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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система моніторингу мережі вендингових автоматів*

Виконав: студент 4 курсу, групи СІ-42
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

		Котюк О.А.
	(підпис)	(прізвище та ініціали)
Керівник		Яцишин В.В.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Тиш Є.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Осухівська Г.М.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2025

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

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

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
«27» січня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Котюку Олегу Андрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система моніторингу мережі вендингових автоматів*

Керівник роботи *Яцишин Василь Володимирович, к.т.н доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від *« 27 » січня 2025 року № 4/7-53*
2. Термін подання студентом завершеної роботи
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз вимог та існуючих рішень щодо управління мережею вендингових автоматів.
2. Проєктування системи управління вендинговими автоматами.
3. Розробка програмної реалізації системи управління вендинговими автоматами.
4. Безпека життєдіяльності, основи охорони праці.
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Структурна схема системи
2. Блок-схема алгоритму
3. ER-діаграма бази даних
4. Схема електрична монтажна (з'єднань)

6. Консультанти розділів роботи

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

7. Дата видачі завдання *27.01.2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*27.01 – 02.02*	*Викон.*
2.	*Аналіз вимог та існуючих рішень щодо управління мережею вендингових автоматів*	*01.03 – 31.03*	*Викон.*
3.	*Проєктування системи управління вендинговими автоматами*	*01.04 – 30.04*	*Викон.*
4.	*Розробка програмної реалізації системи управління вендинговими автоматами*	*01.05 – 31.05*	*Викон.*
5.	*Безпека життєдіяльності, основи охорони праці*	*12.06 – 13.06*	*Викон.*
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*01.06 – 13.06*	*Викон.*
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*9.06 – 15.06*	
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*16.06 – 22.06*	*Викон.*
9.	*Захист кваліфікаційної роботи бакалавра*		*Викон.*

Студент _____ *Котюк О.А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Яцишин В.В.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Котюк О.А. Комп’ютеризована система моніторингу мережі вендингових автоматів: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 123 — комп’ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2025.

Ключові слова: комп’ютерна система, вендинговий автомат, Інтернет речей, мікроконтролер ESP32, REST API, моніторинг, база даних, FastAPI, SQLModel, дистанційний контроль.

У кваліфікаційній роботі розроблено комп’ютеризовану систему моніторингу мережі вендингових автоматів, що забезпечує централізований збір, обробку та аналіз інформації про стан пристроїв у режимі реального часу. Система побудована на базі сучасних технологій Інтернету речей (IoT) та використовує мікроконтролери ESP32 для бездротової передачі даних на серверну частину через REST API. Серверна частина реалізована за допомогою вебфреймворку FastAPI і взаємодіє з базою даних PostgreSQL через ORM SQLModel.

У роботі здійснено структурно-функціональний аналіз системи, розроблено схему інформаційної взаємодії між компонентами, реалізовано програмне забезпечення для контролю автоматів, а також вебінтерфейс для адміністратора. Описано процес обробки подій, механізми захисту від втрати даних та забезпечення безпеки інформаційного обміну.

Система дозволяє здійснювати моніторинг товарних залишків, продажів, подій обслуговування та технічного стану автоматів. Результати роботи підтверджують ефективність побудованої архітектури для задач віддаленого контролю та аналітики в розподілених мережах пристроїв.

ANNOTATION

Kotyuk O.A. Computerized system for monitoring a vending machine network: Bachelor's Graduation Thesis: speciality 123 — Computer Engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2025.

Keywords: computer system, vending machine, Internet of Things, ESP32 microcontroller, REST API, monitoring, database, FastAPI, SQLModel, remote control.

The bachelor's thesis presents the development of a computerized system for monitoring a network of vending machines, enabling centralized collection, processing, and analysis of device status data in real time. The system is built on modern Internet of Things (IoT) technologies and uses ESP32 microcontrollers for wireless data transmission to the server via a REST API. The backend is implemented using the FastAPI web framework and interacts with a PostgreSQL database through the SQLModel ORM.

The work includes a structural and functional analysis of the system, the design of the information interaction scheme between components, the implementation of software for vending machine control, and the creation of an administrator web interface. The event processing logic, data loss prevention mechanisms, and data security measures are also described.

The system enables monitoring of inventory levels, sales, maintenance events, and the technical status of vending machines. The results confirm the effectiveness of the designed architecture for remote control and analytics in distributed device networks.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

API – Application Programming Interface

CPU – Central Processing Unit

ESP – Espressif Systems Platform

ESP32 – Espressif Systems Platform 32-bit

HTTP – HyperText Transfer Protocol

IoT – Internet of Things

JSON – JavaScript Object Notation

ORM – Object-Relational Mapping

RAM – Random Access Memory

REST – Representational State Transfer

SQL – Structured Query Language

SQLModel – Structured Query Language Model

UI – User Interface

Wi-Fi – Wireless Fidelity

IPC – Inter-Process Communication

ВСТУП

У сучасних умовах стрімкої цифровізації бізнес-процесів, що охоплює практично всі сфери економічної діяльності, усе більшого значення набуває комплексна автоматизація процесів обліку, контролю та управління. Ця тенденція є особливо вираженою у галузях, які оперують значною кількістю автономних пристроїв. Однією з таких галузей є торгівля за допомогою вендингових автоматів – автономних пристроїв для продажу товарів без безпосередньої участі оператора. З огляду на інтенсивне розширення мереж таких автоматів, що охоплюють значні географічні території та функціонують у режимі 24/7, виникає гостра потреба у впровадженні високоефективних засобів централізованого моніторингу та дистанційного керування. Це, у свою чергу, безпосередньо вимагає створення відповідних комп’ютеризованих систем, здатних забезпечити надійну комунікацію, збір та обробку даних.

Аналіз ринку наявних програмно-апаратних рішень для управління вендинговими мережами показує, що вони часто характеризуються значною вартістю, закритою архітектурою, що обмежує можливості кастомізації, або високою складністю інтеграції з існуючими бізнес-процесами. Такі обмеження створюють об'єктивний попит на розробку власних, гнучких, оптимізованих під конкретні потреби користувача, систем. Зважаючи на вищезазначене, актуальним є завдання створення доступної, масштабованої, безпечної та зручної у використанні програмно-апаратної системи, яка забезпечить ефективне управління мережею вендингових автоматів.

Метою кваліфікаційної роботи є розробка програмно-апаратної системи, яка забезпечить комплексний збір статистичних даних про продажі з вендингових автоматів, їх надійну обробку, збереження в централізованій базі даних, візуалізацію даних у зручному для аналізу форматі, а також централізоване адміністрування та дистанційне керування функціями автоматів.

Розроблена система дозволить в режимі реального часу відстежувати оперативну роботу кожного автомата, включаючи його статус (онлайн/офлайн), наявність товарів та технічний стан. Вона забезпечить отримання деталізованих звітів про продажі, що дозволить аналізувати ефективність розміщення автоматів, попит на окремі товари та оптимізувати логістичні маршрути. Крім того, система надасть можливість оперативно реагувати на можливі збої або нештатні ситуації, мінімізуючи час простою обладнання. Впровадження такої інтегрованої системи значно підвищить загальну ефективність управління мережею автоматів, зменшить вплив людського фактора на операційні процеси та надасть керівництву можливість приймати обґрунтовані рішення на основі об'єктивних аналітичних даних.

Сервіс, що буде розроблений в рамках даної роботи, включатиме декілька ключових компонентів: централізовану базу даних для зберігання всієї інформації; інтуїтивно зрозумілий веб-інтерфейс адміністратора, що забезпечує перегляд аналітики, управління автоматами та їх налаштуваннями; а також набір програмних інтерфейсів (API) для безшовної взаємодії з вендинговими автоматами та потенційної інтеграції з іншими зовнішніми системами.

Загалом, кваліфікаційна робота демонструє комплексний підхід до вирішення актуальної задачі автоматизації управління вендинговими мережами, пропонуючи сучасне, ефективне та масштабоване рішення. Її результати мають значну практичну цінність, оскільки сприяють оптимізації операційних витрат, підвищенню якості обслуговування та створенню нових можливостей для розвитку бізнесу в умовах зростаючої конкуренції та вимог до технологічності.

РОЗДІЛ 1 АНАЛІЗ ВИМОГ ТА ІСНУЮЧИХ РІШЕНЬ ЩОДО УПРАВЛІННЯ МЕРЕЖЕЮ ВЕНДИНГОВИХ АВТОМАТІВ

1.1 Підхід до сфери застосування комп'ютеризованої системи управління вендинговими автоматами

Система управління мережею вендингових автоматів розроблена з метою автоматизації процесів моніторингу, управління та збору даних з вендингових автоматів. Така система необхідна для централізованого управління мережею торгових автоматів, що дозволяє здійснювати моніторинг їх стану, здійснювати облік продажів, а також приймати оперативні рішення щодо обслуговування або змін у асортименті. Система буде використовуватись у таких умовах, як торгові центри, супермаркети, магазини, офіси та інші громадські місця, де встановлені вендингові автомати для продажу різноманітної продукції, включаючи напої, снекові продукти, канцелярські товари та інші товари для користувачів.

Основні вигоди для бізнесу, які надає ця система, включають можливість відстеження продажів і стану товарів у реальному часі, що дає змогу оперативно реагувати на потреби споживачів і оптимізувати асортимент товарів. Крім того, автоматизоване обслуговування та моніторинг автоматів допомагає знизити витрати на персонал і забезпечити безперебійне функціонування торгових точок без необхідності частого фізичного контролю.

Система управління мережею вендингових автоматів надає можливість централізованого моніторингу всіх автоматів, що дозволяє керівництву підприємства отримувати доступ до детальних звітів про стан кожного окремого автомата. Це включає інформацію про залишки товарів, рівень

монет, можливі технічні несправності або потребу в обслуговуванні. Такий підхід забезпечує оперативне виявлення проблем і дозволяє вчасно планувати ремонтні роботи чи поповнення товарних запасів, що значно знижує кількість несправностей і втрат доходу через неефективну роботу автоматів.

Автоматизоване управління мережею вендингових автоматів забезпечує інтеграцію з іншими системами, такими як платіжні платформи, що дозволяє здійснювати контроль за транзакціями, обліком фінансів і звітністю. Це дозволяє уникнути помилок у фінансових звітах і мінімізувати ймовірність шахрайства чи неправильного обліку. Завдяки автоматичним оновленням даних про фінансові операції, система забезпечує зручний доступ до інформації про прибутки в реальному часі.

Використання такої системи також дає змогу знижувати операційні витрати завдяки централізованому управлінню. Замість того, щоб кожен вендинговий автомат був ізольованою одиницею, що потребує окремого обслуговування і моніторингу, централізована система дозволяє об’єднати всі автомати в єдину мережу, що значно полегшує процеси контролю і обслуговування. Це дозволяє підприємствам ефективно розподіляти ресурси, контролювати витрати на технічне обслуговування і оптимізувати логістику.

Система управління мережею вендингових автоматів повинна відповідати низці основних вимог, які забезпечують її ефективну роботу та максимальну зручність для користувачів.

Реєстрація та ідентифікація автоматів: система повинна дозволяти реєстрацію нових вендингових автоматів із присвоєнням унікального ідентифікатора, зазначенням місця встановлення, типу пристрою та інших параметрів. Це є основою для подальшого збору статистики та адміністрування мережі.

Автоматизований збір і точна обробка даних: система має забезпечити достовірне зчитування та збереження даних з автоматів. Це включає інформацію про продажі, залишки товарів, технічний стан, а також події чи

помилки. Важливо, щоб дані збиралися автоматично та без втручання користувача.

Оперативна передача даних у реальному часі: зібрані дані повинні передаватися на сервер без затримок. Це дозволяє адміністраторам вчасно реагувати на критичні зміни — наприклад, вичерпання товарів або виникнення технічних несправностей.

Швидкість оновлення інформації: система має швидко відображати зміни на стороні сервера, щоб інтерфейс користувача завжди показував актуальні показники. Це особливо важливо для ефективного обслуговування та логістики.

Стабільність і надійність функціонування: система повинна бути готовою до роботи з великою кількістю підключених пристроїв одночасно, не втрачаючи продуктивності. Важливою є також здатність до масштабування та відновлення після збоїв.

Зручний та інтуїтивний інтерфейс: користувацький інтерфейс повинен бути зрозумілим навіть для не технічних користувачів. Він має забезпечити легкий доступ до звітів, налаштувань, аналітики та можливостей адміністрування автоматів.

Гнучке адміністрування користувачів і прав доступу: система має підтримувати багаторівневу систему доступу з розмежуванням прав. Це забезпечить контроль над тим, хто і до яких даних має доступ, запобігаючи несанкціонованим діям.

Формування звітів і статистичних зведень: повинна бути можливість створення докладних звітів за періодами, за автоматами, за категоріями товарів. Звіти мають експортуватися у стандартні формати (CSV, PDF) для подальшого аналізу.

Безпека даних та авторизація користувачів: необхідно реалізувати захищену автентифікацію (логін, пароль, токени) та шифрування даних при

передачі. Це гарантує конфіденційність комерційної інформації та захист від атак.

Підтримка автономної роботи: вендингові автомати можуть тимчасово втрачати з'єднання з мережею, тому система повинна вміти працювати автономно з локальним збереженням даних, що пізніше синхронізуються з сервером після відновлення зв'язку.

Загалом, система повинна забезпечити точність, швидкість, зручність та безпеку, а також бути стабільною і надійною в роботі в будь-яких умовах.

1.2 Аналітичний огляд існуючих рішень у сфері моніторингу торгових пристроїв

Існуючі рішення на ринку для моніторингу вендингових автоматів включають в себе такі системи, як Nayax, VendSoft, та SmartVend. Ці системи пропонують комплексний підхід до управління мережею торгових автоматів, включаючи можливість моніторингу стану автоматів у реальному часі, збирати статистику продажів та передавати дані на сервер для подальшої обробки. Крім того, вони дозволяють здійснювати автоматичне оновлення даних щодо товарних залишків, поповнення запасів, а також отримувати повідомлення про технічні проблеми або необхідність обслуговування автоматів.

Багато з цих рішень також мають функції для інтеграції з платіжними системами, що дозволяє здійснювати контроль за фінансовими операціями. Проте, більшість таких систем мають певні обмеження в гнучкості налаштувань, користувацькому інтерфейсі, а також в адаптації до специфічних потреб різних бізнесів. Наприклад, деякі з них обмежуються тільки базовими функціями моніторингу і збору статистики без можливості кастомізації звітів чи інтеграції з іншими системами управління.

Інші системи можуть мати високі витрати на впровадження та обслуговування, що робить їх менш доступними для малих і середніх підприємств.

Для розширення функціональних можливостей і забезпечення кращої гнучкості, ринок потребує рішень, які надають користувачам більше можливостей для налаштування, інтеграції та адаптації до індивідуальних потреб.

Наприклад, система Nayax (рис. 1.1) забезпечує інтеграцію з платіжними системами і дозволяє здійснювати повний моніторинг вендингових автоматів.

Основними компонентами системи Nayax є універсальні пристрої, такі як VPOS Touch та Onyx, які поєднують функції платіжного термінала та модуля телеметрії. Вони підтримують прийом оплат банківськими картками (магнітними, з чіпом, безконтактними), а також мобільними платежами через NFC та QR-коди.

Рисунок 1.1 – Моніторинговий термінал Nayax

Однак, вона має високі витрати на впровадження та обслуговування. VendSoft, (рис. 1.2) в свою чергу, пропонує більш доступну за ціною платформу, але з обмеженим набором функцій [23].

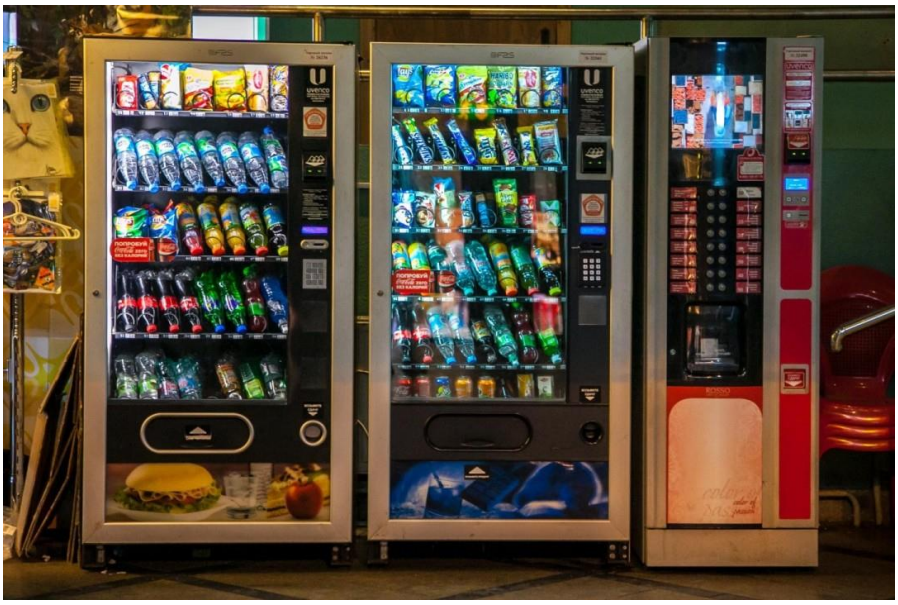

Рисунок 1.2 – Вендингові автомати оснащені системою VendSoft

Автомати, підключені до системи VendSoft, оснащуються передовими телеметрійними модулями й інтеграцією з хмарним сервісом, що дозволяє отримувати безперервний доступ до даних про роботу, продажі й обслуговування.

Перш за все, кожен автомат конфігурується у системі через веб-інтерфейс: йому присвоюється унікальний код, модель, тип («напій», «перекус»), серійний номер, а також налаштовуються параметри моніторингу, такі як лічильники транзакцій і готівки. Далі формується планограма — список слотів з товарами, цінами й максимальними обсягами, що дозволяє системі оцінювати залишки в автоматі й формувати рекомендації по поповненню (рис. 1.3).

Рисунок 1.3 –Інтерфейс системи VendSoft

Існуючі системи моніторингу та управління вендинговими автоматами, такі як Nayax, VendSoft та SmartVend, мають свої переваги та недоліки, що можуть бути корисними для формування вимог до розробки нової системи для даного проєкту. Nayax є потужною та широко використовуваною системою, що забезпечує комплексний набір інструментів для моніторингу та управління вендинговими автоматами, включаючи управління платежами, інвентаризацію, моніторинг технічного стану та віддалене налаштування автоматів.

Однією з основних переваг є гнучкість у виборі способу оплати, оскільки система підтримує різні платіжні методи, включаючи банківські картки, мобільні платежі та навіть криптовалюти. Це робить її зручною для міжнародного використання. Проте, одним з головних недоліків є висока вартість обладнання та підключення до платформи, що може бути недоцільним для малих або середніх бізнесів.

Для інтеграції в систему Nayax необхідно використовувати спеціалізоване обладнання, що створює залежність від постачальника послуг і ускладнює модернізацію або зміни в майбутньому. Також система складніша в адаптації до специфічних потреб, що може бути проблемою для певних типів автоматів.

VendSoft має простий та зручний інтерфейс, що дозволяє користувачам швидко отримувати доступ до звітів і налаштувань автоматів. Система підтримує різноманітні типи автоматів, що дозволяє інтегрувати її з більшістю моделей, які є на ринку. Однак її можливості для масштабування можуть бути обмеженими при великих обсягах даних або великій кількості автоматів, також, система потребує регулярних оновлень, що може викликати труднощі для користувачів, особливо якщо компанія не має можливості своєчасно виконувати ці оновлення.

Ще одним недоліком є потенційні проблеми з безпекою, зокрема зберіганням чутливих даних про платежі та користувачів, що вимагає додаткових заходів безпеки.

VendSoft пропонує можливість моніторити стан автоматів у реальному часі, що дозволяє швидко реагувати на будь-які технічні проблеми або зміни в асортименті. Програмне забезпечення також підтримує мобільні додатки для адміністраторів, що дозволяє віддалено керувати автоматами.

Однією з переваг є інтеграція з фінансовими системами, що спрощує управління фінансами. Проте у системи є і свої недоліки. Вона може мати проблеми з інтеграцією з різними моделями автоматів, що ускладнює процес підключення нових пристроїв. Як і в Nayax, висока вартість налаштування та обладнання може бути обмеженням для малого бізнесу. Також система може не працювати з усіма типами автоматів, що обмежує її універсальність на ринку.

Враховуючи ці переваги і недоліки існуючих рішень, можна зазначити, що вони мають суттєві переваги в плані функціональності, але і значні

обмеження щодо вартості, гнучкості та масштабування. Це дає можливість для вдосконалення розробленої системи, що буде більш доступною, гнучкою і масштабованою, з можливістю адаптації до специфічних вимог бізнесу та різних моделей вендингових автоматів. Розробка нової системи може врахувати ці недоліки, забезпечивши більш ефективне, економічне та зручне рішення для управління мережею торгових автоматів.

1.3 Потенційні шляхи реалізації системи моніторингу вендингових автоматів

Для реалізації комп'ютеризованої системи управління мережею вендингових автоматів можна використовувати модульний підхід, який включатиме кілька ключових компонентів. Кожен із цих компонентів взаємодіятиме між собою, забезпечуючи автоматизацію збору даних, їх передачу на сервер та зручне адміністрування. Структура можливої реалізації буде складатися з наступних основних частин.

Модуль збору даних з автоматів. Основним завданням цього модуля є отримання інформації про стан вендингових автоматів, включаючи продажі, рівень товарів, рівень монет, статус обладнання та інші показники. Для цього можна використовувати мікроконтролери (наприклад, ESP32 або аналогічні) з підключеними датчиками для збору цих даних. Мікроконтролери будуть отримувати дані від автоматів, а потім передавати їх через Wi-Fi або Ethernet на сервер за допомогою HTTP-запитів. Цей модуль також відповідає за обробку помилок, якщо автомат не відповідає або дані не можуть бути передані.

Система передачі даних на сервер. Для забезпечення безперервної передачі даних на сервер використовується стандартний механізм HTTP-запитів, що дозволяє здійснювати безперервне оновлення інформації на сервері в реальному часі. Використання RESTful API для комунікації між

автоматами та сервером забезпечить швидку обробку запитів, що дозволить оперативно отримувати актуальні дані без значних затримок. З сервера дані зберігаються у базі даних, що забезпечує їх централізоване зберігання та доступність для подальшого аналізу.

Модуль зберігання та обробки даних. Для зберігання даних про роботу вендингових автоматів використовуватиметься база даних, яка оброблятиме великий обсяг інформації. Залежно від вимог, можна вибрати реляційну базу даних, таку як MySQL або PostgreSQL, що забезпечить високу продуктивність при зберіганні та обробці структурованих даних. Для масштабованих рішень можна також використовувати NoSQL бази даних, наприклад MongoDB, щоб забезпечити гнучкість у зберіганні неструктурованих даних. База даних зберігатиме всі необхідні метадані про автоматах, продажах, рівнях товарів та інші параметри.

Вебсервіс для адміністрування мережі. Вебсервіс буде основним інтерфейсом для адміністрування мережі вендингових автоматів. Він надасть доступ до статистики та звітів, а також дасть можливість налаштовувати автоматизовані процеси обслуговування і керування асортиментом. Адміністратор через вебсервіс зможе отримати актуальну інформацію про стан кожного автомату, здійснювати налаштування автоматів, переглядати звіти про продажі і здійснювати аналіз ефективності роботи автоматів. Для реалізації цього інтерфейсу можна використовувати сучасні фреймворки для створення вебдодатків, такі як Vue.js або React [25].

Система безпеки та автентифікації. Вебсервіс та API повинні бути забезпечені надійним рівнем безпеки для захисту даних і доступу до системи. Для цього передбачено використання методів автентифікації, таких як OAuth або JWT (JSON Web Tokens), які дозволяють захищати доступ до адміністративного інтерфейсу та персональних даних. Додатково, всі запити до сервера повинні здійснюватися через HTTPS для забезпечення шифрування даних при передачі.

Система сповіщень та моніторингу. Щоб своєчасно реагувати на проблеми, система повинна надсилати сповіщення адміністратору або відповідальному персоналу. Це можуть бути сповіщення про технічні збої, низький рівень товару, потребу в обслуговуванні автоматів чи інші критичні події. Сповіщення можуть надсилатися через електронну пошту, SMS або через інші канали зв'язку.

Загалом, така структура забезпечить високу ефективність роботи системи, дозволяючи зібрати дані з великої кількості вендингових автоматів, зберігати їх у централізованій базі даних, а також надати зручний інтерфейс для моніторингу і керування мережею автоматів в реальному часі.

Основні задачі, що стоять перед системою, мають чітке відображення в проектуванні та реалізації програмного забезпечення. Наприклад, автоматизований збір даних з торгових автоматів є ключовою задачею, і для її виконання було розроблено API, яке дозволяє здійснювати запити до кожного автомату та отримувати актуальну інформацію про стан роботи автомату, рівень товарів, фінансові операції тощо. Ці дані передаються на сервер для подальшої обробки і відображення в інтерфейсі системи в реальному часі, що дає змогу отримати детальну статистику.

Для цього у дипломній роботі передбачено використання технології HTTP для взаємодії між ESP32 (мікроконтролером автоматів) та сервером. Завдяки цьому забезпечується стабільна та безперервна передача даних без втрат, що є важливим для ефективного моніторингу.

Зручне адміністрування мережі автоматів є важливим аспектом реалізації системи. Для цього в дипломній роботі передбачено створення інтуїтивно зрозумілого вебінтерфейсу, через який адміністратор може здійснювати налаштування автоматів, переглядати звіти про продажі, виявляти технічні проблеми та відправляти сповіщення. Інтерфейс також включає можливості для дистанційного управління мережою і отримання

оперативної інформації про кожен автомат, що забезпечує гнучкість в управлінні мережею.

Якщо узагальнюючи, то кожна із задач, поставлених для системи, реалізується через конкретні етапи проекту, що включають розробку серверної частини, API, взаємодію з обладнанням, а також створення зручного користувацького інтерфейсу, що дозволяє ефективно керувати мережею вендингових автоматів.

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ВЕНДИНГОВИМИ АВТОМАТАМИ

2.1 Структурна модель системи управління мережею пристроїв

Система управління мережею вендингових автоматів має клієнт-серверну архітектуру, що забезпечує централізовану обробку та зберігання даних.

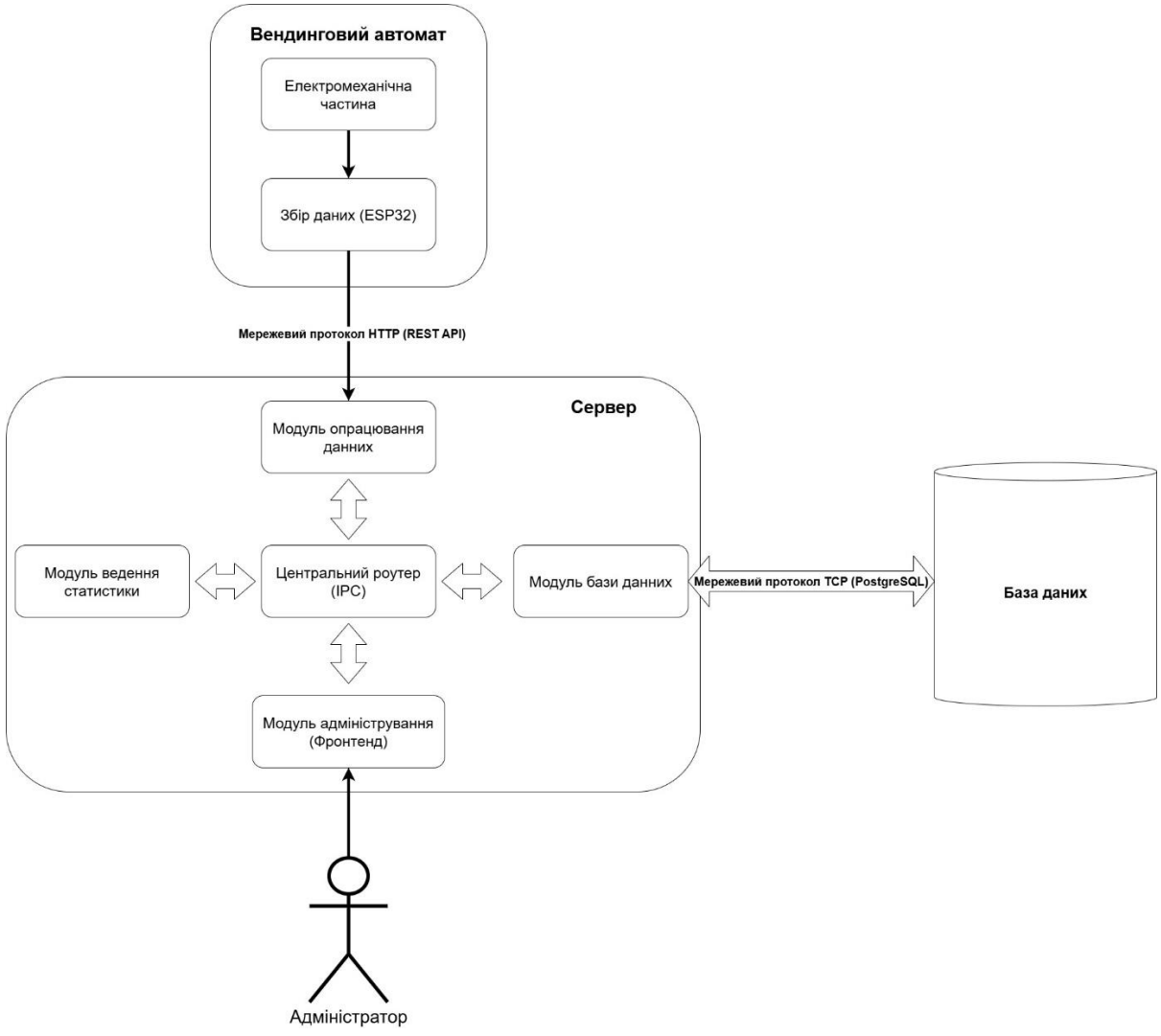

Рисунок 2.1 – Структурна схема системи мережі вендингових автоматів

Основними елементами цієї структури є: вендинговий автомат, сервер, база даних та користувач-адміністратор. Взаємодія між цими елементами представлена на структурній схемі (рис. 2.1).

Вендинговий автомат оснащений мікроконтролером (ESP32), яки збирає статистику про транзакції, кількість товарів, технічний стан пристрою та інші параметри. Зібрані дані надсилаються на сервер за допомогою HTTP-запитів через REST API. Сервер, отримавши запит, виконує його обробку, зберігає інформацію у відповідні таблиці бази даних, а також може генерувати відповідь для автомату, якщо це необхідно (наприклад, команди на оновлення параметрів).

База даних виконує функції довготривалого зберігання інформації, включаючи реєстраційні дані автоматів, історію продажів, інформацію про товари, лог обслуговування тощо. Сервер взаємодіє з базою даних за допомогою SQL-запитів через відповідний протокол бази даних.

Адміністратор взаємодіє з сервером через веб інтерфейс, реалізований як фронтенд, що надсилає запити до REST API. Через цей інтерфейс адміністратор може переглядати інформацію про автомати, генерувати звіти, змінювати налаштування, переглядати статистику та отримувати сповіщення про стан пристроїв.

Система працює у режимі постійного обміну даними між пристроями та сервером, забезпечуючи надійну передачу інформації, її централізовану обробку та візуалізацію для керування мережею торгових автоматів.

Блок-схема, (рис. 2.2) ілюструє алгоритм обробки замовлення, що надходить до серверної частини системи управління мережею вендингових автоматів. Цей алгоритм охоплює повний цикл взаємодії між апаратною частиною (вендінговим автоматом із мікроконтролером ESP32) та сервером, зосереджуючись на перевірці коректності даних, авторизації пристрою та збереженні інформації до бази даних. Реалізація цього процесу є критично важливою для забезпечення цілісності, достовірності та безперервності збору

комерційної інформації про продажі.

Процес розпочинається з початкової умови – ініціації перевірки, чи працює сервер.

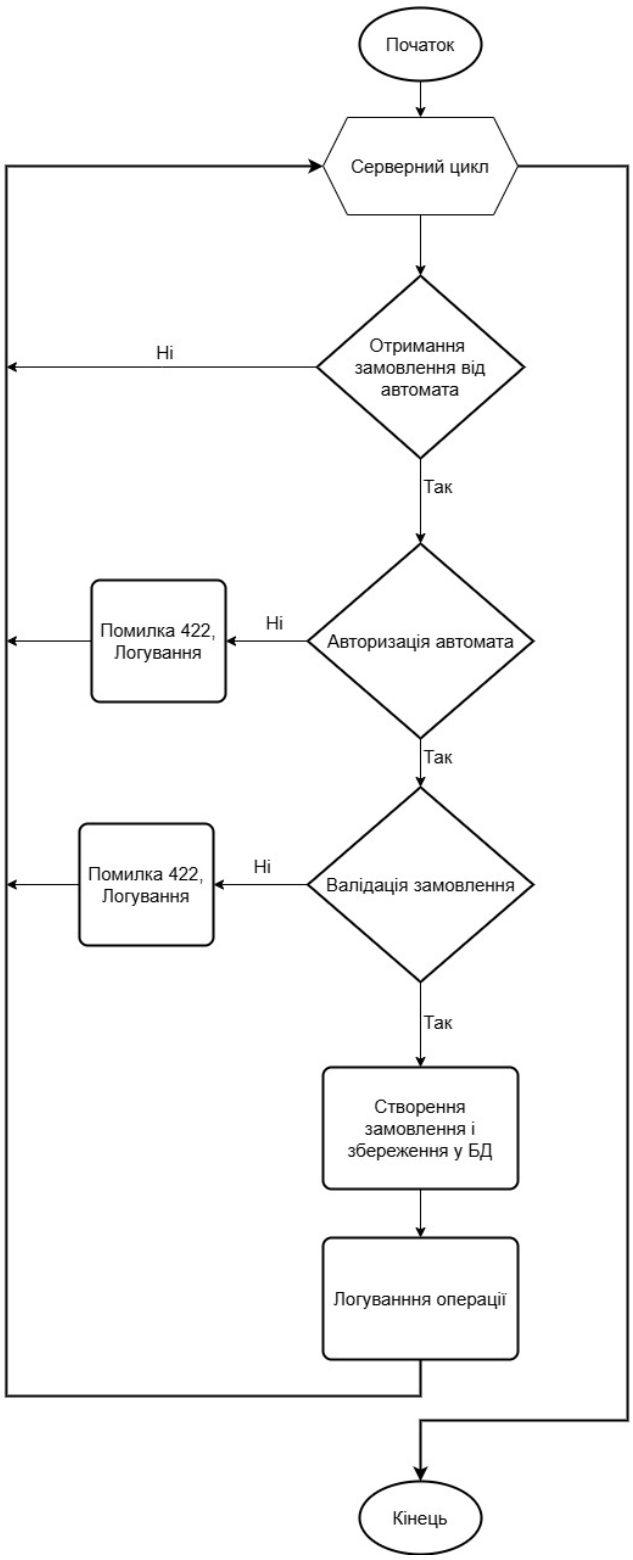

Рисунок 2.2 – Блоксхема алгоритму, управління автоматизованої системою керування вендинговими автоматами.

Якщо серверне середовище є недоступним або перебуває у неробочому стані, подальші кроки алгоритму не виконуються. Це дозволяє уникнути обробки запитів у некоректному стані системи та є базовим етапом перевірки життєздатності системи.

У разі підтвердження працездатності сервера здійснюється прийом замовлення від автомата. На цьому етапі сервер очікує POST-запит, який повинен містити дані у форматі JSON. Якщо запит не відповідає очікуваній структурі (наприклад, містить помилки у форматі або відсутні обов'язкові поля), ініціюється обробка помилки з поверненням статус-коду HTTP 422 (Unprocessable Entity), а сама помилка реєструється у системі логування для подальшого аналізу.

Далі система переходить до етапу авторизації автомата. Для цього перевіряється, чи є в базі даних запис, що відповідає ідентифікатору наданому у запиті (automat_id). Якщо автомат не авторизовано (наприклад, він не зареєстрований або ID вказано з помилкою), знову ж таки формується помилка з кодом 422 і здійснюється логування інциденту.

Наступним етапом є валідація замовлення. Це перевірка достовірності переданих даних щодо самого продажу: назва продукту, кількість, вартість одиниці, а також правильність вказаного часу транзакції. За відсутності помилок структура даних вважається валідною. У протилежному випадку сервер повертає відповідний код помилки (422), а невдала спроба зберігається у логах.

Якщо всі попередні перевірки пройдено успішно, сервер створює новий запис замовлення у базі даних. Це відбувається шляхом ініціалізації об'єкта типу Sale та збереження його до таблиці через ORM-рівень. Після збереження, додатково виконується логування успішної операції з відповідним повідомленням, яке може бути використане для аудиту або моніторингу.

Таким чином, завершення процесу досягається після успішного збереження замовлення та фіксації факту операції в логах. Якщо на будь-

якому з етапів виникає помилка, алгоритм завершується раніше, гарантуючи контрольований вихід із процесу та надання зворотного зв’язку у вигляді помилкового HTTP-коду.

Загалом, представлена блок-схема демонструє структурований і захищений підхід до обробки вхідних даних, з належним контролем помилок та реєстрацією кожного етапу взаємодії, що повністю відповідає вимогам до надійності промислових IoT-систем.

2.2 Функціональний розподіл підсистем

Система управління мережею вендингових автоматів складається з низки взаємопов’язаних підсистем (модулів), кожна з яких виконує окремі функції в межах загального процесу збору, обробки, збереження та аналізу даних. Такий функціональний розподіл забезпечує гнучкість, масштабованість і зручність підтримки всієї системи.

Модуль збору даних реалізується на стороні вендингового автомата на базі мікроконтролера ESP32. Цей компонент відповідає за зчитування технічних параметрів пристрою, фіксацію подій (таких як продаж товарів, поповнення запасів, помилки чи збої в роботі), а також підрахунок кількості товарів у наявності. Дані формуються у структурованому вигляді (JSON) і передаються на сервер за допомогою HTTP-запитів через REST API.

Модуль прийому та обробки запитів функціонує на серверній частині системи та реалізується з використанням фреймворку FastAPI. Його основними завданнями є прийом вхідних запитів як від автоматів, так і від адміністративної частини системи, перевірка достовірності та коректності отриманих даних (валідація), маршрутизація запитів до відповідних внутрішніх сервісів, а також формування відповіді для клієнтів у стандартизованому форматі.

Модуль збереження даних безпосередньо взаємодіє з базою даних і

відповідає за довготривале зберігання всієї інформації про функціонування системи. У базі фіксується реєстраційна інформація про автомати, історія продажів, поточні залишки товарів у кожному автоматі, події обслуговування та облікові записи користувачів. У системі застосовується реляційна база даних PostgreSQL, доступ до якої здійснюється через ORM-бібліотеку SQLModel, що забезпечує зручну інтеграцію та захищену роботу з даними.

Модуль адміністрування реалізований у вигляді вебінтерфейсу адміністратора і є клієнтською частиною системи. Він надає доступ до основних функцій управління мережею автоматів, включно з переглядом поточного стану кожного пристрою в реальному часі, генерацією статистичних звітів за певний період, зміною налаштувань конкретного автомата, а також контролем доступу користувачів до системи [16].

Модуль статистики виконує обробку накопичених у базі даних записів з метою створення аналітичних звітів, які дозволяють оцінити ефективність функціонування мережі. Серед показників, які аналізуються: обсяг продажів, частота технічного обслуговування, найпопулярніші продукти, кількість збоїв або технічних несправностей. Звіти можуть експортуватися у зручні формати, зокрема CSV або PDF, для подальшого використання або архівування.

Модуль безпеки відповідає за автентифікацію та авторизацію користувачів, управління правами доступу до функцій системи, захист REST API-ендпоінтів від несанкціонованого доступу, логування підозрілої активності, а також шифрування конфіденційної інформації. Завдяки цьому забезпечується надійний рівень безпеки як для внутрішніх даних системи, так і для її користувачів.

2.3 Організація обміну даними між модулями

У системі управління мережею вендингових автоматів ключову роль відіграє ефективна організація обміну даними між її компонентами. Для

забезпечення надійної та масштабованої взаємодії використовуються стандартизовані протоколи зв'язку та формати передавання інформації. Комунікація між вендинговими автоматами та сервером здійснюється за допомогою протоколу HTTP з використанням REST API, що дозволяє забезпечити уніфіковану модель взаємодії.

Вендингові автомати надсилають серверу звіти у форматі JSON — це спрощує процес інтеграції та дозволяє зручно обробляти дані на стороні бекенду. REST API також активно використовується зі сторони адміністратора: він надає можливість отримувати звіти, здійснювати моніторинг стану пристроїв, керувати їх налаштуваннями та переглядати аналітичну інформацію.

Такий підхід дозволяє централізовано керувати мережею автоматів і ефективно реагувати на зміни в режимі реального часу [16].

Кожен звіт, що надсилається автоматом, містить набір структурованих полів (рис. 2.3).

```json
{
  "device_id": "VND-001",
  "timestamp": "2025-06-15T15:23:45Z",
  "status": "OK",
  "products": [
    {"product_id": "P01", "remaining": 10},
    {"product_id": "P02", "remaining": 5}
  ],
  "sales": [
    {"product_id": "P01", "quantity": 2, "price": 30.0}
  ],
  "error_code": null
}
```

Рисунок 2.3 – Структура JSON-звітів від автоматів

Це дозволяє серверу оперативно зберігати всі зміни та формувати історію подій у реальному часі. Порядок надсилання, обробки та відповіді на запити побудований за принципом REST-архітектури. Вендинговий автомат

формує звіт, що містить інформацію про власний поточний стан, виконані операції, події та, за необхідності, додаткові параметри, такі як рівень запасів чи внутрішні помилки. Сформований звіт надсилається на визначений HTTP-ендпоінт сервера.

Після отримання запиту сервер виконує низку перевірок: автентифікація пристрою відбувається за допомогою API-ключа або токена, далі здійснюється валідація структури отриманого JSON-документа. У разі успішного проходження перевірок дані зберігаються у відповідних таблицях бази даних. Після збереження сервер повертає відповідь із кодом статусу 200 OK, що сигналізує про успішне завершення обробки. У випадках помилки повертається відповідне повідомлення, наприклад, 400 Bad Request при порушенні формату або 401 Unauthorized у разі невалідного ключа. Якщо зв'язок із сервером тимчасово недоступний, автомат виконує повторну спробу надсилання звіту з фіксованою затримкою відповідно до встановленого інтервалу.

Для забезпечення надійності обміну повідомленнями реалізовано захист від помилок, дублювання та втрати даних. Кожен звіт містить унікальний ідентифікатор транзакції або часову мітку, що дає змогу на стороні сервера ідентифікувати дублікати й запобігти повторній обробці. У разі виявлення збоїв або відсутності відповіді система автоматично виконує повторну відправку з використанням стратегії “retry with backoff”, що поступово збільшує інтервал між спробами. Зі сторони сервера реалізовано додаткові механізми перевірки повторів, логування помилок, а також аварійне збереження вхідних запитів у тимчасове сховище у разі непередбачених винятків.

Передача даних між автоматами та сервером здійснюється з використанням захищеного протоколу HTTPS, що гарантує конфіденційність та цілісність інформації під час обміну. Такий підхід дозволяє забезпечити безпечну та надійну інтеграцію пристроїв з центральною системою

моніторингу та керування.

Внутрішня архітектура серверної частини системи передбачає активну взаємодію з системою управління базами даних (СУБД) для постійного зберігання, оновлення та доступу до всієї операційної та аналітичної інформації. Як основна СУБД обрано PostgreSQL, що обумовлено її високою надійністю, підтримкою транзакційності, масштабованістю, а також розширеними можливостями щодо роботи з просторовими даними та JSON-типами, що є перевагою для зберігання структурованих звітів.

Зв'язок між серверним застосунком та базою даних PostgreSQL встановлюється через протокол TCP/IP. Для забезпечення безпеки та конфіденційності даних, особливо при розгортанні у хмарному середовищі або розподіленій архітектурі, використовуються зашифровані TCP-з'єднання за допомогою SSL/TLS. Це запобігає несанкціонованому перехопленню та модифікації даних під час їх передачі між сервером застосунків та СУБД. Управління пулом з'єднань реалізується за допомогою спеціалізованих бібліотек або ORM (Object-Relational Mapping), що дозволяє ефективно керувати ресурсами, зменшувати накладні витрати на встановлення нових з'єднань та оптимізувати продуктивність при високому навантаженні. Кожен запит на збереження даних, оновлення статусу або отримання інформації від адміністративного інтерфейсу транслюється у відповідні SQL-запити, які виконуються сервером баз даних, забезпечуючи консистентність та цілісність даних у масштабах усієї системи [18].

2.4 Проєктування інформаційної моделі збереження звітних даних

Проєктування бази даних є критичним етапом розробки комп’ютеризованої системи управління мережею вендингових автоматів, оскільки саме від структури даних залежить ефективність зберігання, обробки й аналізу інформації про функціонування пристроїв. Основна мета побудови

інформаційної моделі полягає у забезпеченні цілісного, логічно структурованого подання інформації про автомати, товари, операції продажів і залишки, що дозволить реалізувати ключові функції системи [5].

У межах даного проєкту розроблено реляційну базу даних, яка складається з чотирьох основних таблиць (рис. 2.4):

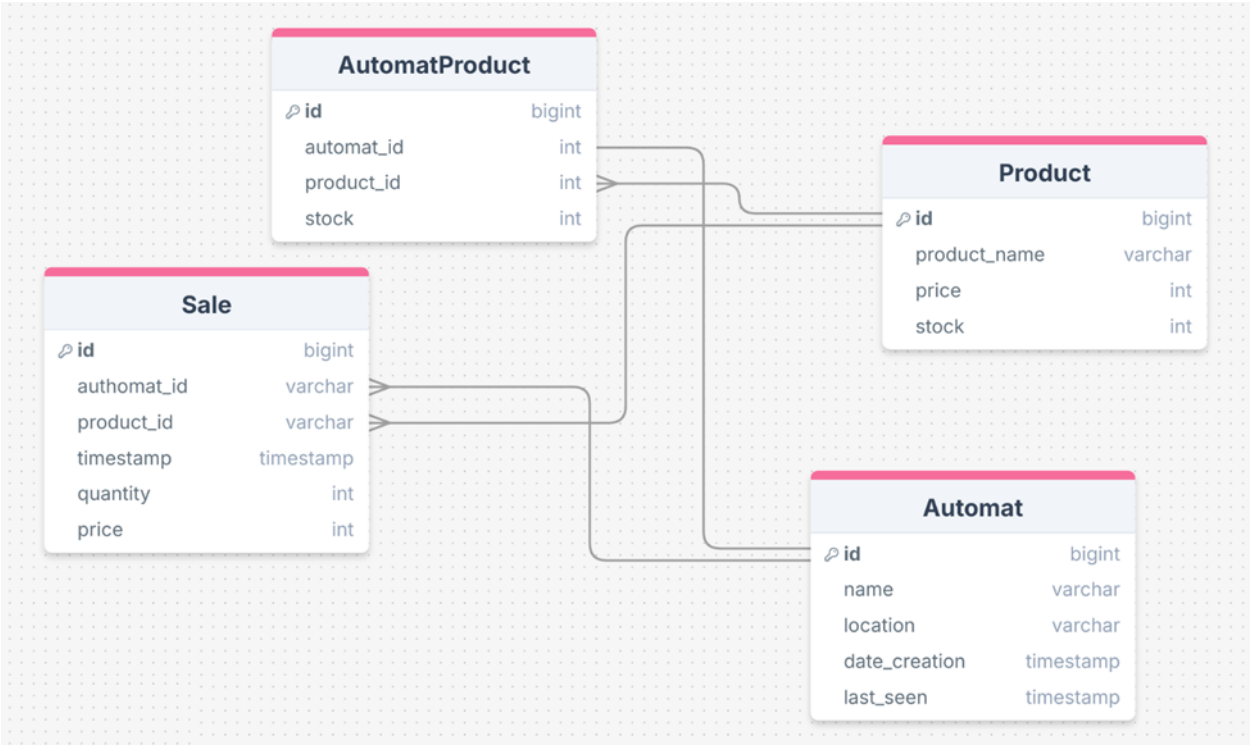

Рисунок 2.4 – Реляційна схема бази даних

У системі управління мережею вендингових автоматів важливою складовою є інформаційна модель бази даних, яка забезпечує структуроване зберігання та обробку звітних даних. Основу цієї моделі становить таблиця Automat, що містить дані про кожен окремий пристрій: унікальний ідентифікатор (id), назву або ідентифікаційну мітку (name), місце встановлення (location), дату додавання в систему (date_creation) та дату останньої активності (last_seen). Ця таблиця слугує точкою відліку для багатьох інших логічних зв’язків у системі.

Інформація про асортимент продукції зберігається в таблиці Product, де кожен запис містить унікальний ідентифікатор товару (id), його назву

(product_name), ціну (price) та загальний запас на складі (stock). Для реалізації зв’язку між автоматами та наявними в них товарами використовується зв’язувальна таблиця AutomatProduct. Вона фіксує, які саме продукти завантажені в кожен конкретний автомат, і в якій кількості. До її атрибутів належать: унікальний id, зовнішні ключі automat_id та product_id, а також поле stock, що вказує на кількість одиниць певного товару в конкретному автоматі.

Факти продажу реєструються у таблиці Sale, яка містить такі атрибути: унікальний ідентифікатор операції (id), посилання на автомат (automat_id) та продукт (product_id), мітку часу продажу (timestamp), кількість реалізованих одиниць (quantity) та підсумкову вартість транзакції (price).

У моделі реалізовано чіткі логічні зв’язки між об’єктами предметної області. Один автомат може містити багато продуктів, і той самий продукт може бути представлений у кількох автоматах. Ці відношення реалізуються через таблицю AutomatProduct. Таблиця Sale, у свою чергу, містить зовнішні ключі до таблиць Automat та Product, що дозволяє точно фіксувати інформацію про кожну продажну операцію, включаючи місце, час та номенклатуру товару.

Структура відповідає вимогам третьої нормальної форми (3NF). Усі атрибути залежні виключно від первинного ключа відповідної таблиці, а функціональні та транзитивні залежності відсутні. Така нормалізація дозволяє уникнути дублювання інформації, зменшити зайве використання пам’яті та мінімізувати ризик логічних помилок при оновленні чи видаленні даних.

Цілісність даних у системі забезпечується за допомогою первинних та зовнішніх ключів. Зокрема, зовнішні ключі automat_id та product_id в таблицях AutomatProduct і Sale гарантують, що всі посилання ведуть виключно на існуючі записи у відповідних таблицях, запобігаючи появі "висячих" або некоректних зв’язків.

Відповідно до розроблених ORM-моделей із використанням бібліотеки SQLModel, наступним кроком є побудова SQL-міграцій, які відображають логічну структуру системи у вигляді фізичних таблиць та зв’язків між ними.

Нижче представлено SQL-код, (рис. 2.5): що відповідає створенню основних таблиць, визначених у моделі, з урахуванням типів даних, первинних і зовнішніх ключів, а також обмежень цілісності.

```sql
-- Створення таблиці automat
CREATE TABLE automat (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    location TEXT NOT NULL,
    date_creation DATETIME NOT NULL DEFAULT (CURRENT_TIMESTAMP),
    last_seen DATETIME
);

-- Створення таблиці product
CREATE TABLE product (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    product_name TEXT NOT NULL,
    price REAL NOT NULL,
    stock INTEGER NOT NULL
);

-- Створення таблиці automat_product
CREATE TABLE automatproduct (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    automat_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    stock INTEGER NOT NULL,
    FOREIGN KEY (automat_id) REFERENCES automat (id),
    FOREIGN KEY (product_id) REFERENCES product (id)
);

-- Створення таблиці sale
CREATE TABLE sale (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    automat_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    timestamp DATETIME NOT NULL,
    quantity INTEGER NOT NULL,
    price REAL NOT NULL,
    FOREIGN KEY (automat_id) REFERENCES automat (id),
    FOREIGN KEY (product_id) REFERENCES product (id)
);
```

Рисунок 2.5 – Вигляд SQL міграцій для бази даних

Алгоритм роботи з базою даних виглядає наступним чином:

1. При додаванні нового пристрою в систему у таблиці Automat створюється новий запис з унікальним ідентифікатором.

2. У разі поповнення автомата товарами оновлюється інформація в таблиці AutomatProduct, де фіксується актуальний перелік і кількість товарів.

3. Після кожного продажу автомат формує звіт, який надсилається на

сервер через REST API. Сервер обробляє запит і додає новий запис у таблицю Sale.

4. Для отримання аналітичної інформації або побудови звітів система виконує агрегаційні SQL-запити до таблиці Sale, об’єднуючи їх з таблицями Automat та Product за допомогою JOIN-зв’язків.

Завдяки такій структурі забезпечується ефективне управління даними, можливість масштабування системи, простота підтримки й розширення, а також точність формування звітів і аналітики.

2.5 Обґрунтування вибору елементної бази пристроїв

Для реалізації системи зв’язку вендингового автомата з сервером було обрано мікроконтролер ESP32, що є оптимальним варіантом для задач, пов’язаних із бездротовою передачею даних, стабільною роботою та гнучким програмуванням (рис. 2.6). Його вбудований модуль Wi-Fi дозволяє напряму підключатися до мережі Інтернет без потреби у додаткових компонентах, що зменшує загальну складність та вартість пристрою.

Рисунок 2.6 – Зовнішній вигляд ESP32

Потужний двоядерний процесор із тактовою частотою до 240 МГц забезпечує достатню обчислювальну здатність для роботи зі стеками

мережевих протоколів та обробки даних. Завдяки підтримці енергозберігаючих режимів (зокрема, deep sleep), ESP32 може працювати тривалий час автономно на акумуляторах або батареях, що важливо для розміщення автоматів у місцях з обмеженим доступом до електромережі.

Додатковою перевагою ESP32 є сумісність з Arduino IDE та іншими популярними середовищами розробки, що значно спрощує створення прошивки, зокрема для реалізації обміну даними з сервером через HTTP або MQTT. Велика кількість виводів дає змогу підключати сенсори, кнопки, світлодіоди та інші електронні компоненти. У даному проєкті до ESP32 підключено кнопки (для симуляції вибору продукту) та світлодіоди (для індикації подій, наприклад, здійснення продажу) [22].

Окрім технічних характеристик, важливу роль відіграє й економічна складова: ESP32 — це доступний, масово вироблений мікроконтролер, який легко закупити у великих кількостях, що робить його ідеальним вибором для комерційного впровадження системи у широку мережу автоматів.

У системі ESP32 виконує функції шлюзу між периферійними компонентами автомата та сервером. Він зчитує дані про кількість товару, події продажів, технічний стан пристрою, формує звіт у форматі JSON і надсилає його на сервер через Wi-Fi, використовуючи HTTP-запит. Таким чином, ESP32 виступає як повноцінний вузол IoT-системи, що поєднує високу функціональність, гнучкість і придатність до масштабного розгортання.

ESP32 відіграє центральну роль у системі, забезпечуючи двосторонній зв'язок. Він постійно опитує підключені сенсори та компоненти вендингового автомата, збираючи інформацію про залишок товарів, стан грошового приймача, температуру всередині автомата, спрацювання датчиків відкриття дверей та інші показники, необхідні для моніторингу та управління. Зібрані дані обробляються та агрегуються ESP32, при цьому події продажу фіксуються, ініціюються оновлення кількості товару, а також збираються дані про помилки або несправності.

Оброблені дані упаковуються в структурований формат, найчастіше JSON, який є компактним, легко читабельним і широко підтримуваним веб-сервісами, що спрощує подальшу обробку на стороні сервера. Для забезпечення конфіденційності та цілісності даних ESP32 може використовувати зашифровані HTTP-запити (HTTPS), що є критично важливим для передачі чутливої інформації. Після формування пакета даних ESP32 ініціює HTTP-запит до API сервера. Сервер отримує ці дані, обробляє їх, зберігає в базі даних і може надсилати відповідні команди назад до ESP32. Ще однією значною перевагою ESP32 є можливість оновлення програмного забезпечення віддалено, без фізичного доступу до пристрою, що дозволяє легко впроваджувати нові функції, виправляти помилки та покращувати безпеку системи протягом її життєвого циклу.

В основі ESP32 лежить система-на-кристалі (SoC), яка інтегрує двоядерний процесор Tensilica Xtensa LX6. Кожне ядро працює незалежно, що дозволяє розподіляти обчислювальні завдання (рис. 2.7).

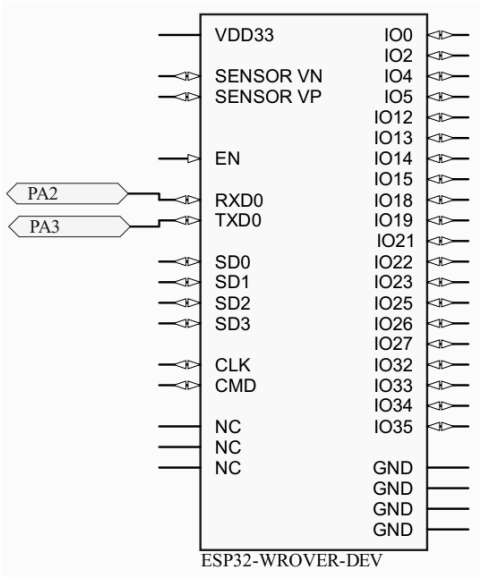

Рисунок 2.7 – Електрично принципова схема платформи ESP32

Одне ядро, як правило, відповідає за виконання основної логіки програми, включаючи зчитування даних із периферійних пристроїв, обробку

внутрішніх подій та формування звітів. Інше ядро може бути задіяне для більш інтенсивних операцій, таких як підтримка мережевого стеку Wi-Fi, обробка криптографічних алгоритмів для HTTPS-з'єднань та управління чергами передачі даних. Такий архітектурний підхід значно підвищує ефективність та відгук системи, дозволяючи одночасно виконувати операції збору даних та підтримувати стабільний зв'язок із сервером без затримок.

Крім обчислювальних ядер, ESP32 має інтегровану пам'ять, включаючи SRAM та Flash-пам'ять. SRAM використовується для зберігання тимчасових даних та стека програми, тоді як Flash-пам'ять призначена для зберігання прошивки (програмного забезпечення) та постійних даних, таких як конфігураційні параметри. Внутрішня архітектура також включає численні периферійні контролери: GPIO (для керування входами/виходами), UART (для послідовного зв'язку з іншими компонентами), SPI та I2C (для підключення зовнішніх сенсорів та дисплеїв). Всі ці компоненти працюють злагоджено під керуванням операційної системи реального часу (RTOS), такої як FreeRTOS, яка оптимізує планування завдань і ресурсів, забезпечуючи надійну та передбачувану роботу системи зв'язку вендингового автомата.

ESP32 ідеально вписується в концепцію Інтернету речей (IoT) для вендингових автоматів. Він є "крайовим" пристроєм (edge device), який збирає дані безпосередньо з фізичного світу і передає їх у "хмару" для аналізу та прийняття рішень. Такий підхід дозволяє моніторинг у реальному часі, коли власники автоматів можуть відстежувати продажі, наявність товарів та технічний стан у будь-який час. На основі зібраних даних можна передбачати потенційні поломки або необхідність поповнення товару, оптимізуючи маршрути обслуговування.Збір та аналіз даних про продажі допомагає в оптимізації асортименту, ціноутворення та розміщення автоматів [24].

Завдяки своїм технічним характеристикам, гнучкості та економічній ефективності, ESP32 є наріжним каменем для побудови сучасної, інтелектуальної системи зв'язку вендингових автоматів із центральним

сервером, забезпечуючи високий рівень автоматизації та контролю.

Для емуляції вендингового апарата було розроблено макет апаратної частини на базі мікроконтролера ESP32, змонтований на безпайковій макетній платі (рис. 2.8).

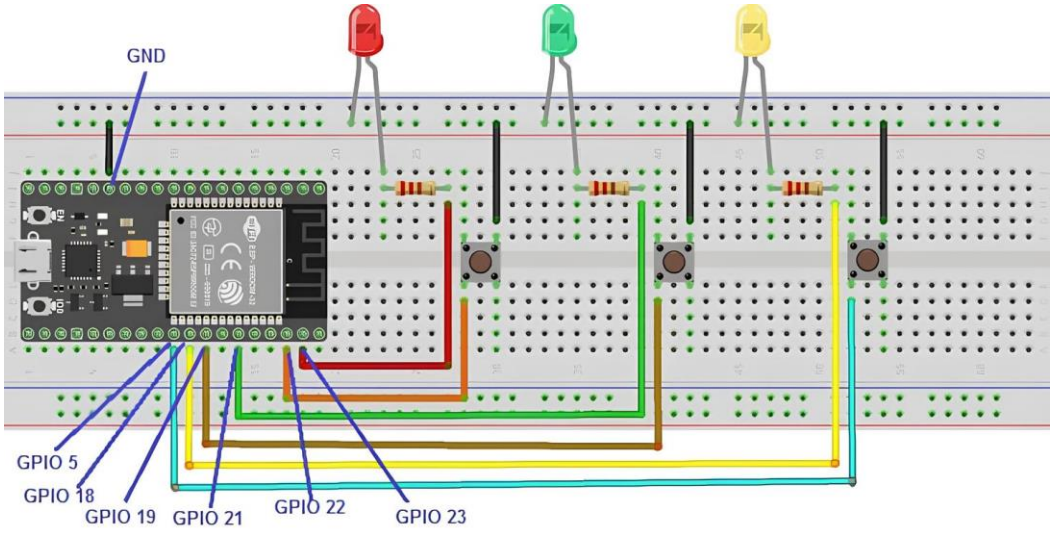

Рисунок 2.8 – Схема підключень емулятора вендингового автомата

Конструкція передбачає підключення трьох світлодіодів (червоного, зеленого та жовтого кольорів), які імітують світлові індикатори стану, а також трьох кнопок, що відтворюють події взаємодії користувача з автоматом. Кожен світлодіод з'єднаний послідовно з резистором для обмеження струму та підключений до окремого цифрового виходу мікроконтролера: червоний — до GPIO18, зелений — до GPIO19, жовтий — до GPIO5. Кнопки з'єднані із входами GPIO21, GPIO22 та GPIO23 відповідно, при цьому застосовано підтягування до землі через резистори, а подача сигналу здійснюється на високий рівень при натисканні. Загальна "земля" (GND) всіх компонентів об'єднана з відповідним контактом ESP32 для забезпечення єдиного потенціалу. Така конфігурація дозволяє протестувати логіку обробки подій, візуального індикаційного супроводу та загальну працездатність прототипу перед його інтеграцією з серверною частиною системи.

.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ВЕНДИНГОВИМИ АВТОМАТАМИ

3.1 Модуль обробки вхідних повідомлень

Однією з ключових складових системи керування мережею вендингових автоматів є модуль, який відповідає за прийом інформації про продажі, що надходить безпосередньо з пристроїв на базі ESP32. Цей модуль побудовано на базі фреймворку FastAPI, що дозволяє створювати високопродуктивні вебсервіси з використанням сучасних підходів до обробки HTTP-запитів.

Модуль прийому реалізовано у вигляді HTTP-ендпоінту POST /api/sales, який очікує надходження JSON-документів від вендингових автоматів. Основною метою цього ендпоінту є перевірка валідності надісланих даних, їх збереження до бази даних та повернення відповідного статусу у відповідь (лістинг 3.1).

```
1  class SaleCreate(BaseModel):
2      automat_id: int
3      product_name: str
4      price: float
5      quantity: int
6      timestamp: datetime
7
```

Лістинг 3.1 – Модель замовлення

Спочатку, FastAPI використовує механізм автоматичної валідації даних за допомогою Pydantic-схем. У нашій реалізації було визначено клас SaleCreate, який описує очікувану структуру вхідного JSON. Зокрема, дані

мають містити automat_id, product_name, price, quantity та timestamp. Завдяки цьому підходу вже на етапі отримання запиту виконується перевірка формату та наявності усіх обов'язкових полів, що значно підвищує надійність і безпеку системи.

Після проходження валідації, система переходить до основної логіки обробки. Перш за все, виконується перевірка наявності автомата з вказаним automat_id у базі даних (лістинг 3.2). Якщо автомат не зареєстрований у системі, сервер повертає повідомлення про помилку з кодом 404, що попереджає про спробу несанкціонованої передачі даних [16].

```
automat = session.get(Automat, sale_data.automat_id)
if automat is None:
    raise HTTPException(status_code=404, detail="Automat not found")
```

Лістинг 3.2 – Фрагмент коду перевірки автомата

У випадку успішної перевірки, наступним кроком є пошук продукту за назвою (лістинг 3.3). Оскільки пристрій передає лише назву продукту, а не його ідентифікатор, необхідно виконати SQL-запит для отримання відповідного запису з таблиці Product.

```
product = session.exec(select(Product).where(Product.product_name == sale_data.product_name)).first()
if product is None:
    raise HTTPException(status_code=404, detail="Product not found")

```

Лістинг 3.3 – Фрагмент коду пошуку продукту

Далі створюється об'єкт продажу — екземпляр моделі Sale, який наповнюється даними з вхідного запиту (лістинг 3.4). Його атрибути включають посилання на відповідний автомат і продукт, кількість проданого товару, час продажу та ціну.

```python
sale = Sale(
    automat_id=sale_data.automat_id,
    product_id=product.id,
    quantity=sale_data.quantity,
    price=sale_data.price,
    timestamp=sale_data.timestamp
)

```

Лістинг 3.4 –Вигляд об’єкта продажу

Після цього дані зберігаються до бази за допомогою SQLModel, а сесія комітиться (лістинг 3.5). У разі виникнення помилки, наприклад через втрату з’єднання або порушення цілісності даних, спрацьовує механізм обробки винятків, який дозволяє або повернути повідомлення про помилку, або занотувати її до системного журналу.

```python
session.add(sale)
session.commit()
session.refresh(sale)
```

Лістинг 3.5 –Комміт об’єкта продажі у базі даних

Після обробки продажу виконується оновлення часу останньої активності автомата (last_seen), (лістинг 3.6) що дозволяє системі в режимі реального часу відслідковувати активність кожного з підключених пристроїв.

```python
automat.last_seen = datetime.now()
session.add(automat)
session.commit()
```

Лістинг 3.6 –Оновлення стану автомата

Окрему увагу приділено системі логування. Для цього застосовується

стандартний модуль logging, який дозволяє фіксувати як інформаційні події (успішне збереження даних), так і критичні помилки (відсутність автомата, збої у БД тощо) (лістинг 3.7). Завдяки логам адміністратор може оперативно реагувати на неполадки в роботі системи або окремих автоматів.

```
1 logger.info(f"Sale recorded: Automat ID {sale_data.automat_id}, Product {sale_data.product_name}")
2
```

Лістинг 3.7 –Логування замовлення

Підсумовуючи, модуль обробки вхідних повідомлень виконує декілька ключових функцій: валідацію структури запиту, перевірку існування автомата і товару, збереження інформації про продаж у базу даних, оновлення статусу автомата, а також реєстрацію подій у логах. Такий підхід забезпечує надійність та масштабованість системи при зростанні кількості пристроїв.

3.2 Побудова серверної логіки обробки даних

Після реалізації модуля прийому даних із вендингових автоматів, наступним ключовим етапом є побудова серверної логіки для обробки накопиченої інформації, формування запитів до бази даних, а також створення відповідних API-ендпоінтів для взаємодії з клієнтською частиною. Даний підпункт описує архітектуру запитів, реалізацію обчислення прибутку та агрегаційної статистики, а також надає приклади коду, що демонструють організацію цієї логіки у середовищі FastAPI.

У проєкті використовується бібліотека SQLModel, яка забезпечує зручний ORM-шар на основі SQLAlchemy з підтримкою типізації Pydantic. Для отримання аналітичної інформації система реалізує запити із використанням агрегатних функцій, таких як SUM, COUNT, Для отримання загального прибутку системи реалізовано наступний запит (лістинг 3.8) [19]:

```python
from sqlmodel import select, func
from app.models import Sale
from app.db import SessionLocal

def get_total_revenue():
    with SessionLocal() as session:
        total = session.exec(
            select(func.sum(Sale.price * Sale.quantity))
        ).one()
        return total
```

Лістинг 3.8 –Функція отримання загального прибутку

У даному прикладі здійснюється множення ціни на кількість одиниць товару в кожному продажу, після чого результати агрегуються через SUM. Аналогічно реалізовується підрахунок загальної кількості продажів (лістинг 3.9):

```python
def get_total_sales_count():
    with SessionLocal() as session:
        count = session.exec(select(func.count()).select_from(Sale)).one()
        return count
```

Лістинг 3.9 –Підрахунок загальної кількості продажів

Для побудови статистики за кожним автоматом необхідно виконати групування по automat_id (лістинг 3.10):

```python
def get_automat_summary():
    with SessionLocal() as session:
        results = session.exec(
            select(
                Sale.automat_id,
                func.sum(Sale.price * Sale.quantity).label("total_revenue"),
                func.count(Sale.id).label("sales_count")
            ).group_by(Sale.automat_id)
        ).all()
        return results
```

Лістинг 3.10 –Метод побудови статистики

Ці результати дозволяють збудувати узагальнену таблицю з фінансовими та кількісними показниками по кожному окремому автомату.

API-сервер побудовано з використанням FastAPI – сучасного асинхронного фреймворку для побудови RESTful-сервісів. Для взаємодії з фронтендом реалізовано декілька ендпоінтів, що повертають або повні HTML-сторінки через Jinja2Templates, або JSON-дані у відповідь на AJAX-запити (через HTMX) [16].

Ендпоінт для загальної статистики (лістинг 3.11):

```python
from fastapi import APIRouter
from fastapi.responses import JSONResponse

router = APIRouter()

@router.get("/api/stats/summary")
def stats_summary():
    return JSONResponse({
        "total_revenue": get_total_revenue(),
        "total_sales": get_total_sales_count()
    })

```

Лістинг 3.11 –Функція повернення статистики

Для отримання даних по конкретному автомату реалізовано обробник який відповідає за обчислення зведеної статистики продажів для одного вендингового автомата за його унікальним ідентифікатором. У тілі функції відбувається відкриття сеансу взаємодії з базою даних за допомогою контекстного менеджера SessionLocal(), що забезпечує безпечне й гарантоване закриття сесії після завершення запиту. У тілі функції відбувається відкриття сеансу взаємодії з базою даних за допомогою контекстного менеджера SessionLocal(), що забезпечує безпечне й гарантоване закриття сесії після завершення запиту (лістинг 3.12).

```python
@router.get("/api/stats/automat/{automat_id}")
def automat_stats(automat_id: int):
    with SessionLocal() as session:
        revenue = session.exec(
            select(func.sum(Sale.price * Sale.quantity))
            .where(Sale.automat_id == automat_id)
        ).one()
        sales = session.exec(
            select(func.count())
            .where(Sale.automat_id == automat_id)
        ).one()
    return {
        "revenue": revenue,
        "sales": sales
    }
```

Лістинг 3.12 – Функція повернення статистики по конкретному автомату

Для фронтенду ці API є джерелом даних, які через HTMX або Chart.js виводяться у вигляді графіків, таблиць або карток зі статистикою. Використання асинхронної архітектури FastAPI дозволяє забезпечити швидку реакцію сервера навіть при високому навантаженні.

Щоб уникнути надмірного навантаження на БД, частина запитів може кешуватися або агрегуватися у фонових процесах. Також рекомендується використання індексів на полях automat_id, timestamp для прискорення фільтрації й сортування [19].

3.3 Формування звітності на основі зібраної інформації

Формування звітності є ключовою функцією системи, що дозволяє адміністраторам аналізувати ефективність мережі вендингових автоматів та приймати обґрунтовані управлінські рішення. Серверна логіка забезпечує обробку накопичених даних для створення інформативних та візуально зрозумілих звітів.

Основна логіка формування звітів базується на агрегаційних запитах до бази даних, аналогічних тим, що використовуються для підрахунку загальної

статистики. Однак, для звітності додається можливість динамічної фільтрації та сортування за різними параметрами, такими як:

– Період часу: користувач може вибрати довільний діапазон дат (наприклад, день, тиждень, місяць) для аналізу продажів.

– Конкретний автомат: можливість перегляду статистики по одному або декількох обраних автоматах.

– Категорія товару: фільтрація даних для аналізу популярності окремих продуктів.

Для реалізації цієї функціональності розроблено API-ендпоінти, що приймають параметри фільтрації та повертають відформатовані дані. Наприклад, для отримання звіту про продажі за певний період використовується наступний запит (лістинг 3.13):

```python
@router.get("/api/reports/sales_by_date")
def get_sales_report(start_date: date, end_date: date):
    with SessionLocal() as session:
        report_data = session.exec(
            select(Sale)
            .where(Sale.timestamp >= start_date, Sale.timestamp <= end_date)
        ).all()
        return report_data
```

Лістинг 3.13 – Приклад функції для отримання звіту з фільтрацією за датою

Для візуалізації даних, зокрема для побудови графіків, сервер агрегує дані за днями або годинами. Наприклад, для побудови графіка динаміки продажів за тиждень, дані групуються за датою, а потім підсумовуються щоденні прибутки (лістинг 3.14):

```python
from sqlalchemy import func

def get_daily_revenue(start_date: date, end_date: date):
    with SessionLocal() as session:
        results = session.exec(
            select(
                func.date(Sale.timestamp).label("day"),
                func.sum(Sale.price * Sale.quantity).label("total_revenue")
            )
            .where(Sale.timestamp.between(start_date, end_date))
            .group_by(func.date(Sale.timestamp))
            .order_by(func.date(Sale.timestamp))
        ).all()
        return results
```

Лістинг 3.14 – Агрегація даних для графіка

Отримані дані передаються на фронтенд, де за допомогою бібліотеки Chart.js або аналогічної, перетворюються на інтерактивні графіки (лінійні, стовпчасті), що дозволяє користувачеві наочно оцінити тенденції продажів, виявити пікові навантаження та проаналізувати ефективність роботи мережі.

3.4 Розробка інтерфейсу для перегляду та контролю даних

Інтерфейс адміністратора є основним інструментом для взаємодії з системою. Він розроблений з урахуванням принципів UX/UI для забезпечення інтуїтивності, зручності та функціональності. Вебінтерфейс реалізовано з використанням сучасних вебтехнологій, що забезпечують швидку та динамічну взаємодію без перезавантаження сторінки. Розглянемо основні компоненти інтерфейсу [21].

Панель моніторингу – це головна сторінка, на якій відображаються ключові показники ефективності: загальний прибуток, кількість продажів за сьогодні, кількість активних автоматів та останні події (рис. 3.1). Картки з показниками оновлюються в реальному часі за допомогою HTMX-запитів до API, що забезпечує актуальність відображеної інформації без необхідності повного оновлення сторінки.

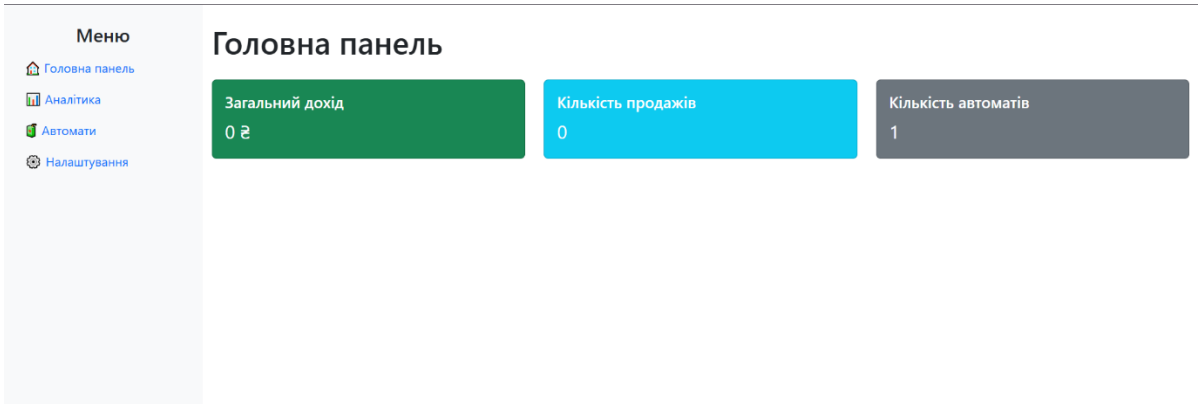

Рисунок 3.1 – Вигляд головної панелі моніторингу

Сторінка "Список автоматів" є центральним місцем для огляду та базового управління всією мережею вендингових пристроїв. Вона представлена у вигляді інтерактивної таблиці, де для кожного зареєстрованого в системі автомата відображається його ключова інформація. Це включає унікальну назву автомата, що дозволяє легко ідентифікувати кожен пристрій, а також його точне місцезнаходження, що є критично важливим для логістики обслуговування.

Особливу увагу приділено відображенню поточного статусу автомата: "онлайн" або "офлайн". Ця інформація оновлюється в реальному часі і дозволяє адміністратору миттєво визначити, які автомати функціонують належним чином, а які можуть потребувати уваги через втрату зв'язку. Додатково відображається час останньої активності, що дає змогу оцінити актуальність даних і швидкість реакції системи (рис. 3.2).

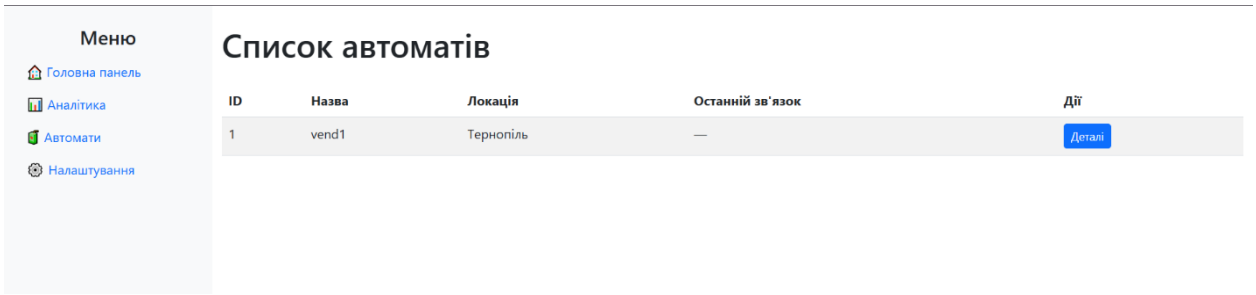

Рисунок 3.2 – Вигляд панелі списку автоматів

Сторінка "Налаштування автоматів" є невід'ємною частиною адміністративного інтерфейсу розробленої системи управління вендинговими автоматами, (рис. 3.3) призначеною для централізованої конфігурації та адміністрування торгових пристроїв. Архітектурно, сторінка виконана у стандартному для веб-додатків форматі, поєднуючи бічне навігаційне меню з основною робочою областю. Заголовок "Налаштування автоматів" чітко ідентифікує функціональне призначення розділу. З метою розширення мережі, праворуч від заголовка інтегрована кнопка "Додати новий автомат", яка ініціює процес реєстрації нового вендингового пристрою в системі.

Центральним елементом сторінки є табличне представлення даних, що містить вичерпний перелік усіх зареєстрованих автоматів. Кожен рядок таблиці надає унікальну ідентифікаційну інформацію про автомат, включаючи його назву (наприклад, "vend1") та точне географічне місцезнаходження ("Тернопіль"), що є критично важливим для логістичних операцій та ефективного планування обслуговування. Додатково, для кожного автомата вказується дата його створення в системі (наприклад, "2023-06-15"). Важливим показником є "Останній зв'язок", який відображає час або дату останньої успішної комунікації автомата із сервером. У випадку, коли значення відображається як "Н/Д", це може вказувати на первинну реєстрацію пристрою без подальшого встановлення зв'язку або на наявність технічних проблем з комунікацією. У повноцінно функціонуючих системах цей параметр є ключовим індикатором актуальності даних та працездатності пристрою в реальному часі. Для забезпечення повного контролю над кожним автоматом у колонці "Дії" передбачені кнопки "Редагувати" для модифікації параметрів пристрою та "Видалити" для його деактивації або повного вилучення з реєстру системи.

Навігаційне меню, розташоване в лівій частині екрану, забезпечує інтуїтивно зрозумілу навігацію між ключовими функціональними модулями системи: "Головна панель" для швидкого огляду основних показників, "Аналітика" для доступу до звітів та статистичних даних, "Автомати" для

загального моніторингу стану пристроїв та "Налаштування" як поточний активний розділ. Ця архітектура сприяє ефективній взаємодії адміністратора з системою, забезпечуючи швидкий доступ до необхідної інформації та функціоналу управління. Таким чином, сторінка "Налаштування автоматів" є фундаментальним компонентом для підтримки актуальності даних про вендингові пристрої та оперативного виконання адміністративних завдань, що забезпечує стабільне та ефективне функціонування всієї мережі [20].

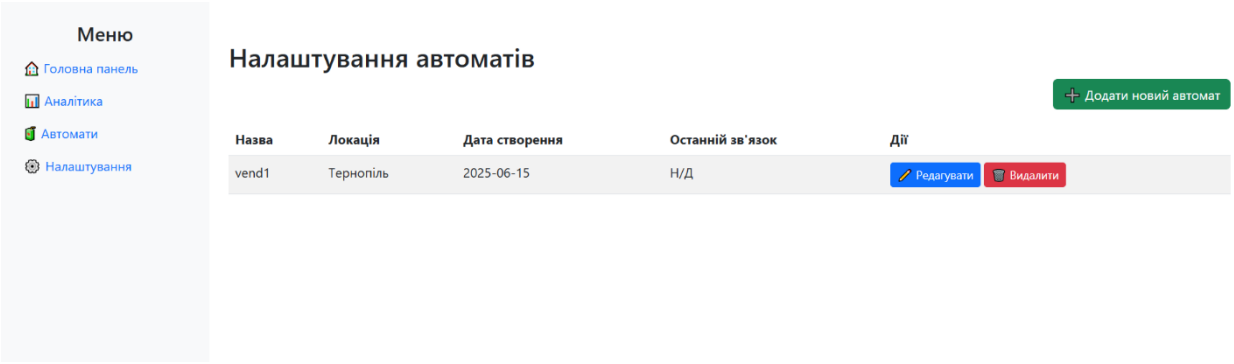

Рисунок 3.3 – Вигляд панелі налаштування автоматів

Зі сторінки "Список автоматів" адміністратор може перейти на сторінку детальної статистики для кожного окремого автомата (рис. 3.4). Ця сторінка є розширеним дашбордом конкретного пристрою і надає вичерпну інформацію про його роботу. Тут можна побачити детальні дані про продажі за певний період, включаючи графіки динаміки, найпопулярніші товари та середній чек. Також відображається актуальний перелік товарів, що знаходяться в автоматі, з їхньою поточною кількістю, що дозволяє ефективно планувати поповнення.

Крім комерційних даних, сторінка детальної статистики містить технічну інформацію, таку як журнал подій, який фіксує всі важливі операції та можливі помилки (наприклад, застрягання монети, відсутність сигналу, несправність датчиків). Це допомагає швидко діагностувати проблеми та вживати заходів для їх усунення. Також доступні опції для віддаленого управління, такі як зміна цін на товари, тимчасове відключення слотів або

перезавантаження модуля зв'язку автомата, що значно підвищує ефективність обслуговування та зменшує час простою обладнання.

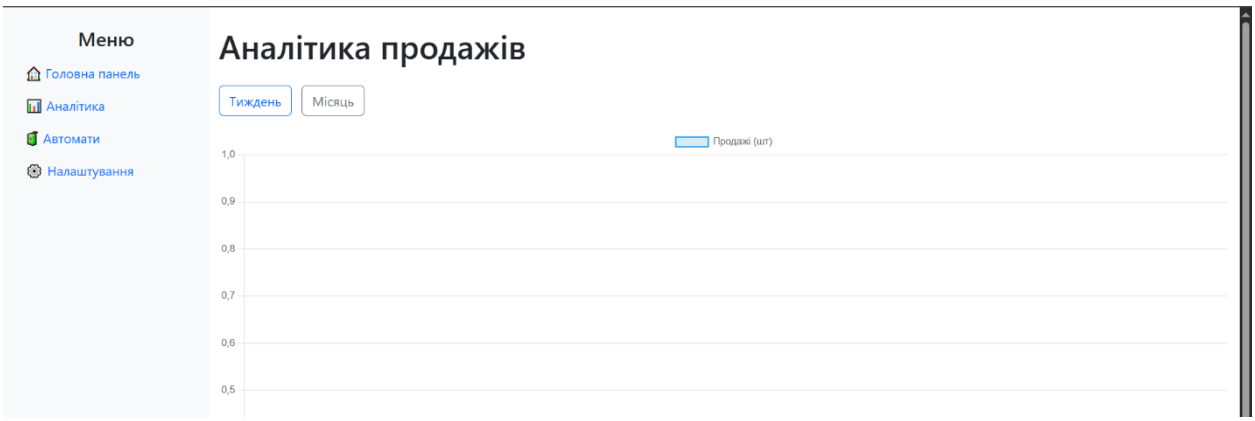

Рисунок 3.4 – Вигляд панелі аналітики

Інтерфейс побудовано за принципом "mobile-first", що забезпечує коректне відображення на пристроях з різною роздільною здатністю екрана. Інтерактивні елементи, такі як кнопки та фільтри, мають чіткі візуальні стани (hover, active, disabled), що покращує користувацький досвід [20].

Система сповіщень реалізована для інформування адміністратора про критичні події. Наприклад, якщо автомат не виходив на зв'язок протягом певного часу, його статус у списку змінюється на "офлайн", а у верхній частині інтерфейсу з’являється відповідне сповіщення. Помилки, що виникають на стороні сервера (наприклад, неможливість завантажити дані), також обробляються і виводяться у вигляді зрозумілих для користувача повідомлень.

3.5 Тестування функціональних сценаріїв роботи системи

Тестування є невід'ємним і критично важливим етапом розробки, що дозволяє всебічно перевірити надійність, стабільність та коректність роботи всіх компонентів системи. Для забезпечення повноти перевірки застосовувався комплексний підхід, що включав тестування на різних рівнях: від модульного (unit-тести) до інтеграційного та мануального тестування всієї

системи. Основна увага була зосереджена на перевірці ключових функціональних сценаріїв, які охоплюють повний життєвий цикл даних: від моменту їх відправки вендинговим автоматом до візуалізації в адміністративному інтерфейсі.

Першим і найважливішим сценарієм була перевірка успішної передачі звіту. В рамках цього тесту емулятор пристрою на базі ESP32 надсилав коректно сформований POST-запит, що містив дані про продаж, на відповідний ендпоінт /api/sales. Як і очікувалося, сервер успішно обробив запит, повернувши у відповідь статус 200 OK. Аналіз стану бази даних підтвердив, що в таблицях Sale та Automat було створено нові записи, а дані на панелі моніторингу в інтерфейсі адміністратора, зокрема показники прибутку та кількості продажів, оновилися в режимі реального часу. Цей тест підтвердив працездатність основного бізнес-процесу системи.

Наступним кроком стало тестування реакції системи на позаштатні ситуації, зокрема, на спробу передачі звіту від незареєстрованого автомата. Для цього було надіслано запит, що містив automat_id, який свідомо був відсутній у базі даних. Система коректно ідентифікувала цю спробу як неавторизовану. Серверна логіка відхилила запит і повернула відповідь з кодом помилки 404 Not Found та інформативним повідомленням "Automat not found", як і було передбачено. Важливо, що жодних змін у базі даних не відбулося, що свідчить про надійний захист від несанкціонованого внесення даних [6].

Ще одним критичним сценарієм була перевірка стійкості до передачі некоректних даних. Під час цього тесту на сервер надсилалися запити з пошкодженою або неповною структурою JSON-документа — наприклад, з відсутнім обов'язковим полем price або з невідповідним типом даних. Завдяки вбудованому механізму валідації на основі Pydantic, система миттєво розпізнавала такі запити і повертала код помилки 422 Unprocessable Entity. Це підтвердило, що механізми валідації ефективно запобігають потраплянню пошкоджених або неконсистентних даних до бази даних, забезпечуючи її

цілісність.

Нарешті, було проведено тестування сценарію втрати зв'язку з сервером, що є поширеною проблемою для IoT-пристроїв. Під час тесту імітувалася тимчасова недоступність сервера в момент, коли емулятор ESP32 намагався надіслати звіт. Прошивка мікроконтролера, що містить логіку повторних спроб відправки через зростаючі інтервали, спрацювала коректно. Після відновлення роботи сервера емулятор успішно надіслав накопичені дані, які були коректно оброблені та збережені. Цей результат продемонстрував стійкість системи до проблем з мережею та її здатність забезпечувати доставку даних без втрат. Окрім перелічених функціональних сценаріїв, окрему увагу було приділено тестуванню швидкості відгуку адміністративного інтерфейсу та його здатності обробляти великі обсяги даних.

Результати проведеного тестування підтвердили, що розроблена система повністю відповідає заявленим функціональним вимогам, є стабільною та спроможною адекватно обробляти як стандартні робочі процеси, так і різноманітні помилкові ситуації. Це створює міцну основу для подальшого розгортання та експлуатації системи у реальних умовах [4].

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Характеристика життєдіяльності людини у системі "людина - машина - середовище існування"

Система "людина - машина - середовище існування" є складною взаємодією трьох основних компонентів: людини, технічних засобів і навколишнього середовища, в якому відбувається ця взаємодія. Людина в цій системі виконує ключову роль, оскільки вона є як активним учасником процесу керування технічними пристроями, так і об’єктом впливу середовища, в якому ця діяльність здійснюється. Машина, у свою чергу, виступає як інструмент розширення можливостей людини, забезпечуючи виконання складних або повторюваних операцій, підвищення продуктивності праці, точності та ефективності. Проте для ефективного функціонування системи необхідно враховувати особливості фізіології та психіки людини, обмеження її уваги, швидкості реакції, стійкості до навантажень, а також ризики, пов’язані з перевтомою чи стресом.

У КРБ враховано що людина має обмежену здатність до тривалої концентрації уваги, зниження якої з часом може призводити до виникнення помилок, особливо при рутинних операціях. В умовах автоматизованих та інформаційно-насичених середовищ додатковим навантаженням є необхідність обробки великого обсягу інформації в обмежений час. Це може викликати ментальне перенапруження, що впливає як на якість прийняття рішень, так і на загальний стан здоров’я працівника. Враховані біоритми, особистісні особливості та рівень підготовки фахівця, а також передбачені психофізіологічні методи оцінки працездатності [9].

Середовище існування, в якому розташовується людина разом із технікою, накладає додаткові вимоги до безпеки та комфорту. Температурний режим, освітлення, рівень шуму, вібрації, наявність шкідливих речовин або електромагнітного випромінювання – усе це може безпосередньо впливати на працездатність, здоров'я та загальну ефективність взаємодії людини з машиною. Важливо забезпечити такі умови праці, які б зменшували ризики негативного впливу зовнішніх чинників, а також передбачали раціональне розміщення технічних засобів, ергономічність робочого місця та наявність засобів індивідуального захисту [11].

Крім фізичних факторів, важливу роль відіграє й соціально-психологічний мікроклімат у колективі. Наявність підтримки з боку керівництва, чітке визначення обов'язків, справедлива оцінка результатів праці та можливість професійного зростання – усе це є критичними аспектами для формування позитивного ставлення працівника до праці та забезпечення його мотивації. Тому організація праці спрямована не лише на фізичний комфорт, а й на створення сприятливої соціально-психологічної атмосфери.

У контексті функціонування комп'ютеризованої системи управління мережею вендингових автоматів особливу увагу необхідно приділяти психоемоційному навантаженню на персонал, який обслуговує систему. Адміністратори, оператори або технічні працівники, які взаємодіють з інтерфейсами системи, мають змогу працювати в умовах, що забезпечують мінімальну кількість відволікаючих факторів і зручність виконання завдань. Інтерфейс програмного забезпечення є інтуїтивно зрозумілим, логічно структурованим, що дозволяє знизити рівень помилок та підвищити ефективність роботи. Організовано технічне обслуговування та ремонт автоматів з урахуванням безпечної експлуатації інструментів, доступу до електричних мереж та впливу мікроклімату на місцях встановлення автоматів. [10].

Важливим є запровадження заходів з охорони праці, спрямованих на запобігання професійним ризикам: інструктажі з техніки безпеки, навчання

діям у разі аварійних ситуацій, забезпечення доступу до аптечок та засобів першої допомоги, а також проведення періодичних медичних оглядів персоналу.

Успішне функціонування системи "людина - машина - середовище існування" можливе лише за умов оптимального балансу між характеристиками людини, технічними параметрами машини та умовами навколишнього середовища. Забезпечення цього балансу є головною умовою безпечної, ефективної та стійкої роботи в будь-якій виробничій або інформаційно-керуючій системі [12].

При проєктуванні комп'ютеризованої системи (КС) управління мережею вендингових автоматів, враховані вимоги по створенню безпечного, ергономічного та ефективного середовища для користувача. Система розроблена з урахуванням фізіологічних і психоемоційних особливостей людини, що дозволяє мінімізувати навантаження на оператора та знизити ймовірність виникнення помилок під час взаємодії з технічними засобами. Враховано обмежену здатність до тривалої концентрації уваги, що компенсується спрощеним та логічним інтерфейсом користувача, а також зменшенням кількості непотрібних дій у процесі керування.

Система побудована так, щоб забезпечити максимально комфортні умови праці як для операторів програмного забезпечення, так і для персоналу, що здійснює технічне обслуговування. Зокрема, приділено увагу раціональному розміщенню елементів керування, забезпеченню зручного доступу до обладнання, контролю мікроклімату в місцях роботи та збереженню допустимих рівнів шуму й вібрацій. Усі заходи організації робочого процесу в системі відповідають сучасним гігієнічним, ергономічним і безпековим нормам.

Особливе значення надано зниженню ризиків психоемоційного вигорання шляхом створення зрозумілої логіки роботи системи, зменшення інформаційного навантаження та врахування особистісних і професійних характеристик користувача.

Реалізована система відповідає принципам збалансованої взаємодії людини, технічного засобу та середовища, в якому вони функціонують, що забезпечує її безпечну, надійну та ефективну експлуатацію.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Організація робочого місця оператора персонального комп'ютера відповідає сучасним вимогам ергономіки з метою забезпечення безпечних, комфортних та ефективних умов праці. Врахування ергономічних норм зменшують втому, підвищують продуктивність і знижують ймовірність виникнення професійних захворювань, пов'язаних із тривалим перебуванням у статичній позі, напруженням зору або повторюваними рухами.

Ергономіка враховує взаємозв'язок між фізичними параметрами робочого місця та анатомо-фізіологічними особливостями людини. За наявності тривалого перебування в незручному положенні виникає ризик розвитку опорно-рухових порушень, таких як шийно-комірцевий остеохондроз, тунельний синдром чи сколіоз. Особливо вразливими є м'язи спини, шиї, зап'ясть і очі, тому запобігання перенапруженню є пріоритетом при розробці умов праці [13].

Основними елементами ергономічно організованого робочого місця є правильно підібрані меблі, технічне обладнання, освітлення та мікроклімат. Стіл врегульований по висоті, щоб забезпечити правильне положення тіла користувача: спина – пряма, ступні – на підлозі або підставці, передпліччя – вільно лежать на поверхні або підлокітниках. Сидіння має заокруглений передній край, м'яку оббивку та вентиляційні отвори для запобігання перегріванню [15].

Розташування периферійних пристроїв відповідає принципам ергономіки. Клавіатура розташована на відстані 10–15 см від краю стола, а миша — на одному рівні з клавіатурою. Для запобігання м'язовій втомі використовують килимки з гелевою підкладкою під зап'ястя.

Монітор розташований на відстані 50–70 см від очей користувача, а верхній край екрана – на рівні або трохи нижче рівня очей. Яскравість, контрастність і антивідблискове покриття екрана відповідає освітленню в приміщенні, щоб уникнути напруження очей і шиї. Згідно з ДСанПіН 3.3.2.007-98, рівень освітленості в робочій зоні повинен становити 300–500 лк, а штучне й природне світло рівномірно розподілені, без пульсацій або відблисків [14].

Оптимальні значення мікроклімату в офісі: температура повітря 22–25 °C, вологість — 40–60 %, швидкість руху повітря — ≤0,1 м/с. Забезпечення цих параметрів можливе через системи вентиляції та кондиціювання. Вирішено питання протягів, різких коливань температури та надмірної сухості повітря, які негативно впливають на самопочуття й концентрацію уваги.

Крім фізичного середовища, важливі організаційні аспекти: регулярні перерви (рекомендовано 45–60 хв), методика 20-20-20 (через кожні 20 хв – перерва 20 с з фокусом на об’єкті 6 м) сприяють зниженню втоми зору й мускулатури. Також доцільно проводити розминки або прості фізичні вправи для кистей, шиї та спини, щоб зменшити ризик застійних явищ у м’язах.

Ергономічна організація простору передбачає, що необхідні предмети розташовані в межах досяжності, а непотрібні — поза зоною робочого процесу. Робоче місце має бути впорядкованим, з чистими поверхнями та матовими покриттями, щоб уникнути концентрованого накопичення пилу [ ].

Враховано психоемоційні аспекти роботи з комп’ютером. Високий рівень інформаційного навантаження, часті перемикання уваги, потреба в прийнятті оперативних рішень можуть призводити до хронічної втоми, зниження мотивації та зростання кількості помилок. Створено не лише фізично комфортне, але й психологічно сприятливе середовище: з помірним рівнем шуму, відсутністю візуальних подразників, можливістю індивідуального налаштування робочого простору та підтримки належного рівня приватності.

Дотримання вимог ергономіки при організації робочого місця оператора ПК є важливою умовою збереження його здоров’я, підвищення працездатності та забезпечення безпечного функціонування всієї інформаційної системи. Це дозволяє досягти оптимального балансу між технічними засобами, фізіологічними можливостями людини та середовищем [13].

Ці чинники враховані при проєктуванні та організації робочого місця оператора у КРБ. Забезпечено дотримання сучасних ергономічних норм, спрямованих на підтримання фізичного і психоемоційного здоров’я користувача, а також на підвищення ефективності виконання професійних завдань. Робоче місце обладнано відповідно до анатомо-фізіологічних особливостей людини, що дозволяє уникнути надмірного навантаження на опорно-руховий апарат, зменшити вірогідність розвитку професійних захворювань та підтримувати зручну позу протягом тривалого часу.

Особлива увага приділена розміщенню технічних засобів: монітор, клавіатура, миша та інші периферійні пристрої розташовані відповідно до ергономічних стандартів, що дозволяє уникнути перенапруження м’язів і зору. Створені сприятливі мікрокліматичні умови – температура, вологість і швидкість руху повітря знаходяться в межах рекомендованих значень, що забезпечує комфортну атмосферу протягом усього робочого дня. Використано якісне освітлення з урахуванням рівня яскравості екрана і відсутності відблисків, що мінімізує втому очей.

Реалізоване ергономічне середовище сприяє стабільній і безпечній роботі оператора, підтримує його працездатність і знижує ризики, пов’язані з тривалою експлуатацією комп’ютерної техніки. Це забезпечує надійне функціонування системи загалом та сприяє досягненню її високої ефективності.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи була розроблена та імплементована програмно-апаратна система для централізованого управління мережею вендингових автоматів. Проведене дослідження та аналіз ринкових потреб підтвердили актуальність створення власного, гнучкого та економічно доцільного рішення, яке дозволяє подолати недоліки існуючих комерційних пропозицій.

Досягнутої метою було створення комплексної системи, що забезпечує збір статистичних даних про продажі, їх надійну обробку та збереження в централізованій базі даних PostgreSQL, а також ефективну візуалізацію та адміністрування через інтуїтивно зрозумілий веб-інтерфейс. Вибір мікроконтролера ESP32 для комунікації автоматів із сервером виявився оптимальним рішенням завдяки його вбудованому Wi-Fi модулю, високій обчислювальній потужності та енергоефективності, що є критично важливим для автономних пристроїв. Взаємодія між автоматами та сервером, а також між адміністративним інтерфейсом та сервером, реалізована за допомогою протоколу HTTP з використанням REST API та формату JSON, що гарантує уніфікованість, гнучкість та масштабованість системи. Застосування HTTPS для передачі даних забезпечує необхідний рівень безпеки та конфіденційності.

Розроблена система демонструє здатність у режимі реального часу відстежувати оперативну роботу кожного автомата, надавати деталізовані звіти про продажі та технічний стан, а також дозволяє оперативно реагувати на потенційні збої.

Отримані результати свідчать про можливість ефективного впровадження даного рішення у комерційних умовах, що може значно покращити операційні процеси, зменшити витрати на обслуговування та підвищити прибутковість вендингового бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 39 с.

2. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Комп'ютерні мережі. Книга 1. Львів: «Магнолія 2006», 2024. 256 с.

3. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Комп'ютерні мережі. Книга 2. Львів: «Магнолія 2006», 2024. 328 с.

4. Kharchenko A., Bodnarchuk I., Yatcysyn V. The Method for Comparative Evaluation of Software Architecture with Accounting of Trade-offs. American Journal of Information Systems. 2014. Vol. 2, No. 1. P. 20-25.

5. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyy R. Technology of relational database management systems performance evaluation during computer systems design. Scientific Journal of TNTU, Ternopil, Ukraine, 2023. Vol. 109, No 1. P. 54–65.

6. Yatsyshyn V., Pastukh O., Zharovskyi R., Shabliy N.Software tool for productivity metrics measure of relational database management system. Mathematical Modeling. No 1 (48). 2023. P. 7-17.

7. Ліщина В., Жаровський Р. Методи підвищення пропускної здатності в мережах LTE. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (7-8 грудня 2022 року). Тернопіль: ТНТУ. 2022. С. 86.

8. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. Вісник ТДТУ. Тернопіль, 2009. Т. 14. №1. С. 201-207.

9. ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора, Національний орган стандартизації України.URL: https://ukrstandart.ua/7299 (дата звернення: 05.06.2025).

10. ДСТУ 8604:2015 Робоче місце для виконання робіт у положенні сидячи, Національний орган стандартизації України. URL: https://ukrstandart.ua/8604 (дата звернення: 06.06.2025).

11. Держпраці. Вимоги виробничої санітарії до робочого місця, новина. URL: https://labor.gov.ua/sanitary-requirements (дата звернення: 04.06.2025).

12. Санітарно-гігієнічні вимоги до офісу, Naurok.com.ua. URL: https://naurok.com.ua/sanitary-office (дата звернення: 03.06.2025).

13. Санітарно-гігієнічні вимоги роботи на комп'ютері, Osvita.ukr-lit.com. URL: https://osvita.ukr-lit.com/sanitary-computer (дата звернення: 07.06.2025).

14. Ергономічні вимоги до організації робочих місць, Studopedia.org. https://studopedia.org/ergonomics-office (дата звернення: 07.06.2025).

15. Zolochiv.net. Умови праці працівників офісу, аналітика. URL: https://zolochiv.net/office-conditions (дата звернення: 07.06.2025).

16. FastAPI Documentation. FastAPI – The modern web framework for building APIs with Python 3.6+. URL: https://fastapi.tiangolo.com (дата звернення: 07.06.2025).

17. Arduino. ESP32 – Getting Started with ESP32 on Arduino IDE. URL: https://docs.arduino.cc/tutorials/esp32 (дата звернення: 07.06.2025).

18. PostgreSQL Documentation. The world's most advanced open source relational database. URL: https://www.postgresql.org/docs (дата звернення: 07.06.2025).

19. SQLModel Documentation. SQLModel – SQL Databases in Python, designed for FastAPI. URL: https://sqlmodel.tiangolo.com (дата звернення: 07.06.2025).

20. Bootstrap. The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com (дата звернення: 07.06.2025).

21. HTMX. High power tools for HTML. URL: https://htmx.org (дата звернення: 07.06.2025).

22. Espressif Systems. ESP32 Technical Reference Manual. URL: https://www.espressif.com/en/products/socs/esp32/resources (дата звернення: 07.06.2025).

23. GitHub. Example Vending Machine Monitoring System using FastAPI and ESP32. URL: https://github.com/vending_example/vending-monitoring (дата звернення: 07.06.2025).

24. Internet of Things (IoT). What is IoT and how does it work? URL: https://www.ibm.com/topics/iot (дата звернення: 07.06.2025).

25. W3Schools. HTML, CSS, JavaScript Tutorials. URL: https://www.w3schools.com (дата звернення: 07.06.2025).

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"Затверджую"

Завідувач кафедри КС

_____ Осухівська Г.М.

"___" _____2025 р

КОМП'ЮТЕРИЗОВАНА СИСТЕМА МОНІТОРИНГУ МЕРЕЖІ ВЕНДИНГОВИХ АВТОМАТІВ

ТЕХНІЧНЕ ЗАВДАННЯ

на _*9*_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-42
_____ к.т.н., доц. Яцишин В.В.	_____ Котюк О.А.
«____» _____ 2025 р.	«____» _____ 2025 р.

Тернопіль 2025

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютеризована система моніторингу мережі вендингових автоматів».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.221.00.00

1.2 Виконавець

Студент групи СІ-42, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Котюк О.А.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/7-53 від 27.01.2025 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 27.01.2025 р.

Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2025 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи – наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Комп’ютеризована система, що розробляється, призначена для:

– автоматизованого збору даних від вендингових автоматів;

– централізованого моніторингу та управління мережею пристроїв;

– забезпечення зручного інтерфейсу для адміністраторів системи;

– обміну даними між апаратною частиною (вендингом) і серверною частиною системи.

2.2 Мета створення системи

Метою кваліфікаційної роботи є розробка комп’ютеризованої системи управління мережею вендингових автоматів із використанням мікроконтролерів (ESP32), вебсервера на базі FastAPI та клієнтської адміністративної панелі для керування обладнанням у реальному часі.

2.3 Характеристика об'єкту

Розроблювана система призначена для використання в комерційних мережах вендингових автоматів, що потребують централізованого керування, оперативного реагування на збої та автоматичного збору статистичних даних для подальшого аналізу.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

Система повинна складатись з таких основних компонентів:

– серверної частини на основі FastAPI;

– модуля бази даних на основі SQLModel;

– вебінтерфейсу адміністратора, реалізованого з використанням HTMX та Bootstrap;

– мікроконтролерів ESP32, встановлених у вендингових автоматах;

– програмного забезпечення на ESP32 для зчитування подій і керування апаратними елементами.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Передача даних між апаратною та серверною частиною має здійснюватися через REST API, з використанням HTTP-запитів. Комунікація повинна бути узгодженою, безконфліктною, з мінімальною затримкою в обробці запитів.

3.1.3 Вимоги до режимів функціонування системи

Система повинна підтримувати такі режими:

– нормальний робочий режим – всі вузли функціонують згідно призначення;

– діагностичний режим – для перевірки справності елементів;

– аварійний режим – при виявленні відмов у з’єднанні чи в роботі обладнання.

Нормальний режим є основним. У разі виникнення аварійної ситуації система повинна повідомити адміністратора через вебінтерфейс.

3.1.4 Вимоги по діагностуванню системи

Серверна частина повинна мати вбудовані логери для фіксації запитів, подій, збоїв у роботі пристроїв. Мікроконтролери повинні передавати сигнали про статуси у фоновому режимі з використанням API. Усі події мають зберігатися в базі даних з прив’язкою до часу та ідентифікатора пристрою.

3.1.5 Перспективи розвитку, проектування системи

Система повинна мати модульну структуру з можливістю масштабування – підключення нових автоматів, створення додаткових ролей користувачів, розширення функціональності вебінтерфейсу (графіки, карти розміщення, аналітика тощо).

3.2 Показники призначення

Система повинна забезпечувати стабільну роботу при одночасному підключенні щонайменше 50 автоматів. Сервер повинен обробляти до 500 запитів на хвилину без втрати продуктивності.

3.2.1 Вимоги до надійності

Система має бути стійкою до збоїв. У разі короткочасної втрати зв’язку мікроконтролер повинен повторити запит. Сервер повинен працювати у середовищі з UPS, резервним копіюванням бази даних. Програмне забезпечення має бути протестоване на типові сценарії збоїв.

3.3 Вимоги до безпеки

Усі пристрої, що працюють під напругою, повинні мати захисні оболонки. Всі підключення мають бути ізольованими. Апаратні компоненти слід заземлити згідно з вимогами електробезпеки.

3.3.1 Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Усі апаратні компоненти мають експлуатуватись у температурному діапазоні від +5°C до +40°C при відносній вологості від 30% до 80%. Планове обслуговування мікроконтролерів та інтерфейсних з’єднань повинно проводитися не рідше одного разу на півроку. Рекомендовано регулярне оновлення прошивки на ESP32 та серверного програмного забезпечення.

3.4 Вимоги до захисту інформації від несанкціонованого доступу

Система повинна реалізовувати автентифікацію користувачів через логін/пароль. Доступ до API повинен бути захищений токенами. Усі критичні операції мають логуватись із фіксацією користувача, що їх виконує. Паролі мають зберігатись у хешованому вигляді.

3.4.1 Вимоги по збереженню інформації при аваріях

Резервне копіювання бази даних повинно здійснюватися автоматично щонайменше один раз на добу із збереженням копій на окремому захищеному носії або у хмарному сховищі, недоступному для змін з боку основної системи. Кожна резервна копія повинна зберігатися протягом періоду не менше ніж 7 діб, з можливістю відновлення даних з будь-якої точки відліку.

У разі аварійної ситуації (втрата електроживлення, пошкодження серверного обладнання, збої ПЗ, атаки на систему) інформація повинна бути відновлена з останнього збереженого знімка автоматично або за участю адміністратора з мінімальними втратами.

Процес резервного копіювання має бути задокументований і підтримувати логування, а у разі невдалого збереження — надсилати відповідне сповіщення адміністратору.

Серверна частина повинна підтримувати автоматичне створення дампів бази даних PostgreSQL з використанням вбудованих засобів (pg_dump), а також можливість гарячого резервування без зупинки роботи системи.

Крім бази даних, також повинні зберігатися конфігураційні файли, журнали подій та важливі системні параметри, що забезпечують цілісність і відновлення роботи системи після збою.

Рекомендується впровадження механізму перевірки цілісності резервних копій за допомогою контрольних сум або інших механізмів валідації, а також періодичне тестування процедури відновлення для гарантії її працездатності в екстрених ситуаціях.

3.4.2 Вимоги по стандартизації і уніфікації

Система повинна відповідати стандартам з обміну даними JSON та специфікаціям RESTful API, що забезпечують платформонезалежність і взаємодію між клієнтом та сервером.

Всі програмні інтерфейси мають бути реалізовані з урахуванням принципів ідентифікації ресурсів за допомогою URI, використання HTTP-методів відповідно до їх семантики (GET, POST, PUT, DELETE), а також чіткої обробки кодів відповіді HTTP.

У системі слід використовувати сучасні вебтехнології:

– FastAPI як серверний фреймворк, що підтримує OpenAPI (Swagger) для документування API;

– SQLModel як ORM для роботи з реляційною базою даних з уніфікованими схемами моделей;

– HTMX та Bootstrap — для уніфікації інтерфейсів взаємодії адміністратора з системою;

– PostgreSQL як надійне та масштабоване рішення для роботи з даними.

Програмні компоненти повинні супроводжуватися технічною документацією, що описує їх структуру, інтерфейси, вхідні/вихідні параметри, приклади використання та порядок оновлення.

Код програмного забезпечення повинен бути структурованим, модульним, відповідати принципам DRY (Don’t Repeat Yourself) і KISS (Keep It Simple, Stupid), а також підлягати уніфікації шляхом дотримання єдиних правил іменування, форматування та структурування.

Формати взаємодії між компонентами повинні відповідати відповідним відкритим галузевим стандартам, що дозволить інтеграцію з іншими інформаційними системами у майбутньому.

Також необхідно забезпечити уніфіковану систему логування подій з однаковим форматом журналів, а всі повідомлення про помилки повинні мати стандартну структуру для полегшення діагностики та автоматичного аналізу.

3.4.3 Вимоги до функцій (завдань), що виконуються системою:

– забезпечення наочного інтерфейсу для моніторингу стану автоматів;

– обробка сигналів з апаратної частини;

– фіксація подій у базі даних;

– передача команд на ESP32;

– підтримка перегляду логів і статистики.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ. Комплект повинен включати:

– пояснювальну записку;

– графічні матеріали:

а) структурну схему системи;

б) блок-схему обробки подій;

в) схему електричну монтажну;

г) ER-діаграма бази даних.

*Примітка: У комплект документації можуть вноситися зміни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка технічного завдання.	27.01.2025р. – 02.02.2025р.
2	Аналіз вимог та існуючих рішень щодо управління мережею вендингових автоматів	01.03.2025р. – 31.03.2025р.
3	Проєктування системи управління вендинговими автоматами	01.04.2025р. – 30.04.2025р
4	Розробка програмної реалізації системи управління вендинговими автоматами	01.05.2025р. – 31.05.2025р
5	Безпека життєдіяльності, основи охорони праці	12.06.2025р. – 13.06.2025р.
6	Оформлення пояснювальної записки і графічного матеріалу	01.06.2025р. – 13.06.2025р.
7	Перевірка на академічний плагіат, перевірка керівником та консультантами	9.06.2025р. – 15.06.2025р.
8	Попередній захист кваліфікаційної роботи бакалавра	16.06.2025р. – 22.06.2025р.
9	Захист кваліфікаційної роботи бакалавра	24.06.2025

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Лістинг коду

```
import uvicorn
from app.db.database import init_db
if __name__ == "__main__":
    init_db()
    print("Database created.")
    uvicorn.run("app.main:app", host="127.0.0.1", port=8000,
reload=True)

# app/main.py
from fastapi import FastAPI
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from app.api import automats
from app.api import sales
from app.api import stats
from app.views import dashboard
from app.views import automats as auto_view
from app.views import analytics
from app.views import settings
app = FastAPI(title="Vending Management System")
# Підключення шаблонів та статики
app.mount("/static",    StaticFiles(directory="app/static"),
name="static")
templates = Jinja2Templates(directory="app/templates")
app.include_router(automats.router)
app.include_router(sales.router)
app.include_router(stats.router)
app.include_router(dashboard.router)
app.include_router(auto_view.router)
app.include_router(analytics.router)
app.include_router(settings.router)

# app/api/automats.py
from fastapi import APIRouter, HTTPException, Depends
from sqlmodel import select
from app.models import Automat
from app.db.database import get_session
from sqlmodel import Session
from typing import List
router          =          APIRouter(prefix="/api/automats",
tags=["Automats"])
@router.get("/", response_model=List[Automat])
def get_automats(session: Session = Depends(get_session)):
    automats = session.exec(select(Automat)).all()
    return automats
@router.post("/", response_model=Automat, status_code=201)
def  create_automat(automat:  Automat,  session:  Session  =
Depends(get_session)):
    session.add(automat)
```

```
        session.commit()
        session.refresh(automat)
        return automat
    @router.get("/{automat_id}", response_model=Automat)
    def get_automat(automat_id: int, session: Session =
Depends(get_session)):
        automat = session.get(Automat, automat_id)
        if not automat:
            raise HTTPException(status_code=404, detail="Автомат
не знайдено")
        return automat

    # app/api/sales.py
    from fastapi import APIRouter, Depends, HTTPException, Query
    from sqlmodel import Session, select
    from typing import List, Optional
    from datetime import datetime
    from app.models import Sale, Automat, Product
    from app.db.database import get_session
    router = APIRouter(prefix="/api/sales", tags=["Sales"])
    @router.get("/", response_model=List[Sale])
    def get_sales(
        automat_id: Optional[int] = Query(default=None),
        session: Session = Depends(get_session),
    ):
        query = select(Sale)
        if automat_id:
            query = query.where(Sale.automat_id == automat_id)
        sales = session.exec(query).all()
        return sales
    @router.post("/", response_model=Sale, status_code=201)
    def register_sale(sale: Sale, session: Session =
Depends(get_session)):
        # Перевірка чи існує автомат
        automat = session.get(Automat, sale.automat_id)
        if not automat:
            raise HTTPException(status_code=404, detail="Автомат
не знайдено")
        # Перевірка чи існує товар
        product = session.get(Product, sale.product_id)
        if not product:
            raise HTTPException(status_code=404, detail="Товар
не знайдено")
        session.add(sale)
        # Оновлюємо час останнього звернення автомату
        automat.last_seen = datetime.utcnow()
        session.add(automat)
        session.commit()
        session.refresh(sale)
        return sale

    # app/api/stats.py
    from fastapi import APIRouter, Depends, HTTPException
```

```
    from sqlmodel import Session, select, func
    from app.db.database import get_session
    from app.models import Sale, Automat
    router = APIRouter(prefix="/api/stats", tags=["Statistics"])
    @router.get("/summary")
    def global_stats(session: Session = Depends(get_session)):
        total_income = session.exec(select(func.sum(Sale.price *
Sale.quantity))).one()
        total_sales                                              =
session.exec(select(func.sum(Sale.quantity))).one()

        return    {"total_income":    total_income    or    0.0,
"total_sales": total_sales or 0}
    @router.get("/automat/{automat_id}")
    def  automat_stats(automat_id:  int,  session:  Session  =
Depends(get_session)):
        automat = session.get(Automat, automat_id)
        if not automat:
            raise HTTPException(status_code=404, detail="Автомат
не знайдено")
        total_income = session.exec(
            select(func.sum(Sale.price * Sale.quantity)).where(
                Sale.automat_id == automat_id
            )
        ).one()
        total_sales = session.exec(

select(func.sum(Sale.quantity)).where(Sale.automat_id          ==
automat_id)
        ).one()
        return {
            "automat_id": automat_id,
            "total_income": total_income or 0.0,
            "total_sales": total_sales or 0,
        }

    # app/db/database.py
    from sqlmodel import SQLModel, create_engine, Session
    import os

    # Якщо .env ще не використовуєш — можна хардкодити шлях до БД
    DATABASE_URL           =           os.getenv("DATABASE_URL",
"sqlite:///./vending.db")
    engine = create_engine(DATABASE_URL, echo=True)
    def get_session():
        return Session(engine)
    def init_db():
        from   app.models   import   Automat,   Product,   Sale,
AutomatProduct
        SQLModel.metadata.create_all(engine)

    # app/models/automat_product.py
    from sqlmodel import SQLModel, Field, ForeignKey
```

```
    from typing import Optional
    class AutomatProduct(SQLModel, table=True):
        id: Optional[int] = Field(default=None, primary_key=True)
        automat_id: int = Field(foreign_key="automat.id")
        product_id: int = Field(foreign_key="product.id")
        stock: int

    # app/models/automat.py
    from sqlmodel import SQLModel, Field
    from typing import Optional
    from datetime import datetime
    class Automat(SQLModel, table=True):
        id: Optional[int] = Field(default=None, primary_key=True)
        name: str
        location: str
        date_creation:                datetime                =
Field(default_factory=datetime.utcnow)
        last_seen: Optional[datetime] = None

    # app/models/product.py
    from sqlmodel import SQLModel, Field
    from typing import Optional
    class Product(SQLModel, table=True):
        id: Optional[int] = Field(default=None, primary_key=True)
        product_name: str
        price: float
        stock: int

    # app/models/sale.py
    from sqlmodel import SQLModel, Field, ForeignKey
    from typing import Optional
    from datetime import datetime
    class Sale(SQLModel, table=True):
        id: Optional[int] = Field(default=None, primary_key=True)
        automat_id: int = Field(foreign_key="automat.id")
        product_id: int = Field(foreign_key="product.id")
        timestamp: datetime
        quantity: int
        price: float

    # app/views/analytics.py
    from fastapi import APIRouter, Request, Depends, Query
    from fastapi.templating import Jinja2Templates
    from sqlmodel import Session, select, func
    from datetime import datetime, timedelta
    from app.db.database import get_session
    from app.models import Sale
    router = APIRouter()
    templates = Jinja2Templates(directory="app/templates")
    @router.get("/analytics", include_in_schema=False)
    def analytics_page(request: Request):
        # default = week
        return templates.TemplateResponse(
```

```
            "analytics.html", {"request": request, "labels": [],
"data": []}
        )
    @router.get("/analytics/chart", include_in_schema=False)
    def analytics_chart(
        request: Request,
        period: str = Query("week"),
        session: Session = Depends(get_session),
    ):
        now = datetime.now()
        if period == "month":
            start = now - timedelta(days=30)
            step = "day"
        else:
            start = now - timedelta(days=7)
            step = "day"
        query = (
            select(
                func.date(Sale.timestamp).label("date"),
                func.sum(Sale.quantity).label("qty"),
            )
            .where(Sale.timestamp >= start)
            .group_by(func.date(Sale.timestamp))
            .order_by(func.date(Sale.timestamp))
        )
        results = session.exec(query).all()
        labels = [r.date.strftime("%Y-%m-%d") for r in results]
        data = [r.qty for r in results]
        return templates.TemplateResponse(
            "partials/chart.html",     {"request":     request,
"labels": labels, "data": data}
        )

    # app/views/automats.py
    from   fastapi   import   APIRouter,   Request,   Depends,
HTTPException
    from fastapi.templating import Jinja2Templates
    from sqlmodel import Session, select
    from app.db.database import get_session
    from app.models import Automat, Sale
    router = APIRouter()
    templates = Jinja2Templates(directory="app/templates")
    @router.get("/automats", include_in_schema=False)
    def  list_automats(request:  Request,  session:  Session  =
Depends(get_session)):
        automats = session.exec(select(Automat)).all()
        return templates.TemplateResponse(
            "automat_list.html",      {"request":      request,
"automats": automats}
        )
    @router.get("/automats/{automat_id}",
include_in_schema=False)
    def automat_detail(
```

```
        automat_id: int, request: Request, session: Session =
Depends(get_session)
    ):
        automat = session.get(Automat, automat_id)
        if not automat:
            raise HTTPException(status_code=404, detail="Автомат
не знайдено")
        sales = session.exec(
            select(Sale)
            .where(Sale.automat_id == automat_id)
            .order_by(Sale.timestamp.desc())
        ).all()
        return templates.TemplateResponse(
            "automat_detail.html",     {"request":     request,
"automat": automat, "sales": sales}
        )

    # app/views/dashboard.py
    from fastapi import APIRouter, Request, Depends
    from fastapi.templating import Jinja2Templates
    from sqlmodel import Session, select, func
    from app.db.database import get_session
    from app.models import Automat, Sale
    router = APIRouter()
    templates = Jinja2Templates(directory="app/templates")
    @router.get("/", include_in_schema=False)
    def   dashboard(request:   Request,   session:   Session   =
Depends(get_session)):
        automat_count                                          =
session.exec(select(func.count(Automat.id))).one()
        total_income = session.exec(select(func.sum(Sale.price *
Sale.quantity))).one() or 0
        total_sales                                            =
session.exec(select(func.sum(Sale.quantity))).one() or 0
        return templates.TemplateResponse(
            "dashboard.html",
            {
                "request": request,
                "automat_count": automat_count,
                "stats":     {"total_income":     total_income,
"total_sales": total_sales},
            },
        )

    # app/views/settings.py
    from fastapi import APIRouter, Request, Form, Depends
    from fastapi.responses import RedirectResponse, HTMLResponse
    from fastapi.exceptions import HTTPException
    from fastapi.templating import Jinja2Templates
    from sqlmodel import Session, select
    from app.models import Automat
    from app.db.database import get_session
    router = APIRouter()
```

```
templates = Jinja2Templates(directory="app/templates")
@router.get("/settings", include_in_schema=False)
def settings_page(request: Request, session: Session =
Depends(get_session)):
    automats = session.exec(select(Automat)).all()
    return templates.TemplateResponse(
        "settings.html", {"request": request, "automats":
automats}
    )
@router.get("/settings/new", include_in_schema=False)
def new_automat_form(request: Request):
    return templates.TemplateResponse(
        "form_create_edit.html",
        {
            "request": request,
            "title": "Додати автомат",
            "action": "/settings/new",
            "automat": None,
        },
    )
@router.post("/settings/new", include_in_schema=False)
def create_automat(
    request: Request,
    name: str = Form(...),
    location: str = Form(...),
    session: Session = Depends(get_session),
):
    automat = Automat(name=name, location=location)
    session.add(automat)
    session.commit()
    return RedirectResponse("/settings", status_code=303)
@router.get("/settings/edit/{automat_id}",
include_in_schema=False)
def edit_form(
    request: Request, automat_id: int, session: Session =
Depends(get_session)
):
    automat = session.get(Automat, automat_id)
    return templates.TemplateResponse(
        "form_create_edit.html",
        {
            "request": request,
            "title": "Редагувати автомат",
            "action": f"/settings/edit/{automat_id}",
            "automat": automat,
        },
    )
@router.post("/settings/{automat_id}/edit",
include_in_schema=False)
def update_automat(
    request: Request,
    automat_id: int,
    name: str = Form(...),
```

```
        location: str = Form(...),
        session: Session = Depends(get_session),
    ):
        automat = session.get(Automat, automat_id)
        automat.name = name
        automat.location = location
        session.add(automat)
        session.commit()
        return RedirectResponse("/settings", status_code=303)
    @router.delete("/settings/{automat_id}/delete")
    def delete_automat(automat_id: int, db: Session =
Depends(get_session)):
        automat = db.get(Automat, automat_id)
        if automat is None:
            raise HTTPException(status_code=404, detail="Автомат
не знайдено")
        db.delete(automat)
        db.commit()
        return HTMLResponse("")  # Порожня відповідь — елемент
буде видалено з DOM
```