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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»
спеціальності 121 Інженерія програмного забезпечення
на тему: «Програмне забезпечення для управління роботою складських приміщень»

Виконав:
студент IV курсу, групи КП-03
Шевченко Гліб Олегович _____

Керівник:
Асистент кафедри ПЗКС, д-р філософії,
Юсин Яків Олексійович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ПСТ ФІТ «КНУ
ім. Тараса Шевченка», д.т.н., с.н.с.,
Порєв Геннадій Володимирович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2023 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Шевченку Глібу Олеговичу

1. Тема проєкту «Програмне забезпечення для управління роботою складських приміщень», керівник проєкту Юсин Яків Олексійович, асистент кафедри ПЗКС, доктор філософії, затверджені наказом по університету від «30»__травня__ 2024_ р. №2205-с

2. Термін подання студентом проєкту «16» червня 2024 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - огляд та аналіз існуючих рішень;
   - обґрунтування вибору засобів реалізації;
   - розроблення вебсервісу;
   - аналіз розробленого вебсервісу.

5. Перелік обов'язкового графічного матеріалу:
   - структура бази даних (креслення);
   - структура класів серверної частини (креслення);

- схема модулів системи вебзастосунку (плакат);
- схема архітектури автентифікації Laravel (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2023	
2.	Розроблення та узгодження технічного завдання	28.11.2023	
3.	Розроблення структури програмного забезпечення	15.12.2023	
4.	Підготовка матеріалів дипломного проєкту	15.01.2024	
5.	Розроблення дизайну програмного забезпечення	28.02.2024	
6.	Підготовка матеріалів дипломного проєкту	10.03.2024	
7.	Розроблення програмного забезпечення	30.03.2024	
8.	Тестування програмного забезпечення	30.04.2024	
9.	Підготовка матеріалів дипломного проєкту	15.05.2024	
10.	Підготовка матеріалів дипломного проєкту	20.05.2024	
11.	Оформлення документації дипломного проєкту	30.05.2024	

Студент Гліб ШЕВЧЕНКО

Керівник проєкту Яків ЮСИН

АНОТАЦІЯ

Дипломний проєкт присвячено розробленню програмного забезпечення для управління роботою складських приміщень, яке надає користувачам ефективний інструмент для організації їх робочого процесу.

У рамках проєкту була розроблена архітектура серверної та клієнтської частини вебсервісу. Серверна частина забезпечує збереження та оброблення даних, включаючи створення, збереження та оновлення інформації, що відображає стан складських приміщень. Клієнтська частина, з свого боку, надає інтерфейс користувача, який дозволяє зручно взаємодіяти з програмним забезпеченням.

Для збереження даних була розроблена структура бази даних, яка включає таблиці для зберігання інформації про користувачів, товари, графіки постачання товару та інші необхідні дані. Це дозволяє ефективно керувати складськими приміщенням і забезпечує швидкий доступ до необхідної інформації.

Одна з основних відмінностей розробленого вебсервісу – це основна функція, а саме наявність системи контролю роботи співробітників. Користувач може ставити завдання, де він описує завдання, встановлює час виконання та обирає виконавців. Обрані працівники отримують завдання у власний кабінет. Користувач може стежити за процесом виконання, та отримувати звіт про виконання у застосунку. Також, користувач зможе переглядати історію виконаних завдань кожним користувачем.

Ще однією функцією є можливість створювати та зберігати дані про товар на складі. Користувач матиме доступ до панелі, де зможе описати новий товар за назвою, ціною, вагою, термін придатності та додати власні коментарі. Якщо товар не новий, його можна занести до необхідного сектору складу з датою внесення. Після цього, користувачем редагується лише його кількість в певному секторі складських приміщень.

Цей проєкт має потенціал для полегшення організації та керування складськими приміщеннями, покращення продуктивності та ефективності роботи працівників складу.

ABSTRACT

The diploma project is dedicated to the development of software for warehouse management, providing users with an effective tool to organize their workflow.

The project involved developing the architecture for both the server and client parts of the web service. The server part ensures data storage and processing, including creating, saving, and updating information that reflects the state of the warehouse facilities. The client part, in turn, provides a user interface that allows for convenient interaction with the software.

A database structure was developed for data storage, including tables for storing information about users, products, delivery schedules, and other necessary data. This enables efficient warehouse management and provides quick access to the required information.

One of the main features of the developed web service is the employee work control system. Users can assign tasks, describing the task, setting a completion time, and selecting executors. The selected employees receive the tasks in their personal accounts. Users can monitor the progress and receive a completion report within the application. Additionally, users can view the history of tasks completed by each employee.

Another feature is the ability to create and store data about warehouse products. Users will have access to a panel where they can describe a new product by name, price, weight, expiration date, and add their comments. If the product is not new, it can be added to the necessary warehouse sector with the date of entry. Subsequently, the user can only edit its quantity in a specific warehouse sector.

This project has the potential to streamline the organization and management of warehouse facilities, enhancing the productivity and efficiency of warehouse staff.

ДП.045440-01-90 Програмне забезпечення для управління роботою складських приміщень. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмне забезпечення	5	
	для управління роботою		
	складських приміщень		
	Технічне завдання		
ДП.045440-03-81	Програмне забезпечення	65	
	для управління роботою		
	складських приміщень		
	Пояснювальна записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для управління роботою		
	складських приміщень		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмне забезпечення	17	
	для управління роботою		
	складських приміщень		
	Керівництво користувача		
ДП.045440-06-99	Програмне забезпечення	1	
	для управління роботою		
	складських приміщень		
	Структура класів		
	серверної частини		
ДП.045440-07-99	Програмне забезпечення	1	
	для управління роботою		
	складських приміщень		
	Схема бази даних		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-08-98	Програмне забезпечення	1	
	для управління роботою		
	складських приміщень		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ РОБОТОЮ СКЛАДСЬКИХ ПРИМІЩЕНЬ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Гліб ШЕВЧЕНКО

2023

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для управління роботою складських приміщень.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання користувачам зручного та ефективного інструменту для організації роботи складських приміщень. Програмне забезпечення дозволяє: забезпечити централізацію основних процесів роботи складських приміщень; організувати розподіл завдань між працівниками; відстежувати стан доставки та прийому вантажу.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

1) можливість динамічного оновлення вмісту вебсторінок;
2) можливість реєстрації у сервісі;
3) можливість авторизації;
4) можливість створювати, оновлювати, видаляти складські приміщення та їх сектори;
5) можливість створювати, оновлювати, видаляти інформацію про товар та його статус в певних секторах складу;
6) можливість створювати, оновлювати, видаляти графік постачань;

7) можливість створювати, оновлювати, видаляти завдання для співробітників;
8) можливість переглядати статистику за різний період.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Схема бази даних системи, ER-діаграма»;
   - «Діаграма класів серверної частини»;

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи..... 16.11.2023
Розроблення та узгодження технічного завдання..... 27.11.2023
Розроблення структури вебзастосунку 15.12.2023
Розроблення дизайну сторінок та графічних елементів 03.02.2024
Програмна реалізація вебзастосунку 15.03.2024
Тестування вебзастосунку..... 07.04.2024
Підготовка матеріалів текстової частини проєкту 28.04.2024
Підготовка матеріалів графічної частини проєкту 12.05.2024
Оформлення технічної документації проєкту..... 25.05.2024

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"
Завідувач кафедри
_____ Євгенія СУЛЕМА
"___" _____ 2024 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ РОБОТОЮ СКЛАДСЬКИХ ПРИМІЩЕНЬ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль: Виконавець:

_____ Микола ОНАЙ _____Гліб ШЕВЧЕНКО

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення, яке складається з програм та операційних систем, що виконуються на комп'ютерах або інших пристроях.

CRM – Customer Relationship Management (управління відносинами з клієнтами), система або програмне забезпечення, яке використовується для управління взаємодією компанії з поточними та потенційними клієнтами.

WMS – Warehouse Management System (система управління складом), програмне забезпечення, що допомагає керувати операціями на складі, включаючи прийом, зберігання, комплектацію та відправку товарів.

PHP – мова програмування, яка широко використовується для розробки вебсайтів та вебзастосунків. Вона є скриптовою мовою, що виконується на сервері.

MySQL – реляційна система керування базами даних (СКБД), яка використовується для зберігання та управління даними в структурованому вигляді.

JavaScript – мова програмування, яка використовується для створення інтерактивних вебсторінок. Вона виконується на боці клієнта в браузері.

Laravel – популярний PHP-фреймворк для розробки вебзастосунків з використанням архітектурного шаблону MVC. Він забезпечує зручний синтаксис та багатий набір функцій для швидкої розробки.

Symfony – PHP-фреймворк для розробки вебзастосунків, відомий своєю модульною структурою та можливістю використовувати окремі компоненти в інших проєктах.

OPcache – розширення для PHP, яке прискорює виконання PHP-коду шляхом зберігання попередньо скомпільованих скриптів в оперативній пам'яті.

SQL-ін'єкція – метод атаки на базу даних, коли зловмисник вставляє зловмисний SQL-код в запит, щоб отримати несанкціонований доступ до даних.

XSS-атака – Cross-Site Scripting (міжсайтовий скриптинг), тип атаки на вебзастосунки, при якому зловмисник вставляє шкідливий скрипт у вебсторінку, яку переглядають інші користувачі.

CSRF-атака – Cross-Site Request Forgery (міжсайтове підроблення запиту), атака, яка примушує користувача виконати небажані дії на вебсайті, на якому він автентифікований.

Node.js – середовище виконання JavaScript-коду на сервері, побудоване на базі рушія V8. Дозволяє створювати серверні застосунки на JavaScript.

V8 – високопродуктивний JavaScript-рушій від Google, який використовується в браузері Chrome та в Node.js.

Callback – функція, яка передається як аргумент іншій функції та викликається після завершення певної операції.

Promises – об’єкт, що представляє результат асинхронної операції та може перебувати в одному з трьох станів: очікування, виконаний або відхилений.

Async/await – синтаксис для написання асинхронного коду, який дозволяє працювати з промісами у більш зрозумілому та лінійному стилі.

GitHub – вебсервіс для хостингу та спільної розробки програмного забезпечення, який використовує систему контролю версій Git.

Angular – популярний фреймворк для розробки вебсервісів з використанням мови програмування TypeScript.

ES6 – шоста версія ECMAScript, стандарту, на якому базується JavaScript. Вона містить нові можливості та синтаксис для спрощення розробки.

СУБД – система управління базами даних, програмне забезпечення, яке забезпечує зберігання, управління та доступ до даних.

CLI – Command Line Interface (інтерфейс командного рядка), текстовий інтерфейс для взаємодії з операційною системою або програмним забезпеченням через введення команд.

Artisan – командний інтерфейс у Laravel, який надає набір корисних команд для розробки та управління застосунками.

ORM – Object-Relational Mapping (об'єктно-реляційне відображення), технологія для перетворення даних між несумісними типами систем у об'єктно-орієнтованих мовах програмування.

Eloquent – ORM для фреймворку Laravel, який дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу.

Middleware – проміжний програмний шар у вебзастосунках, який обробляє запити перед тим, як вони досягають основного логічного коду програми.

PHPUnit – тестовий фреймворк для PHP, який використовується для написання та виконання модульних тестів.

API – Application Programming Interface (інтерфейс програмування застосунків), набір правил та протоколів для взаємодії між різними програмними компонентами.

RESTful API – тип API, який використовує принципи REST для взаємодії між клієнтами та серверами.

SPA – Single Page Application (односторінковий застосунок), вебзастосунок, який завантажується як одна HTML-сторінка та динамічно оновлює вміст без перезавантаження сторінки.

Sanctum – пакет для Laravel, який надає простий спосіб автентифікації користувачів API за допомогою токенів.

MFA – Multi-Factor Authentication (багатофакторна автентифікація), метод захисту, який використовує кілька незалежних факторів для підтвердження особи користувача.

OTP – One-Time Password (одноразовий пароль), пароль, який дійсний тільки для одного сеансу або транзакції.

HTML – HyperText Markup Language (мова розмітки гіпертексту), стандартна мова для створення вебсторінок.

DI – Dependency Injection (ін’єкція залежностей), шаблон проєктування, при якому залежності об’єкта передаються йому ззовні, а не створюються ним самостійно.

AOT Compilation – Ahead-Of-Time Compilation (компіляція на етапі випередження), метод компіляції, при якому код компілюється під час розробки, а не під час виконання.

HTTPS – HyperText Transfer Protocol Secure (захищений протокол передачі гіпертексту), розширення HTTP, яке використовує SSL/TLS для захисту даних, що передаються між клієнтом та сервером.

GDPR – General Data Protection Regulation (Загальний регламент захисту даних), законодавчий акт Європейського Союзу, який встановлює правила щодо обробки персональних даних фізичних осіб. Введений у дію 25 травня 2018 року, GDPR посилює права громадян на захист їхніх даних та накладає суворі вимоги на компанії щодо збору, зберігання та обробки персональних даних.

HIPAA – Health Insurance Portability and Accountability Act (Закон про перенесення та підзвітність медичного страхування), законодавчий акт США, прийнятий у 1996 році, який встановлює стандарти для захисту конфіденційності та безпеки медичної інформації. HIPAA зобов’язує медичні установи, страхові компанії та інші організації, що працюють з медичними даними, забезпечувати конфіденційність, цілісність та доступність цієї інформації.

AWS – Amazon Web Services, платформа хмарних обчислень, яка надає широкий спектр послуг, включаючи обчислювальні потужності, зберігання даних, бази даних, аналітику, машинне навчання та інші інструменти для розробки та розгортання застосунків. AWS є дочірньою компанією Amazon і є однією з найбільших хмарних платформ у світі.

MVC – Model-View-Controller (архітектурний шаблон, який використовується під час проєктування та розроблення програмного забезпечення. Він розділяє застосунок на три основні компоненти: модель

(Model) – відповідає за управління даними і логікою бізнесу; представлення (View) – відповідає за відображення інформації користувачу; контролер (Controller) – відповідає за обробку користувацьких запитів і взаємодію між моделлю та представленням).

ORM – Object-Relational Mapping (технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування. ORM дозволяє розробникам працювати з базою даних за допомогою об'єктів у своєму коді, замість того щоб писати SQL-запити. Це спрощує розробку і підтримку коду, а також забезпечує абстракцію від специфіки конкретної СУБД).

Фреймворк – Framework (програмне середовище, яке спрощує та прискорює створення програмного забезпечення. Фреймворки надають набір готових компонентів, бібліотек і шаблонів для вирішення типових завдань, що дозволяє розробникам зосередитись на унікальних аспектах своїх проєктів замість того, щоб повторно вирішувати загальні проблеми).

ВСТУП

У сучасному світі, де бізнес процвітає у динамічному середовищі, ефективне управління складськими приміщеннями має ключове значення для забезпечення конкурентоспроможності підприємств. З ростом обсягів виробництва та збільшенням кількості товарів, які надходять і відправляються щодня, необхідність розроблення надійного програмного забезпечення для управління складськими приміщеннями стає все більш очевидною.

Програмне забезпечення для управління складськими приміщеннями дозволяє оптимізувати складські операції, підвищуючи продуктивність і точність обробки замовлень. Воно сприяє покращенню організації складських процесів, включаючи відстеження запасів, контроль над логістикою, а також аналіз і візуалізацію даних, пов’язаних із роботою складу.

Метою даного дипломного проєкту є створення програмного забезпечення для управління складськими приміщеннями, яке забезпечуватиме повний спектр функцій, необхідних для ефективного контролю над складськими операціями. Воно має надавати зручний інтерфейс для планування та обліку запасів, а також розширені можливості для збору, обробки та аналізу даних.

Розроблене програмне забезпечення допоможе підвищити ефективність роботи складських приміщень, оптимізувати процеси обліку та управління запасами, що сприятиме більш раціональному використанню ресурсів. Воно також дозволить поліпшити комунікацію між різними підрозділами підприємства та забезпечити своєчасну та точну обробку даних.

Застосовуючи сучасні підходи та технології в управлінні складськими приміщеннями, компанії отримають можливість досягти високого рівня

організованості та продуктивності у своїй діяльності, що в свою чергу позитивно позначиться на їх конкурентоспроможності.

1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Zoho Inventory

Zoho Inventory – це комплексний інструмент для управління складськими операціями та запасами [1]. З його допомогою користувачі можуть ефективно керувати товарами, відслідковувати їх рух, управляти замовленнями та постачаннями. Zoho Inventory підтримує різноманітні платформи продажу та легко інтегрується з іншими бізнес-застосунками, надаючи все необхідне для успішного управління складськими процесами.

Основні функції Zoho Inventory включають:

1. Відстеження руху товарів у режимі реального часу, облік на різних складах, ведення складських звітів.
2. Управління замовленнями від клієнтів, постачальників та обробка повернень.
3. Підключення до інших сервісів, таких як бухгалтерія, CRM та платформи електронної комерції для повної автоматизації бізнес-процесів.
4. Отримання інформації про запаси, продажі та продуктивність для ухвалення ефективних рішень.

Основна функція Zoho Inventory – це управління «списками», що відповідають за облік та організацію товарів, замовлень та операцій на складі, як на рисунку 1.1. Користувач може створювати нові списки для різних категорій товарів або замовлень, наприклад, «Постачання», «Продажі», «Запаси», «Транзит» та інші. Кожен список може містити детальний опис позицій, їх кількість, вартість та інші дані для ефективного управління. Ця функція дозволяє користувачам створювати списки товарів і визначати терміни їх оброблення або виконання. Користувачі можуть планувати та керувати запасами, ставити пріоритети та вказувати тимчасові рамки для кожної категорії товарів або замовлень. Крім того, Zoho Inventory дозволяє користувачам організовувати запаси за допомогою тегів, що

сприяє більш зручному пошуку та класифікації товарів. Таким чином, функція «списків» у Zoho Inventory надає користувачам можливість створювати та організовувати свої товари, замовлення та складські операції у вигляді списків, що спрощує їх контроль та управління.

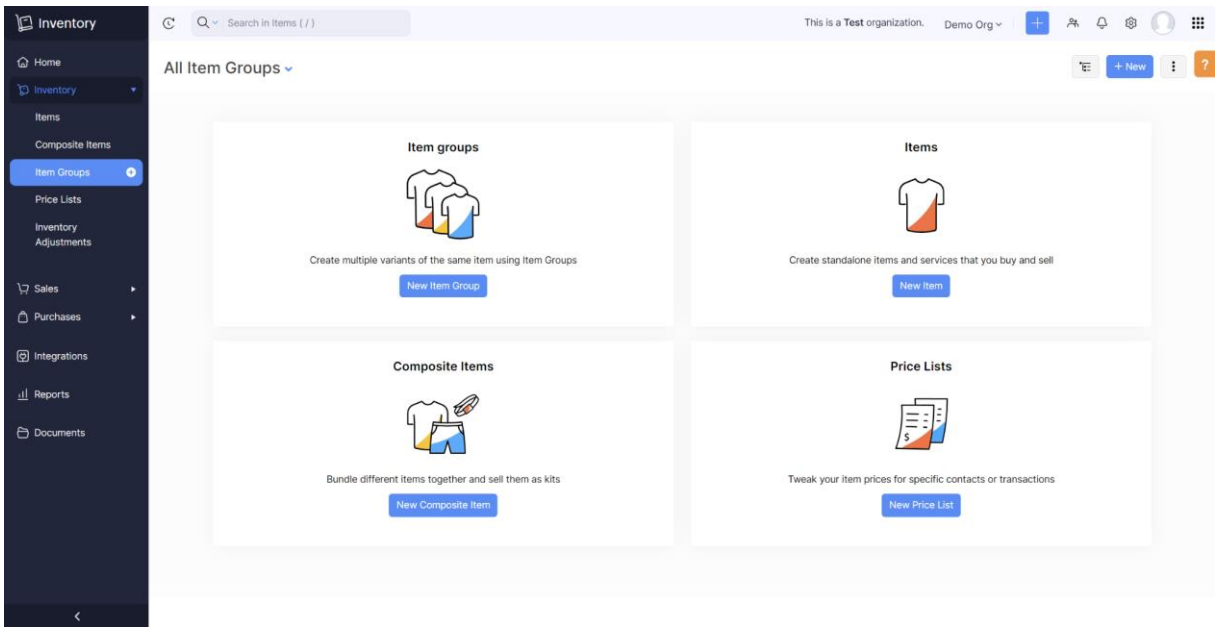


Рис. 1.1. Створення списків

Zoho Inventory підтримує інтеграцію з різними застосунками, як Google Workspace, так і власної розробки (рис. 1.2). Інтеграція з іншими застосунками важлива, оскільки дозволяє бізнесу працювати більш ефективно та цілісно. Коли Zoho Inventory інтегрується з бухгалтерськими системами, CRM, платформами електронної комерції та іншими інструментами, це забезпечує безперервний обмін даними між ними. Така взаємодія автоматизує процеси, зменшує ризик помилок та економить час, оскільки користувачі можуть працювати з повним набором даних у єдиній системі. Крім того, це сприяє кращому контролю над операціями та покращенню якості обслуговування клієнтів завдяки синхронізації інформації між різними департаментами.

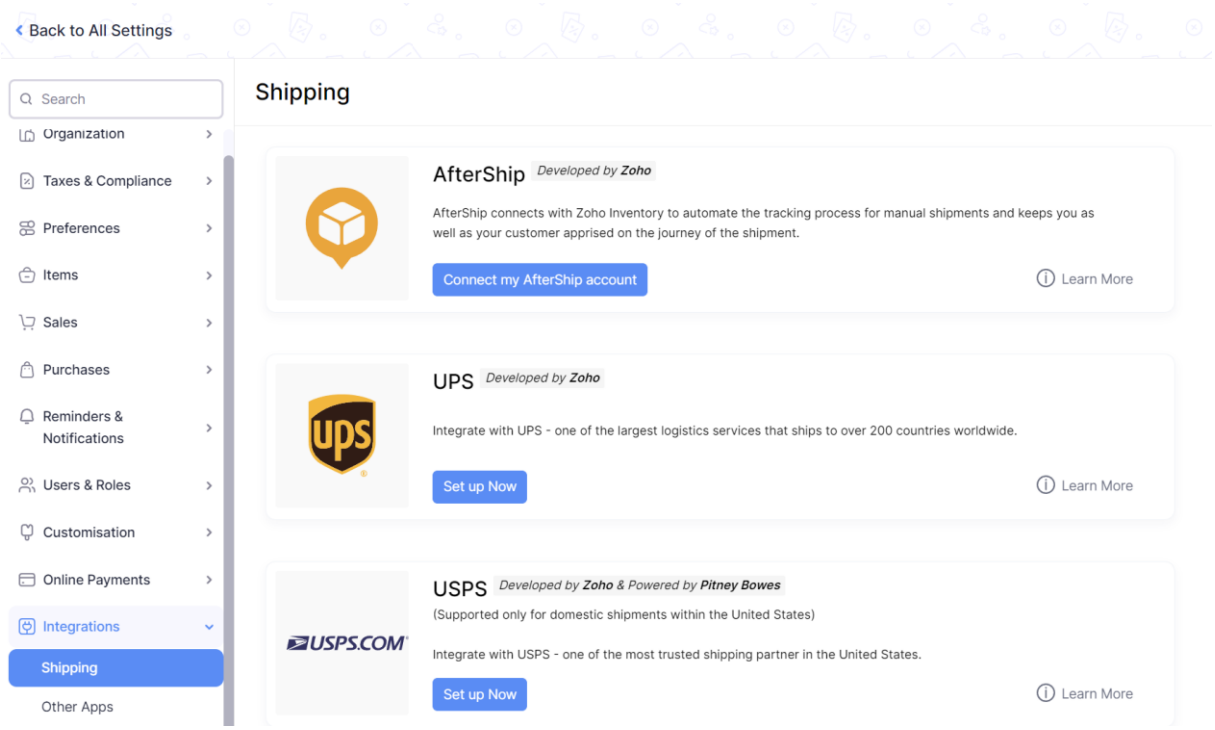


Рис. 1.2. Список застосунків для інтеграції

Іншою корисною функцією є головна панель застосунку (рис. 1.3), на якому зображена велика кількість корисної інформації: огляд запасів, статистика продажів, замовлення клієнтів, закупівлі, нагадування та сповіщення, та інше. Це дозволяє користувачам швидко оцінити поточний стан складського господарства та виконувати необхідні дії. На сторінці розташовано вікно підтримки, та можливість потрапити в особистий кабінет. Дизайн сторінки є простим та зрозумілим. Знизу зображеня детальна статистика по кожному товару. Показано скільки продається товару. Який прибуток йде з цього товару. Можна побачити з яких регіонів більш усього замовлють певний товар. Також показана загальна кількість товару і груп товарів. Є можливість передивитися посібник для початківців та побачити, як працювати з головною сторінкою. Користувач може здійснити пошук по товарам та інформації на сторінці. Сторінка дозволяє змінити показ статистики згідно з бажання користувача. Найголовніше, що користувач може побачити скільки товару він має відправити, та скільки вже надійшло до покупців.

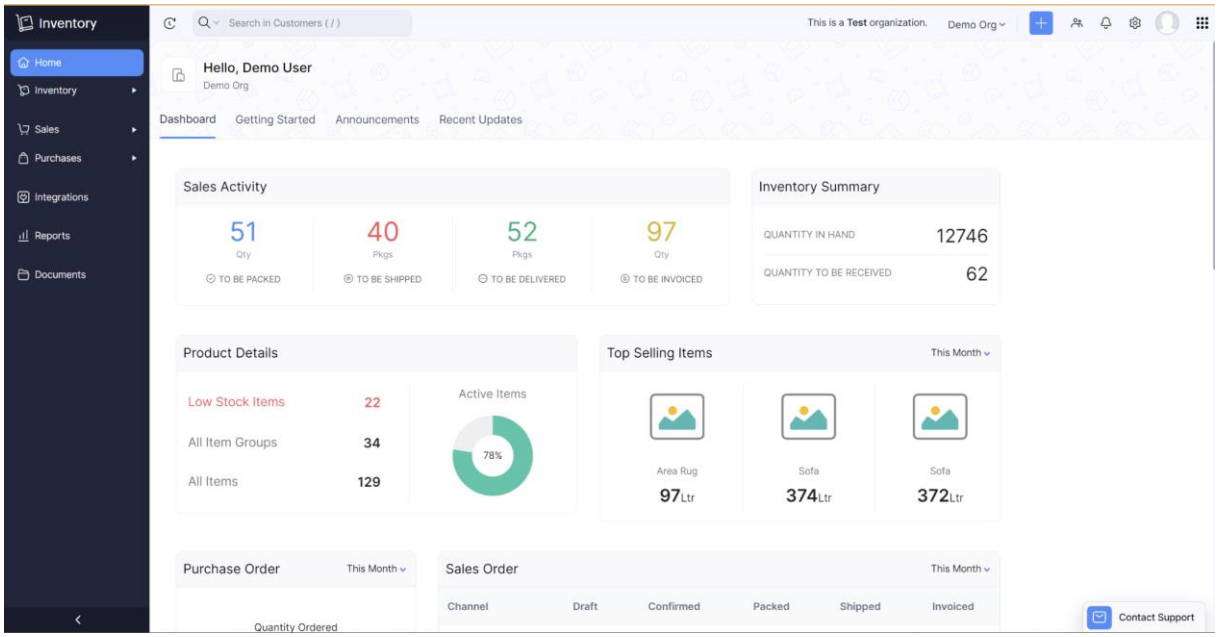


Рис. 1.3. Вікно «Home»

Також, Zoho Inventory має інструмент, який надає користувачам доступ до детальної аналітики та звітів про різні аспекти складських операцій. Сторінка «Reports» допомагає користувачам отримати вичерпну інформацію про ефективність бізнес-процесів, а також приймати обґрунтовані рішення на основі даних, як на рисунку 1.4.

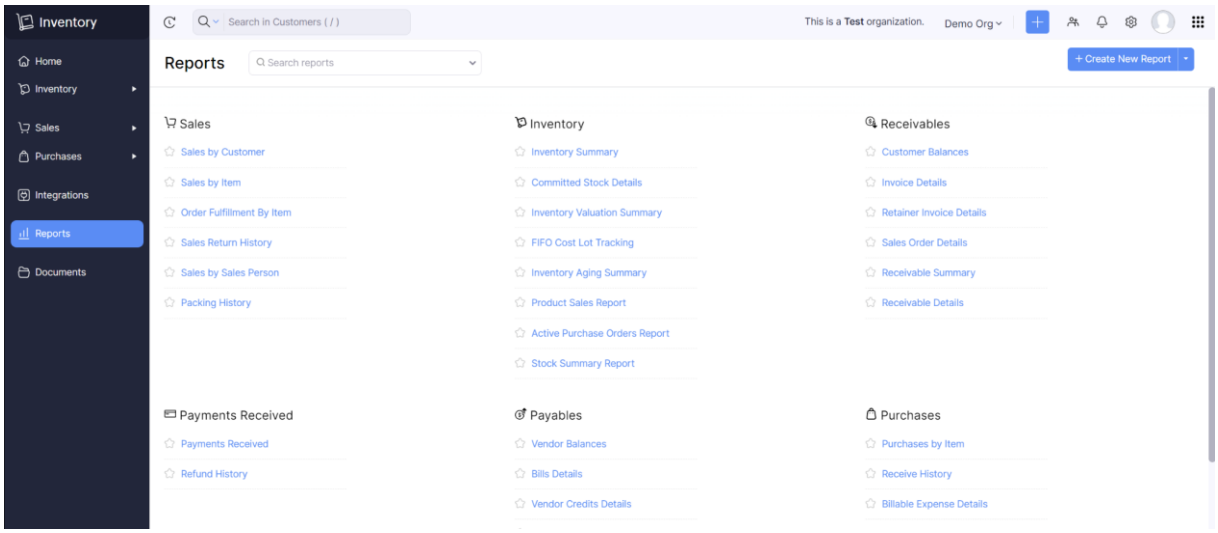


Рис. 1.4. Вікно «Reports»

Сторінка Reports також надає можливість візуалізувати дані за допомогою графіків, діаграм та таблиць, що полегшує розуміння інформації та допомагає користувачам приймати зважені рішення. Вона надає

можливість експортувати звіти в різних форматах, таких як Excel або PDF, для подальшого використання чи обміну з іншими.

Загалом, Reports дозволяє користувачам контролювати ефективність бізнесу, оперативно оцінювати поточний стан справ і приймати стратегічні рішення на основі точних даних. Завдяки цьому інструменту користувачі можуть швидко виявляти можливості для покращення процесів та оптимізації роботи складу.

Незважаючи на всі переваги, Zoho Inventory має певні недоліки. Наприклад, деякі користувачі зазначають, що тарифні плани можуть бути дорогими для малого бізнесу, обмежуючи доступ до розширених функцій. Крім того, система може мати певні обмеження на кількість користувачів або транзакцій, що може стати проблемою для великих підприємств з високим обсягом операцій.

Іншим недоліком є те, що новим користувачам може знадобитися деякий час для освоєння інтерфейсу та функціональних можливостей Zoho Inventory, оскільки програма пропонує багато можливостей для управління запасами та замовленнями. Деякі користувачі також скаржаться на обмежену інтеграцію з деякими системами, які вони хотіли б використовувати разом з Zoho Inventory.

1.2. NetSuite WMS

NetSuite WMS – це комплексне рішення для управління складськими операціями, яке забезпечує ефективну організацію та контроль складських процесів [2]. За допомогою NetSuite WMS користувачі можуть оптимізувати облік товарів, керувати запасами, відстежувати переміщення та виконувати замовлення. Головна перевага застосунку – програмне забезпечення доступне через різні пристрої, що дозволяє користувачам керувати складом з мобільних пристроїв або через веббраузер.

Сервіс пропонує такі функції:

1. Відстеження рівнів запасів та автоматичне оновлення даних.

2. Планування та організація товарів на складі для максимальної ефективності.
3. Взаємодія з різними бізнес-застосунками для цілісного управління.
4. Прискорення процесів прийому, обробки та відвантаження замовлень через функцію виконання замовлень.

Функція виконання замовлень в NetSuite WMS надає користувачам інтуїтивно зрозумілий спосіб оброблення та відвантаження замовлень. Коли клієнт розміщує замовлення, система автоматично відстежує запаси та підказує користувачу, з яких локацій складу можна відвантажити потрібні товари. Це допомагає оптимізувати процеси та прискорити виконання замовлень.

Крім того, користувач може розпочати процес виконання замовлення з вибору конкретного замовлення зі списку. Система виводить всі необхідні дані, включаючи список товарів, що входять до замовлення, та їхні доступні кількості на складі. Після цього користувач може почати збирати товари для замовлення, використовуючи функції сканування штрих-кодів для точного обліку. Користувач може легко переходити між завданнями та контролювати статус кожного замовлення. Наприклад є можливість самостійно обрати який товар буде запакований в різні групи через технологію «Wave Pick-up», як на рисунку 1.5.

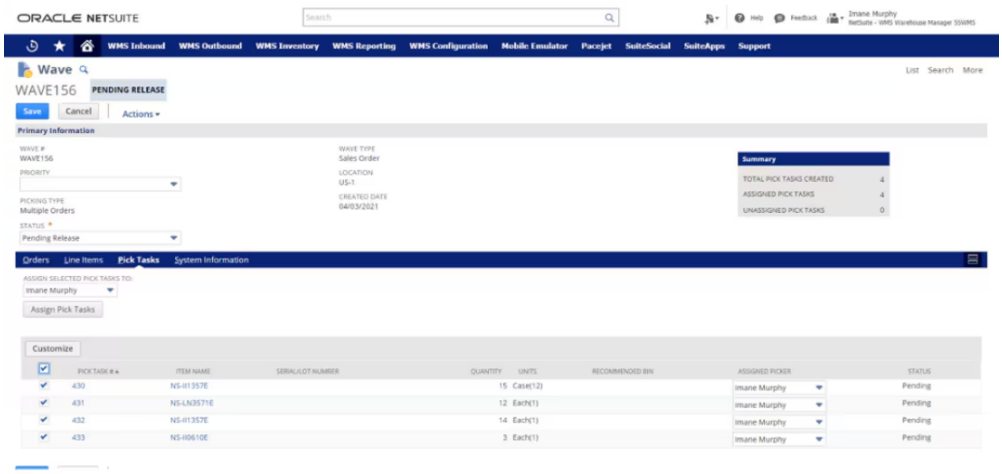


Рис. 1.5. Вікно «Wave Pick-up»

Загалом функція виконання замовлень в NetSuite WMS полегшує користувачам управління всім процесом обробки замовлень, забезпечуючи високу точність та ефективність у роботі зі складом і замовленнями.

Іншою важливою перевагою застосунку можна вважати «Mobile Warehouse Management» (рис. 1.6). Виходячи з назви, дозволяє користувачам легко відстежувати та управляти своїми складськими операціями прямо зі свого мобільного пристрою. За допомогою мобільного застосунка користувачі можуть здійснювати всі основні функції, включаючи перевірку рівня запасів, відстеження замовлень та виконання складських операцій.

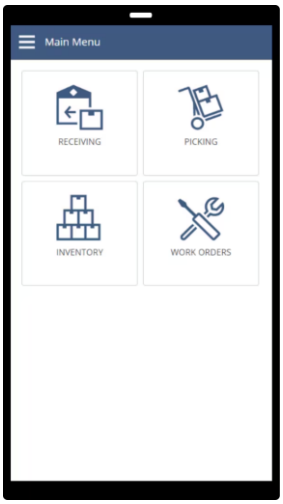


Рис. 1.6. Головне меню мобільного застосунку NetSuite WMS

Після завантаження та встановлення мобільного застосунка користувачі отримують доступ до всіх наявних складських даних, які автоматично синхронізуються з основною системою NetSuite WMS. Користувач може переглядати списки замовлень, перевіряти статуси виконання, здійснювати пошук товарів та багато іншого.

Також, користувач може виконувати функції збору та упаковки замовлень, скануючи штрих-коди товарів за допомогою камери телефону. Це допомагає підвищити точність та швидкість виконання замовлень. Для відстеження замовлень та доставки користувач може використовувати інструменти відстеження в реальному часі, доступні через мобільний

застосунок.

Загалом, «Mobile Warehouse Management» з NetSuite WMS забезпечує користувачам гнучкість та зручність у керуванні складськими операціями з будь-якого місця та в будь-який час.

Незважаючи на всі переваги, NetSuite WMS має певні недоліки:

1. NetSuite WMS пропонує багато функцій і можливостей для налаштування, що може зробити процес налаштування та конфігурування системи складним для нових користувачів.
2. У деяких випадках користувачі можуть відчувати зниження продуктивності або повільне завантаження сторінок, особливо коли в системі працює велика кількість користувачів.
3. Для користувачів з унікальними потребами або специфічними вимогами деякі функції NetSuite WMS можуть бути обмеженими або не надавати всіх необхідних можливостей.
4. Впровадження NetSuite WMS може зайняти значний час, особливо для великих підприємств з великим обсягом складських операцій. Це може уповільнити впровадження та вимагає ретельного планування.

Отже, NetSuite WMS – це потужне рішення для управління складськими операціями, яке покриває більшість вимог підприємств до оптимізації роботи складу. Серед його переваг варто виділити можливість детального управління запасами та замовленнями, автоматизацію складських процесів, та тісна інтеграція з мобільними пристроями. А серед недоліків складність налаштування та час, необхідний для повного впровадження системи на підприємстві. Це може бути перешкодою для нових користувачів.

1.3. Dilovod

Dilovod – це інтуїтивно зрозумілий інструмент для обліку складу, який допомагає організовувати та контролювати товарні запаси на складі [3]. З

Dilovod можна легко відстежувати кількість товару, рух товару, а також проводити інвентаризацію. Доступ до системи можна отримати через веббраузер, а також за допомогою мобільного застосунку.

Основні функції застосунку Dilovod:

1. Детальний опис кожного товару, включаючи назву, категорію, ціну та кількість.
2. Можливість встановлювати мінімальний рівень запасу та отримувати сповіщення про необхідність поповнення.
3. Реєстрація всіх надходжень та вибуттів товарів зі складу.
4. Функція для проведення швидкої та точної інвентаризації товарних запасів.
5. Автоматичне генерування звітів щодо стану складу, продажів та закупівель.

Завдяки функціям автоматизації обліку товарів на складі, Dilovod забезпечує ефективне управління складськими операціями. Можливо прискорити процеси списання, оприбуткування та інвентаризації товару, підключивши сканер штрих-кодів до програми обліку матеріалів на складі (рис. 1.7).

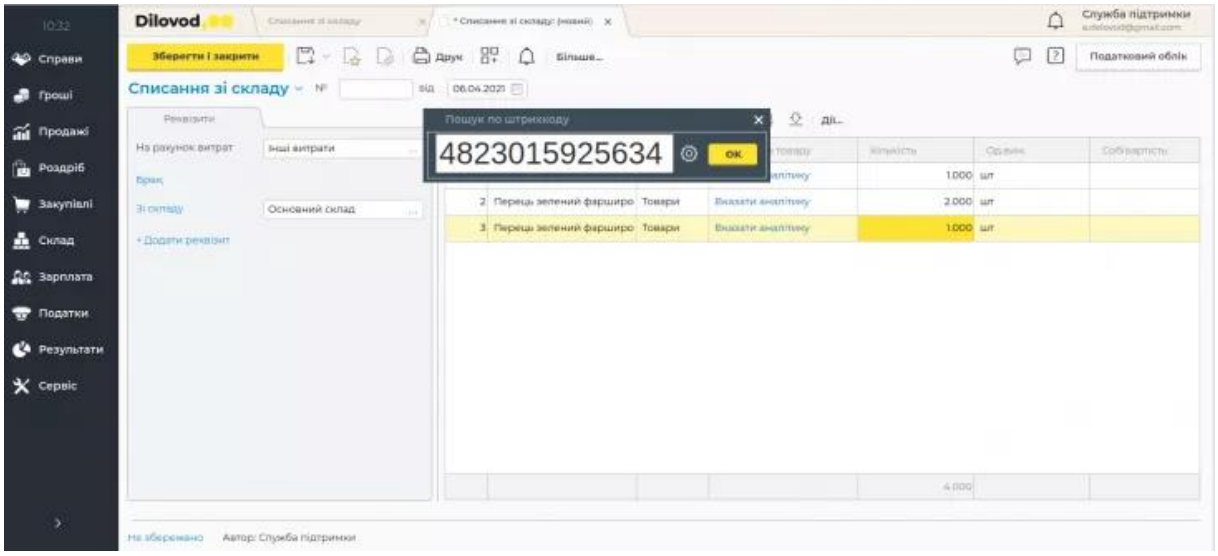


Рис. 1.7. Зчитування та пошук за штрих-кодом

Іншою корисною функцією є підключення служб доставки в системі обліку складу. Dilovod надає можливість ефективно керувати службами доставки для забезпечення швидкого та надійного обслуговування клієнтів. Функція дозволяє організувати власну доставку або підключити сторонні служби. Здійснюється процес через друк транспортних накладних (рис. 1.8).

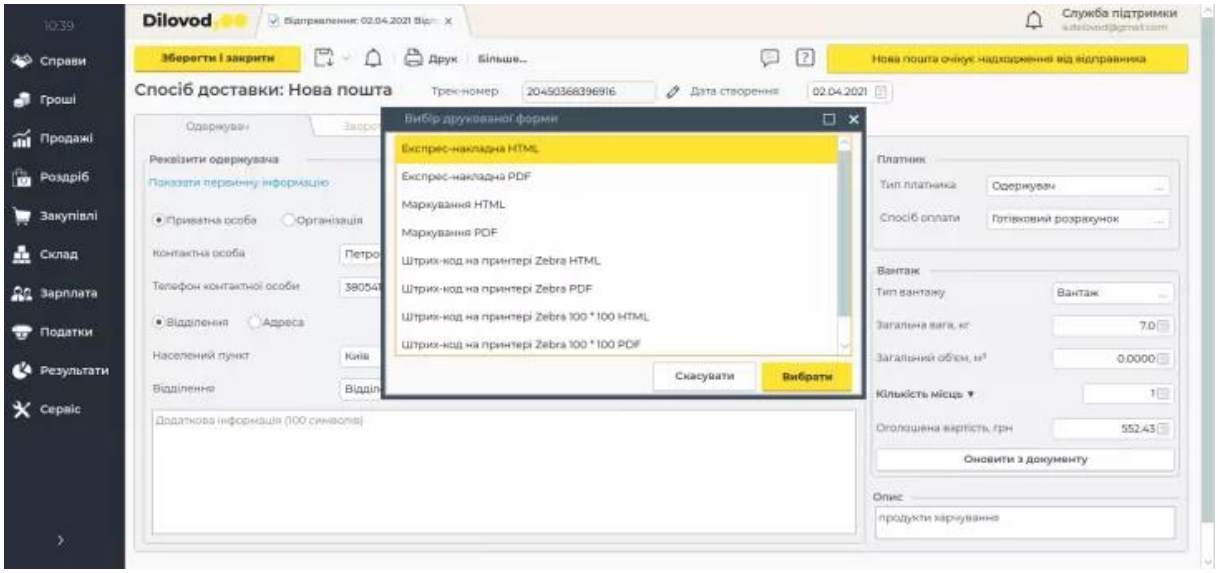


Рис. 1.8. Меню обрання способу доставки

Також, з Dilovod можна оперативно відстежувати склад онлайн, за допомогою докладних складських звітів. Це допоможе підтримувати оптимальний набір товарів, забезпечити вчасне постачання, та управляти роботою складу. Здійснюється це через функцію звіт «Аналіз складських запасів» (рис. 1.9).

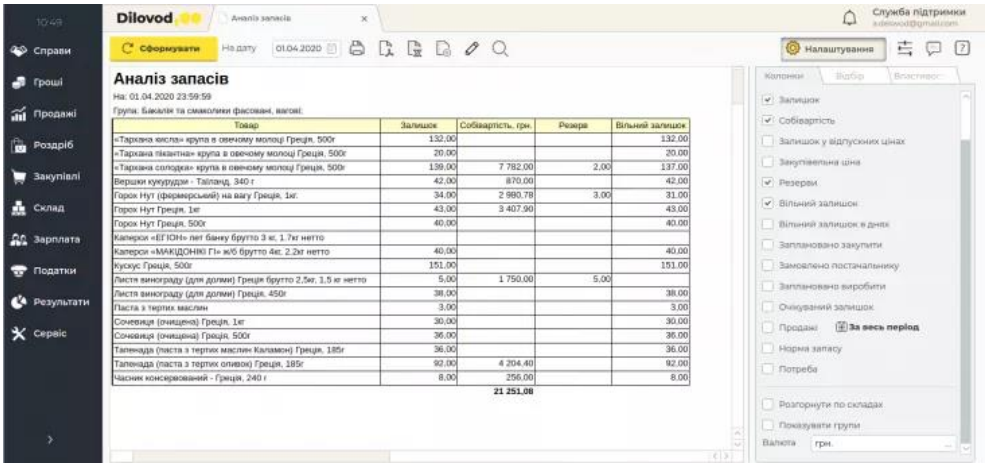


Рис. 1.9. Меню аналізу запасів

Функція надає глибокий огляд стану складських запасів, допомагаючи підприємствам оптимізувати управління асортиментом. Цей звіт ідентифікує товари, які користуються найбільшим попитом, а також ті, що мають низьку популярність. Завдяки цьому, можна ефективно зосередити зусилля на своєчасному поповненні популярних позицій та переглянути доцільність зберігання товарів, що не користуються попитом. Аналіз руху товарів та історичні дані дозволяють приймати обґрунтовані рішення щодо замовлень та оптимізації складських операцій. Звіт також містить візуалізацію даних у вигляді графіків та діаграм, що спрощує розуміння ситуації зі складськими запасами та допомагає приймати правильні рішення щодо управління складом.

Хоча Dilovod є потужною системою для обліку складу та торгівлі, проте, як і будь-який інший програмний продукт, має свої недоліки. Деякі з можливих недоліків Dilovod можуть включати:

1. Враховуючи специфічність українського ринку, можливості мови інтерфейсу чи документації можуть бути обмеженими, особливо якщо розробник не надає повну підтримку української мови.
2. Обмежена технічна підтримка призводить до ступору в роботі. Доступність та якість технічної підтримки можуть впливати на загальний досвід використання Dilovod, особливо якщо є складні питання або потреба в оперативному вирішенні проблем.
3. Вартість підписки або використання Dilovod може стати важливим фактором для невеликих бізнесів, оскільки витрати на програмне забезпечення можуть бути високими.
4. Dilovod підтримує інтеграцію зі своїми сервісами, можливості інтеграції через це, можуть бути обмеженими, що може спричинити труднощі для тих, хто використовує різноманітні програмні рішення.

1.4. Висновки

Ринок програмного забезпечення для управління складом досить зрілий, оскільки існує кілька потужних та добре відомих гравців, які вже завоювали довіру користувачів. У цій сфері новим учасникам ринку потрібно пропонувати унікальні функції та надавати бездоганне обслуговування, щоб стати конкурентоспроможними. Відомі сервіси, такі як NetSuite WMS, Zoho inventory та Dilovod облік складу, надають базову функціональність для управління запасами, контролю замовлень та аналітики. Проте жоден з цих застосунків не надає можливості видачі завдань конкретним робітникам. Всі ці застосунки розраховані на великий бізнес і через це вони занадто дорогі.

З точки зору перспектив нових розробників, варто зосередитися на розробленні інноваційних функцій, таких як інтеграція з сучасними технологіями, вдосконалення автоматизації складських процесів та підвищення користувацького досвіду за рахунок інтуїтивно зрозумілих інтерфейсів. Розроблення сервісів із функціями для різних бізнес-моделей може стати ключовою перевагою.

Для того щоб завоювати ринок та переманити користувачів від відомих постачальників, новим розробникам необхідно впроваджувати маркетингові стратегії, піар-акції та програми лояльності. Це допоможе створити новий образ бренду, який буде асоціюватися з інноваціями, якістю та надійністю.

Загалом, перспективи для нових розробників у цій сфері існують, але успіх буде залежати від здатності пропонувати нові підходи та краще обслуговування, щоб здобути довіру та задовольнити потреби користувачів.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Мови програмування

2.1.1. *PHP*

PHP – це високорівнева мова програмування для створення динамічних вебсайтів та вебзастосунків, яка широко використовується завдяки своїй простоті, гнучкості та потужності. PHP є основним вибором для багатьох розробників вебсайтів завдяки своїм численним перевагам і підтримці великих спільнот користувачів [4].

Однією з ключових переваг PHP є його легкість у вивченні та використанні. Вбудований синтаксис PHP нагадує C та Perl, що робить його зрозумілим для програмістів, знайомих з цими мовами. Навіть новачки можуть швидко освоїти PHP завдяки великій кількості доступних навчальних матеріалів та активній спільноті, яка готова допомогти. Ця простота дозволяє швидко створювати вебзастосунки та прототипи, що є важливим у швидкому світі веброзробки [5].

PHP легко інтегрується з багатьма базами даних, такими як MySQL, PostgreSQL, Oracle, та іншими. Це дозволяє розробникам створювати потужні та масштабовані вебзастосунки. Крім того, PHP підтримує інтеграцію з іншими мовами програмування та технологіями, такими як HTML, CSS, JavaScript, що дозволяє створювати повнофункціональні вебсайти. Гнучкість PHP забезпечує можливість використання різних архітектур і підходів до розробки, що робить його універсальним інструментом для веброзробників.

Також, PHP має багатий набір фреймворків, які полегшують розробку вебзастосунків. До популярних фреймворків належать Laravel, Symfony, CodeIgniter, Zend Framework та інші. Ці фреймворки надають готові рішення для типових завдань веброзробки, що дозволяє розробникам зосередитися на унікальних аспектах своїх проєктів. Вони також сприяють кращій організації коду, що покращує його читабельність та підтримку. Багато з цих

фреймворків також включають вбудовані засоби безпеки, що забезпечує захист вебзастосунків від поширених загроз.

Важливо, що PHP має одну з найбільших та найактивніших спільнот серед мов програмування. Це означає, що розробники можуть легко знайти допомогу, навчальні ресурси, плагіни та бібліотеки, що полегшує та прискорює процес розробки. Активна спільнота також сприяє швидкому виправленню помилок та постійному вдосконаленню мови. Завдяки великій кількості доступних ресурсів PHP залишається актуальним і затребуваним у світі веброзробки.

PHP забезпечує високу продуктивність вебзастосунків завдяки своїй здатності ефективно обробляти великий обсяг запитів. Завдяки кешуванню та оптимізації коду, PHP-застосунки можуть працювати швидко та з мінімальними затримками. Це особливо важливо для вебсайтів з високим трафіком, де швидкість завантаження сторінок є критичною для користувацького досвіду. PHP підтримує різноманітні кешувальні механізми, такі як OPcache, що дозволяють зменшити час виконання скриптів та покращити загальну продуктивність системи.

Безпека є ключовим аспектом будь-якої веброзробки, і PHP пропонує численні вбудовані функції для захисту вебзастосунків. Це включає засоби для запобігання SQL-ін’єкціям, XSS-атакам та CSRF-атакам. Крім того, активна спільнота PHP постійно працює над виявленням та виправленням вразливостей, що забезпечує високий рівень безпеки для розробників, які використовують цю мову. Використання сучасних фреймворків, таких як Laravel, додатково підвищує безпеку завдяки вбудованим механізмам автентифікації та авторизації [6].

Відомо, що PHP підтримує об’єктно-орієнтоване програмування. Ця можливість дозволяє розробникам створювати більш структуровані коди. ООП у PHP включає підтримку наслідування, інкапсуляції та поліморфізму, що полегшує створення масштабованих та підтримуваних застосунків. Завдяки ООП розробники можуть організовувати код у класи та об’єкти, що

робить його більш зрозумілим та легким у підтримці. Це особливо корисно для великих проєктів, де важлива модульність та структурованість коду.

PHP дозволяє створювати масштабовані вебзастосунки, які можуть легко адаптуватися до зростання трафіку та вимог користувачів. Завдяки підтримці різноманітних фреймворків та бібліотек, PHP-застосунки можуть ефективно обробляти великі обсяги даних та забезпечувати стабільну роботу навіть при значному навантаженні. Масштабованість є важливою властивістю для сучасних вебсайтів та сервісів, що постійно розвиваються.

Підсумовуючи, мова програмування PHP є потужним та гнучким інструментом для створення динамічних вебзастосунків. Його легкість у використанні, інтеграційні можливості, широкий вибір фреймворків, підтримка великих спільнот, висока продуктивність, безпека, підтримка ООП та масштабованість роблять PHP відмінним вибором для розробників, які прагнуть створювати ефективні та надійні вебсайти.

2.1.2. *JavaScript*

JavaScript є однією з найпопулярніших мов програмування у світі, і це не випадково. Вона володіє численними перевагами, що робить її незамінною для розробників застосунків. Розглянемо детальніше основні переваги JavaScript [7].

JavaScript спочатку створювався як мова для написання скриптів на стороні клієнта, але з часом він значно розширив свою сферу застосування. Сьогодні JavaScript використовується завдяки таким середовищам виконання, як Node.js. Це дозволяє розробникам використовувати одну мову для створення всього застосунка, що значно спрощує процес розробки та підтримки коду [8].

Також, JavaScript є відносно легкою для вивчення мовою, особливо для тих, хто вже має базові знання в програмуванні. Синтаксис JavaScript є досить простим і інтуїтивно зрозумілим, що дозволяє швидко опанувати основи мови та почати створювати прості скрипти. Крім того, існує безліч

ресурсів для навчання, таких як онлайн-курси, книги, форуми та документація, що допомагають новачкам швидко стати на ноги.

Підтримується всіма сучасними веббраузерами, такими як Google Chrome, Mozilla Firefox, Safari, Edge та інші. Це означає, що скрипти на JavaScript можуть виконуватися без необхідності встановлення додаткового програмного забезпечення або плагінів, що забезпечує високу сумісність та доступність для кінцевих користувачів.

Завдяки таким сучасним рушіям JavaScript, як V8 від Google, продуктивність JavaScript значно покращилася. Ці рушії оптимізують виконання коду, забезпечуючи високу швидкість роботи застосунків. Крім того, завдяки Node.js, JavaScript дозволяє створювати високопродуктивні серверні застосунки, які здатні обробляти великі обсяги запитів у реальному часі.

Однією з головних особливостей JavaScript є його асинхронна природа. Використання функцій callback, promises та async/await дозволяє ефективно оброблювати події та виконувати завдання без блокування основного потоку. Це особливо важливо для вебзастосунків, які повинні швидко реагувати на дії користувача та обробляти дані у фоновому режимі [9].

JavaScript має одну з найбільших та найактивніших спільнот розробників. Це означає, що завжди можна знайти допомогу та підтримку у вирішенні проблем, а також безліч готових рішень для різних завдань. На таких платформах, як GitHub, доступні мільйони відкритих проєктів та бібліотек, які можна використовувати у своїх застосунках.

Середовище JavaScript багате на різноманітні фреймворки та бібліотеки, що значно полегшують розробку застосунків. Наприклад, React, Angular і Vue.js є популярними фреймворками для розробки, які спрощують створення складних користувацьких інтерфейсів. Використовують Node.js разом з фреймворками, такими як Express.js, дозволяє швидко створювати масштабовані серверні застосунки.

Сучасний JavaScript підтримує модульність через стандарти ES6 та інші. Використання модулів дозволяє розробникам розділяти код на окремі файли та компоненти, що значно полегшує підтримку та масштабування великих проєктів. Модулі дозволяють легко повторно використовувати код та уникати дублювання, що сприяє підвищенню ефективності розробки [10].

2.2. Бази даних

2.2.1. *MySQL*

MySQL – це потужна система управління реляційними базами даних, яка широко використовується завдяки своїй надійності, високій продуктивності та відкритому коду. MySQL є вибором багатьох розробників і компаній для зберігання та обробки даних у вебзастосунках та корпоративних системах завдяки своїм численним перевагам і підтримці великої спільноти користувачів [11]. Розглянемо MySQL детальніше.

Однією з основних переваг MySQL є її висока продуктивність. MySQL оптимізована для роботи з великими обсягами даних та високими навантаженнями, що робить її відмінним вибором для вебзастосунків, які обробляють мільйони запитів на день. MySQL використовує ефективні алгоритми для роботи з пам’яттю та індексами, що дозволяє досягати високої швидкості обробки запитів. Масштабованість MySQL також заслуговує на увагу. Вона підтримує горизонтальне та вертикальне масштабування, що дозволяє збільшувати потужність системи за рахунок додавання нових серверів або покращення апаратного забезпечення. Це особливо важливо для великих проєктів, які потребують обробки величезних обсягів даних у реальному часі.

MySQL є відкритою СУБД, що означає, що її вихідний код доступний для всіх бажаючих. Це дозволяє розробникам вносити зміни та оптимізації під свої потреби. Відкритість коду також забезпечує високу прозорість і безпеку, оскільки будь-хто може перевірити код на наявність вразливостей. Вартість використання MySQL є ще однією важливою перевагою. Більшість

можливостей доступні безкоштовно, що робить її привабливим варіантом для стартапів та невеликих компаній, які мають обмежений бюджет. Водночас, існують платні версії MySQL з застосунковими можливостями та підтримкою від розробників, що може бути корисним для великих підприємств.

СУБД підтримує широкий спектр операційних систем, включаючи Windows, Linux, macOS, а також різні дистрибутиви Unix. Це дозволяє інтегрувати MySQL у будь-яке IT-середовище без значних зусиль. Крім того, MySQL підтримує більшість популярних мов програмування, таких як PHP, Java, Python, C++, Ruby і багато інших. Це робить її універсальним рішенням для різних типів застосунків, від вебсайтів до мобільних застосунків та великих корпоративних систем.

Також, MySQL відома своєю простотою використання. Інсталяція MySQL займає мінімум часу, а процес налаштування є досить інтуїтивно зрозумілим. Розробники можуть швидко налаштувати сервер MySQL і розпочати роботу з базами даних. MySQL також пропонує зручні графічні інтерфейси для управління базами даних, такі як MySQL Workbench. Ці інструменти дозволяють виконувати адміністративні завдання, розробляти та оптимізувати бази даних без необхідності використання командного рядка, що значно полегшує роботу, особливо для початківців [12].

Безпека є критично важливою складовою будь-якої бази даних, і MySQL пропонує потужні засоби захисту. Вона підтримує шифрування даних, аутентифікацію користувачів і контроль доступу, що дозволяє захищати дані від несанкціонованого доступу. Надійність MySQL також є важливим аспектом. Вона забезпечує механізми резервного відновлення та копіювання даних, а також підтримує реплікацію. Це дозволяє створювати резервні копії баз даних у реальному часі. Це забезпечує високу доступність даних і зменшує ризик втрати інформації у разі збою системи [13].

MySQL має велику та активну спільноту користувачів і розробників. Це означає, що завжди можна знайти допомогу або поради щодо вирішення

тих чи інших питань. Існує безліч форумів, блогів, документацій і посібників, які допомагають розв’язувати проблеми та оптимізувати роботу з MySQL. Офіційна підтримка від компанії Oracle, яка володіє MySQL, також є значним плюсом. Компанія надає професійну підтримку, регулярні оновлення та патчі безпеки, що гарантує актуальність та безпеку використання MySQL [14].

Зазначимо, що MySQL добре інтегрується з іншими популярними технологіями та платформами. Наприклад, вона відмінно працює з вебсерверами Apache та Nginx, а також з різними системами керування контентом, такими як WordPress, Joomla і Drupal. Це дозволяє створювати потужні та ефективні вебзастосунки з мінімальними зусиллями. Сумісність MySQL з різними системами реплікації та кластеризації дозволяє забезпечувати високу доступність і відмовостійкість застосунків. Це особливо важливо для критично важливих систем, які потребують постійного доступу до даних.

MySQL підтримує сучасні стандарти SQL, що робить її сумісною з іншими популярними СУБД. Це дозволяє легко переносити дані та програми між різними системами, знижуючи витрати на міграцію та інтеграцію. MySQL також пропонує розширені можливості SQL, включаючи підтримку складних запитів, транзакцій, індексів та тригерів, що дозволяє розробляти високоефективні та функціональні застосунки.

2.3. Фреймворки

2.3.1. *Laravel*

Laravel – це популярний PHP-фреймворк, створений для полегшення розробки вебзастосунків. З моменту свого випуску у 2011 році Laravel швидко завоював прихильність розробників завдяки своїм багатим можливостям, зручності використання та масштабованості. Нижче розглянемо основні переваги Laravel, які роблять його таким привабливим для веброзробників [15].

Laravel пропонує елегантний синтаксис, який спрощує написання коду. Він використовує модульний підхід до архітектури, що дозволяє розробникам використовувати та розширювати функціональність фреймворку за допомогою пакетів та модулів. Це значно спрощує процес розробки та підтримки великих проєктів.

Завдяки системі пакетів, розробники можуть легко додавати нову функціональність до своїх проєктів. Laravel має власний менеджер пакетів – Composer, який дозволяє швидко встановлювати та оновлювати залежності. Крім того, спільнота Laravel створила велику кількість готових пакетів, які можна використовувати для розширення функціональності застосунків.

Artisan – це командний інтерфейс Laravel, який надає розробникам безліч корисних команд для автоматизації різноманітних завдань. З його допомогою можна легко створювати міграції баз даних, генерувати код моделей, контролерів та інші компоненти проєкту. Це дозволяє значно прискорити процес розробки та зменшити кількість рутинної роботи.

Також, Laravel підтримує роботу з різними системами управління базами даних, такими як MySQL, SQL Server, SQLite та PostgreSQL. Він надає зручний ORM – Eloquent, який дозволяє легко працювати з базами даних за допомогою об’єктно-орієнтованого підходу. Це спрощує роботу з базами даних та підвищує продуктивність розробки [16].

Laravel надає гнучку систему маршрутизації. Це дозволяє легко налаштовувати шляхи для обробки запитів. За допомогою Middleware можна додавати різноманітні шари обробки запитів, такі як аутентифікація, кешування, логування та інші. Це дозволяє будувати більш захищені та оптимізовані вебзастосунки.

Фреймворк має вбудовані засоби для забезпечення безпеки вебзастосунків. Він пропонує функціональність для захисту від поширених вразливостей, таких як SQL-ін’єкції, XSS та CSRF. Завдяки цим

інструментам, розробники можуть бути впевнені, що їхні застосунки будуть захищеними від основних типів атак.

Laravel має вбудовану підтримку для тестування коду. Він постачається з PHPUnit, що дозволяє легко писати та виконувати модульні тести. Крім того, Laravel надає зручні інструменти для написання функціональних тестів, що дозволяє забезпечити високу якість перевірки коду та зменшити кількість випадків виникнення помилок.

Сучасні вебзастосунки часто взаємодіють з іншими системами за допомогою API. Laravel має можливість створення RESTful API, що дозволяє легко реалізовувати інтерфейси для взаємодії з іншими сервісами. Це робить Laravel одним з найкращих продуктів для розробки застосунків, які потребують інтеграції з іншими системами.

2.3.2. *Sanctum*

Laravel Sanctum – це легкий пакет автентифікації для фреймворку Laravel, що надає прості рішення для управління аутентифікацією в застосунках, зокрема SPA, мобільних застосунках та простих API. Sanctum має багато переваг, що роблять його чудовим вибором для розробників, які шукають ефективний та безпечний спосіб автентифікації користувачів [17]. Розглянемо переваги фреймворку.

Sanctum підтримує різні типи застосунків, включаючи SPA, мобільні застосунки та прості API. Це досягається за рахунок використання токенів, які можуть бути легко інтегровані у застосунки. Наприклад, для SPA застосунків можна використовувати Sanctum разом із CSRF токенами, що забезпечує додатковий рівень безпеки, а для мобільних застосунків – доступні API токени, які можна використовувати для автентифікації користувачів.

Безпека є одним з ключових аспектів будь-якої системи автентифікації, і Sanctum не є винятком. Він надає можливість створювати та керувати персональними токенами доступу, які можуть мати різні рівні

дозволів. Це дозволяє контролювати доступ до різних частин API застосунку. Крім того, Sanctum підтримує використання CSRF захисту для SPA застосунків, що допомагає запобігти атакам типу CSRF.

Також, Sanctum легко інтегрується з існуючими застосунками Laravel. Він добре працює з іншими пакетами та функціями Laravel, такими як Eloquent, маршрутами, середовищем та middleware. Це означає, що розробникам не потрібно робити значні зміни в їхньому коді для інтеграції Sanctum, що дозволяє економити час та зусилля.

Sanctum надає розробникам гнучкість у налаштуванні автентифікації відповідно до їхніх потреб. Наприклад, можна легко налаштувати політики доступу для різних типів користувачів або створювати спеціальні middleware для обробки автентифікації. Це дозволяє створювати складні системи з багаторівневою аутентифікацією, що може бути корисним для великих проєктів з високими вимогами до безпеки та функціональності.

Фреймворк дозволяє створювати персональні токени доступу, які можна використовувати для автентифікації API запитів. Кожен токен може мати певні дозволи, що дозволяє контролювати, які дії можуть виконуватися за допомогою цього токена. Це особливо корисно для мобільних застосунків або інших клієнтів, які взаємодіють з вашим API.

Sanctum забезпечує легке управління сесіями користувачів, що дозволяє легко керувати активними сесіями, переглядати та відкликати токени. Це може бути корисним для відстеження активності користувачів та забезпечення додаткового рівня безпеки, особливо у випадках, коли є підозра на компрометацію облікового запису.

Однією з нових функцій, що були додані до Sanctum, є підтримка MFA. Це додає додатковий рівень безпеки, вимагаючи від користувачів підтвердити свою особу через додаткові методи автентифікації, такі як OTP або біометричні дані. Це стає все більш важливим у сучасному світі, де атаки на облікові записи стають все більш складними та частими.

Sanctum добре працює з іншими популярними бібліотеками та фреймворками, такими як Vue.js, React, Angular. Це дозволяє розробникам створювати комплексні рішення, використовуючи якісні інструменти для кожного конкретного завдання. Завдяки цьому можна легко інтегрувати Sanctum в існуючі проєкти, що вже використовують ці фреймворки, без значних змін у коді.

2.3.3. *Angular*

Angular – це потужний і популярний фреймворк для створення вебзастосунків, розроблений компанією Google. З моменту свого запуску у 2010 році, він пройшов значний шлях розвитку та трансформації, і на сьогоднішній день є одним з найпоширеніших інструментів у сфері розробки. Розглянемо основні переваги Angular, які роблять його таким привабливим для розробників [18].

Однією з найбільших переваг Angular є його модульна структура. Це дозволяє розробникам розділяти код на логічно ізольовані модулі, що полегшує його підтримку, тестування і повторне використання. Модулі можуть включати в себе компоненти, сервіси, директиви та інші елементи, що робить процес розробки більш організованим та зручним [19].

Angular використовує декларативний підхід до розробки інтерфейсу користувача. Це означає, що розробники можуть описувати вигляд і поведінку застосунка за допомогою HTML-шаблонів, що спрощує розуміння і підтримку коду. Angular розширює можливості HTML за допомогою власних директив, що дозволяють легко керувати відображенням елементів в залежності від стану застосунка.

Двостороння прив’язка даних є однією з ключових особливостей Angular. Вона забезпечує автоматичне оновлення моделі даних у відповідь на зміни в інтерфейсі користувача і навпаки. Це дозволяє розробникам значно скоротити кількість коду, необхідного для синхронізації стану

застосунка з його представленням, і робить процес розробки більш інтуїтивно зрозумілим.

Також, Angular пропонує потужний механізм для роботи із залежностями. Це дозволяє розробникам легко управляти сервісами, які використовуються в застосунку, забезпечуючи високий рівень гнучкості та тестованості. За допомогою DI, сервіси можуть бути легко замінені, що особливо корисно для модульного тестування та створення масштабованих застосунків.

Angular CLI є незамінним інструментом для створення і управління Angular застосунками. Він автоматизує багато рутинних завдань, таких як генерація компонентів, модулів, сервісів та інших елементів застосунка. CLI також спрощує процес збірки, тестування і розгортання застосунка, забезпечуючи єдиний стандарт для структурування проєктів і підвищуючи продуктивність розробників.

Фреймворк включає ряд оптимізацій для підвищення продуктивності застосунків. Серед них – tree shaking, який видаляє невикористовуваний код з фінального бандлу, і AOT компіляція, яка перетворює Angular-шаблони в високоефективний JavaScript-код під час збірки. Це дозволяє зменшити час завантаження і збільшити швидкодію застосунків.

2.4. Розгортання

2.4.1. *Google Cloud*

Google Cloud, один з провідних постачальників хмарних послуг у світі, пропонує широкий спектр інноваційних рішень для бізнесу, урядових організацій, стартапів та розробників. Ця платформа надає комплексні сервіси, що охоплюють обчислення, зберігання даних, аналітику, машинне навчання та інші сучасні технології. Розглянемо детальніше переваги Google Cloud, які роблять його привабливим вибором для багатьох організацій [20].

Google Cloud дозволяє компаніям масштабувати свої ресурси відповідно до потреб бізнесу. Високий рівень продуктивності досягається завдяки інфраструктурі Google, яка включає сучасні дата-центри та потужні сервери. Ця масштабованість особливо важлива для компаній, що швидко зростають або переживають пікові навантаження, наприклад, під час запуску нових продуктів чи рекламних кампаній. Завдяки Google Cloud організації можуть легко збільшувати або зменшувати обчислювальні потужності та обсяги зберігання без значних інвестицій у власну інфраструктуру [21].

Безпека даних є однією з головних переваг Google Cloud. Платформа забезпечує багаторівневу безпеку, включаючи шифрування даних як під час передачі, так і під час зберігання. Google також має одну з найсильніших систем автентифікації, що включає двофакторну аутентифікацію та інтеграцію з системами керування доступом. Крім того, Google постійно проводить аудит своєї інфраструктури і дотримується найвищих стандартів відповідності, таких як GDPR та HIPAA, що гарантує відповідність вимогам конфіденційності та захисту даних.

Однією з переваг Google Cloud є його здатність інтегруватися з іншими сервісами та платформами, включаючи інші хмарні провайдери. Це забезпечує гнучкість у виборі технологічних рішень та дозволяє використовувати мультихмарний підхід, коли організації можуть об’єднувати найкращі сервіси з різних платформ для оптимального виконання своїх завдань. Наприклад, компанії можуть використовувати Google Cloud для машинного навчання та AWS для зберігання даних, об’єднуючи переваги обох провайдерів.

2.5. Висновки

Аналіз різних технологій та мов програмування є ключовим етапом при створенні будь-якого вебзастосунку, оскільки це дозволяє вибрати найоптимальніші інструменти для реалізації проєкту. Для даного

вебсервісу, що займається організацією складських приміщень, був проведений ґрунтовний аналіз різних технологій та мов програмування.

Було досліджено різні мови програмування та фреймворки для розробки вебзастосунків, ретельно проаналізовано їхні переваги та відповідність вимогам проєкту.

На основі проведеного аналізу було вирішено використати мови програмування PHP та JavaScript, а також фреймворки Laravel, Sanctum та Angular для реалізації цього вебсервісу.PHP – це мова програмування, яка використовується для створення вебзастосунків, серверних скриптів та інтерактивних вебсайтів. PHP відзначається своєю простотою у вивченні, широкими можливостями для інтеграції з різними базами даних та високою продуктивністю при роботі з вебконтентом. Крім того, PHP є однією з найпопулярніших мов програмування для веброзробки, що забезпечує велику кількість інструментів та бібліотек. Тож PHP була обрана через її простоту, гнучкість, високу продуктивність та багатий набір функцій для розробки вебзастосунків.

JavaScript – це мова програмування, яка використовується для створення динамічних вебзастосунків, інтерактивних елементів на вебсторінках та інших вебпрограм. JavaScript має широку підтримку, високу гнучкість, ефективність у роботі з користувацькими інтерфейсами та велику спільноту розробників. Крім того, JavaScript є однією з найпопулярніших мов програмування на сьогоднішній день, що забезпечує доступ до великої кількості фреймворків та бібліотек. Тож JavaScript був обраний через його широку підтримку, гнучкість, ефективність та можливість створення інтерактивних вебзастосунків.

Laravel – це фреймворк для створення вебзастосунків на мові програмування PHP. Його вибір обумовлений простотою використання, зручною конфігурацією та легкою інтеграцією з різними інструментами. Laravel також дозволяє суттєво зменшити обсяг коду, що прискорює процес розробки.

Sanctum – це пакет для Laravel, який забезпечує аутентифікацію та авторизацію для вебзастосунків на мові програмування PHP. Sanctum був обраний через його простоту використання та можливість інтеграції з різними застосунками. Використання Sanctum дозволяє значно спростити процес розробки аутентифікаційної системи, що в свою чергу прискорює створення вебзастосунків.

Angular – це фреймворк для створення клієнтської частини вебзастосунків мовою програмування JavaScript. Його вибір обумовлений легкістю розробки та підтримки, а також можливістю створення багатокомпонентних і інтерактивних інтерфейсів.

3. РОЗРОБЛЕННЯ ВЕБСЕРВІСУ

3.1. Загальний опис програми

Були проаналізовані наявні застосунки, виявлені їхні переваги та недоліки, що дозволило сформулювати вимоги до нового застосунку. Вебсервіс розроблявся з урахуванням усіх вимог до інтерфейсу, безпеки та функціональних можливостей. Оскільки вебзастосунок призначений для використання великою кількістю робочого персоналу, у системі передбачено три ролі: «Працівник», «Управляючий», яких має зареєструвати адміністратор, та «Власник», який має права адміністратора.

На початку роботи з вебсервісом користувач потрапляє на головну сторінку, де повинен ввести дані для авторизації у системі завдяки логіну та паролю.

Після входу у систему, користувач потравляє на сторінку складських приміщень. На ньому користувачу показані доступні для нього складські приміщення та надається можливість обрати потрібне.

Користувач з роллю «Власник» може додати складське приміщення завдяки спеціальній кнопці.

При натисканні на потрібне складське приміщення користувач переходить до сторінки перегляду складу. На ній користувач має можливість переглянути інформацію про складське приміщення завдяки вкладці «Information», доступні сектори завдяки вкладці «Sectors», список поставок завдяки вкладці «Deliveries», список товарів завдяки вкладці «Goods», та може сгенерувати звіт завдяки вкладці «Reports».

При переході на вкладку «Sectors» користувач може побачити інформацію про сектор «Information», або про товари наявні в секторі «Goods List»

Користувач з роллю «Власник», може додати нові сектори завдяки спеціальній кнопці.

При переході на вкладку «Deliveries» користувач може переглянути та додати інформацію про поставки.

При переході на вкладку «Goods» користувач може переглянути та додати інформацію про товари.

Для користувача з роллю «Управляючий» та «Власник» також доступна вкладка «Tasks». При переході на сторінку постановки завдання користувач може перейти на дві вкладки «New Task» та «History». При переході до вкладки «New Task» користувач може у спеціальному вікні задати завдання для групи користувачів. Він може встановлювати строки роботи, описати, що саме потребується та відправити завдання користувачам. При переході до вкладки «History» користувач може побачити збережену історію поставлених завдань в певному складському приміщенні.

Коли користувач отримує завдання він отримує повідомлення до особистого кабінету. Там він отримує всю необхідну інформацію про завдання та має можливість підтвердити виконання через кнопку «Confirm», або сповістити про невиконання завдання через кнопку «Decline». Якщо завдання було надіслано групі користувачів, то завдання буде вважатися виконаним тільки після підтвердження кожного члена групи. У іншому випадку завдання буде вважатися невиконаним.

Користувач може викликати особистий профіль завдяки спеціальній кнопці. При натисканні здійснює перехід до сторінки користувача на якій доступна наступна інформація: Ім’я, роль та вкладка «Tasks», де відображаються активні завдання користувача. Також там присутня кнопка виходу з профіля, яка здійснює перехід користувача на екран логіну.

Користувач з роллю «Власник» в кабінеті замість вкладки «Tasks» знаходиться вкладка «New User». Там має бути задано логін, пароль, ім’я користувача та його роль.

3.2. Аналіз функціональних вимог до вебзастосунку

При розробленні вебсервісу кожній з функціональних вимог було надано пріоритет: значення «1» надане ключовим функціональним можливостям і є обов’язковим для виконання інакше сервіс не зможе працювати. Розподіл пріоритетів наведено у таблиці 3.1.

Таблиця 3.1

Функціональні вимоги до вебзастосунку

Код вимоги	Назва вимоги	Пріоритет вимоги
01	Реєстрація в системі	1
02	Авторизація в системі за допомогою логіну та паролю	1
03	Редагування даних користувача	1
04	Функція створення складів	1
05	Функція редагування складів	1
06	Функція створення секторів	1
07	Функція редагування секторів	1
08	Функція створення товарів	1
09	Функція редагування товарів	2
10	Функція створення завдань	1
11	Функція відстеження виконання завдань	1
12	Функція призначення відповідальних за завдання	1
13	Функція генерації звітів	2
14	Функція створення поставок	3

3.3. Архітектура системи

Для створення вебсервісу було прийнято рішення використовувати клієнт-серверну архітектуру, яка розділяє застосунок на серверну та клієнтську частини. Це дозволяє оптимально розподіляти навантаження між сервером, який надає послуги, і клієнтом, який взаємодіє з користувачем та відображає дані.

3.3.1. *Клієнт-серверна архітектура*

Клієнт-серверна архітектура є методом побудови програмних систем, який передбачає розподіл функціональності між двома основними частинами: клієнтською та серверною. Це розділення дозволяє оптимально використовувати ресурси і забезпечувати ефективну взаємодію між користувачами і сервером.

3.3.2. *Серверна частина*

Серверна частина виконує роль центрального вузла, який обробляє запити від клієнтів. Вона відповідає за виконання основних операцій, таких як обробка бізнес-логіки, зберігання та управління даними, а також забезпечення безпеки системи. Сервер приймає запити від клієнтів, виконує необхідні обчислення або операції і повертає результати назад клієнту.

3.3.3. *Клієнтська частина*

Клієнтська частина представляє собою інтерфейс, з яким взаємодіють користувачі. Вона відправляє запити до сервера і відображає отримані відповіді у вигляді зрозумілого інтерфейсу. Клієнтська частина забезпечує зручність користувача, дозволяючи йому вводити дані, переглядати інформацію і виконувати різні операції у системі.

3.3.4. *Взаємодія між клієнтом і сервером*

Клієнт і сервер взаємодіють між собою через мережу, використовуючи певні протоколи для обміну даними. Клієнт надсилає

запити до сервера, який їх обробляє і повертає відповіді. Така архітектура дозволяє централізовано зберігати і управляти даними, зменшуючи навантаження на клієнтські пристрої та підвищуючи загальну безпеку системи.

Клієнт-серверна архітектура забезпечує масштабованість, оскільки серверна частина може обслуговувати численні клієнти одночасно, а також спрощує підтримку і оновлення системи, оскільки основна логіка зосереджена на сервері.

Model-View-Controller (MVC) – це архітектурний шаблон програмного забезпечення, що поділяє застосунок на три взаємодіючі компоненти.

Модель управляє даними та бізнес-логікою. Вона містить усі необхідні для роботи застосунку, дані та логіку, що обробляє ці дані, забезпечуючи їх цілісність. Модель є незалежною від інших компонентів сервісу, що спрощує її розширення та модифікацію.

Подання (або вигляд) відповідає за відображення даних користувачу та їх інтерактивне управління. Цей компонент створює графічний інтерфейс користувача, дозволяючи взаємодіяти із застосунком. Подання включає різноманітні елементи, такі як кнопки, текстові поля, таблиці та інші, які відображають дані з Моделі.

Контролер обробляє запити користувача та керує взаємодією між Моделлю та Поданням. Він перетворює запити користувача на дії з даними, які необхідно виконати, і передає їх Моделі. Після обробки даних Модель повертає результати Контролеру, який, своєю чергою, передає їх Поданню для відображення.

3.4. Шаблон Model-View-Controller (MVC)

Model-View-Controller (MVC) – це архітектурний шаблон, що забезпечує розділення програми на три основні компоненти: Модель, Подання та Контролер. Кожен із цих компонентів має свої завдання, що

допомагає зберігати чітку структуру та організацію коду, роблячи його легшим для підтримки та розширення [22].

3.4.1. *Компоненти MVC*

MVC поділяється на три компоненти, а саме:

1. Модель.

Модель відповідає за управління даними та бізнес-логікою застосунку. Вона виконує такі основні функції, як зберігання даних, бізнес-логіка та оновлення стану. Функція зберігання даних утримує дані, які використовуються в застосунку, незалежно від їхнього джерела (база даних, зовнішні сервіси тощо). Функція бізнес-логіка реалізує правила та процедури обробки даних, включаючи валідацію, обчислення та транзакції. А функція оновлення стану інформує інші компоненти про зміни у даних.

Модель є автономною, що дозволяє легко змінювати або розширювати її, не впливаючи на інші частини застосунку.

2. Подання.

Подання відповідає за відображення інформації користувачу та забезпечення інтерактивної взаємодії. Основні функції полягають у відображенні даних та інтерактивності. Функція відображення даних презентує інформацію, отриману від моделі, у зручному для користувача вигляді. Функція інтерактивності дозволяє користувачу вводити дані та виконувати дії за допомогою елементів інтерфейсу (кнопки, поля введення, списки тощо).

Подання отримує дані від контролера та відображає їх, надаючи користувачу можливість взаємодіяти із системою.

3. Контролер.

Контролер діє як посередник між моделлю та поданням. Його основні функції полягають у обробці запитів, взаємодії з моделлю та оновлення подання. Функція обробки запитів приймає запити від користувача і визначає, які дії потрібно виконати. Функція взаємодії з моделлю

звертається до моделі для виконання необхідних операцій (читання, оновлення, створення, видалення даних). Функція оновлення даних після обробки даних оновлює подання, щоб відобразити актуальний стан інформації.

Контролер координує роботу між користувачем та моделлю, забезпечуючи їхню ефективну взаємодію.

3.4.2. *Взаємодія між компонентами*

Описати взаємодію між компоненти можна наступним ланцюгом:

1. Користувач взаємодіє з поданням через інтерфейс користувача.
2. Подання відправляє запити до контролера для обробки дій користувача.
3. Контролер обробляє запити, звертаючись до моделі для виконання необхідних операцій.
4. Модель оновлює свої дані відповідно до дій, виконаних контролером.
5. Контролер отримує оновлені дані від моделі і передає їх поданню.
6. Подання оновлює відображення даних для користувача.

3.4.3. *Реальний приклад*

Маємо вебзастосунок для управління складськими приміщеннями з функцією управління завданнями. У цьому випадку:

- Модель буде містити інформацію про завдання, включаючи назву, опис, статус та дедлайн. Модель також буде містити логіку для створення, редагування, видалення та отримання завдань.
- Подання буде відображати список завдань користувачу, дозволяючи переглядати деталі кожного завдання та вводити нові завдання через форми.
- Контролер буде обробляти дії користувача, такі як натискання кнопки «Додати завдання», взаємодіяти з моделлю для створення

нового завдання та оновлювати подання для відображення нового списку завдань.

Цей підхід забезпечує гнучкість, зручність підтримки та масштабованість застосунку, дозволяючи легко додавати нові функції або змінювати існуючі без впливу на інші частини системи.

Схему, на якій зображено взаємодію між модулями системи та користувачем, можна побачити на рис. 3.1.

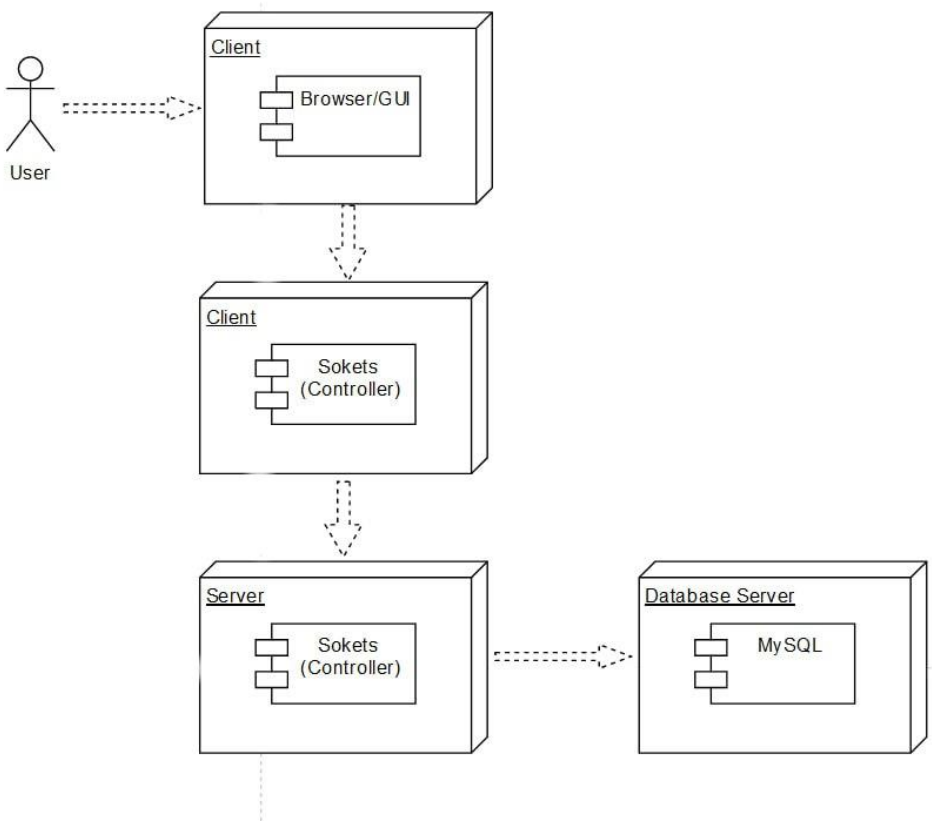


Рис. 3.1. Діаграма модулів системи

Для реалізації клієнт-серверної архітектури прикладу було розроблено 3 модулі:

- вебсервер;
- вебклієнт;
- модуль для роботи з базою даних.

Вебсервер

Вебсервер – це програмний комплекс, який приймає та обробляє запити від користувачів, що надходять через браузери або інші клієнтські програми. Він генерує відповіді у вигляді вебсторінок, зображень, відео та інших даних.

Вебклієнт

Вебклієнт – це програмне забезпечення, що використовується користувачами для доступу до серверів. Найпоширенішим видом вебклієнтів є браузери. браузери дозволяють користувачам вводити адреси вебсайтів, переглядати вебсторінки, завантажувати файли та взаємодіяти з вебсерверами іншими способами.

Взаємодія з СУБД

Взаємодія з системою управління базами даних полягає у створенні 8 моделей, таких як users, tasks, roles, sectors, warehouses, reports, deliveries, goods, які містять дані про основні та допоміжні сутності у вебсервісі.

Users

Дана модель зберігає дані про користувачів, а саме: дані про логін, шифрований пароль, повне ім’я, яке видно на сторінці користувача, та його роль (табл. 3.2).

Таблиця 3.2

Поля моделі Users

Назва поля	Тип поля
user_id	Integer AUTOINCREMENT
login	String
password	String
role_id	String FK
full_name	String

Tasks

Дана модель зберігає дані про завдання, що містить автора завдання, назву, опис, та дедлайн завдання, статус виконання завдання, сектор, склад та доставка, що стосуються цього завдання (табл. 3.3).

Таблиця 3.3

Поля моделі Tasks

Назва поля	Тип поля
task_id	Integer AUTOINCREMENT
author_id	Integer FK
title	String
description	String
delivery_id	Integer FK
warehouse_id	Integer FK
sector_id	Integer FK
deadline	Date
is_done	Boolean

Deliveries

Дана модель зберігає дані про доставки товарів, опис самої доставки (табл. 3.4).

Таблиця 3.4

Поля моделі Deliveries

Назва поля	Тип поля
delivery_id	Integer AUTOINCREMENT
description	String

Roles

Дана модель зберігає дані про ролі користувачів в системі (табл. 3.5).

Таблиця 3.5

Поля моделі Roles

Назва поля	Тип поля
role_id	Integer AUTOINCREMENT
name	String

Goods

Модель Goods зберігає дані про товар. Таблиця містить назву товару, його опис, ціну, та термін придатності (табл. 3.6).

Таблиця 3.6

Поля моделі Goods

Назва поля	Тип поля
good_id	Integer AUTOINCREMENT
name	String
description	String
price	Float
shelf_life	Date

Sectors

Модель Sectors зберігає дані про сектори скпладів. Сектор має назву та опис (табл. 3.7).

Таблиця 3.7

Поля моделі Sectors

Назва поля	Тип поля
sector_id	Integer AUTOINCREMENT
name	String
description	String

Warehouses

Дана модель зберігає дані про склади. Модель має номер складу, його адресу та опис (табл. 3.8).

Таблиця 3.8

Поля моделі Warehouses

Назва поля	Тип поля
warehouse_id	Integer AUTOINCREMENT
number	Integer
address	String
description	String

Reports

Модель Reports – допоміжна, зберігає дані про категорії, які використовуються у нотатках користувача (табл. 3.9).

Таблиця 3.9

Поля моделі Reports

Назва поля	Тип поля
report_id	Integer AUTOINCREMENT
start_date	Date

Продовження табл. 3.9

end_date	Date
full_sold_price	Float

Схему архітектури розробленої бази даних можна переглянути на рисунку 3.2.

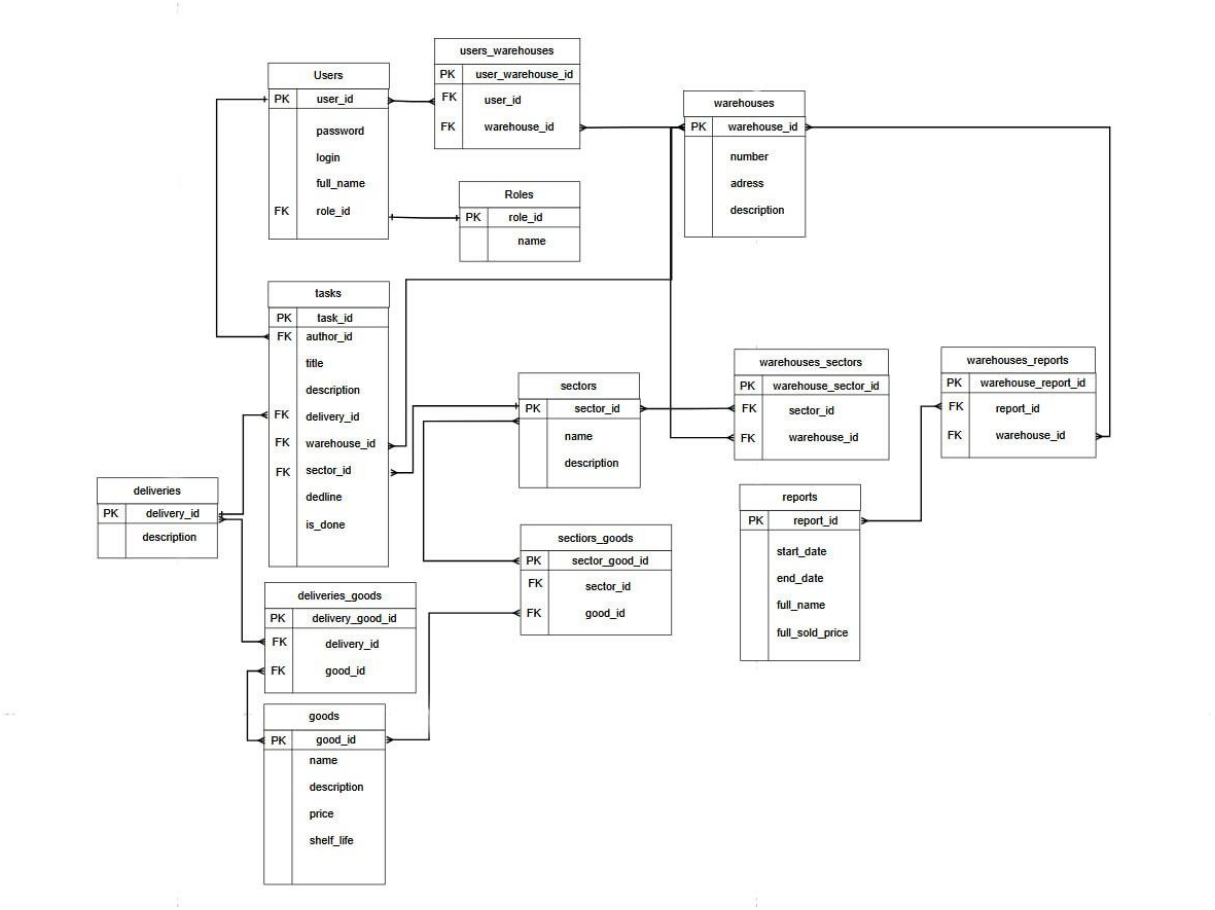


Рис. 3.2. Схема бази даних системи

3.5. Особливості реалізації

3.5.1. *Використання MVC*

Використання шаблону MVC дозволяє створювати чітко структуровані, масштабовані та легко підтримувані програми, що значно полегшує процес розробки та подальшої експлуатації.

Для розробки вебсервісу була вибрана клієнт-серверна архітектура, яка ділить застосунок на дві частини: серверну та клієнтську. Серверна частина відповідає за надання послуг і доступ до ресурсів, тоді як клієнтська

частина займається взаємодією з сервером і відображенням інформації користувачеві. Логіка роботи з базою даних була реалізована на серверній стороні.

Серверна частина обробляє запити від клієнта, виконує бізнес-логіку та взаємодіє з базою даних для зберігання та отримання необхідних даних. Вона надає API, яке клієнтська частина використовує для отримання даних та виконання операцій.

Клієнтська частина відповідає за відображення інформації для користувача та взаємодію з сервером через API. Вона забезпечує інтерфейс користувача, де користувач може переглядати, додавати, редагувати або видаляти дані. Клієнтська частина надсилає запити до серверної частини і отримує відповіді, які потім відображаються в зручному вигляді для користувача.

Цей підхід забезпечує чітке розділення обов'язків між серверною та клієнтською частинами, що сприяє кращій структурованості коду, полегшує розробку та підтримку застосунку, а також дозволяє легко масштабувати систему при зростанні навантаження. Діаграму класів можна побачити на рисунку 3.3.

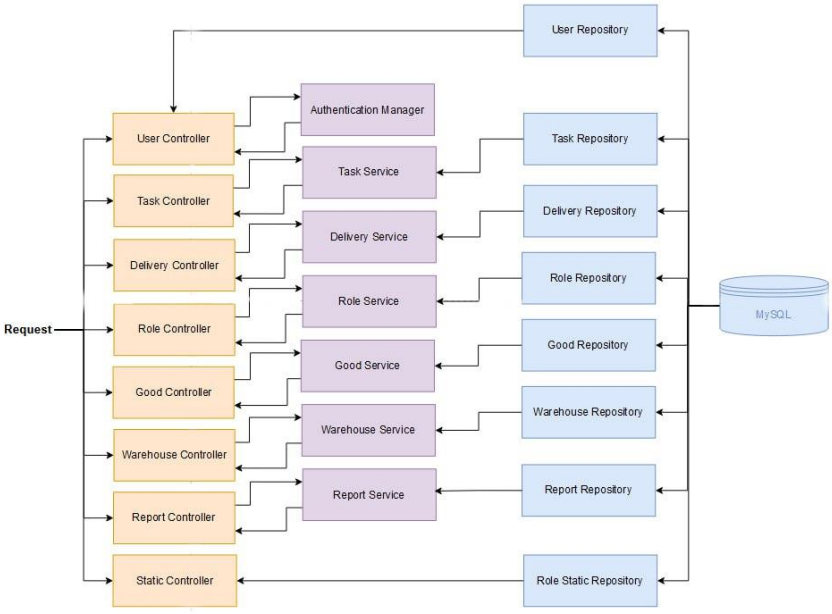


Рис. 3.3. Діаграма класів серверної частини

3.5.2. *Використання Laravel*

Laravel – це сучасний фреймворк для мови програмування PHP, створений для розробки вебзастосунків за допомогою елегантного та простого синтаксису. Його основна мета – зробити процес розробки зручнішим і швидшим, забезпечуючи розробникам широкий набір інструментів та можливостей. Laravel базується на архітектурі Model-View-Controller, що допомагає розділити логіку застосунку на три основні частини: модель, представлення та контролер.

Аутентифікація в Laravel – це комплексний процес, який забезпечує безпечний доступ користувачів до вебзастосунків. Laravel надає вбудовані інструменти для реалізації аутентифікації, такі як пакети Laravel Breeze і Laravel Fortify. Розглянемо детальний покроковий процес аутентифікації в Laravel.

На початку необхідно переконатися, що конфігурація підключення до бази даних у файлі .env налаштована правильно. Це дозволяє зберігати дані користувачів, які будуть використовуватися для автентифікації.

Для реалізації автентифікації у Laravel можна використовувати пакет Laravel Breeze. Він встановлюється за допомогою Composer і надає базові шаблони для реєстрації та входу користувачів.

На цьому етапі користувачі вводять свої дані, такі як ім’я, електронну адресу та пароль, через форму реєстрації. Ці дані перевіряються на коректність та унікальність, а потім зберігаються в базі даних. Після успішної реєстрації користувач може увійти до системи.

Під час входу користувач надає свою електронну адресу та пароль. Система перевіряє ці дані, порівнюючи їх із записами в базі даних. Якщо дані співпадають, користувач успішно аутентифікується.

Після успішного входу Laravel створює об’єкт аутентифікації, який містить інформацію про користувача, таку як його ідентифікатор, ім’я та ролі. Цей об’єкт використовується для подальшого процесу авторизації та доступу до ресурсів.

Авторизація визначає, які дії може виконувати користувач у системі на основі його ролей та дозволів. Laravel використовує політики та шлюзи для управління доступом. Політики дозволяють визначити, чи може користувач виконувати певні дії над певними ресурсами.

Laravel також надає можливості для управління сеансами користувачів, такі як обмеження кількості одночасно активних сеансів та управління часом життя сеансу. Це забезпечує додатковий рівень безпеки для користувачів.

Процес автентифікації Laravel складається з кількох етапів (рис. 3.4).

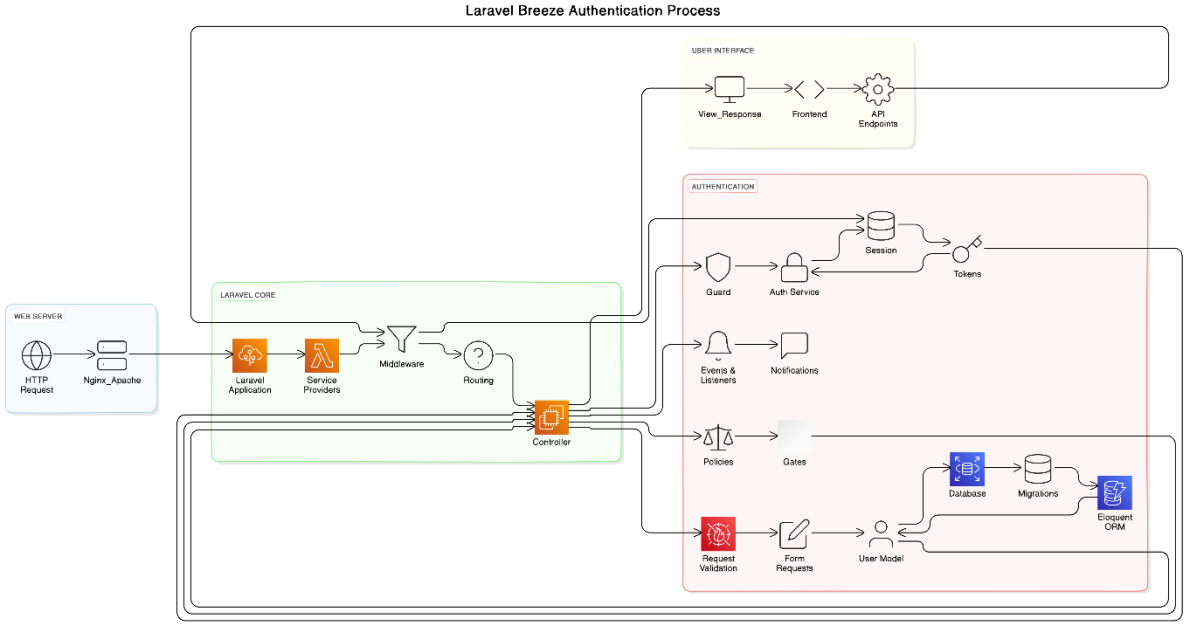


Рис. 3.4. Схема архітектури автентифікації Laravel

На клієнтській частині було розроблено модулі графічного відображення певних функцій системи.

Було розроблено 13 графічних інтерфейсів:

1. Сторінка логіну.
2. Головна сторінка.
3. Сторінка перегляду складу.
4. Сторінка перегляду списку секторів.
5. Сторінка перегляду сектору.

6. Сторінка постановки завдання.
7. Сторінка створення нового завдання.
8. Сторінка історії завдань.
9. Сторінка поставок.
10. Сторінка додавання нової поставки.
11. Сторінка генерації звітів.
12. Сторінка списку товарів.
13. Кабінет користувача.

3.6. Висновки

Вебсервіс розроблений з урахуванням високих вимог до інтерфейсу, безпеки та функціональних можливостей. Він включає ієрархічну структуру, яка забезпечує різні рівні доступу та можливості для користувачів.

Система побудована на клієнт-серверній архітектурі, використовуючи MVC (Model-View-Controller) шаблон, що є стандартом у розробці вебзастосунків.

Інтерфейс розроблений з акцентом на зручність і інтуїтивність, кожна сторінка забезпечує доступ до необхідних функцій.

Створений вебсервіс охоплює всі аспекти необхідні для забезпечення високого рівня функціональності, безпеки, та користувацької зручності. Ретельно продумана архітектура та гнучка система авторизації і реєстрації сприяють забезпеченню стабільності і надійності в роботі. Функціональні можливості, включаючи управління складами, завданнями, поставками та товарними позиціями, дають можливість користувачам ефективно керувати своїми обов’язками та відповідальностями. Інтерфейс, з орієнтацією на інтуїтивне управління, забезпечує легку взаємодію між користувачем і системою, а додаткові модулі статистики та звітності дозволяють моніторити успіхи і налаштовувати процеси бізнесу залежно від потреб.

Таким чином, цей вебсервіс є високоякісним інструментом, який дозволяє компаніям та індивідуальним користувачам ефективно управляти своїми ресурсами, оптимізувати процеси і підвищувати загальну продуктивність.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБСЕРВЕРУ

4.1. Аналіз реалізованого вебзастосунку

В результаті розробки програмного забезпечення було створено вебсервіс для контролю роботи складських приміщень. Цей вебзастосунок пропонує широкий спектр функцій для ефективного моніторингу виконання робіт та відображення наявних товарів на складі. Основна функція полягає у можливості створювати завдання для працівників складу та отримувати своєчасну та детальну інформацію про стан їх виконання.

Основна ідея застосунку надає можливість користувачу вводити інформацію про складські приміщення та ділити їх на сектори. Це дає змогу керуючому отримувати детальну інформацію про окремі частини складу та діяльність працівників.

Для зберігання даних використовується база даних, яка забезпечує надійне збереження і швидкий доступ до інформації.

Для забезпечення безпеки та конфіденційності даних вебсервіс використовує протокол HTTPS, що захищає передачу даних між клієнтом і сервером. Реалізовано механізми авторизації користувачів.

Вебсервіс був розгорнутий на платформі Google Cloud, яка надає зручні можливості для розгортання, керування та масштабування вебзастосунків.

Всі функціональні та нефункціональні вимоги до сервісу були реалізовані. Система завдань дозволяє відображати завдання та інформацію про них, задавати їх певній групі працівників та забезпечувати можливість звітування про виконання. Завдання можуть мати статуси виконаних або невиконаних, зберігається їх історія, що дозволяє аналізувати ефективність роботи працівників і складу в цілому.

Також, вебсервіс дозволяє створювати списки товарів і додавати нові товари з відповідними параметрами. Також реалізована функція створення

поставок, при якій товар автоматично додається до списку товарів у відповідному секторі після виконання завдання на прийняття товару.

Генерація звітів є ще однією важливою функцією вебсервісу. Звіти автоматично генеруються та містять інформацію про надходження і вилучення товарів за останній місяць, що дозволяє ефективно моніторити роботу складських приміщень.

Інтерфейс вебсервісу відзначається практичним дизайном, який був ретельно протестований. Він орієнтований на зручність та легкість у використанні, забезпечуючи зручну навігацію, інтуїтивно зрозумілі елементи керування та чітку організацію інформації.

4.2. Тестування розробленого вебзастосунку

Тестування є невід'ємною частиною процесу розробки. Під час тестування виконуються різномантні дії, спрямовані на виявлення та усунення помилок, а також перевірку роботи системи.

Під час розробки цього програмного забезпечення тестування проводилося одразу після реалізації функції. Кожен модуль тестувався після повної реалізації, що дозволило виявляти та усувати можливі помилки на усіх етапах розробки. Після розробки модулю програми, він також одразу тестувався. Такий метод допоміг знизити кількість помилок та підвищити якість розроблюваного продукту.

Під час проведення тестування програмного забезпечення було обрано метод ручного тестування. Його суть полягає в тому, що тестувальник виконує роль користувача та перевіряє функціональні можливості програми та її поведінку, використовуючи різні ролі. Цей підхід дозволяє виявити можливі проблеми системи, як це міг би зробити користувач.

Для проведення тестування були залучені потенційні користувачі, і їхні відгуки враховувалися в процесі доопрацювання.

Основні зауваження до розроблюваного програмного забезпечення включали:

1. Додати список працівників складу.
2. Додати кнопку профілю на усі сторінки вебзастосунку.
3. Додати систему перевірки підтвердження виконання завдання усією групою користувачів.

В рамках робіт також було виконано тестування програмного забезпечення на відповідність вимогам безпеки. Під час тестування було підтверджено, що всі вимоги безпеки були дотримані. База даних була надійно захищена від зовнішніх ін’єкцій, вебсервіс мав захист від JS-ін’єкцій, а програмне забезпечення забезпечувало валідацію даних, що вводяться користувачами.

Усі виявлені зауваження були детально розглянуті, після чого до програмного забезпечення було внесено необхідні корективи.

По завершенню фінального тестування було визначено, що у розробленому програмному забезпеченні не залишилося жодних помилок. Це вказує на успішне завершення тестування та готовність програмного забезпечення до подальшої експлуатації.

4.3. Порівняння з аналогами

У першому розділі проведено детальне дослідження різних рішень, які присутні на ринку. Було розглянуто їх особливості, переваги та недоліки. Дослідження показало, що існують декілька важливих функціональних можливостей, які варто врахувати:

1. Зручний поділ та перехід між складськими приміщеннями.
2. Додавання та створення користувачів в системі людиною на підприємстві, а не компанією, що надала програмне забезпечення.
3. Відмова від комп’ютерного застосунку у бік вебзастосунку.
4. Відстеження поставок, завдяки яким реалізовано автоматичне

додавання товару до сектору.

5. Можливість відстежувати ефективність роботи складських приміщень завдяки генерації звітів.
6. Відсутність способу контроля роботи певного працівника, або групи працівників.

Також були проаналізовані різні способи комунікації між підлеглими та керівниками. Проте ні в одному з досліджених застосунків не був застосований підхід постановки завдань – зручний альтернативний спосіб відстеження роботи працівників, та можливості збільшити ефективність контролю та комунікації. Використання постановки завдань дозволяє ставити завдання, які відносяться до певної поставки, долучати до нього певну групу працівників та асоціювати завдання з певним сектором.

Розроблений вебзастосунок має ключові функціональні можливості, що необхідні для програмного забезпечення для керування роботою складських приміщень. Було реалізовано можливість отримувати всю необхідну інформацію про складські приміщення, та редагувати її. Окрім цього, завдяки використанню функції постановки завдань у програмному забезпеченні, пропонується нове рішення для організації працівників та контролю за якістю праці. Отже, розроблене програмне забезпечення може конкурувати з аналогами, що представлені на ринку.

4.4. Рекомендації для подальшого вдосконалення

Розроблений вебзастосунок пропонує різноманітні інструменти, що відповідають потребам майбутніх користувачів та забезпечують їм зручність в користуванні. У зв’язку з швидким розвитком області використання цього програмного забезпечення та великою конкуренцією, регулярне оновлення вебсервісу стає важливим для забезпечення його актуальності та конкурентоздатності.

Завдяки зворотньому зв’язку від цільової аудиторії, були сформульовані рекомендації для вдосконалення сервісу:

1. Розробка системи відстеження поставок у реальному часі.
2. Розроблення окремого програмного застосунку для мобільних пристроїв, який буде синхронізований з вебсервісом. Це дозволить отримувати завдання та всю необхідну інформацію в будь-якому зручному місці для працівника. Також це допоможе своєчасно повідомити про статус виконання завдання.
3. Використання системи оповіщень про нові завдання. Це дозволить швидко реагувати на нові поставки та види робіт.
4. Деталізація процесу постановки завдань. Додавання нових статусів виконання завдань. Додавання шаблонів завдань та впровадження системи виконання та контролю декількох завдань в одному.
5. Створення зв’язку між різними складськими приміщеннями.
6. Додавання технічної підтримки.
7. Додавання системи створення власних шаблонів звітів та вибір формату отримуваного файлу звіту.

У майбутньому, цей вебсервіс може розвинутися до комплексного програмного рішення, що пропонуватиме широкі функціональні можливості та буде відповідати не лише потребам у керування складськими приміщеннями, а й у керуванні роботи відділів у великих компаніях. Це розширить сферу застосування даного програмного забезпечення, надаючи користувачам змогу ефективно співпрацювати та керувати роботою відділів та працівників.

4.5. Висновки

В цьому розділі детально проаналізовано розроблений вебсервіс, включаючи опис реалізації кожної функціональної та нефункціональної вимоги.

Було здійснено порівняння розробленого програмного забезпечення з аналогами. Порівняння показало, що вебсервіс не лише має ключові функціональні можливості, необхідні для контролю роботи складських приміщень, але й пропонує нове рішення для контролю ефективності праці певної групи працівників завдяки системі постановки завдань. Це надає йому конкурентну перевагу порівняно з іншими подібними рішеннями на ринку.

Завдяки системі ручного тестування було усунуто ряд помилок. Крім того, були проведені окремі тести з безпеки, що підтвердили виконання всіх вимог до безпеки. Отримані зауваження під час тестування були проаналізовані та виправлені.

На підставі аналізу сучасних тенденцій, вивчення особливостей аналогічного програмного забезпечення та опитування потенційних користувачів було складено перелік пропозицій та нововведень, які були важливі для подальшого розвитку та вдосконалення розробленого застосунку. Ці рекомендації мали на меті поліпшення користувацького досвіду та розширення функціональних можливостей, і всі вони були успішно реалізовані.

ВИСНОВКИ

Даний дипломний проєкт присвячений створенню програмного забезпечення для управління роботою складських приміщень.

У першому розділі проведено аналіз існуючих аналогів на ринку, з визначенням їх переваг та недоліків. На основі цього аналізу сформульовано перелік вимог до розробленої системи. Однією з головних переваг програмного забезпечення для управління складськими приміщеннями є постійний моніторинг та автоматизований збір даних. Проте існуючі застосунки мають обмежені функціональні можливості та недостатню гнучкість, що не задовольняє індивідуальні потреби користувачів.

У другому розділі проведено аналіз потенційних технологій розробки сервісу та обрано найбільш доцільні інструменти. Для реалізації сервісу було вибрано мови програмування PHP та JavaScript, фреймворки Laravel, Sanctum і Angular, а також базу даних MySQL.

Третій та четвертий розділи присвячені аналізу розробленого сервісу для планування та аналізу статистики, структури бази даних та візуалізації. Для забезпечення безпеки та обмеженого доступу до даних передбачена система авторизації. Порівняння розробленого програмного забезпечення з аналогами показало, що вебсервіс має ключові функціональні можливості для моніторингу стану виконання поставлених завдань, що надає йому конкурентну перевагу порівняно з іншими подібними рішеннями на ринку.

У результаті роботи над дипломним проєктом було розроблено програмне забезпечення для управління роботою складських приміщень, що враховує основні принципи цієї системи. Система надає користувачам можливість ефективно управляти складськими приміщеннями, планувати поставки, відстежувати прогрес виконання завдань та створювати звіти. Зручний інтерфейс вебсервісу дозволяє швидко створювати, редагувати та переглядати доступну інформацію.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

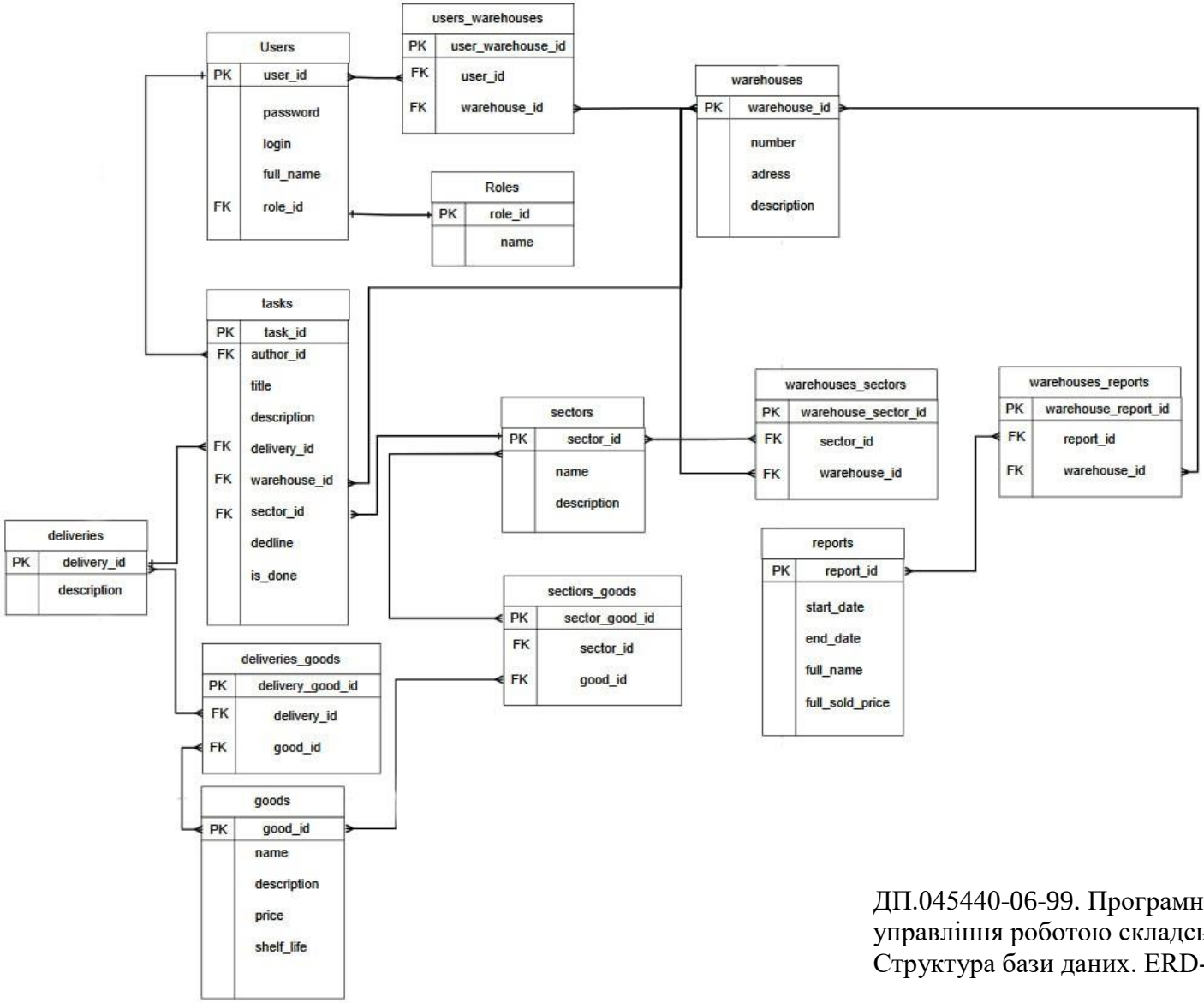
1. Zoho Inventory: офіційний вебсайт Zoho. [Електронний ресурс]. — Режим доступу: https://www.zoho.com/inventory/ — (27.04.2024)
2. NetSuite WMS: офіційний вебсайт NetSuite. [Електронний ресурс]. — Режим доступу: https://www.netsuite.com/portal/products/erp/warehouse-fulfillment/order-fulfillment.shtml/ — (27.04.2024)
3. Dilovod: офіційний вебсайт Dilovod. [Електронний ресурс]. — Режим доступу: https://dilovod.ua/uchet-sklada/ — (27.04.2024)
4. PHP: офіційний вебсайт PHP. [Електронний ресурс]. — Режим доступу: https://www.php.net/ — (27.04.2024)
5. PHP The Right Way: рекомендований підручник з PHP. [Електронний ресурс]. — Режим доступу: https://phptherightway.com/ — (27.04.2024)
6. PHP Security Guide: ресурс для вивчення безпеки в PHP. [Електронний ресурс]. — Режим доступу: https://phpsecurity.readthedocs.io/en/latest/index.html/ — (29.04.2024)
7. JavaScript: офіційний вебсайт JavaScript. [Електронний ресурс]. — Режим доступу: https://www.javascript.com/ — (29.04.2024)
8. Node.js: офіційний вебсайт Node.js. [Електронний ресурс]. — Режим доступу: https://nodejs.org/ — (29.04.2024)
9. JavaScript.info: документація по JavaScript. [Електронний ресурс]. — Режим доступу: https://javascript.info/ — (29.04.2024)
10. ECMAScript 6: офіційний вебсайт ECMAScript 2015. [Електронний ресурс]. — Режим доступу: https://262.ecma-international.org/6.0/ — (29.04.2024)
11. MySQL: офіційний вебсайт MySQL. [Електронний ресурс]. — Режим доступу: https://www.mysql.com/ — (01.05.2024)
12. MySQL Documentation: офіційна документація MySQL. [Електронний ресурс]. — Режим доступу: https://dev.mysql.com/doc/ — (01.05.2024)

13. MySQL Security Guide: керівництво з безпеки MySQL. [Електронний ресурс]. — Режим доступу: https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/ — (01.05.2024)
14. MySQL Community Forums: форум спільноти користувачів MySQL. [Електронний ресурс]. — Режим доступу: https://forums.mysql.com/ — (01.05.2024)
15. Laravel: офіційний вебсайт Laravel. [Електронний ресурс]. — Режим доступу: https://laravel.com/ — (01.05.2024)
16. Laravel Documentation: офіційна документація Laravel. [Електронний ресурс]. — Режим доступу: https://laravel.com/docs/ — (01.05.2024)
17. Laravel Sanctum: офіційна документація Laravel Sanctum. [Електронний ресурс]. — Режим доступу: https://laravel.com/docs/11.x/sanctum/ — (06.05.2024)
18. Angular: офіційний сайт. [Електронний ресурс]. — Режим доступу: https://angular.io/ — (06.05.2024)
19. Angular: документація. [Електронний ресурс]. — Режим доступу: https://angular.io/guide/architecture/ — (06.05.2024)
20. Google Cloud: офіційний вебсайт. [Електронний ресурс]. — Режим доступу: https://cloud.google.com/ — (06.05.2024)
21. Google Cloud Documentation: офіційна документація Google Cloud. [Електронний ресурс]. — Режим доступу: https://cloud.google.com/docs/ — (06.05.2024)
22. Microsoft ASP.NET MVC: офіційний вебсайт ASP.NET MVC. [Електронний ресурс]. — Режим доступу: https://dotnet.microsoft.com/apps/aspnet/mvc/ — (06.05.2024)

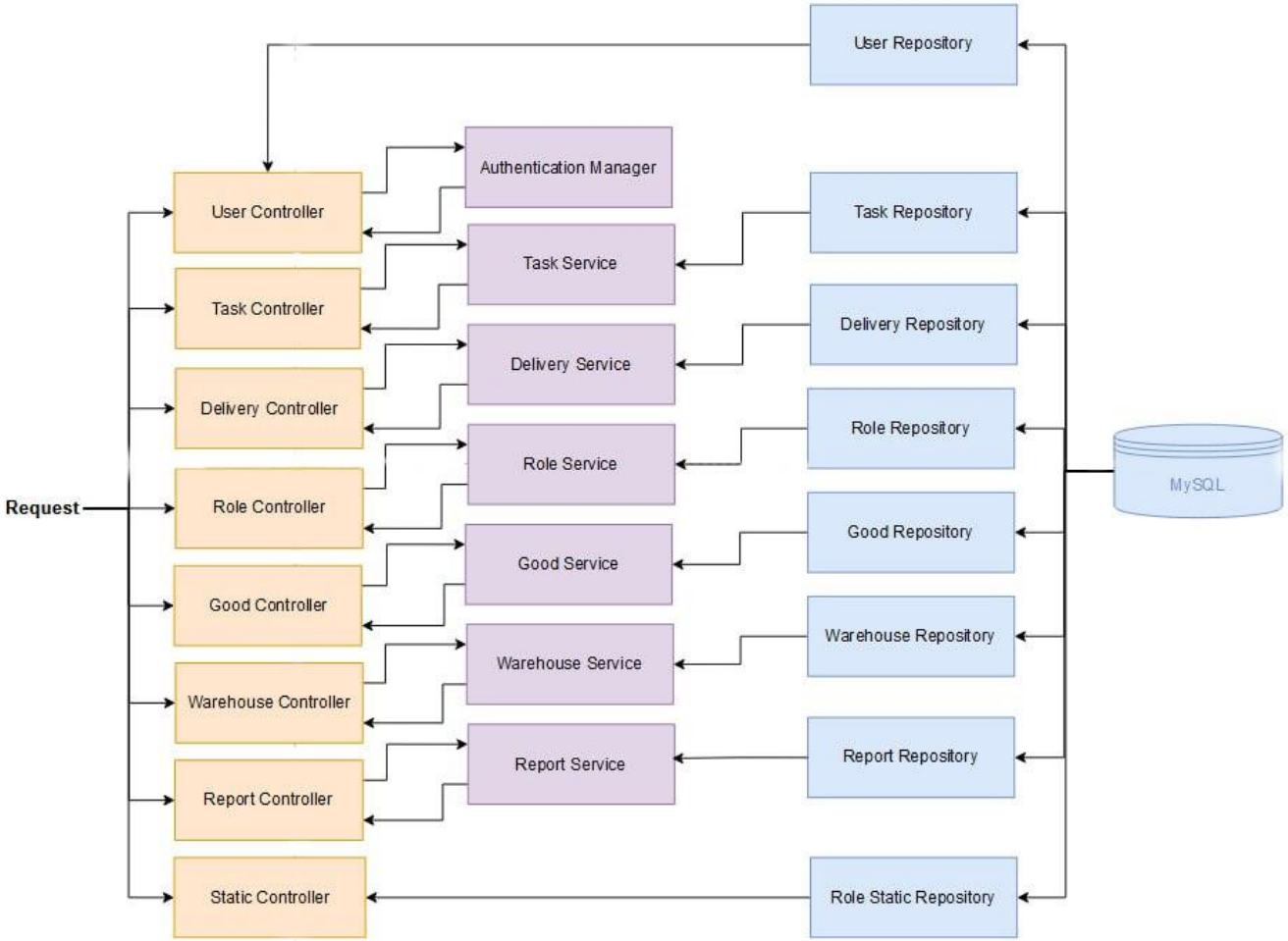
ДОДАТКИ

Додаток 1

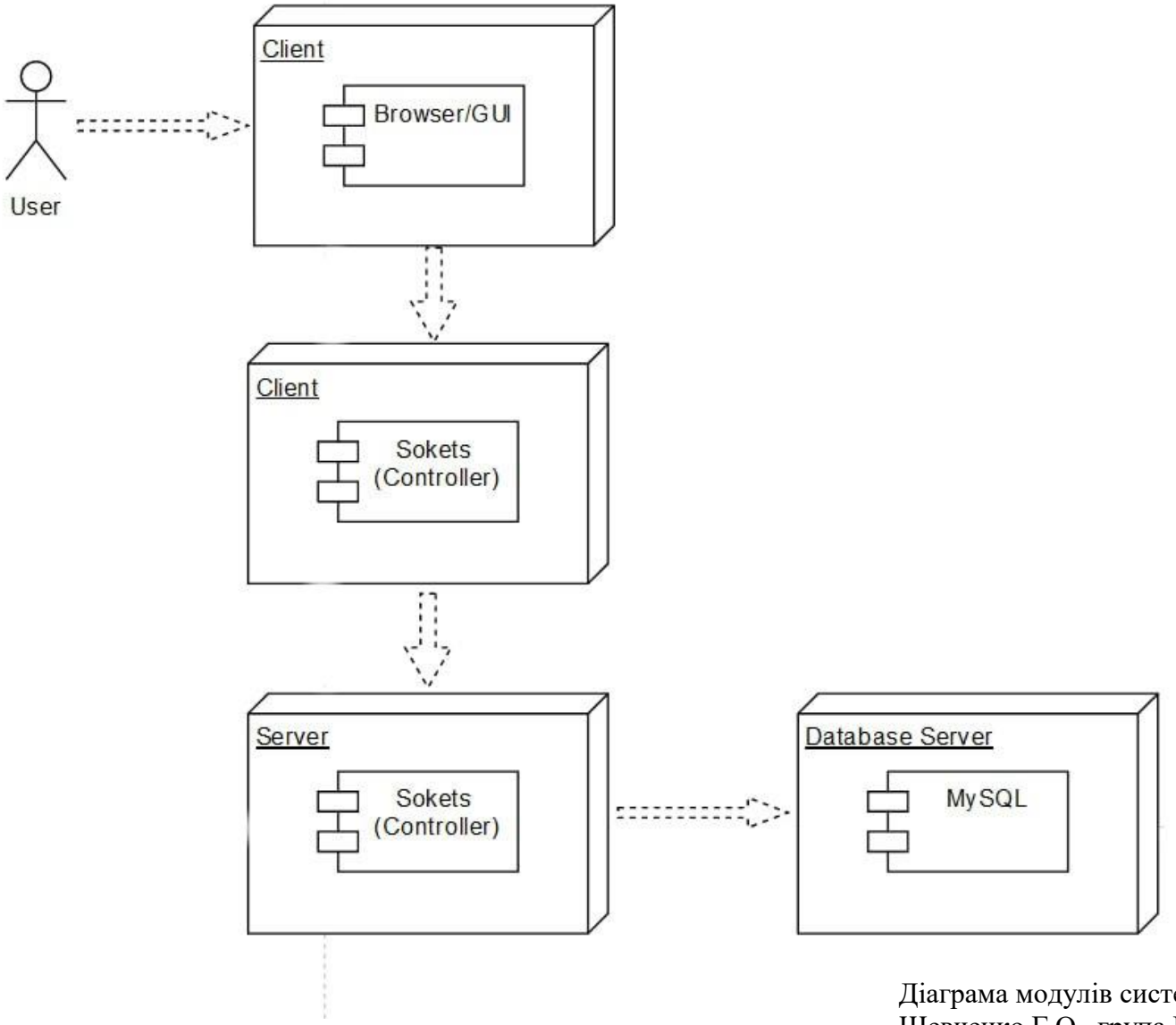
Копії графічних матеріалів



ДП.045440-06-99. Програмне забезпечення для управління роботою складських приміщень. Структура бази даних. ERD-діаграма



ДП.045440-07-99. Програмне забезпечення для управління роботою складських приміщень. Структура класів серверної частини. Діаграма Компонент



Діаграма модулів системи вебзастосунку
Шевченко Г.О., група КП-03

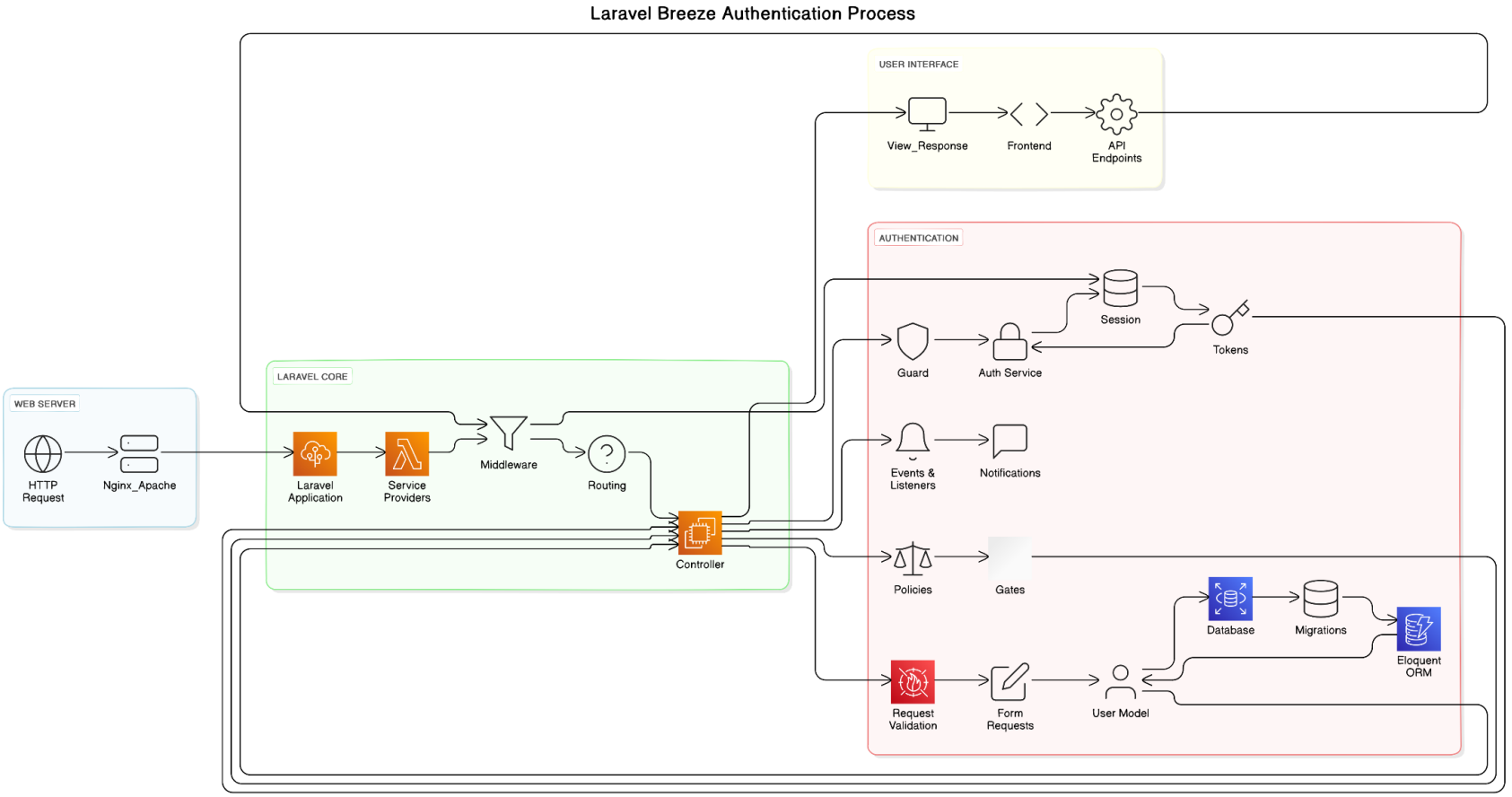


Схема архітектури автентифікації Laravel
Шевченко Г.О., група КП-03

Додаток 2

Лістинг програми

```
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
class User extends Authenticatable {
    use HasFactory, Notifiable;
    public function role() {
        return $this->belongsTo(Role::class);
    }
    public function tasks() {
        return $this->hasMany(Task::class, 'author_id');
    }
}
// app/Models/Role.php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Role extends Model {
    use HasFactory;
    public function users() {
        return $this->hasMany(User::class);
    }
}
// app/Models/Task.php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Task extends Model {
    use HasFactory;
    protected $fillable = ['author_id', 'title', 'description', 'delivery_id',
'warehouse_id', 'sector_id', 'deadline', 'is_done'];
    public function author() {
        return $this->belongsTo(User::class, 'author_id');
    }
    public function delivery() {
        return $this->belongsTo(Delivery::class);
    }

    public function warehouse() {
        return $this->belongsTo(Warehouse::class);
    }
    public function sector() {
        return $this->belongsTo(Sector::class);
    }
}
// app/Models/Warehouse.php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Warehouse extends Model {
    use HasFactory;
    protected $fillable = ['address', 'description'];
    public function tasks() {
        return $this->hasMany(Task::class);
    }
}
// app/Models/Sector.php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Sector extends Model {
    use HasFactory;
```

```
    protected $fillable = ['name', 'description'];
    public function tasks() {
        return $this->hasMany(Task::class);
    }
}
// app/Models/Delivery.php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Delivery extends Model {
    use HasFactory;
    protected $fillable = ['description'];
    public function tasks() {
        return $this->hasMany(Task::class);
    }
}
namespace App\Http\Requests;
use Illuminate\Foundation\Http\FormRequest;
class StoreTaskRequest extends FormRequest {
    public function authorize() {
        return $this->user()->role->name === 'admin';
    }
    public function rules() {
        return [
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'delivery_id' => 'nullable|exists:deliveries,id',
            'warehouse_id' => 'required|exists:warehouses,id',
            'sector_id' => 'required|exists:sectors,id',
            'deadline' => 'required|date',
            'is_done' => 'boolean',
        ];
    }
}
namespace App\Http\Controllers;
use App\Http\Requests\StoreTaskRequest;
use App\Models\Task;
class TaskController extends Controller {
    public function store(StoreTaskRequest $request) {
        $task = Task::create([
            'author_id' => $request->user()->id,
            'title' => $request->title,
            'description' => $request->description,
            'delivery_id' => $request->delivery_id,
            'warehouse_id' => $request->warehouse_id,
            'sector_id' => $request->sector_id,
            'deadline' => $request->deadline,
            'is_done' => $request->is_done ?? false,
        ]);
        return response()->json(['task' => $task], 201);
    }
}
use App\Http\Controllers\TaskController;
Route::middleware(['auth'])->group(function () {
    Route::post('/tasks', [TaskController::class, 'store']);
});
namespace App\Models;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
    protected $fillable = [
        'login',
        'password',
        'full_name',
        'role_id',
    ];
    protected $hidden = [
        'password',
        'remember_token',
    ];
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
php artisan make:migration create_roles_table
php artisan make:migration create_users_table
php artisan make:migration create_tasks_table
php artisan make:migration create_deliveries_table
php artisan make:migration create_goods_table
php artisan make:migration create_sectors_table
php artisan make:migration create_warehouses_table
php artisan make:migration create_reports_table
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateRolesTable extends Migration
{
    public function up()
    {
        Schema::create('roles', function (Blueprint $table) {
            $table->id('role_id');
            $table->string('name');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('roles');
    }
}
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id('user_id');
            $table->string('login')->unique();
            $table->string('password');
            $table->string('full_name');
            $table->unsignedBigInteger('role_id');
```

```
            $table->timestamps();
            $table->foreign('role_id')->references('role_id')->on('roles')-
>onDelete('cascade');
        });
    }
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
php artisan make:controller Auth/RegisterController
php artisan make:controller Auth/LoginController
namespace App\Http\Controllers\Auth;
use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
class RegisterController extends Controller
{
    public function showRegistrationForm()
    {
        return view('auth.register');
    }

    public function register(Request $request)
    {
        $request->validate([
            'login' => 'required|string|unique:users',
            'password' => 'required|string|confirmed',
            'full_name' => 'required|string',
            'role_id' => 'required|exists:roles,role_id',
        ]);
        $user = User::create([
            'login' => $request->login,
            'password' => Hash::make($request->password),
            'full_name' => $request->full_name,
            'role_id' => $request->role_id,
        ]);
        auth()->login($user);
        return redirect()->route('home');
    }
}
namespace App\Http\Controllers\Auth;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class LoginController extends Controller
{
    public function showLoginForm()
    {
        return view('auth.login');
    }

    public function login(Request $request)
    {
        $request->validate([
            'login' => 'required|string',
            'password' => 'required|string',
        ]);
```

```
        if (Auth::attempt($request->only('login', 'password'))) {
            return redirect()->route('home');
        }
        return back()->withErrors([
            'login' => 'The provided credentials do not match our records.',
        ]);
    }

    public function logout()
    {
        Auth::logout();
        return redirect()->route('home');
    }
}
use App\Http\Controllers\Auth\RegisterController;
use App\Http\Controllers\Auth\LoginController;
Route::get('register', [RegisterController::class, 'showRegistrationForm'])-
>name('register');
Route::post('register', [RegisterController::class, 'register']);
Route::get('login', [LoginController::class, 'showLoginForm'])->name('login');
Route::post('login', [LoginController::class, 'login']);
Route::post('logout', [LoginController::class, 'logout'])->name('logout');
php artisan make:controller WarehouseController
use App\Http\Controllers\WarehouseController;
Route::get('warehouses', [WarehouseController::class, 'index'])-
>name('warehouses.index');
Route::get('warehouses/create', [WarehouseController::class, 'create'])-
>name('warehouses.create');
Route::post('warehouses', [WarehouseController::class, 'store'])-
>name('warehouses.store');
Route::get('warehouses/{id}/edit', [WarehouseController::class, 'edit'])-
>name('warehouses.edit');
Route::put('warehouses/{id}', [WarehouseController::class, 'update'])-
>name('warehouses.update');
Route::delete('warehouses/{id}', [WarehouseController::class, 'destroy'])-
>name('warehouses.destroy');
namespace App\Http\Controllers;

use App\Models\Warehouse;
use Illuminate\Http\Request;
class WarehouseController extends Controller
{
    public function index()
    {
        $warehouses = Warehouse::all();
        return view('warehouses.index', compact('warehouses'));
    }

    public function create()
    {
        return view('warehouses.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'number' => 'required|integer',
            'address' => 'required|string',
            'description' => 'nullable|string',
```

```
        ]);
        Warehouse::create($request->all());
        return redirect()->route('warehouses.index')->with('success', 'Warehouse
created successfully.');
    }
    public function edit($id)
    {
        $warehouse = Warehouse::findOrFail($id);
        return view('warehouses.edit', compact('warehouse'));
    }
    public function update(Request $request, $id)
    {
        $request->validate([
            'number' => 'required|integer',
            'address' => 'required|string',
            'description' => 'nullable|string',
        ]);
        $warehouse = Warehouse::findOrFail($id);
        $warehouse->update($request->all());
        return redirect()->route('warehouses.index')->with('success', 'Warehouse
updated successfully.');
    }
    public function destroy($id)
    {
        $warehouse = Warehouse::findOrFail($id);
        $warehouse->delete();
        return redirect()->route('warehouses.index')->with('success', 'Warehouse
deleted successfully.');
    }
}
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Warehouse extends Model
{
    use HasFactory;
    protected $fillable = [
        'number',
        'address',
        'description',
    ];
    protected $primaryKey = 'warehouse_id';
}
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Програмне забезпечення для управління роботою складських приміщень

Виконав: студент групи КП-03 Шевченко Гліб Олегович

Керівник: асистент кафедри ПЗКС, д-р філософії, Юсин Яків Олексійович

Київ – 2024

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмне забезпечення для управління складськими приміщеннями.

Завдання:

1. Проаналізувати існуючи аналоги, та проблематику предметної області.
2. Обрати технологію для розроблення ПЗ, та розробити всі складові програмного забезпечення.
3. Протестувати та проаналізувати розроблене ПЗ.

АКТУАЛЬНІСТЬ

1. Зростання потреби в ефективному управлінні складськими приміщеннями.
2. Потреба в гнучкому рішенні, що задовольнить більшість вимог.
3. Недосконалість існуючих рішень:
    a) Відсутність системи контролю роботи працівників.
    b) Використання складних для розуміння та впровадження рішень.

ІСНУЮЧІ АНАЛОГИ

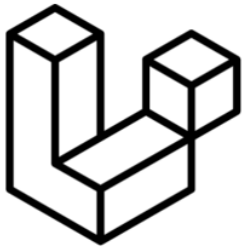
ZOHO Inventory

ORACLE NETSUITE WMS

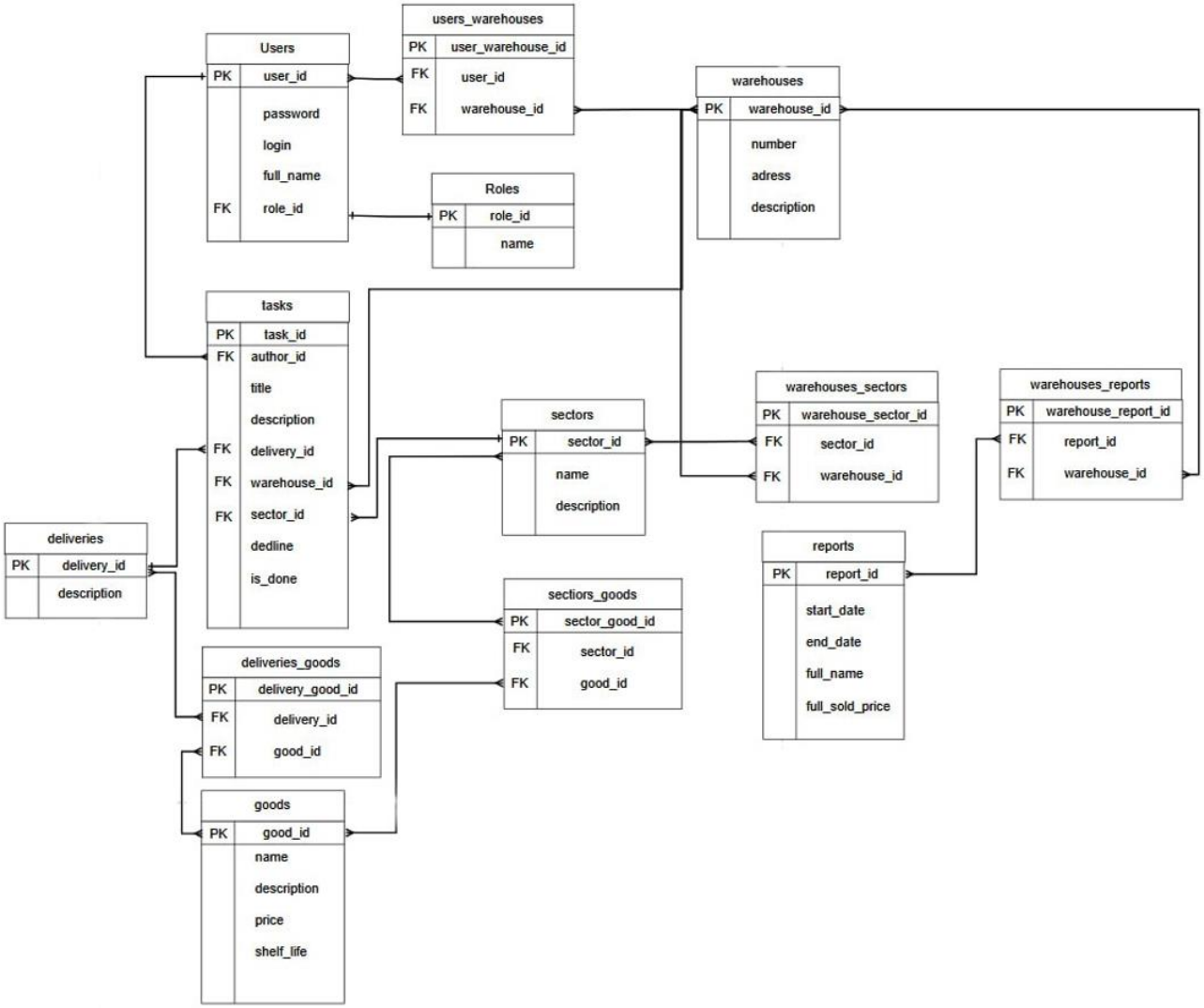
Dilovod

ЗАСОБИ РЕАЛІЗАЦІЇ

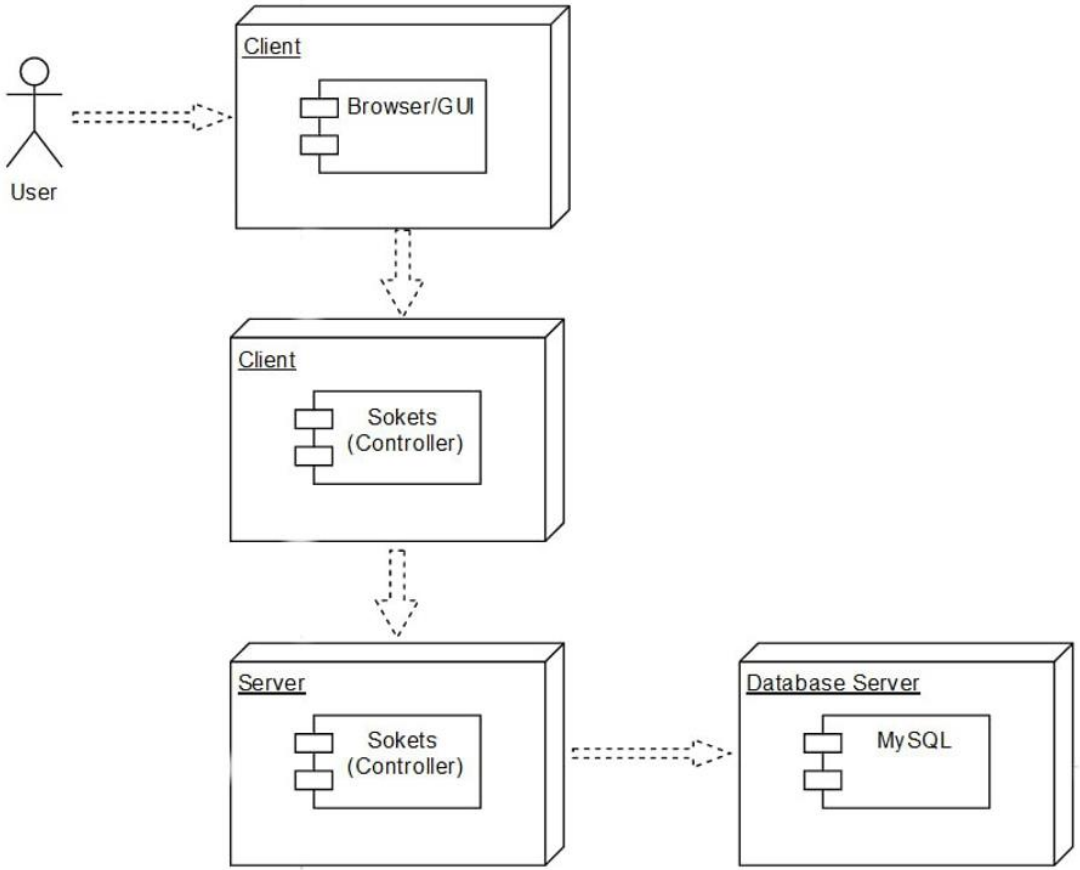
1. Мови програмування:
   a) PHP
   b) JavaScript
2. Бази даних:
   a) MySQL
3. Фреймворки:
   a) Laravel
   b) Sanctum
   c) Angular
4. Розгортання:
   a) Google Cloud



СТРУКТУРА БАЗИ ДАНИХ



ДІАГРАМА МОДУЛІВ СИСТЕМИ (MVC)



ДІАГРАМА КЛАСІВ

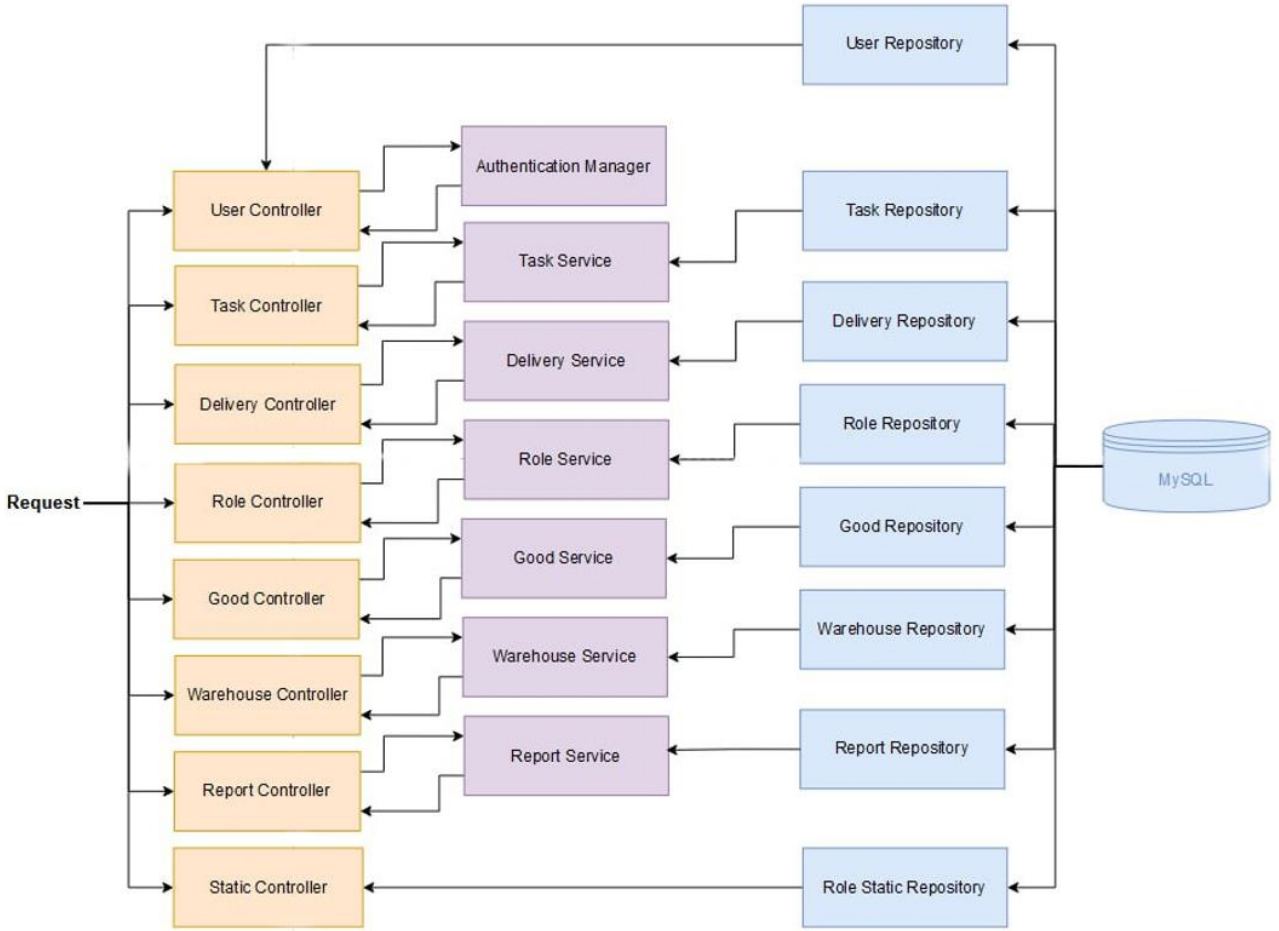
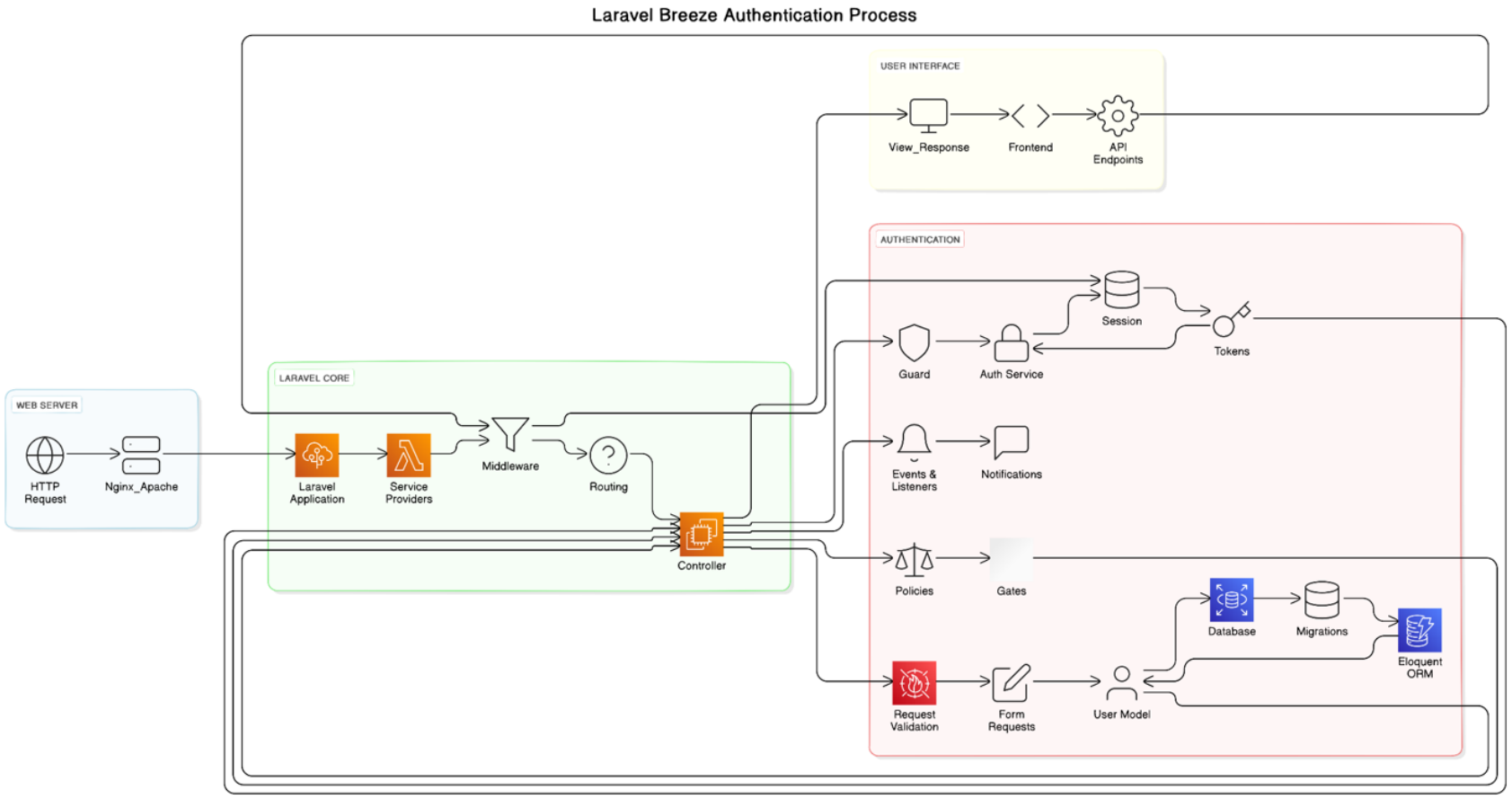
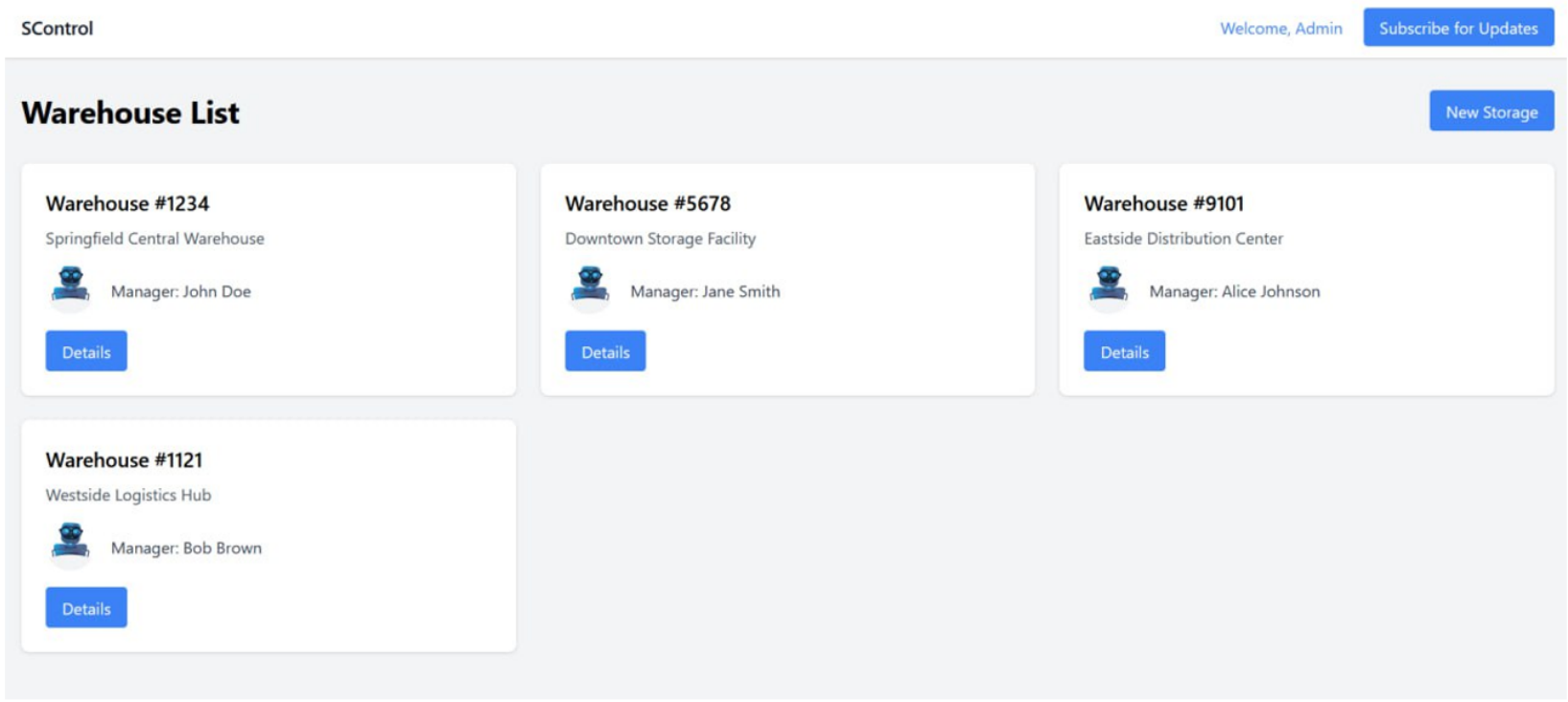


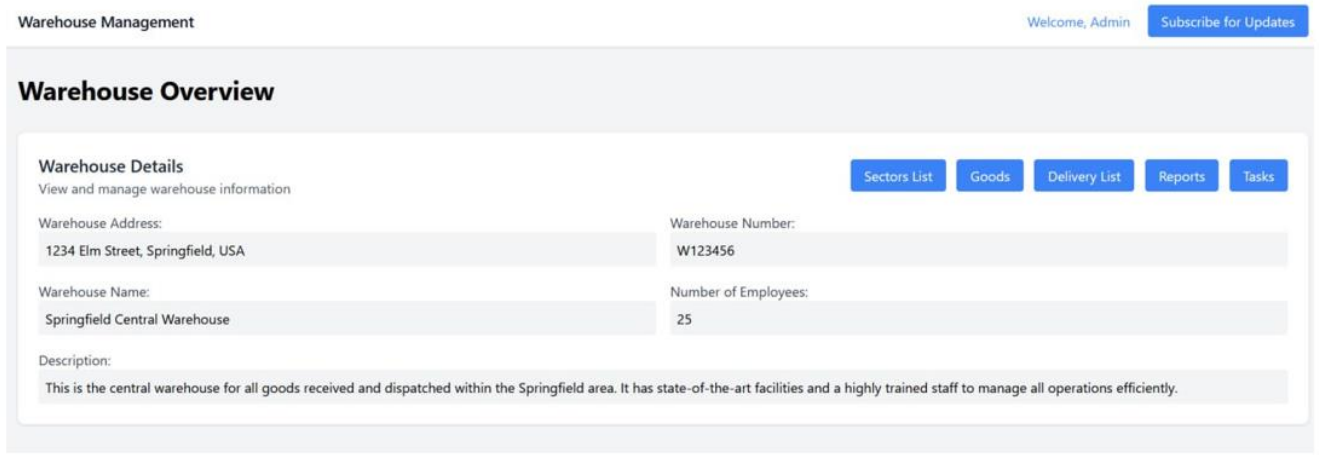
Схема архітектури автентифікації Laravel

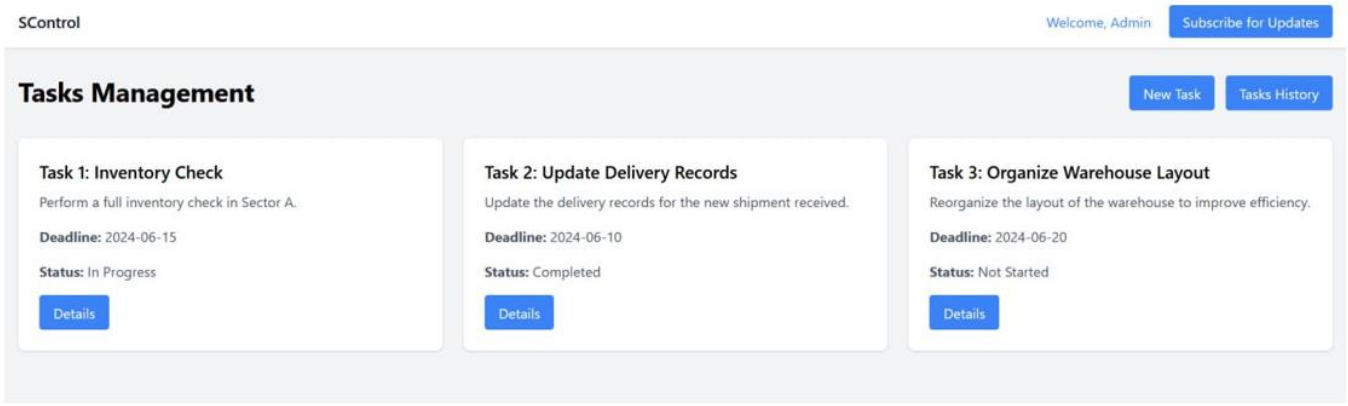


ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

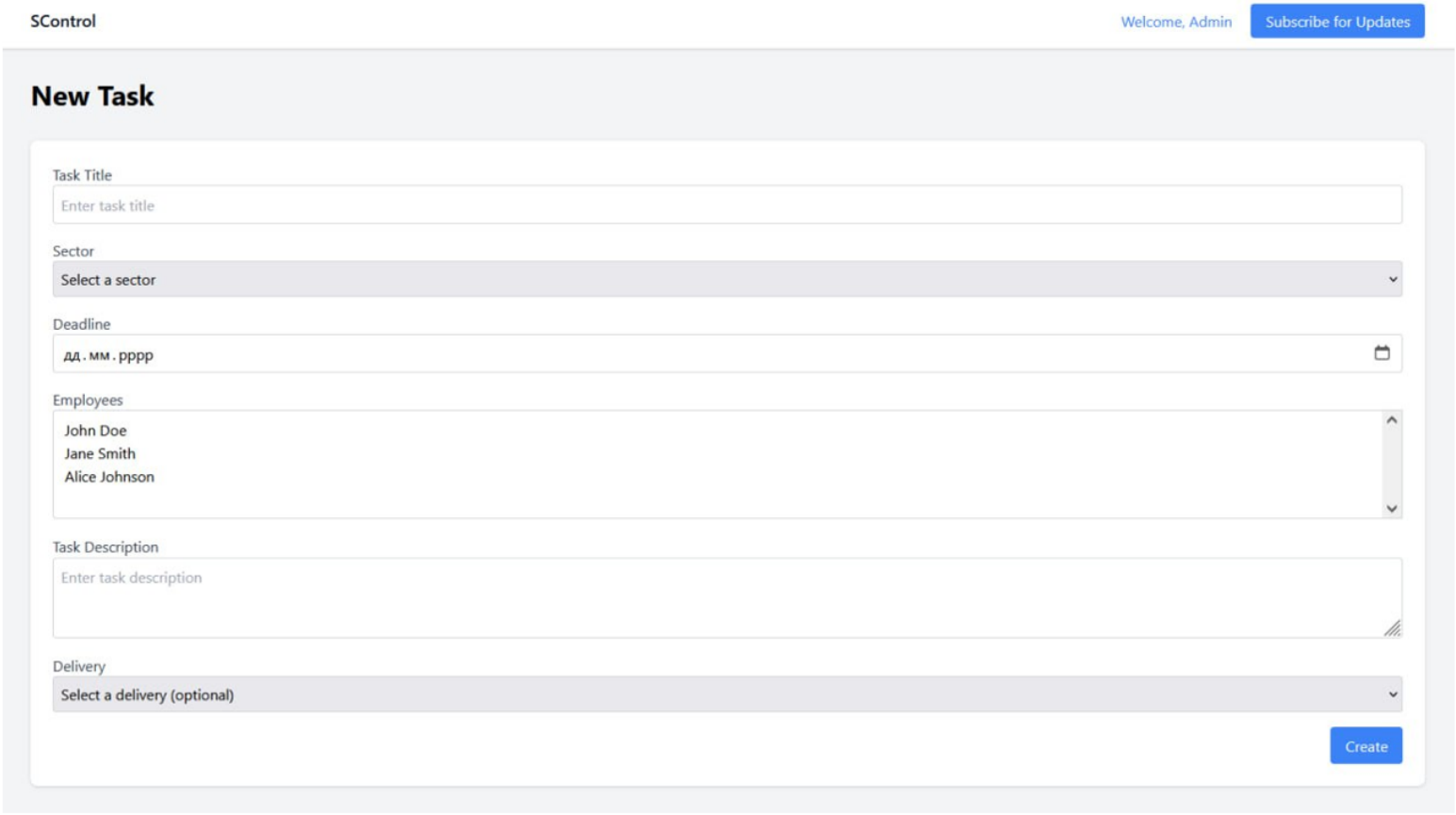


ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

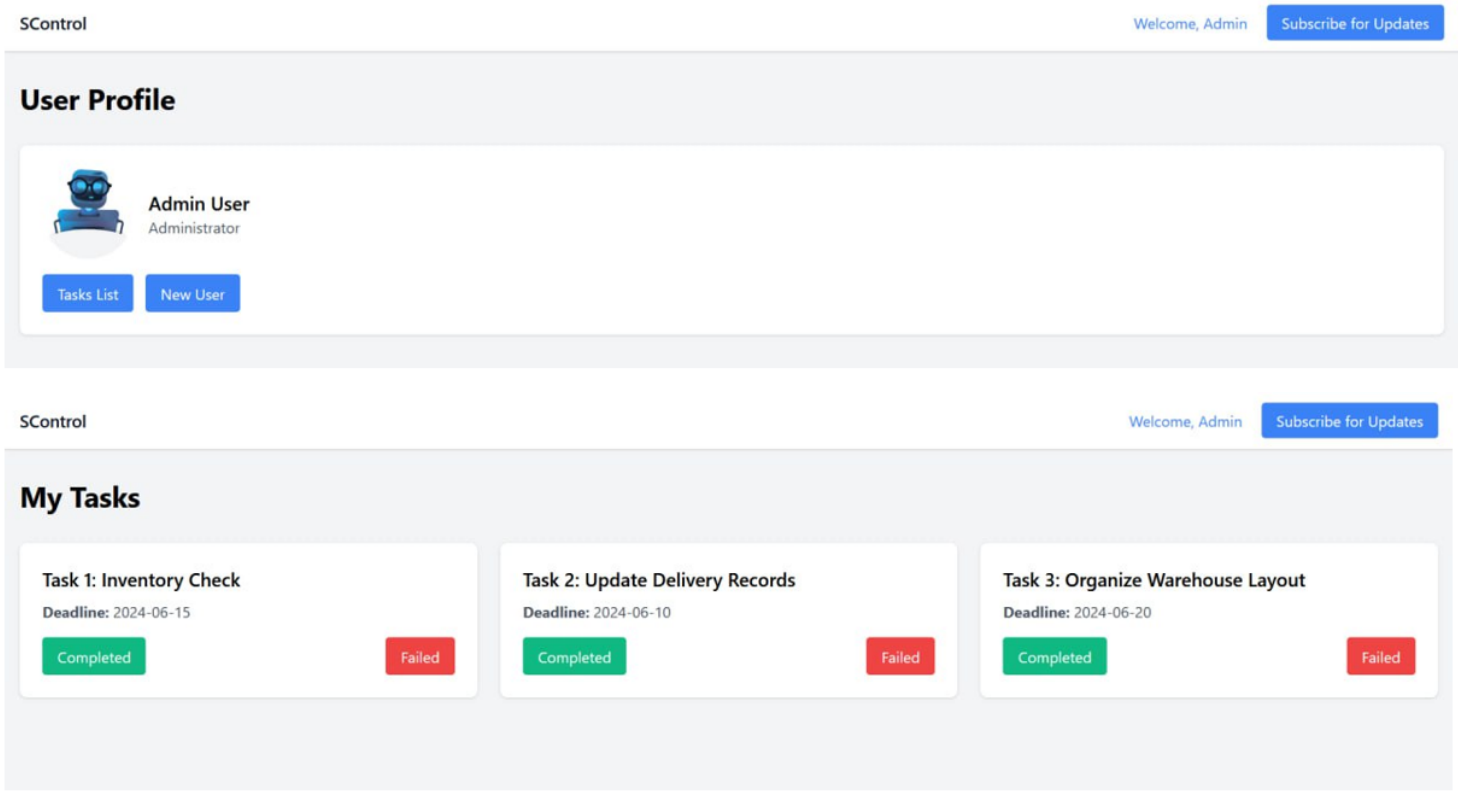




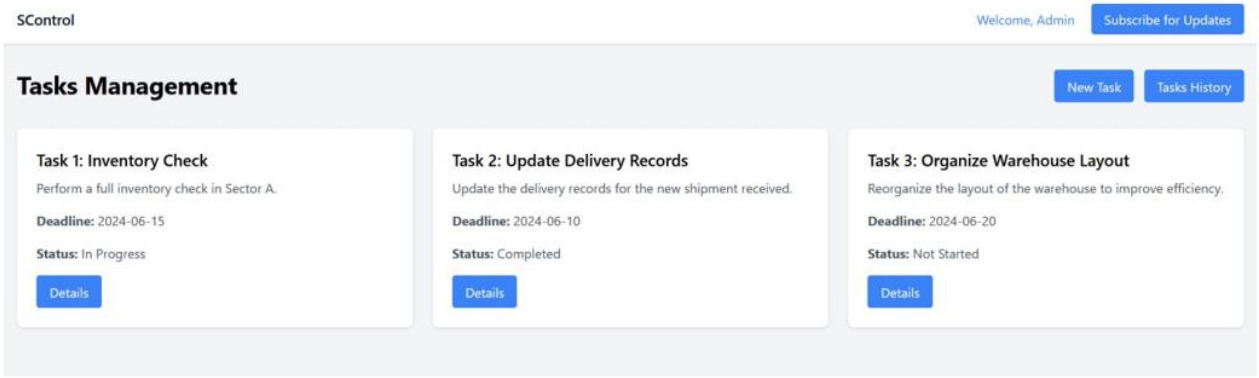
ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

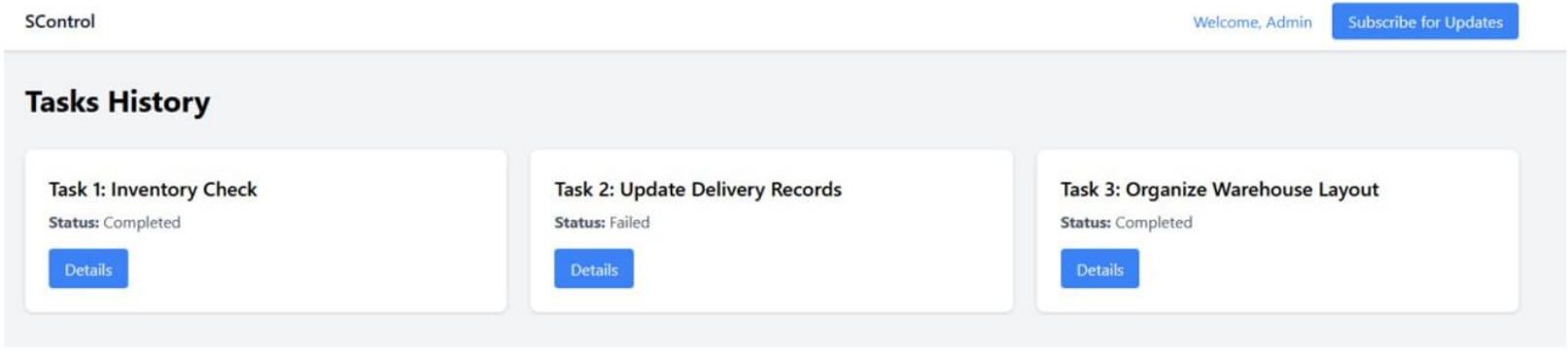


ПРИКЛАД РОБОТИ ЗАСТОСУНКУ



ПРИКЛАД РОБОТИ ЗАСТОСУНКУ





ШЛЯХИ РОЗВИТКУ

1. Створення мобільного застосунку 
2. Розробка системи відстеження поставок у реальному часі 
3. Створення зв’язку між різними складськими приміщеннями 
4. Додавання нових шаблонів завдань 
5. Вровадження системи оповіщень про нові завдання 

ВИСНОВКИ

1. Проаналізовано аналіз програмних рішень.
2. Спроєктовано архітектуру системи та схему бази даних.
3. Розроблено серверну та клієнтську частини застосунку.
4. Розроблено та протестовано ПЗ для організації роботи працівників складу.
5. Представлені напрямки подальшого вдосконалення.

Розробка виконана у повному обсязі згідно з положенням Технічного завдання, тестування продукту проведено відповідно до затвердженої програми та методики тестування.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ РОБОТОЮ СКЛАДСЬКИХ ПРИМІЩЕНЬ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Гліб ШЕВЧЕНКО

2023

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для управління роботою складських приміщень, який являє собою вебсайт. В якості засобів реалізації застосунку було обрано мову програмування PHP та JavaScript, фреймворки Laravel, Sanctum і Angular, базу даних MySQL.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Функціональна працездатність елементів сторінок вебзастосунку.
2. Наявність доступу до бази даних.
4. Забезпечення належного рівня безпеки даних.
5. Зручність роботи з вебсайтом.
6. Відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

У процесі розробки даного програмного забезпечення було виконано тестування протягом всього періоду розробки. Кожен з модулів був протестований після його реалізації, що дозволило виявляти та усувати можливі помилки на ранніх етапах розробки. Цей підхід допоміг знизити кількість потенційних помилок та мінімізувати ймовірність некоректної поведінки програмного забезпечення.

Для тестування програмного забезпечення був обраний метод ручного тестування. Суть методики полягає в тому, що тестувальник виступає в ролі кінцевого користувача та перевіряє функції програми та її поведінку використовуючи різні ролі. Цей метод дозволяє виявляти потенційні проблеми системи з точки зору кінцевого користувача.

Окрім цього, було проведено тестування програмного забезпечення на відповідність вимогам безпеки. У результаті тестування було виявлено,

що вимоги щодо безпеки були виконані належним чином. База даних була захищена від сторонньої ін’єкції, вебсервіс був захищений від JS-ін’єкцій, а програмне забезпечення мало валідацію даних, введених користувачем.

Усі зібрані зауваження були ретельно опрацьовані, і як результат, були внесені необхідні зміни до розроблюваного програмного забезпечення.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність вебсервісу перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування вебсервісу в різних браузерах.
5. Тестування стабільності роботи при різних умовах.
6. Тестування зручності використання.
7. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ РОБОТОЮ СКЛАДСЬКИХ ПРИМІЩЕНЬ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____Гліб ШЕВЧЕНКО

2024

ЗМІСТ

1. Опис структури вебзастосунку

Вебзастосунок для управління роботою складських приміщень реалізовано у вигляді вебсайту. Він містить у собі наступні складові:

1. Сторінка авторизації.
2. Головну сторінку зі списком складських приміщень.
3. Сторінку перегляду складу.
4. Сторінку перегляду секторів.
5. Сторінка списку товарів.
6. Сторінка поставок.
7. Сторінка генерації звітів.
8. Сторінка кабінету користувача.
9. Сторінка наявних завдань.
10. Сторінка завaднь.
11. Сторінка нового завдання.
12. Сторінка історії завдань.
13. Сторінка створення користувача.

Остання сторінка, згадана у переліку існуючих, відносяться до адміністративної панелі вебзастосунку, доступної лише адміністратору системи.

2. Процедура авторизації користувача

При відвідуванні вебзастосунку для управління роботою складських приміщень користувач потрапляє на сторінку авторизації. На сторінці користувач має заповнити поля форми авторизації: логін та пароль. У результаті коректного внесення інформації та натиснення на клавішу «Sing In» користувач потрапить на головну сторінку вебзастосунку. У іншому випаду, користувач побачить повідомлення про помилку (рис. 1).

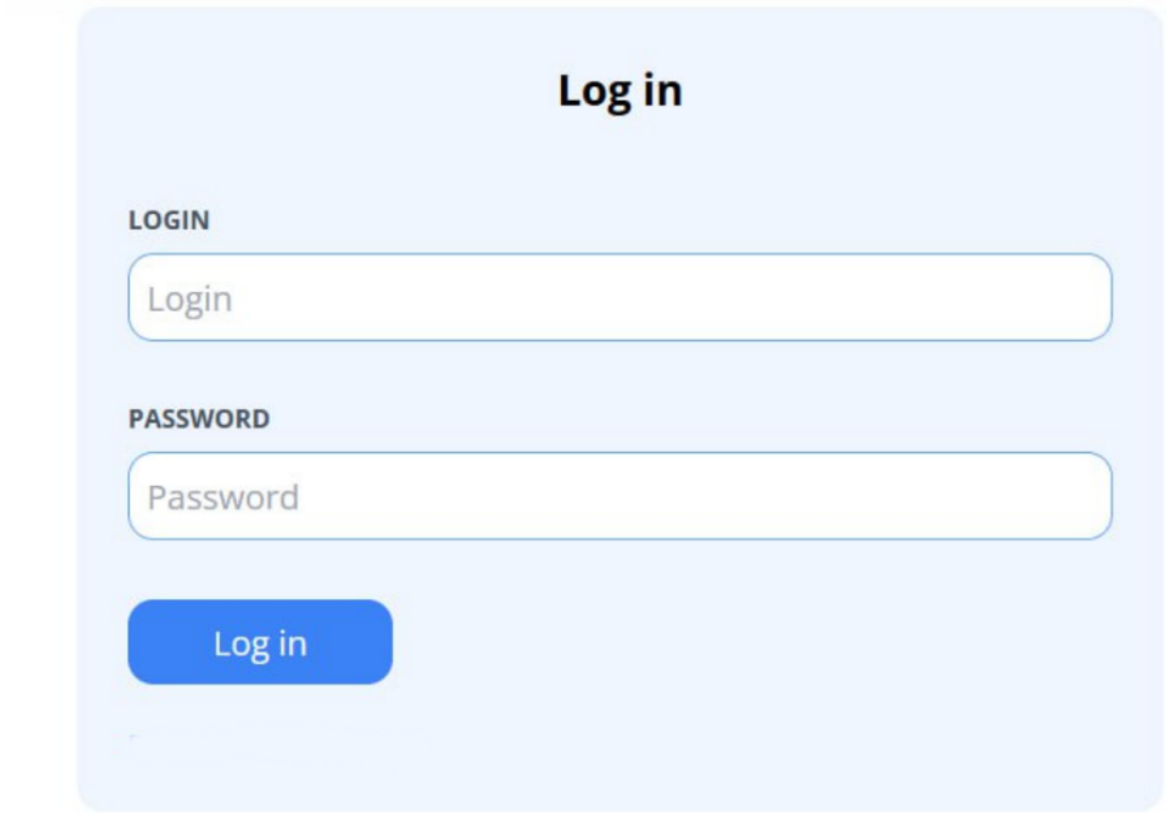


Рис. 1. Форма авторизації користувача

3. Сторінка зі списком складських приміщень

Після успішної процедури авторизації користувач направляється на головну сторінку системи. На цій сторінці користувач може побачити список доступних складських приміщень та інформацію про управляючого. Інформація про управляючого представлена у вигляді блоку з фотографією та іменем. Інформація про склади представлена у вигляді блоків з номерами складу та назвами. У кожному блоці також присутня кнопка «Details», при натисканні переводить користувача на сторінку перегляду складу.

Користувач з правами адміністратора може побачити кнопку «New storage». При натисканні користувач може побачити меню з формою створення складу, де повинен вказати: адресу складу, номер складу, назву складу, опис складу, кількість співробітників. Після заповнення інформації

користувач має натиснути на кнопку «Submit», для створення складу з заданою інформацією (рис. 2).

Крім того, користувач може потрапити на цю сторінку з будь-якого місця програми, натиснувши на логотип «SContorl» у верхньому лівому куті сторінки (рис. 3).

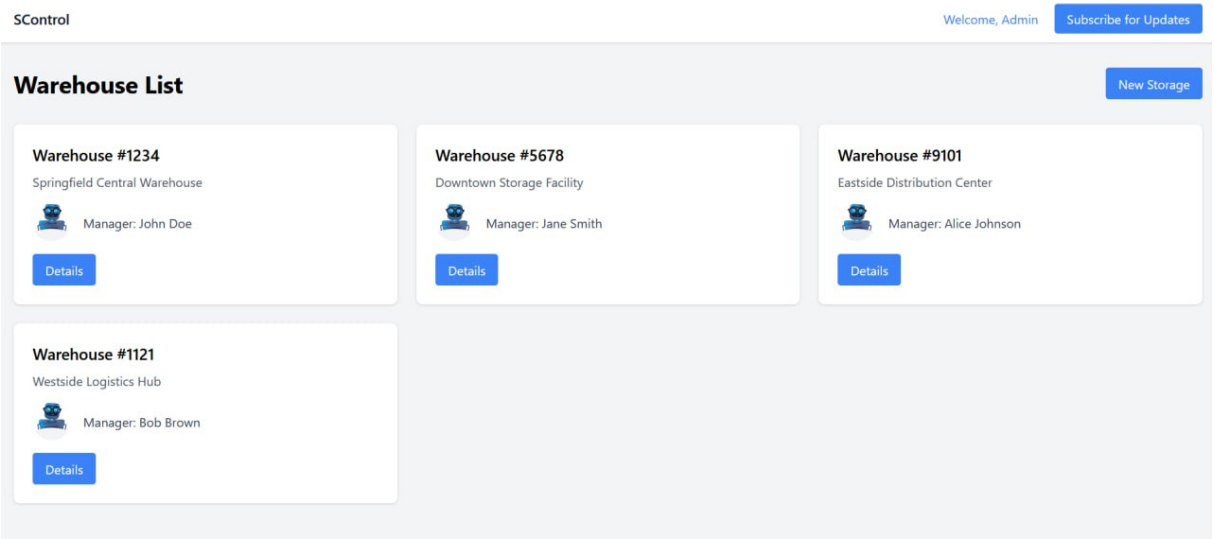


Рис. 2. Список складських приміщень

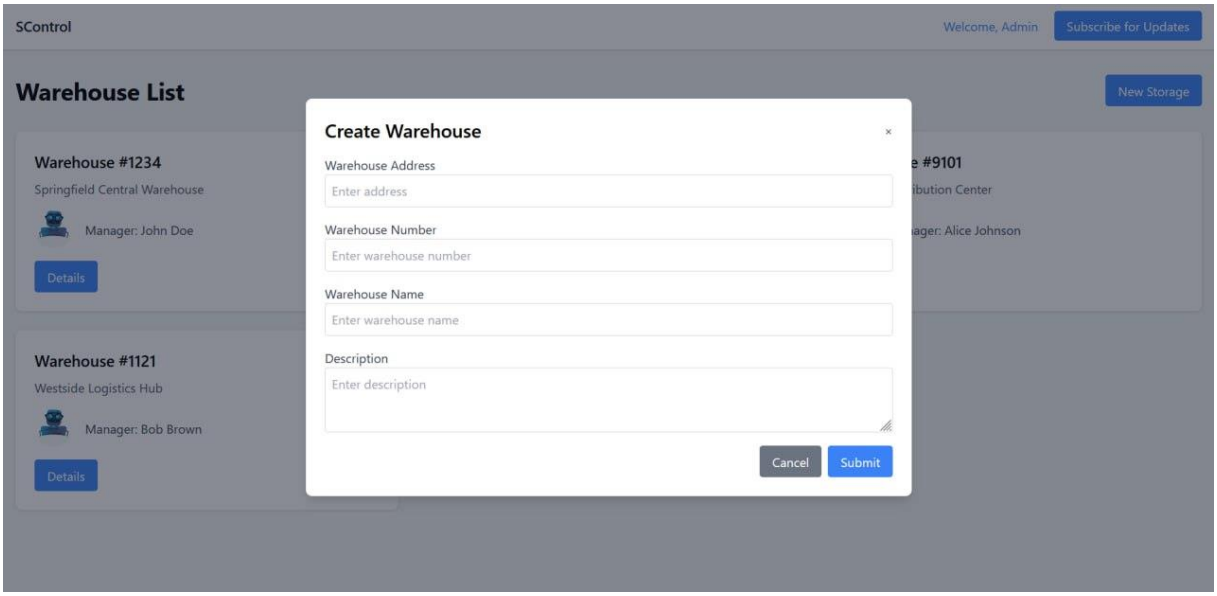


Рис. 3. Меню створення складу

4. Сторінка перегляду складу

Після того, як користувач натискає на кнопку «Details», система направляє його на сторінку перегляду інформації про склад. Там відображається детальна інформація про склад (рис. 4).

Користувач може побачити наступну інформацію: адреса складу, номер складу, назва складу, опис складу, кількість співробітників.

Також, на сторінці можна побачити:

1. Кнопку «Sectors List». При натисканні переводить користувача на сторінку списку секторів.
2. Кнопку «Goods». При натисканні переводить користувача на сторінку списку товарів.
3. Кнопку «Delivery List». При натисканні переводить користувача на сторінку списку поставок.
4. Кнопку «Reports». При натисканні переводить користувача на сторінку генерації звітів.
5. Кнопку «Tasks». При натисканні переводить користувача на сторінку завдань. Цю кнопку бачить лише адміністратор та користувач з роллю «Управляючий».

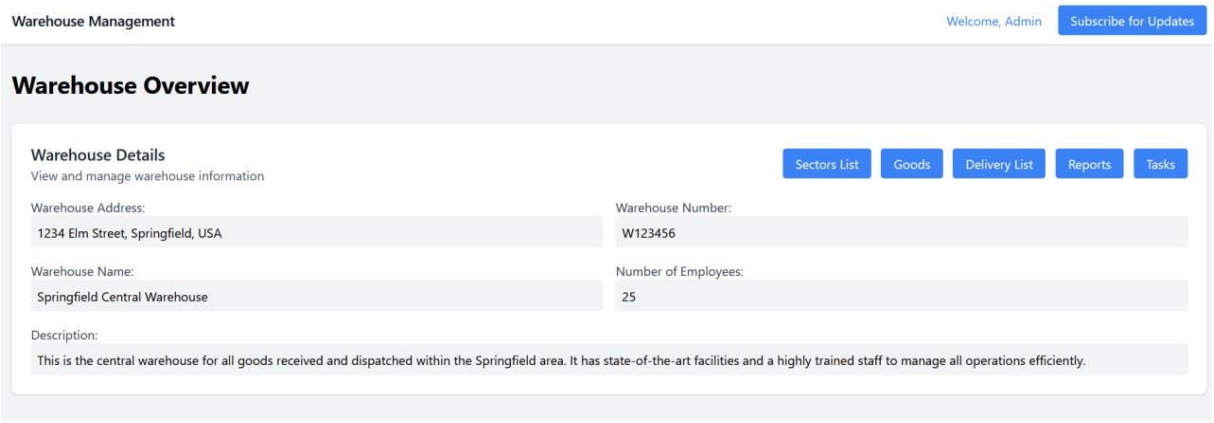


Рис. 4. Сторінка складу

5. Сторінка перегляду секторів

Після того, як користувач натискає на кнопку «Sectors List», система направляє його на сторінку списку секторів. Там відображається детальна інформація про сектор: номер сектору, назва сектору, опис сектору (рис. 5).

На цій сторінці користувач може побачити список доступних секторів. Інформація про сектори представлена у вигляді блоків, що містять номер та назву. Також, в кожному блоці присутня кнопка «Goods List», яка при натисканні відображає меню зі списком товарів сектору. Користувач може отримати інформацію про кількість та назву товарів на складі (рис. 6).

Користувач з правами адміністратора, або роллю «Управляючий» може побачити кнопку «New sector». При натисканні користувач може побачити меню з формою створення сектору, де повинен вказати: номер складу, назву складу, опис складу. Після заповнення інформації користувач має натиснути на кнопку «Submit», для створення сектору з заданою інформацією (рис. 7).

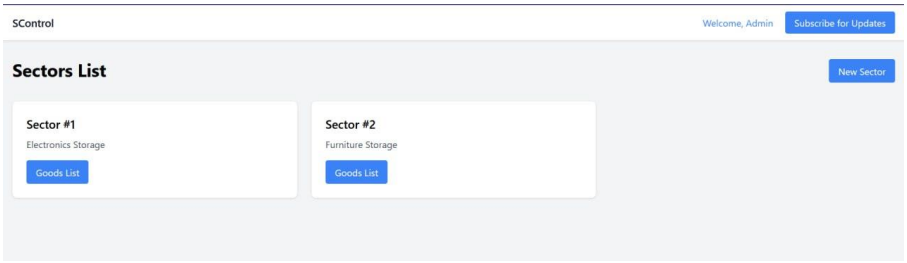


Рис. 5. Список секторів

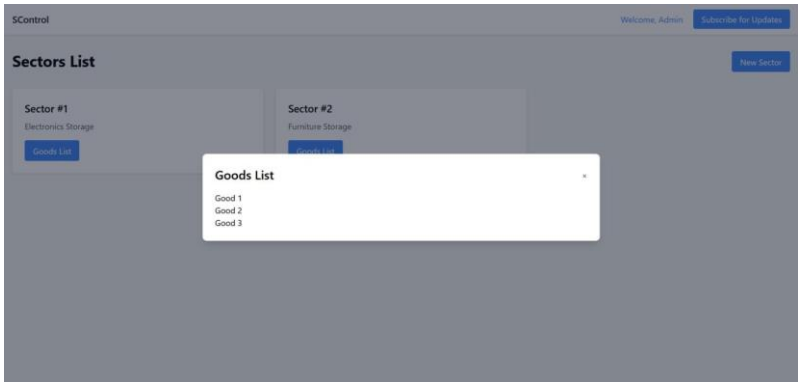


Рис. 6. Меню списку товарів сектору

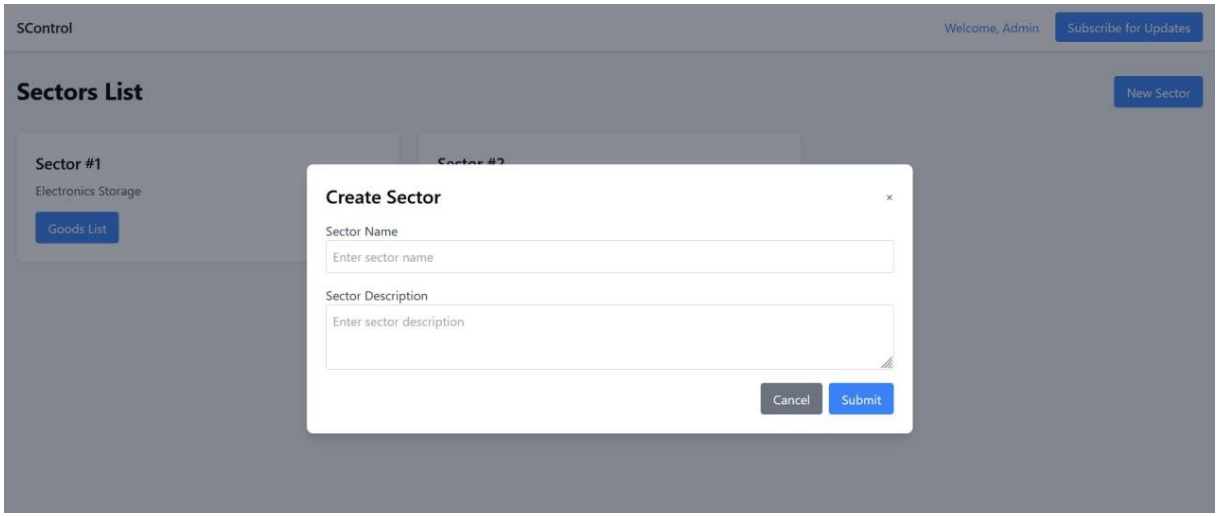


Рис. 7. Меню створення сектору

6. Сторінка списку товарів

Після того, як користувач натискає на кнопку «Goods», система направляє його на сторінку перегляду товарів, що знаходяться на обраному секторі. Там відображається інформація про товари на складі у вигляді списку. Користувач може натиснути на назву товару, щоб переглянути меню з детальною інформацією про товар: назву, опис, ціну, строк придатності (рис. 8).

Користувач з правами адміністратора, або роллю «Управляючий» може побачити кнопку «New good». При натисканні користувач може побачити меню з формою створення складу, де повинен вказати: назву, опис, ціну, строк придатності. Після заповнення інформації користувач має натиснути на кнопку «Submit», для створення товару з заданою інформацією (рис. 9).

Також, користувач з правами адміністратора, або роллю «Управляючий» бачить кнопку у вигляді кошику поряд з кожним товаром. При її натисканні товар буде видалено зі списку.

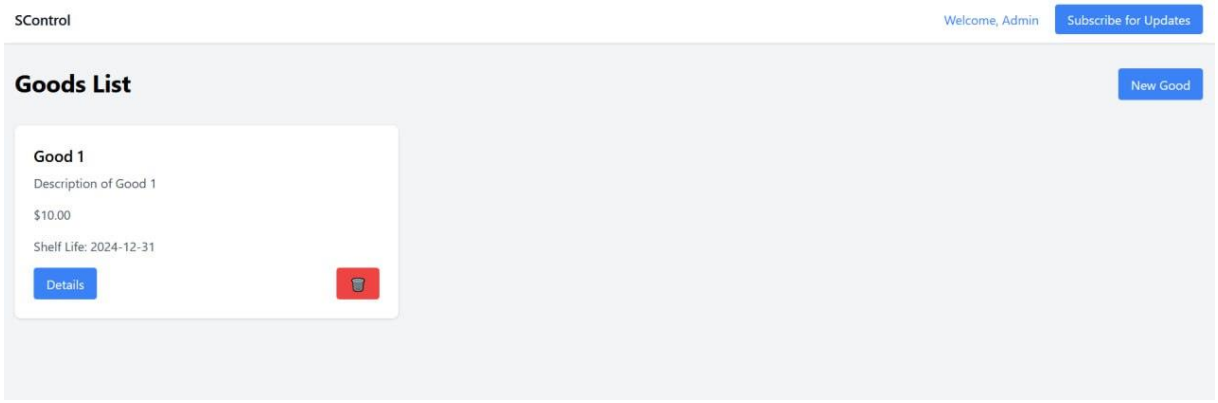


Рис. 8. Список товарів

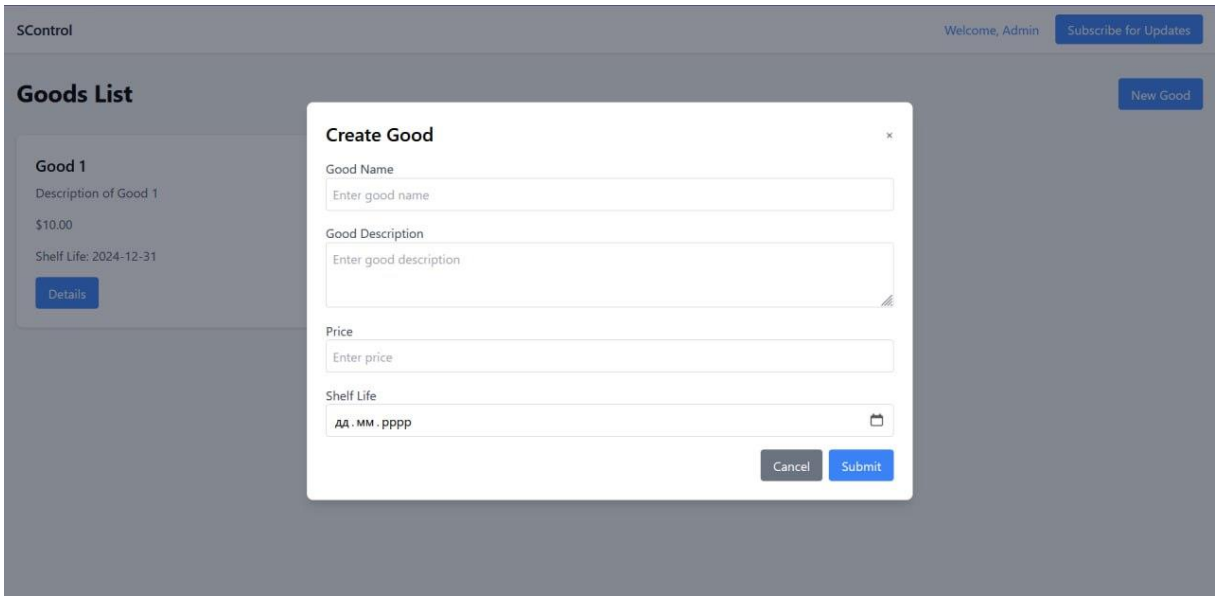


Рис. 9. Меню створення товару

7. Сторінка поставок

Після того, як користувач натискає на кнопку «Delivery List», система направляє його на сторінку перегляду поставок. Там відображається інформація про актуальні поставки до складу у вигляді списку. Користувач може натиснути на номер поставки, щоб переглянути меню з детальною інформацією про поставку: номер, опис, дату, список товарів (рис. 10).

Користувач з правами адміністратора, або роллю «Управляючий» може побачити кнопку «New delivery». При натисканні користувач може побачити меню з формою створення поставки, де повинен вказати: номер, опис, дату, список товарів. Після заповнення інформації користувач має

натиснути на кнопку «Submit», для створення поставки з заданою інформацією (рис. 11).

Також, користувач з правами адміністратора, або роллю «Управляючий» бачить кнопку у вигляді кошику поряд з кожною поставкою. При її натисканні доставку буде видалено зі списку.

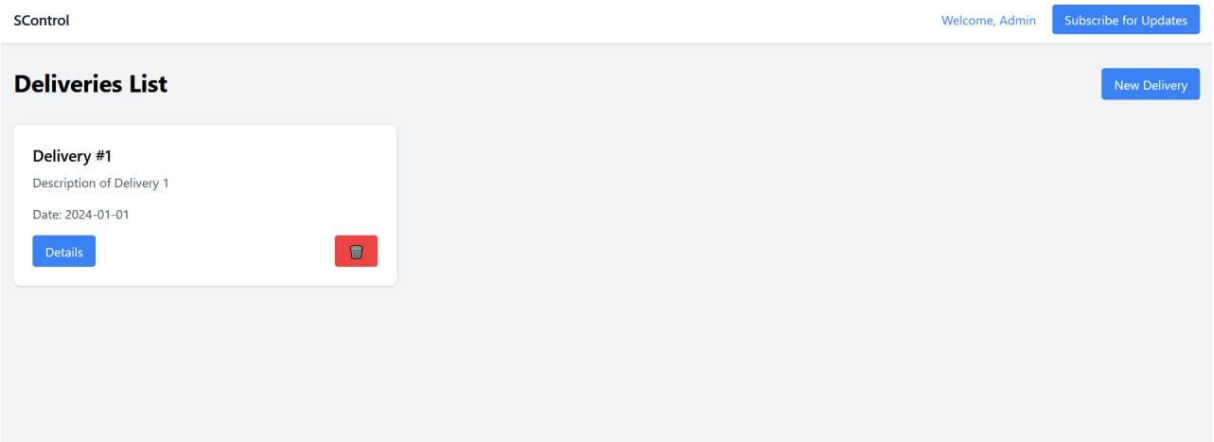


Рис. 10. Список поставок

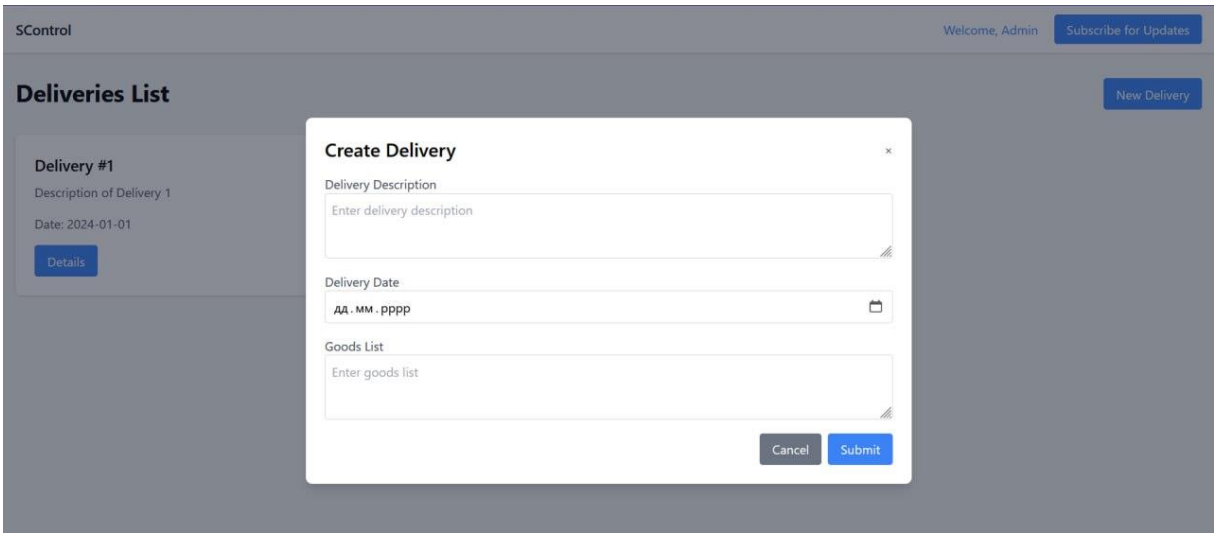


Рис. 11. Меню створення поставки

8. Сторінка генерації звітів

Після того, як користувач натискає на кнопку «Reports», система направляє його на сторінку створення звітів. Користувач може побачити наступну кнопку «Make Report» (рис. 12). При натисканні створює звіт за

останній місяць та скачує його у вигляді pdf файлу. У звіті користувач отримує детальну інформацію про роботу складу: кількість поставок, кількість завдань, кількість виконаних завдань, кількість провалених завдань (рис. 13).

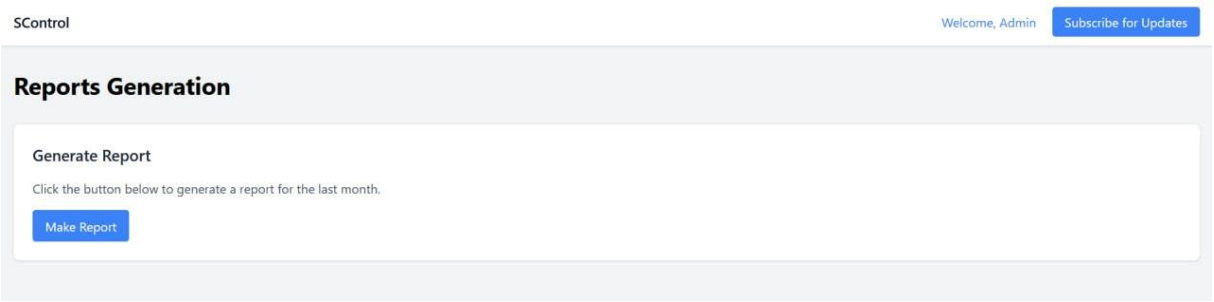


Рис. 12. Сторінка генерації звіту

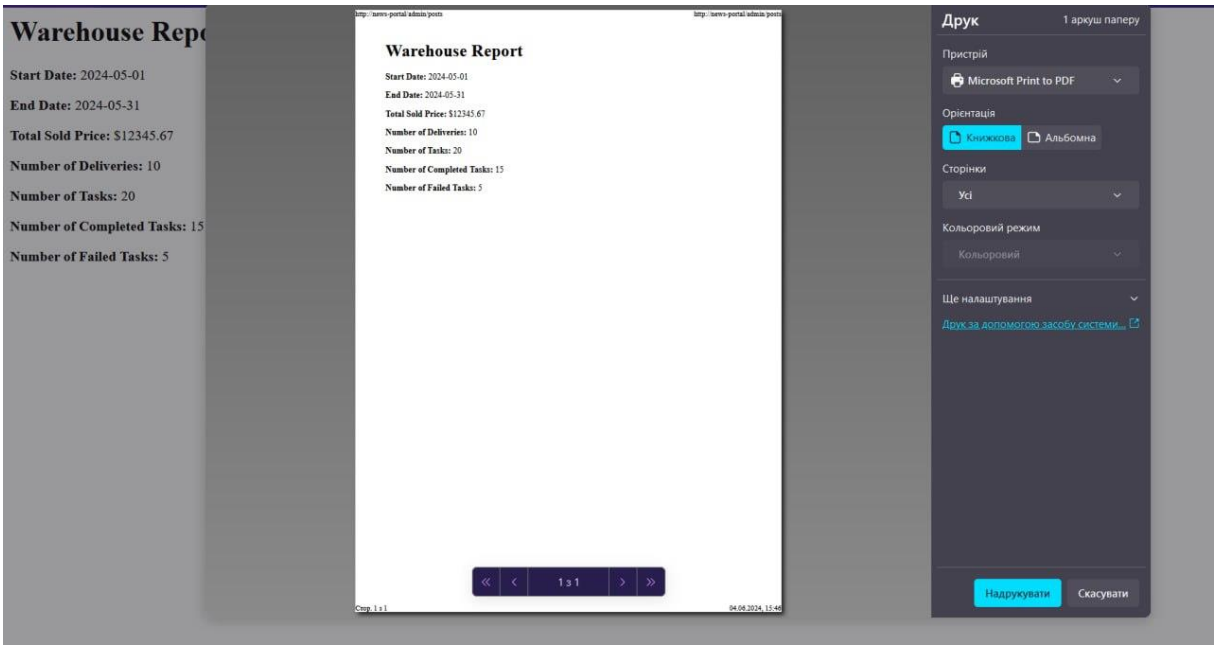


Рис. 13. Приклад звіту

9. Сторінка завдань

Після того, як користувач натискає на кнопку «Tasks», система направляє його на сторінку управління завданнями (рис. 14). Користувач може побачити кнопки:

1. «New Task». При натисканні переводить користувача на сторінку нового завдання.

2. «Tasks History». При натисканні переводить користувача на сторінку історії завдань.

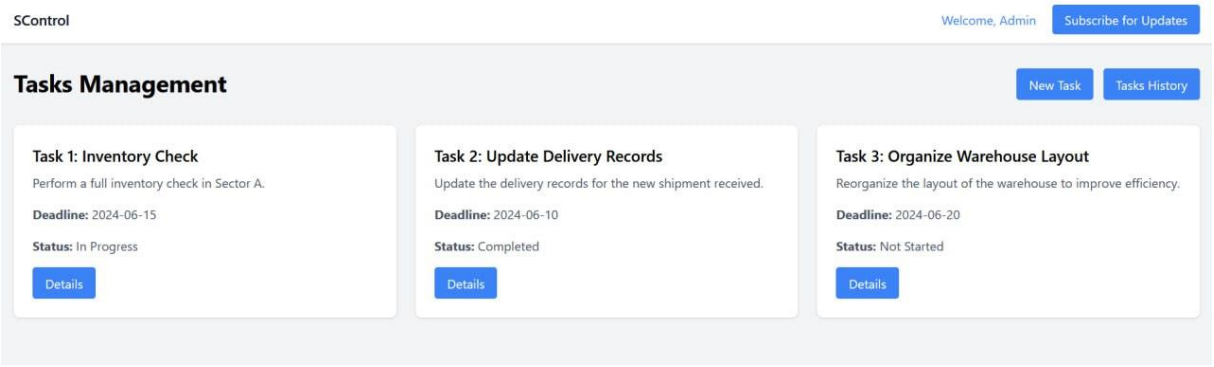


Рис. 14. Сторінка завдань

10. Сторінка нового завдання

Після того, як користувач натискає на кнопку «New Task», система направляє його на сторінку створення завдання. Вся сторінка є формою, де користувач має вказати: тему завдання, сектор, який відноситься до завдання, час виконання завдання, працівники, що виконують завдання, опис завдання, поставку, що відноситься до завдання. Тему завдання та опис завдання користувач вписує в поле власноруч. Натомість сектор, час, поставку та працівників користувач обирає з розкривного списку. Користувач може обрати декілька працівників, але сектор, поставку та час лише один кожного списку окремо.

По завершенню заповнення форми користувач може натиснути кнопку «Create», щоб підтвердити завдання. Програма здійснює перевірку, що всі обов’язкові поля були заповнені. Обов’язковими полями рахуються усі поля окрім поля поставки. Якщо все заповнено правильно, то завдання буде створене, а користувача поверне до сторінки задвань. У іншому випадку – користувачу покажуть поле, що він не заповнив, або заповнив неправильно і виведуть повідомлення з помилкою (рис. 15).

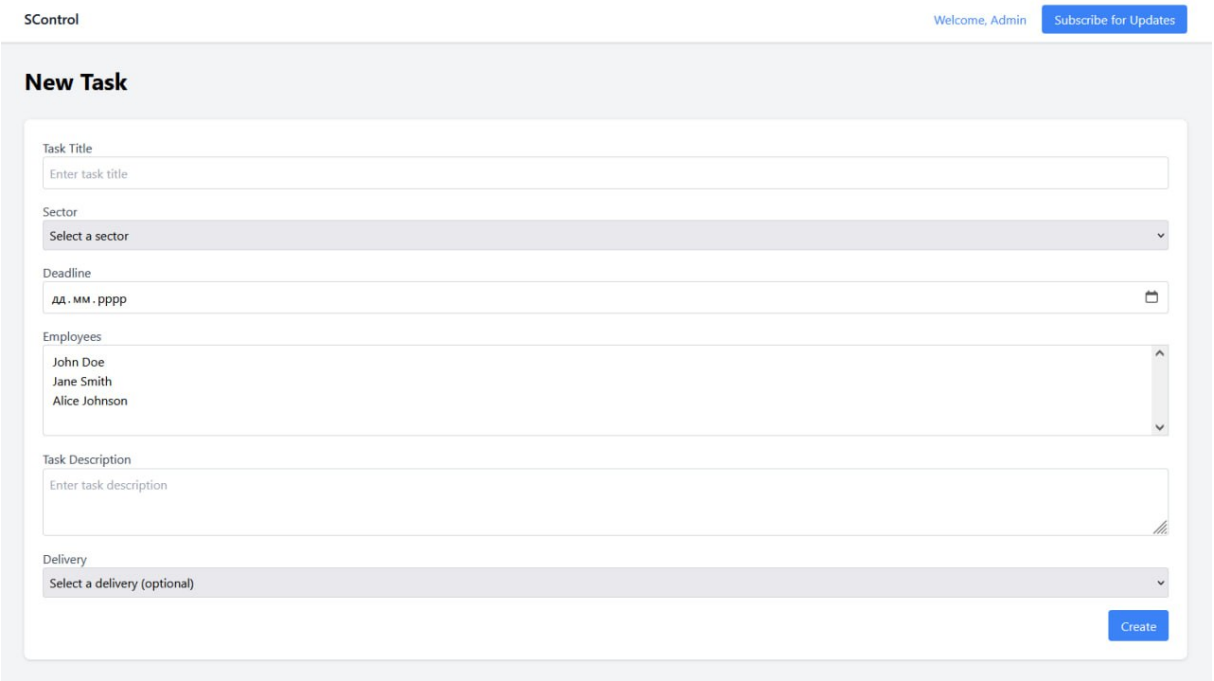


Рис. 15. Сторінка нового завдання

11. Сторінка історії завдань

Після того, як користувач натискає на кнопку «Tasks History», система направляє його на сторінку перегляду історії завдань. Там відображається інформація про завершені завдань у вигляді списку. Користувач може натиснути на назву завдання, щоб переглянути меню з детальною інформацією про завдання: тему завдання, сектор, який відноситься до завдання, час виконання завдання, працівники, що виконують завдання, опис завдання, поставку, що відноситься до завдання, результат виконання завдання. Результат завдання може поділятися на «Completed», у разі успішного виконання, або «Failed» у разі невиконання (рис. 16).

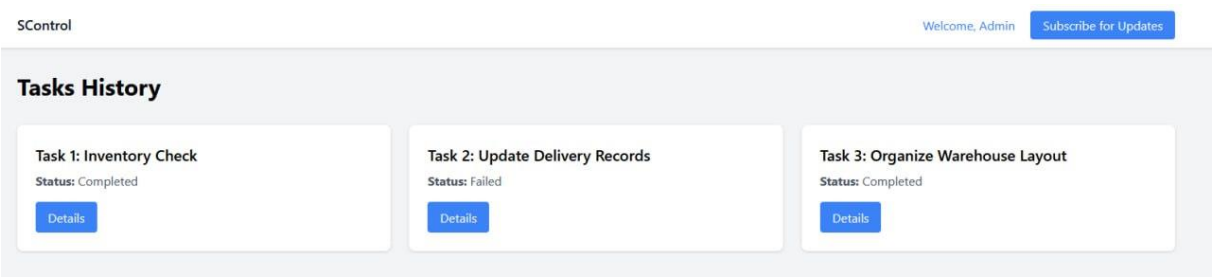


Рис. 16. Сторінка історії завдань

12. Сторінка кабінету користувача

Після того, як користувач натискає на кнопку зі своїм ім’ям, система направляє його на сторінку користувача. Користувач може побачити наступну інформацію: фото профілю, ім’я користувача, роль користувача.

Користувач з роллю «працівник», або з роллю «Управляючий» бачить кнопку «Tasks List». При натисканні переводить користувача на сторінку наявних завдань.

Користувач з правами адміністратора бачить кнопку «New User». При натисканні переводить користувача на сторінку створення користувача.

Майже кожна сторінка містить посилання на кабінет користувача у вигляді імені користувача у правому верхньому кутку вікна (рис. 17).

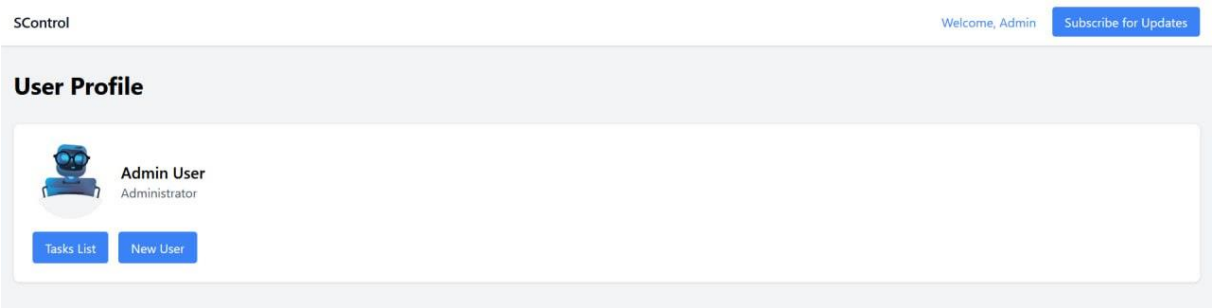


Рис. 17. Сторінка кабінету користувача

13. Сторінка наявних завдань

Після того, як користувач натискає на кнопку «Tasks List», система направляє його на сторінку перегляду наявних завдань користувача (рис. 18). Там відображається інформація про завдання, які користувач має виконати у вигляді списку. Поруч з назвою завдання користувач може побачити кнопки:

1. «Completed». При натисканні користувач повідомляє про успішне виконання завдання. Статус завдання зміниться на «Completed» лише при натисканні цієї кнопки усіма працівниками, що відносяться до завдання.

2. «Failed». При натисканні користувач повідомляє про невдале виконання завдання. Статус завдання зміниться на «Failed», якщо один з працівників, що відноситься до завдання, натисне кнопку «Failed», або коли час завдання сплине.

Користувач може натиснути на назву завдання, щоб переглянути меню з детальною інформацією про завдання: тему завдання, сектор, який відноситься до завдання, час виконання завдання, працівники, що виконують завдання, опис завдання, поставку, що відноситься до завдання (рис. 19).

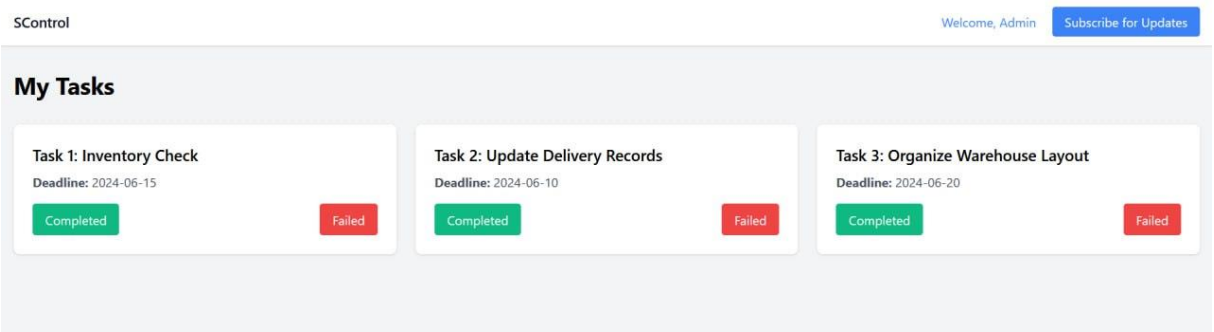


Рис. 18. Сторінка наявних завдань

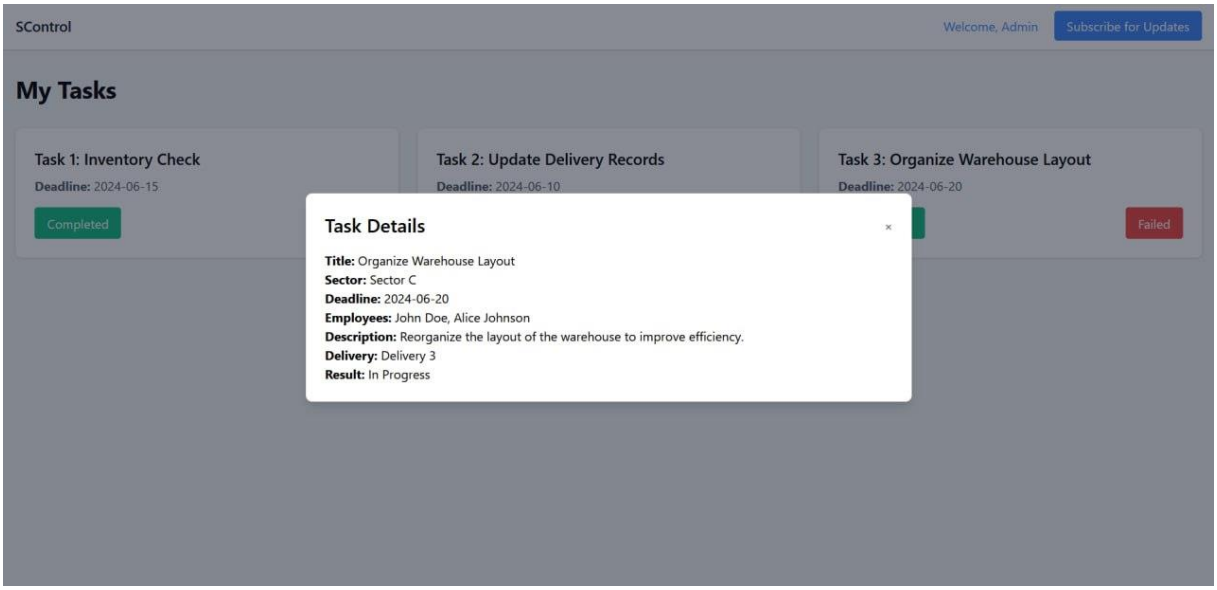


Рис. 19. Меню деталей завдання

14. Сторінка створення користувача

Після того, як користувач натискає на кнопку «New User», система направляє його на сторінку створення користувача. Сторінка складається з форми, в якій потрібно вказати: логін, пароль, роль користувача, ім’я користувача, склад, на якому працює користувач. Також присутнє поле для завантаження фотографії користувача. Після завершення заповнення форми, користувач має натиснути кнопку «Sign Up» і його буде перенесено на сторінку свого кабінету користувача, а створеного користувача буде занесено в систему (рис. 20).

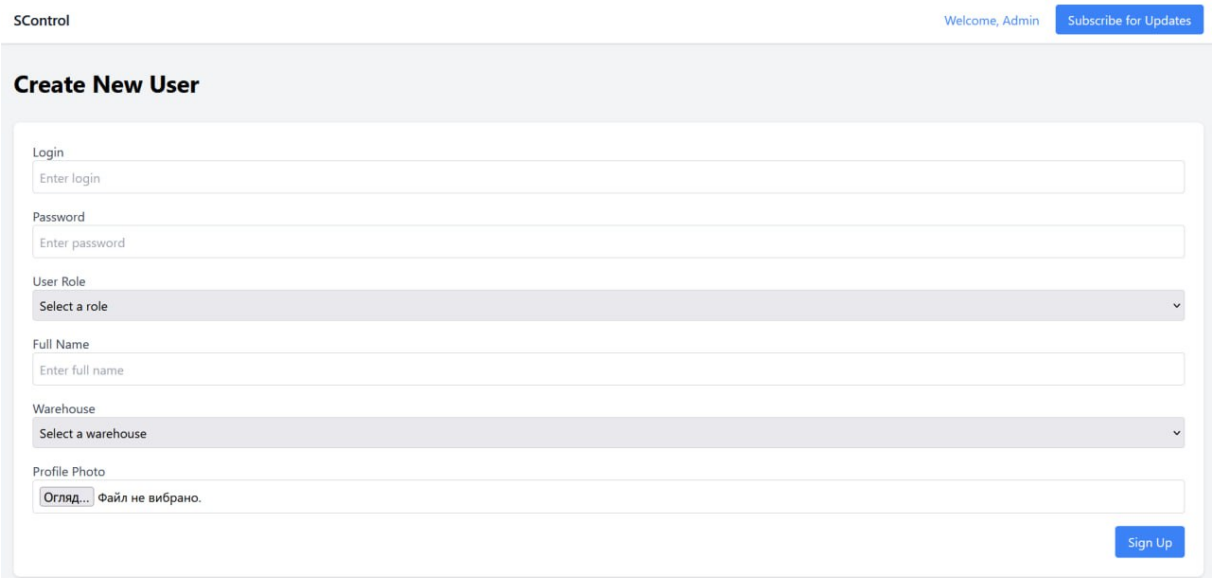


Рис. 20. Сторінка створення користувача