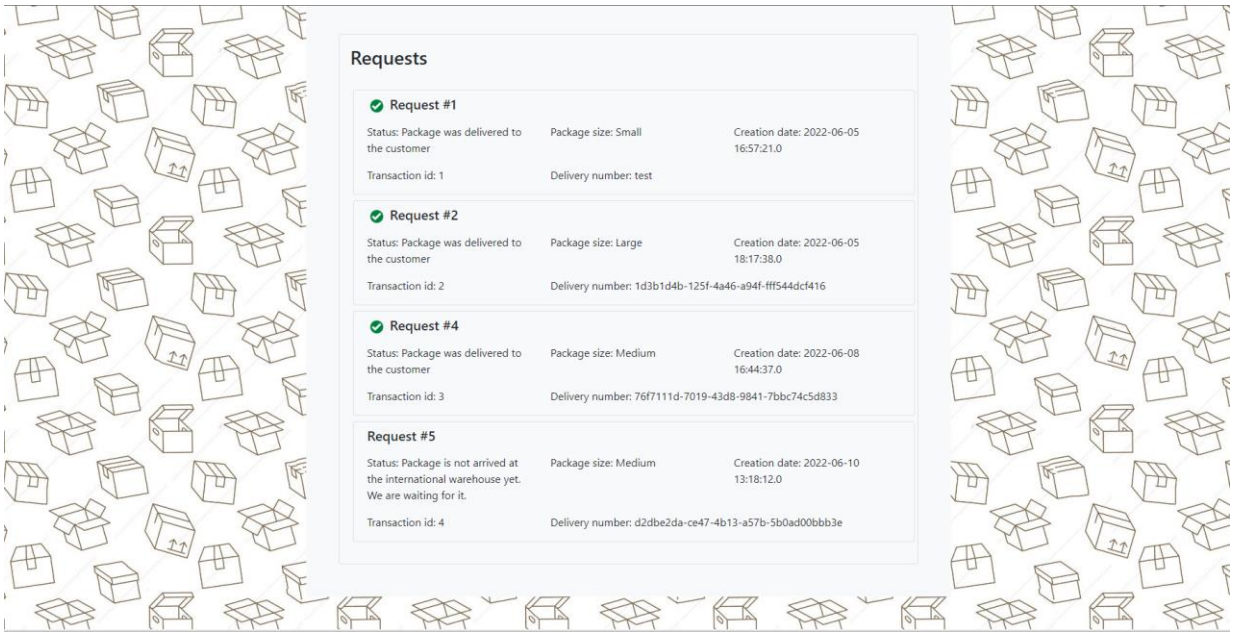


Рисунок 3.17 – Список замовлень клристувача

Наступним кроком є отримання посилки персоналом доставки. Це здійснюється на міжнародному складі. А також віжбувається зміна статуса посилки в меню її отримання (рис. 3.18).

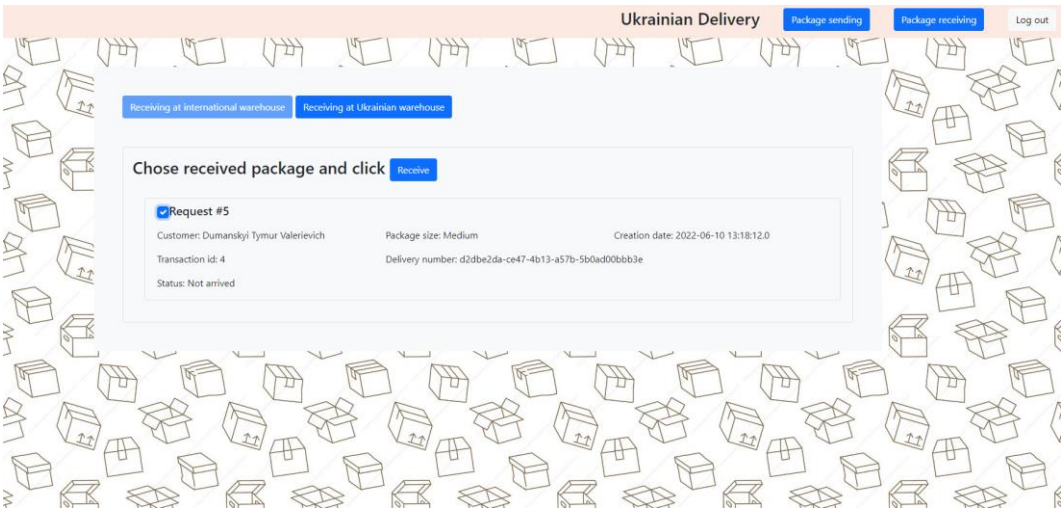


Рисунок 3.18 – Форма отримання посилки в міжнародному складі

Далі персонал доставки полвинен відправити посилку на склад в Україні та змінити її статус із «Arrived at the international warehouse» на «Sent to Ukraine» (рис. 3.19).

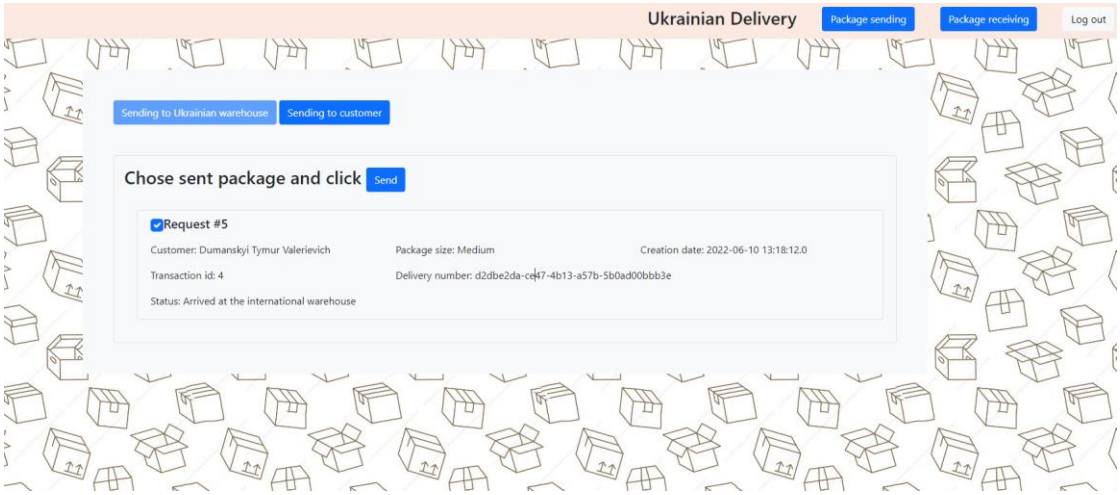


Рисунок 3.19 – Форма відправлення посилки на склад в Україні

Наступним кроком є зміна персоналом статусу посилки на «Arrived at the Ukrainian warehouse» (рис. 3.20).

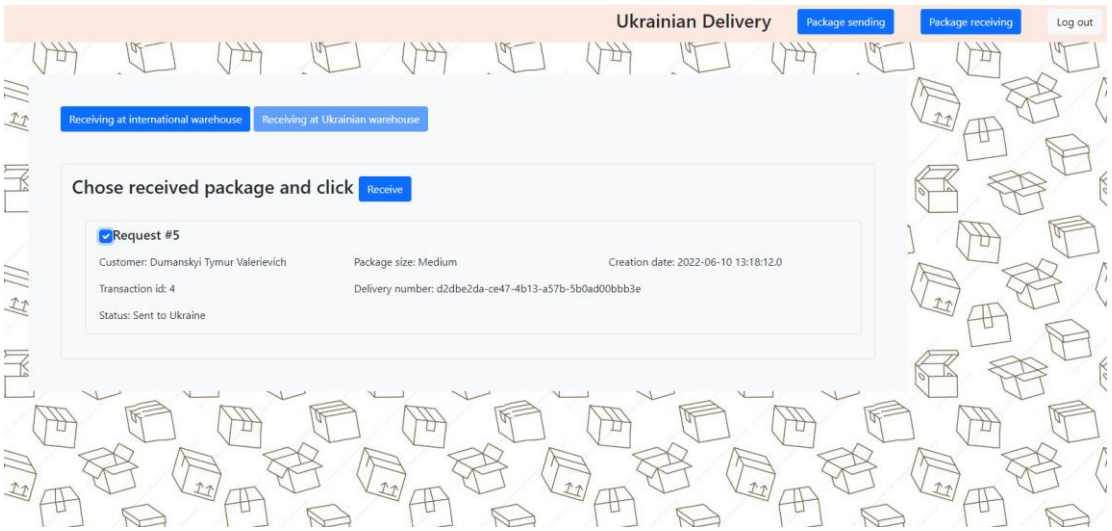


Рисунок 3.20 – Форма отримання посилки на склад в Україні

Завершальним кроком є відправка отриманої посилки персоналом доставки за адресою, яка була вказана користувачем при замовленні (рис. 3.21).

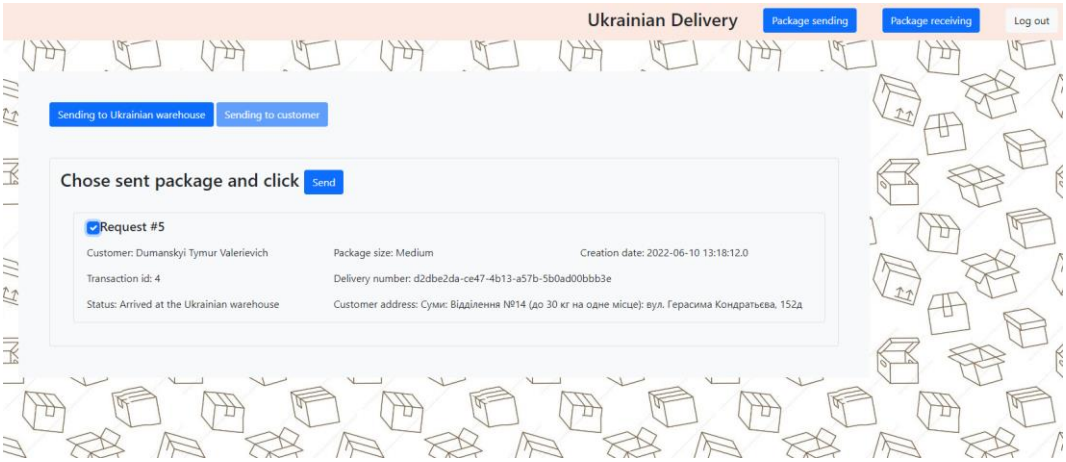


Рисунок 3.21 – Форма відправлення посилки користувачу
за адресою в Україні

Після зміни статусу замовлення на «Package was sent to the customer», клієнт має можливість отримати власне замовлення із-за кордону (рис. 3.22).

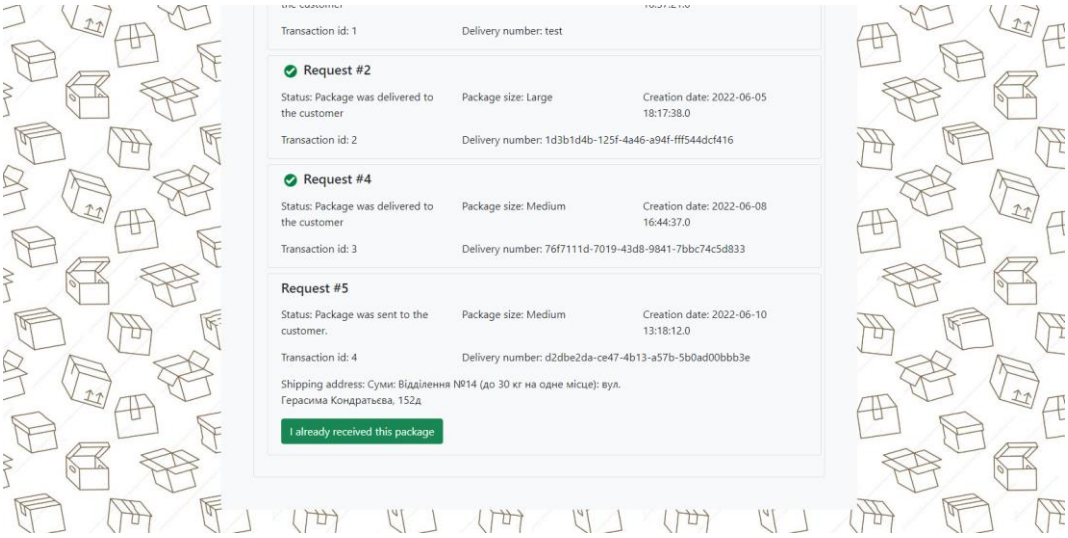


Рисунок 3.22 – Список замовлень користувача

Після отримання замовлення запит вважається завершеним (рис. 3.23).

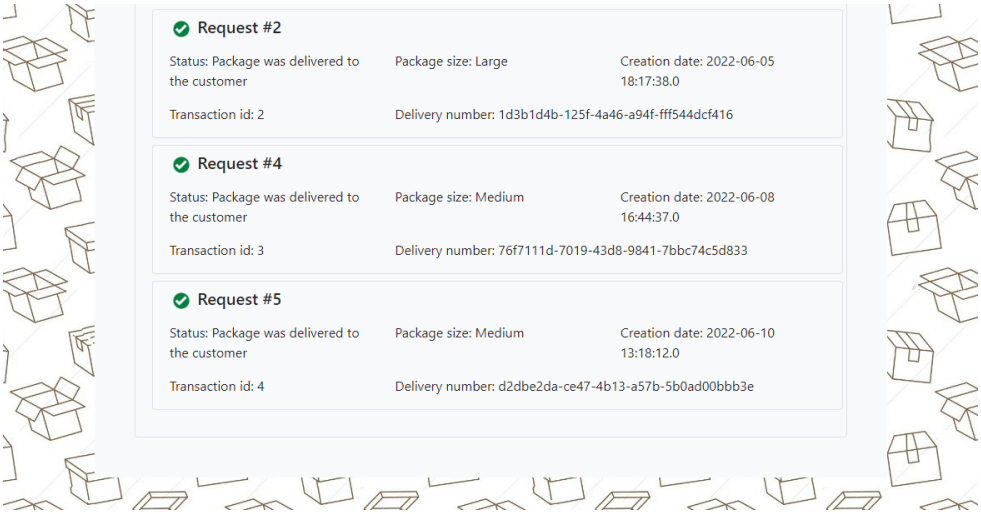
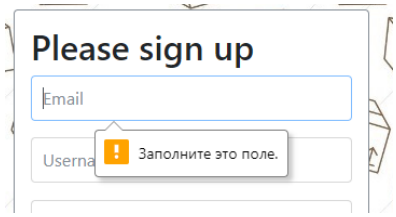
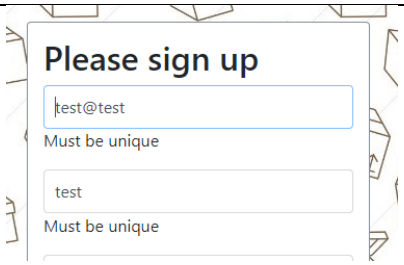
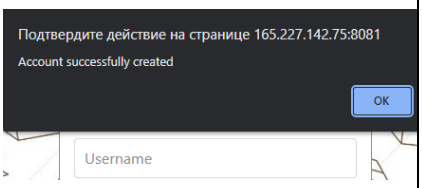
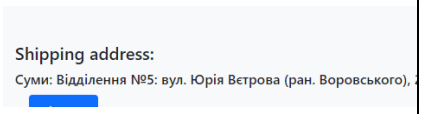
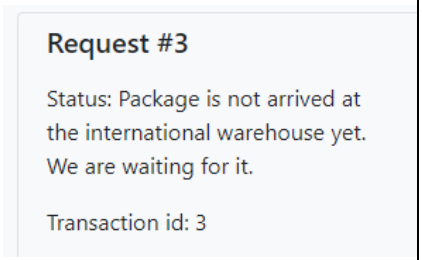


Рисунок 3.23 – Замовлення виконане

3.4 Тестування web-додатку

Останнім кроком розробки web-додатку є тестування його основних функціональних частин. Було перевірено етап створення облікового запису клієнта. Також було протестовано додавання та зміну адресу доставки клієнта. Далі перевірено створення нового запису клієнта й облікового запису адміністратора й персоналу доставки. Процес управління посилкою під час тестування збоїв не показав. Результат перевірки основних функцій даного web-додатку представлено у таблиці 3.1.

Таблиця 3.1 – Тестування функціоналу web-додатку

№	Назва	Очікуваний результат	Фактичний результат	0/1
1	Перевірка відправлення некоректних даних реєстрації	Обліковий запис не створено. Повідомлення про некоректні дані		1
2	Перевірка відправлення не унікальних даних реєстрації	Обліковий запис не створено. Повідомлення про неунікальність даних		1
3	Перевірка відправлення коректних даних реєстрації	Обліковий запис створено. Повідомлення про створення облікового запису		1
4	Перевірка форми додавання адреси доставки з коректними даними	Особистий кабінет оновлено. Присутня адреса доставки		1
5	Перевірка форми створення нового замовлення	Користувача переадресовано в особистий кабінет. Створене замовлення присутнє в списку		1

Згідно результатам вищезазначеної таблиці, тестування відбулось успішно. Багів виявлено не було.

ВИСНОВКИ

У результаті виконання даної кваліфікаційної роботи бакалавра було створено web-додаток організації замовлення товарів із-за кордону.

На початку реалізації проекту було проаналізовано предметну область. Визначено актуальність роботи та специфіку замовлень товарів із-за кордону. Проведено аналіз відповідних аналогів даного web-додатку, виявлено їх недоліки та переваги. Було сформовано мету та задачі проекту. Визначено основні функції, які повинен мати створюваний web-додаток. Був проведений аналіз засобів реалізації та обрано ті інструменти, які будуть використані. Сформовано технічне завдання на розробку даного проекту (Додаток А).

Розроблено графічне представлення web-додатку в нотаціях UML і IDEF0. Створено фізичну модель даних БД. Також було виконано планування робіт проекту (Додаток Б).

Розроблена архітектура web-додатку з використанням HLD діаграми. Представлена програмна реалізація головних елементів web-додатку. Також було протестовано його основний функціонал.

Повний лістинг коду модулів даної розробки наведено в додатку В.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Economic Forum [Електронний ресурс] – Режим доступу до ресурсу: https://www.weforum.org/agenda/2021/07/global-consumer-behaviour-trends-online-shopping/ (дата звернення: 25.05.2022).

2. Beginner’s Guide for WordPress [Електронний ресурс] – Режим доступу до ресурсу: https://www.wpbeginner.com/showcase/24-must-have-wordpress-plugins-for-business-websites/ (дата звернення: 25.05.2022).

3. AliExpress [Електронний ресурс] – Режим доступу до ресурсу: https://aliexpress.com/ (дата звернення: 20.12.2022).

4. Bunddler [Електронний ресурс] – Режим доступу до ресурсу: https://www.bunddler.com/ (дата звернення: 20.12.2021).

5. Meest [Електронний ресурс] – Режим доступу до ресурсу: https://ua.meest.com/ (дата звернення: 20.12.2021).

6. HTML Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/html/ (дата звернення: 25.05.2022).

7. CSS Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/css/ (дата звернення: 25.05.2022).

8. JavaScript Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/javascript/ (дата звернення: 25.05.2022).

9. MySQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/ (дата звернення: 25.05.2022).

10. Java Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/ (дата звернення: 25.05.2022).

11. Strojniški vestnik – Journal of Mechanical Engineering [Електронний ресурс] – Режим доступу до ресурсу: https://www.sv-

jme.eu/?ns_articles_pdf=/ns_articles/files/ojs3/1563/submission/1563-1-1898-1-2-20171103.pdf&id=4931 (дата звернення: 23.05.2022).

12. Oracle [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/technetwork/developer-tools/jdev/gettingstartedwithusecasemodeling-133857.pdf (дата звернення: 23.05.2022).

13. Sawakinome [Електронний ресурс] - Режим доступу до ресурсу: https://ua.sawakinome.com/articles/technology/difference-between-logical-and-physical-data-model-2.html (дата звернення: 25.12.2021)

14. High-Level Design Application [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/mandiri-engineering/high-level-design-application-2130642b9496 (дата звернення: 25.05.2022).

15. Docker Overview [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/get-started/overview/ (дата звернення: 25.05.2022).

16. Dockerfile reference [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/engine/reference/builder/ (дата звернення: 25.05.2022).

17. Use Docker Compose [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/get-started/08_using_compose/#:~:text=Docker%20Compose%20is%20a%20tool,or%20tear%20it%20all%20down (дата звернення: 25.05.2022).

18. Spring Security Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-security/reference/servlet/getting-started.html (дата звернення: 25.05.2022).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи

«Web-додаток організації замовлення товарів із-за кордону»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Антипенко В.П.

Студент групи ІТ-81

_____ Думанський Т.В.

2022

1 Основи для розробки

Розробка виконується за наказом на випускну роботу бакалавра.

2 Призначення розробки

Розробка повинна сформувати навички створення програмних продуктів, використовуючи знання з різних дисциплін і продемонструвати вміння формувати пакет документації, а також представляти результати виконаного проекту.

Тема проекту: «Web-додаток організації замовлення товарів із-за кордону».

3 Вимоги до програмного виробу

Web-додаток організації замовлення товарів із-за кордону повинен бути реалізований за допомогою web-інструментів та забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проекту має бути представлений web-додатком, який містить якісне інформаційне наповнення.

3.1 Вимоги до програмного продукту

Web-додаток організації замовлення товарів із-за кордону повинен бути реалізований за допомогою web-інструментів та забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проекту має бути представлений web-додатком, який містить якісне інформаційне наповнення.

bКористувачі не повинні мати особливих технічних навичок для роботи з web-додатком і його підтримкою. Єдиною вимогою є наявність навичок користування персональним комп’ютером та web-браузером.

Розроблюваний web-додаток має бути загальнодоступним у мережі Інтернет. Права доступу до інформації розмежовані за групами користувачів: адміністратор, персонал доставки та користувач. Адміністратор має необмежений доступ до даних користувачів та їх замовлень з правами перегляду, додавання, редагування та видалення.

Користувач може переглядати інформацію на web-сторінках з правом редагування особистої інформації та формування замовлення, оплачувати замовлення та його доставку.

Уся інформація надана на web-додаток повинна зберігатися у базі даних реалізованій засобами системи управління базами даних MySQL.

Для зручної навігації повинно бути створене меню, яке забезпечить швидке переміщення користувача по всім доступним сторінкам web-додатку. Меню має бути закріплене й розташовуватися зверху (у шапці) на кожній сторінці.

3.2 Вимоги до інформаційної та програмної сумісності

Увесь текст у web-додатку має бути виконаний українською мовою та англійською за потреби. Web-додаток має бути розміщений на хостингу. Для забезпечення стабільної роботи web-додатку web-браузер має бути Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

3.3 Вимоги до функціональних характеристик

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Реєстрація	Користувач
UN-02	Створення нового замовлення	Користувач
UN-03	Відправлення замовлення	Користувач
UN-04	Оплата замовлення	Користувач
UN-05	Редагування даних для доставки	Користувач, персонал доставки
UN-06	Створення нового акаунту для адміністратора та персоналу доставки	Адміністратор, персонал доставки
UN-07	Відправлення повідомлення про отримання замовлення	Персонал доставки
UN-08	Відправлення повідомлення про відправлення замовлення користувачу	Персонал доставки

Проаналізувавши потреби користувачів, додаток повинен забезпечувати виконання наступних функцій:

- перегляд активних замовлень;
- редагування особистих даних;
- оформлення замовлення обраного виробника товарів;
- оплата замовлення та доставки;
- нотифікація користувача про стан замовлення.

4 Перелік програмної документації

Розроблюваний web-додаток повинен супроводжуватися наступною документацією:

– специфікація – склад програми та документації на неї;

– текст програми – запис програми з необхідними коментарями;

– опис програми – відомості про логічну структуру і функціонуванні програми;

– програма і методика випробувань – вимоги, що підлягають перевірці при випробуванні програми, а також порядок і методи їх контролю;

– технічне завдання – призначення і область застосування програми, технічні, техніко-економічні та спеціальні вимоги, що пред'являються до програми, необхідні стадії і терміни розробки, види випробувань;

– пояснювальна записка – схема алгоритму, загальний опис алгоритму та (або) функціонування програми, а також обґрунтування прийнятих технічних і техніко-економічних рішень;

– експлуатаційні документи – відомості для забезпечення функціонування та експлуатації програми.

ДОДАТОК Б

Планування робіт

На сьогоднішній день неможливо уявити повсякденне життя без використання інтернет-технологій. Вони стали частиною багатьох сфер нашого життя, починаючи від роботи та навчання, закінчуючи розвагами та відпочинком. Через високу швидкість та доступність мережі Інтернет особливим попитом почав користується онлайн-шопінг. Останній передбачає значну економію часу покупців та зручність.

Проте, навіть у сучасному світі онлайн-покупок існують такі перешкоди як доставка товару з-за кордону. Часто в замовників виникає ряд прооблем з цим. Основною є те, що для доставки товару до їхньої країни мінімальна вартість замовлення виходить завеликою для бюджету. Існує варіант домовитися з іншими користувачами оформити спільне замовлення, але цей варіант є ризиковим та не завжди можливим.

Саме тому було вирішено створити web-додаток організації замовлення товарів із-за кордону. Його використання заощадить кошти покупця та повністю гарантує доставку замовлення.

Деталізацію мети проекту виконують за допомогою SMART-методу. SMART-метод дозволяє конкретизувати поставлені цілі проекту, обмежити в часі та деталізувати мету проекту. Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific	Заощадження коштів користувачів-замовників за доставку з міжнародних інтернет-магазинів
Measurable	Стабільно працюючий веб-додаток який формує спільне замовлення, розраховує вартість доставки між учасниками, нотифікує користувача, коли товар буде доставлено в Україну, а також що товар було відправлено українським перевізником на вказану користувачем адресу
Achievable	Мета досяжна, є затверджене технічне завдання, необхідні програмні засоби.
Relevant	Для організації замовлення товарів із-за кордону
Time-framed	Є конкретний термін – до кінця 4 курсу (10 червня 2022 р.).

Планування змісту робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічне представлення елементів проекту, кожний елемент якого згрупований ієрархією у єдиний проект. Структура декомпозиції робіт орієнтована на виконання робіт по частинам та розподіл робіт проекту між учасниками. Структура, окрім розділених робіт проекту, також вміщує в себе часові обмеження для виконання тієї, чи іншої роботи.

На найвищому (першому) рівень розміщений продукт проекту. Основні дії і етапи котрі необхідно виконати для завершення проекту знаходяться на другому рівні. Декомпозицію необхідно проводити до тих пір, поки вони не стануть елементарними, тобто матимуть однозначний чіткий результат, матимуть одну відповідальну особу та обмеження в часі. На рисунку Б.1 представлено WBS з розробки web-додатку організації замовлення товарів із-за кордону.

Планування структури виконавців. Після визначення структури робіт проекту, необхідно визначити відповідальних осіб для кожної елементарної задачі,

котрі були визначені в WBS. Для цього необхідно створити організаційну структуру виконавців або OBS. Вона має вигляд графічної структури з відображенням учасників які беруть участь у реалізації проекту.

На рисунку Б.2 представлено організаційну структуру планування проекту. Список виконавців, що функціонують в проекті описано в таблиці Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Думанський Т.В.	Виконує front-end та back-end розробку
Проектувальник	Думанський Т.В. Антипенко В.П.	Виконує проектування бази даних та розробляє структуру web-додатку.
Тестувальник	Думанський Т.В.	Відповідає за тестування функціоналу та дизайну web-додатку.
Керівник проекту	Антипенко В.П.	Формує завдання на розробку проекту.
Менеджер проекту	Думанський Т.В.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

Діаграма Ганта. Побудова календарного графіку (діаграма Ганта) є дуже важливим етапом планування проекту, оскільки завдяки цьому графіку можна отримати уявлення про тривалість процесів з урахуванням вихідних, та святкових днів. Календарний графік проекту представлено на рисунку Б.3.

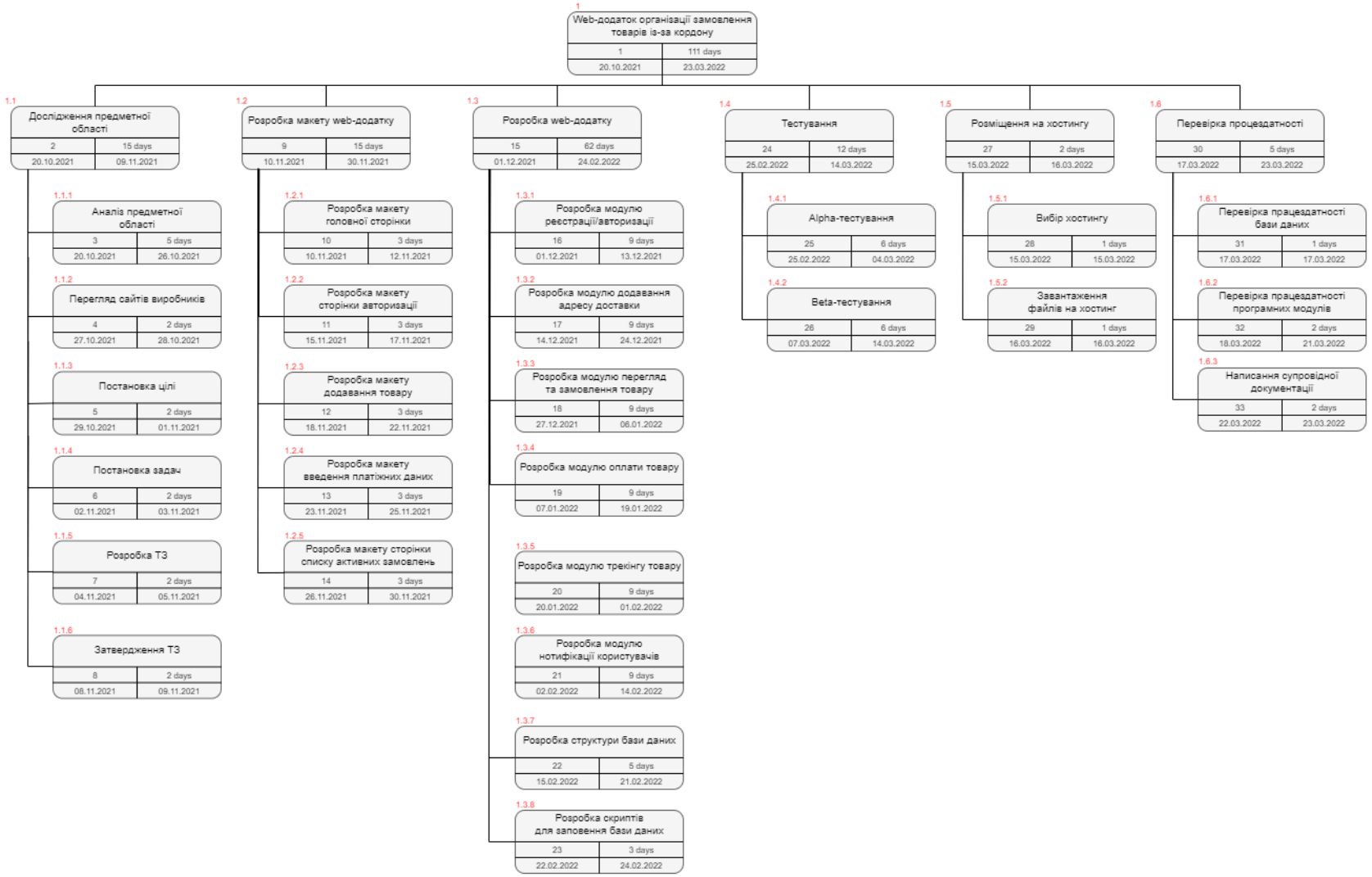


Рисунок Б.1 – WBS-структура робіт проекту

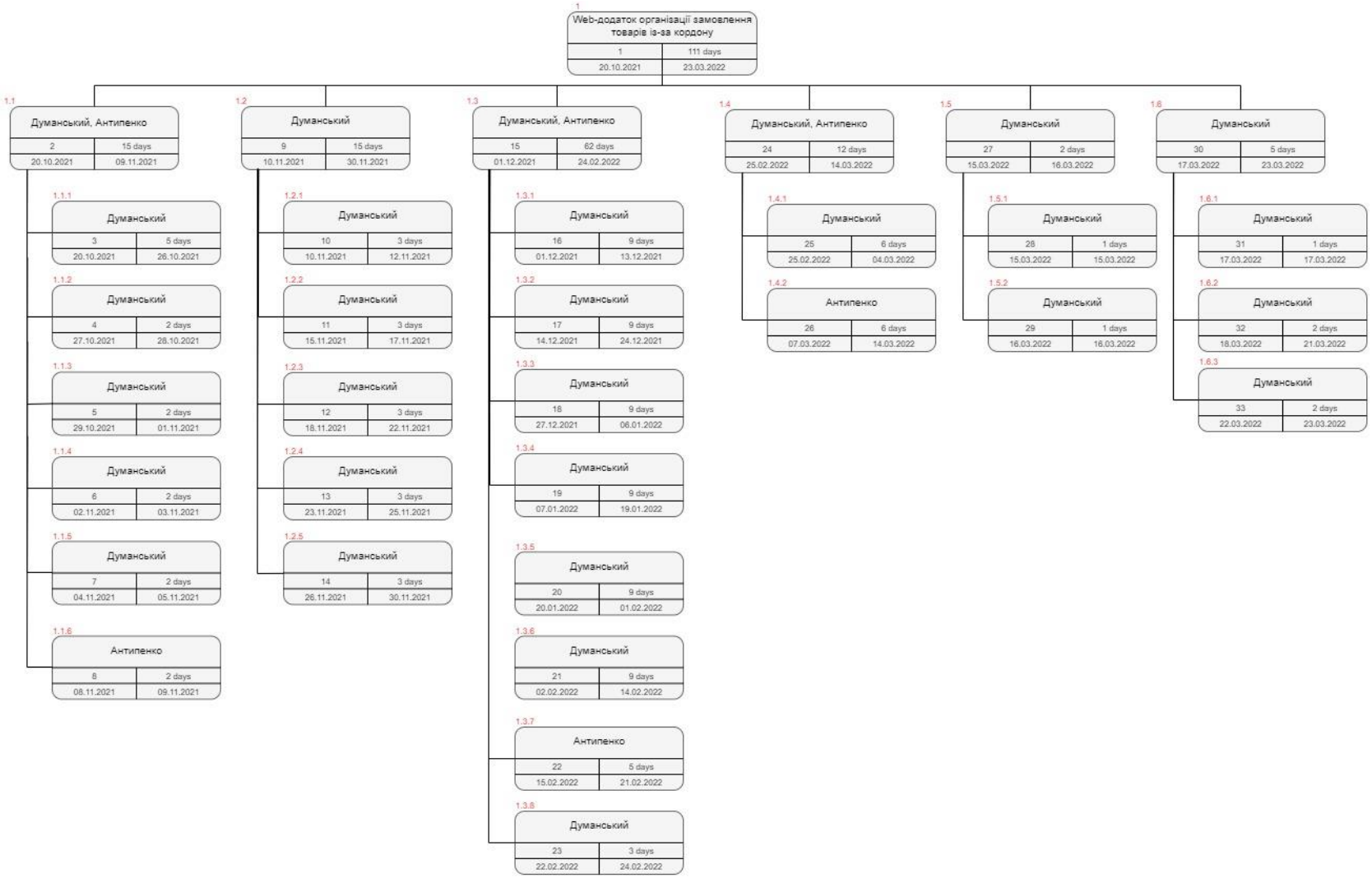


Рисунок Б.2 – OBS-структура робіт проекту

	Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предшественн
1		**Web-додаток організації замовлення товарів із-за кордону**	**111 дней**	**Ср 20.10.21**	**Ср 23.03.22**	
2		**Дослідження предметної області**	**15 дней**	**Ср 20.10.21**	**Вт 09.11.21**	
3		Аналіз предметної області	5 дней	Ср 20.10.21	Вт 26.10.21	
4		Перегляд сайтів виробників	2 дней	Ср 27.10.21	Чт 28.10.21	3
5		Постановка цілі	2 дней	Пт 29.10.21	Пн 01.11.21	4
6		Постановка задач	2 дней	Вт 02.11.21	Ср 03.11.21	5
7		Розробка ТЗ	2 дней	Чт 04.11.21	Пт 05.11.21	6
8		Затвердження ТЗ	2 дней	Пн 08.11.21	Вт 09.11.21	7
9		**Розробка макету web-додатку**	**15 дней**	**Ср 10.11.21**	**Вт 30.11.21**	**2**
10		Розробка макету головної сторінки	3 дней	Ср 10.11.21	Пт 12.11.21	
11		Розробка макету сторінки авторизації	3 дней	Пн 15.11.21	Ср 17.11.21	10
12		Розробка макету сторінки додавання товару	3 дней	Чт 18.11.21	Пн 22.11.21	11
13		Розробка макету сторінки введення платіжних даних	3 дней	Вт 23.11.21	Чт 25.11.21	12
14		Розробка макету сторінки списку активних замовлень	3 дней	Пт 26.11.21	Вт 30.11.21	13
15		**Розробка web-додатку**	**62 дней**	**Ср 01.12.21**	**Чт 24.02.22**	**9**
16		Розробка модулю реєстрації/авторизації	9 дней	Ср 01.12.21	Пн 13.12.21	
17		Розробка модулю додавання адресу доставки	9 дней	Вт 14.12.21	Пт 24.12.21	16
18		Розробка модулю перегляду та замовлення товару	9 дней	Пн 27.12.21	Чт 06.01.22	17
19		Розробка модулю оплати товару	9 дней	Пт 07.01.22	Ср 19.01.22	18
20		Розробка модулю трекінгу товару	9 дней	Чт 20.01.22	Вт 01.02.22	19
21		Розробка модулю нотифікації користувачів	9 дней	Ср 02.02.22	Пн 14.02.22	20
22		Розробка структури бази даних	5 дней	Вт 15.02.22	Пн 21.02.22	21
23		Розробка скриптів для заповення бази даних	3 дней	Вт 22.02.22	Чт 24.02.22	22
24		**Тестування**	**12 дней**	**Пт 25.02.22**	**Пн 14.03.22**	**15**
25		Alpha-тестування	6 дней	Пт 25.02.22	Пт 04.03.22	
26		Beta-тестування	6 дней	Пн 07.03.22	Пн 14.03.22	25
27		**Розміщення на хостингу**	**2 дней**	**Вт 15.03.22**	**Ср 16.03.22**	**24**
28		Вибір хостингу	1 день	Вт 15.03.22	Вт 15.03.22	
29		Завантаження файлів на хостинг	1 день	Ср 16.03.22	Ср 16.03.22	28
30		**Перевірка процездатності**	**5 дней**	**Чт 17.03.22**	**Ср 23.03.22**	**27**
31		Перевірка працездатності бази даних	1 день	Чт 17.03.22	Чт 17.03.22	
32		Перевірка працездатності програмних модулів	2 дней	Пт 18.03.22	Пн 21.03.22	31
33		Написання супровідної документації	2 дней	Вт 22.03.22	Ср 23.03.22	32

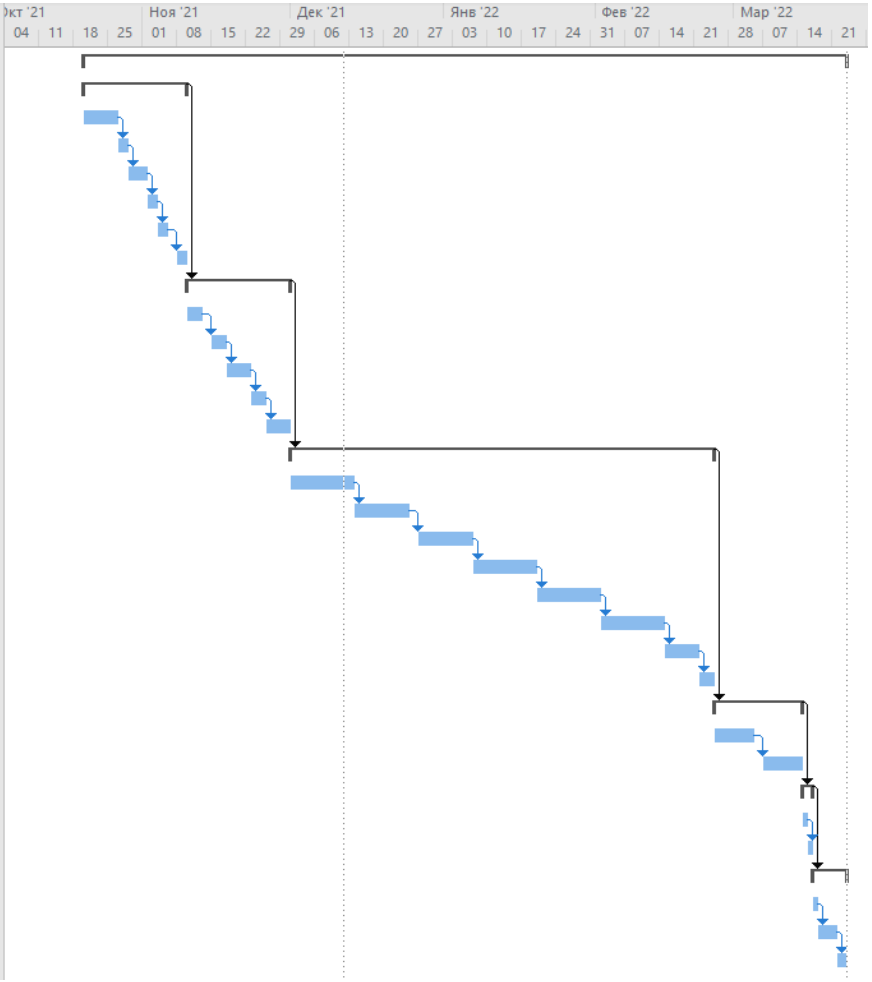


Рисунок Б.3 – Календарний графік проекту

Управління ризиками проекту. Під час виконання якісної оцінки ризиків треба визначити ризики, які мають бути усунені якнайшвидше. В залежності від ступеня важливості ризику – реагування буде відповідне. Наступним етапом є виконання кількісного оцінювання ризиків. Кількісне та якісне оцінювання можуть виконувати одночасно або окремо, що залежить від ступеня забезпечення проекту. У таблиці Б.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

Таблиця Б.3 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу.

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для того, щоб знизити негативний вплив ризиків на проект треба виконати планування реагування на них. До нього входить визначення ефективності розробки та оцінка наслідків впливу на проект. Оцінювання виконується за показниками, що описані в таблиці Б.3. У результаті планування реагування було отримано матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на таблиці Б.4. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.4. – Матриця ймовірності

Вплив	Ймовірність		
	Низька 1	Середня 2	Велика 3
Великий 3	RS_5		RS_9
Середній 2	RS_8, RS_10	RS_4, RS_7	RS_3
Низький 1	RS_1, RS_6	RS_2,	

Класифікація ризиків за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.5. У таблиці Б.6 описано ризики та стратегії реагування на кожен з них.

Таблиця Б.5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1, 2, 6, 8, 10
2	Виправдані	3≤R≤4	4, 5, 7
3	Недопустимі	6≤R≤9	3, 9

Таблиця Б.6 – Ризики та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	План А	План Б
RS_1	Відкритий	Непорозуміння з замовником	Низька	Низький	1	Налагодження контакту	Дотримання ділового етикету
RS_2	Відкритий	Низька кваліфікованість розробників	Середня	Низький	2	Переглянути необхідну літературу та онлайн ресурси	Використання більш знайомої технології для розробки
RS_3	Відкритий	Неоптимальний розподіл часу	Висока	Середній	6	Внести поправки в розподіл часу між задачами	Обговорити варіант внесення поправок до термінів виконання проекту.
RS_4	Відкритий	Внесення змін до ТЗ	Середня	Середній	4	Внести зміни в список робіт для виконання проекту	Обговорити варіант внесення поправок до термінів виконання проекту.
RS_5	Відкритий	Вибір неефективної технології розробки	Низька	Високий	3	Виділити час на пошук варіантів покращення вибраної технології	Обрати більш легку та зрозумілу технологію розробки
RS_6	Відкритий	Збої роботи хостингу	Низька	Низький	1	Виділити час на пошук кращого варіанту для хостингу	Скористатись платними аналогами

Продовження табл. Б.6

RS_7	Відкритий	Лікарняні	Середня	Середній	4	Обговорити варіант внесення поправок до термінів виконання проекту	Внести поправки в розподіл часу між задачами
RS_8	Відкритий	Технічні збої у розробника	Низька	Середній	2	Улагодження збоїв	Використання іншої машини для розробки
RS_9	Відкритий	Помилки в проектуванні	Висока	Високий	9	Здійснювати контроль результату на кожному етапі проектування	Обговорити варіант внесення поправок до термінів виконання проекту
RS_10	Відкритий	Відсутність резервних копій даних	Низька	Середній	2	Налаштувати автоматичне збереження даних.	Робити копію даних після кожного виконаного етапу.

ДОДАТОК В

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНИХ МОДУЛІВ WEB-ДОДАТКУ

AppInitializer.java

```
package com.dumanskyi.delivery.config;

import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.ContextLoaderListener;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;
import org.springframework.web.filter.DelegatingFilterProxy;

import javax.servlet.ServletContext;

public class AppInitializer implements WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext sc) {

        AnnotationConfigWebApplicationContext root = new
AnnotationConfigWebApplicationContext();
        root.register(SecurityConfig.class);

        sc.addListener(new ContextLoaderListener(root));

        sc.addFilter("securityFilter", new
DelegatingFilterProxy("springSecurityFilterChain"))
                .addMappingForUrlPatterns(null, false, "/*");
    }
}
```

DeliveryApplication.java

```
package com.dumanskyi.delivery.config;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication(scanBasePackages = "com.dumanskyi.delivery")
@ComponentScan({"com.dumanskyi.delivery"})
@EnableJpaRepositories(basePackages = "com.dumanskyi.delivery.persistence")
@EntityScan("com.dumanskyi.delivery.entities")
public class DeliveryApplication extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
        return builder.sources(DeliveryApplication.class);
    }
    public static void main(String[] args) {
        SpringApplication.run(DeliveryApplication.class, args);
    }
```

```
}
```

MappingConfig.java

```
package com.dumanskyi.delivery.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;
import org.springframework.web.servlet.resource.PathResourceResolver;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

import java.util.Locale;

@EnableWebMvc
@Configuration
@ComponentScan("com.dumanskyi.delivery")
public class MappingConfig implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry
                .addResourceHandler("/resources/**")
                .addResourceLocations("/WEB-INF/views/components/")
                .resourceChain(true)
                .addResolver(new PathResourceResolver());
    }
    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/views/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
    @Bean()
    public LocaleResolver localeResolver() {
        SessionLocaleResolver slr = new SessionLocaleResolver();
        slr.setDefaultLocale(Locale.ENGLISH);
        return slr;
    }
}
```

SecurityConfig.java

```
package com.dumanskyi.delivery.config;

import com.dumanskyi.delivery.entities.db.Role;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationM
anagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
```

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final UserDetailsService userDetailsService;
    public SecurityConfig(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @Override
    protected void configure(final AuthenticationManagerBuilder auth) throws Exception
{

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Override
    protected void configure(final HttpSecurity http) throws Exception {
        http
                .csrf().disable()
                .authorizeRequests()
                .antMatchers("/admin/**").hasAuthority(Role.ADMIN.name())
                .antMatchers("/customer/**").hasAuthority(Role.CUSTOMER.name())
                .antMatchers("/deliveryman/**").hasAuthority(Role.DELIVERYMAN.name())
                .antMatchers("/city", "/cities", "/post", "/anonymous/**", "/login",
                        "/registration", "/resources/**", "/css/**", "/welcome", "/",
"/createCustomer").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .defaultSuccessUrl("/welcome", true)
                .failureUrl("/login?error=true")
                .and()
                .logout()
                .logoutUrl("/perform_logout")
                .logoutSuccessUrl("/welcome")
                .deleteCookies("JSESSIONID");
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance();
    }
}
```

ServletInitializer.java

```
package com.dumanskyi.delivery.config;

import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application)
{
        return application.sources(DeliveryApplication.class);
    }

}
```

AdminController.java

```
package com.dumanskyi.delivery.controller;

import com.dumanskyi.delivery.entities.db.User;
```

```
import com.dumanskyi.delivery.persistence.UserRepository;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.validation.Valid;

@Controller
@RequestMapping("admin")
public class AdminController {
    private final UserRepository userRepository;

    public AdminController(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @GetMapping(path = "/registration")
    public String registrationPage(@ModelAttribute("user") User user) {
        return "admin_registration";
    }

    @PostMapping(path = "/registration")
    public String RegisterNewUser(@Valid @ModelAttribute("user") User user,
BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            return "admin_registration";
        }
        userRepository.save(user);
        return "redirect:/admin/registration?customerUserCreated=true";
    }
}
```

DeliveryController.java

```
package com.dumanskyi.delivery.controller;

import com.dumanskyi.delivery.persistence.RequestRepository;
import com.dumanskyi.delivery.services.api.NPService;
import com.dumanskyi.delivery.services.api.RequestService;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.util.List;

@Controller
@RequestMapping("/deliveryman")
public class DelivérymanController {
    private final RequestRepository requestRepository;
    private final NPService npService;
    private final RequestService requestService;

    public DelivérymanController(RequestRepository requestRepository, NPService
npService, RequestService requestService) {
        this.requestRepository = requestRepository;
        this.npService = npService;
        this.requestService = requestService;
    }

    @GetMapping("/package_receiving")
    public String packageReceivingPage(HttpServletRequest request) {
        request.setAttribute("requestRepository", requestRepository);
        return "package_receiving";
```

```
    }
    @GetMapping("/package_sending")
    public String packageSendingPage(HttpServletRequest request) {
        request.setAttribute("npService", npService);
        request.setAttribute("requestRepository", requestRepository);
        return "package_sending";
    }
    @PostMapping(value = "/updateRequestStatus")
    @ResponseBody
    public    String    packageReceiving(HttpServletRequest    request,    @RequestBody
List<Integer> requestIds) {
        requestService.updateRequestsStatus(requestIds);
        return "Cool";
    }
}
```

MainController.java

```
package com.dumanskyi.delivery.controller;

import com.dumanskyi.delivery.entities.db.User;
import com.dumanskyi.delivery.entities.novaposhta.responses.KeyValue;
import com.dumanskyi.delivery.services.api.NPService;
import com.dumanskyi.delivery.services.api.UserService;
import com.liqpay.LiqPay;
import org.hibernate.validator.constraints.CreditCardNumber;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import javax.validation.Valid;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Controller
public class MainController {
    private final UserService userService;
    private final NPService NPService;
    @Value("${liqpay.api.public-key}")
    private String publicKey;
    @Value("${liqpay.api.private-key}")
    private String privateKey;

    public MainController(UserService userService, NPService NPService) {
        this.userService = userService;
        this.NPService = NPService;
    }

    @GetMapping(path = "/welcome")
    public String welcomePage() {
        return "index";
    }

    @GetMapping(path = "/")
    public ModelAndView indexPage() {
        return new ModelAndView("redirect:/welcome");
    }
```

```
    @GetMapping(path = "/login")
    public String loginPage() {
        return "login";
    }

    @GetMapping(path = "/registration")
    public String registrationPage(@ModelAttribute("user") User user) {
        return "registration";
    }

    @PostMapping(path = "/registration")
    public String RegisterNewUser(@Valid @ModelAttribute("user") User user,
BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            return "registration";
        }
        userService.createCustomer(user);
        return "redirect:/login?customerUserCreated=true";
    }
    @GetMapping(path = "/post", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    @ResponseBody
    public String post() {
        return NPService.getStandardWarehouseTypeId();
    }
    @GetMapping(path = "/cities", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    @ResponseBody
    public List<KeyValue> cities(@RequestParam(required = false) String name,
@RequestParam int amount) {
        return NPService.getCitiesIdByName(name, amount);
    }
    @GetMapping(path = "/warehouses", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    @ResponseBody
    public List<KeyValue> warehouses(@RequestParam String city, int number) {
        return NPService.getWarehouseByCityIdAndNumber(city, number);
    }
    @GetMapping(path = "/test")
    @ResponseBody
    public String test() throws Exception {
        HashMap<String, String> params = new HashMap<String, String>();
        params.put("action", "p2p");
        params.put("version", "3");
        params.put("phone", "380503079905");
        params.put("amount", "1");
        params.put("currency", "USD");
        params.put("description", "Payment for service");
        params.put("order_id", "4");
        params.put("receiver_card", "5168757371045958");
        params.put("card", "5354322072921397");
        params.put("card_exp_month", "10");
        params.put("card_exp_year", "23");
        params.put("card_cvv", "950");
        LiqPay liqpay = new LiqPay(publicKey, privateKey);
        Map<String, Object> res = liqpay.api("request", params);
        System.out.println(res.get("status"));
        return res.toString();
    }
}
```

UserController.java

```
package com.dumanskyi.delivery.controller;

import com.dumanskyi.delivery.entities.AddRequest;
import com.dumanskyi.delivery.entities.db.PackageSize;
import com.dumanskyi.delivery.entities.db.ShippingAddress;
import com.dumanskyi.delivery.entities.db.User;
```

```
import com.dumanskyi.delivery.persistence.RequestRepository;
import com.dumanskyi.delivery.persistence.ShippingAddressRepository;
import com.dumanskyi.delivery.persistence.UserRepository;
import com.dumanskyi.delivery.services.api.NPService;
import com.dumanskyi.delivery.services.api.RequestService;
import com.dumanskyi.delivery.services.api.UserService;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.Optional;

@Controller
@RequestMapping("/customer")
public class UserController {

    private final UserRepository userRepository;
    private final ShippingAddressRepository addressRepository;
    private final UserService userService;
    private final NPService npService;
    private final RequestService requestService;
    private final RequestRepository requestRepository;

    public  UserController(UserRepository  userRepository,  ShippingAddressRepository
addressRepository,
                           UserService userService, NPService npService, RequestService
requestService, RequestRepository requestRepository) {
        this.userRepository = userRepository;
        this.addressRepository = addressRepository;
        this.userService = userService;
        this.npService = npService;
        this.requestService = requestService;
        this.requestRepository = requestRepository;
    }

    @GetMapping("/cabinet")
    public ModelAndView customerCabinet(HttpServletRequest request) {
        User user = userService.getCurrentUser();
        request.setAttribute("userRepository", userRepository);
        request.setAttribute("npService", npService);
        request.setAttribute("user", user);
        return new ModelAndView("customer_cabinet", "user", user);
    }
    @PostMapping(path = "/shipping-address")
    public String setShippingAddress(HttpServletRequest request, String warehouse_id) {
        System.out.println(request.getContextPath());
        User user = userService.getCurrentUser();
        Optional<ShippingAddress>              shippingAddressOptional              =
addressRepository.findShippingAddressByNpWarehouseId(warehouse_id);
        ShippingAddress shippingAddress;
        if (shippingAddressOptional.isPresent()) {
            shippingAddress = shippingAddressOptional.get();
        } else {
            shippingAddress = new ShippingAddress();
            shippingAddress.setNpWarehouseId(warehouse_id);
            addressRepository.save(shippingAddress);
        }
        user.setShippingAddress(shippingAddress);
        userRepository.flush();
        return "redirect:/customer/cabinet";
    }
    @GetMapping(path = "/request/add")
```

```
    public String addRequestPage(HttpServletRequest request, @ModelAttribute("request")
AddRequest addRequest) {
        request.setAttribute("userRepository", userRepository);
        request.setAttribute("npService", npService);
        request.setAttribute("user", userService.getCurrentUser());
        return "add_request";
    }
    @PostMapping(path = "/request/add")
    public       String       addRequest(HttpServletRequest       request,       @Valid
@ModelAttribute("request") AddRequest addRequest, BindingResult bindingResult,
                            @ModelAttribute("packageSizeId") int packageSizeId) {
        request.setAttribute("userRepository", userRepository);
        request.setAttribute("npService", npService);
        request.setAttribute("user", userService.getCurrentUser());
        if                          (bindingResult.hasErrors()                          ||
userService.getCurrentUser().getShippingAddress() == null) {
            return "add_request";
        }
        addRequest.setPackageSize(PackageSize.getById(packageSizeId));
        requestService.createNewRequest(addRequest);
        return "redirect:/customer/cabinet?request_created=true";
    }
    @PostMapping(path = "/request/finished")
    @ResponseBody
    public String finishRequest(@RequestBody int requestId) {
        requestService.finishRequest(requestRepository.getById(requestId));
        return "Cool";
    }
}
```

PackageSize.java

```
package com.dumanskyi.delivery.entities.db;

import lombok.Getter;

import java.util.Arrays;

@Getter
public enum PackageSize {
    SMALL(0, 1, 3.99f),
    MEDIUM(1, 3, 9.99f),
    LARGE(2, 10, 19.99f);
    private final int id;
    private final int maxWeight;
    private final float price;
    PackageSize(int id, int maxWeight, float price) {
        this.id = id;
        this.maxWeight = maxWeight;
        this.price = price;
    }
    public static PackageSize getById(int id) {
        return Arrays.stream(values())
                .filter(x -> x.getId() == id)
                .findAny()
                .get();
    }
}
```

Request.java

```
package com.dumanskyi.delivery.entities.db;

import lombok.*;

import javax.persistence.*;
```

```
import javax.validation.constraints.NotBlank;
import java.util.Date;

@Entity
@Table(name = "requests")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Request {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int requestId;
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    @Column(name = "status_id")
    @Enumerated(EnumType.ORDINAL)
    private Status status;

    @Column(name = "package_size_id")
    @Enumerated(EnumType.ORDINAL)
    private PackageSize packageSize;

    @Temporal(TemporalType.TIMESTAMP)
    private Date creationDate;
    @NotBlank
    private String deliveryNumber;
    private String transactionId;
    @ManyToOne
    @JoinColumn(name = "shipping_address_id")
    private ShippingAddress shippingAddress;
}
```

Role.java

```
package com.dumanskyi.delivery.entities.db;

import lombok.Getter;

@Getter
public enum Role {
    CUSTOMER(0),
    ADMIN(1),
    DELIVERYMAN(2);
    private final int id;
    Role(int id) {
        this.id = id;
    }
}
```

ShippingAddress.java

```
package com.dumanskyi.delivery.entities.db;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import java.util.List;

@Entity
@Table(name = "shipping_addresses")
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class ShippingAddress {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int shippingAddressId;
    private String npWarehouseId;
    @JsonIgnore
    @OneToMany(mappedBy = "shippingAddress", cascade = CascadeType.ALL)
    private List<User> users;
    @JsonIgnore
    @OneToMany(mappedBy = "shippingAddress", cascade = CascadeType.ALL)
    private List<Request> requests;
}
```

Status.java

```
package com.dumanskyi.delivery.entities.db;

import lombok.Getter;

@Getter
public enum Status {
    NOT_ARRIVED(0, "Not arrived", "Package is not arrived at the international warehouse
yet. We are waiting for it."),
    ARRIVED_AT_THE_INTERNATIONAL_WAREHOUSE(1, "Arrived at the international warehouse",
"Package was arrived at the international warehouse. We are going to send it to
Ukraine."),
    SENT_TO_UKRAINE(2, "Sent to Ukraine", "Package was sent to Ukrainian warehouse."),
    ARRIVED_AT_THE_UKRAINIAN_WAREHOUSE(3,"Arrived at the Ukrainian warehouse", "Package
was arrived at the Ukrainian warehouse. We are going to send it to the customer."),
    SENT_TO_CUSTOMER(4, "Sent to the customer", "Package was sent to the customer."),
    DELIVERED(5, "Delivered", "Package was delivered to the customer");
    private final int id;
    private final String shortMessage;
    private final String message;
    Status(int id, String shortMessage, String message) {
        this.id = id;
        this.shortMessage = shortMessage;
        this.message = message;
    }

}
```

User.java

```
package com.dumanskyi.delivery.entities.db;

import com.dumanskyi.delivery.entities.validators.annotations.DatabaseField;
import com.dumanskyi.delivery.entities.validators.annotations.Unique;
import lombok.*;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
```

```
import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import java.util.Collection;
import java.util.Collections;
import java.util.List;

@Entity
@Table(name = "users")
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
@EqualsAndHashCode
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userId;
    @NotNull
    @Size(min=2, max=50)
    private String firstName;
    @NotNull
    @Size(min=2, max=50)
    private String lastName;
    private String patronymic;
    @NotNull
    @Size(min=5, max=30)
    @Unique(databaseField = DatabaseField.EMAIL)
    private String email;
    @NotNull
    @Size(min=8, max=30)
    private String password;
    @NotNull
    @Size(min=4, max=30)
    @Unique(databaseField = DatabaseField.USERNAME)
    private String username;
    @ManyToOne
    @JoinColumn(name = "shipping_address_id")
    private ShippingAddress shippingAddress;

    @Column(name = "role_id")
    @Enumerated(EnumType.ORDINAL)
    private Role role;

    @OneToMany(mappedBy = "user")
    private List<Request> requestList;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        SimpleGrantedAuthority authority = new SimpleGrantedAuthority(role.name());
        return Collections.singleton(authority);
    }
    public String getFullName() {
        return lastName + " " + firstName + " " + patronymic;
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
```

```
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

NPRequestBody.java

```
package com.dumanskyi.delivery.entities.novaposhta.requests;

import lombok.*;
import org.springframework.beans.factory.annotation.Value;

import java.util.Map;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Builder
public class NPRequestBody {
    @Value("${novaposhta.api.key}")
    private static String apiKey;
    private String modelName;
    private String calledMethod;
    private Map<String, String> methodProperties;
}
```

KeyValue.java

```
package com.dumanskyi.delivery.entities.novaposhta.responses;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
public class KeyValue {
    private String key;
    private String value;
}
```

NPDataConstants.java

```
package com.dumanskyi.delivery.entities.novaposhta.responses;

public interface NPDataConstants {
    String DESCRIPTION = "Description";
    String DESCRIPTION_RU = "DescriptionRu";
    String REF = "Ref";
    String CITY_REF = "CityRef";
    String CITY_DESCRIPTION = "CityDescription";
    String FIND_BY_STRING = "FindByString";
    String PAGE = "Page";
```

```
    String LIMIT = "Limit";
    String WAREHOUSE_ID = "WarehouseId";
    String TYPE_WAREHOUSE_REF = "TypeOfWarehouseRef";

}
```

NPRequestBody.java

```
package com.dumanskyi.delivery.entities.novaposhta.responses;

import lombok.*;

import java.util.List;
import java.util.Map;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class NPResponseBody {
    private boolean success;
    private List<Map<String, Object>> data;
}
```

DataBaseField.java

```
package com.dumanskyi.delivery.entities.validators.annotations;

public enum DatabaseField {
    EMAIL("email"),
    USERNAME("username");
    final String value;
    DatabaseField(String value) {
        this.value = value;
    }
    public String getValue() {
        return value;
    }
}
```

Unique.java

```
package com.dumanskyi.delivery.entities.validators.annotations;

import com.dumanskyi.delivery.entities.validators.UniqueFieldValidator;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Documented
@Constraint(validatedBy = UniqueFieldValidator.class)
@Target( { ElementType.METHOD, ElementType.FIELD })
@Retention(RetentionPolicy.RUNTIME)
public @interface Unique {
    String message() default "Must be unique";
    DatabaseField databaseField() default DatabaseField.USERNAME;
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

UniqueFieldValidator.java

```
package com.dumanskyi.delivery.entities.validators;

import com.dumanskyi.delivery.entities.validators.annotations.Unique;
import com.dumanskyi.delivery.persistence.UserRepository;
import org.hibernate.AssertionFailure;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;
@Component
public class UniqueFieldValidator implements ConstraintValidator<Unique, String> {

    static private UserRepository userRepository;
    private Unique annotation;
    @Autowired
    public void setUserRepository(UserRepository userRepository) {
        UniqueFieldValidator.userRepository = userRepository;
    }


    @Override
    public void initialize(Unique constraintAnnotation) {
        annotation = constraintAnnotation;
    }

    @Override
    public      boolean      isValid(String      field,      ConstraintValidatorContext
constraintValidatorContext) {
        try {
            switch (annotation.databaseField()) {
                case USERNAME:
                    return userRepository.findUserByUsernameEquals(field).isEmpty();
                case EMAIL:
                    return userRepository.findUserByEmailEquals(field).isEmpty();
            }
            throw new IllegalArgumentException("There is no such field");
        } catch (AssertionFailure ignored) {
            return true;
        }
    }
}
```

AddRequest.java

```
package com.dumanskyi.delivery.entities;

import com.dumanskyi.delivery.entities.db.PackageSize;
import lombok.*;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class AddRequest {
    @NotNull
    @NotBlank
    private String deliveryNumber;
    private PackageSize packageSize;
```

```
    @Valid
    private CreditCard creditCard;
}
```

CreditCard.java

```
package com.dumanskyi.delivery.entities;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.validator.constraints.CreditCardNumber;

import javax.validation.constraints.Digits;
import javax.validation.constraints.Pattern;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class CreditCard {
    @Pattern(regexp = "\\d{16}", message = "invalid credit card number")
    private String number;
    @Digits(integer = 2, fraction = 0, message = "should contains 2 digits")
    private int expMonth;
    @Digits(integer = 2, fraction = 0, message = "should contains 2 digits")
    private int expYear;
    @Digits(integer = 3, fraction = 0, message = "should contains 3 digits")
    private int cvv;
    @Pattern(regexp="\\d{10,12}", message = "should contains less then 10 and more then
12")
    private String phoneNumber;
}
```

RequestRepository.java

```
package com.dumanskyi.delivery.persistence;

import com.dumanskyi.delivery.entities.db.Request;
import com.dumanskyi.delivery.entities.db.Status;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.util.List;
import java.util.Optional;

public interface RequestRepository extends JpaRepository<Request, Integer> {
    @Query(value = "SELECT MAX(r.transaction_id) FROM requests r", nativeQuery = true)
    Optional<Integer> findMaxTransactionId();
    List<Request> findRequestByStatus(Status status);
}
```

ShippingAddressRepository.java

```
package com.dumanskyi.delivery.persistence;

import com.dumanskyi.delivery.entities.db.ShippingAddress;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface ShippingAddressRepository extends JpaRepository<ShippingAddress,
Integer> {
    Optional<ShippingAddress> findShippingAddressByNpWarehouseId(String npWarehouseId);
```

```
}
```

UserRepository.java

```
package com.dumanskyi.delivery.persistence;

import com.dumanskyi.delivery.entities.db.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User, Integer> {

    Optional<User>    findUserByEmailEqualsOrUsernameEquals(String    email,    String
username);

    @Query(value = "SELECT count(*) FROM users u WHERE u.email = ?1", nativeQuery =
true)
    Optional<Integer> findUserByEmailEquals(String email);

    @Query(value = "SELECT count(*) FROM users u WHERE u.username = ?1", nativeQuery =
true)
    Optional<Integer> findUserByUsernameEquals(String username);

    User findByUsername(String username);
}
```

LiqPayService.java

```
package com.dumanskyi.delivery.services.api;

import com.dumanskyi.delivery.entities.AddRequest;

public interface LiqPayService {
    String processRequest(AddRequest request);
}
```

NPService.java

```
package com.dumanskyi.delivery.services.api;

import com.dumanskyi.delivery.entities.novaposhta.responses.KeyValue;

import java.util.List;
import java.util.Map;

public interface NPService {
    String getStandardWarehouseTypeId();
    List<KeyValue> getCitiesIdByName(String name, int amount);
    List<KeyValue> getWarehouseByCityId(String id, int amount);
    List<KeyValue> getWarehouseByCityId(String id);
    List<KeyValue> getWarehouseByCityIdAndNumber(String id, int number);
    List<KeyValue> getWarehouseById(String id);
    List<KeyValue> getCityByWarehouseId(String id);

    String getWarehouseFullAddressById(String id);
}
```

RequestService.java

```
package com.dumanskyi.delivery.services.api;

import com.dumanskyi.delivery.entities.AddRequest;
import com.dumanskyi.delivery.entities.db.Request;
```

```
import java.util.List;

public interface RequestService {
    void createNewRequest(AddRequest input);
    void updateRequestsStatus(List<Integer> ids);
    void finishRequest(Request request);
}
```

UserService.java

```
package com.dumanskyi.delivery.services.api;

import com.dumanskyi.delivery.entities.db.User;
import org.springframework.security.core.userdetails.UserDetails;

public interface UserService {
    void createCustomer(User user);
    UserDetails getCurrentUserDetails();
    User getCurrentUser();
}
```

LiqPayServiceImpl.java

```
package com.dumanskyi.delivery.services.impl;

import com.dumanskyi.delivery.entities.AddRequest;
import com.dumanskyi.delivery.persistence.RequestRepository;
import com.dumanskyi.delivery.services.api.LiqPayService;
import com.liqpay.LiqPay;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

@Service
public class LiqPayServiceImpl implements LiqPayService {
    @Value("${liqpay.api.public-key}")
    private String publicKey;
    @Value("${liqpay.api.private-key}")
    private String privateKey;
    @Value("${liqpay.card-receiver}")
    private String receiverCard;
    private final RequestRepository requestRepository;

    public LiqPayServiceImpl(RequestRepository requestRepository) {
        this.requestRepository = requestRepository;
    }

    @Override
    public String processRequest(AddRequest request) {
        Optional<Integer>                  maxTransactionIdOptional                  =
requestRepository.findMaxTransactionId();
        String transactionId = maxTransactionIdOptional
                .map(integer -> Integer.toString(integer + 1))
                .orElse("1");
        HashMap<String, String> params = new HashMap<String, String>();
        params.put("action", "p2p");
        params.put("version", "3");
        params.put("phone", request.getCreditCard().getPhoneNumber());
        params.put("amount", Float.toString(request.getPackageSize().getPrice()));
        params.put("currency", "USD");
        params.put("description", "Payment for service");
        params.put("order_id", transactionId);
```

```
        params.put("receiver_card", receiverCard);
        params.put("card", request.getCreditCard().getNumber());
        params.put("card_exp_month",
Integer.toString(request.getCreditCard().getExpMonth()));
        params.put("card_exp_year",
Integer.toString(request.getCreditCard().getExpYear()));
        params.put("card_cvv", Integer.toString(request.getCreditCard().getCvv()));
        LiqPay liqpay = new LiqPay(publicKey, privateKey);
        Map<String, Object> res;
        try {
            res = liqpay.api("request", params);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return transactionId;
    }
}
```

NPServiceImpl.java

```
package com.dumanskyi.delivery.services.impl;

import com.dumanskyi.delivery.entities.novaposhta.requests.NPRequestBody;
import com.dumanskyi.delivery.entities.novaposhta.responses.KeyValue;
import com.dumanskyi.delivery.entities.novaposhta.responses.NPDataConstants;
import com.dumanskyi.delivery.entities.novaposhta.responses.NPResponseBody;
import com.dumanskyi.delivery.services.api.NPService;
import com.dumanskyi.delivery.utils.NPRequestBodyUtil;
import com.dumanskyi.delivery.utils.RequestsUtil;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

@Service
public class NPServiceImpl implements NPService {
    @Value("${novaposhta.api.entry-point.json}")
    private String apiUri;

    @Override
    public String getStandardWarehouseTypeId() {
        NPRequestBody requestBody = NPRequestBodyUtil.getWarehouseTypes();
        NPResponseBody types = RequestsUtil.sendRequestNP(apiUri, requestBody);

        return (String)types.getData().stream()
                .filter(type                          ->                          "Поштове
відділення".equals(type.get(NPDataConstants.DESCRIPTION)))
                .findAny()
                .orElseThrow(() -> new IllegalStateException("Can't find Post type"))
                .get(NPDataConstants.REF);
    }

    @Override
    public List<KeyValue> getCitiesIdByName(String name, int amount) {
        NPRequestBody requestBody;
        if (name != null && !name.isEmpty()) {
            requestBody = NPRequestBodyUtil.getCitiesByName(name, amount);
        } else {
            requestBody = NPRequestBodyUtil.getAllCities(amount);
        }
        NPResponseBody responseBody = RequestsUtil.sendRequestNP(apiUri, requestBody);
        return requestBodyToKeyValueMap(responseBody);
    }
```

```
    @Override
    public List<KeyValue> getWarehouseByCityId(String id, int amount) {
        NPRequestBody    requestBody    =    NPRequestBodyUtil.getWarehousesByCityId(id,
amount);
        NPResponseBody responseBody = RequestsUtil.sendRequestNP(apiUri, requestBody);
        return requestBodyToKeyValueMap(responseBody);
    }

    @Override
    public List<KeyValue> getWarehouseByCityId(String id) {
        NPRequestBody    requestBody    =    NPRequestBodyUtil.getWarehousesByCityId(id,
getStandardWarehouseTypeId());
        NPResponseBody responseBody = RequestsUtil.sendRequestNP(apiUri, requestBody);
        return requestBodyToKeyValueMap(responseBody);
    }

    @Override
    public List<KeyValue> getWarehouseByCityIdAndNumber(String id, int number) {
        NPRequestBody                              requestBody                              =
NPRequestBodyUtil.getWarehousesByCityIdAndNumber(id, number);
        NPResponseBody responseBody = RequestsUtil.sendRequestNP(apiUri, requestBody);
        return requestBodyToKeyValueMap(responseBody);
    }

    @Override
    public List<KeyValue> getWarehouseById(String id) {
        NPRequestBody requestBody = NPRequestBodyUtil.getWarehousesById(id);
        NPResponseBody responseBody = RequestsUtil.sendRequestNP(apiUri, requestBody);
        return requestBodyToKeyValueMap(responseBody);
    }

    @Override
    public List<KeyValue> getCityByWarehouseId(String id) {
        NPRequestBody requestBody = NPRequestBodyUtil.getWarehousesById(id);
        NPResponseBody responseBody = RequestsUtil.sendRequestNP(apiUri, requestBody);
        List<KeyValue> keyValues = new ArrayList<>();
        for (Map<String, Object> data : responseBody.getData()) {
            keyValues.add(new     KeyValue((String)data.get(NPDataConstants.CITY_REF),
(String)data.get(NPDataConstants.CITY_DESCRIPTION)));
        }
        return keyValues;
    }

    @Override
    public String getWarehouseFullAddressById(String id) {
        KeyValue city = getCityByWarehouseId(id).get(0);
        KeyValue warehouse = getWarehouseById(id).get(0);
        return city.getValue() + ": " + warehouse.getValue();
    }

    private List<KeyValue> requestBodyToKeyValueMap(NPResponseBody responseBody) {
        List<KeyValue> keyValues = new ArrayList<>();
        for (Map<String, Object> data : responseBody.getData()) {
            keyValues.add(new          KeyValue((String)data.get(NPDataConstants.REF),
(String)data.get(NPDataConstants.DESCRIPTION)));
        }
        return keyValues;
    }
}
```

RequestServiceImpl.java

```
package com.dumanskyi.delivery.services.impl;

import com.dumanskyi.delivery.entities.AddRequest;
import com.dumanskyi.delivery.entities.db.Request;
```

```
import com.dumanskyi.delivery.entities.db.Status;
import com.dumanskyi.delivery.persistence.RequestRepository;
import com.dumanskyi.delivery.services.api.LiqPayService;
import com.dumanskyi.delivery.services.api.RequestService;
import com.dumanskyi.delivery.services.api.UserService;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.List;

@Service
public class RequestServiceImpl implements RequestService {

    private final RequestRepository requestRepository;
    private final UserService userService;
    private final LiqPayService liqPayService;

    public    RequestServiceImpl(RequestRepository    requestRepository,    UserService
userService, LiqPayService liqPayService) {
        this.requestRepository = requestRepository;
        this.userService = userService;
        this.liqPayService = liqPayService;
    }

    @Override
    public void createNewRequest(AddRequest input) {
        Request request = Request.builder()
                .creationDate(new Date())
                .deliveryNumber(input.getDeliveryNumber())
                .packageSize(input.getPackageSize())
                .status(Status.NOT_ARRIVED)
                .user(userService.getCurrentUser())
                .transactionId(liqPayService.processRequest(input))
                .build();
        request.setCreationDate(new Date());
        request.setUser(userService.getCurrentUser());
        requestRepository.save(request);
    }

    @Override
    public void updateRequestsStatus(List<Integer> ids) {
        List<Request> requests = requestRepository.findAllById(ids);
        for (Request request: requests) {
            switch (request.getStatus()) {
                case NOT_ARRIVED:
                    request.setStatus(Status.ARRIVED_AT_THE_INTERNATIONAL_WAREHOUSE);
                    break;
                case ARRIVED_AT_THE_INTERNATIONAL_WAREHOUSE:
                    request.setStatus(Status.SENT_TO_UKRAINE);
                    break;
                case SENT_TO_UKRAINE:
                    request.setStatus(Status.ARRIVED_AT_THE_UKRAINIAN_WAREHOUSE);
                    break;
                case ARRIVED_AT_THE_UKRAINIAN_WAREHOUSE:

request.setShippingAddress(request.getUser().getShippingAddress());
                    request.setStatus(Status.SENT_TO_CUSTOMER);
                    break;
            }
        }
        requestRepository.flush();
    }

    @Override
    public void finishRequest(Request request) {
```

```
        if           (!request.getStatus().equals(Status.SENT_TO_CUSTOMER)           &&
!request.getUser().equals(userService.getCurrentUser())) {
            throw new IllegalStateException("Request cannot be finished");
        }
        request.setStatus(Status.DELIVERED);
        requestRepository.flush();
    }

}
```

UserDetailsServiceImpl.java

```
package com.dumanskyi.delivery.services.impl;

import com.dumanskyi.delivery.entities.db.User;
import com.dumanskyi.delivery.persistence.UserRepository;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    private final static String USER_NOT_FOUND_MSG = "User with email %s not found";
    private final UserRepository userRepository;

    public UserDetailsServiceImpl(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public      UserDetails      loadUserByUsername(String      username)      throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);
        if (user == null) {
            throw    new    UsernameNotFoundException(String.format(USER_NOT_FOUND_MSG,
username));
        }
        return user;
    }
}
```

UserServiceImpl.java

```
package com.dumanskyi.delivery.services.impl;

import com.dumanskyi.delivery.entities.db.Role;
import com.dumanskyi.delivery.entities.db.User;
import com.dumanskyi.delivery.persistence.UserRepository;
import com.dumanskyi.delivery.services.api.UserService;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

@Service
public class UserServiceImpl implements UserService {
    private final UserRepository userRepository;

    public UserServiceImpl(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public void createCustomer(User user) {
        user.setRole(Role.CUSTOMER);
        userRepository.save(user);
```

```
    }

    @Override
    public UserDetails getCurrentUserDetails() {
        Object                                principal                                =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        return (UserDetails)principal;
    }

    @Override
    public User getCurrentUser() {
        return userRepository.findByUsername(getCurrentUserDetails().getUsername());
    }

}
```

NPRequestBodyUtil.java

```
package com.dumanskyi.delivery.utils;

import com.dumanskyi.delivery.entities.novaposhta.requests.NPRequestBody;
import com.dumanskyi.delivery.entities.novaposhta.responses.NPDataConstants;
import lombok.experimental.UtilityClass;

import java.util.HashMap;
import java.util.Map;

@UtilityClass
public class NPRequestBodyUtil {
    public NPRequestBody getWarehouseTypes() {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getWarehouseTypes")
                .methodProperties(new HashMap<>())
                .build();
    }
    public NPRequestBody getCitiesByName(String name, int amount) {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getCities")
                .methodProperties(Map.of(
                        NPDataConstants.FIND_BY_STRING, name,
                        NPDataConstants.PAGE, Integer.toString(1),
                        NPDataConstants.LIMIT, Integer.toString(amount)
                ))
                .build();
    }
    public NPRequestBody getAllCities(int amount) {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getCities")
                .methodProperties(Map.of(
                        NPDataConstants.PAGE, Integer.toString(1),
                        NPDataConstants.LIMIT, Integer.toString(amount)
                ))
                .build();
    }
    public NPRequestBody getWarehousesByCityId(String id, int amount) {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getWarehouses")
                .methodProperties(Map.of(
                        NPDataConstants.CITY_REF, id,
                        NPDataConstants.PAGE, Integer.toString(1),
                        NPDataConstants.LIMIT, Integer.toString(amount)
                ))
```

```
                .build();
    }
    public NPRequestBody getWarehousesByCityId(String id, String warehouseId) {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getWarehouses")
                .methodProperties(Map.of(
                        NPDataConstants.CITY_REF, id,
                        NPDataConstants.PAGE, Integer.toString(1),
                        NPDataConstants.TYPE_WAREHOUSE_REF, warehouseId
                ))
                .build();
    }
    public NPRequestBody getWarehousesByCityIdAndNumber(String id, int number) {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getWarehouses")
                .methodProperties(Map.of(
                        NPDataConstants.WAREHOUSE_ID, Integer.toString(number),
                        NPDataConstants.CITY_REF, id,
                        NPDataConstants.PAGE, Integer.toString(1),
                        NPDataConstants.LIMIT, Integer.toString(20)
                ))
                .build();
    }
    public NPRequestBody getWarehousesById(String id) {
        return NPRequestBody.builder()
                .modelName("Address")
                .calledMethod("getWarehouses")
                .methodProperties(Map.of(
                        NPDataConstants.REF, id
                ))
                .build();
    }
}
```

RequestUtil.java

```
package com.dumanskyi.delivery.utils;

import com.dumanskyi.delivery.entities.novaposhta.requests.NPRequestBody;
import com.dumanskyi.delivery.entities.novaposhta.responses.NPResponseBody;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.experimental.UtilityClass;
import org.apache.commons.lang.StringEscapeUtils;
import org.springframework.util.ResourceUtils;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.Map;
import java.util.Scanner;

@UtilityClass
public class RequestsUtil {

    private String getRequestBodyByFileName(String fileName) {
        StringBuilder request = new StringBuilder();
        try {
```

```
            File file = ResourceUtils.getFile("classpath:requests/" + fileName);
            InputStream in = new FileInputStream(file);
            Scanner scanner = new Scanner(in);
            while (scanner.hasNext()) {
                request.append(scanner.nextLine());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return request.toString();
    }

    public String getRequestBody(String filename, Map<String, String> params) {
        String request = getRequestBodyByFileName(filename);
        for (Map.Entry<String, String> param: params.entrySet()) {
            request = request.replace("{{ " + param.getKey() + " }}", param.getValue());
        }
        return request;
    }
    public  String sendRequest(String uri, String body) {
        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(uri))
                .header("accept", "appication/json")
                .header("Accept-Charset", "utf-8")
                .header("accept-language",                              "ru-RU,ru;q=0.9,en-
US;q=0.8,en;q=0.7,uk;q=0.6")
                .method("GET", HttpRequest.BodyPublishers.ofString(body))
                .build();
        HttpResponse<String> response;
        try {
            response = client.send(request,
                    HttpResponse.BodyHandlers.ofString());
        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
        return StringEscapeUtils.unescapeJava(response.body());
    }
    public NPResponseBody sendRequestNP(String apiUri, NPRequestBody requestBody) {
        ObjectMapper objectMapper = new ObjectMapper()
                .configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
        try {
            String requestBodyString = objectMapper.writeValueAsString(requestBody);
            return    objectMapper.readValue(sendRequest(apiUri,    requestBodyString),
NPResponseBody.class);
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
    }
}
```

StringUtil.java

```
package com.dumanskyi.delivery.utils;

import lombok.experimental.UtilityClass;

@UtilityClass
public class StringUtil {
    public String capitalizeFirstLetter(String string) {
        return            string.substring(0,            1).toUpperCase()            +
string.substring(1).toLowerCase();
    }
}
```

address_form.css

```
#filter {
    box-sizing: border-box;
    /*background-image: url('searchicon.png');*/
    background-position: 14px 12px;
    background-repeat: no-repeat;
    font-size: 16px;
    padding: 14px 20px 12px 45px;
    border: none;
    border-bottom: 1px solid #ddd;
}

#filter:focus {outline: 3px solid #ddd;}

.dropdown {
    position: relative;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f6f6f6;
    min-width: 230px;
    border: 1px solid #ddd;
    z-index: 1;
}

.dropdown-content option {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
}

.dropdown-content option:hover {background-color: #e1e1e1}

.show {display:block;}
```

customer_cabinet.css

```
.request:hover {
    background-color: #efefef;
}
.address-form {
    width: 75%;
}
```

registration_form.css

```
.form-signin input[type="password"], .form-signin input[type="email"] {
    border-radius: .25rem;
}
.form-signin input[type="button"] {
    width:100%;
}
```

address_from.js

```
function showCityList() {
    document.getElementById("dropdownWarehouseList").classList.toggle("show");
    filterFunction();
}

function filterFunction() {
    let filter = document.getElementById("filter");
    console.log(filter);
    $.ajax({
```

```
        url:"/cities?amount=5&name=" + filter.value,
        type:"get",
        complete: [
            function (response) {
            $("#options").empty();
            let cities = $.parseJSON(response.responseText);
            for(let i = 0; i < cities.length; i++) {
                $('<option>' + cities[i].value + '</option>').attr({
                    value: cities[i].key,
                    onclick: "chooseCity(event)"
                }).appendTo("#options");
            }
        }
        ]
    })
}
function chooseCity(event) {
    document.getElementById("city_name").value = event.target.innerHTML;
    document.getElementById("city_id").value = event.target.value;
    document.getElementById("warehouse_number").disabled = false;
    document.getElementById("warehouse_search_button").disabled = false;
    showCityList();
}
function searchWarehouse() {
    let cityId = document.getElementById("city_id").value;
    let warehouseNumber = document.getElementById("warehouse_number").value;
    $.ajax({
        url:"/warehouses?city=" + cityId + "&number=" + warehouseNumber,
        type:"get",
        complete: [
            function (response) {
                $("#options").empty();
                let warehouses = $.parseJSON(response.responseText);
                if (warehouses.length === 0) {
                    document.getElementById("warehouse_address_msg").innerHTML = "Can't
find warehouse #" + warehouseNumber + "!";
                } else {
                    document.getElementById("set_address_button").disabled = false;
                    document.getElementById("warehouse_address_msg").innerHTML = "Found
warehouse: " + warehouses[0].value;
                    document.getElementById("warehouse_id").value = warehouses[0].key;
                }
                for(let i = 0; i < warehouses.length; i++) {
                    $('<option>' + warehouses[i].value + '</option>').attr({
                        value: warehouses[i].key,
                        onclick: "chooseCity(event)"
                    }).appendTo("#warehouse");
                }
            }
        ]
    })
}
```

address_from.html

```
<form class="border rounded address-form" method="post" action="/customer/shipping-
address">
    <div class="container p-3">
        <p id="form-label" class="h5 p-2">Shipping address</p>
        <div class="form-group dropdown p-2">
            <label for="city_name" class="m-2">City</label>
            <input type="text" id="city_id" style="display: none">
            <input class="form-control mx-2 w-50" type="text" id="city_name"
onclick="showCityList()" placeholder="Choose city" style="background-color: white"
readonly>
            <div id="dropdownWarehouseList" class="dropdown-content">
```

```
                <input      type="text"      placeholder="Search.."      id="filter"
onkeyup="filterFunction()" class="border p-2 px-4">
                <div class="options" id="options">
                </div>
            </div>
        </div>
        <div class="form-group p-2">
            <label for="warehouse_number" class="m-2">Warehouse</label>
            <input  class="form-control  mx-2  w-50"  type="text"  id="warehouse_number"
placeholder="Enter warehouse number" disabled>
        </div>
        <div class="form-group p-2">
            <input     class="btn     btn-secondary     m-2     my-3"     type="button"
id="warehouse_search_button" onclick="searchWarehouse()" value="Search" disabled>
            <input    class="form-control"    name="warehouse_id"    id="warehouse_id"
style="display: none">
            <p class="p-2" id="warehouse_address_msg"></p>
        </div>
        <input  type="submit"  class="btn  btn-primary  mx-3"  id="set_address_button"
value="Set shipping address" disabled>
    </div>
</form>
```

admin_header.html

```
<div class="row justify-content-end" style="background-color: #fce9e1">
    <div class="col-auto m-2" style="text-align: center;">
        <h3       style="display:       flex;align-items:       center;justify-content:
center;">Ukrainian Delivery</h3>
    </div>
    <div   class="col-auto   m-2"><button   type="button"   class="btn   btn-primary"
onclick="{location.href='/admin/registration'}">Create new account</button></div>
    <div    class="col-auto    m-2"><button    type="button"    class="btn    btn-light"
onclick="{location.href='/perform_logout'}">Log out</button></div>
</div>
```

anonym_header.html

```
<div class="row justify-content-end" style="background-color: #fce9e1">
    <div class="col-auto m-2" style="text-align: center;">
        <h3       style="display:       flex;align-items:       center;justify-content:
center;">Ukrainian Delivery</h3>
    </div>
    <div    class="col-auto    m-2"><button    type="button"    class="btn    btn-light"
onclick="{location.href='/login'}">Sign in</button></div>
    <div class="col-auto m-2 d-none d-lg-block"><button type="button" class="btn btn-
light" onclick="{location.href='/registration'}">Sign up</button></div>
</div>
```

customer_header.html

```
<div class="row justify-content-end" style="background-color: #fce9e1">
    <div class="col-auto m-2" style="text-align: center;">
        <h3       style="display:       flex;align-items:       center;justify-content:
center;">Ukrainian Delivery</h3>
    </div>
    <div class="col-auto m-2" id="add-request-button"><button type="button" class="btn
btn-primary"                      onclick="{location.href='/customer/request/add'}">New
Request</button></div>
    <div class="col-auto m-2" id="cabinet-button"><button type="button" class="btn btn-
primary" onclick="{location.href='/customer/cabinet'}">Cabinet</button></div>
    <div    class="col-auto    m-2"><button    type="button"    class="btn    btn-light"
onclick="{location.href='/perform_logout'}">Log out</button></div>
</div>
```

deliveryman_header.html

```
<div class="row justify-content-end" style="background-color: #fce9e1">
    <div class="col-auto m-2" style="text-align: center;">
        <h3       style="display:       flex;align-items:       center;justify-content:
center;">Ukrainian Delivery</h3>
    </div>
    <div   class="col-auto   m-2"><button   type="button"   class="btn   btn-primary"
onclick="{location.href='/deliveryman/package_sending'}">Package
sending</button></div>
    <div   class="col-auto   m-2"><button   type="button"   class="btn   btn-primary"
onclick="{location.href='/deliveryman/package_receiving'}">Package
receiving</button></div>
    <div   class="col-auto   m-2"><button   type="button"   class="btn   btn-light"
onclick="{location.href='/perform_logout'}">Log out</button></div>
</div>
```

add_request.jsp

```
<%@ page import="com.dumanskyi.delivery.entities.db.User" %>
<%@ page import="com.dumanskyi.delivery.persistence.UserRepository" %>
<%@ page import="com.dumanskyi.delivery.services.api.NPService" %>
<%@ page import="com.dumanskyi.delivery.entities.db.PackageSize" %>
<%@ page import="com.dumanskyi.delivery.utils.StringUtil" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<!DOCTYPE html>
<html lang="en">
<%
    UserRepository            userRepository            =            (UserRepository)
request.getAttribute("userRepository");
    NPService npService = (NPService) request.getAttribute("npService");
    User user = (User) request.getAttribute("user");
    boolean shippingAddressIsPresent = user.getShippingAddress() != null;
    String warehouseName = "";
    if (shippingAddressIsPresent) {
        warehouseName                                                                =
npService.getWarehouseById(user.getShippingAddress().getNpWarehouseId()).get(0).getVal
ue();
    }
%>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
    <link href="/resources/css/address_form.css" rel="stylesheet">
    <script src="/resources/scripts/address_form.js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <style>
        .address-form {
            padding-bottom: 1rem!important;
        }
        .error-msg {
            color: red;
        }
    </style>
    <script>
        $(function(){
            $("#header").load("/resources/customer_header.html");
```

```
            $("#address-form").load("/resources/address_form.html");
        });
        function setPrice(index) {
            console.log(index);
            document.getElementById("price").innerHTML                                    =
document.getElementsByTagName("option")[index].id;
        }
        function onload() {
            <%if (!shippingAddressIsPresent) {%>
            document.getElementById("address-form").hidden = false;
            document.getElementById("add_request_btn").disabled = true;
            document.getElementById("deliveryNumber").disabled = true;
            document.getElementById("packageSizeId").disabled = true;
            document.getElementById("creditCard.phoneNumber").disabled = true;
            document.getElementById("creditCard.number").disabled = true;
            document.getElementById("creditCard.expMonth").disabled = true;
            document.getElementById("creditCard.expYear").disabled = true;
            document.getElementById("creditCard.cvv").disabled = true;
            alert("First of all you need to choose shipping address")
            <%}%>
            document.getElementById('creditCard.expMonth').value='';
            document.getElementById('creditCard.expYear').value='';
            document.getElementById('creditCard.cvv').value='';
        }
    </script>
</head>
<body                      onload="onload()"                      style="background-image:
url('/resources/imgs/background.jpg');">
    <div id="header"></div>
    <div class="w-75 container my-4 p-5 rounded bg-light">
        <div class="row justify-content-center">
            <form:form  modelAttribute="request"  class="col-lg-6  rounded  border  p-4"
method="post" action="/customer/request/add" id="add_request_form">
                <p class="h2">Create new request</p>
                <div class="form-group my-5">
                    <form:label    path="deliveryNumber"    class="h4">Enter    delivery
number</form:label>
                    <form:input path="deliveryNumber" type="text" name="deliveryNumber"
class="form-control mt-2 w-75" placeholder="Shop's delivery number" required="true"/>
                    <form:errors path="deliveryNumber" cssClass="error-msg"/>
                </div>
                <div class="form-group my-5">
                    <label class="h4">Choose size of your package</label>
                    <select id="packageSizeId" name="packageSizeId" class="form-select
mt-2 w-75" onclick="setPrice(this.selectedIndex)">
                        <%for (PackageSize size: PackageSize.values()) {%>
                        <option value="<%=size.getId()%>" id="<%=size.getPrice()%>">
                            <%=StringUtil.capitalizeFirstLetter(size.name())%>    (less
than <%=size.getMaxWeight()%> kilograms) - <%=size.getPrice()%>$
                        </option>
                        <%}%>
                    </select>
                </div>
                <div class="form-group my-5">
                    <form:label  path="creditCard.phoneNumber"  class="h4">Enter  your
phone number</form:label>
                    <form:input        path="creditCard.phoneNumber"        typ="text"
name="phoneNumber"    class="form-control    mt-2    w-75"    placeholder="38XXXXXXXXX"
required="true"/>
                    <form:errors path="creditCard.phoneNumber" cssClass="error-msg"/>
                </div>
                <div class="rounded border py-3 px-5 w-75">
                    <div class="form-group my-2 w-75">
                        <form:label  path="creditCard.number"  class="h4">Card  number:
</form:label>
```

```
                        <form:input path="creditCard.number" type="text" name="number"
class="form-control mt-2" placeholder="0000 0000 0000 0000" maxlength="16" title="Should
be 16 digits" required="true"/>
                        <form:errors path="creditCard.number" cssClass="error-msg"/>
                    </div>
                    <div class="form-group my-2 container px-0 mt-3">
                        <div class="row">
                            <div class="form-group col-5">
                                <label class="h6">Expiration date: </label>
                                <div class="input-group">
                                    <form:input path="creditCard.expMonth" type="text"
name="expMonth"     class="form-control     w-50"     placeholder="MM"     maxlength="2"
required="true"/>
                                    <form:input  path="creditCard.expYear"  type="text"
name="expYear"     class="form-control     w-50"     placeholder="YY"     maxlength="2"
required="true"/>
                                </div>
                                <form:errors                path="creditCard.expMonth"
cssClass="error-msg"/>
                                <form:errors path="creditCard.expYear" cssClass="error-
msg"/>
                            </div>
                            <div class="col-3">
                                <form:label    path="creditCard.cvv"    class="h6">CVV:
</form:label>
                                <form:input      path="creditCard.cvv"      type="text"
name="cvv" class="form-control" maxlength="3" required="true"/>
                                <form:errors   path="creditCard.cvv"   cssClass="error-
msg"/>
                            </div>
                        </div>
                    </div>
                </div>
            </form:form>
            <div class="col-lg-4 mx-2 px-0 py-2">
                <div id="address-form" hidden="true" class="mb-2"></div>
                <div class="h-auto border rounded p-4">
                    <p class="h3">Request summary</p>
                    <p       class="h5       my-4">Total       to       pay:       <span
id="price"><%=PackageSize.SMALL.getPrice()%></span>$</p>
                    <input     class="btn     btn-primary     px-4"     type="submit"
id="add_request_btn" form="add_request_form" value="Place order"/>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

admin_registration.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta  name="viewport"  content="width=device-width,  initial-scale=1,  shrink-to-
fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Please sign in</title>
    <link                        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css"             rel="stylesheet"             integrity="sha384-
/Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfaDjzALeQsN6M"
crossorigin="anonymous">
```

```
    <link          href="https://getbootstrap.com/docs/4.0/examples/signin/signin.css"
rel="stylesheet" crossorigin="anonymous"/>
    <link href="/resources/css/registration_form.css" rel="stylesheet"/>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <% if("true".equals(request.getParameter("customerUserCreated"))) {%>
    <script>
        alert("Account successfully created");
    </script>
    <%}%>
</head>
<body style="background-image: url('/resources/imgs/background.jpg');">
<div class="container">
    <form:form class="form-signin bg-white border rounded border-secondary" id="signup-
form" action="/admin/registration" modelAttribute="user">
        <h2 class="form-signin-heading">Account registration</h2>
        <p>
            <form:label path="role" for="role" class="sr-only">Role</form:label>
            <form:select path="role" id="role" name="role" class="form-control p-1">
                <form:option class="form-control" value="ADMIN">Admin</form:option>
                <form:option                                        class="form-control"
value="DELIVERYMAN">Deliveryman</form:option>
            </form:select>
            <form:errors path="role"/>
        </p>
        <p>
            <form:label path="email" for="email" class="sr-only">Email</form:label>
            <form:input path="email" type="email" id="email" name="email" class="form-
control" placeholder="Email" required="true" autofocus="true"/>
            <form:errors path="email"/>
        </p>
        <p>
            <form:label        path="username"        for="username"        class="sr-
only">Username</form:label>
            <form:input  path="username"  type="text"  id="username"  name="username"
class="form-control" placeholder="Username" required="true"/>
            <form:errors path="username"/>
        </p>
        <p>
            <form:label    path="firstName"    for="firstname"    class="sr-only">First
name</form:label>
            <form:input  path="firstName"  type="text"  id="firstname"  name="firstName"
class="form-control" placeholder="First name" required="true"/>
            <form:errors path="firstName"/>
        </p>
        <p>
            <form:label    path="lastName"    for="secondname"    class="sr-only">Second
name</form:label>
            <form:input  path="lastName"  type="text"  id="secondname"  name="lastName"
class="form-control" placeholder="Last name" required="true"/>
            <form:errors path="lastName"/>
        </p>
        <p>
            <form:label      path="patronymic"      for="patronymic"      class="sr-
only">Patronymic</form:label>
            <form:input path="patronymic" type="text" id="patronymic" name="patronymic"
class="form-control" placeholder="Patronymic"/>
            <form:errors path="patronymic"/>
        </p>
        <p>
            <form:label        path="password"        for="password"        class="sr-
only">Password</form:label>
            <form:input  path="password"  type="password"  id="password"  name="password"
class="form-control" placeholder="Password" required="true"/>
            <form:errors path="password"/>
        </p>
```

```
        <input type="submit" value="Submit" class="btn btn-lg btn-primary btn-block my-
3">
        <input  type="button"  value="Back"  class="btn  btn-lg  btn-secondary  my-3"
onclick="window.location.href = '/welcome';">
    </form:form>
</div>
</body></html>
```

customer_cabinet.jsp

```
<%@ page import="com.dumanskyi.delivery.entities.db.User" %>
<%@ page import="com.dumanskyi.delivery.persistence.UserRepository" %>
<%@ page import="com.dumanskyi.delivery.services.api.NPService" %>
<%@ page import="com.dumanskyi.delivery.entities.db.Request" %>
<%@ page import="com.dumanskyi.delivery.utils.StringUtil" %>
<%@ page import="com.dumanskyi.delivery.entities.db.Status" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
<!DOCTYPE html>
<html lang="en">
    <%
        UserRepository           userRepository           =           (UserRepository)
request.getAttribute("userRepository");
        NPService npService = (NPService) request.getAttribute("npService");
        User user = (User) request.getAttribute("user");
        boolean shippingAddressIsPresent = user.getShippingAddress() != null;
        String warehouseName = "";
        String cityName = "";
        if (shippingAddressIsPresent) {
            warehouseName                                                              =
npService.getWarehouseById(user.getShippingAddress().getNpWarehouseId()).get(0).getVal
ue();
            cityName                                                                   =
npService.getCityByWarehouseId(user.getShippingAddress().getNpWarehouseId()).get(0).ge
tValue();
        }
    %>
    <head>
        <meta charset="UTF-8">
        <title>Title</title>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
              integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
        <link href="/resources/css/address_form.css" rel="stylesheet">
        <link href="/resources/css/customer_cabinet.css" rel="stylesheet">
        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
        <script src="/resources/scripts/address_form.js"></script>
        <script>
            $(function(){
                $("#header").load("/resources/customer_header.html");
                $("#address-form").load("/resources/address_form.html");
                $("#shipping-address-btn").on("click", function () {
                    document.getElementById('form-
label').innerHTML='<%=(shippingAddressIsPresent ? "Changing" : "Adding") + " shipping
address"%>';
                    if (document.getElementById('address-form').hidden) {
                        document.getElementById('address-form').hidden=false;
                    } else {
                        document.getElementById('address-form').hidden=true;
                    }
                })
            });
```

```
            function finishRequest(id) {
                const request = new XMLHttpRequest();
                const url = '/customer/request/finished';
                request.open("POST", url);
                request.setRequestHeader('Content-Type', 'application/json');
                request.send(JSON.stringify(id));

                request.onreadystatechange = (e) => {
                    window.location.reload();
                }
            }
        </script>
    </head>
    <body style="background-image: url('/resources/imgs/background.jpg');">
        <div id="header"></div>
        <div class="container my-5 p-5 rounded bg-light w-50">
            <p class="h1">Hi ${user.username}</p>
            <p class="h3">First name: ${user.firstName}</p>
            <p class="h3">Last name: ${user.lastName}</p>
            <p class="h3">Patronymic: ${user.patronymic}</p>
            <div class="mb-2"><p class="h5 pt-5 font-weight-bold">Shipping address:
            <%if(shippingAddressIsPresent) {%>
                <p class="h6">

<%=npService.getWarehouseFullAddressById(user.getShippingAddress().getNpWarehouseId())
%>
                </p>
            <%} else {%>
                Empty
            <%}%>
            </div>
                <button    id="shipping-address-btn"    class="btn    btn-primary    mx-
3"><%=shippingAddressIsPresent ? "Change" : "Add"%></button>
            </p>
            <span hidden="true" id="address-form"></span>
            <%if (!user.getRequestList().isEmpty()) {%>
            <div class="border rounded mt-5 p-3 container">
                <p class="h3 mb-3">Requests</p>
                <div class="container p-2 m-2">

                    <%for (Request userRequest : user.getRequestList()) {%>
                    <div class="border rounded row p-2 mt-2 request">
                        <p class="h5">
                            <%if(userRequest.getStatus().equals(Status.DELIVERED)) {%>
                                <img            height="20px"            class="mx-1"
src="/resources/imgs/check_mark.png">
                            <%}%>
                            Request #<%=userRequest.getRequestId()%>
                        </p>
                        <div    class="col-sm-12    col-md-6    col-lg-4    py-2">Status:
<%=userRequest.getStatus().getMessage()%></div>
                        <div  class="col-sm-12  col-md-6  col-lg-4  py-2">Package  size:
<%=StringUtil.capitalizeFirstLetter(userRequest.getPackageSize().name())%></div>
                        <div  class="col-sm-12  col-md-6  col-lg-4  py-2">Creation  date:
<%=userRequest.getCreationDate()%></div>
                        <div class="col-sm-12 col-md-6 col-lg-4 py-2">Transaction id:
<%=userRequest.getTransactionId()%></div>
                        <div class="col-sm-12 col-md-12 col-lg-8 py-2">Delivery number:
<%=userRequest.getDeliveryNumber()%></div>
                        <%if  (userRequest.getStatus().equals(Status.SENT_TO_CUSTOMER))
{%>
                            <div  class="col-sm-12  col-md-12  col-lg-8  py-2">Shipping
address:
<%=npService.getWarehouseFullAddressById(userRequest.getShippingAddress().getNpWarehou
seId())%></div>
```

```
                            <div  class="col-sm-12  col-md-12  col-lg-8  py-2"><input
type="button" onclick="finishRequest(<%=userRequest.getRequestId()%>)" class="btn btn-
success" value="I already received this package"></div>
                        <%}%>
                    </div>
                    <%}%>
                </div>
            </div>
            <%}%>
        </div>
    </body>
</html>
```

index.jsp

```
<%@                                                                                page
import="org.springframework.security.authentication.UsernamePasswordAuthenticationToke
n" %>
<%@ page import="org.springframework.security.core.GrantedAuthority" %>
<%@ page import="java.util.stream.Collectors" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Delivery - Welcome</title>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
              integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>
        <script>
            $(function(){
                let header_url = "resources/anonym_header.html";
                <sec:authorize access="hasAuthority('CUSTOMER')">
                    header_url = "/resources/customer_header.html";
                </sec:authorize>
                <sec:authorize access="hasAuthority('ADMIN')">
                    header_url = "/resources/admin_header.html"
                </sec:authorize>
                <sec:authorize access="hasAuthority('DELIVERYMAN')">
                header_url = "/resources/deliveryman_header.html";
                </sec:authorize>
                $("#header").load(header_url);
            });
        </script>
    </head>
    <body style="background-image: url('/resources/imgs/background.jpg');">
        <div id="header"></div>
        <div class="container bg-light p-5">
            <p class="h3">How it works:</p>
            <p class="h6">1) You send your order to our international warehouse</p>
            <p class="h6">2) We receive it</p>
            <p class="h6">3) And send it to you</p>
        </div>
    </body>
</html>
```

login.jsp

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta  name="viewport"  content="width=device-width,  initial-scale=1,  shrink-to-
fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Please sign in</title>
    <link                        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css"             rel="stylesheet"             integrity="sha384-
/Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfaDjzALeQsN6M"
crossorigin="anonymous">
    <link          href="https://getbootstrap.com/docs/4.0/examples/signin/signin.css"
rel="stylesheet" crossorigin="anonymous"/>
    <% if("true".equals(request.getParameter("customerUserCreated"))) {%>
    <script>
        alert("Account successfully created");
    </script>
    <%}%>
</head>
<body style="background-image: url('/resources/imgs/background.jpg');">
<div class="container">
    <form class="form-signin bg-white border rounded border-secondary" method="post"
action="/login">
        <h2 class="form-signin-heading">Please sign in</h2>
        <p>
            <label for="username" class="sr-only">Username</label>
            <input  type="text"  id="username"  name="username"  class="form-control"
placeholder="Username" required autofocus>
        </p>
        <p>
            <label for="password" class="sr-only">Password</label>
            <input type="password" id="password" name="password" class="form-control"
placeholder="Password" required>
        </p>
        <button class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
        <p class="mt-2">Don't have an account? <a href="/registration">Sign up</a></p>
        <p><a></a></p>
    </form>
</div>
</body></html>
```

package_receiving.jsp

```
<%@ page import="com.dumanskyi.delivery.persistence.RequestRepository" %>
<%@ page import="com.dumanskyi.delivery.entities.db.Request" %>
<%@ page import="com.dumanskyi.delivery.entities.db.Status" %>
<%@ page import="com.dumanskyi.delivery.utils.StringUtil" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
```

```
          integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
    <link href="/resources/css/address_form.css" rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="/resources/scripts/address_form.js"></script>
    <script>
        var pageNumber = 0;
        $(function() {
            $("#header").load("/resources/deliveryman_header.html");
        });
        function setPage(pageNumber) {
            if (pageNumber === 0) {
                document.getElementById("first-btn").disabled = true;
                document.getElementById("second-btn").disabled = false;
                document.getElementById("first-page").hidden = false;
                document.getElementById("second-page").hidden = true;
            } else {
                document.getElementById("first-btn").disabled = false;
                document.getElementById("second-btn").disabled = true;
                document.getElementById("first-page").hidden = true;
                document.getElementById("second-page").hidden = false;
            }
        }
        function checkboxClick(checkboxClass, buttonId) {
            var checkboxes = document.getElementsByClassName(checkboxClass);
            for (let i = 0; i < checkboxes.length; i++) {
                if (checkboxes[i].checked === true) {
                    document.getElementById(buttonId).disabled = false;
                    return;
                }
            }
            document.getElementById(buttonId).disabled = true;
        }
        function updateStatus(key) {
            var body = [];
            if (key === 'receive-international') {
                var checkboxes = document.getElementsByClassName("not-arrived");
                for (let i = 0; i < checkboxes.length; i++) {
                    if (checkboxes[i].checked === true) {
                        body.push(checkboxes[i].value);
                    }
                }
            } else if (key === 'receive-ukrainian') {
                var checkboxes = document.getElementsByClassName("arrived-to-ukrainian-
warehouse");
                for (let i = 0; i < checkboxes.length; i++) {
                    if (checkboxes[i].checked === true) {
                        body.push(checkboxes[i].value);
                    }
                }
            }
            const request = new XMLHttpRequest();
            const url = '/deliveryman/updateRequestStatus';
            request.open("POST", url);
            request.setRequestHeader('Content-Type', 'application/json');
            request.send(JSON.stringify(body));

            request.onreadystatechange = (e) => {
                window.location.reload();
            }
        }
    </script>
</head>
<%
```

```
    RequestRepository requestRepository = (RequestRepository)
request.getAttribute("requestRepository");
%>
<body style="background-image: url('/resources/imgs/background.jpg');">
    <div id="header"></div>
    <div class="container my-5 p-5 rounded bg-light">
        <input id="first-btn" type="button" class="btn btn-primary" value="Receiving at
international warehouse" onclick="setPage(0)">
        <input id="second-btn" type="button" class="btn btn-primary" value="Receiving
at Ukrainian warehouse" onclick="setPage(1)">
        <div id="first-page" hidden>
            <div class="border rounded mt-5 p-3 container">
                <form action="/deliveryman/package_receiving" method="post">
                    <p class="h3 mb-3">Chose received package and click <input
type="button" id="receive-international" onclick="updateStatus('receive-
international')" class="btn btn-primary" disabled value="Receive"></p>
                    <div class="container p-2 m-2">
                        <%for (Request userRequest :
requestRepository.findRequestByStatus(Status.NOT_ARRIVED)) {%>
                        <div class="border rounded row p-2 m-1">
                            <p class="h5"><input onclick="checkboxClick('not-arrived',
'receive-international')" class="form-check-input not-arrived" type="checkbox"
value="<%=userRequest.getRequestId()%>">Request #<%=userRequest.getRequestId()%></p>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Customer:
<%=userRequest.getUser().getFullName()%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Package size:
<%=StringUtil.capitalizeFirstLetter(userRequest.getPackageSize().name())%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Creation
date: <%=userRequest.getCreationDate()%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Transaction
id: <%=userRequest.getTransactionId()%></div>
                            <div class="col-sm-12 col-md-12 col-lg-8 py-2">Delivery
number: <%=userRequest.getDeliveryNumber()%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Status:
<%=userRequest.getStatus().getShortMessage()%></div>
                        </div>
                        <%}%>
                    </div>
                </form>
            </div>
        </div>
        <div id="second-page" hidden>
            <div class="border rounded mt-5 p-3 container">
                <form>
                    <p class="h3 mb-3">Chose received package and click <button
class="btn btn-primary" id="receive-ukrainian" onclick="updateStatus('receive-
ukrainian')" disabled>Receive</button></p>
                    <div class="container p-2 m-2">
                        <%for (Request userRequest :
requestRepository.findRequestByStatus(Status.SENT_TO_UKRAINE)) {%>
                        <div class="border rounded row p-2 m-1">
                            <p class="h5"><input onclick="checkboxClick('arrived-to-
ukrainian-warehouse', 'receive-ukrainian')" class="form-check-input arrived-to-
ukrainian-warehouse" type="checkbox" value="<%=userRequest.getRequestId()%>">Request
#<%=userRequest.getRequestId()%></p>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Customer:
<%=userRequest.getUser().getFullName()%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Package size:
<%=StringUtil.capitalizeFirstLetter(userRequest.getPackageSize().name())%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Creation
date: <%=userRequest.getCreationDate()%></div>
                            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Transaction
id: <%=userRequest.getTransactionId()%></div>
                            <div class="col-sm-12 col-md-12 col-lg-8 py-2">Delivery
number: <%=userRequest.getDeliveryNumber()%></div>
```

```
                            <div  class="col-sm-12  col-md-6  col-lg-4  py-2">Status:
<%=userRequest.getStatus().getShortMessage()%></div>
                        </div>
                        <%}%>
                    </div>
                </form>
            </div>
        </div>
    </div>
</body>
</html>
```

package_sending.jsp

```
<%@ page import="com.dumanskyi.delivery.persistence.RequestRepository" %>
<%@ page import="com.dumanskyi.delivery.entities.db.Request" %>
<%@ page import="com.dumanskyi.delivery.entities.db.Status" %>
<%@ page import="com.dumanskyi.delivery.utils.StringUtil" %>
<%@ page import="com.dumanskyi.delivery.services.api.NPService" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link  href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
  <link href="/resources/css/address_form.css" rel="stylesheet">
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script src="/resources/scripts/address_form.js"></script>
  <script>
    var pageNumber = 0;
    $(function() {
      $("#header").load("/resources/deliveryman_header.html");
    });
    function setPage(pageNumber) {
      if (pageNumber === 0) {
        document.getElementById("first-btn").disabled = true;
        document.getElementById("second-btn").disabled = false;
        document.getElementById("first-page").hidden = false;
        document.getElementById("second-page").hidden = true;
      } else {
        document.getElementById("first-btn").disabled = false;
        document.getElementById("second-btn").disabled = true;
        document.getElementById("first-page").hidden = true;
        document.getElementById("second-page").hidden = false;
      }
    }
    function checkboxClick(checkboxClass, buttonId) {
      var checkboxes = document.getElementsByClassName(checkboxClass);
      for (let i = 0; i < checkboxes.length; i++) {
        if (checkboxes[i].checked === true) {
          document.getElementById(buttonId).disabled = false;
          return;
        }
      }
      document.getElementById(buttonId).disabled = true;
    }
    function updateStatus(key) {
      var body = [];
      if (key === 'sent-to-ukrainian-warehouse') {
```

```
        var checkboxes = document.getElementsByClassName("arrived-to-international-warehouse");
        for (let i = 0; i < checkboxes.length; i++) {
          if (checkboxes[i].checked === true) {
            body.push(checkboxes[i].value);
          }
        }
      } else if (key === 'sent-to-customer') {
        var checkboxes = document.getElementsByClassName("arrived-to-ukrainian-warehouse");
        for (let i = 0; i < checkboxes.length; i++) {
          if (checkboxes[i].checked === true) {
            body.push(checkboxes[i].value);
          }
        }
      }
      const request = new XMLHttpRequest();
      const url = '/deliveryman/updateRequestStatus';
      request.open("POST", url);
      request.setRequestHeader('Content-Type', 'application/json');
      request.send(JSON.stringify(body));

      request.onreadystatechange = (e) => {
        window.location.reload();
      }
    }
  </script>
</head>
<%
  RequestRepository requestRepository = (RequestRepository) request.getAttribute("requestRepository");
  NPService npService = (NPService) request.getAttribute("npService");
%>
<body style="background-image: url('/resources/imgs/background.jpg');">
<div id="header"></div>
<div class="container my-5 p-5 rounded bg-light">
  <input id="first-btn" type="button" class="btn btn-primary" value="Sending to Ukrainian warehouse" onclick="setPage(0)">
  <input id="second-btn" type="button" class="btn btn-primary" value="Sending to customer" onclick="setPage(1)">
  <div id="first-page" hidden>
    <div class="border rounded mt-5 p-3 container">
      <form action="/deliveryman/package_receiving" method="post">
        <p class="h3 mb-3">Chose sent package and click <input type="button" id="sent-to-ukrainian-warehouse" onclick="updateStatus('sent-to-ukrainian-warehouse')" class="btn btn-primary" disabled value="Send"></p>
        <div class="container p-2 m-2">
          <%for (Request userRequest : requestRepository.findRequestByStatus(Status.ARRIVED_AT_THE_INTERNATIONAL_WAREHOUSE)) {%>
          <div class="border rounded row p-2 m-1">
            <p class="h5"><input onclick="checkboxClick('arrived-to-international-warehouse', 'sent-to-ukrainian-warehouse')" class="form-check-input arrived-to-international-warehouse" type="checkbox" value="<%=userRequest.getRequestId()%>">Request #<%=userRequest.getRequestId()%></p>
            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Customer: <%=userRequest.getUser().getFullName()%></div>
            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Package size: <%=StringUtil.capitalizeFirstLetter(userRequest.getPackageSize().name())%></div>
            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Creation date: <%=userRequest.getCreationDate()%></div>
            <div class="col-sm-12 col-md-6 col-lg-4 py-2">Transaction id: <%=userRequest.getTransactionId()%></div>
            <div class="col-sm-12 col-md-12 col-lg-8 py-2">Delivery number: <%=userRequest.getDeliveryNumber()%></div>
```

```
            <div      class="col-sm-12      col-md-6      col-lg-4      py-2">Status:
<%=userRequest.getStatus().getShortMessage()%></div>
          </div>
          <%}%>
        </div>
      </form>
    </div>
  </div>
  <div id="second-page" hidden>
    <div class="border rounded mt-5 p-3 container">
      <form>
        <p class="h3 mb-3">Chose sent package and click <button class="btn btn-primary"
id="sent-to-customer"                          onclick="updateStatus('sent-to-customer')"
disabled>Send</button></p>
        <div class="container p-2 m-2">
          <%for                  (Request                  userRequest                  :
requestRepository.findRequestByStatus(Status.ARRIVED_AT_THE_UKRAINIAN_WAREHOUSE)) {%>
          <div class="border rounded row p-2 m-1">
            <p      class="h5"><input      onclick="checkboxClick('arrived-to-ukrainian-
warehouse',    'sent-to-customer')"    class="form-check-input    arrived-to-ukrainian-
warehouse"       type="checkbox"       value="<%=userRequest.getRequestId()%>">Request
#<%=userRequest.getRequestId()%></p>
            <div      class="col-sm-12      col-md-6      col-lg-4      py-2">Customer:
<%=userRequest.getUser().getFullName()%></div>
            <div    class="col-sm-12    col-md-6    col-lg-4    py-2">Package    size:
<%=StringUtil.capitalizeFirstLetter(userRequest.getPackageSize().name())%></div>
            <div    class="col-sm-12    col-md-6    col-lg-4    py-2">Creation    date:
<%=userRequest.getCreationDate()%></div>
            <div    class="col-sm-12    col-md-6    col-lg-4    py-2">Transaction    id:
<%=userRequest.getTransactionId()%></div>
            <div   class="col-sm-12   col-md-12   col-lg-8   py-2">Delivery   number:
<%=userRequest.getDeliveryNumber()%></div>
            <div      class="col-sm-12      col-md-6      col-lg-4      py-2">Status:
<%=userRequest.getStatus().getShortMessage()%></div>
            <div   class="col-sm-12   col-md-12   col-lg-8   py-2">Customer   address:
<%=npService.getWarehouseFullAddressById(userRequest.getUser().getShippingAddress().ge
tNpWarehouseId())%></div>
          </div>
          <%}%>
        </div>
      </form>
    </div>
  </div>
</div>
</body>
</html>
```

registration.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta  name="viewport"  content="width=device-width,  initial-scale=1,  shrink-to-
fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Please sign in</title>
    <link                       href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css"             rel="stylesheet"             integrity="sha384-
/Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfaDjzALeQsN6M"
crossorigin="anonymous">
    <link          href="https://getbootstrap.com/docs/4.0/examples/signin/signin.css"
rel="stylesheet" crossorigin="anonymous"/>
```

```
    <link href="/resources/css/registration_form.css" rel="stylesheet"/>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
<body style="background-image: url('/resources/imgs/background.jpg');">
<div class="container">
    <form:form class="form-signin bg-white border rounded border-secondary" id="signup-
form" action="/registration" modelAttribute="user">
        <h2 class="form-signin-heading">Please sign up</h2>
        <p>
            <form:label path="email" for="email" class="sr-only">Email</form:label>
            <form:input path="email" type="email" id="email" name="email" class="form-
control" placeholder="Email" required="true" autofocus="true"/>
            <form:errors path="email"/>
        </p>
        <p>
            <form:label        path="username"        for="username"        class="sr-
only">Username</form:label>
            <form:input  path="username"  type="text"  id="username"  name="username"
class="form-control" placeholder="Username" required="true"/>
            <form:errors path="username"/>
        </p>
        <p>
            <form:label    path="firstName"    for="firstname"    class="sr-only">First
name</form:label>
            <form:input  path="firstName"  type="text"  id="firstname"  name="firstName"
class="form-control" placeholder="First name" required="true"/>
            <form:errors path="firstName"/>
        </p>
        <p>
            <form:label    path="lastName"    for="secondname"    class="sr-only">Second
name</form:label>
            <form:input  path="lastName"  type="text"  id="secondname"  name="lastName"
class="form-control" placeholder="Last name" required="true"/>
            <form:errors path="lastName"/>
        </p>
        <p>
            <form:label       path="patronymic"       for="patronymic"       class="sr-
only">Patronymic</form:label>
            <form:input path="patronymic" type="text" id="patronymic" name="patronymic"
class="form-control" placeholder="Patronymic"/>
            <form:errors path="patronymic"/>
        </p>
        <p>
            <form:label        path="password"        for="password"        class="sr-
only">Password</form:label>
            <form:input  path="password"  type="password"  id="password"  name="password"
class="form-control" placeholder="Password" required="true"/>
            <form:errors path="password"/>
        </p>
        <input type="submit" value="Sign up" class="btn btn-lg btn-primary btn-block">
        <p class="mt-2">Already have an account? <a href="/login">Log In</a></p>
    </form:form>
</div>
</body></html>
```

shema.sql

```
CREATE TABLE `delivery_db`.`users`
(
    `user_id`             INT         NOT NULL AUTO_INCREMENT,
    `first_name`          VARCHAR(45) NOT NULL,
    `last_name`           VARCHAR(45) NOT NULL,
    `patronymic`          VARCHAR(45) NOT NULL,
    `shipping_address_id` INT         NULL,
    `email`               VARCHAR(45) NOT NULL,
```

```
    `password`            VARCHAR(45) NOT NULL,
    `role_id`             INT         NOT NULL,
    `username`            VARCHAR(45) NOT NULL,
    PRIMARY KEY (`user_id`)
);


CREATE TABLE `delivery_db`.`shipping_addresses`
(
    `shipping_address_id`    INT         NOT NULL AUTO_INCREMENT,
    `np_warehouse_id` VARCHAR(40) NOT NULL,
    PRIMARY KEY (`shipping_address_id`)
);


CREATE TABLE `delivery_db`.`requests`
(
    `request_id`     INT         NOT NULL AUTO_INCREMENT,
    `user_id`         INT         NOT NULL,
    `creation_date`   DATETIME    NOT NULL,
    `status_id`       INT         NOT NULL,
    `delivery_number` VARCHAR(45) NOT NULL,
    `package_size_id` INT NOT NULL,
    `transaction_id` VARCHAR(45) NOT NULL,
    `shipping_address_id` INT,
    PRIMARY KEY (`request_id`)
);

CREATE TABLE `delivery_db`.`roles` (
    `role_id` INT NOT NULL,
    `role_name` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`role_id`));

CREATE TABLE `delivery_db`.`package_sizes` (
    `package_size_id` INT NOT NULL,
    `name` VARCHAR(45) NOT NULL,
    `max_weight` INT NOT NULL,
    `price` FLOAT NOT NULL,
    PRIMARY KEY (`package_size_id`));

CREATE TABLE `delivery_db`.`statuses` (
    `status_id` INT NOT NULL,
    `status_name` VARCHAR(45) NOT NULL,
    `message` VARCHAR(45) NOT NULL,
    `short_message` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`status_id`));


-- INDEXES/FOREIGN_KEYS --

ALTER TABLE `delivery_db`.`users`
    ADD INDEX `shipping_address_id_idx` (`shipping_address_id` ASC) VISIBLE;
ALTER TABLE `delivery_db`.`users`
    ADD CONSTRAINT `shipping_address_id` FOREIGN KEY (`shipping_address_id`)
        REFERENCES `delivery_db`.`shipping_addresses` (`shipping_address_id`)
        ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE `delivery_db`.`requests`
    ADD INDEX `user_id_idx` (`user_id` ASC) VISIBLE;
ALTER TABLE `delivery_db`.`requests`
    ADD CONSTRAINT `user_id` FOREIGN KEY (`user_id`)
        REFERENCES `delivery_db`.`users` (`user_id`)
        ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `delivery_db`.`requests`
    ADD INDEX `shipping_address_request_idx` (`shipping_address_id` ASC) VISIBLE;
```

```
ALTER TABLE `delivery_db`.`requests`
    ADD CONSTRAINT `shipping_address_request`
        FOREIGN KEY (`shipping_address_id`)
            REFERENCES `delivery_db`.`shipping_addresses` (`shipping_address_id`)
            ON DELETE RESTRICT
            ON UPDATE CASCADE;
```

data.sql

```
/*insert into delivery_db.statuses (status_id, status_name, status_description)
values (0, 'NOT_RECEIVED', 'Package is not received yet. We are waiting for it.'),
       (1, 'RECEIVED', 'Package is already received. We are going to send it.'),
       (2, 'SENT', 'Package is sent. Wait for it.'),
       (3, 'DELIVERED', 'Package is delivered.');
*/

insert into delivery_db.users (first_name, last_name, patronymic, email, password,
role_id, username)
values ('Petro', 'Petrov', 'Petrovich', 'petr.petrov2000@gmail.com', 'password1', 1,
'petr'),
       ('Dmitry',    'Dmitrov',    'Dmitrovich',    'dmitr.dmitrovich1999@gmail.com',
'password1', 2, 'dmitr'),
       ('Tymur', 'Dumanskyi', 'Valerievich', 'cap.tim.2001@gmail.com', 'password1', 0,
'timr'),
       ('Oleh', 'Lukyanenko', 'Pavlovich', 'oleh.luk1969@gmail.com', 'password1', 0,
'oleh1969');
```

Dockerfile

```
FROM maven:3.8.2-jdk-11
WORKDIR /Delivery
COPY . .
RUN mvn clean install
CMD mvn spring-boot:run
```

docker-compose.yml

```
version: '3'

services:
  mysql:
    container_name: mysqldb
    image: mysql:8.0
    ports:
      - "3307:3306"
    environment:
      - MYSQL_DATABASE=delivery_db
      - MYSQL_USER=dev
      - MYSQL_PASSWORD=student
      - MYSQL_ROOT_PASSWORD=root
  main-service:
    container_name: main_service
    build: ./
    volumes:
      - .m2:/root/.m2
      - ./:/Delivery
    ports:
      - "8081:8080"
    environment:
      SPRING_APPLICATION_JSON: '{
                "spring.datasource.url"                                                :
"jdbc:mysql://mysqldb:3306/delivery_db?allowPublicKeyRetrieval=true&useSSL=false",
                "spring.datasource.username" : "dev",
                "spring.datasource.password" : "student"
              }'
    depends_on:
      - mysql
```