СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

на тему „Створення веб-додатку для реєстрації та управління відправленнями вантажів”

| | | |
|---|---|---|
| Виконав: студент групи ІПЗ-20д | _____ (підпис) | Щіров Д.С. (ініціали і прізвище) |
| Керівник | _____ (підпис) | Ратов Д.В. (ініціали і прізвище) |
| Завідувач кафедри | _____ (підпис) | О.І. Захожай (ініціали і прізвище) |

Рецензент___В.О. Лифар_____

Київ – 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
_____ Захожай О.І.
“___” _____ 2024 року

# З А В Д А Н Н Я
## НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Щіров Дмитро Сергійович
(прізвище, ім’я, по батькові)

1. Тема роботи: Створення веб-додатку для реєстрації та управління відправленнями вантажів
керівник роботи Ратов Денис Валентинович, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “_06_”_травня_2024 року №171/15.15-С

2. Строк подання студентом роботи 15.06.2024р.
3. Вихідні дані до роботи: Об'єктом даної роботи є процес розробки веб-додатку для реєстрації додатку для реєстрації та управління відправленнями вантажів
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Вступ. Аналітичний огляд, з висвітленням наступних питань: поширення аналогічних систем у світі, важливість для логістичних компаній мати додаток, порівняння мов програмування для розробки додатку. Основна частина, в якій висвітлити інформаційну та функціональну модель додатку, продемонструвати графічний вид додатку, етап створення додатку та тестування. Висновки. Перелік використаних джерел.

5. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

6. Дата видачі завдання 30.03.2024р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання кваліфікаційної випускної роботи | Строк виконання етапів | Примітка |
|---|---|---|---|
| 1 | Одержання завдання на виконання роботи | 30.03.24 | виконано |
| 2 | Укладання і погодження з керівником плану і етапів виконання роботи | 06.04.24 | виконано |
| 3 | Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі» | 13.04.24 | виконано |
| 4 | Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху | 20.04.24 | виконано |
| 5 | Укладання та тестування програмного продукту | 27.04.24 | виконано |
| 6 | Укладання, оформлення та погодження пояснювальної записки з керівником | 04.05.24 | виконано |
| 7 | Здача готової пояснювальної записки на кафедру | 08.06.24 | виконано |
| 8 | Укладання доповіді і презентації | 10.06.24 | виконано |

**Студент** _____ Д.С. Щіров
підпис (ініціали і прізвище)

**Керівник роботи** _____ Д.В. Ратов
підпис (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ
дипломної роботи студента гр. ІПЗ-20д Тесля М.С.

Науковий керівник:

Доцент, к.т.н. _____ Ратов Д. В.

Оцінка наукового керівника: _____

Рецензент _____
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:
_____

Голова ЕК
Професор кафедри ІТП
д.т.н. _____ Меняйленко О.С.
підпис

# РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Створення веб-додатку для реєстрації та управління відправленнями вантажів» містить: 44 основних сторінок, 24 додаткових, зображення, 19 інформаційних джерел.

**Об’єкт дослідження** – система поставок товарів у торговельні мережі.

**Предмет дослідження** – технології автоматизації процесів реєстрації поставок товарів.

**Мета кваліфікаційної роботи** – розробити автоматизовану систему поставок товарів, використовуючи найефективніші інструменти та програмне забезпечення, визначені у ході дослідження.

**Методи дослідження** – аналіз, порівняння, обробка літературних джерел та проектування.

Розроблений прототип системи є цінним матеріалом, який може бути використаний для багатьох цілей. Він може служити основою для подальшого розвитку та вдосконалення системи управління поставками. Крім того, прототип можна використовувати для демонстрації потенціалу та можливостей цієї розробки для широкої публіки.

Для створення автоматизованої системи реєстру поставок товарів використані інструменти та забезпечення, які були обрані у ході дослідження як найбільш ефективні в порівнянні з їх аналогами. Використано мову програмування Python та фреймворк Django для бекенд-розробки, а також базу даних PostgreSQL для надійного зберігання даних.

# ЗМІСТ

# ВСТУП

У сучасному світі управління поставками товарів у торговельні мережі набуває все більшого значення. Ефективна автоматизація цього процесу дозволяє значно знизити витрати, підвищити точність обліку та покращити якість обслуговування клієнтів. Це обумовлює актуальність дослідження теми створення та автоматизації реєстру системи поставок товарів.

Мета цієї роботи полягає у розробці та впровадженні системи, яка автоматизує процеси реєстрації та управління поставками товарів у торговельні мережі. Основними завданнями дослідження є аналіз існуючих рішень, вибір найбільш ефективних технологій та засобів програмної реалізації, проектування та розробка системи, а також тестування та оцінка її ефективності.

**Актуальність теми**

Станом на зараз управління поставками товарів є одним з найважливіших аспектів, що впливає на ефективність та конкурентоспроможність торговельних мереж. Ефективне управління логістикою дозволяє не лише знизити витрати на логістику, але й забезпечити своєчасне поповнення запасів, що особливо важливо для задоволення попиту споживачів.

Одним з основних викликів у сфері управління поставками є необхідність оперативної обробки великого обсягу даних про товари, постачальників, замовлення та доставку. Традиційні методи управління, що базуються на ручному введенні даних та використанні паперових носіїв, не лише займають багато часу, але й підвищують ризик помилок та неточностей.

В умовах стрімкого розвитку інформаційних технологій зростає потреба у автоматизації процесів управління поставками. Сучасні технології дозволяють створювати системи, які можуть автоматично збирати, обробляти та аналізувати дані, забезпечуючи високу точність і швидкість обміну інформацією між усіма учасниками ланцюга поставок.

Створення додатку сприяє підвищенню прозорості та контрольованості логістичних процесів. Це дозволяє оперативно реагувати на зміни в попиті, оптимізувати запаси та мінімізувати затримки у постачанні товарів.

Значна увага до даної теми також обумовлена вимогами сучасного ринку до якості обслуговування клієнтів. Споживачі очікують своєчасного та точного виконання замовлень, а автоматизовані системи дозволяють забезпечити високу швидкість обробки замовлень та своєчасну доставку товарів.

Таким чином, тема створення та автоматизації реєстру системи поставок товарів є надзвичайно актуальною та перспективною для дослідження, оскільки вона не лише відповідає вимогам сучасного ринку, але й сприяє підвищенню ефективності управління логістичними процесами.

**Мета і завдання роботи**

Мета цієї роботи полягає в розробці та впровадженні автоматизованої системи реєстру поставок товарів для торговельних мереж. Це включає створення програмного забезпечення, яке забезпечить ефективне управління інформацією про постачальників, товари, замовлення та доставку, а також оптимізацію логістичних процесів для підвищення ефективності та якості обслуговування клієнтів.

**Для досягнення цієї мети в роботі ставляться такі завдання:**

1. Провести аналіз сучасних систем управління поставками товарів, визначити їх переваги та недоліки, а також провести порівняльний аналіз існуючих рішень.
2. Вибрати найдоцільніші технології та інструменти для розробки системи, включаючи мову програмування, базу даних та середовище розробки.
3. Розробити архітектуру системи, моделі даних та спроектувати структуру бази даних для ефективного зберігання та обробки інформації.
4. Створити інтуїтивно зрозумілий та зручний користувацький інтерфейс, забезпечивши простоту навігації та доступність основних функцій.
5. Реалізувати програмний код для всіх компонентів системи, включаючи модулі для введення, зберігання та обробки даних, а також забезпечити інтеграцію з існуючими програмними рішеннями.

6. Провести тестування розробленої системи, виявити та усунути помилки, оптимізувати продуктивність та забезпечити надійність роботи системи.
7. Проаналізувати ефективність роботи системи на основі отриманих даних, підготувати рекомендації щодо її впровадження у торговельні мережі та розробити пропозиції щодо подальшого вдосконалення.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ (АНАЛІТИЧНИЙ ОГЛЯД)

## 1.1. Об’єкт та редмет дослідження

**Об’єктом дослідження** є процеси управління поставками товарів у торговельні мережі. Це включає всі аспекти, пов'язані з організацією, контролем та оптимізацією потоків товарів від постачальників до кінцевих споживачів. Управління поставками є критично важливим для ефективної роботи торговельних мереж, оскільки воно впливає на вчасність доставки товарів, рівень запасів, витрати на логістику та загальну задоволеність клієнтів. Дослідження охоплює аналіз існуючих логістичних практик, систем обліку та управління даними, а також технологій, що застосовуються для покращення цих процесів.

**Предметом дослідження** є методи та засоби автоматизації реєстру системи поставок товарів у торговельні мережі. Це включає розробку та впровадження програмного забезпечення, яке дозволить автоматизувати процеси збору, обробки та аналізу даних про поставки товарів. Основна увага приділяється розробці ефективної бази даних для зберігання інформації про постачальників, товари, замовлення та доставки, створенню інтуїтивно зрозумілого користувацького інтерфейсу, а також інтеграції розробленої системи з існуючими програмними рішеннями в торговельних мережах. Автоматизація цих процесів сприятиме підвищенню точності даних, скороченню часу на обробку інформації та оптимізації логістичних операцій.

## 1.2. Методологія дослідження

Для досягнення мети та вирішення завдань цієї дипломної роботи була використана комплексна методологія дослідження, яка включає різні методи та підходи:

1. **Аналіз літературних джерел та аналогічних систем:**

- Проведення детального огляду наукової літератури, технічної документації та інших інформаційних джерел з метою вивчення сучасного стану проблеми автоматизації управління поставками товарів у торговельні мережі.

- Аналіз існуючих систем управління поставками, виявлення їхніх переваг та недоліків, а також можливостей для покращення.

2. **Системний підхід:**

- Використання системного підходу для проектування архітектури системи, що дозволяє врахувати всі компоненти та їх взаємодію.

- Моделювання логістичних процесів та структуризація даних для забезпечення ефективного управління інформацією про поставки.

3. **Вибір та обґрунтування технологій:**

- Обґрунтування вибору мови програмування, бази даних та середовища розробки на основі аналізу вимог до системи та можливостей різних технологій.

- Оцінка технічних характеристик та можливостей обраних технологій для забезпечення високої продуктивності та надійності системи.

4. **Проектування та розробка системи:**

- Використання методів об’єктно-орієнтованого проектування для створення моделей даних, розробки архітектури та проектування користувацького інтерфейсу.

- Програмування модулів системи з використанням сучасних інструментів розробки.

5. **Тестування та верифікація:**

- Проведення тестування розробленої системи на різних етапах для виявлення та усунення помилок.

- Використання методів функціонального та нефункціонального тестування для забезпечення відповідності системи вимогам.

6. **Аналіз результатів та підготовка рекомендацій:**

- Аналіз ефективності роботи системи на основі отриманих результатів тестування.

- Підготовка рекомендацій щодо впровадження системи у торговельні мережі та її подальшого вдосконалення.

### 1.3. Поняття та важливість реєстрів у системах поставок

Реєстр поставок товарів є ключовим елементом управління логістичними процесами. Він являє собою централізовану базу даних, яка містить інформацію про всі етапи поставок: від замовлення товару до його доставки кінцевому споживачу. Реєстр включає дані про постачальників, товари, замовлення, маршрути доставки, склади та інші важливі аспекти.

**Важливість реєстрів у системах поставок:**

1. Централізація даних:

- Реєстри дозволяють зберігати всю інформацію в одному місці, що спрощує доступ до даних та їх обробку.

- Це забезпечує єдність та узгодженість даних, що важливо для прийняття управлінських рішень.

2. Підвищення точності та зменшення помилок:

- Використання автоматизованих реєстрів зменшує ймовірність людських помилок, пов’язаних з ручним введенням даних.

- Автоматизація процесів дозволяє швидко та точно обробляти великі обсяги інформації.

3. Оптимізація логістичних процесів:

- Реєстри допомагають оптимізувати маршрути доставки, планувати запаси та керувати складськими операціями.

- Це сприяє зниженню витрат на логістику та підвищенню ефективності управління поставками.

4. Підвищення рівня обслуговування клієнтів:

- Своєчасна та точна інформація про поставки дозволяє забезпечити вчасну доставку товарів, що покращує задоволеність клієнтів.

- Реєстри також дозволяють відслідковувати статус замовлень та оперативно реагувати на зміни.

**Класифікація систем поставок товарів**

Системи поставок товарів можна класифікувати за різними ознаками, що дозволяє краще розуміти їх особливості та обирати найбільш підходящі рішення для конкретних потреб.

**За рівнем автоматизації:**

- Традиційні системи: Використовують ручне введення даних та паперову документацію. Характеризуються низькою швидкістю обробки інформації та високою ймовірністю помилок.

- Автоматизовані системи: Використовують спеціалізоване програмне забезпечення для автоматизації процесів управління поставками. Забезпечують високу точність даних та ефективність логістичних операцій.

**За типом системи:**

- ERP-системи (Enterprise Resource Planning): Інтегровані системи управління ресурсами підприємства, які включають модулі для управління поставками, виробництвом, фінансами, людськими ресурсами та іншими аспектами діяльності.

- WMS-системи (Warehouse Management System): Системи управління складом, які забезпечують автоматизацію складських операцій, облік запасів, оптимізацію розміщення товарів та контроль за їх рухом.

- CRM-системи (Customer Relationship Management): Системи управління взаємовідносинами з клієнтами, які включають функції обліку замовлень, управління контактами та підтримки клієнтів.

**За галузевою спеціалізацією:**

- Універсальні системи: Підходять для широкого кола галузей та видів діяльності. Мають гнучку структуру та налаштування, що дозволяє адаптувати їх під конкретні потреби.

- Спеціалізовані системи: Розроблені для конкретних галузей (наприклад, роздрібна торгівля, виробництво, логістика). Враховують особливості та специфіку конкретної галузі.

### 1.4. Сучасні тенденції торгових мереж

У сучасних умовах розвитку технологій та зростаючих вимог до ефективності бізнесу автоматизація замовлень набуває все більшого значення. Основні тенденції включають впровадження штучного інтелекту та машинного навчання, використання Інтернету речей (IoT), застосування хмарних технологій, розвиток мобільних додатків та технологій, а також аналіз великих даних (Big Data).

Впровадження штучного інтелекту та машинного навчання є однією з ключових тенденцій у сучасній автоматизації. Використання AI та ML дозволяє прогнозувати попит, оптимізувати запаси, аналізувати поведінку клієнтів та підвищувати ефективність маркетингових кампаній. Інтелектуальні системи можуть автоматично аналізувати великі обсяги даних та приймати рішення на основі отриманих результатів, що значно підвищує точність та швидкість обробки інформації.

Хмарні технології надають нові можливості для автоматизації бізнес-процесів. Впровадження хмарних рішень дозволяє забезпечити доступ до даних з будь-якого місця та будь-якого пристрою, що підвищує мобільність та гнучкість бізнесу. Хмарні платформи забезпечують високу масштабованість, надійність та безпеку зберігання даних, що є критично важливим для сучасних торгових мереж.

Мобільні додатки та технології стають невід'ємною частиною автоматизації торгових мереж. Використання мобільних додатків дозволяє співробітникам торговельних мереж оперативно отримувати та вводити дані, відслідковувати статус замовлень, керувати запасами та здійснювати інші операції. Мобільні технології підвищують швидкість та зручність виконання логістичних процесів, що сприяє підвищенню ефективності роботи.

Аналіз великих даних (Big Data) є ще однією важливою тенденцією в автоматизації торгових мереж. Використання технологій Big Data дозволяє обробляти та аналізувати великі обсяги даних про клієнтів, продажі, запаси та інші аспекти діяльності. Аналітичні інструменти дозволяють виявляти приховані закономірності, прогнозувати тренди та приймати обґрунтовані рішення, що сприяє підвищенню ефективності управління бізнесом.

### 1.5. Огляд та аналіз існуючих рішень (аналогічних систем)

У сучасному світі існує багато програмних рішень для управління поставками товарів. Нижче наведено огляд деяких з них:

**SAP SCM:**

Опис: Комплексна система управління ланцюгами поставок, яка забезпечує планування, виконання, координацію та моніторинг всіх логістичних процесів.

Переваги: Висока функціональність, інтеграція з іншими модулями SAP, можливість масштабування.

Недоліки: Висока вартість впровадження та обслуговування, складність налаштування та використання.

**Oracle SCM Cloud:**

Опис: Хмарна платформа для управління ланцюгами поставок, що включає модулі для планування, обліку, транспортування та інших логістичних операцій.

Переваги: Хмарна інфраструктура, гнучкість та мобільність, висока безпека даних.

Недоліки: Залежність від інтернет-з’єднання, висока вартість підписки.

**Microsoft Dynamics 365 Supply Chain Management:**

Опис: Інтегрована система, що забезпечує автоматизацію процесів управління запасами, виробництвом, логістикою та іншими аспектами ланцюга поставок.

Переваги: Інтеграція з іншими продуктами Microsoft, гнучкість налаштувань, зручний інтерфейс.

Недоліки: Може бути дорогим для малих підприємств, потребує високої кваліфікації для налаштування.

**Infor SCM:**

Опис: Система управління ланцюгами поставок, що забезпечує повну прозорість і контроль над всіма аспектами ланцюга поставок.

Переваги: Можливість інтеграції з іншими системами, потужні аналітичні інструменти.

Недоліки: Складність впровадження, висока вартість.

**JDA Software:**

Опис: Програмне забезпечення для управління поставками, яке включає рішення для планування, виконання, аналізу та оптимізації ланцюгів поставок.

Переваги: Висока функціональність, можливість налаштування під конкретні потреби.

Недоліки: Високі вимоги до ресурсів, складність навчання персоналу.

Цей аналітичний огляд дає змогу зрозуміти основні тенденції та існуючі рішення в області автоматизації управління поставками товарів у торговельних мережах, що допоможе обґрунтовано підходити до вибору та впровадження найбільш ефективних інструментів для досягнення бізнес-цілей.

## Поняття реєстрів

Реєстр поставок товарів – це централізована база даних, яка містить детальну інформацію про всі аспекти процесу постачання. Основні компоненти включають:

**1 Дані про постачальників:**

- Контактна інформація, рейтинги надійності, історія співпраці.

- Умови договорів, обсяги замовлень, строки постачання.

**2. Дані про товари:**

- Найменування, опис, артикул, класифікація.

- Ціни, одиниці виміру, запаси на складах.

**3. Дані про замовлення:**

- Ідентифікаційний номер замовлення, дата створення, статус виконання.

- Деталі замовлення: перелік товарів, кількість, вартість.

**4. Дані про транспортування та логістику:**

- Маршрути доставки, транспортні засоби, відповідальні особи.

- Час відправлення та прибуття, умови зберігання та транспортування.

**5. Дані про склади:**

- Локації складів, наявність товарів, умови зберігання.

- Операції на складах: прийом, відправка, інвентаризація.

**Важливість реєстрів у системах поставок**

Реєстри відіграють важливу роль у забезпеченні ефективного управління ланцюгами поставок, надаючи такі переваги:

Реєстри забезпечують централізоване зберігання інформації, що дозволяє мати єдиний доступ до всіх даних про поставки. Централізація спрощує управління інформацією та дозволяє легко отримувати потрібні дані в режимі реального часу.

Автоматизовані реєстри зменшують ризик людських помилок, які можуть виникати при ручному введенні даних. Це підвищує точність інформації та дозволяє уникати проблем, пов’язаних з невірними даними.

Завдяки реєстрам можливо оптимізувати маршрути доставки, планувати запаси та керувати складськими операціями. Це дозволяє знижувати витрати на логістику, підвищувати ефективність використання ресурсів та покращувати координацію між різними етапами постачання.

Своєчасна та точна інформація про поставки дозволяє забезпечити вчасну доставку товарів та підвищувати задоволеність клієнтів. Реєстри дозволяють швидко відслідковувати статус замовлень та оперативно реагувати на зміни, що підвищує довіру клієнтів до компанії.

Реєстри забезпечують наявність великої кількості даних, що може бути використано для аналітики та прийняття обґрунтованих управлінських рішень. Аналітичні інструменти дозволяють виявляти тренди, прогнозувати попит, аналізувати ефективність логістичних процесів та вносити необхідні корективи.

**Функції реєстрів у системах поставок**

**Зберігання даних:**

- Реєстри забезпечують надійне зберігання великих обсягів інформації про постачання, яка може бути швидко та легко доступною.

**Обробка та аналіз даних:**

- Реєстри дозволяють автоматично обробляти дані, виконувати аналітичні обчислення та генерувати звіти для керівництва.

**Моніторинг та контроль:**

- Реєстри забезпечують постійний моніторинг усіх етапів процесу постачання, що дозволяє вчасно виявляти та усувати проблеми.

**Планування та оптимізація:**

- Реєстри дозволяють планувати обсяги поставок, оптимізувати маршрути доставки та керувати запасами на складах.

**Інтеграція з іншими системами:**

- Сучасні реєстри можуть інтегруватися з ERP, CRM та іншими системами, що забезпечує єдину інформаційну базу для всіх бізнес-процесів.

Реєстри забезпечують централізацію, точність, ефективність та оптимізацію всіх логістичних процесів, що дозволяє підвищувати рівень обслуговування клієнтів та знижувати витрати на логістику. Успішне впровадження та використання реєстрів сприяє розвитку бізнесу та зміцненню конкурентних позицій на ринку.

**Проблема використання Excel у системах поставок**

Однією з основних проблем, на яку звертають увагу більшість компаній, є те, що вони часто покладаються на функціонал Excel для управління системами поставок. Використання Excel і функцій формул в ньому є поширеною практикою, оскільки ця програма доступна, зрозуміла та відносно проста у використанні. Однак, Excel має значні обмеження, особливо коли мова йде про управління складними логістичними процесами.

Excel не забезпечує автоматичної інтеграції з іншими системами, що робить процес оновлення даних трудомістким і схильним до помилок. Крім того, великі

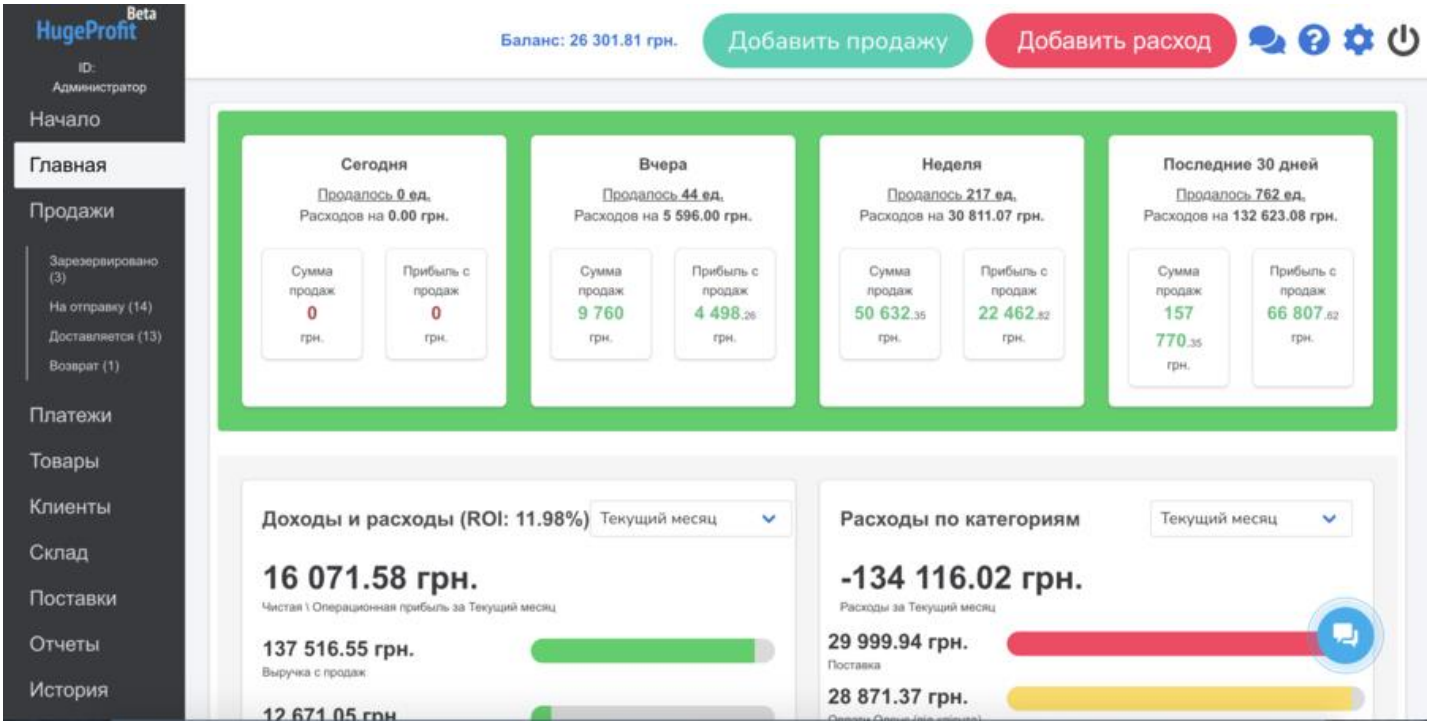
обсяги даних можуть призвести до уповільнення роботи програми, що негативно впливає на ефективність бізнес-процесів. Відсутність спеціалізованих інструментів для логістичного планування і оптимізації також є серйозним недоліком.

### 1.6 Програмні рішення для автоматизації систем

Щоб вирішити ці проблеми, багато компаній звертаються до спеціалізованих програмних рішень, які забезпечують більш високу ефективність та надійність у порівнянні з Excel.

Приклади таких рішень включають:

**HugeProfit** - це потужна система управління поставками, яка пропонує широкий спектр функцій для оптимізації логістичних процесів. Основні переваги HugeProfit включають:



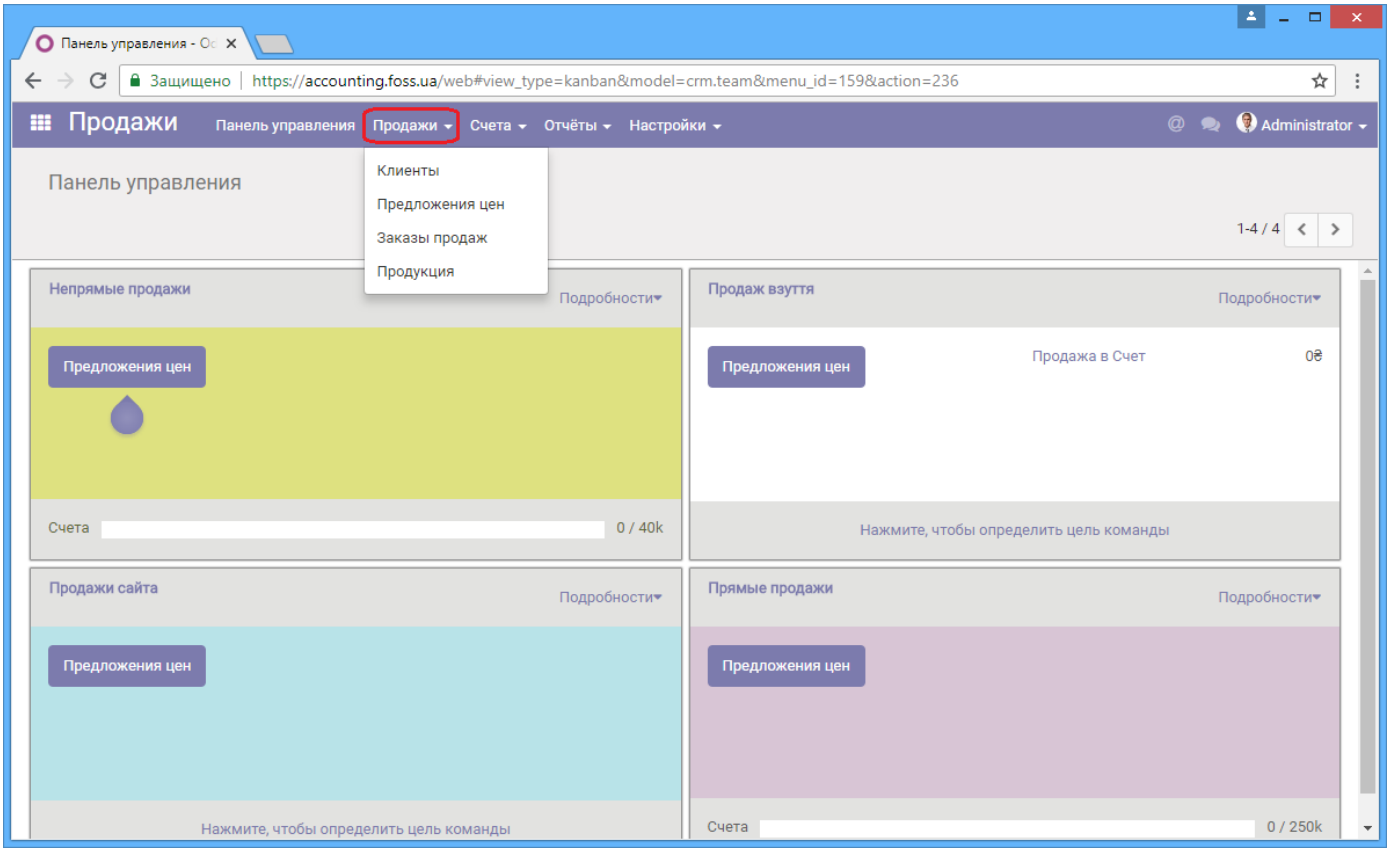

- Інтеграція з ERP та CRM системами: забезпечує централізоване управління всіма бізнес-процесами.

- Автоматизація процесів: дозволяє автоматично оновлювати дані, що зменшує ризик людських помилок.

- Аналітичні інструменти: включає модулі для прогнозування попиту, аналізу ефективності логістичних процесів та оптимізації запасів.

- Мобільність та доступність: можливість доступу до системи з будь-якого пристрою, що підвищує мобільність співробітників та оперативність прийняття рішень.

**ERP FOSS** - це ще одне потужне рішення для автоматизації управління ланцюгами поставок. Основні характеристики ERP FOSS:



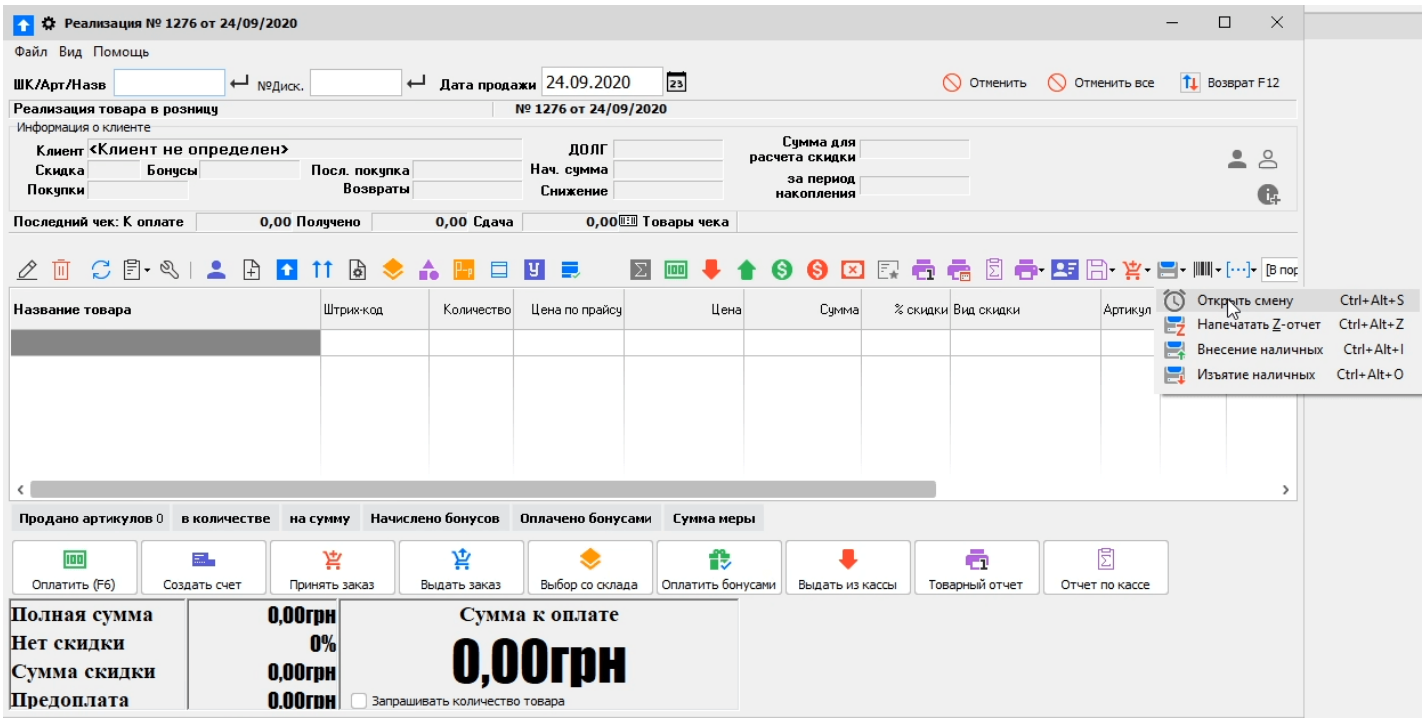

- Модульна структура: дозволяє налаштовувати систему під конкретні потреби компанії, включаючи модулі для управління запасами, виробництвом, фінансами та логістикою.

- Висока масштабованість: система легко адаптується до змін у бізнесі та зростання обсягів даних.

- Відкритий код: забезпечує гнучкість налаштувань та можливість інтеграції з іншими системами.

- Аналітика та звітність: потужні інструменти для збору та аналізу даних, що дозволяє приймати обґрунтовані управлінські рішення.

**Торгсофт** - це українська програма для автоматизації торгівлі, яка також може бути використана для управління поставками. Переваги:




- Інтеграція з касовими апаратами та торговим обладнанням: забезпечує ефективне управління продажами та обліком товарів.

- Облік запасів* можливість автоматизованого обліку товарів на складах та в магазинах, що зменшує ризик втрат та помилок.

- Управління лояльністю клієнтів: функції для роботи з програмами лояльності, обліку бонусів та знижок.

- Мобільні рішення: можливість використання мобільних додатків для віддаленого управління бізнесом.

**Переваги наведених спеціалізованих програмних рішень**

**Автоматизація та інтеграція:**

Спеціалізовані програми, такі як HugeProfit, ERP FOSS та Торгсофт, забезпечують автоматизацію більшості процесів управління поставками, що значно знижує ризик людських помилок та підвищує точність даних.

Інтеграція з іншими системами дозволяє централізовано управляти всіма аспектами бізнесу, що підвищує ефективність та узгодженість даних.

**Аналітика та прогнозування:**

Потужні аналітичні інструменти дозволяють компаніям виявляти тенденції, прогнозувати попит та планувати запаси, що допомагає уникнути дефіциту або надлишку товарів.

Автоматизовані звіти та дашборди забезпечують керівництво необхідною інформацією для прийняття обґрунтованих рішень.

**Оптимізація логістичних процесів:**

Програмні рішення дозволяють оптимізувати маршрути доставки, що знижує витрати на логістику та підвищує швидкість обробки замовлень.

Автоматизоване управління складами дозволяє ефективніше використовувати простір та ресурси.

**Підвищення рівня обслуговування клієнтів:**

Завдяки точній та своєчасній інформації про поставки, компанії можуть забезпечити вчасну доставку товарів та підвищити задоволеність клієнтів.

Інструменти управління лояльністю клієнтів, такі як у програмі Торгсофт, допомагають залучати нових клієнтів та утримувати існуючих.

Заміна Excel на спеціалізовані програмні рішення для управління системами поставок є непоганим кроком для підвищення ефективності та конкурентоспроможності компанії. Під час створення проекту необхідно детально розглянути сильні сторони наведених програмних продуктів, та звернути увагу на автоматизацію процесів, точність даних, аналітику та інтеграцію. Використання цих рішень дозволить компаніям ефективно управляти своїми ланцюгами поставок і задовольняти потреби клієнтів на високому рівні. Інтеграція найкращих практик і функціональностей з інших систем допоможе створити надійний і ефективний продукт, забезпечуючи оптимізацію процесів та підвищення загальної продуктивності.

# РОЗДІЛ 2
# ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 2.1 Вибір мови програмування

У процесі створення та автоматизації системи поставок важливим є вибір мови програмування, яка забезпечить ефективність розробки, гнучкість інтеграції та підтримку необхідних функціональних можливостей. Розглянемо кілька основних мов програмування, які можуть бути використані для даної задачі.

### 2.1.1. Java

Java є однією з найбільш поширених мов програмування, яка була розроблена компанією Sun Microsystems у 1995 році і згодом придбана Oracle Corporation. Завдяки своїй концепції "пиши один раз, запускай де завгодно" (Write Once, Run Anywhere), Java забезпечує високу кросплатформеність і є вибором багатьох великих корпорацій для розробки масштабованих і надійних додатків. Її екосистема включає численні фреймворки, такі як Spring та Hibernate, що значно полегшують розробку корпоративних додатків.

**Переваги:**

- Висока продуктивність та масштабованість
- Широке використання в корпоративних додатках
- Платформа незалежності завдяки JVM (Java Virtual Machine)
- Велика кількість бібліотек та фреймворків (Spring, Hibernate)

**Недоліки:**

- Відносна складність синтаксису для новачків
- Вимагає більше ресурсів під час виконання

### 2.1.2 Python

Python була створена Гвідо ван Россумом і випущена в 1991 році. З тих пір вона стала однією з найпопулярніших мов програмування у світі, завдяки своїй простоті та широким можливостям. Python використовується в різних галузях, включаючи

веб-розробку, аналіз даних, машинне навчання та автоматизацію процесів. Бібліотеки, такі як Pandas і NumPy, роблять Python потужним інструментом для обробки даних, а фреймворки Django і Flask дозволяють швидко створювати веб-додатки.

**Переваги:**

- Простота та зрозумілість синтаксису

- Велика кількість бібліотек та фреймворків для автоматизації та роботи з даними (Pandas, NumPy, SciPy)

- Широкі можливості інтеграції з іншими системами та сервісами (REST API, SOAP)

- Активна спільнота та велика кількість ресурсів для навчання та підтримки

- Підтримка асинхронного програмування (Asyncio)

**Недоліки:**

- Відносно повільніше виконання у порівнянні з компільованими мовами

- Можливість виникнення проблем з керуванням пам’яттю в великих проектах

### 2.1.3 C#

C# була розроблена компанією Microsoft і випущена у 2000 році як частина .NET Framework. Ця мова є основним інструментом для розробки додатків у середовищі Windows, але завдяки .NET Core вона стала кросплатформенною. C# поєднує в собі елементи об'єктно-орієнтованого та функціонального програмування, що робить її гнучким вибором для різних типів проектів. Розробка в C# часто здійснюється у Visual Studio, одному з найпотужніших інтегрованих середовищ розробки (IDE).

**Переваги:**

- Висока продуктивність та безпека

- Добре інтегрується з продуктами Microsoft (SQL Server, Azure)

- Розвинена екосистема для розробки додатків (Visual Studio, .NET)

- Підтримка об'єктно-орієнтованого та функціонального програмування

**Недоліки:**

- Основне використання в середовищі Windows

- Відносно складний синтаксис у порівнянні з Python

### 2.1.4. JavaScript (Node.js)

JavaScript був розроблений Бренданом Айком у 1995 році як мова сценаріїв для веб-браузерів. Згодом він став одним з основних інструментів для веб-розробки. Node.js, платформа для виконання JavaScript на сервері, дозволяє використовувати цю мову для створення серверних додатків з високою продуктивністю завдяки асинхронній обробці подій. JavaScript та Node.js мають широку екосистему модулів та бібліотек, доступних через npm (Node Package Manager), що значно спрощує розробку різних видів веб-додатків.

**Переваги:**

- Висока продуктивність у реальному часі завдяки асинхронній обробці подій

- Велика кількість бібліотек та модулів (npm)

- Добре підходить для розробки веб-додатків

- Підтримка як серверної, так і клієнтської сторін

**Недоліки:**

- Відносна складність у налагодженні та відстеженні помилок

- Обмежена підтримка для обчислювально інтенсивних задач

## 2.2 Обґрунтування вибору мови програмування

Після аналізу переваг і недоліків різних мов програмування, було прийнято рішення використовувати **Python** як основну мову програмування для створення та автоматизації реєстру поставок. Цей вибір обумовлений наступними причинами:

Простота та зрозумілість синтаксису: Python відомий своєю простотою і зрозумілістю, що дозволяє швидко розпочати розробку та знижує поріг входу для нових розробників.

Велика кількість бібліотек та фреймворків: Python має велику кількість готових бібліотек і фреймворків, таких як Pandas, NumPy, SciPy, Django, Flask, що дозволяють швидко реалізувати необхідний функціонал для автоматизації процесів, роботи з даними та інтеграції.

Широкі можливості інтеграції: Python забезпечує легку інтеграцію з різними системами та сервісами через REST API, SOAP, що є важливим для створення комплексної системи управління поставками.

Активна спільнота та велика кількість ресурсів: Завдяки активній спільноті та великій кількості ресурсів для навчання та підтримки, Python забезпечує швидке вирішення проблем, які можуть виникнути в процесі розробки.

Підтримка асинхронного програмування: Завдяки можливостям асинхронного програмування, Python дозволяє ефективно обробляти великі обсяги даних та виконувати багатозадачні процеси, що є критичним для систем управління поставками.

Підсумуємо: вибір Python як основної мови програмування для проекту обґрунтований його численними перевагами, які дозволять створити ефективну, гнучку та масштабовану систему управління поставками товарів.

## 2.3 АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ ДЛЯ АВТОМАТИЗАЦІЇ

### Огляд ERP систем

ERP (Enterprise Resource Planning) системи є комплексними програмними рішеннями, які забезпечують інтеграцію всіх основних бізнес-процесів компанії, таких як фінанси, постачання, виробництво, управління кадрами та інше. Вони дозволяють оптимізувати ресурси підприємства, забезпечуючи централізоване управління та аналіз даних.

### SAP

SAP є однією з провідних ERP-систем у світі, розробленою німецькою компанією SAP SE. Вона надає повний набір інструментів для управління різними аспектами бізнесу, включаючи фінанси, логістику, виробництво та HR. Система підтримує багатомовність та багатовалютність, що робить її придатною для використання у міжнародних компаніях.

**Переваги:**

- Висока функціональність та модульність
- Підтримка масштабованості для великих корпорацій
- Надійність та безпека даних
- Потужні аналітичні інструменти

**Недоліки:**

- Висока вартість ліцензії та впровадження
- Складність у налаштуванні та інтеграції
- Необхідність у кваліфікованих спеціалістах для підтримки

### Oracle ERP

Oracle ERP є потужним інструментом для автоматизації бізнес-процесів, розробленим компанією Oracle Corporation. Вона пропонує широкий спектр функцій

для управління фінансами, постачаннями, проектами та кадрами. Система забезпечує високу продуктивність та надійність, а також підтримує хмарні рішення.

**Переваги:**

- Широкий функціонал та інтеграція з іншими продуктами Oracle
- Підтримка хмарних рішень
- Висока продуктивність та безпека

**Недоліки:**

- Висока вартість
- Складність у налаштуванні та впровадженні
- Необхідність у висококваліфікованих IT-ресурсах

**Microsoft Dynamics**

Microsoft Dynamics є набором ERP та CRM рішень, розроблених компанією Microsoft. Ця система інтегрується з іншими продуктами Microsoft, такими як Office 365 та Azure, що робить її привабливою для компаній, які вже використовують ці продукти. Dynamics забезпечує управління фінансами, операціями, продажами та маркетингом.

**Переваги:**

- Інтеграція з іншими продуктами Microsoft
- Гнучкість та модульність
- Підтримка хмарних рішень

**Недоліки:**

- Висока вартість ліцензій та впровадження
- Складність у налаштуванні
- Вимоги до висококваліфікованих спеціалістів

### Огляд спеціалізованих платформ

На додаток до універсальних ERP систем існують спеціалізовані платформи, які орієнтовані на конкретні бізнес-процеси або галузі. Раніше в тексті було розглянуто такі платформи, як HugeProfit і Торгсофт. Ці системи можуть бути ефективно використані для інтеграції з основною розробкою, забезпечуючи вузькоспеціалізовані рішення для автоматизації систем поставок та управління торгівлею.

#### HugeProfit

HugeProfit, згадана раніше, є спеціалізованою платформою для автоматизації систем поставок і управління запасами. Вона пропонує інструменти для відстеження запасів, управління постачаннями, аналітики та прогнозування попиту. Завдяки інтеграції з існуючими ERP рішеннями, HugeProfit забезпечує високу точність даних і може бути ефективно інтегрована в нашому прикладі для підвищення ефективності управління поставками.

#### Торгсофт

Торгсофт, також, як було зазначено раніше, є програмою українського виробництва для автоматизації торгівлі, яка пропонує інструменти для управління товарообігом, обліку товарів, аналізу продажів та клієнтської бази. Ця система є особливо привабливою для малого та середнього бізнесу завдяки своїй простоті у використанні та доступній вартості. Торгсофт можна інтегрувати з нашою системою для покращення процесів управління торгівлею та підвищення ефективності бізнесу.

Порівняння платформ за функціональністю, масштабованістю, вартістю та іншими критеріями дозволить визначити найкращий варіант для конкретної компанії.

#### Функціональність:

ERP системи, такі як SAP, Oracle ERP та Microsoft Dynamics, забезпечують широку функціональність, охоплюючи всі основні бізнес-процеси.

Спеціалізовані платформи, такі як HugeProfit і Торгсофт, пропонують вузькоспеціалізовані функції, орієнтовані на конкретні потреби.

**Масштабованість:**

Великі ERP системи можуть масштабуватись для великих корпорацій і підтримувати велику кількість користувачів та обробку великих обсягів даних.

Спеціалізовані платформи можуть бути обмежені в масштабованості і підходять більше для малого та середнього бізнесу.

**Вартість:**

ERP системи зазвичай мають високу вартість ліцензій та впровадження, що робить їх дорогими для малих компаній.

Спеціалізовані платформи зазвичай мають нижчу вартість і доступні для менших компаній.

**Інтеграція:**

ERP системи часто легко інтегруються з іншими продуктами та системами завдяки своїй універсальності.

Спеціалізовані платформи можуть вимагати додаткових зусиль для інтеграції з існуючими системами.

Таким чином, можна стверджувати, що вибір платформи залежить від конкретних потреб компанії, її масштабу, функціональних вимог та фінансових можливостей. Використання найкращих практик та інтеграція сильних сторін різних платформ дозволять створити ефективну систему управління поставками.

## 2.4. ВИБІР БАЗИ ДАНИХ

### Поняття бази даних

База даних (БД) є структурованим набором даних, який використовується для зберігання, управління та пошуку інформації. У системах управління поставками БД відіграє ключову роль, забезпечуючи зберігання даних про товари, постачальників, замовлення, логістику та інші аспекти діяльності. Існують різні типи баз даних, кожен з яких має свої переваги та недоліки, що визначають їх придатність для конкретних завдань.

### Реляційні бази даних

Реляційні бази даних (РБД) є найпоширенішим типом баз даних, які використовуються для зберігання структурованої інформації. Вони організовані у вигляді таблиць, що дозволяє легко маніпулювати даними за допомогою мови запитів SQL.

#### MySQL

MySQL є однією з найпопулярніших реляційних баз даних, що поширюється з відкритим кодом. Вона відома своєю високою швидкістю обробки даних і підтримкою великих обсягів інформації, що робить її підходящою для багатьох веб-додатків і систем управління поставками. MySQL є безкоштовною, що знижує витрати на впровадження та підтримку. Однак, обмежена масштабованість у порівнянні з іншими БД може стати проблемою для дуже великих проектів.

#### PostgreSQL

PostgreSQL відома своєю високою надійністю, стабільністю та підтримкою складних запитів. Вона має гнучку структуру, що дозволяє адаптувати базу даних до специфічних потреб проекту. Однією з ключових переваг PostgreSQL є її здатність обробляти складні транзакції та забезпечувати цілісність даних. Однак, налаштування та адміністрування PostgreSQL можуть бути складнішими у порівнянні з іншими БД, що вимагає наявності кваліфікованих фахівців.

#### Oracle Database

Oracle Database є потужною комерційною реляційною базою даних, яка відома своєю високою продуктивністю та надійністю. Вона здатна обробляти великі обсяги даних та складні транзакції, що робить її підходящою для великих корпорацій. Oracle Database пропонує широкий спектр інструментів для управління даними, аналізу та безпеки. Проте, висока вартість ліцензії та складність навчання можуть стати бар'єром для її використання у менших компаніях.

**NoSQL бази даних**

NoSQL бази даних пропонують гнучку структуру зберігання даних і високу масштабованість, що робить їх підходящими для великих розподілених систем та роботи з неструктурованими даними.

**MongoDB**

MongoDB є популярною NoSQL базою даних, що використовує документно-орієнтовану модель зберігання даних. Вона забезпечує високу гнучкість структури, дозволяючи зберігати різні типи даних у одному документі. Це робить MongoDB дуже ефективною для швидкої обробки документів та масштабованості. Проте, відсутність підтримки складних транзакцій може бути обмеженням для деяких додатків.

**Cassandra**

Cassandra є ще однією популярною NoSQL базою даних, розробленою для роботи з великими розподіленими системами. Вона забезпечує високу масштабованість і стійкість до збоїв, що робить її ідеальною для додатків, які вимагають безперебійної роботи та швидкого доступу до даних. Однак, обмежена функціональність для складних запитів може стати недоліком при використанні Cassandra для деяких задач.

**Вибір оптимальної бази даних для поточного проекту**

Вибір оптимальної бази даних для системи управління поставками залежить від специфічних вимог проекту, таких як обсяги даних, вимоги до масштабованості, тип даних та складність транзакцій. У цьому проекті важливо враховувати як реляційні,

так і NoSQL бази даних для забезпечення максимальної ефективності та гнучкості системи.

MySQL та PostgreSQL є чудовими варіантами для реляційних баз даних, пропонуючи високу швидкість обробки та надійність відповідно. PostgreSQL може бути кращим вибором для більш складних та вимогливих до цілісності даних систем. Oracle Database є оптимальним вибором для великих корпоративних проектів, хоча її вартість може бути занадто високою для менш масштабних проектів.

Серед NoSQL баз даних, MongoDB забезпечує високу гнучкість і швидкість обробки документів, тоді як Cassandra пропонує високу масштабованість і надійність для розподілених систем.

На основі аналізу вимог проекту та характеристик різних баз даних, було прийнято рішення використовувати PostgreSQL для даного проекту. Це забезпечить необхідну надійність, гнучкість та підтримку складних запитів, що є критично важливим для системи управління поставками у торговельних мережах.

## 3.5. ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ

Процес розробки програмного забезпечення потребує використання різноманітних інструментів, які допомагають ефективно створювати, тестувати, розгортати та підтримувати програмні продукти. Вибір правильних інструментів є ключовим фактором для успіху проекту, оскільки вони впливають на продуктивність розробників, якість коду та загальний прогрес проекту.

### Середовища розробки (IDE)

Інтегроване середовище розробки (IDE) є основним інструментом для програмістів, яке забезпечує всі необхідні функції для написання, тестування та налагодження коду. Серед найпопулярніших IDE можна виділити кілька:

### PyCharm

PyCharm, розроблене компанією JetBrains, є одним з найпопулярніших середовищ розробки для мови програмування Python. Воно пропонує широкий спектр функцій, включаючи автодоповнення коду, інтеграцію з системами контролю версій, потужний дебаггер та можливості для тестування. PyCharm підтримує різні фреймворки та бібліотеки, що робить його ідеальним для розробки складних проектів.

### Visual Studio Code

Visual Studio Code (VS Code) від Microsoft є багатофункціональним редактором коду, який підтримує численні мови програмування, включаючи Python. Його популярність обумовлена можливістю встановлення різноманітних розширень, які додають нові функції та покращують продуктивність розробників. VS Code також підтримує інтеграцію з Git, що дозволяє легко керувати версіями коду.

### Atom

Atom, розроблений GitHub, є ще одним потужним текстовим редактором, який можна налаштовувати за допомогою пакетів та тем. Він підтримує автодоповнення коду, інтеграцію з Git та інші корисні функції, які полегшують роботу розробників.

Atom є відкритим програмним забезпеченням, що робить його доступним для широкого кола користувачів.

**Sublime Text**

Sublime Text є легким, але потужним текстовим редактором, який часто використовується розробниками завдяки своїй швидкості та можливості налаштування. Незважаючи на те, що Sublime Text не є повноцінним IDE, його можна легко адаптувати для розробки на Python завдяки встановленню відповідних пакетів та плагінів.

**Інші інструменти для розробки**

Окрім середовищ розробки, існують інші інструменти, які можуть бути корисними під час створення програмного забезпечення. До них належать системи контролю версій, інструменти для тестування та деплою, а також інструменти для документування коду.

**Вибір Sublime Text**

Після вивчення всіх варіантів, було вирішено вибрати Sublime Text як основне середовище розробки. Однією з головних причин вибору Sublime Text є його простота та швидкість, та особисті уподобання. Він завантажується майже миттєво і не споживає багато ресурсів, що робить його ідеальним для розробки на менш потужних комп’ютерах. Крім того, Sublime Text підтримує широкий спектр плагінів, які розширюють його функціональність.

# РОЗДІЛ 3
# РЕАЛІЗАЦІЯ ПРОЕКТУ

## 3.1 Опис проекту

Проект полягає у створенні веб-додатку для реєстрації та управління відправленнями вантажів. Основні функції додатку включають реєстрацію користувачів, створення, редагування та видалення інформації про відправлення, а також відстеження стану відправлення.

## 3.2 Вибір технологій

Для розробки веб-додатку були обрані наступні технології:

- Мова програмування: Python

- Веб-фреймворк: Django

- База даних: SQLite (для розробки) та PostgreSQL (для продуктивного середовища)

- Інтерфейс користувача: HTML, CSS, JavaScript (з використанням фреймворку Bootstrap)

- Система контролю версій: Git

## 3.3 Етапи розробки проекту

### 3.3.1 Перша версія коду

На початковому етапі розробки проекту була створена базова структура додатку. Це включало налаштування Django-проекту та створення основних моделей бази даних для зберігання інформації про відправлення.

Налаштування проекту Django

Створення нового проекту за допомогою Django-команди:

```
django-admin startproject shipment_registry
```

Створення моделі Shipment

На першому етапі була створена модель для зберігання даних про відправлення:

```python
from django.db import models

class Shipment(models.Model):
    name = models.CharField(max_length=255)
    status = models.CharField(max_length=255)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name
```

**Міграції бази даних**

Після створення моделі було здійснено міграцію для створення відповідних таблиць у базі даних:

```
python manage.py makemigrations
python manage.py migrate
```

### 3.3.2 Додавання функції реєстрації та входу користувачів

Після створення базової моделі було вирішено додати функції реєстрації та аутентифікації користувачів для забезпечення безпеки даних. Для цього були створені відповідні представлення та форми.

**Реєстрація користувачів**

**Створення форми реєстрації:**

```python
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class RegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

```

**Створення представлення для реєстрації користувачів:**

```python
from django.contrib.auth import login
from django.shortcuts import render, redirect
from .forms import RegisterForm

def register(request):
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('list_shipments')
    else:
        form = RegisterForm()
    return render(request, 'registration/register.html', {'form': form})
```

**Створення представлення для входу користувачів:**

```python
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from django.contrib.auth.forms import AuthenticationForm

def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('list_shipments')
    else:
        form = AuthenticationForm()
    return render(request, 'registration/login.html', {'form': form})
```

### 3.3.3 Додавання функцій управління відправленнями

Наступним кроком було додавання функцій для створення, редагування, видалення та перегляду відправлень. Це дозволяє користувачам якщо з'явиться потреба взаємодіяти зі своїми відправленнями.

**Форма для створення відправлення:**

```python
from django import forms
from .models import Shipment

class ShipmentForm(forms.ModelForm):
    class Meta:
        model = Shipment
        fields = ['name', 'status']
```

**Представлення для створення відправлення:**

```python
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import ShipmentForm

def create_shipment(request):
    if request.method == "POST":
        form = ShipmentForm(request.POST)
        if form.is_valid():
            shipment = form.save(commit=False)
            shipment.user = request.user
            shipment.save()
            messages.success(request, 'Shipment created successfully!')
            return redirect('success')
    else:
        form = ShipmentForm()
    return render(request, 'shipments/create_shipment.html', {'form': form})

```

**Редагування відправлення**

```python
from django.shortcuts import get_object_or_404

def edit_shipment(request, shipment_id):
    shipment = get_object_or_404(Shipment, id=shipment_id)
    if request.method == "POST":
        form = ShipmentForm(request.POST, instance=shipment)
        if form.is_valid():
            form.save()
            messages.success(request, 'Shipment updated successfully!')
            return redirect('success')
    else:
        form = ShipmentForm(instance=shipment)
    return render(request, 'shipments/edit_shipment.html', {'form': form})
```

**Видалення відправлення**

```python
def delete_shipment(request, shipment_id):
    shipment = get_object_or_404(Shipment, id=shipment_id)
    if request.method == "POST":
        shipment.delete()
        messages.success(request, 'Shipment deleted successfully!')
        return redirect('success')
    return render(request, 'shipments/delete_shipment.html', {'shipment': shipment})

```

### 3.3.4 Відображення списку відправлень

Для зручності користувачів було додано функцію відображення списку відправлень. Це дозволяє користувачам переглядати усі свої відправлення на одній сторінці.

```python
def list_shipments(request):
    shipments = Shipment.objects.filter(user=request.user)
    return render(request, 'shipments/list_shipments.html', {'shipments': shipments})
```

**Шаблон для відображення списку відправлень:**

```
{% extends 'base_generic.html' %}

{% block content %}
  <h2>Your Shipments</h2>
  <ul>
    {% for shipment in shipments %}
      <li>{{ shipment.name }} - {{ shipment.status }}</li>
    {% endfor %}
  </ul>
  <a href="{% url 'create_shipment' %}">Create New Shipment</a>
{% endblock %}
```

### 3.3.5 Виправлення непередбачених помилок

На цьому етапі проект було піддано тестуванню для виявлення та виправлення можливих помилок. Були проведені як функціональні, так і інтеграційні тести для забезпечення стабільної роботи додатку. Було виявлено багато прикладів, які довелося розбирати.

**Приклади виправлення помилок**

Однією з помилок була проблема з аутентифікацією користувачів при редагуванні відправлень. Було додано перевірку користувача:

```
def edit_shipment(request, shipment_id):
    shipment = get_object_or_404(Shipment, id=shipment_id)
    if shipment.user != request.user:
        return HttpResponseForbidden()
    if request.method == "POST":
        form = ShipmentForm(request.POST, instance=shipment)
        if form.is_valid():
            form.save()
            messages.success(request, 'Shipment updated successfully!
            return redirect('success')
    else:
        form = ShipmentForm(instance=shipment)
    return render(request, 'shipments/edit_shipment.html', {'form': f
```

## 3.4 Налаштування бази даних

Для зберігання даних було обрано базу даних SQLite для середовища розробки та PostgreSQL для продуктивного середовища. Налаштування бази даних здійснюється через файл `settings.py`.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Для продуктивного середовища використовувалася база даних PostgreSQL:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'shipment_registry',
        'USER': 'mydatabaseuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

### 3.5 Реалізація інтерфейсу користувача

Для створення інтерфейсу користувача використовувалися шаблони HTML, стилі CSS та JavaScript. Bootstrap був обраний для забезпечення адаптивного дизайну.

**Сторінка входу**

```
{% extends 'base_generic.html' %}

{% block content %}
  <h2>Login</h2>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
  </form>
{% endblock %}
```

**Шаблон списку відправлень**

```
{% extends 'base_generic.html' %}

{% block content %}
  <h2>Your Shipments</h2>
  <ul>
    {% for shipment in shipments %}
      <li>{{ shipment.name }} - {{ shipment.status }}</li>
    {% endfor %}
  </ul>
  <a href="{% url 'create_shipment' %}">Create New Shipment</a>
{% endblock %}
```

### 3.6 Розгортання проекту

Для розгортання проекту на продуктивному сервері використовувався сервіс Heroku. Процес розгортання включав налаштування бази даних PostgreSQL та використання системи керування версіями Git для завантаження коду на сервер.

**Кроки розгортання проекту**

**1. Ініціалізація Git-репозиторію:**

```
$ git init
$ git add .
$ git commit -m "Initial commit"
```

**2. Створення нового додатку на Heroku:**

```
$ heroku create
```

**3. Завантаження коду на Heroku:**

```
$ git push heroku master
```

**4. Виконання міграцій бази даних на Heroku:**

```
$ heroku run python manage.py migrate
```

В результаті реалізації проекту був створений повнофункціональний веб-додаток для реєстрації та управління відправленнями вантажів. Додаток забезпечує зручний інтерфейс для користувачів та підтримує основні функції для роботи з відправленнями. Реалізація проекту включала кілька етапів розробки, починаючи від налаштування базової структури до впровадження функцій аутентифікації та управління відправленнями. Завершення проекту включало тестування та виправлення помилок, а також розгортання на продуктивному сервері.

# ВИСНОВКИ РОБОТИ

Розвиток інформаційних технологій продовжує розширювати можливості компаній у сфері автоматизації процесів постачання товарів, що є критичним для підвищення ефективності та конкурентоспроможності на ринку. Автоматизація процесів дозволяє оптимізувати роботу, мінімізувати ризики людських помилок та забезпечити точність і оперативність в обробці даних.

У зв’язку з ростом обсягів торгівлі та зростанням вимог до оперативності поставок, все більше компаній звертаються до впровадження автоматизованих систем управління постачанням. Такі системи дозволяють значно поліпшити процеси реєстрації, контролю та аналізу поставок, забезпечуючи високу точність даних та оперативність у прийнятті рішень.

Враховуючи все сказане, можна зробити висновок, що впровадження веб-додатку для реєстрації та управління відправленнями вантажів має значний потенціал для розвитку. Розробка такої системи дозволяє задовольнити потреби різноманітних користувачів, включаючи операторів логістики, менеджерів та аналітиків, одночасно розширюючи можливості управління поставками. Завдяки покращенню технологій, зростанню зацікавленості в автоматизації та доступності для розробки програмних проектів ця галузь приваблює молодих спеціалістів та підприємців.

У процесі роботи було проведено аналіз предметної області, щоб дослідити основні елементи автоматизації процесів постачання, такі як ідеї, класифікація та основні етапи створення систем. Аналіз ефективності інструментів і технологій, використовуваних для розробки програмних додатків, дозволив визначити найкращий набір інструментів для успішного завершення проекту, який є актуальним і працездатним.

Дослідження та аналіз подібних систем дозволили зрозуміти їхні особливості та технічні рішення, які використовуються в сучасних автоматизованих системах, що призвело до покращення розуміння та технічної компетентності. У результаті цього аналізу були визначені як переваги, так і недоліки цих систем. Результати цього аналізу були використані для подальшої розробки власного програмного додатку.

Вибір мови програмування Python та фреймворку Django був зумовлений широким спектром можливостей, які пропонує цей стек технологій. Зручний і ефективний інструментарій, який надає Python, дозволяє швидко розробляти та масштабувати додатки, забезпечуючи високу продуктивність та надійність. Також було обрано середовище розробки Sublime Text з відповідними плагінами, такими як

Anaconda, SublimeLinter та Jedi, для забезпечення ефективної розробки та тестування коду.

У процесі проектування була розроблена концепція та структура системи, а також визначено цільову аудиторію. Оформлення системи включало розробку інтерфейсу користувача, бази даних та модулів для автоматизації процесів реєстрації поставок. Крім того, було розглянуто різні етапи проектування системи, такі як визначення архітектури, розробка алгоритмів обробки даних та забезпечення безпеки інформації.

Після аналізу результатів можна сказати, що проект був успішно завершений і що його цілі були досягнуті. Система має потенціал для подальшого розвитку. Можливість додати нові функціональні можливості, розширити інтерфейси та інтеграції є дуже привабливими перспективами. Розробка системи дала можливість розширити знання та навички в галузі автоматизації процесів постачання товарів, зокрема щодо використання сучасних технологій та інструментів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


**Документація Django:**

1. Офіційна документація Django: https://docs.djangoproject.com

2. Django REST framework: https://www.django-rest-framework.org

**Документація Python:**

3. Офіційна документація Python: https://docs.python.org

**Книги та навчальні матеріали з Django:**

4. "Django for Beginners" by William S. Vincent

5. "Django for Professionals" William S. Vincent

6. "Two Scoops of Django" Audrey Roy Greenfeld і Daniel Roy Greenfeld

**Ресурси з фронтенд розробки:**

7. Документація Bootstrap:

https://getbootstrap.com/docs/5.0/getting-started/introduction/

8. MDN Web Docs (HTML, CSS, JavaScript):

https://developer.mozilla.org

**Навчальні платформи:**

9. Coursera (курси по Django та веб-розробці)

10. Udemy (курси по Django та веб-розробці)

11. Pluralsight (курси по Django та веб-розробці)

**Спільноти та форуми:**

12. Stack Overflow: https://stackoverflow.com

13. Reddit (підредди з програмування та Django)

14. Django Users Google Group: https://groups.google.com/g/django-users

**Блоги та статті:**

15. Simple is Better Than Complex: https://simpleisbetterthancomplex.com

16. Real Python: https://realpython.com

17. The Django Blog: https://www.djangoproject.com/weblog/

**Допомога у виправленні помилок та консультуванні:**

18. ChatGPT, language model by OpenAI

**Репозиторії з прикладами коду:**

19. GitHub (репозиторії з прикладами проектів на Django): https://github.com

## ПРОГРАМНИЙ КОД

### 1. shipment_registration/manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys



def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'shipment_registration.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)



if __name__ == '__main__':
    main()
```

**2. shipment_registration/shipment_registration/asgi.py**

"""

ASGI config for shipment_registration project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/5.0/howto/deployment/asgi/

"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'shipment_registration.settings')

application = get_asgi_application()

**3. shipment_registration/shipment_registration/settings.py**

"""

Django settings for shipment_registration project.

Generated by 'django-admin startproject' using Django 5.0.6.

For more information on this file, see

```
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-
+(_3qc$n*r8!#$7x&eqj^c06mkd&(np^w+0qq^3$wujaqtu)%g'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'shipments',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'shipment_registration.urls'

TEMPLATES = [
    {
```

```
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'shipment_registration.wsgi.application'



# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'shipment_db',
        'USER': 'your_db_user',
```

```
        'PASSWORD': 'your_db_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}



# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

DATE_INPUT_FORMATS = ['%Y-%m-%d']


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
LOGIN_REDIRECT_URL = 'list_shipments'

LOGOUT_REDIRECT_URL = 'login'
```

**4. shipment_registration/shipment_registration/urls.py**

```
from django.contrib import admin

from django.urls import path

from shipments import views as shipment_views


urlpatterns = [

    path('admin/', admin.site.urls),

    path('register/', shipment_views.register, name='register'),

    path('login/', shipment_views.login_view, name='login'),

    path('logout/', shipment_views.logout_view, name='logout'),

    path('shipments/', shipment_views.list_shipments, name='list_shipments'),

    path('shipments/create/', shipment_views.create_shipment, name='create_shipment'),

    path('shipments/edit/<int:shipment_id>/', shipment_views.edit_shipment,
name='edit_shipment'),

    path('shipments/delete/<int:shipment_id>/', shipment_views.delete_shipment,
name='delete_shipment'),

    path('success/', shipment_views.success, name='success'),

]
```

**5. shipment_registration/shipment_registration/wsgi.py**

```
"""

WSGI config for shipment_registration project.
```

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/5.0/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'shipment_registration.settings')

application = get_wsgi_application()

**6. shipment_registration/shipments/admin.py**

from django.contrib import admin

# Register your models here.

**7. shipment_registration/shipments/apps.py**

from django.apps import AppConfig

class ShipmentsConfig(AppConfig):

    default_auto_field = 'django.db.models.BigAutoField'

```
    name = 'shipments'
```

**8. shipment_registration/shipments/forms.py**

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from .models import Shipment

class RegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

class ShipmentForm(forms.ModelForm):
    class Meta:
        model = Shipment
        fields = ['name', 'description', 'date_shipped', 'status']
```

**9. shipment_registration/shipments/models.py**

```
from django.db import models
from django.contrib.auth.models import User

class Shipment(models.Model):
```

```
    STATUS_CHOICES = (
        ('pending', 'Pending'),
        ('shipped', 'Shipped'),
        ('delivered', 'Delivered'),
    )

    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    description = models.TextField()
    date_shipped = models.DateField()
    status = models.CharField(max_length=20, choices=STATUS_CHOICES)

    def __str__(self):
        return self.name
```

**10. shipment_registration/shipments/urls.py**

```
from django.urls import path
from . import views

urlpatterns = [
    path('create/', views.create_shipment, name='create_shipment'),
    path('success/', views.success, name='success'),
    path('list/', views.list_shipments, name='list_shipments'),  # Добавьте эту строку
]
```

**11. shipment_registration/shipments/views.py**

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.forms import AuthenticationForm
from .forms import ShipmentForm, RegisterForm
from .models import Shipment


def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('list_shipments')
    else:
        form = AuthenticationForm()
    return render(request, 'registration/login.html', {'form': form})


def logout_view(request):
    logout(request)
    return redirect('login')
```

```
def create_shipment(request):
    if request.method == "POST":
        form = ShipmentForm(request.POST)
        if form.is_valid():
            shipment = form.save(commit=False)
            shipment.user = request.user
            shipment.save()
            messages.success(request, 'Shipment created successfully!')
            return redirect('success')
    else:
        form = ShipmentForm()
    return render(request, 'shipments/create_shipment.html', {'form': form})

def edit_shipment(request, shipment_id):
    shipment = get_object_or_404(Shipment, id=shipment_id)
    if request.method == "POST":
        form = ShipmentForm(request.POST, instance=shipment)
        if form.is_valid():
            form.save()
            messages.success(request, 'Shipment updated successfully!')
            return redirect('success')
    else:
        form = ShipmentForm(instance=shipment)
    return render(request, 'shipments/edit_shipment.html', {'form': form})
```

```
def delete_shipment(request, shipment_id):
    shipment = get_object_or_404(Shipment, id=shipment_id)
    if request.method == "POST":
        shipment.delete()
        messages.success(request, 'Shipment deleted successfully!')
        return redirect('success')
    return render(request, 'shipments/delete_shipment.html', {'shipment': shipment})


def list_shipments(request):
    shipments = Shipment.objects.filter(user=request.user)
    return render(request, 'shipments/list_shipments.html', {'shipments': shipments})


def success(request):
    return render(request, 'shipments/success.html')


def register(request):
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('list_shipments')
    else:
        form = RegisterForm()
```

```
    return render(request, 'registration/register.html', {'form': form})
```

**12. shipment_registration/shipments/migrations/0001_initial.py**

```
# Generated by Django 5.0.6 on 2024-06-12 08:14

import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Shipment',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('name', models.CharField(max_length=100)),
                ('description', models.TextField()),
                ('date_shipped', models.DateField()),
```

```
            ('status', models.CharField(choices=[('pending', 'Pending'), ('shipped', 'Shipped'), ('delivered', 'Delivered')], max_length=20)),

            ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),

        ],

    ),

]
```

**13. shipment_registration/shipments/templates/base.html**

```
<!DOCTYPE html>

<html>

<head>

    <title>{% block title %}{% endblock %}</title>

</head>

<body>

    {% if messages %}

        <ul class="messages">

            {% for message in messages %}

                <li{% if message.tags %} class="{{ message.tags }}"{% endif %}>{{ message }}</li>

            {% endfor %}

        </ul>

    {% endif %}

    {% block content %}{% endblock %}

</body>

</html>
```

**14. shipment_registration/shipments/templates/registration/login.html**

```
<!-- shipments/templates/registration/login.html -->
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

**15. shipment_registration/shipments/templates/registration/register.html**

```
{% extends "base.html" %}

{% block title %}Register{% endblock %}

{% block content %}
    <h2>Register</h2>
    <form method="post">
```

```
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Register</button>
    </form>
{% endblock %}
```

**16. shipment_registration/shipments/templates/shipments/create_shipment.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Create Shipment</title>
</head>
<body>
    <h2>Create Shipment</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Save</button>
    </form>
</body>
</html>
```

**17. shipment_registration/shipments/templates/shipments/delete_shipment.html**

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Delete Shipment</title>
</head>
<body>
    <h2>Are you sure you want to delete this shipment?</h2>
    <form method="post">
        {% csrf_token %}
        <button type="submit">Yes, delete</button>
    </form>
    <a href="{% url 'create_shipment' %}">Cancel</a>
</body>
</html>
```

**18. shipment_registration/shipments/templates/shipments/edit_shipment.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Edit Shipment</title>
</head>
<body>
    <h2>Edit Shipment</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Save changes</button>
```

```
    </form>
</body>
</html>
```

**19. shipment_registration/shipments/templates/shipments/list_shipments.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>My Shipments</title>
</head>
<body>
    <h2>My Shipments</h2>
    <ul>
        {% for shipment in shipments %}
            <li>
                {{ shipment.name }} -
                <a href="{% url 'edit_shipment' shipment.id %}">Edit</a> |
                <a href="{% url 'delete_shipment' shipment.id %}">Delete</a>
            </li>
        {% endfor %}
    </ul>
    <a href="{% url 'create_shipment' %}">Create a new shipment</a>
</body>
</html>
```

**20. shipment_registration/shipments/templates/shipments/shipmentsutils.py**

```
import pdfplumber


def parse_invoice(file_path):
    with pdfplumber.open(file_path) as pdf:
        first_page = pdf.pages[0]
        text = first_page.extract_text()
        # Логика для парсинга текста и выделения данных
        parsed_data = {'item_name': 'Parsed Item', 'supplier': 'Parsed Supplier', 'delivery_date': '2024-01-01'}
        return parsed_data
```

**21. shipment_registration/shipments/templates/shipments/success.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Success</title>
</head>
<body>
    <h2>Shipment created successfully!</h2>
    <a href="{% url 'create_shipment' %}">Create another shipment</a>
</body>
</html>
```

**22. shipment_registration/shipments/templates/shipments/tests.py**

```
import pytest
```

```
from django.test import Client

from .models import Shipment


@pytest.fixture

def client():

    return Client()


def test_shipment_creation(client):

    response = client.post('/shipments/create/', {'item_name': 'Test Item', 'supplier': 'Test Supplier', 'delivery_date': '2024-01-01'})

    assert response.status_code == 200

    assert Shipment.objects.count() == 1
```