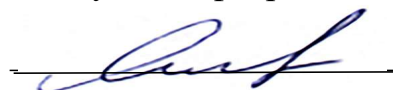


Національний університет кораблебудування імені адмірала Макарова
Навчально науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

«Допущений до захисту»

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

«__»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: «Розробка інформаційної системи мережі аптек "Аптека плюс"»

Виконав: студент групи



Борко Ю.А.

(підпис)

Керівник роботи:

Старший викладач _____

(посада, науковий ступінь вчене звання)



Беркунський Є.Ю.

(підпис)

Миколаїв – 2026 р.

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої
програми



к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття ступеня вищої освіти "бакалавр"

Студенту Борко Юлії Андріївни

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інформаційної системи мережі аптек "Аптека плюс"

Керівник роботи: Беркунський Євген Юрійович

Затверджені наказом ректора № 270-уч від "31" березня 2026 року.

2. Термін подання роботи: 22 червня 2026 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем

4. Перелік питань, що належать до розробки (найменування розділів): аналіз предметної області і постановка задачі, розробка проекту інформаційної системи, реалізація проекту інформаційної системи, розділ з охорони праці, висновки.

5. Перелік презентаційних матеріалів _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Беркунський Є.Ю.	02.02.2026	23.02.2026
2	Беркунський Є.Ю.	23.02.2026	20.03.2026
3	Беркунський Є.Ю.	20.03.2026	24.04.2026
4	Беркунський Є.Ю.	24.04.2026	01.06.2026

7. Дата видачі завдання 02 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	02.02.2026	
2	Постановка задачі	16.02.2026	
3	Розробка концепції	23.02.2026	
4	Розробка проекту ІС	23.03.2026	
5	Реалізація проекту	27.04.2026	
6	Розробка питань з охорони праці	01.06.2026	
7	Розробка графічних матеріалів	05.06.2026	
8	Подання роботи на перевірку на плагіат	08.06.2026	
9	Подання роботи рецензенту	15.06.2026	
10	Подання роботи на захист	22.06.2026	

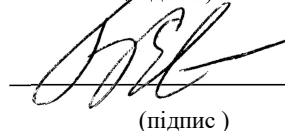
Студент


(підпис)

Борко Юлія Андріївна

(ПІБ)

Керівник роботи


(підпис)

Беркунський Євген Юрійович

(ПІБ)

АНОТАЦІЯ

У роботі досліджено теоретичні засади створення інформаційних систем у фармацевтичній сфері та проаналізовано сучасний стан аптечного бізнесу в Україні. Визначено проблеми аптеки «Аптека плюс» та сформульовано вимоги до нової системи, що забезпечує структуроване зберігання даних, автоматизацію обліку та захист інформації. Розроблено концепцію багаторівневої архітектури, спрямовану на оптимізацію бізнес-процесів і підвищення якості обслуговування клієнтів.

Робота викладена на 89 сторінках, містить 4 розділи, 15 рисунків, 4 додатків, і список використаних джерел з 13 найменувань.

Ключові слова: аптека, «Аптека плюс», медикаменти, інформаційна система, база даних, Java, Spring, SQL.

ABSTRACT

This thesis examines the theoretical foundations of information systems in the pharmaceutical sector and analyzes the current state of the pharmacy business in Ukraine. It identifies key challenges in the “Apteka Plus” network and proposes a multi-level information system architecture to ensure structured data storage, automated accounting, and secure customer interaction. The solution is aimed at optimizing business processes, improving service quality, and integrating modern digital technologies into healthcare.

The work is presented on 89 pages, contains 4 sections, 15 pictures, 4 applications and 13 references to the sources.

Keywords: pharmacy, "Pharmacy Plus", medicines, information system, database, Java, Spring, SQL.

АНОТАЦІЯ.....	4
ВСТУП.....	6
РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ФОРМУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	7
1.1 Опис об'єкта дослідження	7
1.2 Аналіз предметної області	12
1.3 Формування вимог до інформаційної системи	13
1.4 Розробка концепції інформаційної системи.....	15
РОЗДІЛ 2. РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ АПТЕКИ	21
2.1 Загальносистемні рішення	21
2.2 Рішення по інформаційному забезпеченню	24
2.3 Рішення по математичному забезпеченню	34
2.4 Рішення по програмному забезпеченню	36
2.5 Рішення по технічному забезпеченню	43
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЄКТА ІНФОРМАЦІЙНОЇ СИСТЕМИ	47
РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ	54
ВИСНОВОК	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58
ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ	59
ДОДАТОК Б – ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ	62
ДОДАТОК В – ПРИКЛАДИ ЕКРАННИХ ФОРМ	68
ДОДАТОК Г – КОД ПРОГРАМИ ТА КОД СТВОРЕННЯ ТАБЛИЦЬ БАЗИ ДАНИХ	71

У даній кваліфікаційній роботі демонструється розробка інформаційної системи веб-сайту аптеки, яка спростить роботу адміністраторам в організації товарів на віртуальних полицях свого закладу, дозволить швидко додавати нові ліки та редагувати старі, а також полегшить взаємодію з клієнтами та дасть можливість відкрити нові способи комунікації з ними. Покупці в свою чергу зможуть дивитися ціни та характеристики товарів в реальному часі та замовляти їх не виходячи з дому. Зручний інтерфейс разом із доступним функціоналом зроблять рутинну закупку ліків більш приємним процесом для усіх користувачів.

В сучасному світі неможливо уявити сферу, яка частково або повністю не перейшла б в онлайн простір. Незважаючи на складність такого переходу для більшості з нас, дані зміни кардинально змінюють підхід до продажу товарів та надання послуг в позитивну сторону. Тому у контексті глобальної цифровізації важливо забезпечити таку затребувану сферу життя як здоров'я - зручними інструментами для користування.

Метою цієї кваліфікаційної роботи є розширення можливостей людей при закупці ліків в аптеках. Основною ціллю є розробка інформаційної системи, яка дозволить користувачам завчасно вибрати необхідні товари на сайті в зручній для себе обстановці та забронювати їх. Це дасть можливість покупцям краще планувати свій графік та оцінити об'єм товару, який можна купити без необхідності виходити з дому.

У розділі 1 зроблений опис об'єкту дослідження та аналіз предметної області. Сформовані вимоги до інформаційної системи та на їх основі розроблена концепція, яка визначає її функціонал.

1.1 Опис об'єкта дослідження

Середня тривалість життя – базовий показник здоров'я населення. В Україні в 1990 році вона склала майже 70,5 років. Для чоловіків – близько 65,5; для жінок – більше 74,8 [1]. Станом на 2025-2026 роки середня тривалість життя в Україні суттєво знизилась через війну, але водночас є вищою ніж 30 років тому, складаючи близько 67 років для чоловіків та 77 років для жінок [2]. За останні кілька десятиліть сфера здоров'я зазнала величезних змін. Завдяки технічному прогресу та розвитку медицини людство зараз має можливість лікувати багато серйозних захворювань, більшість з яких напряду впливають на тривалість життя і її якість в будь-якому віці.

Ліки, залежно від рівня розвитку суспільства в різні часи мали різний вигляд і якість, але завжди використовувалися з метою лікування та запобігання хворобам. Вони використовувалися з сивої давнини і лише на підставі археології, етнографії та інших даних можна робити висновок про те, що відбувалось у ті далекі часи. Використовуючи досвід попередніх поколінь, людство емпіричним шляхом відкривало лікувальні властивості багатьох рослин. Майже до XIX ст. використовувалася сировина рослинного, тваринного або мінерального походження, що піддавалася відносно простому обробленню. Згодом з рослин та іншої сировини почали виділяти або концентрувати діючі речовини, встановлювати їх структуру, здійснювати синтез біологічно активних речовин (БАР). Значно пізніше, коли була встановлена певна залежність між хімічною будовою БАР та їх фармакологічною дією, були отримані аналоги природних сполук.

Нарешті, з вивченням механізму дії багатьох фармакотерапевтичних груп, був започаткований цілеспрямований пошук нових ефективних лікарських речовин. Почали розвиватися нові, більш сучасні методи синтезу лікарських речовин (комбінаторний, комп'ютерний тощо). На різних етапах розвитку

фармацевтичної технології в медицині використовувалися ЛП у різних лікарських формах (кашки, збори, порошки, пілюлі, пластирі, настої, примочки, гранули, таблетки, мазі тощо). Нині фармацевтичні фірми пропонують більш складні лікарські системи, які, на відміну від звичайних лікарських препаратів, характеризуються контрольованим вивільненням діючої субстанції та її цілеспрямованою (адресною) доставкою до певних органів або тканин організму.

Залежно від бажаного фармакологічного ефекту ліки вводять в організм різними шляхами. Для кожного шляху введення лікарських препаратів розробляється певна лікарська форма, метод застосування, а інколи і спеціальний інструментарій. І.П. Павлов ще в 1895 р. відзначав універсальність лікарської терапії, яка використовується навіть при різних медичних процедурах, що не втратило свого значення донині. Ліки стали могутньою зброєю в боротьбі з найтяжчими захворюваннями; вони прискорюють видужання, поліпшують самопочуття, підвищують працездатність, мобілізують захисні сили організму.

У будь-які часи люди продовжують хворіти й в зв'язку з цим змушені купувати лікарські препарати, іноді навіть на шкоду власному шлунку. Незважаючи на величезну кількість працівників аптек і аптечних пунктів, аптечний бізнес продовжує вважатися досить прибутковим і залучає нових підприємців, які відкривають додаткові точки з продажу ліків. За різними оцінками рентабельність аптек становить від 15% до 40% і залежить від місця їх розташування, професіоналізму персоналу та цінової політики закладу [4]. Попит на медичні товари з роками тільки зростає. На заміну старих ліків на полиці аптек виставляють нові та більш ефективні товари. Водночас дивлячись на економічну ситуацію та появлення більш агресивних захворювань, люди вимушені втрачати велику частку своїх грошей на аптечні товари. Основними категоріями витрат є продукти харчування (85%), комунальні послуги (72%), та ліки (60%). Очікувано, що серед респондентів віком 60+ років більшість (78%) назвала ліки однією з основних статей витрат. Найбільша частка людей, для яких ліки є основною категорією витрат, проживає в селах — 71%, тоді як в містах із населенням до 50

тис. осіб цей показник становить 50%. Це, вочевидь, пов'язано з тим, що в селах проживає значно більше людей старшого віку [5].

Розвиток фармацевтичної галузі та зростання попиту на лікарські засоби зумовлюють необхідність пошуку нових підходів до організації аптечного бізнесу. Традиційні методи управління та обслуговування клієнтів уже не повною мірою відповідають сучасним викликам, адже вони обмежують швидкість доступу до інформації та зручність взаємодії з пацієнтами. У цьому контексті дедалі більшого значення набувають інноваційні технології, що дозволяють оптимізувати процеси та забезпечити якісно новий рівень сервісу. Саме тому важливим етапом розвитку є впровадження цифрових рішень, які змінюють підхід до охорони здоров'я та фармацевтичної діяльності.

Цифрова трансформація охорони здоров'я сьогодні є одним із ключових чинників підвищення ефективності медичних систем та розвитку біотехнологічного сектору. В Україні тенденції цифрової трансформації охорони здоров'я стають особливо актуальними у контексті війни та відновлення медичної інфраструктури. Використання цифрових технологій дозволяє підвищити ефективність медичного обслуговування, забезпечити стійкість системи охорони здоров'я та інтегрувати українські біотехнологічні компанії у глобальні науково-технологічні процеси.

Погоджуюсь з думкою Деміховим О. О., що «цифрова медицина покликана забезпечити взаємодію між пацієнтами, медпрацівниками та лікарнями за допомогою інформаційних технологій. Перехід медичного документообігу України у цифровий формат є головною метою цифрової медицини». Автоматизація процесів у сфері охорони здоров'я спрямована на підвищення ефективності обслуговування пацієнтів, зменшення навантаження на лікарів і медичних працівників та оптимізацію управлінських рішень. Однак в Україні досі зберігається низка проблем, пов'язаних із використанням Інтернету та цифрових технологій, зокрема у сільській місцевості, де спостерігається недостатній рівень цифрової інфраструктури та обмежений

доступ до якісного зв'язку. М.В.Макаренко вважає, що «Важливим кроком до цифровізації медичних та супутніх послуг є створення повнофункціональної цифрової медичної платформи. Для сільській місцевості особливо важливим є впровадження телекомунікаційних систем для надання дистанційних медичних послуг, а для цього потрібен широкосмуговий доступ до інтернету. Це дозволить забезпечити проведення онлайн діагностування, дасть можливість проводити моніторинг стану хворого тощо»[6].

Мережа аптек «Аптека плюс» позиціонується як сучасна система закладів, що об'єднує центральний офіс, регіональні представництва та численні роздрібні точки в містах і селах. Основною метою мережі є забезпечення населення якісними лікарськими засобами, медичними виробами та супутніми товарами для здоров'я за доступними цінами.

Організаційна структура аптеки «Аптека плюс» представлена на рисунку 1.1

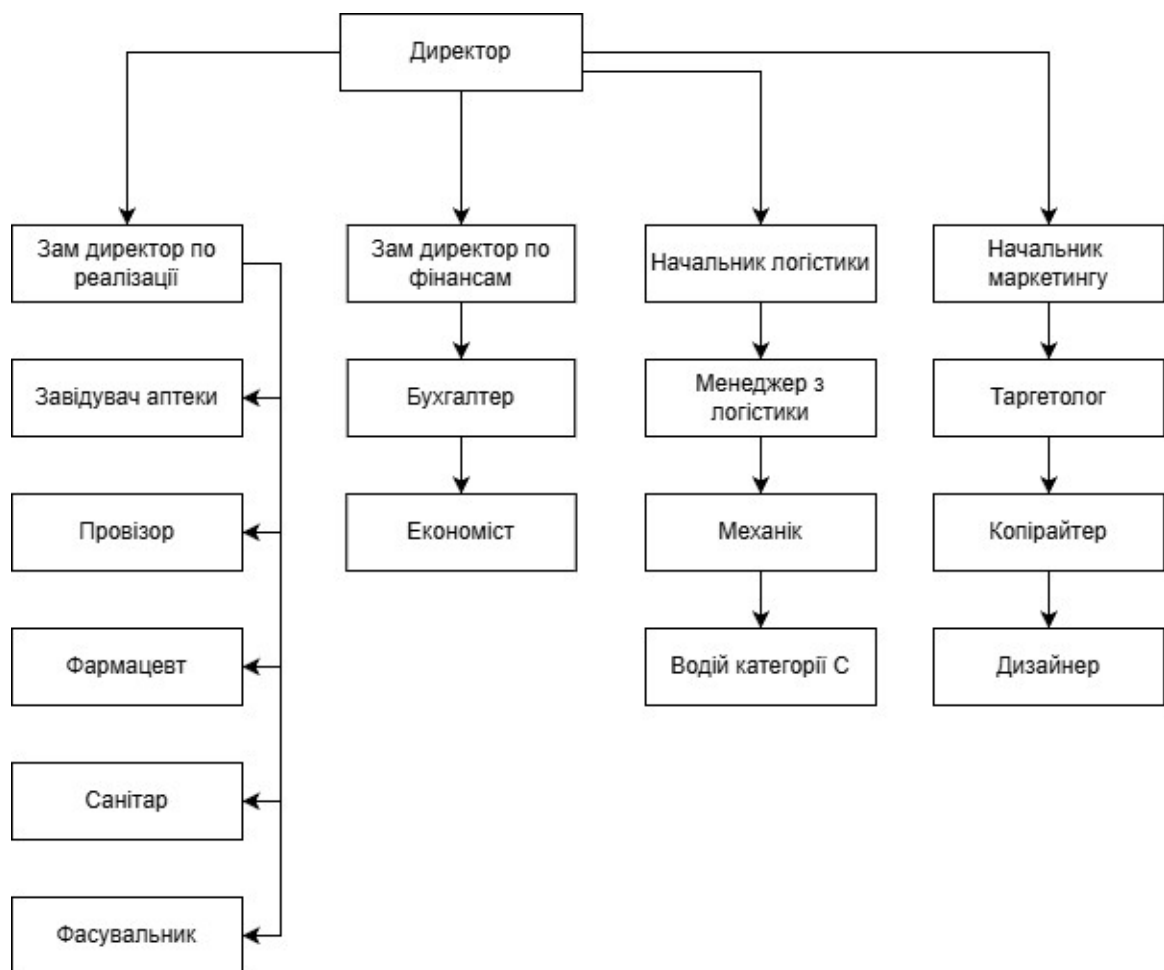


Рис. 1.1 – Організаційна структура аптеки «Аптека плюс»

Ця схема показує ієрархічну організаційну структуру підприємства, де всі підрозділи підпорядковуються директору. Вона демонструє розподіл управлінських і виконавчих функцій між чотирма основними напрямками діяльності:

- Директор — головна посадова особа, що здійснює стратегічне управління підприємством, координує роботу всіх підрозділів і приймає ключові рішення.
- Зам директор по реалізації відповідає за продаж продукції, роботу аптек і обслуговування клієнтів. Підпорядковані працівники:
 - Завідувач аптеки — організовує роботу торгової точки.
 - Провізор і фармацевт — здійснюють відпуск лікарських засобів, консультують клієнтів.
 - Санітар — підтримує санітарні норми.
 - Фасувальник — займається пакуванням продукції.
- Зам директор по фінансам контролює фінансові потоки, звітність і планування бюджету. Підпорядковані:
 - Бухгалтер — веде облік операцій.
 - Економіст — аналізує витрати, прибутковість і ефективність діяльності.
- Начальник логістики організовує доставку товарів і технічне обслуговування транспорту. Підпорядковані:
 - Менеджер з логістики — планує маршрути й контролює поставки.
 - Механік — відповідає за технічний стан транспорту.
 - Водій категорії С — здійснює перевезення продукції.
- Начальник маркетингу формує імідж підприємства та просуває продукцію. Підпорядковані:
 - Таргетолог — налаштовує рекламні кампанії.
 - Копірайтер — створює текстовий контент.

- Дизайнер — розробляє візуальні матеріали.

Таким чином, аналіз історичного розвитку та сучасного стану фармацевтичної галузі показав, що ліки залишаються ключовим елементом у забезпеченні здоров'я населення. Водночас зростаючий попит на аптечні товари та розширення мереж аптек формують нові виклики для їх організації та управління. Це створює підґрунтя для подальшого аналізу предметної області, який дозволить визначити специфіку функціонування аптечної мережі та її інформаційних потреб.

1.2 Аналіз предметної області

Автоматизована інформаційна система мережі «Аптека плюс» інтегрована лише між центральним офісом, регіональними представництвами та аптеками, що дає змогу ефективно управляти залишками, здійснювати внутрішній облік і логістику, а також контролювати процес бронювання товарів між підрозділами. Однак для покупців не передбачено електронного доступу: немає платформи для формування онлайн-замовлень чи розрахунку їхньої вартості. Високий рівень обслуговування досягається за рахунок професійної підготовки персоналу, впровадження новітніх технологій у внутрішніх процесах та гнучкої системи знижок для пенсіонерів і соціально вразливих груп населення.

За відсутності електронного доступу до системи виникають наступні проблеми:

1. Неможливість купити ліки в аптеці зі зручним для себе розташуванням. Людина прийде купити ліків, а в наявності їх може не виявитися. Окрім цього, існують цілі групи тих, хто взагалі не має можливості далеко пересуватися щоб придбати необхідні товари: інваліди, тяжко хворі тощо.
2. Можуть виникнути проблеми з оплатою, бо людина завчасно не знає ціну товарів за якими йде до аптеки.

3. Відсутність інструкції та опису товару ще до покупки для тих, хто має протипоказання чи алергію на якісь препарати. В результаті це може призвести до непорозумінь чи даремно витрачених грошей і в результаті незадоволення клієнта.

З метою підвищення ефективності роботи мережі аптек «Аптека плюс» доцільно розробити інформаційну систему мережі аптек.

Отже, дослідження предметної області показало, що існуюча автоматизована система аптечної мережі забезпечує внутрішні бізнес-процеси, проте не враховує потреби кінцевих користувачів у зручному онлайн-сервісі. Виявлені проблеми — відсутність доступу до інформації про товари, неможливість формування замовлень та обмеженість комунікації з клієнтами — визначають необхідність формування вимог до нової інформаційної системи, яка усуне ці недоліки.

1.3 Формування вимог до інформаційної системи

Формування вимог до інформаційної системи є одним із ключових етапів її проектування, адже саме на цьому рівні визначається функціональність, структура та технічні характеристики майбутнього програмного продукту. Вимоги виступають своєрідним «мостом» між аналізом предметної області та розробкою концепції системи, забезпечуючи узгодженість між потребами користувачів та можливостями сучасних технологій.

У контексті аптечної мережі вимоги до інформаційної системи мають особливе значення, оскільки вони визначають не лише базові функції — облік товарів, формування замовлень, управління бонусами, — але й гарантують відповідність нормативним стандартам, захист персональних даних та зручність користування для різних категорій клієнтів. Важливо врахувати, що сучасні інформаційні системи повинні бути масштабованими, інтегрованими з корпоративними ресурсами та здатними до адаптації під нові бізнес-процеси.

Таким чином, у цьому підрозділі буде сформульовано комплекс вимог, які охоплюють функціональні, інформаційні, технічні та програмні аспекти системи. Їхнє чітке визначення дозволить перейти до розробки концепції, що інтегрує ці вимоги у цілісну архітектуру майбутнього веб-застосунку аптечної мережі.

Щоб вирішити проблеми, з якими стикаються буденні клієнти аптеки, потрібно розробити інформаційну систему з наступними вимогами:

1. Приймання та запис даних у систему відбувається тільки після перевірки
2. Зберігання даних має бути структурованим
3. Шаблонне виведення даних для звітів
4. Чітке сповіщення користувача про статус замовлення
5. Автоматичне сповіщення адміністраторів про надходження нових даних для оброки
6. Зрозумілий інтерфейс для всіх користувачів системи

Для виконання функціональних вимог необхідно застосувати наступні програмні засоби:

1. Операційна система
2. Конструктор для розробки сайту
3. Реляційна СУБД
4. Браузер для відображення та використання системи
5. Середовище розробки програмного забезпечення

Для розробки необхідні такі програмні засоби:

1. Інтегроване середовище розробки;
2. Система контролю версій;
3. Фреймворк генерування документації програмного коду;
4. Графічний редактор для створення діаграм.

Таким чином, сформульовані вимоги окреслюють ключові функціональні та технічні характеристики майбутньої системи: структуроване зберігання даних, автоматизація обліку, зручний інтерфейс та захист інформації. Визначені програмні й технічні засоби створюють основу для розробки концепції інформаційної системи, яка інтегрує ці вимоги у цілісну архітектуру.

1.4 Розробка концепції інформаційної системи

Концепція інформаційної системи передбачає створення багаторівневої архітектури, що складається з бази даних, серверної частини та клієнтського інтерфейсу. Основними функціями є автоматизація обліку, аналітика даних та формування звітності. Система має забезпечувати захист інформації, масштабованість та інтеграцію з існуючими корпоративними ресурсами.

Основними функціональними можливостями цієї підсистеми мають бути:

- Авторизація користувачів у системі. Передбачено реєстрацію та вхід для співробітників аптеки (касирів, фармацевтів, адміністраторів) та клієнтів, що забезпечить захист персональних даних і контроль доступу до різних рівнів функціоналу.
- Облік покупок. Система має фіксувати всі здійснені покупки, формувати електронні чеки, вести історію транзакцій для кожного клієнта.
- Інтеграція з бонусною/лояльнісною програмою. Завдяки обліку покупок система може автоматично нараховувати бонуси, знижки чи інші заохочення для постійних клієнтів, особливо для пенсіонерів і соціально вразливих груп.

Реалізація цієї підсистеми дозволить не лише оптимізувати процеси продажу й обслуговування, а й підвищити лояльність клієнтів, оперативно реагувати на їхні потреби та формувати довгострокові стосунки з покупцями,

тож впровадження сучасної автоматизованої підсистеми стане вагомим кроком до сталого розвитку мережі аптек.

За результатами сформованих вище вимог розроблено концепцію інформаційної підсистеми та її модель, відповідно до функціональних вимог а також зображено на рисунку 1.2:

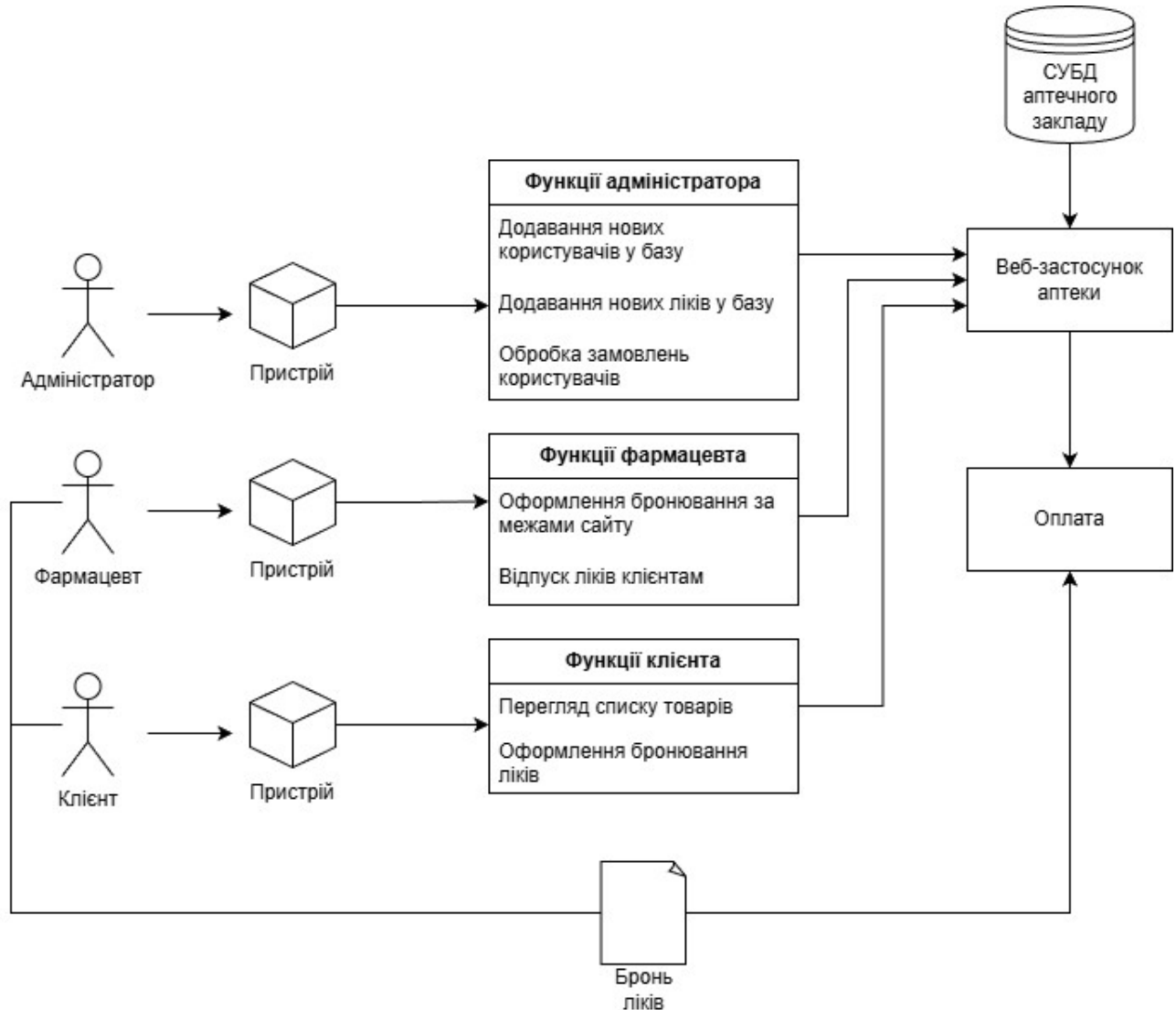


Рис. 1.2 Модель концепції інформаційної системи

Виходячи зі схеми моделі інформаційної підсистеми, можна побачити що побутовий клієнт взаємодіє з базою даних через адміністратора або напряду. Це дозволяє автоматизувати процес, дозволяючи людині економити свій час на поході в фізичний заклад.

Технічні засоби:

Системні вимоги для нормальної роботи клієнтської частини:

1. Процесор — багатоядерний (наприклад, Intel Core i5 10-го покоління або AMD Ryzen 5), що забезпечує стабільну роботу веб-застосунку.
2. Оперативна пам'ять — від 8 ГБ, що дозволяє комфортно працювати з браузером та багатозадачністю.
3. Графічна підсистема — інтегрована або дискретна відеокарта з підтримкою апаратного прискорення (Intel UHD Graphics, AMD Radeon Vega або NVIDIA GeForce GTX серії 16 і вище).
4. Місце на диску — від 20 ГБ вільного простору, бажано SSD-накопичувач для швидкого завантаження та роботи системи.
5. Операційна система — Windows 10/11, macOS 12+, або сучасні дистрибутиви Linux (Ubuntu, Fedora), що підтримують сучасні браузери.
6. Браузер — останні версії Chrome, Edge, Firefox або Safari з підтримкою HTML5, CSS3 та JavaScript ES6+.
7. Мережа — стабільне інтернет-з'єднання зі швидкістю від 10 Мбіт/с для коректної роботи з онлайн-замовленнями та інтерактивними елементами.
8. Додаткові пристрої — клавіатура, миша/тачпад, монітор з роздільною здатністю Full HD (1920×1080) або вище.

Системні вимоги які стосуються серверної частини:

1. Процесор — багатоядерний серверний CPU (наприклад, Intel Xeon Silver або AMD EPYC), що забезпечує високу продуктивність при багатопоточних обчисленнях.
2. Оперативна пам'ять — від 32 ГБ, з можливістю масштабування до 64–128 ГБ для роботи з великими обсягами даних та одночасними запитами користувачів.
3. Накопичувачі — SSD-масиви від 1 ТБ з підтримкою RAID для швидкого доступу та резервування даних.
4. Мережа — гігабітний Ethernet (1 Gbps) або вище, з можливістю підключення до оптичних каналів для високої пропускної здатності.

5. Операційна система — сучасні серверні ОС: Linux (Ubuntu Server, CentOS, Debian) або Windows Server 2022, що підтримують роботу з контейнерами та віртуалізацією.
6. Віртуалізація та контейнеризація — підтримка Docker та Kubernetes для масштабованості та гнучкого розгортання застосунків.
7. Безпека — апаратна підтримка шифрування (TPM-модулі), регулярні оновлення ОС та використання міжмережевих екранів.
8. Моніторинг та адміністрування — системи моніторингу (Prometheus, Grafana) для контролю продуктивності та стану серверів.

Програмні засоби:

Для розробки веб-застосунку були обрані мова програмування Java та система збірки Maven, так як автоматизована інформаційна система мережі аптек «Аптека плюс» інтегрована лише між центральним офісом, регіональними представництвами та аптеками вже використовує ці засоби. А для збереження даних використовується СУБД MySQL (з аналогічних причин, приведених вище).

Код програмних модулів системи розроблятиметься мовою Java. Згідно з дослідженням[7] IDE відіграють важливу роль у тому, як розробники Java виконують свої завдання, і IntelliJ IDEA залишається беззаперечним лідером із 72% респондентів, при цьому 95% користувачів IntelliJ IDEA повідомляють про зростання продуктивності, а дві третини з них вважають це зростання значним. Тому для написання коду обрано середовище IntelliJ IDEA.

Серверна частина системи розробляється за допомогою фреймворка Spring [8]. Spring є одним із найпопулярніших Java-фреймворків, що забезпечує створення масштабованих та гнучких корпоративних застосунків. Його модульна архітектура включає Spring Core, Spring MVC та Spring Data JPA, що дозволяє ефективно реалізовувати інверсію керування, веб-інтерфейси та роботу з базами даних. Використання Spring Boot значно спрощує процес конфігурації

та розгортання, забезпечуючи швидкий старт проєктів і зменшуючи кількість рутинного коду. Завдяки активній спільноті та широкій екосистемі Spring підтримує інтеграцію з сучасними технологіями, що робить його оптимальним вибором для серверної частини системи.

Авторизація користувачів та шифрування деяких даних здійснюється за допомогою Spring Security, що є потужним модулем для реалізації механізмів автентифікації та контролю доступу. Spring Security — це фреймворк, який зосереджується на забезпеченні як автентифікації, так і авторизації для Java-застосунків. Як і всі проєкти Spring, справжня сила Spring Security полягає в тому, наскільки легко його можна розширити, щоб задовольнити індивідуальні вимоги. Він підтримує різні методи авторизації, включно з ролями та правами доступу, інтеграцію з OAuth2 та JWT, а також забезпечує захист від поширених атак, таких як CSRF чи XSS. Це дозволяє гарантувати високий рівень безпеки системи та захист конфіденційних даних користувачів.[9]

Спливаючі вікна були запрограмовані мовою JavaScript. Сучасний JavaScript — це “безпечна” мова програмування. Вона не надає низькорівневого доступу до пам’яті чи процесора, оскільки була створена для браузерів, які цього не потребують. Можливості JavaScript значно залежать від середовища, у якому виконується скрипт. Наприклад, Node.js підтримує функції, які дозволяють JavaScript читати/записувати довільні файли, здійснювати мережеві запити тощо. Вбудована в браузер JavaScript може робити все, що пов’язано з управлінням вебсторінками, взаємодією з користувачем та вебсервером.[10]

Зовнішній вигляд має більш приємний для людського ока вигляд завдяки механізму шаблонів Thymeleaf, мови програмування CSS та іконок Bootstrap. Для створення діаграм під час розробки використовується програма Draw.io.

Отже, концепція системи передбачає багаторівневу архітектуру з базою даних, серверною частиною та клієнтським інтерфейсом, що забезпечує автоматизацію процесів та захист інформації. Вона відповідає сформованим вимогам і створює фундамент для переходу до технічного проєктування, де

будуть визначені загальносистемні, інформаційні, математичні, програмні та технічні рішення.

Висновок

У цьому розділі було здійснено комплексний аналіз теоретичних засад формування інформаційної системи для аптечної мережі. Проведено опис об'єкта дослідження, що дозволив визначити ключові характеристики фармацевтичної галузі та її сучасні тенденції. Аналіз предметної області виявив проблеми існуючої системи, зокрема відсутність електронного доступу для клієнтів та обмеженість функціональних можливостей у сфері комунікації й замовлення товарів. На основі цього сформовано вимоги до інформаційної системи, які окреслюють її функціональні, технічні та програмні характеристики. Розроблена концепція системи визначає багаторівневу архітектуру та основні принципи її роботи, що забезпечують автоматизацію процесів, захист даних та інтеграцію з корпоративними ресурсами.

Таким чином, перший розділ створює теоретичне підґрунтя для подальшої розробки технічного проєкту, у якому будуть визначені загальносистемні, інформаційні, математичні, програмні та технічні рішення, необхідні для практичної реалізації системи.

РОЗДІЛ 2. РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ АПТЕКИ

2.1 Загальносистемні рішення

У процесі розробки концепції інформаційної системи було визначено низку загальносистемних рішень, що забезпечують її ефективність та надійність. Архітектура побудована за клієнт-серверним принципом, що дозволяє розподілити навантаження між користувацьким інтерфейсом та серверною частиною. Для зберігання даних обрано реляційну СУБД, яка підтримує транзакційність та забезпечує цілісність інформації. Особливу увагу приділено питанням безпеки: впроваджено механізми автентифікації користувачів, контроль доступу за ролями. Такі рішення формують основу системи, створюючи умови для її стабільного функціонування та подальшого розвитку.

Виходячи з цього було визначено, що користувачами інформаційної системи будуть клієнти аптек, адміністратори чи провізори. Інформаційні процеси користувачів інформаційної системи та їх взаємодії зображено на діаграмі прецедентів:



Рисунок 2.1 – Діаграма прецедентів

Дана діаграма відображає основні сценарії взаємодії користувачів із системою та демонструє розподіл функціональних можливостей між різними категоріями акторів. Адміністратор відповідає за управління системою, підтвердження реєстрації клієнтів та організацію пунктів доставки, що забезпечує контроль і підтримку інфраструктури. Провізор виконує ключову операційну функцію – обробку замовлень, яка безпосередньо пов’язує клієнта з процесом отримання медикаментів. Клієнт, у свою чергу, взаємодіє із системою через реєстрацію, авторизацію та формування замовлень, що забезпечує зручність і доступність сервісу. Таким чином, діаграма прецедентів ілюструє логіку роботи системи, визначає ролі та відповідальність кожного учасника, а також формує основу для подальшого технічного проєктування.

Для початку роботи через сайт користувач повинен зареєструвати новий обліковий запис, ввівши основні дані, які просять у вікні реєстрації. Якщо ж у людини вже є акаунт, то потрібно залогінитися і після цього стає можливим виконувати базові функції користувача.

Для забезпечення прозорості та зручності, бронювання ліків представлено у вигляді діаграми, яка відображає послідовність дій від моменту додавання товару до кошика до остаточного отримання або анулювання замовлення(рис. 2.2).

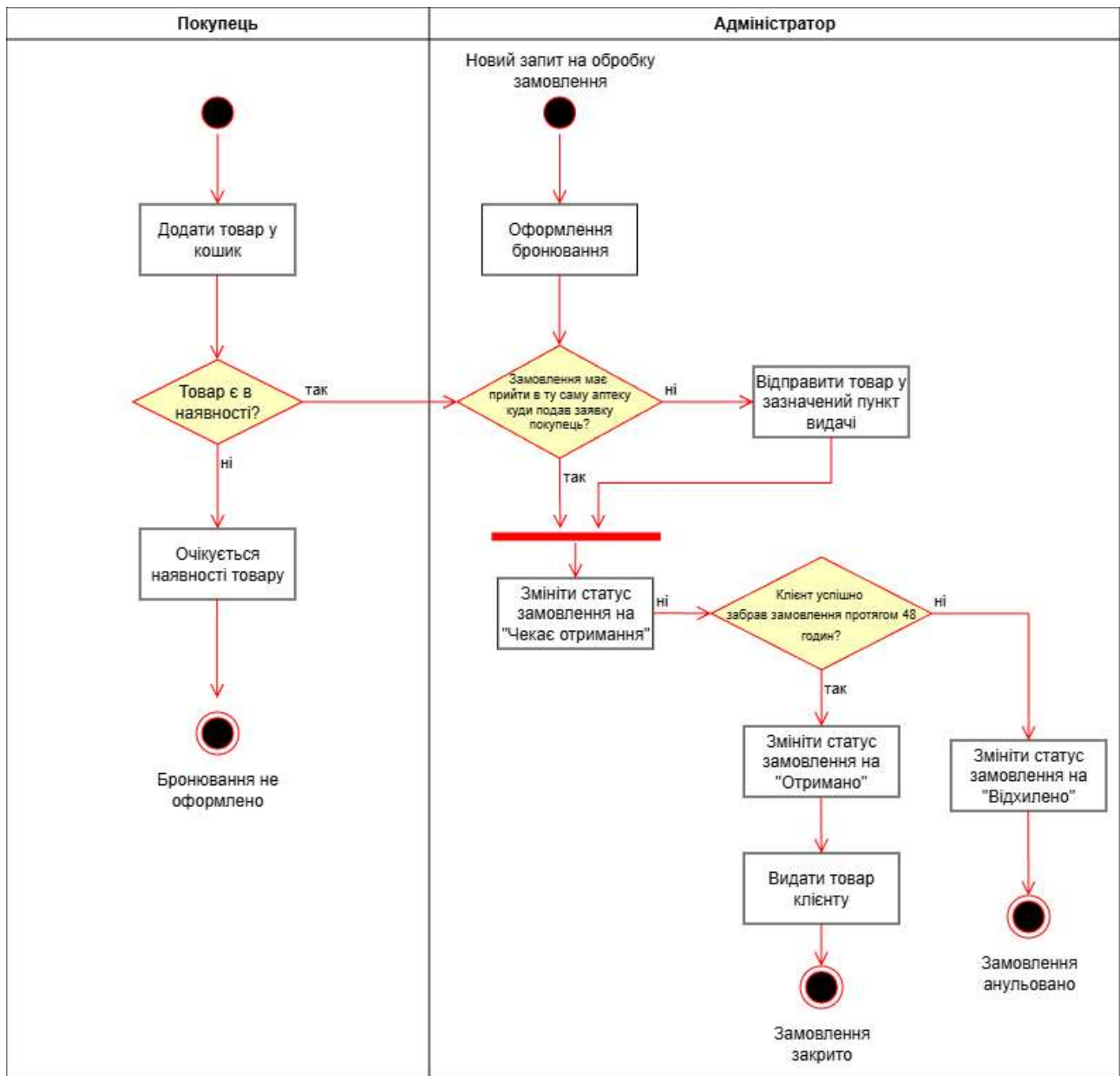


Рисунок 2.2 – Діаграма діяльності процесу обробки заявки на бронювання ліків

Діаграма демонструє два паралельні рівні роботи: дії покупця, який формує замовлення, та дії адміністратора, який відповідає за його обробку. Вона ілюструє ключові етапи: перевірку наявності товару, оформлення бронювання, визначення пункту видачі, зміну статусів замовлення та контроль отримання товару клієнтом.

Після оформлення бронювання адміністратор визначає, чи може замовлення бути видане у тій самій аптеці, де його було зроблено, або потребує перенаправлення до іншого пункту видачі. Завершальним етапом є

підтвердження отримання товару клієнтом протягом визначеного часу. Якщо замовлення не було забране, система автоматично змінює його статус на «Відхилено» та анулює бронювання.

Визначені вище загальносистемні рішення окреслюють архітектурні та організаційні засади функціонування інформаційної системи аптеки. Вони створюють основу для подальшого розгляду інформаційного забезпечення, яке деталізує способи зберігання та обробки даних у системі.

2.2 Рішення по інформаційному забезпеченню

Інформаційне забезпечення є ключовим компонентом будь-якої інформаційної системи, оскільки саме воно визначає способи збирання, зберігання, обробки та подання даних користувачам. У контексті розробки підсистеми веб-застосунку мережі аптек інформаційне забезпечення виступає основою для формування єдиної бази даних лікарських засобів, клієнтів та транзакцій, що забезпечує узгодженість і цілісність інформаційних потоків. Від якості організації інформаційного забезпечення залежить ефективність роботи всієї системи, адже воно визначає доступність даних для адміністраторів, фармацевтів та клієнтів, а також можливість інтеграції з іншими підсистемами – бухгалтерською, логістичною чи маркетинговою.

У цьому підрозділі розглядаються принципи побудови інформаційного забезпечення, структура бази даних, а також вимоги до її актуальності та захисту. Особлива увага приділяється питанням стандартизації даних, що дозволяє уникнути дублювання та забезпечити коректність обліку лікарських засобів і клієнтських операцій. Таким чином, інформаційне забезпечення виступає фундаментом для реалізації функціональних можливостей системи та гарантує її стабільну роботу в умовах зростаючих потреб фармацевтичної галузі.

Основні принципи побудови інформаційного забезпечення:

1. Цілісність та узгодженість даних

Інформаційна система повинна забезпечувати єдине джерело правдивих даних. Це означає, що всі записи про товари, клієнтів та транзакції зберігаються централізовано, без дублювання чи суперечностей. Такий підхід гарантує коректність обліку та дозволяє уникнути помилок при аналізі чи формуванні звітності.

2. Стандартизація та уніфікація

Використання єдиних форматів і класифікаторів (наприклад, міжнародних кодів лікарських засобів, стандартів для персональних даних) забезпечує сумісність системи з іншими програмними продуктами та мінімізує ризик помилок при введенні інформації. Це особливо важливо для аптечної мережі, де дані мають відповідати вимогам фармацевтичної галузі.

3. Актуальність та своєчасність

Дані повинні регулярно оновлюватися, щоб користувачі отримували достовірну інформацію про наявність товарів, їхню ціну та характеристики. Несвоєчасне оновлення може призвести до невдоволення клієнтів та втрати довіри до системи, тому актуальність є критичною умовою її ефективності.

4. Доступність та зручність подання

Інформація має бути представлена у зрозумілому вигляді для різних категорій користувачів – адміністраторів, фармацевтів та клієнтів. Це передбачає створення інтуїтивного інтерфейсу, багаторівневого доступу та використання зрозумілих форм звітності. Зручність подання даних напряму впливає на швидкість роботи персоналу та задоволеність клієнтів.

5. Захист та конфіденційність

Інформаційне забезпечення повинно відповідати вимогам законодавства щодо захисту персональних даних (GDPR, українські норми). Реалізація

механізмів авторизації, шифрування та контролю доступу гарантує безпеку інформації та захист від несанкціонованого використання.

6. Масштабованість та гнучкість

Система має бути здатною до розширення – додавання нових модулів, інтеграції з іншими сервісами (CRM, бухгалтерія, логістика, тощо) без порушення роботи основних процесів. Це дозволяє адаптувати систему до змін у фармацевтичному бізнесі та зростаючих потреб користувачів.

7. Автоматизація процесів

Максимальне використання автоматичних механізмів введення, обробки та аналізу даних підвищує ефективність роботи системи. Наприклад, автоматичне формування електронних чеків, нарахування бонусів чи ведення історії покупок зменшує навантаження на персонал і підвищує якість обслуговування клієнтів.

Таким чином, дотримання цих принципів дозволяє створити надійне, ефективне та сучасне інформаційне забезпечення, яке стане фундаментом для функціонування всієї системи аптечної мережі.

Інформаційне забезпечення будь-якої інформаційної системи неможливе без чітко організованої бази даних, яка виступає центральним сховищем усіх відомостей про товари, клієнтів, транзакції та внутрішні бізнес-процеси. Саме база даних забезпечує систематизацію інформації, її цілісність та узгодженість, а також створює основу для автоматизації операцій та інтеграції з іншими підсистемами.

У випадку аптечної мережі база даних виконує функцію «ядра» системи, оскільки вона акумулює ключові дані про лікарські засоби, персонал та клієнтів, а також забезпечує можливість швидкого пошуку, оновлення та аналізу інформації. Від правильності її побудови залежить ефективність роботи всієї

системи: від обліку товарів і формування замовлень до реалізації програм лояльності та контролю логістики.

Важливим аспектом є також стандартизація та уніфікація даних, що дозволяє уникнути дублювання та забезпечує сумісність із зовнішніми сервісами. Крім того, база даних повинна відповідати вимогам безпеки та конфіденційності, адже вона містить як комерційно важливу інформацію, так і персональні дані клієнтів. Побудова структури бази даних є ключовим етапом проєктування інформаційної системи. Для більш зручної роботи з таблицями баз даних використовується Hibernate.

Hibernate ORM (або просто Hibernate) – це інструмент об'єктно-реляційного відображення для мови програмування Java. Він надає основну основу для відображення об'єктно-орієнтованої моделі предметної області на реляційну базу даних. Hibernate вирішує проблеми невідповідності об'єктно-реляційного імпедансу, замінюючи прямий, постійний доступ до бази даних високорівневими функціями обробки об'єктів.

Основною його функцією є відображення класів Java на таблиці бази даних та відображення типів даних Java на типи даних SQL. Також надає засоби запитів та пошуку даних. Він генерує SQL-виклики та звільняє розробника від ручної обробки та перетворення об'єктів результуючого набору.

Зіставлення класів Java з таблицями бази даних реалізується шляхом конфігурації XML-файлу або за допомогою анотацій Java. При використанні XML-файлу Hibernate може генерувати скелет вихідного коду для класів персистенції. Це допоміжно, коли використовуються анотації. Hibernate може використовувати XML-файл або анотації Java для підтримки схеми бази даних.

Передбачено засоби для організації зв'язків «один до багатьох» та «багато до багатьох» між класами. Окрім керування зв'язками між об'єктами, Hibernate також може керувати рефлексивними зв'язками, коли об'єкт має зв'язок «один до багатьох» з іншими екземплярами цього типу класу.

Hibernate надає мову, подібну SQL, під назвою Hibernate Query Language (HQL), для написання SQL-подібних запитів до об'єктів даних Hibernate. Критеріальні запити надаються як об'єктно-орієнтована альтернатива HQL. Критеріальні запити використовуються для модифікації об'єктів та встановлення обмежень для об'єктів. HQL (Hibernate Query Language) – це об'єктно-орієнтована версія SQL. Вона генерує незалежні від бази даних запити, тому немає потреби писати запити, специфічні для бази даних. Без цієї можливості зміна бази даних вимагала б також зміни окремих SQL-запитів, що призводило б до проблем з обслуговуванням [11].

Нижче наведено фізичну модель бази даних для інформаційної системи мережі аптек, побудовану для СУБД MySQL. Генерація моделі відбувалась із використанням ORM-фреймворку Hibernate, що відображає класи-сутності предметної області та їхні взаємозв'язки, реалізовані мовою Java із відповідними анотаціями, у таблиці реальної СУБД. Таким чином отримаємо діаграму фізичної моделі даних (рис. 2.3).

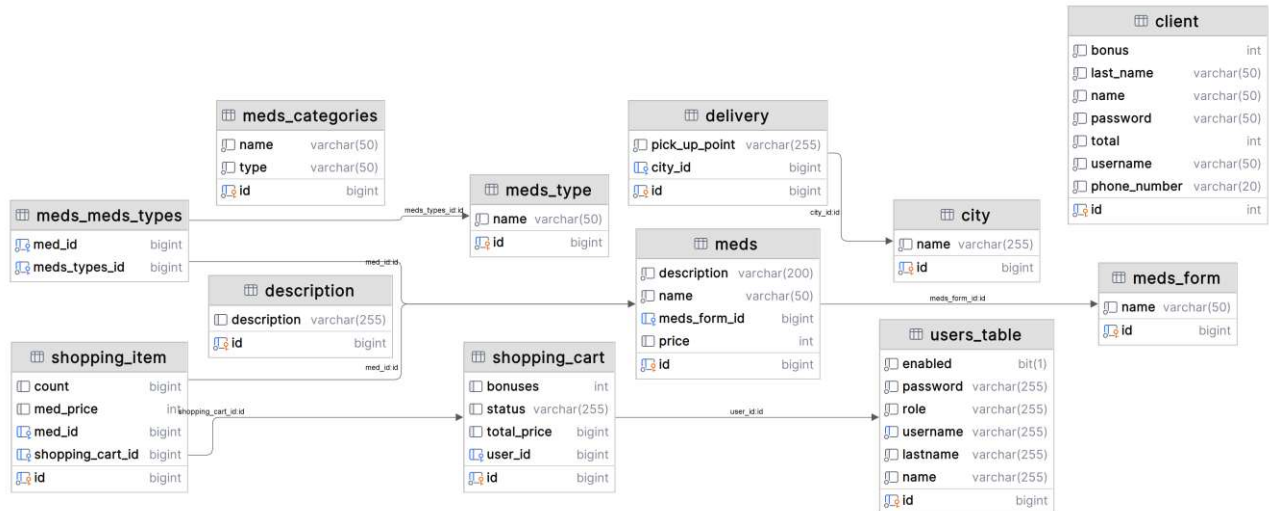


Рисунок 2.3 – Фізична модель даних

Реляційна модель бази даних системи управління медикаментами та процесами онлайн-шопінгу представлена у вигляді ERD-діаграми, що відображає основні сутності, їх атрибути та взаємозв'язки.

Сутності та атрибути

- *City* – сутність для збереження інформації про міста, що містить атрибут ідентифікатора (id) та назву (name).
- *Delivery* – сутність, що моделює пункти видачі, має атрибут pick_up_point та зовнішній ключ city_id, який забезпечує зв'язок із таблицею *City*.
- *Users_table* – сутність для системних користувачів, що включає атрибути автентифікації (username, password), ролі (role), стан активності (enabled) та персональні дані (name, lastname).
- *Client* – сутність для клієнтів, яка містить персональні дані (ім'я, прізвище, телефон), а також бізнес-атрибути (bonus, total).
- *Meds_form* – сутність, що визначає форму випуску медикаментів (наприклад, таблетки, капсули).
- *Meds* – центральна сутність, яка описує медикаменти: назву (name), опис (description), ціну (price) та зовнішній ключ meds_form_id.
- *Meds_type* – сутність для класифікації медикаментів за типами.
- *Meds_meds_types* – проміжна сутність, що реалізує зв'язок «багато-до-багатьох» між *Meds* та *Meds_type*.
- *Description* – сутність для збереження розширених текстових описів ліків.
- *Shopping_cart* – сутність, що моделює кошик користувача, містить атрибути bonuses, status, total_price та зовнішній ключ user_id.
- *Shopping_item* – сутність для конкретних позицій у кошику, що включає кількість (count), ціну (med_price), а також зовнішні ключі med_id та shopping_cart_id.

Таблиця 1. Опис сутності City

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
name	VARCHAR	Назва міста

Таблиця 2. Опис сутності Delivery

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
pick_up_point	VARCHAR	Назва пункту видачі
city_id	INT(FK)	Зовнішній ключ на City

Таблиця 3. Опис сутності Users_table

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
username	VARCHAR	Ім'я користувача (логін)
password	VARCHAR	Пароль
role	VARCHAR	Роль користувача
enabled	BOOLEAN	Стан активності
name	VARCHAR	Ім'я користувача
lastname	VARCHAR	Прізвище користувача

Таблиця 4. Опис сутності Client

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
name	VARCHAR	Ім'я клієнта
lastname	VARCHAR	Прізвище клієнта
phone	VARCHAR	Телефон
bonus	INT	Бонуси клієнта
total	DECIMAL	Сума покупок

Таблиця 5. Опис сутності Meds_form

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
form_name	VARCHAR	Форма випуску (наприклад, таблетки, капсули)

Таблиця 6. Опис сутності Meds

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
name	VARCHAR	Назва медикаменту
description	TEXT	Опис медикаменту
price	DECIMAL	Ціна
meds_form_id	INT(FK)	Зовнішній ключ на Meds_form

Таблиця 7. Опис сутності Meds_type

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
type_name	VARCHAR	Тип медикаменту

Таблиця 8. Опис сутності Meds_meds_types

Поле	Тип даних	Опис
meds_id	INT(PK,FK)	Зовнішній ключ на Meds
meds_type_id	INT(PK,FK)	Зовнішній ключ на Meds_type

Таблиця 9. Опис сутності Description

Поле	Тип даних	Опис
meds_id	INT(PK,FK)	Первинний ключ, Зовнішній ключ на Meds
text_description	TEXT	Розширений текстовий опис ліків

Таблиця 10. Опис сутності Shopping_cart

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
bonuses	INT	Нараховані бонуси
status	VARCHAR	Статус кошика
total_price	DECIMAL	Загальна сума
user_id	INT(FK)	Зовнішній ключ на Users_table

Таблиця 11. Опис сутності Shopping_item

Поле	Тип даних	Опис
id	INT(PK)	Ідентифікатор запису, Первинний ключ
count	INT	Кількість одиниць товару
med_price	DECIMAL	Ціна медикаменту
med_id	INT(FK)	Зовнішній ключ на Meds
shopping_cart_id	INT(FK)	Зовнішній ключ на Shopping_cart

Зв'язки між сутностями

- *City* ↔ *Delivery*: зв'язок «один-до-багатьох».
- *Users_table* ↔ *Shopping_cart*: зв'язок «один-до-багатьох».
- *Shopping_cart* ↔ *Shopping_item*: зв'язок «один-до-багатьох».
- *Shopping_item* ↔ *Meds*: зв'язок «багато-до-одного».
- *Meds* ↔ *Meds_form*: зв'язок «багато-до-одного».
- *Meds* ↔ *Meds_type* через *Meds_meds_types*: зв'язок «багато-до-багатьох».
- *Meds* ↔ *Description*: зв'язок «один-до-одного».

Таким чином, модель даних забезпечує комплексне представлення предметної області: від класифікації та опису медикаментів до організації процесів покупки, бонусної системи та доставки. Використання зовнішніх ключів та проміжних таблиць дозволяє реалізувати як прості, так і складні зв'язки між сутностями, що забезпечує цілісність та узгодженість даних у системі.

Побудова структури бази даних не обмежується лише визначенням таблиць та їхніх атрибутів. Важливим етапом є також встановлення взаємозв'язків між сутностями, адже саме вони забезпечують логіку функціонування всієї системи. Завдяки цьому база даних перетворюється на цілісну модель, яка відображає реальні бізнес-процеси аптечної мережі – від обліку товарів і клієнтів до контролю транзакцій та реалізації програм лояльності.

Для наочності використовується діаграма «сутність-зв'язок», яка демонструє, як окремі таблиці взаємодіють між собою. Вона дозволяє зрозуміти структуру даних, визначити ключові зв'язки та забезпечити узгодженість інформаційних потоків. Саме ця діаграма є основою для подальшої реалізації програмного забезпечення, оскільки вона формує концептуальну модель даних, що відповідає вимогам користувачів та технічним стандартам.

Нижче наведено діаграму класів сутностей, яка наочно показує зв'язки між ними та параметри які має в собі кожен клас (рис. 2.4).

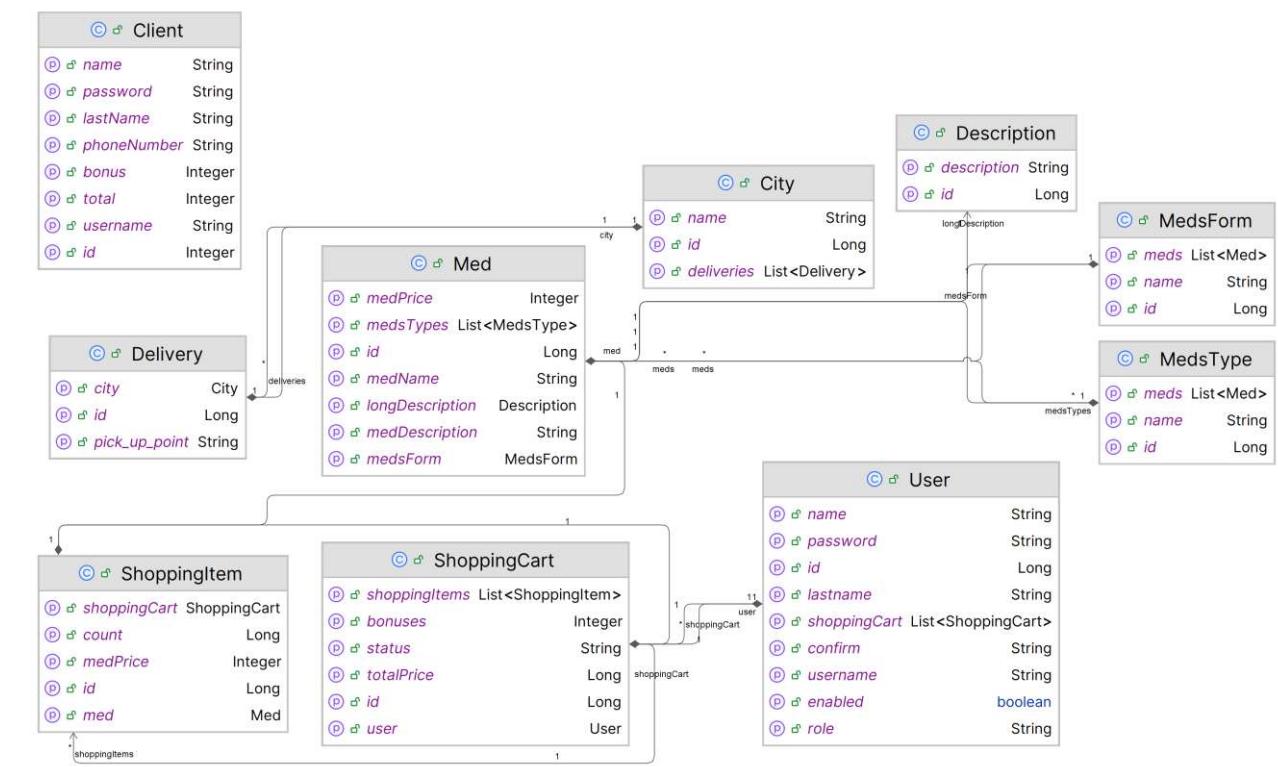


Рис. 2.4 – Діаграма класів сутностей

Ця діаграма класів описує структуру інформаційної системи для управління медикаментами та онлайн-шопінгу. Вона показує основні сутності (класи), їхні атрибути та зв'язки між ними, показує комплексну модель онлайн-аптеки: від опису ліків і їх класифікації до процесу покупки, бонусної системи та доставки. Вона поєднує бізнес-логіку (кошик, бонуси, клієнти) з довідковою інформацією (категорії, типи, форми медикаментів).

Основні блоки системи

- Місто та доставка
 - Клас *City* зберігає назву та список доставок.
 - Клас *Delivery* містить інформацію про місто, ідентифікатор та пункт видачі. → Це дозволяє організувати логістику та відстежувати, де і як відбувається доставка.
- Медикаменти та їх опис
 - *Med* – центральний клас, що описує ліки: назву, ціну, форму, типи, короткий і довгий опис.

- *MedsForm* визначає форму випуску (таблетки, капсули тощо).
- *MedsType* групує ліки за типами.
- *Description* дає розширений текстовий опис.
- Кошик та покупки
 - *ShoppingCart* містить список товарів (*ShoppingItem*), бонуси, статус, загальну ціну та користувача.
 - *ShoppingItem* пов'язаний із конкретним медикаментом, кількістю та ціною. → Це моделює процес покупки та накопичення бонусів.
- Користувачі системи
 - *User* має атрибути для авторизації (username, password, role, enabled), а також список кошиків.
 - *Client* містить додаткові дані про клієнта: ім'я, прізвище, телефон, бонуси, загальні витрати. → Відображає різні рівні користувачів: системні та клієнтські.

Логіка взаємодії

- Користувач (*User/Client*) формує *ShoppingCart*, додає *ShoppingItem*, які посилаються на *Med*.
- *Med* належить до певної *MedsForm*, має *MedsType* та *Description*.
- Доставка організовується через *Delivery*, що прив'язана до *City*.

2.3 Рішення по математичному забезпеченню

Математичне забезпечення інформаційної системи аптечної мережі визначає сукупність методів, алгоритмів та моделей, що забезпечують коректність і ефективність її функціонування. Основною метою математичного забезпечення є формалізація процесів обробки даних, оптимізація пошуку лікарських засобів, управління запасами та прогнозування попиту.

1. Алгоритми пошуку та фільтрації.

Для реалізації швидкого доступу до інформації про лікарські засоби застосовується алгоритм оптимізованого пошуку за ключовими параметрами – назвою препарату, діючою речовиною, виробником, категорією та ціновим діапазоном. Використовується комбінований підхід, що поєднує методи індексації та часткової відповідності (fuzzy search), що дозволяє враховувати можливі орфографічні помилки користувача.

2. Алгоритми лояльності та бонусних нарахувань.

Для реалізації програми лояльності використовується математична модель нарахування бонусів, яка базується на функції залежності між сумою покупки та коефіцієнтом клієнтської активності.

$$B = S \cdot ka$$

де B – кількість бонусних балів, S – сума покупки, ka – коефіцієнт активності клієнта, що визначається частотою покупок та середнім чеком. Ця модель дозволяє адаптивно змінювати рівень заохочення залежно від поведінки користувача.

3. Моделі прогнозування попиту.

Для оцінки майбутнього попиту на лікарські засоби застосовується регресійна модель, що враховує сезонні коливання, історичні дані продажів та зовнішні фактори (епідеміологічні тенденції, зміни цін).

$$Dt = \alpha + \beta_1 Pt + \beta_2 St + \beta_3 Tt + \varepsilon t$$

де Dt – прогнозований попит у момент часу t , Pt – ціна препарату, St – сезонний коефіцієнт, Tt – тренд продажів, εt – випадкова похибка. Модель дозволяє формувати рекомендації щодо оптимального поповнення запасів та уникати дефіциту товарів.

4. Алгоритми оптимізації запасів.

Для управління складськими залишками використовується метод економічного розміру замовлення (EOQ), який мінімізує сумарні витрати на зберігання та поповнення запасів.

$$Q^* = 2DSH$$

де Q^* – оптимальний розмір замовлення, D – річний попит, S – витрати на оформлення замовлення, H – витрати на зберігання одиниці товару. Цей підхід забезпечує баланс між наявністю товарів і витратами підприємства.

5. Алгоритми безпеки даних.

Для захисту інформації застосовуються криптографічні хеш-функції (SHA-256) та алгоритми шифрування паролів користувачів. Це гарантує цілісність даних і захист персональної інформації.

2.4 Рішення по програмному забезпеченню

Для реалізації інформаційної системи було обрано комбінацію сучасних технологій, які забезпечують надійність, масштабованість та кросплатформеність. Основними складовими є система управління базами даних MySQL, мова програмування Java та набір фреймворків Spring.

У якості СУБД використано MySQL – потужну об'єктно-реляційну систему з відкритим вихідним кодом, яка підтримує стандарт SQL та розширює його додатковими можливостями. MySQL забезпечує відповідність принципам ACID, що гарантує цілісність даних, високу продуктивність та надійність при роботі з великими обсягами інформації. Вона сумісна з різними платформами та інтегрується з основними мовами програмування, що робить її оптимальним вибором для проєктів, які перебувають на стадії ініціалізації та можуть змінювати вимоги з часом. Код створення таблиць наведено у додатку Г.

Веб-додаток системи реалізовано мовою Java, яка є об'єктно-орієнтованою та широко застосовується у корпоративних рішеннях завдяки своїй кросплатформенності. Програма компілюється у байт-код, що виконується у середовищі JVM, забезпечуючи незалежність від операційної системи. Для розробки використано інтегроване середовище IntelliJ IDEA Ultimate Edition, яке надає широкий спектр інструментів для ефективного програмування, включаючи розвинений редактор коду та засоби інтеграції з фреймворками. Для побудови серверної частини застосовано Spring Framework – один із найпопулярніших каркасів Java EE, який забезпечує підтримку інфраструктури для створення високопродуктивних додатків. Використання Spring Boot дозволило швидко налаштувати середовище розробки та реалізувати архітектуру на основі мікросервісів, що забезпечує гнучкість та масштабованість системи. Для взаємодії з базою даних застосовано Spring Data JPA, який спрощує роботу з репозиторіями та реалізує доступ до даних на основі JPA. Механізми автентифікації та авторизації користувачів реалізовано за допомогою Spring Security, що гарантує захист персональних даних та контроль доступу до функціоналу системи.

Методи розробки програмного забезпечення

Розробка програмного забезпечення інформаційної системи аптечної мережі здійснюється на основі сучасних підходів, які поєднують гнучкість, структурованість та орієнтацію на кінцевого користувача. Основна мета застосування методів розробки полягає у забезпеченні високої якості програмного продукту, його адаптивності до змін вимог та можливості масштабування у майбутньому.

Процес створення системи організовано як послідовність етапів: аналіз вимог, проєктування, реалізація, тестування та впровадження. На кожному етапі застосовуються інструменти та практики, що дозволяють мінімізувати ризики, забезпечити прозорість процесу та контроль якості. Особлива увага приділяється

тестуванню, яке охоплює перевірку функціональності, продуктивності та безпеки системи.

Важливим аспектом є ітеративність процесу розробки, що дозволяє поступово вдосконалювати систему, враховуючи нові вимоги та відгуки користувачів. Такий підхід забезпечує постійне вдосконалення функціоналу та підвищення рівня задоволеності клієнтів.

Операційна система

Для функціонування інформаційної системи аптеки використовуються різні операційні системи, залежно від типу користувача та призначення частини системи. Серверна частина, що відповідає за обробку запитів, зберігання даних та забезпечення безпеки, розгорнута на Arch Linux. Ця операційна система характеризується високою стабільністю, гнучкістю налаштувань та мінімалістичним підходом, що дозволяє оптимізувати використання ресурсів сервера. Arch Linux підтримує сучасні технології контейнеризації, віртуалізації та автоматизації, що робить її ефективним рішенням для розробників і системних адміністраторів.

Для звичайних користувачів, які взаємодіють із веб-інтерфейсом системи, рекомендовано використання операційної системи Windows. Вона забезпечує зручність роботи, сумісність із більшістю браузерів та програмного забезпечення, а також має інтуїтивно зрозумілий інтерфейс. Windows є оптимальним вибором для кінцевих користувачів завдяки широкій підтримці драйверів, регулярним оновленням безпеки та простоті налаштування.

Поєднання Arch Linux для серверної частини та Windows для клієнтської забезпечує баланс між продуктивністю, стабільністю та зручністю використання, що сприяє ефективній роботі всієї інформаційної системи.

Сервісне програмне забезпечення

До складу сервісного програмного забезпечення входять системні утиліти, драйвери, засоби адміністрування баз даних, антивірусні програми, а також інструменти резервного копіювання та моніторингу. На серверній частині використовуються сервіси для автоматичного оновлення системи, управління мережею та контролю стану серверів. Це дозволяє підтримувати безперервну роботу веб-додатку, своєчасно виявляти збої та запобігати втраті даних. Для клієнтської частини застосовуються стандартні засоби обслуговування операційної системи, зокрема програми для очищення тимчасових файлів, оновлення драйверів та антивірусного захисту.

Аптечні комп'ютери отримують доступ до Інтернету за допомогою мережевого обладнання – маршрутизаторів, комутаторів та модемів, які з'єднують локальну мережу аптеки з глобальною мережею. Основним каналом є широкосмуговий Ethernet або оптоволоконні лінії, що забезпечують стабільну швидкість передачі даних. У деяких випадках використовується мобільний інтернет (3G/4G/5G) як резервний канал. Програмне забезпечення операційної системи реалізує протоколи TCP/IP, що дозволяють комп'ютерам здійснювати запити до веб-серверів, отримувати дані з бази та взаємодіяти з клієнтськими пристроями.

Протоколи Інтернету

Для коректної роботи системи застосовуються стандартні протоколи:

- TCP/IP – базовий набір протоколів для маршрутизації та доставки даних. TCP гарантує надійність передачі, IP відповідає за адресацію.
- HTTP/HTTPS – протоколи взаємодії між браузером користувача та сервером. HTTPS забезпечує захищений канал передачі даних завдяки SSL/TLS-шифруванню.
- DNS – система перетворення доменних імен у IP-адреси, що дозволяє користувачам працювати з веб-інтерфейсом за зрозумілими адресами.

- SMTP/IMAP/POP3 – протоколи електронної пошти, які можуть використовуватися для відправки повідомлень клієнтам (наприклад, підтвердження замовлення чи нагадування про бронювання).

Функціональні можливості інформаційної системи

Інформаційна система аптечної мережі створена таким чином, щоб забезпечити зручний доступ до даних про лікарські засоби та оптимізувати процес їх замовлення. Користувацький інтерфейс побудований на принципах простоти та інтуїтивності, що дозволяє швидко знаходити потрібну інформацію та здійснювати необхідні операції. Основна увага приділяється організації довідкових даних про ліки, їх категорії та форми прийняття, а також забезпеченню можливості замовлення товарів через інтегрований веб-додаток.

Система передбачає багаторівневу структуру доступу: звичайні користувачі мають можливість переглядати інформацію про товари, формувати замовлення та взаємодіяти зі службою підтримки, тоді як адміністратори отримують розширений набір інструментів для управління базою даних, клієнтами та пунктами видачі. Такий підхід забезпечує баланс між зручністю для клієнтів та ефективністю управління для персоналу.

Користувацький інтерфейс реалізовано у вигляді вкладок, кожна з яких відповідає за окремий блок функціональності. Це дозволяє структурувати роботу системи та зробити її зрозумілою навіть для користувачів без спеціальної технічної підготовки. У наступних підпунктах наведено детальний опис можливостей системи для клієнтів та адміністраторів.

- Перегляд списку всіх категорії ліків
- Перегляд списку всіх форм прийняття ліків
- Перегляд докладного списку ліків, які відносяться до вибраної категорії
- Перегляд докладного списку ліків, які відносяться до вибраної форми прийняття

1. Вкладка «Замовити ліки»:

- Перегляд список всіх аптечних товарів на сайті
- Пошук необхідних товарів через фільтр категорії ліків
- Пошук необхідних товарів через фільтр форми ліків
- Пошук необхідних товарів використовуючи пошукове поле
- Додавання вибраних товарів до кошика

2. Вкладка «Кошик»

- Перегляд вибору товарів в кошику
- Редагування кількості товарів у кошику
- Видалення товарів із кошика
- Перегляд конечної суми вартості товарів у кошику
- Вибір пункту видачі товарів після бронювання ліків у кошику
- Відслідковування поточного статусу замовлення
- Перегляд замовлень зроблених за увесь час існування акаунту та їх статус

3. Вкладка «Служба підтримки»

- Звернення до служби підтримки з поточною проблемою

Для адміністратора в свою чергу завчасно вже доданий обліковий запис, входячи в яких людина має можливість переглядати всі сторінки зі сторони звичайного клієнта а також доступ до специфічного інструментарію, дозволяючи додавати, редагувати та видаляти дані з бази даних. Основні функції приведені нижче:

1. Вкладка «Адміністрація»:

- Перегляд списку всіх існуючих в базі даних товарів
- Пошук необхідних товарів за допомогою фільтру форми прийняття ліків
- Пошук необхідних товарів за допомогою фільтру типу ліків
- Пошук необхідних товарів за допомогою поля пошуку

- Додавання нових препаратів (з параметрами: назва, короткий опис, ціна, форма прийняття ліків)
- Редагування основних параметрів кожного препарату
- Видалення товарів з бази даних
- Доступ до вкладки «додатково» де можливо додати повний опис товару, очистити поле з описом, додати тип ліків, видалити конкретний тип)

2. Вкладка «Довідник»:

- Перегляд всіх існуючих в базі типів ліків
- Перегляд всіх існуючих форм прийняття ліків
- Додання нових типів ліків
- Редагування існуючих типів ліків
- Видалення існуючих типів ліків з бази даних
- Додання нових форм прийняття ліків
- Редагування існуючих форм прийняття ліків
- Видалення існуючих форм прийняття ліків

3. Вкладка «Клієнти»:

- Перегляд списку всіх зареєстрованих в базі клієнтів
- Перегляд кількості бонусів, які має клієнт за покупки
- Додавання нових клієнтів до бази даних(у разі якщо у людини не вийшло зробити це через сайт з будь-яких причин)
- При додаванні нового клієнта вводити наступні дані: Ім'я, Прізвище, Нікнейм, номер телефону
- Редагування даних конкретного клієнта
- Видалення існуючого клієнта з бази даних

4. Вкладка «Пункти видачі»:

- Перегляд списку всіх пунктів видачі товарів

- Додавання нових пунктів видачі товарів
- Редагування даних конкретних пунктів видачі товарів
- Видалення існуючих пунктів видачі товарів з бази даних

2.5 Рішення по технічному забезпеченню

Технічне забезпечення є фундаментальною складовою інформаційної системи, адже саме апаратні ресурси визначають її продуктивність, надійність та масштабованість. Вибір обладнання має враховувати специфіку предметної області, обсяг даних, кількість користувачів та вимоги до безпеки. Для аптечної мережі технічні засоби повинні гарантувати безперервну роботу веб-застосунку, швидкий доступ до бази даних, стабільність мережових з'єднань та можливість інтеграції з іншими сервісами.

Основними компонентами технічного забезпечення є серверна інфраструктура, клієнтські пристрої, мережеве обладнання та засоби захисту інформації. Сервери виконують ключову роль, адже саме вони забезпечують зберігання бази даних, виконання бізнес-логіки та обробку запитів користувачів. Від їхньої продуктивності залежить швидкість роботи системи, можливість масштабування та безперервність функціонування. Серверна частина також відповідає за резервне копіювання та відновлення даних у разі збоїв.

Клієнтські пристрої – комп'ютери адміністраторів, касові термінали фармацевтів та мобільні пристрої клієнтів – забезпечують взаємодію користувачів із системою через веб-інтерфейс. Важливо, щоб вони підтримували сучасні браузерери, мали достатню продуктивність та були зручними для щоденного використання. Мережеві комутатори, маршрутизатори та точки доступу формують основу для стабільного обміну даними між сервером, аптеками та клієнтами. Висока пропускна здатність та надійність мережі гарантують швидкий доступ до інформації, синхронізацію даних у реальному часі та інтеграцію з зовнішніми сервісами.

До технічного забезпечення входять також апаратні та програмні засоби безпеки: міжмережеві екрани, системи шифрування, антивірусні рішення. Вони захищають персональні дані клієнтів, комерційну інформацію та фінансові транзакції від несанкціонованого доступу чи кібератак.

Особливу увагу приділено носіям даних. Найбільш поширеними залишаються HDD (жорсткі магнітні диски) та SSD (твердотільні накопичувачі). HDD є традиційною технологією зберігання з нижчою вартістю при великій ємності, тоді як SSD забезпечують значно вищу швидкість доступу, надійність та енергоефективність завдяки відсутності рухомих частин. SSD доступні у діапазоні від 120 ГБ до понад 30 ТБ, HDD – від 250 ГБ до 20 ТБ. Використання SSD у серверних системах дозволяє досягати більшої продуктивності при меншій кількості пристроїв, що знижує загальну вартість володіння (TCO).

Основою SSD є NAND-пам'ять – енергонезалежна флеш-технологія, здатна зберігати дані навіть без живлення. Існують різні її типи: SLC (найнадійніший, але дорогий), MLC (компроміс між ціною та продуктивністю), TLC (дешевший, але менш витривалий) та 3D NAND (сучасна багатошарова технологія з високою щільністю). Для серверної частини системи було обрано SSD TLC, які забезпечують достатню швидкість роботи при оптимальній вартості. Для резервного копіювання баз даних використовуються HDD, що дозволяє економно зберігати великі обсяги інформації. На клієнтських комп'ютерах достатньо HDD, оскільки основні дані зберігаються на сервері.

Відповідно до системних вимог було визначено таке технічне забезпечення:

Клієнтський комп'ютер

- Процесор: Intel Core i5-12400 (6 ядер, 12 потоків, 2.5–4.4 GHz)
- Оперативна пам'ять: 8 GB DDR4
- Накопичувач: SSD 256 GB (для швидкого завантаження ОС та роботи з браузером)
- Монітор: Dell P2419H (24", Full HD)

- Периферія: Logitech MK270 (клавіатура + мишка)

Серверний комп'ютер

- Процесор: AMD EPYC 7302P (16 ядер, 32 потоки, 3.0 GHz)
- Оперативна пам'ять: 32 GB DDR4 ECC
- Накопичувачі: SSD NVMe 512 GB (для основної бази даних) + HDD 2 TB (для резервного зберігання)
- Джерело безперебійного живлення: APC Smart-UPS 2200VA
- Мережеве обладнання: Cisco RV340 Dual WAN Gigabit Router

Додаткове обладнання

- Мережеве підключення: оптоволоконний канал 1 Gbps + резервний LTE-роутер для аварійного доступу

Висновок

У другому розділі було розглянуто ключові аспекти технічного та програмного забезпечення інформаційної системи аптечної мережі. Системний аналіз показав, що ефективність роботи залежить від комплексного поєднання серверної інфраструктури, клієнтських пристроїв, мережевого обладнання та засобів захисту даних. Вибір операційних систем, сервісного програмного забезпечення та інтернет-протоколів забезпечує стабільність функціонування, безпеку та можливість масштабування системи.

Особливу увагу приділено носіям даних, адже саме вони визначають швидкість доступу до інформації та надійність її зберігання. Порівняння HDD та SSD показало, що використання твердотільних накопичувачів у серверній частині дозволяє досягти високої продуктивності при оптимальних витратах, тоді як HDD залишаються доцільними для резервного копіювання великих обсягів даних.

Технічне забезпечення системи сформовано з урахуванням сучасних вимог до продуктивності, надійності та безпеки. Воно створює основу для реалізації програмних рішень, забезпечує безперервну роботу веб-застосунку та підтримує інтеграцію з іншими сервісами. Це дозволяє інформаційній системі аптечної мережі відповідати потребам користувачів та забезпечувати високий рівень сервісу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЄКТА ІНФОРМАЦІЙНОЇ СИСТЕМИ

Програмна частина інформаційної системи «Аптека плюс» реалізована як веб-додаток, що забезпечує взаємодію між клієнтами та адміністраторами. Основні модулі системи включають:

- Модуль управління товарами – дозволяє адміністраторам додавати нові препарати, редагувати характеристики та ціни, видаляти застарілі позиції.
- Модуль клієнтського доступу – забезпечує перегляд каталогу, пошук ліків за назвою чи категорією, формування замовлень та бронювання препаратів онлайн.
- Модуль авторизації та безпеки – реалізує реєстрацію користувачів, розмежування прав доступу (адміністратор, фармацевт, клієнт), захист персональних даних відповідно до вимог GDPR та українського законодавства.
- Модуль обліку покупок та бонусної програми – фіксує транзакції, формує електронні чеки, нараховує бонуси та знижки постійним клієнтам.

Реалізація є ключовим етапом, що забезпечує перехід від теоретичних рішень та технічного проєктування до створення робочої системи, здатної функціонувати у реальному середовищі. Тут детально описуються програмні модулі, їх інтеграція з базами даних, налаштування інтерфейсу користувача та організація технічної інфраструктури. Особливу увагу приділено питанням захисту даних, масштабованості та надійності системи, що є критично важливим у фармацевтичній сфері.

Інструкція користувача

В інструкції наведено опис основних дій для двох категорій користувачів – звичайних клієнтів та адміністраторів системи. Сфера застосування інформаційної системи – аптеки «Аптека плюс», де вона автоматизує процеси обліку товарів, формування замовлень та управління клієнтами.

Завдяки зрозумілому та інтуїтивному інтерфейсу будь-який користувач, незалежно від рівня технічної підготовки, може швидко освоїти роботу з системою. Функціональні модулі розташовані структуровано за розділами, що спрощує навігацію. Інформація відображається у вигляді таблиць: вибір рядка відкриває детальні дані з можливістю редагування, збереження чи видалення. Елементи інтерфейсу мають кольорове маркування – синій використовується для дій, пов'язаних із додаванням даних, червоний для операції видалення, жовтий позначає операцію редагування даних. Для зручності пошуку окремі таблиці містять поля фільтрації.

Інструкція адміністратора

Адміністратор інформаційної системи «Аптека плюс» має доступ до кількох функціональних розділів, що забезпечують повне управління даними системи. Кожен із них виконує окрему роль у підтримці актуальності інформації та контролі бізнес-процесів аптеки.

Після входу до системи адміністратор переходить до модуля довідкових даних, де здійснюється керування типами та формами медикаментів. У цьому розділі відображено дві таблиці: одна містить категорії препаратів (антибіотики, болезаспокійливі, вітаміни тощо), інша – лікарські форми (капсули, таблетки, краплі, шрот, фіточай тощо). Для кожного запису передбачено можливість редагування або видалення, а також функцію додавання нових елементів. Такий підхід дозволяє швидко оновлювати довідкову інформацію, що використовується в інших частинах системи.

Далі адміністратор може перейти до розділу управління клієнтами, який містить таблицю з основними даними користувачів: ідентифікатор, ім'я, прізвище, логін, номер телефону, кількість бонусів та загальну суму покупок. Тут можна редагувати або видаляти записи, а також створювати нові облікові записи через кнопку «Додати». Цей модуль забезпечує контроль над клієнтською базою та підтримує систему лояльності шляхом обліку бонусів.

Наступним етапом роботи адміністратора є керування пунктами видачі товарів. У цьому розділі наведено інформацію про міста та адреси, де клієнти можуть отримати свої замовлення. Для кожного пункту доступні функції редагування, видалення та додавання нових записів. Це дозволяє оперативно оновлювати логістичні дані та підтримувати актуальність інформації про місця видачі продукції в різних регіонах.

Завдяки цим функціональним можливостям адміністратор має змогу ефективно керувати всіма аспектами роботи системи – від довідкових даних до клієнтських записів і пунктів видачі. Інтерфейс побудований логічно, з чітким розподілом функцій, що забезпечує швидке виконання операцій та мінімізує ризик помилок при роботі з базою даних. Вигляд сторінок адміністратора приводиться в додатку В.

Інструкція клієнта

Клієнт інформаційної системи «Аптека плюс» має доступ до зручного веб-інтерфейсу, який дозволяє переглядати товари, формувати замовлення та взаємодіяти з аптечною мережею онлайн.

Після входу до системи користувач потрапляє на головну сторінку, де розміщено навігаційне меню з основними розділами: «Головна сторінка», «Кошик» та «Підтримка». У правому верхньому куті знаходиться кнопка «Вийти», яка завершує сеанс роботи.

У центральній частині інтерфейсу розташований каталог медикаментів, що містить зображення товарів, їх назви, ціни та кнопку «Додати у кошик». Користувач може переглядати асортимент, обирати потрібні препарати та додавати їх до кошика для подальшого оформлення замовлення.

Для зручності пошуку передбачено панель фільтрів, де можна вибрати лікарську форму, тип препарату або виконати текстовий пошук за назвою. Після

натискання кнопки «Пошук» система відображає лише ті товари, що відповідають заданим параметрам.

Після вибору товарів користувач переходить до розділу кошика, де відображається перелік доданих позицій із зазначенням форми, ціни, кількості та загальної суми покупки. Тут можна змінювати кількість товарів або видаляти непотрібні позиції. Загальна сума автоматично оновлюється після кожної зміни.

Завершальним етапом є підтвердження покупки. На сторінці оформлення замовлення користувач бачить чек із зазначенням суми та кнопкою «Підтвердити», яка активує процес бронювання або оплати товарів. Після підтвердження система формує запис про замовлення та передає його до пункту видачі, обраного клієнтом. Вигляд сторінок клієнта приводиться в додатку В.

Інструкція по формуванню і веденню бази даних

На початковому етапі здійснюється збір вихідної інформації з внутрішніх документів аптечної мережі, що містять дані про товари, клієнтів та пункти видачі. Для створення та адміністрування бази даних використовується база даних MySQL.

Введення нових записів здійснюється співробітниками аптек та адміністраторами через веб-застосунок, що дозволяє додавати препарати, редагувати їх характеристики та формувати замовлення. Усі внесені дані зберігаються у спільній базі, яка використовується на всіх автоматизованих робочих місцях і гарантує їх актуальність та унікальність. Операції редагування та видалення можуть виконувати лише користувачі з відповідними правами доступу. Адміністратор відповідає за підтримку цілісності даних, контроль правильності введених відомостей та своєчасне оновлення інформації.

Системний адміністратор бази даних здійснює постійний моніторинг її роботи, контролює використання дискового простору та організовує резервне копіювання. Копії даних створюються регулярно за визначеним графіком:

- щоденні – зберігаються протягом одного тижня;
- щотижневі – зберігаються протягом одного місяця;
- щомісячні – зберігаються протягом двох років;
- щорічні – зберігаються безстроково.

Такий підхід забезпечує захист інформації від втрат, дозволяє швидко відновити систему у випадку збоїв та підтримує стабільність роботи інформаційної системи аптечної мережі «Аптека плюс».

Підготовка до роботи інформаційної системи

Перед початком використання інформаційної системи аптечної мережі «Аптека плюс» системний адміністратор має ознайомитися з інструкцією щодо формування та ведення бази даних. Після заповнення бази актуальною інформацією про лікарські засоби, клієнтів та пункти видачі, а також після налаштування резервного копіювання, система готова до роботи.

Наступним етапом є створення облікових записів для співробітників та клієнтів. Для кожного користувача визначаються ролі та права доступу: звичайні клієнти можуть переглядати каталог товарів і формувати замовлення, фармацевти – вести облік продажів, а адміністратори – редагувати дані та контролювати роботу системи.

Налаштування робочих комп'ютерів, реєстрація користувачів та призначення їхніх ролей здійснюється системним адміністратором. Саме він відповідає за коректність роботи системи, безпеку даних та доступність усіх функцій для різних категорій користувачів.

Інструкція на випадок аварійних ситуацій

Аварійні ситуації в інформаційній системі аптечної мережі «Аптека плюс» розглядаються як будь-які збої, що можуть призвести до втрати доступу до даних, порушення процесу обслуговування клієнтів або загрози цілісності бази даних. У випадку виникнення проблеми співробітники аптеки повинні одразу

повідомити адміністратора системи та тимчасово перейти на паперове оформлення замовлень і видачу товарів за накладними. Усі операції, виконані під час збою, фіксуються у спеціальному журналі, щоб після відновлення роботи їх можна було внести до системи.

Адміністратор у такій ситуації перевіряє стан серверів і бази даних, визначає характер збою та вживає заходів для його усунення. Якщо пошкоджено дані, проводиться відновлення з останньої резервної копії. У випадку перевантаження системи може бути тимчасово обмежений доступ клієнтів, залишаючи лише внутрішні функції для персоналу. При апаратних несправностях адміністратор звертається до технічної служби та організовує заміну обладнання.

Резервне копіювання виконується регулярно, що дозволяє мінімізувати втрати інформації. Щоденні копії зберігаються протягом тижня, щотижневі – протягом місяця, щомісячні – протягом двох років, а щорічні – постійно. Це забезпечує можливість швидкого відновлення роботи системи у разі критичних збоїв.

Після усунення аварії адміністратор інформує керівництво аптечної мережі про характер проблеми та час, необхідний для її вирішення. Далі проводиться перевірка цілісності даних і тестування основних функцій системи. Усі дії та результати відновлення фіксуються у звіті, який використовується для аналізу та запобігання подібним ситуаціям у майбутньому.

Додатково передбачено інтеграцію з мобільними пристроями для адміністраторів та фармацевтів, що дозволяє оперативно контролювати залишки товарів та бронювання.

Висновок

Розробка та впровадження інформаційної системи аптечної мережі «Аптека плюс» підтвердила практичну реалізованість сформованих вимог і

концепції. У процесі реалізації було створено веб-застосунок із функціоналом для адміністраторів та клієнтів, що забезпечує структуроване зберігання даних, автоматизацію обліку та зручний інтерфейс. Система дозволяє оперативно оновлювати асортимент, формувати замовлення, контролювати їх статус і забезпечує інтеграцію з внутрішніми бізнес-процесами. Таким чином, реалізація проєкта довела ефективність використання сучасних технологій для оптимізації роботи аптечної мережі та підвищення рівня обслуговування клієнтів.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ

Під охороною праці в аптеці розуміють систему заходів, що гарантує безпечні умови роботи фармацевтів та інших співробітників, які щодня взаємодіють із лікарськими засобами, комп'ютерною технікою та клієнтами. Основна мета цієї системи – мінімізувати ризик травмувань, професійних захворювань та забезпечити стабільне функціонування аптечної мережі.

У процесі роботи працівники стикаються з різними факторами, які можуть впливати на їхнє здоров'я. До небезпечних факторів належать ті, що здатні спричинити травму або різке погіршення стану, а до шкідливих – ті, що поступово знижують працездатність чи викликають захворювання. У фармацевтичній сфері вони можуть бути фізичними (недостатнє освітлення торгового залу, шум від обладнання, неправильний мікроклімат), хімічними (контакт із лікарськими препаратами, дезінфікуючими засобами, побутовою хімією), біологічними (бактерії, грибки, мікроорганізми, що можуть потрапляти у приміщення ззовні або з препаратів), а також психофізіологічними (емоційне навантаження при роботі з клієнтами, високий темп роботи, тривале перебування у статичній позі).

Техніка безпеки в аптеці включає правила та умови роботи, що гарантують захист персоналу. Це правильна організація робочих місць, використання засобів індивідуального захисту (рукавички, маски), дотримання санітарних норм при роботі з медикаментами, а також безпечне використання комп'ютерної техніки та електрообладнання.

Законодавчо, охорона праці регулюється Конституцією України та Законом про охорону праці. Статтею 43 Конституції України зазначено: «Кожен має право на належні, безпечні та здорові умови праці, на заробітну плату, не нижчу від визначеної законом».

Законом «Про охорону праці» визначаються основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у

процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні [13].

Для аптек це означає обов'язкове проведення інструктажів для співробітників: вступного – для нових працівників, первинного – при зміні посади чи обов'язків, повторного – для оновлення знань, позапланового – при впровадженні нового обладнання чи технологій, а також цільового – у випадку аварійних ситуацій чи надзвичайних подій.

У процесі роботи фармацевти та інші співробітники аптеки стикаються з низкою виробничих факторів, які можуть негативно впливати на їхнє здоров'я та працездатність. До таких факторів належать фізичні (освітлення, шум, мікроклімат), хімічні (контакт із лікарськими препаратами, дезінфікуючими засобами), біологічні (мікроорганізми, що можуть потрапляти у приміщення разом із клієнтами чи препаратами), а також психофізіологічні (емоційне навантаження, високий темп роботи, тривале перебування у статичній позі).

Для зменшення їхнього впливу необхідно впроваджувати комплекс заходів. Передусім слід забезпечити оптимальні умови праці: правильне освітлення торгового залу та робочих місць, підтримання комфортної температури та вологості, регулярне провітрювання приміщень. Важливим є використання джерел безперебійного живлення та якісного електрообладнання, що знижує ризик ураження електричним струмом.

З огляду на специфіку аптечної діяльності особливу увагу потрібно приділяти хімічним ризикам. Працівники повинні користуватися засобами індивідуального захисту – рукавичками, масками, захисними окулярами – під час роботи з препаратами, що мають токсичні властивості або можуть подразнювати шкіру та слизові оболонки. Зберігання медикаментів має здійснюватися

відповідно до санітарних норм, із чітким розподілом за групами та умовами температури.

Для зниження біологічних ризиків необхідно забезпечити регулярну дезінфекцію приміщень, використання бактерицидних ламп у складських зонах та суворе дотримання правил особистої гігієни персоналом.

Психофізіологічні фактори зменшуються завдяки раціональній організації робочого процесу: чергуванню завдань, запровадженню перерв для відпочинку, створенню комфортних умов для спілкування з клієнтами. Важливим є також проведення інструктажів та навчальних курсів з охорони праці, що підвищують рівень обізнаності співробітників та формують культуру безпечної роботи.

Незважаючи на всі заходи, повністю усунути вплив шкідливих факторів неможливо, тому керівництво аптек повинно дотримуватися пріоритетів: максимально знижувати ризики, застосовувати безпечні методи роботи, контролювати джерела небезпеки та забезпечувати персонал засобами індивідуального захисту.

Висновок

Аналіз питань охорони праці та техніки безпеки показав, що створення та експлуатація інформаційної системи потребує дотримання нормативних вимог щодо організації робочих місць, електробезпеки, ергономіки та захисту персональних даних. Впровадження системи супроводжується мінімальними ризиками для здоров'я працівників, оскільки робота здійснюється у стандартних умовах використання комп'ютерної техніки. Основна увага приділяється забезпеченню безпечного середовища праці, профілактиці професійних захворювань та створенню комфортних умов для користувачів. Це дозволяє не лише гарантувати безпеку персоналу, але й підвищити продуктивність роботи та якість обслуговування клієнтів.

У даній дипломній роботі розроблено концепцію інформаційної системи аптеки «Аптека плюс». Під час виконання кваліфікаційної роботи було здійснено комплексний аналіз предметної області, сформовано вимоги до інформаційної системи та розроблено її концепцію. Реалізація проєкта підтвердила ефективність застосування сучасних інформаційних технологій для оптимізації діяльності аптеки «Аптека плюс». Створений веб-застосунок забезпечує інтеграцію внутрішніх бізнес-процесів із сервісами для клієнтів, що дозволяє підвищити якість обслуговування, зменшити витрати часу та забезпечити прозорість взаємодії між адміністраторами й покупцями. Особливу увагу приділено питанням охорони праці та техніки безпеки, що гарантує безпечні умови роботи персоналу та відповідність нормативним вимогам. Це створює підґрунтя для стабільної експлуатації системи та її подальшого розвитку. Таким чином, дипломна робота довела актуальність та практичну значущість впровадження інформаційних систем у фармацевтичній сфері. Розроблена система не лише вирішує існуючі проблеми організації аптечного бізнесу, але й відкриває нові можливості для цифрової трансформації галузі охорони здоров'я в Україні.

1. Статистика середньої тривалості життя на момент 1990 року [Електронний ресурс]: <https://nv.ua/ukr/opinion/den-nezalezhnosti-yak-zminilasya-medicina-za-30-rokiv-novini-ukrajini-50177925.html>
2. Статистика середньої тривалості життя станом на 2025-2026 років [Електронний ресурс]: <https://tsn.ua/other/serednia-tryvalist-zyttia-v-ukrayini-rizko-znyzylas-shcho-hovoriat-statystyka-eksperty-2880671.html>
3. Про ліки [Електронний ресурс]: Фармацевтична енциклопедія: <https://www.pharmencyclopedia.com.ua/article/2082/liki>
4. Про рентабельність аптечного бізнесу [Електронний ресурс]: <https://regserv.biz/yak-vidkriti-apteku/>
5. Статистика споживання аптечних виробів [Електронний ресурс]: <https://gradus.app/uk/open-reports/consumer-trends-part-1-pharma/>
6. «ТЕНДЕНЦІЇ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ СФЕРИ ОХОРОНИ ЗДОРОВ'Я» І.О. Іртищева, О. С. Іртищев, А.В. Гарагуля, Ю.В. Хігазі, В.В. Палій (с.4 – 6)
7. Клевцов С.І. Аналіз та формування вимог до програмного забезпечення інформаційних систем збору та обробки даних: навчальний посібник. Таганрог: Вид-во ТТІУФ, 2007. 100с.
8. The State of Java 2025 [Електронний ресурс]: <https://lp.jetbrains.com/the-state-of-java-2025>
9. Spring Framework [Електронний ресурс]: <https://spring.io>
10. Spring Security [Електронний ресурс] <https://spring.io/projects/spring-security>
11. Вступ до JavaScript[Електронний ресурс]: <https://uk.javascript.info/intro>
12. Hibernate(framework) [Електронний ресурс]: [https://en.wikipedia.org/wiki/Hibernate_\(framework\)](https://en.wikipedia.org/wiki/Hibernate_(framework))
13. Закон про «Охорону праці» [Електронний ресурс]: <https://zakon.rada.gov.ua/laws/show/2695-12#Text>

1. Вступ

1.1 Назва проекту – Інформаційна система мережі аптек "Аптека плюс"

1.2 Призначення – спростити роботу співробітників аптеки та надання зручних інструментів для покупців в умовах глобальної цифровізації

1.3 Область застосування – аптечні пункти «Аптека плюс»

2. Призначення розробки

Призначена для спрощення, прискорення та структурованості роботи аптечних пунктів. Розширення доступності закладів для покупців

3. Вимоги до системи

3.1 Вимоги до функціональних характеристик

Система буде містити наступні функціональні характеристики:

- Управління інформацією про ліки, користувачів та пунктів видачі товарів
- Синхронізація даних та змін в реальному часі

3.2 Вимоги надійності

3.2.1 Вимоги для забезпечення надійності функціонування програми

Надійне (стійке) функціонування системи має бути забезпечено надійним (стійким) функціонуванням бездротової мережі і мережі електропостачання в аптечному закладі, або у іншому місці, де буде розміщений сервер.

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача.

- Перевірка формату вводу даних для коректної роботи.
- Функція автоматичного backup кожен день, тиждень, місяць, рік.

3.3 Умови експлуатації

Данною програмою будуть користуватися працівники, які є впевненими користувачами комп'ютера, та не потребують додаткового навчання по даному ПЗ, але можуть вимагати додаткову допомогу у роботі з вихідними даними.

3.4 Вимоги до інформаційної та програмної сумісності

3.4.1 Вимоги до інформаційних структур і методів розв'язання

Вимоги до інформаційних структур і методів розв'язання відсутні.

3.4.2 Вимоги до вихідних кодів і мов програмування

Додаткові вимоги не відсутні.

3.4.3 Вимоги до програмних засобів , які використовуються програмою

Програмні засоби що використовуються мають бути вільно поширювані

3.5 Вимоги до маркування та упаковки

3.5.1 Вимоги до маркування :

- Читабельність тексту, чіткість ілюстрацій;
- Достовірність інформації про систему

3.5.2 Вимоги до упаковки:

Вимоги до упаковки відсутні.

3.6 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутні.

3.7 Спеціальні вимоги

Спеціальні вимоги відсутні.

4. Вимоги до програмної документації

4.1 Попередній склад програмної документації

Попередній склад програмної документації повинен включати в себе технічне завдання;

4.2 Склад програмної документації

Програмна документація складається з наступних програмних документів:

- Технічне завдання ;
- Опис системи ;

- Код програми ;
- Керівництво користувача.

Сертифікація містить список всіх програмних документів .

Технічне завдання містить опис завдання роботи, склад функціональних характеристик програми, опис вимог для розробки.

Опис програми містить опис логічної структури програми, форматів вхідних та вихідних даних, використовуваних технічних засобів.

Код програми містить опис блоків програми і запис блоків в програмному коді.

Керівництво користувача містить інформацію , необхідну для ознайомлення користувача з програмним забезпеченням.

5. Техніко-економічні показники

5.1 Економічні переваги розробки

Орієнтовна економічна ефективність не розраховується.

6. Стадії та етапи розробки програмного забезпечення

6.1 Стадії розробки

Розробка повинна бути проведена в 5 стадій:

1. Технічне завдання
2. Концепція проекту
3. Технічний проект
4. Робочий проект
5. Впровадження

Об'єкт випробувань

Назва об'єкта випробувань: інформаційна система обліку та продажу ліків аптечних пунктів, надалі називається «Програма». Програма дозволяє користувачеві пройти автентифікацію та отримати доступ до відповідного модуля.

Ціль випробувань

Перевірка придатності програми до використання за призначенням, її відповідність заданим вимогам і програмній документації. Перевірка реально досягнутих результатів програми.

Вимоги до програми

Програма повинна задовольняти наступним функціональним характеристикам:

- 1) перевірка правильності імені та паролю користувачів;
- 2) перевірка правильності даних при додаванні нового користувача в систему.

Адміністратор повинен вказати ім'я та пароль, ввести дані про користувача.

Вимоги до програмної документації

У комплект програмної документації повинні входити наступні документи:

- Загальний опис системи.
- Посібник користувача – адміністратора.
- Інструкція з формування і ведення бази даних.

- Програма і методика випробувань.
- Текст програми.

Засоби і порядок випробувань

Вимоги до складу і параметрів технічних засобів:

Сервер на базі i586_64 або x86.

Вимоги до складу і параметрів програмних засобів :

- 1) операційна система Arch Linux;
- 2) встановлене з'єднання з сервером баз даних SQL.

Група людей, що будуть проводити випробування, повинна складатися з автора програми (розробника) і представника і/або самого замовника. При наявності відповідної можливості, ця група може бути розширена програмістом, спеціалістом-тестувальником програмного забезпечення і фахівцем-проектувальником, які не брали участь у розробці програми.

Попередньо (не пізніше 3 днів) до початку випробувань уточнюються кінцевий склад групи, що буде проводити випробування. Кожен член групи одержує необхідну документацію до програми (список вимог до програми, опис програми, керівництво користувача, програму і методику випробувань).

Порядок проведення випробувань:

- послідовно виконуються контрольні тести; результати заносяться у відповідний журнал;
- виробляється аналіз отриманих результатів і встановлюється відповідність програми заданим вимогам і програмним документам.

Методи випробувань

Робота програми залежить від параметрів, що вводяться користувачем: від імені та паролю користувача, а також параметрів, що відповідають інформації про покупця (ім'я, прізвище, нікнейм, електрона пошта та інші). Адміністратор також має слідкувати за правильністю найменувань та додаткових даних ліків

Іспити будуть проводитися методом «чорної шухляди».

Тести будуть створюватися на основі класів еквівалентності.

Класи еквівалентності виділяються шляхом вибору кожної вхідної умови і розбивкою його на двох або більш груп. Розрізняють два типи класів еквівалентності: правильні класи еквівалентності, що представляють правильні вхідні дані програми, і неправильні класи еквівалентності, що представляють всі інші можливі стани умов (помилкові вхідні значення). При цьому існує ряд правил:

- один тест може покривати тільки один неправильний клас еквівалентності;
- один тест повинний покривати як можна більше число непокритих правильних класів еквівалентності.

Виділені класи еквівалентності для складання тестів мають унікальні номери і приводяться в таблиці Б.1.

Таблиця Б.1 – Класи еквівалентності

Операція, що тестується	Вихідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Автентифікація користувача за ім'ям та паролем	Ім'я та пароль правильний	Так (1)	Ні (2)

Збереження даних про покупця	Коректні дані про покупця	Так (3)	Ні (4)
------------------------------------	------------------------------	---------	--------

Тести

Тести складені на основі класів еквівалентності таблиці Б.1. Тести використовують наступні початкові умови:

1) ім'я користувача – user , його пароль – password;

2) коректні дані про учня: ім'я – Петро, прізвище – Бандарюк, ім'я користувача – a petrik, електронна пошта: shev@gmail.com, номер телефону: +380939393939.

Тест 1

Класи еквівалентності:

1

Вхідні дані:

Ім'я користувача: user

Пароль користувача: password

Результат:

Ім'я та пароль користувача знайдено. Надано доступ до інтерфейсу користувача.

Тест 2

Класи еквівалентності:

2

Вхідні дані:

Ім'я користувача: usr

Пароль користувача: passwd

Результати:

Ім'я та пароль користувача не знайдено. У доступі до інтерфейсу користувача відмовлено з повідомленням про помилку введення імені та паролю.

Тест 3

Класи еквівалентності :

3

Вхідні дані:

Ім'я: Петро

Прізвище: Бандарюк

Ім'я користувача – petrik

Електронна пошта: shev@gmail.com

Номер телефону: +380939393939.

Результати:

Дані перевірено на коректність та збережено. Видане повідомлення про успішне збереження даних про учня.

Тест 4

Класи еквівалентності:

4

Вхідні дані:

Ім'я: Петро

Прізвище: Бандарчук

Ім'я користувача – petrikov

Електронна пошта: shev@gmail.com

Номер телефону: +380.

Результати:

Дані перевірено на коректність та не збережено. Видане повідомлення про помилку в даних про користувача, некоректність прізвища, ім'я користувача та номеру телефону.

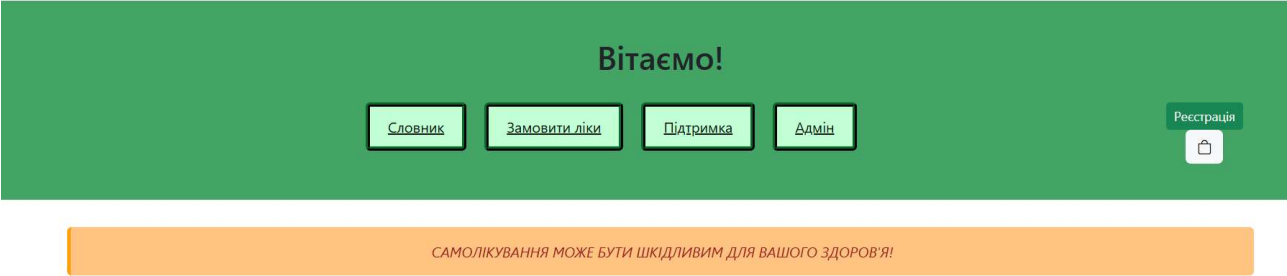


Рисунок В1 – Головна сторінка веб-застосунку

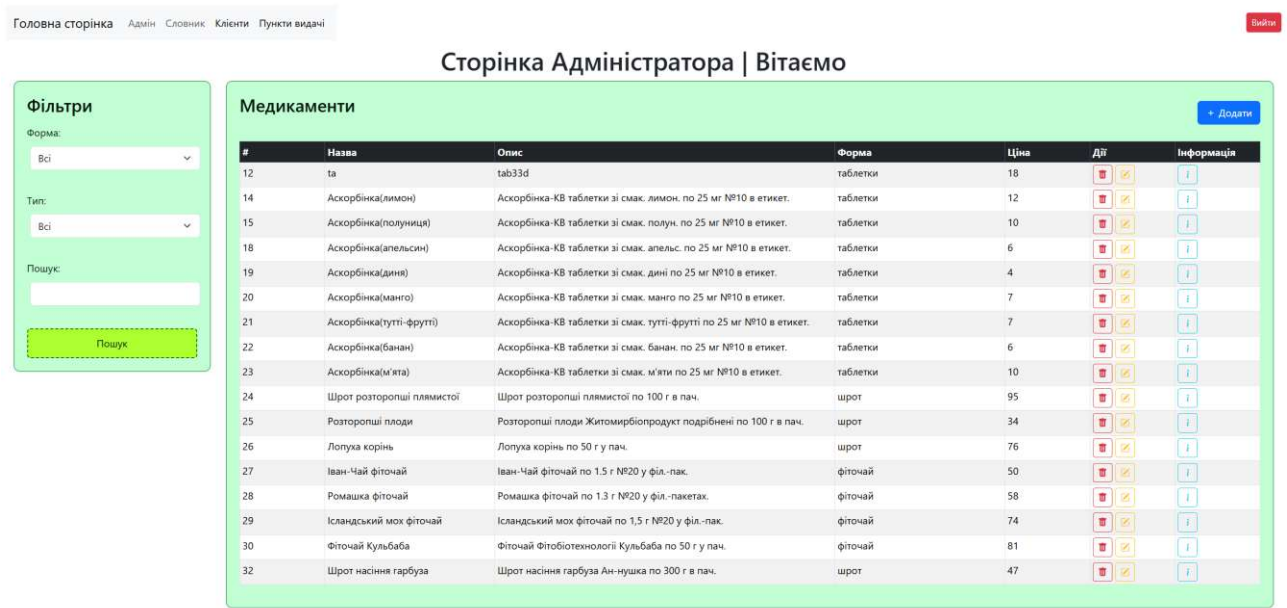


Рисунок В2 – Сторінка керування ліками



Рисунок В3 – Сторінка керування клієнтами

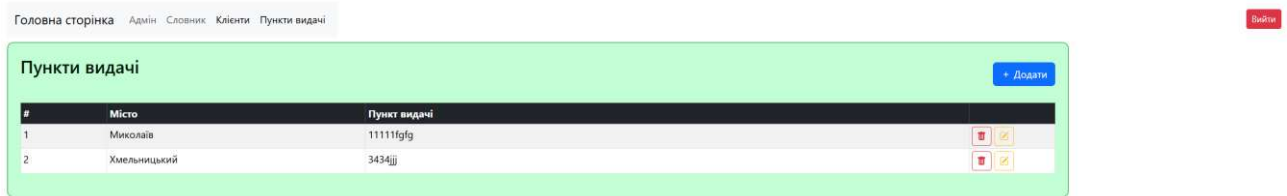


Рисунок В4 – Сторінка керування пунктами видачі

Логін

Пароль

Підтвердить Пароль

Ім'я

Фамілія

Рисунок В5 – Форма реєстрації нового користувача

Логін

Логін

Пароль

Рисунок В6 – Форма для входу у систему

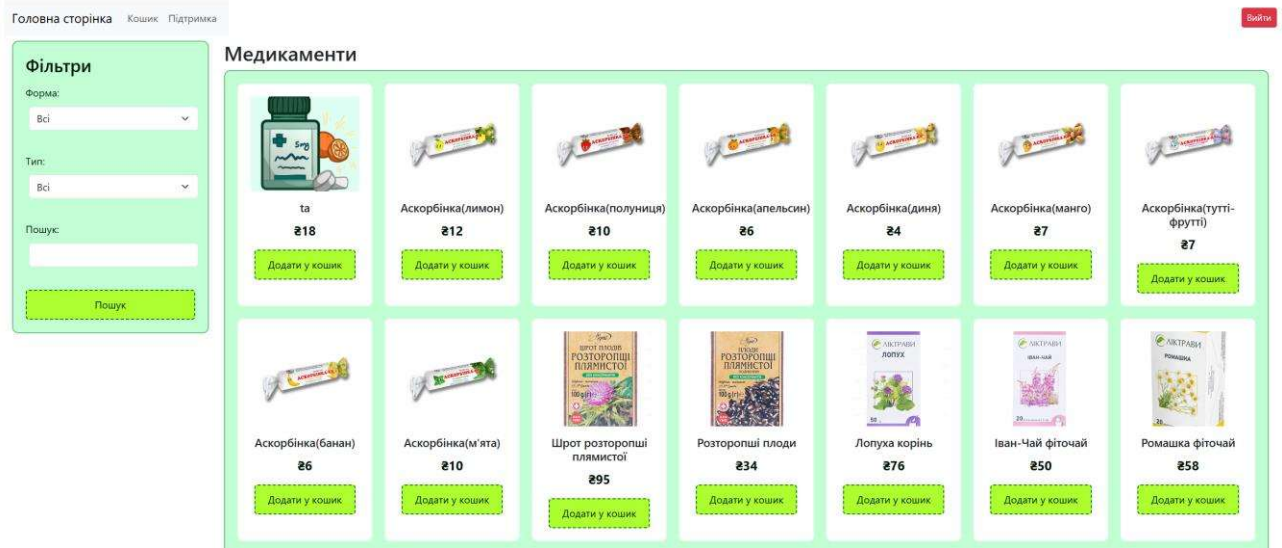


Рисунок В7 – Сторінка замовлення ліків



Рисунок В8 – Форма редагування кошика

Чек для: Іусу
Сума: 243

Аскорбінка(лимон)	таблетки 12	2
Аскорбінка(полуниця)	таблетки 10	2
Аскорбінка(лимон)	таблетки 12	1
Аскорбінка(апельсин)	таблетки 6	1
Лопуха корінь	шрот 76	1
Аскорбінка(м'ята)	таблетки 10	1
Шрот розторопші плямистої	шрот 95	1

Підтвердити

Рисунок В9 – Форма чеку

ДОДАТОК Г – КОД ПРОГРАМИ ТА КОД СТВОРЕННЯ ТАБЛИЦЬ БАЗИ ДАНИХ

Створення таблиці Client:

```
package org.example.grandproject.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@Entity
@NoArgsConstructor
@Table(name = "client")
public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Integer id;

    @Column(name = "username", nullable = false, length = 50)
    private String username;

    @Column(name = "name", nullable = false, length = 50)
    private String name;

    @Column(name = "last_name", nullable = false, length = 50)
    private String lastName;

    @Column(name = "password", nullable = false, length = 50)
    private String password;

    @Column(name = "total", nullable = false)
    private Integer total;

    @Column(name = "bonus", nullable = false)
    private Integer bonus;

    @Column(name = "phone_number", nullable = false, length = 20)
    private String phoneNumber;

    public Client(String username, String name, String lastName,
String password, Integer total, Integer bonus, String phoneNumber)
    {
        this.username = username;
        this.name = name;
        this.lastName = lastName;
        this.password = password;
        this.total = total;
        this.bonus = bonus;
        this.phoneNumber = phoneNumber;
    }
}
```

```
}  
}
```

Створення таблиці User:

```
package org.example.grandproject.entity;  
  
import jakarta.persistence.*;  
import lombok.*;  
import org.hibernate.proxy.HibernateProxy;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
@Builder  
@NoArgsConstructor  
@AllArgsConstructor  
@Entity  
@Getter  
@Setter  
@Table(name = "users_table")  
public class User {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(nullable = false)  
    private Long id;  
  
    @Column(unique = true, nullable = false)  
    private String username;  
  
    @Column(nullable = false)  
    private String name;  
  
    @Column(nullable = false)  
    private String lastname;  
  
    @Column(nullable = false)  
    private String password;  
  
    @Column(nullable = false)  
    private String role;  
  
    private boolean enabled;  
  
    @Transient  
    private String confirm;  
  
    @OneToMany(mappedBy = "user", orphanRemoval = true)  
    private List<ShoppingCart> shoppingCart = new ArrayList<>();  
  
    public User(String username,  
                String password,  
                String confirm,
```

```

        String name,
        String lastname) {
    this.username = username;
    this.password = password;
    this.role = "ROLE_USER";
    this.enabled = true;
    this.lastname = lastname;
    this.name = name;
    this.confirm = confirm;
}

@Override
public final boolean equals(Object o) {
    if (this == o) return true;
    if (o == null) return false;
    Class<?> oEffectiveClass = o instanceof HibernateProxy ?
((HibernateProxy)
o).getHibernateLazyInitializer().getPersistentClass() :
o.getClass();
    Class<?> thisEffectiveClass = this instanceof
HibernateProxy ? ((HibernateProxy)
this).getHibernateLazyInitializer().getPersistentClass() :
this.getClass();
    if (thisEffectiveClass != oEffectiveClass) return false;
    User user = (User) o;
    return getId() != null && Objects.equals(getId(),
user.getId());
}

@Override
public final int hashCode() {
    return this instanceof HibernateProxy ? ((HibernateProxy)
this).getHibernateLazyInitializer().getPersistentClass().hashCode(
) : getClass().hashCode();
}
}

```

Створення таблиці Med:

```

package org.example.grandproject.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.proxy.HibernateProxy;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

@Getter
@Setter

```

```

@Entity
@NoArgsConstructor
@Table(name = "meds")
public class Med {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @Column(name = "name", nullable = false, length = 50)
    private String medName;

    @Column(name = "description", nullable = false, length = 200)
    private String medDescription;

    @Column(name = "price", nullable = false)
    private Integer medPrice;

    @ManyToMany
    @JoinTable(name = "meds_medsTypes",
        joinColumns = @JoinColumn(name = "med_id"),
        inverseJoinColumns = @JoinColumn(name =
"medsTypes_id"))
    private List<MedsType> medsTypes = new ArrayList<>();

    @ManyToOne
    @JoinColumn(name = "meds_form_id")
    private MedsForm medsForm;

    @OneToOne(orphanRemoval = true)
    @JoinColumn(name = "id")
    private Description longDescription;

    public Med(String medName, String description, Integer price,
MedsForm medsForm) {
        this.medName = medName;
        this.medDescription = description;
        this.medPrice = price;
        this.medsForm = medsForm;
    }

    @Override
    public String toString() {
        return medName;
    }

    @Override
    public final boolean equals(Object o) {
        if (this == o) return true;
        if (o == null) return false;
        Class<?> oEffectiveClass = o instanceof HibernateProxy ?
((HibernateProxy)
o).getHibernateLazyInitializer().getPersistentClass() :
o.getClass();

```



```

        Class<?> thisEffectiveClass = this instanceof
HibernateProxy ? ((HibernateProxy)
this).getHibernateLazyInitializer().getPersistentClass() :
this.getClass();
        if (thisEffectiveClass != oEffectiveClass) return false;
        Med med = (Med) o;
        return getId() != null && Objects.equals(getId(),
med.getId());
    }

    @Override
    public final int hashCode() {
        return this instanceof HibernateProxy ? ((HibernateProxy)
this).getHibernateLazyInitializer().getPersistentClass().hashCode(
) : getClass().hashCode();
    }
}

```

Створення таблиці MedsFrom:

```

package org.example.grandproject.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.ArrayList;
import java.util.List;

@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name = "meds_form")
public class MedsForm {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @Column(name = "name", nullable = false, length = 50)
    private String name;

    @OneToMany(mappedBy = "medsForm", orphanRemoval = true)
    private List<Med> meds = new ArrayList<>();

    public MedsForm(String formName) {
        this.name = formName;
    }

    @Override
    public String toString() {

```

```

        return name;
    }
}

```

Створення таблиці MedsType:

```

package org.example.grandproject.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.proxy.HibernateProxy;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name = "meds_type")
public class MedsType {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @Column(name = "name", nullable = false, length = 50)
    private String name;

    @ManyToMany(mappedBy = "medsTypes")
    private List<Med> meds = new ArrayList<>();

    public MedsType(String typeName) {
        this.name = typeName;
    }

    @Override
    public String toString() {
        return name;
    }
}

```

Створення таблиці ShoppingCart:

```

package org.example.grandproject.entity;

import jakarta.persistence.*;
import lombok.Getter;

```

```

import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.ArrayList;
import java.util.List;

@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name = "shopping_cart")
public class ShoppingCart {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    private String status;

    @Column(name = "total_price")
    private Long totalPrice;

    @Column(name = "bonuses")
    private Integer bonuses;

    @OneToMany(mappedBy = "shoppingCart", orphanRemoval = true)
    private List<ShoppingItem> shoppingItems = new ArrayList<>();

    public ShoppingCart(User user) {
        this.user = user;
        status = "NEW";
    }
}

```

Створення таблиці ShoppingItem:

```

package org.example.grandproject.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "shopping_item")
public class ShoppingItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

    @Column(name = "id", nullable = false)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "med_id")
    private Med med;

    @Column(name = "count")
    private Long count;

    @Column(name = "med_price")
    private Integer medPrice;

    @ManyToOne
    @JoinColumn(name = "shopping_cart_id")
    private ShoppingCart shoppingCart;
}

```

Контролер AdminController:

```

package org.example.grandproject.controller;

import jakarta.persistence.criteria.CriteriaBuilder;
import jakarta.servlet.http.HttpSession;
import lombok.AllArgsConstructor;
import org.example.grandproject.entity.Med;
import org.example.grandproject.entity.ShoppingCart;
import org.example.grandproject.repository.MedsFormRepository;
import org.example.grandproject.service.*;
/*import org.springframework.security.core.Authentication;
import
org.springframework.security.core.context.SecurityContextHolder;*/
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import
org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.List;
import java.util.Optional;

@Controller
@RequestMapping("/admin")
public class AdminController {

    private final MedsService medsService;
    private final FormService formService;
    private final TypeService typeService;
    private final DescriptionService descriptionService;
    private final ClientService clientService;
    private final DeliveryService deliveryService;

```

```

private final ShoppingCartService shoppingCartService;

@GetMapping("")
public String admin(Model model, HttpSession session) {
    /*Authentication auth =
SecurityContextHolder.getContext().getAuthentication();*/
    /*System.out.println(auth.getPrincipal());*/
    model.addAttribute("meds", medsService.findAll());
    if (session.getAttribute("name") != null) {
        model.addAttribute("username",
session.getAttribute("name"));
    }else{
        model.addAttribute("username", "");
    }
    return showAdminPage(model);
}

private String showAdminPage(Model model) {
    model.addAttribute("medForms", formService.findAll());
    model.addAttribute("medTypes", typeService.findAll());
    return "admin";
}

@PostMapping("/client/add")
public String addClient(@RequestParam("client_name") String
name,
                        @RequestParam("client_lastname")
String lastName,
                        @RequestParam("client_nickname")
String nickname,
                        @RequestParam("client_phonenumber")
String phoneNumber,
                        @RequestParam("client_password")
String password,
                        @RequestParam("client_total") Integer
total,
                        @RequestParam("client_bonus") Integer
bonus/*,
                        @RequestParam("client_id") Integer
id*/) {
    clientService.addClient(name, lastName, nickname,
phoneNumber, password, total, bonus);
    return "redirect:/admin/clients";
}

@PostMapping("/client/edit")
public String editClient(@RequestParam("client_name") String
name,
                        @RequestParam("client_lastname")
String lastName,
                        @RequestParam("client_nickname")
String nickname,
                        @RequestParam("client_phonenumber")

```

```

String phoneNumber,
                                @RequestParam("client_password")
String password,
                                @RequestParam("client_total") Integer
total,
                                @RequestParam("client_bonus") Integer
bonus,
                                @RequestParam("client_id") Integer
clientId) {

    clientService.updateClientData(name, lastName, nickname, phoneNumber,
password, total, bonus, clientId);
    return "redirect:/admin/clients";
}

@GetMapping("/client/delete/{id}")
public String deleteClient(@PathVariable("id") Integer id) {
    clientService.deleteClient(id);
    return "redirect:/admin/clients";
}

@PostMapping("/med/add")
public String addMeds(@RequestParam("med_name") String
medName,
                                @RequestParam("description_name") String
description,
                                @RequestParam("price_name") Integer
price,
                                @RequestParam("form_id") Long formId) {
    medsService.addMed(medName, description, price, formId );
    return "redirect:/admin";
}

@GetMapping("/med/delete/{id}")
public String deleteMeds(@PathVariable("id") Long id) {
    medsService.deleteMedById(id);
    return "redirect:/admin";
}

@GetMapping("/type/delete/{id}")
public String deleteType(@PathVariable("id") Long id) {
    typeService.deleteTypeById(id);
    return "redirect:/admin/dictionaries";
}

@GetMapping("/form/delete/{id}")
public String deleteForm(@PathVariable("id") Long id) {
    formService.deleteFormById(id);
    return "redirect:/admin/dictionaries";
}

@PostMapping("/med/edit")
public String editMeds(@RequestParam("med_name") String
medName,

```

```

        @RequestParam("description_name")
String description,
        @RequestParam("price_name") Integer
price,
        @RequestParam("form_id") Long formId,
        @RequestParam("med_id") Long id) {
    medsService.updateMedData(medName, description, price, formId,
id);
    return "redirect:/admin";
}

@PostMapping("/type/edit")
public String editType(@RequestParam("type_name") String
typeName,
        @RequestParam("type_id") Long typeId) {
    typeService.updateTypeData(typeName, typeId);
    return "redirect:/admin/dictionaries";
}

@PostMapping("/form/edit")
public String editForm(@RequestParam("form_name") String
formName,
        @RequestParam("form_id") Long formId) {
    formService.updateFormData(formName, formId);
    return "redirect:/admin/dictionaries";
}

@PostMapping("/type/add")
public String addType(@RequestParam("type_name") String
typeName) {
    typeService.addType(typeName);
    return "redirect:/admin/dictionaries";
}

@PostMapping("/form/add")
public String addForm(@RequestParam("form_name") String
formName) {
    formService.addForm(formName);
    return "redirect:/admin/dictionaries";
}

@GetMapping("/dictionaries")
public String dictionaries(Model model) {
    model.addAttribute("medsForms", formService.findAll());
    model.addAttribute("medsTypes", typeService.findAll());
    return "admin_dictionaries";
}

@GetMapping("/med/med_info/{id}")
public String medInfo(@PathVariable("id") Long id, Model
model) {
    Optional<Med> optionalMed = medsService.findById(id);
    if (optionalMed.isPresent()) {

```

```

        model.addAttribute("med", optionalMed.get());
        model.addAttribute("types_available",
medsService.findAvailableTypes(id));
        model.addAttribute("description",
descriptionService.findById(id));
        return "admin_med_info";
    }else {
        return "redirect:/admin";
    }
}

@GetMapping("/med/type/remove/{mId}/{tId}")
public String removeMedType(@PathVariable("mId") Long mId,
@PathVariable("tId") Long tId) {
    medsService.removeType(mId,tId);
    return "redirect:/admin/med/med_info/"+ mId;
}

@PostMapping("/med/type/add")
public String addMedType(@RequestParam("med_id") Long mId,
@RequestParam("type_id") Long tId){
    medsService.addType(mId, tId);
    return "redirect:/admin/med/med_info/"+ mId;
}

@PostMapping("/med/description/save")
public String saveDescription(@RequestParam("description")
String description,

@RequestParam("description_id")Long descriptionId) {
    descriptionService.save(description,descriptionId);
    return "redirect:/admin/med/med_info/"+ descriptionId;
}

@GetMapping("/med/description/remove/{mId}")
public String removeDescription(@PathVariable("mId") Long
mId){
    descriptionService.clear(mId);
    return "redirect:/admin/med/med_info/"+ mId;
}

@RequestMapping("/filter")
public String filter(Model model,
                    @RequestParam(required = false, name =
"search") String filter,
                    @RequestParam("formFilter") Long formId,
                    @RequestParam("typeFilter") Long typeId)
{
    /*model.addAttribute("meds",
medsService.findByMedNameContainsIgnoreCase(filter));*/
    model.addAttribute("meds",
medsService.findByMedNameContainsIgnoreCaseInFormWithType(filter,f
ormId,typeId));
    return showAdminPage(model);
}

```



```

    }

    @GetMapping("/clients")
    public String clients(Model model) {
        model.addAttribute("clients", clientService.findAll());
        return "clients";
    }

    @GetMapping("/delivery_page")
    public String deliveryPage(Model model) {
        model.addAttribute("delivery", deliveryService.findAll());
        model.addAttribute("city",
deliveryService.findAllCities());
        return "delivery_page";
    }

    //Delivery action controls

    @GetMapping("/delivery/delete/{id}")
    public String deleteDelivery(@PathVariable("id") Long id) {
        deliveryService.deleteDelivery(id);
        return "redirect:/admin/delivery_page";
    }

    @PostMapping("/delivery/add")
    public String addDelivery(@RequestParam("city_id") Long
cityId,
                                @RequestParam("pick_up_point")
String pickUpPoint) {
        deliveryService.addDelivery(cityId,pickUpPoint);
        return "redirect:/admin/delivery_page";
    }

    @PostMapping("/delivery/edit")
    public String editDelivery(@RequestParam("city_id") Long
cityId,
                                @RequestParam("pick_up_point")
String pickUpPoint,
                                @RequestParam("delivery_id") Long
deliveryId) {
        deliveryService.updateDeliveryInfo(deliveryId,
pickUpPoint, cityId);
        return "redirect:/admin/delivery_page";
    }

    @GetMapping("/orders/in_progress")
    public String ordersInProgress(Model model) {
        List<ShoppingCart> cartList =
shoppingCartService.findInProgress();
        model.addAttribute("cartList", cartList);
        return "orders_admin";
    }

    @GetMapping("/orders/more_info")

```

```

        public String checkout(@RequestParam("id") Long cartId,
                               Model model) {
            ShoppingCart cart =
shoppingCartService.showAction(cartId);
            model.addAttribute("cart", cart);
            return "more_info";
        }

        @PostMapping("/orders/complete")
        public String complete(@RequestParam("id") Long cartId){
            shoppingCartService.completeOrder(cartId);
            return "redirect:/admin/orders/in_progress";
        }
    }
}

```

Контролер OrdersController:

```

package org.example.grandproject.controller;

import jakarta.servlet.http.HttpSession;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.example.grandproject.entity.ShoppingCart;
import org.example.grandproject.entity.ShoppingItem;
import org.example.grandproject.entity.User;
import org.example.grandproject.service.*;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Slf4j
@Controller
@RequestMapping("/orders")
public class OrdersController {

    private final FormService formService;
    private final TypeService typeService;
    private final MedsService medsService;
    private final ShoppingCartService shoppingCartService;
    private final UserService userService;

    @GetMapping("")
    public String orders(Model model) {

```

```

        model.addAttribute("medForms", formService.findAll());
        model.addAttribute("medTypes", typeService.findAll());
        model.addAttribute("meds", medsService.findAll());
        model.addAttribute("form_id", 0);
        model.addAttribute("type_id", 0);
        return "orders";
    }

    @RequestMapping("/client/filter")
    public String filter(Model model,
        @RequestParam(required = false, name =
"search") String filter,
        @RequestParam("formFilter") Long formId,
        @RequestParam("typeFilter") Long typeId){
        /*System.out.println(filter);*/

        model.addAttribute("meds", medsService.findByMedNameContainsIgnoreCaseInFormWithType(filter, formId, typeId));
        model.addAttribute("medForms", formService.findAll());
        model.addAttribute("medTypes", typeService.findAll());
        model.addAttribute("form_id", formId);
        model.addAttribute("type_id", typeId);
        return "orders";
    }

    @PostMapping("/client/add_to_cart")
    public String addMedToCart(@RequestParam("med_id") Long medId,
        HttpSession session
        /*@RequestParam("filter") String
filter,
        @RequestParam("formFilter") Long
formId,
        @RequestParam("typeFilter") Long
typeId,*/
        ){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        UserDetails principal = (UserDetails)
authentication.getPrincipal();
        String username = principal.getUsername();
        Long userId =
userService.findByUsername(username).map(User::getId).orElse(0L);

        log.info("Adding med to cart");
        log.info("MedId: {}", medId);
        log.info("UserId: {}", session.getAttribute("userId"));
        session.setAttribute("user_id", userId);
        ShoppingCart cart =
shoppingCartService.findById(userId);
        ShoppingItem item = new ShoppingItem();
        item.setShoppingCart(cart);
        item.setMed(medsService.findById(medId).get());
        item.setCount(1L);
        shoppingCartService.saveItem(item);
    }

```

```

        //cart.getShoppingItems().add(item);
        return "redirect:/orders";
    }
}

```

Контролер RegisterController:

```

package org.example.grandproject.controller;

import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.example.grandproject.logic.RegisterRecord;
import org.example.grandproject.service.RegisterService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Slf4j
@Controller
@AllArgsConstructor
@RequestMapping("")
public class RegisterController {

    private final RegisterService registerService;

    @GetMapping("/register")
    public String register() {
        return "register_form";
    }

    @PostMapping("/register")
    public String register(@RequestParam("username") String
username,
                           @RequestParam("password") String
password,
                           @RequestParam("password2") String
confirm,
                           @RequestParam("name") String name,
                           @RequestParam("lastname") String
lastname,
                           Model model) {
        log.info("Registering user: {}", username);
        RegisterRecord register =
registerService.register(username,password,confirm,name,lastname);
        if (register.message().contains("successfully")) {
            return "redirect:/";
        }else {
            model.addAttribute("register",register);
            return "redirect:/register";
        }
    }
}

```

```

    }
}

```

Файл pom.xml, який містить інформацію про залежності проекту:

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>4.0.2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>org.example</groupId>
  <artifactId>grandProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>grandProject</name>
  <description>grandProject</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>21</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>

```

```

        <scope>runtime</scope>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.webjars.npm/bootstrap -->
    <dependency>
        <groupId>org.webjars.npm</groupId>
        <artifactId>bootstrap</artifactId>
        <version>5.3.8</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.webjars.npm/bootstrap-icons
-->
    <dependency>
        <groupId>org.webjars.npm</groupId>
        <artifactId>bootstrap-icons</artifactId>
        <version>1.13.1</version>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.thymeleaf.extras</groupId>
        <artifactId>thymeleaf-extras-
springsecurity6</artifactId>
    </dependency>

</dependencies>

<build>
    <plugins>

```

```

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <annotationProcessorPaths>
          <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </path>
        </annotationProcessorPaths>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>

```