Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій
(повна найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: "Розроблення вебсистеми обліку продукції підприємств з використанням фреймворків Laravel та Vue.js"

Виконав: студент 6 курсу групи КН(м)
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Марадь Ю.І.
(прізвище та ініціали)

Керівник Карашецький В.П.
(прізвище та ініціали)

Рецензент Пірко І.Б.
(прізвище та ініціали)

Львів – 2024 року

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Борецька І.Б.
"05" січня 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Марадь Юрій Ігорович
(прізвище, ім'я, по батькові)

1. Тема магістерської роботи Розроблення вебсистеми обліку продукції підприємств з використанням фреймворків Laravel та Vue.js

керівник роботи Карашецький Володимир Петрович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "13" 02 2023 року № С-49
2. Термін подання студентом проєкту (роботи) 05 січня 2024 року
3. Вихідні дані до проєкту (роботи) Розробити програмне забезпечення вебсистеми обліку продукції підприємств з використанням фреймворків Laravel та Vue.js.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області.
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
Розділ 5. Розроблення стартап-проєкту
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 15 лютого 2023 року

2

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних та інших джерел згідно досліджуваної теми.	1.03.23 – 27.03.23	виконано
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки.	4.04.23 – 24.04.23	виконано
3	Постановка задачі та її формалізація.	25.04.23 – 1.05.23	виконано
4	Вибір та обгрунтування методів і засобів проведення дослідження.	2.05.23 – 22.05.23	виконано
5	Розроблення коцептувальної схеми реалізації завдання.	23.05.23 – 29.05.23	виконано
6	Програмна реалізація завдання.	30.05.23 – 30.08.23	виконано
7	Тестування програмного продукту та отриманих результатів	31.08.23 – 30.10.23	виконано
8	Розробка пояснювальної записки та презентації дипломної роботи	31.10.23 – 30.12.23	виконано

Студент _____ (підпис) Мараль Ю.І. (прізвище та ініціали)

Керівник роботи _____ (підпис) Карашенський В.П. (прізвище та ініціали)

АНОТАЦІЯ

Магістерська дипломна робота складається із 78 стор., 8 рис., 4 табл., 2 додатків, 15 джерел.

В даному дипломному проєкті розроблено вебсистему, яка дозволяє зручно вести облік прийому та відвантаження продукції підприємств. Передбачено можливість заповнення та редагування інформації адміністратором. Виконано тестування розробленої системи.

Ключові слова: вебсистема, облік продукції, PHP, Laravel, Vue,js, MVC, MySQL, реляційна БД.

ABSTRACT

The master's thesis consists of 78 pages, 8 figures, 4 tables, 2 appendixes, 15 sources.

In this diploma project, a web system was developed that allows you to conveniently keep records of the receipt and shipment of products of enterprises. It is possible to fill in and edit information by the administrator. The developed system was tested.

Key words: web system, product accounting, PHP, Laravel, Vue.js, MVC, MySQL, relational database.

ТЕХНІЧНЕ ЗАВДАННЯ

Провести огляд літератури по даній тематиці та проаналізувати існуючі системи для ведення обліку продукції на підприємствах, розробити вебсистему, яка дозволяє зручно вести облік прийому та відвантаження продукції на підприємстві. Передбачити можливість заповнення та редагування інформації адміністратором. Виконати тестування розробленої системи.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ІСР – інтегроване середовище розробки;

БД – база даних;

ООП – об’єктно-орієнтоване програмування;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

API – Application Programming Interface;

CSS – Cascading Style Sheets;

HTML – HyperText Markup Language;

JS – JavaScript;

IDE – Integrated Development Environment;

MVC – Model-View-Controller;

PHP – Hypertext Preprocessor;

SPA – Single Page Application;

VS Code – Visual Studio Code.

ВСТУП

Дипломна робота присвячена розробці вебсистеми для ведення обліку прийому та відвантаження продукції підприємств. Головна мета проєкту – спрощення та автоматизація процесів обліку продукції, зменшення можливості людської помилки та підвищення загальної ефективності роботи підприємств. У роботі розглядаються сучасні підходи та інструменти веброзробки, а також особливості проєктування та реалізації подібних систем.

Даний проєкт є актуальним, оскільки процеси автоматизації та цифровізації набувають все більшого значення в сучасному світі. Використання веб-технологій дозволяє створити масштабовану, доступну та легку у використанні систему.

Проєкт передбачає аналіз існуючих рішень, визначення вимог до системи, її проєктування, розробку та тестування. Особлива увага приділяється зручності інтерфейсу, безпеці даних та інтеграції з іншими системами та сервісами.

Дана дипломна робота може стати важливим внеском у поліпшення процесів обліку та управління на підприємствах, а також допоможе в підготовці фахівців у галузі ІТ та веброзробки.

Об’єкт дослідження – системи ведення обліку прийому та відвантаження продукції на підприємствах.

Предмет дослідження – вебсистема обліку продукції підприємств з використанням фреймворків Laravel та Vue.js [1].

Наукова новизна – Розробка інноваційного підходу до інтеграції фреймворків Laravel та Vue.js для створення ефективної вебсистеми обліку. Впровадження передових методик управління даними та розробки інтерфейсу користувача.

Практична значимість – система сприятиме покращенню точності обліку на підприємствах, зниженню витрат часу та ресурсів, а також полегшить процеси прийняття управлінських рішень.

Актуальність роботи – автоматизація процесів стає все більш важливою у сучасному бізнес-середовищі. Розробка надійних і ефективних вебсистем є ключовою для підвищення продуктивності та конкурентоспроможності підприємств.

Мета дослідження – розробка вебсистеми для ведення обліку прийому та відвантаження продукції, яка була б легкою у використанні, надійною і ефективною, з використанням сучасних технологій і фреймворків.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Сервіси для ведення обліку прийому та відвантаження продукції підприємств

Облік прийому та відвантаження продукції є ключовим аспектом управління матеріальними ресурсами будь-якого підприємства. Сучасні технології надають можливості для автоматизації цих процесів, підвищуючи ефективність та знижуючи ризики помилок.

1.2 Сервіс “RemOnline”

“RemOnline” – це вебзастосунок, який дозволяє автоматизувати процеси відстеження та обліку продукції (рис. 1.1) Даний хмарний сервіс надає широкий спектр функціональності, включаючи реєстрацію товарів, відстеження їх переміщення, а також генерацію звітів. Особливістю “RemOnline” є використання штрих-кодів для швидкої ідентифікації товарів, що значно спрощує процеси прийому та відвантаження. Однак, система має обмеження у кастомізації інтерфейсу та інтеграції з іншими системами.

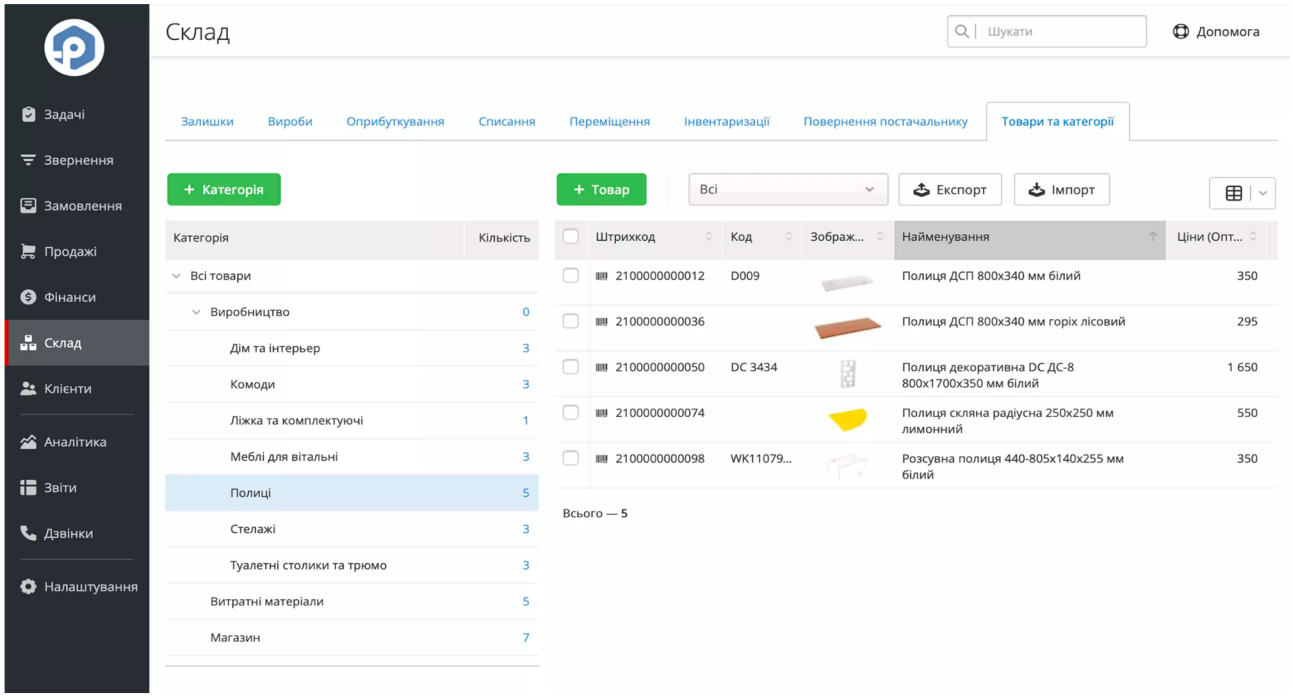

Рис. 1.1. Головна сторінка сервісу “RemOnline”
(розділ ведення обліку на складі)

1.3 Сервіс "DNTrade"

"DNTrade" – це інноваційна платформа, яка зосереджена на автоматизації взаємодій між постачальниками та покупцями. Платформа надає можливості для управління замовленнями, складським обліком, а також плануванням логістики (рис.1.2). "DNTrade" інтегрується з різними ERP-системами, що дозволяє забезпечити високий рівень синхронізації даних. Однак, складність конфігурації та висока вартість можуть бути перешкодами для її використання малими та середніми підприємствами.

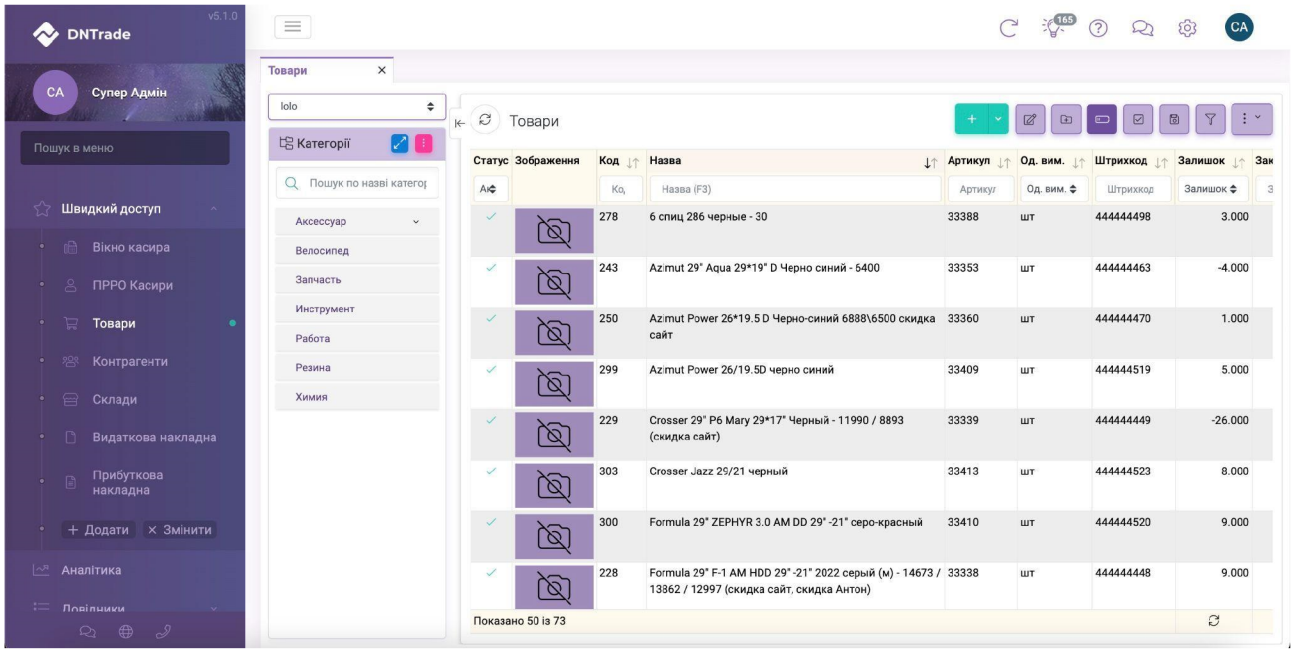

Рис. 1.2. Головна сторінка сервісу “DNTrade” (розділ ведення обліку)

Висновки. Аналіз існуючих сервісів, таких як “RemOnline” та "DNTrade", виявляє ключові вимоги до ефективних систем обліку – гнучкість, інтеграційна здатність, простота використання, та економічна доступність. Вони також підкреслюють потребу в розробці більш адаптивних рішень, які зможуть задовольнити специфічні потреби різних типів підприємств.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування та фреймворку

Вибір мови програмування для розробки вебсистеми – це ключовий крок, який впливає на архітектуру, розширюваність та ефективність проєкту. Для бекенду було обрано PHP з огляду на його широке використання у веб-розробці, велику спільноту та підтримку. PHP ефективний для розробки складних ввебзастосунків завдяки своїй гнучкості та вбудованим функціям [2].

Laravel було обрано як основний фреймворк для бекенду через його зрілу архітектуру, високу продуктивність та легкість у використанні. Laravel пропонує численні функції, такі як ORM Eloquent, систему шаблонів Blade, вбудовану підтримку тестування, що значно спрощує розробку та підтримку великих застосунків. Також, Laravel підтримує MVC-паттерн, що сприяє чіткому розділенню логіки та представлення, забезпечуючи кращу організацію коду [4].

Для фронтенду було обрано JavaScript (JS) – мову, яка є стандартом для веброзробки [5]. JS дозволяє створювати динамічні, інтерактивні вебінтерфейси. Вона інтегрується із HTML та CSS, забезпечуючи гнучкість та широкі можливості для розробки користувацьких інтерфейсів.

Vue.js був обраний для фронтенду з огляду на його легкість, гнучкість та високу швидкість. Vue.js – це прогресивний JavaScript фреймворк, який дозволяє створювати інтуїтивно зрозумілі та реактивні інтерфейси. Його легка інтеграція з іншими бібліотеками та проєктами, а також простота у використанні робить його ідеальним вибором для даного проєкту.

2.2 Вибір середовища розробки

Середовище розробки є ключовим інструментом для будь-якого розробника. Воно об’єднує в собі редактор коду, інструменти для відлагодження та засоби автоматизації. Вибір правильного ICP може значно підвищити продуктивність розробки та спростити багато процесів.

PhpStorm від компанії JetBrains є одним з найпопулярніших інтегрованих середовищ розробки (ICP) для PHP розробки. Його переваги включають підтримку різних PHP фреймворків, зокрема Laravel, а також інтеграцію з сучасними інструментами для розробки, такими як Composer. PhpStorm також надає розширену підтримку для фронтенду, включаючи JavaScript, HTML і CSS, що робить його ідеальним вибором для повноцінної веброзробки [6].

Visual Studio Code (VS Code) від Microsoft також є чудовим вибором для веброзробки. Це безкоштовний, легкий, але потужний редактор коду, який підтримує широкий спектр мов програмування та фреймворків. Його гнучкість дозволяє легко інтегрувати як Laravel, так і Vue.js через розширення та плагіни. VS Code відомий своєю швидкістю, зручним користувацьким інтерфейсом та великою кількістю доступних розширень.

Хоча PhpStorm та VS Code є основними рекомендованими варіантами, існують інші IDE, які також можуть бути розглянуті, залежно від особистих переваг та вимог проєкту. Наприклад, Sublime Text та Atom пропонують гнучкість та швидкість, але можуть вимагати більше налаштувань для інтеграції з конкретними фреймворками.

Вибір середовища розробки залежить від багатьох факторів, включаючи особисті уподобання, специфіку проєкту та необхідні інструменти. Для даного проєкту, PhpStorm та VS Code є найкращими варіантами з огляду на їх широку підтримку PHP та JavaScript, включаючи фреймворки.

2.3 Проєктування архітектури системи

При проєктуванні архітектури вебсистеми важливо забезпечити її масштабованість, безпеку, та ефективність. Система буде розділена на три основні компоненти: бекенд (серверна частина), фронтенд (клієнтська частина), та база даних (БД).

Бекенд системи буде розроблений з використанням PHP і Laravel фреймворку. Laravel забезпечує чітке відокремлення бізнес-логіки від представлення даних, дотримуючись архітектурного шаблону MVC (рис. 2. 1).

Це сприяє легшій підтримці коду та його розширенню. Серверна частина буде відповідати за обробку запитів від клієнта, взаємодію з БД, авторизацію користувачів, та логіку обробки даних.

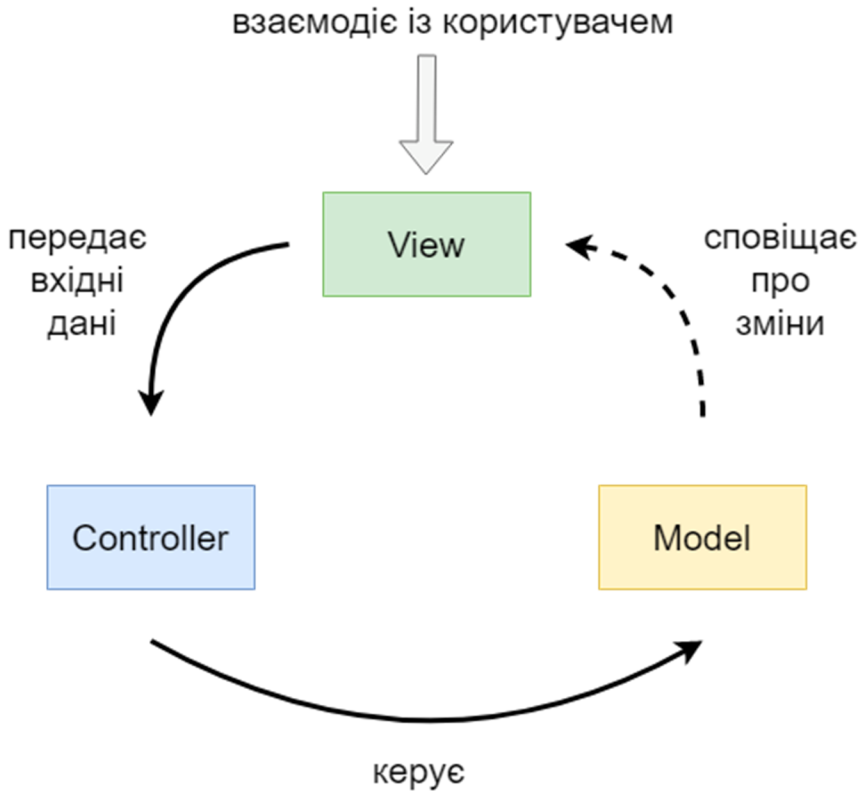

Рис. 2.1. Шаблон MVC

Фронтенд системи буде розроблений з використанням JavaScript та Vue.js фреймворку. Vue.js дозволяє створювати динамічні, інтерактивні користувацькі інтерфейси. Використання компонентного підходу сприяє більшій організованості та легкій підтримці коду. Фронтенд буде взаємодіяти з бекендом через REST API для отримання та відправлення даних.

База даних є ключовим компонентом, який забезпечує зберігання та управління даними системи. Вибір системи керування базами даних (СКБД) залежить від вимог до швидкості, надійності, та легкості масштабування. Для даної системи можна використовувати MySQL або PostgreSQL [8]. Обидві

системи є потужними, надійними та масштабованими, та мають широку підтримку у Laravel.

Інтеграція між бекендом, фронтендом та БД є критичним аспектом для забезпечення плавної роботи системи. REST API буде використовуватися для комунікації між клієнтом та сервером. Це забезпечить чітке розділення логіки користувацького інтерфейсу та серверної обробки, полегшуючи тестування та розширення системи.

Висновки. Архітектура системи є фундаментом для створення надійної, ефективної та легко підтримуваної вебсистеми. Використання Laravel для бекенду, Vue.js для фронтенду, та реляційної СКБД для зберігання даних забезпечить міцну основу для нашого проєкту.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Використання математичних методів для обліку продукції

Математика грає ключову роль у веденні статистики та аналізі даних. Вона дозволяє точно обчислювати важливі показники, аналізувати тенденції та робити обґрунтовані прогнози. Використання математичних методів у вебсистемі для обліку продукції сприяє підвищенню точності даних та ефективності управлінських рішень.

Статистичний аналіз даних забезпечує глибоке розуміння процесів прийому та відвантаження продукції. Для цього використовуються такі методи, як описова статистика (середні значення, медіана, мода), дисперсійний аналіз, та лінійна регресія. Ці методи допомагають ідентифікувати закономірності, виявити аномалії та оцінювати ефективність бізнес-процесів.

Прогнозування є важливим аспектом у плануванні та управлінні запасами. Використання алгоритмів машинного навчання, таких як часові ряди та нейронні мережі, може значно підвищити точність прогнозів. Це дозволяє підприємствам оптимізувати запаси та знизити витрати.

Візуалізація даних є важливим інструментом для зручного представлення статистичних результатів. Графіки, діаграми, та інші візуальні засоби допомагають легко інтерпретувати великі обсяги даних та приймати рішення. Використання бібліотек JavaScript, таких як D3.js або Chart.js, може забезпечити ефективну візуалізацію даних у вебінтерфейсі.

У проєкті мною були використані наступні методи:

– розрахунок середнього обсягу продажів (Mean Sales Volume);

– розрахунок дисперсії продажів (Sales Variance);

– прогнозування з використанням лінійної регресії.

Додатково до використання формул, ефективним засобом аналізу є візуалізація даних. Наприклад, створення графіку часових рядів продажів дозволить візуально оцінити тенденції та сезонні коливання. Використання

гістограм чи коробкових діаграм (box plots) може допомогти в ідентифікації розподілів та викидів у даних.

Висновки. Використання математичних методів та алгоритмів для ведення статистики дозволяє системі забезпечувати точні та надійні дані для аналізу. Це сприяє підвищенню якості управлінських рішень та оптимізації бізнес-процесів на підприємстві.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Створення бази даних

База даних для системи обліку продукції буде розроблена так, щоб забезпечити ефективне зберігання, обробку та витягування інформації. Основою стане реляційна модель даних з використанням MySQL або PostgreSQL.

Основні таблиці БД та їх призначення:

- Продукти (Products). Поля: ProductID (ключ), Name, Description, CategoryID, Price, Quantity.Зберігає інформацію про кожен продукт.
- Категорії Продуктів (ProductCategories). Поля: CategoryID (ключ), CategoryName. Класифікація продуктів за категоріями.
- Замовлення (Orders). Поля: OrderID (ключ), CustomerID, OrderDate, TotalAmount. Інформація про замовлення від клієнтів.
- Деталі Замовлень (OrderDetails). Поля: OrderDetailID (ключ), OrderID, ProductID, Quantity, Price. Деталізація замовлень, включаючи продукти та їх кількість.
- Клієнти (Customers). Поля: CustomerID (ключ), Name, ContactInfo, Address. Зберігання інформації про клієнтів.

Для забезпечення ефективності та надійності БД буде нормалізована до 3НФ (третьої нормальної форми). Це зменшить дублювання даних та спростить їх оновлення. Також будуть встановлені обмеження цілісності, як-от ключі та обмеження зв'язків між таблицями, щоб запобігти некоректному введенню даних.

Реалізація БД включає створення таблиць, визначення зв'язків між ними та наповнення тестовими даними. SQL-скрипти будуть використані для створення структури БД та її первинного наповнення [10].

Створення добре структурованої, нормалізованої БД є критично важливим для забезпечення ефективного та безпечного зберігання та обробки інформації у вебсистемі. Використання сучасних реляційних СКБД і

найкращих практик проєктування даних забезпечить надійну основу для нашої системи.

4.2 Розробка бізнес-логіки

Бізнес-логіка вебсистеми визначає, як вона обробляє вхідні дані, виконує операції, інтегрується з іншими системами та повертає результати. Цей розділ розглядає, як логіка системи буде реалізована з використанням Laravel та як вона взаємодіятиме з БД та користувацьким інтерфейсом.

4.2.1 Ключові компоненти бізнес-логіки

Ключовими компонентами бізнес-логіки розроблюваної вебсистеми є:

– Логіка для обробки замовлень, яка включатиме створення, оновлення та відстеження статусу замовлення. Це включає інтеграцію з таблицею “Замовлення” та “Деталі Замовлень” у БД.

– Функціонал для додавання, оновлення, видалення та перегляду продуктів. Інтеграція з таблицею “Продукти” дозволить забезпечити актуальність інформації про продукти.

– Реалізація системи авторизації, реєстрації працівників, управління ролями та доступами. Це важливо для забезпечення безпеки системи та контролю доступу.

– Можливість інтеграції з зовнішніми системами, такими як ERP, CRM, або логістичні сервіси, через API.

Laravel надає різні інструменти для розробки бізнес-логіки:

– Eloquent ORM. Для інтеракції з БД, забезпечує чистий та зрозумілий синтаксис для запитів.

– Middleware. Для реалізації логіки авторизації та контролю доступу.

– Service Container та Service Providers. Для управління залежностями та організації бізнес-логіки.

Розробка бізнес-логіки є важливим етапом у створенні вебсистеми. Вона

повинна бути гнучкою для адаптації до змін у бізнес-процесах, масштабованою для обробки зростаючого обсягу даних, та надійною для забезпечення безперебійної роботи системи.

Основними контролерами вебсистеми є:

1. ProductController. Методи: create(), store(), edit(), update(), destroy(), show(). Управління каталогом продуктів (додавання, редагування, видалення, перегляд).

2. OrderController. Методи: create(), store(), update(), show(). Обробка замовлень (створення нових замовлень, оновлення статусу замовлень, перегляд деталей замовлення).

3. InventoryController. Методи: index(), update(). Управління запасами (відстеження кількості наявних товарів, оновлення запасів).

4. CustomerController. Методи: create(), store(), edit(), update(). Управління інформацією про клієнтів (реєстрація нових клієнтів, редагування інформації клієнтів).

5. AuthController. Методи: login(), logout(), changePassword(). Авторизація та аутентифікація працівників (вхід, вихід, зміна паролю).

6. ReportController. Методи: salesReport(), inventoryReport(), performanceReport(). Генерація звітів (продажі, запаси, продуктивність).

7. APIController. Методи: fetchData(), postData(), syncData(). Інтеграція з зовнішніми API, наприклад, для обміну даними з ERP системами.

Кожен з цих контролерів відіграє важливу роль у системі, забезпечуючи необхідну логіку та взаємодію з іншими частинами системи. Вони допоможуть забезпечити структурований та організований підхід до розробки, відповідаючи за конкретні функціональні аспекти вебсистеми.

4.3 Розробка інтерфейсу користувача

У сучасному світі веб-технологій, ефективний та інтуїтивно зрозумілий інтерфейс користувача є ключем до успіху будь-якої системи. Процес розробки фронтенду вебсистеми для ведення обліку прийому та відвантаження продукції

включає не тільки візуальне представлення інформації, а й забезпечення зручної та логічної взаємодії з користувачем.

Мета полягає в розробці набору сторінок та компонентів, які разом формують користувацький інтерфейс, орієнтований на задоволення потреб і очікувань користувачів. Важливість уваги до деталей, що стосуються дизайну та взаємодії, не може бути недооцінена, адже вона безпосередньо впливає на продуктивність роботи та задоволення користувачів системою.

Використовуючи сучасні підходи та інструменти розробки, як Vue.js, створимо інтерфейс, який не тільки задовольняє функціональні вимоги, але й приносить задоволення від його використання. Розглядаючи важливі аспекти інтерфейсу, від головної панелі управління до деталізованих сторінок управління продуктами та замовленнями, забезпечимо, що кожна частина системи буде легкою у використанні та ефективною в роботі.

Почнемо з входу. Працівник має можливість зареєструватися або увійти, використовуючи email/пароль або скинути пароль якщо забув (рис. 4.1).

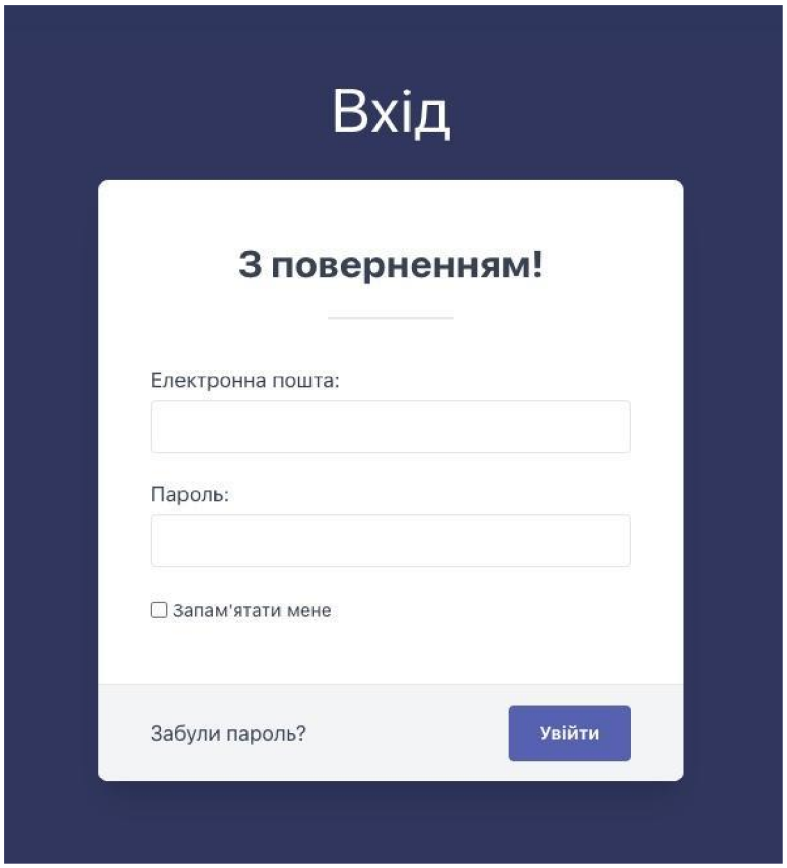

Рис. 4.1. Вхід в систему

Після входу в систему, працівник потрапляє на панель управління (рис. 4.2). Тут можна переглядати загальну статистику, останні записи про прийом і відвантаження продукції. Vue.js динамічно оновлює інформацію.

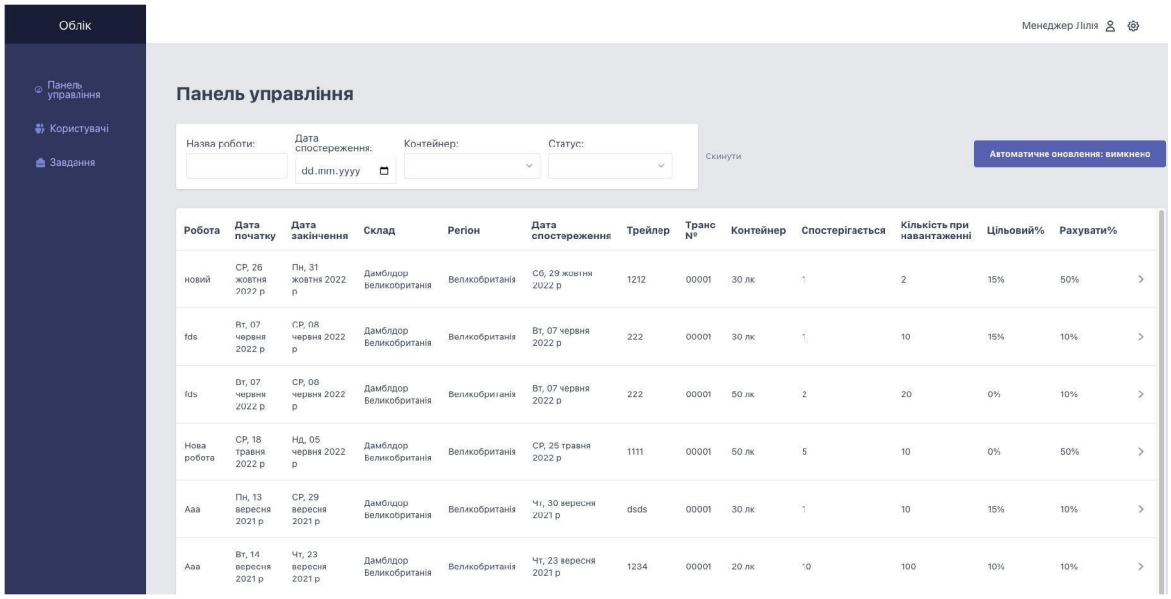

Рис. 4.2. Панель управління

Адміністратор системи має змогу додавати та редагувати працівників у системі (рис. 4.3). Також він має змогу давати їм ролі як от: адміністратор, менеджер.

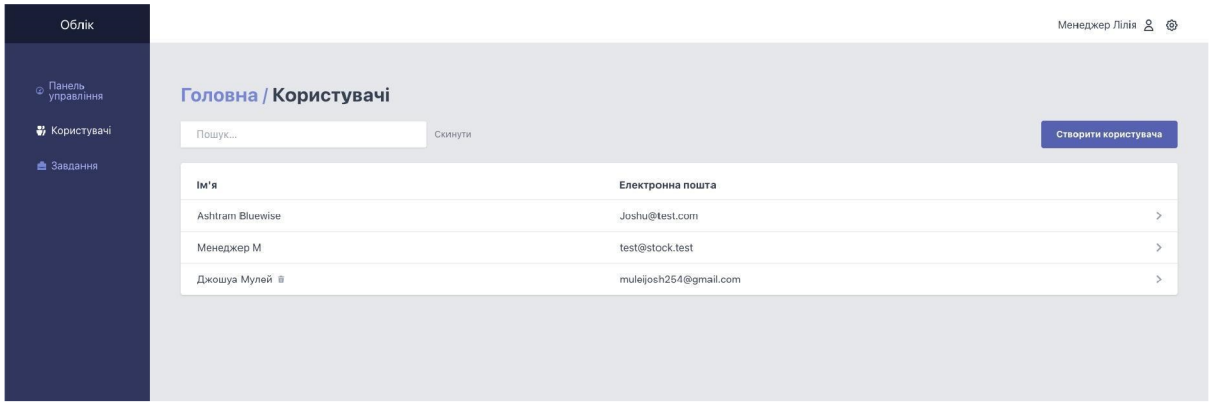

Рис. 4.3. Сторінка “Користувачі”

Адміністратор системи має змогу створювати завдання для менеджерів (рис. 4.4). Основні критерії – це назва завдання, дата його початку і закінчення та склад, на якому ведеться облік продукції.

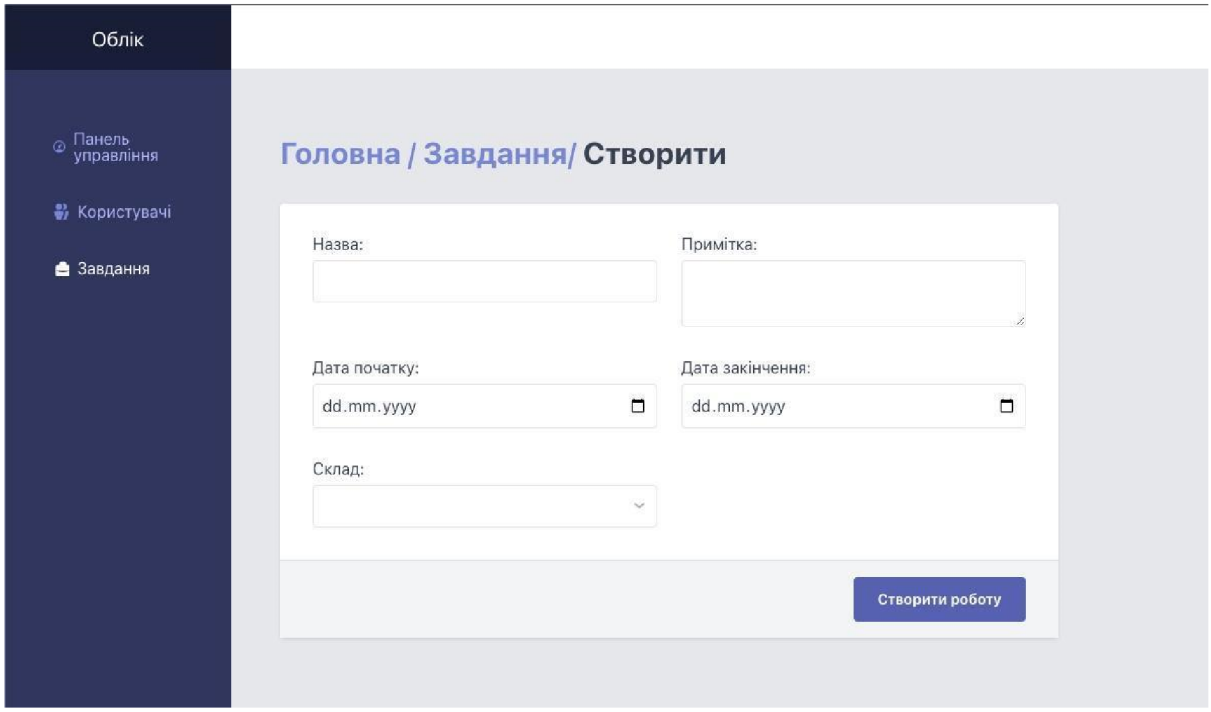

Рис. 4.4. Сторінка створення завдань для менеджерів

Працівник з ролю “Менеджер” може увійти в систему та за допомогою форми, розробленої на Vue.js, може внести дані про новий прийом і відвантаження продукції (рис. 4.5). Форма взаємодіє з Laravel API, забезпечуючи валідацію на фронтенді і бекенді.

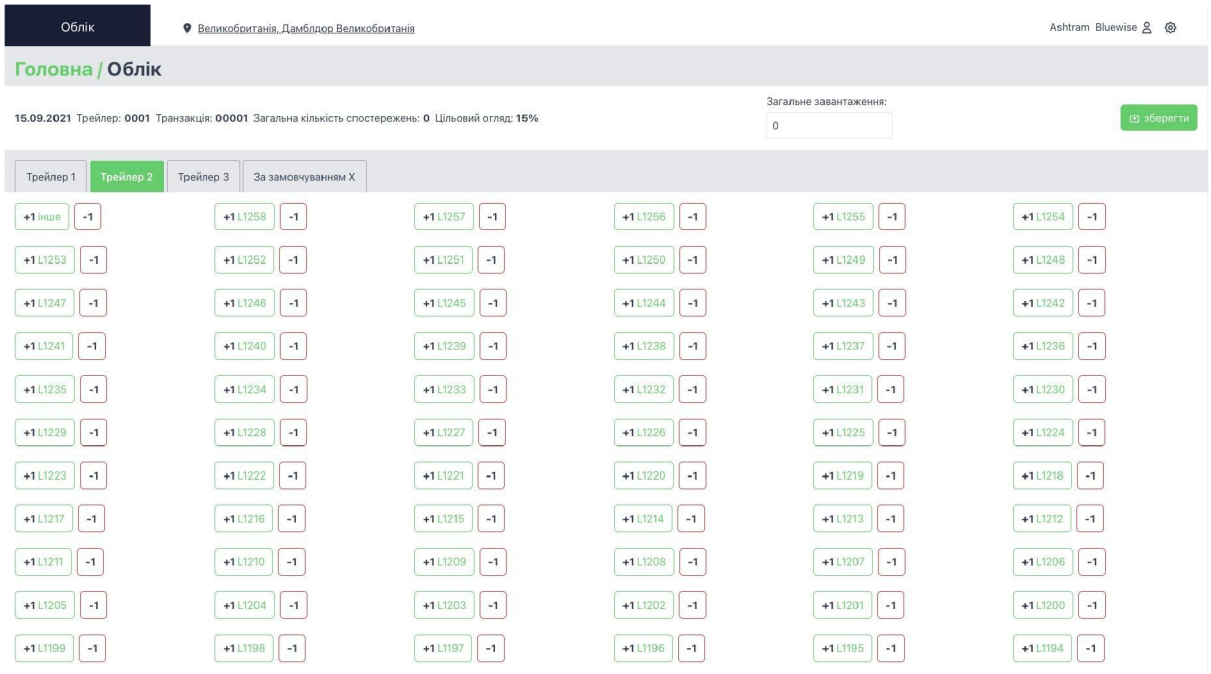

Рис. 4.5. Сторінка внесення облікових даних в систему

За допомогою кнопки “Налаштування профілю” можна змінити свої персональні дані, пароль і так далі. Є можливість двофакторної авторизації для підвищення безпеки.

4.4 Формування валідаторів та тестування

Валідація даних та тестування є невід’ємною частиною процесу розробки, що забезпечує надійність, безпеку та ефективність вебсистеми. Опишемо методи та практики, які будуть застосовані для перевірки вхідних даних та забезпечення якості коду.

Валідація даних – це процес перевірки того, що вхідні дані відповідають заданим критеріям перед їх обробкою або зберіганням.

Фронтенд-валідація реалізується на стороні клієнта за допомогою JavaScript та Vue.js для забезпечення негайного зворотного зв’язку.

Серверна валідація виконується на стороні сервера (Laravel), перевіряючи дані на відповідність бізнес-правилам та безпековим вимогам.

Тестування включає ряд дій, спрямованих на виявлення помилок та перевірку функціональності системи.

Модульне тестування включає тестування окремих модулів коду для перевірки їх правильності. Використовується PHPUnit для тестування PHP коду.

Функціональне тестування виконує перевірку окремих функцій системи щодо відповідності специфікаціям.

Інтеграційне тестування виконує тестування взаємодії між різними частинами системи, такими як серверний бекенд і клієнтський фронтенд.

Тестування користувацького інтерфейсу перевіряє елементи інтерфейсу та їх взаємодію з користувачем.

Автоматизація тестування та впровадження практик неперервної інтеграції (CI) допомагають забезпечити сталість та якість коду. Використовуються такі інструменти як Jenkins або GitLab CI для автоматизованого запуску тестів при кожному оновленні коду.

Висновки. В даному розділі описано розроблення бази даних та інтерфейсу користувача вебсистеми, представлено ключові компоненти бізнес-логіки. Ретельна валідація та всебічне тестування є ключовими для створення надійної, безпечної та ефективної вебсистеми. Вони дозволяють не лише запобігти помилкам, але й забезпечити високий рівень якості програмного продукту.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Таблиця 5.1. Інформаційна карта проЄкту

Назва номінації	Вебсистема
Назва проєкту	Вебсистема обліку продукції підприємств
Назва ВНЗ, спеціальності	НЛТУ, кафедра комп'ютерних наук, "Комп'ютерні науки"
Прізвище, ім'я, по-батькові автора	Марадь Юрій Ігорович
Цілі і задачі проєкту	Спроєктувати та розробити вебсистему для оптимізації процесів обліку прийому та відвантаження продукції на підприємстві
Короткий зміст проєкту	Розробка вебсистеми, яка дозволяє автоматизувати облік прийому та відвантаження продукції, ведення статистики, аналізу запасів та формування звітності для підприємств. Включає розробку зручного користувацького інтерфейсу та реляційної бази даних.
Цільова аудиторія	Керівники підприємств, логісти, бухгалтери, менеджери по роботі з клієнтами
Терміни виконання проєкту	3 місяці
Бюджет проєкту	50 000 грн.

Суть проєкту полягає в тому, щоб спроєктувати та розробити вебсистему, яка автоматизує та оптимізує процеси обліку прийому та відвантаження продукції на підприємствах. Система надаватиме можливість ефективного моніторингу запасів, аналізу даних продажів, формування звітів та підтримки рішень управління. Ключовою особливістю є розробка інтуїтивно зрозумілого користувацького інтерфейсу та забезпечення надійного зберігання даних у

реляційній БД. Даний проєкт має на меті значно підвищити ефективність робочих процесів, зменшити людські помилки у обліку та сприяти кращому управлінню ресурсами підприємства.

5.2 Розроблення ринкової стратегія проєкту

Розроблення ринкової стратегії програмного продукту передбачає аналіз цільових груп потенційних клієнтів, оцінку готовності ринку до прийняття продукту, інтенсивність конкуренції та оцінку простоти виходу на ринок.

Таблиця 5.2. Опис цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції на ринку	Простота виходу на ринок
1.	Малі та середні підприємства з активним товарообігом	Висока	Високий	Конкуренція на ринку помірна, проте внесення унікальних рішень допоможе отримати місце на ринку	Розробка вебсистеми не потребує великих затрат, ймовірність використання цільовою аудиторією є доволі високою
2.	Великі підприємства зі складними логістичними ланцюгами	Середня	Середній		
3.	Інноваційні компанії з потребою в автоматизації процесів	Висока	Середній		
4.	Логістичні та	Висока	Високий		

	транспортні компанії				

Стратегія охоплення ринку базується на індивідуальному підході до кожної цільової групи, з акцентом на унікальних особливостях та вимогах кожної з них. Важливим елементом стратегії є розробка гнучкої системи, яка дозволить легко адаптуватися до специфічних потреб різних груп клієнтів. Маркетингові зусилля будуть зосереджені на демонстрації вигод та ефективності системи, використовуючи кейси успішного впровадження, діджитал-маркетинг та участь у галузевих заходах.

Основною метою ринкової стратегії є створення стабільного попиту на продукт, визнання на ринку та забезпечення довгострокового росту.

5.3 Розробка маркетингової програми стартап-проєкту

Таблиця 5.3. Визначення ключових переваг концепції потенційного продукту.

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Доступність системи з різних пристроїв	Можливість роботи зі системою з будь-якого пристрою з доступом до Інтернету	Платформна незалежність, висока адаптивність до різних пристроїв
2	Інтуїтивний інтерфейс	Простота використання, відсутність надлишкових елементів	Зручність та швидкість освоєння системи, висока користувацька лояльність
3	Ефективність	Швидкість обробки	Швидкість та точність відповідей,

	обробки та аналізу даних потрібної інформації	запитів, оптимізація алгоритмів інформації, оптимізація обчислювально складних алгоритмів	забезпечення актуальної інформації

Основною стратегією є висвітлення унікальних переваг продукту, що відрізняють його від конкурентів. Основні напрямки:

- Просування через цифрові канали. Використання соціальних медіа, блогів, партнерських платформ для досягнення широкої аудиторії.
- Демонстрація продукту на спеціалізованих заходах. Участь у виставках, конференціях, вебінарах для демонстрації можливостей системи.
- Співпраця з інфлюенсерами та експертами. Залучення відомих особистостей у галузі для підвищення довіри до продукту.

Таблиця 5.4. Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Невідомо	Невідомо	300 $ +	100-200$

Висновки. Отримана вебсистема для обліку прийому та відвантаження продукції має позитивні прогнози на ринку. Завдяки своїм унікальним особливостям система зустрічає потреби різних груп споживачів, включаючи малі та середні підприємства, стартапи, а також великі корпорації з складними логістичними потребами. Оптимізація обробки даних та адаптивність інтерфейсу сприяють збільшенню ефективності роботи працівників і зниженню витрат на технічне обслуговування, що робить систему конкурентоспроможною на ринку.

ВИСНОВКИ

За допомогою Laravel та Vue.js мною розроблено вебсистему, яка дозволяє зручно вести облік прийому та відвантаження продукції на підприємстві. Передбачена можливість заповнення та редагування інформації адміністратором. Реалізовані основні функції, включаючи статистику та звітність. Система розроблена з урахуванням сучасних стандартів безпеки та вимог бізнесу. Реалізація SPA на Vue.js забезпечує швидку взаємодію без перезавантаження сторінок. Виконано тестування розробленої системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PHP: Hypertext Preprocessor [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.php.net/manual/uk/intro-whatis.php (дата звернення 16.11.2023). – Назва з екрана.

2. Посібник по JavaScript [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://developer.mozilla.org/uk/docs/Web/JavaScript/Guide (дата звернення 16.11.2023). – Назва з екрана.

3. Laravel - PHP фреймворк для веб-майстрів [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://laravel.com/docs/8.x (дата звернення 16.11.2023). – Назва з екрана.

4. Vue.js - Прогресивний JavaScript фреймворк [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://vuejs.org/v2/guide/ (дата звернення 16.11.2023). – Назва з екрана.

5. Клієнтська валідація форм [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://developer.mozilla.org/uk/docs/Learn/Forms/Form_validation (дата звернення 16.11.2023). – Назва з екрана.

6. MySQL [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.mysql.com/ (дата звернення 16.11.2022). – Назва з екрана.

7. Введення в SQL [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.w3schools.com/sql/ (дата звернення 16.11.2023). – Назва з екрана.

8. Практика роботи з PHP та MySQL [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.tutorialspoint.com/php/php_and_mysql.htm (дата звернення 16.11.2023). – Назва з екрана.

9. Інтеграція PHP та JavaScript [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.geeksforgeeks.org/how-to-call-a-javascript-function-from-php/ (дата звернення 16.11.2023). – Назва з екрана.

10. Структура та принципи роботи реляційних баз даних [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.studytonight.com/dbms/database-models.php (дата звернення 16.11.2023). – Назва з екрана.

11. Найкращі практики розробки у PHP [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.phptherightway.com/ (дата звернення 16.11.2023). – Назва з екрана.

12. Основи Vue.js для початківців [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://vuejs.org/v2/guide/index.html (дата звернення 16.11.2023). – Назва з екрана.

13. Використання Ajax у PHP [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.w3schools.com/php/php_ajax_intro.asp (дата звернення 16.11.2023). – Назва з екрана.

14. Розробка веб-додатків з Laravel [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://laravel-news.com/your-first-laravel-application (дата звернення 16.11.2022). – Назва з екрана.

15. Керівництво по створенню RESTful API на Laravel [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.toptal.com/laravel/restful-laravel-api-tutorial (дата звернення 16.11.2022). – Назва з екрана.

ДОДАТКИ

Додаток А

```
CREATE DATABASE  IF NOT EXISTS `census` /*!40100 DEFAULT CHARACTER SET utf8mb4 */;

USE `census`;
-- MySQL dump 10.13  Distrib 8.0.26, for macos11 (x86_64)
--
-- Host: localhost    Database: census
-- ------------------------------------------------------
-- Server version  5.7.23

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `action_events`
--

DROP TABLE IF EXISTS `action_events`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `action_events` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `batch_id` char(36) COLLATE utf8mb4_unicode_ci NOT NULL,
  `user_id` bigint(20) unsigned NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `actionable_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `actionable_id` bigint(20) unsigned NOT NULL,
  `target_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `target_id` bigint(20) unsigned NOT NULL,
  `model_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `model_id` bigint(20) unsigned DEFAULT NULL,
```

```
  `fields` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `status` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'running',
  `exception` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `original` text COLLATE utf8mb4_unicode_ci,
  `changes` text COLLATE utf8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  KEY `action_events_actionable_type_actionable_id_index` (`actionable_type`,`actionable_id`),
  KEY `action_events_batch_id_model_type_model_id_index` (`batch_id`,`model_type`,`model_id`),
  KEY `action_events_user_id_index` (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=199 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=936;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `audits`
--

DROP TABLE IF EXISTS `audits`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `audits` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `user_type` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `user_id` bigint(20) unsigned DEFAULT NULL,
  `event` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `auditable_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `auditable_id` bigint(20) unsigned NOT NULL,
  `old_values` text COLLATE utf8mb4_unicode_ci,
  `new_values` text COLLATE utf8mb4_unicode_ci,
  `url` text COLLATE utf8mb4_unicode_ci,
  `ip_address` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `user_agent` varchar(1023) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `tags` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `audits_auditable_type_auditable_id_index` (`auditable_type`,`auditable_id`),
  KEY `audits_user_id_user_type_index` (`user_id`,`user_type`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=71 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=1638;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `cache`
--

DROP TABLE IF EXISTS `cache`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `cache` (
  `key` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `value` mediumtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `expiration` int(11) NOT NULL,
  UNIQUE KEY `cache_key_unique` (`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
AVG_ROW_LENGTH=16384;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `count_users`
--

DROP TABLE IF EXISTS `count_users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `count_users` (
  `created_at` timestamp NULL DEFAULT NULL,
  `customer_id` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `deleted_at` timestamp NULL DEFAULT NULL,
  `email` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `first_name` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `last_name` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `region_selected` int(11) DEFAULT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `role` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '0',
  `updated_at` timestamp NULL DEFAULT NULL,
  `warehouse_selected` int(11) DEFAULT NULL,
```

```
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_email_unique` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=24 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=2730;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `counts_report`
--

DROP TABLE IF EXISTS `counts_report`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `counts_report` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `trailer_number` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `transaction_number` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `customer_id` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `total_observed` int(11) NOT NULL,
  `total_qty_on_load` int(11) NOT NULL,
  `job_id` int(11) NOT NULL,
  `container_id` int(11) NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `failed_jobs`
--

DROP TABLE IF EXISTS `failed_jobs`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `failed_jobs` (
  `connection` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `exception` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `failed_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `payload` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `queue` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `uuid` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `failed_jobs_uuid_unique` (`uuid`)
) ENGINE=InnoDB AUTO_INCREMENT=138 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=19833;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_user`
--

DROP TABLE IF EXISTS `job_user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `job_user` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `job_id` int(10) unsigned NOT NULL,
  `user_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `job_user_job_id_foreign` (`job_id`),
  KEY `job_user_user_id_foreign` (`user_id`),
  CONSTRAINT `job_user_job_id_foreign` FOREIGN KEY (`job_id`) REFERENCES `jobs` (`id`) ON
DELETE CASCADE,
  CONSTRAINT `job_user_user_id_foreign` FOREIGN KEY (`user_id`) REFERENCES `count_users` (`id`)
ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=60 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=16384;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `jobs`
--

DROP TABLE IF EXISTS `jobs`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `jobs` (
  `created_at` timestamp NULL DEFAULT NULL,
```

```
  `description` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `end_date` date DEFAULT NULL,
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `note` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `status` int(11) NOT NULL DEFAULT '1',
  `updated_at` timestamp NULL DEFAULT NULL,
  `warehouse_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `jobs_warehouse_id_foreign` (`warehouse_id`),
  CONSTRAINT `jobs_warehouse_id_foreign` FOREIGN KEY (`warehouse_id`) REFERENCES `warehouses` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=39 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=16384;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `migrations`
--

DROP TABLE IF EXISTS `migrations`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `migrations` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `migration` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `batch` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=80 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=744;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `model_has_permissions`
--

DROP TABLE IF EXISTS `model_has_permissions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `model_has_permissions` (
  `permission_id` bigint(20) unsigned NOT NULL,
```

```
  `model_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `model_id` bigint(20) unsigned NOT NULL,
  PRIMARY KEY (`permission_id`,`model_id`,`model_type`),
  KEY `model_has_permissions_model_id_model_type_index` (`model_id`,`model_type`),
  CONSTRAINT `model_has_permissions_permission_id_foreign` FOREIGN KEY (`permission_id`) REFERENCES `permissions` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `model_has_roles`
--

DROP TABLE IF EXISTS `model_has_roles`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `model_has_roles` (
  `role_id` bigint(20) unsigned NOT NULL,
  `model_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `model_id` bigint(20) unsigned NOT NULL,
  PRIMARY KEY (`role_id`,`model_id`,`model_type`),
  KEY `model_has_roles_model_id_model_type_index` (`model_id`,`model_type`),
  CONSTRAINT `model_has_roles_role_id_foreign` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=16384;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `notifications`
--

DROP TABLE IF EXISTS `notifications`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `notifications` (
  `id` char(36) COLLATE utf8mb4_unicode_ci NOT NULL,
  `type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `notifiable_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `notifiable_id` bigint(20) unsigned NOT NULL,
  `data` text COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `read_at` timestamp NULL DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `notifications_notifiable_type_notifiable_id_index` (`notifiable_type`,`notifiable_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
AVG_ROW_LENGTH=615;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `nova_notifications`
--

DROP TABLE IF EXISTS `nova_notifications`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `nova_notifications` (
  `id` char(36) COLLATE utf8mb4_unicode_ci NOT NULL,
  `type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `notifiable_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `notifiable_id` bigint(20) unsigned NOT NULL,
  `data` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `read_at` timestamp NULL DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `deleted_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `nova_notifications_notifiable_type_notifiable_id_index` (`notifiable_type`,`notifiable_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `nova_pending_trix_attachments`
--

DROP TABLE IF EXISTS `nova_pending_trix_attachments`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `nova_pending_trix_attachments` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `draft_id` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `attachment` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `disk` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `nova_pending_trix_attachments_draft_id_index` (`draft_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `nova_trix_attachments`
--

DROP TABLE IF EXISTS `nova_trix_attachments`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `nova_trix_attachments` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `attachable_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `attachable_id` int(10) unsigned NOT NULL,
  `attachment` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `disk` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `url` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `nova_trix_attachments_attachable_type_attachable_id_index` (`attachable_type`,`attachable_id`),
  KEY `nova_trix_attachments_url_index` (`url`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `password_resets`
--

DROP TABLE IF EXISTS `password_resets`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `password_resets` (
  `created_at` timestamp NULL DEFAULT NULL,
  `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `token` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  KEY `password_resets_email_index` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
AVG_ROW_LENGTH=8192;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `permissions`
--

DROP TABLE IF EXISTS `permissions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `permissions` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `guard_name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `personal_access_tokens`
--

DROP TABLE IF EXISTS `personal_access_tokens`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `personal_access_tokens` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `tokenable_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `tokenable_id` bigint(20) unsigned NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `token` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL,
  `abilities` text COLLATE utf8mb4_unicode_ci,
  `last_used_at` timestamp NULL DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
```

```
  UNIQUE KEY `personal_access_tokens_token_unique` (`token`),
  KEY `personal_access_tokens_tokenable_type_tokenable_id_index` (`tokenable_type`,`tokenable_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `region_container`
--

DROP TABLE IF EXISTS `region_container`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `region_container` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `region_id` int(10) unsigned NOT NULL,
  `container_id` int(10) unsigned NOT NULL,
  `target_observation` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `region_container_region_id_foreign` (`region_id`),
  KEY `region_container_container_id_foreign` (`container_id`),
  CONSTRAINT `region_container_container_id_foreign` FOREIGN KEY (`container_id`) REFERENCES `tblcontainer` (`id`),
  CONSTRAINT `region_container_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=30 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `region_user`
--

DROP TABLE IF EXISTS `region_user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `region_user` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `region_id` int(10) unsigned NOT NULL,
  `user_id` int(10) unsigned NOT NULL,
```

```
  PRIMARY KEY (`id`),
  KEY `region_user_region_id_foreign` (`region_id`),
  KEY `region_user_user_id_foreign` (`user_id`),
  CONSTRAINT `region_user_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`) ON DELETE CASCADE,
  CONSTRAINT `region_user_user_id_foreign` FOREIGN KEY (`user_id`) REFERENCES `count_users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=34 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=3276;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `role_has_permissions`
--

DROP TABLE IF EXISTS `role_has_permissions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `role_has_permissions` (
  `permission_id` bigint(20) unsigned NOT NULL,
  `role_id` bigint(20) unsigned NOT NULL,
  PRIMARY KEY (`permission_id`,`role_id`),
  KEY `role_has_permissions_role_id_foreign` (`role_id`),
  CONSTRAINT `role_has_permissions_permission_id_foreign` FOREIGN KEY (`permission_id`) REFERENCES `permissions` (`id`) ON DELETE CASCADE,
  CONSTRAINT `role_has_permissions_role_id_foreign` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `roles`
--

DROP TABLE IF EXISTS `roles`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `roles` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `guard_name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=16384;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `sessions`
--

DROP TABLE IF EXISTS `sessions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `sessions` (
  `id` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `user_id` bigint(20) unsigned DEFAULT NULL,
  `ip_address` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `user_agent` text COLLATE utf8mb4_unicode_ci,
  `payload` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `last_activity` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `sessions_last_activity_index` (`last_activity`),
  KEY `sessions_user_id_index` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
AVG_ROW_LENGTH=1365;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `stocks`
--

DROP TABLE IF EXISTS `stocks`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `stocks` (
  `container_id` int(10) unsigned NOT NULL,
  `counter_name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `customer_id` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
```

```
  `ip_address` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `job_id` int(10) unsigned NOT NULL,
  `label_date` timestamp NULL DEFAULT NULL,
  `location` json DEFAULT NULL,
  `number_observations` int(11) DEFAULT NULL,
  `observation_date` date NOT NULL,
  `region_id` int(10) unsigned NOT NULL,
  `trailer_number` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `transaction_number` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `user_id` int(10) unsigned NOT NULL,
  `total_qty_on_load` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `stocks_container_id_foreign` (`container_id`),
  KEY `stocks_job_id_foreign` (`job_id`),
  KEY `stocks_region_id_foreign` (`region_id`),
  KEY `stocks_user_id_foreign` (`user_id`),
  CONSTRAINT `stocks_container_id_foreign` FOREIGN KEY (`container_id`) REFERENCES `tblcontainer` (`id`),
  CONSTRAINT `stocks_job_id_foreign` FOREIGN KEY (`job_id`) REFERENCES `jobs` (`id`),
  CONSTRAINT `stocks_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`),
  CONSTRAINT `stocks_user_id_foreign` FOREIGN KEY (`user_id`) REFERENCES `count_users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=172 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=1638;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblcensus`
--

DROP TABLE IF EXISTS `tblcensus`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblcensus` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `census_identity` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` text COLLATE utf8mb4_unicode_ci,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `max_trt` int(11) DEFAULT NULL,
```

```
  `min_trt` int(11) DEFAULT NULL,
  `note` text COLLATE utf8mb4_unicode_ci,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `tblcensus_census_identity_unique` (`census_identity`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=1820;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblcontainer`
--

DROP TABLE IF EXISTS `tblcontainer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblcontainer` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `container` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `color` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '#6574cd',
  `default` smallint(6) DEFAULT '0',
  `rank` smallint(6) DEFAULT '0',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=5461;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblconttype`
--

DROP TABLE IF EXISTS `tblconttype`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblconttype` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `census_id` int(10) unsigned NOT NULL,
```

```
  `container_id` int(10) unsigned NOT NULL,
  `book_stock` int(11) NOT NULL DEFAULT '0',
  `new_containers` int(11) NOT NULL DEFAULT '0',
  `scrap` int(11) NOT NULL DEFAULT '0',
  `repair` int(11) NOT NULL DEFAULT '0',
  `export` int(11) NOT NULL DEFAULT '0',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_census_ref_container_id` (`census_id`,`container_id`),
  KEY `idx_container_id` (`container_id`),
  CONSTRAINT `tblconttype_census_id_foreign` FOREIGN KEY (`census_id`) REFERENCES `tblcensus` (`id`) ON DELETE CASCADE,
  CONSTRAINT `tblconttype_container_id_foreign` FOREIGN KEY (`container_id`) REFERENCES `tblcontainer` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=819;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tbldailyproduction`
--

DROP TABLE IF EXISTS `tbldailyproduction`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tbldailyproduction` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `rack_date` date DEFAULT NULL,
  `number_racked` int(11) DEFAULT NULL,
  `number_observations` int(11) DEFAULT NULL,
  `region_id` int(10) unsigned DEFAULT NULL,
  `container_id` int(10) unsigned NOT NULL,
  `product_id` int(10) unsigned NOT NULL,
  `census_reference_id` int(10) unsigned NOT NULL,
  `note` text COLLATE utf8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  KEY `idx_census_reference_id` (`census_reference_id`),
  KEY `idx_container_id` (`container_id`),
  KEY `idx_product_id` (`product_id`),
  KEY `idx_region` (`region_id`),
```

```
  KEY `idx_all` (`region_id`,`container_id`,`product_id`,`census_reference_id`,`rack_date`),
  CONSTRAINT `tbldailyproduction_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=7068 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=73;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblerrorcode`
--

DROP TABLE IF EXISTS `tblerrorcode`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblerrorcode` (
  `error_code` varchar(1) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` text COLLATE utf8mb4_unicode_ci,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  UNIQUE KEY `tblerrorcode_error_code_unique` (`error_code`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=2730;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblholdprobs`
--

DROP TABLE IF EXISTS `tblholdprobs`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblholdprobs` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `census_reference_id` int(10) unsigned NOT NULL,
  `container_id` int(10) unsigned NOT NULL,
  `region_id` int(10) unsigned DEFAULT NULL,
  `product_id` int(10) unsigned NOT NULL,
  `trt_days` int(11) NOT NULL,
  `number_observations` int(11) DEFAULT NULL,
```

```
  `prod_date` date DEFAULT NULL,
  `number_racked` int(11) DEFAULT NULL,
  `VNOBS_PROD` int(11) DEFAULT NULL,
  `vnm1prob` double(53,30) DEFAULT NULL,
  `vnm2prob` double(53,30) DEFAULT NULL,
  `vndm1prob` double(53,30) DEFAULT NULL,
  `vndm2prob` double(53,30) DEFAULT NULL,
  `run_number` int(11) DEFAULT NULL,
  `statistics_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_all_trt_days` (`census_reference_id`,`run_number`,`container_id`,`product_id`,`trt_days`),
  KEY `idx_all_prod_date` (`census_reference_id`,`container_id`,`product_id`,`prod_date`),
  KEY `idx_census_reference_id` (`census_reference_id`),
  KEY `idx_container_id` (`container_id`),
  KEY `idx_product_id` (`product_id`),
  KEY `idx_region_id` (`region_id`),
  KEY `tblholdprobs_statistics_id_foreign` (`statistics_id`),
  CONSTRAINT `tblholdprobs_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`) ON DELETE CASCADE,
  CONSTRAINT `tblholdprobs_statistics_id_foreign` FOREIGN KEY (`statistics_id`) REFERENCES `tblresults` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=15180 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=89;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblobservation`
--

DROP TABLE IF EXISTS `tblobservation`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblobservation` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `region_id` int(10) unsigned DEFAULT NULL,
  `container_id` int(10) unsigned NOT NULL,
  `product_id` int(10) unsigned NOT NULL,
  `label_date` date DEFAULT NULL,
  `rack_date` date DEFAULT NULL,
  `observation_date` date DEFAULT NULL,
```

```
  `number_observations` int(11) DEFAULT NULL,
  `error_code` char(1) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `ignore_flag` tinyint(1) NOT NULL DEFAULT '0',
  `census_reference_id` int(10) unsigned NOT NULL,
  `note` text COLLATE utf8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  KEY `idx_census_reference_id` (`census_reference_id`),
  KEY `idx_container_id` (`container_id`),
  KEY `idx_product_id` (`product_id`),
  KEY `idx_all` (`census_reference_id`,`container_id`,`product_id`,`error_code`,`ignore_flag`,`rack_date`,`observation_date`),
  KEY `idx_null` (`census_reference_id`,`container_id`),
  KEY `idx_census_site_container` (`census_reference_id`,`container_id`),
  KEY `idx_region_id` (`region_id`),
  CONSTRAINT `tblobservation_product_id_foreign` FOREIGN KEY (`product_id`) REFERENCES `tblproduct` (`id`) ON DELETE CASCADE,
  CONSTRAINT `tblobservation_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=15339 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=146;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblproduct`
--

DROP TABLE IF EXISTS `tblproduct`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblproduct` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `product_code` varchar(10) COLLATE utf8mb4_unicode_ci NOT NULL,
  `product_description` text COLLATE utf8mb4_unicode_ci,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=1820;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```
-- Table structure for table `tblregions`
--

DROP TABLE IF EXISTS `tblregions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblregions` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=8192;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblresults`
--

DROP TABLE IF EXISTS `tblresults`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblresults` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `census_reference_id` int(10) unsigned NOT NULL,
  `region_id` int(10) unsigned DEFAULT NULL,
  `container_id` int(10) unsigned NOT NULL,
  `product_id` int(10) unsigned NOT NULL,
  `run_number` smallint(6) DEFAULT NULL,
  `number_observations` double(53,30) DEFAULT NULL,
  `M1AVGTRT` double(53,30) DEFAULT NULL,
  `M2AVGTRT` double(53,30) DEFAULT NULL,
  `M1STDTRT` double(53,30) DEFAULT NULL,
  `M2STDTRT` double(53,30) DEFAULT NULL,
  `M1POPEST` double(53,30) DEFAULT NULL,
  `M2POPEST` double(53,30) DEFAULT NULL,
  `M3POPEST` double(53,30) DEFAULT NULL,
  `accuracy` double(53,30) DEFAULT NULL,
  `accuracy2` double(53,30) DEFAULT NULL,
  `accuracy3` decimal(53,30) DEFAULT NULL,
```

```
`M1Weighting` double(53,30) DEFAULT NULL,
`M3TRT` double(53,30) DEFAULT NULL,
`static_stock` int(11) DEFAULT NULL,
`projected_stock` int(11) DEFAULT NULL,
`total_pop_estimateM1` int(11) DEFAULT NULL,
`total_pop_estimateM2` int(11) DEFAULT NULL,
`total_pop_estimateM3` int(11) DEFAULT NULL,
`start_date` date DEFAULT NULL,
`end_date` date DEFAULT NULL,
`note` text COLLATE utf8mb4_unicode_ci,
`created_at` timestamp NULL DEFAULT NULL,
`plot` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
`update_production_table` tinyint(1) DEFAULT '0',
`filter_observations` tinyint(1) DEFAULT '0',
`M1_0pct` double(53,2) DEFAULT NULL,
`M1_25pct` double(53,2) DEFAULT NULL,
`M1_50pct` double(53,2) DEFAULT NULL,
`M1_75pct` double(53,2) DEFAULT NULL,
`M1_100pct` double(53,2) DEFAULT NULL,
`M2_0pct` double(53,2) DEFAULT NULL,
`M2_25pct` double(53,2) DEFAULT NULL,
`M2_50pct` double(53,2) DEFAULT NULL,
`M2_75pct` double(53,2) DEFAULT NULL,
`M2_100pct` double(53,2) DEFAULT NULL,
`PM3_0pct` double(53,2) DEFAULT NULL,
`PM3_25pct` double(53,2) DEFAULT NULL,
`PM3_50pct` double(53,2) DEFAULT NULL,
`PM3_75pct` double(53,2) DEFAULT NULL,
`PM3_100pct` double(53,2) DEFAULT NULL,
`MaxTRT` int(11) DEFAULT NULL,
`MinTRT` int(11) DEFAULT NULL,
`M1_InCirc` double(53,2) DEFAULT NULL,
`M2_InCirc` double(53,2) DEFAULT NULL,
`M2b_InCirc` double(53,2) DEFAULT NULL,
`kM1_InCirc` double(53,2) DEFAULT NULL,
`kM2_InCirc` double(53,2) DEFAULT NULL,
`kM2b_InCirc` double(53,2) DEFAULT NULL,
`PM1_InCirc` double(53,2) DEFAULT NULL,
`PM2_InCirc` double(53,2) DEFAULT NULL,
`PM3_InCirc` double(53,2) DEFAULT NULL,
`updated_at` timestamp NULL DEFAULT NULL,
```

```
  PRIMARY KEY (`id`),
  KEY `idx_all` (`census_reference_id`,`container_id`,`product_id`),
  KEY `idx_census_reference_id` (`census_reference_id`),
  KEY `idx_container_id` (`container_id`),
  KEY `idx_product_id` (`product_id`),
  KEY `idx_region` (`region_id`),
  CONSTRAINT `tblresults_product_id_foreign` FOREIGN KEY (`product_id`) REFERENCES `tblproduct` (`id`) ON DELETE CASCADE,
  CONSTRAINT `tblresults_region_id_foreign` FOREIGN KEY (`region_id`) REFERENCES `tblregions` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=290 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=416;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblsite`
--

DROP TABLE IF EXISTS `tblsite`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblsite` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `site` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `region_id` int(11) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `tblsite_site_unique` (`site`),
  KEY `idx_description` (`description`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=1365;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tblstaticstock`
--

DROP TABLE IF EXISTS `tblstaticstock`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tblstaticstock` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `static_stock` int(11) NOT NULL,
  `container_id` int(10) unsigned NOT NULL,
  `census_reference_id` int(10) unsigned NOT NULL,
  `site_id` int(10) unsigned NOT NULL,
  `product_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_container_id_uniq` (`container_id`,`census_reference_id`,`site_id`,`product_id`),
  KEY `idx_census_reference_id` (`census_reference_id`),
  KEY `idx_container_id` (`census_reference_id`,`container_id`),
  KEY `idx_site_id` (`site_id`),
  KEY `tblstaticstock_product_id_foreign` (`product_id`),
  CONSTRAINT `tblstaticstock_census_reference_id_foreign` FOREIGN KEY (`census_reference_id`) REFERENCES `tblcensus` (`id`) ON DELETE CASCADE,
  CONSTRAINT `tblstaticstock_container_id_foreign` FOREIGN KEY (`container_id`) REFERENCES `tblcontainer` (`id`) ON DELETE CASCADE,
  CONSTRAINT `tblstaticstock_product_id_foreign` FOREIGN KEY (`product_id`) REFERENCES `tblproduct` (`id`),
  CONSTRAINT `tblstaticstock_site_id_foreign` FOREIGN KEY (`site_id`) REFERENCES `tblsite` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=94 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=197;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `users` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email_verified_at` timestamp NULL DEFAULT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
```

```
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_email_unique` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci AVG_ROW_LENGTH=16384;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Додаток Б

Розробка такої системи - це великий проєкт, тому я надаю лише базові уривки коду для системи на Laravel і Vue.js.

```
<?php

namespace App\Http\Controllers;

use App\Models\Container;
use App\Models\Job;
use App\Models\Region;
use App\Models\Stock;
use App\Models\User;
use App\Models\Warehouse;
use Carbon\Carbon;
use Carbon\CarbonPeriod;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\RedirectResponse;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
use Illuminate\Support\Facades\Request;
use Illuminate\Validation\ValidationException;
use Inertia\Inertia;
use Inertia\Response;
use Stevebauman\Location\Facades\Location;

class StocksController extends Controller
{
    /**
```

```
 * @return Response
 */
public function index()
{
    return Inertia::render('Stocks/Index', [
        'filters' => Request::all('search'),
        'jobs' => Auth::user()->jobs()
            ->where('status', 1)
            ->filter(Request::only('search'))
            ->filterByWarehouse(Auth::user()->warehouse_selected)
            ->get()
            ->transform(function ($job) {
                return [
                    'id' => $job->id,
                    'description' => $job->description,
                    'start_date' => $job->start_date,
                    'end_date' => $job->end_date,
                    'note' => $job->note,
                    'days' => $this->getDays($job->start_date, $job->end_date),
                    'showDays' => false,
                ];
            }),
        'customer' => Auth::user()->customer_id,
    ]);
}

/**
 * @return Response
```

```
     */
    public function location()
    {
        return Inertia::render('Stocks/Location', [
            'user' => Auth::user(),
            'regions' => Auth::user()->regions()->get(),
            'warehouses' => Auth::user()->warehouses()->get(),
        ]);
    }

    /**
     * @return RedirectResponse
     */
    public function locationStore()
    {
        Request::validate([
            'region' => 'required',
            'warehouse' => 'required',
        ]);

        if (Warehouse::find(Request::get('warehouse'))->region_id != Request::get('region')) {
            throw ValidationException::withMessages(['warehouse' => 'The warehouse field is required.']);
        }

        $user = User::find(Auth::user()->id);
        $user->region_selected = Request::get('region');
        $user->warehouse_selected = Request::get('warehouse');
        $user->save();
```

```
    return Redirect::route('stocks');
}

/**
 * @param Job $job
 * @param $size
 * @param $date
 * @param $trailer
 * @param $transaction
 * @return Response
 */
public function edit(Job $job, $size, $date, $trailer, $transaction)
{
    if (! Auth::user()->jobs()->get()->contains('id', $job->id)) {
        return Redirect::route('stocks')->with('error', 'Forbidden.');
    }

    $stockCountPeriod = $this->initStock($job->id, $size, $date, $trailer, $transaction);

    $containers = Container::orderBy('rank')->get();
    $regionContainers = Region::find(Auth::user()->region_selected)->containers()->get();
    foreach ($containers as $container) {
        $container->targetObservation = 0;
        foreach ($regionContainers as $regionContainer) {
            if ($regionContainer->id === $container->id && $regionContainer->pivot->target_observation) {
                $container->targetObservation = (int) $regionContainer->pivot->target_observation;
            }
```

```
        }
    }

    return Inertia::render('Stocks/Edit', [
        'stockCountPeriod' => $stockCountPeriod,
        'size' => $size,
        'jobId' => $job->id,
        'date' => $date,
        'trailer' => $trailer,
        'transaction' => $transaction,
        'customer' => Auth::user()->customer_id,
        'total' => $this->getTotal($job->id, $size, $date, $trailer, $transaction),
        'containers' => $containers,
    ]);
}

/**
 * @param Job $job
 * @param $date
 * @param $trailer
 * @param $transaction
 * @return JsonResponse
 */
public function checkIsCompleted(Job $job, $date, $trailer, $transaction): JsonResponse
{
    $stocks = Stock::where([
        'job_id' => $job->id,
        'observation_date' => $date,
```

```
            'trailer_number' => $trailer,
            'transaction_number' => $transaction,
        ])->get();

        return new JsonResponse(['completed' => count($stocks) > 0]);
    }

    /**
     * @return Response
     */
    public function updateAllSizes()
    {
        Request::validate([
            'observationDate' => 'required',
            'jobId' => 'required',
            'trailer' => 'required',
            'customer' => 'required',
            'counterName' => 'required',
            'transaction' => 'required',
        ]);

        $totalSaved = 0;
        $location = [];
        if (Location::get(Request::ip())) {
            $location = Location::get(Request::ip())->toArray();
        }
        foreach (Request::get('stockToSave') as $size => $stock) {
            foreach ($stock as $stockItem) {
```

```
            Stock::create([
                'job_id' => Request::get('jobId'),
                'region_id' => Auth::user()->region_selected,
                'user_id' => Auth::user()->id,
                'counter_name' => Request::get('counterName'),
                'label_date' => $stockItem['date'],
                'observation_date' => Request::get('observationDate'),
                'trailer_number' => Request::get('trailer'),
                'transaction_number' => Request::get('transaction'),
                'customer_id' => Request::get('customer'),
                'container_id' => $size,
                'number_observations' => $stockItem['localTotal'],
                'ip_address' => Request::ip(),
                'location' => json_encode($location),
                'total_qty_on_load' => Request::get('totalQty')[$size],
            ]);
            $totalSaved += $stockItem['localTotal'];
        }
    }

    return Inertia::render('Stocks/Finish', [
        'observationDate' => Request::get('observationDate'),
        'trailer' => Request::get('trailer'),
        'transaction' => Request::get('transaction'),
        'customer' => Request::get('customer'),
        'size' => 'All',
        'totalSaved' => $totalSaved,
        'containers' => Container::get(),
```

```
            'totalObservedAllSizes' => Request::get('totalObserved'),
            'qtyOnLoad' => Request::get('totalQty'),
        ]);
    }

    /**
     * @param int $jobId
     * @param string $size
     * @param string $date
     * @param string $trailer
     * @param string $transaction
     * @return array
     */
    public function initStock(int $jobId, string $size, string $date, string $trailer, string $transaction): array
    {
        $job = Job::findOrFail($jobId);
        $period = CarbonPeriod::create(Carbon::parse($job->start_date)->subYear(), Carbon::parse($job->start_date));
        $stockCountPeriod = [];
        $totals = $this->getTotalAll($jobId, $size, $date, $trailer, $transaction);
        foreach ($period as $day) {
            array_push($stockCountPeriod, [
                'year' => \substr($day->format('Y'), -1),
                'day' => \sprintf('%03d', $day->format('z') + 1),
                'total' => isset($totals[$day->format('Y-m-d H:i:s')]) ? $totals[$day->format('Y-m-d H:i:s')] : 0,
                'date' => $day->format('Y-m-d'),
            ]);
        }
```

```
    $day = Carbon::createFromTimestamp(86400);
    array_push($stockCountPeriod, [
        'year' => \substr($day->format('Y'), -1),
        'day' => \sprintf('%03d', $day->format('z') + 1),
        'total' => $this->getTotal($jobId, $size, $date, $trailer, $transaction, $day->format('Y-m-d')),
        'date' => $day->format('Y-m-d'),
        'other' => true,
    ]);

    return \array_reverse($stockCountPeriod);
}

/**
 * @param string $startDate
 * @param string $endDate
 * @return array
 */
public function getDays(string $startDate, string $endDate): array
{
    $period = CarbonPeriod::create($startDate, $endDate);
    $stockCountPeriod = [];
    foreach ($period as $date) {
        if ($date->isFuture()) {
            continue;
        }
        array_push($stockCountPeriod, [
            'dateToDisplay' => $date->format('l d F, Y'),
            'date' => $date->format('Y-m-d'),
```

```
            'current' => $date->isToday(),
        ]);
    }

    return $stockCountPeriod;
}

/**
 * @param int $jobId
 * @param string $size
 * @param string $observationDate
 * @param string $trailer
 * @param string $transaction
 * @param string|null $day
 * @return int
 */
public function getTotal(
    int $jobId,
    string $size,
    string $observationDate,
    string $trailer,
    string $transaction,
    ?string $day = null
): int {
    $condition = [
        'job_id' => $jobId,
        'trailer_number' => $trailer,
        'transaction_number' => $transaction,
```

```
            'container_id' => $size,
            'observation_date' => $observationDate,
        ];
        if ($day) {
            $condition['label_date'] = $day;
        }

        return Stock::where($condition)->sum('number_observations');
    }

    public function getTotalAll(
        int $jobId,
        string $size,
        string $observationDate,
        string $trailer,
        string $transaction
    ) {
        $result = [];
        $condition = [
            'job_id' => $jobId,
            'trailer_number' => $trailer,
            'transaction_number' => $transaction,
            'container_id' => $size,
            'observation_date' => $observationDate,
        ];

        $stocks = Stock::where($condition)->get();
```

```
    foreach ($stocks as $stock) {
        if (isset($result[$stock['label_date']])) {
            $result[$stock['label_date']] += $stock['number_observations'];
        } else {
            $result[$stock['label_date']] = $stock['number_observations'];
        }
    }

    return $result;
}

/**
 * @param int $jobId
 * @param string $observationDate
 * @return JsonResponse
 */
public function suggestedTrailerNumbers(int $jobId, string $observationDate): JsonResponse
{
    $numbers = Stock::select(['trailer_number', 'transaction_number'])->where([
        'job_id' => $jobId,
        'observation_date' => $observationDate,
    ])->distinct()->get();
    $formattedNumbers = [];
    foreach ($numbers as $number) {
        $formattedNumbers[$number['trailer_number']][] = [
            'transactionNumber' => $number['transaction_number'],
            'createdAt' => Stock::select('created_at')->where([
                'job_id' => $jobId,
```

```
                'observation_date' => $observationDate,
                'trailer_number' => $number['trailer_number'],
                'transaction_number' => $number['transaction_number'],
            ])->orderBy('id', 'ASC')->limit(1)->first()->created_at,
        ];
    }

    return new JsonResponse($formattedNumbers);
  }
}
<template>
  <div>
    <Head title="Dashboard" />
    <h1 class="font-bold text-3xl mb-2">
      Dashboard
    </h1>
    <div class="mb-4 flex justify-between items-center">
      <DashboardFilter class="mt-6 mr-4" @reset="reset">
        <div>
          <text-input v-model="form.jobName" class="m-w-160" :type="'text'" label="Job Name"/>
        </div>
        <div class="ml-3">
          <text-input v-model="form.observationDate" class="m-w-160" :type="'date'" label="Observation
Date"/>
        </div>
        <div class="ml-3 w-full lg:w-1/3">
          <label class="form-label">Container:</label>
          <select v-model="form.containerId" class="form-select">
            <option :value="container.container" v-for="container in containers">
```

```
                {{ container.description }}
              </option>
            </select>
          </div>
          <div class="ml-3 pr-6 w-full lg:w-1/3">
            <label class="form-label">Status:</label>
            <select v-model="form.status" class="form-select">
              <option value="1" selected>Opened</option>
              <option value="0">Closed</option>
            </select>
          </div>
        </DashboardFilter>

        <button class="btn-indigo focus:bg-indigo-600 hover:bg-indigo-600" @click="autoRefresh"
            :disabled="refreshProcess === true">
          <span v-show="refreshProcess">
            Refreshing...
          </span>
          <span v-show="!refreshProcess">
            <span>Auto Refresh:</span>
            <span class="hidden md:inline" v-show="autorefresh"> On</span>
            <span class="hidden md:inline" v-show="!autorefresh"> Off</span>
          </span>
        </button>
      </div>
      <div class="bg-white rounded shadow overflow-x-auto">
        <table class="w-full whitespace-no-wrap">
          <tr class="text-left font-bold">
```

```
    <th class="px-3 py-4">Job</th>
    <th class="px-3 py-4">Start Date</th>
    <th class="px-3 py-4">End Date</th>
    <th class="px-3 py-4">Warehouse</th>
    <th class="px-3 py-4">Region</th>
    <th class="px-3 py-4">Observation Date</th>
    <th class="px-3 py-4">Trailer</th>
    <th class="px-3 py-4">Trans No.</th>
    <th class="px-3 py-4">Container</th>
    <th class="px-3 py-4">Observed</th>
    <th class="px-3 py-4">Qty On Load</th>
    <th class="px-3 py-4">Target%</th>
    <th class="px-3 py-4">Count%</th>
  </tr>
  <tr v-for="job in jobs.data" :key="job.id" @click="goToDetail(job.id)" class="hover:bg-gray-100
focus-within:bg-gray-100 cursor-pointer">
    <td class="border-t">
      <span class="px-3 py-4 text-sm flex items-center">
        {{ job.job }}
      </span>
    </td>
    <td class="border-t">
      <span class="px-3 py-4 text-sm flex items-center">
        {{ formatDate(job.start_date) }}
      </span>
    </td>
    <td class="border-t">
      <span class="px-3 py-4 text-sm flex items-center">
```

```
      {{ formatDate(job.end_date) }}
    </span>
  </td>
  <td class="border-t">
    <span class="px-3 py-4 text-sm flex items-center">
      {{ job.warehouse }}
    </span>
  </td>
  <td class="border-t">
    <span class="px-3 py-4 text-sm flex items-center">
      {{ job.region }}
    </span>
  </td>
  <td class="border-t">
    <span class="px-3 py-4 text-sm flex items-center">
      {{ formatDate(job.observation_date) }}
    </span>
  </td>
  <td class="border-t">
    <span class="px-3 py-4 text-sm flex items-center">
      {{ job.trailer }}
    </span>
  </td>
  <td class="border-t">
    <span class="px-3 py-4 text-sm flex items-center">
      {{ job.transaction_number }}
    </span>
  </td>
```

```
<td class="border-t">
  <span class="px-3 py-4 text-sm flex items-center">
    {{ job.container }}
  </span>
</td>
<td class="border-t">
  <span class="px-3 py-4 text-sm flex items-center">
    {{ job.observed }}
  </span>
</td>
<td class="border-t">
  <span class="px-3 py-4 text-sm flex items-center">
    {{ job.total_qty_on_load }}
  </span>
</td>
<td class="border-t">
  <span class="px-3 py-4 text-sm flex items-center">
    {{ job.target_observation }}%
  </span>
</td>
<td class="border-t">
  <span class="px-3 py-4 text-sm flex items-center">
    {{ job.perc }}%
  </span>
</td>
<td class="border-t">
  <span class="px-3 py-4 text-sm flex items-center">
    <icon name="cheveron-right" class="block w-6 h-6 fill-gray-400"/>
```

```
                    </span>
                </td>
            </tr>
            <tr v-if="jobs.data.length === 0">
                <td class="border-t px-6 py-4" colspan="4">No observations found.</td>
            </tr>
        </table>
        <pagination class="p-4" v-if="jobs.data.length !== 0" :links="jobs.links"/>
    </div>
  </div>
</template>

<script>
import { Head, Link } from '@inertiajs/inertia-vue3'
import Layout from '@/Shared/Layout'
import moment from 'moment'
import axios from "axios";
import Pagination from '@/Shared/Pagination'
import Icon from '@/Shared/Icon'
import JobsFilter from '@/Shared/JobsFilter'
import TextInput from '@/Shared/TextInput'
import DashboardFilter from "../../Shared/DashboardFilter";
import mapValues from "lodash/mapValues";
import throttle from "lodash/throttle";
import pickBy from "lodash/pickBy";
import SelectInput from '@/Shared/SelectInput';

export default {
```

```
layout: Layout,
filters: {
  formatDate: function (date) {
    if (date) {
      return moment(String(date)).format('ddd, DD MMMM YYYY')
    }
  },
},
components: {
  DashboardFilter,
  Pagination,
  Icon,
  JobsFilter,
  TextInput,
  SelectInput,
  Head,
  Link
},
props: {
  jobs: [Array, Object],
  containers: [Array, Object],
  filters: [Array, Object],
},
data() {
  return {
    'autorefresh': false,
    'jobsData': [],
    'timer': '',
```

```
            'refreshProcess': false,
            'status': 1,
            'observationDate': '',
            'form': {
                jobName: this.filters.jobName,
                observationDate: this.filters.observationDate,
                containerId: this.filters.containerId,
                status: this.filters.status,
            }
        }
    },
    watch: {
        form: {
            handler: throttle(function () {
                let query = pickBy(this.form)
                this.$inertia.replace(this.route('dashboard', Object.keys(query).length ? query : {remember: 'forget'}))
            }, 150),
            deep: true,
        },
    },
    methods: {
        formatDate(date) {
            if (date) {
                return moment(String(date)).format('ddd, DD MMMM YYYY')
            }
        },
        reset() {
            this.form = mapValues(this.form, () => null)
```

```
        },
        autoRefresh() {
            this.autorefresh = !this.autorefresh;
            if (!this.autorefresh) {
                clearInterval(this.timer);
            } else {
                this.timer = setInterval(this.getJobs, this.$page.props.auth.user.dashboardRefreshInterval);
            }
        },
        getJobs() {
            let query = pickBy(this.form)
            this.$inertia.replace(this.route('dashboard', Object.keys(query).length ? query : {remember: 'forget'}))
        },
        cancelAutoUpdate() {
            clearInterval(this.timer);
        },
        goToDetail(id) {
            this.$inertia.get(this.route('dashboard.details', {'id': id}));
        }
    },
    beforeUnmount() {
        this.cancelAutoUpdate();
    }
}

</script>

<style>
```

```
.m-w-160 {
    min-width: 160px;
}
.date-height {
    height: 42px;
}
</style>
```