КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок для управління запасами ліків»

Виконав _____

(Підпис)

Молчанов Богдан Станіславович

(прізвище, ім'я, по батькові)

Керівник _Краснощок Віктор Миколайович

(прізвище, ім'я, по батькові)

_____

(Резолюція «До захисту»)

Попередній захист:

_____

_____

(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.

(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок для управління запасами ліків»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Молчанов Богдан Станіславович	

Назва роботи українською та англійською мовами

Веб-застосунок для управління запасами ліків Web service for drug inventory management

Мета бакалаврської роботи, завдання

Мета бакалаврської роботи: Підвищення ефективності управління запасами лікарських засобів

План роботи:
1. Сучасні підходи до розроблення і впровадження веб сервісів
2. Аналіз архітектурних рішень і вибір програмних засобів для реалізації веб – систем
3. Програмна реалізація веб-сервісу управліннями запасами ліків

ПІБ, ступінь, звання наукового керівника роботи: _Краснощок В.М., к.т.н., доцент

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	Виконав
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	Заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	Виконав
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	Виконав
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	Виконав
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	Виконав
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	Виконав
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	Виконав
9.	Подання роботи у першому варіанті	11.05.2021	Виконав
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	Виконав
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	16.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Зміст пояснювальної записки (перелік питань під час дослідження)

Складові частини дипломної роботи	Обсяг, арк. 78
Титульний аркуш	1 ст.
Завдання до дипломної роботи (календарний план проекту)	1 ст.
Відомість дипломної роботи	1 ст.
Анотація	1 ст.
Анотація (іноземною мовою-англійською)	1 ст.
Зміст	1 ст.
Вступ	3 ст.
1. Загальносистемні питання. Аналіз розв’язуваної задачі й огляд наявних результатів. Постановка задачі та проектування	24 ст.
2. Проектні і технічні рішення. Види забезпечення	29 ст.
3. Опис роботи програми	6 ст.
Висновки	1 ст.
Перелік посилань	4 ст.
Додатки	5 ст.

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.	Макаренко С.А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Дипломна робота: 78с., 19 рис., 4 табл., 40 джерел, 2 дод.

Ця дипломна робота присвячена проектуванню та розробленню веб-застосунку для управління запасами ліків.

Метою дипломної роботи є створення застосунку із ефективним управлінням запасами ліків у аптечних закладах України.

Для досягнення поставленої мети треба вирішити такі **завдання**:

- Дослідити теоретичні основи побудови програмної системи управління складом медичних препаратів та системи пошуку лікарських засобів;
- Провести аналіз необхідних технічних рішень для зручної побудови системи;
- Спроектувати і реалізувати програмну систему «Веб-застосунок для управління запасами ліків».

Об'єкт дослідження - процес управління запасами ліків.

Предмет дослідження - програмні засоби для управління запасами ліків.

Методи дослідження.

- Тестування та порівняння існуючих програм-прототипів.
- Аналіз контенту вітчизняних і іноземних інтернет-форумів різної спрямованості (вибірково-тематичний аналіз).
- Логічні методи і прийоми пізнання: аналіз, синтез, аналогія, узагальнення, моделювання, абстрагування, індукція.

Ключові слова: програмна система, управління запасами, аптечні заклади, технологія ASP.NET, технологія React.

ABSTRACT

Thesis: 78 pages, 19 figures, 4 tables, 40 sources, 2 appendices.

This thesis is devoted to the design and development of web service for drug inventory management.

The purpose of the thesis is to create an application through effective management of drug stocks in pharmacies in Ukraine.

To complete the goal, the thesis has to solve the **following problem**:

- Investigate the theoretical foundations of building a software system for managing the composition of drugs and drug search system;
- Analyze the necessary technical solutions for convenient construction of the system;
- Design and implement a software system "Web service for drug inventory management".

The object of study is the process of drug inventory management.

Subject of study - software for drug inventory management.

Research methods.

- Testing and comparing existing prototype programs.
- Analysis of the content of domestic and foreign Internet forums of different orientation (selective thematic analysis).
- Logical methods and techniques of cognition: analysis, synthesis, analogy, generalization, modeling, abstraction, induction.

Keywords: software system, inventory management, pharmacies, ASP.NET technology, React technology.

ЗМІСТ

Вступ

Сучасні аптечні заклади завжди мають потребу у контролі запасів лікарських засобів. Нажаль, станом на сьогоднішній день людство не в силах побороти значну потребу в медичних препаратах. Як показує світова пандемія коронавірусу 2019-nCoV, неналежна і повільна реакція на медичні проблеми серед населення може призвести до колапсу світової економіки на рівні із Великою депресією 1930-х років, наслідки якої люди будуть відчувати протягом наступних кількох років.

Статистика поширення коронавірусної хвороби вказує на те, що найбільшого розростання і пагубного впливу хвороба набула у здебільшого таких країнах, де мало розвинена медична сфера, зокрема її інформаційно-технологічна частина.

Окрім вакцинування, яке стало теоретично доступним лише через рік після початку пандемії, дієвим способом побороти хворобу стало симптоматичне лікування. Після діагностики лікарі виписували пацієнту рецепт, згідно якого захворілий пацієнт повинен знайти список лікарських засобів якомога швидше і почати лікування.

На превеликий жаль, окрім можливої проблеми низької економічної забезпеченості населення, у недостатньо забезпечених сучасними інформаційними технологіями країн, серед захворілих постала проблема пошуку ліків. Не всі аптечні заклади можуть вести облік лікарських засобів, та що стає ще більшою проблемою – не всі аптечні заклади мають змогу із наявним програмним забезпеченням надати клієнтам можливість переглянути актуальну наявність медичних препаратів у закладі, а тим паче у конкретних відділеннях конкретних населених пунктів.

Розвиток технології Інтернет має дуже високий вплив на кожну сферу життя громадян. Пристрої, які використовуються для доступу в мережу Інтернет стають дедалі доступнішими для кожного. Сьогодні дуже складно знайти хоч

одну родину, де не буде жодного смартфона, ноутбука, настільного ПК або планшетного ПК.

Високий рівень доступності до мережі Інтернет у населення дає змогу спроектувати систему управління запасами ліків, яка буде мати можливість взаємодіяти із навколишнім світом, надаючи можливість не просто корегувати і керувати наявністю препаратів на складі аптеки, а й можливість надати пацієнтам клінік доступ до перегляду наявності лікарського засобу у потрібному аптечному відділенні. Ще минулого десятиліття було фактично неможливо дізнатися, чи наявні потрібні ліки у запасах хоча би одного аптечного закладу міста чи району. Сьогодні ж це питання вирішується створенням веб-застосунку для управління запасами лікарських засобів із доступом до мережі Інтернет та можливістю відкрити статус наповнення складу для всіх користувачів веб-мережі.

Актуальність дослідження теми бакалаврської роботи зумовлена важливістю ведення аптечними закладами електронного журналу складських запасів медичних препаратів за засобів медичного захисту та можливістю часткового відкриття інформації про наявність таких засобів у конкретних відділеннях всіх необхідних населених пунктів в сучасних реаліях для поліпшення стану здоров’я населення і, відповідно, світової економіки. Наразі в Україні тільки планується введення системи управління складськими запасами для аптечних закладів, що також додає актуальності даній роботі.

Наукова новизна одержаних результатів - виявлено змістовні особливості аптечної системи управління запасами ліків як спеціалізований спосіб керування складом препаратів та засобом надання пацієнтам інформації про наявність ліків у відділенні аптечного закладу.

Мета дослідження:

- Дослідити теоретичні основи побудови програмної системи управління складом медичних препаратів та системи пошуку лікарських засобів;

- Провести аналіз необхідних технічних рішень для зручної побудови системи;
- Спроектувати і реалізувати програмну систему «Веб-застосунок для управління запасами ліків».

Завдання дослідження – створення веб-застосунку для управління запасами ліків.

Об’єктом дослідження є процес управління запасами ліків.

Предметом дослідження є програмні засоби для управління запасами ліків.

Методи дослідження – системний аналіз роботи й застосування клієнт-серверної архітектури для досліджуваної системи та збір інформації про лікарські засоби й їх облік. Дослідження проводиться за допомогою наступних методів дослідження:

1. Тестування та порівняння існуючих програм-прототипів.
2. Аналіз контенту вітчизняних і іноземних інтернет-форумів різної спрямованості (вибірково-тематичний аналіз).
3. Логічні методи і прийоми пізнання: аналіз, синтез, аналогія, узагальнення, моделювання, абстрагування, індукція.

Теоретичне значення роботи: розглянуті можливості сучасних систем з управління запасами лікарських засобів.

Практичне значення одержаних результатів – полягає в можливості використання теоретичних положень і результатів емпіричного дослідження в навчальному процесі і побудові програмної системи за темою бакалаврської роботи.

РОЗДІЛ 1. ЗАГАЛЬНОСИСТЕМНІ ПИТАННЯ. АНАЛІЗ РОЗВ'ЯЗУВАНОЇ ЗАДАЧІ Й ОГЛЯД НАЯВНИХ РЕЗУЛЬТАТІВ. ПОСТАНОВКАЗАДАЧІ ТА ПРОЕКТУВАННЯ

1.1 Огляд поняття «система управління запасами». Моделі управління запасами на підприємствах

Веб-застосунок із теми бакалаврської роботи «Веб-застосунок для управління запасами ліків» - це поєднання двох понять:

- Система управління запасами;
- Система пошуку ліків.

В першу чергу це саме система управління запасами ліків. Така програма створюється для підприємців, власників і працівників аптечних відділень, які у сучасних реаліях повинні відмовитись від ведення паперових журналів і довіритися електронним інформаційним системам з ведення управлінням запасів.

Розглянемо поняття «система управління запасами»: в англійській мові його часто називають «Warehouse Management System» (система управління складом). У літературі також зустрічаються варіанти «inventory management system» та «warehouse complex management system». Переклад в своїй суті вони мають той же самий. Система управління запасами – це сукупність методів управління, навантажених правилами і показниками, які дозволяють створювати, змінювати, зберігати запаси та визначати момент і обсяг необхідної закупівлі продукції для поповнення запасів та циклічно відслідковувати необхідність їх поповнення, що почали своє існування ще в 1980-х роках разом із розвитком технології зберігання даних «в реальному часі».

Для комерційного підприємства дуже важливим є параметр забезпечення стійкості наявного асортименту товарів задля підтримки високого рівня обслуговування клієнтів. Якщо підприємство не має можливості надати покупцю необхідний товар в необхідній кількості – таке підприємство може втратити базу клієнтури. Задля вирішення такої проблеми бізнесу необхідні саме системи

управління запасами, що надають можливість безперебійно обслуговувати покупців на оптимальному рівні, коли на логістичному рівні стає неможливим дефіцит або надлишок запасів.

Система управління запасами повинна вміти своєчасно реагувати на дії користувачів над складом запасів. Це означає, що при будь-яких діях над системою їй необхідно відслідковувати ці дії та динамічно оцінювати новий стан складу з урахуванням отримання нових даних про зміну в системі. Наприклад, при продажу (видаленню) зі складу певної кількості елементів товару, система повинна оперативно відреагувати й автоматично створити й запропонувати надіслати замовлення необхідної кількості нового товару для поповнення сховища.

Використання системи управління запасами потребує певного досвіду у роботі з інформаційними технологіями та вміння керування реальними поставками товарів на склад. До правильного налаштування система може не побороти й не знати про певні нюанси, які викликають складнощі при веденні складу. Відтак, персонал, який використовує таку систему, повинен вміти передбачити:

1. Можливі випадкові коливання попиту на товар за період в інтервалах між постачанням. Наприклад, в деяких аптечних закладах під час першого спалаху пандемії коронавірусу виникла велика недостача протизастудних засобів та препаратів вітаміну-D, проте була можливість цього уникнути завдяки надійній контролю запасів та достатньому рівні підготовки персоналу;
2. Географічну віддаленість відділення чи складу від постачальника, що не дозволяє отримати необхідні засоби у той час і в тому обсязі, в якому вони необхідні;
3. Урахування сезонності збуту або виготовлення товарів, що можуть змінити кількість поставки або потреби у необхідній сировині у певні

періоди річного проміжку та негативно повпливати на репутацію місця збуту серед споживачів;

4. Ризик зміни ринкових цін на кінцеву продукцію.

Як вже стало зрозуміло, основне завдання системи управління запасами і аналізу товарно-матеріальних запасів – можливість побачити, коли краще замовляти товари та яким повинен бути розмір замовлення.

Необхідно виділити наступні параметри системи управління:

- Точка замовлення, що є контрольним рівнем запасів продукції, при якому необхідним стає виклик повторного створення нового замовлення;
- Нормативний, або ж страховий рівень запасів, що є достатнім рівнем підтримки кількості товару на складі підприємства, відділення чи складського приміщення;
- Частота здійснення закупівель є періодичністю поповнення необхідних запасів продукції та тривалістю в межах інтервалу між двома можливими закупівлями;
- Поповнювана кількість продукції, яка визначає не тільки кількість товару, а й досягнення мінімальних витрат на зберігання запасів згідно до заданих витрат на поповнення.

Система повинна надавати можливість проводити операції із нормування запасів, тобто контролю обсягу для поточних та страхових запасів на економічно обґрунтованих рівнях. Такі величини повинні керуватися за допомогою спеціального програмного забезпечення, яке дає можливість проаналізувати залишки товарів на початку і в кінці місяця, порівняти їх і допомогти підприємцю, користувачу системи зробити висновки для регулювання складських ресурсів, підтримуючи їх на необхідному рівні в залежності від попиту серед споживачів та періодів доставки нових запасів, не допускаючи надлишку чи недостачі запасів.

В предметі теорії управління запасами було розроблено дві системи управління, які допомагають досягти мету безперервного постачання споживачу матеріальних ресурсів у ситуаціях, коли відсутні відхилення від своєчасно запланованих показників і запаси завжди споживаються рівномірно, а саме система з управління запасами з фіксованим розміром замовлення і система управління запасами з фіксованим інтервалом часу між замовленнями.

Окрім двох даних систем, існують і їх модифікації, такі як система з встановленою періодичністю поповнення запасів до встановленого рівня і система "мінімум-максимум".

Із даних модифікацій для побудування нової системи управління запасами необхідно вибрати одну або створити нову, поєднавши декілька систем в одну. Нижче проведемо детальний аналіз всіх сучасних технологічних систем управління запасами.

Система з фіксованим розміром замовлення – це модель управління запасами, в якій встановлюється чітко зафіксована константа на оптимальну величину замовлення, після чого вона не ні за яких умов не змінюється. Оптимальний розмір замовлення в такій системі вираховується з урахуванням мінімізації загальних витрат на зберігання і повторне створення замовлення.

Розмір резервного запасу встановлюється для передбачення дефіциту на запаси через логістичні причини, такі як часовий інтервал між замовленням та отриманням поставки запасів.

Граничний рівень запасу визначає, коли рівень запасу є максимально критичним для здійснення наступного замовлення. Граничний рівень розраховується так, щоб надходження замовлення до відділення чи складу відбувалося в момент зниження поточного запасу торгівельних матеріалів до резервного рівня.

Максимально бажаним запасом є сума розміру резервного та розміру нової поставки, що визначається з урахуванням особливостей самого товару, попиту на нього і доступним місцем на складі підприємства.

У такій системі замовлення на постачання нової партії продукції здійснюється при умові досягнення наявним складом запасу мінімального критичного рівня, який задається «точкою замовлення». Інтервали постачання залежать від інтенсивності продажів запасів споживачам, тобто, потреба у продукті серед клієнтури, оптимальний розмір замовлення, інтервал між надсиланням замовлення і отриманням товарів згідно замовлення є регулюючими параметрами для даної системи.

У деяких таких системах точка замовлення не фіксується заздалегідь. Замість цього момент подачі замовлення відбувається або з урахуванням виконання постачальником точних зобов'язань з доставки товару, або з урахуванням коливання споживчого попиту на продаж запасів. У такому випадку точку замовлення називають «плаваючою».

Часто таку систему скорочено називають «системою двох бункерів», так як запас зберігається ніби в двох сховищах, один з яких витрачається з моменту отримання чергової партії запасів до моменту досягнення критичної точки замовлення, а другий – у період між періодом подачі замовлення і виконанням постачальником своїх зобов'язань.

Система управління запасами з фіксованим розміром замовлення рекомендується до застосування, якщо:

- Можливі комерційні провали при відсутності запасів;
- Вартість зберігання великої кількості запасів є значною;
- Недопустимі перевищення поповнюваної кількості замовлення через високу вартість продукції у постачальника;
- Ще не зрозумілий рівень попиту;
- Зниження ціни на товар при замовленні конкретних кількостей в партії;

- Постачальницька партія має мінімальний поріг замовлення.

Серед мінусів таких систем головним чином являється високий вхідний поріг для співробітника сховища облікованих ресурсів, що зумовлений необхідністю вміти правильно обрати мінімальну точку відправлення замовлення. Неправильний вибір точки замовлення в даній системі може призвести до значних бізнесових втрат.

Система управління запасами з фіксованим інтервалом часу між замовленнями, або ж система періодичного замовлення, де чітко встановлений період закупівель, але обсяг закупівель – різний. Згідно до цієї системи власники складу повинні кожного разу в конкретний період, наприклад, кожен тиждень чи місяць, перевіряти залишки товару та створювати замовлення на основі спостереження за збутом складських запасів.

Величина замовлення в кожний період між постачаннями може бути різна. Через це така модель системи допустима до використання лише за умови, що постачальник може забезпечити логістичний перевіз різної величини запасів у кожній окремій партії. Така практика буде недопустима, якщо склад великий і поставки здійснюються по кількості контейнерів або вагонів. Часто постачальник може запросити більшу вартість замовлення за гнучку зміну габаритності товару. В таких випадках використання періодичної системи замовлення також неможлива.

Додатково варто відмітити, що така система може допустити дефіцит запасів. Попит на товар може різко і неочікувано зрости в інтервалі між замовленнями, що спровокує нестачу товару.

Така модель рекомендована до впровадження у випадках, коли:

- Методи доставлення запасів дозволяють змінити розмір партії;
- Тарифи серед постачальників на перевезення і розміщення замовлення не високі;
- Нестача споживчого товару не є проблемою для підприємства.

Систему періодичних замовлень можна назвати відносно простою через внесення правок в неї лише один раз в період між поставками, проте проблема дефіциту товарів є критичною для аптечних закладів.

Система із заданою періодичністю поповнення запасів до встановленого рівню є поєднанням двох перелічених вище систем. Така система найкраще підійде для впровадження у підприємства, де можуть бути присутні часті значні коливання споживчих запасів.

У такій системі замовлення поділяються на планові, які оформлюються в кінці кожного заданого періоду, де розмір наступного замовлення вираховується на основі прогнозованого рівню продажів до наступного моменту завершення поставки нової партії, та на додаткові, які оформлюються при досягненні запасами на складі критичної точки замовлення.

У системі «мінімум-максимум» замовлення формуються виключно за умови досягнення мінімального або меншого рівню запасів на складі чи підприємстві, при чому розмір замовлення нових запасів завжди має бути таким, щоб негайно поповнити рівень ресурсів до максимального. При цьому виникає високий ризик отримання дефіциту запасів.

Така модель замовлень допустима лише за умови значних витрат на зберігання запасів та їх поновлення, коли необхідно швидко збути товар будь-якою ціною, навіть ціною створення дефіциту.

Підсумовуючи матеріал даного підрозділу, можна сказати, що інженери розробили достатню кількість моделей управління запасами на складі, проте найбільш підходящою для аптечних закладів залишиться тільки система із заданою періодичністю поповнення запасів до встановленого рівню через передбачення можливого дефіциту медичних засобів та готовність до сезонних змін попиту на препарати.

1.2 Поняття веб-застосунку

Програмний застосунок – це програма або система, яка встановлюється безпосередньо на пристрій споживача і може бути використана без доступу в мережу Інтернет. Така програма потребує попереднього встановлення необхідних драйверів та середовищ запуску програм, як наприклад будь-яка програма, написана на мові програмування C# потребує такої ж версії середовища запуску .NET Runtime, як і версія .NET Framework або .NET Core, в інакшому випадку користувач отримає помилку на запуску застосунку. Також такі застосунки можуть займати багато місця на пристроях підприємства, що породжує додаткові витрати на закупівлю комплектуючих у вигляді жорстких або твердих накопичувачів.

Програмні застосунки потребують великих затрат на розробку через неможливість запуску одного коду на різних платформах і пристроях. Розроблені програми для операційних систем Microsoft Windows, Google Android чи Apple IOS потребують знань в різних мовах програмування та вміння адаптувати додатки під різні операційні системи та пристрої. Відповідно, це підвищує вартість виконання робіт для створення і подальшого покращення системи, що зовсім не вигідно для замовника.

Окрім проблем із витратами на розробку та адміністрування пристроями, на яких запускається програмна система, існує ризик зловмисницької атаки, що може виникнути як внаслідок спрямованого так і випадкового порушення безпеки адміністрування, таких як встановлення небезпечного неліцензійного програмного забезпечення, приєднання незнайомих зовнішніх електронних пристроїв запам’ятовування інформації чи завантаження файлів з невідомих ресурсів та поштових повідомлень.

Проблеми використання програмного забезпечення на основі роботи з операційною системою можуть призвести до значних втрат на підприємстві, а саме локального чи глобального витоку інформації про підприємство, його співробітників чи бази даних підприємства, перебоїв у роботі пристроїв чи

виводу пристроїв з ладу, що часто може призвести до довгострокового порушення режиму роботи певних ланок або всього підприємства.

Необхідно боротися із поставленими проблемами, і одним із найбільш дієвих способів гаранту безпеки інформаційних технологій всередині підприємства є використання веб-застосунків, що використовують хмарні технології для збереження даних.

Веб-застосунок – це комплексна інформаційна технологія, яка взаємодіє з клієнтом через веб-браузер та доступ до мережі Інтернет, що надає можливість клієнту зв’язатися із веб-системою обробки даних, що обробляє та обчислює запити відповідно до команди, що надсилає клієнт, за необхідності звертається до бази даних, яка знаходиться у спеціальних дата-центрах з підвищеною системою обслуговування апаратних пристроїв та накопичувачів та повертає клієнту відповідь у зрозумілому форматі.

У разі використання веб-застосунку як системне рішення управління запасами, підприємство позбавляється відповідальності за безпеку даних від атак зловмисників, направлених на погіршення стабільності роботи системи. Натомість сервер обробки даних захищений згідно стандартів, заданих розробником, а дані зберігаються у віддаленій хмарній базі даних і не підлягають видаленню. Єдиним способом нанести шкоду підприємству у разі нападу на систему буде викрадення даних для входу до веб-застосунку, таких як паролі співробітників та неправильне розподілення ролей управління системою.

Окрім високого рівню безпеки програмної системи у веб-застосунку, її замовники отримують можливість використовувати таке програмне забезпечення на всіх доступних апаратних платформах, які підтримують можливість виходу в мережу Інтернет та містять підтримку встановлення сучасних версії веб-браузерів, що означає кросплатформенність. Не важливо, з якого пристрою був відправлений запит на отримання чи обробку даних на стороні серверу, сервер завжди повертає дані у такому форматі, яку зрозуміє

клієнт у веб-браузері. Тобто, таку систему можна відкрити як із ноутбуку, планшету, так навіть із смартфону.

Для використання веб-застосунку системні адміністратори підприємства не зобов’язані завантажувати будь-яке стороннє програмне забезпечення, займаючи місце на фізичних носіях пристроїв компанії. Тобто, за умови роботи з веб-системою замість прикладних програм підприємство може зекономити на пристроях зберігання даних, купуючи накопичувачі з найменшим об’ємом, необхідним лише для встановлення на них будь-якої зручної операційної системи. А кросплатформенність веб-браузерів дозволяє не встановлювати на жорсткі диски великі і навантажені системи з лишнім функціоналом, замість яких можна поставити простіші в роботі і більш оптимізовані системи на основі ядра Linux.

Архітектурна система веб-застосунків цілком ізольована від користувача. Користувач лише завантажує веб-сторінку, вводячи у пошуковий рядок назву або адресу підключення до веб-системи. Користувач не повинен проводити будь-які додаткові налаштування. Логіка роботи програми знаходиться на виділеному сервері, який адмініструється розробниками такої системи.

Структура роботи веб-додатків принципово відрізняється від роботи прикладних систем, які зберігаються і виконуються виключно на пристрої, на якому вони були запущені. Нижче надано пояснення роботи веб-застосунків згідно рис. 1.1.

Перед поясненням діаграми необхідно отримати розуміння понять асинхронності і синхронності процесів:

- Процес називається синхронним, коли він виконується в рамках одного потоку, тобто наступна дія після даного процесу почнеться тільки тоді, коли поточна дія буде закінчена;

- Асинхронним процес можна вважати тоді, коли одразу після моменту виконання команди для його початку починає виконуватися наступний процес.

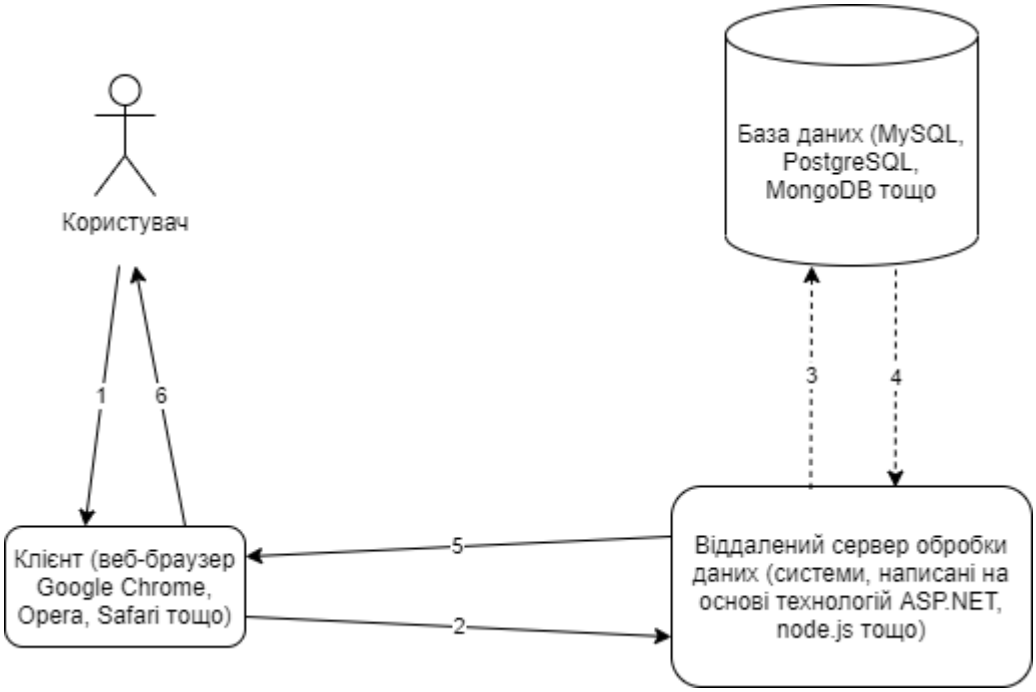

Рисунок 1.1 – Базова структура використання веб-застосунку

Пояснення структури роботи веб-застосунків зображеної на рис. 1.1 згідно номерам стрілок XML- діаграми:

1. Користувач вводить запит у веб-браузер. Це може бути прохання показати нову сторінку або запит на управління даними (їх збереження, редагування, видалення тощо);
2. Сторінка, відкрита у веб-браузері, може обробити дані, надані користувачем і надіслати команду у вигляді HTTP-запиту на адресу серверу обробки даних;
3. Сервер обробки даних може обчислити інформацію і без участі бази даних, проте в більшості випадків присутня взаємодія, яка провокує зміну вмісту сутності в базі даних або створення нової.
4. В залежності від типу запиту, база даних синхронно або асинхронно виконує поставлене завдання. У випадку збереження даних, їх

редагування або видалення, база даних не повертає відповідь у зв’язку із асинхронною моделлю роботи процесу. Коли дані необхідно отримати – сервер виконує запит і отримує дані синхронно.

5. Сервер повертає результат дії, який супроводжується HTTP статус-кодом, що дозволяє зрозуміти ступінь успішності виконаної задачі та за необхідності тілом даних, які були отримані в результаті звернення до бази даних та (або) відповідних обчислень. Це можуть бути дані у вигляду набору байтів, що можуть, наприклад, сформувати документ форматів Portable Document Format (PDF) або Microsoft Word з розширенням DOCX, можуть бути дані у вигляді документу JavaScript Object Notation або у вигляді Hyper Text Markup Language (HTML) сторінки.
6. Клієнт веб-браузеру виводить користувачу інформацію про виконання запиту та (або) дані, які користувач запитував на першому кроці.

Як видно, переваги веб-застосунків над прикладними системами є досить очевидними. Проте у такого виду застосунків є і недолік у вигляді низької продуктивності при відображенні величезної кількості даних або тривимірної графіки. Проте тривимірна графіка не відноситься до теми даного дослідження, а відображення великої кількості даних можна сортувати, вводячи у веб-застосунок технологію пагінації сторінок, що дозволяє відображати не весь набір даних із сховища на сервері, а лише його частину, фільтруючи дані у лімітах відображення певної кількості одиниць на одній сторінці та даючи можливість користувачу переключати ці сторінки.

Висока продуктивність коду є важливою частиною створення веб-застосунку. Якщо у прикладних застосунках допускається низький рівень оптимізації програмної системи з розрахунком на використання більш продуктивних апаратних застосунків, то особливістю веб-застосунків є можливість відкрити систему на системах із найбільш слабким апаратним забезпеченням. Окрім цього, продуктивність платформ і операційних систем, на

базі яких можна запустити веб-браузер, дуже варіюється від найслабших мобільних пристроїв до продуктивних настільних рішень. Через це необхідно постаратися забезпечити якомога більшу плавність відображення інтерфейсу і якнайшвидшу архітектуру роботи системи на стороні як клієнтської частини застосунку, так і серверної і бази даних.

Популярність веб-застосунки набули із розвитком мови програмування JavaScript. До початку XXI століття веб-сайти здебільшого були статичними. Технології не дозволяли впровадити високонавантажену систему у веб-браузер. Проте сьогодні JavaScript дозволяє розробити будь-яку систему управління даними у веб-браузері.

Отож, через високий рівень гнучкості, високу доступність та простоту у доступі веб-застосунки на сьогоднішній день набули надзвичайно високої популярності серед підприємців та користувачів мережі Інтернет. Такий тип архітектури програми дозволяє розгорнути застосунок на будь-якому пристрої, навіть не проводячи жодних завантажень і встановлень. При правильному підході до розробки веб-застосунок може працювати із великою кількістю даних, а окрім цього вона дозволяє використовувати одну копію застосунку для покриття всіх бажаних клієнтів, для чого достатньо лише додати додаткові механізми у розробку системи авторизації, зокрема можливість реєструвати мережі підприємств та створювати окремі відділення для них. За потреби простота розгортки веб-застосунку може зіграти ключову роль у швидкому захопленні ринку систем управління запасами для фірм із продажу фармацевтичних товарів.

1.3 Огляд існуючих систем управління запасами

У даному підрозділі будуть розглянуті системи управління запасами для проведення аналізу над існуючими рішеннями, знаходження їх переваг і недоліків та створення висновку, що підкреслить найважливіші рішення впроваджених систем та виявить можливості, які додатково необхідно реалізувати в новій системі, розроблюваній в рамках даної дипломної роботи.

Нижче розглянемо наступні актуальні станом на дату підготовки бакалаврської роботи системи управління запасами на підприємстві:

- Zoho Inventory Management;
- Oracle;
- Галактика;
- 4Psite;
- SAP.

Система Zoho є однією з найбільш відомих систем на зарубіжному ринку. Zoho – повноцінна система для ведення бізнесу в електронній комерції, яка дозволяє провести інтеграцію із такими відомими мережами, як Amazon, eBay та Shopify.

Zoho дозволяє в рамках однієї системи проводити управління прямими поставками, створювати замовлення та отримувати рекомендації по веденню бізнесу. Ця система більше підходить для підприємців, бізнес яких пов'язаний із продажом товарів через доставку, наприклад, для власників сучасних інтернет-магазинів. Інтерфейс сторінки управління замовленнями даної системи зображений на рис. 1.2.

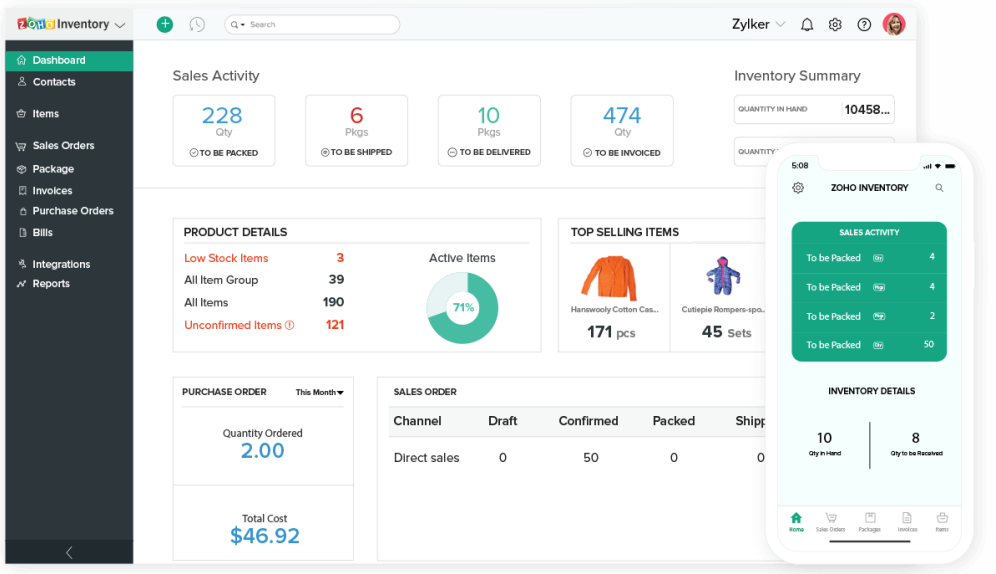

Рисунок 1.2 – Інтерфейс управління замовленнями Zoho

До плюсів даної системи можна віднести:

- Інтегровану систему управління відносин з клієнтами;
- Повноцінну систему відслідковування запасів;
- Інтеграцію із найбільшими площадками електронної комерції;
- Зрозумілий програмний інтерфейс;
- Низька ціна у порівнянні з конкурентами і безкоштовний доступ до багатьох функцій;
- Інструменти управління маркетингом і людськими ресурсами;
- Можливість управління замовленнями через прикладну програму у смартфоні.

Серед мінусів відгуки користувачів системи виділяють:

- Система інтеграції із зазначеними ресурсами не завжди працює коректно;
- Жахлива документація роботи з системою;
- Повільна робота з оновленнями системи після частих скарг користувачів;
- Затримка в роботі синхронізатора із хмарою при великих кількостях замовлень, що викликає некоректну роботу в інтеграції з такими сервісами, як Amazon;
- Після переходу із безкоштовної версії на платну скидаються дані по співробітниках.

Отож, система Zoho не є досконалою і може слугувати хорошим варіантом для використання виключно як програмна система управління невеликими ресурсами запасів та система з управління продажів у малому та середньому бізнесі. Така система не підходить для використання на великому ринку електронної комерції через велику кількість несправностей, які недопустимі для більшості підприємств.

Перейдемо до огляду системи Oracle. Повна назва системи управління запасами від цієї компанії – Oracle Warehouse Management Cloud. Це прикладна програма із підтримкою хмарних технологій, яка славиться зручним інтерфейсом і високою швидкістю роботи.

Це по-справжньому провідний лідер у сфері логістики. Інтеграції, судячи з відгуків користувачів, працюють майже з усіма можливими системами. Офіційний сайт заявляє, що система швидко впроваджується і гнучка у налаштуваннях. Інтерфейс системи показаний на рис. 1.3.

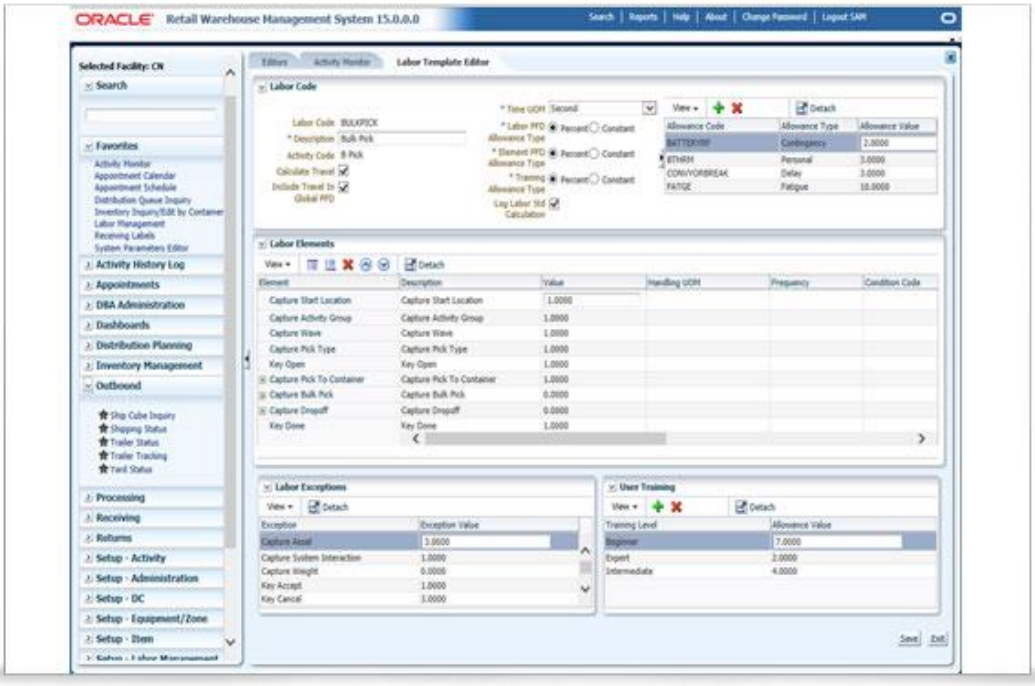

Рисунок 1.3 – Інтерфейс системи управління запасами Oracle

Серед переваг і можливостей можна виділити:

- Управління запасами із поставки дозволяє відправити запаси прямо в інвентар, друкувати наліпки зі штрих-кодами, налаштовувати правила розміщення запасів на складі;
- Забезпечення відслідковування проданих запасів від виходу зі складу до полички магазину;
- Управління роботи персоналом, що дозволяє відстежити його ефективність;

- Широкий функціонал для аналітики складських операцій;
- Наявність рекомендацій щодо вибору постачальників.

Користувачі системою виділяють наступні недоліки:

- Занадто велика кількість налаштувань, система перенавантажена функціями, як не потрібні для роботи малих підприємств;
- Застаріла система. Наприклад, для пошуку, окрім фільтрів, використовується мова по принципу схожа на SQL, але не настільки популярна і знайти документацію та рекомендації до її використання на сьогоднішній день складно;
- Повільна робота служби підтримки.

Така система хоч і є світовим лідером у сфері логістики, та все ж являється застарілою і використовує функціонал, мало описаний у документації. Через складність у знайомстві з системою необхідно витратити чимало часу на ознайомлення з нею та навчання персоналу. В той же час служба підтримки працює повільно, що для малих підприємств, які обирають першу логістичну систему управління запасами, може стати ключовою причиною для відмови у використанні даної системи.

Комлекс «Галактика» являється не світовим лідером в управлінні запасами, проте є явним лідером у СНГ-регіоні. Система вміщає в себе широкий комплекс для виконання задач із оперативного управління та стратегічного планування.

Система надає можливість купувати лише необхідні компоненти. Вона побудована на сервіс-орієнтованій архітектурі, що дозволяє легко інтегрувати програмні системи від сторонніх виробників. Також «Галактика» надає можливість працювати з системою у вигляді веб-сервісу. Окрім управління запасами, продукт пропонує роботу з управління фінансами, виробництвом, персоналом, проектами, логістикою, бухгалтерським обліком та цінними активами. Інтерфейс систем изображений на рис. 1.4.

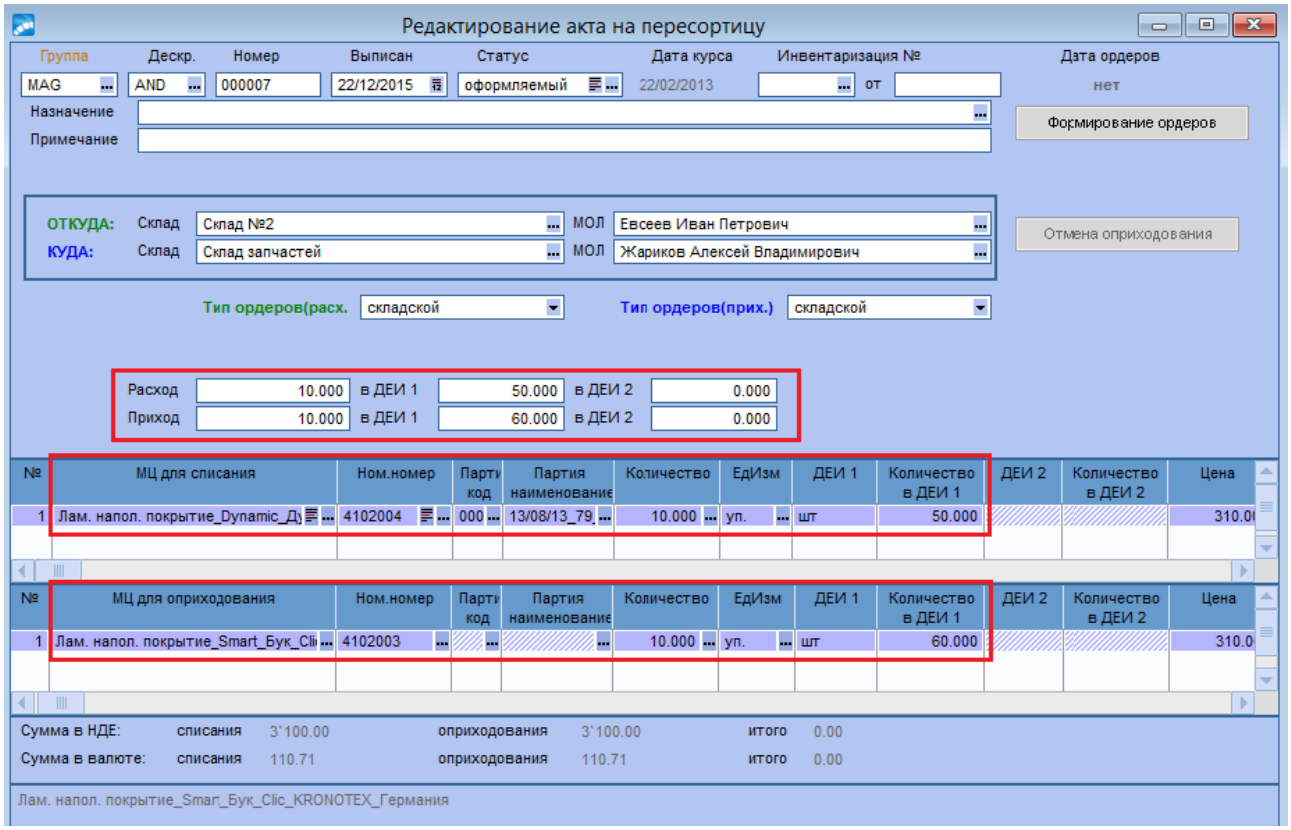

Рисунок 1.4 – Прийом поставки у складське приміщення у системі «Галактика»

Модуль з управління запасами у системі «Галактика» містить наступні функції:

- Можливість реєстрації і збереження заявок на послуги від покупців або замовників, що можуть оброблятися автоматично на основі аналізу документів, наданих в заявці. При формуванні заявки замовник повинен вказати кількість товару для замовлення та ціну. Після отримання система обчислює приблизний період, який знадобиться для виготовлення та відпуску товару;
- Автоматичне створення підготовленого списку замовлень, які надсилалися підприємству за період планування на основі налаштувань власника системи;
- Автоматичне формування плану продажу послуг на основі підготовлених списків;
- Можливість виводу аналітичних форм на основі роботи з системою.

На основі відгуків користувачів системою з’являються і негативні сторони:

- Повільна робота системи і високе споживання ресурсів комп’ютера;
- Надзвичайно складний і непродуманий інтерфейс;
- Проблеми при передачі даних між значною кількістю модулів;
- Висока ціна у порівнянні з якістю.

Як висновок для даної системи, можна сказати, що «Галактика» є слабо оптимізованою системою, яка більше підходить для управління замовленнями і продажами невеликої логістичної компанії, ніж для управління запасами аптечної мережі, де головне – простота і швидкість використання та відсутність надлишкових модулів.

Система управління запасами 4Psite виділяється серед попередніх систем. Це повноцінна веб-система для підтримки роботи сайтів електронної комерції. Вона дозволяє управляти замовленнями, інвентарем і покупками, способами доставки, фінансами, містить влаштовану систему управління відносин з клієнтами, систему електронного обміну даними та потужну систему аналізу даних. Це дозволяє системі бути мультиканальною, вміщаючи у себе весь тільки потрібний для сайту електронної комерції функціонал. Інтерфейс зображений на рис. 1.5.

Серед переваг системи в модулі управління запасами виділяються:

- Можливість зберігати запаси в кількох локаціях;
- Підтримка управління кількома складами;
- Управління поставками партій;
- Підтримка роботи системи у браузері;
- Автоматичне поновлення запасів;
- Простий інтерфейс у порівнянні з іншими системами через наявність лише найважливішого функціоналу;
- Розробники швидко додають новий функціонал за бажання користувачів;

- Потужна аналітична система.

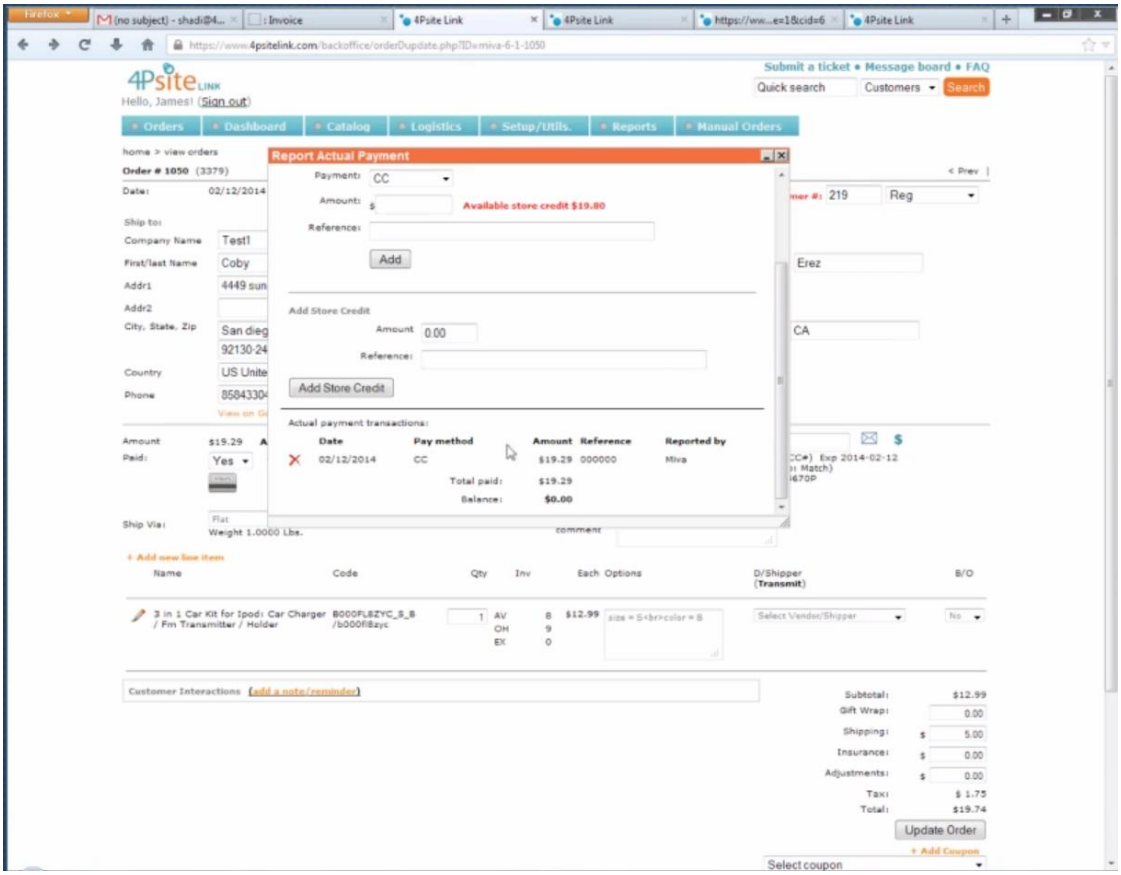

Рисунок 1.5 – Здійснення замовлення у системі 4Psite

Проте є і недоліки. Користувачі скаржаться на застарілий вигляд інтерфейсу, який створеній «ніби у 80-х» та на поганий сервіс синхронізації інвентарів між різними системами.

З такої системи можна взяти приклад у порівняній простоті. Для аптечних закладів вона все ще буде не підходящою, проте структура у вигляді веб-застосунку, який дозволяє в одному місці сполучити всі заклади, можна запозичити.

Останньою системою, яка буде розглянута в даному підрозділі, стане німецька система управління запасами SAP.

SAP дозволяє проводити будь-які операції по управлінню запасами, незалежно від їх складності, надає можливість управління всіма складськими підрозділами, які підприємство побажає ввести до системи. Система навіть

дозволяє відокремлювати особливо небезпечні запаси та створювати та зчитувати бар-коди для швидкого управління системою. Більшість процесів, які в інших системах займають багато часу, SAP дозволяє автоматизувати. Однією із ключових переваг в рамках побудови проекту для дипломної роботи є можливість придбати децентралізовану систему управління запасами, без інтеграції системи управління ресурсами організації (ERP). Інтерфейс модулю списку замовлень в рамках системи управління запасами показаний на рис. 1.6.

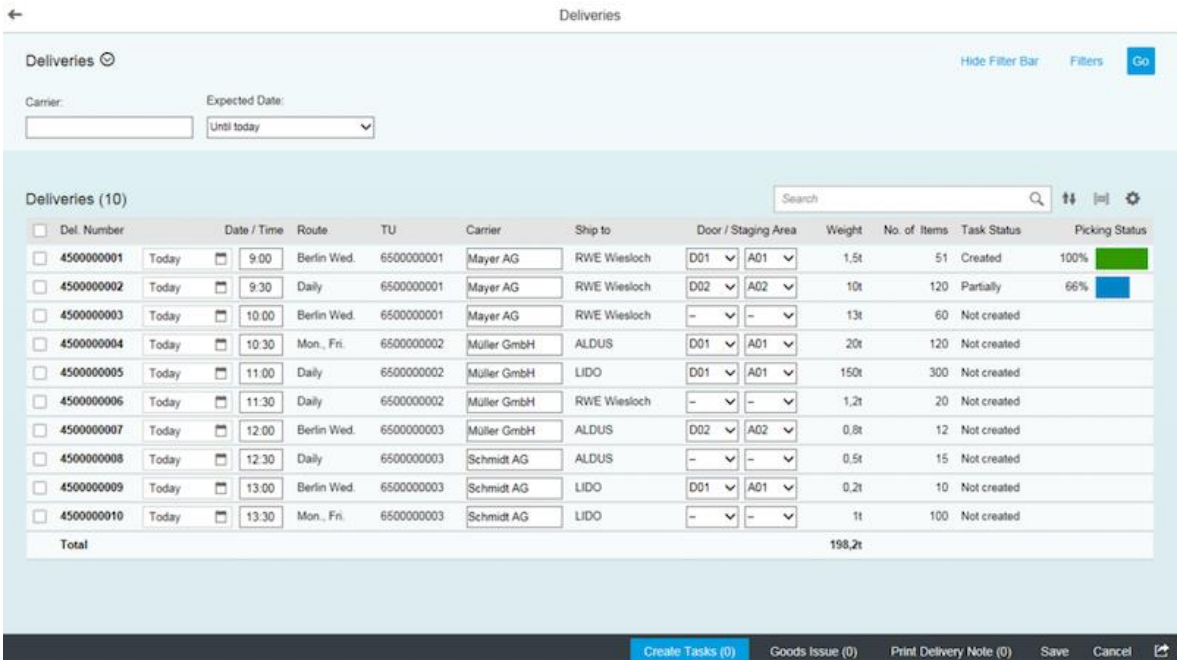

Рисунок 1.6 – Список замовлень у системі SAP

Серед переваг і основних функцій системи управління запасами SAP можна виділити:

- Можливість децентралізації системи;
- Система відстеження пересування запасів;
- Зменшення впливу людського фактору за допомогою автоматизації часомістких процесів;
- Можливість налаштувати масив складських приміщень;
- Можливість за потреби налаштувати функції програми у спеціальному порядку.

Недоліки системи SAP:

- Не підтримує функціоналу друкування рахунків за замовлення
- Має обмежений функціонал для побудови аналітичної звітності;
- Не підтримує функціонал контролю людських ресурсів;
- Не підтримує функціонал проведення оплати;
- Не має інтеграції із бухгалтерськими програмами.

Отож, приклад із системою управління запасів від програмного продукту SAP являється чудовим показником простоти інтерфейсу, яка не потребує великого вкладання часу на її вивчення. Проте, з точки зору користувача, функціоналу може бути занадто мало. Якщо для системи управління запасами, для прикладу, аптечного закладу, явно не потрібен модуль управління людськими ресурсами, то повноцінна система із аналітичними показниками є дуже необхідною.

1.4 Постановка задачі. Технічне завдання на розробку

На основі аналізу знань про побудову і структуру веб-систем, літератури із теорії управління запасами, переглядом інформації про існуючі аналоги, а також враховуючи специфіку збереження, замовлення і продажу запасів на медичних підприємствах із продажу ліків, необхідно постановити задачу, яка буде реалізована в даній бакалаврській роботі.

Існуючі системи мають безліч недоліків як самі по собі, так і для окремих організацій. Відтак, однією із основних проблем існуючих систем, навіть серед світових лідерів, являється перевантаженість інтерфейсу не потрібними для користувача модулями. Через перевантаженість такі системи не можуть працювати на слабких пристроях, бо на них не вистачає ресурсів для швидкої обробки даних. Іноді системи управління запасами навіть не мають хмарної бази даних для збереження «прогресу» ведення складу ресурсів підприємства, через що при несправності у роботі програми або пристрою можлива втрата даних, у зв’язку з чим бізнес може понести великі втрати. Окрім того, системи, дані яких зберігаються безпосередньо на пристроях, можуть бути втрачені у зв’язку із

зловмисницькими атаками на пристрої підприємства, навмисними чи ненавмисними. Також такі системи не дозволяють проводити чітку синхронізацію із віддаленими відділеннями тієї ж організації при переведенні запасів із приміщення одного відділення до іншого, яке може знаходитися у іншому місті.

Через це застосунок повинен бути саме веб-застосунком, працювати на віддаленому сервері, звертаючись до хмарної бази даних. Така архітектура захистить дані користувачів від зловмисників та не буде викликати навантаження на пристрої користувачів. Віддавання даних із сервера на клієнт повинно відбуватися швидко і дозовано, не допускаючи появи занадто великої кількості даних у браузері клієнта в один час і на одній сторінці. Повинні бути присутні фільтри і можливість перегляду списків даних через вибір сторінки.

Аналізуючи сезонні перепади попиту на медичні товари, система управління запасів у аптеці повинна мати модель, яка не допустить дефіциту за жодних обставин. Такою моделлю є система із заданою періодичністю поповнення запасів до встановленого рівню. Із такою моделлю відповідальна особа буде бачити рекомендації щодо поповнення запасів наприкінці кожного періоду поповнення і до того ж, при високому попиті на товар при досягненні критичної точки рівня запасів буде автоматично сформоване замовлення на покупку нової партії запасів. Найкращим рішенням буде дати користувачу можливість вибору моделі системи управління для кожного товару окремо.

Повинна бути присутня система відправки заявок на замовлення запасів та система прийому замовлень, яка дозволить поповнити запас на відділенні. Система повинна бути багатою на вже внесені у довідник лікарські засоби, які зареєстровані у державному реєстрі лікарських засобів України та можливість додати нові препарати у довідник при формуванні замовлення. Для поповнення складу обов’язково необхідно спочатку створити заявку на отримання нової партії, після чого підтвердити прибуття партії із даної заявки на склад підприємства.

Необхідно передбачити і систему збуту товару, яка дозволить змоделювати систему збуту запасів зі складу. Такою системою стане каса, у якій фармацевт кожного відділення зможе поштучно продати медичні препарати із запасів. Окрім цього, необхідно додати систему видалення препаратів зі складу і окрему сторінку зі списком препаратів, термін придатності яких закінчився або скоро закінчиться.

Програма повинна бути побудована на архітектурі клієнт-сервер для уникнення навантажень на пристрій користувача. Жодні обчислення не мають проходити на стороні клієнтського застосунку. Сам браузер повинен тільки відображати дані, які приходять по запитам із серверу. Бази даних повинні вміти зберігати величезні об’єми даних. Одне аптечне відділення в день може прийняти десятки замовлень, а система повинна бути побудована для використання необмеженою кількістю відділень, тому теоретично програмна система буде працювати із величезною кількістю збережених даних. Мова програмування для побудови серверу обчислень повинна бути достатньо універсальною для реалізації готової до великих навантажень системи масового обслуговування.

ВИСНОВКИ

У першому розділі були розглянуті ключові поняття із побудови систем управління запасами на підприємстві, розглянута різниця між прикладними системами та веб-застосунками а також проаналізовані найбільш популярні сучасні програмні рішення, які пов’язані із темою управління запасами.

За допомогою теоретичних відомостей у даному розділі були сформовані основні вимоги до побудови системи управління запасами із урахуванням необхідності використання цієї системи тільки у медичних організаціях із продажу ліків.

РОЗДІЛ 2. ПРОЕКТНІ І ТЕХНІЧНІ РІШЕННЯ. ВИДИ ЗАБЕЗПЕЧЕННЯ

2.1 Інформаційне забезпечення проектованої системи

Технічне завдання на розробку із першого розділу вимагає розробки системи у вигляді клієнт-серверної архітектури. Побудова системи у такій реалізації дозволить простіше вносити зміни у процесі підтримки проекту через розділення програми на логічні частини, серед яких:

- Клієнт, який дозволяє вводити і виводити дані;
- Сервер, що займається обчисленням прикладних функцій, які запрограмовані згідно до потреб області дослідження;
- Сховище, яке реалізує збереження даних.

Клієнтська частина повинна використовувати мову програмування, яка може без труднощів взаємодіяти із сервером. Окрім цього, вона має бути достатньо популярною, що допоможе знайти і прочитати більше документації по ній і відповідно, створити більш універсальне програмне забезпечення.

Для створення клієнтської частини проекту використовується мова програмування JavaScript. Вона дає можливість керування потоком даних на клієнтській стороні. Це – так звані м’язи сторінки (рис. 2.1). JavaScript дозволяє додати сторінці динамічність, можливість взаємодії з нею.

Рисунок 2.1 – Структура веб-сайту

JavaScript є кросплатформенною технологією, що дозволяє запустити програми, створені на її основі, на будь-якому пристрої. Проте, в рамках дослідження дипломної роботи, ця технологія буде використовуватись у веб-браузері, що також являється кросплатформенною технологією.

Для динамічної побудови нових елементів на сторінці HTML-документу JavaScript використовує DOM-дерево (Document Object Model). Воно дозволяє динамічно будувати нові елементи на HTML-сторінці та реагувати на взаємодію користувача із елементами. У стандартному DOM-дереві JavaScript при взаємодії запускає створення нової сторінки або додає нові елементи до дерева.

Такі дії є дуже затратними згідно ресурсів системи, тому творці системи соціальної мережі Facebook розробили свою JavaScript-бібліотеку, яка по-іншому взаємодіє із HTML-документами. Ця бібліотека називається React, а заміну DOM-дереву назвали віртуальним DOM-деревом.

React – декларативна, гнучка бібліотека для створення користувацьких інтерфейсів. Дана бібліотека дозволяє розбити інтерфейс на маленькі частини, що називаються компонентами. Такий підхід до розробки підвищує швидкість роботи клієнтської частини, а окрім того ще й дозволяє використати компоненти для розробки інших програмних систем.

Віртуальне DOM-дерево працює за допомогою Stateful-компонентів (state – стан). Стан – це об’єкт, що описує внутрішній стан компоненту, що робить сторінку компонентно-структурованою. Коли стан компоненту оновлюється, веб-сервіс порівнює минуле віртуальне DOM-дерево з новим. При знаходженні змін, компонент, який використовує змінений стан, оновлюється без використання потреби у зміні структури всієї сторінки.

Система управління запасами ліків повинна використовувати систему авторизації на основі управління через ролі. Вона необхідна для розмежування можливостей різних користувачів у роботі із системою. Наприклад, адміністратор системи повинен перевіряти, чи організація, що реєструється,

реальна, а фармацевт повинен мати можливість збуту ліків через продаж, перевіряючи їх наявність на складі та вибираючи їх у списку замовлення покупця відділення.

Одним із поширених і найбільш безпечних методів створення авторизації між користувачем, клієнтом та сервером є авторизація через створення і передачу спеціального ключа, названого JWT-токеном (JSON Web Token).

При введенні вхідних даних користувача, клієнт надсилає його логін та пароль до серверу. Там перевіряється наявність профілю даного користувача у системі і відповідно до цього видається відмова або створюється тимчасовий ключ авторизації, який передається клієнту разом із потрібними клієнту даними про користувача. Такий ключ також містить в собі інформацію про сервер і права, надані користувачу при вході до системи.

Після отримання із серверу JWT-токену, клієнт має зберегти його для виконання подальших викликів методів серверу. Стандартні методи бібліотеки React дозволяють зберігати стани тільки у компонентах, проте JWT-токен необхідно зберігати для всього застосунку. Для збереження стану даних у системі дипломної роботи використовується технологія Redux. Вона дозволяє зберігати для кожного клієнта стан не тільки в окремих компонентах, а і у всій системі в цілому. Redux - це глобальне сховище документу змінних системи, що зберігається у форматі JSON і дозволяє обійти систему передачі станів бібліотеки React із принципом «зверху-вниз».

Після формування віртуального DOM-дерева сторінка однаково перетворюється в документ з HTML-розміткою (Hyper Text Markup Language). Це гіпертекстова розмітка, що використовується для створення й відображення веб-сторінки.

HTML – основа будь-якої сторінки, її каркас. Сучасний HTML 5 описує і семантику вмісту документу, складається з тегів та їх атрибутів.

Під час перегляду HTML-документа браузер інтерпретує його, будуючи структуру DOM, відображаючи її відносно коду, прописаному в файлі. Інтерпретація при цьому починається ще до того, як сторінка повністю завантажується на веб-сторінці. Перед цим браузер «будує» структуру, використовуючи так само й CSS (Cascading Stylesheets).

CSS – код, який використовується для стилізації сайтів. Він дозволяє унікально оформити будь-який елемент DOM-дерева HTML-сторінки. Можна змінювати кольори, розміри, положення, шрифти в документі. Також за допомогою CSS можна створювати унікальні анімації, наприклад, при використанні додаткових бібліотек або JavaScript-коду, можна створити анімацію прокрутки зображень.

За допомогою CSS можна адаптувати розміри веб-сторінки під роздільну здатність будь-якого екрану, наприклад дисплею ноутбуку, телефону, планшету чи телевізору.

CSS також дозволяє імпортувати вже готові стилі, розроблені іншими авторами. Для спрощення роботи із стилями використовується бібліотека Bootstrap. На основі препроцесора SASS Bootstrap дозволяє спростити користування «анатомією» CSS коду, змінюючи правила внесення його структури а також дає можливість використовувати вже готові, розроблені користувачами мережі Інтернет стилі, для оформлення елементів HTML, таких як форми, кнопки, текстові поля і подібні.

Прикладним застосунком для обробки клієнтських даних на сервері буде слугувати Web API (Application Programming Interface, інтерфейс прикладного програмування) на основі модульної платформи для програмування .NET Core 5. Така система дозволяє отримувати і обробляти дані за допомогою типізованої мови програмування C#.

Web API дозволяє створити таку веб-службу, до якої за наявності необхідних дозволів може звернутися будь-який клієнт. Тому вона має бути

надзвичайно продуктивною на програмному і апаратному рівні для своєчасного обчислення всіх даних від масового клієнта. .NET 5 надає таку можливість.

Для обробки запитів ASP.NET Core використовує спеціальні класи-контролери, які унаслідують клас Controller. Вони дозволяють прописати визначений Route (шлях) для запуску необхідних нам дій, оброблюваних сервером та повернути дані у вигляді, наприклад, файлу в форматі JSON (JavaScript Object Notation). При отриманні запиту клас-контролер починає синхронне обчислення даних, повертаючи відповідь клієнту одразу після виконання всіх процесів, запрограмованих у контролері. Однак, контролер може запустити і асинхронний потік, що буде обчислений у фоні.

Також .NET дозволяє створити так звані «воркери». Це прикладні рішення, які працюють безперервно, виконуючи обчислення за певних умов. Наприклад, в рамках даної системи воркер може постійно перевіряти рівень запасів кожного продукту і при досягненні ними критичної точки викликати метод створення замовлення на нову партію товару.

Перевагою .NET у розробці є надзвичайно високий рівень підтримки середовища програмування як серед розробників системи, компанії Microsoft, так і від користувачів, які з кожним днем додають тисячі нових сторінок документації та сотні статей на тему використання технології. Також слід відмітити, що дана технологія є кросплатформенною, тобто її можна розгорнути для запуску в будь-якій операційній системі чи архітектурі.

Вибір бази даних є останнім із головних моментів у виборі напрямку розробки. Всі сучасні бази даних досить швидкі, проте існує різниця у швидкодії між великим і малим розміром даних, а також у виборі підходу до їх збереження.

Реляційні бази даних, такі як MySQL або PostgreSQL не дозволяють зберігати різні типи даних для однієї змінної, тобто колонки. Можна тільки змінити її тип даних, проте запуск команд із такими змінами зупинить усі процеси звернень до бази даних і поставить їх у чергу, що при великих об’ємах

даних може заблокувати роботу бази даних на кілька годин, що може негативно повпливати на бізнес-процеси.

Отож, нереляційні бази даних будуть гнучким професійним рішенням для великих об’ємів даних, які зберігаються у одній колекції. У свою чергу, реляційні бази швидше працюють із відносно малими об’ємами даних та незамінні при складних запитах (агрегаціях), які одночасно працюють із кількома таблицями даних.

Враховуючи проведений вище аналіз порівняння реляційних та нереляційних баз даних, можна розподілити навантаження на дві різні бази: MongoDB та PostgreSQL.

Принцип зберігання інформації в MongoDB дуже відрізняється від стандартних, реляційних баз даних із чітко структурованими таблицями, схемами, SQL-запитами, зовнішніми ключами та іншими об’єктно-реляційними рішеннями.

MongoDB використовує документно-орієнтовану модель даних. Завдяки такій структурі база даних працює швидше та її використання значно спрощується. Поняття таблиць в MongoDB заміняється поняттям колекцій.

Потрібно розуміти принципово новий для розробників на стандартному SQL формат зберігання даних у MongoDB. Для цього MongoDB використовує формат BSON (Binary JSON), що дозволяє швидше виконувати функції пошуку і обробки даних. Недоліком такого формату слугує більший розмір документів порівняно із звичайним JSON, проте BSON виправляє це перевагою у вигляді високої швидкості роботи із індексованими даними. Окрім того, можна створювати один індекс на кілька властивостей даних, які до того ж можуть мати різні типи.

MongoDB працює швидше не тільки за рахунок формату зберігання даних, а й через свою основу розробленої архітектури, що побудована на мові програмування C++. Більше того, MongoDB дозволяє запускати в свою консоль виконання команд на мові JavaScript, які виконає сервер бази даних.

PostgreSQL – об’єктно-реляційна система управління базами даних. Вона підтримує більшість функцій стандарту SQL та додає нові можливості, такі як дозвіл на створення власних типів даних, функцій, операторів та навіть створення власних процедурних мов виконання SQL-запитів. Ця система дозволяє легко проводити виконання великих транзакцій, що є основою запитів до баз даних у системах управління запасами.

Довідник медичних препаратів, дані про замовлення, дані про рівень запасів на складі та дати їх поповнення, тобто інформація, яка буде сягати величезних розмірів, буде індексуватися і зберігатися у базі даних PostgreSQL.

MongoDB, як нереляційна база даних, буде використовуватись для збереження інформації, що потребує швидкої реакції на запити та низький час обробки. Це можуть бути дані про користувачів, про корпорації, що використовують систему і про відділення, в яких вона використовується. Також MongoDB буде зберігати історичні дані про продаж товарів, так як нереляційні бази чудово працюють із горизонтальним зчитуванням даних.

Підсумок усіх використаних технологій у вигляді їх логотипів для впізнаваності зображений на рисунку 2.2.

Рисунок 2.2 – Технології, використані для побудови веб-застосунку

2.1.1 Розподіл ролей та структура доступних їм сторінок у проектованій системі

Тематика роботи являє собою веб-застосунок для управління запасами ліків у аптечному відділенні. Таке рішення дозволяє організувати єдину систему доступу до управління запасів для всіх організацій та користувачів, що в ній зареєструються. Окрім цього, у даній роботі реалізується механізм пошуку наявних у відділеннях ліків для неавторизованих користувачів. Тобто, клієнти підприємств зможуть бачити, у яких відділеннях поруч із пацієнтом наявні необхідні лікарські засоби.

Така система потребує великих затрат на час розробки. Проте його можна зменшити, підготувавши чітку документацію до побудови системи. В даному підрозділі буде розпочата аналітична складова для розробки, а саме вимоги до побудови структурованих сторінок веб-системи для кожної системної ролі.

Так як однією із головних скарг на системи-конкуренти була перевантаженість інтерфейсу – це необхідно виправити в рамках дипломної роботи. Інтерфейс не повинен містити модулі, які являються лишніми для функціоналу управління запасами у аптеках та адмініструванні веб-застосунку.

Розмежування функціоналу, необхідного для побудови веб-сторінок для кожної ролі у системі подано у таблиці 2.1.

Таблиця 2.1 – Ролі та їх функціонал

Назва ролі	Опис доступного функціоналу
Незареєстрований користувач	- Пошук аптечних відділень за спеціальними фільтрами по адресі;

Продовження до таблиці 2.1

Назва ролі	Опис доступного функціоналу
Незареєстрований користувач	- Пошук медичних препаратів у рамках всієї системи за фільтрами по назві та формі випуску; - Подання заявки на створення організації та профілю директора організації.
Директор організації	- Перегляд статистичних даних по роботі із запасами відділень організації; - Реєстрація фармацевтів і управляючих запасами на існуючих відділеннях; - Надсилання заявки на технічну підтримку.
Адміністратор системи	- Перегляд і управління заявками на реєстрацію організацій і відділень організацій та запити стосовні технічної підтримки; - Перегляд статистичних звітів кожної організації; - Додавання нових медичних препаратів до довідника.

Продовження таблиці 2.1

Назва ролі	Опис доступного функціоналу
Фармацевт відділення організації	- Управління продажами запасів
Управляючий запасами відділення	- Створення запиту на отримання нової партії запасів; - Прийом нової партії запасів від постачальника; - Надсилання заявки на технічну підтримку.

Отож, згідно таблиці 2.1 для клієнтської презентативної частини застосунку необхідно побудувати і стилізувати наступні сторінки:

- Сторінка пошуку відділень у системі;
- Сторінка пошуку медичних препаратів у всій системі;
- Сторінка відділення із пошуком медичних препаратів у ньому;
- Сторінка створення заявки на реєстрацію організації та профілю директора організації;
- Сторінка створення заявки на реєстрацію відділень;
- Сторінка управління персоналом відділення;
- Сторінка створення заявки із зверненням до технічної підтримки системи, наприклад, із проханням додати новий препарат до довіднику медичних засобів;
- Сторінка перегляду статистики по закладу та системі;
- Сторінка перегляду заявок від користувачів;
- Сторінка управління запасами: із можливістю налаштовувати модель поповнення запасів, створення заявки на замовлення до постачальника, підтвердження збуту запасів, термін придатності яких вичерпано.

Керування переходами між сторінками на клієнтській частині веб-застосунку керується використанням бібліотеки React, а налаштування і створення ролей – на серверній стороні з боку системи .NET. При створенні профілю користувача йому присвоюється певна роль. Дана роль може перевірятися перед виконанням запиту певного методу обчислення серверу або використовуватися безпосередньо у такому методі. При створенні авторизації JWT-токену користувачу на клієнтську частину також приходить його роль, яку запам’ятовує Redux. На клієнтській частині застосунок може відображати вміст компонентів React в залежності від ролі користувача, яка перевіряється перед відображенням HTLM-сторінки. У неавторизованих користувачів ролі немає, тому вони також обмежені в доступі, отримуючи доступ лише до того функціоналу, який не містить в собі перевірку на роль співробітника, директора чи адміністратора.

Рисунок 2.3, як висновок, показує стислу UML-діаграму (UML – Unified Modeling Language, уніфікована мова моделювання) взаємодії користувача із веб-застосунком згідно аналітичних вимог до побудови структури веб-сторінок.

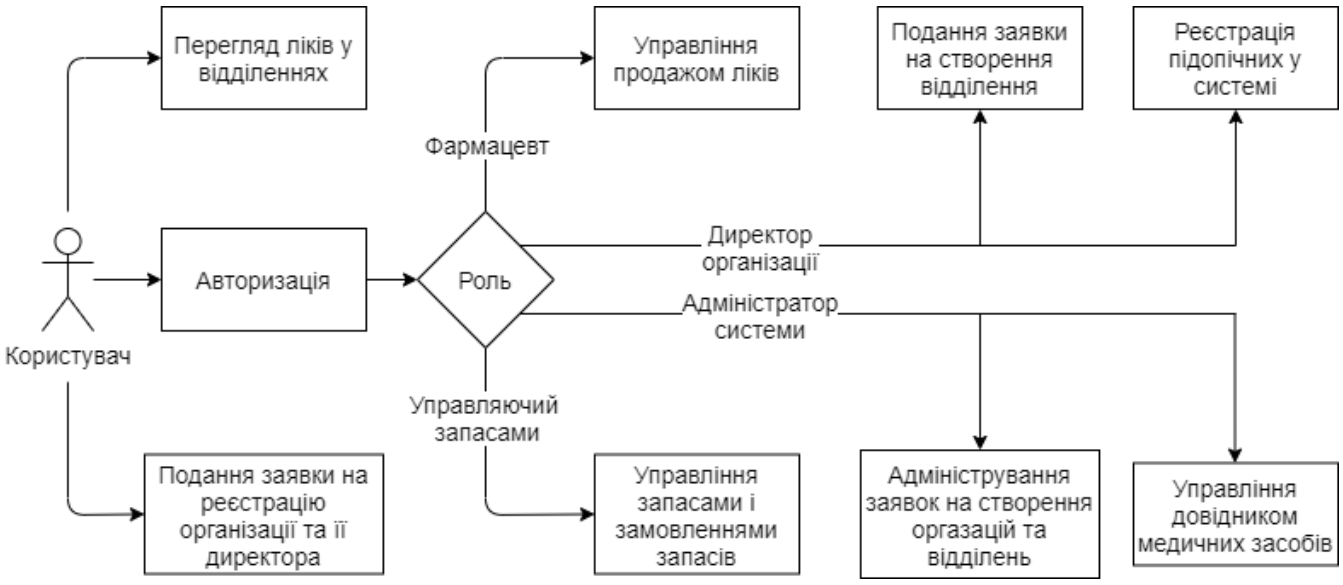

Рисунок 2.3 Розмежування взаємодії користувача із веб-застосунком

2.1.2 Функціональні можливості проектованої системи

Після проведеного аналізу теоретичних відомостей з теми управління запасами, аналізу рішень від конкурентів, формування загальних вимог та розподілу веб-сторінок відносно узагальненого функціоналу необхідно сформувати уточнені вимоги щодо роботи кожної частини системи. Точні вимоги сформовані у даному підрозділі.

Система в цілому повинна містити наступний функціонал:

- Створення заявки на реєстрацію в системі організації та її відділень;
- Реєстрація співробітників організації;
- Управління запасами відділення;
- Продаж запасів у відділенні;
- Аналітична сторінка директора;
- Управління довідником запасів;
- Управління заявками на реєстрацію організацій та відділень;
- Пошук лікарських засобів;
- Система авторизації.

Першим розглянемо функціонал для створення електронний заявок для реєстрації в системі організації та її відділень.

Даний функціонал надає користувачу можливість надіслати автоматично структуроване повідомлення адміністратору системи на основі вказаних даних про структуру закладу. Спочатку повинна бути подана заявка на створення організації разом із профілем її директора. Реєстрація організації може вважатись підтвердженою, якщо адміністратор схвалив таку заявку і надіслав повідомлення на пошту директора.

Після реєстрації організації система дає можливість надіслати заявки і на створення відділень аналогічним способом, проте уже тільки для авторизованих директорів організацій.

Функціонал заявок створений через необхідність відображати в частині системи для пошуку ліків та відділень тільки існуючі заклади і обмеження входу і використання системи для закладів, які не попереджують про наміри співпраці із розробниками веб-застосунку. Така практика буде дуже корисна, якщо застосунок буде впроваджений у реальне використання.

Функціонал реєстрації співробітників організації повинен бути доступним директору організації після створення відділення. Серед ролей, які може створити директор – фармацевт та управляючий складом. Через те, що співробітники відділення не відображаються в рамках функціоналу пошуку ліків для незареєстрованих користувачів, для реєстрації таких співробітників не потрібні додаткові підтвердження. Ця відповідальність лежить виключно на підприємстві, яке надає права доступу директора адміністратора фізичній особі.

Управління запасами у відділенні – основний функціонал бакалаврської роботи. Згідно теоретичних вимог, він має базуватись на основі методу системи із заданою періодичністю поповнення запасів до встановленого рівню. В рамках аптечних закладів така система не потребує функції відслідковування відправки товару, інтеграції з іншими системами та відстеження масштабів складських приміщень, так як зазвичай у аптечних відділень складські модулі – це маленькі кімнати із запасами, де рівень наповненості сховища легше дослідити наочно, аніж натискаючи лишні кнопки в системі.

Основними логічними методами в управлінні запасами повинні бути створення, редагування та видалення із системи слоту із назвою медичного препарату та характеристиками, необхідними для створення замовлення.

При додаванні товару до системи складу користувач повинен налаштувати характеристики товару, «точку замовлення», при досягненні якої система автоматично буде замовляти товар та (або) періодичність замовлення нових партій. Окрім цього, має бути налаштована максимальна кількість товару на складі, аби при використанні системи періодичних замовлень не створювати надлишки на складі.

В рамках функціоналу замовлення товарів управляючий запасами повинен вибрати постачальника, маючи можливість ввести нового або вибрати із уже існуючих в базі застосунку. Коли нова партія прибуває на відділення, управляючий повинен підтвердити прибуття, після чого партія буде успішно додана до складу відділення.

Для правильної роботи системи необхідно налаштувати сервіс, який буде працювати асинхронно у фоні на сервері, постійно звертаючись до бази даних і перевіряючи рівень запасів товару задля автоматичного створення нового замовлення при досягненні «точки замовлення».

Функціонал списку запасів для управляючого складу повинен надавати користувачу таблицю з фільтрами. У даній таблиці без вибору фільтрів будуть відображатися всі товари згідно наступного сортування:

- Товари, які необхідно замовити згідно із досягненням періоду замовлення;
- Товари, які досягли кінця терміну придатності. Виділення цієї категорії необхідне для підтвердження видалення таких товарів зі складу;
- Всі інші товари, фільтровані по зростанню кількості ресурсів в одиницях.

Відповідно до цих даних, у списку можна буде застосувати фільтр по відображенню тільки запасів, які необхідно поповнити, товарів, які необхідно звільнити зі складського приміщення та всіх товарів, які придатні до продажу.

Кожна категорія товару повинна виділятися окремим кольором. Відтак, перша категорія має оформитись у жовтих тонах, друга – у червоних, третя – у нейтральних, синіх.

Для фармацевтів виділяється функціонал продажу запасів. Це можна вважати окремим модулем, інтегрованим в систему управління запасами для їх збуту. Не потрібно проводити інтеграцію із іншими системами оплати, натомість аптечні заклади будуть використовувати просту систему продажу в рамках

одного веб-застосунку. Використання власної системи продажу також покращить вивід статистичних даних для директора організації, де він буде бачити статистику по продажам.

У процесі продажу товару фармацевт обирає товар зі списку запасів, які присутні на складі відділення. Якщо товар раніше був на складі відділення, але закінчився – користувач отримає відповідне повідомлення. Таке саме повідомлення користувач отримає стосовно товарів, присутніх на складі, але термін придатності яких закінчився. Якщо товар був обраний для продажу і оплата була успішно підтверджена – введена кількість ресурсу віднімається від загальної кількості лоту на складі. Не можна продати кількість товару більшу, ніж присутню на складі.

Як і в будь-якій конкурентній системі з управління запасами, повинна бути присутня аналітична сторінка. Доступ до неї надається тільки директору організації. В ній за вибраний період, з мінімальним порогом вибору в один день, повинна відображатися статистика по приросту та збуту товарів, кількості замовлень, збитків та прибутку по організації та по кожному відділенню окремо.

Адміністратору системи доступний функціонал управління довідником медичних засобів. Через систему він може додати певний лікарський засіб, вказавши його характеристики. Внесення таких даних повинне проходити із високим рівнем відповідальності адміністратора задля попередження випадків додавання не існуючих медичних препаратів, які зможе використовувати кінцевий користувач. Препарати адміністратор вносить згідно до прохань користувачів, які надходять через влаштовану систему підтримки, або згідно зміни державного реєстру лікарських засобів.

Перед додаванням препарату до системи, адміністратор повинен ввести назву лікарського засобу у пошуковий довідник із вже наявними в системі препаратами чи засобами для уникнення створення дублів.

Також адміністратор повинен керувати системою внесення до застосунку нових організацій та відділень. Заявки на реєстрацію може надіслати будь-який незареєстрований користувач, а адміністратор в свою чергу повинен перевірити, чи вже існує ця організація в системі, чи зареєстрована ця організація в єдиному державному реєстрі юридичних осіб та за необхідності зв’язатися із особою, що реєструє заклад, через надані нею засоби зв’язку задля уточнення інформації. Структуру відділень також має перевіряти адміністратор системи задля перевірки правильності наданих директорами адрес чи координат.

Система пошуку лікарських засобів грає важливу роль у даному застосунку. Це є однією із ключових особливостей дипломної роботи, яка вирізняє дану систему управління запасами серед всіх конкурентів.

Будь-якому користувачу мережі Інтернет надається можливість переглянути список всіх ліків, доданих до системи та їх наявність у найближчих відділеннях. Таку можливість дають пошукові фільтри, які дозволяють відфільтрувати область, район та назву населеного пункту. Разом з цим, ліки можна шукати по приблизному збігу в назві, що в кінцевому результаті дозволить швидко знайти медичний засіб, який цікавить потенційного клієнта відділення організації, що зареєстрована в системі.

Система авторизації необхідна для розподілу ролей і прав доступу в організаціях. Вона працює таким чином:

- При першому запуску системи формується профіль адміністратора. У нього спочатку сталий пароль, який безпечно задається в конфігураціях серверної частини застосунку.
- Незареєстровані користувачі можуть подати заявку на створення організації. Цю заявку підтверджує адміністратор. Разом із організацією створюється і профіль директора організації;
- Директор організації може створювати профілі для фармацевтів та управляючих складом у відділеннях.

Первинні логіни та паролі для всіх користувачів надають ті, ким вони були зареєстровані. Після першого входу пароль необхідно змінити.

2.1.3 Схеми та опис баз даних

Раніше в цьому розділі був проведений аналіз основних характеристик нереляційних та реляційних баз даних. Згідно нього, у системі будуть використовуватися дві бази даних: MongoDB для зберігання інформації про авторизацію користувача і PostgreSQL для збереження даних про товари, замовлення, партії замовлень, історичні дані та статистичні дані.

MongoDB відіграє роль сховища даних про організації, їх відділення та користувачів. Схема бази умовно зображена на рисунку 2.4. Зв'язки на рисунку додані виключно для кращого розуміння взаємодії документів. Слід також зазначити, що для полів ідентифікаторів у колекціях використовуються текстові значення унікальних глобальних ідентифікаторів GUID (Globally Unique Identifier) замість стандартних ідентифікаторів MongoDB BsonId. Така заміна необхідна через те, що PostgreSQL не підтримує такий тип даних.

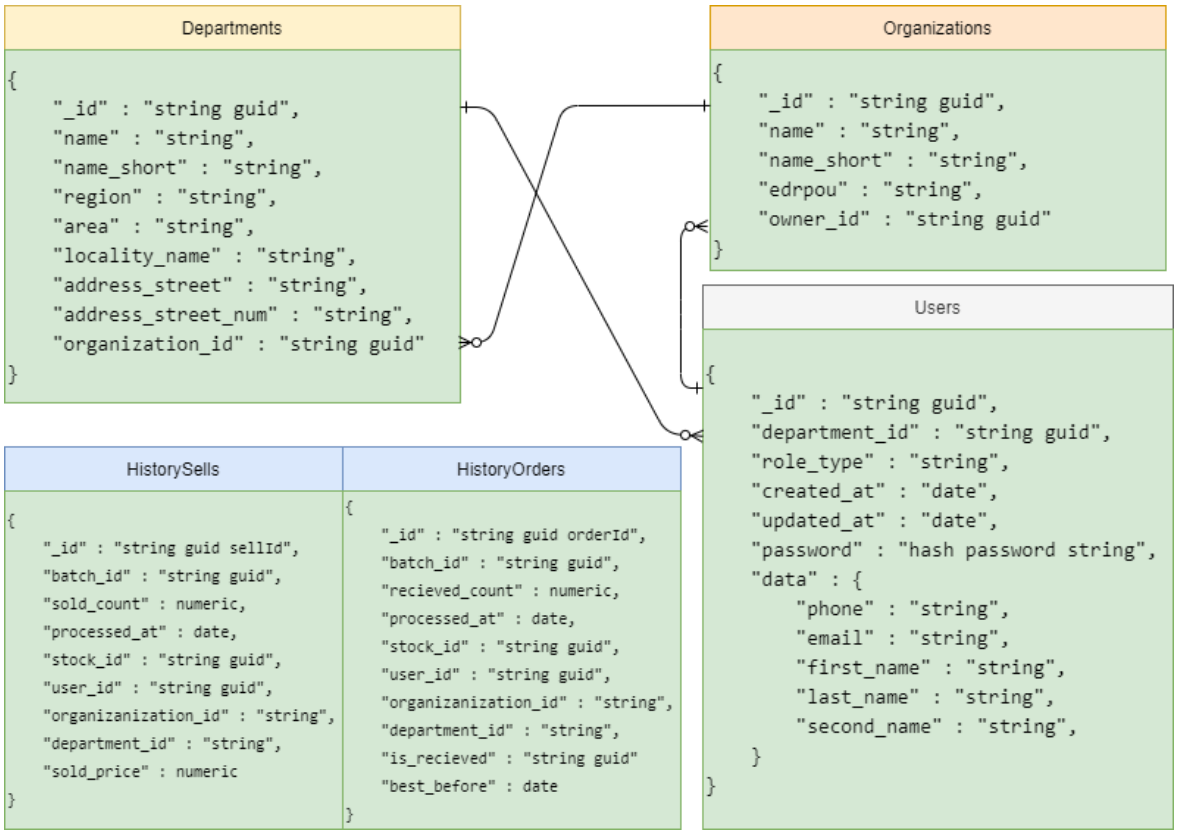

Рисунок 2.4 – Схема бази даних MongoDB

Документи у колекції організації Organizations містять короткі, але надзвичайно важливі дані про організацію в системі – її код єдиного державного реєстру підприємств та організацій України (ЄДРПОУ), назву та ідентифікатор директора організації.

Колекція із відділеннями, або ж Departments, містить в собі інформацію про прив’язку відділень до організацій, адресу відділень, яка поділяється на область, район, населений пункт, адресу та адресний номер, а також про назву.

Колекція із користувачами Users вміщає в собі документи, що описують зареєстрованих користувачів системи. Такі документи містять інформацію про ідентифікатор відділення (який указується не обов’язково, так як адміністратор системи не має організації), роль користувача, його пароль та персональні дані, такі як прізвище, ім’я, по-батькові, електронна пошта та номер телефону.

Історичні дані зберігаються у таблицях HistorySells, записи в яку додаються в момент продажу товару фармацевтом та HistoryOrders, яка зберігає в собі моменти обробки замовлення партій при створенні заявок на їх отримання та безпосередньо при позначенні прибуття.

На рисунку 2.5 зображена схема бази даних, які зберігаються у PostgreSQL. Як зазначалося раніше у вимогах до оформлення баз даних, таблиці, що містять великий набір даних, повинні зберігатися у реляційних SQL.

Серед таких наборів даних виділені синім – таблиці взаємодії для управління запасами, червоним – таблиця для збору даних по статистиці.

Основа управління запасами реалізується за допомогою трьох таблиць: таблиці з довідником товарів Product, таблиці із позначенням запасів Stocks та таблиці із позначенням партій Batches. Таблиця Product містить у собі основні дані медичних препаратів та інших товарів, які можуть бути наявними на складі аптечного відділення. Серед таких даних – назва українською та англійською, тип продукту для визначення категорії та додаткова інформація, що слугує

виключно для опису продукту незареєстрованому користувачу, який може скористатися сервісом пошуку ліків.

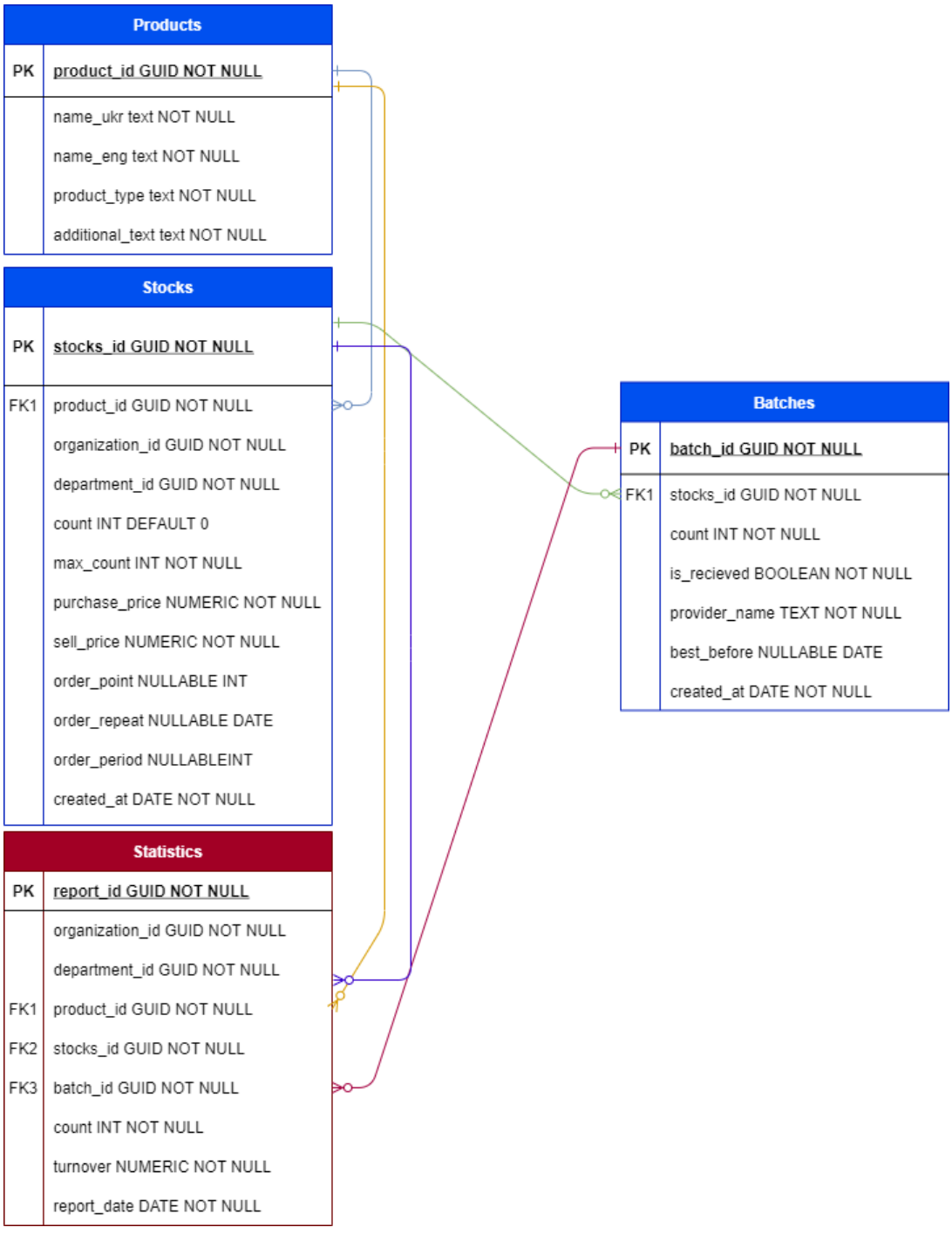

Рисунок 2.5 – Схема бази даних PostgreSQL

Stocks представляє собою сховище запасів певного продукту для кожного відділення. У ній обчислюється сума запасів із таблиці Batches, позначається ціна для закупки і продажу товарів поштучно. Для управління створеннями замовлень у цій таблиці є поля «точки замовлення», періоду замовлення у днях і дата

наступного замовлення. Також існують поля організації та відділення, які зберігаються в іншій базі даних.

Batches – збірник усіх партій товарів, прив’язаних до таблиці із сховищем запасів. У кожної партії є своя кількість, відмітка про її отримання та опціональне поле із датою закінчення терміну придатності товару.

Статистична таблиця показує кількість товару, яка була у відділенні на момент збереження статистики, показує грошовий оборот за день і містить посилання на всі інші таблиці, необхідні для управління запасами.

2.1.4 Системні елементи

Система управління запасами ліків складається із трьох основних системних елементів – бази даних, структура якої була розглянута в попередньому підрозділі, а також елементу клієнтського застосунку і серверного.

Серверна частина розроблена для обробки та обчислення даних. Клієнтська частина розроблена як точка входу і виходу користувацьких даних. Ці дві частини застосунку створюють кодову основу системи.

Для розмежування функціоналу серверної частини використовуються класи-контролери (Controller). Принципи об’єктно-орієнтованого програмування вимагають від розробника певних навичок для створення добре зрозумілого коду. Необхідно вміти відокремити функціональні класи і методи коду так, щоб розбити їх на маленькі частини, що дозволить оброблювати тільки необхідні дані та зберігати певну інкапсуляцію даних.

Розробка в середовищі .NET дозволяє умовно поділити рівні обчислення функціоналу на:

- Контролери – класи, що приймають на повертають дані;
- Менеджери – класи, що перевіряють цілісність даних та підготовлюють їх до обчислення;
- Сервіси – класи, що обчислюють дані;

- Репозиторії – класи, що взаємодіють із базою даних.

Така структура системи дозволяє їй бути більш зрозумілою для інших розробників, що можуть читати код та доповняти його. Також, рівні сервісів та репозиторіїв спрощують процес написання коду, дозволяючи багато разів використовувати одні і ті ж методи, які в різних місцях системи можуть обробляти дані.

Точкою входу до методів серверу слугують методи класів-контролерів, позначених спеціальними шляхами до них. Програмісти називають їх «роутами».

В таблиці 2.2 показані класи-контролери серверної частини веб-застосунку, їх роль у системі та опис процесу обробки даних всередині них.

Таблиця 2.2. Роути серверу та їх опис

Контролер	Опис функціоналу
AuthenticationController	Слугує класом обробки даних користувачів для входу. Містить метод тільки метод для входу користувачів у систему.
UserController	Даний клас дозволяє реєструвати співробітників у системі. Містить методи з управління додавання нових користувачів різних ролей та редагування інформації про них
UserRequestController	Дозволяє користувачам створювати заявку на реєстрацію організації з її директором, реєстрацію відділень, додавання нових засобів до довідника запасів та створення заявок на отримання технічної підтримки без визначеної теми

Продовження до таблиці 2.2

Контролер	Опис функціоналу
AdministrationController	Містить у собі функціонал із управління функціями адміністратора. Через нього можна підтверджувати заявки на реєстрації організацій, директорів та відділень та додавати нові лікарські засоби
SellsContoller	Дозволяє управляти продажом запасів
SearchController	Управляє пошуком інформації про відділення та запаси для незареєстрованих користувачів
WarehouseManagementController	Містить весь функціонал по управлінню запасами. Включає методи по створенню сутностей запасів, управлінню над їх замовленнями, видаленням їх зі складів відділень
ReportsController	Статистичний контролер. Повертає аналітичні дані згідно вимог до статистичної системи, описаних раніше у розділі

Клієнтська частина застосунку замість класів-контролерів містить компоненти. Компоненти у застосунку – це частини основної логіки веб-сторінок, які можуть включати в собі менші компоненти, створені для більшої конкретизації і розбиття функціоналу.

На відміну від контролерів на сервері, компоненти у клієнті слугують для зручного виводу та прийому даних користувача, а також для навігації по системі. Компоненти дозволяють реалізувати механізм повторного використання, тому

деякі компоненти можуть містити лише загальний контекст. Наприклад, таким компонентом може бути зв’язка надпису біля поля вводу.

У клієнті на React також присутні і свої сервіси. Сервіси – це логічні методи внутрішньої обробки даних у мові JavaScript. Таким сервісом може слугувати, наприклад, сервіс авторизації. Він буде приймати із html-сторінки логін та пароль для входу, відправляти їх на сервер та приймати із серверу JWT-токен для авторизації, після чого додаючи його для зберігання до стану Redux.

В таблиці 2.3 показані основні великі компоненти системи, які вирізняються наповненістю унікальним функціоналом взаємодії з користувачем та сервіси.

Таблиця 2.3. Компоненти та сервіси клієнтської частини застосунку

Сервіс/компонент	Опис функціоналу
Сервіс AuthenticationProvider	Повертає методи для авторизації із серверною частиною, утриманням JWT-токену, поверненням запису поточного користувача, реєстрацій та вихід із системи
Компонент NavBar	Шапка веб-сайту. Використовується для простої та зручної навігації всередині веб-сервісу. Знає, які сторінки відобразити для певних ролей.
Компоненти OrganizationRegistration та DepartmentRegistration	Компоненти для оформлення заявок на створення організації та відділень

Продовження до таблиці 2.3

Сервіс/компонент	Опис функціоналу
Компонент App	Серце компонентів бібліотеки React. Містить шапку веб-сайту, Router, де вказані шляхи до всіх компонентів-сторінок та футер (footer)
Компонент HomePage	Головна сторінка, однакова для користувачів всіх ролей. Показує список із препаратів та містить функцію їх пошуку
Компонент WarehouseManagement	Компонент із управлінням над замовленнями. Містить в собі компоненти для створення замовлення, створення заявки на замовлення та видалення запасів зі складу
Сервіс WarehouseProvider	Слугує місцем обробки запитів на управління складськими запасами

Окрім перелічених вище сервісів та компонентів, у клієнтській частині проекту існують і спеціальні статичні допоміжні сервіси. Вони допомагають реалізувати в одному методі функціонал підготовки даних до виводу або обробки. Наприклад, таким методом може бути функція поєднання кількох текстових об’єктів в один із заміною символів.

2.2 Програмне забезпечення

Веб-застосунок в дипломній роботі розроблений за допомогою мов програмування C# (ASP.NET Core Framework), JavaScript (бібліотека React), каскадної мови стилів CSS, мови розмітки гіпертексту HTML та бази даних MongoDB.

Всі ці інструменти кросплатформенні, тобто, дозволяють розробляти програмне забезпечення використовуючи будь-яку архітектурну систему.

Для роботи над проектом була використана операційна система Microsoft Windows 10 та програмне забезпечення JetBrains Rider (для серверної частини) і Microsoft Visual Studio Code для клієнтської частини. Для взаємодії і перевірки даних із нереляційною базою даних використовується хмарне сховище бази даних MongoDB Atlas та програмне забезпечення Robo3T, а для використання реляційної PostgreSQL – програмне забезпечення JetBrains DataGrip.

Всі застосовані рішення – сучасні рішення для продуктивної розробки із високим рівнем підтримки серед розробників у вигляді чітко описаних документацій а також веб-форумів, створених користувачами для спілкування із спеціалістами.

Не зважаючи на велику кількість використаного програмного забезпечення для розробки, для використання кінцевим споживачем не потрібно встановлювати нічого, окрім веб-браузеру, здатного компілювати JavaScript ES10, тобто користувач може використовувати веб-сайт як зі смартфону, так і з планшету, так і з ноутбука чи комп’ютера. Головне – не забути про підключення до інтернету задля первинного завантаження сторінки та зв’язку із сервером.

Для впровадження на сервері можна використовувати будь-яку операційну систему. Наприклад, Windows або Linux. Головне – завантажити пакети і драйвери, необхідні для запуску серверного застосунку.

Встановлення баз даних потребує лише налаштування на серверному пристрої середовища PostgreSQL. MongoDB знаходиться у хмарному сховищі

Atlas, тому не потребує жодних додаткових налаштувань, окрім зміни рядку підключення у файлах конфігурації основного серверного застосунку.

2.3 Опис структури системи

В даному підрозділі розглянуто структуру клієнтської та серверної частини клієнту. Це допоможе краще орієнтуватися всередині програмного коду веб-застосунку.

Схема структури серверної частини зображена на рис. 2.6:

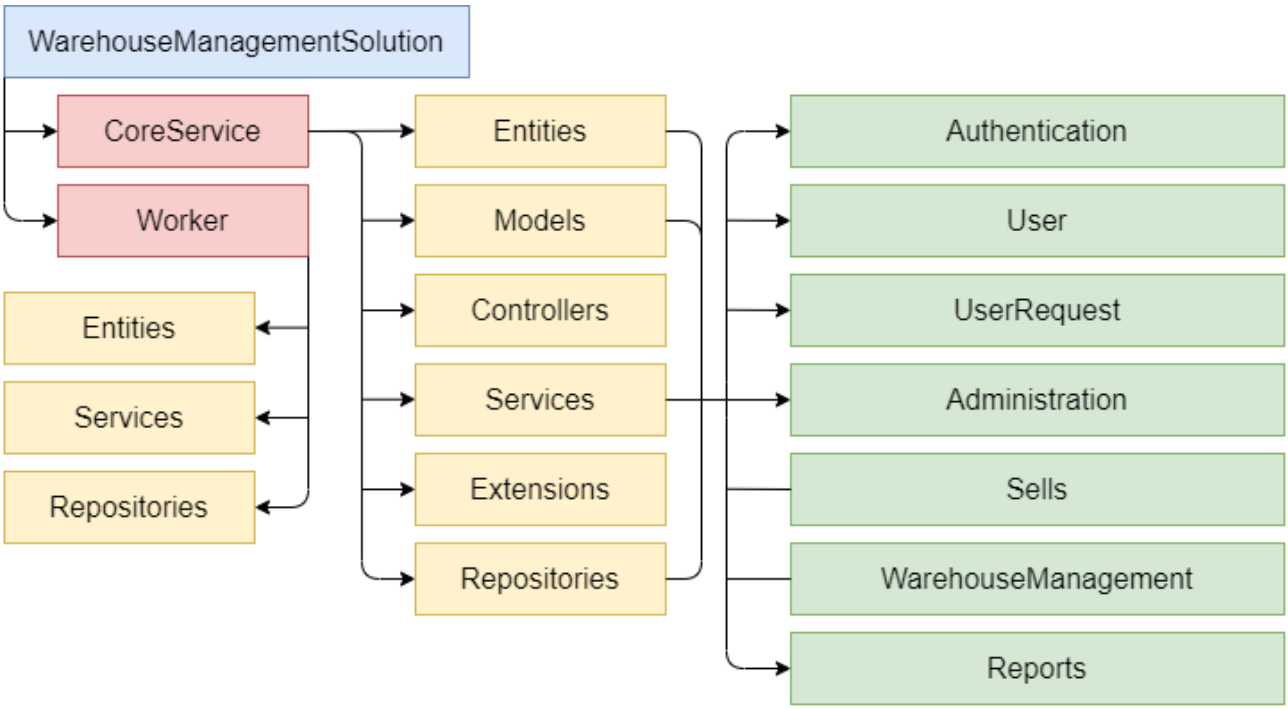

Рис. 2.6 Структура серверної частини

Як видно із рисунку 2.6, проект системи управління запасами WarehouseManagementSolution містить два сервіси: основний сервіс CoreService та сервіс опитування бази даних Worker.

CoreService слугує центральною базою для обробки і обчислення даних, яка включає в себе всі класи управління, з якими клієнт може взаємодіяти через клієнтську частину застосунку.

Worker використовується для двох цілей. Перша з них – збереження статистичної інформації. Запускається цей процес один раз в день, згідно конфігурації, збираючи необхідну статистичну інформацію із всіх необхідних

таблиць та додаючи рядок у таблицю Statistics бази даних PostgreSQL. Друге завдання сервісу – перевіряти рівень запасів на відділеннях. При досягненні точки замовлення сервіс опитування звертається до основного сервісу із проханням відправити заявку на замовлення нової партії запасів.

Призначення директорій обох сервісів проекту на сервері детально описано у таблиці 2.4, що є поясненням до рисунку 2.6.

Таблиця 2.4. Опис структури серверної частини

Директорія	Опис вмісту
Корінь проекту	Слугує збірником сервісів для обслуговування системи управління запасами ліків
Extensions	Містить допоміжні статичні методи обробки даних
Controllers	Містить класи управління, що слугують точкою входу для клієнтських запитів а також запитів сторонніх сервісів, таких як сервіс опитування у проекті управління запасами, яка після обробки повертає дані у зрозумілому для клієнту чи стороннього сервісу вигляді
Entities, Models	Зберігають в собі моделі (об’єкти) даних, необхідних для побудови бази даних, отримання відповідей з неї та формування відповідей для клієнтської частини
Repositories	Містить репозиторії взаємодії із базами даних

Продовження таблиці 2.4

Директорія	Опис вмісту
Services	Містить сервіси та менеджери обробки даних

Схема структури клієнтської знаходиться на рисунку 2.6 частини має наступний вигляд:

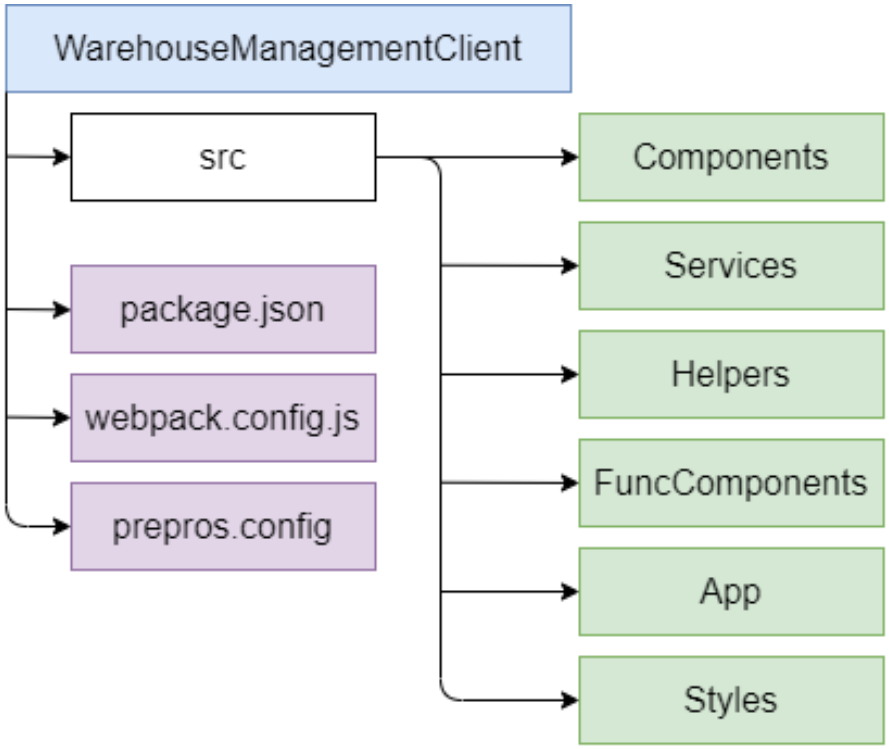

Рис. 2.7 Схема структури клієнтської частини

Структура клієнтської частини сформована згідно до принципу зрозумілого розмежування функціоналу. У корні проекту лежить директорія src, що містить весь функціонал клієнтського застосунку, а також файли налаштування:

- Файл package.json містить запис про всі бібліотеки та їх версії, завантажені за допомогою утиліти npm node.js.
- Webpack.config.js містить налаштування для роботи постачальнику модулів JavaScript, трансформації елементів HTML, CSS.
- Prepros.config містить налаштування для роботи препроцесора SCSS.

На основі внутрішніх директорій папки src будується основа клієнтської частини веб-застосунку. Її структурні директорії мають наступний опис:

- Components – збірники компонентів (по суті, веб-сторінки);
- Services – методи збору даних для відправки на сервер та обробку отриманих даних;
- Helpers – допоміжні методи;
- FuncComponents – компоненти багаторазового використання;
- App – вхідна точка для старту системи з React;
- Styles – оформлення веб-застосунку.

ВИСНОВКИ

На основі теоретичних знань із першого розділу, у другому розділі були сформовані детальні технічні вимоги, побудований веб-застосунок та показана його кінцева структура.

Застосунок, побудований з нуля, базується на клієнт-серверній архітектурі. Містить в собі два шари роботи системи – клієнт, що показує користувачу дані та дає змогу управляти ними та сервер, що обробляє дані від клієнта, за необхідності звертаючись до двох, розмежованих по функціоналу на основі теоретичного аналізу, баз даних.

Створена система запасами має різні рівні доступу, розмежовані ролями та в цілому дозволяє створювати сутності запасів, управляти ними, створювати заявки на поповнення запасів, управляти створенням організацій, відділень та користувачів, управляти довідником медичних засобів, продавати їх користувачу, переглядати статистику в рамках організації та дає змогу побачити медичні запаси на відділеннях потенційним клієнтам аптечних закладів.

Система вміє автоматично формувати статистичні звіти та створювати замовлення при досягненні критичної точки рівню запасів, що задає користувач.

РОЗДІЛ 3. ОПИС РОБОТИ ПРОГРАМИ

3.1 Інструкція користувача для клієнтів аптечних відділень

Доступ до веб-застосунку не має обмежень. Це означає, що будь-який користувач мережі Інтернет може відвідати даний сервіс. Це дозволяє просувати платформу як веб-сайт для пошуку ліків.

Користувач може знайти необхідне йому відділення в системі за допомогою спеціальних фільтрів адреси. Фільтр дозволяє знайти відділення за областю, районом, населеним пунктом, адресом та номером будинку.

Якщо користувачу необхідно уточнити наявність конкретного лікарського засобу, можна додатково ввести його назву у пошуковий рядок головної сторінки або перевірити його наявність на сторінках знайдених відділень. Приклад результатів пошуку відділення за фільтром показаний на рисунку 3.1.

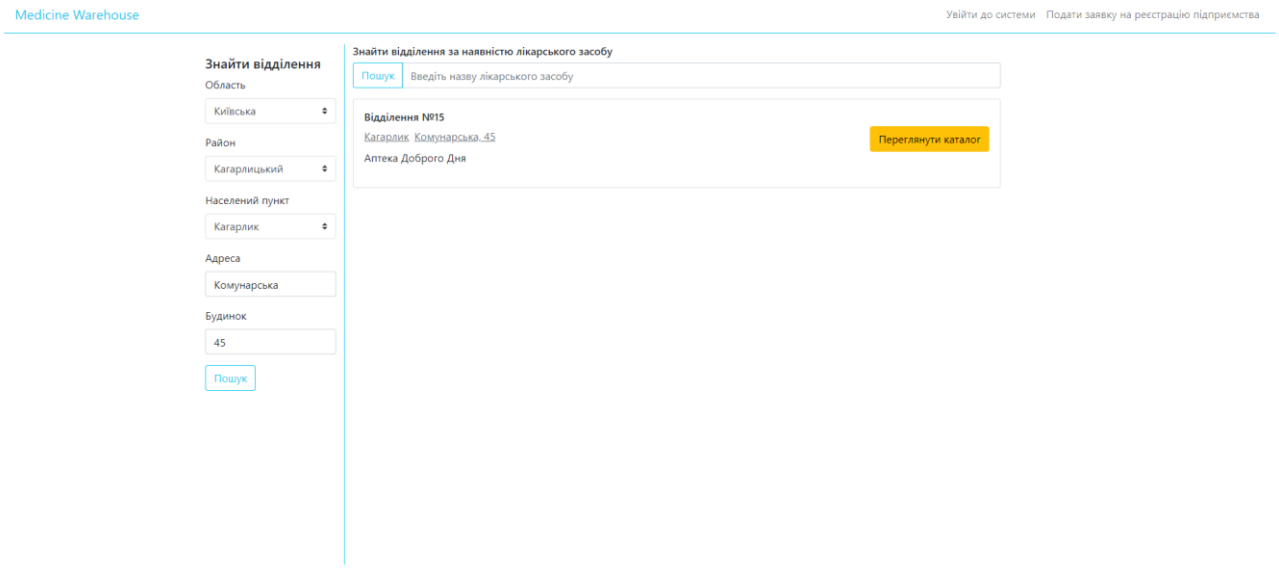

Рисунок 3.1 – Головна сторінка веб-застосунку

Якщо користувач натискає кнопку «Переглянути каталог» - він потрапляє на унікальну сторінку вибраного відділення. На ній користувач може побачити список всіх препаратів, наявних у запасах закладу. Якщо список здається занадто великим – користувач може скористатися пошуковим полем для знаходження необхідного препарату.

Інформація про препарати містить у собі його назву українською та англійською, ціни на препарати у відділенні а також його опис. Каталог відділення зображений на рисунку 3.2.

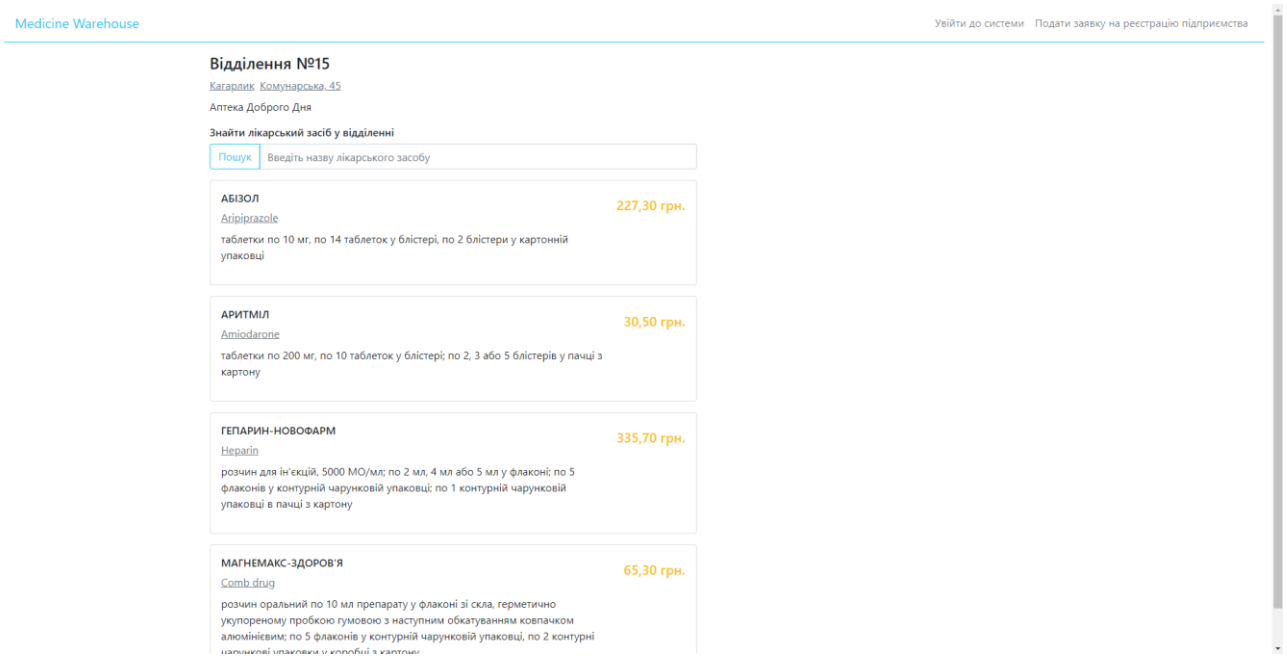

Рисунок 3.2 – Каталог препаратів відділення

Таким чином, користувач може не тільки знайти ліки у найбільш зручних для себе аптечних відділеннях, а і знайти відділення із найзручнішою ціною.

3.2 Інструкція для прийняття замовлення запасів

Для управління запасами у системі повинні бути виконані спеціальні передумови. Розглянемо їх перед поданням інструкції:

1. Якщо препарат відсутній у Державному реєстрі лікарських засобів – адміністратор системи повинен додати у довідник медичних препаратів відповідний тип запасів;
2. Реєстрація організації. Керівник повинен подати заявку на реєстрацію підприємства. Після чого адміністратор за умови успішної перевірки юридичної особи підтвердить таку реєстрацію і дозволить вхід;
3. Створення відділення. Після входу в особистий кабінет, керівник повинен створити заявку на додавання відділення аптечного закладу, яку адміністратор розгляне аналогічно заявці на створення організації.

4. Створення користувача із роллю управляючого організацією та прив’язка його до відділення. Цим займається користувач із роллю директора закладу.

Після виконання усіх передумов працівник аптечного відділення із створеним аккаунтом повинен увійти до системи за допомогою своїх персональних логіну та паролю.

За умови виконання успішного входу до системи, управляючий запасами бачить таблицю запасів. При першому вході першого управляючого запасами таблиця буде порожня.

Для її наповнення необхідно над таблицею натиснути кнопку «Створити сутність запасів». Відкриється меню додавання нової сутності запасу до відділення. Користувач повинен заповнити всі поля та вибрати, за якою моделлю повинна працювати система управління замовленнями для даного медичного препарату. Інтерфейс додавання сутності запасу показаний на рисунку 3.3.

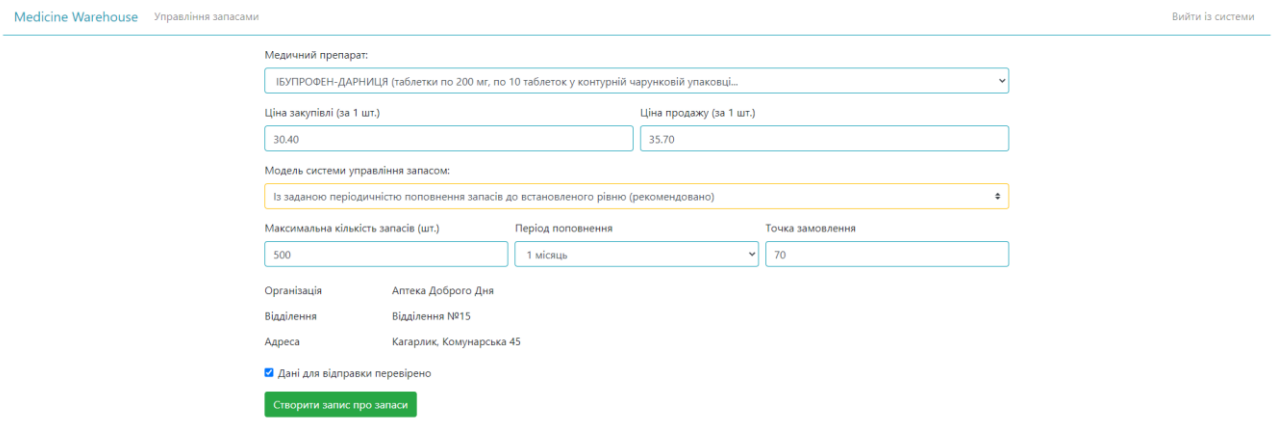

Рисунок 3.3 – Інтерфейс додавання сутності запасів до відділення

Створена сутність запасів потрапляє на сторінку управління запасами. Такій сутності виставляється статус «Потребує замовлення». Така ситуація змодельована на рисунку 3.4.

Рисунок 3.4 – Сторінка управління запасами

Перша партія товару повинна бути обов’язково замовлена вручну. Це необхідно того, щоб наступне автоматичне замовлення при досягненні критичної точки могло зчитати назву організації-постачальника та створити на нього наступне замовлення.

Для того, щоб замовити партію товару, необхідно перейти на сторінку управління замовленнями, натиснувши на кнопку «Таблиця управління замовленнями». На цій сторінці необхідно натиснути на кнопку «Додати замовлення», вибрати із списку необхідну категорію запасу, ввести назву постачальника та ввести кількість товару, що необхідно замовити. При цьому слід врахувати, що система автоматично обчислить кількість товару для замовлення на основі даних, введених у сутність запасу. Після заповнення даних про замовлення необхідно натиснути кнопку «Підтвердити відправку замовлення».

Далі замовлення з’явиться на у таблиці сторінки управління замовленнями, як на рисунку 3.5. Нові замовлення завжди виділяються зеленим кольором. Для підтвердження отримання в рядку із замовленим товаром у колонці «Статус та управління» необхідно натиснути кнопку «Підтвердити отримання». Натискання на цю кнопку приведе користувача до сторінки вводу даних про отримане замовлення. Там необхідно ввести дату отримання замовлення, на основі якого

сформується дата для проведення наступного замовлення, відповідно до зазначеного в налаштуваннях запасу періоду поповнення.

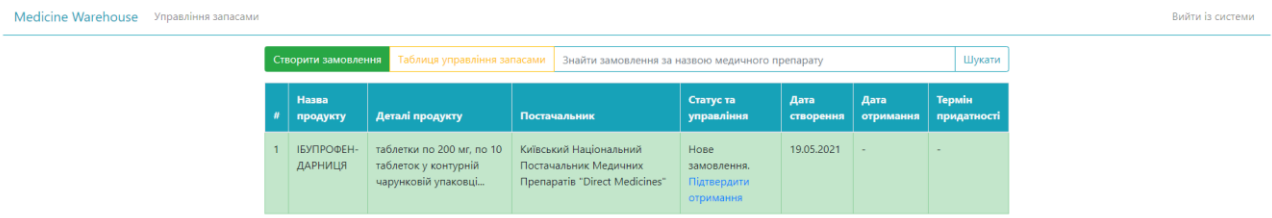

Рисунок 3.5 – Сторінка управління замовленнями партій товару

Після підтвердження статус замовлення зміниться на «Отримано», а рядок запасу на сторінці оновиться відповідно до отриманих даних і не буде виділятися червоним кольором. Оновлена сторінка управління запасами показана на рисунку 3.6. Якщо все проведено правильно і рівень запасу не близький до точки замовлення, статус зміниться на «ОК».

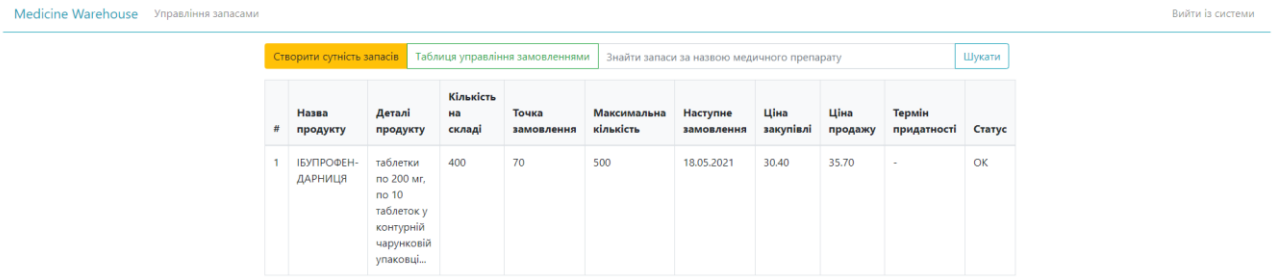

Рисунок 3.6 – Успішно додана сутність запасу

ВИСНОВКИ

Даний розділ демонструє частини роботи веб-застосунку, такі як пошук ліків та управління запасами.

Була надана інструкція та опис функціоналу для користувача, який має намір скористатися послугами аптечних закладів та бажає знайти підходящі ліки в запасі найбільш зручного для користувача відділення. Показано, як користуватися пошуковою системою веб-застосунку за проводити навігацію по сторінках пошуку ліків.

Для підприємств аптечних закладів був детально описаний основний процес створення та отримання замовлення партії ліків з нуля, починаючи із переліку передумов, таких як реєстрація організації, створення відділень і профілів користувачів, а також додавання медичного препарату до довіднику. Показано, як створити сутність запасу, створити замовлення на його поповнення і прийняти його.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Г. І. Капінос, І. В. Бабій. Операційний менеджмент : навч. посіб. Київ : Центр учбової літератури, 2013. 352 с.
2. Н. М. Тюріна, І. В. Гой, І. В. Бабій. Логістика : навч. посіб. Київ : Центр учбової літератури, 2015. 392 с.
3. В Україні впровадять електронну систему управління запасами ліків eStock. Ukrinform : веб-сайт. URL : https://www.ukrinform.ua/rubric-society/2632308-v-ukraini-vprovadat-elektronnu-sistemu-upravlinna-zapasami-likiv-estock.html (дата звернення: 02.05.2021)
4. Нова Велика депресія: наслідки коронавірусу для світової та української економіки. Hromadske : веб-сайт. URL : https://hromadske.ua/posts/nova-velika-depresiya-naslidki-koronavirusu-dlya-svitovoyi-ta-ukrayinskoyi-ekonomiki (дата звернення: 02.05.2021)
5. Що таке WMS. Habr : веб-сайт. URL : https://habr.com/ru/post/252889/ (дата звернення: 02.05.2021)
6. What is a Warehouse Management System (WMS). Erpblog : веб-сайт. URL : https://erpblog.iqms.com/what-is-warehouse-management-system (дата звернення: 02.05.2021)
7. Основні системи управління запасами. Fnow : веб-сайт. URL : https://fnow.ru/articles/osnovnye-sistemy-upravleniya-zapasami (дата звернення: 02.05.2021)
8. В.М. Гончаров, Н.В. Касьянова, Н.В. Вецепура, Д.В. Солоха. Внутрішній економічний механізм підприємства: навч. посіб. Донецьк : СПД Купріянов В.С., 2007. 284 с.
9. Пономарьова Ю.В. Логістика : навч. посіб. Харків : Центр навчальної літератури, 2005. 328 с.
10. Єжова Л. Ф. Інформаційний маркетинг : навч. посіб. Київ : КНЕУ, 2002. 560 с.

11. Як захистити систему від хакерських атак. Uteka.ua : веб-сайт. URL : https://uteka.ua/ua/publication/news-14-delovye-novosti-36-kak-zashhitit-sistemu-ot-xakerskix-atak (дата звернення: 04.05.2021)
12. Найпопулярніші операційні системи світу 2020. Uaspectr : веб-сайт. URL : https://uaspectr.com/2020/05/06/najpopulyarnishi-operatsijni-systemy-svitu-2020/ (дата звернення: 04.05.2021)
13. Синхронність і асинхронність процесів. Habr : веб-сайт. URL : https://habr.com/ru/post/453192 (дата звернення: 04.05.2021)
14. Desktop vs. Web Applications: A Deeper Look and Comparison. Seguetech : веб-сайт. URL : https://www.seguetech.com/desktop-vs-web-applications (дата звернення: 04.05.2021)
15. Top 10 Warehouse Management Systems. thirdstage-consulting : веб-сайт. URL : https://www.thirdstage-consulting.com/top-10-warehouse-management-systems (дата звернення: 04.05.2021)
16. Best Inventory Management Software (May 2021). ecommerce-platforms : веб-сайт. URL : https://ecommerce-platforms.com/articles/best-inventory-management-software (дата звернення: 04.05.2021)
17. Cloud-based warehouse management application. Zoho : веб-сайт. URL : https://www.zoho.com/inventory/warehouse-management/ (дата звернення: 04.05.2021)
18. Zoho Inventory Software Review. Softwareadvice : веб-сайт. URL : https://www.softwareadvice.com/inventory-management/zoho-inventory-profile/ (дата звернення: 04.05.2021)
19. Система управління запасами Oracle. Oracle : веб-сайт. URL : https://www.oracle.com/ru/scm/logistics/warehouse-management/ (дата звернення: 04.05.2021)
20. Oracle Warehouse Management Cloud Reviews & Product Details. G2 : веб-сайт. URL : https://www.g2.com/products/oracle-oracle-warehouse-management-cloud/reviews (дата звернення: 04.05.2021)

21. Oracle Warehouse Management User's Guide. Docs.Oracle : веб-сайт. URL : https://docs.oracle.com/cd/E18727_01/doc.121/e13433/index.html (дата звернення: 04.05.2021)
22. Управління замовленнями. Galactica.ua : веб-сайт. URL : http://galaktika.ua/erp/upravlenie-zakazami.html (дата звернення: 05.05.2021)
23. Warehouse Management System. 4PSite : веб-сайт. URL : https://www.4psite.com/wms (дата звернення: 05.05.2021)
24. 4PSite WMS reviews. Capterra : веб-сайт. URL : https://www.capterra.com/p/142449/4Psite/#reviews (дата звернення: 05.05.2021)
25. SAP Extended Warehouse Management. SAP : веб-сайт. URL : https://www.sap.com/products/extended-warehouse-management.html (дата звернення: 05.05.2021)
26. SAP – управління запасами. Coderlessons : веб-сайт. URL : https://coderlessons.com/tutorials/sap/izuchite-sap-mm/sap-mm-upravlenie-zapasami (дата звернення: 05.05.2021)
27. What is SAP Warehouse Management. Selecthub : веб-сайт. URL : https://www.selecthub.com/warehouse-management-software/sap-warehouse-management (дата звернення: 05.05.2021)
28. Client-server architecture. Britannica : веб-сайт. URL : https://www.britannica.com/technology/client-server-architecture (дата звернення: 06.05.2021)
29. Client-Server Architecture. Teachcomputerscience : веб-сайт. URL : https://teachcomputerscience.com/client-server-architecture/ (дата звернення: 06.05.2021)
30. Get started with Bootstrap. Getbootstrap : веб-сайт. URL : https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 06.05.2021)

31. Tutorial: Intro to React. Reactjs : веб-сайт. URL : https://reactjs.org/tutorial/tutorial.html (дата звернення: 08.05.2021)
32. HTML Basics — The 10 Concepts. Medium : веб-сайт. URL : https://medium.com/@readizo.com/html-basics-the-10-important-concepts-afeedcbe8e7d (дата звернення: 08.05.2021)
33. Top JavaScript frameworks for cross-platform development. Who is the leader. ClockwiseSoftware : веб-сайт. URL : https://clockwise.software/blog/top-javascript-frameworks-for-cross-platform-development (дата звернення: 08.05.2021)
34. Redux. A Predictable State Container for JS Apps : веб-сайт. URL : https://redux.js.org (дата звернення: 08.05.2021)
35. What's the Difference? Relational vs Non-Relational Databases. Izenda : веб-сайт. URL : https://www.izenda.com/relational-vs-non-relational-databases (дата звернення: 09.05.2021)
36. MongoDB. The application data platform. Mongodb : веб-сайт. URL : https://www.mongodb.com/1 (дата звернення: 09.05.2021)
37. PostgreSQL: The World's Most Advanced Open Source Relational Database. Postgresql : веб-сайт. URL : https://www.postgresql.org (дата звернення: 09.05.2021)
38. .NET. Free. Cross-platform. Open source. Microsoft : веб-сайт. URL : https://dotnet.microsoft.com (дата звернення: 09.05.2021)
39. C# documentation. Microsoft : веб-сайт. URL : https://docs.microsoft.com/en-us/dotnet/csharp (дата звернення: 09.05.2021)
40. Сучасний посібник по JavaScript. LearnJavascript. URL : https://learn.javascript.ru (дата звернення: 09.05.2021)

ДОДАТКИ

Додаток А. Код клієнтської сторінки створення замовлення

```
import React, { Component, useState } from "react";
import Form from "react-bootstrap/Form";
import Col from "react-bootstrap/Col";
import Button from "react-bootstrap/Button";
import Row from "react-bootstrap/Row";
import AsyncSelect from "react-select/async";
import { WarehouseService } from "../../../Services/WarehouseService";
class CreateBatch extends Component {
  constructor(props) {
    super();
    this.state = {
      stockId: null,
      providername:"", count:"",
      selectedStockValue: "",
      responseSuccess: "",
      responseError: "",
    };
    this.handleProductSelectChange = this.handleProductSelectChange.bind(this);
    this.handleChange = this.handleChange.bind(this);
    this.createProductRequest = this.createProductRequest.bind(this);
  }

  handleChange(e) {
    console.log(e);
    const { name, value } = e.target;
    this.setState({ [name]: value });
  }
  handleProductSelectChange(e) {
    console.log(e.stockId);
    console.log("aaaaa");
    this.setState({
      stockId: e.stockId,
      selectedStockValue: e,
    });
  }

  loadOptions = (inputValue) => {
    console.log(inputValue);
    if (inputValue.length > 2) {
      return WarehouseService.GetStockNames(inputValue);
    }
  };

  handleInputChange = (value) => {
    console.log(value);
    this.setState({
      selectedProductValue: value,
```

```
    stockId: value.stockId,
  });
};

handleChange(e) {
  const { name, value } = e.target;
  this.setState({ [name]: value });
}

createProductRequest(e) {
  e.preventDefault();
  const { stockId, providername, count, responseError, responseSuccess } =
    this.state;
  console.log("stockId");
  console.log(stockId);

  WarehouseService.CreateBatch(stockId, providername, count).then(
    (result) =>
      this.setState({
        responseSuccess: "Заявка на поповнення була успішно створена.",
      }),
    (error) =>
      this.setState({
        responseError: `Виникла помилка під час відправки запиту. ${error}`,
      })
  );
}

render() {
  const {
    selectedStockValue,
    providername,
    count,
    responseError,
    responseSuccess,
  } = this.state;
  return (
    <Form>
      {responseError && (
        <div class="alert alert-danger" role="alert">
          {responseError}
        </div>
      )}
      {responseSuccess && (
        <div class="alert alert-success" role="alert">
          {responseSuccess}
        </div>
      )}
      <Form.Group controlId="formGridAddress1">
        <Form.Label>Виберіть сутність запасу:</Form.Label>
        <AsyncSelect
```

```
    defaultOptions
    placeholder="Знайдіть сутність запасу по назві лікарського засобу"
    value={selectedStockValue}
    getOptionLabel={(e) => e.name}
    getOptionValue={(e) => e.stockId}
    loadOptions={this.loadOptions}
    onChange={this.handleProductSelectChange}
  />
</Form.Group>
<Form.Row>
  <Form.Group as={Col}>
    <Form.Label>Введіть назву фірми-постачальника</Form.Label>
    <Form.Control
      type="text"
      name="providername"
      value={providername}
      placeholder="Назва фірми-постачальника"
      className="border border-info"
      onChange={this.handleChange}
    />
  </Form.Group>

  <Form.Group as={Col} controlId="formGridPassword">
    <Form.Label>
      Введіть точну кількість одиниць медичних засобів для замовлення
    </Form.Label>
    <Form.Control
      type="number"
      name="count"
      value={count}
      placeholder="Кількість препаратів"
      className="border border-info"
      onChange={this.handleChange}
    />
  </Form.Group>
</Form.Row>

<Form.Group as={Row} controlId="formPlaintextEmail" className="mb-0">
  <Form.Label column sm="2">
    Організація
  </Form.Label>
  <Col sm="10">
    <Form.Control
      plaintext
      readOnly
      defaultValue="Аптека Доброго Дня"
    />
  </Col>
</Form.Group>
<Form.Group as={Row} controlId="formPlaintextEmail" className="mb-0">
  <Form.Label column sm="2">
```

```
            Відділення
          </Form.Label>
          <Col sm="10">
            <Form.Control plaintext readOnly defaultValue="Відділення №15" />
          </Col>
        </Form.Group>

        <Form.Group as={Row} controlId="formPlaintextEmail">
          <Form.Label column sm="2">
            Адреса
          </Form.Label>
          <Col sm="10">
            <Form.Control
              plaintext
              readOnly
              defaultValue="Кагарлик, Комунарська 45"
            />
          </Col>
        </Form.Group>

        <Form.Group id="formGridCheckbox">
          <Form.Check type="checkbox" label="Дані для відправки перевірено" />
        </Form.Group>

        <Button
          variant="success"
          type="submit"
          onClick={this.createProductRequest}
        >
          Створити запис про запаси
        </Button>
      </Form>
    );
  }
}

export default CreateBatch;
```

Додаток Б. Механізм пошуку назви товару використовуючи TsVector у PostgreSQL через EntityFramework

```
public static class TsVectorHelper
{
    public static string ToTsQueryString(string query)
        => $"{string.Join(":* & ", Rgx.Replace(query,"\\$1").Split(' ',
StringSplitOptions.RemoveEmptyEntries))}:*";

    private static readonly Regex Rgx = new
Regex("(['^$.|?*+()/\\\\#!\"\\\\{\\}\\[\\]\\:\\<\\>\\&])",
RegexOptions.Compiled);
}
...
```

```
public static string GetShortString(string text, int length = 20)
{
    return !string.IsNullOrEmpty(text) && text.Length > length
        ? text.Substring(0, length) + "..."
        : text ?? "";
}
```

...

```
var result = await _context.Stocks.AsQueryable()
    .Include(x => x.Product)
    .Where(x => x.DepartmentId == departmentId &&

x.Product.SearchVector.Matches(EF.Functions.ToTsQuery(TsVectorHelper.ToTsQuerySt
ring(search))))
    .Take(20)
    .OrderBy(o => o.Product.NameUkr)
    .ThenByDescending(t => t.TableKey)
    .ToListAsync();
var response = result.Select(stockEntity => new StockNamesResponseModel()
{
    StockId = stockEntity.Id,
    Name = $"#{stockEntity.TableKey}, {stockEntity.Product.NameUkr},
{TextHelper.GetShortString(stockEntity.Product.Description, 80)}"
}).ToList();
```

...