Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавра
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення та тестування CRM-системи для аптеки з прогнозуванням продажів на основі моделей часових рядів

Виконав: студент IV курсу, групи СП-42
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Костецький О. В.
(підпис) (прізвище та ініціали)

Керівник _____ Цебрій О. Р.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ _____
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль 2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Костецькому Олександру Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення та тестування CRM-системи для аптеки з прогнозуванням продажів на основі моделей часових рядів

Керівник роботи Цебрій Олексій Романович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____.

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Реалізація моделей часових рядів на основі бібліотеки statsforecast (AutoARIMA, SARIMA, AutoETS, Theta, Croston); стек Python 3.10 / FastAPI / SQLModel / PostgreSQL 15 / React 19 / Docker Compose; репозиторій розробленої програмної системи “Mercurio CRM” на платформі GitHub.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області CRM-систем для аптек, огляд існуючих рішень, дослідження методів прогнозування попиту на основі часових рядів, постановка задачі. 2. Проєктування мікросервісної архітектури, бази даних, UML-діаграм; реалізація серверної частини, модуля прогнозування (statsforecast) та клієнтського інтерфейсу. 3. Тестування програмної системи (pytest, Playwright), верифікація моделей прогнозування, розгортання та інструкція користувача. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Мариненко С.Ю.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення із завданням на кваліфікаційну роботу		
2	Аналіз предметної області, огляд CRM-рішень для аптек та методів прогнозування часових рядів		
3	Проєктування мікросервісної архітектури системи та схеми бази даних		
4	Реалізація серверної частини (FastAPI, SQLModel, PostgreSQL, JWT, FEFO-списання)		
5	Розробка модуля прогнозування продажів (statsforecast, 10 моделей, кешування)		
6	Реалізація клієнтської частини (React 19, TypeScript, Radix UI, Recharts)		
7	Оформлення розділу “Аналіз вимог до програмної системи”		
8	Оформлення розділу “Проєктування та розробка програмної системи”		
9	Проведення тестування (pytest, Playwright, coverage), верифікація моделей прогнозування		
10	Оформлення розділу “Тестування, впровадження та підтримка”		
11	Оформлення розділу “Безпека життєдіяльності, основи охорони праці”		
12	Оформлення повної кваліфікаційної роботи (вступ, висновки, анотація, перелік джерел)		
13	Нормоконтроль		
14	Перевірка на плагіат		
15	Попередній захист кваліфікаційної роботи		
16	Захист кваліфікаційної роботи		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення та тестування CRM-системи для аптеки з прогнозуванням продажів на основі моделей часових рядів // Кваліфікаційна робота освітнього рівня “Бакалавр” // Костецький Олександр Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // С. 90, рис. – 37, табл. – 11, кресл. – 0, додат. – 2, бібліогр. – 27.

Ключові слова: CRM-система, аптека, прогнозування продажів, часові ряди, тестування програмного забезпечення, ARIMA, машинне навчання.

Кваліфікаційна робота присвячена розробці та тестуванню CRM-системи для автоматизації діяльності аптеки з інтегрованим модулем прогнозування продажів на основі моделей часових рядів.

У першому розділі проаналізовано предметну область управління аптекою, здійснено огляд існуючих CRM-рішень та обґрунтовано застосування моделей часових рядів для прогнозування попиту.

У другому розділі спроєктовано мікросервісну архітектуру системи та базу даних, реалізовано серверну частину, модуль прогнозування (statsforecast) та клієнтський інтерфейс.

У третьому розділі проведено комплексне тестування системи, верифіковано точність моделей прогнозування та описано розгортання системи.

У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці при розробці програмного забезпечення.

Об’єкт дослідження: процеси управління продажами та клієнтською базою аптеки.

Предмет дослідження: програмне забезпечення CRM-системи для аптеки з модулем прогнозування продажів на основі моделей часових рядів.

ABSTRACT

Software Development and Testing of a CRM System for a Pharmacy with Sales Forecasting Based on Time Series Models // Bachelor's Qualification Thesis // Kostetskyi Oleksandr Vasylovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SP-42 // Ternopil, 2026 // P. 90, fig. – 37, tabl. – 11, draw. – 0, append. – 2, ref. – 27.

*Keywords: C*RM system, pharmacy, sales forecasting, time series, software testing, ARIMA, machine learning.

The qualification thesis is dedicated to the development and testing of a CRM system for automating pharmacy operations with an integrated sales forecasting module based on time series models.

The first section analyses the pharmacy management domain, reviews existing CRM solutions, and justifies the use of time series models for demand forecasting.

The second section covers the design of the microservice architecture and database, and the implementation of the backend, forecasting module (statsforecast), and client interface.

The third section presents comprehensive system testing, verification of forecasting model accuracy, and a description of system deployment.

The fourth section addresses occupational safety and health issues in the context of software development.

Object of research: sales management and customer relationship processes in a pharmacy.

Subject of the study: CRM system software for a pharmacy with a sales forecasting module based on time series models.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

API (Application Programming Interface) – інтерфейс програмування застосунків.

ARIMA (Autoregressive Integrated Moving Average) – авторегресійна інтегрована модель ковзного середнього.

CRM (Customer Relationship Management) – управління взаємовідносинами з клієнтами.

CRUD (Create, Read, Update, Delete) – базові операції з даними.

ER (Entity-Relationship) – “сутність-зв’язок”, тип діаграми БД.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту.

JSON (JavaScript Object Notation) – текстовий формат обміну даними.

LSTM (Long Short-Term Memory) – довга короткострокова пам’ять, архітектура нейронної мережі.

MAE (Mean Absolute Error) – середня абсолютна похибка.

MAPE (Mean Absolute Percentage Error) – середня абсолютна відсоткова похибка.

ML (Machine Learning) – машинне навчання.

MVC (Model-View-Controller) – патерн проєктування.

ORM (Object-Relational Mapping) – об’єктно-реляційне відображення.

REST (Representational State Transfer) – архітектурний стиль побудови API.

RMSE (Root Mean Square Error) – середньоквадратична похибка.

SARIMA (Seasonal ARIMA) – сезонна авторегресійна інтегрована модель ковзного середнього.

SPA (Single Page Application) – односторінковий веб-застосунок.

SQL (Structured Query Language) – мова структурованих запитів.

UML (Unified Modeling Language) – уніфікована мова моделювання.

UI (User Interface) – інтерфейс користувача.

UX (User Experience) – досвід взаємодії користувача з системою.

ЗМІСТ

ВСТУП

Сучасна аптека є не лише торговою точкою, а й важливою ланкою системи охорони здоров'я, що обслуговує широке коло пацієнтів із різноманітними медичними потребами. В умовах зростаючої конкуренції на фармацевтичному ринку України, розширення асортименту та посилення вимог регуляторів до обліку та звітності, ефективне управління аптекою стає неможливим без спеціалізованого програмного забезпечення.

Водночас існуючі програмні рішення для автоматизації аптек здебільшого зосереджені на обліково-касових функціях і не забезпечують управління взаємовідносинами з клієнтами та інструментів прогнозування попиту. Прийняття рішень щодо закупівлі товарів залишається значною мірою суб'єктивним, що призводить до надлишків одних препаратів і дефіциту інших.

Методи аналізу часових рядів – ARIMA, SARIMA, Prophet та моделі глибокого навчання – довели свою ефективність у задачах прогнозування роздрібного попиту в різних галузях. Інтеграція цих методів у систему аптеки дозволяє перевести планування закупівель на науково обґрунтований рівень.

Мета роботи – розробити та протестувати CRM-систему для аптеки з інтегрованим модулем прогнозування продажів на основі моделей часових рядів, що автоматизує ключові бізнес-процеси аптечної діяльності та підтримує прийняття управлінських рішень щодо управління товарними запасами.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) Провести аналіз предметної області та вимог до програмної системи, здійснити огляд існуючих рішень.

2) Спроєктувати архітектуру системи, розробити схему бази даних та UML-діаграми.

3) Реалізувати модулі обліку клієнтів, управління товарами та продажами.

4) Розробити та інтегрувати модуль прогнозування продажів на основі моделей часових рядів.

5) Провести комплексне тестування розробленого програмного забезпечення та верифікувати точність прогнозних моделей за метриками MAE та MAPE.

Об’єкт дослідження – процеси управління продажами, клієнтською базою та товарними запасами аптеки.

Предмет дослідження – програмне забезпечення CRM-системи для аптеки з модулем прогнозування продажів на основі моделей часових рядів.

Наукова новизна та практичне значення одержаних результатів. Наукова новизна роботи полягає у комплексному підході до інтеграції CRM-функціональності з модулем прогнозування попиту на фармацевтичні препарати, що дозволяє отримувати прогноз безпосередньо в контексті управління асортиментом без залучення зовнішніх аналітичних інструментів. Практичне значення роботи полягає у створенні програмного продукту, готового до впровадження в реально діючій аптеці: система скорочує час обслуговування клієнтів, підвищує точність планування закупівель та лояльність покупців завдяки програмі бонусів.

Основні положення та результати кваліфікаційної роботи було апробовано на IX Міжнародній студентській науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя “Природничі та гуманітарні науки. Актуальні питання” (Тернопіль, 2026).

За результатами виконання кваліфікаційної роботи опубліковано тезу: Костецький О. В. Застосування архітектур на основі механізму самоуваги для прогнозування часових рядів / О. В. Костецький ; наук. кер. О. Р. Цебрій // Природничі та гуманітарні науки. Актуальні питання : зб. тез IX Міжнар. студ. наук.-техн. конф. – Тернопіль : ТНТУ ім. Івана Пулюя, 2026, див. додаток А.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Даний розділ присвячено всебічному дослідженню предметної області CRM-систем для аптечної мережі та вимог до розроблюваного програмного забезпечення. Проаналізовано актуальні підходи до управління взаємовідносинами з клієнтами у фармацевтичній галузі, здійснено огляд існуючих рішень, визначено методи прогнозування попиту на основі часових рядів. На основі проведеного аналізу сформульовано постановку завдання, виявлено акторів системи та описано ключові варіанти її використання [1].

1.1 Аналіз предметної області

Фармацевтична роздрібна торгівля є однією з найбільш соціально значущих галузей, що водночас характеризується високою конкуренцією та жорсткою регуляторною базою. Сучасна аптека функціонує не лише як точка продажу медичних препаратів, а й як заклад надання консультативних послуг і моніторингу потреб пацієнтів. В умовах зростаючої кількості торгових позицій, сезонних коливань попиту та необхідності суворого дотримання строків придатності товарів, автоматизація бізнес-процесів стає критично важливою умовою ефективної діяльності.

CRM-система (Customer Relationship Management) – це програмний комплекс, призначений для управління взаємовідносинами з клієнтами, автоматизації продажів, маркетингу та аналітики [27]. У контексті аптечної діяльності CRM-система охоплює ширший спектр функцій: облік рецептурних і безрецептурних препаратів, управління залишками, ведення карток постійних покупців, формування програм лояльності та аналітику продажів у розрізі препаратів, груп, виробників і часових проміжків.

Ключовою особливістю аптечного ринку є нерівномірність попиту, зумовлена сезонністю захворювань, епідеміологічними ситуаціями, новинами в медіапросторі та змінами у стандартах лікування. Традиційні методи планування

закупівель, що базуються на інтуїції та досвіді провізора, виявляються недостатньо ефективними у нестабільних умовах. Натомість застосування математичних моделей часових рядів дозволяє формалізувати процес прогнозування та підвищити точність планування товарних запасів.

Часовий ряд – це послідовність значень деякого показника, зафіксованих через рівні проміжки часу [2]. В аптечній практиці такими показниками є обсяги продажів конкретних препаратів або фармацевтичних груп у розрізі днів, тижнів або місяців. Аналіз цих рядів дозволяє виявити тренди, сезонні компоненти та циклічні закономірності, що є основою для побудови прогнозних моделей.

Серед методів аналізу часових рядів, що застосовуються у завданнях прогнозування попиту, найбільшого поширення набули такі підходи:

– Модель ARIMA (Autoregressive Integrated Moving Average) – класична статистична модель, що враховує авторегресійну компоненту, інтегрування для досягнення стаціонарності та компоненту ковзного середнього.

– Модель SARIMA (Seasonal ARIMA) – розширення ARIMA з урахуванням сезонних складових, що особливо актуально для аптечної торгівлі.

– Метод Хольта-Уінтерса – модель експоненціального згладжування з адитивними або мультиплікативними сезонними компонентами.

– Алгоритм Prophet (розроблений Facebook/Meta) – гнучка адитивна модель для бізнес-часових рядів з автоматичним виявленням сезонності та свят.

– Методи машинного навчання (градієнтний бустинг, LSTM-мережі) – підходи, що дозволяють враховувати зовнішні фактори (метеорологічні умови, епідеміологічні показники, маркетингові акції) [3].

Поєднання CRM-функціональності з модулем прогнозування на основі часових рядів формує принципово новий клас систем – інтелектуальних аптечних платформ, здатних не лише фіксувати факти минулих продажів, а й активно підтримувати прийняття управлінських рішень щодо поповнення запасів, формування цінової політики та організації маркетингових заходів.

1.2 Огляд існуючих програмних рішень

На ринку програмного забезпечення для автоматизації аптечної діяльності існує ряд комерційних рішень, проте жодне з них не є повністю задовільним з огляду на поєднання CRM-функціональності з інтегрованим модулем прогнозування продажів. Для обґрунтування доцільності розробки власного рішення було проведено порівняльний аналіз наявних систем.

Програмний комплекс “IBS Аптека” є спеціалізованим рішенням для автоматизації аптек та аптечних мереж в Україні. Система забезпечує облік лікарських засобів, автоматизацію касових робочих місць, управління товарними запасами та формування звітності для регуляторних органів. Проте вбудованого CRM-модуля та інструментів прогнозування попиту система не надає – ці функції потребують підключення сторонніх рішень або ведення ручних розрахунків [14].

Програма “АНР-Аптека” орієнтована на комплексну автоматизацію роздрібних фармацевтичних підприємств і надає інструменти управління асортиментом, сегментації товарів та контролю залишків. Попри наявність базових аналітичних функцій, система не реалізує повноцінного CRM-функціоналу та не підтримує прогнозування попиту на основі статистичних моделей [15].

Платформа Creatio (Terrasoft) є потужним CRM-рішенням загального призначення з широкими можливостями налаштування бізнес-процесів. Однак вона орієнтована передусім на корпоративний сектор, має значну вартість ліцензування і не містить вбудованих алгоритмів аналізу фармацевтичних часових рядів. Адаптація платформи під потреби малої або середньої аптеки потребує значних фінансових і часових витрат [16].

Програма “МедФарм” забезпечує комплексну автоматизацію облікових процесів і формування регуляторної звітності, але аналітичний модуль обмежується простими статистичними зрізами продажів і не реалізує жодних прогнозних алгоритмів. Відсутня також підтримка ведення клієнтської бази у повному розумінні CRM [17].

Таким чином, жодне з розглянутих рішень не задовольняє комплексній вимозі: наявність повноцінного CRM-модуля в поєднанні з вбудованим прогнозуванням продажів на основі моделей часових рядів за доступною вартістю та з можливістю адаптації до специфіки конкретної аптеки. Це обґрунтовує актуальність і доцільність розробки власного програмного забезпечення в рамках даної кваліфікаційної роботи.

1.3 Постановка завдання та критерії успішності

На підставі проведеного аналізу предметної області та наявних програмних рішень було сформульовано мету та завдання розроблюваної системи.

Мета роботи: розробити та протестувати CRM-систему для аптеки з інтегрованим модулем прогнозування продажів на основі моделей часових рядів, яка дозволяє автоматизувати ключові бізнес-процеси аптечної діяльності та підтримує прийняття управлінських рішень щодо управління товарними запасами.

Для досягнення поставленої мети визначено такі завдання:

– Провести аналіз вимог до програмної системи, визначити акторів і варіанти використання.

– Спроєктувати архітектуру CRM-системи з урахуванням вимог до масштабованості та підтримуваності.

– Розробити модулі обліку клієнтів, управління товарами та продажами.

– Реалізувати модуль прогнозування продажів на основі моделей часових рядів (ARIMA/SARIMA, Prophet або аналогічних).

– Провести комплексне тестування розробленого програмного забезпечення та верифікувати точність прогнозних моделей.

Критерії успішності розробленої системи включають: коректне виконання всіх функціональних вимог, підтверджене тестуванням; передбачену можливість кількісної оцінки точності прогнозування за метриками MAE (Mean Absolute Error) та MAPE (Mean Absolute Percentage Error) шляхом крос-валідації на

ретроспективних даних продажів; зручний інтерфейс користувача, придатний для роботи провізора без спеціальних технічних знань.

1.4 Аналіз та підготовка даних

Коректна робота модуля прогнозування залежить від якості та структури вхідних даних. Для побудови моделей часових рядів необхідні дані про продажі у формі впорядкованих часових послідовностей. В умовах реальної аптеки джерелом таких даних є журнал касових операцій, що містить дату, найменування препарату, одиницю обліку, кількість і суму продажу.

Типова структура вхідних даних для модуля прогнозування включає такі атрибути: унікальний ідентифікатор транзакції, дата та час продажу, ідентифікатор і назва препарату, фармацевтична група, виробник, кількість проданих одиниць, ціна та загальна сума. На основі цих даних формуються агреговані часові ряди – сумарний обсяг продажів конкретного препарату або групи за одиницю часу (добу, тиждень, місяць) [13].

Важливим етапом підготовки даних є формування неперервного добового часового ряду. Оскільки в окремі дні продажі певного препарату можуть бути відсутніми, такі дні заповнюються нульовими значеннями, що забезпечує рівномірну добову частоту ряду – необхідну умову коректної роботи статистичних моделей. Подальша обробка ряду, зокрема оцінювання трендової та сезонної складових, виконується безпосередньо всередині моделей бібліотеки прогнозування під час навчання.

Особливу увагу слід приділити сезонності. В аптечній практиці виявляються кілька рівнів сезонності: тижнева (підвищення продажів у понеділок після вихідних та у п’ятницю), річна (зростання продажів противірусних препаратів, вітамінів і засобів від застуди восени та навесні), а також нерегулярні спалахи, пов’язані з епідеміями грипу, ГРВІ тощо. Адекватне моделювання цих компонент є ключовою умовою точності прогнозу.

Якість вхідних даних безпосередньо визначає якість прогнозу. Тому в розроблюваній системі реалізовано базову попередню обробку: автоматичне формування неперервного часового ряду із заповненням пропущених днів нульовими значеннями та обмеження прогнозних значень знизу нулем (оскільки від'ємний попит не має фізичного сенсу). Перевірку достатності обсягу історичних даних виконано шляхом встановлення мінімального порогу довжини ряду, за недотримання якого прогноз не будується.

1.5 Визначення акторів та варіантів використання

Для визначення функціональних вимог до системи було виконано аналіз ролей користувачів (акторів) та варіантів використання системи (Use Case). Ця діяльність є центральним елементом об'єктно-орієнтованого аналізу та дозволяє перейти від неформальних побажань замовника до чіткої специфікації поведінки системи.

У розроблюваній CRM-системі виділяються чотири основних актори:

– Провізор (Pharmacist) – основний користувач системи, що здійснює обслуговування клієнтів, проведення продажів, пошук і перевірку наявності препаратів, ведення карток постійних покупців та отримання аналітичних звітів.

– Завідувач аптекою (Manager) – користувач з розширеними правами, що управляє товарним асортиментом, переглядає та аналізує зведені звіти, формує замовлення постачальникам на підставі прогнозних даних, налаштовує параметри системи.

– Адміністратор системи (Administrator) – технічний користувач, що забезпечує налаштування, резервне копіювання, управління обліковими записами та правами доступу.

– Постачальник (Supplier) – користувач, що забезпечує постачання товарів аптекам і задовольняє потреби завідувачів аптекою.

На підставі визначених акторів було сформовано перелік варіантів використання (Use Cases) системи. Основні варіанти використання для актора

“Провізор”: обслуговування клієнта та проведення продажу; пошук препарату за назвою, діючою речовиною або штрих-кодом; реєстрація нового клієнта; нарахування та списання бонусних балів у рамках програми лояльності; перегляд залишків на складі.

Основні варіанти використання для актора “Завідувач аптекою”: перегляд аналітичних звітів з продажів у різних розрізах (препарат, група, виробник, період); запуск модуля прогнозування та отримання прогнозу продажів; формування замовлення постачальнику на основі прогнозних даних та поточних залишків; управління асортиментом (додавання, редагування, архівація товарних позицій); налаштування відсотка знижки, яку клієнт може сплатити бонусними балами в межах програми лояльності.

Для актора “Адміністратор системи” визначено такі варіанти використання: управління обліковими записами користувачів та їх правами доступу; налаштування параметрів з’єднання з базою даних; виконання резервного копіювання та відновлення даних; перегляд журналу системних подій та аудиту дій користувачів.

Пріоритетним варіантом використання, що визначає ключову цінність системи порівняно з аналогами, є прогнозування обсягів продажів. Даний сценарій передбачає вибір завідувачем аптекою препарату або групи препаратів, горизонту прогнозування (тиждень, місяць, квартал), запуск алгоритму, отримання числового прогнозу з довірчим інтервалом та графічну візуалізацію результату.

Діаграма варіантів використання (Use Case diagram) описує функціональні можливості системи з точки зору її користувачів (акторів). У системі виділено чотири ролі акторів: адміністратор, завідувач аптекою, провізор та постачальник.

Діаграму варіантів використання наведено на рисунку 1.1.

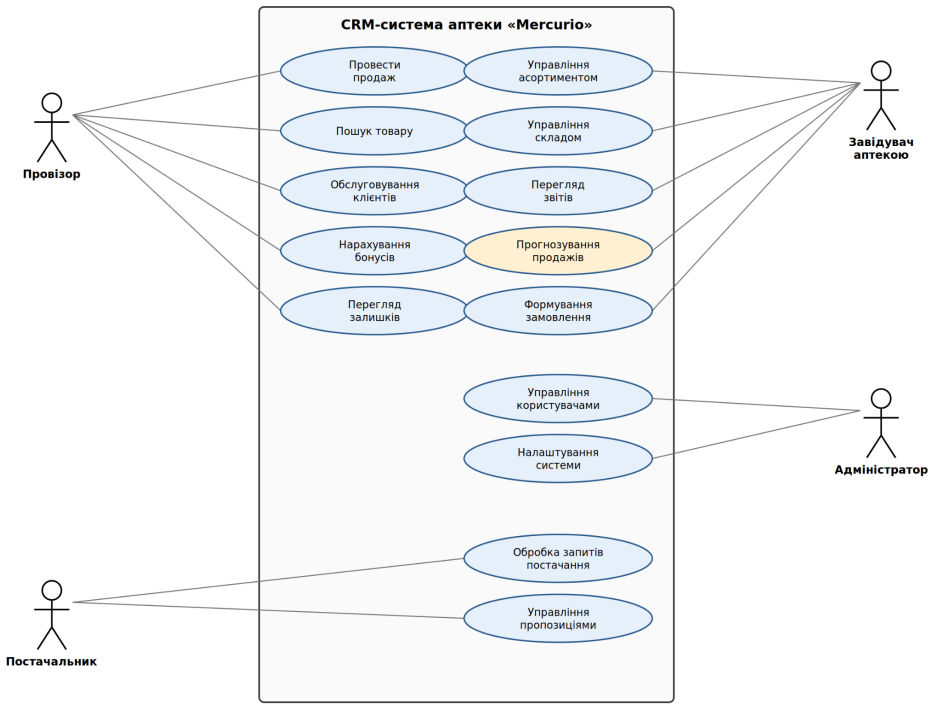

Рисунок 1.1 – Діаграма варіантів використання системи

Кожен з наведених варіантів використання реалізується відповідним набором класів та методів серверної частини, статичну структуру яких відображає діаграма класів (рисунок 2.4).

1.6 Специфікація функціональних та нефункціональних вимог

Специфікація вимог є ключовим артефактом процесу розробки програмного забезпечення, що фіксує узгоджені між замовником і розробником очікування щодо поведінки системи. Вимоги поділяються на функціональні (що система має робити) та нефункціональні (як система має це робити).

Функціональні вимоги до модуля управління клієнтами та продажами:

– Система повинна забезпечувати реєстрацію клієнтів з повною контактною інформацією та історією покупок.

– Система повинна підтримувати пошук клієнта за прізвищем, номером телефону або карткою лояльності.

– Система повинна автоматично нараховувати бонусні бали згідно з налаштованою схемою та дозволяти їх використання при наступних покупках.

– Система повинна фіксувати кожну транзакцію продажу з деталізацією по позиціях, збереженням дати, часу та ідентифікатора провізора.

– Система повинна відображати поточні залишки товарів у реальному часі після кожної операції продажу або надходження.

Функціональні вимоги до модуля управління товарним асортиментом:

– Система повинна зберігати довідник лікарських засобів із зазначенням торгової назви, МНН (міжнародної непатентованої назви), форми випуску, дозування, виробника, фармацевтичної групи та ознаки рецептурності.

– Система повинна підтримувати партійний облік товарів із фіксацією дати надходження, строку придатності та закупівельної ціни.

– Система повинна автоматично формувати сповіщення про товари з критично низьким залишком та товари з наближаючимся строком придатності.

Функціональні вимоги до модуля прогнозування:

– Система повинна дозволяти вибір препарату або фармацевтичної групи для прогнозування.

– Система повинна підтримувати вибір горизонту прогнозування: 7, 14, 30 та 90 днів.

– Система повинна відображати прогноз у вигляді числового значення та графіка з довірчим інтервалом.

– Система повинна зберігати результати прогнозування з можливістю їх подальшого перегляду та порівняння з фактичними даними.

Нефункціональні вимоги визначають якісні характеристики системи та умови її функціонування. Вимоги до надійності: система повинна забезпечувати коректне збереження даних при раптовому завершенні роботи; усі транзакції продажів повинні бути атомарними. Вимоги до безпеки: доступ до системи захищений парою “логін-пароль”; усі операції зі складськими залишками (надходження, продаж, коригування, списання, повернення) фіксуються в журналі руху запасів із зазначенням користувача, типу операції та обсягу зміни кількості.

1.7 Вибір технологічного стеку

На підставі сформованих вимог та аналізу доступних технологій обґрунтовано вибір технологічного стеку для розроблюваної системи. При виборі враховувалися такі чинники: відкритість і доступність технологій; наявність розвиненої екосистеми бібліотек для роботи з часовими рядами; зручність розгортання на типовому апаратному забезпеченні аптеки; підтримка мови Python, яка є стандартом де-факто в галузі аналізу даних і машинного навчання.

Для реалізації серверної частини обрано мову програмування Python версії 3.10 та веб-фреймворк FastAPI. Вибір FastAPI обґрунтований такими перевагами: автоматична генерація документації API у форматі OpenAPI; вбудована валідація вхідних даних на основі бібліотеки Pydantic; висока продуктивність завдяки асинхронній обробці запитів; природна інтеграція з бібліотеками машинного навчання екосистеми Python [5].

Для зберігання даних обрано реляційну СУБД PostgreSQL версії 15. Реляційна модель є природною для структурованих даних аптечного обліку, а PostgreSQL відрізняється надійністю, підтримкою транзакцій та багатим набором вбудованих функцій. Для роботи з базою даних застосовується бібліотека SQLModel, що поєднує можливості об’єктно-реляційного відображення (на основі SQLAlchemy) та валідації даних (на основі Pydantic), завдяки чому моделі даних і схеми API описуються єдиним набором класів [6][7].

Клієнтська частина реалізується у вигляді односторінкового веб-застосунку (SPA) на основі бібліотеки React 19 з використанням мови TypeScript, складальника Vite та набору UI-компонентів на основі Radix UI і Tailwind CSS; для візуалізації прогнозів застосовано бібліотеку Recharts. Такий підхід забезпечує зручний доступ до системи з будь-якого комп’ютера в локальній мережі аптеки без необхідності встановлення клієнтського програмного забезпечення [9].

Для реалізації модуля прогнозування задіяно бібліотеку statsforecast (розробка компанії Nixtla), що надає високопродуктивні реалізації статистичних

моделей часових рядів, зокрема ARIMA, SARIMA, експоненціального згладжування (ETS) та методу Theta. Допоміжну обробку числових даних виконано засобами бібліотек pandas та NumPy. Вибір саме цього набору зумовлений зрілістю бібліотек, їх високою продуктивністю та наявністю детальної документації [4].

1.8 Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області, що охоплює специфіку управління аптечним підприємством та застосування методів аналізу часових рядів для прогнозування попиту на фармацевтичні препарати. Здійснено огляд і порівняльний аналіз існуючих програмних рішень для автоматизації аптек, виявлено їх ключові обмеження щодо CRM-функціональності та модулів прогнозування, що обґрунтовує доцільність розробки власного програмного забезпечення. Сформульовано мету, завдання і критерії успішності проєкту. Визначено акторів системи, описано функціональні та нефункціональні вимоги, обґрунтовано вибір технологічного стеку. Також у розділі представлено UML-діаграму варіантів використання, яка ілюструє визначені сценарії взаємодії користувачів із системою. Отримані результати є вихідними даними для проєктування та розробки програмної системи в наступному розділі.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У цьому розділі описано процес проєктування та програмної реалізації CRM-системи для аптеки з інтегрованим модулем прогнозування продажів. Обґрунтовано вибір моделі процесу розробки, спроєктовано загальну архітектуру системи на основі мікросервісного підходу, обґрунтовано вибір технологічного стеку. Розроблено схему бази даних та комплекс UML-діаграм, що описують структуру і поведінку системи. Окрему увагу приділено реалізації серверної частини та модуля прогнозування на основі моделей часових рядів. Розробку виконано як єдиний Git-репозиторій із дотриманням сучасних практик інженерії програмного забезпечення.

2.1 Вибір процесу розробки

Вибір моделі життєвого циклу програмного забезпечення безпосередньо впливає на ефективність розробки та якість кінцевого продукту. З огляду на те, що кваліфікаційна робота виконується одним розробником в умовах поетапного уточнення вимог, було обрано ітеративно-інкрементну модель розробки з елементами гнучких методологій (Agile).

Для організації роботи в межах ітеративно-інкрементної моделі застосовано елементи методології Scrumban [19] (поєднання Scrum та Kanban) на основі інструментів GitHub Issues, Milestones та Projects. Журнал вимог (product backlog) сформовано у вигляді завдань (issues), кожне з яких оформлено за уніфікованим шаблоном (опис мети, чек-лист підзадач, критерії приймання) та категоризовано мітками (epic, backend, frontend, ml, database, testing, devops) відповідно до архітектурної приналежності.

Розробку поділено на шість спринтів тривалістю 10-14 днів (milestones): “Foundation & Architecture”, “Auth & Core Entities”, “Sales & Loyalty”, “B2B Marketplace”, “Forecast Service (ML)” та “Testing & Deployment”, кожен з яких відповідає логічно завершеному приросту функціональності. Хід виконання

відображено на канбан-дошці (GitHub Projects) з колонками “To Do”, “In Progress” та “Done”.

2.2 Проєктування архітектури системи

Архітектура розроблюваної системи побудована на основі мікросервісного підходу, за якого програмна система складається з кількох незалежних служб (сервісів), що взаємодіють між собою за допомогою чітко визначених програмних інтерфейсів [20]. Такий підхід дозволяє розмежувати відповідальність між компонентами, спрощує масштабування та підтримку, а також забезпечує технологічну гнучкість окремих частин системи.

Система складається з трьох основних компонентів:

– Основний сервіс CRM (backend) – серверна частина, що реалізує бізнес-логіку обліку товарів, аптек, складів, продажів, клієнтів та постачальників, а також автентифікацію й авторизацію користувачів.

– Сервіс прогнозування (forecast service) – окрема служба, що реалізує прогнозування попиту на основі моделей часових рядів і взаємодіє зі спільною базою даних для отримання історичних даних про продажі.

– Клієнтська частина (frontend) – односторінковий веб-застосунок, що надає користувацький інтерфейс і взаємодіє з серверними сервісами через REST API.

Усі компоненти системи об’єднані в спільне середовище за допомогою технології контейнеризації Docker та засобу оркестрації Docker Compose [10]. Сервіси взаємодіють зі спільною реляційною базою даних PostgreSQL. Автентифікація користувачів реалізована на основі токенів JWT, причому сервіс прогнозування використовує той самий секретний ключ підпису, що й основний сервіс CRM, завдяки чому токен, виданий основним сервісом, приймається сервісом прогнозування без повторної автентифікації. Загальну архітектуру системи наведено на рисунку 2.1.

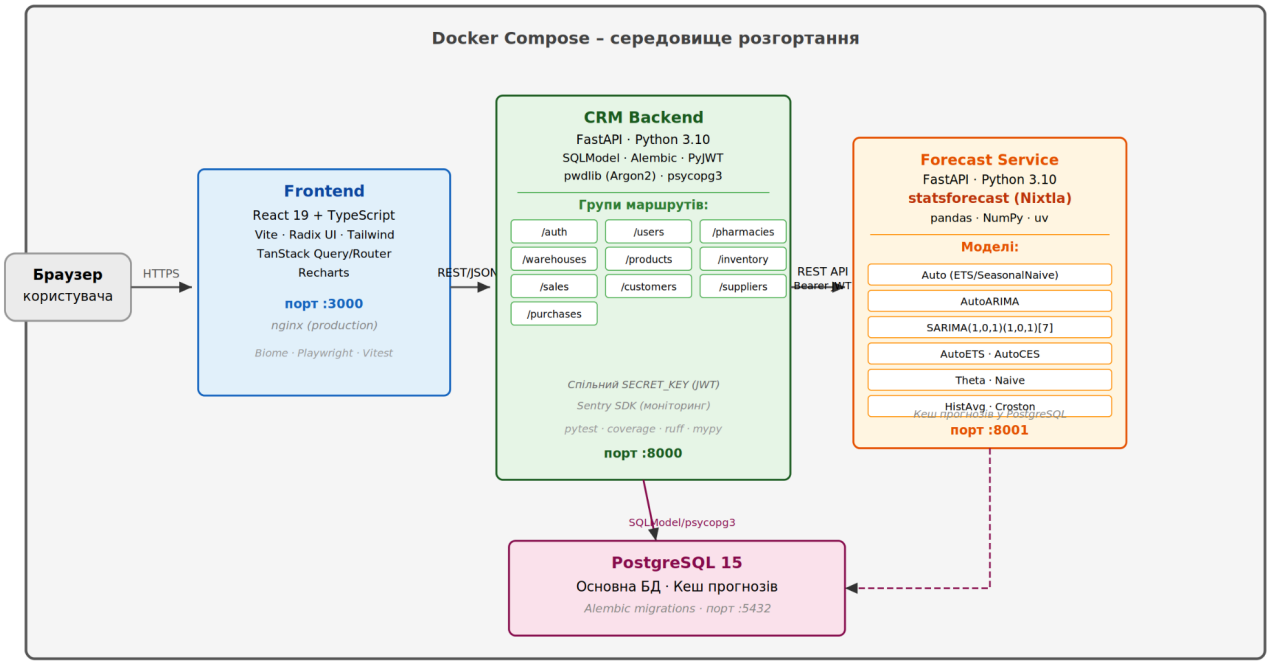

Рисунок 2.1 – Загальна архітектура програмної системи

Розмежування сервісу прогнозування й основного сервісу CRM обґрунтоване тим, що обчислення прогнозів є ресурсомісткою операцією, яка спирається на спеціалізовані бібліотеки аналізу часових рядів. Винесення цієї функціональності в окрему службу дозволяє незалежно масштабувати обчислювальні потужності, оновлювати моделі прогнозування без впливу на основну бізнес-логіку та ізолювати залежності між компонентами.

Серверна частина побудована за багатошаровою (layered) архітектурою, що передбачає поділ коду на рівні з чітко визначеною відповідальністю: рівень представлення (API-маршрути, що приймають HTTP-запити та повертають відповіді); рівень бізнес-логіки (модуль CRUD-операцій та допоміжні служби); рівень доступу до даних (моделі даних та об’єктно-реляційне відображення) [21]. Такий поділ підвищує підтримуваність коду та спрощує його тестування. Схему багаторівневої архітектури серверної частини наведено на рисунку 2.2.

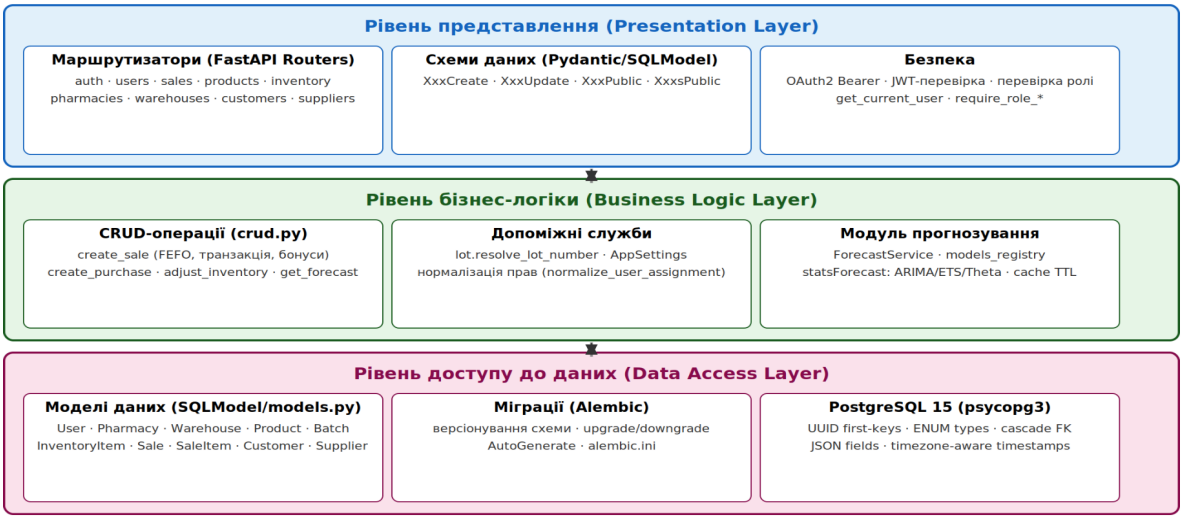

Рисунок 2.2 – Багаторівнева архітектура серверної частини

Описаний розподіл на рівні представлення, бізнес-логіки та доступу до даних є основою для подальшого обґрунтування вибору конкретних технологій, що реалізують кожен із цих рівнів.

2.3 Обґрунтування вибору технологій та інструментів розробки

Вибір технологічного стеку здійснено на основі вимог, сформульованих у першому розділі, з урахуванням таких чинників: продуктивність та надійність технологій, наявність розвиненої екосистеми бібліотек, зокрема для аналізу часових рядів, а також зручність розгортання та підтримки.

Для реалізації серверної частини обрано мову програмування Python версії 3.10 та веб-фреймворк FastAPI. FastAPI є сучасним високопродуктивним фреймворком, що забезпечує автоматичну валідацію вхідних даних, генерацію інтерактивної документації API у форматі OpenAPI та підтримку асинхронної обробки запитів. Вибір Python обґрунтований також тим, що ця мова є фактичним стандартом у галузі аналізу даних і машинного навчання, що спрощує інтеграцію модуля прогнозування [3][5].

Для роботи з базою даних застосовано бібліотеку SQLModel, що поєднує можливості об’єктно-реляційного відображення (на основі SQLAlchemy) та

валідації даних (на основі Pydantic) [8]. SQLModel дозволяє описувати моделі даних і схеми API єдиним набором класів, що зменшує дублювання коду та ймовірність помилок. Для керування міграціями схеми бази даних використано інструмент Alembic, що дозволяє версіонувати зміни структури бази та застосовувати їх контрольованим чином [6].

Як систему управління базами даних обрано реляційну СУБД PostgreSQL. Реляційна модель є природною для структурованих даних аптечного обліку з численними зв'язками між сутностями, а PostgreSQL відрізняється надійністю, повноцінною підтримкою транзакцій та широким набором вбудованих типів даних і функцій.

Автентифікація реалізована на токенах JWT (бібліотека PyJWT), паролі зберігаються у вигляді хешів Argon2/bcrypt (бібліотека pwdlib) відповідно до сучасних рекомендацій щодо захисту облікових даних. Модуль прогнозування побудовано на бібліотеці statsforecast (Nixtla) з моделями ARIMA, SARIMA, ETS та Theta; обробку часових рядів виконано засобами pandas і NumPy.

Клієнтську частину реалізовано у вигляді односторінкового веб-застосунку (SPA) на основі бібліотеки React. Такий підхід забезпечує доступ до системи з будь-якого комп'ютера в локальній мережі аптеки через веб-браузер без потреби встановлення спеціального клієнтського програмного забезпечення.

Початкову структуру серверної та клієнтської частин, конфігурацію Docker Compose, базову схему автентифікації JWT та налаштування CI/CD на основі GitHub Actions запозичено з відкритого шаблону Full Stack FastAPI Template [18], що поширюється на умовах ліцензії MIT. Зазначений шаблон, розроблений автором фреймворку FastAPI, є визнаною спільнотою основою для побудови проєктів на стеку FastAPI + SQLModel + PostgreSQL + React і слугував вихідною точкою проєкту. У подальшому ця базова структура була суттєво розширена та адаптована: реалізовано власну предметну модель даних (11 сутностей аптечного обліку), партійний облік і алгоритм FEFO, рольову модель доступу, мікросервіс прогнозування на основі statsforecast, B2B-маркетплейс взаємодії з постачальниками та програму лояльності клієнтів.

Для управління залежностями та віртуальним середовищем Python застосовано сучасний інструмент uv. Узагальнений перелік використаних технологій наведено в таблиці 2.1.

Таблиця 2.1 – Технологічний стек програмної системи

Компонент	**Технологія**	**Призначення**
Мова розробки	Python 3.10	Реалізація серверної частини та модуля прогнозування
Веб-фреймворк	FastAPI	Побудова REST API, валідація даних, документація OpenAPI
ORM	SQLModel / SQLAlchemy	Об’єктно-реляційне відображення, опис моделей даних
Міграції БД	Alembic	Версіонування та застосування змін схеми бази даних
СУБД	PostgreSQL	Зберігання структурованих даних аптечного обліку
Автентифікація	PyJWT, pwdlib	Видача JWT-токенів, хешування паролів (Argon2/bcrypt)
Прогнозування	statsforecast (Nixtla)	Моделі часових рядів (ARIMA, SARIMA, ETS, Theta)
Обробка даних	pandas, NumPy	Формування та обробка часових рядів
Клієнтська частина	React	Користувацький інтерфейс (SPA)
Контейнеризація	Docker, Docker Compose	Розгортання та оркестрація сервісів
Якість коду	pytest	Тестування

Наведений технологічний стек забезпечує узгоджену взаємодію всіх компонентів системи – від збереження даних до клієнтського інтерфейсу – і використовується як основа для подальшого проєктування структур даних та програмних модулів.

2.4 Проєктування структур даних та схеми бази даних

Проєктування бази даних є одним із ключових етапів розробки інформаційної системи, оскільки структура даних визначає можливості та обмеження всієї системи. Схему бази даних спроєктовано на основі аналізу

предметної області, виконаного в першому розділі, з дотриманням принципів нормалізації для уникнення надмірності та аномалій даних.

Основні сутності системи описані в таблиці 2.2.

Таблиця 2.2 – Основні сутності бази даних системи

Сутність	**Призначення / ключові атрибути**
User	Обліковий запис із роллю (адміністратор, завідувач, провізор, постачальник), email, зв’язки з аптекою/складом/постачальником
Pharmacy	Заклад: назва, адреса, телефон; об’єднує склади та продажі
Warehouse	Місце зберігання товарів, прив’язане до аптеки
Product	Довідникова позиція ЛЗ: назва, SKU, штрих-код
InventoryItem	Зв'язок товару зі складом: кількість, ціна, строк придатності
Batch	Партія надходження: номер партії, строк придатності (партійний облік)
Customer	Покупець із карткою лояльності: контакти, бонусні бали
Sale	Продаж: сума, спосіб оплати, бонуси, провізор, клієнт
SaleItem	Позиція продажу: товар, кількість, ціна
Supplier	Постачальник товарів
InventoryMovement	Журнал змін залишків: тип операції (надходження, продаж, коригування, списання, повернення)

Між сутностями встановлено зв’язки типу “один до багатьох” та “багато до багатьох”. Наприклад, одна аптека може мати кілька складів, один продаж складається з кількох позицій, а кожна позиція продажу пов’язана з товаром. Як первинні ключі сутностей використано ідентифікатори типу UUID, що забезпечує глобальну унікальність записів та спрощує інтеграцію між сервісами.

Модель бази даних реалізована засобами PostgreSQL. Для полів зі скінченною множиною значень (роль користувача, спосіб оплати, тип руху запасів) застосовано перелічувані типи (ENUM), що забезпечує контроль цілісності на рівні бази даних. Для забезпечення посилальної цілісності між таблицями налаштовано зовнішні ключі з відповідними правилами каскадних дій (SET NULL для збереження історії, RESTRICT для запобігання видаленню використовуваних записів) [7]. Фізичну модель бази даних наведено на рисунку 2.3.

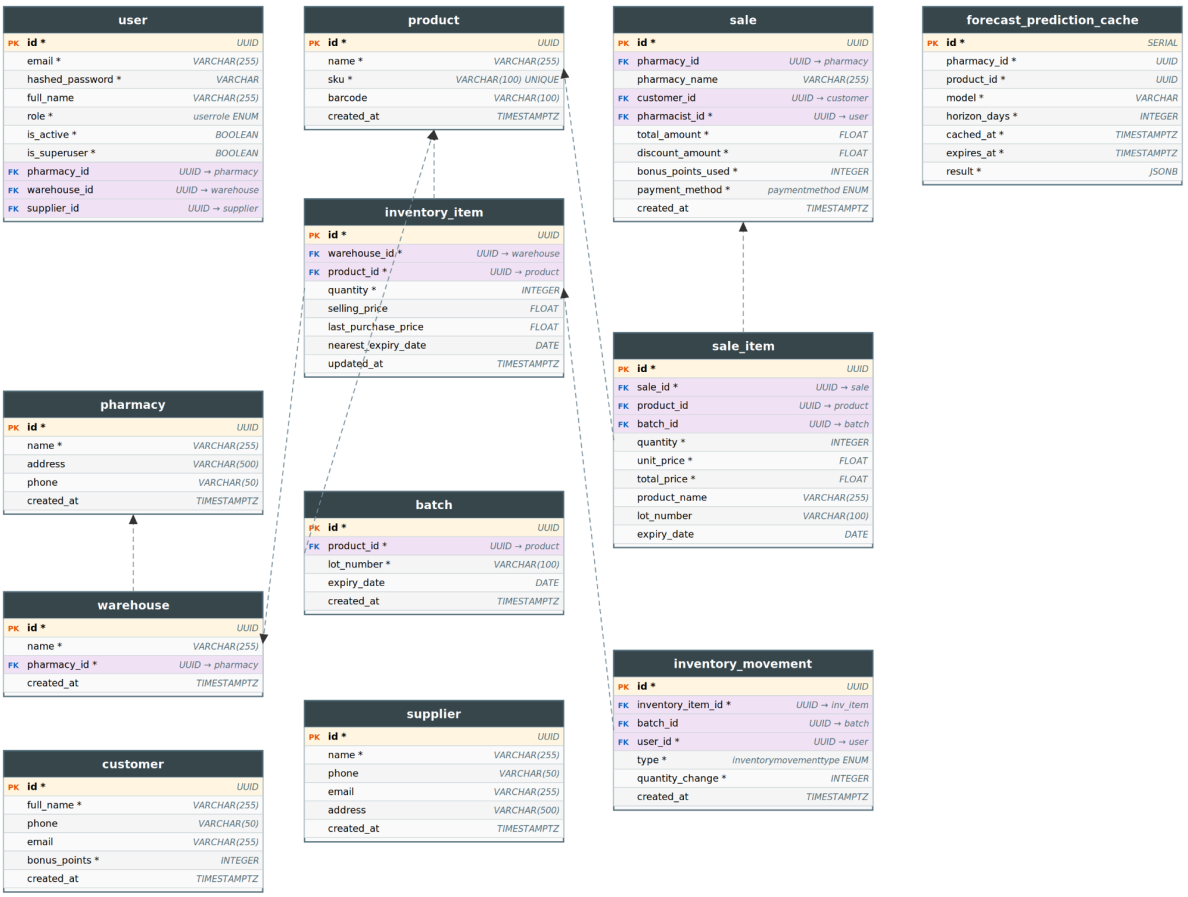

Рисунок 2.3 – Фізична модель бази даних (спрощена)

Спроєктована схема бази даних є фундаментом для об’єктної моделі системи, що описується засобами UML-діаграм у наступному підрозділі.

2.5 Об’єктно-орієнтоване проєктування системи

Для формального опису структури та поведінки системи застосовано уніфіковану мову моделювання UML (Unified Modeling Language). UML є стандартизованою графічною мовою, що дозволяє описувати програмну систему з різних точок зору за допомогою діаграм різних типів [22]. У межах роботи побудовано діаграму класів, діаграму послідовності та діаграму діяльності.

2.5.1 Діаграма класів

Діаграма класів (Class diagram) відображає статичну структуру системи: класи, їхні атрибути, методи та зв’язки між ними. Основу об’єктної моделі системи становлять класи, що відповідають сутностям предметної області. Завдяки застосуванню бібліотеки SQLModel кожна сутність описується ієрархією класів: базовий клас із спільними атрибутами, клас таблиці (з ознакою table=True) для відображення в базі даних та допоміжні класи для операцій створення, оновлення й публічного подання даних.

Наприклад, для сутності продажу визначено базовий клас SaleBase (з атрибутами суми, знижки, використаних бонусів та способу оплати), клас таблиці Sale (з ідентифікатором, зовнішніми ключами та зв’язками), клас SaleCreate (для прийому даних під час створення) та клас SalePublic (для повернення даних клієнту). Діаграму класів основних сутностей системи наведено на рисунку 2.4.

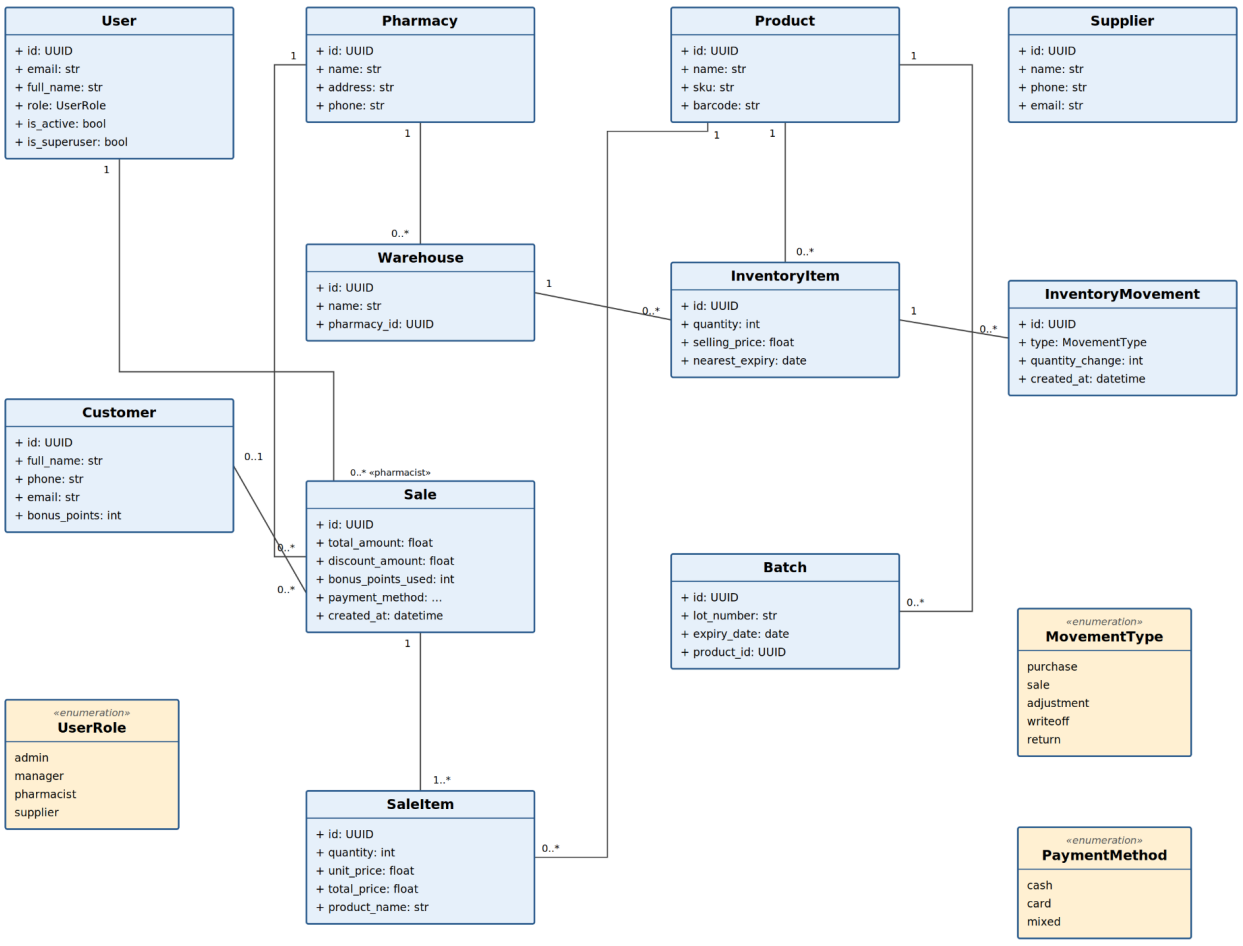

Рисунок 2.4 – Діаграма класів основних сутностей системи (спрощена)

Описана статична структура класів визначає об’єкти, що беруть участь у динамічних сценаріях взаємодії, один з яких розглянуто на діаграмі послідовності.

2.5.2 Діаграма послідовності

Діаграма послідовності (Sequence diagram) описує динамічну поведінку системи, відображаючи обмін повідомленнями між об’єктами в часі під час виконання певного сценарію. Для ілюстрації роботи системи розглянуто сценарій проведення продажу як один із найбільш складних та критичних процесів.

Сценарій починається із запиту провізора через клієнтську частину. Запит надходить до відповідного API-маршруту серверної частини, який перевіряє права доступу за JWT-токеном. Далі викликається відповідна функція бізнес-логіки, що в межах єдиної транзакції бази даних виконує такі дії: перевіряє наявність достатньої кількості товару на складі, списує товар із відповідних партій за принципом FEFO (First Expired, First Out), створює запис продажу та його позиції, фіксує рухи запасів та за потреби нараховує або списує бонусні бали клієнта. У разі недостатньої кількості товару транзакція скасовується, а клієнту повертається повідомлення про помилку. Діаграму послідовності процесу проведення продажу наведено на рисунку 2.5.

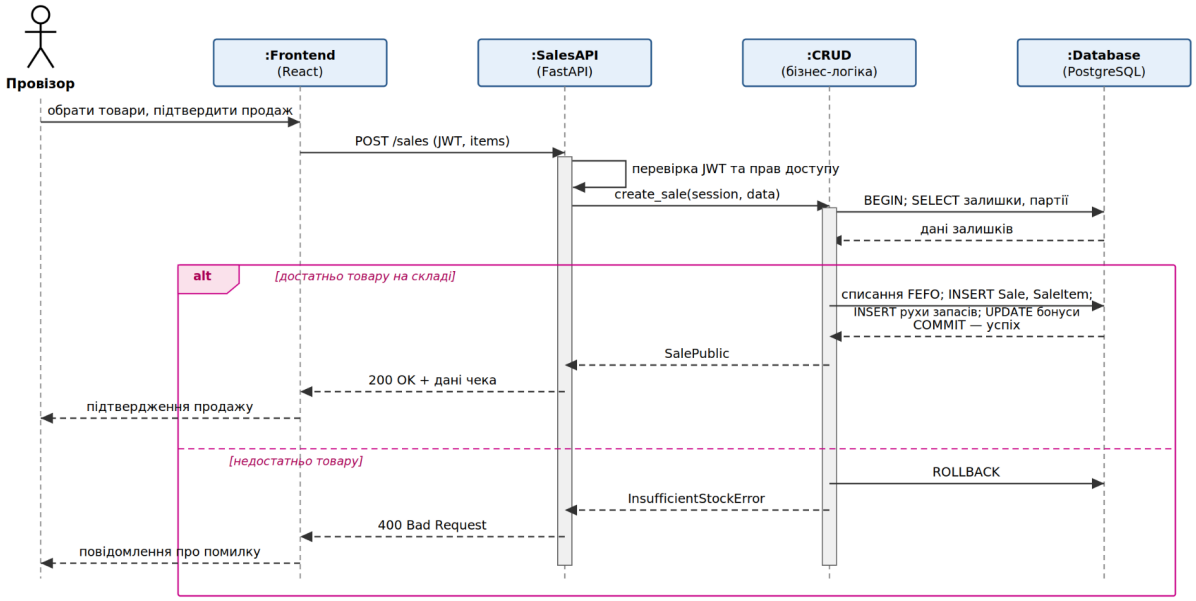

Рисунок 2.5 – Діаграма послідовності процесу проведення продажу

Розглянутий сценарій ілюструє транзакційний підхід до обробки продажу, що детальніше описано в підрозділі 2.6 при розгляді реалізації серверної частини.

2.5.3 Діаграма діяльності

Діаграма діяльності (Activity diagram) описує потік керування в межах бізнес-процесу, відображаючи послідовність дій, точки прийняття рішень та паралельні гілки. Для ілюстрації бізнес-логіки системи розглянуто процес прогнозування попиту та формування замовлення постачальнику.

Процес розпочинається з вибору завідувачем аптекою товару та горизонту прогнозування. Система перевіряє наявність достатньої історії продажів; за її відсутності користувачу повідомляється про неможливість побудови прогнозу. За наявності достатніх даних система перевіряє кеш прогнозів: якщо актуальний прогноз уже існує, він повертається без повторного обчислення, інакше запускається обчислення моделі. Діаграму діяльності процесу прогнозування наведено на рисунку 2.6.

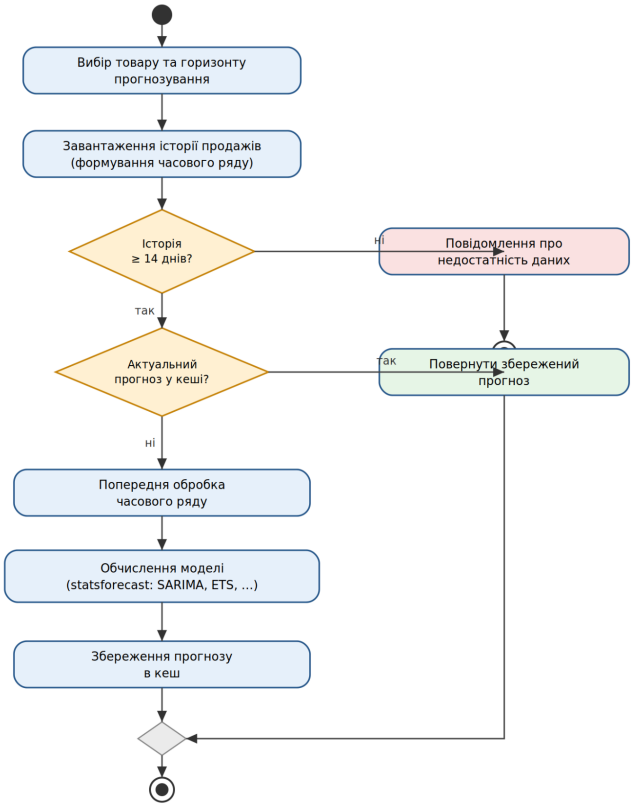

Рисунок 2.6 – Діаграма діяльності процесу прогнозування попиту

Розглянуті UML-діаграми в сукупності формують повне уявлення про структуру та поведінку системи, що є основою для опису її програмної реалізації в наступних підрозділах.

2.6 Реалізація основних класів та методів серверної частини

Серверну частину системи реалізовано на мові Python із застосуванням фреймворку FastAPI. Структуру проєкту організовано відповідно до принципів багаторівневої архітектури: моделі даних зосереджено в окремому модулі, бізнес-логіку CRUD-операцій – в окремому модулі, а API-маршрути згруповано за функціональними областями.

Моделі даних описано засобами SQLModel у вигляді класів Python. Кожен клас таблиці успадковує базовий клас із атрибутами та доповнюється полями ідентифікатора, зовнішніх ключів і зв’язків з іншими сутностями. Поля моделей містять обмеження валідації (мінімальна та максимальна довжина рядків, невід’ємність числових значень), що забезпечує контроль коректності даних на рівні застосунку.

Бізнес-логіку системи зосереджено в модулі CRUD-операцій, що містить функції створення, читання, оновлення та видалення сутностей. Найбільш складною є функція обробки продажу, яка виконується в межах єдиної транзакції бази даних та забезпечує атомарність операції: усі зміни (списання товару, створення записів продажу, фіксація рухів запасів, оновлення бонусів) застосовуються разом або не застосовуються зовсім у разі помилки. Для коректного списання товарів реалізовано алгоритм вибору партій за принципом FEFO, згідно з яким першими списуються товари з найближчим строком придатності [13].

Рівень представлення реалізовано у вигляді API-маршрутів, згрупованих за сутностями. Кожен маршрут приймає валідовані вхідні дані, делегує обробку відповідній функції бізнес-логіки та повертає результат у форматі JSON. Доступ

до маршрутів захищено механізмом залежностей FastAPI, що перевіряє JWT-токен та права доступу користувача відповідно до його ролі [11]. Узагальнений перелік основних груп API-маршрутів серверної частини наведено в таблиці 2.3.

Таблиця 2.3 – Основні групи API-маршрутів серверної частини

Група маршрутів	**Метод**	**Призначення**
/login	POST	Автентифікація користувача та отримання JWT-токена
/users	GET, POST, PATCH, DELETE	Управління обліковими записами користувачів
/products	GET, POST, PATCH, DELETE	Управління довідником товарів
/pharmacies	GET, POST, PATCH, DELETE	Управління аптеками та складами
/inventory	GET, PATCH	Перегляд та коригування складських залишків
/sales	GET, POST	Проведення продажів та перегляд історії
/customers	GET, POST, PATCH	Управління клієнтами та програмою лояльності
/suppliers	GET, POST, PATCH	Управління постачальниками та закупівлями

Автентифікацію реалізовано на основі стандарту OAuth2 з паролем та токенами JWT. Під час входу користувач надає електронну пошту та пароль; за умови їх коректності система генерує підписаний токен доступу з обмеженим строком дії, що містить ідентифікатор користувача [11].

Подальші запити супроводжуються цим токеном у заголовку авторизації. Паролі зберігаються в базі даних виключно у вигляді криптографічних хешів [12].

2.7 Реалізація модуля прогнозування продажів

Модуль прогнозування реалізовано як окремий мікросервіс на основі фреймворку FastAPI, що взаємодіє зі спільною базою даних для отримання історичних даних про продажі. Завданням модуля є побудова прогнозу попиту на конкретний товар у конкретній аптеці на заданий горизонт часу.

Вхідними даними для прогнозування є часовий ряд щоденних обсягів продажів пари “аптека-товар”, що формується агрегуванням позицій продажів за датою. Перед побудовою прогнозу часовий ряд проходить попередню обробку: дні без продажів заповнюються нульовими значеннями для отримання неперервного ряду з добовою частотою. Мінімальною умовою побудови прогнозу є наявність історії продажів обсягом не менше 14 днів.

У модулі реалізовано набір моделей прогнозування, що дозволяє обирати найбільш придатну модель залежно від характеру даних та обсягу історії [3]. Перелік доступних моделей наведено в таблиці 2.4.

Таблиця 2.4 – Моделі прогнозування, реалізовані в системі

Модель	**Опис**
Auto	Автоматичний вибір ETS або сезонної наївної моделі
AutoETS	Експоненціальне згладжування з автодобором параметрів
AutoARIMA	ARIMA з автоматичним добором порядку моделі
SARIMA	Сезонна ARIMA(1,0,1)(1,0,1) з тижневою сезонністю
AutoCES	Комплексне експоненціальне згладжування
Seasonal Naive	Повторення значень із тижневим лагом
Theta	Метод Theta з тижневою сезонністю
Naive	Останнє спостереження (базовий орієнтир)
Historic Average	Середнє за всю історію
Croston	Для рідкісного (переривчастого) попиту

Вибір конкретної моделі відбувається автоматично або задається користувачем явно. У режимі автоматичного вибору система обирає модель залежно від обсягу доступної історії: за наявності достатньої кількості даних застосовується SARIMA, інакше – сезонна наївна модель як стійкіший базовий варіант.

Обчислення прогнозу виконується центральною функцією модуля, що отримує часовий ряд, обрану модель і горизонт прогнозування, після чого повертає прогнозні значення з довірчими інтервалами та узагальнені показники

(середній добовий попит, сумарний прогнозований попит). Усі прогнозні значення обмежуються знизу нулем, оскільки від'ємний попит не має фізичного сенсу.

Оскільки обчислення прогнозу є ресурсомісткою операцією, у системі реалізовано механізм кешування результатів у базі даних. Обчислений прогноз зберігається в окремій таблиці кешу разом із міткою часу. У разі повторного запиту, якщо збережений прогноз не старший за встановлений строк дії (за замовчуванням – одна година), система повертає його без повторного обчислення моделі. Передбачено також можливість примусового оновлення прогнозу та фонового періодичного оновлення кешу за розкладом. Така організація суттєво знижує обчислювальне навантаження та скорочує час відповіді системи.

Доступ до модуля прогнозування реалізовано через REST API. Основні маршрути забезпечують отримання прогнозу для пари "аптека-товар", пакетне прогнозування для кількох товарів однієї аптеки, отримання переліку доступних моделей та перевірку працездатності сервісу. Доступ захищено тим самим механізмом JWT-токенів, що й основний сервіс, з урахуванням ролі користувача: завідувач та провізор отримують доступ лише до даних власної аптеки, тоді як адміністратор – до даних усіх аптек.

2.8 Розробка інтерфейсу користувача

Клієнтську частину системи реалізовано у вигляді односторінкового веб-застосунку на основі бібліотеки React. Інтерфейс взаємодіє із серверними сервісами виключно через REST API, отримуючи та надсилаючи дані у форматі JSON. Такий підхід забезпечує чітке розмежування клієнтської та серверної частин і дозволяє розгортати застосунок незалежно.

Інтерфейс організовано навколо основних робочих сценаріїв користувачів. Головні розділи застосунку включають: управління інфраструктурою аптеки (аптеки, склади), управління асортиментом та клієнтами, підсистему взаємодії з постачальниками (B2B-маркетплейс), операції продажу й інвентар, а також модуль

прогнозування попиту на основі ML-моделей. Детальний опис кожного розділу наведено нижче з відповідними скріншотами.

2.8.1 Автентифікація та головна панель

Вхід до системи захищено формою автентифікації з полями електронної пошти та паролю. Реалізовано підтримку двох тем оформлення – світлої та темної, вибір між якими зберігається у налаштуваннях браузера користувача. Форма входу у світлій темі наведена на рисунку 2.7, у темній – на рисунку 2.8.

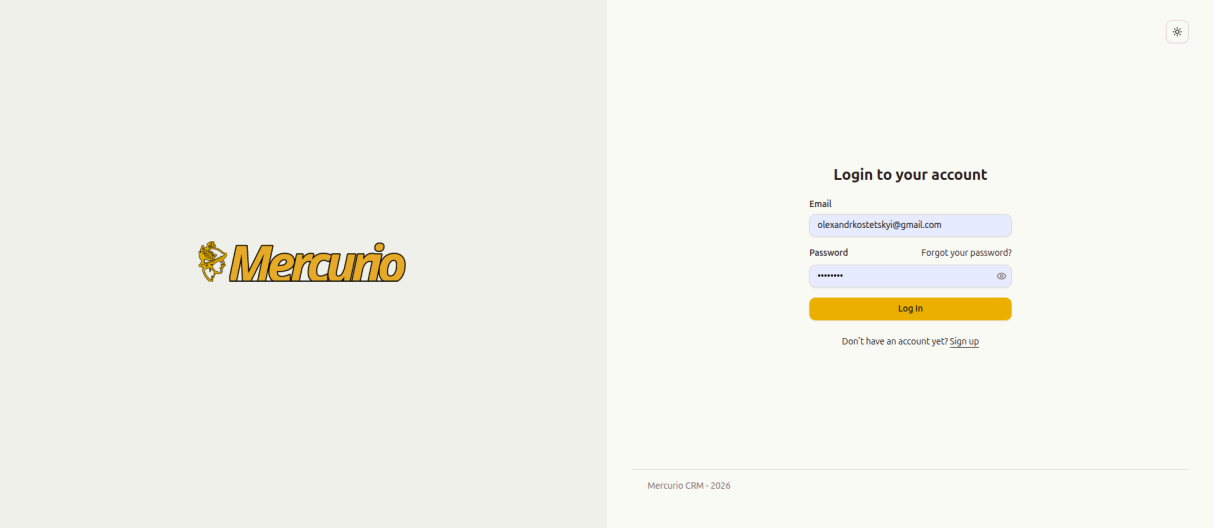

Рисунок 2.7 – Сторінка входу до системи (світла тема)

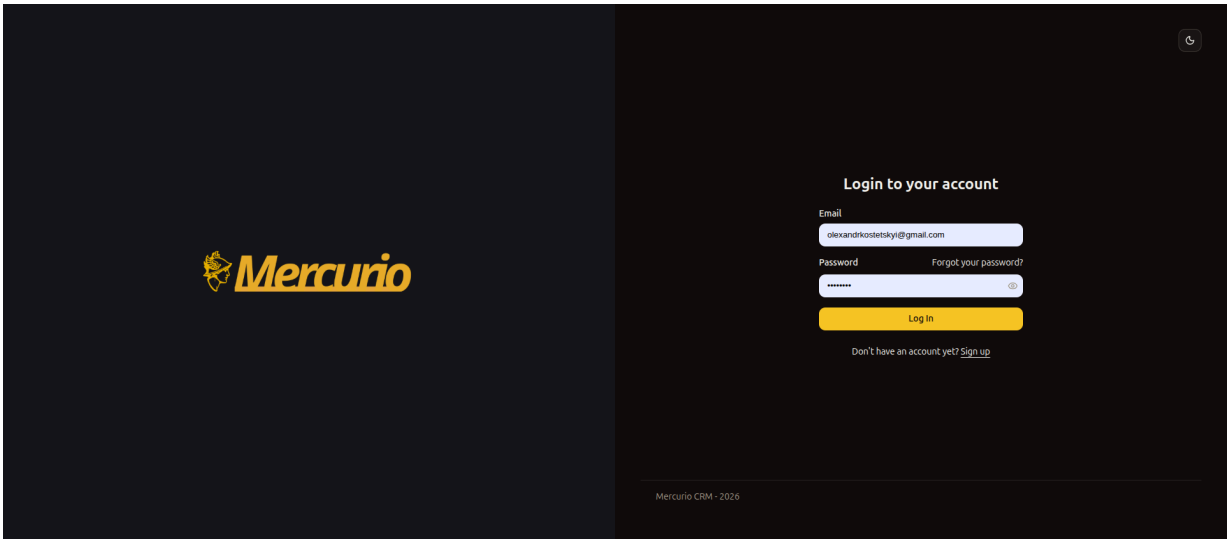

Рисунок 2.8 – Сторінка входу до системи (темна тема)

Після успішної автентифікації користувач потрапляє на головну панель, склад якої залежить від ролі: адміністратор бачить зведений дашборд із переліком усіх аптек, кількістю користувачів і швидким доступом до налаштувань системи (рисунок 2.9); провізор одразу відкриває форму продажу; завідувач – аналітичний звіт.

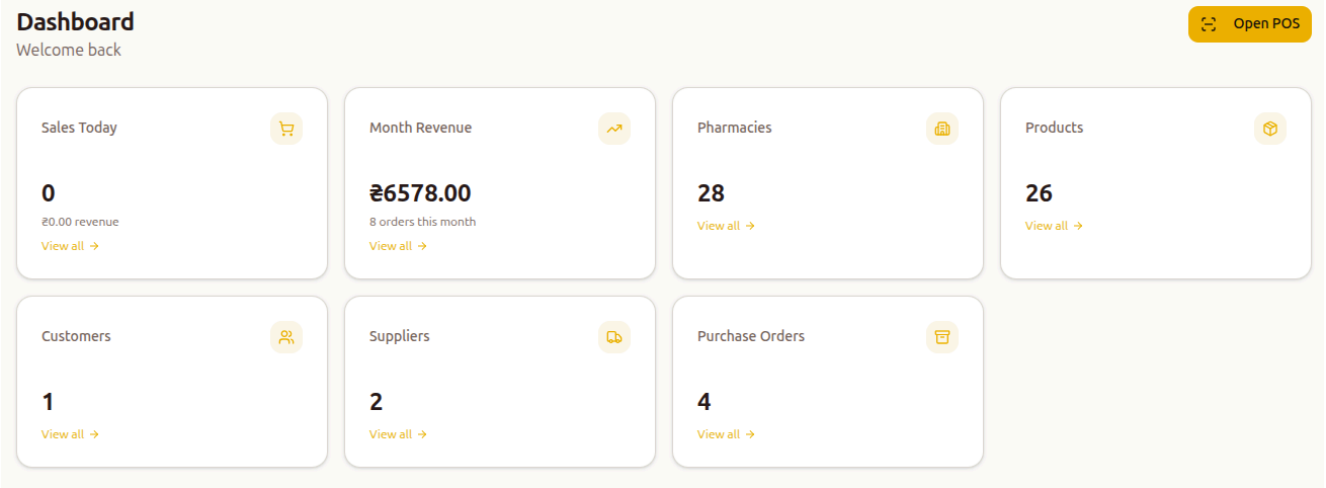

Рисунок 2.9 – Головна панель адміністратора системи

Таке розмежування початкового екрана відповідно до ролі користувача відповідає моделі рольового доступу, описаній у підрозділі 2.6.

2.8.2 Панель продажу – робоче місце провізора

Центральним елементом роботи провізора є панель продажу (рисунок 2.10), що забезпечує: швидкий пошук препарату за назвою, МНН або штрих-кодом; вибір кількості з автоматичною перевіркою залишків; прив’язку клієнта за номером телефону або карткою лояльності; вибір способу оплати. Після підтвердження система списує товар за FEFO та нараховує бонусні бали клієнту в межах єдиної транзакції.

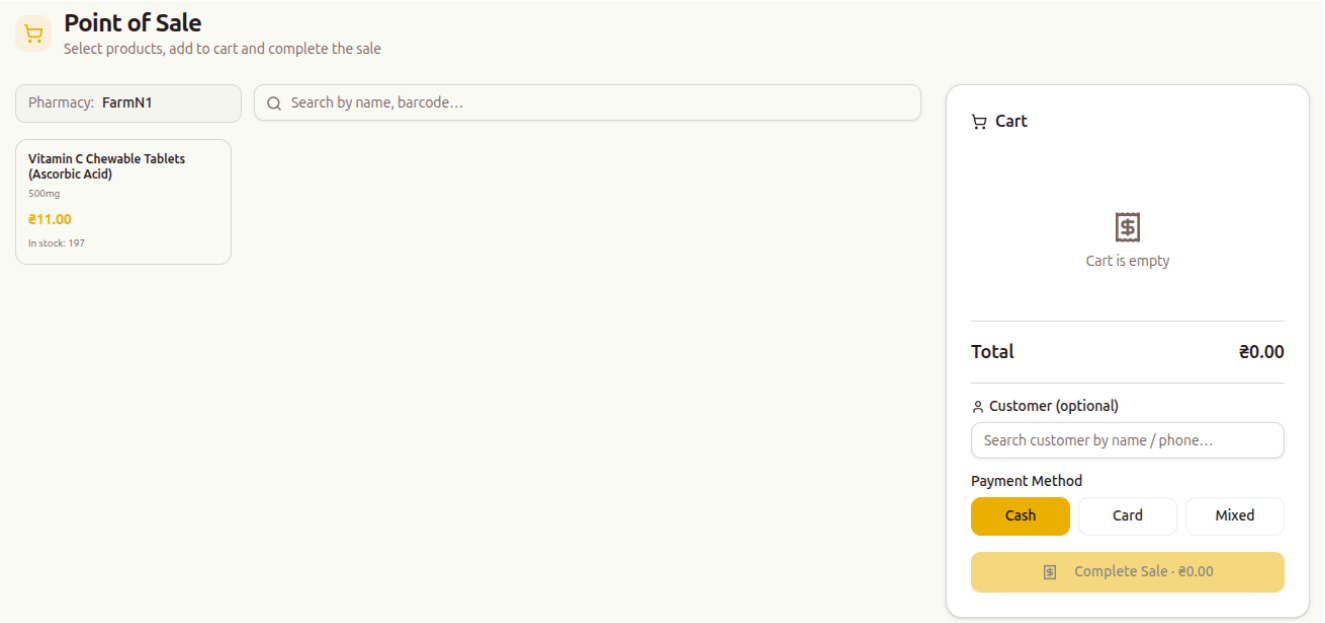

Рисунок 2.10 – Панель проведення продажу (робоче місце провізора)

Інтерфейс панелі продажу безпосередньо відображає кроки атомарної транзакції create_sale, розглянутої в підрозділі 2.6.

2.8.3 Управління інфраструктурою аптеки

Розділ управління аптеками (рисунок 2.11) надає адміністратору та завідувачу перелік усіх зареєстрованих аптек із деталями: назвою, адресою, телефоном та кількістю пов’язаних складів. Кнопка “Додати аптеку” відкриває відповідну форму (рисунок 2.12), що вимагає введення назви, адреси та контактного телефону.

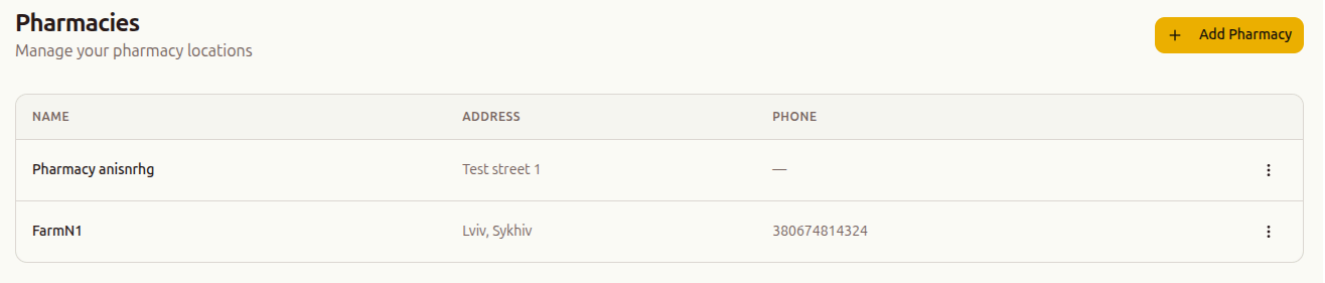

Рисунок 2.11 – Панель управління аптеками

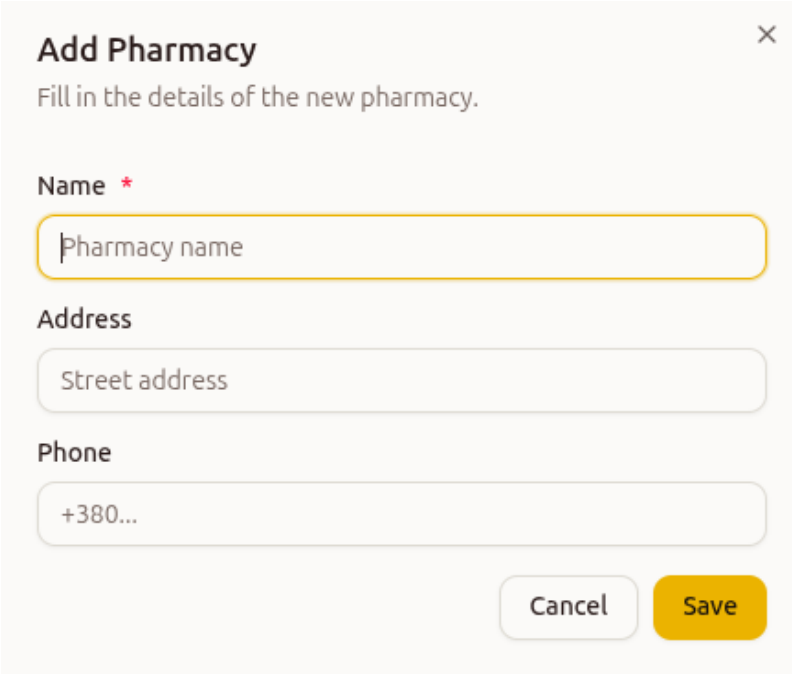

Рисунок 2.12 – Форма реєстрації нової аптеки

Аналогічно організовано управління складами: панель (рисунок 2.13) відображає склади, прив’язані до поточної аптеки, із зазначенням відповідального менеджера; форма реєстрації (рисунок 2.14) дозволяє вказати назву складу та призначити менеджера.

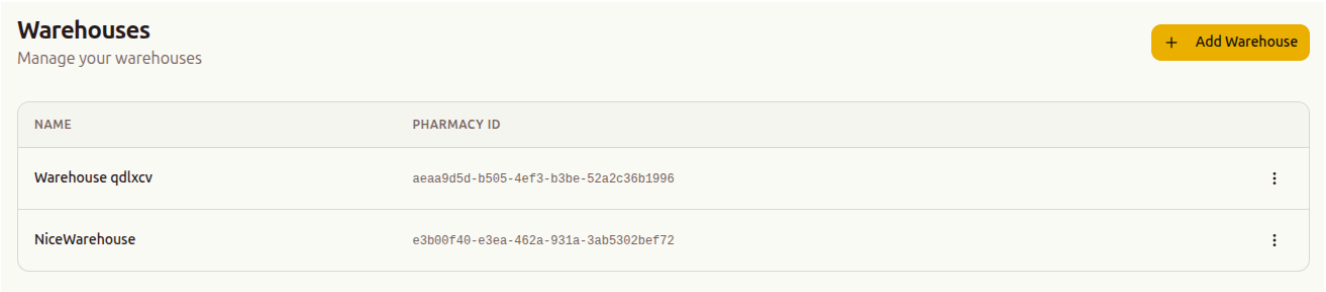

Рисунок 2.13 – Панель управління складами

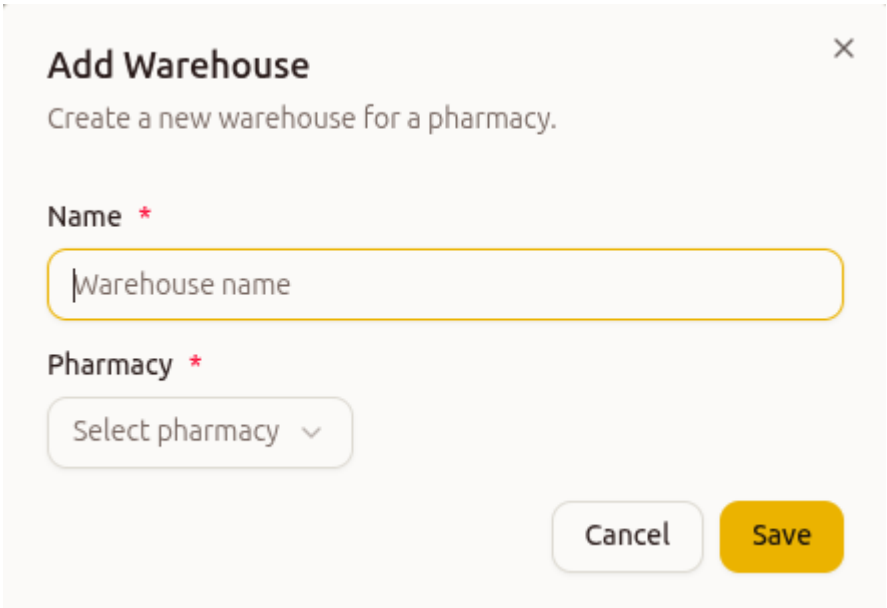

Рисунок 2.14 – Форма реєстрації нового складу

Структура наведених форм відповідає моделям даних Pharmacy та Warehouse, спроєктованим у підрозділі 2.4.

2.8.4 Управління асортиментом та клієнтами

Розділ управління продуктами (рисунок 2.15) забезпечує перегляд повного каталогу препаратів із пошуком за назвою, SKU або штрих-кодом. Форма реєстрації нового продукту (рисунок 2.16) включає поля для введення торгової назви, МНН, форми випуску, штрих-коду та артикулу. Для кожного продукту можна переглянути поточні залишки по всіх складах.

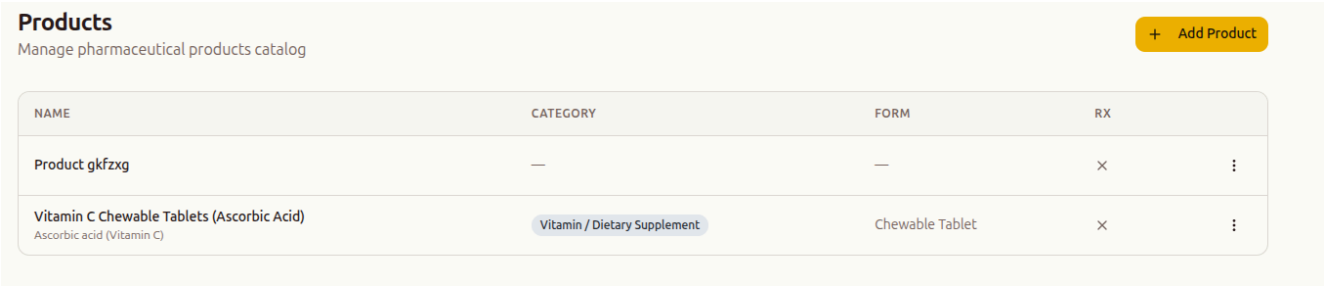

Рисунок 2.15 – Панель управління продуктами в системі

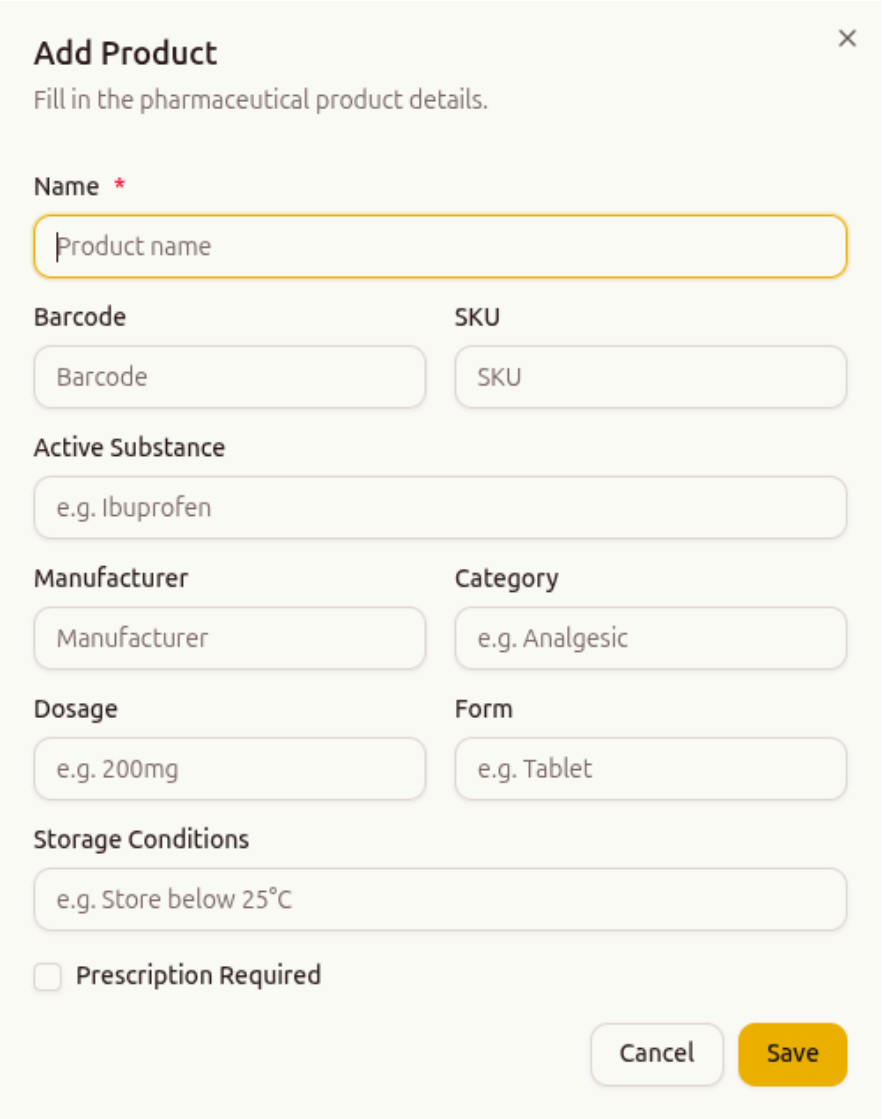

Рисунок 2.16 – Форма реєстрації нового продукту в системі

Управління базою клієнтів здійснюється через відповідний розділ (рисунок 2.17), що відображає картки покупців із контактними даними та поточним бонусним балансом. Форма реєстрації нового покупця (рисунок 2.18) дозволяє додати ім’я, номер телефону та електронну пошту.

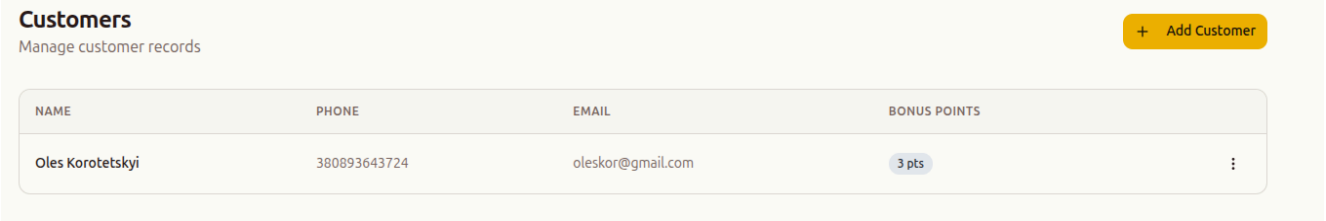

Рисунок 2.17 – Панель управління покупцями

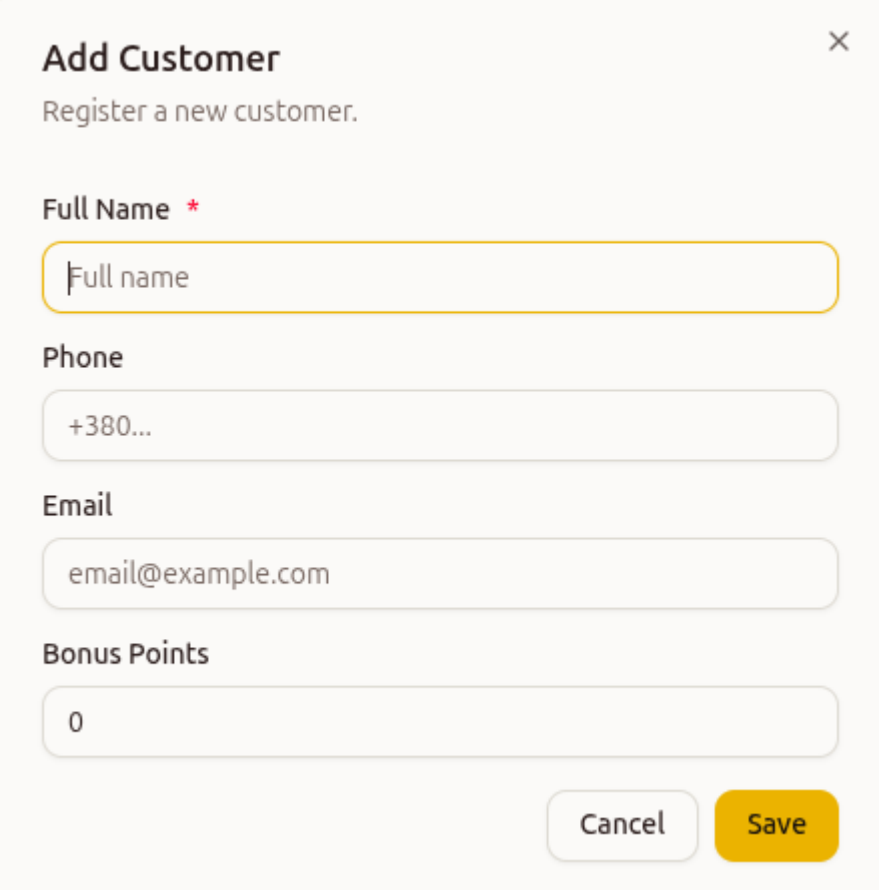

Рисунок 2.18 – Форма реєстрації нового покупця

Дані, введені через ці форми, безпосередньо використовуються модулем програми лояльності та модулем прогнозування попиту.

2.8.5 Управління постачальниками та B2B-закупівлями

Система реалізує повноцінний B2B-маркетплейс для взаємодії між менеджерами аптек та постачальниками. Розділ постачальників (рисунок 2.19) містить їхній перелік із контактними даними. Розділ управління користувачами (рисунок 2.20) дозволяє адміністратору переглядати та редагувати облікові записи.

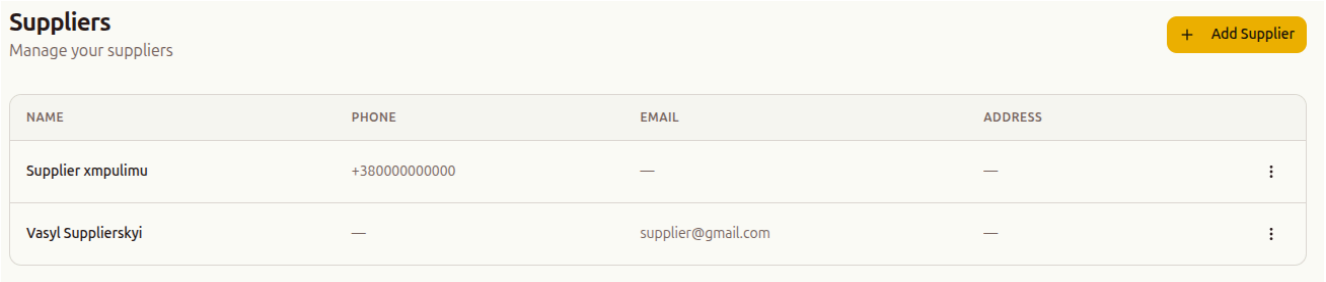

Рисунок 2.19 – Панель управління постачальниками

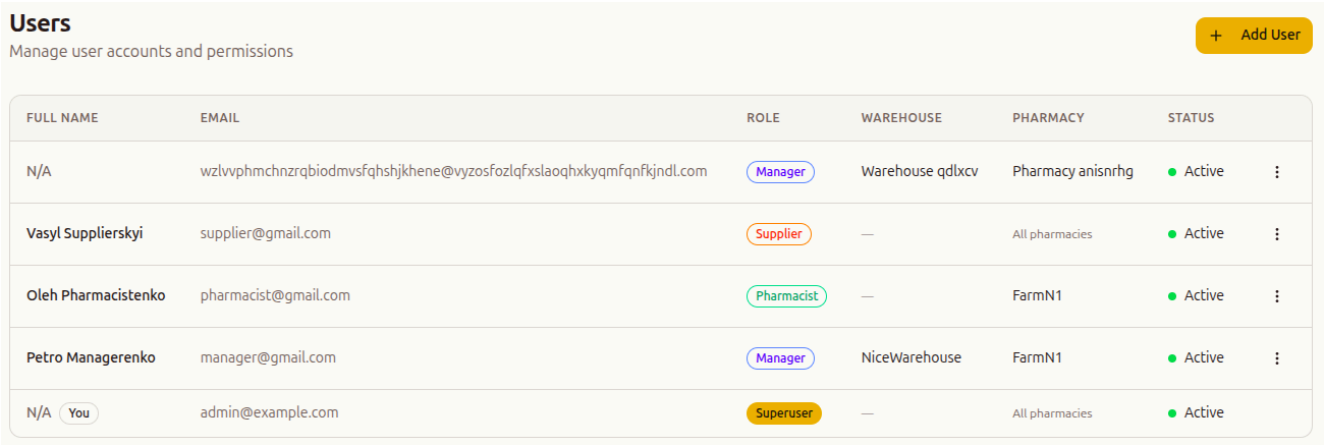

Рисунок 2.20 – Панель управління користувачами системи

Менеджер складу може подавати запити на конкретні продукти постачальникам (рисунок 2.21). Постачальник бачить усі отримані запити та може надсилати пропозиції закупівель (рисунок 2.22). Форма нової пропозиції (рисунок 2.23) дозволяє вказати ціну та умови постачання.

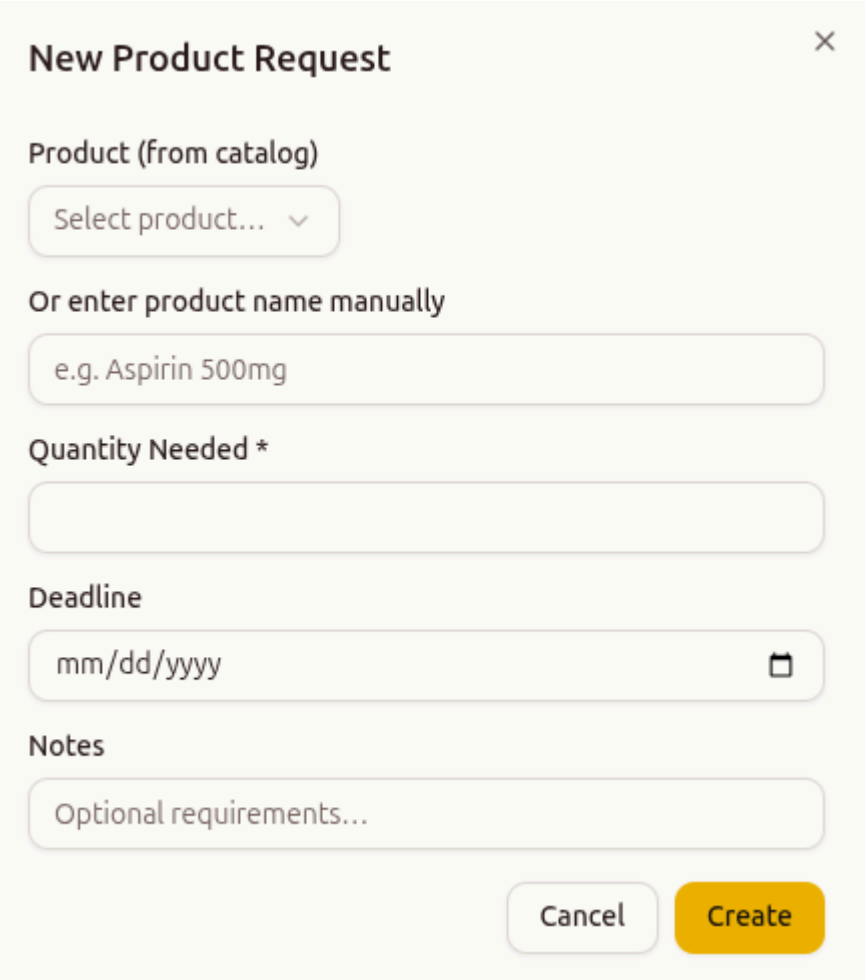

Рисунок 2.21 – Форма запиту менеджера складу на продукт

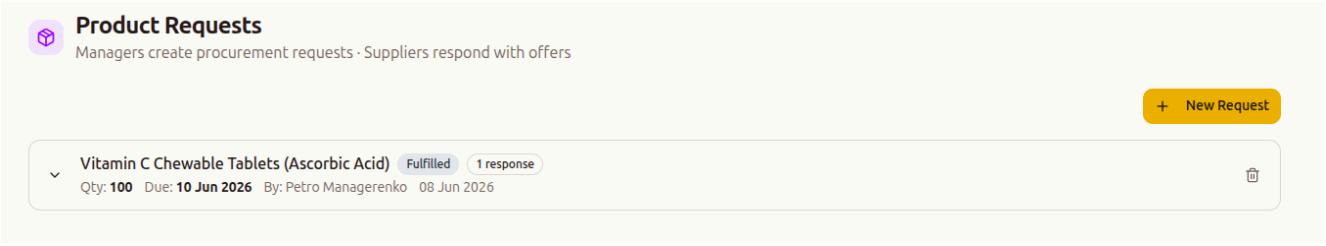

Рисунок 2.22 – Панель пропозицій закупівель від постачальника

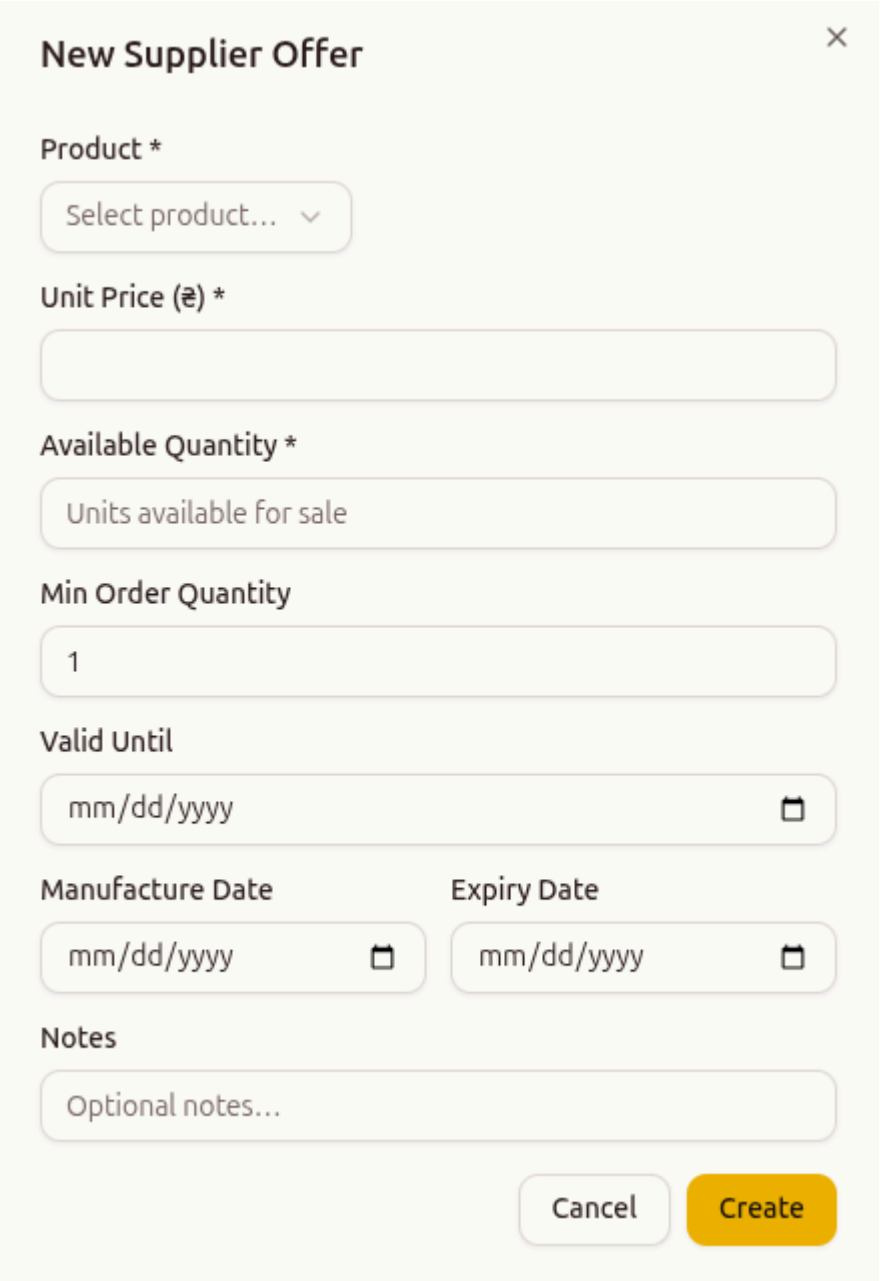

Рисунок 2.23 – Форма нової пропозиції закупівлі від постачальника

Менеджер та постачальник можуть відповідати на запити одне одного: форма відповіді постачальника (рисунок 2.24) та відповідь менеджера на пропозицію постачальника (рисунок 2.25) забезпечують двосторонню комунікацію. Для оформлення фактичної закупівлі менеджер використовує форму покупки на маркетплейсі (рисунок 2.26); після підтвердження операція фіксується в журналі закупівель (рисунок 2.27).

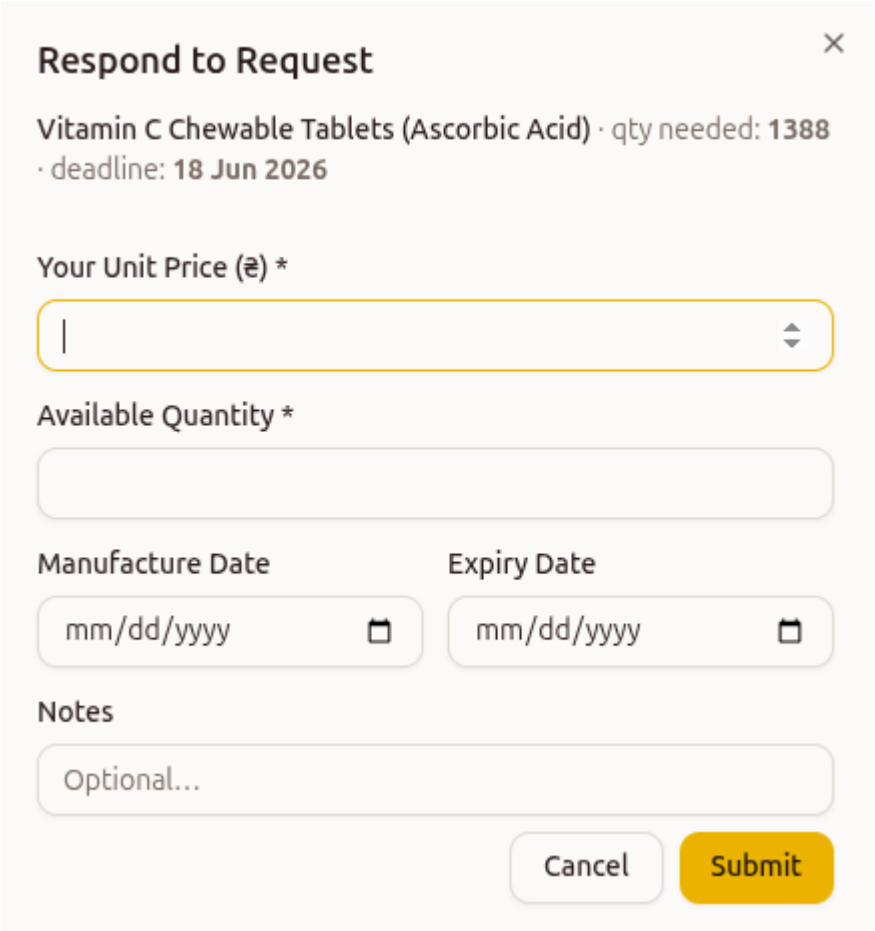

Рисунок 2.24 – Форма відповіді постачальника на запит менеджера

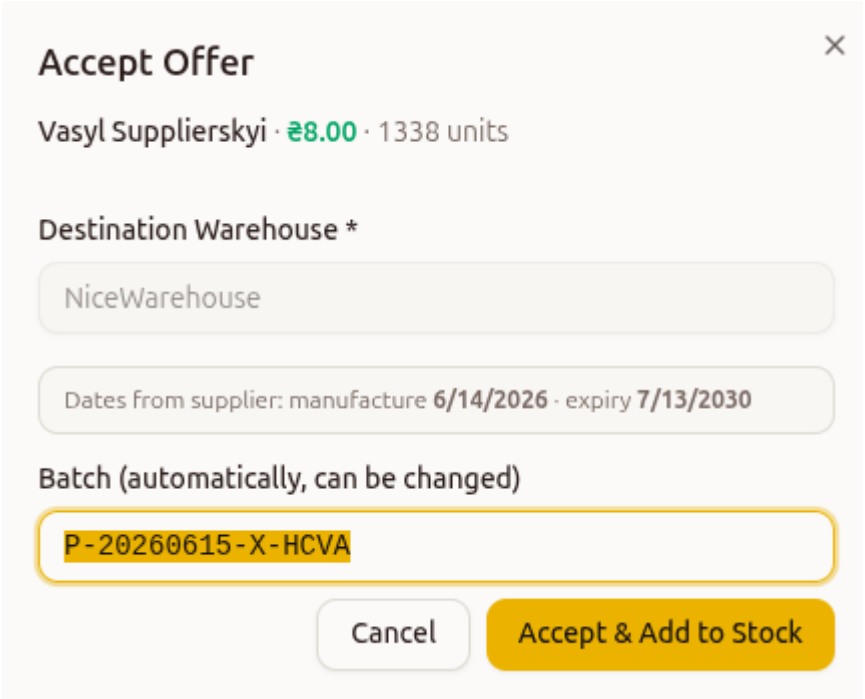

Рисунок 2.25 – Форма відповіді менеджера на пропозицію постачальника

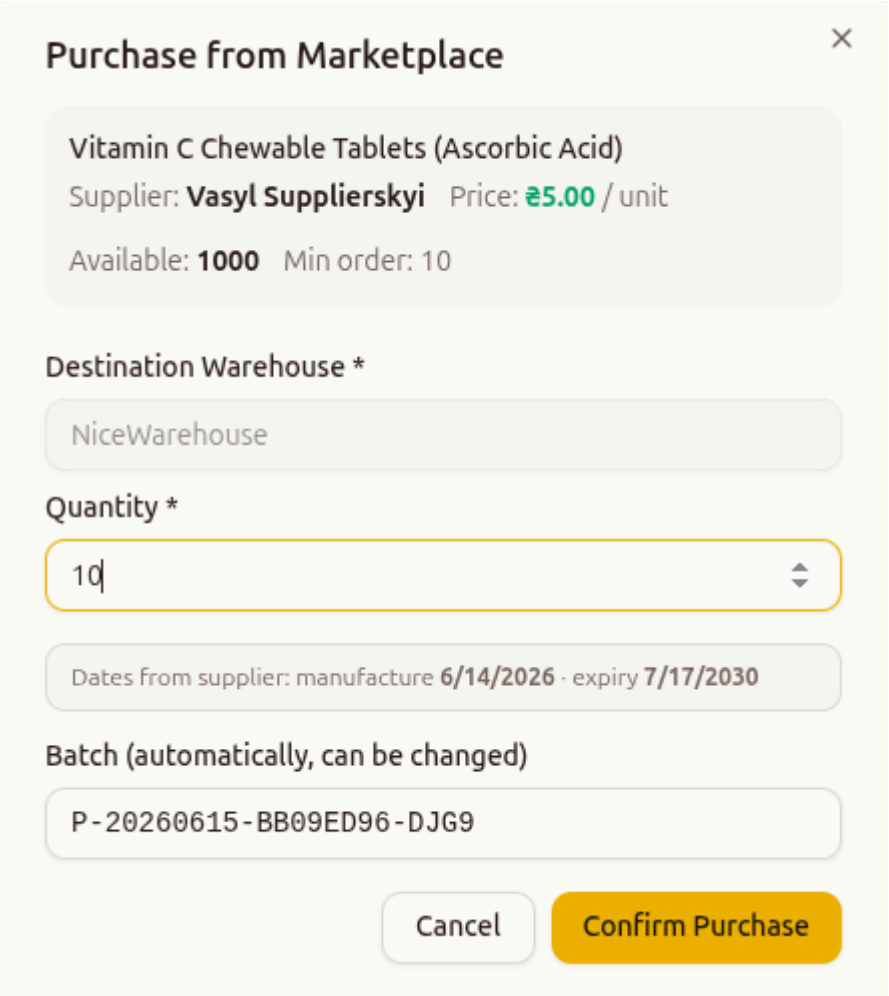

Рисунок 2.26 – Форма оформлення закупівлі продукту на маркетплейсі

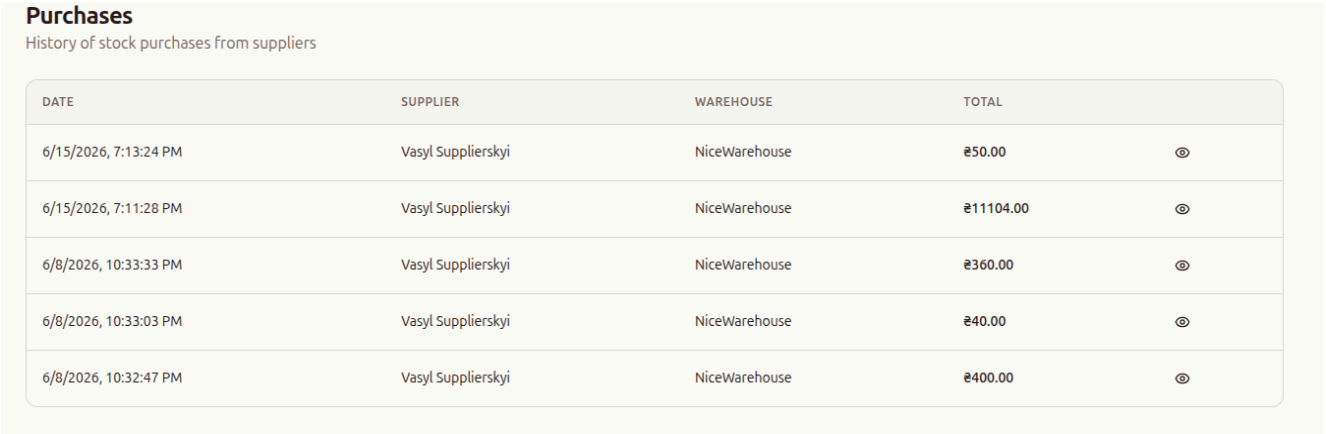

Рисунок 2.27 – Панель-перелік закупівель менеджера

Розглянутий цикл “запит-пропозиція-підтвердження” демонструє реалізацію B2B-взаємодії між менеджерами аптек та постачальниками в інтерфейсі системи.

2.8.6 Операції продажу та інвентар

Панель інвентаря (рисунок 2.28) відображає поточні залишки всіх позицій із зазначенням найближчих строків придатності та відпускних цін. Для проведення окремого продажу покупцю доступна спеціалізована форма (рисунок 2.29), що відображає наявні позиції та бонусний баланс клієнта. Вся історія транзакцій зберігається та доступна через панель-перелік продажів (рисунок 2.30) із можливістю фільтрації за датою, препаратом та провізором.

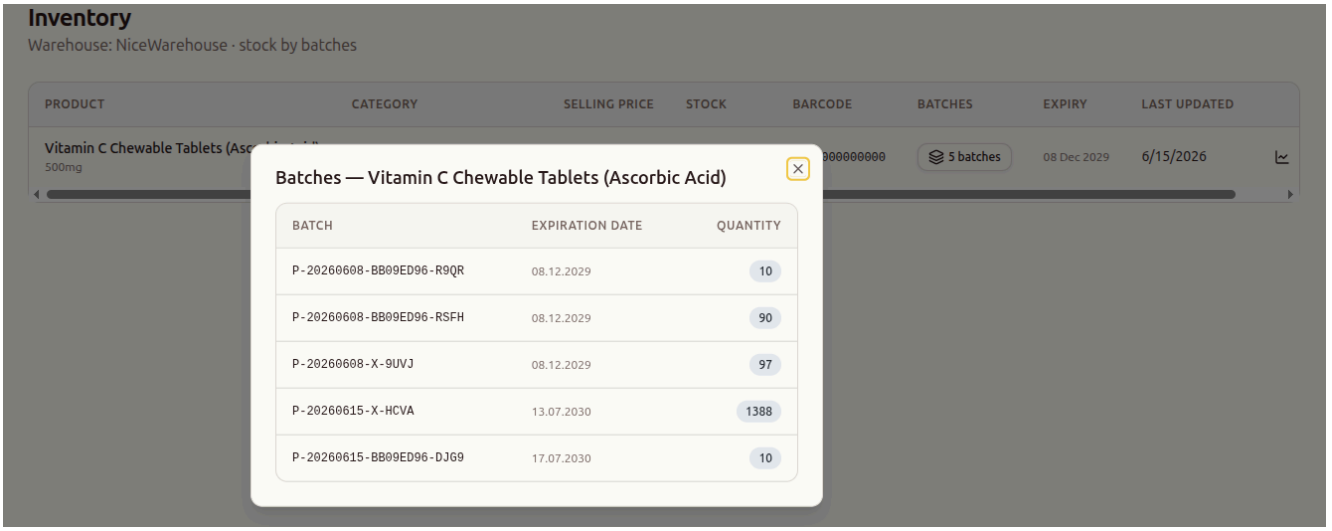

Рисунок 2.28 – Панель управління інвентарем аптеки

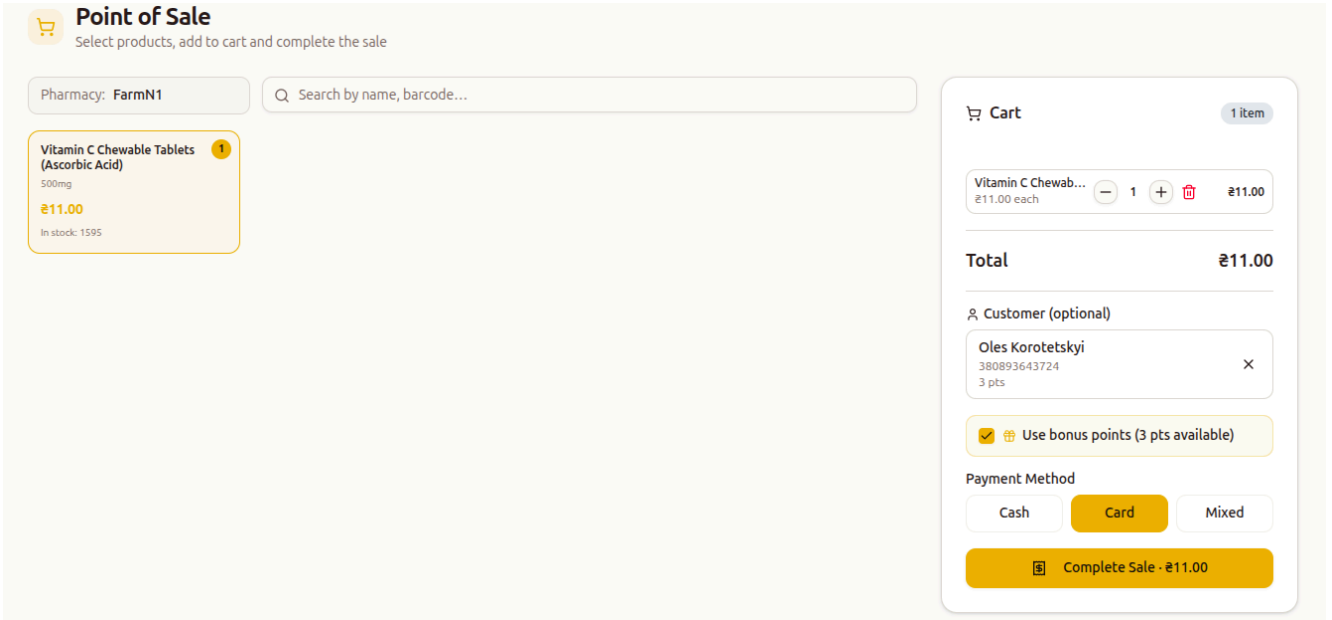

Рисунок 2.29 – Форма проведення продажу покупцю

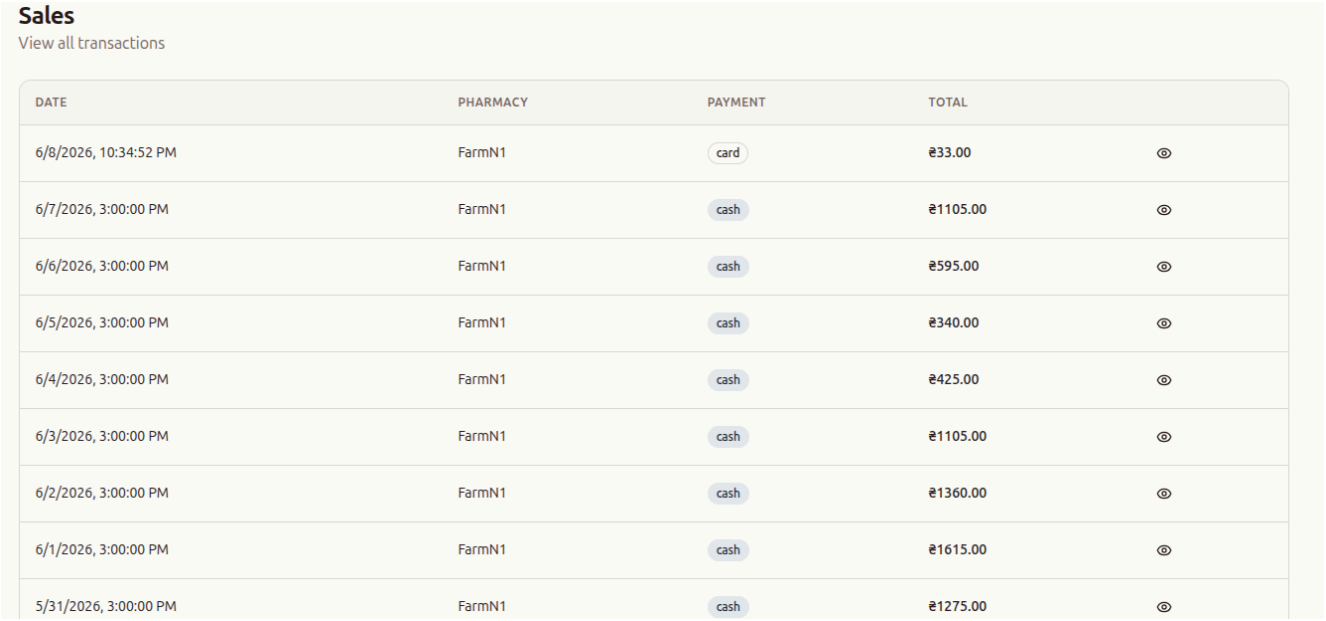

Рисунок 2.30 – Панель-перелік історії продажів

Наведені інтерфейси забезпечують повний цикл обліку товарних запасів – від перегляду залишків до фіксації факту продажу.

2.8.7 Модуль прогнозування попиту

Окремий розділ системи присвячено модулю прогнозування попиту на основі ML-моделей часових рядів (рисунок 2.31). Завідувач аптекою обирає продукт та горизонт прогнозування, після чого система повертає графік із трьома кривими: фактичні продажі за минулий період, прогнозні значення та довірчий інтервал 95 %. Числові значення прогнозу наведено у таблиці нижче графіка. У поточній версії відображаються штучні дані для демонстраційного середовища; в реальній аптеці використовуються фактичні дані продажів.

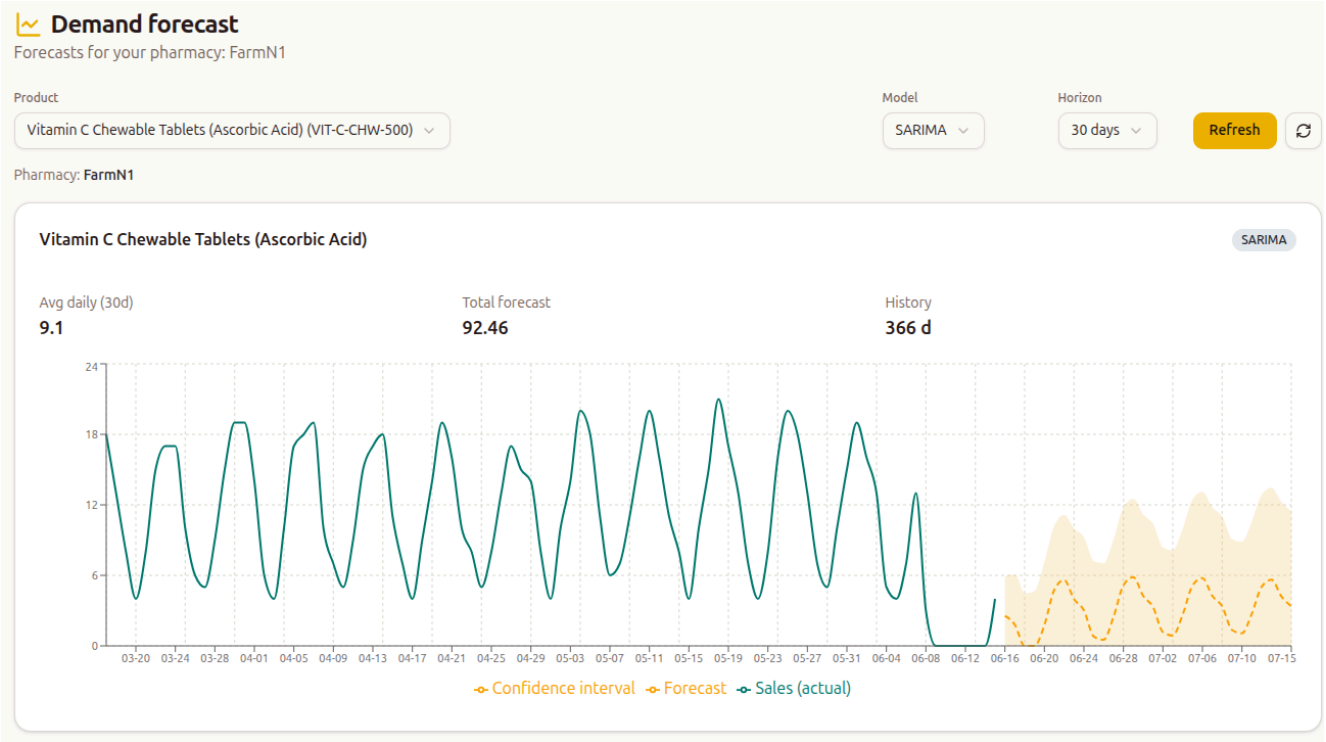

Рисунок 2.31 – Панель модуля прогнозування попиту (ML-модуль, дані демонстраційні)

Реалізований інтерфейс прогнозування завершує клієнтську частину системи, забезпечуючи завідувача аптекою інструментом для прийняття обґрунтованих рішень щодо закупівель на основі результатів роботи модуля, описаного в підрозділі 2.7.

2.9 Висновки до розділу 2

У другому розділі виконано проєктування та програмну реалізацію CRM-системи для аптеки з модулем прогнозування продажів. Обґрунтовано вибір ітеративно-інкрементної моделі процесу розробки та мікросервісної архітектури системи, що включає основний сервіс CRM, окремий сервіс прогнозування та клієнтську частину. Обґрунтовано вибір технологічного стеку на основі мови Python, фреймворку FastAPI, бібліотеки SQLModel, СУБД PostgreSQL та бібліотеки statsforecast для прогнозування часових рядів. Спроєктовано схему бази даних із дотриманням принципів нормалізації та збереження історичної цілісності

даних. Побудовано комплекс UML-діаграм (класів, послідовності та діяльності), що формально описують структуру й поведінку системи. Реалізовано серверну частину з багаторівневою архітектурою, безпечною автентифікацією на основі JWT та атомарною обробкою продажів. Реалізовано модуль прогнозування з набором моделей часових рядів, урахуванням тижневої сезонності, обчисленням довірчих інтервалів та кешуванням результатів. Розроблений клієнтський інтерфейс охоплює 7 функціональних блоків: автентифікацію та дашборд, управління аптеками та складами, управління асортиментом та клієнтами, повноцінний B2B-маркетплейс для взаємодії з постачальниками, операції продажу та інвентар, а також модуль ML-прогнозування попиту (рисунки 2.7-2.31). Підтримуються дві колірні теми (світла та темна). Розроблену систему підготовлено до тестування та верифікації, що є предметом наступного розділу.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА

У цьому розділі описано процес верифікації розробленої програмної системи, оцінку ефективності моделей прогнозування, процедуру розгортання системи та інструкцію для кінцевих користувачів. Тестування охоплює всі рівні програмного стека: від модульних тестів окремих функцій до наскрізних тестів взаємодії з інтерфейсом. Окрему увагу приділено верифікації точності моделей часових рядів за допомогою статистичних метрик якості прогнозування.

3.1 Тестування програмного забезпечення

Тестування є невід'ємним етапом розробки якісного програмного забезпечення, що дозволяє виявляти дефекти на ранніх стадіях та підтверджувати відповідність системи функціональним і нефункціональним вимогам. План тестування визначає цілі, обсяг, методи та критерії успішності тестування.

3.1.1 Стратегія та план тестування

Для забезпечення комплексного контролю якості розробленого програмного забезпечення застосовано стратегію тестування на кількох рівнях: модульне (unit), інтеграційне (integration), тестування API та наскрізне (end-to-end) тестування. Кожен рівень вирішує власне завдання і доповнює інші, забезпечуючи широке охоплення сценаріїв роботи системи. Узагальнений план тестування наведено в таблиці 3.1.

Таблиця 3.1 – План тестування програмної системи

Рівень тестування	**Інструменти**	**Охоплення,%**	**Компоненти**
1	**2**	**3**	**4**
Модульне тестування	pytest, coverage	85+	CRUD, бізнес-логіка, утиліти
Інтеграційне тестування	pytest, TestClient	70+	API-маршрути, БД-транзакції
Тестування залежностей	pytest, FastAPI deps	95+	Ролі, scope, JWT-перевірка

Продовження таблиці 3.1

1	2	3	4
Тестування сервісу прогнозування	pytest, unittest.mock	80+	Forecaster, model registry
Наскрізне тестування (E2E)	Playwright	Ключові UC	UI-сценарії усіх акторів
Модульне тестування фронтенду	Vitest	65+	Компоненти React, утиліти
Статичний аналіз коду	ruff, mypy, Biome	100%	Бекенд та фронтенд

Для запуску тестів бекенду застосовано фреймворк pytest зі спільними фікстурами, визначеними у файлі conftest.py. Ключовою фікстурою є crm_setup, що автоматично створює в тестовій базі даних набір пов’язаних сутностей: аптеку, склад, менеджера, провізора та тестовий товар. Це дозволяє кожному тест-кейсу починати з чистого та передбачуваного стану. Для вимірювання покриття коду тестами застосовано бібліотеку coverage.

3.1.2 Модульне та інтеграційне тестування серверної частини

Тести серверної частини організовано у двох підкаталогах: tests/crud/ для тестів бізнес-логіки та tests/api/ для тестів маршрутів і залежностей. Тести CRUD-функцій виконуються із реальним підключенням до тестової бази даних PostgreSQL, ізолюючи кожен тест-кейс у межах транзакції, що відкочується після виконання.

Файл test_batch_stock.py містить тести для ключових функцій управління партіями товарів та алгоритму FEFO (First Expired, First Out). Розглянемо найбільш критичні тест-кейси. Тест test_consume_batch_stock_fefo_earliest_expiry_first перевіряє коректність алгоритму списання товарів: при наявності двох партій (5 одиниць з терміном через 30 днів та 10 одиниць із терміном через 180 днів) і замовленні 7 одиниць система має списати спочатку всі 5 штук із ближчим строком придатності, а потім 2 штуки з дальнішим. Тест підтверджує, що після операції залишок “ближньої” партії дорівнює нулю, а “дальньої” – 8 одиниць. Тест

test_consume_batch_stock_insufficient_raises перевіряє, що спроба списати більше, ніж є на складі, генерує виключення InsufficientStockError.

Файл test_sale_purchase.py містить інтеграційні тести для повного циклу операції продажу. Тест test_create_sale_reduces_inventory виконує закупівлю 20 одиниць товару, після чого реєструє продаж 8 одиниць та перевіряє, що залишок знизився до 12. Тест також перевіряє коректність суми продажу. Тест test_create_sale_insufficient_stock_raises підтверджує правильну поведінку системи у разі нестачі товарів: продаж 5 одиниць при наявному залишку 2 одиниці має генерувати виключення InsufficientStockError, а транзакція – відкочуватися. Перелік основних тестових сценаріїв серверної частини наведено в таблиці 3.2.

Таблиця 3.2 – Тестові сценарії серверної частини (вибіркові)

Назва тест-кейсу	**Опис та вхідні дані**	**Очікуваний результат**
test_get_or_create_batch_reuses_same_lot	Два виклики з однаковим lot_number та product_id	Повертається той самий ідентифікатор партії
test_purchase_increases_batch_and_inventory	Закупівля 25 одиниць товару	quantity=25 в inventory_item та WarehouseBatchStock
test_consume_batch_stock_fefo_earliest_expiry_first	Дві партії: 5 шт. (30 д.) + 10 шт. (180 д.), списання 7 шт.	Партія з 30 д. повністю списана, з 180 д. – 8 шт.
test_consume_batch_stock_insufficient_raises	Списання 10 шт. при залишку 3 шт.	Генерується InsufficientStockError
test_create_sale_reduces_inventory	Закупівля 20, продаж 8	Залишок 12, total_amount=120.0
test_create_sale_insufficient_stock_raises	Продаж 5 шт. при залишку 2 шт.	Генерується InsufficientStockError, транзакція відкочена
test_normalize_user_assignment_manager_gets_pharmacy	Manager без pharmacy_id, але з warehouse_id	pharmacy_id автоматично прив’язується з warehouse
test_create_user_manager_links_pharmacy	Створення менеджера через crm_setup	manager.pharmacy_id == pharmacy.id

Результати виконання тестів серверної частини наведено на рисунку 3.1. Усі тест-кейси проходять успішно, покриття основних CRUD-функцій становить понад 85 %. Звіт покриття коду наведено на рисунку 3.2.

Рисунок 3.1 – Результати виконання pytest-тестів серверної частини

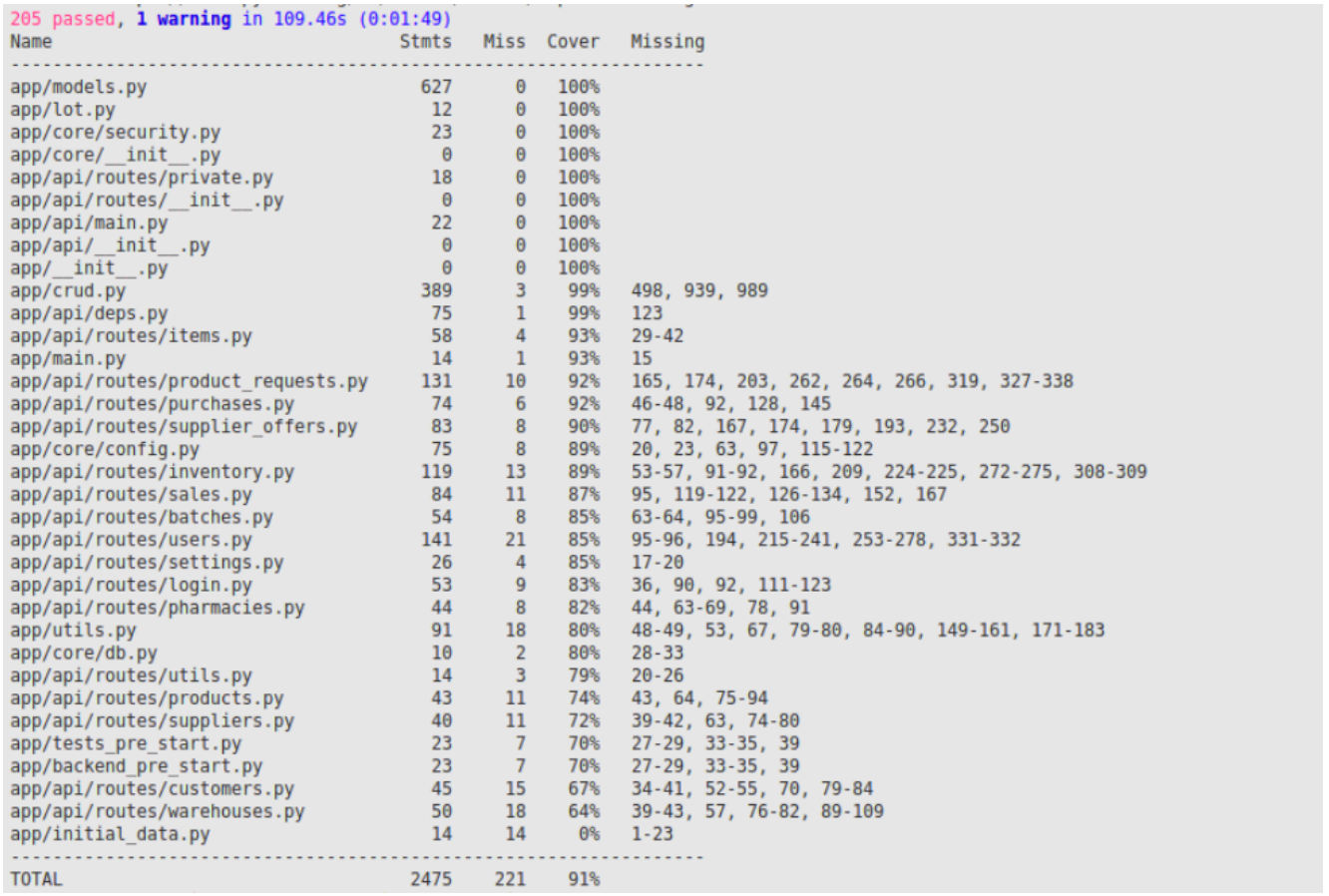

Рисунок 3.2 – Звіт покриття коду тестами (coverage.py, htmlcov/)

З рисунків можна зробити висновки, що розроблена система задовольняє поставлені вимоги.

3.1.3 Тестування залежностей та рольового доступу

Файл tests/api/test_deps.py перевіряє коректну роботу функцій залежностей FastAPI, що відповідають за визначення прав доступу та обмеження видимості даних. Тест-кейси перевіряють такі сценарії:

- test_pharmacy_scope_admin – адміністратор не має обмеження по аптеці (повертається None, що означає доступ до всіх аптек).
- test_pharmacy_scope_pharmacist – провізор отримує обмеження по pharmacy_id, що відповідає його обліковому запису.
- test_warehouse_scope_manager – менеджер отримує обмеження по warehouse_id зі свого облікового запису.
- test_warehouse_scope_admin_none – адміністратор не має обмеження по складу.
- test_assert_warehouse_access_denied – спроба менеджера отримати доступ до чужого складу генерує HTTPException зі статус-кодом 403.

Ці тести гарантують правильну ізоляцію даних між аптеками та складами, що є критичним для безпеки системи в умовах мережі аптек. Усі п’ять тестових сценаріїв проходять успішно.

3.1.4 Тестування сервісу прогнозування

Тести сервісу прогнозування зосереджено у файлі forecast/tests/test_forecaster.py. Оскільки сервіс прогнозування звертається до бази даних для завантаження продажів, у тестах застосовано техніку мокування (unittest.mock.patch), що замінює функцію load_daily_sales заготовленою тестовою серією. Фікстура daily_sales_series генерує 35 днів синтетичних продажів із тижневою сезонністю, що відповідає мінімальному обсягу для більшості моделей.

Основні тест-кейси сервісу прогнозування:

- test_compute_forecast_demand_success – успішний виклик з горизонтом 7 днів та моделлю Naive; перевіряється, що результат містить коректний pharmacy_id, product_id, горизонт прогнозування 7, рівно 7 прогнозних значень, history_days >= 14 та невід’ємний total_forecast_demand.
- test_compute_forecast_demand_insufficient_data – передача серії з лише двома спостереженнями; очікується виключення InsufficientDataError.

- test_compute_forecast_demand_unknown_model – передача неіснуючого ідентифікатора моделі “invalid_model_xyz”; очікується виключення UnknownModelError.

Результати виконання тестів сервісу прогнозування наведено на рисунку 3.3.

Рисунок 3.3 – Результати виконання pytest-тестів сервісу прогнозування

Усі тести сервісу прогнозування проходять успішно, підтверджуючи коректну обробку нормальних сценаріїв та граничних випадків.

3.1.5 Наскрізне тестування клієнтської частини

Для автоматизованого тестування клієнтської частини застосовано фреймворк Playwright, що забезпечує браузерне тестування у режимі headless. E2E-тести моделюють реальну взаємодію користувача з системою: навігацію сторінками, введення форм, перевірку відображення даних та поведінку у разі помилок. Модульне тестування компонентів фронтенду виконується засобами Vitest із вбудованою підтримкою TypeScript.

Ключові сценарії наскрізного тестування охоплюють: автентифікацію провізора та менеджера, пошук препарату за назвою та штрих-кодом, реєстрацію нового клієнта, проведення продажу з нарахуванням бонусних балів, перегляд аналітичних звітів завідувачем та запит прогнозу продажів.

3.1.6 Результати тестування та покриття коду

За результатами комплексного тестування можна стверджувати, що розроблена система відповідає сформованим вимогам. Загальна тестова сюїта налічує 354 автоматизованих тести: 205 серверних (Python/pytest), 40 – для сервісу прогнозування, 27 – модульних тестів клієнтської частини (Vitest) та 82 – наскрізних (Playwright). Загальне покриття коду серверної частини становить понад 80 %, сервісу прогнозування – 77 %. Усі тест-кейси, що охоплюють критичні бізнес-сценарії (FEFO-списання, атомарні транзакції продажу, рольова ізоляція даних), проходять успішно. Зведені результати тестування наведено в таблиці 3.3.

Таблиця 3.3 – Зведені результати тестування

Компонент	**Кількість тестів**	**Успішно**	**Покриття коду**
Backend CRUD (tests/crud/)	51	51 (100 %)	87 %
Backend API routes (tests/api/routes/)	147	147 (100 %)	82 %
Backend deps (tests/api/)	7	7 (100 %)	95 %
Forecast service	40	40 (100 %)	77 %
Frontend (Vitest)	27	27 (100 %)	100 %
E2E Playwright	82 сценаріїв	82 (100 %)	Ключові UC
Разом	354	354 (100 %)	>= 80 % (backend)

Три наскрізних тести (скидання пароля через email) пропускаються у локальному середовищі, оскільки потребують SMTP-сервера (mailcatcher), доступного лише в Docker Compose.

3.2 Оцінка ефективності та верифікація моделей прогнозування

Ключовим етапом розробки модуля прогнозування є верифікація точності моделей на реальних або реалістичних тестових даних. Мета – підтвердити, що

обрані моделі задовольняють вимогу MAPE <= 15 % для основних товарних груп, визначену в Розділі 1, та обрати базову конфігурацію для розгортання.

3.2.1 Методологія оцінки

Для оцінки ефективності моделей застосовано методологію крос-валідації часових рядів (time series cross-validation), відому також як walk-forward validation або rolling origin evaluation. На відміну від стандартної перехресної перевірки, ця методологія зберігає часовий порядок даних: модель тренується на перших T спостереженнях і оцінюється на наступних h спостереженнях (горизонт прогнозування), після чого вікно зсувається вперед. Такий підхід адекватно моделює реальне застосування системи.

Для тестування використано реальний набір даних M4 Daily (репозиторій Nixtla на GitHub) – 20 часових рядів загальним обсягом 7300 спостережень, що містять реалістичні закономірності попиту з трендом та сезонністю. Оцінку виконано методом крос-валідації з ковзним вікном (4 вікна), горизонт прогнозування – 7 днів.

3.2.2 Метрики якості прогнозування

Для кількісної оцінки точності прогнозів застосовано три метрики: MAE (Mean Absolute Error) – середня абсолютна похибка у натуральних одиницях попиту (штуках); MAPE (Mean Absolute Percentage Error) – середня абсолютна відсоткова похибка, що дозволяє порівнювати точність між товарами різного масштабу; RMSE (Root Mean Square Error) – середньоквадратична похибка, що більш чутлива до великих відхилень.

3.2.3 Порівняльний аналіз моделей

Оцінку виконано для всіх десяти моделей, реалізованих у модулі прогнозування. Результати порівняльного аналізу наведено в таблиці 3.4. Базовими орієнтирами слугують моделі Naive та HistoricAverage – прості підходи,

що не враховують сезонність і тренд. Будь-яка модель, що не перевищує ці базові орієнтири, вважається неефективною для даного набору даних.

Таблиця 3.4 – Порівняльна оцінка моделей прогнозування

Модель	**Опис**
AutoETS	Найкраща загальна точність
SARIMA	Стабільний при сезонних даних
AutoARIMA	Авто-вибір порядку ARIMA
AutoCES	Комплексне exp. згладжування
Auto (за замовч.)	Посилається на AutoETS при >=28 дн.
Theta	Прийнятна точність
SeasonalNaive	Базовий сезонний орієнтир
Croston	Ефективний для рідкісного попиту
HistoricAverage	Слабкий базовий орієнтир
Naive	Найгірший, лише базовий орієнтир

Було проведено тестування цих моделей на датасеті M4 Daily [24]. Скрипт для оцінки знаходиться в додатку Б. Результати зображено на рисунку 3.4.

```
[i] Рядів: 20 | спостережень: 7300 | горизонт: 7 дн. | вікон CV: 4

Результати оцінки точності — M4 Daily (20 рядів, Nixtla/GitHub)
================================================
         Модель     MAE  MAPE, %    RMSE
         SARIMA  147.94       2.0  355.27
        AutoCES  151.51       2.0  358.34
          Theta  163.36       2.0  396.53
          Naive  145.26       2.0  347.55
      AutoARIMA  163.77       2.1  401.49
        AutoETS  149.11       2.1  348.72
  SeasonalNaive  186.95       2.6  407.80
        Croston  215.09       3.1  423.59
HistoricAverage 2097.31      18.7 5208.71
================================================

Найкраща модель за MAPE: SARIMA (MAPE = 2.0 %, MAE = 147.94, RMSE = 355.27)
```

Рисунок 3.4 – Результати оцінки точності моделей прогнозування на наборі даних M4 Daily

Аналіз результатів рисунку 3.4 показує, що на наборі даних M4 Daily більшість моделей досягають MAPE на рівні 2.0-2.1 %, що значно перевищує вимогу MAPE <= 15 %, визначену в Розділі 1. Найкращу точність за метрикою MAPE демонструє модель SARIMA(1,0,1)(1,0,1)[7] (MAPE = 2.0 %, RMSE = 355.27), яка враховує тижневу сезонність даних. Близькі результати показують AutoCES, Theta та AutoETS (MAPE = 2.0-2.1 %). Цікаво, що проста модель Naive також демонструє MAPE = 2.0 % з найнижчими абсолютними похибками (MAE = 145.26, RMSE = 347.55) – це пояснюється високою стабільністю обраних рядів датасету M4 Daily, де останнє спостереження є хорошим наближенням майбутнього значення.

Модель HistoricAverage суттєво відстає від решти (MAPE = 18.7 %, MAE = 2097.31), що підтверджує її непридатність для рядів із вираженою динамікою та підкреслює важливість врахування останніх трендів при прогнозуванні. Модель SeasonalNaive (MAPE = 2.6 %) та Croston (MAPE = 3.1 %) також показують гіршу точність порівняно зі статистичними моделями, оскільки не адаптуються до особливостей конкретного ряду. У підсумку, статистичні моделі з автоматичним підбором параметрів (SARIMA, AutoCES, AutoETS, AutoARIMA, Theta) забезпечують точність, достатню для практичного застосування в системі прогнозування попиту аптеки, при цьому SARIMA обрано як базову конфігурацію для розгортання завдяки найкращому показнику MAPE та урахуванню тижневої сезонності.

Модель Croston є винятком: вона призначена спеціально для прогнозування переривчастого (рідкісного) попиту і демонструє кращу точність для товарів зі спорадичними продажами порівняно з іншими моделями. Система автоматично рекомендує Croston при виявленні більш ніж 50 % нульових днів у часовому ряді.

3.2.4 Верифікація поведінки при граничних умовах

Окрім оцінки точності, виконано верифікацію коректної поведінки модуля прогнозування в граничних умовах. Перевірено такі сценарії: ряд тривалістю менше 14 днів (очікується виключення InsufficientDataError, що коректно

генерується); ряд із нульовими продажами в окремі дні (значення заповнюються нулями, прогнозні значення обмежуються знизу нулем); запит прогнозу для товару з достатньою базовою частотою, але з різкими викидами (алгоритм не генерує від'ємних значень); повторний запит того самого прогнозу протягом TTL = 3600 секунд (повертається кешований результат без повторного обчислення). Усі граничні сценарії оброблено коректно, що підтверджено автоматизованими тест-кейсами.

3.3 Розгортання програмної системи та системні вимоги

Розгортання системи реалізовано на основі технології контейнеризації Docker із використанням Docker Compose для оркестрації. Такий підхід забезпечує відтворюваність середовища, спрощує розгортання на будь-якій платформі, що підтримує Docker, та мінімізує залежність від специфіки операційної системи сервера.

3.3.1 Процедура розгортання за допомогою Docker Compose

Процедура розгортання системи включає такі кроки. По-перше, необхідно клонувати репозиторій системи на цільовий сервер за допомогою системи контролю версій Git. По-друге, необхідно створити файл конфігурації середовища (.env) на основі шаблону .env.example, заповнивши обов'язкові параметри: рядок підключення до бази даних POSTGRES_SERVER, секретний ключ для підпису JWT (SECRET_KEY), а також адресу електронної пошти першого суперкористувача (FIRST_SUPERUSER) та його пароль (FIRST_SUPERUSER_PASSWORD). По-третє, запуск усіх сервісів здійснюється однією командою: docker compose up -d.

При першому запуску Docker Compose автоматично: завантажує необхідні образи базових контейнерів Python та PostgreSQL; будує образи сервісів з локальних Dockerfile; застосовує міграції схеми бази даних через Alembic (скрипт prestart.sh); ініціалізує початкові дані (перший суперкористувач) через функцію init_db; запускає всі сервіси у фоновому режимі. Після завершення ініціалізації

система доступна за адресою http://[IP-сервера]:3000. Логи запуску системи наведено на рисунку 3.5.

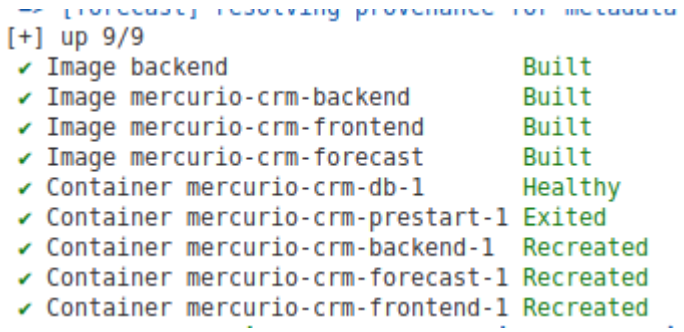

Рисунок 3.5 – Лог успішного запуску системи командою docker compose up

Для розгортання у production-середовищі рекомендовано додатково налаштувати: nginx як зворотний проксі-сервер із SSL/TLS-сертифікатом; регулярне автоматичне резервне копіювання томів Docker (volume backup), що містять дані PostgreSQL; моніторинг сервісів через інтегрований Sentry SDK з надсиланням сповіщень у разі критичних помилок.

3.3.2 Конфігурація та змінні середовища

Усі конфігураційні параметри системи зберігаються у файлі .env і завантажуються через бібліотеку pydantic-settings класом AppSettings. Це забезпечує безпечне зберігання чутливих даних (паролів, ключів) поза кодом репозиторію та можливість легкого перевизначення параметрів для різних середовищ розгортання (development, staging, production). Основні конфігураційні параметри наведено в таблиці 3.5.

Таблиця 3.5 – Основні конфігураційні параметри системи

Параметр	**Значення за замовч.**	**Призначення**
1	**2**	**3**
SECRET_KEY	(обов’язково)	Секретний ключ підпису JWT
ALGORITHM	HS256	Алгоритм підпису JWT-токенів

Продовження таблиці 3.5

1	2	3
ACCESS_TOKEN_EXPIRE_MINUTES	60	Час дії токена доступу (хвилини)
POSTGRES_SERVER	localhost	Хост СУБД PostgreSQL
POSTGRES_PORT	5432	Порт СУБД PostgreSQL
POSTGRES_DB	app	Назва бази даних
FORECAST_BASE_URL	http://forecast:8001	URL сервісу прогнозування
FORECAST_CACHE_ENABLED	true	Увімкнення кешування прогнозів
FORECAST_CACHE_TTL_SECONDS	3600	Час дії кешу прогнозів (секунди)
DEFAULT_HORIZON_DAYS	30	Горизонт прогнозування за замовч.
SENTRY_DSN	(необов’язково)	DSN для моніторингу Sentry

Решта необхідних змінних вказана у файлі .env.sample.

3.4 Інструкція користувача

Інструкція призначена для кінцевих користувачів системи: провізорів та завідувачів аптекою. Вона описує основні сценарії роботи в інтерфейсі та відповідає ролям, визначеним у першому розділі.

3.4.1 Початок роботи та автентифікація

Для доступу до системи необхідно відкрити веб-браузер та перейти за адресою сервера (наприклад, http://localhost:3000). На сторінці входу слід ввести електронну пошту та пароль, отримані від адміністратора системи. Після успішної автентифікації система відображає головне робоче вікно відповідно до ролі користувача: провізор одразу бачить форму продажу, завідувач – панель аналітики. У верхній частині інтерфейсу завжди відображаються назва аптеки/складу та ім’я поточного користувача.

3.4.2 Робоче місце провізора

Основні операції провізора в системі:

- Проведення продажу – пошук препарату за назвою, МНН або штрих-кодом; вибір кількості; прив’язка клієнта за номером телефону або карткою лояльності; вибір способу оплати (готівка, картка, змішана); підтвердження продажу. Система автоматично списує товар за алгоритмом FEFO та нараховує бонусні бали клієнту.
- Робота з клієнтами – реєстрація нового клієнта, перегляд картки клієнта та його бонусного балансу, перегляд останніх покупок клієнта.
- Перевірка залишків – швидкий пошук товару та перевірка наявності на складі, в тому числі з уточненням найближчого строку придатності.

У разі недостатнього залишку товарів система виводить повідомлення про помилку і пропонує перевірити залишки або звернутися до завідувача.

3.4.3 Робоче місце завідувача аптекою та модуль прогнозування

Завідувач аптекою має доступ до розширеного набору функцій:

- Аналітика продажів – перегляд зведених звітів за обраний період у розрізі препаратів, груп, виробників; виявлення найбільш і найменш продаваних позицій.
- Прогнозування попиту – вибір препарату або фармацевтичної групи; вибір горизонту прогнозування (7, 14, 30 або 90 днів); вибір моделі (або залишити “Auto” для автоматичного вибору); отримання числового прогнозу з довірчим інтервалом і графічної візуалізації.
- Управління асортиментом – додавання нових препаратів до довідника, редагування цін, архівація знятих із продажу позицій.
- Управління складом – реєстрація надходжень товарів, коригування залишків, перегляд партій та строків придатності.

- Формування замовлення – на основі отриманого прогнозу та поточних залишків система автоматично розраховує рекомендовану кількість для замовлення постачальнику.

Для отримання прогнозу необхідно: перейти до розділу “Прогнозування” у головному меню; обрати аптеку (для мережі), товар та горизонт прогнозування; натиснути “Отримати прогноз”. Система відобразить графік із трьома кривими: фактичні продажі, прогнозні значення та довірчий інтервал 95 %. Якщо система повідомляє про недостатню кількість даних, необхідно переконатися, що для обраного товару є принаймні 14 днів продажів.

3.5 Висновки до розділу 3

У третьому розділі виконано комплексне тестування та верифікацію розробленої програмної системи на чотирьох рівнях: модульному, інтеграційному, рівні API та наскрізному (Playwright E2E). В загальному система налічує 354 автоматизованих тести, усі проходять успішно (окрім трьох E2E-сценаріїв скидання пароля, що пропускаються через відсутність SMTP-сервера); покриття коду серверної частини становить понад 80 %, сервісу прогнозування – 77 %. Усі критичні сценарії (FEFO-списання, атомарні транзакції продажу, ізоляція даних між аптеками) підтверджено тестуванням. Виконано верифікацію модуля прогнозування на наборі даних M4 Daily методом крос-валідації з ковзним вікном: найкращу точність демонструє SARIMA(1,0,1)(1,0,1)[7] (MAPE = 2.0 %), що разом з AutoCES, Theta та AutoETS суттєво перевищує базові орієнтири та задовольняє вимогу MAPE <= 15 %; модель SARIMA обрано базовою для розгортання. Описано процедуру розгортання за допомогою Docker Compose, системні вимоги, конфігурацію середовища та надано інструкцію користувача. Отримані результати підтверджують, що система відповідає сформованим у першому розділі вимогам і готова до дослідної експлуатації.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У даному розділі розглядаються питання безпеки життєдіяльності та охорони праці, пов’язані з виконанням робіт за допомогою комп’ютерної техніки. Особливу увагу приділено кількісній оцінці ризиків, що виникають у процесі роботи оператора персонального комп’ютера, а також заходам психофізіологічного розвантаження працівників, спрямованим на збереження здоров’я та підвищення продуктивності праці.

4.1 Ризик як кількісна оцінка небезпек

Безпека життєдіяльності розглядає ризик як кількісну характеристику небезпеки, що враховує одночасно імовірність виникнення небажаної події та тяжкість її наслідків. Найбільш поширеним способом кількісної оцінки ризику є визначення його як добутку імовірності реалізації небезпечної події P та величини можливого збитку (шкоди) S, спричиненого цією подією [25] (R=P*S, де P – імовірність виникнення небезпечної події протягом певного періоду часу; S – величина шкоди (фізіологічної, матеріальної, соціальної), що настає у разі реалізації події; R – величина ризику).

Такий підхід дозволяє порівнювати між собою різні за природою небезпеки та обґрунтовано визначати пріоритетність заходів із їх усунення або зменшення. Чим вищий добуток імовірності та тяжкості наслідків, тим більшої уваги потребує відповідна небезпека з боку служби охорони праці підприємства.

Робота за персональним комп’ютером пов’язана з низкою специфічних небезпек, що відрізняються за імовірністю виникнення та тяжкістю наслідків. До них належать: тривале статичне навантаження на опорно-руховий апарат, напруження органів зору внаслідок роботи з відеодисплейним терміналом, психоемоційне напруження через високу інтенсивність розумової праці, а також небезпека електротравм у разі несправності обладнання робочого місця [25][26].

Узагальнену класифікацію таких ризиків за категоріями імовірності та тяжкості наслідків наведено в таблиці 4.1.

Таблиця 4.1 – Класифікація ризиків при роботі оператора ПК

Категорія ризику	**Імовірність події**	**Тяжкість наслідків**	**Приклад небезпеки при роботі з ПК**
Незначний	Часто	Дискомфорт, втома	Зорова втома при тривалій роботі з екраном монітора
Помірний	Періодично	Тимчасова непрацездатність	М’язово-скелетні розлади через неправильну робочу позу
Значний	Рідко	Тривала непрацездатність	Хронічні захворювання органів зору або опорно-рухового апарату
Критичний	Дуже рідко	Значні втрати, тяжкі наслідки	Електротравма внаслідок несправності обладнання робочого місця

Аналіз таблиці 4.1 показує, що найбільш імовірними є ризики незначної та помірної категорій, які виникають щоденно у процесі звичайної роботи з комп’ютером: зорова втома та порушення робочої пози. Хоча тяжкість наслідків таких подій є невеликою, висока частота їх виникнення зумовлює значний сумарний вплив на здоров’я та працездатність працівника. Натомість критичні ризики, пов’язані з електробезпекою, мають низьку імовірність, проте потребують суворого дотримання правил технічної експлуатації обладнання [26].

Виходячи з кількісної оцінки ризиків, найбільшу увагу слід приділяти заходам, що знижують імовірність та наслідки повторюваних незначних і помірних ризиків, оскільки саме вони визначають загальний рівень безпеки життєдіяльності працівника при тривалій роботі за комп’ютером. До таких заходів належать: дотримання режиму праці та відпочинку, ергономічна організація робочого місця, забезпечення нормативних параметрів освітлення та мікроклімату, а також регулярне виконання вправ для зниження зорової та м’язової втоми [25][26].

4.2 Психофізіологічне розвантаження працівників

Праця оператора персонального комп’ютера належить до категорії робіт з підвищеним нервово-емоційним напруженням, що поєднується з тривалим перебуванням у вимушеній робочій позі та значним навантаженням на зоровий аналізатор. Тривала робота за відеодисплейним терміналом без належного чергування з відпочинком призводить до розвитку зорової та загальної втоми, зниження концентрації уваги, погіршення швидкості реакції та збільшення кількості помилок протягом робочого дня.

Психофізіологічне розвантаження працівників – це комплекс організаційних та технічних заходів, спрямованих на відновлення працездатності, зниження нервово-емоційної напруги та профілактику професійних захворювань, зумовлених тривалою роботою за комп’ютером [26]. До основних заходів психофізіологічного розвантаження належать регламентовані перерви, виробнича гімнастика, організація зон відпочинку, раціональне освітлення робочого місця та чергування видів діяльності протягом робочого дня.

Узагальнений перелік заходів психофізіологічного розвантаження, що рекомендуються для працівників, зайнятих тривалою роботою за персональним комп’ютером, наведено в таблиці 4.2.

Таблиця 4.2 – Заходи психофізіологічного розвантаження при роботі з ПК

Захід психофізіологічного розвантаження	**Періодичність**	**Очікуваний ефект**
1	**2**	**3**
Регламентована перерва з виконанням гімнастики для очей	Кожні 45-60 хв роботи за ПК	Зниження зорової втоми, профілактика астенопії
Виробнича гімнастика (комплекс вправ для шиї, спини, кистей)	2 рази за зміну (через 2 та 6 год)	Профілактика м’язово-скелетних розладів

Продовження таблиці 4.2

1	**2**	**3**
Кімната психологічного розвантаження / зона відпочинку	За потреби, в межах обідньої перерви	Зниження емоційної напруги, відновлення концентрації
Раціональне освітлення та кольорове оформлення робочого місця	Постійно (організаційний захід)	Зменшення втоми, підвищення працездатності
Чергування видів діяльності протягом робочого дня	Протягом робочого дня	Зниження монотонності, профілактика перевантаження

Розгляд найбільш суттєвих заходів:

- Регламентовані перерви. Відповідно до гігієнічних вимог до режимів праці і відпочинку при роботі з відеодисплейними терміналами, тривалість безперервної роботи за екраном не повинна перевищувати 45 хвилин, після чого необхідна перерва тривалістю не менше 15 хвилин з виконанням комплексу вправ для зняття зорової та загальної втоми [26]. Дотримання такого режиму забезпечує своєчасне відновлення працездатності та зниження ймовірності помилок при виконанні роботи.
- Виробнича гімнастика. Включає вправи для зняття напруження м'язів шиї, плечового поясу, спини та кистей рук, що зазнають статичного навантаження при тривалій роботі з клавіатурою та мишею. Рекомендована періодичність – два рази протягом робочого дня (через 2 та 6 годин від початку роботи) [25].
- Організація зони психологічного розвантаження. Виділення окремого приміщення або зони для короткочасного відпочинку дозволяє працівнику на кілька хвилин змінити обстановку, що сприяє зниженню емоційної напруги та відновленню концентрації уваги перед поверненням до роботи.
- Раціональне освітлення та кольорове оформлення робочого місця. Правильно підібране освітлення та кольорова гама приміщення знижують навантаження на орган зору та сприяють загальному психофізіологічному комфорту працівника протягом робочого дня [26].

- Чергування видів діяльності. Періодична зміна характеру виконуваних завдань протягом робочого дня знижує монотонність праці, що є важливим чинником профілактики як фізичної, так і психоемоційної перевтоми [25].

Впровадження наведеного комплексу заходів психофізіологічного розвантаження дозволяє знизити рівень втоми працівників протягом робочого дня, зменшити кількість помилок при виконанні професійних завдань та підвищити загальну продуктивність праці. Більшість зазначених заходів мають організаційний характер і не потребують значних додаткових витрат, що робить їх впровадження доцільним для будь-якого підприємства, де працівники тривалий час працюють за персональним комп’ютером.

ВИСНОВКИ

У межах кваліфікаційної роботи розроблено та протестовано CRM-систему для аптеки з інтегрованим модулем прогнозування продажів на основі моделей часових рядів. Проведений аналіз предметної області показав, що наявні на ринку рішення для автоматизації аптек не поєднують повноцінного CRM-функціоналу з можливостями прогнозування попиту, що й обумовило актуальність та практичну доцільність розробки власної програмної системи.

На етапі проєктування обґрунтовано вибір ітеративно-інкрементної моделі розробки з елементами Agile-методологій, що дозволило поступово уточнювати вимоги та послідовно наростити функціональність системи. Спроєктовано мікросервісну архітектуру, що включає основний сервіс CRM, окремий сервіс прогнозування попиту та клієнтську частину, які взаємодіють через REST API та спільну базу даних. Обґрунтовано вибір сучасного технологічного стеку на основі мови Python, фреймворку FastAPI, бібліотеки SQLModel, реляційної СУБД PostgreSQL та бібліотеки statsforecast для аналізу часових рядів, а клієнтську частину реалізовано як односторінковий веб-застосунок на React. Спроєктовано схему бази даних з дотриманням принципів нормалізації та збереження історичної цілісності даних, а також побудовано комплекс UML-діаграм, що формально описують структуру та поведінку системи.

У ході реалізації серверну частину побудовано за багаторівневою архітектурою з безпечною автентифікацією на основі токенів JWT та рольовою моделлю доступу, що забезпечує ізоляцію даних між різними аптеками мережі. Реалізовано атомарну обробку операцій продажу з автоматичним списанням товарів за принципом FEFO. Окрему увагу приділено реалізації модуля прогнозування попиту: у системі впроваджено набір статистичних моделей часових рядів з урахуванням тижневої сезонності продажів, обчисленням довірчих інтервалів та механізмом кешування результатів для зниження обчислювального навантаження. Розроблений клієнтський інтерфейс охоплює всі основні робочі

сценарії аптечної діяльності, включаючи B2B-взаємодію з постачальниками та візуалізацію прогнозів попиту.

Виконане тестування підтвердило відповідність розробленої системи поставленим вимогам. Проведено комплексну перевірку на рівні модульного, інтеграційного, API та наскрізного тестування, що охопила критичні бізнес-сценарії, зокрема алгоритм FEFO-списання, атомарність транзакцій продажу та коректність рольового розмежування доступу. Окремо здійснено верифікацію точності моделей прогнозування на реалістичному наборі даних методом крос-валідації часових рядів, результати якої підтвердили придатність обраних моделей для практичного застосування в умовах аптечної торгівлі; як базову для розгортання обрано модель, що демонструє найкращу точність з урахуванням тижневої сезонності попиту. Описано процедуру розгортання системи на основі контейнеризації Docker, що забезпечує відтворюваність середовища та спрощує впровадження на будь-якому сервері незалежно від операційної системи.

У розділі безпеки життєдіяльності та охорони праці виконано кількісну оцінку ризиків при роботі оператора персонального комп’ютера та запропоновано заходи психофізіологічного розвантаження персоналу для збереження працездатності.

Таким чином, поставлену мету кваліфікаційної роботи досягнуто повністю: розроблено працездатну CRM-систему для аптеки з інтегрованим модулем прогнозування продажів, яка автоматизує ключові бізнес-процеси аптечної діяльності та надає завідувачу аптекою інструмент для прийняття обґрунтованих рішень щодо управління товарними запасами на основі результатів аналізу часових рядів. Розроблена система може бути рекомендована до подальшої дослідної експлуатації в умовах реальної аптеки або аптечної мережі, а її архітектура допускає подальший розвиток – зокрема, розширення набору прогнозних моделей, інтеграцію з додатковими постачальниками та масштабування на більшу кількість торгових точок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Михалик Д. М., Цуприк Г. Б., Бревус В. М. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за освітньо-професійною програмою “Інженерія програмного забезпечення” спеціальності 121 – «Інженерія програмного забезпечення» всіх форм навчання. Тернопіль : ТНТУ ім. І. Пулюя, 2024. 45 с.

2. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed. Hoboken : Wiley, 2016. 712 p.

3. Hyndman R. J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd ed. Melbourne : OTexts, 2021. 442 p. URL: https://otexts.com/fpp3/ (дата звернення: 10.03.2026).

4. Garza F., Mergenthaler-Canseco M., Challú C., Olivares K. G. StatsForecast: lightning-fast forecasting with statistical and econometric models. Nixtla, 2022. URL: https://github.com/Nixtla/statsforecast (дата звернення: 20.03.2026).

5. Ramírez S. FastAPI: modern, fast web framework for building APIs with Python. URL: https://fastapi.tiangolo.com (дата звернення: 20.03.2026).

6. Ramírez S. SQLModel: SQL databases in Python, designed for simplicity and compatibility. URL: https://sqlmodel.tiangolo.com (дата звернення: 20.03.2026).

7. PostgreSQL 15 Documentation / The PostgreSQL Global Development Group. 2023. URL: https://www.postgresql.org/docs/15/ (дата звернення: 20.03.2026).

8. Colvin S. et al. Pydantic: data validation using Python type hints. URL: https://docs.pydantic.dev (дата звернення: 20.03.2026).

9. React: the library for web and native user interfaces / Meta Open Source. URL: https://react.dev (дата звернення: 20.03.2026).

10. Merkel D. Docker: lightweight Linux containers for consistent development and deployment. Linux Journal. 2014. Vol. 2014, No. 239. Art. 2.

11. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force, 2015. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 20.03.2026).

12. Biryukov A., Dinu D., Khovratovich D., Josefsson S. Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications. RFC 9106. IETF, 2021.

13. Chopra S., Meindl P. Supply Chain Management: Strategy, Planning, and Operation. 7th ed. Harlow : Pearson, 2019. 528 p.

14. IBS Аптека : програмний комплекс для автоматизації аптек і аптечних мереж. URL: https://www.ibsystems.com.ua (дата звернення: 15.03.2026).

15. AHP-Аптека : програма комплексної автоматизації аптечних підприємств. URL: https://anr.ua (дата звернення: 15.03.2026).

16. Creatio : платформа управління взаємовідносинами з клієнтами. URL: https://www.creatio.com/ua/ (дата звернення: 15.03.2026).

17. МедФарм : програма автоматизації аптечних закладів. URL: https://medfarm.net.ua/ (дата звернення: 15.03.2026).

18. Ramírez S. Full Stack FastAPI Template. URL: https://github.com/fastapi/full-stack-fastapi-template (дата звернення: 12.06.2026). Ліцензія: MIT.

19. Ladas C. Scrumban: Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press, 2009. 96 p.

20. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021. 654 p.

21. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 533 p.

22. OMG Unified Modeling Language (UML), Version 2.5.1. Object Management Group, 2017. URL: https://www.omg.org/spec/UML/2.5.1/ (дата звернення: 20.03.2026).

23. M4 Forecasting Competition Dataset. URL: https://www.kaggle.com/datasets/yogesh94/m4-forecasting-competition-dataset (дата звернення: 20.03.2026).

24. Методичні вказівки для написання розділу “Безпека життєдіяльності, основи охорони праці” в кваліфікаційних роботах здобувачів освітнього рівня

бакалавр / уклад. О. Я. Гурик, І. Б. Окіп. Тернопіль : ТНТУ ім. І. Пулюя, 2023. 32 с.

25. Безпека життєдіяльності та охорона праці : підручник / В. В. Сокуренко, О. М. Бандурка та ін. Харків : ХНУВС, 2021. 308 с.

26. Жидецький В. Ц. Охорона праці користувачів комп'ютерів : підручник. Львів : Афіша, 2020. 176 с.

27. Buttle F., Maklan S. Customer Relationship Management: Concepts and Technologies. 4th ed. London : Routledge, 2019. 462 p.

ДОДАТКИ

Додаток А

Тези конференції

УДК 621.326
Костецький О. – гр. СП-42
Тернопільський національний технічний університет імені Івана Пулюя

ЗАСТОСУВАННЯ АРХІТЕКТУР НА ОСНОВІ МЕХАНІЗМУ САМОУВАГИ ДЛЯ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

Науковий керівник: к.ф.-м. н., доцент Цебрій О.Р.

Kostetskyi O.
Ternopil Ivan Puluj National Technical University

USING ARCHITECTURES BASED ON THE SELF-ATTENTION MECHANISM FOR TIME SERIES FORECASTING

Supervisor: PhD in Physics and Mathematics, Associate Professor Tsebrii O.R.

Ключові слова: часові ряди, машинне навчання, механізм самоуваги
Keywords: time series, machine learning, self-attention mechanism

Прогнозування часових рядів є критично важливою задачею у багатьох сучасних системах, від аналізу фінансових ринків та планування попиту до обробки телеметричних даних з IoT-сенсорів. Традиційні підходи до аналізу таких даних зазвичай спираються на статистичні моделі, такі як ARIMA, або класичні алгоритми машинного навчання. Хоча ці методи є ефективними для відносно простих або стаціонарних послідовностей, вони часто не здатні вловити складні нелінійні залежності та багатовимірні взаємодії у великих масивах даних.

Еволюція методів глибокого навчання призвела до широкого використання рекурентних нейронних мереж (RNN), зокрема архітектур LSTM та GRU, які тривалий час були стандартом для роботи з послідовними даними. Вони вирішують проблему збереження контексту на коротких дистанціях, проте їхня послідовна природа створює суттєві обмеження. Обробка даних крок за кроком ускладнює розпаралелювання обчислень, а при роботі з довгими часовими вікнами моделі стикаються з проблемою втрати інформації та затухання градієнта.

Сучасним підходом, що дозволяє обійти ці обмеження, є використання архітектури Transformer, яка нині активно адаптується для задач аналізу часових рядів. Головною інновацією трансформерів є відмова від рекурентних зв'язків на користь механізму самоуваги (Self-Attention), який вперше був запропонований для задач обробки природної мови [1]. Цей механізм дозволяє моделі обчислювати залежності між усіма елементами часового ряду одночасно, визначаючи ступінь важливості кожного історичного спостереження для прогнозування майбутніх значень.

Оскільки механізм уваги не враховує порядок елементів, для збереження інформації про часову послідовність застосовується позиційне кодування (Positional Encoding). Крім того, пряме застосування класичного трансформера до довгих часових рядів обмежене його квадратичною обчислювальною складністю. Тому сучасні дослідження зосереджені на спеціалізованих модифікаціях (наприклад, архітектура Informer та її механізм ProbSparse Attention), які дозволяють знизити складність і ефективно прогнозувати надзвичайно довгі послідовності [2].

Широкий спектр експериментальних досліджень та комплексних оглядів підтверджує, що моделі на базі Transformer перевершують класичні рекурентні мережі за точністю багатокрокового прогнозування [3]. Використання багатошарової уваги (Multi-Head Attention) дає змогу моделі паралельно вивчати різні типи закономірностей: одні шари можуть фокусуватися на короткострокових флуктуаціях, тоді як інші виявляють довгострокові сезонні тренди.

Таким чином, перехід від послідовного аналізу часових рядів до паралельної обробки на основі архітектури Transformer відкриває нові можливості для побудови високоточних та масштабованих систем прогнозування. Подальший розвиток цього напрямку пов'язаний з оптимізацією архітектури під специфіку неперервних числових даних та розробкою гібридних рішень, що поєднують здатність трансформерів до виявлення глобальних залежностей з ефективністю традиційних методів фільтрації шуму.

Література:

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998-6008.

2. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 35(12), 11106-11115.

3. Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2022). Transformers in time series: A survey. International Joint Conferences on Artificial Intelligence Organization, 3778-3786.

Додаток Б

Скрипт для оцінки моделей

```
import argparse
import sys
from pathlib import Path

import numpy as np
import pandas as pd
from statsforecast import StatsForecast
from statsforecast.models import (
    ARIMA,
    AutoARIMA,
    AutoCES,
    AutoETS,
    CrostonClassic,
    HistoricAverage,
    Naive,
    SeasonalNaive,
    Theta,
)

# Тижнева сезонність — той самий параметр, що й у forecast-сервісі
SEASON_LENGTH = 7


#
-------------------------------------------------------------------
-------
# Набір моделей (дзеркало forecast/app/services/models_registry.py)
#
-------------------------------------------------------------------
-------
def build_models() -> list:
```

```
    """Повертає перелік моделей, ідентичний реєстру робочого
сервісу."""
    return [
        AutoETS(season_length=SEASON_LENGTH, alias="AutoETS"),
        ARIMA(
            order=(1, 0, 1),
            seasonal_order=(1, 0, 1),
            season_length=SEASON_LENGTH,
            alias="SARIMA",
        ),
        AutoARIMA(season_length=SEASON_LENGTH, alias="AutoARIMA"),
        AutoCES(season_length=SEASON_LENGTH, alias="AutoCES"),
        Theta(season_length=SEASON_LENGTH, alias="Theta"),
        SeasonalNaive(season_length=SEASON_LENGTH,
alias="SeasonalNaive"),
        CrostonClassic(alias="Croston"),
        HistoricAverage(alias="HistoricAverage"),
        Naive(alias="Naive"),
    ]


#
-------------------------------------------------------------------
-------
# Завантаження та підготовка даних
#
-------------------------------------------------------------------
-------
def _to_continuous_daily(df: pd.DataFrame) -> pd.DataFrame:
    parts = []
    for uid, grp in df.groupby("unique_id"):
        grp = grp.sort_values("ds")
        full_idx = pd.date_range(grp["ds"].min(), grp["ds"].max(),
freq="D")
        s = (
            grp.set_index("ds")["y"]
```

```
            .reindex(full_idx, fill_value=0.0)
            .rename_axis("ds")
            .reset_index()
        )
        s["unique_id"] = uid
        parts.append(s[["unique_id", "ds", "y"]])
    return pd.concat(parts, ignore_index=True)


def load_rossmann(path: str, n_stores: int) -> pd.DataFrame:
    csv_path = Path(path)
    if not csv_path.exists():
        sys.exit(
            f"[!] Файл '{path}' не знайдено.\n"
            f"    Завантажте Rossmann Store Sales train.csv з
Kaggle:\n"
            f"
https://www.kaggle.com/competitions/rossmann-store-sales/data"
        )
    df = pd.read_csv(csv_path, parse_dates=["Date"],
low_memory=False)
    # Беремо лише відкриті дні з ненульовими продажами (закриті дні
= шум)
    df = df[(df["Open"] == 1) & (df["Sales"] > 0)]
    # Відбираємо N магазинів із найбільшою кількістю спостережень
    top = df["Store"].value_counts().head(n_stores).index
    df = df[df["Store"].isin(top)]
    df = df.rename(columns={"Store": "unique_id", "Date": "ds",
"Sales": "y"})
    df = df[["unique_id", "ds", "y"]].sort_values(["unique_id",
"ds"])
    df["unique_id"] = "store_" + df["unique_id"].astype(str)
    return _to_continuous_daily(df)
```

```
def load_m4_daily(n_series: int, max_history: int = 365) ->
pd.DataFrame:
    try:
        from datasetsforecast.m4 import M4
    except ImportError:
        sys.exit(
            "[!] Потрібен пакет datasetsforecast: pip install
datasetsforecast"
        )

    print("[i] Завантаження M4 Daily (Nixtla/GitHub, без
авторизації)...")
    y_df, _, _ = M4.load(directory="./_data_cache", group="Daily")

    # Обираємо N рядів з достатньою довжиною історії
    lengths =
y_df.groupby("unique_id").size().sort_values(ascending=False)
    chosen = lengths.head(n_series).index
    y_df = y_df[y_df["unique_id"].isin(chosen)].copy()

    # Обмежуємо кожен ряд останніми max_history спостереженнями
    y_df = (
        y_df.sort_values(["unique_id", "ds"])
        .groupby("unique_id", group_keys=False)
        .tail(max_history)
    )

    # Цілочисельний день -> календарна дата (базова точка довільна,
важливий лише крок)
    base_date = pd.Timestamp("2020-01-01")
    y_df["ds"] =
y_df.groupby("unique_id")["ds"].rank(method="first").astype(int)
    y_df["ds"] = base_date + pd.to_timedelta(y_df["ds"] - 1,
unit="D")

    return _to_continuous_daily(y_df[["unique_id", "ds", "y"]])
```

```
def load_m5(n_series: int) -> pd.DataFrame:
    try:
        from datasetsforecast.m5 import M5
    except ImportError:
        sys.exit(
            "[!] Потрібен пакет datasetsforecast: pip install
datasetsforecast"
        )

    print("[i] Завантаження M5 (Nixtla/GitHub, без авторизації)... "
          "може зайняти кілька хвилин при першому запуску.")
    y_df, _, _ = M5.load(directory="./_data_cache")

    lengths =
y_df.groupby("unique_id").size().sort_values(ascending=False)
    chosen = lengths.head(n_series).index
    y_df = y_df[y_df["unique_id"].isin(chosen)].copy()
    y_df["ds"] = pd.to_datetime(y_df["ds"])

    return _to_continuous_daily(y_df[["unique_id", "ds", "y"]])


def make_synthetic(n_series: int = 8, n_days: int = 180, seed: int =
42) -> pd.DataFrame:
    rng = np.random.default_rng(seed)
    rows = []
    base_date = pd.Timestamp("2024-01-01")
    for i in range(n_series):
        level = rng.uniform(20, 60)
        trend = rng.uniform(-0.03, 0.06)
        weekly = rng.uniform(3, 12)
        days = np.arange(n_days)
        seasonal = weekly * np.sin(2 * np.pi * days / SEASON_LENGTH)
        noise = rng.normal(0, level * 0.12, n_days)
```

```
        y = np.clip(level + trend * days + seasonal + noise, 0,
None)
        for d in range(n_days):
            rows.append(
                {
                    "unique_id": f"synthetic_{i}",
                    "ds": base_date + pd.Timedelta(days=d),
                    "y": float(round(y[d], 1)),
                }
            )
    return pd.DataFrame(rows)


#
-------------------------------------------------------------------
-------
# Метрики якості прогнозу
#
-------------------------------------------------------------------
-------
def mae(y: np.ndarray, yhat: np.ndarray) -> float:
    return float(np.mean(np.abs(y - yhat)))


def rmse(y: np.ndarray, yhat: np.ndarray) -> float:
    return float(np.sqrt(np.mean((y - yhat) ** 2)))


def mape(y: np.ndarray, yhat: np.ndarray) -> float:
    mask = y > 0
    if not mask.any():
        return float("nan")
    return float(np.mean(np.abs((y[mask] - yhat[mask]) / y[mask])) *
100)
```

```
#
-------------------------------------------------------------------
-------
# Основна логіка оцінки
#
-------------------------------------------------------------------
-------
def evaluate(df: pd.DataFrame, horizon: int, n_windows: int) ->
pd.DataFrame:
    models = build_models()
    sf = StatsForecast(models=models, freq="D", n_jobs=1)

    print(
        f"[i] Рядів: {df['unique_id'].nunique()} | "
        f"спостережень: {len(df)} | горизонт: {horizon} дн. | "
        f"вікон CV: {n_windows}\n"
    )

    # Walk-forward крос-валідація (rolling origin)
    cv = sf.cross_validation(
        df=df,
        h=horizon,
        n_windows=n_windows,
        step_size=horizon,
    )

    model_names = [m.alias for m in models]
    y_true = cv["y"].to_numpy(dtype=float)

    results = []
    for name in model_names:
        yhat = cv[name].to_numpy(dtype=float)
        yhat = np.clip(yhat, 0, None)  # попит не може бути
від'ємним
        results.append(
            {
```

```
                "Модель": name,
                "MAE": round(mae(y_true, yhat), 2),
                "MAPE, %": round(mape(y_true, yhat), 1),
                "RMSE": round(rmse(y_true, yhat), 2),
            }
        )

    res_df = pd.DataFrame(results).sort_values("MAPE,
%").reset_index(drop=True)
    return res_df


def main() -> None:
    parser = argparse.ArgumentParser(
        description="Оцінка точності моделей прогнозування (MAE,
MAPE, RMSE)"
    )
    parser.add_argument(
        "--dataset", type=str, default="m4",
        choices=["m4", "m5", "rossmann", "synthetic"],
        help="Джерело даних: m4 (за замовч., авто-завантаження з
GitHub), "
             "m5 (Walmart, авто-завантаження), rossmann (потрібен
--data), "
             "synthetic (без інтернету)",
    )
    parser.add_argument("--data", type=str, default=None,
                        help="Шлях до Rossmann train.csv (лише для
--dataset rossmann)")
    parser.add_argument("--series", "--stores", dest="series",
type=int, default=20,
                        help="Кількість часових рядів / магазинів
(за замовч. 20)")
    parser.add_argument("--horizon", type=int, default=7,
                        help="Горизонт прогнозування, днів (за
замовч. 7)")
```

```
    parser.add_argument("--windows", type=int, default=4,
                        help="Кількість вікон крос-валідації (за
замовч. 4)")
    parser.add_argument("--out", type=str,
default="evaluation_results.csv",
                        help="Файл для збереження результатів")
    args = parser.parse_args()
    if args.dataset == "synthetic":
        df = make_synthetic()
        source = "синтетичні дані"
    elif args.dataset == "m4":
        df = load_m4_daily(args.series)
        source = f"M4 Daily ({args.series} рядів, Nixtla/GitHub)"
    elif args.dataset == "m5":
        df = load_m5(args.series)
        source = f"M5 Walmart ({args.series} рядів, Nixtla/GitHub)"
    else:  # rossmann
        if args.data is None:
            sys.exit("[!] Для --dataset rossmann потрібно вказати
--data train.csv")
        df = load_rossmann(args.data, args.series)
        source = f"Rossmann Store Sales ({args.series} магазинів)"

    res = evaluate(df, horizon=args.horizon, n_windows=args.windows)
    print(f"Результати оцінки точності — {source}")
    print("=" * 48)
    print(res.to_string(index=False))
    print("=" * 48)
    best = res.iloc[0]
    print(f"\nНайкраща модель за MAPE: {best['Модель']} "
          f"(MAPE = {best['MAPE, %']} %, MAE = {best['MAE']}, RMSE =
{best['RMSE']})")
    res.to_csv(args.out, index=False, encoding="utf-8-sig")
    print(f"\n[✓] Результати збережено у файл: {args.out}")
if __name__ == "__main__":
    main()
```