Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

**КУРСОВА РОБОТА**

з дисципліни «Веб-технології та веб-дизайн»

Тема «Розробка веб-додатка з ведення інвентаризації»

Студента 3 курсу АІ-221 групи

Спеціальності 122 – «Комп’ютерні науки»

Попазова П.І.

Керівник: д. Червоненко П.П.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Одеса – 2024 рік

ЗМІСТ

[ЗАВДАННЯ 4](#_Toc183084704)

[НА КУРСОВУ РОБОТУ 4](#_Toc183084705)

[АНОТАЦІЯ 5](#_Toc183084706)

[ВСТУП 6](#_Toc183084707)

[1 АКТУАЛЬНІСТЬ, АНАЛІЗ АНАЛОГІВ 7](#_Toc183084708)

[1.1 Актуальність веб-додатка 7](#_Toc183084709)

[1.2 Аналіз додатка-аналога Zoho Inventory 8](#_Toc183084710)

[1.3 Аналіз додатка-аналога Uspacy 10](#_Toc183084711)

[1.4 Бачення програмного продукту 11](#_Toc183084712)

[2 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-САЙТУ 13](#_Toc183084713)

[2.1 Реєстрація, авторизація користувача 13](#_Toc183084714)

[2.2 Бізнес-логіка застосунка 17](#_Toc183084715)

[3 ПРОЕКТУВАННЯ ВЕБ-САЙТУ 20](#_Toc183084716)

[3.1 Мета та завдання ІС 20](#_Toc183084717)

[3.2 Кінцева аудиторія ІС 20](#_Toc183084718)

[3.3 Функціональні вимоги ІС 21](#_Toc183084719)

[3.4 Моделювання прецедентів 24](#_Toc183084720)

[3.5 Нефункціональні вимоги 27](#_Toc183084721)

[3.6 Architecture Constraints 28](#_Toc183084722)

[3.7 UI View 30](#_Toc183084723)

[3.8 Logical View 37](#_Toc183084724)

[3.9 Deployment View 39](#_Toc183084725)

[3.10 Design View 40](#_Toc183084726)

[3.11 Process View 40](#_Toc183084727)

[3.12 Data View 42](#_Toc183084728)

[3.13 Interface View 45](#_Toc183084729)

[3.14 Security View 47](#_Toc183084730)

[3.15 Infrastructure View 47](#_Toc183084731)

[3.16 Delivery Strategy View 48](#_Toc183084732)

[4 РЕАЛІЗАЦІЯ ВЕБ САЙТУ 50](#_Toc183084733)

[5 ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ВЕБ-САЙТУ. 51](#_Toc183084734)

[ВИСНОВКИ 52](#_Toc183084735)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 53](#_Toc183084736)

[ДОДАТОК А – Сповіщення про недостатню кількість товарів 54](#_Toc183084737)

Національний університет «Одеська політехніка»

Інститут компʼютерних систем

Кафедра інформаційних систем

# ЗАВДАННЯ

# НА КУРСОВУ РОБОТУ

студентці Попазову Петру Івановичу група АІ-221

1. Тема роботи «Розробка веб-додатка з продажу техніки»

2. Термін здачі студентом закінченої роботи 09.06.2024

3. Початкові дані до проекту (роботи). Користувачі мають змогу зарєструватися, увійти у системиу та виконати двуфакторну авторизаці. Користувачі додають товари до списку, редагують, видаляють. Отримують сповіщення про занизький рівень товару на счкладі. Виконують трансфери між складами. Оримують детальну аналітику про продажі та бачать прогнозовані продажі.

4. Зміст розрахунково-пояснювальної записки: Розділ 1 – Актуальність, аналіз аналогів. Розділ 2 – Алгоритмічне забезпечення веб-сайту. Розділ 3 – Проектування веб-сайту. Розділ 4 – реалізація веб сайту. Розділ 5 – забезпечення якості веб-сайту.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання видано 01.09.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Завдання прийнято до виконання 10.09.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

# АНОТАЦІЯ

ABSTRACT

# ВСТУП

У сучасному бізнес-середовищі, де конкуренція та технології швидко змінюються, розробка веб-застосунку для інвентаризації стає надзвичайно важливою. Такий застосунок автоматизує процеси управління запасами, що зменшує ризик помилок і економить час. Він забезпечує доступ до актуальної інформації про запаси в реальному часі, що дозволяє приймати обґрунтовані рішення. Мобільний доступ до даних та гнучкість у масштабуванні відповідають потребам бізнесу, а покращене обслуговування клієнтів і зменшення витрат підвищують конкурентоспроможність.

Веб-застосунки для інвентаризації мають ряд важливих функцій, які роблять їх незамінними для ефективного управління запасами. По-перше, вони забезпечують відстеження запасів, що дозволяє моніторити кількість товарів на складі в реальному часі та їхню історію руху. По-друге, функція управління замовленнями автоматизує процеси створення, обробки та виконання замовлень, включаючи повернення та зміни. Також важливою є аналітика та звітність, яка генерує звіти про запаси, продажі та прогнози попиту, що допомагає приймати обґрунтовані рішення. Крім того, такі веб-застосунки дозволяють зберігати інформацію про постачальників, а саме – контактні дані та історію закупівель.

Мобільний доступ до системи дозволяє управляти запасами на ходу, а функція керування цінами допомагає оптимізувати цінову політику, включаючи знижки та акції. Нарешті, захист даних забезпечує безпеку інформації, шифрування даних та регулярне резервне копіювання, що є критично важливим для захисту від втрат.

Всі вищенаведені функції в комплексі дозволяють компаніям ефективно управляти своїми запасами, оптимізувати бізнес-процеси та підвищувати задоволеність клієнтів.

# 1 АКТУАЛЬНІСТЬ, АНАЛІЗ АНАЛОГІВ

## 1.1 Актуальність веб-додатка

Розробка веб-застосунку для інвентаризації залишається надзвичайно актуальною в сучасному бізнес-середовищі, яке швидко змінюється під впливом сучасних технологій. У 2024 році компанії стикаються з необхідністю оптимізації своїх процесів управління запасами, щоб залишатися конкурентоспроможними та задовольняти потреби клієнтів.

Однією з основних причин для розробки такого застосунку є підвищення ефективності. Автоматизація процесів інвентаризації дозволяє компаніям значно зменшити час, витрачений на ручне введення даних і обробку інформації, що не тільки знижує ризики помилок, але й дозволяє співробітникам зосередитися на більш важливих завданнях, таких як стратегічне планування та покращення обслуговування клієнтів.

Крім того, веб-застосунки для інвентаризації надають можливість моніторингу запасів у реальному часі, що дозволяє компаніям оперативно реагувати на зміни в попиті та забезпечувати наявність товарів на складі, що є особливо важливо в умовах нестабільного ринку, де швидкість реакції може стати ключовим фактором успіху. Застосунки також дозволяють генерувати детальні звіти та аналітику, що сприяє прийняттю обґрунтованих рішень на всіх рівнях управління.

В умовах зростаючої конкуренції та змінюваних споживчих уподобань компанії також повинні звертати увагу на можливості масштабування своїх систем. Веб-застосунки легко адаптуються до зростання бізнесу, дозволяючи додавати нові функції або інтегрувати нові платформи за потреби, що забезпечує гнучкість, необхідну для швидкого реагування на зміни на ринку.

Веб-застосунки для інвентаризації використовуються різними типами організацій у багатьох галузях. По-перше, роздрібні магазини—як малі, так і великі мережі—застосовують ці рішення для управління запасами товарів, обробки замовлень і відстеження продажів. Оптові постачальники потребують ефективних інструментів для моніторингу великих обсягів товарів і управління постачаннями. Виробничі підприємства контролюють сировину та готову продукцію, а також запаси на складах. Логістичні компанії використовують ці системи для моніторингу вантажів, оптимізації складу та управління транспортними процесами.

У сфері електронної комерції такі застосугки використовують для обробки замовлень та генерації електронних чеків. Медичні заклади, такі як лікарні та клініки, контролюють рівень медичних запасів. Готелі та ресторани використовують ці рішення для управління запасами продуктів харчування, напоїв та інших матеріалів. Будівельні компанії потребують інструментів для моніторингу матеріалів та обладнання, необхідного для виконання проектів.

Також університети та освітні установи можуть використовувати інвентаризаційні рішення для управління запасами навчальних матеріалів і обладнання. Навіть неприбуткові організації застосовують ці рішення для управління запасами товарів, призначених для благодійних програм.

Веб-застосунки для інвентаризації є універсальними інструментами, що підходять для різних галузей і типів бізнесу, допомагаючи покращувати ефективність управління запасами та оптимізувати бізнес-процеси.

## 1.2 Аналіз додатка-аналога Zoho Inventory

Zoho Inventory — це потужне рішення для управління запасами, яке входить до складу екосистеми Zoho, розробленої для допомоги бізнесам у відстеженні, управлінні та оптимізації їхніх запасів. Zoho Inventory дозволяє відстежувати рівні запасів у режимі реального часу, що допомагає компаніям уникати переповнення складів або нестачі товарів. Користувачі можуть налаштовувати автоматичні сповіщення про низькі запаси, щоб вчасно поповнювати запаси.

Платформа дозволяє управляти як закупівлею, так і продажем товарів, включаючи створення та відстеження замовлень постачальників і клієнтів. Інтерфейс дозволяє легко управляти історією замовлень, що забезпечує повний огляд діяльності. Zoho Inventory інтегрується з популярними платформами електронної комерції, такими як Shopify, Amazon, eBay та Etsy. Це забезпечує синхронізацію запасів та автоматизацію процесу продажу. Користувачі можуть легко управляти замовленнями та відстежувати продажі на різних каналах з одного місця. Платформа дозволяє керувати кількома складами, що корисно для бізнесів з різними локаціями. Користувачі можуть відстежувати товарні запаси в різних місцях та оптимізувати розподіл товарів. Zoho Inventory надає розширені можливості для створення звітів та аналітики. Користувачі можуть отримувати статистику про продажі, запаси, замовлення та постачальників, що дозволяє приймати обґрунтовані рішення на основі даних. Інструменти автоматизації, такі як автоматичне створення замовлень, допомагають зменшити трудозатрати і підвищити ефективність. Користувачі можуть налаштовувати автоматичні робочі процеси відповідно до своїх потреб.

Переваги Zoho Inventory:

* комплексна функціональність;
* інтуїтивно зрозумілий інтерфейс;
* автоматизація процесів;
* гнучка інтеграція;
* потужні аналітичні інструменти;
* підтримка кількох валют і мов.

Недоліки Zoho Inventory

* обмежена безкоштовна версія;
* складність у налаштуванні;
* вартість підписки;
* обмеження у кастомізації.

Отже, Zoho Inventory є потужним інструментом для управління запасами, який підходить для бізнесів різного розміру. Його багатофункціональність, простота використання та інтеграція з іншими платформами роблять його привабливим вибором для компаній, які прагнуть оптимізувати свої процеси управління запасами, підвищити ефективність і забезпечити якісне обслуговування клієнтів.

## 1.3 Аналіз додатка-аналога Uspacy

Uspacy — це український інноваційний веб-застосунок, розроблений для ефективного управління запасами та обліку товарів у бізнесах, що займаються роздрібною торгівлею, оптовими постачаннями та електронною комерцією. Цей інструмент стає все більш популярним серед підприємців завдяки своїй функціональності та простоті використання, що дозволяє значно спростити процеси управління запасами.

Однією з основних функцій Uspacy є управління запасами в режимі реального часу. Це означає, що користувачі можуть миттєво відстежувати наявність товарів на складі, що допомагає уникнути проблем, пов'язаних з нестачею товарів або переповненням складів. Система автоматично оновлює інформацію про запаси, що забезпечує точність даних і дозволяє підприємствам бути в курсі своїх ресурсів.

Автоматизація процесів є ще однією важливою перевагою Uspacy. Застосунок дозволяє автоматично створювати замовлення, обробляти постачання і генерувати звіти, що значно зменшує трудозатрати та підвищує ефективність бізнес-процесів. Автоматизація також допомагає зменшити ризик помилок, які можуть виникнути під час ручного введення даних.

Аналіз даних — це ще одна ключова функція, яка робить Uspacy корисним інструментом для прийняття обґрунтованих рішень. Застосунок надає можливості для створення детальних звітів та аналітики, що дозволяє підприємствам отримувати цінну інформацію про продажі, запаси та ринкові тенденції. Це, в свою чергу, сприяє оптимізації управління запасами та підвищенню прибутковості.

Uspacy також підтримує інтеграцію з іншими платформами та сервісами, такими як електронна комерція та системи управління бізнесом за допомогою REST API.

Переваги Uspacy:

* інтуїтивно зрозумілий інтерфейс;
* управління запасами в реальному часі;
* автоматизація процесів, що знижує трудозатрати та підвищує ефективність;
* аналітика та звітність;
* інтеграція з іншими платформами;
* гнучкість.

Недоліки Uspacy:

* обмежена функціональність безкоштовної версії;
* необхідність в Інтернет-з'єднанні;
* потреба в навчанні для більш складних функцій.

Отже, Uspacy є потужним інструментом для управління запасами, який допомагає бізнесам оптимізувати їхні процеси, зменшити витрати та підвищити ефективність. Його функціональність, простота використання та можливості інтеграції роблять його привабливим вибором для компаній, які прагнуть вдосконалити управління запасами та забезпечити якісне обслуговування клієнтів.

## 1.4 Бачення програмного продукту

У книзі «Перетинаючи прірву» маркетолога Джеффрі А. Мур (Crossing the Chasm, Geoffrey A. Moore) описується, як формалізація бізнес (часткова) вимог може привабити перших прихильників, які з ентузіазмом приймуть продукт і можуть самі почати рекламувати та переконати або навчити інших людей користуватися цим продуктом. Існує багато різних шаблонів опису бачення продукту. Один з них описано у вказаній книзі у вигляді 7-ми елементів [3]:

* For: «target customer» (Для кого: цільовий споживач);
* Who: «needs» (Хто: потреби);
* The: «product name» ( назва продукту);
* Is: «product category» (Є: категорія продукту);
* That: «product benefit. Reason to buy» (Що: переваги продукту, причина
* купувати);
* Unlike: «competitors» (на відміну: продукти-конкуренти);
* Our product: «differentiation or value proposition» (наш продукт: позитивні
* відмінності від інших продуктів або суттєві пропозиції).

Бачення програмного продукту: «FOR споживача, який є повнолітньою працюючою особою з середнім або вище середнього рівнем доходу, WHO має потребу в ефективному управлінні запасами, точній інформації про наявні товари та оптимізації процесів обліку, THE Easy Store App IS веб-додаток THAT розроблений для забезпечення користувачів зручними інструментами для ведення обліку товарів, відстеження запасів та автоматизації процесів замовлень. UNLIKE інші програмні продукти, такі як Zoho Inventory та Uspacy, які є повністю комерційними рішеннями та мають заскладний інтерфейс користувача, OUR продукт має повністю безоплатний доступ до усіх функцій та має простий, інтуїтивно зрозумилий інтерфейс користувача».

# 2 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-САЙТУ

## 2.1 Реєстрація, авторизація користувача

Детальний опис алгоритма реєстрації користувачів.

Вхідні параметри:

* email: рядок, що представляє email користувача;
* password: рядок, що представляє пароль користувача.

Вихідні параметри:

* унікальний ідентифікатор користувача (Integer userId) після його успішного збереження в базі даних.

Блок-схема алгоритму реєстрації (рис. 2.1):

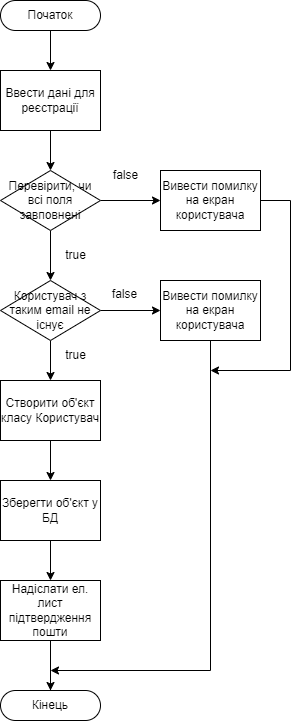


Рисунок 2.1 – Реєстрація користувача

Алгоритм реєстрації:

* перевірка і валідація вхідних даних: перед збереженням користувача виконується валідація введених полів (email та пароль), що гарантує коректність введених даних;
* хешування паролю: пароль користувача хешується за допомогою PasswordEncoder для забезпечення безпеки зберігання;
* створення об'єкта користувача: на основі вхідних даних створюється об'єкт AppUser;
* збереження користувача в базі даних: користувач зберігається через репозиторій appUserRepository;
* надсилання верифікаційного листа: генерується токен активації, який надсилається користувачеві для підтвердження акаунта через email;
* логування реєстрації: генерується повідомлення про реєстрацію, яке надсилається іншому мікросервісу через userRegisteredProducer.

Трудомісткість (Time Complexity):

* хешування пароля: O(1) — операція хешування паролю займає постійний час;
* збереження користувача в базі даних: O(1) — збереження нового запису до бази даних;
* генерація токена активації: O(n) — де n — це довжина токена, яку можна вважати константною, отже, складність генерації також близька до O(1);
* надсилання email: O(1) — це виклик асинхронного процесу, тому не враховується в основній логіці;

Залежність використання пам'яті (Space Complexity):

* збереження паролю: O(1) — для збереження хешу пароля;
* об'єкт користувача: O(1) — структура даних користувача займає постійний обсяг пам'яті;
* токен активації: O(1) — токен займає фіксовану кількість пам'яті;
* база даних: O(n), де n — кількість користувачів, оскільки кожен новий користувач потребує місця для зберігання даних у базі;
* загальна залежність від пам'яті — O(1) для кожного окремого користувача.

Діаграма (рис. 2.2) демонструє архітектуру аутентифікації в додатку на основі Spring Security з використанням JWT (JSON Web Token).

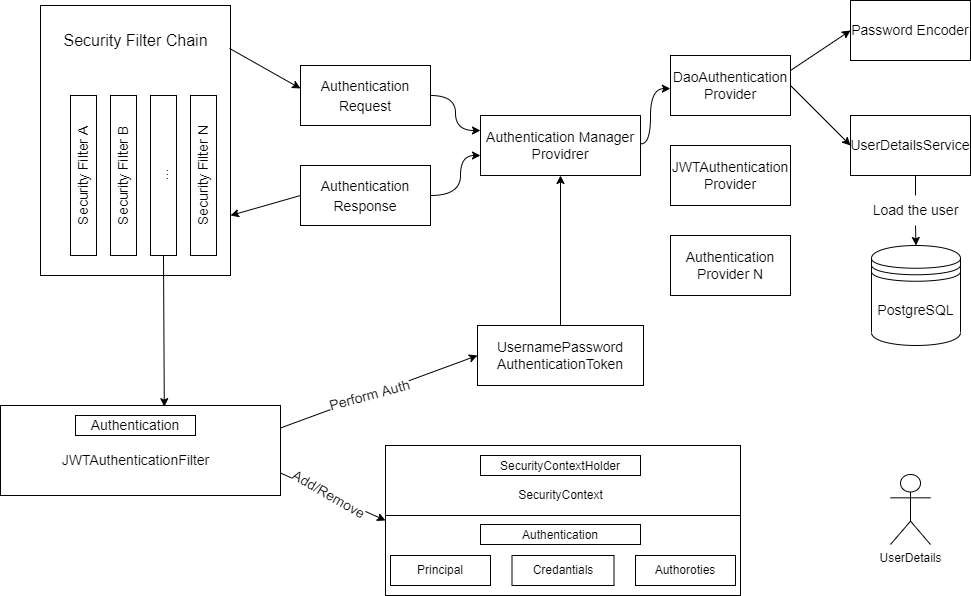


Рисунок 2.2 – Архітектура аутентифікації у Spring Security

Security Filter Chain (Ланцюг фільтрів безпеки) складається з кількох фільтрів безпеки (Security Filter A, B, ..., N). Фільтри взаємодіють з запитами, зокрема, перевіряють наявність токену або валідність аутентифікаційних даних.

Authentication Request & Authentication Response (Запит та відповідь аутентифікації): - запит аутентифікації передається через фільтри до менеджера аутентифікації. Відповідь надсилається назад після успішної або неуспішної аутентифікації.

Authentication Manager Provider (Менеджер аутентифікації) – це ключовий елемент, що делегує аутентифікацію до різних провайдерів (DaoAuthenticationProvider, JWTAuthenticationProvider, і AuthenticationProvider N).

DaoAuthenticationProvider – вкористовує Password Encoder та UserDetailsService для аутентифікації на основі даних із бази. Після отримання даних користувача з бази PostgreSQL через UserDetailsService перевіряє пароль користувача за допомогою Password Encoder.

JWTAuthenticationProvider відповідає за перевірку та обробку JWT-токену. Якщо токен валідний, користувач вважається аутентифікованим.

Authentication Providers (N): можливі інші провайдери аутентифікації для різних типів методів (LDAP, OAuth і т.д.).

UsernamePasswordAuthenticationToken – Використовується для передачі аутентифікаційних даних (логін і пароль) від користувача до менеджера аутентифікації.

JWTAuthenticationFilter – фільтр, який перехоплює запити, що містять JWT, і передає їх для подальшої обробки. Встановлює користувача у SecurityContext після успішної аутентифікації.

SecurityContextHolder – центральний компонент, який зберігає контекст безпеки для поточного потоку. Включає інформацію про аутентифікованого користувача (Principal), його облікові дані (Credentials) та права доступу (Authorities).

За авторизацію уористувача відповідає метод authenticate. Опис алгоритму:

Вхідні параметри: request (типу AuthUserRequest): містить email та пароль користувача для аутентифікації. Вихідні параметри: AuthenticationResponse: об'єкт з JWT-токеном, якщо аутентифікація успішна. Метод працює лише з валідними об'єктами AuthUserRequest. Трудомісткість: O(1) для генерації токена, O(n) для пошуку користувача в базі, де n — кількість користувачів. Обґрунтована залежність споживання пам’яті від розміру вхідних параметрів: використовує постійну кількість пам’яті для claims та AuthenticationResponse, що оцінюється як O(1). Блок-схема алгориму (рис. 2.3).

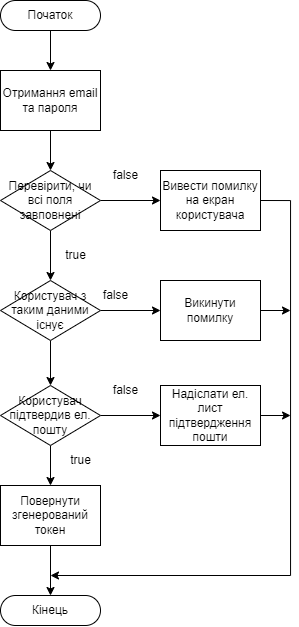


Рисунок 2.3 – Авторизаціякористувача

## 2.2 Бізнес-логіка застосунка

Метод createProduct призначений для зберігання нового товару. Вхідні параметри: productRequest (типу CreateProductRequest): містить дані для створення продукту (ім'я, код, баркод, опис, ціна, кількість, тощо), userId (типу String): ідентифікатор користувача, який створює продукт. Вихідні параметри: ServerResponse<Integer>: об'єкт, що містить ID нового продукту. Пошук категорії: O(n), де n — кількість категорій у репозиторії. Створення нового продукту: O(1) (фіксовані операції). Метод використовує фіксовану кількість пам’яті для об’єктів (Category, Inventory, Product), що оцінюється як O(1). Блок-схема алгоритму (рис. 2.4).

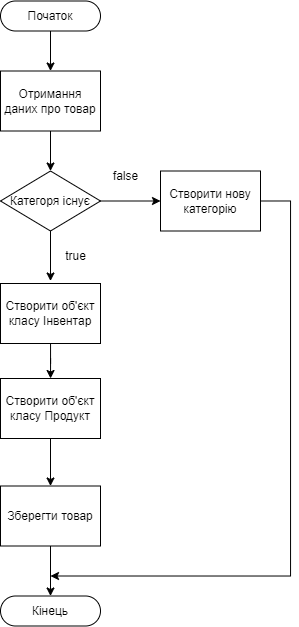


Рисунок 2.4 – Створення нового товару

Метод transfer відповідає за перенаправлення товару з одного складу в інший. Вхідні параметри: request (типу TransferRequest): містить дані для передачі продукту, такі як ідентифікатори продукту, інвентарю та складів, userId (типу String): ідентифікатор користувача, який виконує передачу. Вихідні параметри: ServerResponse<String>: об'єкт, що містить повідомлення про успіх або помилку передачі. Трудомісткість: пошук продукту: O(1) (ідентифікатор продукту), пошук інвентарю: O(1) (ідентифікатор інвентарю). Операції з оновленнями: O(1). Використовує постійну кількість пам’яті для об'єктів (Product, Inventory), що оцінюється як O(1). Блок-схема алгоритму (рис. 2.5).

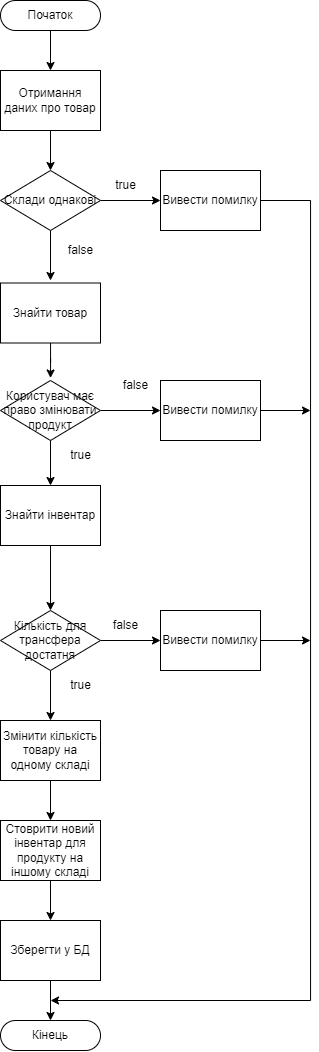


Рисунок 2.4 – Трансфер товару

# 3 ПРОЕКТУВАННЯ ВЕБ-САЙТУ

## 3.1 Мета та завдання ІС

Ціль ІС: створення веб-додатка, що забезпечує ефективне управління обліком товарів на складах, автоматизацію процесів обліку, передачі та управління запасами, підвищуючи точність, швидкість і зручність роботи користувачів.

Мета веб-додатка:

1. Забезпечити користувачів зручними інструментами для автоматичного обліку товарів, що надходять та відправляються зі складів, знижуючи ймовірність людських помилок.
2. Надати можливість в режимі реального часу відстежувати наявність товарів, щоб уникнути дефіциту або надлишків.
3. Спрощення процесів передачі товарів між складами, обліку поставок і контролю за термінами зберігання, що дозволить зекономити час і ресурси.
4. Аналіз даних: Забезпечити аналітичні звіти щодо обігу товарів, використання складів та ефективності управлінських рішень, що допоможе у стратегічному плануванні.

## 3.2 Кінцева аудиторія ІС

Інформаційна система для ведення обліку товарів на складах орієнтована на задоволення потреб користувачів, забезпечуючи зручність, доступ до необхідної інформації та інструменти для ефективного управління запасами та обробки замовлень. Можливі користувачі системи:

1. Менеджери складів:ефективний моніторинг запасів на складах; можливість швидко отримувати звіти про наявність товарів і їх обіг; інструменти для планування поповнення запасів.
2. Співробітники складу: інтуїтивно зрозумілий інтерфейс для обліку надходжень і відправлень товарів; можливість швидкої обробки замовлень і перевірки наявності товарів; системи для сканування штрих-кодів для швидшого внесення даних.
3. Керівництво компанії: стратегічна інформація про запаси, обсяги продажів і тренди для прийняття рішень; інструменти для моніторингу ефективності роботи складу та управлінських процесів.

## 3.3 Функціональні вимоги ІС

FR1. Реєстрація нового користувача:

FR1.1 Користувач повинен ввести обов'язкові дані: електронну пошту, пароль.

FR1.2 Пароль повинен відповідати критеріям безпеки: не менше 8 символів, містити щонайменше одну велику літеру, одну цифру та один спеціальний символ.

FR1.3 У разі введення некоректних або неповних даних система повинна видати повідомлення: «Будь ласка, заповніть усі обов'язкові поля коректно».

FR1.4 Якщо електронна пошта вже зареєстрована, система повинна видати повідомлення: «Ця електронна пошта вже використовується».

FR1.5 При успішній реєстрації користувач повинен отримати повідомлення: «Реєстрація успішна. Ви можете увійти до системи».

FR1.6 Після успішної реєстрації система повинна надіслати електронний лист на вказану адресу для підтвердження реєстрації.

FR1.7 Користувач повинен перейти за посиланням у листі для завершення процесу реєстрації.

FR2. Авторизація користувача

FR2.1 Користувач повинен ввести електронну пошту та пароль для авторизації.

FR2.2 У разі некоректного введення електронної пошти або пароля система повинна видати повідомлення: «Некоректний логін або пароль», без уточнення, що саме було введено неправильно.

FR2.3 При успішній авторизації користувач повинен бути перенаправлений до робочого кабінету.

FR2.4 Система повинна зберігати сесію користувача, щоб він не вимагав повторного входу протягом певного часу (наприклад, 30 хвилин бездіяльності).

FR3: Створення команди

FR3.1 Система повинна дозволяти адміністратору створювати нову команду, вказуючи її назву та опис (той, хто створює команду – автоматично адміністратор).

FR4: Управління учасниками команди

FR4.1 Система повинна дозволяти адміністратору додавати нових учасників за електронною поштою до команди з відповідними ролями.

FR4.2 У разі, якщо такий користувач ще не зареєстрований – надати відповідне повідомлення.

FR4.3: Адміністратор команди повинен мати можливість видаляти учасників із команди.

FR5. Додавання товару в систему

FR5.1 Користувач повинен ввести всю необхідну інформацію про товар: назву, код, штрих-код, опис, ціну, кількість доступних одиниць на відповдному складі, та категорію.

FR5.2 У разі відсутності обов'язкових полів або введення некоректних даних система повинна видати повідомлення: «Будь ласка, заповніть всі обов'язкові поля коректно».

FR5.3 При успішному додаванні товару система повинна відобразити повідомлення: «Товар успішно додано» та надати користувачеві можливість додати новий товар або повернутися до списку товарів.

FR6. Редагування інформації про товар

FR6.1 Користувач повинен мати можливість вибрати товар зі списку для редагування.

FR6.2 Після вибору товару користувач може редагувати будь-які з полів: назву, код, штрих-код, опис, ціну, кількість доступних одиниць, та категорію.

FR6.3 У разі некоректного введення даних система повинна видати повідомлення: «Будь ласка, виправте помилки у формах».

FR6.4 При успішному редагуванні інформації система повинна відобразити повідомлення: «Інформацію про товар успішно оновлено».

FR7. Видалення товару з системи

FR7.1 Користувач повинен мати можливість вибрати товар зі списку для видалення.

FR7.2 Система повинна підтвердити видалення товару через діалогове вікно: «Ви впевнені, що хочете видалити цей товар?».

FR7.3 У разі успішного видалення товару система повинна відобразити повідомлення: «Товар успішно видалено».

FR8 Пошук товарів

FR8.1 Користувач може ввести назву товару для перевірки його наявності.

FR8.2 Система повинна відобразити такий товар або – повідомлення «Товар недоступний», залежно від результату перевірки.

FR9. Передача товару між складами

FR9.1 Користувач повинен вказати товар, який передається, а також склади "звідки" та "куди" здійснюється передача.

FR9.2 У разі недостатньої кількості товару для передачі система повинна видати повідомлення: «Недостатня кількість товару для передачі».

FR9.3 При успішній передачі товару система повинна відобразити повідомлення: «Передача товару успішно виконана».

FR10. Перегляд аналітики товарів

FR10.1 Система повинна відображати загальні дані про кількість товарів на складі, кількість проданих одиниць та їх загальну вартість за вибраний період.

FR9.3 Система повинна генерувати графіки або діаграми, що відображають динаміку продажів товарів за вибраний період.

FR9.4 Користувач може завантажити усі доступні товари у форматі JSON.

FR10. Моніторинг залишків товарів

FR10.1 Система повинна автоматично оновлювати дані про залишки товарів у режимі реального часу при кожній операції (додавання, видалення, передача), а також за графіком, кожні 30 хв.

FR10.2 Користувач отримує повідомлення про нестачу товару на складі у вигляді елекронного листа.

FR10.3 Користувач отримує повідомлення про нестачу товару на складі у вигляді сповіщення на веб-застосунку.

FR10.4 Користувач повинен мати можливість замовляти товари у вигляді PDF форми, що надсилається йому на пошту

FR11. Створення нового замовлення

FR11.1 Користувач повинен мати можливість вибрати товари для замовлення зі списку доступних товарів.

FR11.2 Система повинна дозволяти користувачу вказати кількість кожного товару, яку потрібно замовити.

FR11.3 При спробі замовлення кількості товару, яка перевищує наявний запас, система повинна видати повідомлення: «Недостатня кількість товару на складі для замовлення».

FR12 Перегляд отсанніх змін про товари

FR12.1 Користувач повинен мати можливість переглядати останні зміни про товари.

FR12.1 Користувач повинен бачити, що саму спричинило зміну у товарі, наприклад, співробіиник оновив дані, система замовила товар або покупець купив товар.

## 3.4 Моделювання прецедентів

Діаграма прецедентів (use case diagram) (рис. 3.5) відображає взаємодії користувачів із системою в контексті управління товарами, створення замовлень, та аналітики. Основні елементи діаграми:

Актори:

* незареєстрований користувач може зареєструватися та авторизуватися в системі. Під час реєстрації проходить верифікація електронної пошти.
* співробітник має доступ до таких функцій:
* адміністратор відповідає за управління командою, включаючи додавання та видалення учасників, призначення ролей.
* менеджер займається створенням замовлень для поповнення товарів або виконання клієнтських запитів.
* замовник (зовнішній користувач) може створювати замовлення на товари.

Основні прецеденти (Use Cases):

* реєстрація: користувач може зареєструватися в систем;
* верифікація пошти: підтвердження електронної пошти після реєстрації;
* авторизація: процес входу в систему після реєстрації;
* управління товарами (рис. 2.6);
* управління командою (рис. 2.7);
* перегляд аналітики (рис. 2.8);
* моніторинг заришків;
* стоврити замовлення для співробітника;
* стоврити замовлення для замовника;

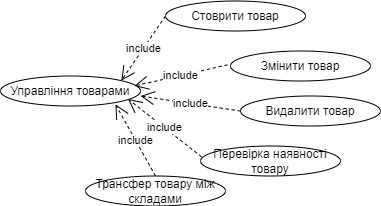


Рисунок 2.6 – Прецедент управління товарами

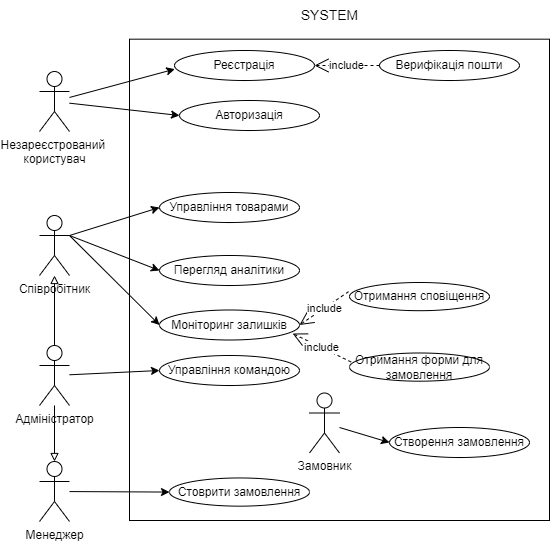


Рисунок 2.5 – Діаграма прецедентів

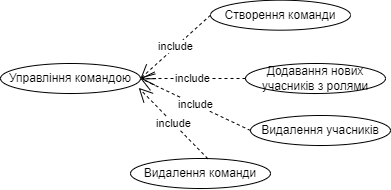


Рисунок 2.7 – Прецедент управління командою

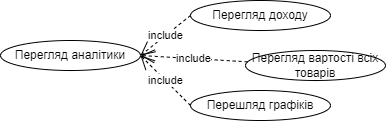


Рисунок 2.8 – Прецедент перегляд аналітики

## 3.5 Нефункціональні вимоги

NFR1. Вимоги платформи:

NFR1.1 Додаток повинен підтримувати основні веб-браузери:

* Google Chrome (версія не нижче 90);
* Mozilla Firefox (версія не нижче 85);
* Microsoft Edge (версія не нижче 90);
* Safari (версія не нижче 14).

NFR1.2 Додаток повинен коректно працювати на операційних системах Windows 10, macOS 11, і Linux (версія Ubuntu 20.04 і вище).

NFR2. Надійність:

NFR2.1 У разі збою підключення до бази даних система повинна автоматично спробувати повторне підключення протягом 5 секунд.

NFR3. Безпека

NFR3.1 Усі дані, що передаються між клієнтом і сервером, повинні бути зашифровані з використанням протоколу HTTPS.

NFR3.2 Система повинна використовувати алгоритми шифрування AES з довжиною ключа не менше 256 біт.

NFR1.3 Система шифрує конфеденційні дані такі, як пароль коритсувача.

NFR4. Продуктивність

NFR4.1 Максимальний час відповіді системи на запит користувача не повинен перевищувати ~3 секунди.

NFR4.2 Система повинна витримувати одночасно до 1000 активних користувачів.

NFR4.3 Додаток повинен підтримувати до 5000 відвідувань користувачів на добу.

NFR4.4 Система повинна обробляти до 200 транзакцій в секунду.

NFR5. Обсяг даних

NFR5.1 Система повинна підтримувати зберігання даних не менше ніж 100,000 користувачів.

NFR5.2 Максимальний обсяг даних, який система повинна обробляти, становить 1 ТБ.

NFR6. Доступність

NW6.1 Система повинна бути доступною 24/7.

NW6.2 Максимально допустиме час простою системи не повинно перевищувати 1 години на рік (99.99% доступності).

NW6.3 Під час планового обслуговування система повинна сповіщати користувачів заздалегідь, не менш ніж за 24 години.

## 3.6 Architecture Constraints

Застосунок для ведення інвентаризації можна віднести до архітектурного патерну Мікросервісної архітектури WA (Web Application) архітепу.

1. Мова програмування: Java.

Джерело: Вимога замовника (дипломний керівник).

Опис: Основна мова програмування для розробки всіх компонентів додатку, забезпечує сумісність з існуючими рішеннями та вимагається для використання в проєкті.

1. Фреймворк: Spring Boot.

Джерело: Вимога замовника (дипломний керівник).

Опис: Вибір Spring Boot як основного фреймворку для розробки серверної частини додатку обумовлений вимогою до створення сучасного, кросплатформеного та масштабованого веб-застосунку.

1. Система управління базами даних: PostgreSQL.

Джерело: Інтеграція з існуючою інформаційною системою (ІС).

Опис: Використання PostgreSQL як системи управління базами даних для зберігання структурованих даних додатку через наявність сумісності з існуючими ІС, що також використовують PostgreSQL.

1. Система управління NoSQL базами даних: MongoDB.

Джерело: Інтеграція з ІС та вимога замовника (дипломний керівник).

Опис: MongoDB використовується для зберігання неструктурованих даних та масштабованих об'єктів, що не підпадають під реляційну модель даних.

1. Система обміну повідомленнями: Apache Kafka.

Джерело: Вимога замовника (дипломний керівник).

Опис: Kafka використовується для передачі повідомлень між мікросервісами, забезпечуючи надійну та масштабовану систему обробки потоків даних у реальному часі.

1. Формат передачі даних: JSON.

Джерело: Вимога замовника (дипломний керівник).

Опис: Формат JSON обрано для обміну даними між клієнтською та серверною частинами додатку через його легкість та широке використання в сучасних веб-застосунках.

1. Інтерфейс взаємодії між сервісами: REST API.

Джерело: Інтеграція з існуючими ІС.

Опис: Взаємодія між мікросервісами та іншими системами відбувається через REST API, що забезпечує стандартизовану передачу даних та легкість інтеграції.

1. Протокол безпеки: JWT (JSON Web Tokens).

Джерело: Вимога замовника (дипломний керівник).

Опис: Використання JWT для аутентифікації та авторизації користувачів. Цей протокол забезпечує безпечну передачу даних між клієнтом та сервером, що відповідає вимогам до безпеки додатку.

1. Фронтенд: React.

Джерело: Вимога замовника (дипломний керівник).

Опис: Вибір React для побудови клієнтської частини додатку обумовлений необхідністю створення динамічного, швидкого та інтерактивного інтерфейсу користувача.

1. UI бібліотека: ShadCN UI.

Джерело: Вимога замовника (дипломний керівник).

Опис: Для побудови інтерфейсу користувача використовується ShadCN UI, яка надає готові компоненти для швидкої розробки сучасних інтерфейсів.

1. Контейнеризація: Docker.

Джерело: Вимога інтеграції з існуючими ІС.

Опис: Використання Docker для контейнеризації компонентів додатку з метою спрощення деплойменту та забезпечення кросплатформеності рішення.

## 3.7 UI View

Cторінка реєстраці дозволяє користувачу, ввевши валідні дані зареєструватися у системі (рис. 2.9).

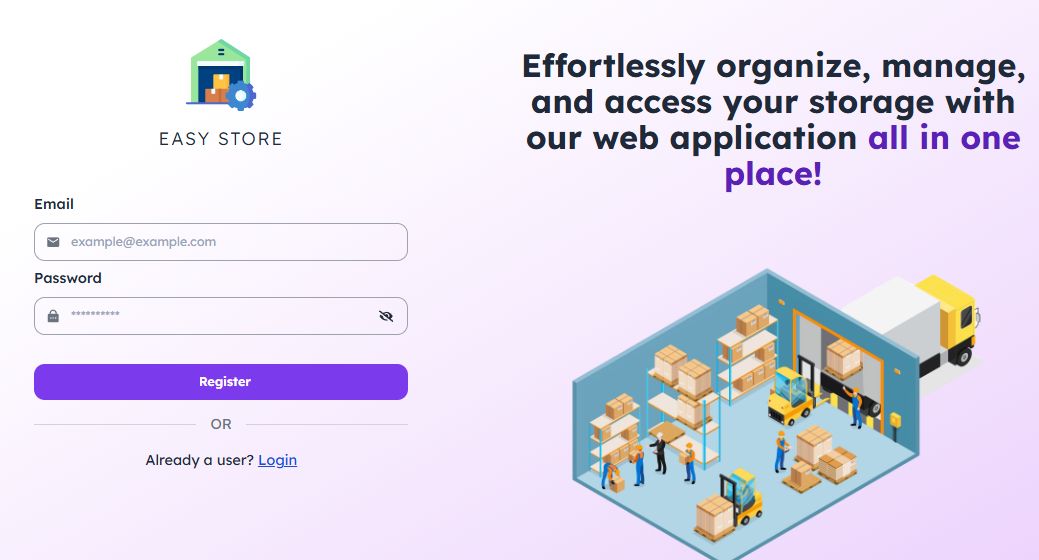


Рисунок 2.9 – Сторінка реєстрації

Cторінка авторизації дозволяє користувачу, ввевши валідні дані увійти у систему (рис. 2.10).

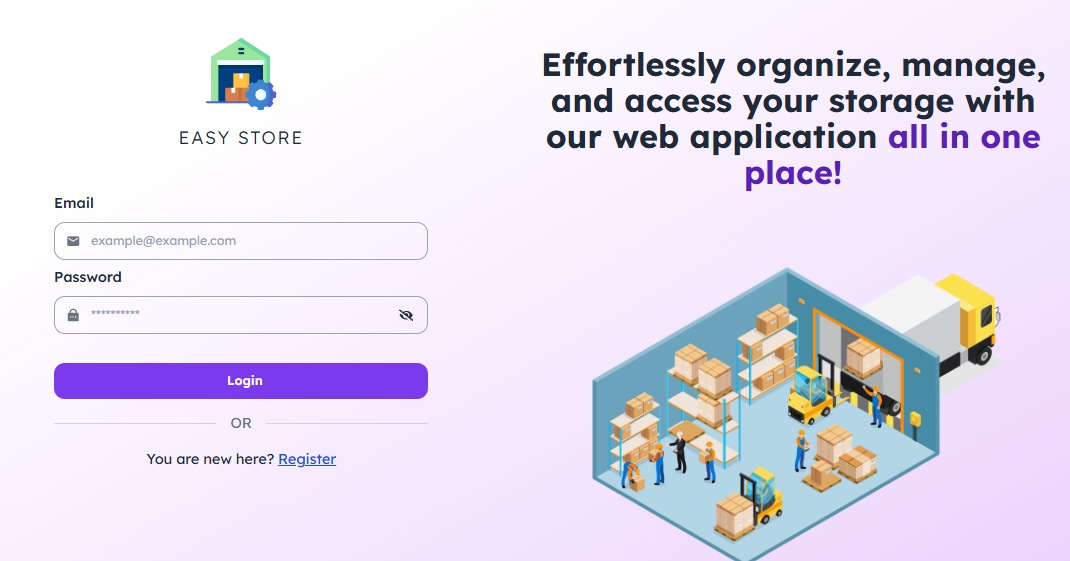


Рисунок 2.9 – Сторінка входу

Сторінка «Інформаційна панель» є центральним центром для користувачів, що забезпечує огляд ключової інформації та швидкий доступ до важливих дій (рис. 2.10). Ключові розділи:

* огляд системних показників;
* прибуток;
* вартість товарів на складах;
* останні переміщення операції над товрами;
* графкіи.
* загальна кількість товарів;
* кількість користувачів у команді.

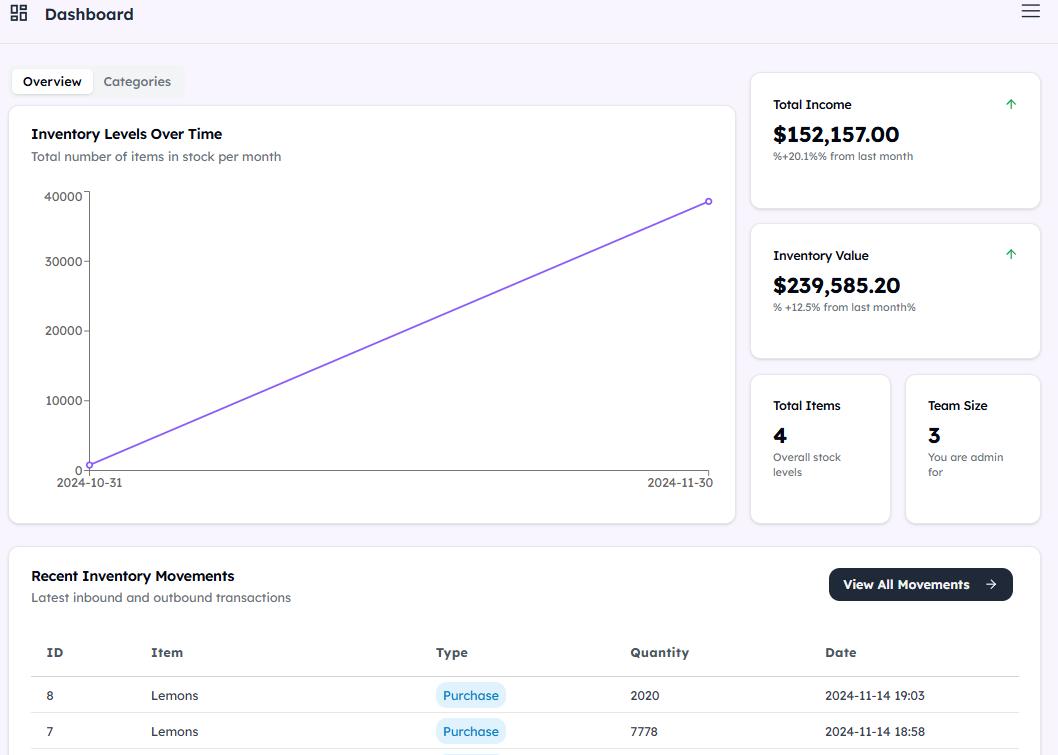


Рисунок 2.10 – Сторінка «Інформаційна панель»

На рисунку (рис. 2.11) показано сторінку керування командою зі зрозумілим і простим інтерфейсом користувача. Сторінка складається з трьох основних вкладок: «Створити команду» (рис. 2.12), «Запросити учасників» (рис. 2.13) і «Керувати командою» (рис. 2.14).

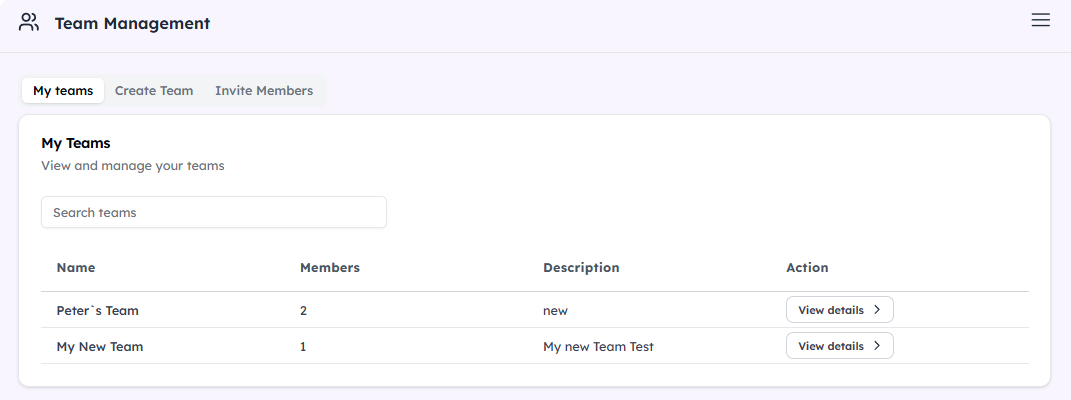


Рисунок 2.11 – Сторінка «Команда»

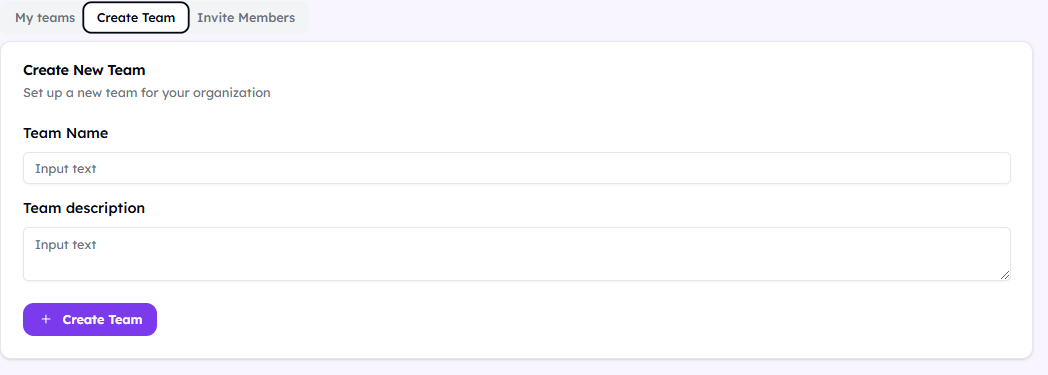


Рисунок 2.12 – Створити команду

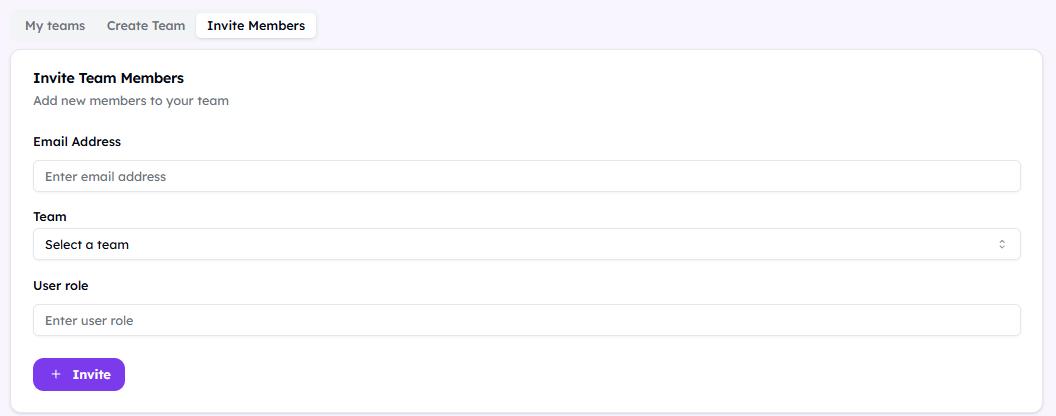


Рисунок 2.13 – Запросити учасників

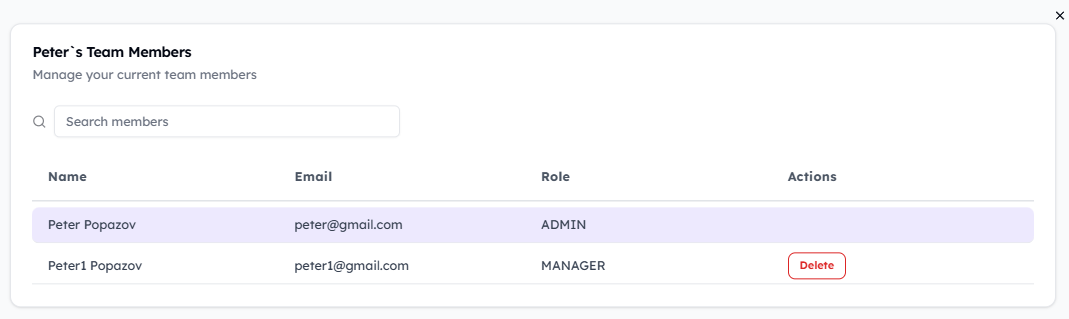


Рисунок 2.14 – Керувати командою

На зображенні (рис. 2.15) відображена сторінка Інвентаризації з чистим і організованим інтерфейсом користувача. Сторінка містить такі елементи:

Пошуковий рядок у верхній частині, де можна вводити назву товару для пошуку.

Таблиця з кількома колонками, які містять наступну інформацію:

* Product (Товар);
* Price (Ціна);
* Quantity (Кількість);
* Category (Категорія);
* Reorder Level (Рівень поповнення);
* і т.д.

У верхньому правому куті сторінки розташовані кнопки:

* Add Item (Додати товар) (рис. 2.16);
* Export (Експортувати): для експорту даних з таблиці.

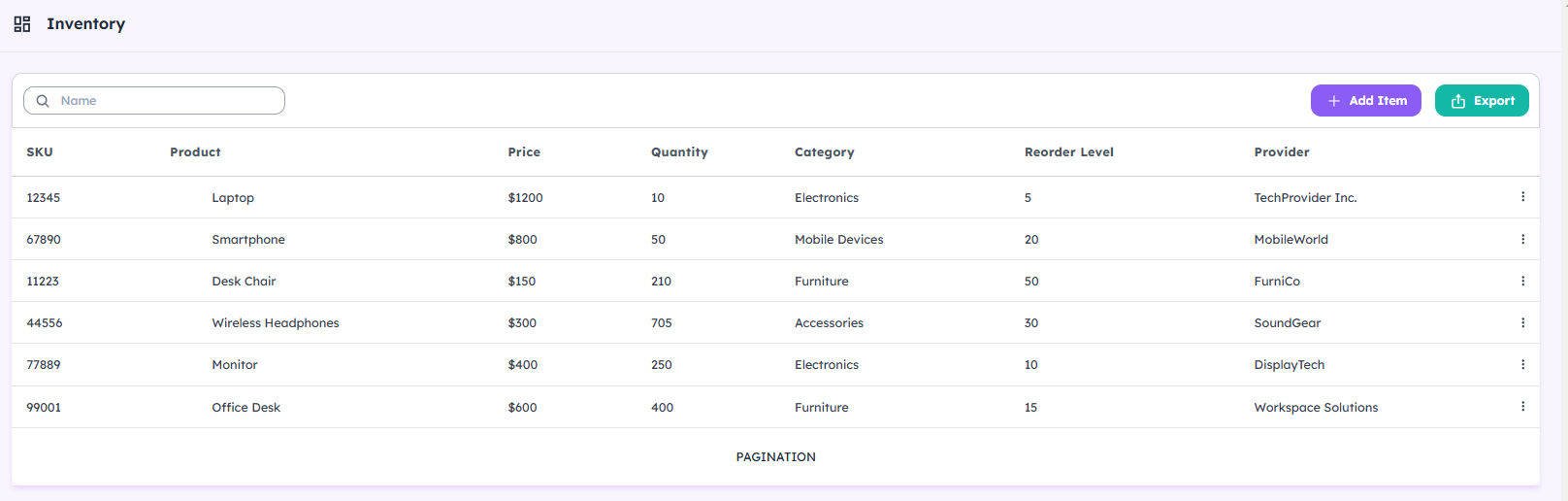


Рисунок 2.15 – Сторінка «Інвентаризація»

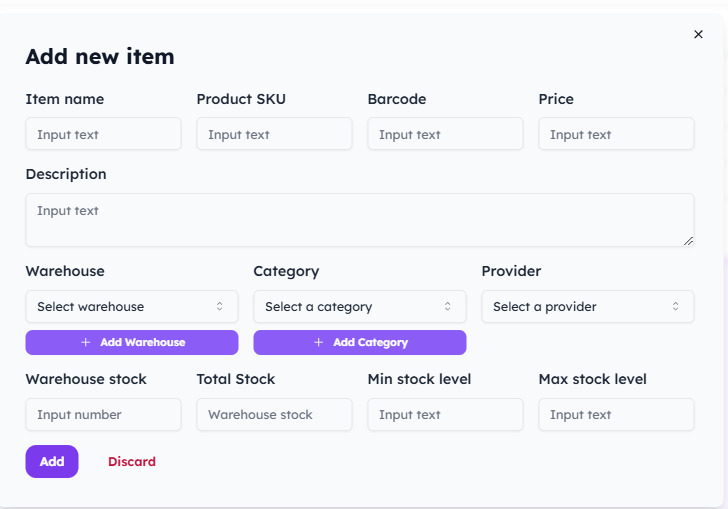


Рисунок 2.16 – Модальне вікно «Додати новий товар»

Для роботи із товаром створено спеціальну панель (рис. 2.17), яка містить кнопки «Змінити» - (рис. 2.18), «Видалити» - (рис. 2.19).

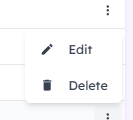


Рисунок 2.17 – Панель роботи з товаром

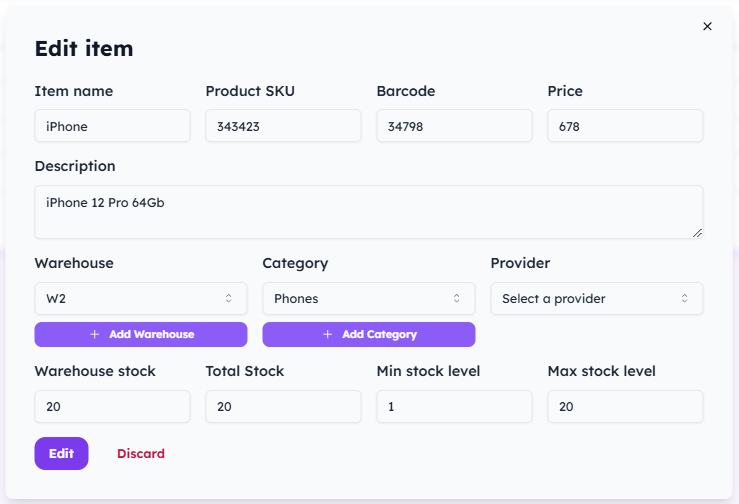


Рисунок 2.18 – Модальне вікно «Змінити товар»

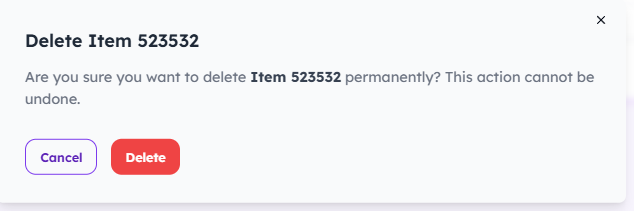


Рисунок 2.19 – Модальне вікно «Видалити товар»

Сторінка про рухи товарів стоврена для зручного відображення останніх тразацій здійснені над товарами (рис. 2.20).

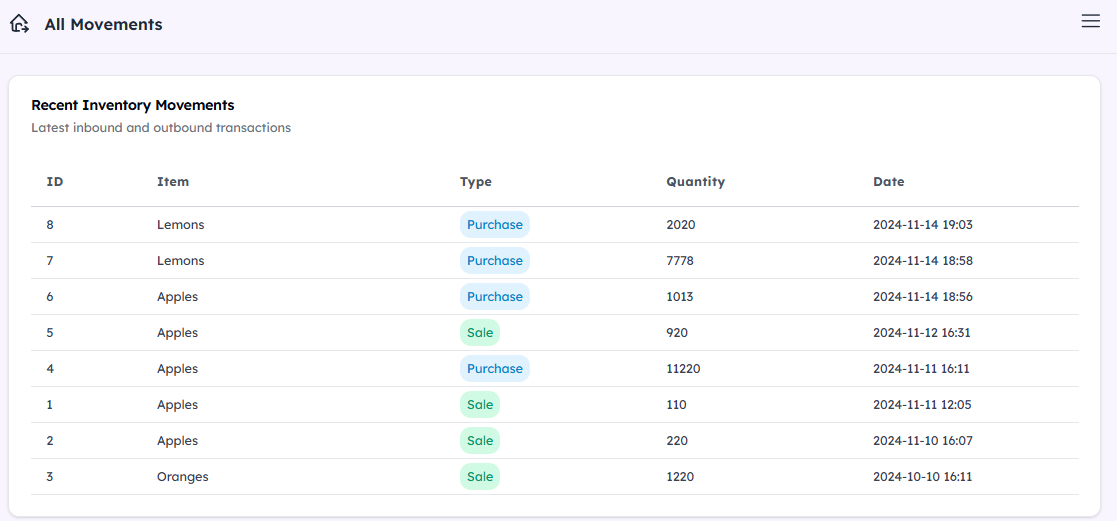


Рисунок 2.20 – Сторінка про рухи товарів

Сторінка «Панель сповіщень» (рис. 2.21) служить для наступних цілей:

1. Відстеження низьких запасів: виділяє товари, які потрібно незабаром поповнити, щоб уникнути їх вичерпання.
2. Кнопку для отримання форми для замовлення товарів на ел. пошту.
3. Інформаційна панель надає приблизну загальну вартість поповнення запасів усіх товарів, які наразі позначені як низькі, що допомагає спланувати бюджет для поповнення.

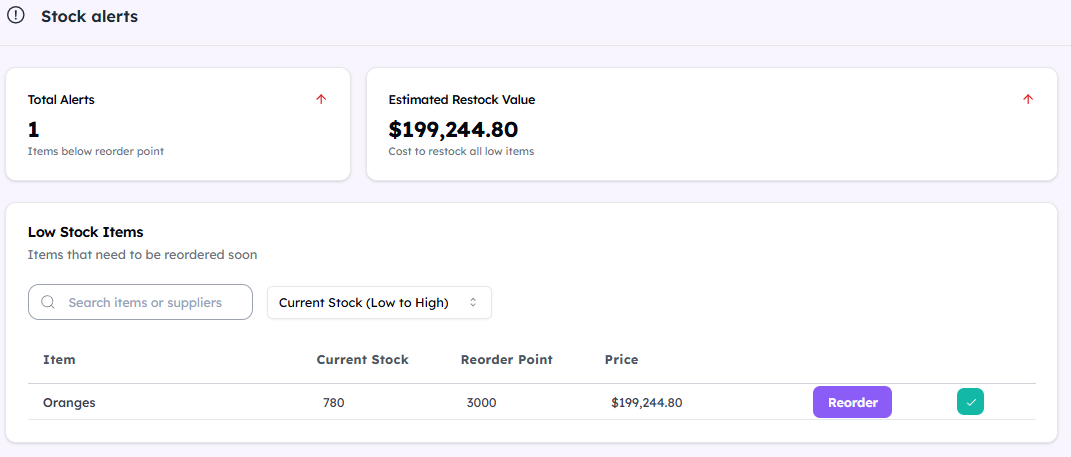


Рисунок 2.21 – Сторінка «Панель сповіщень»

## 3.8 Logical View

Зображено логіне представлення (рис. 2.22). Користувач взаємодіє з клієнтом (веб), який надсилає HTTP запити на API Gateway, який у свою чергу аутентифікує користувача та додає у загловки HTTP запиту необхідну інформацію. API Gateway знає про всі інстанси мікросервісів та має load balancer, який здатен автоматично направляти запити до потрібного мікросервісу. Комунікація між мікросервісами реалізована у приватній мережі (клієнт не може надсилати запити на мікросервіс, тільки через API Gateway). Присутній Message Broker Apache Kafka та мікросервіси, які реєструють повідомлення та ті, що слухають. Наявні дві СУБД PostgreSQL – таблична (RDBMS) та MongoDB – документноорієнтована.

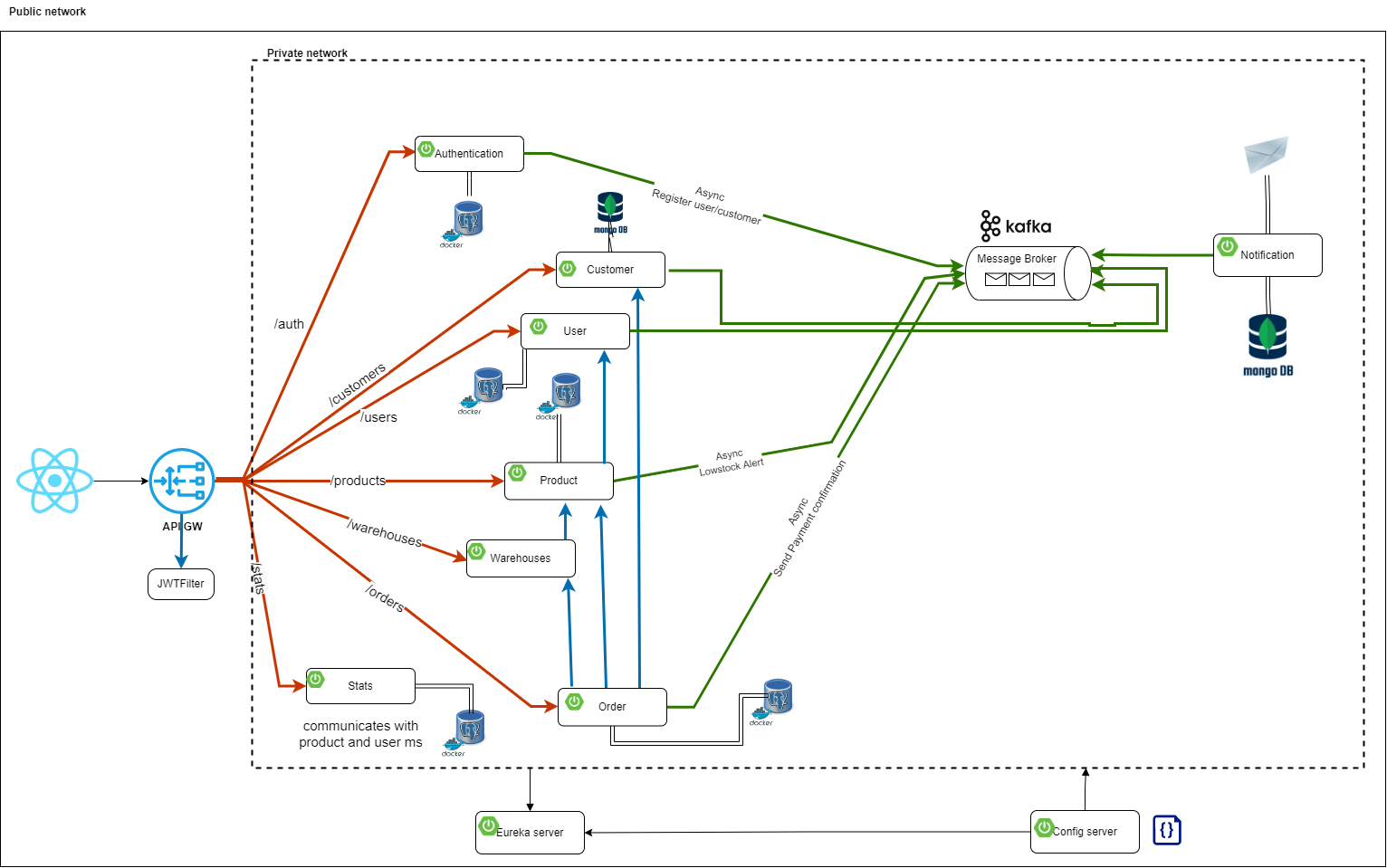


Рисунок 2.22 – Logical View

## 3.9 Deployment View

Кожен мікросервіс запускається на окремому сервері. Кожен мікросервіс використовує свою СУБД, що запускається в окремому контейнері. СУБД не були позначені на діаграмі для візуального спрощення, адже кожен мікросервіс використовує СУБД і загальна кількість контейнерів перевищувала 20 (рис. 2.23).

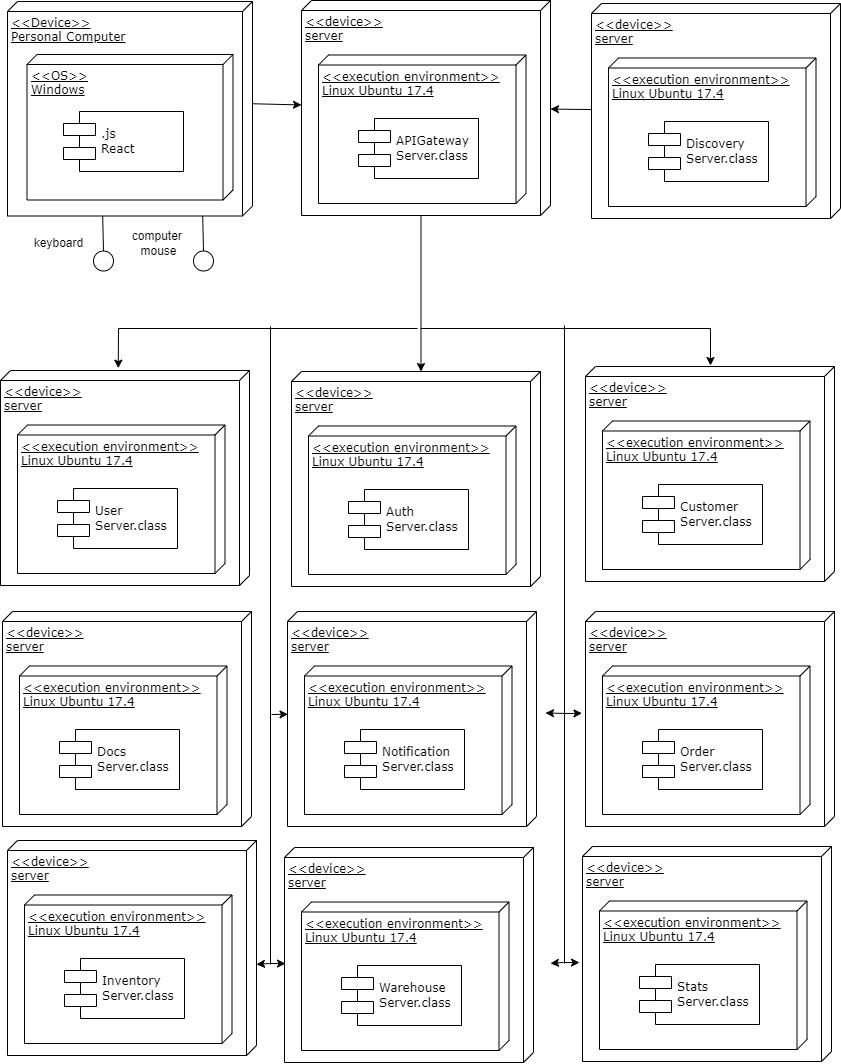


Рисунок 2.22 – Deployment View

## 3.10 Design View

UML діаграма Design View (рис. 2.23). Рівні пов’язані з логічним розподілом компонентів і функціональних можливостей і не враховують фізичне розташування компонентів, тоді як рівні описують фізичний розподіл функціональних можливостей і компонентів на окремих серверах, комп’ютерах, мережах або віддалених локації. Модель додатка показує кілька рівнів: рівень презентації, Presentation Layer, Business Layer, Data Layer.

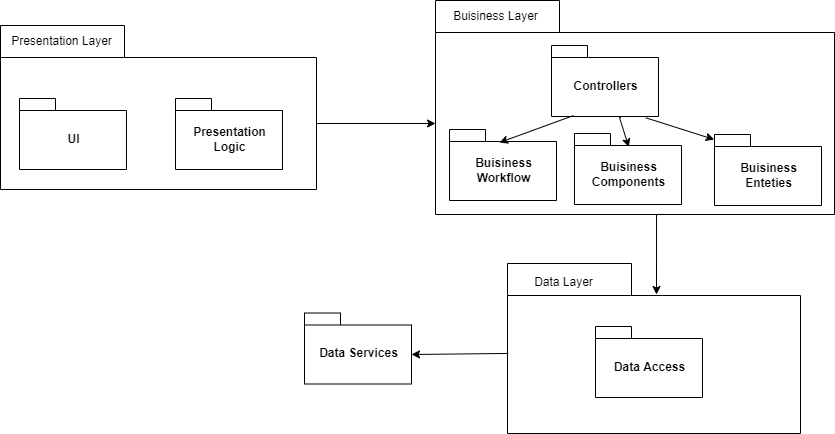


Рисунок 2.23 – Design View

## 3.11 Process View

Нижче наведено схему процесу додавання нового продукту до системи (рис. 2.24). Діаграма процесу візуально представляє послідовність дій і моментів прийняття рішень, які беруть участь у процесі додавання продукту.

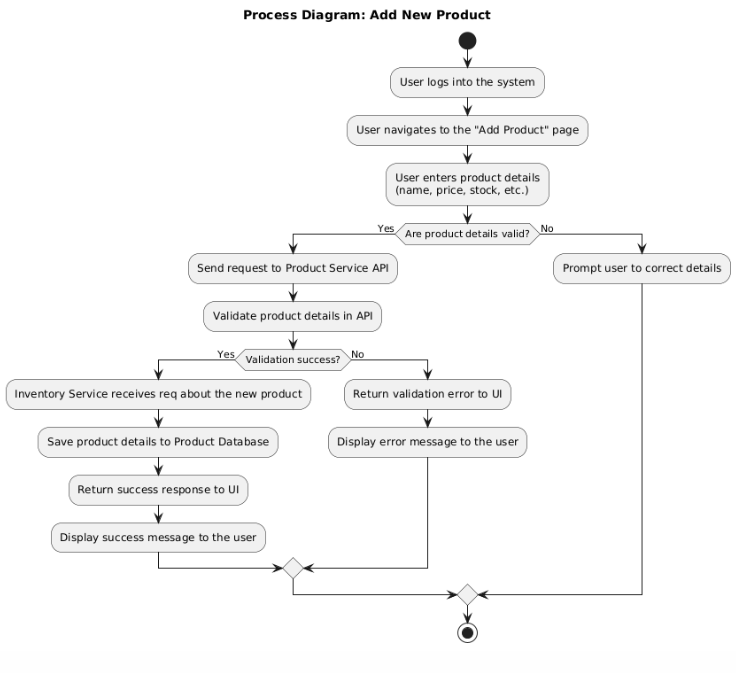


Рисунок 2.24 – Додавання нового продукту до системи

Нижче наведено діаграму процесу, яка представляє алгоритм для перевірки недостатнього запасу (рис. 2.25).

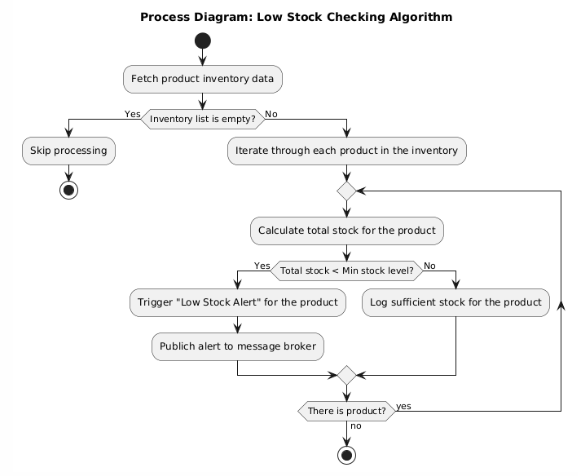


Рисунок 2.24 – Алгоритм для перевірки недостатнього запасу

Нижче наведено діаграму процесу, яка представляє процес додавання нового користувача до команди (рис. 2.26).

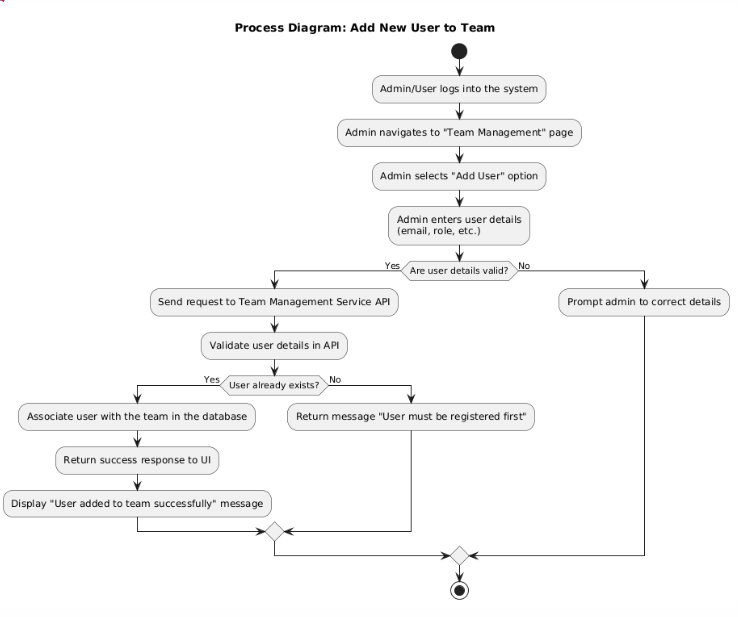
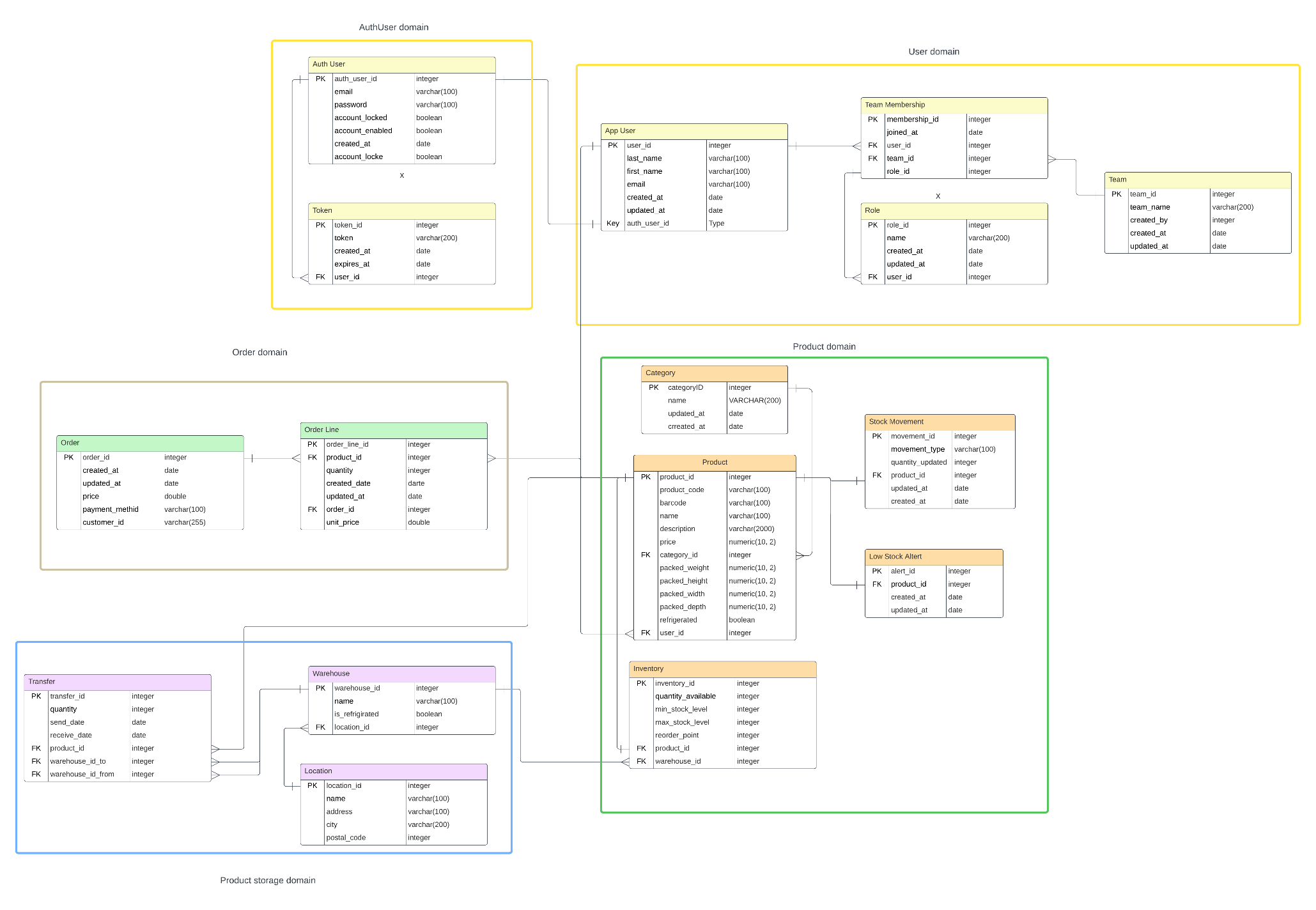


Рисунок 2.24 – Додавання нового користувача до команди

## 3.12 Data View

При плануванні розробки застосунка було обрано використовувати дві СУБД різного типу - SQL (PosrgreSQL) та NoSQL (MongoDB). Розроблено ER-діаграму із зазначеним первинними та вторинними ключами (рис. 2.25). Так як архітектура застосунка передбачає використання мікросервісів, усю діаграму було розбито на домени за технікою (Domain Driven Development). Було створено індекси для полів, за якими найчастіше планується здійснювати пошук, а саме у таблиці User (email), Category (name).

Рисунок 2.25 – ER-діаграма

Наведено діаграми для документоорієнтованої СУБД. Призначення — зберігання даних про надіслані електронні листи (рис. 2.26) зберігання даних про покупців (рис. 2.27).

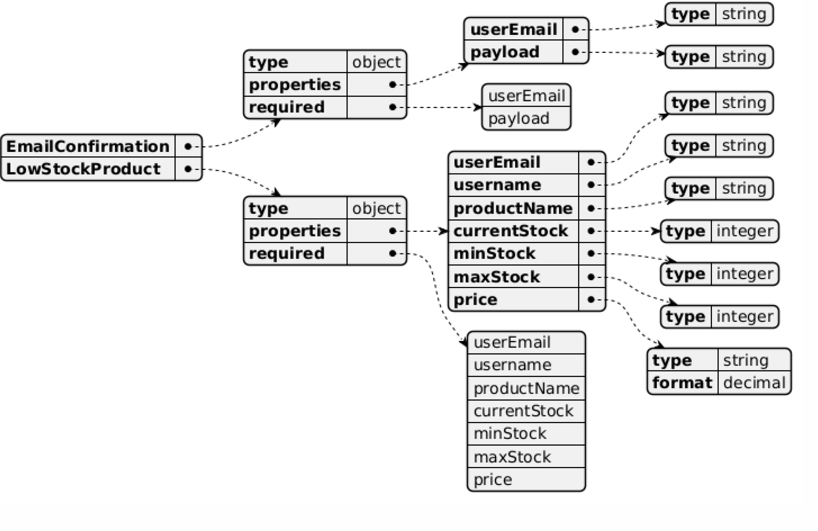


Рисунок 2.26 – Зберігання даних про надіслані електронні листи

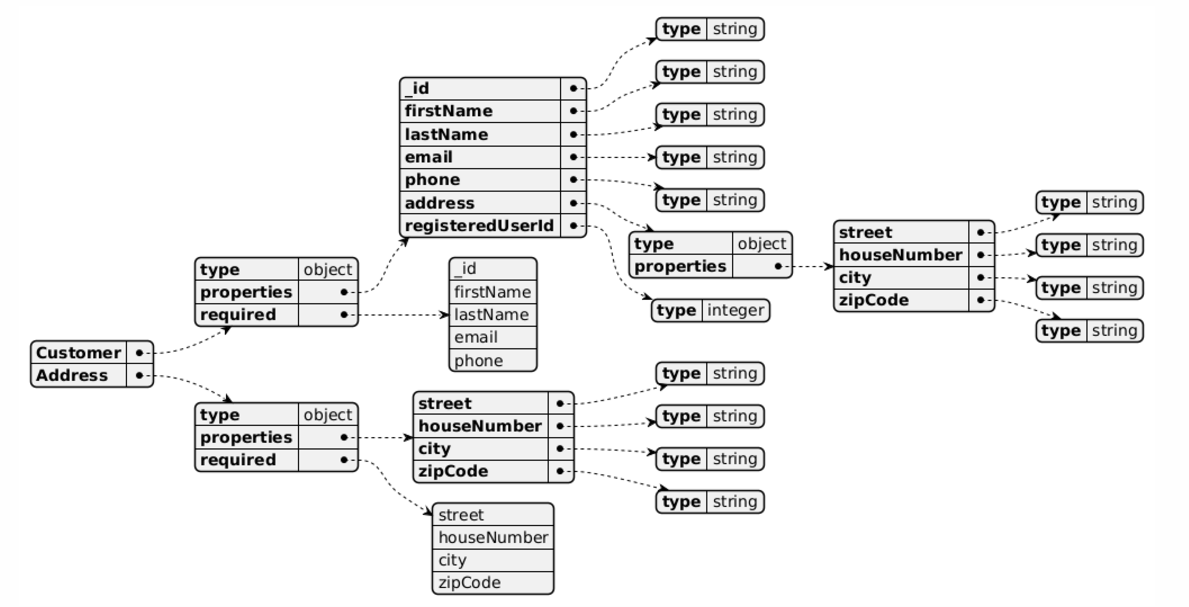


Рисунок 2.27 – Зберігання даних про покупців

## 3.13 Interface View

Система Smart Inventory Management System (рис. 2.28) розроблена з використанням архітектури мікросервісів із API Gateway, який діє як центральна точка входу для зв’язку між інтерфейсом і сервером. Frontend, створений як React SPA, надає інтерфейси для керування сповіщеннями про низькі запаси, керування продуктами та автентифікації користувачів (частина функціоналу). Зовнішні компоненти надсилають запити REST API до шлюзу API, який потім направляє ці запити до відповідних серверних служб. Для автентифікації шлюз API пересилає запити користувачів на вхід до служби автентифікації, де автентифікація JWT забезпечує безпечний доступ. Після автентифікації користувачі можуть взаємодіяти з іншими серверними службами, такими як Product Service для операцій CRUD над продуктами та Alert Service, щоб ініціювати та керувати сповіщеннями про низькі запаси.

Сервіс складається з різних мікросервісів Spring Boot, включаючи Product Service, який взаємодіє з таблицею Products у базі даних PostgreSQL для обробки даних продукту. Служба сповіщень обробляє сповіщення про низький запас, зберігаючи їх у таблиці сповіщень і використовуючи службу електронної пошти для надсилання сповіщень. Служба електронної пошти також взаємодіє зі службою PDF Generator Service, щоб додавати звіти до електронних листів і зберігати пов’язані з електронною поштою дані в MongoDB. Крім того, Служба продукту публікує сповіщення для брокера повідомлень Kafka, який прослуховує Служба сповіщень для обробки сповіщень у реальному часі. Ця система обміну повідомленнями допомагає роз’єднати служби та забезпечити асинхронну обробку. Усі серверні служби підключено до відповідних баз даних, таких як PostgreSQL для продуктів, сповіщень і даних автентифікації, що забезпечує надійну та масштабовану архітектуру. API Gateway спрощує взаємодію інтерфейсу з кількома мікросервісами, пропонуючи єдину точку доступу для маршрутизації запитів, одночасно вирішуючи такі проблеми, як автентифікація та балансування навантаження.

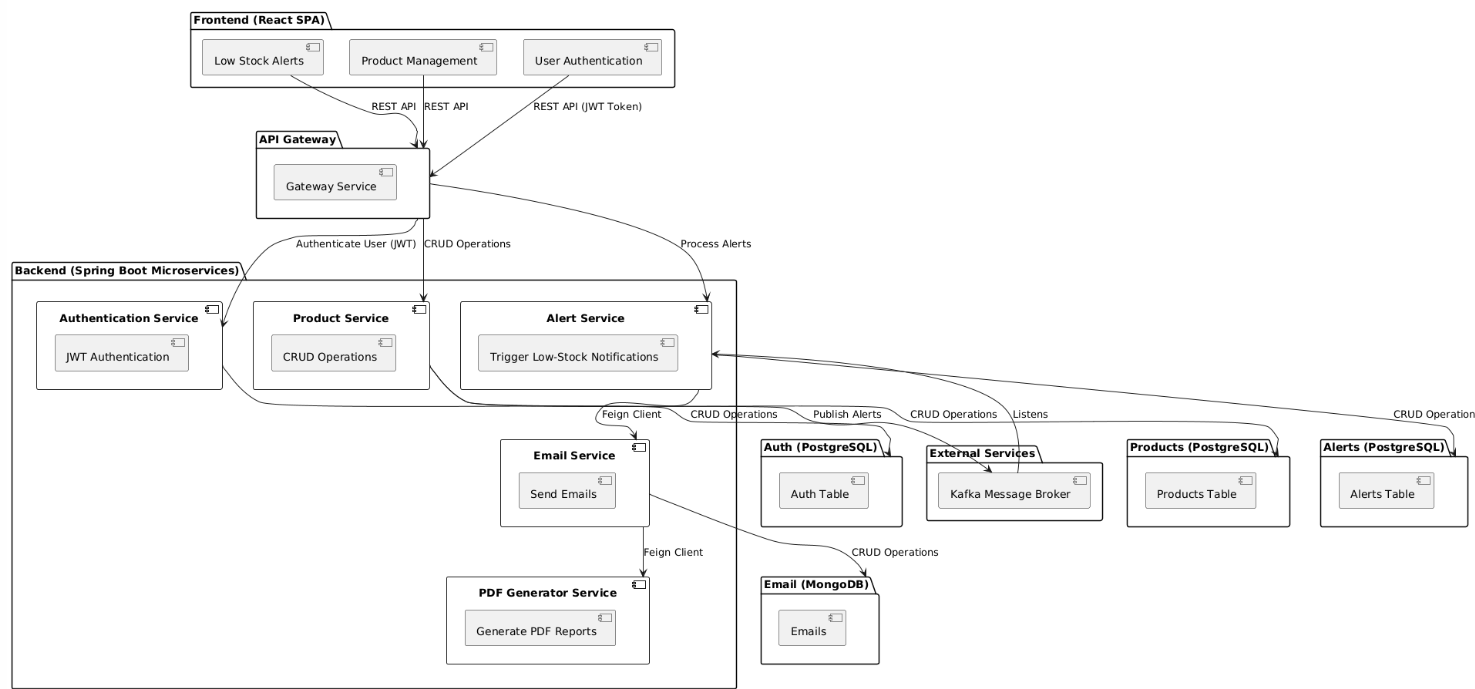


Рисунок 2.28 – Interface View

## 3.14 Security View

Для системи, побудованої на архітектурі мікросервісів з використанням Spring Boot, API Gateway та окремого мікросервісу для аутентифікації, використані наступні кроки забезпечення безпеки:

1. Аутентифікація та авторизація. Використовується аутентифікація за допомогою токенів JWT — JSON Web Token. Клієнт передає токен у заголовку запиту HTTP (Authorization Header), який перевіряється на API Gateway. Для реалізації ролей та дозволів застосовується Spring Security з підтримкою механізмів авторизації на рівні API Gateway. Використання HTTPS (SSL/TLS) для захищеного передавання даних між клієнтом і сервером, а також між мікросервісами.

2. Захист чутливих даних у базі даних. Паролі користувачів зберігаються у зашифрованому вигляді, використовуючи алгоритм BCrypt

3. Захищені з'єднання. Всі канали зв'язку між клієнтом і сервером, а також між мікросервісами, повинні бути зашифровані за допомогою SSL/TLS для захисту даних від перехоплення та підміни. API Gateway та інші мікросервіси забезпечують підтримку TLS 1.2 або 1.3 для безпечного з'єднання.

4. Запобігання ін'єкціям. Використання параметризованих запитів (prepared statements) та ORM-бібліотек, таких як JPA або Hibernate, для захисту від SQL-ін'єкцій. Використання механізмів захисту від CSRF, таких як токени CSRF (вони повинні передаватися у кожному запиті від клієнта і перевірятися сервером). Spring Security підтримує захист від CSRF за замовчуванням.

5. Моніторинг і аудит. Використання централізованих систем для збору логів та моніторингу безпеки, таких як ELK Stack (Elasticsearch, Logstash, Kibana) або Prometheus + Grafana.

## 3.15 Infrastructure View

Infrastructure View (Інфраструктурне представлення) — це архітектурне представлення інформаційної системи, яке описує її фізичну та програмну інфраструктуру (табл. 2.1, табл. 2.2). Його основна мета — деталізувати технічне середовище, в якому працює система, щоб забезпечити її ефективність, надійність та масштабованість.

Таблиця 2.1 – Application Infrastructure View

|  |  |  |  |
| --- | --- | --- | --- |
| Сервер додатків | База даних (PostgreSQL): | AWS (Amazon Web Services) | ПЗ |
| Оперативна пам'ять: 16 ГБ  ЦП: 8 ядер (3,0 ГГц)  GPU: відсутній | Оперативна пам'ять: 32 ГБ  ЦП: 12 ядер (2,8 ГГц)  Хранилище: SSD 1 ТБ | EC2 для серверів програм, RDS для бази даних | Java 17 (Spring Boot 3.3)  Kafka 3.0 |

|  |  |  |
| --- | --- | --- |
| PCs | Mobiles | Веб-браузери |
| Оперативна пам'ять: мінімум 4 ГБ  ЦП: 2 ядра (2.0 ГГц)  GPU: базовий рівень, інтегрована графіка | Оперативна пам'ять: мінімум 2 ГБ  ОС: Android 8.0+ або iOS 13+  Підключення: 4G або Wi-Fi (мінімум 10 Мбіт/с) | Google Chrome 90+, Firefox 88+, Safari 14+ |

Таблиця 2.2 – Users Infrastructure View

Мережа:

* пропускна здатність: 1 Гбіт/с;
* зовнішнє підключення до Інтернету: мінімум 100 Мбіт/с.

## 3.16 Delivery Strategy View

Система побудована на мікросервісній архітектурі, де кожен сервіс (управління продуктами, складами, сповіщеннями тощо) контейнеризовано через Docker. Для доступу користувачів використовуються React SPA та REST API.

Оновлення здійснюється через CI/CD з використанням Blue/Green та Canary Deployment, забезпечуючи безперебійність роботи. Комунікація між сервісами реалізована через Kafka (асинхронно) та API (синхронно).

Захист даних забезпечується JWT-токенами, SSL/TLS та ролями (RBAC). Масштабованість досягається завдяки горизонтальному масштабуванню сервісів і Load Balancer.

# 4 РЕАЛІЗАЦІЯ ВЕБ САЙТУ

# 5 ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ВЕБ-САЙТУ.

# ВИСНОВКИ

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

# ДОДАТОК А – Сповіщення про недостатню кількість товарів

