Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп’ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

перший (бакалаврський)
(рівень вищої освіти)

Розроблення системи автоматизованого управління складською системою підприємства з використання хмарних технологій
(тема)

Виконав:
студент 4 курсу, групи ТРІТЗР-20-1

Науменко Д. І.

Спеціальності 172 Телекомунікації та радіотехніка

Тип програми Освітньо-професійна

Освітня програма Інтелектуальні технології засобів радіоелектроніки

Керівник доц. каф. КІТАР Бронніков А. І.

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав та не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

17.червня.2024 Науменко Д.І.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

АКТ

Кафедра КІТАР

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації та радіотехніка

Тип програми Освітньо-професійна

Освітня програма Інтелектуальні технології засобів радіоелектроніки

(шифр і назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри КІТАР _____

(підпис)

«__04__» __квітня__ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Науменко Данилу Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи автоматизованого управління складською системою з використання хмарних технологій

Затверджена наказом по університету від 20.05.2024 р. № 479 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024 р.

3. Вихідні дані до роботи мова JavaScript, автоматизація управління, хмарні технології, бази даних

4. Перелік питань, що потрібно опрацювати в роботі Вступ, аналіз літератури та інформації, вибір і обґрунтування технічних засобів, розроблення системи автоматизованого управління складською системою з використання хмарних технологій, висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайди у форматі Power Point у кількості 12 слайдів з розширенням .pptx

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Змістовний опис та аналіз структурних і функціональних особливостей предметної області університету та опис об'єкта автоматизації	20.05 – 21.05.24	Виконано
2	Огляд і аналіз сучасного стану розглянутої задачі, а також існуючих методів і засобів вирішення задач кваліфікаційної роботи	22.05– 24.05.24	Виконано
3	Формулювання завдання розробки	25.05– 26.05.24	Виконано
4	Опис архітектури об'єкта розробки на рівні функцій	27.05– 28.05.24	Виконано
5	Обґрунтування літератури інформаційної системи	29.05– 30.05.24	Виконано
6	Вибір і обґрунтування технічних засобів	01.06– 02.06.24	Виконано
7	Розробка і обґрунтування елементів програмної забезпечуючих системи	03.06– 05.06.24	Виконано
8	Подання роботи на перевірку Інтернет-сервісом Unicheck	07.06 – 08.06.24	Виконано
9	Оформлення пояснювальної записки	08.06 – 10.06.22	Виконано
10	Подання роботи на рецензію	11.06 – 13.06.22	Виконано
11	Подання роботи на підпис зав. кафедри	14.06 – 10.06.22	Виконано
12	Подання атестаційної роботи в ЕК	20.06.2022	Виконано

Дата видачі завдання 25.04.2024 р.

Студент _____ Науменко Д. І.
(підпис)

Керівник роботи _____ доц. Бронніков А. І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 91 стор., 24 рис., 2 дод., 19 джерел.

ІТ-СЕРВІС, МОНІТОРИНГ, ЗАМОВЛЕННЯ, АНАЛІТИКА, СИСТЕМА АВТОМАТИЗОВАНОГО УПРАВЛІННЯ, АВТОРИЗАЦІЯ, NODE.JS, DIRECTUS, EXPRESS.JS, REACT.JS, TYPESCRIPT, POSTGRESQL,

У сучасному світі інформаційні технології відіграють ключову роль у всіх сферах бізнесу, включаючи управління складською системою. Метою роботи є створення ефективного та надійного інструменту для автоматизації складських операцій, що дозволить знизити витрати на обслуговування складу, підвищити продуктивність праці та забезпечити прозорість процесів. Об’єкт розробки: процес управління розумним виробництвом виробництвом. Предмет: автоматизоване управління складською системою як частиною розумного виробництва з використанням хмарних технологій. Методи розробки: аналіз наукових статей, книг, та інших ресурсів для збору інформації про сучасні практики управління складами та хмарні технології, аналіз даних для збору та обробка даних про роботу складських систем для виявлення тенденцій та закономірностей, прототипування - для створення робочої моделі системи і перевірки її ефективності та можливостей.

Для розробки застосунку були використані сучасні технології: frontend на основі Next.js React.js та TypeScript, backend на основі Node.js, Express.js та Directus, база даних PostgreSQL. Система включає модулі авторизації, керування замовленнями, створення нового замовлення, аналітики та статистики.

У результаті роботи було розроблено систему автоматизованого управління складською системою, яка забезпечує ефективне управління складськими операціями, підвищення продуктивності праці та прозорість

процесів. Використання сучасних технологій, таких як Next.js, React, Node.js, Directus та PostgreSQL, а також інтеграція з хмарними сервісами, дозволяють досягти високої продуктивності та надійності системи. Забезпечення охорони праці та дотримання нормативних вимог сприяють створенню безпечних умов праці для працівників.

ABSTRACT

The explanatory note: 91 p., 24 fig., 2 app., 19 sources.

IT-SERVICE, MONITORING, RECORDING, ANALYTICS, AUTOMATED MANAGEMENT SYSTEM, AUTHORIZATION, NODE.JS, DIRECTUS, EXPRESS.JS, REACT.JS, TYPESCRIPT, POSTGRESQL,

Today, information technologies play a key role in all areas of business, including warehouse system management. The goal of the project is to create an effective and reliable tool for automating warehouse operations, which will reduce costs for warehouse maintenance, increase productivity and ensure transparency of processes.

To explore the details, we used current technologies: frontend based on Next.js React.js and TypeScript, backend based on Node.js, Express.js and Directus, PostgreSQL database. The system includes authorization modules, transaction processing, new contract creation, analytics and statistics.

As a result of the work, an automated warehouse system management system was developed, which ensures effective management of warehouse operations, increased productivity and process visibility. The use of current technologies, such as Next.js, React, Node.js, Directus and PostgreSQL, as well as integration with advanced services, allows us to achieve high productivity and reliability of the system. The security of the protection of the praci and the development of normative regulations can be used to protect the creation of careless minds for practitioners.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АСУ – автоматичні системи управління;

САУ – системи автоматизованого та автоматичного управління;

СУБД – система управління базами даних;

ШІ – штучний інтелект;

AR – доповнена реальність;

API – інтерфейс програмування додатків;

BCI – інтерфейси мозок-комп'ютер;

CI/CD – безперервна інтеграція/безперервне розгортання;

IaaS – інфраструктура як послуга;

IoT – інтернет речі;

MTBF – середній час між відмовами;

MTTR – середній час відновлення;

PaaS – платформа як послуга;

SaaS – програмне забезпечення як послуга;

UI – користувацький інтерфейс;

VR – віртуальна реальність.

ВСТУП

В сучасному світі ефективне управління складськими системами є ключовим елементом успішної діяльності будь-якого підприємства. Склади відіграють важливу роль у забезпеченні безперервності виробничих процесів, швидкого та своєчасного постачання продукції споживачам, а також оптимізації логістичних витрат. З розвитком технологій та зростанням обсягів даних, традиційні методи управління складом стають все менш ефективними, що вимагає впровадження інноваційних рішень [1].

Одним із таких інноваційних рішень є застосування хмарних технологій для автоматизації складських процесів. Хмарні технології надають можливість зберігати та обробляти великі обсяги даних, забезпечувати доступ до них у режимі реального часу з будь-якої точки світу, а також інтегрувати різні системи та платформи для досягнення високої ефективності. Використання хмарних сервісів дозволяє знижувати витрати на інфраструктуру, забезпечувати гнучкість та масштабованість системи, а також підвищувати рівень безпеки даних.

Метою даної дипломної роботи є розробка системи автоматизованого управління складською системою з використанням хмарних технологій. Ця система повинна забезпечити автоматизацію основних складських процесів, таких як прийом товарів, інвентаризація, обробка замовлень та відвантаження продукції, а також надавати інструменти для моніторингу та аналізу роботи складу.

Для досягнення цієї мети у роботі передбачено вирішення наступних завдань:

– аналіз сучасних тенденцій в області автоматизованих складських систем та хмарних технологій;

– вибір та обґрунтування технічних засобів для розробки системи;

– розробка програмного забезпечення, включаючи проектування архітектури, реалізацію інтерфейсів користувача та серверної частини, а також інтеграцію з хмарними сервісами;

– проведення заходів з охорони праці для забезпечення безпеки працівників у процесі впровадження та експлуатації системи.

Об’єкт розробки: процес управління розумним виробництвом виробництвом. Предмет: автоматизоване управління складською системою як частиною розумного виробництва з використанням хмарних технологій. Методи розробки: аналіз наукових статей, книг, та інших ресурсів для збору інформації про сучасні практики управління складами та хмарні технології, аналіз даних для збору та обробка даних про роботу складських систем для виявлення тенденцій та закономірностей, прототипування - для створення робочої моделі системи і перевірки її ефективності та можливостей.

У процесі розробки буде використано сучасні технології та інструменти, такі як JavaScript, Visual Studio Code, Next.js, React, Node.js, Directus та PostgreSQL. Це дозволить створити надійну, масштабовану та ефективну систему, яка відповідатиме вимогам сучасного ринку та забезпечуватиме високий рівень автоматизації складських операцій.

Розроблена система дозволить підприємствам підвищити продуктивність праці, знизити витрати на управління складськими операціями, забезпечити точність та своєчасність виконання замовлень, а також покращити загальну ефективність бізнес-процесів. Впровадження таких систем є важливим кроком на шляху до цифрової трансформації та підвищення конкурентоспроможності підприємств в умовах сучасної економіки

Робота виконана згідно з [2] та [3].

1 АНАЛІЗ ЛІТЕРАТУРИ ТА ІНФОРМАЦІЇ

1.1 Індустрія 5.0 та її основні компоненти

Світ стоїть на порозі нової промислової революції, відомої як Індустрія 5.0. Ця ера ґрунтується на досягненнях Індустрії 4.0, яка зосереджувалась на автоматизації та цифрових технологіях [4]. Індустрія 5.0 робить крок далі, ставлячи в центр уваги людину, стійкість та співпрацю.

Основні принципи Індустрії 5.0:

– людиноцентричність. Індустрія 5.0 прагне до гармонійного поєднання людських здібностей та передових технологій. Це передбачає розширення можливостей людей, а не їх заміну роботами;

– стійкість. Ця ера виробництва ставить за мету мінімізувати негативний вплив на довкілля та максимізувати використання ресурсів. Це включає в себе принципи циркулярної економіки та "зеленого" виробництва;

– співпраця. Індустрія 5.0 стимулює співпрацю між людьми, машинами та компаніями для досягнення спільних цілей. Це передбачає відкритість даних, спільні платформи та нові моделі співпраці (рис 1.1).

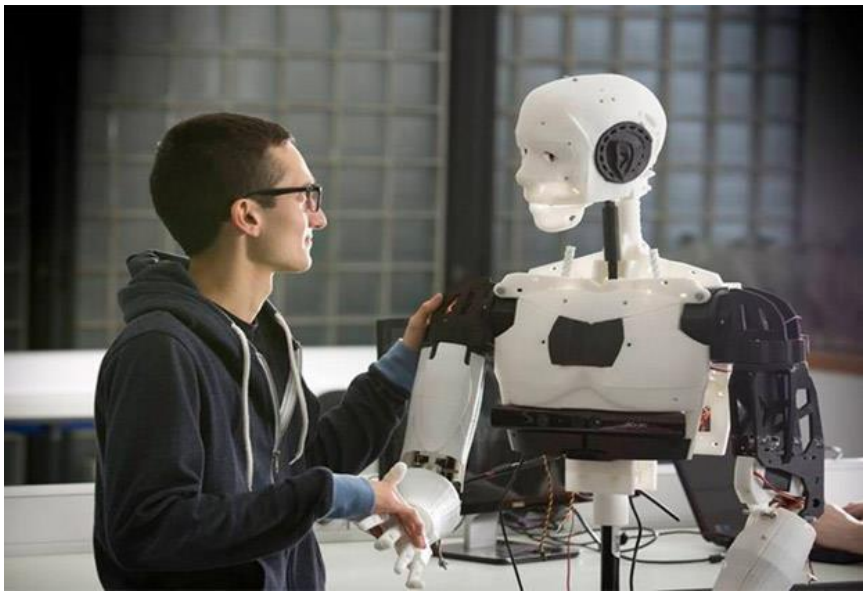


Рисунок 1.1 – Взаємодія людини і робота

Ключові технології Індустрії 5.0:

– штучний інтелект (ШІ). ШІ буде відігравати ключову роль у забезпеченні співпраці людини та машини, прийнятті рішень та оптимізації процесів;

– інтернет речей (IoT). IoT дозволить збирати та аналізувати дані в режимі реального часу з датчиків, вбудованих у машини та продукти;

– хмарні обчислення. Хмарні платформи забезпечать масштабованість, гнучкість та безпеку для даних та програмного забезпечення Індустрії 5.0;

– робототехніка. Роботи нового покоління, здатні до співпраці та навчання, будуть працювати пліч-о-пліч з людьми [5];

– 3D-друк. 3D-друк революціонізує виробництво, дозволяючи створювати складні продукти та персоналізувати їх [6].

Індустрія 5.0 знаменує собою нову еру в промисловому розвитку, де люди та машини працюють пліч-о-пліч, а не один замість одного. На відміну від попередніх індустріальних революцій, які фокусувалися на автоматизації та ефективності, Індустрія 5.0 ставить пріоритетом співпрацю, стійкість та персоналізацію. На рис 1.2 можна побачити одні з її переваг.

Переваги Індустрії 5.0:

– підвищення продуктивності та ефективності. Інтеграція технологій та співпраця призведуть до значного скорочення часу та ресурсів, необхідних для виробництва;

– покращення якості. ШІ та аналітика даних допоможуть виявляти та усувати дефекти, гарантуючи високу якість продукції. Нові продукти та послуги. Індустрія 5.0 відкриває можливості для створення інноваційних продуктів та послуг, які раніше були неможливими;

– збільшення стійкості. Перехід до циркулярної економіки та "зеленого" виробництва зменшить негативний вплив на довкілля;

– покращення умов праці. Людська праця стане більш творчою та цікавою, а небезпечні та рутинні завдання будуть автоматизовані.

Виклики Індустрії 5.0:

– кібербезпека. Зростання кількості підключених пристроїв та обсяг даних збільшують ризики кібератак;

– етичні питання. Використання ШІ та інших технологій викликає етичні питання, пов'язані з конфіденційністю, упередженнями та відповідальністю;

– соціальні та економічні зміни. Перехід до Індустрії 5.0 може призвести до втрати робочих місць у деяких секторах, потребуючи перекваліфікації та створення нових робочих місць;

– інклюзивність. Важливо забезпечити доступ до переваг Індустрії 5.0 для всіх людей та країн, щоб уникнути поглиблення нерівності.

Рисунок 1.2 – Переваги Індустрії 5.0

Індустрія 5.0 не є монолітною концепцією, а складається з декількох взаємопов’язаних компонентів:

а) людиноцентричність:

– розширення можливостей людей. Інтеграція технологій має підсилювати, а не замінювати людські здібності. Це передбачає навчання протягом життя, розвиток нових навичок та створення робочих місць, які потребують творчості, співпраці та критичного мислення;

– ергономіка та безпека. Робочі місця та технології повинні бути розроблені з урахуванням людської фізіології та психології, щоб забезпечити комфорт, безпеку та здоров'я працівників;

– етика та відповідальність. Важливо встановити чіткі етичні рамки для використання ШІ та інших технологій, щоб гарантувати їхнє використання на благо людей та суспільства.

б) стійкість:

– циркулярна економіка. Індустрія 5.0 прагне до мінімізації відходів та максимізації повторного використання ресурсів. Це включає в себе розробку продуктів, які можна легко ремонтувати, переробляти та повторно використовувати, а також нові моделі споживання, такі як оренда та спільне використання;

– «зелене» виробництво. Виробничі процеси повинні бути екологічно чистими та енергоефективними, щоб мінімізувати викиди парникових газів та забруднення довкілля;

– відновлювані джерела енергії. Індустрія 5.0 має ґрунтуватися на відновлюваних джерелах енергії, таких як сонячна, вітрова та геотермальна, щоб зменшити залежність від викопного палива.

в) співпраця:

– співпраця людина-машина. Люди та машини повинні працювати разом як партнери, доповнюючи сильні та слабкі сторони один одного;

– співпраця між компаніями. Відкритість даних, спільні платформи та нові моделі співпраці між компаніями сприятимуть інноваціям та вирішенню складних проблем;

– співпраця з зацікавленими сторонами. Уряди, наукові кола, громадські організації та інші зацікавлені сторони повинні співпрацювати, щоб розробити та впровадити принципи Індустрії 5.0.

г) інтегровані цифрові технології:

– великі дані. Аналітика великих даних допоможе виявляти нові закономірності, приймати кращі рішення та оптимізувати виробничі процеси.

– мобільні технології. Мобільні пристрої та програми дозволять людям отримувати доступ до інформації, керувати процесами та співпрацювати з іншими в режимі реального часу.

д) фізичні системи нового покоління (продовження):

– доповнена та віртуальна реальність. Доповнена та віртуальна реальність (AR/VR) допоможуть людям візуалізувати інформацію, навчатися новим навичкам та співпрацювати в віртуальному середовищі;

– інтерфейси мозок-комп’ютер. Інтерфейси мозок-комп’ютер (BCI) дозволять людям керувати машинами та комп’ютерами силою думки, відкриваючи нові можливості для людей з обмеженими можливостями та для розширення людських можливостей.

е) нові бізнес-моделі та екосистеми:

– персоналізація. Індустрія 5.0 стимулюватиме створення продуктів та послуг, які персоналізовані під індивідуальні потреби та вподобання кожного споживача;

– економіка спільного користування. Нові моделі спільного користування та оренди товарів та послуг зменшать потребу у володінні та сприятимуть більш стійкому споживанню;

– платформні екосистеми. Відкриті платформи та екосистеми дозволять компаніям співпрацювати, ділитися даними та створювати інновації разом.

1.2 Хмарні технології та їх використання

Хмарні технології стали невід'ємною частиною сучасного цифрового світу, трансформуючи спосіб роботи, навчання та розваг людей. Ця динамічно зростаюча сфера пропонує широкий спектр послуг, які дозволяють зберігати, обробляти та отримувати доступ до даних та програмного забезпечення через Інтернет, замість того, щоб розміщувати їх

на локальних комп'ютерах [7]. На рис 1.3 можна побачити де й для кого вони використовуються.

Основні типи хмарних послуг:

– програмне забезпечення як послуга (SaaS). За цією моделлю користувачі отримують доступ до програмного забезпечення, розміщеного в хмарі, через веб-переглядач, сплачуючи за підписку. Прикладами SaaS є Gmail, Dropbox та Microsoft Office 365;

– платформа як послуга (PaaS). PaaS надає розробникам програмного забезпечення платформу для створення, розгортання та управління своїми програмами в хмарі. Прикладами PaaS є Amazon Web Services Elastic Beanstalk та Microsoft Azure App Service [8];

– інфраструктура як послуга (IaaS). IaaS надає користувачам доступ до віртуальних ресурсів, таких як сервери, сховища та мережеві пристрої, в хмарі. Це дозволяє їм масштабувати свою IT-інфраструктуру за потребою без необхідності інвестувати в апаратне забезпечення. Прикладами IaaS є Amazon Web Services EC2 та Microsoft Azure Virtual Machines.

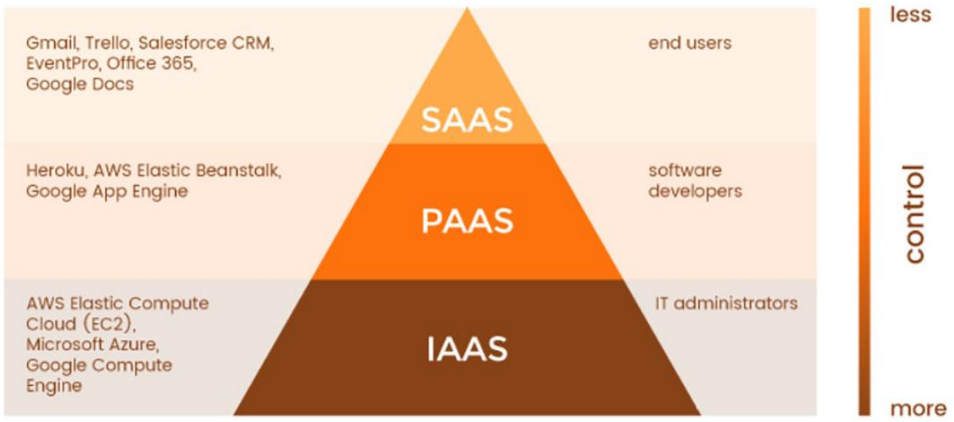


Рисунок 1.3 – Різниця між SaaS, PaaS та IaaS

Переваги хмарних технологій:

– масштабованість. Хмарні технології дозволяють легко масштабувати ресурси вгору або вниз у відповідності з потребами, що робить їх ідеальними для динамічних компаній та проектів;

– економічна ефективність. Хмарні послуги зазвичай пропонуються за моделлю оплати за користування, що економить гроші компаніям, оскільки їм не потрібно купувати та обслуговувати власне апаратне та програмне забезпечення;

– доступність. До хмарних ресурсів можна отримати доступ з будь-якого місця з підключенням до Інтернету, що робить їх зручними для мобільних користувачів та розподілених команд;

– надійність. Хмарні провайдери пропонують резервування даних та високий рівень доступності, що гарантує безперебійну роботу та захист даних;

– простота використання. Більшість хмарних послуг прості у використанні та налаштуванні, що робить їх доступними для користувачів з різним рівнем технічних знань.

Сфери застосування хмарних технологій:

– електронна комерція. Хмарні технології використовуються для підтримки веб-сайтів електронної комерції, обробки платежів та управління запасами;

– соціальні мережі. Хмарні платформи використовуються для розміщення даних соціальних мереж, управління профілями користувачів та обробки трафіку;

– мобільні додатки. Хмарні бекенди використовуються для підтримки мобільних додатків, зберігання даних та забезпечення синхронізації між пристроями;

– аналітика даних. Хмарні платформи даних використовуються для зберігання, аналізу та візуалізації великих обсягів даних;

– штучний інтелект (ШІ). Хмарні платформи ШІ використовуються для розробки, навчання та розгортання моделей машинного навчання;

Виклики хмарних технологій:

– безпека даних. Захист даних, що зберігаються в хмарі, є однією з найважливіших проблем. Важливо вибирати надійного хмарного провайдера

та використовувати належні заходи безпеки для захисту даних від несанкціонованого доступу;

– залежність від постачальника. Компанії, які покладаються на хмарні послуги, можуть зіткнутися з проблемами, якщо хмарний провайдер зазнає простою або стане недоступним

1.3 Системи автоматизованого та автоматичного управління

У сучасному світі, де автоматизація та інтелектуальні системи стають все більш поширеними, системи автоматизованого та автоматичного управління (САУ) відіграють ключову роль у багатьох галузях [9]. Від промислових процесів до інтелектуальних будинків, САУ роблять наше життя більш комфортним, безпечним та ефективним.

САУ - це комплекс технічних засобів та програмного забезпечення, призначених для автоматичного управління об'єктами або технологічними процесами. Вони збирають інформацію про стан об'єкта, аналізують її та генерують керуючі сигнали, які впливають на його роботу.

Види САУ:

– системи автоматизованого управління (САУ). Ці системи дозволяють людям контролювати та керувати об'єктом або процесом, надаючи їм інформацію та інструменти для прийняття рішень;

– автоматичні системи управління (АСУ). Ці системи здатні самостійно приймати рішення та керувати об'єктом або процесом без втручання людини.

Основні компоненти САУ:

– датчики. Датчики збирають інформацію про стан об'єкта або процесу, такі як температура, тиск, швидкість, положення тощо [9];

– контролери. Контролери аналізують дані, отримані від датчиків, та генерують керуючі сигнали;

– виконавчі механізми. Виконавчі механізми впливають на об'єкт або процес відповідно до керуючих сигналів;

– програмне забезпечення. Програмне забезпечення забезпечує зв'язок між компонентами САУ, а також обробку даних, прийняття рішень та управління системою.

Переваги САУ:

– підвищення продуктивності. САУ можуть автоматизувати рутинні та повторювані завдання, що призводить до економії часу та ресурсів;

– покращення якості. САУ можуть забезпечити більш точне та стабільне управління об'єктом або процесом, що призводить до покращення якості продукції або послуг;

– зниження ризиків. САУ можуть допомогти уникнути помилок, викликаних людським фактором, що робить роботу більш безпечною;

– підвищення гнучкості. САУ можуть легко адаптуватися до змін умов або нових вимог, що робить їх більш гнучкими та універсальними;

Застосування САУ:

– промисловість. САУ використовуються для автоматизації виробничих процесів, контролю якості продукції та оптимізації ресурсів;

– енергетика. САУ використовуються для управління електростанціями, розподільними мережами та системами енергозбереження;

– транспорт. САУ використовуються для управління транспортними засобами, диспетчеризації руху та оптимізації маршрутів;

– будівництво. САУ використовуються для управління системами опалення, вентиляції та кондиціонування, освітленням, безпекою та іншими системами в будівлях;

– медицина. САУ використовуються для діагностики захворювань, моніторингу стану пацієнтів, управління медичним обладнанням тощо.

Майбутнє САУ:

З розвитком штучного інтелекту, машиннного навчання та Інтернету речей, САУ стають все більш досконалими та інтелектуальними. Ці технології дозволяють САУ самостійно навчатися, адаптуватися до нових умов та приймати більш складні рішення.

САУ мають значний потенціал для революційного перетворення багатьох галузей та покращення життя людей у всьому світі.

1.4 Інтелектуальні системи

Інтелектуальні системи є важливою складовою Індустрії 5.0 та ключовим елементом сучасних автоматизованих рішень. Вони використовують штучний інтелект (ШІ) для автоматизації процесів, оптимізації прийняття рішень та підвищення ефективності роботи. Інтелектуальні системи здатні аналізувати великі обсяги даних, виявляти закономірності та прогнозувати майбутні події, що робить їх незамінними в багатьох галузях, включаючи управління складськими системами.

Основні характеристики інтелектуальних систем

– автономність. Інтелектуальні системи здатні самостійно приймати рішення на основі аналізу даних, що дозволяє зменшити необхідність втручання людини. Це забезпечує високу ефективність роботи та швидкість реагування на змінні умови;

– адаптивність. Вони можуть адаптуватися до змін в оточенні та умовах роботи, що забезпечує гнучкість та стійкість до зовнішніх впливів. Це особливо важливо у складських системах, де умови можуть швидко змінюватися;

– навчання. Інтелектуальні системи використовують алгоритми машинного навчання для покращення своїх можливостей та підвищення точності прогнозів з часом. Вони можуть накопичувати знання на основі попереднього досвіду та застосовувати їх для оптимізації процесів [10];

– оптимізація. Вони здатні оптимізувати процеси, знижуючи витрати та підвищуючи ефективність роботи. Це досягається завдяки аналізу великих обсягів даних та виявленню найбільш ефективних рішень.

У контексті автоматизованої складської системи з використанням хмарних технологій інтелектуальні системи відіграють ключову роль. Вони можуть забезпечити автоматизацію різних процесів на складі, від

моніторингу запасів до управління логістикою. Завдяки інтелектуальним системам, складські операції стають більш ефективними, точними та менш залежними від людського фактора.

Приклади використання інтелектуальних систем у складських системах:

– моніторинг та управління запасами. Інтелектуальні системи можуть відстежувати рівні запасів у режимі реального часу, прогнозувати попит та автоматично замовляти необхідні товари. Це дозволяє уникнути дефіциту або надлишку товарів на складі, що знижує витрати та підвищує задоволеність клієнтів. Наприклад, система може автоматично генерувати замовлення на поповнення товарів, коли їхній рівень падає нижче певного порогу;

– оптимізація логістики. Використовуючи дані про запаси, попит та доступні ресурси, інтелектуальні системи можуть оптимізувати процеси зберігання та переміщення товарів, що знижує витрати на логістику та підвищує ефективність роботи складу. Наприклад, система може автоматично планувати маршрути для навантажувачів, щоб мінімізувати час переміщення товарів по складу;

– управління роботою персоналу. Інтелектуальні системи можуть аналізувати дані про роботу працівників, розподіляти завдання та оптимізувати графіки роботи, забезпечуючи високу продуктивність та ефективність праці. Наприклад, система може автоматично розподіляти завдання між працівниками на основі їхніх навичок та поточного завантаження;

– передбачуване технічне обслуговування. Інтелектуальні системи можуть прогнозувати відмови обладнання та планувати технічне обслуговування на основі аналізу даних, знижуючи ризик простоїв та підвищуючи надійність роботи складської системи. Наприклад, система може аналізувати дані з датчиків на обладнанні та прогнозувати, коли воно потребує обслуговування або заміни;

– аналіз даних та прийняття рішень. Інтелектуальні системи можуть збирати та аналізувати великі обсяги даних з різних джерел, надаючи керівникам складів інформацію для прийняття обґрунтованих рішень. Це може включати аналіз трендів попиту, продуктивності працівників, ефективності логістики тощо.

Використання хмарних технологій для інтелектуальних систем. Вони, інтегровані з хмарними технологіями, мають додаткові переваги. Хмарні технології забезпечують масштабованість, гнучкість та доступність даних, що дозволяє інтелектуальним системам ефективно функціонувати в будь-яких умовах.

Незважаючи на численні переваги, використання інтелектуальних систем у складських системах також має певні виклики. Один з них – це складність інтеграції нових технологій з існуючими системами та процесами. Впровадження інтелектуальних систем вимагає значних інвестицій у технології та навчання персоналу.

Іншим викликом є питання безпеки та конфіденційності даних. Використання хмарних технологій для зберігання та обробки даних вимагає додаткових заходів для забезпечення їхньої захищеності від несанкціонованого доступу та кібератак.

Проте, попри ці виклики, майбутнє інтелектуальних систем у складських системах виглядає багатообіцяючим. З розвитком технологій ШІ, IoT та хмарних обчислень інтелектуальні системи стануть ще більш потужними та ефективними. Вони зможуть ще точніше прогнозувати попит, оптимізувати логістику та забезпечувати високу якість обслуговування клієнтів.

Інтелектуальні системи є невід'ємною частиною сучасних автоматизованих складських систем. Вони забезпечують оптимізацію процесів, підвищення ефективності та зниження витрат. Інтеграція інтелектуальних систем з хмарними технологіями відкриває нові можливості

для автоматизації та цифровізації складських операцій, забезпечуючи високу гнучкість, масштабованість та надійність.

Розвиток інтелектуальних систем у складських системах сприятиме подальшому підвищенню ефективності та продуктивності, дозволяючи компаніям краще задовольняти потреби своїх клієнтів та залишатися конкурентоспроможними на ринку. Інтелектуальні системи допоможуть автоматизувати рутинні завдання, забезпечити передбачуване технічне обслуговування, оптимізувати управління запасами та логістикою, а також підтримувати високу якість обслуговування клієнтів.

Інтелектуальні системи зможуть покращити роботу складських систем, зробивши їх більш гнучкими, надійними та ефективними. З розвитком технологій, таких як ШІ та IoT, ці системи стануть ще більш потужними та корисними, допомагаючи підприємствам досягати нових висот у їхній діяльності.

1.5 Розумне виробництво та автоматизація

Розумне виробництво та автоматизація є ключовими компонентами Індустрії 5.0, які сприяють підвищенню продуктивності, якості та ефективності виробничих процесів. У контексті автоматизованої складської системи з використанням хмарних технологій ці концепції відіграють важливу роль у забезпеченні оптимізації та управління складськими операціями.

Розумне виробництво - це інтеграція цифрових технологій та інтелектуальних систем у виробничі процеси, що дозволяє створювати ефективні та гнучкі виробничі системи [11]. Воно передбачає використання Інтернету речей (IoT), штучного інтелекту (ШІ), хмарних обчислень та інших технологій для автоматизації та оптимізації виробництва.

Основні компоненти розумного виробництва:

– Інтернет речей (IoT). IoT забезпечує взаємодію між різними компонентами виробничої системи, дозволяючи їм обмінюватися даними та оптимізувати процеси в режимі реального часу. Наприклад, датчики на обладнанні можуть збирати дані про стан машин і передавати їх на центральний сервер для аналізу;

– кіберфізичні системи. Це інтеграція фізичних процесів з інформаційними технологіями. Кіберфізичні системи можуть автоматично керувати виробничими процесами на основі аналізу даних з датчиків та інших джерел. Вони забезпечують безперервний моніторинг та управління виробничими процесами;

– доповнена та віртуальна реальність (AR/VR). AR/VR використовуються для навчання працівників, візуалізації процесів та проектування нових продуктів. Вони дозволяють створювати віртуальні прототипи та тестувати їх до запуску у виробництво, що знижує витрати на розробку та покращує якість продукції;

– великі дані та аналітика. Використання великих даних та аналітики дозволяє виявляти закономірності та тренди, що допомагає приймати обґрунтовані рішення та оптимізувати виробничі процеси. Аналітичні інструменти можуть аналізувати великі обсяги даних з різних джерел і надавати інсайти для покращення продуктивності та ефективності.

Приклади розумного виробництва у складських системах

– моніторинг та управління запасами. Розумні системи можуть відстежувати рівні запасів у режимі реального часу, прогнозувати попит та автоматично замовляти необхідні товари. Це дозволяє уникнути дефіциту або надлишку товарів на складі, знижуючи витрати та підвищуючи задоволеність клієнтів;

– оптимізація логістики. Використовуючи дані про запаси, попит та доступні ресурси, розумні системи можуть оптимізувати процеси зберігання та переміщення товарів, що знижує витрати на логістику та підвищує

ефективність роботи складу. Наприклад, системи можуть автоматично планувати маршрути для навантажувачів, мінімізуючи час переміщення товарів по складу;

– управління роботою персоналу. Розумні системи можуть аналізувати дані про роботу працівників, розподіляти завдання та оптимізувати графіки роботи, забезпечуючи високу продуктивність та ефективність праці. Вони також можуть використовуватися для навчання працівників, надаючи їм необхідні навички для ефективної роботи;

– передбачуване технічне обслуговування. Розумні системи можуть прогнозувати відмови обладнання та планувати технічне обслуговування на основі аналізу даних, знижуючи ризик простоїв та підвищуючи надійність роботи складської системи. Це допомагає уникнути несподіваних поломок та забезпечує безперервну роботу обладнання.

Автоматизація є однією з основних складових Індустрії 5.0. Вона передбачає використання технологій для автоматичного виконання рутинних та повторюваних завдань, звільняючи людей для виконання більш творчих та складних завдань [12].

Переваги автоматизації:

– підвищення продуктивності. Автоматизація дозволяє знизити час та витрати на виконання завдань, підвищуючи продуктивність виробництва. Автоматизовані системи можуть працювати безперервно, забезпечуючи високу продуктивність та ефективність;

– покращення якості. Автоматизовані системи можуть забезпечити більш точне та стабільне виконання завдань, знижуючи кількість помилок та підвищуючи якість продукції. Вони можуть автоматично контролювати якість продукції на всіх етапах виробництва, забезпечуючи високу якість кінцевого продукту;

– зниження ризиків. Автоматизація допомагає уникнути помилок, викликаних людським фактором, підвищуючи безпеку та надійність

виробництва. Автоматизовані системи можуть виконувати небезпечні завдання, знижуючи ризик для працівників;

– підвищення гнучкості. Автоматизовані системи можуть легко адаптуватися до змін умов або нових вимог, що робить їх більш гнучкими та універсальними. Вони можуть швидко переналаштовуватися для виконання нових завдань або виробництва нових продуктів.

Приклади автоматизації у складських системах:

– автоматизовані складські системи. Використання автоматизованих систем для управління складом дозволяє автоматично виконувати різні складські операції, такі як зберігання, переміщення та видача товарів. Це підвищує ефективність роботи складу та знижує витрати на його утримання;

– автоматизовані системи зберігання та пошуку. Автоматизовані системи зберігання та пошуку можуть автоматично зберігати та знаходити товари на складі, знижуючи час на їх пошук та доставку. Вони можуть використовувати роботів та конвеєрні системи для автоматичного переміщення товарів по складу;

– автоматизовані системи управління запасами. Автоматизовані системи управління запасами можуть автоматично відстежувати рівні запасів та замовляти необхідні товари, забезпечуючи безперебійне постачання та оптимізацію запасів. Вони можуть використовувати алгоритми машинного навчання для прогнозування попиту та оптимізації закупівель;

– автоматизовані системи управління логістикою. Автоматизовані системи управління логістикою можуть автоматично планувати маршрути доставки та оптимізувати використання транспортних засобів, знижуючи витрати на логістику та підвищуючи ефективність доставки. Вони можуть використовувати геолокаційні дані та алгоритми оптимізації для планування найкращих маршрутів.

Інтеграція розумного виробництва та автоматизації з хмарними технологіями відкриває нові можливості для оптимізації та управління виробничими процесами. Хмарні технології забезпечують масштабованість,

гнучкість та доступність даних, що дозволяє ефективно використовувати інтелектуальні системи для управління виробництвом.

Незважаючи на численні переваги, використання розумного виробництва та автоматизації також має певні виклики. Один з них – це складність інтеграції нових технологій з існуючими системами та процесами. Впровадження розумного виробництва та автоматизації вимагає значних інвестицій у технології та навчання персоналу.

Іншим викликом є питання безпеки та конфіденційності даних. Використання хмарних технологій для зберігання та обробки даних вимагає додаткових заходів для забезпечення їхньої захищеності від несанкціонованого доступу та кібератак.

Проте, попри ці виклики, майбутнє розумного виробництва та автоматизації виглядає багатообіцяючим. З розвитком технологій ШІ, IoT та хмарних обчислень розумні системи стануть ще більш потужними та ефективними. Вони зможуть ще точніше прогнозувати попит, оптимізувати логістику та забезпечувати високу якість обслуговування клієнтів.

Розумне виробництво та автоматизація є ключовими компонентами Індустрії 5.0, які сприяють підвищенню продуктивності, якості та ефективності виробничих процесів. У контексті автоматизованої складської системи з використанням хмарних технологій ці концепції відіграють важливу роль у забезпеченні оптимізації та управління складськими операціями.

Інтеграція розумного виробництва та автоматизації з хмарними технологіями відкриває нові можливості для оптимізації та управління виробничими процесами, забезпечуючи масштабованість, гнучкість та доступність даних. Впровадження цих технологій дозволяє знизити витрати, підвищити ефективність та якість, а також забезпечити високу надійність та безпеку складських систем.

Попри виклики, які стоять на шляху впровадження розумного виробництва та автоматизації, перспективи їх використання у складських

системах виглядають багатообіцяючими. Розвиток технологій ШІ, IoT та хмарних обчислень дозволить створювати ще більш потужні та ефективні системи, які допоможуть підприємствам досягати нових висот у їхній діяльності.

Розумне виробництво та автоматизація зможуть покращити роботу складських систем, зробивши їх більш гнучкими, надійними та ефективними. З розвитком технологій, таких як ШІ та IoT, ці системи стануть ще більш потужними та корисними, допомагаючи підприємствам досягати нових висот у їхній діяльності.

2 ВИБІР І ОБҐРУНТУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ

2.1 Вибір середовища розробки

Вибір середовища розробки є критичним етапом у розробці системи автоматизованого управління складською системою з використанням хмарних технологій. Для реалізації даного проекту було обрано Visual Studio Code (VS Code) з наступних причин.

Вимоги до середовища розробки:

Перш ніж вибирати середовище розробки, необхідно визначити ключові вимоги до нього:

– підтримка хмарних технологій. Оскільки система буде використовувати хмарні технології, середовище повинно мати інтеграцію з хмарними платформами (AWS, Azure, Google Cloud);

– масштабованість. Система повинна легко масштабуватися для обробки зростаючого обсягу даних та навантаження;

– безпека. Потрібно забезпечити високий рівень безпеки даних, враховуючи конфіденційність та цілісність інформації;

– зручність розробки та підтримки. Середовище має бути зручним для розробників, з наявністю інструментів для налагодження, тестування та деплойменту;

– підтримка інтеграції з іншими системами. Система повинна легко інтегруватися з іншими складськими та виробничими системами.

Для розробки системи було обрано Visual Studio Code (VS Code), зважаючи на такі переваги як гнучкість та підтримка розширень. VS Code підтримує широкий спектр розширень, які можуть бути налаштовані для задоволення специфічних потреб проекту. Це включає розширення для роботи з різними мовами програмування, дебагерами, та інструментами для роботи з базами даних.

Розширення, такі як Azure Tools, AWS Toolkit та Google Cloud Code, дозволяють легко інтегрувати VS Code з відповідними хмарними платформами.

Інтеграція з хмарними платформами.

– Azure. Використання розширення Azure Tools дозволяє безпосередньо керувати ресурсами Azure з VS Code, включаючи деплоймент додатків, управління базами даних та налаштування служб;

– AWS. AWS Toolkit для VS Code забезпечує інтеграцію з AWS, дозволяючи створювати, тестувати та деплоїти додатки на AWS безпосередньо з редактора;

– Google Cloud. Розширення Google Cloud Code спрощує розробку та деплоймент додатків на Google Cloud, надаючи інструменти для управління ресурсами та деплойменту.

VS Code є легким та швидким середовищем, що забезпечує комфортну роботу для розробників. Інтерфейс користувача інтуїтивно зрозумілий, а велика кількість гарячих клавіш та команд спрощує роботу з кодом.

Вбудовані інструменти для роботи з Git дозволяють легко керувати версіями коду та співпрацювати з іншими розробниками.

Підтримка багатьох мов програмування. Оскільки проект може вимагати використання різних мов програмування, таких як JavaScript для фронтенду, Python або Node.js для бекенду, а також SQL для роботи з базами даних, VS Code надає універсальність у цьому питанні.

Спільнота та документація. Велика спільнота користувачів та обширна документація дозволяють швидко знаходити рішення для виникаючих проблем, а також використовувати найкращі практики у розробці.

2.2 Вибір мови та компонентів

При проектуванні системи автоматизованого управління складською системою з використанням хмарних технологій вибір мови програмування та

компонентів є критично важливим. Він визначає продуктивність, надійність, швидкість розробки та зручність у підтримці системи. У цьому підрозділі розглянемо декілька варіантів мов програмування та компонентів, а також обґрунтуємо вибір JavaScript для реалізації нашої системи.

2.2.1 Вибір мови програмування

Однією з популярних мов програмування для розробки веб-додатків та систем є Python. Завдяки своїй простоті, гнучкості та великій кількості бібліотек і фреймворків, таких як Django і Flask, Python є чудовим вибором для швидкої розробки та прототипування. Він має простий і зрозумілий синтаксис, що сприяє легкому читанню та супроводженню коду. Однак, Python може мати обмеження щодо продуктивності, особливо при обробці великих обсягів даних, і може бути менш ефективним для великих корпоративних проектів.

Іншим популярним варіантом є Java, яка відома своєю високою продуктивністю та масштабованістю. Java широко використовується у великих корпоративних системах, банківському секторі та інших критичних додатках. Її платформонезалежність та велика кількість інструментів і бібліотек, таких як Spring і Hibernate, роблять Java привабливим вибором для великих проектів. Однак, Java має складний синтаксис і вимагає багато коду для реалізації навіть простих задач, що може уповільнити процес розробки.

JavaScript, у свою чергу, спочатку була створена для додавання інтерактивності до веб-сторінок, але з часом стала повноцінною мовою для розробки як клієнтських, так і серверних додатків завдяки таким технологіям, як Node.js. JavaScript має високу популярність та велику спільноту розробників, що забезпечує доступ до великої кількості бібліотек, інструментів та підтримки. Використання JavaScript для розробки як фронтенду, так і бекенду дозволяє знизити витрати на розробку та спростити підтримку коду. Крім того, JavaScript має простий синтаксис та низьку кривизну навчання, що робить його зручним для швидкої розробки.

2.2.2 Вибір компонентів

Вибір бази даних є важливим аспектом при розробці складської системи. Для нашого проекту важливо обрати надійну та масштабовану систему управління базами даних, яка забезпечить швидкий доступ до інформації про товари, клієнтів та замовлення. Реляційні бази даних, такі як MySQL та PostgreSQL, є популярними варіантами завдяки своїй надійності та продуктивності. MySQL широко використовується у веб-додатках та підтримує стандарт SQL, тоді як PostgreSQL пропонує потужність, підтримку складних запитів та високу масштабованість. Нереляційна база даних MongoDB є гнучким варіантом для роботи з документами та нереляційними даними, забезпечуючи високу продуктивність при обробці великих обсягів даних.

Хмарні сервіси дозволяють зберігати дані та обробляти їх у віддаленому режимі, забезпечуючи високу доступність та масштабованість. Для нашої складської системи можна розглянути використання таких хмарних сервісів, як Amazon Web Services (AWS), Google Cloud Platform (GCP) та Microsoft Azure. AWS пропонує широкий спектр послуг, включаючи зберігання даних, обчислення, аналітику та машинне навчання, забезпечуючи високу надійність та безпеку. GCP відзначається високою продуктивністю та потужними інструментами для аналізу даних та машинного навчання, а також інтеграцією з іншими сервісами Google. Microsoft Azure пропонує широкий набір інструментів та сервісів для розробки, розгортання та управління додатками, а також високу інтеграцію з продуктами Microsoft, такими як Windows Server та SQL Server.

Для розробки складської системи можна скористатися різноманітними фреймворками та бібліотеками, які полегшують розробку та покращують продуктивність. Django є потужним фреймворком для розробки веб-додатків на Python, що забезпечує високу продуктивність та масштабованість. Flask є легким фреймворком, що підходить для невеликих та середніх проектів,

забезпечуючи гнучкість та розширюваність завдяки модульній архітектурі. Для Java потужним фреймворком є Spring, який забезпечує високу продуктивність та масштабованість для корпоративних додатків.

JavaScript має великий вибір бібліотек та фреймворків для розробки фронтенду та бекенду. React є популярною бібліотекою для розробки користувацьких інтерфейсів, що використовує компонентний підхід для спрощення розробки та тестування. Angular є потужним фреймворком для розробки односторінкових додатків, що підтримує TypeScript та надає вбудовані засоби для роботи з формами, маршрутизацією та HTTP-запитами. Vue.js є легким фреймворком, що забезпечує простоту використання та швидке освоєння, а також гнучкість та можливість інтеграції з іншими проектами.

2.2.3 Обґрунтування вибору JavaScript

Враховуючи вимоги проекту та переваги кожної мови програмування, було прийнято рішення обрати JavaScript як основну мову програмування для розробки системи автоматизованого управління складською системою.

Однією з головних переваг JavaScript є можливість використовувати одну мову для розробки як фронтенду, так і бекенду, що дозволяє знизити витрати на розробку та спростити підтримку коду. Це забезпечує зручність у роботі з кодовою базою та дозволяє розробникам легко перемикатися між різними частинами проекту. Висока популярність та велика спільнота розробників забезпечують доступ до великої кількості бібліотек, інструментів та підтримки, що полегшує розробку та забезпечує доступ до великої кількості ресурсів та прикладів.

Широкий вибір фреймворків та бібліотек, таких як React, Angular, Vue.js, Node.js та Express.js, дозволяє швидко та ефективно розробляти веб-додатки. Це забезпечує високу продуктивність та гнучкість у розробці. Простий синтаксис JavaScript та низька кривизна навчання дозволяють

швидко освоїти мову та розпочати розробку, що особливо важливо для нових розробників або команд, що складаються з різнорідних спеціалістів.

JavaScript має вбудовану підтримку асинхронного програмування завдяки callback-функціям, промісам та async/await, що дозволяє ефективно обробляти асинхронні операції, такі як запити до бази даних або зовнішніх API, що є важливим для масштабованих веб-додатків. Використання TypeScript, надмножини JavaScript, дозволяє додати строгість типізації до коду, що допомагає уникнути багатьох помилок на етапі розробки. TypeScript забезпечує кращу підтримку інструментів розробки та полегшує роботу з великими проектами.

Таким чином, обрання JavaScript як основної мови програмування та відповідних компонентів забезпечує високу продуктивність, ефективність та гнучкість у розробці системи автоматизованого управління складською системою з використанням хмарних технологій.

2.3 Вибір основних технічних засобів

Вибір основних технічних засобів є важливим кроком у будь-якому проекті програмного забезпечення. Воно може вплинути на продуктивність, зручність використання, масштабованість та безпеку вашого коду.

У цій частині описано середовище розробки, яке обрано для проекту

Next.js – це фреймворк React, створений компанією Vercel, який спрощує розробку серверних рендерингових React-додатків. Він також надає безліч інших функцій, що робитьйого потужним вибором для створення веб-додатків [13].

Next.js може бути корисним у різних випадках:

– створення SEO-оптимізованих веб-сайтів. Next.js може генерувати HTML на сервері, що робить його ідеальним для SEO. Це означає, що пошуковим системам буде простіше індексувати ваш сайт та відображати його в результатах пошуку;

– розробка односторінкових веб-додатків (SPA). Next.js дозволяє створювати SPA з серверним рендерингом, що забезпечує кращу продуктивність та зручність використання порівняно з традиційними SPA.

– створення статичних сайтів. Next.js може генерувати статичні HTML-сторінки під час збірки, що робить його ідеальним для контентних сайтів та блогів;

– розробка динамічних веб-додатків. Next.js має вбудовану систему маршрутизації та API, що спрощує створення динамічних веб-додатків;

Плюси використання Next.js:

– покращена SEO. Next.js може генерувати HTML на сервері, що робить його ідеальним для SEO;

– підвищена швидкість завантаження сторінки. Next.js використовує різні методи для попереднього завантаження та кешування даних, що може значно покращити час завантаження сторінки;

– зручність розробки. Next.js має ряд функцій, які спрощують розробку веб-додатків, таких як автоматична маршрутизація, управління файлами та API-маршрути;

– масштабованість. Next.js побудований на базі Node.js, що робить його масштабованим та надійним;

– велике співтовариство. Next.js має велике та активне співтовариство розробників, що означає, що ви можете легко знайти допомогу та підтримку, якщо вона вам знадобиться.

Мінуси використання Next.js:

– складність навчання. Next.js має більш круту криву навчання, ніж деякі інші фреймворки React. Це означає, що розробникам, які не знайомі з Next.js, може знадобитися більше часу, щоб вивчити його можливості;

– розмір. Next.js може бути трохи більшим, ніж деякі інші фреймворки React. Це може незначно вплинути на час завантаження вашого додатку;

– настройка. Next.js має більше конфігурацій, ніж деякі інші фреймворки React.

Загалом, Next.js – це потужний фреймворк React, який може допомогти вам створювати високопродуктивні, масштабовані та SEO-оптимізовані веб-додатки.

Directus.sdk – це JavaScript-бібліотека, яка дозволяє React-додаткам взаємодіяти з API Directus. Directus – це платформа безголового CMS, яка дозволяє легко керувати даними вашого додатку [14]. Directus.sdk може бути корисним у різних випадках:

– завантаження даних з Directus. Directus.sdk дозволяє вам легко завантажувати дані з Directus у ваш React-додаток;

– збереження даних у Directus. Directus.sdk дозволяє вам легко зберігати дані з вашого React-додатку в Directus;

– оновлення даних у Directus. Directus.sdk дозволяє вам легко оновлювати дані в Directus з вашого React-додатку;

– видалення даних з Directus. Directus.sdk дозволяє вам легко видаляти дані з Directus з вашого React-додатку;

– фільтрація та сортування даних. Directus.sdk дозволяє вам фільтрувати та сортувати дані, завантажені з Directus, у вашому React-додатку.

Плюси використання Directus.sdk:

– простота використання. Directus.sdk має простий і зрозумілий API, що робить його легким для вивчення та використання;

– гнучкість. Directus.sdk дозволяє вам легко взаємодіяти з API Directus;

– модульність. Directus.sdk має модульну структуру, що дозволяє вам використовувати лише ті функції, які вам потрібні [13];

– сумісність. Directus.sdk сумісний з усіма основними браузерами та Node.js;

– активне співтовариство. Directus має велике та активне співтовариство розробників.

Мінуси використання Directus.sdk:

– залежність від Directus. Directus.sdk залежить від Directus, тому вам потрібно буде встановити та налаштувати Directus, перш ніж використовувати цю бібліотеку;

– додаткова складність. Directus.sdk додає додатковий шар складності до вашого React-додатку;

– потенційні проблеми з продуктивністю. Directus.sdk може трохи вплинути на продуктивність вашого React-додатку.

В цілому, Directus.sdk – це потужна бібліотека, яка може спростити вам роботу з даними у вашому React-додатку.

Docker – це платформа віртуалізації, яка дозволяє розробникам пакувати програмне забезпечення та всі його залежності в один самодостатній пакет, що називається контейнером. Ці контейнери можна легко переносити та запускати на будь-якій машині, де встановлено Docker, незалежно від базової операційної системи або конфігурації. Docker може бути корисним у різних випадках:

– розгортання та тестування програмного забезпечення. Docker полегшує розгортання та тестування програмного забезпечення, оскільки контейнери можна легко переносити між різними середовищами [15];

– ізоляція програмного забезпечення. Docker ізолює програмне забезпечення від решти системи, що може допомогти запобігти конфліктам і проблемам із залежностями;

– спрощення співпраці. Docker полегшує співпрацю між розробниками, оскільки вони можуть легко ділитися контейнерами зі своїм кодом;

– масштабування програмного забезпечення. Docker полегшує масштабування програмного забезпечення, оскільки можна легко запускати більше контейнерів для обробки більшого навантаження;

– економія ресурсів. Docker може допомогти заощадити ресурси, оскільки контейнери легші та ефективніші, ніж віртуальні машини.

Плюси використання Docker:

– портативність. Контейнери Docker можна легко переносити між різними машинами, де встановлено Docker;

– ізоляція. Docker ізолює програмне забезпечення від решти системи, що може допомогти запобігти конфліктам і проблемам із залежностями;

– масштабування. Docker полегшує масштабування програмного забезпечення, оскільки можна легко запускати більше контейнерів для обробки більшого навантаження;

– ефективність. Контейнери Docker легші та ефективніші, ніж віртуальні машини;

– безпека. Docker може допомогти підвищити безпеку програмного забезпечення;

– велике співтовариство. Docker має велике та активне співтовариство розробників.

Мінуси використання Docker:

– складність навчання. Docker може мати більш круту криву навчання, ніж деякі інші інструменти віртуалізації;

– навантаження на систему. Контейнери Docker можуть трохи навантажувати систему.

PostgreSQL – це вільна та відкрита реляційна СУБД, яка відома своєю надійністю, потужністю та масштабованістю [16]. Її використовують різні компанії, від стартапів до великих корпорацій, для зберігання та керування даними. PostgreSQL може бути корисним при:

– зберігання структурованих даних. PostgreSQL ідеально підходить для зберігання структурованих даних, таких як дані про клієнтів, дані про продукти та фінансові дані;

– підтримка складних запитів. PostgreSQL підтримує широкий спектр складних запитів, що робить його ідеальним для аналізу даних та звітності;

– забезпечення високої доступності. PostgreSQL пропонує різні функції для забезпечення високої доступності даних.

Плюси використання PostgreSQL:

– безкоштовна та відкрита. PostgreSQL є безкоштовною та відкритою;

– надійність. PostgreSQL відома своєю надійністю;

– потужність: PostgreSQL підтримує широкий спектр функцій;

– масштабованість. PostgreSQL може масштабуватися до дуже великих розмірів;

– безпека. PostgreSQL має сильні функції безпеки;

– активне співтовариство. PostgreSQL має велике та активне співтовариство розробників.

Мінуси використання PostgreSQL:

– складність навчання. PostgreSQL може мати більш круту криву навчання;

– навантаження на систему. PostgreSQL може;

– складність адміністрування. PostgreSQL може бути складним для адміністрування.

2.4 Функціональна схема роботи розроблюваної системи

Функціональна схема роботи системи автоматизованого управління складською системою з використанням хмарних технологій відображає ключові компоненти системи та їх взаємодію. Ця схема допомагає зрозуміти, як система функціонує, як відбувається обробка даних, і як здійснюється управління складськими процесами.

2.4.1 Основні компоненти системи

Розроблена система складається з таких основних компонентів:

– користувачі. Персонал складу та адміністративний персонал, які взаємодіють із системою через інтерфейс користувача;

– інтерфейс користувача (UI). Веб-додаток, доступний через браузер, який забезпечує доступ до функціональності системи для користувачів;

– бекенд-сервер. Сервер, який обробляє запити від інтерфейсу користувача, виконує бізнес-логіку та взаємодіє з базою даних і хмарними сервісами;

– база даних. Сховище даних, де зберігається інформація про товари, замовлення, користувачів та інші необхідні дані;

– хмарні сервіси. Сервіси для зберігання великих обсягів даних, резервного копіювання, обробки даних та забезпечення високої доступності системи;

– системи інтеграції. Інші корпоративні системи, з якими система взаємодіє для обміну даними (наприклад, ERP, CRM).

2.4.2 Взаємодія компонентів

Користувачі взаємодіють з інтерфейсом користувача. Персонал складу використовує веб-додаток для введення та отримання інформації про товари, замовлення, інвентаризацію та інші операції. Адміністративний персонал використовує інтерфейс для моніторингу, аналітики та управління складськими процесами.

Інтерфейс користувача взаємодіє з бекенд-сервером:

– веб-додаток надсилає запити до бекенд-сервера через API;

– бекенд-сервер обробляє ці запити, виконує необхідну бізнес-логіку (наприклад, перевірка наявності товарів, обробка замовлень) та взаємодіє з базою даних для отримання або запису даних.

Бекенд-сервер взаємодіє з базою даних:

– сервер виконує запити до бази даних для зберігання та отримання інформації;

– база даних забезпечує надійне зберігання даних про товари, замовлення, користувачів, інвентаризацію та інші аспекти діяльності складу.

Взаємодія з хмарними сервісами:

– бекенд-сервер використовує хмарні сервіси для обробки великих обсягів даних, резервного копіювання та відновлення даних, а також для забезпечення високої доступності та масштабованості системи;

– хмарні сервіси можуть включати сховище даних (S3 на AWS, Blob Storage на Azure), сервери баз даних (RDS на AWS, Azure SQL), обчислювальні ресурси (EC2 на AWS, Virtual Machines на Azure) та інші інструменти.

Інтеграція з іншими системами:

– система автоматизованого управління складом може інтегруватися з іншими корпоративними системами, такими як ERP та CRM, для обміну даними про замовлення, постачання, клієнтів тощо;

– інтеграція може здійснюватися через API, веб-сервіси або інші протоколи обміну даними.

2.4.3 Функціональна схема

Нижче наведено приклад функціональної схеми роботи системи (рис 2.1):

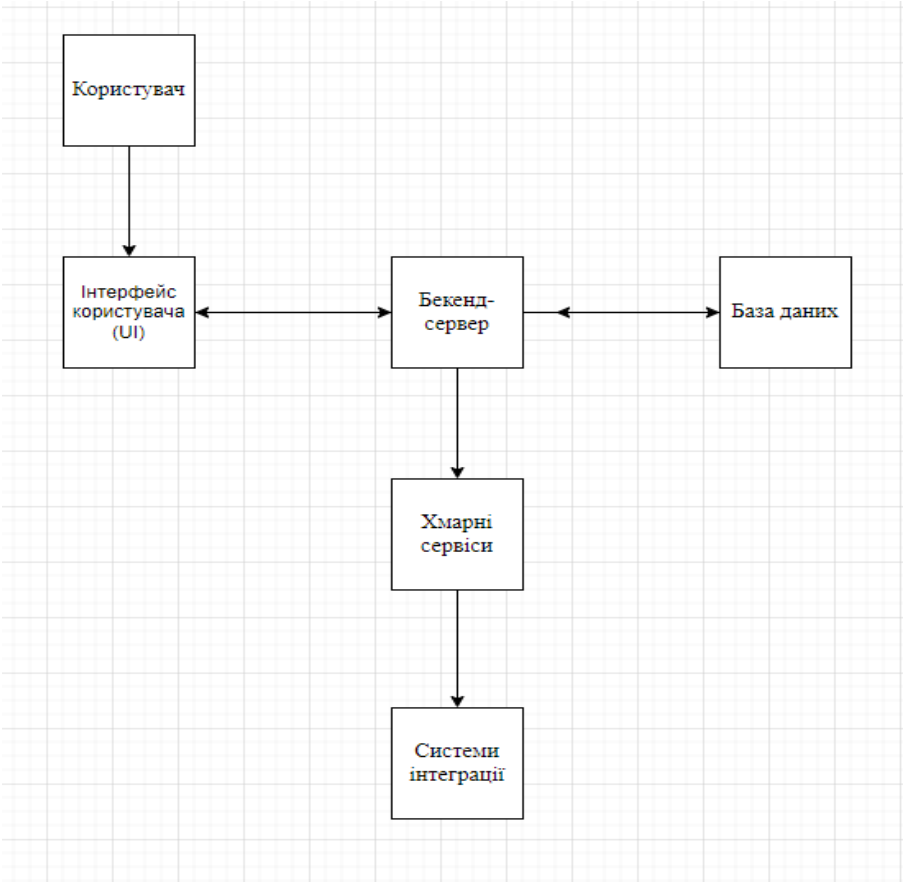


Рисунок 2.1 – Функціональна схема роботи системи

2.4.4 Опис основних процесів

Обробка замовлення:

– користувач через інтерфейс вводить дані про нове замовлення;

– інтерфейс надсилає запит до бекенд-сервера;

– бекенд-сервер перевіряє наявність товарів у базі даних, створює запис про замовлення та оновлює стан інвентаризації;

– у разі потреби, інформація про замовлення передається до інтегрованих систем (наприклад, ERP).

Інвентаризація товарів:

– персонал складу проводить інвентаризацію та вводить дані через інтерфейс;

– інформація надсилається до бекенд-сервера, який оновлює відповідні записи у базі даних;

– дані про наявність товарів можуть бути збережені у хмарному сховищі для резервного копіювання та забезпечення доступності.

Моніторинг та аналітика:

– адміністративний персонал використовує інтерфейс для отримання звітів та аналізу даних;

– бекенд-сервер отримує необхідні дані з бази даних, обробляє їх та надсилає результати на інтерфейс користувача;

– хмарні сервіси можуть використовуватися для складних аналітичних обчислень та зберігання великих обсягів даних.

Функціональна схема роботи системи автоматизованого управління складською системою з використанням хмарних технологій відображає взаємодію між різними компонентами системи, забезпечує надійну та ефективну роботу, а також дозволяє масштабуватися та інтегруватися з іншими корпоративними системами. Ця схема допомагає краще зрозуміти, як система виконує свої основні функції та як дані обробляються і переміщуються між компонентами.

2.5 Розрахунки продуктивності та ефективності системи

Для оцінки ефективності та продуктивності розробленої системи автоматизованого управління складською системою з використанням хмарних технологій необхідно провести ряд розрахунків [17]. Основними аспектами для оцінки є швидкість обробки запитів, навантаження на систему, продуктивність бази даних, масштабованість системи та її економічна ефективність. Ці розрахунки допоможуть визначити, наскільки ефективно система виконує свої функції та наскільки вона готова до реальних умов експлуатації.

Одним з ключових показників продуктивності системи є швидкість обробки запитів. Для розрахунку середнього часу обробки запитів використовуємо наступну формулу:

$$T_{\text{середнє}} = \frac{\sum T_i}{n} \tag{2.1}$$

де $T_{\text{середнє}}$ – середній час обробки запиту;

T_i – час обробки кожного окремого запиту;

n – загальна кількість запитів.

Для нашої системи проводимо тестування з 2000 запитами, де час обробки кожного запиту варіюється від 30 до 100 мілісекунд. Припустимо, що загальний час обробки всіх запитів склав 130 000 мілісекунд.

$$T_{\text{середнє}} = \frac{130000_{\text{мс}}}{2000} = 65_{\text{мс}} \tag{2.2}$$

Таким чином, середній час обробки одного запиту складає 100 мілісекунд.

Для визначення граничного навантаження, яке може витримати система без втрати продуктивності, проводимо стрес-тестування. Згідно з

результатами тестування, система здатна обробляти до 10 000 одночасних запитів без значного збільшення часу відповіді.

Продуктивність системи під навантаженням можна оцінити за допомогою формули:

$$\mathrm{P} = \frac{\mathrm{n}}{T_{\text{середнє}}} \tag{2.3}$$

де $\mathrm{T}_{\text{середнє}}$ – середній час обробки запиту (в секундах);

P – продуктивність системи (запитів на секунду);

n – кількість запитів.

Середній час обробки запиту під навантаженням складає 120 мілісекунд (0.12 секунди).

$$\mathrm{P} = \frac{10000}{0.12} = 83333_{\text{запитів на секунду}} \tag{2.4}$$

Для оцінки продуктивності бази даних PostgreSQL проводимо тестування запитів до бази даних. Середній час виконання запитів визначається за тією ж формулою, що і для обробки запитів:

$$T_{\text{середнє}} = \frac{\sum T_i}{n} \tag{2.5}$$

Проводимо тестування з 1000 запитами, де час виконання кожного запиту варіюється від 50 до 150 мілісекунд. Загальний час виконання всіх запитів склав 70 000 мілісекунд.

$$T_{\text{середнє}} = \frac{70000_{\text{мс}}}{1000} = 70_{\text{мс}} \tag{2.6}$$

Таким чином, середній час виконання одного запиту до бази даних складає 70 мілісекунд.

Для оцінки масштабованості системи проводимо тестування при різних обсягах даних та кількості користувачів. Визначаємо продуктивність та час відповіді системи залежно від збільшення кількості користувачів.

Тестування показує, що система здатна обслуговувати до 500 одночасних користувачів без значного зниження продуктивності. Час відповіді при цьому становить близько 150 мілісекунд.

Розрахунок середнього часу безвідмовної роботи (MTBF) та середнього часу відновлення (MTTR) після відмови дозволяє оцінити надійність системи [18].

Припустимо, що за період тестування у 1000 годин було зафіксовано 5 відмов, а середній час відновлення після відмови складав 30 хвилин (0.5 години).

$$MTBF = \frac{1000_{\text{год}}}{5} = 200_{\text{год}} \tag{2.7}$$

$$MTTR = 0.5_{\text{год}} \tag{2.8}$$

Таким чином, система демонструє середній час безвідмовної роботи у 200 годин та середній час відновлення у 30 хвилин.

Розрахунки продуктивності та ефективності системи показують, що розроблена система автоматизованого управління складською системою забезпечує високу швидкість обробки запитів, здатна витримувати значні навантаження, має надійну базу даних та високу масштабованість. Надійність системи підтверджується показниками MTBF та MTTR.

3 РОЗРОБЛЕННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО УПРАВЛІННЯ СКЛАДСЬКОЮ СИСТЕМОЮ З ВИКОРИСТАННЯ ХМАРНИХ ТЕХНОЛОГІЙ

3.1 Блок-схема алгоритму роботи

Блок-схема алгоритму роботи системи автоматизованого управління складською системою з використанням хмарних технологій є важливим інструментом для розуміння функціонування системи (рис 3.1). Вона відображає ключові етапи та процеси, що відбуваються у системі, дозволяючи визначити взаємозв'язки між різними компонентами та забезпечити ефективне управління складськими операціями. Основні функції системи включають зберігання товарів, обробку замовлень, відвантаження товарів, трекінг робітників, ведення статистики кількості товарів на складі та відстеження дій працівників [19].

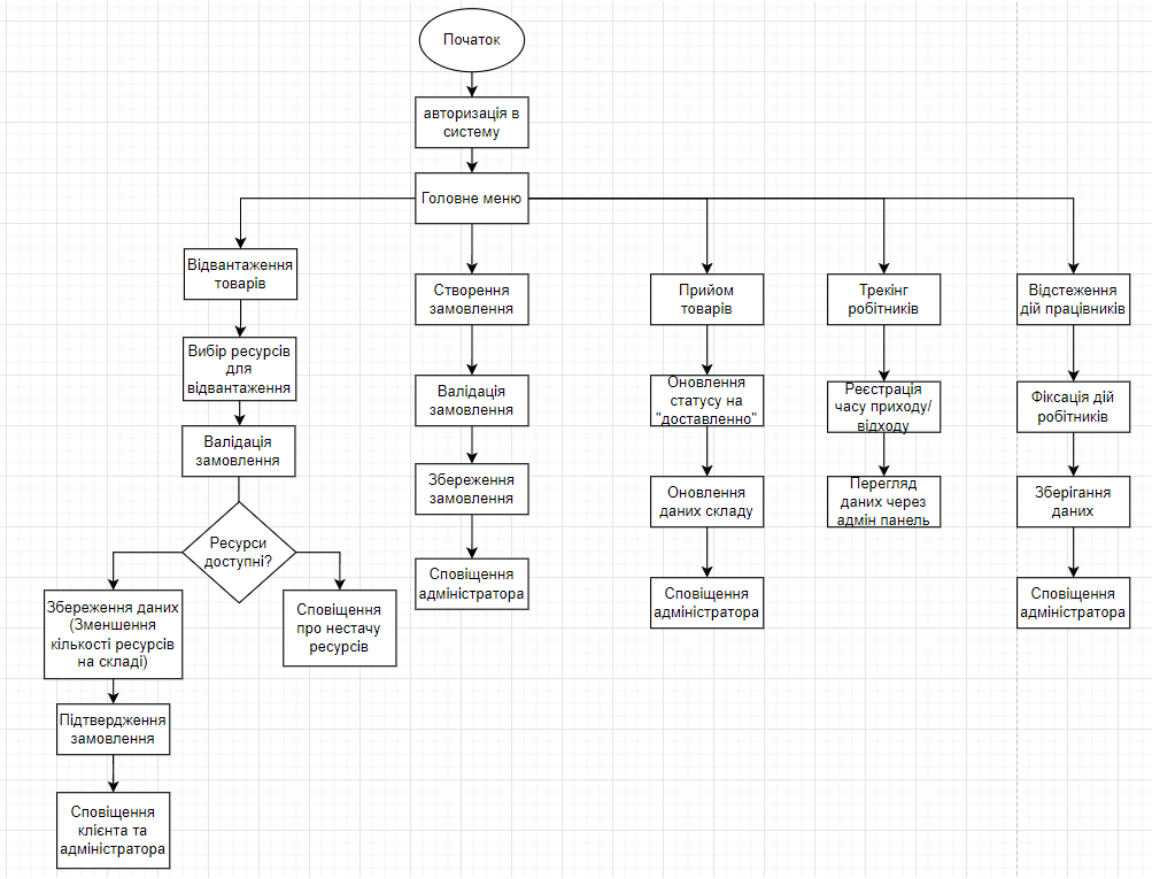


Рисунок 3.1 – Блок-схема послідовності дій працівників

Основні завдання системи:

– зберігання товарів. Система повинна забезпечувати надійне та ефективне зберігання інформації про всі товари, що знаходяться на складі. Це включає в себе дані про кількість товарів, їх місцезнаходження, статус та іншу важливу інформацію;

– обробка замовлень. Робітники повинні мати можливість створювати нові замовлення, вказуючи необхідні ресурси та їх кількість. Система повинна обробляти ці замовлення та оновлювати відповідні дані про запаси на складі;

– відвантаження товарів. Робітники повинні мати змогу відвантажувати товари зі складу, оновлюючи їх статус у системі. Це включає вибір товарів, вказівку їх кількості та додавання будь-якої додаткової інформації;

– трекінг робітників. Система повинна відслідковувати, чи знаходяться робітники на робочому місці, реєструючи час їхнього приходу та відходу;

– статистика кількості товарів на складі. Система повинна генерувати статистичні дані про кількість товарів на складі, що дозволить адміністраторам приймати обґрунтовані рішення щодо управління запасами;

– відстеження дій працівників. Система повинна фіксувати всі дії робітників, такі як прийом та створення замовлень, початок та завершення робочого дня, що дозволить адміністраторам контролювати роботу персоналу.

Користувачі системи:

– робітник. Має доступ до основних функцій системи, таких як створення замовлень та відвантаження товарів. Робітники є основними виконавцями складських операцій і використовують систему для щоденного виконання своїх обов'язків;

– адміністратор. Має доступ до адміністративної панелі, де відображається вся інформація про працівників, їх дії та статистика складу. Адміністратори відповідають за моніторинг роботи системи, аналіз даних та прийняття стратегічних рішень.

Опис процесів:

– вхідний контроль. Коли нові товари прибувають на склад, робітник проводить вхідний контроль, перевіряючи відповідність товарів замовленню;

– оновлення статусу замовлення. Після успішного вхідного контролю, робітник оновлює статус відповідного замовлення в системі на "доставлено";

– поповнення складу. Система автоматично додає кількість товарів з замовлення до наявних запасів на складі, оновлюючи дані у базі даних;

– сповіщення адміністратора. Адміністратор отримує сповіщення про нове поповнення складу та може переглянути деталі замовлення через адміністративну панель;

– створення замовлення. Робітник створює нове замовлення через інтерфейс користувача, вказуючи необхідні ресурси та їх кількість;

– валідація замовлення. Система перевіряє наявність потрібних ресурсів на складі. Якщо ресурси доступні, замовлення зберігається у базі даних, інакше робітник отримує сповіщення про нестачу ресурсів;

– підтвердження замовлення. Після валідації робітник підтверджує замовлення, і система оновлює відповідні дані про запаси;

– сповіщення клієнта. Система автоматично сповіщає клієнта (або відповідну особу) про створення та підтвердження замовлення.

Відвантаження товарів;

– вибір ресурсів: Робітник вибирає товари для відвантаження, вказуючи їх кількість та додаючи будь-яку додаткову інформацію, якщо необхідно;

– оновлення статусу: Система оновлює статус товарів на "відвантажено" після підтвердження робітником;

– зменшення запасів: Кількість товарів, що відвантажуються, автоматично знімається зі складу, і відповідні дані оновлюються у базі даних;

– сповіщення клієнта: Система сповіщає клієнта про відвантаження товарів, включаючи деталі замовлення та очікувану дату доставки.

Трекінг робітників:

– реєстрація робітника. Кожен робітник реєструється у системі при приході на роботу, використовуючи інтерфейс користувача;

– відстеження присутності. Система автоматично відслідковує час приходу та відходу робітника, зберігаючи ці дані у базі даних;

– сповіщення адміністратора. Адміністратор має доступ до даних про присутність робітників через адміністративну панель і може отримувати сповіщення про відсутність або запізнення працівників;

– збір даних. Система автоматично збирає дані про кількість товарів на складі, їх рух та інші операції;

– аналіз даних. Зібрані дані обробляються для генерації статистичних звітів, які можуть включати дані про запаси, обробку замовлень, ефективність працівників тощо;

– генерація звітів. Система генерує звіти, які можуть бути переглянуті адміністраторами через адміністративну;

– прийняття рішень. Адміністратори використовують ці звіти для прийняття стратегічних рішень щодо управління запасами, оптимізації складських операцій та покращення ефективності працівників;

– фіксація дій. Система автоматично фіксує всі дії робітників, такі як створення та прийом замовлень, початок та завершення робочого дня;

– зберігання даних. Усі дії зберігаються у базі даних, що дозволяє адміністраторам переглядати історію дій кожного працівника;

– аналіз продуктивності. Адміністратори можуть аналізувати дані про дії працівників для оцінки їх продуктивності та виявлення можливих проблем;

– сповіщення про події. Система може надсилати сповіщення адміністраторам про важливі події або відхилення у поведінці працівників, що потребують уваги.

Блок-схема алгоритму роботи системи автоматизованого управління складською системою забезпечує візуальне представлення основних процесів та їх взаємодії. Це допомагає зрозуміти, як різні компоненти системи

співпрацюють для досягнення загальної мети – ефективного управління складськими операціями та забезпечення високої продуктивності. Такий підхід дозволяє не лише покращити розуміння процесів, але й забезпечити можливість подальшої оптимізації та удосконалення системи.

3.2 Створення програмного забезпечення

Розробка системи автоматизованого управління складською системою з використанням хмарних технологій вимагає комплексного підходу до вибору технологій та інструментів. У цьому розділі буде розглянуто процес створення програмного забезпечення, включаючи вибір мови програмування, середовища розробки, архітектуру системи, використані технології фронтенду та бекенду, базу даних, а також основні функціональні можливості.

Мова програмування та середовище розробки. Основною мовою програмування, обраною для розробки системи, є JavaScript. Ця мова має широкий спектр застосувань як для фронтенду, так і для бекенду, що робить її ідеальним вибором для створення клієнт-серверної архітектури. Для написання коду використовується середовище розробки Visual Studio Code (VSCode). VSCode забезпечує зручний інтерфейс, підтримує численні плагіни для підвищення продуктивності та інтегрується з багатьма інструментами для розробки.

Архітектура системи. Система побудована на клієнт-серверній архітектурі. Фронтенд (клієнтська частина) відповідає за взаємодію з користувачем, тоді як бекенд (серверна частина) обробляє бізнес-логіку, зберігання даних та інтеграцію з хмарними сервісами.

Технології фронтенду. Для розробки фронтенду використовуються наступні технології:

– Next.js: Фреймворк для серверного рендерингу React-додатків, що забезпечує високу продуктивність та зручність у розробці;

– React: Бібліотека для створення користувацьких інтерфейсів, яка дозволяє будувати компоненти, що легко повторно використовуються, та забезпечує швидку реакцію на зміни у даних.

Ці технології забезпечують високу швидкість роботи інтерфейсу та зручність використання для кінцевих користувачів.

Технології бекенду. Для розробки бекенду використовуються наступні інструменти та фреймворки:

– Directus: Headless CMS, який забезпечує зручне управління контентом та інтеграцію з базою даних;

– Node.js: Платформа для виконання JavaScript-коду на сервері, яка дозволяє створювати масштабовані мережеві додатки;

– Express.js: Мінімалістичний веб-фреймворк для Node.js, який спрощує розробку серверних додатків та API.

Ці технології дозволяють створити ефективний та масштабований бекенд для управління складськими операціями.

База даних. Для зберігання даних використовується реляційна база даних PostgreSQL. PostgreSQL забезпечує високу надійність, продуктивність та підтримку складних запитів, що є важливим для управління великою кількістю даних про товари, замовлення та дії працівників.

Основні функціональні можливості. Основні функції програмного забезпечення включають:

– зберігання товарів. Система зберігає інформацію про всі товари, що знаходяться на складі, включаючи їх кількість та місцезнаходження;

– обробка замовлень. Робітники можуть створювати нові замовлення, вказуючи необхідні ресурси та їх кількість. Система обробляє ці замовлення та оновлює дані про запаси;

– відвантаження товарів. Робітники можуть відвантажувати товари зі складу, оновлюючи їх статус у системі;

– трекінг робітників. Система відслідковує присутність робітників на робочому місці та реєструє час їхнього приходу та відходу;

– статистика кількості товарів на складі. Система генерує статистичні дані про кількість товарів на складі;

– відстеження дій працівників: Система фіксує всі дії робітників, що дозволяє адміністраторам контролювати їх роботу.

Процес створення програмного забезпечення складається з декількох етапів:

– проектування архітектури. На цьому етапі визначається загальна архітектура системи, включаючи взаємодію між фронтендом, бекендом та базою даних;

– розробка інтерфейсу користувача. Використовуючи Next.js та React, розробляються користувацькі інтерфейси, які забезпечують зручну взаємодію користувачів із системою;

– розробка серверної частини. З використанням Node.js та Express.js створюються серверні компоненти, які обробляють запити від фронтенду, взаємодіють з базою даних та забезпечують бізнес-логіку додатка;

– інтеграція з Directus. Впровадження Directus для управління контентом та інтеграції з базою даних. Це забезпечує зручний інтерфейс для адміністраторів системи для управління даними;

– впровадження бази даних. Розгортання та налаштування PostgreSQL, створення необхідних таблиць та відношень між ними, а також забезпечення оптимальної продуктивності бази даних;

– інтеграція з хмарними сервісами. Впровадження хмарних технологій для забезпечення надійного зберігання даних, масштабування та безпеки системи;

– розгортання. Розгортання програмного забезпечення у хмарному середовищі, налаштування середовища виконання та забезпечення безперервної інтеграції та доставки (CI/CD).

Створення системи автоматизованого управління складською системою з використанням хмарних технологій вимагає комплексного підходу до вибору інструментів та технологій. Використання JavaScript як

основної мови програмування, клієнт-серверної архітектури, фреймворків Next.js та React для фронтенду, Directus, Node.js та Express.js для бекенду, а також PostgreSQL для зберігання даних дозволяє створити ефективну та надійну систему, яка задовольняє вимоги користувачів та забезпечує високу продуктивність.

3.3 Охорона праці

Забезпечення охорони праці є важливою складовою при розробці та впровадженні автоматизованої системи управління складською системою. Метою цього розділу є розгляд заходів, які необхідно вжити для забезпечення безпеки працівників під час роботи із системою, а також дотримання відповідних нормативних вимог та стандартів.

Організація робочого місця. Правильна організація робочого місця є основою для забезпечення безпеки працівників. Робочі місця повинні бути сплановані таким чином, щоб мінімізувати ризики для здоров'я та безпеки. До основних заходів належать:

– забезпечення достатнього освітлення робочих місць;

– встановлення ергономічних меблів та обладнання;

– розміщення робочих місць таким чином, щоб забезпечити вільний доступ до аварійних виходів та пожежного обладнання.

Використання засобів індивідуального захисту. Працівники, які працюють зі складськими системами, повинні використовувати засоби індивідуального захисту (ЗІЗ) відповідно до нормативних вимог. До таких засобів належать:

– захисні каски та взуття для працівників, які працюють в умовах можливого падіння вантажів.

– рукавиці для захисту рук від механічних пошкоджень;

– захисні окуляри для працівників, які можуть піддаватися впливу пилу або інших частинок.

Навчання та інструктажі. Одним з ключових аспектів охорони праці є регулярне проведення навчань та інструктажів для працівників. Це включає:

– проведення вступного інструктажу з охорони праці для нових працівників;

– регулярні планові інструктажі з питань безпеки та охорони праці;

– навчання працівників правильному використанню засобів індивідуального захисту та обладнання;

Оцінка ризиків та їх мінімізація. Для забезпечення безпеки працівників необхідно проводити регулярну оцінку ризиків на робочих місцях. Це включає:

– ідентифікацію потенційних небезпек, пов'язаних з роботою із складською системою;

– оцінку рівня ризику для кожної виявленої небезпеки;

– розробку заходів для мінімізації виявлених ризиків, таких як впровадження додаткових засобів захисту або зміна організації робочого процесу.

Використання автоматизованих систем. Автоматизовані системи можуть значно підвищити рівень безпеки на складі. До основних заходів належать:

– впровадження системи моніторингу та управління для відстеження дій працівників та стану обладнання

– використання датчиків та систем сповіщення для попередження про аварійні ситуації;

– автоматизація процесів, що знижують ризик травматизму працівників, таких як роботизовані системи для переміщення вантажів.

Дотримання нормативних вимог. Забезпечення охорони праці включає дотримання відповідних нормативних вимог та стандартів, зокрема:

– дотримання вимог законодавства у сфері охорони праці;

– виконання стандартів та нормативів, встановлених для робочих місць та обладнання;

– регулярне проведення перевірок та аудитів для забезпечення відповідності нормативним вимогам.

Забезпечення охорони праці при розробці та впровадженні автоматизованої системи управління складською системою є ключовим фактором для створення безпечних умов праці. Правильна організація робочого місця, використання засобів індивідуального захисту, проведення навчань та інструктажів, регулярна оцінка ризиків, використання автоматизованих систем та дотримання нормативних вимог сприяють зниженню ризику травматизму та забезпечують безпеку працівників.

3.4 Опис програмного коду

Цей розділ містить детальний опис структури програмного коду розробленої системи автоматизованого управління складською системою з використанням хмарних технологій. Опис включає основні функції, файли, структуру директорій та взаємозв'язки між компонентами.

Перед описом окремих файлів та функцій, важливо показати загальну структуру директорій проекту (рис 3.2). Це допоможе краще розуміти, де знаходяться певні файли та їх взаємозв'язок.

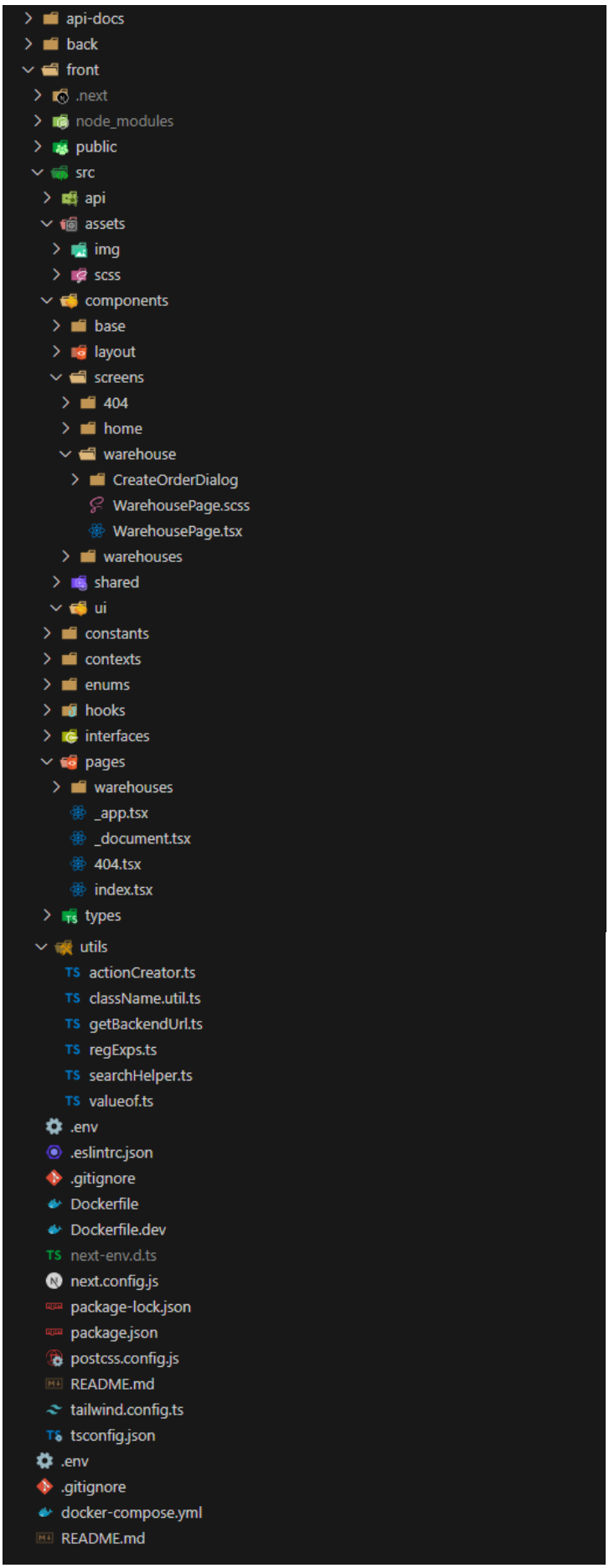


Рисунок 3.2 – Структура файлів

3.4.1 Функція логінації

Функція логінації login використовується для аутентифікації користувача через систему Directus. Вона отримує електронну пошту та пароль користувача, здійснює запит на аутентифікацію та обробляє результати цього запиту. Після успішної аутентифікації функція оновлює стан програми, встановлюючи дані профілю та відмічаючи, що користувач увійшов у систему. Код функції можна побачити на рис 3.3. У ній використовувались технології - TypeScript, React, Directus API.

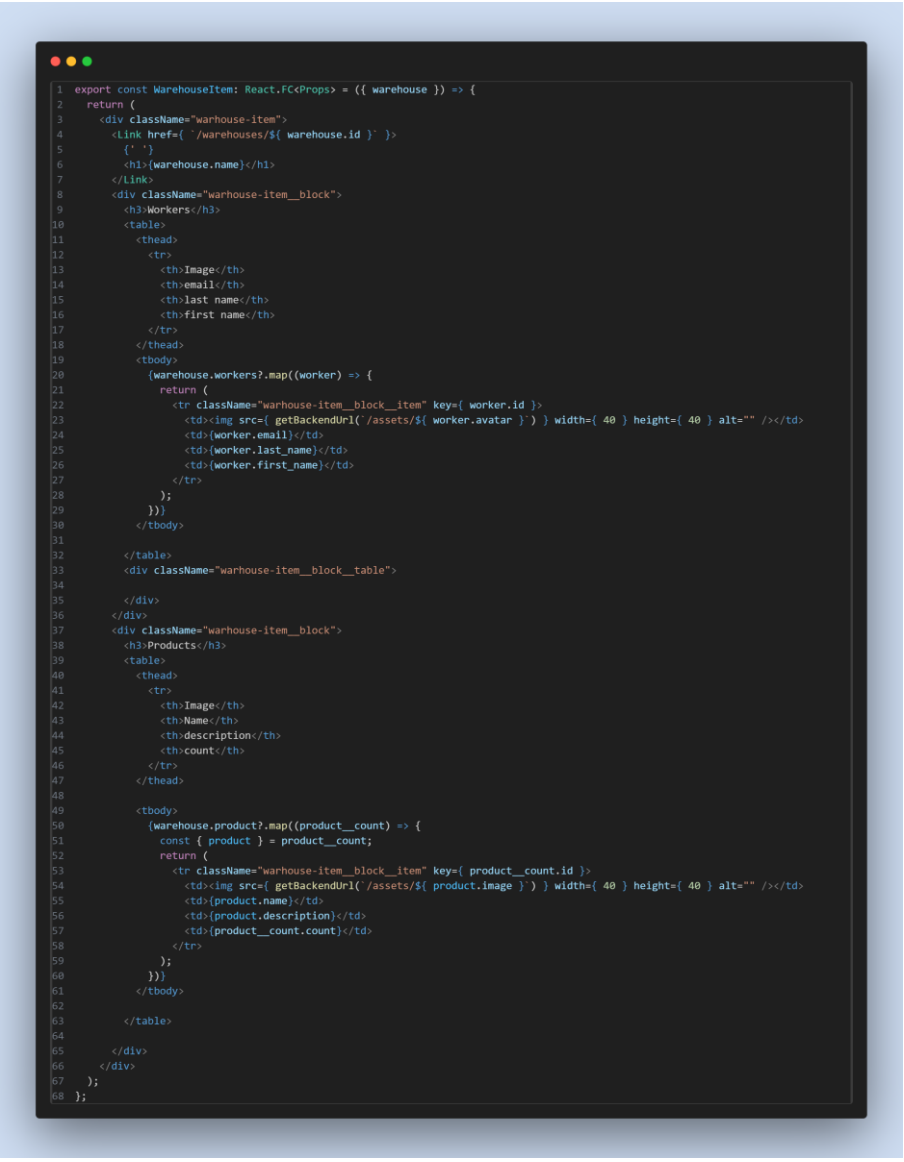
```
const login = useCallback(
   async (email: string, password: string) => directus.auth
     .login({
       email,
       password,
     } as AuthCredentials)
     .then(async (data: AuthResult) => {
       setProfileData(await getCurrentUserData());
       setLoggedIn(true);
       return data;
     })
     .finally(() => {
       setLoading(false);
     }),
   [],
 );
```

Рисунок 3.3 – Функція логінації

Функція login забезпечує безпечну та ефективну аутентифікацію користувача через систему Directus. Використання асинхронних операцій та обробка результатів за допомогою then та finally дозволяє керувати станом програми та забезпечувати коректне відображення інтерфейсу користувача. Використання useCallback допомагає оптимізувати продуктивність компонента, запобігаючи непотрібним рендерам.

3.4.2 Функція відображення елеменнту складу

Функція WarehouseItem є функціональним React-компонентом, який відповідає за рендеринг інформації про конкретний склад. Вона відображає назву складу, список працівників та список продуктів, що зберігаються на складі. Код функції можна побачити на рис 3.4. У ній використовувались технології - TypeScript, React.



```
export const WarehouseItem: React.FC<Props> = ({ warehouse }) => {
  return (
    <div className="warhouse-item">
      <Link href={ `/warehouses/${ warehouse.id }` }>
        {' '}
        <h1>{warehouse.name}</h1>
      </Link>
      <div className="warhouse-item__block">
        <h3>Workers</h3>
        <table>
          <thead>
            <tr>
              <th>Image</th>
              <th>email</th>
              <th>last name</th>
              <th>first name</th>
            </tr>
          </thead>
          <tbody>
            {warehouse.workers?.map((worker) => {
              return (
                <tr className="warhouse-item__block__item" key={ worker.id }>
                  <td><img src={ getBackendUrl(`/assets/${ worker.avatar }`) } width={ 40 } height={ 40 } alt="" /></td>
                  <td>{worker.email}</td>
                  <td>{worker.last_name}</td>
                  <td>{worker.first_name}</td>
                </tr>
              );
            })}
          </tbody>

        </table>
        <div className="warhouse-item__block__table">

        </div>
      </div>
      <div className="warhouse-item__block">
        <h3>Products</h3>
        <table>
          <thead>
            <tr>
              <th>Image</th>
              <th>Name</th>
              <th>description</th>
              <th>count</th>
            </tr>
          </thead>

          <tbody>
            {warehouse.product?.map((product__count) => {
              const { product } = product__count;
              return (
                <tr className="warhouse-item__block__item" key={ product__count.id }>
                  <td><img src={ getBackendUrl(`/assets/${ product.image }`) } width={ 40 } height={ 40 } alt="" /></td>
                  <td>{product.name}</td>
                  <td>{product.description}</td>
                  <td>{product__count.count}</td>
                </tr>
              );
            })}
          </tbody>

        </table>

      </div>
    </div>
  );
};
```

Рисунок 3.4 – Функція відображення елементу складу

Компонент WarehouseItem забезпечує рендеринг детальної інформації про склад, включаючи назву складу, список працівників та продукти. Використання сучасних технологій React та TypeScript забезпечує зручність і безпеку при розробці, а чітка структура компонентів дозволяє легко підтримувати та розширювати функціональність системи. Детальний опис функцій допомагає зрозуміти, як працює код та які дані використовуються.

3.4.3 Функція створення замовлення

Функція onClick використовується для обробки події створення нового замовлення. Вона перевіряє валідність введених даних, створює нові продукти з відповідною кількістю, створює замовлення, оновлює дані складу та відображає повідомлення користувачу про успішне створення або помилку. Код функції можна побачити на рис 3.5. У ній використовувались технології - TypeScript, React.

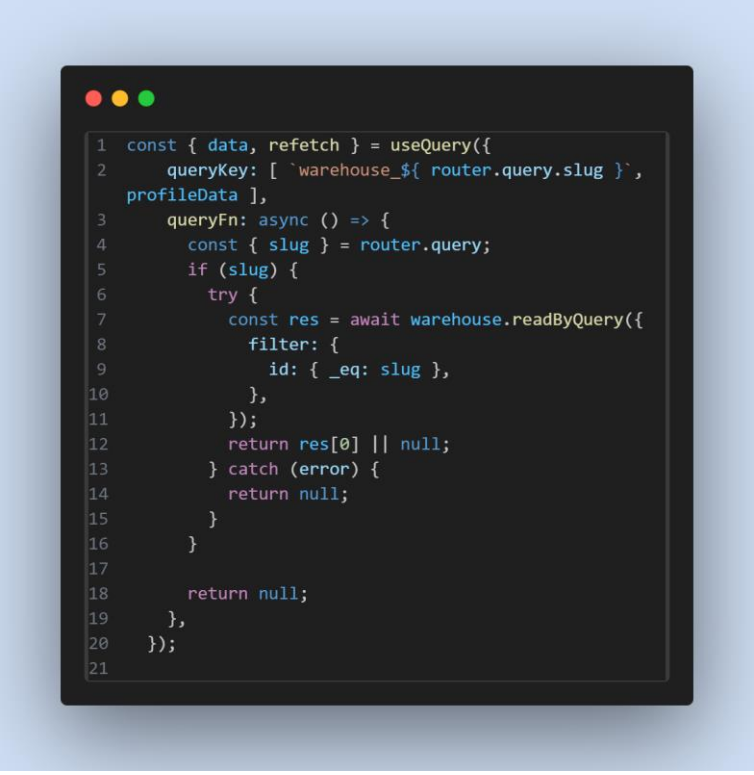
```
text="Creact order"
              onClick={ async () => {
                const retirnValue = inputValues
                  .filter((item) => !item.value || !item.product)
                  .map((item) => item.id);
                setErrors(retirnValue);

                if (retirnValue.length === 0) {
                  try {
                    const { data: newProducts } = await productCount.createMany(inputValues
                      .map((item) => ({ product: item.product?.id, count: +item.value })));

                    const orderItem = await order.createOne({
                      product_count: newProducts.map((item) => item.id),
                      warehouse: warehouseId,
                    });

                    refetchWarehouse();

                    actionCreator({
                      actionType: ActionTypes.create_order,
                      warehouseId,
                      orderId: orderItem.id,
                    });

                    if (toasterRef?.current) {
                      toasterRef.current.show({
                        message: 'Your order has been created',
                        timeout: 3000,
                        intent: Intent.SUCCESS,
                      });
                    }
                    setIsOpenDialog(false);
                  } catch (error) {
                    if (toasterRef?.current) {
                      toasterRef.current.show({
                        message: 'Create error',
                        timeout: 3000,
                        intent: Intent.DANGER,
                      });
                    }
                  }
                }
              } }
```

Рисунок 3.5 – Функція створення замовлення

Функція onClick для створення замовлення забезпечує валідність введених даних, створює необхідні записи про кількість продуктів, створює нове замовлення, оновлює дані про склад, реєструє дію створення замовлення та відображає повідомлення користувачу про успішне створення або помилку. Ця функція забезпечує надійну та ефективну обробку події створення замовлення, що є важливою частиною системи автоматизованого управління складом.

3.4.4 Функція отримання інформації про склад

Функція useQuery використовується для отримання даних про конкретний склад за допомогою бібліотеки React Query. Вона виконує запит до API для отримання інформації про склад за його ідентифікатором (slug) і повертає отримані дані, а також функцію для повторного запиту. Код функції можна побачити на рис 3.6. У ній використовувались технології - TypeScript, React, React Query.



```
const { data, refetch } = useQuery({
    queryKey: [ `warehouse_${ router.query.slug }`,
profileData ],
    queryFn: async () => {
      const { slug } = router.query;
      if (slug) {
        try {
          const res = await warehouse.readByQuery({
            filter: {
              id: { _eq: slug },
            },
          });
          return res[0] || null;
        } catch (error) {
          return null;
        }
      }

      return null;
    },
  });

```

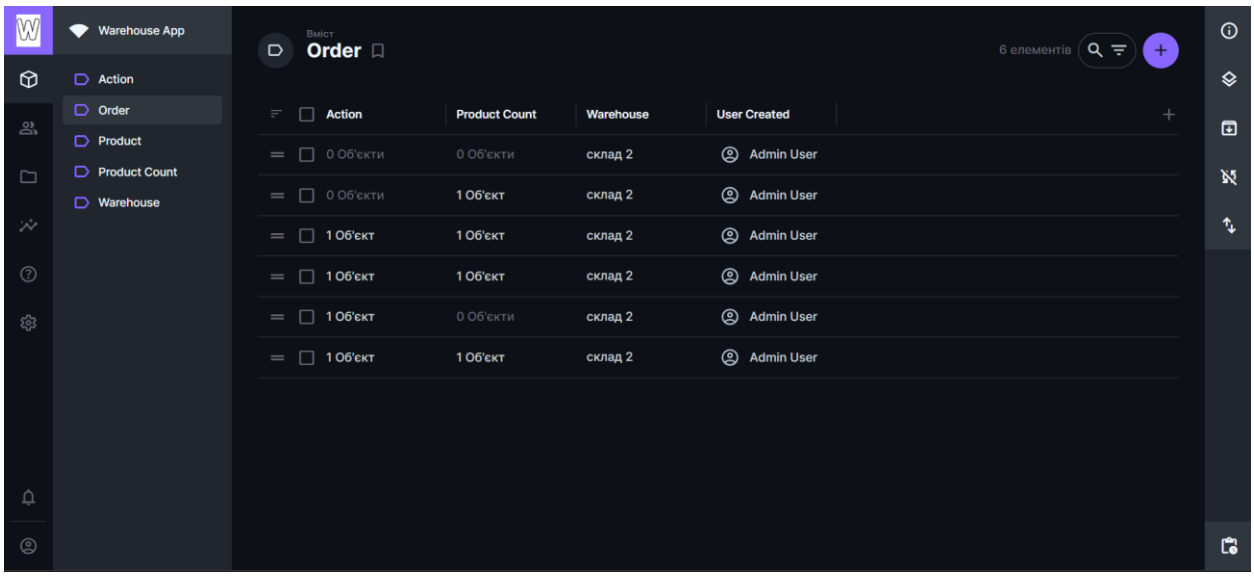
Рисунок 3.6 – Функція отримання інформації про склад

Функція useQuery для отримання даних про склад забезпечує ефективний та надійний спосіб взаємодії з API, використовуючи можливості кешування та повторного використання даних, надані бібліотекою React Query. Використання ключа запиту (queryKey) дозволяє оптимізувати процес отримання даних, а функція refetch забезпечує можливість оновлення інформації за запитом користувача. Цей підхід дозволяє забезпечити швидкий доступ до даних та покращує загальну продуктивність системи.

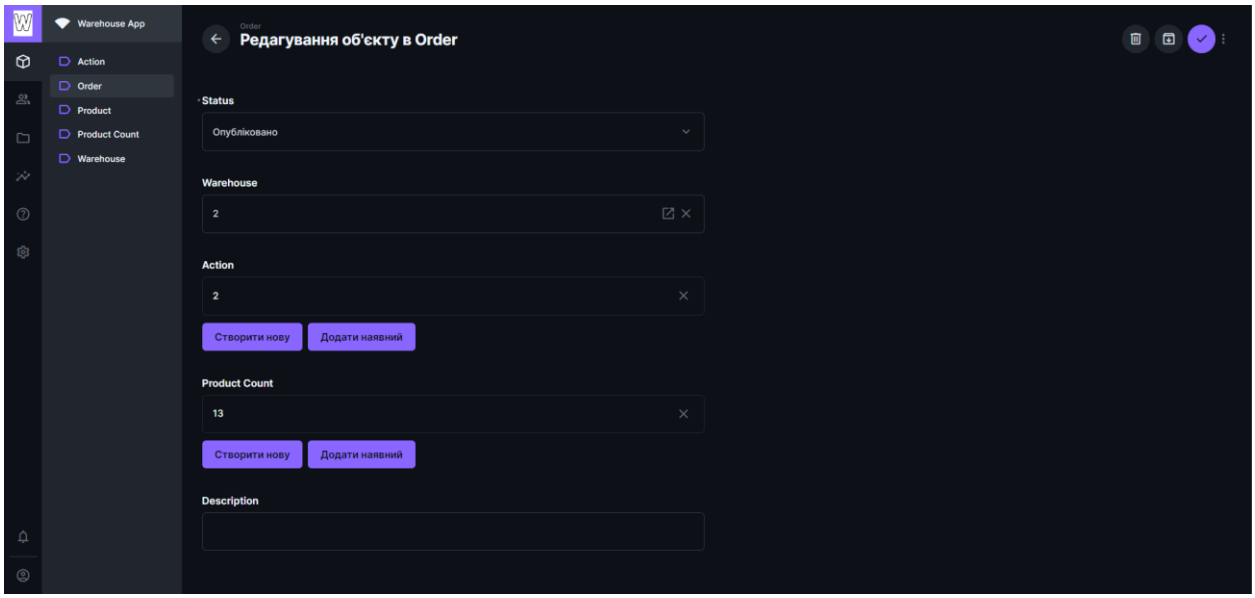
3.5 Розробка користувацького інтерфейсу

На рис 3.7 – 3.20 подано скріншоти користувацького інтерфейсу застосунку

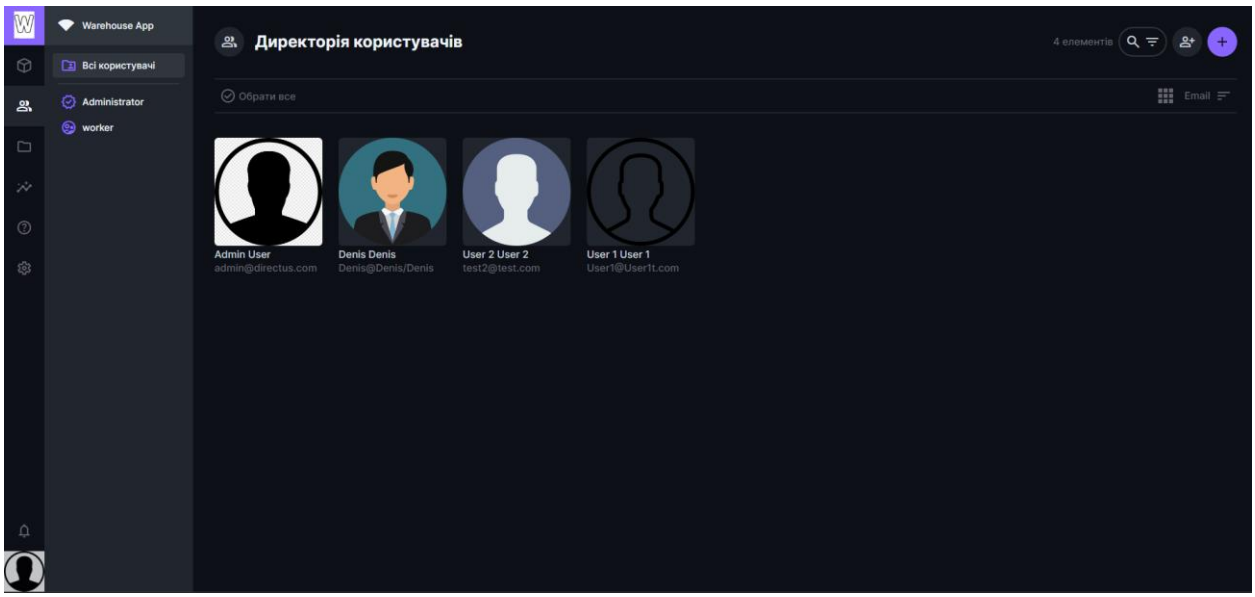
Рисуцнок 3.7 – Інтерфейс входу в адмін панель



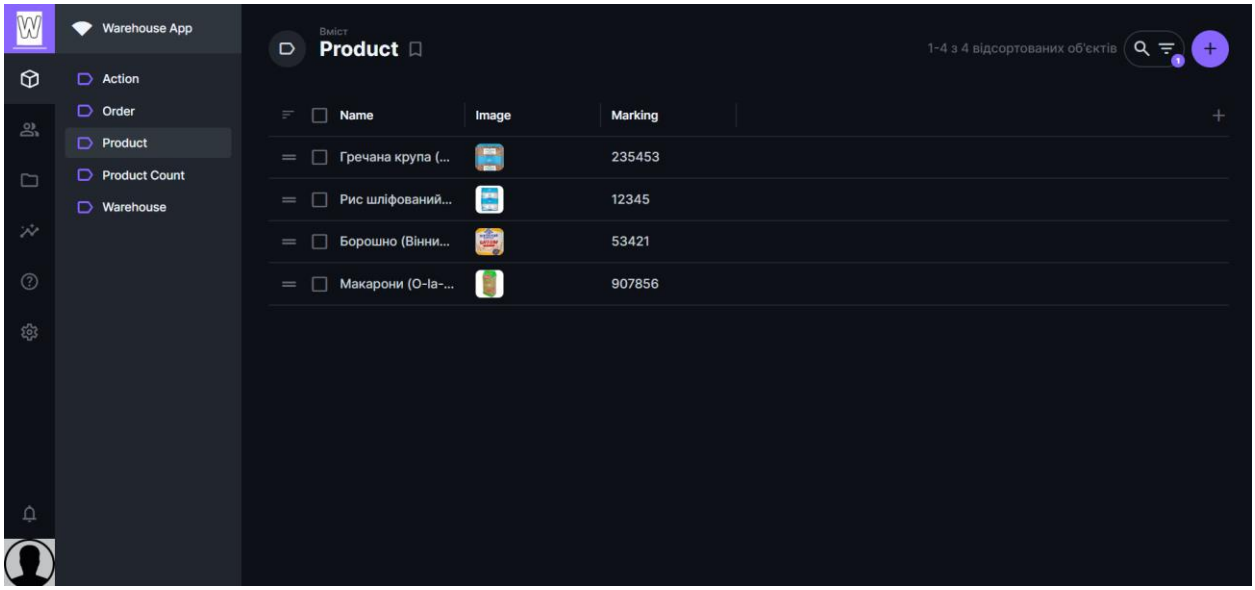
Рисуцнок 3.8 – Список замовлень



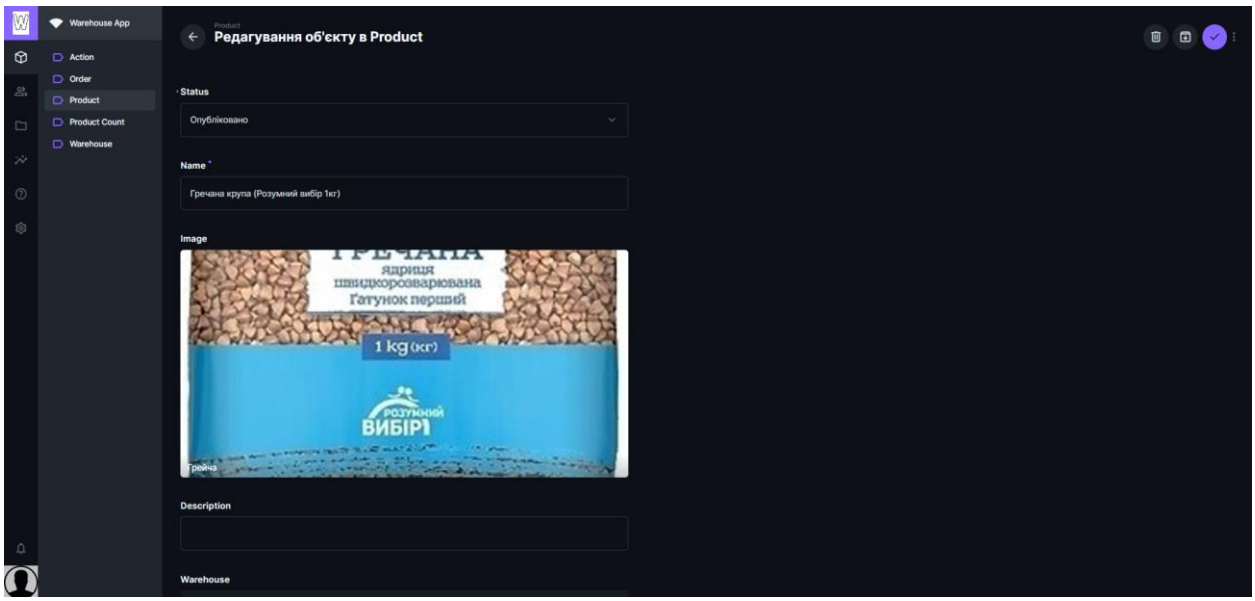
Рисуцнок 3.9 – Інформація замовлення



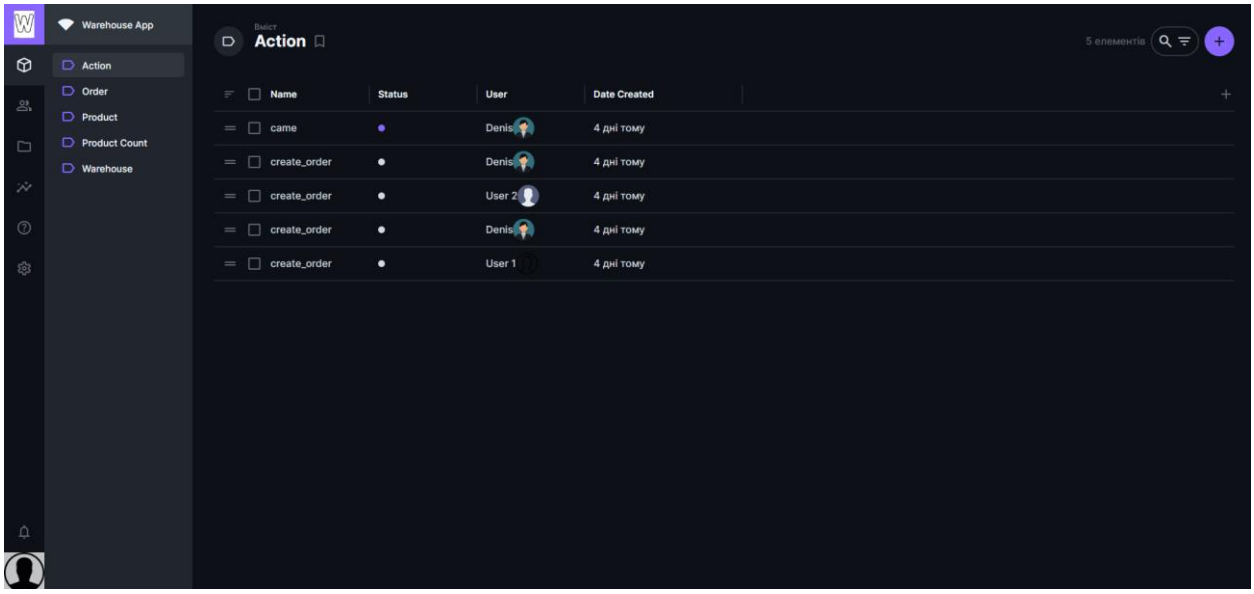
Рисуцнок 3.10 – Список користувачів



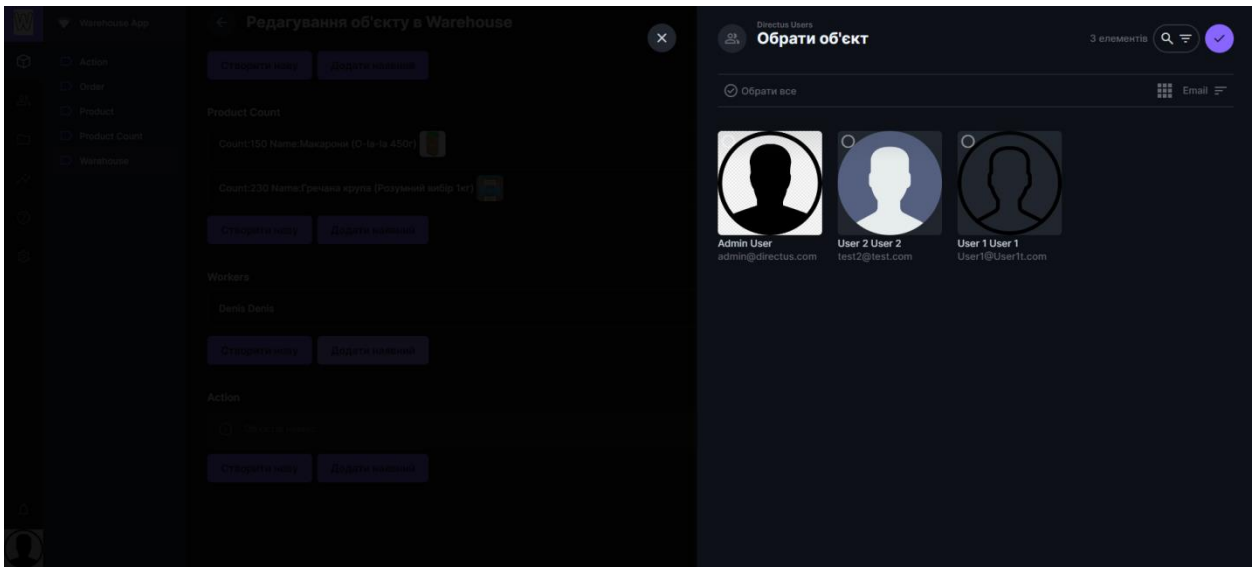
Рисуцнок 3.11 – Перелік продуктів



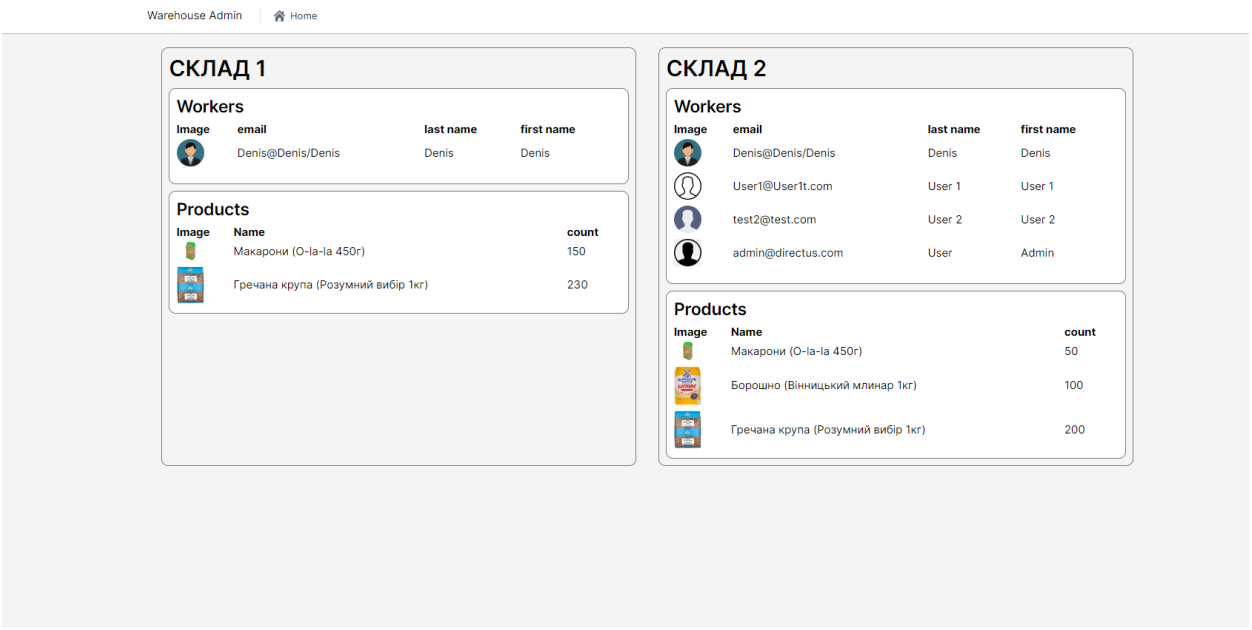
Рисуцнок 3.12 – Інформація продукту



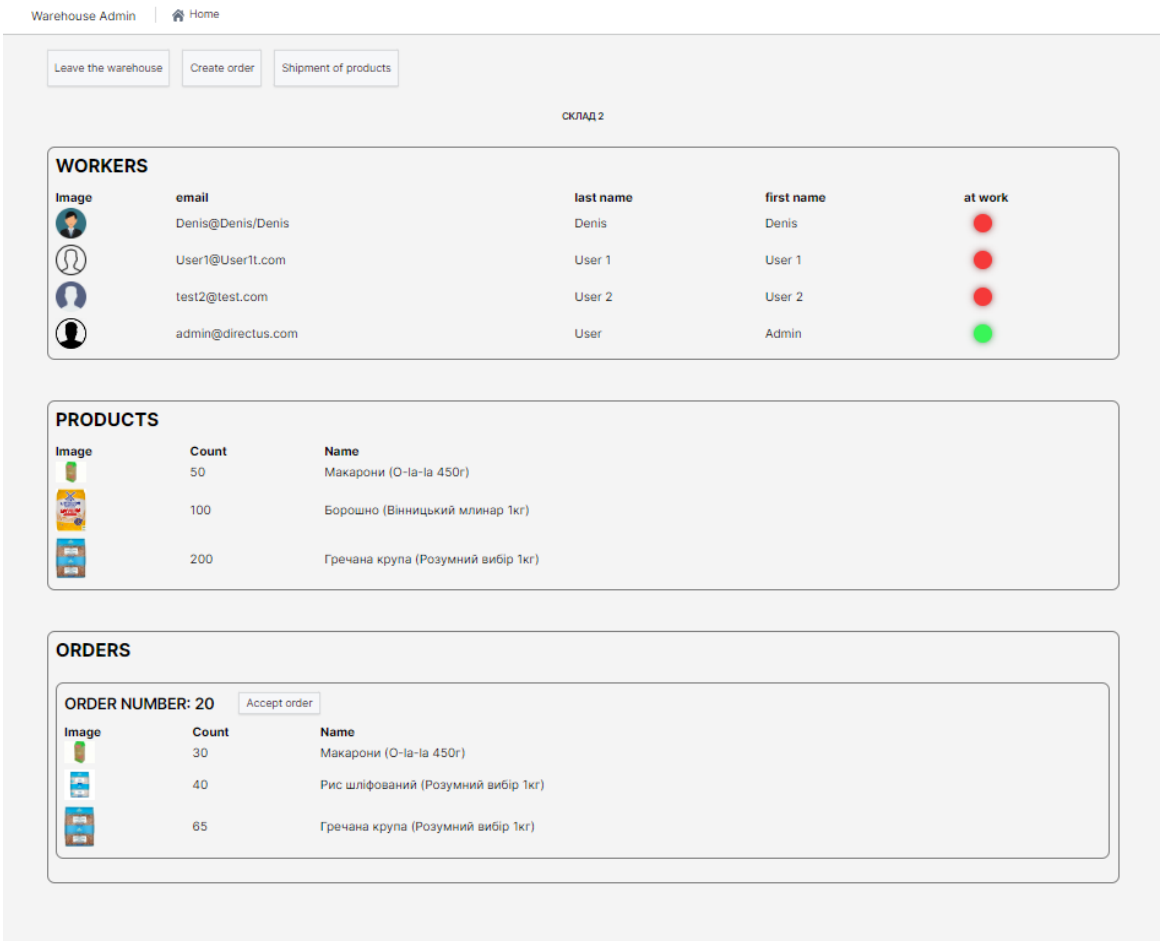
Рисуцнок 3.13 – Список дій



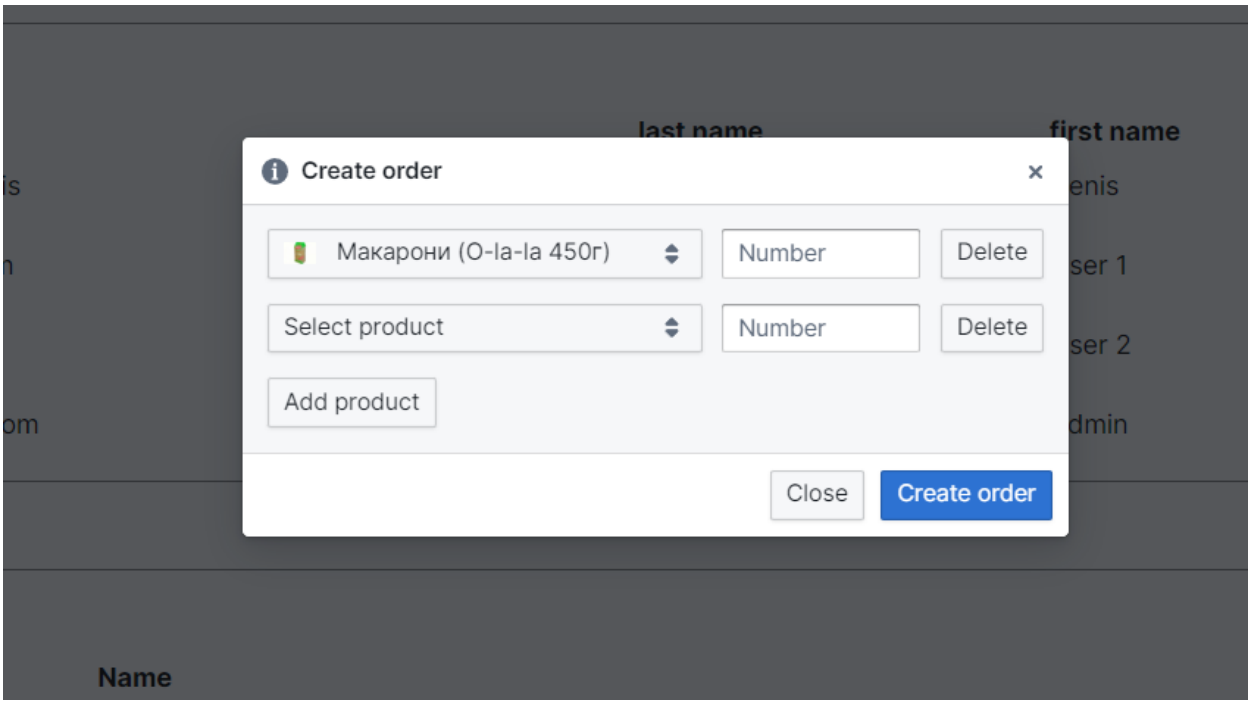
Рисуцнок 3.14 – Інтерфейс додавання працівника на сткад



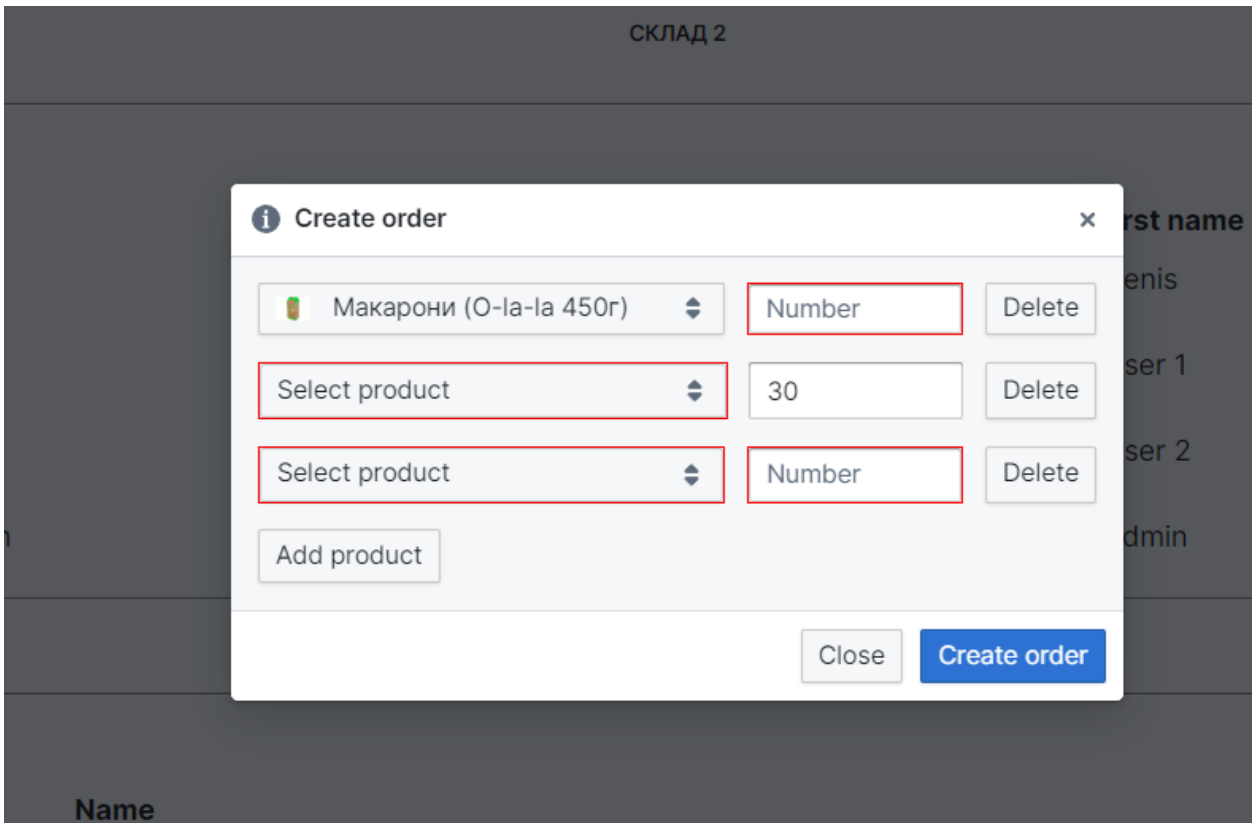
Рисуцнок 3.15 – Список складів



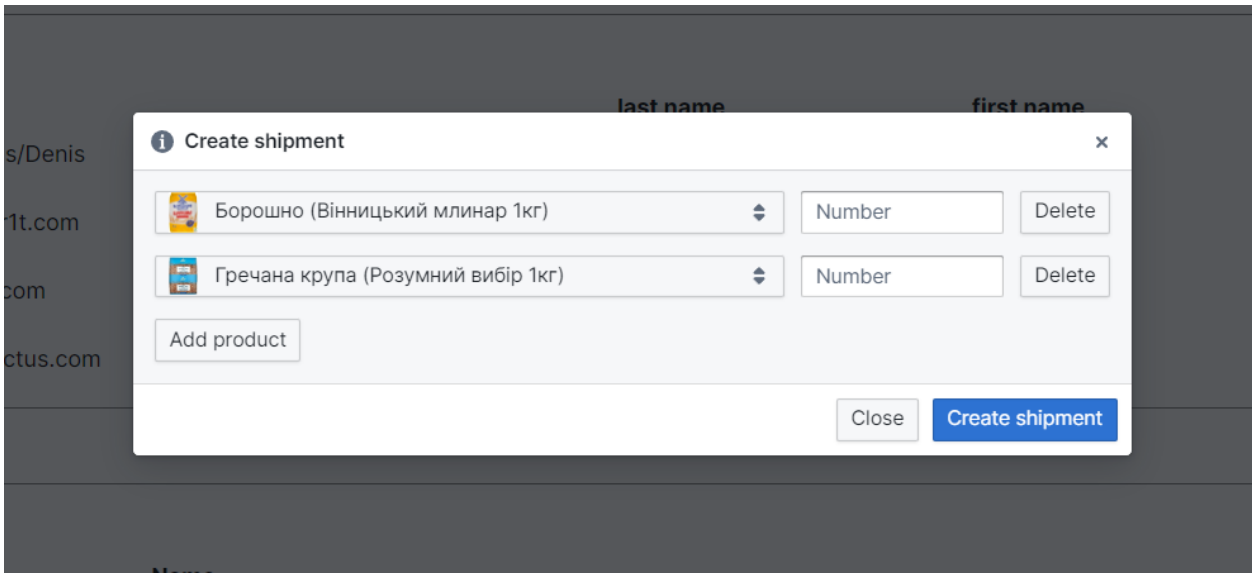
Рисуцнок 3.16 – Інформація про склад



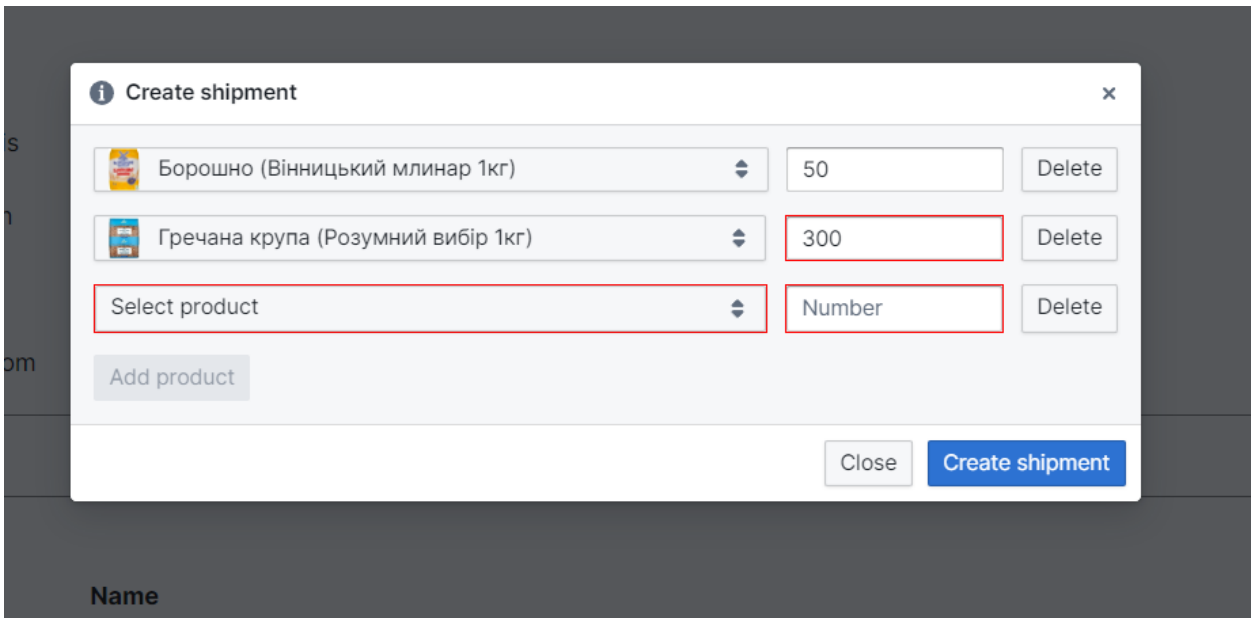
Рисуцнок 3.17 – Модальне вікно створення замовлення



Рисуцнок 3.18 – Валідація модального вікна створення замовлення



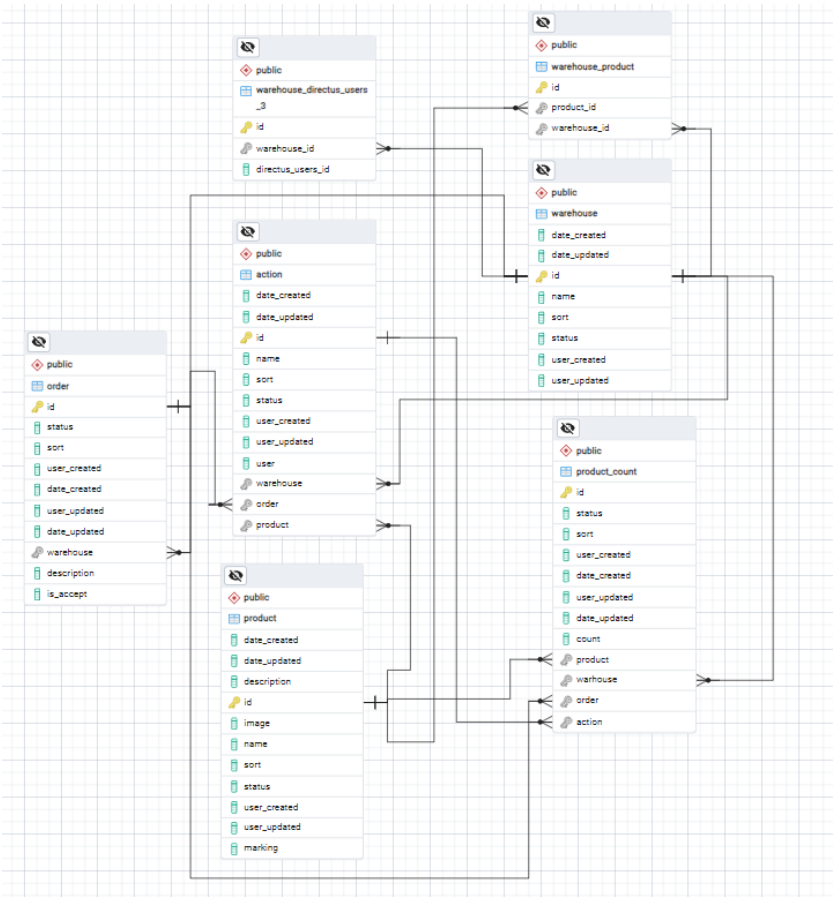
Рисуцнок 3.19 – Модальне вікно створення відвантаження



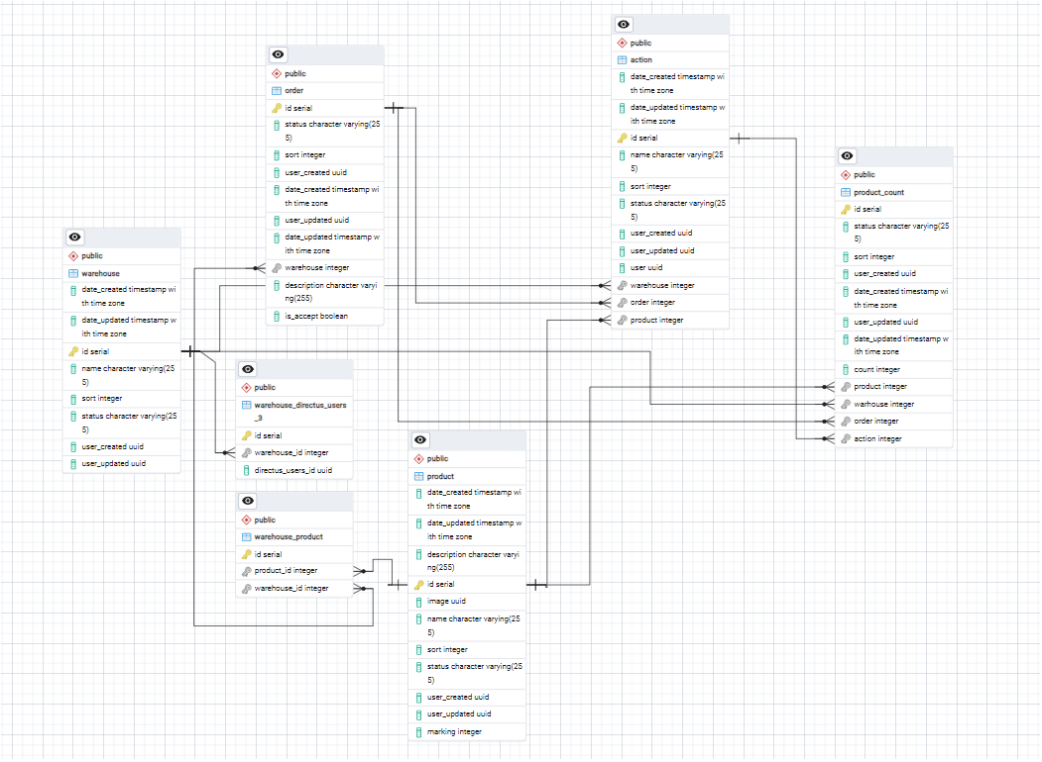
Рисуцнок 3.20 – Валідація модального вікна створення відвантаження

5.6 Структура таблиць бази даних

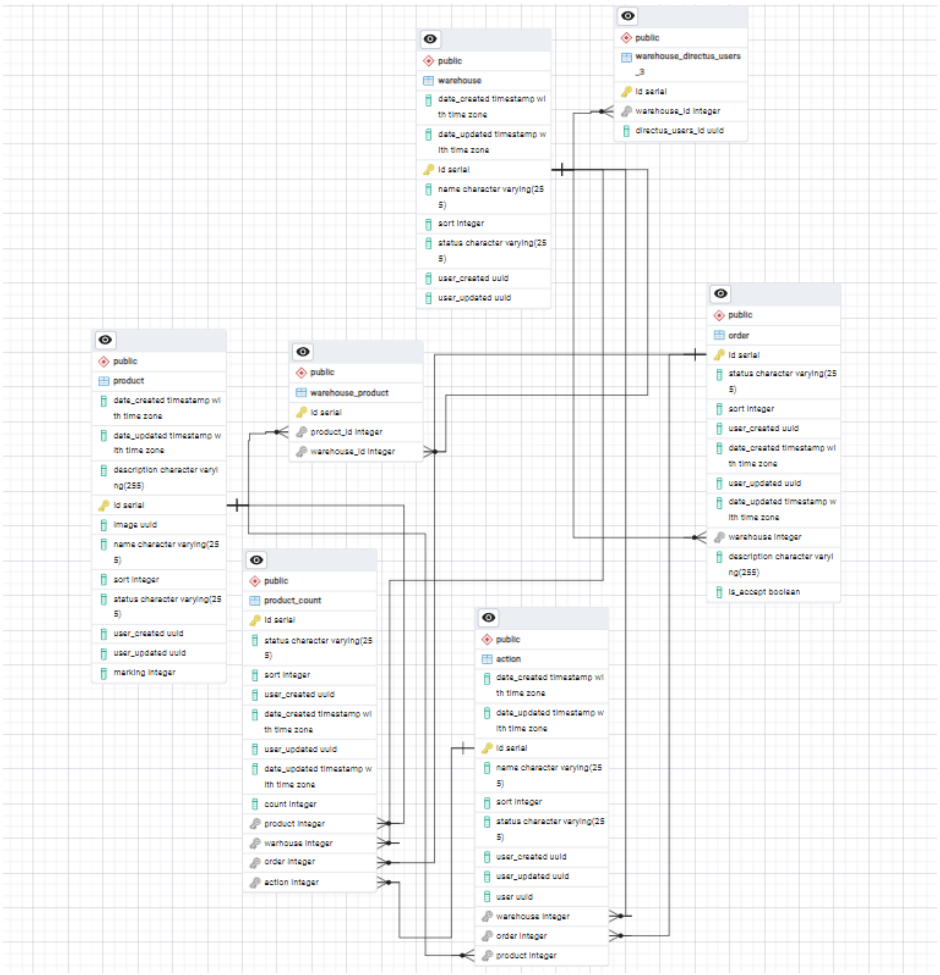
На рис 3.21 – 3.23 подано скріншоти структури таблиць бази данних



Рисуцнок 3.21 – Структура таблиці order



Рисуцнок 3.22 – Структура таблиці warehouse



Рисуцнок 3.23 – Структура таблиці product

ВИСНОВКИ

Інновації у виробництві індустрії 5.0 відображають стрімкий технологічний прогрес та спрямована на забезпечення сталого розвитку виробництва з акцентом на людські цінності та співпрацю між людиною та технологіями.

Цифрова трансформація та хмарні технології є ключовими аспектами цифрової трансформації, що сприяє підвищенню ефективності та доступності IT-ресурсів.

Загалом, ця інформація свідчить про важливість інновацій у галузі технологій, особливо в контексті підвищення продуктивності, сталого розвитку та підтримки людських цінностей у виробничих процесах.

Next.js, Directus.sdk, Docker та PostgreSQL – це потужні технології, які можуть допомогти вам створювати та розгортати веб-застосунки.

Кожна з них має свої плюси та мінуси, тому важливо вибрати правильну технологію для ваших конкретних потреб.

Next.js ідеально підходить для SEO-оптимізованих веб-сайтів, SPA, статичних сайтів та динамічних веб-застосунків.

Directus.sdk спрощує взаємодію React-застосунків з API Directus.

Docker забезпечує портативність, ізоляцію, масштабованість, ефективність та безпеку при розгортанні та тестуванні програмного забезпечення [12].

PostgreSQL – це надійна, потужна та масштабована СУБД, ідеально підходяща для зберігання структурованих даних, підтримки складних запитів, забезпечення високої доступності та масштабування.

Розроблена система автоматизованого управління складською системою з використанням хмарних технологій демонструє високу ефективність, надійність та гнучкість. Використання сучасних технологій та інструментів дозволило створити рішення, яке відповідає вимогам сучасного

ринку та забезпечує оптимізацію складських процесів. Впровадження цієї системи дозволить підприємствам підвищити продуктивність праці, знизити витрати на управління складськими операціями, забезпечити точність та своєчасність виконання замовлень, а також покращити загальну ефективність бізнес-процесів.

Проведені дослідження та розробки підтвердили актуальність використання хмарних технологій у сфері складського управління та відкрили нові перспективи для подальших досліджень та вдосконалення розробленої системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Харківський національний університет радіоелектроніки [Електронний ресурс]. – Режим доступу: www/URL: https://nure.ua/department/kafedra-kompyuterno-integrovanih-tehnologiy avtomatizatsiyi-ta-mehatroniki-kitam.

2. ДСТУ 3008:2015. [Електронний ресурс] /– Режим доступу: www / URL:https://drive.google.com/file/d/16XBXdCm8beHzpriYutDjwWuCkUknH1SO/view?hl=ru

3. Методичні вказівки з підготовки та захисту кваліфікаційної роботи . [Електронний ресурс] /– Режим доступу: www / URL:https://drive.google.com/file/d/1iBbYpJpLQxpgNYUNZG_vKNrGyNlfogPL/view?hl=ru

4. Industry 5.0—A Human-Centric Solution . [Електронний ресурс] /– Режим доступу: www / URL: https://www.mdpi.com/2071-1050/11/16/4371

5. Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В., Новоселов С. П., Демська Н. П. Проектування мобільних маніпуляційних роботів: Монографія. – Х.:, 2022. – 427 с

6. Филипенко А.И., Невлюдов И.Ш. СУЧАСНІ ТЕХНОЛОГІЇ ТА ТЕХНІЧНІ ЗАСОБИ АВТОМАТИЗАЦІЇ ВИРОБНИЦТВА РАДІОЕЛЕКТРОННОГО ПРИЛАДОБУДУВАННЯ. ХНУРЕ. – Харків: ХНУРЕ, 2021. – 202 с.

[Електронний ресурс] /– Режим доступу: www / URL:https://openarchive.nure.ua/server/api/core/bitstreams/f5b3143d-4bd8-432c-bc59-24531fa6e44c/content

7. hub kyivstar. [Електронний ресурс] /– Режим доступу: www / URL: https://hub.kyivstar.ua/articles/perevagy-ta-pryklady-paas-dlya-biznesu

8. Електронний документообіг. [Електронний ресурс] /– Режим доступу: www / URL: https://edin.ua/shho-take-xmarni-texnologi%D1%97-i-

navishho-voni-potribni/

9. Невлюдов І. Ш. Автоматичне управління технологічними об’єктами [Електронний ресурс]: Підручник / І. Ш. Невлюдов, О. В. Токарєва; М-во освіти і науки України, ХНУРЕ. – Харків: ХНУРЕ, 2018. – 190 с.

10. Intelligent Information Technologies and Systems. [Електронний ресурс] /– Режим доступу: www /

URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/47f2177b-d23f-45a3-a441-3f7cfce7ce8c/content

11. Цимбал О. М. Системи адаптації роботів і технологія OpenCV: навч. посіб. / О. М. Цимбал, А. І. Бронніков; Харків. нац. ун-т радіоелектроніки. – Харків: ХНУРЕ, 2019. – 144 с.: іл. – ISBN 978-966-659-268-5.

12. Автоматизація – штучний інтелект сучасного підприємства [Електронний ресурс] /– Режим доступу: www / URL: https://web.kpi.kharkov.ua/acem/uk/avtomatizatsiya/

13. Використання сучасних Web-фреймворків. [Електронний ресурс] /– Режим доступу: www /

URL:https://moodle.znu.edu.ua/pluginfile.php/1162911/mod_resource/content/1/%D0%A0%D0%BE%D0%B7%D0%B4%D1%96%D0%BB_3_%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%97_NextJS.pdf

14. directus.io. [Електронний ресурс] /– Режим доступу: www / URL: https://directus.io/

15. ux-republic. [Електронний ресурс] /– Режим доступу: www / URL: https://www.ux-republic.com/uk/https-wp-me-p5iq6x-9ix/

16. foxminded. [Електронний ресурс] /– Режим доступу: www / URL: https://foxminded.ua/postgresql-shcho-tse/

17. Методичний підхід оцінки ефективності функціонування підприємств. [Електронний ресурс] /– Режим доступу: www / URL: https://api.dspace.khadi.kharkov.ua/server/api/core/bitstreams/9279978e-5d38-4c90-b84e-05f54dba27a6/content

18. Статичні методи контролю і теорія надійності. [Електронний

ресурс] /– Режим доступу: www / URL: http://radio-vtc.inf.ua/Quality/L4_5.pdf

19. Автоматизація складського обліку. [Електронний ресурс] /– Режим доступу: www / URL: https://appointer.ua/blog/avtomatizatsiya-skladskogo-obliku-sho-tse-i-yak-pratsue/