КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 «Комп'ютерні науки» Освітньо-наукова програма «Управління проектами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА на тему:

“Дослідження процесів управління проєктом розробки інформаційної системи з моніторингу цін на споживчі товари”

Студента 2-го курсу групи УП-21	**Науковий керівник:**
Курицького Владислава Сергійовича	к.ф.-м.н., доцент
(прізвище, ім'я, по батькові)	*(науковий ступінь, вчене звання)*
	Стешенко Григорій Миколайович
	(прізвище, ім'я, по батькові)
(підпис студента)	*(дата)* *(підпис)*

Попередній захист:

(Висновок: "До захисту в Екзаменаційній комісії")

Завідувач кафедри технологій управління _____ _____ _____

(підпис) *(прізвище, ініціали)* *(дата)*

Київ - 2024

ЗАТВЕРДЖУЮ

Завідувач кафедри

професор Морозов В.В.

_____

“___” _____ 2024 року

З А В Д А Н Н Я

НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: Курицький Владислав Сергійович

Група УП-21

1. Тема кваліфікаційної роботи: “Дослідження процесів управління проєктом розробки інформаційної системи з моніторингу цін на споживчі товари”.

Затверджена протоколом від №6 від 06.11.2023 року.

2. Строк подання студентом готової роботи - “06” травня 2024 р.

3. Цільова установка та вихідні дані до роботи: дослідження характеристики об’єкту управління планування виконання проєкту (календарне планування, зміст, бюджет, ресурси, ризики, якість) та планування управління іншими галузями управління проєктами.

4. Зміст роботи: актуальність теми, огляд існуючих систем моніторингу цін, огляд публікацій з тематики, опис проблематики, визначення об’єму ринку, аналіз конкурентів, SWOT-аналіз, STEP-аналіз, системний аналіз, визначення контрольних віх, визначення вимог до продукту, дослідження математичних моделей об’єму та часу виконання запитів БД, опис персон користувачів, паспорт проєкту, побудова фінансової моделі, дослідження організаційної структури, складання змісту проєкту, опис технології управління, опис планування за часом, розробка плану управління ризиками, планування управління якістю, проектування моделей БД, опис CAP-теореми в контексті

проєкта, загальний опис архітектури ІС, опис основних модулів системи, опис браузерного застосунку, опис системи розгортання в хмарі.

5. Перелік графічного матеріалу (слайдів): титульна сторінка, мета дипломної роботи, актуальність теми, конкуренти на ринку, об’єм ринку, контрольні віхи, економічна модель, організаційна структура, життєвий цикл, технологія управління, зміст проєкту, календарний план, управління ризиками, управління якістю, CAP теорема, архітектура ПЗ, інфраструктура в хмарі, висновки.

6. Календарний план виконання роботи:

№ з/п	Назва частин роботи	План виконання роботи
1	Пошук та опрацювання інформаційних джерел з предмету дослідження	01.12.23 - 31.12.23
2	Складання плану кваліфікаційної роботи магістра	15.01.24 - 28.02.24
3	Підготовка першого розділу	29.02.24 - 16.02.24
4	Підготовка другого розділу	19.02.24 - 08.03.24
5	Підготовка третього розділу	11.03.24 - 29.03.24
6	Підготовка четвертого розділу	01.04.24 - 23.04.24
7	Оформлення кваліфікаційної роботи	24.04.24 - 03.05.24
8	Передача КРМ на кафедру в електронному вигляді	06.05.24
9	Передача роботи на рецензування	06.05.24
10	Попередній захист	07.05.24 - 10.05.24
11	Захист роботи	21.05.24 - 23.05.24

Дата видачі завдання: “06” листопада 2023 року.

Керівник роботи: к.ф.-м.н., доцент Стешенко Г.М.

_____

(підпис)

Завдання прийняв до виконання студент групи УП-21

Курицький В.С.

_____

(підпис)

ЗМІСТ

АНОТАЦІЯ

Дана кваліфікаційна робота магістра присвячена темі дослідження процесів управління проєктом розробки інформаційної системи з моніторингу цін на споживчі товари. За мету поставлено проєктування та розробка інформаційної системи автоматичного моніторингу цін як важливого інструменту для бізнесу, який допомагає забезпечити конкурентність, оптимізувати стратегії ціноутворення та приймати обґрунтовані бізнес-рішення.

Об’єктом дослідження є процеси створення інформаційної системи з автоматичного моніторингу цін. Предмет дослідження - процеси управління часом, якістю, ризиками, змістом, процеси побудови ефективної організаційної структури та процеси проєктування інформаційної системи. У роботі висвітлено актуальність та проблематику, проведений огляд популярних публікацій за темою. Досліджено ринок інформаційних систем з моніторингу цін, проведений аналіз конкурентів, визначено об’єм ринку, проведені SWOT, STEP аналізи. Побудована концепція проєкту, яка містить: системний аналіз, визначення віх, опис вимог, дослідження та моделювання сховища даних, опис персон користувачів, паспорт проєкту, фінансове моделювання та дослідження організаційної структури. Планування проєкту містить зміст робіт, опис технології управління, планування часу, ризиків та якості. У частині реалізації інформаційної системи надано моделі бази даних, обґрунтування вибору технологій, опис архітектурних складових, проєктування інфраструктури в хмарі, опис браузерного застосунку, приклади коду та демонстрація MVP.

За результатами досліджень отримано оптимальну організаційну структуру проєкту; змодельовано об’єм та швидкодію сховища даних; побудовано інформаційну систему здатну витримувати високі навантаження.

Досліджено та проаналізовано предметну галузь; змодельовано концепцію та план управління проєктом.

ВСТУП

У сучасному глобальному ринковому середовищі бізнеси стикаються зі наростаючою конкуренцією, швидкими змінами цін і вимогами споживачів щодо найкращих пропозицій. Для успішної конкуренції на ринку та забезпечення максимальної ефективності своєї діяльності, підприємства повинні бути в курсі актуальних цінових трендів і аналізувати дії своїх конкурентів.

В цьому контексті автоматичний моніторинг цін стає незамінним інструментом для бізнесу. Інформаційна система автоматичного моніторингу цін використовує передові технології обробки даних та аналітики для систематичного збору, аналізу та порівняння інформації про ціни на товари або послуги, що пропонуються на ринку.

Метою цієї магістерської роботи є проєктування та розробка інформаційної системи автоматичного моніторингу цін як важливого інструменту для бізнесу, який допомагає забезпечити конкурентність, оптимізувати стратегії ціноутворення та приймати обґрунтовані бізнес-рішення. Проєкт розглядає основні аспекти автоматичного моніторингу цін, включаючи методи та технології, переваги та виклики, що виникають.

Перед роботою поставлено такі *задачі*:

— Дослідити зовнішнє та внутрішнє середовище проєкту, впевнитись, що реалізація проєкту можлива.

— Побудувати загальну концепцію проєкту. Вона має продемонструвати життєздатність проєкту та повинна мати необхідні артефакти для подальшого планування та реалізації проєкту.

— Дослідити організаційну структуру при розробці розподілених інформаційних систем та запропонувати шляхи вирішення недоліків.
— Навести математичну модель оптимізації сховища даних.
— Запропонувати план управління проєктом за необхідними аспектами.
— Дизайн, опис та реалізація інформаційної системи з моніторингу цін, яка спроможна конкурувати на чинному ринку.

Об'єктом дослідження є процеси створення інформаційної системи з автоматичного моніторингу цін.

Предмет дослідження - процеси управління часом, якістю, ризиками, змістом, процеси побудови ефективної організаційної структури та процеси проєктування інформаційної системи.

У роботі було використано декілька *методів дослідження*, так у першій частині роботи було проаналізовано внутрішнє та зовнішнє середовище проєкту. Був проведений аналіз галузі, проведені SWOT та PEST аналізи. За допомогою метода зіставлення було розглянуто переваги та недоліки конкурентів. На наступному етапі було проведено системний аналіз інформаційного продукту, проведено моделювання економічної, часової та технічної складової проєкту. При дослідженні організаційної структури була висунута гіпотеза, яка потім перевірялася методом опитування. За допомогою проведених експериментів було виявлено оптимальний розмір сховища даних та за допомогою математичного моделювання вдалось спрогнозувати подальше масштабування.

Новизна отриманих результатів полягає у вдосконаленні організаційної структури при Domain Driven Design підході, що робить процес розробки та архітектуру інформаційної системи більш злагодженими і стабільними. Також отримали подальший розвиток методи оптимізації сховища NoSQL бази даних,

а саме - розроблені методи прогнозування масштабованості та швидкодії БД. У останньому розділі вдосконалено архітектуру інформаційної системи до наближення її до всіх трьох властивостей CAP-теореми.

Практичне значення результатів роботи полягає у тому, що за допомогою аналізу галузі інформаційних систем з моніторингу цін, можна результативно виводити новий продукт на ринок. Маючи концепцію та планування проєкту, можна успішно реалізувати проєкт з прогнозованими результатами. У свою чергу проведені дослідження над базою даних можна використати у багатьох IT-проєктах, де є потреба в оперуванні великими об’ємами даних. Також архітектурні патерни розроблені на основі CAP-теореми є достатньо універсальними та підійдуть до багатьох інформаційних систем. Результати дослідження організаційної структури проєкту є також придатними у використанні для багатьох проєктів де запроваджена DDD розробка і успішно показали себе на базі науково-дослідної практики.

У першому розділі буде розглянуто актуальність теми роботи та проблематика. Далі наведений огляд присутніх систем на ринку та більш детальний аналіз конкурентів. Також у розділі проаналізовано зовнішнє середовище проєкту, а саме, визначено об’єм ринку, проведені SWOT та STEP аналізи.

Наступний розділ передбачає побудову концепції проєкту. Тут наведені контрольні віхи проєкту, фінансова модель, системний аналіз, паспорт проєкту та основні вимоги до продукту. Також розділ містить дослідження математичної моделі для прогнозування місткості сховища для даної інформаційної системи. Завершується розділ описом досліджень організаційної структури.

Третій розділ присвячений плануванню проєктної діяльності. Так розроблений зміст пакетів робіт у вигляді ієрархічної структури, проаналізовано деякі технології управління проєктом та надано обґрунтування остаточного

вибору технології. Також розділ містить планування з управління ризиками, часом та якістю.

Останній розділ описує проєктування та реалізацію інформаційної системи. Він містить моделі бази даних, архітектурний опис системи, приклади основної частини коду, розгляд процесів всередині системи за допомогою UML-діаграм, детальний опис розгортання у хмарному середовищі та приклад роботи системи у стані MVP.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ЗОВНІШНЬОГО ОТОЧЕННЯ ПРОЄКТУ, ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ ПРОЄКТУ

1.1 Актуальність тематики

Останні роки світ сколихнула хвиля економічної нестабільності, так ще по сьогодні відбувається девальвація основних світових валют. У червні минулого року інфляція у Сполучених Штатах Америки сягнула 9.1% у річному вимірі. Для США це небувалий показник, який ФРС намагається приборкати вже майже рік. Зростання цін відчули майже всі країни світу, оскільки доллар США є резервною валютою багатьох центральних банків та займає основний відсоток в обігу грошової маси світу. Розглядаючи український ринок, можна помножувати фактори нестабільності, особливо через війну, демографічні проблеми та занепад експортного потенціалу. Тож в такий час, економічних коливань, буде дуже актуальним мати зручні та надійні інструменти з контролю цін. Це буде найбільш актуально для органів державного регулювання які займаються контролем цін, для комерційних мереж, для інвесторів та міжнародних економічних інститутів.

1.2 Огляд існуючих систем з моніторингу цін

На сьогоднішній день існує багато різних систем моніторингу цін, які допомагають підприємствам, магазинам та іншим суб'єктам господарювання відстежувати ціни на ринку. Серед найпопулярніших систем можна виділити такі:

1. Price2Spy: Це потужна інтернет-платформа з моніторингу цін, яка дозволяє відстежувати ціни на різних інтернет-майданчиках. Вона надає можливість автоматичного збору цінових даних, порівняння цін та аналізу конкурентів [10].

2. Prisync: Ця система моніторингу цін дозволяє відстежувати ціни на конкретні товари в режимі реального часу. Вона надає зручні інструменти для аналізу цінових змін і динаміки ринку, а також можливість отримувати сповіщення про зміни цін [11].

3. Jungle Scout: Це система моніторингу цін для продавців на Amazon. Вона надає розширені можливості для відстеження цін на Amazon-майданчику, аналізу конкурентів, прогнозування попиту та іншого ринкового дослідження [12].

4. SEMrush: Це інструмент для моніторингу цін, який дозволяє відстежувати ціни на товари і послуги в Інтернеті. Він надає можливість аналізувати цінові зміни, стежити за конкурентами та здійснювати ключовий аналіз ринку [13].

5. PriceRunner: Це порівняльний сервіс цін, який дозволяє знайти найкращі пропозиції на товари і послуги. Він надає широкий спектр категорій товарів, а також інструменти для порівняння цін і відгуків споживачів [14].

Ці системи моніторингу цін пропонують різні функції та можливості. При виборі тієї чи іншої системи варто враховувати конкретні потреби, бюджет і особливості бізнесу. Також важливо оцінити надійність та якість збирання та аналізу даних цінової інформації, а також якість підтримки користувачів.

1.3 Огляд популярних публікацій про ІС з моніторингу цін

1. "The method of automated monitoring of product prices and market position determination in relation to competition quotes: Monitoring of product prices and marketability development with continuous assessment of market position in on-line sales" (Radoslav Fasuga; Pavel Stoklasa; Martin Němec). У статті обговорюються виклики та техніки автоматизованого моніторингу цін у конкурентних ринкових умовах, включаючи отримання даних, обробку і оптимізацію стратегій ціноутворення [3].

2. "Research on distributed price monitoring system based on Multi-Agent" (Puyun Bi). У документі висуваються нові ідеї щодо створення розподіленої системи моніторингу цін на основі мультиагентної теорії, щоб зробити процес моніторингу цін інтелектуальним, діагностику несправностей цін у режимі реального часу та інтелектуальне вирішення конфліктів. [4].

3. "Price dialectics and the concept of creating a unified system for monitoring pricing processes in the economy" (Aleksandr Kovalev, Lyubov' Orlova, Pavel Domkin, Sergey Sokolov). У монографії наведено концептуальні підходи та практичні рекомендації щодо формування системи моніторингу процесів ціноутворення в економіці. Основною ідеєю створення такої системи є забезпечення прозорості процесів ціноутворення, виключення практики маніпулювання цінами та реалізація принципу чесного ведення бізнесу. [5].

4. "Blockchain Private Pools and Price Discovery" (Agostino Capponi, Ruizhe Jia). Робота розглядає процеси моніторингу цін з застосуванням технологій Blockchain [2].

5. "Pricing Process Part 5: Monitoring" (Frank Frohmann). Автор описує процеси автоматичного моніторингу та підводні камені, які виникають під час зіткнення з реальними кейсами. Наводяться поради щодо найбільш ефективного виконання моніторингу. [1].

1.4 Проблеми при автоматичному моніторингу цін

Хоча автоматичний моніторинг цін має багато переваг, беручи до уваги розглянуті вище роботи, він також може супроводжуватися деякими проблемами. Ось кілька потенційних проблем, які можуть виникнути при автоматичному моніторингу цін:

1. Блокування анти-скрапінгом: Деякі веб-сайти можуть застосовувати заходи проти скрапінгу (анти-скрапінг), що може призвести до блокування або

обмеження доступу до цінових даних. Це може ускладнити збір актуальної цінової інформації.

2. Зміна формату даних: Веб-сайти можуть змінювати формат та розташування даних, що ускладнює автоматичний збір і аналіз цін. Необхідно встановлювати механізми для адаптації до таких змін і оновлення програмного забезпечення для збору даних.

3. Неправильні дані: В деяких випадках можуть виникати проблеми з точністю зібраних цінових даних. Це може бути спричинено помилками в програмному забезпеченні, проблемами зі збором даних або зміною структури даних на веб-сайті. Потрібно відслідковувати та виправляти такі проблеми, щоб забезпечити точність отриманих даних.

4. Зручність використання: В деяких випадках інструменти моніторингу цін можуть бути складними у використанні або потребувати певного рівня технічних навичок. Це може вимагати додаткового навчання або залучення фахівців для ефективного використання таких систем.

5. Недостатня потужність системи: автоматичний моніторинг цін вимагає досить витратних обчислень, які можуть коливатись з часом. Саме тому така система повинна бути гнучкою та прогнозованою у плані масштабування.

Ці проблеми можуть виникати при автоматичному моніторингу цін, проте багато з них можуть бути вирішені за допомогою належного налаштування системи та регулярного оновлення програмного забезпечення для збору даних. Також варто зазначити, що маючи систему власної розробки, її легше адаптувати до потрібних функціонаьних вимог.

1.5 Визначення об’єму ринку

Для успішної реалізації проєкту важливо зібрати та проаналізувати якомога більше інформації стосовно цільового ринку. Якщо ринок дуже вузький або його потенційні користувачі не мають ресурсів на придбання продукту -

реалізація проєкту може бути під загрозою. Прогнозування об’єму ринку виступає на передній план маркетингово дослідження, коли ми виходимо на ринок з новим продуктом. В американській практиці прийнято використовувати такі показники для оцінки об’єму ринку: TAM, SAM та SOM.

— TAM (Total Addressable Market) - відноситься до загального ринкового попиту на продукт або послугу. Це максимальна сума доходу, яку може отримати бізнес, продаючи свій продукт або послугу на певному ринку.

— SAM (Serviceable Addressable Market) - це та частина TAM, на яку бізнес може реалістично орієнтуватися та обслуговувати, беручи до уваги такі обмеження, як географія, ціноутворення та розподіл.

— SOM (Serviceable Obtainable Market) - це та частина SAM, яку бізнес може реально охопити на основі таких факторів, як можливості компанії в маркетингу та продажах, конкуренція та насиченість ринку.

За інформацією Forbes, 25 найбільших рітейлерів України мають сукупний виторг в 530 млрд грн [15]. Зробимо припущення, що торгові мережі можуть собі дозволити витратити 0,01% від сукупного виторгу на систему моніторингу цін. Тоді TAM буде становити 5,3 млрд грн.

Якщо брати до уваги великих гравців ринку, а вони є переважною цільовою аудиторією продукту, то кількість таких компаній буде до сотні, вимірюючи за показником виторгу. Просуваючи продукт за допомогою B2B продажів, відділ продажу цілком може охопити таку кількість. Тоді SAM дорівнює TAM (5,3 млрд грн).

Беручи до уваги аналіз конкурентів, наведений вище. Можемо орієнтировочно порахувати показник SOM = 5,3 / 10 = 0,53 млрд грн.

Отже аналіз об’єму ринку показує, що реальний об’єм на який може розраховувати молода компанія складає 530 млн грн. Це показує, що гроші на ринку є та можна спробувати впровадити новий продукт.

1.6 Аналіз конкурентів

На таблиці 1.1 представлено порівняння трьох основних конкурентів:

Таблиця 1.1

Порівняння конкурентів

Критерій		**WEBCentric (Price2Spy)**	**Prisync**	**Competera**
1		**2**	**3**	**4**
Специфіка компанії	Кількість співробітників	116 [19]	73 [20]	79 [23]
	Заснування	Заснована у 2010 році у Сербії. Компанія вже мала мережу магазинів і потребувала систему автоматичного моніторингу, так як до цього доводилося це робити вручну [10].	Заснована як стартап командою ентузіастів у 2013 році в Стамбулі [11].	Заснована як маленький стартап орієнтований на внутрішній ринок двома партнерами [24].
	Дохід	5 млн. $ [19]	$17 млн. $ [20]	16.8 млн. $ [23]
	Інвестори	WEBCentric	ESOR Investments, Collective Spark [20]	STRATMINDS, Flyer One Ventures, Verras

Продовження табл. 1.1

1		2	3	4
				Capital, SMRK Fund, and Transform Partners Fund [25]
	Придбання	None	None	None
	Кількість клієнтів	Десятки користувачів	-	Десятки користувачів
Специфіка компанії	Сильні та слабкі сторони	Компанія гнучка та експериментує з новими направленнями. Співробітники скаржаться на овертайми та низьку заробітну плату [18].	Компанія більш орієнтована на маркетинг та продажі ніж на власний продукт, також співробітники скаржаться на неорганізований менеджмент. Але компанія гнучка у прийнятті рішень та має мотивовані висококваліфікова ні кадри [21].	Відзначаються сильні і слабкі сторони притаманні стартапам. Гнучкість, дружнє середовище, команда ентузіастів, але присутня велика турбулентність в управлінні персоналом та стратегічному менеджменті

Продовження табл. 1.1

1		2	3	4
				[24,26].
Персона клієнта	Продукт	Price2Spy та інше програмне забезпечення	Система з моніторингу цін	Система з моніторингу цін
	Основний покупець / Приймає рішення Цільовий клієнт	Великі компанії з світовим іменем, такі як: Phillips, Braun, Delonghi, Kenwood, EPSON. Так і середні за величиною інтернет-магазини [10].	Великі компанії з світовим іменем, такі як: Samsung, Sony, Suzuki. Так і невеликі інтернет-магазини [11].	Переважно мережеві рітейлери серед яких чимало українських компаній [27].
Специфіка продукту	Фічі продукту	— Historical Price reporting — Multi currency — Product matching — Bulk data import	— Alerts — Inventory tracking — API — Bulk data import — High price monitoring frequency [15]	— Historical Price reporting — Multi currency — Product matching — Bulk data import

Продовження табл. 1.1

1		2	3	4
Специфіка продукту		— User management module — Additional product details — inventory tracking — API — Monitoring Bot-aware websites — Alerts — Screenshot capturing — Google analytics 4 [15]		— Additional product details — inventory tracking — API — Monitoring Bot-aware websites — Alerts [15]
	Ціна	135 - 1350 $/місяць [16]	100 - 800 $/місяць [11]	Договірна ціна [27]
	Безкоштовний період	30 днів [16]	14 днів [11]	За домовленістю [27]

Продовження табл. 1.1

1		2	3	4
Специфіка продукту	Сильні сторони продукту	Може працювати на великих сайтах обходячи обмежуючі конфігурації. Гнучка у виявленні ціна та обробки результатів. Має гарну службу підтримки [10].	Зрозумілий інтерфейс, гарна клієнтська підтримка [11].	Багато функціоналу, швидкий саппорт, зрозумілий інтерфейс [28].
	Слабкі сторони продукту	Не завжди точний в оцінці продукту [17]	Обмежено працює з великою кількістю товарів. Недостатність фіч [22].	Користувачі скаржаться на високу ціну і що ціноутворення не завжди прозоре [28].

Продовження табл. 1.1

1		2	3	4
	Клієнтські відгуки	Зазвичай позитивні відгуки, але іноді скаржаться на те, що доводиться перевіряти дані вручну. [17]	Клієнти відмічають гарний інтерфейс, але іноді скаржаться на відсутність обробки деяких помилок [22] .	Не багато відгуків, а деякі викликають сумніви стосовно їхньої відвертості. Загалом відгуки позитивні, але деякі відмічають за високу ціну продукту [28].
Стратегія	Як перемогти/ чому користувачі повинні обрати нас	Стратегія заснована на чотирьох принципах: надійність, легкість у використанні, гнучкість, довговічність [10].	Компанія орієнтована на широкий спектр ринку, від маленьких крамниць до лідерів ринку електроніки [11].	Компанія дуже відома на внутрішньому ринку, але через його фінансову обмеженість вимушена виходити на міжнародні ринки. Компанія робить ставку на диверсифікацію фіч, а також на

Закінчення табл. 1.1

1		2	3	4
Стратегія	Як перемогти/ чому користувачі повинні обрати нас	Компанія намагається утримувати лідируючі позиції на ринку за рахунок досвіду, складним та надійним механізмом роботи та широкої технічної підтримки. Компанія орієнтована не тільки на великі торгові мережі, а і на невеликі крамниці, роблячи свій прайс доступним всім [10].	Компанія ставить першим пріоритетом клієнтське обслуговування та підтримку. Пропонуючи миттєві відповіді на виникаючі питання. Також компанія робить ставку на точність даних і гарантує їх свіжість та відповідність [11].	якісну підтримку. Так як компанія ще досі перебуває у вигляді стартапу, щоб вижити їм доводиться продавати свій продукт якомога дорожче за допомогою b2b продажів [24].

1.7 SWOT-аналіз

На етапі розробки концепції проєкту часто застосовується SWOT-аналіз для окреслення майбутньої стратегії розвитку проєкта. Цей метод дозволяє явно перелічити внутрішні (сильні та слабкі) та зовнішні (можливості та загрози) сторони проєкту. Після переліченя всіх сторін, складається матриця табл. 1.2, у центральних комірках описуються дії, які можливо зробити враховуючи перелічені аспекти.

Таблиця 1.2

Матриця SWOT-аналізу

	Сильні сторони: — Гнучкість у прийнятті рішень — Команда досвідчених професіоналів — Високопродуктивна інформаційна система — Низька бюрократизація — Команда не має жорсткої прив’язки до фізичного місця праці	**Слабкі сторони:** — Залежність від інвесторів — Молода компанія без репутації на ринку — Невеликий досвід у предметній області — Досить повільна швидкість розробки

Продовження табл. 1.2

Можливості: — Низька конкуренція на ринку — Висока волатильність макроекономічних показників, що збільшує зацікавленість у проєкті — Вихід на міжнародні ринки — Зацікавленість держави у проєкті	За рахунок гнучкості та низької бюрократизації, що пришвидшує темпи імплементації нових проривних рішень, є можливість випередити конкурентів та завоювати лідерські позиції на ринку. Через гнучкість компанії та відсутності прив'язки до місця роботи, можна легко переміщуватись кранами та створювати передумови для відкриття філіалів у зарубіжних країнах. Через високу продуктивність та професіоналізм співробітників, держава може обрати цю інформаційну систему поміж інших конкурентів.	Залежність від інвесторів може бути послаблена, якщо держава увійде у проєкт. Оскільки український ринок ще не насичений конкурентами подібних компаній, то малий досвід компанії може не стати проблемою.

Закінчення табл. 1.2

Загрози: — Поява нових конкурентів — Більш якісна та швидка розробка тих самих фіч конкурентами — Подорожчання послуг IT-спеціалістів — Збільшення податкового навантаження — Тиск з боку силовиків	Легкість у прийнятті рішень та професіоналізм команди мають посприяти у боротьбі з конкурентами. Через відсутність прив'язки до робочої локації, є можливість наймати працівників у інших, більш дешевих, країнах, якщо ціна місцевих працівників збільшиться. Тим самим можна подолати збільшення податків та тиск з боку силовиків, просто змінити резидентство компанії.	Молода компанія, поки що не маючи здобутої репутації, може вдатися до методів оптимізації податків. Так як компанія вже має невеликий досвід у предметній галузі та має працюючий продукт, це вже допоможе випередити тих конкурентів, котрі тільки починають заходити на ринок.

Згідно з представленою матрицею можна виділити наступні дії:

— За рахунок гнучкості компанії та низької бюрократизації намагатися знайти і розвивати нові направлення/фічі, щоб мати перевагу серед конкурентів.

— Через мобільність команди треба виходити на ринки іноземних країн.

— Залучати державу до інвестиції/користування продуктом.

— Зберігати сталий швидкий розвиток, щоб випереджати нових конкурентів на ринку.

— Розглянути можливість найму працівників з інших країн для зменшення собівартості продукту.

Всі перелічені дії мають практичну доцільність та мають стати основою розвитку проєкту на початковому етапі. Після досягнення повною мірою зрілості компанії/проєкту, має відбутись перегляд основних положень стратегічного напрямку.

1.8 STEP-аналіз

STEP-аналіз важливий для оцінки зовнішнього середовища проєкту. Це необхідно для розробки стратегії компанії та дає змогу врахувати певні ризики. Вплив на проєкт або компанію оцінюють за чотирма аспектами: політичним, економічним, соціальним і технологічним. Виділяють основні фактори які мають вплив на компанію та заносять у таблицю (табл. 1.3):

Таблиця 1.3

STEP-аналіз

Політика	**Економіка**
Збільшення податкового навантаження Прагнення влади до контролю галузі Захист інтелектуальної власності Бюрократизація і рівень корупції Стабільність уряду	Розмір заробітної плати Вартість та доступність кредитування Динаміка розвитку економіки Курс валют Рівень інфляції Валютні обмеження, обмеження пересування капіталу

Продовження табл. 1.3

Соціум	Технологія
Міграція населення Середній вік Баланс праці та відпочинку Розмір і структура сім'ї Ставлення до ІТ-галузі Темпи споживання	Доступність та швидкість інтернету Доступ до нових технологій Рівень інновації та технічного розвитку Державна технічна політика

Для кожного квадранту таблиці 1.3 було оцінено чотирма незалежними експертами, які знайомі з галуззю проєкту. Кожному фактору була присвоєна характеристика (“+” або “-”) та надана оцінка (max = 3) за силою впливу на проєкт. У загальному стовбці порахоно середньоарифметичне значеня. Результати наведені у Додатку А.

Найвпливовіші фактори винесено в таблицю 1.4.

Таблиця 1.4

Результати STEP-аналізу

Політичні		Економічні	
Фактор	Вага	Фактор	Вага
Захист інтелектуальної власності	+2,5	Вартість та доступність кредитування	+2,75
Збільшення податкового навантаження	-2,5	Розмір заробітної плати	-2,5

Продовження табл. 1.4

Соціальні		**Технологічні**	
Фактор	Вага	Фактор	Вага
Баланс праці та відпочинку	+1,75	Доступність та швидкість інтернету	+3
Міграція населення	-2	Рівень інновації та технічного розвитку	+2,75

Розглядаючи фактори політичного впливу, найбільший вплив на проєкт мають фактори: захист інтелектуальної власності та податкове навантаження. Щоб збільшити позитивний вплив першого фактора, варто приділяти більше уваги юридичній складовій, першим кроком для цього може стати найм юриста, який має досвід у захисті інтелектуальної власності. Щоб зменшити негативний вплив податкового навантаження, варто також найняти юриста та бухгалтера. Ще буде доцільно розглянути можливості з оптимізації податків і оформлення резидентства в іншій країні, якщо цей фактор буде чинити критичний вплив на компанію.

Наступними йдуть економічні фактори: вартість та доступність кредитування і розмір заробітної плати. Для кращого задіяння першого фактора, має сенс найняти банківського консультанта, який може підказати умови отримання більш дешевого та більш об’ємного кредиту. Ще варто розглянути кредитні програми в інших країнах. Розглядаючи другий фактор, а саме розмір заробітної плати, щоб знизити його негативний вплив, доцільно буде розглянути можливість найму спеціалістів з інших країн, наприклад, в Індії.

Найбільш впливові соціальні фактори, це: баланс праці та відпочинку і міграція населення. Для дотримання оптимального балансу роботи та життя,

має сенс проводити для співробітників навчальні сесії, з метою висвітлення питань організації життєвого часу. Варто також створювати організаційну культуру таким чином, щоб мінімізувати шкідливий вплив робочого перевантаження. Негативний вплив відтоку спеціалістів з країни можна також компенсувати наймом співробітників з інших країн. Наразі одним з трендів IT-галузі є віддалена робота, тому ще більше стираються кордони між кранами.

Доступність та швидкість інтернету має критичний вплив на проєкт. Так як відсутність останнього просто зупиняє хід проєкту. Наразі в Україні спостерігаються відключення електроенергії та можливі перебої зі зв'язком. Тому ще до виникнення проблем, варто забезпечити компанію резервними джерелами живлення та супутниковим інтернетом. Також фактор інновації і технічного розвитку має значний вплив на проєкт. Для його застосування з користю, буде доцільно слідкувати за трендами галузі, відвідувати відповідні конференції, слідкувати за державними програмами розвитку технологічного комплексу.

РОЗДІЛ 2. ПОБУДОВА КОНЦЕПЦІЇ ТА ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ ПРОЄКТУ

2.1 Системний аналіз та побудова математичної моделі

1) Система - Інформаційна система з моніторингу цін
2) Аналіз системи з фізичної точки зору:

   Надсистема: служба статистики, комерційні зацікавлені компанії.

   Підсистеми:

   — Співробітники.
   — Сервери.
   — Робоче обладнання.
   — Веб-сторінка.
3) Зовнішні зв’язки системи з надсистемамою:

Зовнішні фактори впливу:

— Служба статистики: запитує потрібну інформацію.

— Комерційні компанії: запитують потрібну інформацію.

— Постачальники даних: забезпечують коректне і безперебійне постачання.

— законодавство: регулює політику стосовно нерозголошення комерційної тайни.

4) Перелік елементів підсистем:

   1) Співробітники:
      a) Проєктний менеджер.
      b) Розробники.
      c) Девопс інженер.
      d) Product owner.

e) Тестувальники.

f) Дизайнер.

2) Робоче обладнання:

a) Комп’ютери.

b) Комунікаційне обладнання.

3) Сервери:

a) AWS Cloud.

4) Веб-сторінка:

a) Фронтенд частина.

b) Бекенд частина.

c) БД.

5) Процеси як елементи системи:

Запит аналітичних даних кінцевим користувачем:

— Зацікавлена сторона заходить на веб-сторінку

— Веб-сторінка формує запит на сервер та відправляє

— Серверна частина обробляє запит та виймає дані з БД

— Веб-сторінка приймає відповідь сервера

— Веб-сторінка обробляє отримані дані та формує графіки

Запит інформації про ціни:

— Сервер запитує продуктове API за певною категорією товарів

— Сервер обробляє та записує відповідну інформацію у БД

— Сервер повторює відповідний запит для кожної

6) Визначення надсистеми для процесу і зовнішній зв’язок з нею:

Після запиту аналітичних даних працівник служби статистики отримав потрібну інформацію стосовно зміни цінових показників.

7) Математичний опис IT системи:

Для опису використовується модель “конус”, яка представлена у формулі (2.1):

$$M = \{X, Y, H\}, \quad (2.1)$$

де $X = \{G, P, R, O, S, Z, K, L, A\}$ - набір вхідних параметрів моделі;

—P - сукупність процесів управління проектами і процесів створення продукту

—R - сукупність матеріальних ресурсів, задіяних у проекті

—O - сукупність учасників проекту (близьке оточення)

—S - послуги, підтримка і сервіси

—Z - сукупність технологій реалізації та підтримки проектного продукту

—K - сукупність параметрів проекту, умови, можливості та обмеження

—L - пропускна можливість компонентів

—A - доступність визначена набором оцінок виконання функцій та вимог (надійність, підтримка, безпека)

$Y = \{C, T, Q\}$ - сукупність вихідних параметрів (2.2)

- C - планова вартість створення елементів проекту
- T - наведена тривалість життєвого циклу проекту
- Q - якість проекту

2.2 Контрольні віхи проєкту

У таблиці 2.1 представлено основні віхи проєкту. Перша віха ознаменує початок розробки концепції проєкту (01.04.24), остання - завершальний етап (26.11.24).

Таблиця 2.1

Віхи проєкту

Дата	Віха
01.04.24	Початок розробки концепції проєкту
04.06.24	Завершено розробку вимог
07.06.24	Сплановано розробку проєкту
21.06.24	Завершено архітектурний дизайн
10.07.24	Завершено графічний дизайн
11.07.24	Початок розробки БД
11.07.24	Початок розробки серверної частини
11.07.24	Початок розробки клієнтської частини
16.07.24	Завершено розробку БД
23.08.24	Налагоджено інфраструктуру системи
02.09.24	Завершено розробку серверної частини
09.10.24	Завершено розробку Frontend
06.11.24	Завершено тестування компонентів системи
13.11.24	Завершено тестування і відладка системи у реальному середовищі
26.11.24	Виконані всі церемонії по завершенню проєкту

2.3 Визначення вимог до IT продукту та його попередня конфігурація

Технічні вимоги системи:

1. Мікросервісна архітектура розгорнута у контейнерах хмарного середовища з Load balancer.
2. Хмарний провайдер - AWS

3. БД - MongoDB розгорнута через Replica set у Mongo Atlas
4. Автоматизоване Unit, Api, E2E тестування
5. AWS Bucket для файлового сховища

Функціональні вимоги:

1. Користувач повинен мати можливість зареєструватись/залогінитись на веб-сторінці
2. Користувач повинен мати можливість обрати торгові мережі за якими йому цікава інформація
3. Користувач повинен мати можливість обрати певні категорії товарів
4. Користувач повинен мати можливість обрати часовий проміжок аналітичних даних
5. Користувач повинен мати можливість обрати графіки та діаграми за якими йому буде зручніше сприймати інформацію

Вимоги до продукту:

— Верстка веб-версії застосунку повинна коректно (без візуальних і технічних помилок) відображатися і бути кроссбраузерною в наступних браузерах: MoziIIa Firefox, Microsoft Edge. Google Chrome та Apple Safari

— Верстка веб-сторінки повинна коректно (без візуальних і технічних помилок) відображатися на мобільних пристроях в операційних системах iOS, Android.

— Користувацький інтерфейс додатку має бути інтуїтивно зрозумілим і виконаним *з* урахуванням правил UI/UX.

— Обробка простих операцій, таких як: вибірка , фільтрація, валідація - не повинна займати більше 0.5 сек. Побудова запиту та обробка не повинні займати більше 2 сек.

Організаційні вимоги:

— Розробка системи та створення супутньої документації виконуються на основі стандарту XYZCo-SP-STAN -95.

Зовнішні вимоги:

— Система не повинна розкривати конфіденційної інформації про користувача системи та конфіденційні аналітичні дані

2.4 Дослідження математичної моделі об’єму та часу виконання запитів у БД

Оскільки даний проєкт розрахований на обробку та зберігання великих масивів даних, то важливо розуміти, який об’єм жорсткого диску знадобиться через певний проміжок часу та як з плином часу, по мірі заповнюваності БД, буде знижуватися швидкість обробки запиту.

За формулою (2.3) розраховується кількість даних за місяць:

$$S_{30} = S_0 * C * N * T \quad (2.3)$$

де S_0 — розмір одного документа (kb)

C — кількість категорій продуктів

N — кількість торгових мереж для моніторингу

T — кількість днів

Знаючи показники даної системи можна розрахувати об’єм даних за місяць:

$S_0 \sim 165\ KB$

C = 10

N = 7

T = 30

S_{30} = 165 * 10 * 7 * 30 = 346 500 kB

S_{30} ~ 338 MB

Отже, на місяць знадобиться приблизно 338 мегабайтів.

Для розрахунку часу виконання запиту використовується формула (2.4):

$$P = f(S, C, D, I, R, Q) \quad (2.4)$$

де P — Query Execution Time

S — Collection Size

C — Query Complexity

D — Data Distribution

I — Indexing

R — Hardware Resources

Q — Query Execution Plan

Отже, час виконання запиту залежить від багатьох змінних і важкопередбачуванних параметрів, тому проведемо заміри виконання найбільш актуального запиту. Для цього сгенеруємо тестові дані за три місяці.

На рисунку 2.1 та рисунку 2.2 представлено скрипти генерації тестових даних для колекцій продуктів з оптимізаційним патерном та без, відповідно. Організація даних у колекції Products використовує патерн Bucket, про який буде йти мова у розділі 4. Навпроти, колекція Products_rare побудована звичайним чином без специфічної організації даних. Тому доцільно оцінити яку перевагу у швидкості виконанні запиту дасть оптимізація та спрогнозувати, де буде та межа заповнення колекції даними, коли швидкість виконання буде вже неприйнятна.

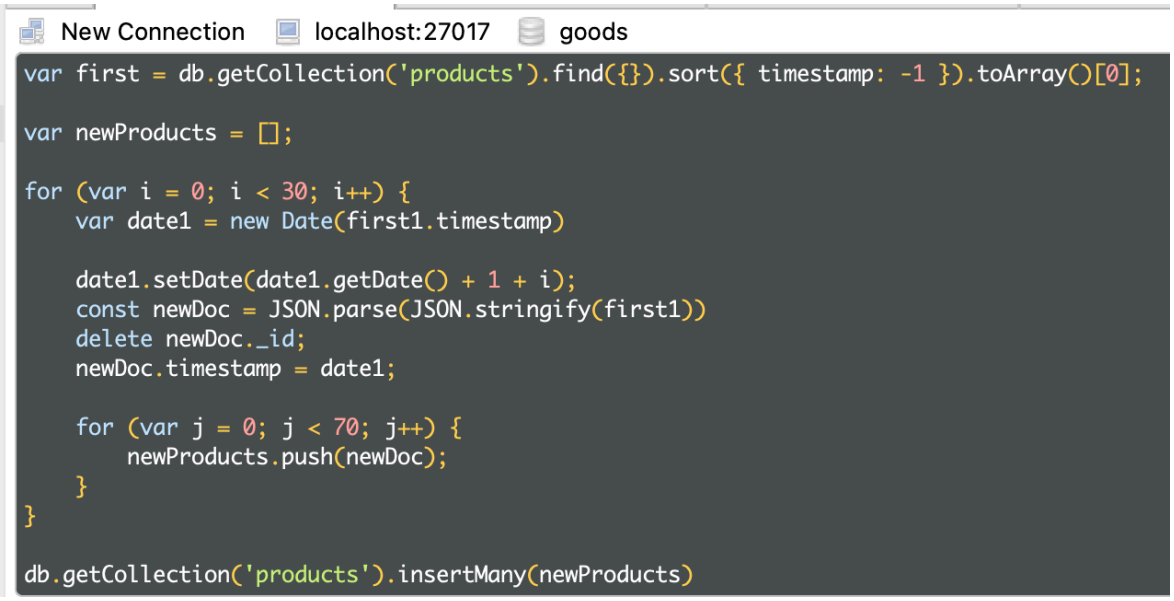


Рис. 2.1. Скрипт генерації тестових даних з оптимізацією

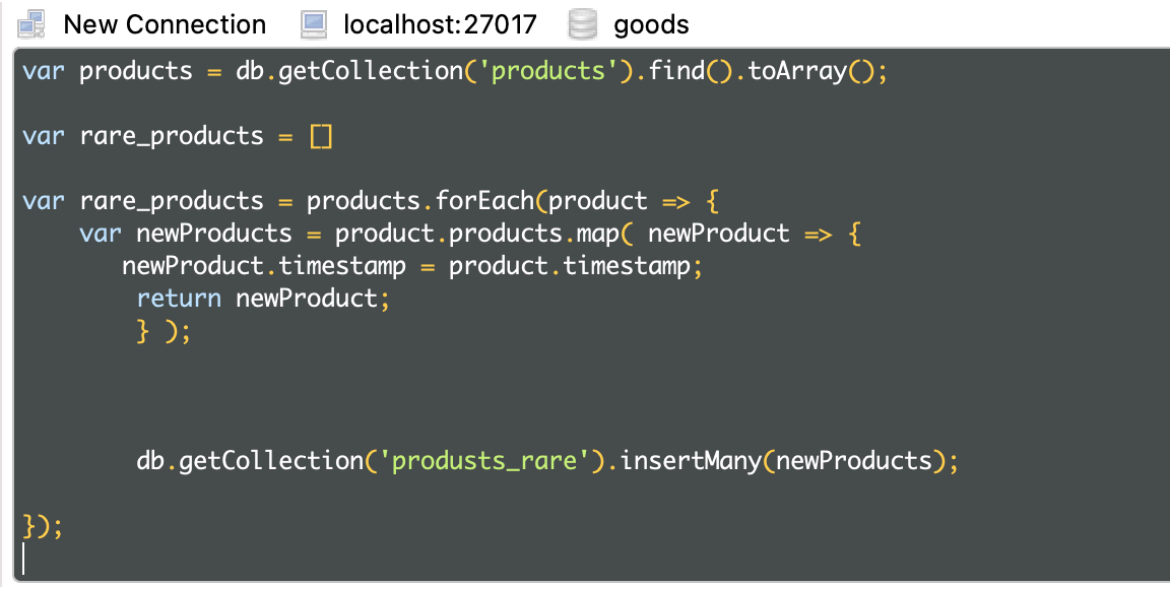


Рис. 2.2. Скрипт генерації тестових даних без оптимізації

Проведемо тестові заміри вибірки даних для колекції Products, за кожний місяць:

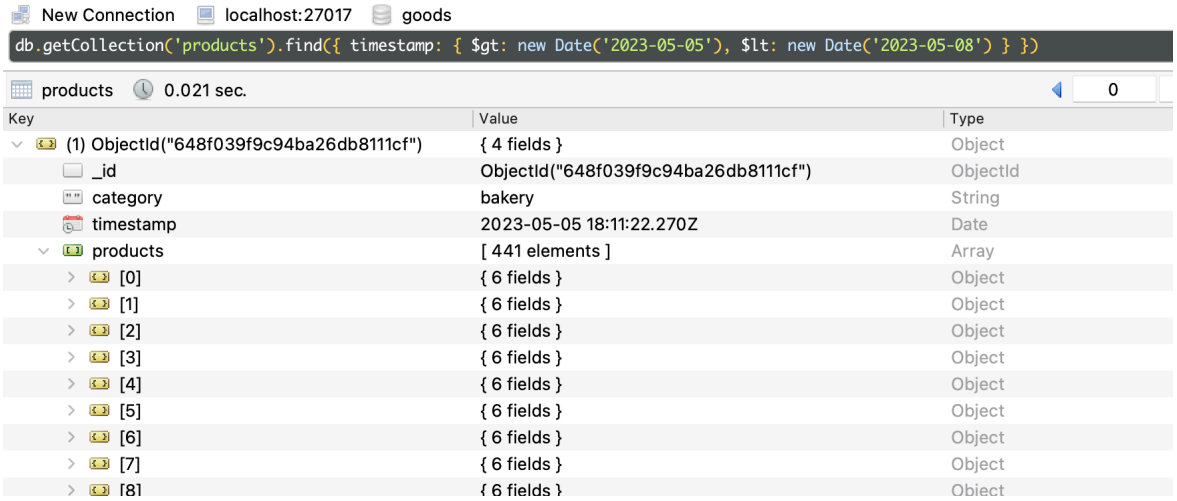


Рис. 2.3. Час виконання с заповненням колекції за перший місяць (0,021 сек.).

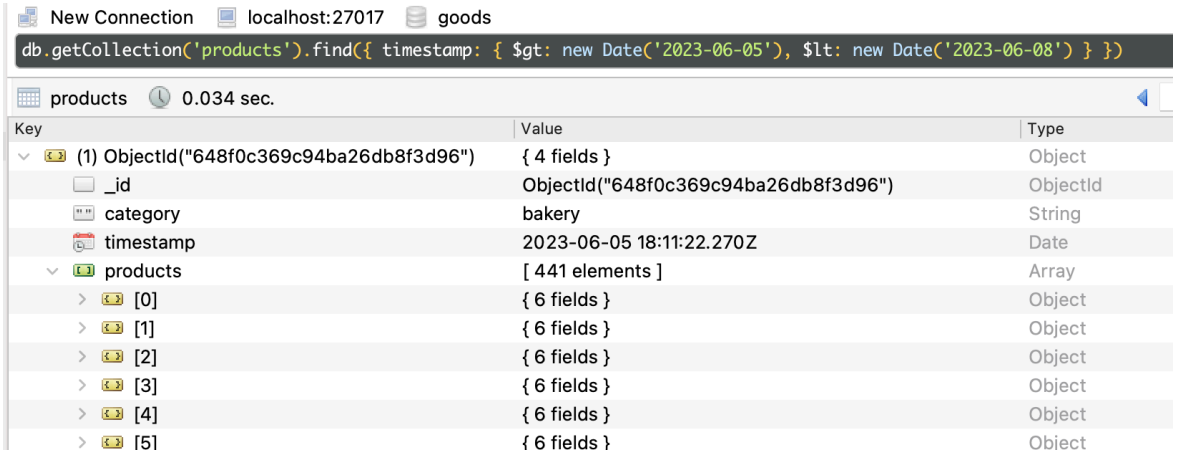


Рис. 2.4. Час виконання с заповненням колекції за другий місяць (0,034 сек).

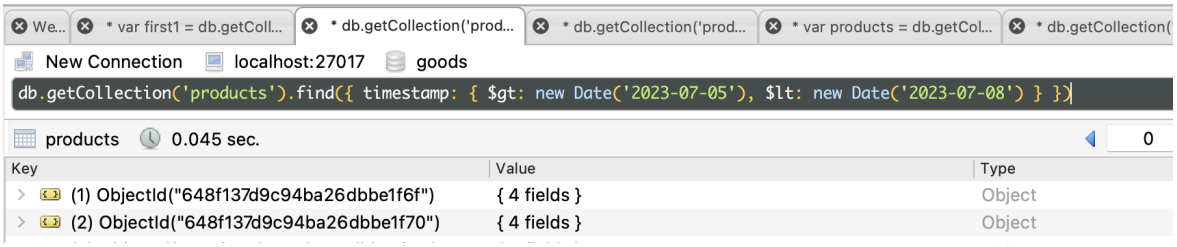


Рис. 2.5. Час виконання с заповненням колекції за третій місяць (0,045сек).

Як видно на рисунках вище, час виконання зростає по мірі заповнення колекції.

Проведено заміри колекції без оптимізації:

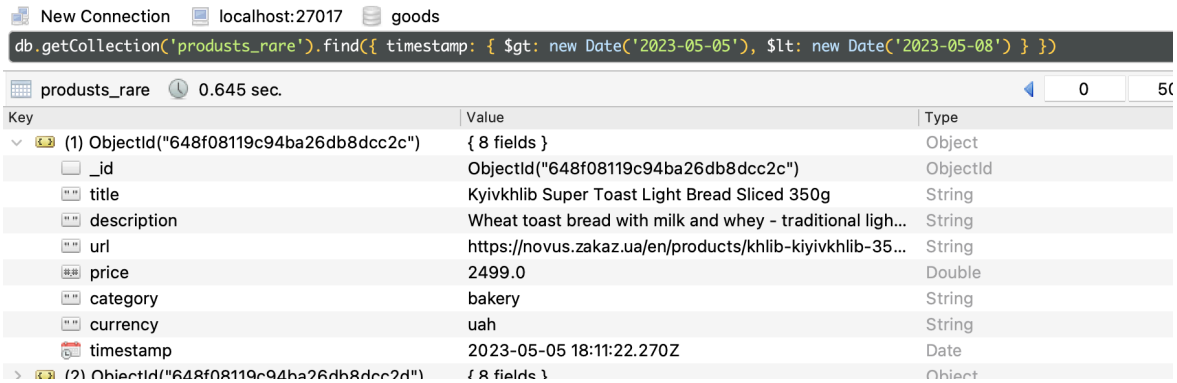


Рис. 2.6. Час виконання с заповненням колекції за перший місяць.

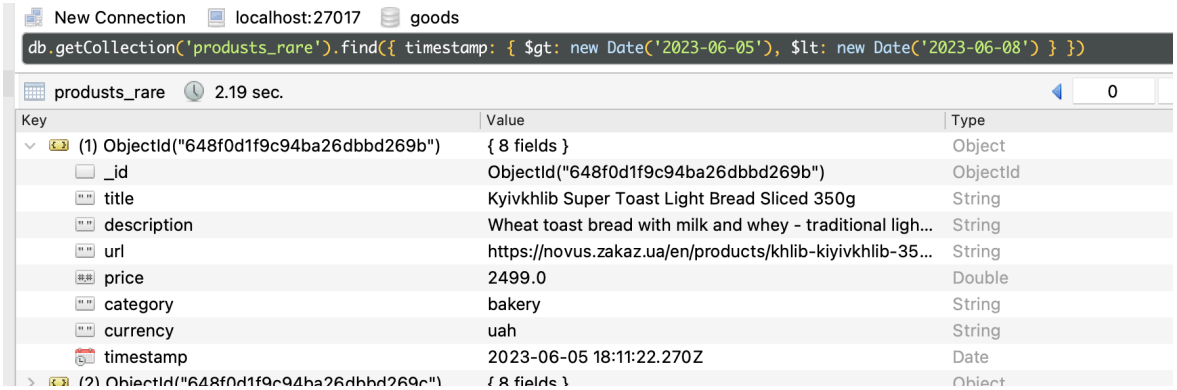


Рис. 2.7. Час виконання с заповненням колекції за другий місяць.

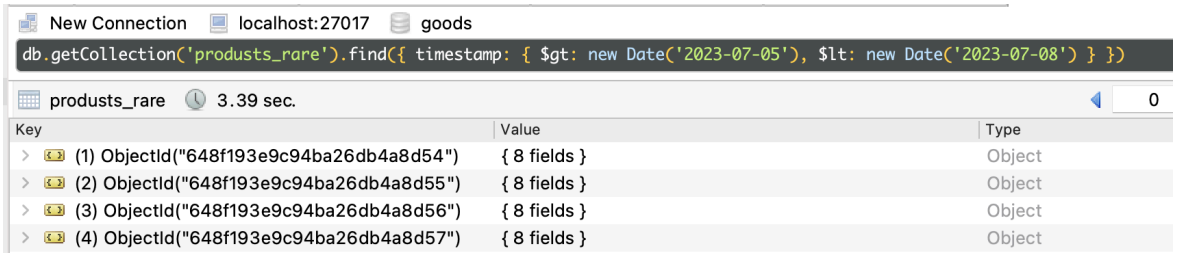


Рис. 2.8. Час виконання с заповненням колекції за третій місяць.

Як видно на замірах, час виконання запиту у коллекції без оптимізації зростає дуже суттєво.

Відкладемо результат на координатній площині та проведемо апроксимальну лінію. На вісі абсцис представлено час у місячному вимірі, а на вісі ординат час виконання у секундах:

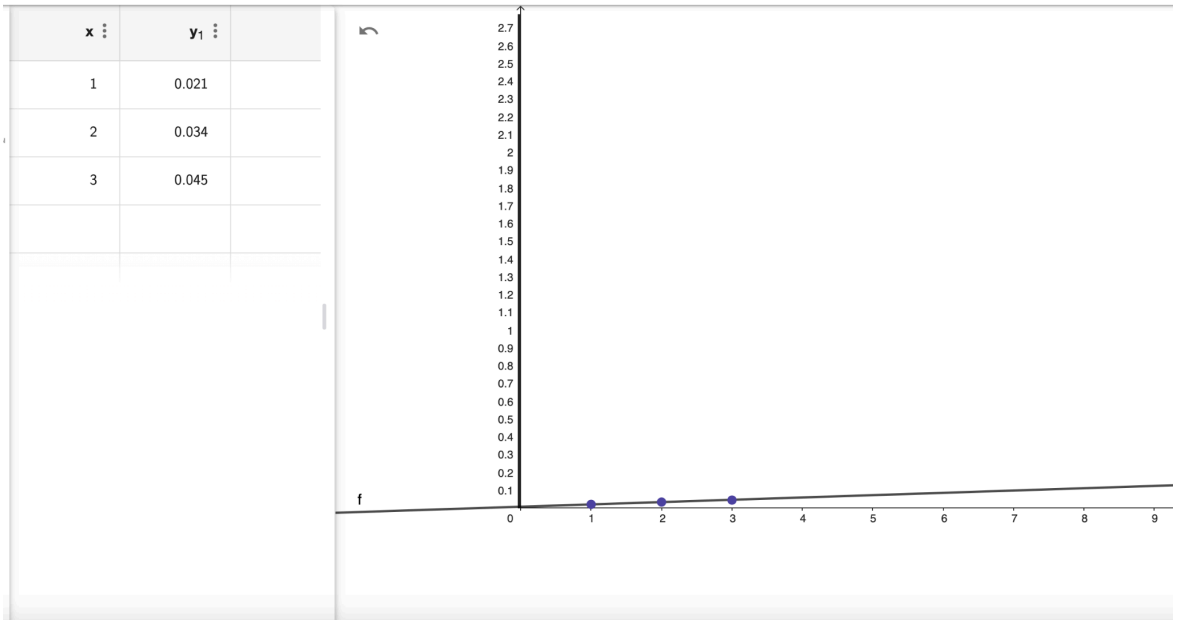


Рис. 2.9. Графік прогнозування часу виконання запиту в залежності від місячного заповнення колекції з оптимізацією.

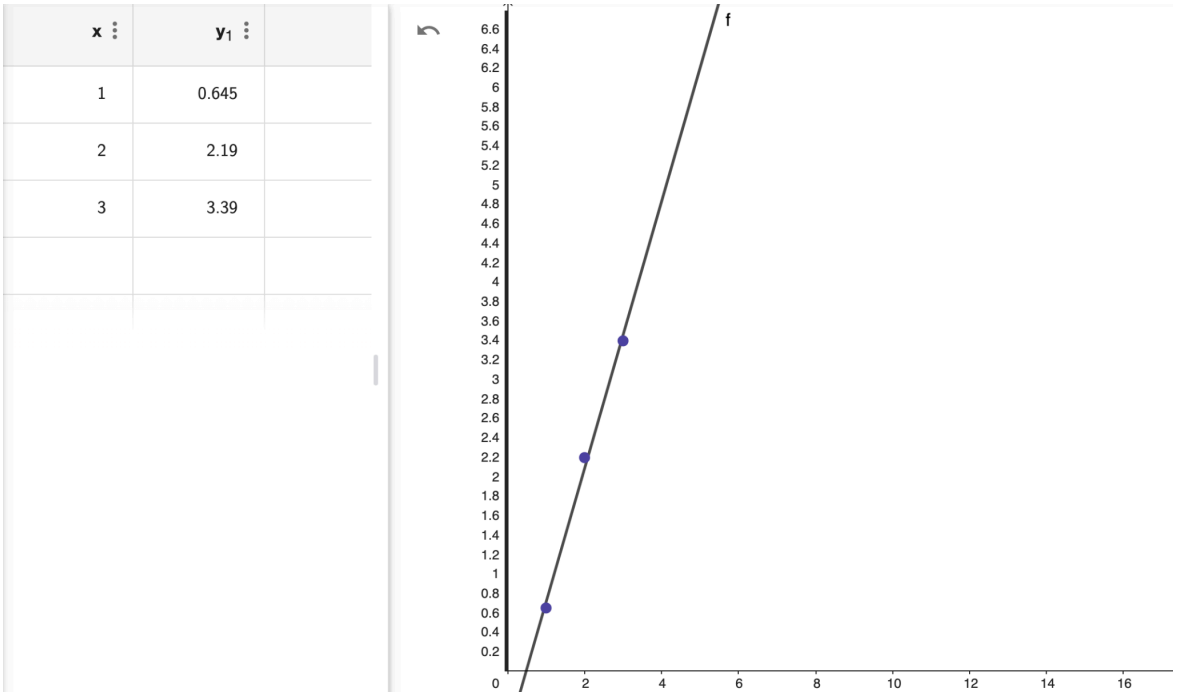


Рис. 2.10. Графік прогнозування часу виконання запиту в залежності від місячного заповнення колекції без оптимізації.

Тож на графіках видно, що колекція з оптимізацією може виконувати запити з прийнятною швидкістю ще довгий термін, а колекція без оптимізації виходить за неприйнятний поріг (2 секунди) вже в кінці другого місяця. Також продовжуючи графіки можно прогнозувати швидкість виконання у майбутньому.

2.5 Опис персон користувачів

Account holder:

Зображення персони користувача (рис. 2.11):

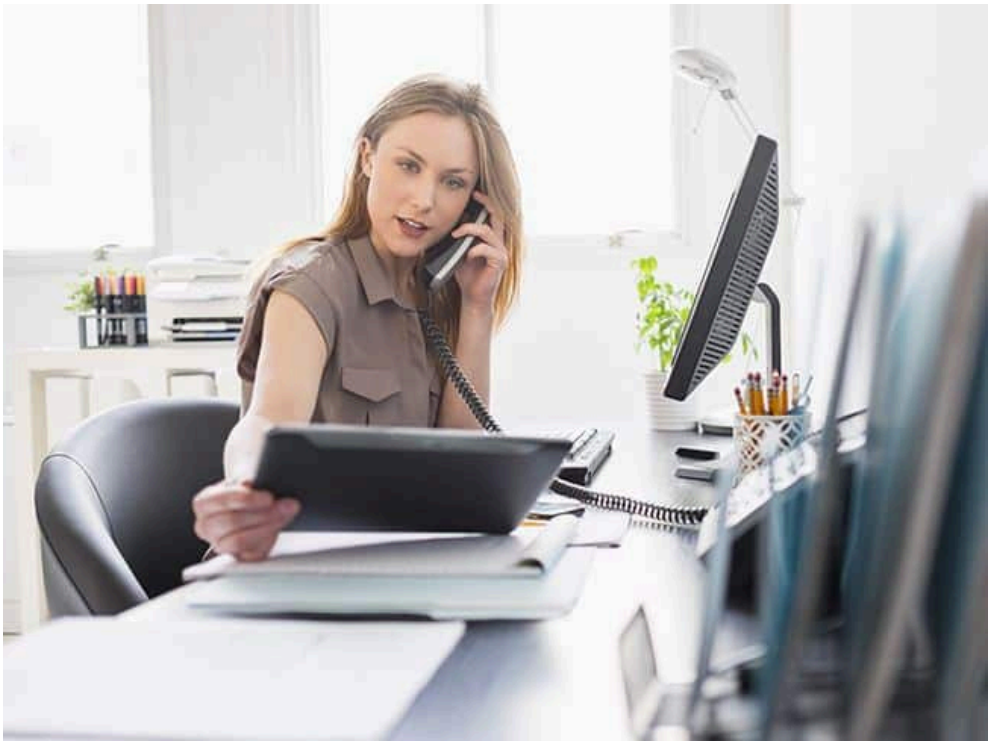


Рис. 2.11 Зображення персони користувача Account holder

Характеристика користувача:

Працює маркетологом у крупній торговельній мережі. Трохи більше 35 років. За плечима чималий досвід з маркетингу у топових компаніях країни та відповідна професійна освіта. Подобається подання інформації у цифрах та графіках. Відвідує професійні конференції та відслідковує тренди галузі.

Цілі та проблеми користувача:

Нестача комплексної інформації про конкурентів. Це не дає можливості більш точно визначити цінову політику. Користувач хоче у режимі “реального часу” відслідковувати зміни конкурентів та вчасно реагувати на них.

Administrator:

Зображення персони користувача (рис. 2.12):

Рис. 2.12 Зображення персони користувача Administrator

Характеристика користувача:

28 років. Має досвід роботи з складними інформаційними системами. Має досвід підтримки та настройки систем на різних платформах. Добре розуміє предметну область продукту.

Цілі та проблеми користувача:

Метою користувача є організація роботи системи з точки зору надання прав певним юзерам та в цілому адміністрування.

Tech support:

Зображення персони користувача (рис. 2.13):

Рис. 2.13 Зображення персони користувача Tech support

Характеристика користувача:

30 років. Має досвід роботи у технічному сапорті великих і малих інформаційних систем. Зоруміє проблеми навантаження інформаційних систем та методи їх контролю. Має досвід підтримки та настройки систем на різних платформах. Має технічну професійну освіту.

Цілі та проблеми користувача:

Метою користувача є організація безперебійної роботи системи та вирішення проблем захисту.

2.6 Макети дизайну інтерфейсу користувача

У цьому підрозділі представлено основні макети дизайну сторінок звичайного юзера у системі.

Користувач починає роботу з системою на сторінці авторизації, представленої на рис. 2.14. Форма авторизації має стандартний вигляд, з полями: логін та пароль користувача. Також є можливість скинути пароль.

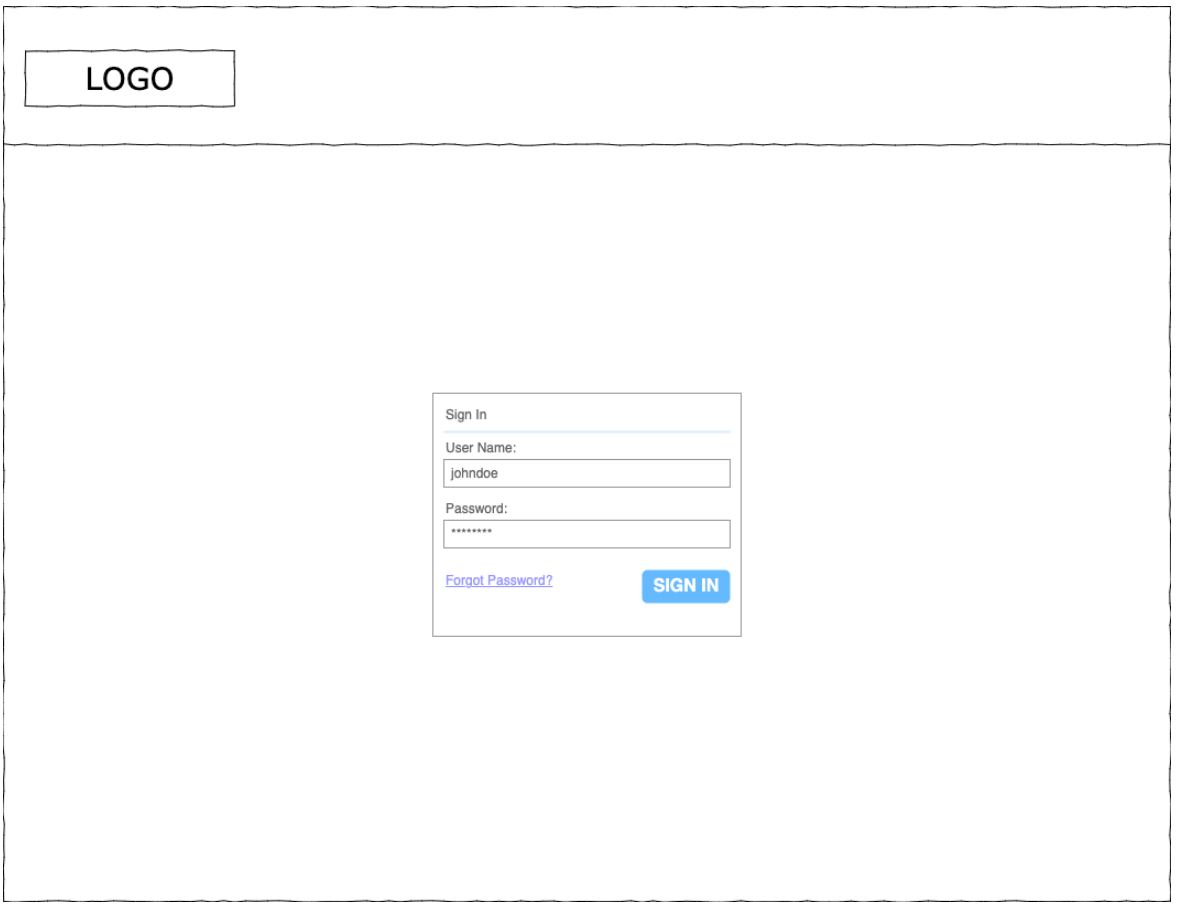


Рис. 2.14 Макет сторінки авторизації

Після авторизації, користувач попадає на сторінку дашборди. У центрі екрану містяться графіки, їх можна додавати, або видаляти завдяки меню зліва.

Вгорі міститься панель з пошуком та налаштуванням фільрів. Фільтри можна налаштовувати за категорію товару, торговельною мережею та часовим проміжком. Ще вище міститься логотип системи, кнопка друкування звіту та кнопка Log out.

Рисунок 2.15 представляє основну сторінку звичайного юзера, саме тут користувач отримує найбільше функціоналу та проводить майже весь час.

Рисунок 2.16 показує ту ж саму дашборду, але демонтрує функціонал фільтрації, це потрібно для зручності налаштування інформації на графіках.

Рис. 2.15. Макет головної сторінки юзера (дашборда)

Рис. 2.16. Макет фільтрів на сторінки дашборд

2.7 Фінансова модель проєкту

Фінансове моделювання проводилось у програмному середовищі Альт-інвест. Ця система дозволяє змоделювати фінансові надходження у проєкт, врахувати виробничі витрати, порахувати видатки та супутні платежі та отримати наглядні звіти у вигляді таблиць або графіків. Виконуючи фінансове моделювання, першим кроком потрібно задати базові параметри проєкту, у даному випадку було задано дату початку проєкту (1 квітня), тривалість - 8 місяців та умовний крок - місяць. Також обрано відобаражати показники у тисячах гривень. На наступному етапі потрібно задати макроекономічні

показники (рис. 2.17). Проставлено приблизний показник інфляції у річному розрахунку для гривні та долара, указано ставку НБУ та задано курсову пропорцію UAH/USD.

86	MACROECONOMICAL ENVIRONMENT			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.
87												
88	Calculation method	**Current prices**		2	*(inflation used in calculations)*							
89												
90	Supposed internal inflation rate											
91	of the main currency		%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%
95												
96	Supposed yearly growth of											
97	exchange rate of foreign currency		%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
98	Exchange rates for foreign currency (gr. 000 / $)			0,0250	0,0250	0,0250	0,0250	0,0250	0,0250	0,0250	0,0250	0,0250
99												
100	Supposed yearly external inflation rate											
101	of the foreign currency		%	4,0%	4,0%	4,0%	4,0%	4,0%	4,0%	4,0%	4,0%	4,0%
105												
106	The Central bank of Russia rate of re-financing		%	14,5%	14,5%	14,5%	14,5%	14,5%	14,5%	14,5%	14,5%	14,5%
107		0										
108	Tax-free rate of interest:											
109	main currency		%	16,0%	16,0%	16,0%	16,0%	16,0%	16,0%	16,0%	16,0%	16,0%
110	foreign currency		%	15,0%	15,0%	15,0%	15,0%	15,0%	15,0%	15,0%	15,0%	15,0%
111												

Рис. 2.17. Показники макроекономічного середовища

Далі було проведено прогнозування плану продажів. Як видно на рис. 2.18, кількість проданих підписок плавно наростає починаючи з 3-го місяця розробки та виходить на номінальний показник у шостому місяці. Орієнтована вартість підписки складає 225 тис. грн., далі ця ціна індексується з урахуванням інфляції.

13	SALES PLAN	Добавить / удалить ...		"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	*TOTAL*
14		Nominal volume											
15	Підписка	21,0	шт.		0%	0%	30%	50%	70%	100%	100%	100%	
16													
17	*Average production level*				*0%*	*0%*	*30%*	*50%*	*70%*	*100%*	*100%*	*100%*	
18													
19													
20	SALES VOLUME (in units)			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	*TOTAL*
21													
22	Підписка		шт.		0,0	0,0	6,3	10,5	14,7	21,0	21,0	21,0	***94,5***
23													
24													
25													
26	SALES PRICES (for unit, including VAT)			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	
27		Currency											
28	Підписка	1	gr. 000 / шт.	225,00	225,92	226,84	227,76	228,69	229,62	230,56	231,50	232,44	
33													
34													
35													
36	SALES REVENUES			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	*TOTAL*
37													
38	Підписка		gr. 000		0	0	1 435	2 401	3 375	4 842	4 861	4 881	***21 796***
51													
52	**= Total**		gr. 000		0	0	1 435	2 401	3 375	4 842	4 861	4 881	***21 796***
55													

Рис. 2.18. План продажів

На рисунку 2.19 показано витрати на персонал. Команда проєкту заначиться у розділі побічного виробничого персоналу, оскільки IT-працівники працюють за місячну ставку, а не за винагородження від об’єму виробництва (якщо помістити команду у розділ основного виробничого персоналу). Також указано адміністративний персонал, який складається з юриста та бухгалтера. У свою чергу комерційний відділ складається з двох менеджерів по продажам. Загальні витрати на оплату послуг персоналу складають приблизно 2,3 млн. грн. на місяць з урахуванням податків.

PERSONNEL AND SALARY	Добавить / удалить ...	"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
General production staff											
Direct Labor Costs	gr. 000		0	0	0	0	0	0	0	0	0
Qnt.	Mon. wages										
Наименование	0	0	0	0	0	0	0	0	0	0	*0*
Auxiliary production staff											
Qnt.	Mon. wages										
Fullstack developer	3	182	546	548	550	553	555	557	559	562	*4 431*
Backend developer	2	182	364	365	367	368	370	371	373	375	*2 954*
Designer	1	56	56	56	56	57	57	57	57	58	*454*
Integration developer	1	160	160	161	161	162	163	163	164	165	*1 298*
QA	2	56	112	112	113	113	114	114	115	115	*909*
Devops	2	133	266	267	268	269	270	271	273	274	*2 159*
Product owner	1	96	96	96	97	97	98	98	98	99	779
Project Manager	1	96	96	96	97	97	98	98	98	99	779
Managerial and administrative staff											
Qnt.	Mon. wages										
Юрист	1	30	30	30	30	30	30	31	31	31	*243*
Бухгалтер	1	25	25	25	25	25	25	26	26	26	*203*
Sales and marketing staff											
Qnt.	Mon. wages										
Sales manager	2	25	50	50	50	51	51	51	51	51	*406*
= Total	gr. 000		**1 801**	**1 808**	**1 816**	**1 823**	**1 831**	**1 838**	**1 845**	**1 853**	***14 615***
Accrued salary tax and insurance			468	470	472	474	476	478	480	482	*3 800*
Salary expenses including tax and insurance			**2 269**	**2 279**	**2 288**	**2 297**	**2 306**	**2 316**	**2 325**	**2 335**	***18 415***
Total employed	*persons*		*17*	*17*	*17*	*17*	*17*	*17*	*17*	*17*	

Рис. 2.19. Витрати на персонал

До прямих продуктових витрат належать закупка ноутбуків, у перший місяць проєкту, та оренда обчислювальних хмарних потужностей, які сплачуються на протязі всього проєкту. У загальні продуктові витрати внесено: оренда офісу, комунальні послуги, доставка води та канцелярія (рис. 2.20).

CURRENT EXPENSES	Добавить / удалить ...		"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
Direct production expenses												
Paid for Raw Materials and Components		gr. 000		0	0	0	0	0	0	0	0	*0*
General production staff salary		gr. 000		0	0	0	0	0	0	0	0	*0*
General production staff salary tax		gr. 000		0	0	0	0	0	0	0	0	*0*
	Currency											
Оренда хмарних потужностей	1	gr. 000		30	30	30	30	30	31	31	31	*243*
Закупка ноутбуків	1	gr. 000		390	0	0	0	0	0	0	0	*390*
General production expenses												
Auxiliary production staff salary		gr. 000		1 696	1 703	1 710	1 717	1 724	1 731	1 738	1 745	*13 763*
Auxiliary production staff salary tax		gr. 000		441	443	445	446	448	450	452	454	*3 578*
Depreciation		gr. 000		0	0	0	0	0	0	0	0	*0*
Land tax and other taxes		gr. 000		0	0	0	0	0	0	0	0	*0*
	Currency											
Оренда офісу	1	gr. 000		30	30	30	30	30	31	31	31	*243*
Телекомунікаційні послуги	1	gr. 000		1	1	1	1	1	1	1	1	*8*
Комунальні послуги	1	gr. 000		4	4	4	4	4	4	4	4	*32*
Доставка води	1	gr. 000		2	2	2	2	2	2	2	2	*16*
Канцелярія	1	gr. 000		1	1	1	1	1	1	1	1	*8*
Managerial and administrative expenses												
Managerial and administrative staff salary		gr. 000		55	55	55	56	56	56	56	57	*446*
Managerial and administrative staff salary tax		gr. 000		14	14	14	14	15	15	15	15	*116*
	Currency											
Наименование	1	gr. 000		0	0	0	0	0	0	0	0	*0*
Sales and marketing expenses												
Sales and marketing staff salary		gr. 000		50	50	50	51	51	51	51	51	*406*
Sales and marketing staff salary tax		gr. 000		13	13	13	13	13	13	13	13	*105*
	Currency											
Наименование	1	gr. 000		0	0	0	0	0	0	0	0	*0*
Sales and marketing expenses as % of sales	0%	gr. 000		0	0	0	0	0	0	0	0	*0*
= Total costs in income statement		gr. 000		2 651	2 335	2 345	2 354	2 364	2 374	2 383	2 393	*19 200*
= Total current payments		gr. 000		2 727	2 347	2 358	2 366	2 376	2 385	2 395	2 405	*19 357*

Рис. 2.20. Поточні витрати

Для старту проєкту було зроблено 2 млн. грн. прямих інвестицій (рис. 2.21) та взято кредит у сумі 2 млн. грн. Ставка кредиту складає 15% річних та за орієнтованими підрахунками (рис. 2.22) він може бути погашеним ще до завершення проєкту. Як видно на рис. 2.22, основний платіж (1,5 млн. грн.) припадає на сьомий місяць.

SHAREHOLDERS' CAPITAL			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
Ordinary shares:		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Preference shares	interest:	0%	0	0	0	0	0	0	0	0	0	*0*
Grants and similar investments		gr. 000	0	2 000	0	0	0	0	0	0	0	*2 000*
Info: Cash at the end of period		gr. 000	*2 000*	*1 980*	*-221*	*-1 253*	*-1 565*	*-1 073*	*614*	*770*	*2 451*	

Рис. 2.21. Інвестиції у проєкт

LOANS	Добавить / удалить ...		"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
Наименование												
Type of the loan	2	*Investment*										
Currency of the loan	1	gr. 000										
Interest rate (yearly)	15%			15%	15%	15%	15%	15%	15%	15%	15%	
Delay of payment of interests	0	mon.										
автоматический подбор кредита...												
Increase of principal		gr. 000	2 000	0	0	0	0	0	0	0	0	*2 000*
Disbursement of principal		gr. 000	0	0	0	100	100	100	200	1 500	0	*2 000*
Interest paid		gr. 000		25	25	25	24	23	21	19	0	*161*
Outstanding debt for the end of the current period		gr. 000	2 000	2 000	2 000	1 900	1 800	1 700	1 500	0	0	
= Total: Proceeds from loans		gr. 000	**2 000**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***2 000***
= Total: Repayments of loans		gr. 000	**0**	**0**	**0**	**100**	**100**	**100**	**200**	**1 500**	**0**	***2 000***
= Total: Interest paid		gr. 000	0	25	25	25	24	23	21	19	0	***161***
= Total: Outstanding debt		gr. 000	**2 000**	**2 000**	**2 000**	**1 900**	**1 800**	**1 700**	**1 500**	**0**	**0**	
Info: Cash at the end of period		gr. 000	*2 000*	*1 980*	*-221*	*-1 253*	*-1 565*	*-1 073*	*614*	*770*	*2 451*	

Рис. 2.22. Кредитування проєкту

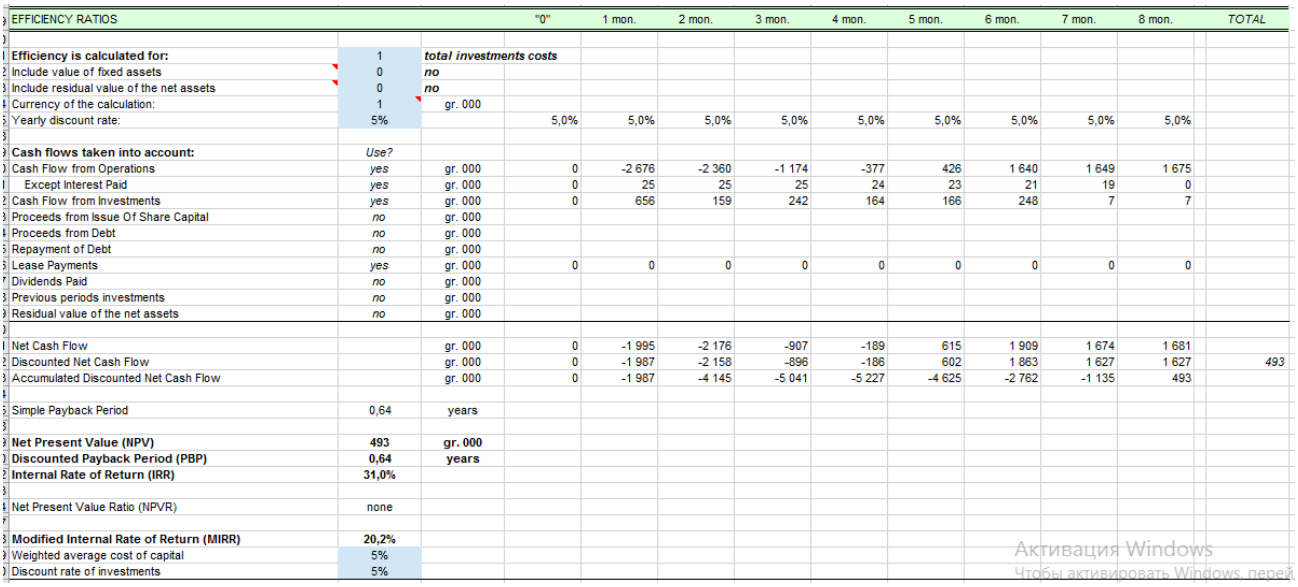
На рисунку 2.23 представлено податкові відрахування. Ставка ПДВ встановлена на рівні 20%, а податок на прибуток сягає 15%. На зведеній таблиці видно суми планових податкових витрат, як видно, найбільші витрати проходять за статтями: ПДВ та соціальні внески з заробітних плат співробітників. Загальна сума податків становить приблизно 7,3 млн. грн. за 8 місяців роботи проєкту. З цього можна зробити висновок, що проєкт приносить значну користь державному бюджету.

TAXES AND SIMILAR PAYMENTS			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
	days											
Excise duty and export tax	30			0	0	0	0	0	0	0	0	*0*
Import tax	30		0	0	0	0	0	0	0	0	0	*0*
1. VALUE ADDED TAX												
tax rate	20,0%											
payment period	30	*days*										
overpaid VAT indemnification method	1	*paid from budget*										
write off VAT on fixed assets	1	*immediately after start of operations*										
VAT received		gr. 000	0	0	0	239	400	563	807	810	814	*3 633*
VAT paid		gr. 000	0	76	11	11	11	12	12	12	12	*157*
VAT paid into budget (or indemnified from budget)		gr. 000	**0**	**-76**	**-11**	**228**	**389**	**551**	**795**	**799**	**802**	***3 476***
2. TAXES INCLUDED IN COST OF GOODS SOLD												
Social payments		gr. 000		**468**	**470**	**472**	**474**	**476**	**478**	**480**	**482**	***3 800***
Land tax		gr. 000		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***0***
Other taxes included in COGS		gr. 000		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***0***
3. TAXES INCLUDED IN GENERAL EXPENSES												
Property tax		gr. 000		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***0***
Other taxes included in general expenses		gr. 000		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***0***
4. PROFIT TAX												
tax rate	15,0%			15%	15%	15%	15%	15%	15%	15%	15%	
payment period	90	*days*										
Profit tax accrued		gr. 000		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***0***
Total tax payments			0	392	459	700	863	1 027	1 273	1 278	1 284	*7 276*

Рис. 2.23. Податки та інші відрахування

Ітогова таблиця ефективності проєкту приведена на рисунку 2.24 та табл. 2.2. Проєкт окупається через 0,64 року, а NPV становить 493 тис. грн. Також показник рентабельності IRR сягає 31%.

Загальний кошторис проєкту без урахування інфляції та налогових платежів представлено в таблиці 2.3. Тобто для реалізації проєкту знадобиться приблизно 15 млн. грн.



EFFICIENCY RATIOS			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
Efficiency is calculated for:	1	***total investments costs***										
Include value of fixed assets	0	***no***										
Include residual value of the net assets	0	***no***										
Currency of the calculation:	1	gr. 000										
Yearly discount rate:	5%		5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	5,0%	
Cash flows taken into account:	*Use?*											
Cash Flow from Operations	*yes*	gr. 000	0	-2 676	-2 360	-1 174	-377	426	1 640	1 649	1 675	
Except Interest Paid	*yes*	gr. 000	0	25	25	25	24	23	21	19	0	
Cash Flow from Investments	*yes*	gr. 000	0	656	159	242	164	166	248	7	7	
Proceeds from Issue Of Share Capital	*no*	gr. 000										
Proceeds from Debt	*no*	gr. 000										
Repayment of Debt	*no*	gr. 000										
Lease Payments	*yes*	gr. 000	0	0	0	0	0	0	0	0	0	
Dividends Paid	*no*	gr. 000										
Previous periods investments	*no*	gr. 000										
Residual value of the net assets	*no*	gr. 000										
Net Cash Flow		gr. 000	0	-1 995	-2 176	-907	-189	615	1 909	1 674	1 681	
Discounted Net Cash Flow		gr. 000	0	-1 987	-2 158	-896	-186	602	1 863	1 627	1 627	*493*
Accumulated Discounted Net Cash Flow		gr. 000	0	-1 987	-4 145	-5 041	-5 227	-4 625	-2 762	-1 135	493	
Simple Payback Period	0,64	years										
Net Present Value (NPV)	**493**	**gr. 000**										
Discounted Payback Period (PBP)	**0,64**	**years**										
Internal Rate of Return (IRR)	**31,0%**											
Net Present Value Ratio (NPVR)	none											
Modified Internal Rate of Return (MIRR)	**20,2%**											
Weighted average cost of capital	5%											
Discount rate of investments	5%											

Рис. 2.24 Ефективність проєкту

Таблиця 2.2

Показники ефективності проєкту

Показник	Значення
Простий строк окупності	0,64
Дисконтований строк окупності	0,64
NPV	493 тис. грн.
IRR	31%

Таблиця 2.3

Вартісна оцінка проєкту

Назва витрат	**Ціна за одиницю (грн)**	**Кількість**	**Тривалість (кількість місяців)**
Проєктний менеджер	96 000	1	8
Розробник	182 000	5	8
Тестувальник	56 000	2	8
Девопс інженер	133 000	2	8
Дизайнер	56 000	1	8
Інтеграційний розробник	160 000	1	8
Юрист	30 000	1	8
Бухгалтер	25 000	1	8
Product Owner	96 000	1	8
Sales manager	25 000	2	8
Загальна вартість:	14 408 000		
Обладнання:			
Хмарне середовище	30 000	1	8
Комп’ютерна техніка	30 000	13	
Загальна вартість:	630 000		
Інші витрати:			
Оренда офісу	30 000		8
Телекомунікаційні послуги	1 000		8
Комунальні послуги	4 000		8
Доставка води	2 000		8

Закінчення табл. 2.3

Канцелярія	1 000		8
Загальна вартість:	304 000		
ВСЬОГО:	15 342 000		

2.8 Паспорт проєкту

На таблиці 2.4 представлено паспорт проєкту:

Таблиця 2.4

Паспорт проєкту

Назва проекту:	Створення інформаційної системи для відстежування цінових змін на продовольчі товари
Замовник проекта	Прямі інвестори
Розробник проекта	Команда технічних спеціалістів
Ціль проекту	Налагодження безперервного постачання коректної інформації щодо цінових змін
Задачі проекту	— Автоматизувати збір цінових показників — Створити БД з історією цінових змін — Зробити зібрані дані репрезентативними і легкими для сприйняття — Підвищити точність та актуальність знань про цінові зміни

Продовження табл. 2.4

Учасники проекту	— Служба статистики - отримання точної та актуальної інформації про ціни. — Команда розробки - отримання грошової винагороди і нових навичок роботи з подібними системами. — Постачальник даних - прибуток та покращення репутації — Мережі супермаркетів - розширення інформаційного впливу щодо позицій товарів та цін — Комерційні компанії, які зацікавлені в наявній інформації - отримання більш прозорої інформації про ціни конкурентів — Державні органи з регулювання цін - отримання актуальної інформації про ціни
Об’єм та джерело фінансування	2 млн. грн. кредиту 2 млн. грн. прямі інвестиції
Час виконання	8 місяці

2.9 Дослідження організаційної структури проєкту

Гіпотеза

Інформаційна система з моніторингу цін складається щонайменше з двох відокремлених модулів. Кожен з яких оперує у своїй окремій предметній галузі, хоча деяким чином вони все ж таки пов’язані. Модульність системи реалізується за рахунок підходів мікросервісної архітектури [30]. Маючи значну розділеність за технічною складовою та за доменами, є сенс розглянути організаційну структуру за принципами Domain Driven Design.

В умовах високого ступеня розподіленості підсистем збільшується непередбачуванність системи в цілому [31]. Так, не погоджена зміна в одному модулі, або баг може проявитися в некоректній роботі інших модулей. З локальної точки зору модуля все може бути цілком логічно і працювати правильно, але при встраюванні в совокупну систему можуть проявлятися суперечності. Це є наслідком слабкої зв’язаності сервісів і комунікації між командами. Враховуючи ще те, що зазвичай розробка ведеться згідно з Agile принципами. Де надається перевага робочому коду замість документації та легкість до змін замість дотримання плану[33]. Це робить систему ще більш непередбачуваною та провокує несвідомі зміни. Можна заперечити, що Agile також спонукає до збільшення комунікації, але ж як зазначалося вище, ця комунікація відбувається переважно у замкненому просторі своєї предметної області, не виходячи за її межі. Відчувається що системі не вистачає централізованого погляду на сукупність децентралізованих, слабо зв'язаних процесів. Шукаючи вирішення даної проблеми, цілком законно можна звернутися до архітектурних патернів. Так як за законом Конвея - архітектура і організаційна система дуже пов’язані між собою [29]. Цей принцип, вперше сформульований мережевим програмістом і організаційним теоретиком Мелвіном Конвеєм в 1968 році. Цей принцип стверджує, що структура організації прямо впливає на структуру систем, які ця організація розробляє.

Закон Конвея визнає, що комунікаційні шляхи та взаємодії між членами команд та підрозділів впливають на архітектурні аспекти розроблюваної системи.

Формулювання закону Конвея може бути висловлене наступним чином: "Організація, що проектує систему, здійснює проектування, яке відображає структуру цієї організації." Це означає, що організації, розглядаючи структуру команд і комунікаційні зв'язки всередині них, можуть передбачати, як буде виглядати структура їхніх програмних продуктів.

Закон Конвея найчастіше асоціюється з галуззю програмного забезпечення та розробкою систем, адже саме в цих областях структура команд та їхні взаємозв'язки можуть визначати архітектуру та дизайн програмних рішень. Організації, що мають розділений структурний склад, схильні розробляти розділені, забезпечені інтерфейсами продукти.

Одним із практичних застосувань закону Конвея є визначення стратегії організаційної архітектури з огляду на майбутні потреби розроблюваних систем. Розуміння, як комунікаційні потоки впливають на архітектурні рішення, може сприяти більш ефективному управлінню процесами розробки та структуризації організації в цілому. Тож за даним законом ми можна розглядати організаційну структуру з точки зору архітектури та навпаки.

Візьмемо до уваги Event Driven Architecture, яка переважно поділяється на два підходи: Broker і Middleware[31]. Broker дуже децентралізована і складна в реалізації та в підтриманні цілісності система (рис. 2.25).

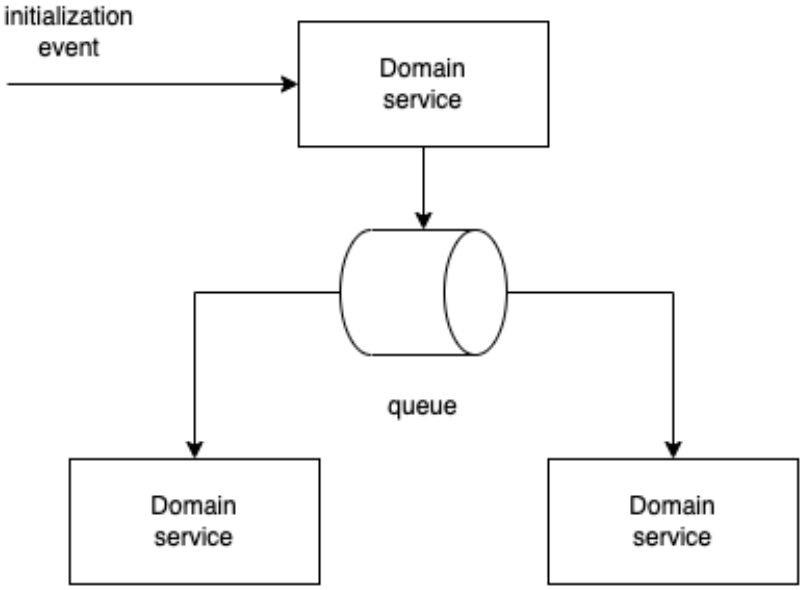


Рис. 2.25. EDA Broker pattern

Навпроти, Middleware додає посередника який оркеструє інформаційні потоки між сервісами (рис. 2.26). Така система більш передбачувана, легко піддається контролю, хоч і додає помірної зв’язаності системі.

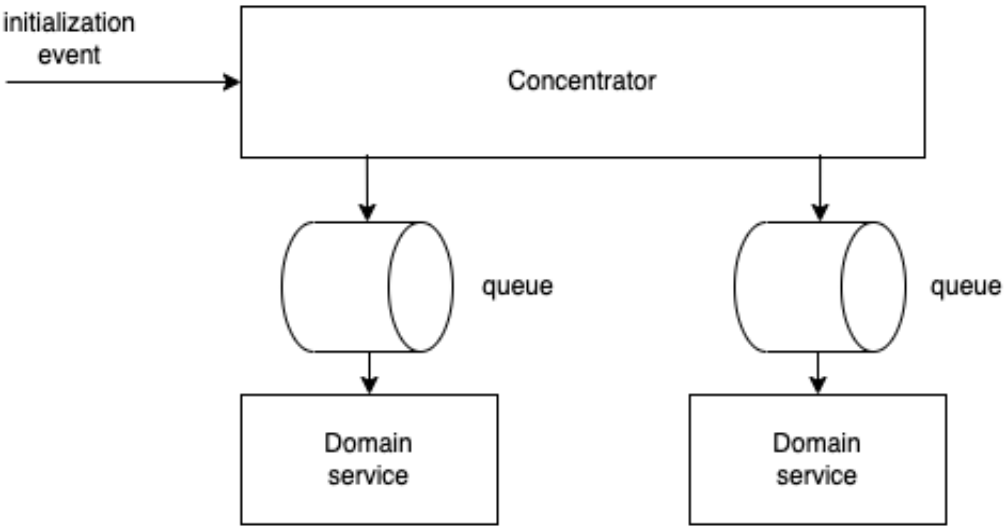


Рис. 2.26. EDA Middleware pattern

Тож для створення організаційної структури, в деяких випадках доречно скористатися паттерном Middleware. Це буде виглядати як одна команда з системним і централізованим баченям усієї системи поверх доменних команд. Також ця команда має містити продуктових спеціалістів для збереження централізованого знання предметної області. Суттєва перевага Domain Driven Design полягає в тому, що досягається пришвидшення роботи команди по мірі набуття знань з предметної області [35]. Тож маючи централізоване сховище та працівників в “над” команді, додається цілісність системі та хеджуються ризики втрати знання в якійсь певній команді (рис. 2.27).

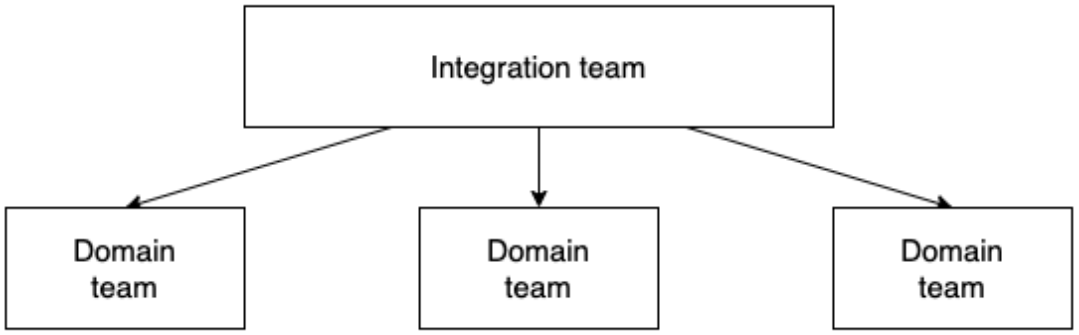


Рис. 2.27. Організаційна структура з інтеграційною командою

У досліджуваному проєкті буде доречно організувати роботу цієї команди всупереч принципам Agile, щоб команда займалася більше документацією, зберіганням та поширенням знання про роботу системи в цілому. Це буде противага Agile підходам які будуть панувати у доменних командах, таким чином вирівнюючи баланс. Також ця команда буде відповідати за системне еволюційне бачення продукту і контролювати відхилення від наміченого плану. Пропонується організувати склад команди з розробників та тестувальників, задачі яких будуть охоплювати всі частини системи, таким чином буде можливо зібрати якомога більше знання про систему та коректно інтегрувати всі частини системи.

Перевірка гіпотези

Для підтвердження цієї гіпотези на базі науково-дослідної практики було проаналізовано декілька команд які працювали над одним проєктом. Команди складались з розробників, які мають навички розробки клієнтських застосунків та серверної частини. Також команди мали по 1-2 мануальных тестувальника. Product Owner працював по запиту, тобто якщо в ньому була необхідність, він приєднався до мітингу. Кожна команда працювала за ітеративним фреймворком Scrum. Було передбачено всі загальні скрам-церемонії, такі як: ранковий дейлі, планування, сесія рефайнменту та ретроспектива. Тож були створені умови для безперешкодної комунікації всередині команд. Результатом кожної ітерації повинен був бути повноцінний відтестований інкремент продукту зі змінами по всій MVC архітектурі.

Інтеграційна команда складалась з декількох осіб, серед яких були інженери-розробники та product owners. Завданням технічних спеціалістів цієї команди було накопичення знань про технічні особливості інформаційної системи, документація їх та підтримка feature команд. У свою чергу Product owners акумулювали знання з продуктового аспекту, також вели певну документацію та підтримували розробників/product owners у scrum командах.

За результатами декількох ітерації деякі учасники були опитані та методом середньозважених експертних оцінок було отримано результати представлені в табл. 2.5.

Таблиця 2.5

Результати впровадження інтеграційної команди.

Критерій	Без інтеграційної команди	З інтеграційною командою	Приріст
Узгодженість архітектурних квантів	7	8	+14%

Продовження табл. 2.5

Зв’язаність архітектури	5	5	0
Вичерпність документації	3	6	+100%
Кількість багів пов’язаних з інтеграцією	4	2	-50%
Швидкість прийняття архітектурного рішення	5	6	+20%
Швидкість початку розробки нової фічі	5	6	+20%
Продуктові знання серед учасників команд	6	8	+33%
Технічні знання системи серед учасників команд	7	8	+14%
Ризик втрати частини зання або співробітника	5	4	-20%

Учасникам запропонували провести оцінку за 10-бальною шкалою за такими критеріями:

— Узгодженість архітектурних квантів - чи зберігається цілісність системи після додавання зміни до якогось певного архітектурного кванту.

— Зв’язаність архітектури - сила залежності архітектурних квантів один від одного

— Вичерпність документації - чи достатньо було інформації в документації для виконання поточного завдання.

— Кількість багів пов’язаних з інтеграцією - яка частота виникнення проблем при інтеграції модуля зі змінами.

— Швидкість прийняття архітектурного рішення - час витрачений на пошук архітектурного рішення.

— Швидкість початку розробки нової фічі - час витрачений на дослідження перед початком розробки.

— Продуктові знання серед учасників команд - вичерпність знань продукту за межами команди.

— Технічні знання системи серед учасників команд - вичерпність технічних знань за межами команди.

— Ризик втрати частини зання або співробітника - оцінка стабільності набутої бази знань при випаданні з неї частини знання або втрата співробітника.

Як видно на табл. 1, в цілому, зміни позитивно вплинули на хід розробки команд. Так, істотно зменшився час на прийняття архітектурного рішення та учасники команд стали менше витрачати часу на аналіз перед розробкою. Система стала більш узгодженою, що говорить про її стабільність. Учасники всіх команд розширили свої знання про сукупну систему з продуктової та технічної точки зору. Це зменшило ризик втрати зання при втраті співробітника.

Варто зазначити, що приведені результати оцінювались на проміжку всього декількох ітерацій, а такі зміни в організаційній структурі зазвичай

вимагають більшого періоду для більш точної оцінки. Але виходячи з позитивної динаміки, можна припустити, що надалі розрив між базою порівняння буде зростати у позитивний бік. Також аналіз проводився на трьох командах, що може вводити в оману. Для більш точного досліду потрібно взяти якомога більше команд, різні предметні області та різні архітектури.

Отже, беручи до уваги проведені дослідження, пропонується організаційна структура проєкту з інтеграційною командою (рис. 2.28), яка складається з розробника, Product Owner та деякі функціїї також виконує проєктний менеджер. Інтеграційна команда наглядає з двома доменими командами: Data representation team та Data source team. Перша команда займається презентацією зібраних даних. Вона складається з backend та frontend розробників та тестувальника. У свою чергу Data source team має тільки backend розробників, так як вона розробляє модуль збору даних з продуктових API. На проєкті також присутня команда девопсів, яка займається налагодженням інфраструктури інформаційної системи та допомагає розробникам з CI/CD.

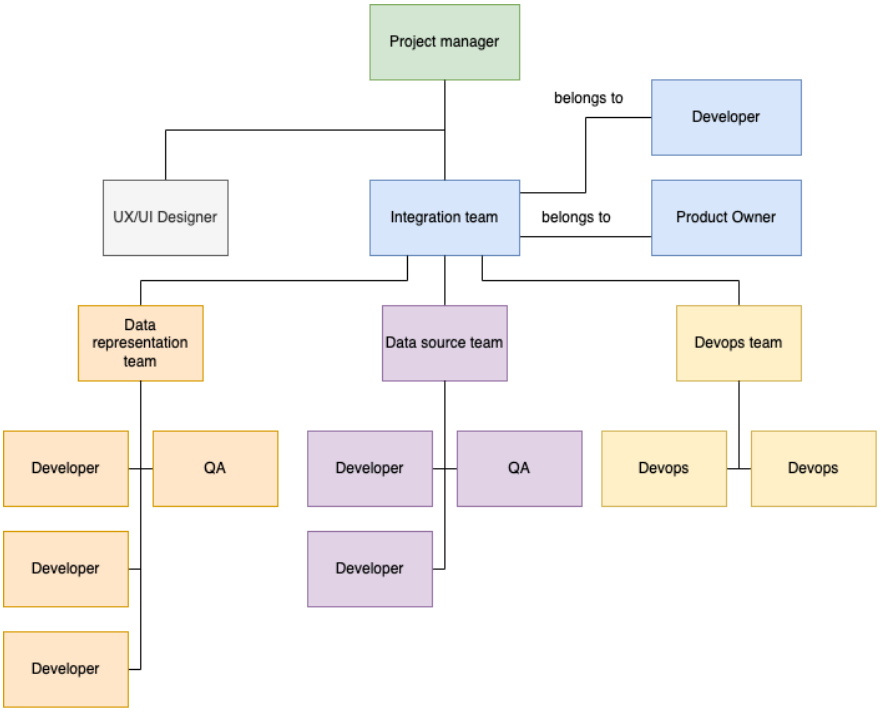


Рис. 2.28. Організаційна структура проєкту

РОЗДІЛ 3. ПЛАНУВАННЯ УПРАВЛІННЯ ПРОЄКТОМ

3.1 Зміст проєкту

За основу змісту проєкту розробки інформаційної системи з моніторингу цін було взято принципи стандартної моделі Software Development Life Cycle. Це класична модель якої дотримуються при плануванні розробки програмного продукту. Як видно на рисунку 3.1, було виділено основні пакети робіт:

— Аналіз.

— Вимоги.

— Дизайн.

— Реалізація.

— Розгортання.

— Документування.

— Завершення.

Спираючись на зазначені вище пакети робіт, було побудовано ієрархічну модель (Work Breakdown Structure). Так найбільше пакетів робіт виявилось у гілці Реалізація, вкладеність якої сягає шостого рівня. У свою чергу гілка Завершення має найменшу кількість робіт та найменший рівень вкладеності.

Варто зазначити, що складання змісту проєкту є одним з найважливіших етапів планування, оскільки після формування пакетів робіт складаються переліки робіт, які потім накладаються на календарний план. Тому загублення певної роботи може спричинити непередбачувані сдвиги у календарному плані.

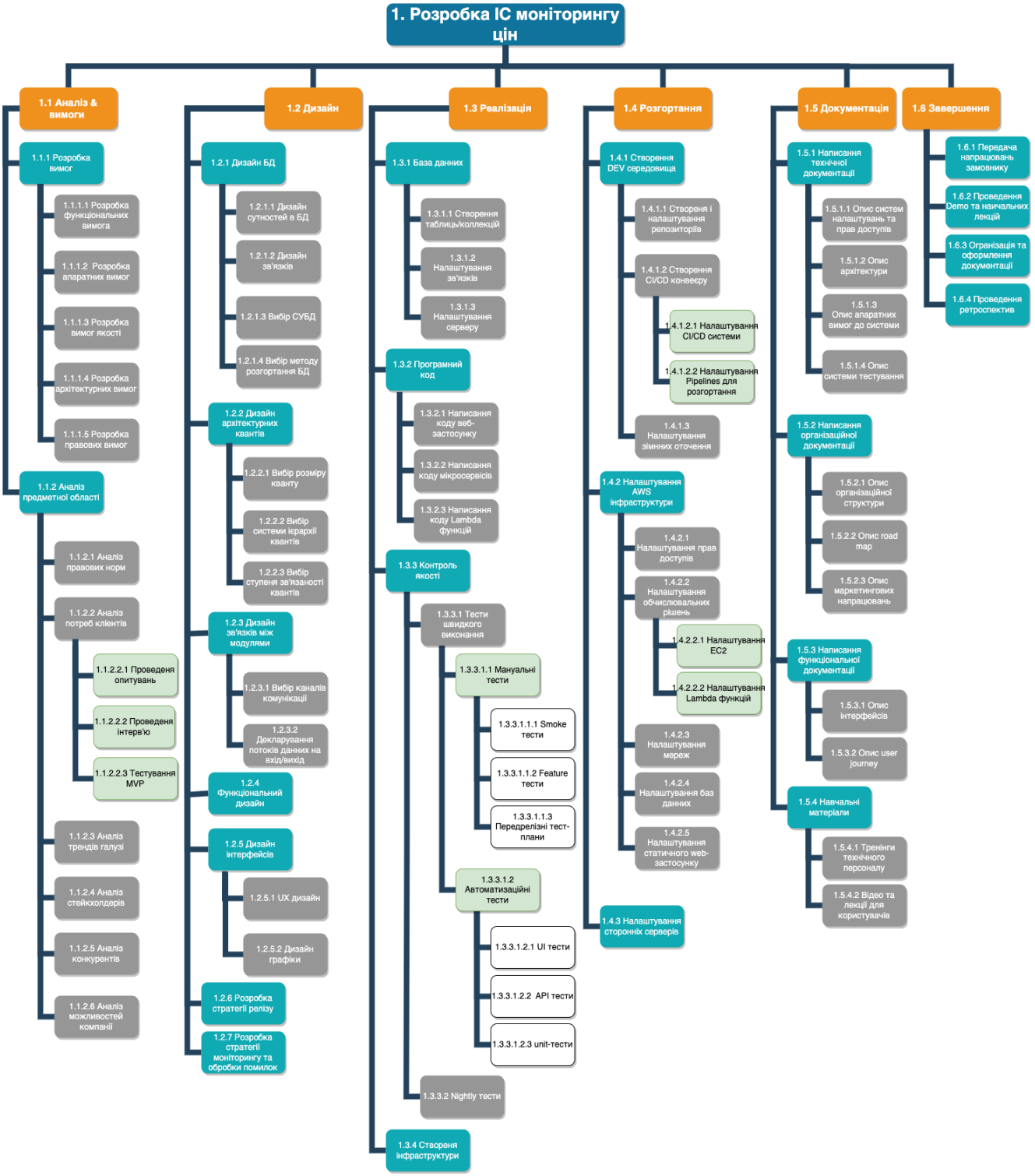


Рис. 3.1. WBS робіт по розробці ІС

3.2 Технологія управління проєктом

У наш час існує багато технологій з управління проєктами, від класичного підходу Каскад до продвинутого Scram, наприклад, Large-Scale Scrum. Усі вони мають свої переваги та недоліки. Одні технології доцільно використовувати на будівництві, деякі у військовому виробництві, ще інші у розробці програмного забезпечення і тд. При виборі технології варто брати до уваги не тільки предметну область проєкту, а ще й організаційну структуру, ресурси проєкту, часові обмеження, умови праці, кваліфікацію персоналу та інші фактори. Неправильний вибір технології може спричинити збитки компанії та віддалить проєкт від його цілей. Тому вибір технології управління проєктом є багатогранною та відповідальною задачею.

Розглянемо проєкт за декількома аспектами, які можуть наштовхнути на вибір доцільної технології. У підрозділі 2.9 розглядалось питання організаційної структури проєкту та дещо торкались вибору технології управління. Так у цьому підрозділі пропонувалось виділити дві доменні команди та одну інтеграційну команду. З урахуванням кількості членів команди, можна допустити, що роботу у цих командах можна організувати за Scrum технологією. Але цього недостатньо для прийняття остаточного рішення. Варто звернути увагу, що Scum побудований за принципами Agile та наслідує всі його цінності. Чотири основних це: люди та співпраця важливіші за процеси та інструменти, працюючий продукт важливіший за вичерпну документацію, позитивна співпраця із замовником важливіша за обговорення умов контракту, готовність до змін важливіша за дотримання плану [33]. Зазвичай такі цінності доречні до переважної більшості IT-продуктів, саме тому Agile набув широкої популярності у Software development. Важливим є те, що гнучкі методології обирають, коли немає впевненості у тому, який кінцевий результат має бути отриманим по закінченню розробки [32]. Завдяки своєї гнучкості Agile дозволяє вносити зміни по ходу проєкту. Розробка інформаційної системи з моніторингу

цін також передбачає деяку невизначеність у кінцевому продукті. Тож за цим критерієм Agile розробка підходить проєкту.

У підрозділі 2.7 представлено фінансову модель проєкту. На ній видно, що заплановані продажі повині початись ще до завершення всієї розробки, отже до цього моменту повинен вже бути якийсь продукт, який можна продемонструвати покупцю. Для цієї мети використовують ітераційну модель розробки, прикладом якої може бути Scrum. Продукт створюється невеликими ітераціями, по завершенню кожної є робочий програмний пакет, котрий нарощує вже наявний функціонал. Також рухаючись ітераціями продукт зменшує невизначеність та контрольовано рухається до цілей проєкту.

Для scrum-команд пропонується взяти двотижневу тривалість спринту. На початку спринта відбуваються колективні планінг та грумінг сесії. Впродовж всього спринта відбуваються ранкові зустрічі, де учасники команд у стислому форматі діляться своїми планами та даосягненями за минулий день. Кожний спринт завершується ретроспективою, на якій команда аналізує минулий спринт складаючи списки про те, що було добре зроблено та що можна покращити у наступній ітерації.

Варто також висвітлити питання естімації задач, так як від цього безпосередньо залежить швидкість та своєчасність виконання проєкту. Пропонується оцінювати задачі на планінг сесії, де всі учасники грають у так званий “скрам-покер”. Вони таємно оцінюють задачу за тривалістю виконання, а потім карти розкриваються та задачі присвоюється обговорена оцінка. Таким чином враховуються думки всіх гравців, незалежно від їхньої позиції у команді, та досягається більш точна оцінка задачі. Часто виникає суперечливе питання, за яким саме критерієм треба оцінювати задачу, за складністю або за тривалістю виконання? За складністю оцінюють у story points, порівнюючи складність задач між собою. Часова ж оцінка проводиться у годинах/днях. Ці підходи мають свої переваги та недоліки. Так вважається, що оцінку за

складністю проводити легше, адже для виставлення певного бала нам потрібно лише порівняти задачі між собою та виставити орієнтований story point. Маючи задачу оцінену у story points, ця оцінка стає універсальною для будь якого учасника команди. Так, наприклад, оцінюючи задачу у часах та присвоюючи оцінку, наприклад, чотири години. Стає не зовсім ясно, для кого саме виконання займає 4 години, для джуніор розробника чи сіньор? Але недоліком естімації в story points є те, що стає складніше планувати часовий хід проєкту, адже story points неможливо закласти в календарний план. Так, можливо вираховувати швидкість скрам-команди, тобто рахувати скільки story-points команда змогла виконати за спринт та приблизно припускати об'єм наступних спринтів. Але все ж таки цей підхід доволі незручний для часового планування та може давати великі відхилення за часом. На мою думку естімацію за складністю буде більш доречна на великих довготривалих проєктах зі значним фінансовим ресурсом, де немає суворого обмеження за часом. У такому випадку команда може дозволити собі працювати у прийнятному для себе темпі без тиску зі сторони менеджменту. Тож враховуючи те, що проєкт, описаний у цій дипломній роботі, має доволі жорсткі обмеження за часом та має чіткий часовий план окупності, варто на перших етапах обрати оцінку задач за часом виконання.

Як зазначалось у розділі 2.9, не всі команди будуть працювати за Scrum технологією. В організаційній структурі є ще інтеграційна команда, яка відповідає за напрямок проєкту та займається підтримкою доменних команд. В цьому ж розділі обгрунтовані причини та цілі цього підходу. Так було зазначено, щоб ефективно виконувати задачі, які поставлені перед цією командою, її роботу варто організувати всупереч принципам Agile. Для цього підійде модель дошки Kanban. Їхні задачі будуть пріоритезовані та будуть виконуватись у потрібній послідовності. Варто зазначити, що Kanban наслідує цінності Agile, а як зазначалося, команда має діяти всупереч цим принципам. Kanban це не

тільки дошка, а ціла методологія, тому пропонується обмежити використання Канбану тільки його дошкою.

Для початку розробки інформаційної системи потрібно зробити підготовчі кроки, такі як: аналіз, підготовка вимог, дизайн системи. Ці етапи окреслюють проєкт, дають йому направлення, але ще мають певний рівень невизначеності. Цю невизначеність має подолати ітераційна розробка, у даному випадку за Scrum моделью. Тож перші та останні фази проєкту проводяться за моделью Waterfall. Зміст та послідовність цих фаз більшою мірою збігаються з фазовою моделью представленою у PMBOK та у класичному Software Development Life Cycle. На рисунку 3.2 представлено послідовну модель життевого циклу. Як видно на малюнку, виділяються підготовчі фази, фази розробки та фаза завершення. Циклічні стрілки позначають замкнутий цикл робіт. Так як розробка відбувається ітеративно, у рамках Scrum технології, то цикл розробка => тестування => розгортання буде повторюватись на кожній ітерації.

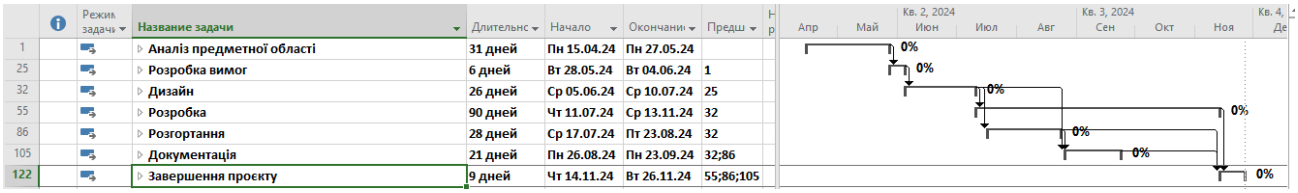
Рис. 3.2. Модель життєвого циклу розробки ІС

Отже, підсумовуючи все вищезазначене, робота у доменних командах буде відбуватись за Scrum технологією, включаючи усі необхідні scrum-церемонії. Задачі будуть оцінюватись за часовим критерієм з використанням Scrum poker. Організація роботи інтеграційної команди відбуватиметься за рахунок Kanban дошки, але всупереч деяким Agile

цінностям. Також початкові та завершальні етапи проєкту будуть йти послідовно за каскадною моделлю.

3.2 Планування управління часом

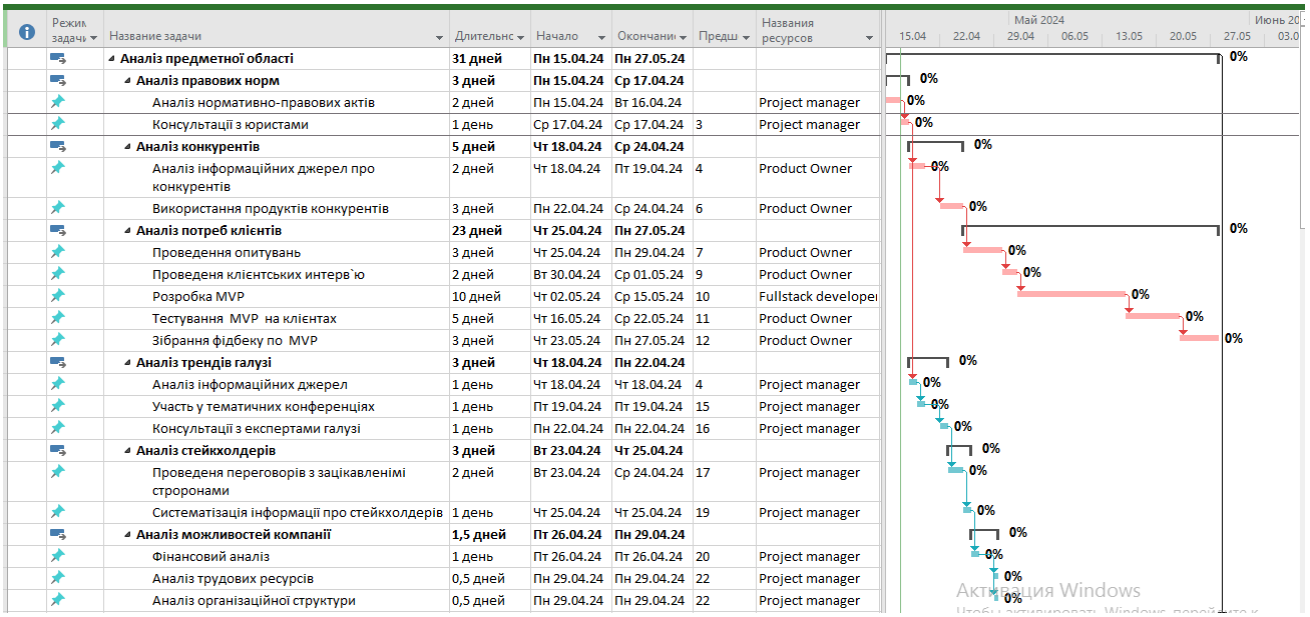
На базі WBS було побудовано календарний план у системі MS Project. Були додані роботи по розробці ІС, які в свою чергу згруповані по пакетам робіт. Загальний план по фазам життєвого цикла представлено на рисунку 3.3. Як видно робота над першим пакетом робіт почалась 15.04.24, а завершення всього життєвого циклу має відбутись 26 листопада 2024 року. Також видно які пакети робіт займають найбільше часу, у даному випадку це пакет розробки, який займає 90 днів. Варто ще звернути увагу на послідовність пакетів: аналіз, розробка вимог та дизайн йдуть послідовно, а розробка, тестування, розгортання та підготовка документації можуть виконуватися паралельно. Останнім пакетом йдуть роботи по завершенню проєкту. Всього календарний план налічує більше 120 робіт.



	Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предш
1		Аналіз предметної області	31 дней	Пн 15.04.24	Пн 27.05.24	
25		Розробка вимог	6 дней	Вт 28.05.24	Вт 04.06.24	1
32		Дизайн	26 дней	Ср 05.06.24	Ср 10.07.24	25
55		Розробка	90 дней	Чт 11.07.24	Ср 13.11.24	32
86		Розгортання	28 дней	Ср 17.07.24	Пт 23.08.24	32
105		Документація	21 дней	Пн 26.08.24	Пн 23.09.24	32;86
122		Завершення проєкту	9 дней	Чт 14.11.24	Вт 26.11.24	55;86;105

Рис. 3.3. Загальний план робіт

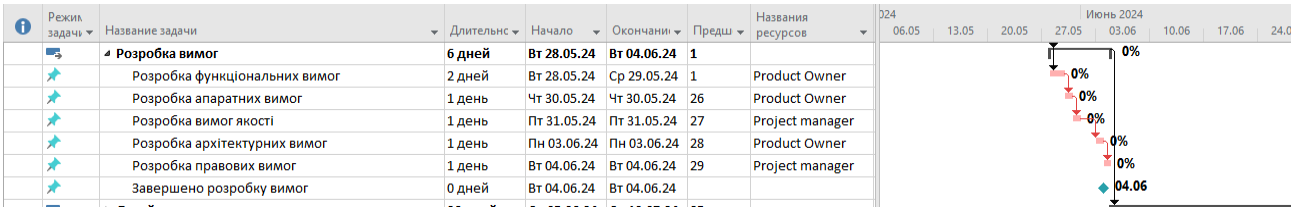
На рисунку 3.4 представлено перелік робіт для робіт аналізу предметної області перед самою розробкою. Роботи починаються 15 квітня та закінчуються 27 травня. Більшість робіт з цього пакету виконують Project manager та Product Owner, але також задіяний Fullstack розробник для підготовки MVP. Як видно на графіку, перша частина робіт проєкту входить в критичний шлях.



Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Аналіз предметної області	**31 дней**	**Пн 15.04.24**	**Пн 27.05.24**		
	Аналіз правових норм	**3 дней**	**Пн 15.04.24**	**Ср 17.04.24**		
	Аналіз нормативно-правових актів	2 дней	Пн 15.04.24	Вт 16.04.24		Project manager
	Консультації з юристами	1 день	Ср 17.04.24	Ср 17.04.24	3	Project manager
	Аналіз конкурентів	**5 дней**	**Чт 18.04.24**	**Ср 24.04.24**		
	Аналіз інформаційних джерел про конкурентів	2 дней	Чт 18.04.24	Пт 19.04.24	4	Product Owner
	Використання продуктів конкурентів	3 дней	Пн 22.04.24	Ср 24.04.24	6	Product Owner
	Аналіз потреб клієнтів	**23 дней**	**Чт 25.04.24**	**Пн 27.05.24**		
	Проведення опитувань	3 дней	Чт 25.04.24	Пн 29.04.24	7	Product Owner
	Проведеня клієнтських интерв`ю	2 дней	Вт 30.04.24	Ср 01.05.24	9	Product Owner
	Розробка MVP	10 дней	Чт 02.05.24	Ср 15.05.24	10	Fullstack developer
	Тестування MVP на клієнтах	5 дней	Чт 16.05.24	Ср 22.05.24	11	Product Owner
	Зібрання фідбеку по MVP	3 дней	Чт 23.05.24	Пн 27.05.24	12	Product Owner
	Аналіз трендів галузі	**3 дней**	**Чт 18.04.24**	**Пн 22.04.24**		
	Аналіз інформаційних джерел	1 день	Чт 18.04.24	Чт 18.04.24	4	Project manager
	Участь у тематичних конференціях	1 день	Пт 19.04.24	Пт 19.04.24	15	Project manager
	Консультації з експертами галузі	1 день	Пн 22.04.24	Пн 22.04.24	16	Project manager
	Аналіз стейкхолдерів	**3 дней**	**Вт 23.04.24**	**Чт 25.04.24**		
	Проведеня переговорів з зацікавленімі строронами	2 дней	Вт 23.04.24	Ср 24.04.24	17	Project manager
	Систематізація інформації про стейкхолдерів	1 день	Чт 25.04.24	Чт 25.04.24	19	Project manager
	Аналіз можливостей компанії	**1,5 дней**	**Пт 26.04.24**	**Пн 29.04.24**		
	Фінансовий аналіз	1 день	Пт 26.04.24	Пт 26.04.24	20	Project manager
	Аналіз трудових ресурсів	0,5 дней	Пн 29.04.24	Пн 29.04.24	22	Project manager
	Аналіз організаційної структури	0,5 дней	Пн 29.04.24	Пн 29.04.24	22	Project manager

Рис. 3.4. Календарний план робіт аналізу

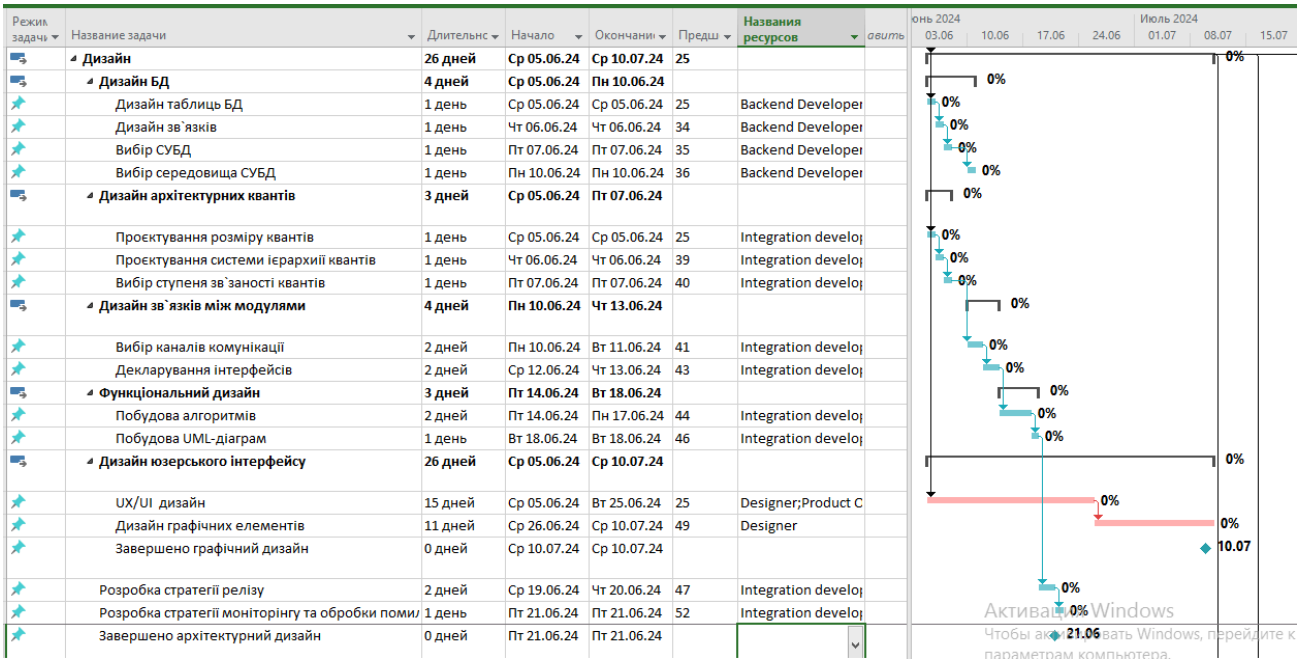
Роботи з розробки вимог виконує переважно Product Owner. Ці роботи також входять в критичний шлях лінійного графіку. Кінцем цього списку є віха “Завершено розробку вимог”, на графіку вона позначена як робота з нульовою тривалістю (рис. 3.5).



Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Розробка вимог	**6 дней**	**Вт 28.05.24**	**Вт 04.06.24**	**1**	
	Розробка функціональних вимог	2 дней	Вт 28.05.24	Ср 29.05.24	1	Product Owner
	Розробка апаратних вимог	1 день	Чт 30.05.24	Чт 30.05.24	26	Product Owner
	Розробка вимог якості	1 день	Пт 31.05.24	Пт 31.05.24	27	Project manager
	Розробка архітектурних вимог	1 день	Пн 03.06.24	Пн 03.06.24	28	Product Owner
	Розробка правових вимог	1 день	Вт 04.06.24	Вт 04.06.24	29	Project manager
	Завершено розробку вимог	0 дней	Вт 04.06.24	Вт 04.06.24		

Рис. 3.5. Календарний план розробки вимог

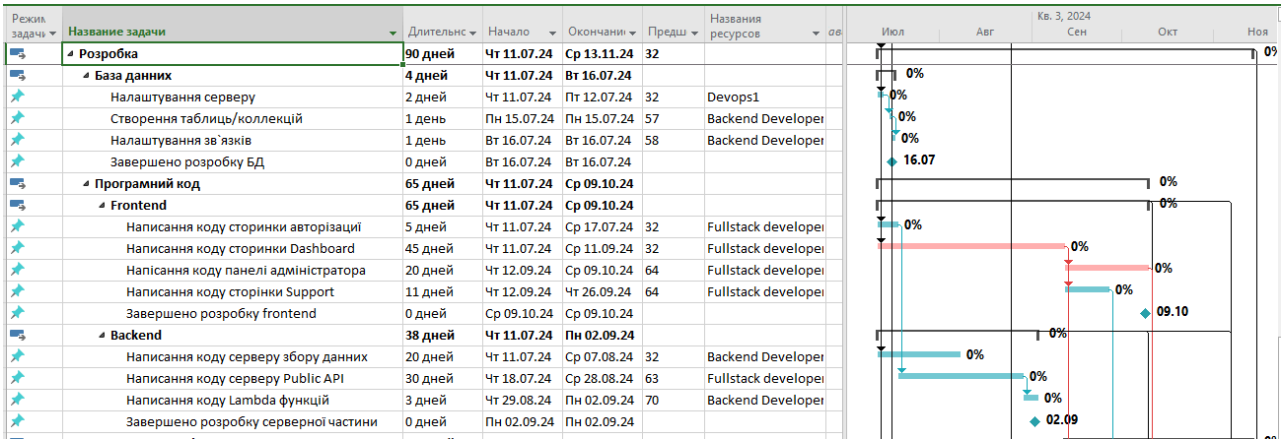
На рисунку 3.6 показано календарний план робіт з дизайну. Дизайн архітектурних компонентів здійснюють технічні спеціалісти: Integration developer та Backend developer. Дизайном клієнтського інтерфейсу займаються графічний дизайнер та Product Owner. Ця секція має дві віхи: завершення графічного дизайну та завершення архітектурного дизайну.



Режим задач	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Дизайн	**26 дней**	**Ср 05.06.24**	**Ср 10.07.24**	**25**	
	Дизайн БД	**4 дней**	**Ср 05.06.24**	**Пн 10.06.24**		
	Дизайн таблиць БД	1 день	Ср 05.06.24	Ср 05.06.24	25	Backend Developer
	Дизайн зв`язків	1 день	Чт 06.06.24	Чт 06.06.24	34	Backend Developer
	Вибір СУБД	1 день	Пт 07.06.24	Пт 07.06.24	35	Backend Developer
	Вибір середовища СУБД	1 день	Пн 10.06.24	Пн 10.06.24	36	Backend Developer
	Дизайн архітектурних квантів	**3 дней**	**Ср 05.06.24**	**Пт 07.06.24**		
	Проєктування розміру квантів	1 день	Ср 05.06.24	Ср 05.06.24	25	Integration develo
	Проєктування системи ієрархії квантів	1 день	Чт 06.06.24	Чт 06.06.24	39	Integration develo
	Вибір ступеня зв`заності квантів	1 день	Пт 07.06.24	Пт 07.06.24	40	Integration develo
	Дизайн зв`язків між модулями	**4 дней**	**Пн 10.06.24**	**Чт 13.06.24**		
	Вибір каналів комунікації	2 дней	Пн 10.06.24	Вт 11.06.24	41	Integration develo
	Декларування інтерфейсів	2 дней	Ср 12.06.24	Чт 13.06.24	43	Integration develo
	Функціональний дизайн	**3 дней**	**Пт 14.06.24**	**Вт 18.06.24**		
	Побудова алгоритмів	2 дней	Пт 14.06.24	Пн 17.06.24	44	Integration develo
	Побудова UML-діаграм	1 день	Вт 18.06.24	Вт 18.06.24	46	Integration develo
	Дизайн юзерського інтерфейсу	**26 дней**	**Ср 05.06.24**	**Ср 10.07.24**		
	UX/UI дизайн	15 дней	Ср 05.06.24	Вт 25.06.24	25	Designer;Product C
	Дизайн графічних елементів	11 дней	Ср 26.06.24	Ср 10.07.24	49	Designer
	Завершено графічний дизайн	0 дней	Ср 10.07.24	Ср 10.07.24		
	Розробка стратегії релізу	2 дней	Ср 19.06.24	Чт 20.06.24	47	Integration develo
	Розробка стратегії моніторінгу та обробки помил	1 день	Пт 21.06.24	Пт 21.06.24	52	Integration develo
	Завершено архітектурний дизайн	0 дней	Пт 21.06.24	Пт 21.06.24		

Рис. 3.6. Календарний план робіт дизайну

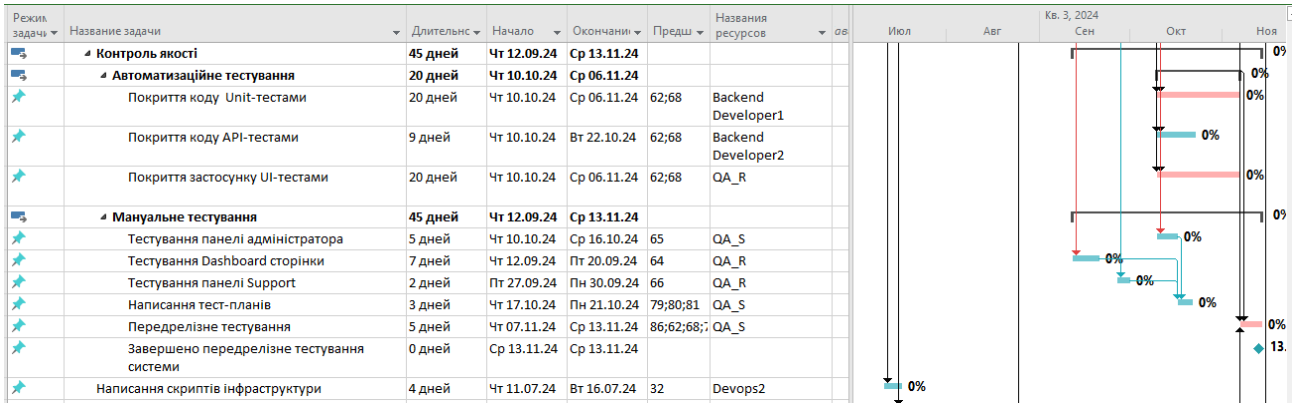
Часове планування розробки представлено на рисунку 3.7. Ці роботи починаються відразу після завершення дизайну системи та продовжуються на протязі 90 днів. На даному етапі задіяні тільки технічні спеціалісти, такі як розробники та девопси. У кінці розробки кожного пакету другого рівня є віха. Розробка фронтенду займає більшу частину часу.



Режим задач	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Розробка	**90 дней**	**Чт 11.07.24**	**Ср 13.11.24**	**32**	
	База данних	**4 дней**	**Чт 11.07.24**	**Вт 16.07.24**		
	Налаштування серверу	2 дней	Чт 11.07.24	Пт 12.07.24	32	Devops1
	Створення таблиць/коллекцій	1 день	Пн 15.07.24	Пн 15.07.24	57	Backend Developer
	Налаштування зв`язків	1 день	Вт 16.07.24	Вт 16.07.24	58	Backend Developer
	Завершено розробку БД	0 дней	Вт 16.07.24	Вт 16.07.24		
	Програмний код	**65 дней**	**Чт 11.07.24**	**Ср 09.10.24**		
	Frontend	**65 дней**	**Чт 11.07.24**	**Ср 09.10.24**		
	Написання коду сторинки авторізації	5 дней	Чт 11.07.24	Ср 17.07.24	32	Fullstack developer
	Написання коду сторинки Dashboard	45 дней	Чт 11.07.24	Ср 11.09.24	32	Fullstack developer
	Напісання коду панелі адміністратора	20 дней	Чт 12.09.24	Ср 09.10.24	64	Fullstack developer
	Написання коду сторінки Support	11 дней	Чт 12.09.24	Чт 26.09.24	64	Fullstack developer
	Завершено розробку frontend	0 дней	Ср 09.10.24	Ср 09.10.24		
	Backend	**38 дней**	**Чт 11.07.24**	**Пн 02.09.24**		
	Написання коду серверу збору данних	20 дней	Чт 11.07.24	Ср 07.08.24	32	Backend Developer
	Написання коду серверу Public API	30 дней	Чт 18.07.24	Ср 28.08.24	63	Fullstack developer
	Написання коду Lambda функцій	3 дней	Чт 29.08.24	Пн 02.09.24	70	Backend Developer
	Завершено розробку серверної частини	0 дней	Пн 02.09.24	Пн 02.09.24		

Рис. 3.7. Календарний план пакету розробки

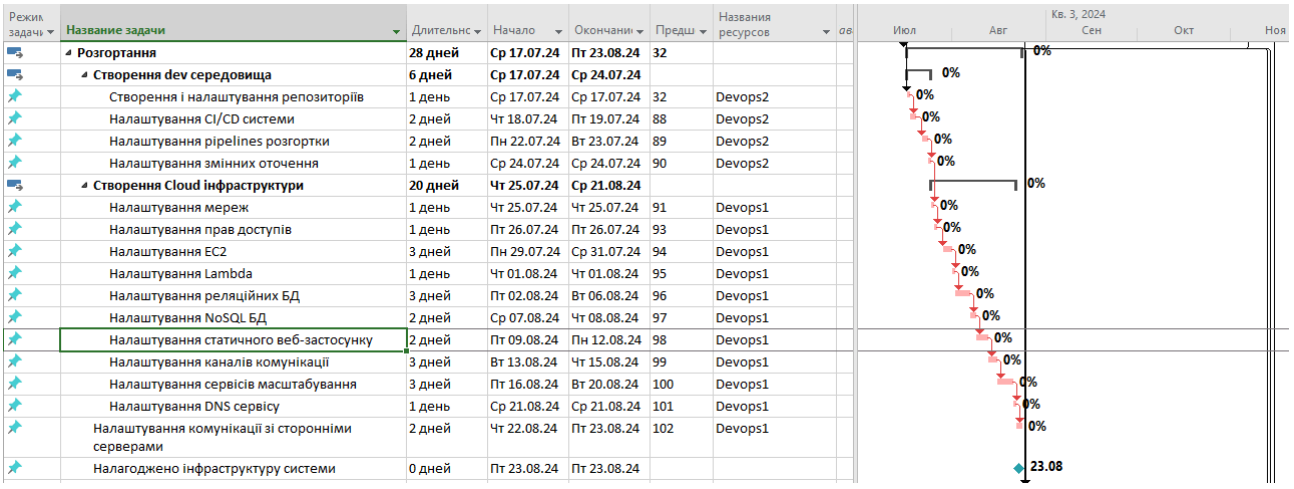
Календарний план з контролю якості представлено на рисунку 3.8. Деякі з цих робіт йдуть паралельно з розробкою, однак деякі види тестування вимагають цілісної інформаційної системи, тому ці роботи виконуються в останню чергу. На даному етапі роботу виконують переважно спеціалісти з контролю якості, але unit та api тестування покладено на розробників. Кінцевим етапом тестування є передрелізне тестування, воно має завершитись 13 листопада, що є віхою по завершенню тестування.



Режим задачі	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Контроль якості	**45 дней**	**Чт 12.09.24**	**Ср 13.11.24**		
	Автоматизаційне тестування	**20 дней**	**Чт 10.10.24**	**Ср 06.11.24**		
	Покриття коду Unit-тестами	20 дней	Чт 10.10.24	Ср 06.11.24	62;68	Backend Developer1
	Покриття коду API-тестами	9 дней	Чт 10.10.24	Вт 22.10.24	62;68	Backend Developer2
	Покриття застосунку UI-тестами	20 дней	Чт 10.10.24	Ср 06.11.24	62;68	QA_R
	Мануальне тестування	**45 дней**	**Чт 12.09.24**	**Ср 13.11.24**		
	Тестування панелі адміністратора	5 дней	Чт 10.10.24	Ср 16.10.24	65	QA_S
	Тестування Dashboard сторінки	7 дней	Чт 12.09.24	Пт 20.09.24	64	QA_R
	Тестування панелі Support	2 дней	Пт 27.09.24	Пн 30.09.24	66	QA_R
	Написання тест-планів	3 дней	Чт 17.10.24	Пн 21.10.24	79;80;81	QA_S
	Передрелізне тестування	5 дней	Чт 07.11.24	Ср 13.11.24	86;62;68;7	QA_S
	Завершено передрелізне тестування системи	0 дней	Ср 13.11.24	Ср 13.11.24		
	Написання скриптів інфраструктури	4 дней	Чт 11.07.24	Вт 16.07.24	32	Devops2

Рис. 3.8. Календарний план робіт з контролю якості

Налагодження інфраструктури починається після завершення дизайну і йдуть паралельно з розробкою (рис. 3.9). На цьому етапі задіяні тільки Devops спеціалісти. 23 серпня є датою закінчення інфраструктурних робіт.



Режим задачі	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Розгортання	**28 дней**	**Ср 17.07.24**	**Пт 23.08.24**	**32**	
	Створення dev середовища	**6 дней**	**Ср 17.07.24**	**Ср 24.07.24**		
	Створення і налаштування репозиторіїв	1 день	Ср 17.07.24	Ср 17.07.24	32	Devops2
	Налаштування CI/CD системи	2 дней	Чт 18.07.24	Пт 19.07.24	88	Devops2
	Налаштування pipelines розгортки	2 дней	Пн 22.07.24	Вт 23.07.24	89	Devops2
	Налаштування змінних оточення	1 день	Ср 24.07.24	Ср 24.07.24	90	Devops2
	Створення Cloud інфраструктури	**20 дней**	**Чт 25.07.24**	**Ср 21.08.24**		
	Налаштування мереж	1 день	Чт 25.07.24	Чт 25.07.24	91	Devops1
	Налаштування прав доступів	1 день	Пт 26.07.24	Пт 26.07.24	93	Devops1
	Налаштування EC2	3 дней	Пн 29.07.24	Ср 31.07.24	94	Devops1
	Налаштування Lambda	1 день	Чт 01.08.24	Чт 01.08.24	95	Devops1
	Налаштування реляційних БД	3 дней	Пт 02.08.24	Вт 06.08.24	96	Devops1
	Налаштування NoSQL БД	2 дней	Ср 07.08.24	Чт 08.08.24	97	Devops1
	Налаштування статичного веб-застосунку	2 дней	Пт 09.08.24	Пн 12.08.24	98	Devops1
	Налаштування каналів комунікації	3 дней	Вт 13.08.24	Чт 15.08.24	99	Devops1
	Налаштування сервісів масштабування	3 дней	Пт 16.08.24	Вт 20.08.24	100	Devops1
	Налаштування DNS сервісу	1 день	Ср 21.08.24	Ср 21.08.24	101	Devops1
	Налаштування комунікації зі сторонніми серверами	2 дней	Чт 22.08.24	Пт 23.08.24	102	Devops1
	Налагоджено інфраструктуру системи	0 дней	Пт 23.08.24	Пт 23.08.24		

Рис. 3.9. Календарний план робіт розгортання ІС

Опис архітектурних та функціональних можливостей системи проводиться паралельно з розробкою. Тож ці роботи починаються після завершення фази дизайну (рис. 3.10). Технічну документацію складають девопси, розробники та тестувальники. Організаційну документацію підготовляє проєктний менеджер, функціональну документацію - дизайнер, а навчальні матеріали складає Product owner. Загалом на документування відводиться 21 день.

Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Документація	**21 дней**	**Пн 26.08.24**	**Пн 23.09.24**	**32;86**	
	Технічна документація	**12 дней**	**Пн 26.08.24**	**Вт 10.09.24**		
	Опис систем налаштувань та розгортання	3 дней	Пн 26.08.24	Ср 28.08.24	32;86	Devops2
	Опис юзерських ролей та прав доступів	1 день	Чт 29.08.24	Чт 29.08.24	107	Product Owner
	Опис архітектури	4 дней	Пт 30.08.24	Ср 04.09.24	108	Integration develo
	Опис апаратних вимог	1 день	Чт 05.09.24	Чт 05.09.24	109	Devops2
	Опис системи тестування	3 дней	Пт 06.09.24	Вт 10.09.24	110	QA_R
	Організаційна документація	**5 дней**	**Пн 26.08.24**	**Пт 30.08.24**		
	Опис організаційної структури	1 день	Пн 26.08.24	Пн 26.08.24	32;86	Project manager
	Опис roadmap	2 дней	Вт 27.08.24	Ср 28.08.24	113	Project manager
	Опис маркетингових напрацювань	2 дней	Чт 29.08.24	Пт 30.08.24	114	Project manager
	Функциональна документація	**4 дней**	**Пн 26.08.24**	**Чт 29.08.24**		
	Опис інтерфейсів	2 дней	Пн 26.08.24	Вт 27.08.24	32;86	Designer
	Опис user journey	2 дней	Ср 28.08.24	Чт 29.08.24	117	Designer
	Навчальні матеріали	**9 дней**	**Ср 11.09.24**	**Пн 23.09.24**		
	Підготовка тренингів для тех. Персоналу	5 дней	Ср 11.09.24	Вт 17.09.24	115;118;1	Product Owner
	Підготовка відеолекцій для користувачів	4 дней	Ср 18.09.24	Пн 23.09.24	120	Product Owner

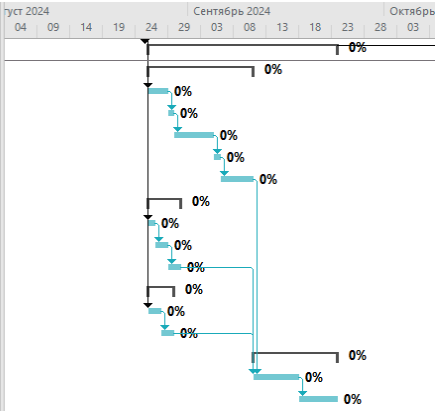


Рис. 3.10. Календарний план пакету робіт з документування

Останнім кроком у життєвому циклі є завершальний пакет робіт (рис. 3.11). Тут включені роботи з остаточного оформлення та організації документації, проведення ретроспектив, демо та фінальна комунікація з замовником. Основну роботу на цьому етапі виконує проєктний менеджер. В кінці робіт стоїть віха - 26 листопада - це планова дата завершення робіт проєкту.

Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предш	Названия ресурсов
	Завершення проєкту	**9 дней**	**Чт 14.11.24**	**Вт 26.11.24**	**55;86;105**	
	Організація та оформлення документації	3 дней	Чт 14.11.24	Пн 18.11.24	**105;55;86**	Project manager
	Проведення ретроспектив	2 дней	Вт 19.11.24	Ср 20.11.24	123	Project manager
	Проведення Demo та навчальних лекцій	2 дней	Чт 21.11.24	Пт 22.11.24	124	Product Owner
	Передача проєктних артефактів замовнику	2 дней	Пн 25.11.24	Вт 26.11.24	125	Project manager
	Виконані всі церемонії по завершеню проєкта	0 дней	Вт 26.11.24	Вт 26.11.24		

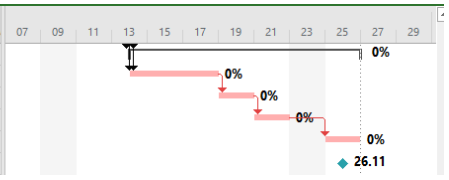


Рис. 3.11. Календарний план завершальних робіт

Далі варто розглянути задіяні ресурси, лист ресурсів представлено на рисунку 3.12. Під час планування робіт були виокремлені трудові ресурси, ресурси з закупівель, наприклад, ноутбуки, будуть пораховані окремо у кошторисі. Кожному трудовому ресурсу була призначена почасова ставка, а також понаднормаова ставка, яка рахувалась як 150% від стандартної ставки. Також обрано пропорційний метод нарахування та стандартний календар.

	Название ресурса	Тип	Единицы измерения материалс	Краткое названи	Группа	Макс. едини	Стандартна ставка	Ставка сверхурочн	Затраты на использ.	Начисление	Базовый календарь
1	Project manager	Трудовой		PM		100%	600,00₴/ч	900,00₴/ч	0,00₴	Пропорционалı	Стандартный
2	Integration developer	Трудовой		ID		100%	1 000,00₴/ч	1 500,00₴/ч	0,00₴	Пропорционалı	Стандартный
3	Product Owner	Трудовой		PO		100%	600,00₴/ч	900,00₴/ч	0,00₴	Пропорционалı	Стандартный
4	Designer	Трудовой		Des		100%	350,00₴/ч	525,00₴/ч	0,00₴	Пропорционалı	Стандартный
5	Fullstack developer1	Трудовой		F1		100%	700,00₴/ч	1 050,00₴/ч	0,00₴	Пропорционалı	Стандартный
6	Fullstack developer2	Трудовой		F2		100%	700,00₴/ч	1 050,00₴/ч	0,00₴	Пропорционалı	Стандартный
7	Fullstack developer3	Трудовой		F3		100%	700,00₴/ч	1 050,00₴/ч	0,00₴	Пропорционалı	Стандартный
8	Backend Developer1	Трудовой		B1		100%	700,00₴/ч	1 050,00₴/ч	0,00₴	Пропорционалı	Стандартный
9	Backend Developer2	Трудовой		B2		100%	700,00₴/ч	1 050,00₴/ч	0,00₴	Пропорционалı	Стандартный
10	Devops1	Трудовой		D1		100%	830,00₴/ч	1 245,00₴/ч	0,00₴	Пропорционалı	Стандартный
11	Devops2	Трудовой		D2		100%	830,00₴/ч	1 245,00₴/ч	0,00₴	Пропорционалı	Стандартный
12	QA_R	Трудовой		QR		100%	350,00₴/ч	525,00₴/ч	0,00₴	Пропорционалı	Стандартный
13	QA_S	Трудовой		QS		100%	350,00₴/ч	525,00₴/ч	0,00₴	Пропорционалı	Стандартный

Рис. 3.12. Лист ресурсів

З точки зору проєктного менеджменту, для розуміння та оптимізації витрат, варто розглянути вартість кожного пакету робіт. Для цього було використано опцію формування звітності в MS Project (рис. 3.13). Як видно в таблиці, найбільшу вартість складають роботи розробки - 1 255 040 грн., а найменшу - розробка вимог - 28 800 грн. Загальна вартість робіт проєкту представлена в Додатку Б та складає більше 2 млн. грн.

У даній звітності представлено вартісне планування тільки трудових ресурсів, вони зазвичай складають найбільшу частину витрат IT-проєктів, тому є вкрай важливими для оцінки. Моделювання загальних витрат представлено у підрозділі 2.7.

СВЕДЕНИЯ О ЗАТРАТАХ

Сведения о затратах для всех задач верхнего уровня.

Название	Фиксированные затраты	Фактические затраты	Оставшиеся затраты	Затраты	Базовые затраты	Отклонение по стоимости
Аналіз предметної області	0,00₴	0,00₴	347 200,00₴	347 200,00₴	0,00₴	347 200,00₴
Розробка вимог	0,00₴	0,00₴	28 800,00₴	28 800,00₴	0,00₴	28 800,00₴
Дизайн	0,00₴	0,00₴	271 200,00₴	271 200,00₴	0,00₴	271 200,00₴
Розробка	0,00₴	0,00₴	1 255 040,00₴	1 255 040,00₴	0,00₴	1 255 040,00₴
Розгортання	0,00₴	0,00₴	185 920,00₴	185 920,00₴	0,00₴	185 920,00₴
Документація	0,00₴	0,00₴	150 160,00₴	150 160,00₴	0,00₴	150 160,00₴
Завершення проєкту	0,00₴	0,00₴	43 200,00₴	43 200,00₴	0,00₴	43 200,00₴

Рис. 3.13. Звіт з вартості робіт

Загальний огляд ресурсів наведено у Додатку Б. На ньому представлено використання трудових ресурсів. Видно, що розробники та Product Owner мають найбільші витрати за часом. У свою чергу, QA_S та Devops2 витрачають найменше часу на роботи проєкту.

3.3 Планування управління ризиками

Ризик менеджмент є одним з ключових процесів фази планування за PMBOK.

Ще на перших етапах визначення та планування проєкту, варто розглянути питання з оцінки ризиків. Оскільки у найгіршому сценарії проєкт може зіштовхнутися з непереборними перешкодами, що може поставити існування всього проєкту під питання. Тож варто визначити модель за якою будуть виявлятися та оцінюватися ризики, окреслити організаційну структуру, яка буде займатися питаннями ризиків та розробити план моніторингу і систему антиризикових заходів.

Обираючи стандарт управління ризиками, варто зазначити, що за проєкт взято розробку інформаційної системи, тому має сенс розглянути стандарт PMBOK, так як він більш специфічно орієнтований на проєкт, а не на загальну компанію, як, наприклад, ISO31000. Він описує достатньо інструментів для виявлення, планування, оцінки, реагування та моніторингу ризиків. Стандарт виокремлює 7 процесів:

— Планування управління ризиками
— Ідентифікація ризиків
— Якісний аналіз ризиків
— Кількісний аналіз ризиків
— Планування реагування на ризики
— Реагування на ризик
— Моніторинг ризиків

Також є доцільним запозичити деякі підходи з MSF, цей фреймворк був розроблений під інформаційні продукти, тому він містить вичерпний опис аспектів які треба брати до уваги розробляючи інформаційну систему.

На початку розробки проєкту доцільно оцінити ризики за стандартом PMBOK, тобто створити план управління та реагування, ідентифікувати та провести аналіз ризиків. Це буде основа для старту проєкту зі сторони ризик менеджменту. Під час виконання та на фазі підтримки проєкту можна застосувати принципи з MSF (рис. 3.14).

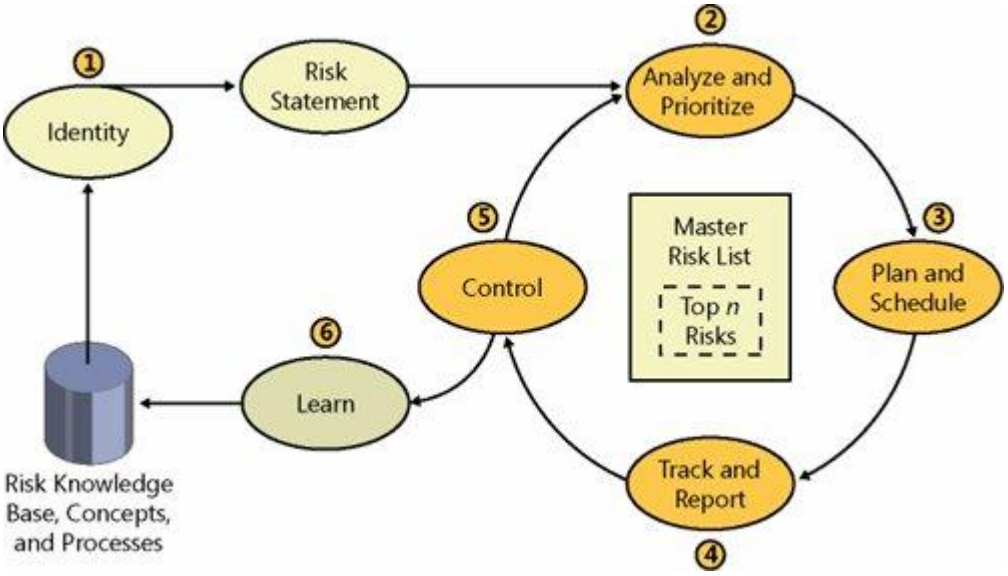


Рис. 3.14. Модель управління ризиками за MSF

MSF пропонує циклічну модель управління ризиками, що буде органічно поєднуватись зі Scrum підходом розробки ІС. По мірі того як інкрементально буде нарощуватися продукт, паралельно буде відбуватися обробка ризиків, які можуть виникати з урахуванням нових обставин.

Описуючи організаційну структуру управління ризиками, варто зазначити, що проєкт передбачає досить обмежений бюджет та склад команди є трохи більше десятка працівників. Тому враховуючи розміри проєкту, доцільно буде сконцентрувати такі повноваження на проєктному менеджері. Він переважно буде сфоккусований на зовнішніх ризиках проєкту, а з внутрішніми ризиками йому буде допомагати працівники технічної команди. Зібравши якомога повну інформацію про можливі ризики, він повинен її оцінити та запровадити план реагування, також по мірі просування проєкту, приділяти час на виявлення потенційних ризиків та контролювати існуючі.

У табл. 3.1 представлено якісну оцінку ризиків за простою шкалою. Усі ризики поділено на групи: організаційні, фінансові, технічні, зовнішні та форс мажори. Як видно на таблиці, найбільш небезпечними для проєкту є ризики пов’язані з затримкою/припиненням фінансування та ризики державного регулювання цін. Ці ризики мають високу силу впливу та низьку керованість.

Отже, цим та іншим ризикам з високою силою впливу потрібно приділити найбільше уваги.

Таблиця 3.1

Якісна оцінка ризиків

№	Тип ризику	Ризикова подія	Сили впливу	Керованість
1	Організаційні	Висока плинність кадрів	Середня	Висока
2		Конфлікти	Середня	Висока
3		Недостатність персоналу	Середня	Висока
4		Недостатність кваліфікації персоналу	Висока	Середня
5	Фінансові ризики	Припинення/затримки фінансування проєкту	Висока	Низька
6		Здорожчання персоналу	Середня	Низька
7		Здорожчання послуг субпідрядника	Середня	Низька
8	Технічні ризики	Обмеження в архітектурі ІС	Висока	Середня
9		Критичні дефекти	Середня	Висока
10		Нестача запланованих обчислювальних потужностей	Середня	Середня
11		Втрата знать про систему	Середня	Висока
12		Хакерська атака	Висока	Середня

Продовження табл. 3.1

13	Зовнішні ризики	Зміна оподаткування	Низька	Низька
14		Вихід на ринок конкурентів з більш досконалим продуктом	Висока	Середня
15		Державне регулювання цін, що робить проєкт непотрібним	Висока	Низька
16	Форс мажори	Військові дії	Низька	Низька
17		Стихійні лиха	Низька	Низька
18		Введення особливих заборонних актів (карантин)	Низька	Низька

На табл. 3.2 представлено кількісну оцінку ризиків. Для оцінки кількісного показника була використана квазікількісна шкала від 0 до 10. Останній параметр обчислено як добуток фінансових втрат на ймовірність. Згідно комплексного показника усі ризики було розділено на групи:

— червоний колір - найважливіші
— жовтий - середні
— сірий - менш важливі

Таблиця 3.2

Кількісна оцінка ризиків

№	Ризикова подія	Затримки у часі		Фінансові втрати		Ймовірність		Частота (за проект)		Важливість ризику (компл. показн.)
		Якіс. оц.	Кільк. оц.	Якіс. оц.	Кільк. оц.	Якіс. оц.	Кільк. оц.	Якіс. оц.	Кільк. оц.	
1	Висока плинність кадрів	с	4	с	3	с	3	н	1	9
2	Конфлікти у команді	с	4	н	2	н	2	с	3	4
3	Недостатність персоналу	в	8	н	2	н	2	н	1	4
4	Недостатність кваліфікації персоналу	в	7	н	2	н	2	н	2	4
5	Припинення/затримки фінансування проєкту	в	9	в	9	н	2	н	2	18
6	Здорожчання персоналу	с	3	с	4	н	2	н	1	8
7	Здорожчання послуг субпідрядника	н	2	с	4	н	2	н	2	8
8	Обмеження в архітектурі ІС	в	8	н	2	с	4	н	2	8
9	Критичні дефекти	с	4	н	2	с	4	с	4	8
10	Нестача запланованих обчислювальних потужностей	н	2	с	3	н	2	н	2	6
11	Втрата знать про систему	с	5	с	4	н	2	н	1	8
12	Хакерська атака	н	2	н	1	н	1	н	1	1
13	Зміна оподаткування	н	1	с	3	н	1	н	1	3

Продовження табл. 3.2

14	Вихід на ринок конкурентів з більш досконалим продуктом	н	2	с	4	с	3	н	1	12
15	Державне регулювання цін, що робить проєкт непотрібним	н	1	в	9	н	2	н	1	18
16	Військові дії	с	3	н	2	н	2	н	1	4
17	Стихійні лиха	с	3	с	3	н	1	н	1	3
18	Введення особливих заборонних актів (карантин)	н	2	н	1	н	1	н	1	1

Як видно на табл. 3.2, ризики державного регулювання, поява нових конкурентів та проблеми фінансування становлять найбільшу загрозу.

На табл. 3.3 представлено план протиризикових заходів для ризиків з найбільшим пріоритетом. Наведено план дій при профілактиці, появі симптомів та при проблемі.

Таблиця 3.3

Протиризикові заходи

№	Ризикова подія	ПРЗ 1	Симптом (рання ознака)	ПРЗ 2	ПРЗ 3
		профілактика		при симптомі	при проблемі
1	2	3	4	5	6
1	Припинення/затримки фінансування проєкту	Укласти контракт за замовником, який передбачає	Виникають невеликі затримки у фінансуванні	Провести консультації з фінансовим менеджером	Якщо це затримка, то провести з командою бесіду,

Продовження табл. 3.3

1	2	3	4	5	6
		оплату заздалегідь на декілька місяців вперед	або компанія втрачає ресурс для інвестування	компанії.	що це тимчасово і фінансові труднощі буде подолано з часом. Якщо повне припинення фінансування, то зупинити проєкт або пошук нових інвесторів.
2	Вихід на ринок конкурентів з більш досконалим продуктом	Моніторити гравців ринку та тренди галузі	Конкуренти змінюють стек технологій або кількість працівників. У галузі з'явилися прогнози щодо нових майбутніх фіч.	Обговорити з менеджером продукту зміни конкурентів або ринку.	Проаналізувати переваги конкурентів та вдосконалити свій продукт.

Закінчення табл. 3.3

1	2	3	4	5	6
3	Державне регулювання цін, що робить проєкт непотрібним	Відслідковувати новини уряду.	Швидке зростання цін.	Провести підготовчі заходи по закриттю/заморозці проєкту, щоб у разі настання несприятливих умов без перешкодно закрити проєкт. Або проаналізувати можливий вихід на ринок іншої країни.	Закрити/заморозити проєкт або спробувати вийти на міжнародний ринок.
4	Висока плинність кадрів	Проводити 1to1 мітинги, для виявлення побажань працівників .Покращувати умови праці. Слідкувати за трендами ринку праці.	Звільнення хоча б одного працівника. Поганий настрій у команді. Працівники виказують невдоволення.	Провести 1to1 мітинги для виявлення невдоволень та намагатися їх подолати.	Виявити причину звільнень та усунути її. Наймати нових працівників по вже набутим критеріях.

Усі перелічені протиризикові заходи не вимагають великих затрат, для їх реалізації потрібно лише час та компетенція проєктного менеджера, а також інших спеціалістів, в залежності від ризику: HR - якщо це плинність кадрів; юрист - якщо це укладання контрактів при фінансових ризиках тощо. Розуміючи, що час проєктного менеджера дуже обмежений, але й ризики вище відносяться до категорії високого впливу, тому буде доцільно виділяти час хоча б раз на тиждень для проведення профілактичних заходів.

3.4 Планування управління якістю

Торкаючись питання управління якістю, варто ще раз поглянути на проєкт з точки зору доменної області, користувача та цілей проєкту. Це потрібно для того, щоб зрозуміти які критерії якості потрібні та кількість ресурсів які можуть бути витрачені для досягнення певного рівня якості.

Розглядаючи доменну область, важливо зазначити, що основним вихідним продуктом інформаційної системи є цифри. Цифри від точності яких залежить правильність та своєчасність дій користувача на своєму ринку. Користувачем системи виступають великі та малі компанії, тобто вони вкрай зацікавлені в точності інформації, яку надає система. В іншому випадку вони можуть понести значні збитки. Зазвичай, так звані ентерпрайз кастомери, дуже ретельно ставляться до вибору інформаційної системи від якої залежить прийняття управлінських рішень. Тому при виборі системи вони звертають увагу на правдивість обчислень та на систему контролю якості. Адже купити кота в мішку - це для них дуже ризикований крок. Тож при тривалій багатоетапній B2B продажі, технічні фахівці користувача можуть навіть зазирнути вглиб продукту, щоб впевнитись, що необхідні норми якості були дотримані. З цього можна зробити висновок, що питання якості для проєкту є важливим та вимагає приділення значної уваги.

Коли йдеться про високу якість IT-проєкта, доцільно організувати автоматизоване тестування у вигляді класичної піраміди тестів [34], вона представлена на рисунку 3.15 з невеликою модифікацією під проєкт.

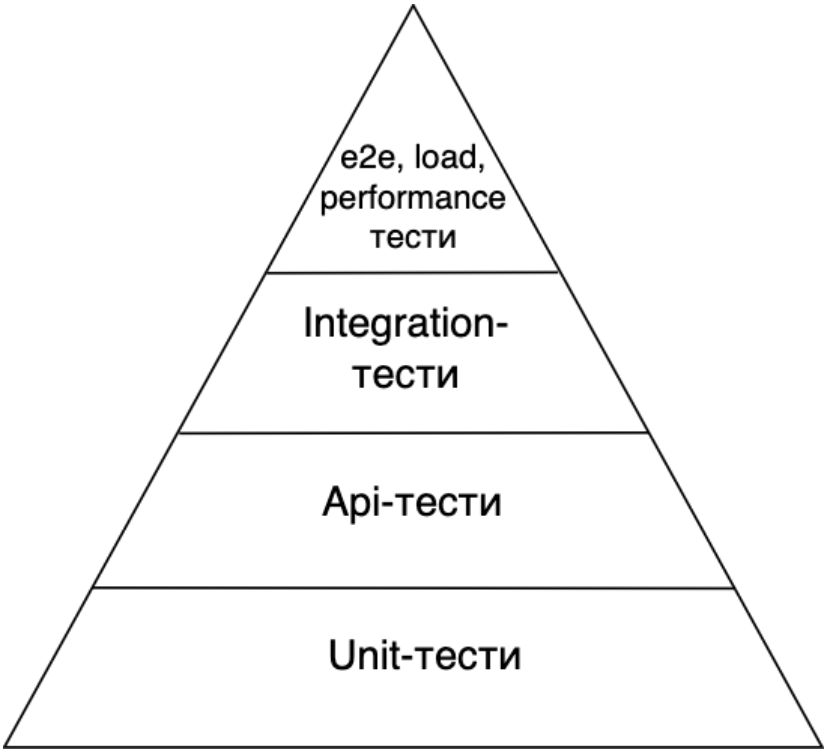


Рис. 3.15. Піраміда тестів

Основу піраміди складають Unit-тести, адже їхня кількість найбільша. Ці тести пишуть розробники зазвичай під час основної розробки коду. Вони потрібні для того, щоб структурувати та збільшити читабельність основного коду, а також завчасно відловлювати функціональні баги. Для початку пропонується зробити покриття unit-тестами на рівні 75%, а далі збільшувати, якщо будуть вспливати помилки.

Api-тести перебувають в середньому шарі піраміди, тобто їхня кількість менша за кількість unit-тестів, але більша за кількість тестів верхньої частини. Ці тести перевіряють функціональність на рівні архітектурного модуля (в даному випадку сервера). Пропонується їх розробити за технологією snapshot.

Тобто отриманий результат виконання тесту буде автоматично звірятися з попередньо згенерованим знімком, який відображає бажаний результат тесту. При виявленні розбіжності тест буде маркуватися як неуспішний та потребуватиме уваги розробника. Api, як і unit-тести, будуть відповідальністю розробників. Рівень планового покриття цим видом тестів - 100%.

Кількість інтеграційних тестів ще менша. Їхня роль полягає у тому, щоб впевнитись у цілісності і злагодженості всієї системи. Дана інформаційна система складається з декількох модулів, тому вкрай важливо контролювати їхню злагоджену роботу. Розробка та впровадження цих тестів покладена на тестувальників-автоматизаторів.

E2e-тести вже є складними та багатокроковими, виконання яких займає доволі багато часу та одночасно торкається багатьох компонентів системи. Через це не має сенсу впроваджувати багато таких тестів. Зазвичай ці тести покривають основні сценарії user-journey. Ці тести також розробляються та підтримуються тестувальниками-автоматизаторами.

Також варто зазначити load та performance тести. Вони покликані перевіряти та звіряти з бенчмарками показники навантаження та швидкодії системи. Зазвичай таку перевірку роблять мануальні тестувальники за допомогою інструменту JMetr.

У системі контролю якості передбачено також мануальне тестування. Тестувальники роблять feature-тести та загальні обширні тести наприкінці кожної ітерації. Для проведення передрелізного тестування заздалегідь складаються тест-плани, слідуючи яким тестувальники можуть відтестувати продукт зі сталою якістю.

Отже, проєкт має багаторівневу систему контролю якості, як з точки зору автоматизованого тестування, так і мануального. Це дозволяє забезпечити високу якість програмного продукту та робить процес тестування прозорим і контрольованим.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З МОНІТОРИНГУ ЦІН

4.1 Проектування бази даних

4.1.1 Концептуальна модель

Для побудови концептуальної моделі було виділено дві сутності: Product та Statistic. У Product пропонується зберігати дані про товари отримані з стороннього API, а сутність Statistic - для зберігання обчислених статистичних даних (рис. 4.1).

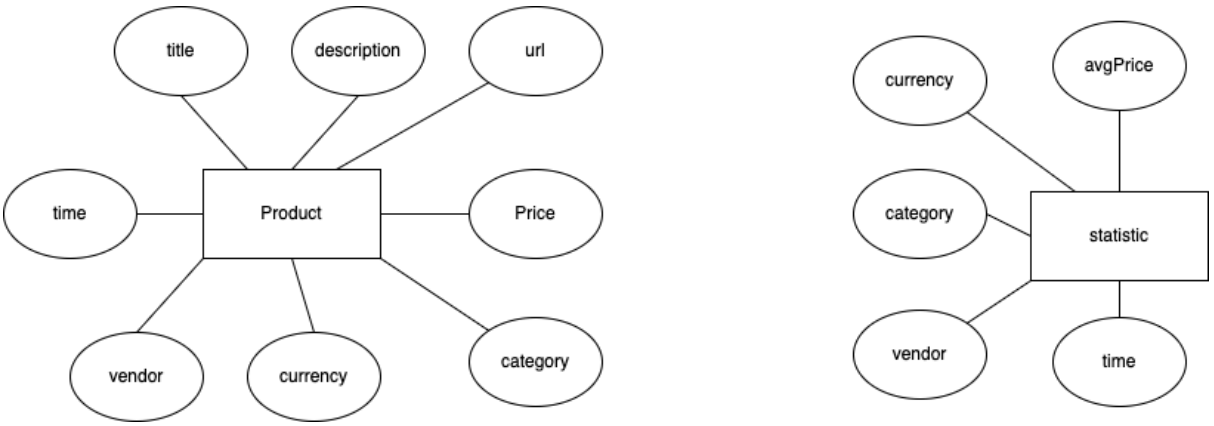


Рис. 4.1. Концептуальна модель

4.1.2 Даталогічна модель

Для проекту була обрана документоорієнтована СУБД, так як немає зв’язків між сутностями (entities), а є вбудовані елементи (embedding pattern) (рис. 4.2). За цих умов документоорієнтована СУБД показує найбільшу продуктивність, а також зручність у використанні.

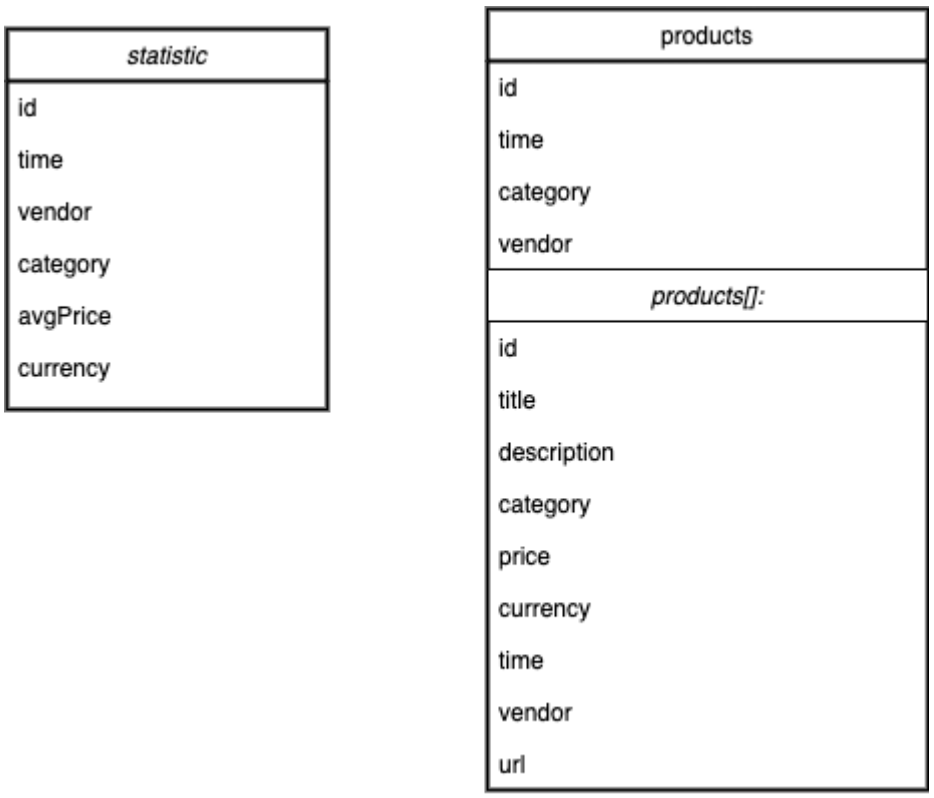


Рис. 4.2. Даталогічна модель

Систему спроектовано з використанням двох колекцій: Statistic - колекція для збереження кумулятивних статистичних даних за часовий проміжок, Products - колекція для зберігання інформації про продукти. При проектуванні схем було використано патерни Counter та Bucket [7]. Counter використовується для полегшення запитів на зчитування статистичної інформації. При записі продуктів у БД, сереверний код прорахує потрібні статистичні дані та запише їх у колекцію Statistic. Таким чином при запиті статистичної інформації, не буде необхідним перебирати мільйони документів у колекції продуктів, достатньо буде лише знайти потрібний документ у колекції Statistic.

Паттерн Bucket використовується для організації документів у колекції, таким чином зменшуючи їх кількість. Тож при запиті продуктів з БД, необхідно

буде лише знайти певний документ за певним критерієм ніж сканувати безліч документів.

4.1.3 Фізична модель

Фізична модель представлена для СУБД MongoDB (рис. 4.3). Ця СУБД була обрана, тому що вона надає зручні можливості для горизонтального масштабування (що вимагає система), а також це лідер серед документоорієнтованих СУБД, що означає її надійність та гарну підтримку з боку спільноти і провайдера СУБД. Модель не використовує joins між таблицями, що є перевагою реялійних СУБД, натомість використовується embeding, що є сильною стороною документоорієнтованих СУБД.

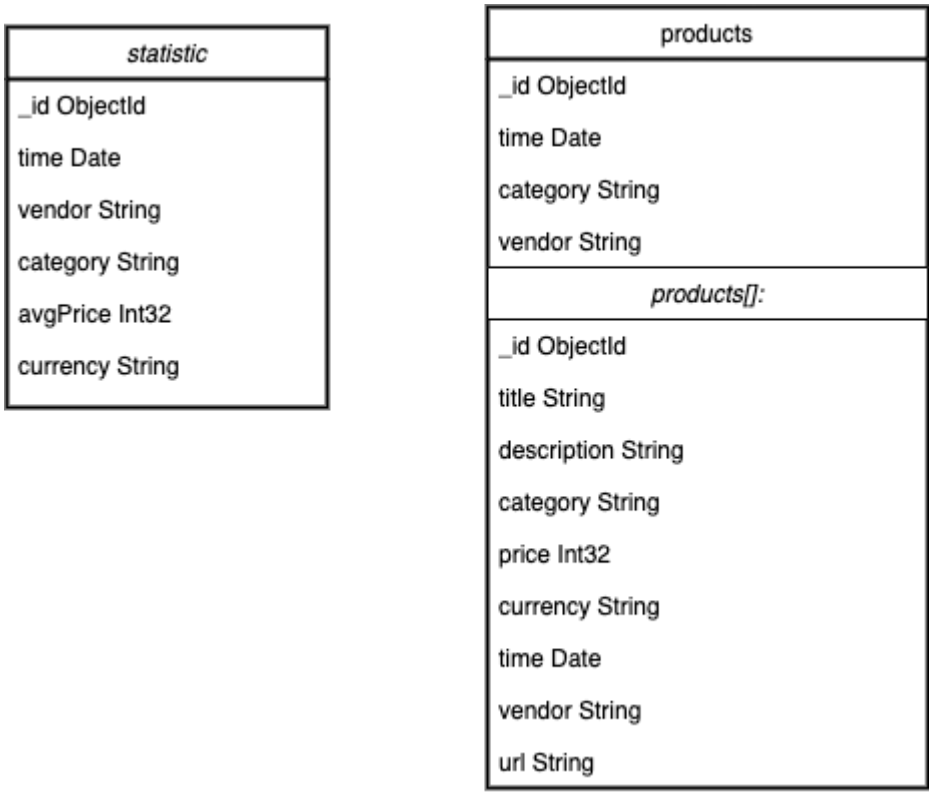


Рис. 4.3. Фізична модель

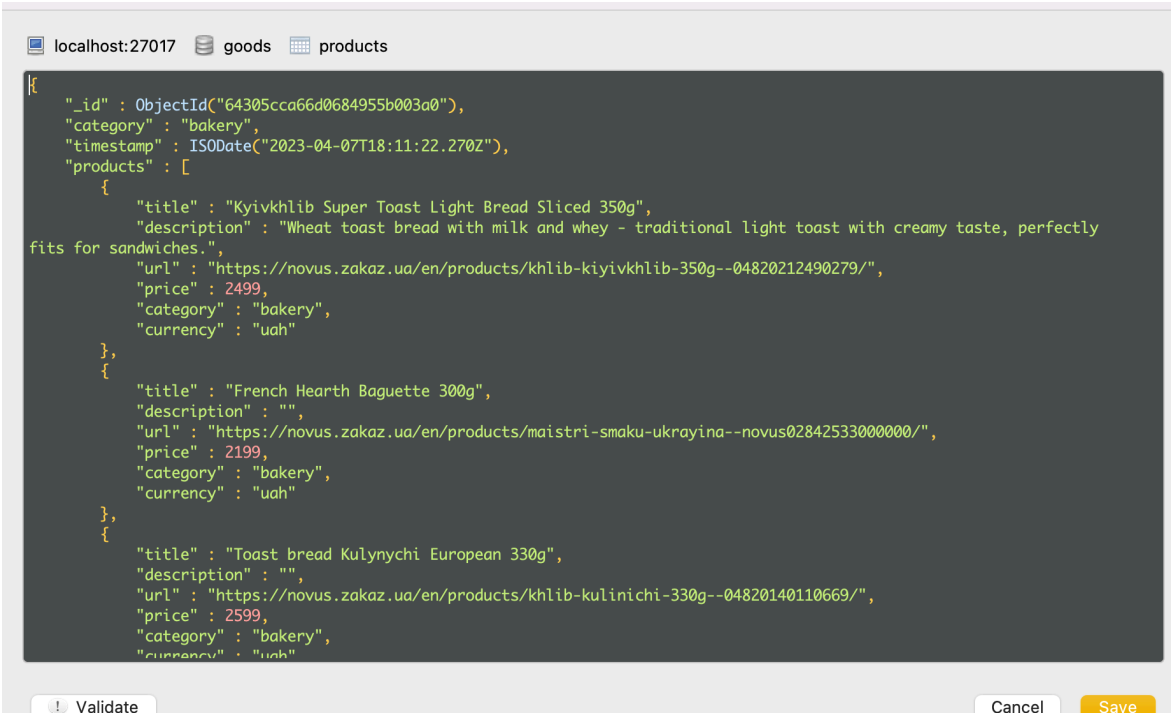


Рис. 4.4. Документ реальних даних у колекції товарів

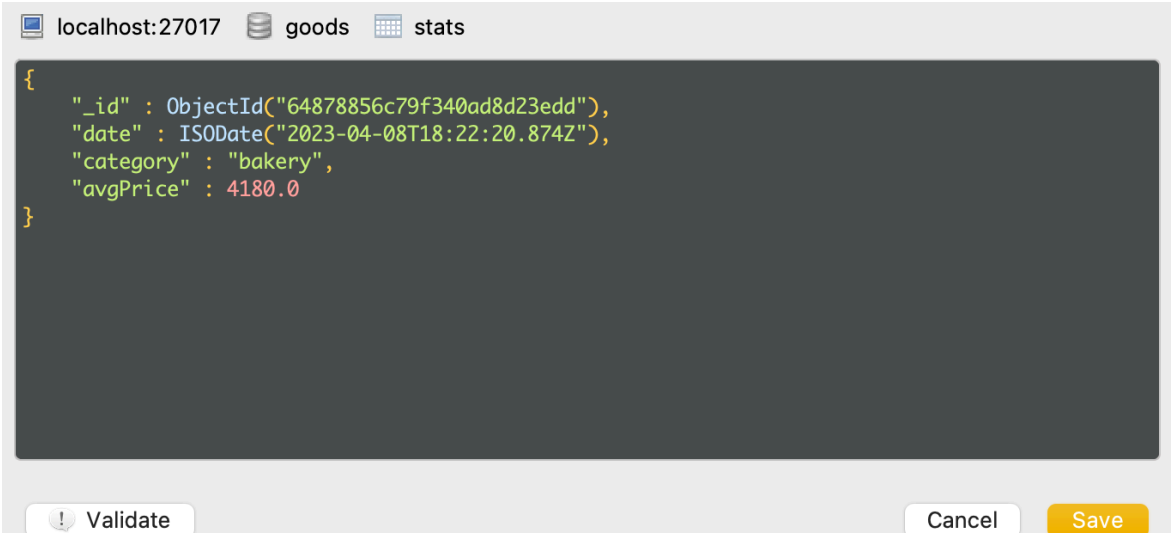


Рис. 4.5. Документ аналітичних даних у колекції статистики

4.1.4 CAP теорема

У світі проектування розподілених систем часто спираються на CAP теорему (рис. 4.6). Суть її полягає в тому, що робота інформаційної системи оцінюється за трьома критеріями:

— Consistency - відповідність даних на всіх вузлах. Тобто при записі у розподілену систему через один вузел, потім під час зчитування система повинна надавати однакові дані.

— Availability - доступність вузлів системи. Тобто роблячи будь-який запит до системи, остання повинна завжди віддавати відповідь.

— Partition-Tolerance - здатність вузла відповідати, навіть якщо втрачено зв’язок з іншими вузлами.

Теорема стверджує, що неможливо одночасно досягти одразу трьох властивостей [36]. Наприклад, якщо система зберігає відповідність даних та її вузли завжди знаходяться у працюючому стані, то при виникненні переривання зв’язку між вузлами вся система може стати недоступною або працювати з перебоями. Принцип Consistency-Availability використовують багато популярних реляційних СУБД.

У даній магістерській роботі буде розглянуто проектування системи таким чином, щоб можна було наблизити її характеристики до одразу трьох критеріїв. Для цього буде використано поєднання двох СУБД: MongoDB та DynamoDB. MongoDB забезпечує Consistency та Partition-Tolerance, а DynamoDB додає Availability.

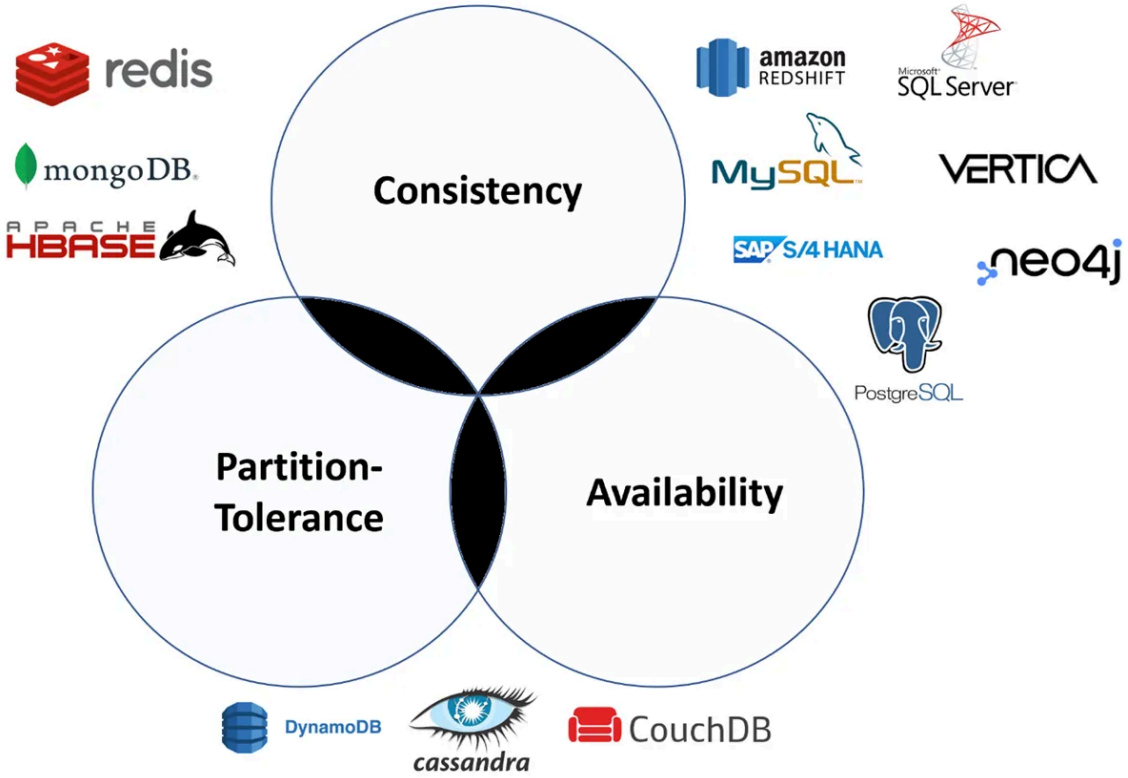


Рис. 4.6. CAP теорема

4.2 Загальний опис архітектури інформаційної системи

Архітектура системи представлена на рисунку 4.7. Вона використовує принципи мікросервісної архітектури [30]. Так можна виділити три сервіси: Node crone job, Authorization service та Public API. Всі вони розгорнуті у контейнерах з можливістю автоскейлінгу. Для збереження даних використовуються чотири бази даних: MongoDB, DynamoDB, Reddis та MySQL. Основною СУБД виступає MongoDB, саме в неї буде записуватися основна інформація з моніторингу цін. DynamoDB обрана як резервне сховище, нижче буде представлено більш детальний опис обраних технологій. MySQL база даних буде зберігати дані самих юзерів у системі, а Reddis використовується як кеш для оптимізації запитів та послаблення навантаження на основні БД. У системі також присутні serverless обчислювальні технології. Було використано AWS Lambda для генерування репортів по запиту.

Вся система “прихована” за фасадом, у ролі якого виступає API Gateway. Він валідує та перенаправляє запити на потрібні сервіси.

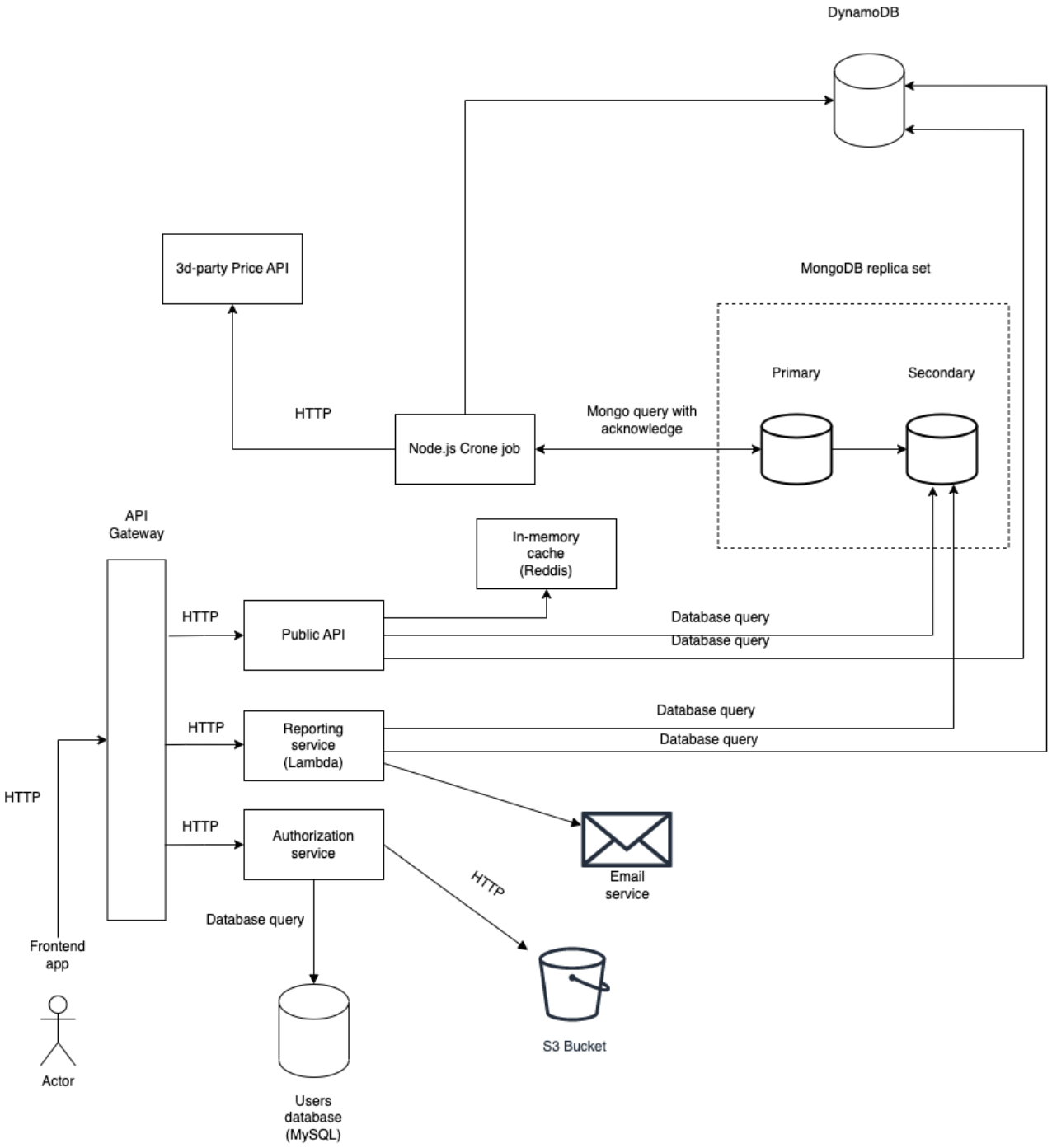


Рис. 4.7. Схема технічної архітектури ІС

Node.js Cron job раз в день опитує сторонні API з інформацією про ціни на споживчі товари, потім записує ці дані у Mongo replica set з підтвердженням

про запис тільки від Primary вузла (тобто при таких даних не обов’язково чекати про підтвердження про запис від усіх вузлів репліка сету, якщо при подальшому зчитуванні дані будуть на декілька секунд/хвилин застарілі - це не критично). На на рис. 4.6 видно, що СУБД MongoDB надає такі властивості системі як: Consistency та Partition-Tolerance. Як було сказано раніше, однією з проблем поставленою перед проєктом, було охоплення всіх трьох властивостей CAP теореми. Тому для покриття останньої властивості (Availability), було використано СУБД DynamoDb. Це NoSQL база даних запроваджена компанією Amazon яка працює тільки в AWS середовищі. Завдяки Serverless технології, ця СУБД легка в налаштуванні та здатна витримувати великі навантаження. Маючи обидві NoSQL бази даних зі схожою структурою збереження даних, можна легко імпортувати/експортувати, виправляти розбіжності та дублювати данні. Тож після запису в основну БД (Mongo replica set), використовуючи паттерн Adapter (рис. 4.8), Node.js Cron job дублює дані у резервну БД DynamoDB. Варто зазначити, що основною СУБД була обрана MongoDB, так як для інформаційної системи дуже важлива саме узгодженість даних (Consistency в CAP теоремі).

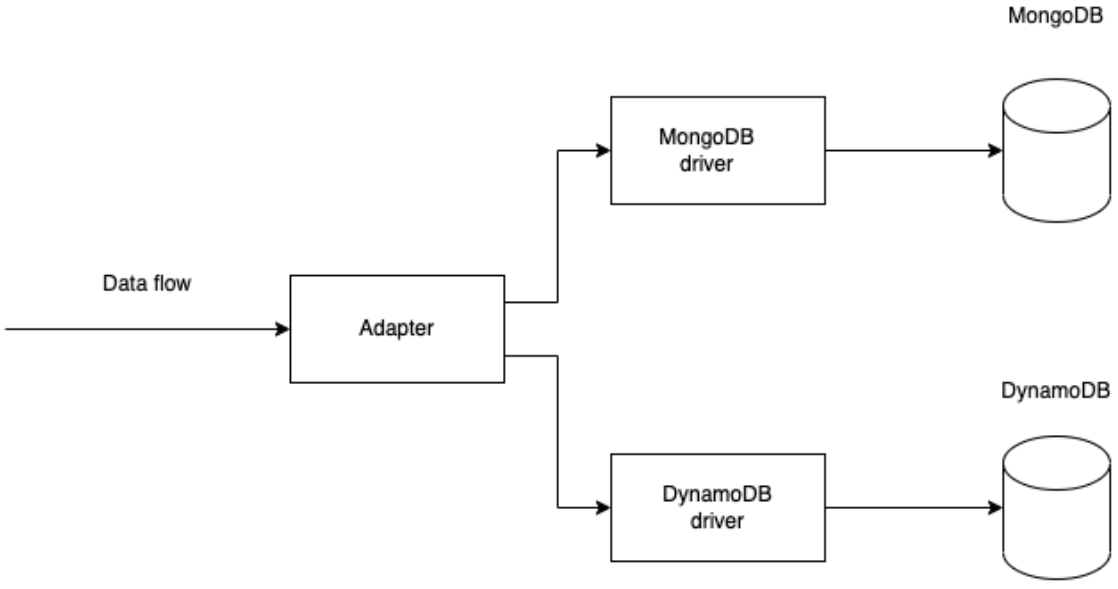


Рис. 4.8. Реалізація паттерну Adapter до двох СУБД

Користувач комунікує з системою за допомогою клієнтського web-застосунку, який також доступний у мобільній версії. Цей застосунок робить асинхронні http-запити до серверної частини. На початку запит потрапляє у API gateway, там він валідується та проходить аутентифікацію за допомогою Authorization service. Якщо юзер авторизований, то цей сервіс видає йому JWT-токен, з яким надалі клієнтський застосунок може безперешкодно комунікувати з серверною частиною. Сам Authorization service також написаний на JavaScript та працює за RESTful принципами. Його задача надати основні CRUD операції для маніпулювання юзерскими сутностями. Він має незалежну MySQL базу даних, така СУБД була обрана через те, що потенційні користувачі системи добре знаються на реляційних базах даних, а це важливо, бо іноді при збої в системі доводиться вручну маніпулювати даними. Також окрема база даних дозволяє не змішувати дані, які збирає Node.js Cron job, тобто це надає зручність у імпорті/експорті даних з основної БД. Варто ще зазначити, що винесення логіки з маніпулювання юзерами в окремий сервіс дозволяє безперешкодно виконувати операції над юзерами, навіть якщо основні сервіси лежать або працюють з затримкою. Також цей підхід додає гнучкості у масштабуванні системи.

Після успішної авторизації юзер має можливість працювати з основним функціоналом. На цьому етапі запити юзера потрапляють через API Gateway до Public API сервіса, який також працює за принципами REST. Цей сервіс надає ендпоінти для отримання даних про ціни. Він формує аналітичні запити до баз даних. Якщо система працює у штатному режимі, дані беруться з MongoDB replica set з secondary вузла, щоб зняти навантаження з Primary пишучого вузла. Але якщо виникають збої в системі, вибірка даних може переключитися на DynamoDB використовуючи той самий адаптер паттерн, що і для Node.js Cron job. За перемикання між базами даних відповідає спеціальна fallback функція, яка в залежності від таймауту та помилок мережі робить перемикання на ту чи

іншу БД. Також передбачена логіка ручного перемикання, шляхом модифікації environment variable, або налаштувань на панелі support-інженера. На рисунку 4.9 представлено послідовність процесів навколо Public API, які починаються з авторизації юзера, продовжуються отриманням аналітичних даних та закінчуються виходом з кабінету.

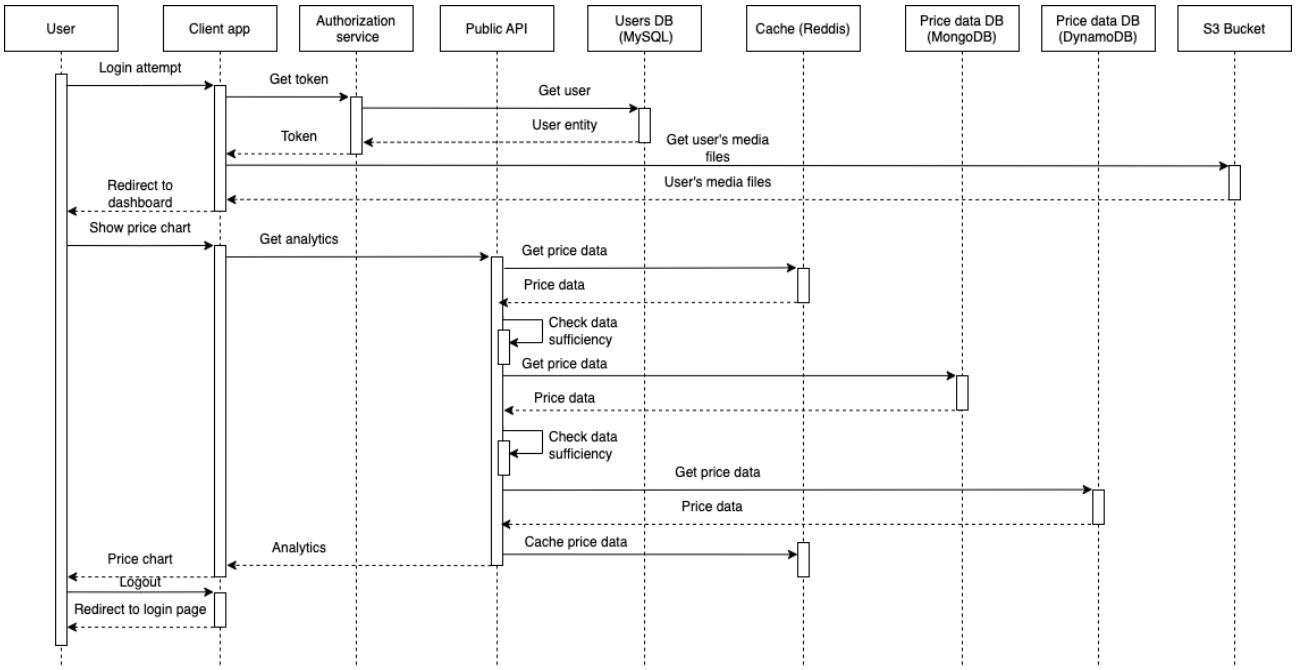


Рис. 4.9. UML-діаграма послідовності процесів отримання аналітичних даних

Між базою даних та Public API існує кеш. Його було додано для пришвидшення запитів юзера та послаблення навантаження на БД. Аналітичні запити, в залежності від складності, вимагають перегляду великих масивів даних у БД, тож це займає час та ресурси. Імплементацією кеша обрано Reddis базу даних, вона має просту структуру даних та є in-memory базою даних, що говорить про її високу швидкість. Сам механізм кешування зроблено за принципом Cache-aside. На рисунку 4.10 показано механізм роботи кешу (номерами над стрілками показано черговість операції). Як видно на схемі, спершу юзер запитує Public API, останній перевіряє чи є потрібні дані у кеші,

якщо відповідних даних не знайшлось, сервер робить запит до бази даних. Отримуючі потрібні дані з БД, сервіс записує їх у кеш та відправляє юзеру. Таким чином при наступному запиті, ті ж самі дані вже будуть братися з кеша.

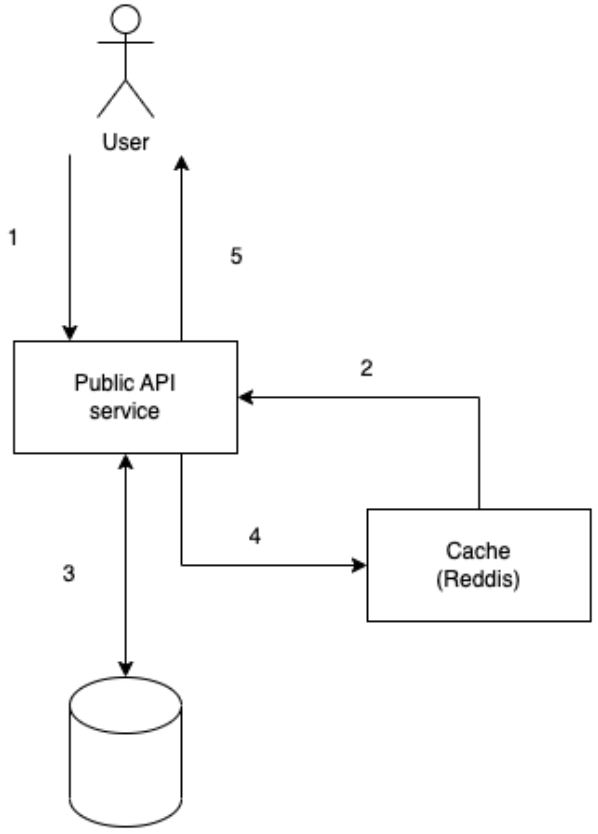


Рис. 4.10 Механізм кешування для Public API сервіса

У системі передбачено функціонал з генерування звітності. Файл звітності може бути згенеровано у таких форматах: EXCEL, CSV та PDF. Після створення відповідного документа, він може бути відправлений на електронну пошту або завантажений у браузері. Було вирішено винести цей функціонал з основного сервісу у Lambda function. Це serverless технологія в AWS середовищі, яка надає обчислювальні потужності по запиту. Тобто ця функція активується, в даному випадку, на HTTP-запит і існує деякий час, доки процес з генерації не буде завершено. Переваги такого підходу в тому, що плата знімається тільки за час виконання функції, а також вона дуже швидко

масштабується при зміні навантаження. Зазвичай звіти складають раз на день/тиждень/місяць, тобто це доволі рідкісна операція і тримати під неї весь час цілий контейнер та систему масштабування, може бути доволі затратно.

На рисунку 4.11 представлена файлова структура проєкту. Для побудови використали принцип монорепозиторія, який має на увазі зберігання всіх компонентів системи в одному репозиторії.

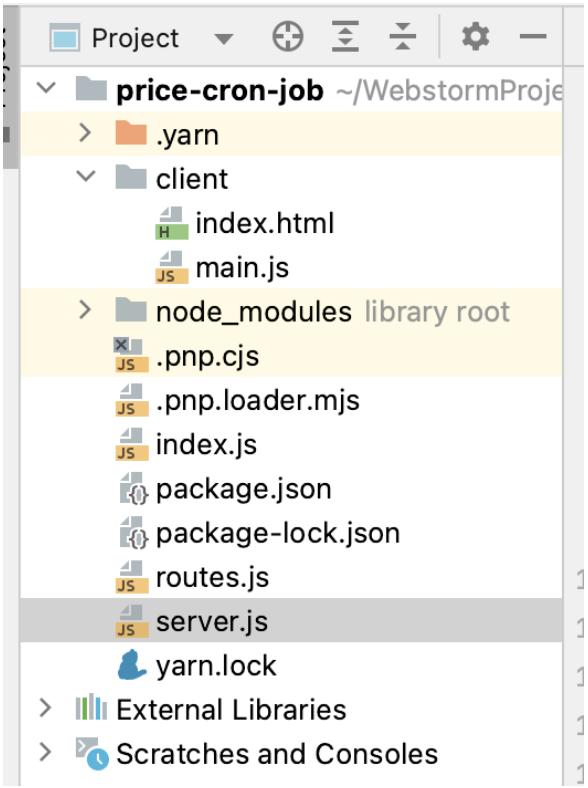


Рис. 4.11. Організація файлової структури проєкту

4.3 Сервер збору продуктових даних

Даний сервер опитує стороні API, які містять інформацію про продукти. Для цього був обран API сервісу zakaz.ua, він містить гарно структуровану інформацію про продукти та найпоширеніші торгові мережі країни. Також цей API публічний і безкоштовний.

Щоб зібрати необхідну інформацію, серверу доводиться робити сотні паралельних запитів. Для вирішення цієї задачі гарно підходить Event-driven модель. Тобто запускаючи події паралельно, які в свою чергу будуть оброблені по мірі їх завершеності, ми виграємо в продуктивності. Для технічної реалізації була обрана Node.js, яка побудована на Event-driven підході.

Збір даних відбувається раз на день, оскільки з точки зору дизайну системи, така свіжість даних є достатньою. Для реалізації цього, був використаний пакет cron-node, застосування якого дозволяє налагодити запуск коду опитування раз на день.

Після отримання даних з API, щоб не перевантажити пам’ять сервера, вони одразу ж фільтруються для виокремлення потрібної інформації та зберігаються у базу даних. На рисунку 4.12 представлено UML sequence діаграму, яка показує послідовність процесів, які відбуваються у даному сервісі. На діаграмі видно, що ініціює послідовність процесів Cron job, яка запускає процеси опитування сторонніх продуктових API та після обробки отриманих даних зберігає їх у дві бази даних. Ці процеси протікають циклічно і завершаться тільки тоді, коли останній в черзі API буде опитаним.

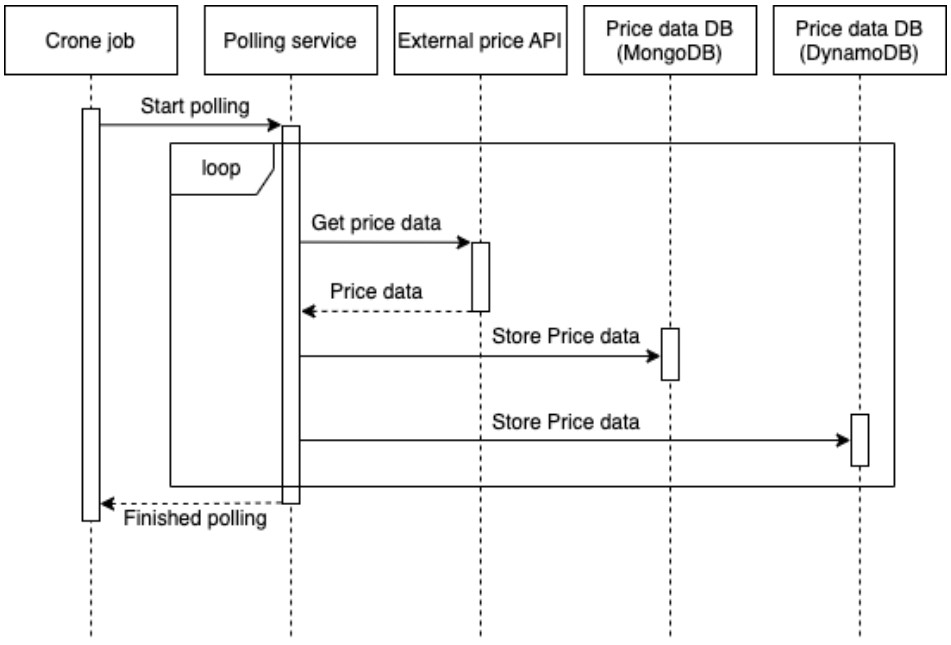


Рис. 4.12. UML-діаграма послідовності процесів для Public api сервісу

Записуючи дані до Mongo DB replica set, сервер отримує підтвердження тільки від Primary вузла. Оскільки допускається застарілість даних в декілька секунд/хвилин, то не обов'язково чекати про запис який завершиться на всіх вузлах, а для пришвидшення достатньо отримати підтвердження лише з першого.

Нижче представлено фрагмент основного коду, який нарізає запити на частини та відправляє запрос на API zakaz.ua. Отримані дані обробляються та зберігаються у БД.

```
const axios = require('axios');
const { MongoClient } = require('mongodb');

const PRODUCTS_API_BASE_URL = `https://stores-api.zakaz.ua`;
async function getProductsByCategoryAndStoreId(category,
storeId) {
                                const url =
`${PRODUCTS_API_BASE_URL}/stores/${storeId}/categories/${cate
gory}/products`
    const response = await axios.get(url);

    const pagesCount = Math.ceil(response.data.count/30);
    const pages = Array.apply(null, Array(pagesCount));

    const requests = pages.map(async (_, indx) => {
        return await axios.get(url + `?page=${++indx}` )
    });
    const rawProducts = await Promise.all(requests);

    return rawProducts.reduce((result, response) => {
        const responseResults = response.data.results;
```

```
            return [...result, ...responseResults.map(({ title,
description, web_url, price, parent_category_id, category_id,
currency }) => ({
                title, description, url: web_url, price, category:
parent_category_id || category_id, currency
            }))];
    }, []);
}

async function getMongoDB() {
                        const        client        =        new
MongoClient('mongodb://localhost:27017/goods');

    try {
        // Connect to the MongoDB cluster
        await client.connect();
    } catch (e) {
        console.error(e);
    }

    return client.db();
}

async function saveProducts(category, products) {
    const date = new Date();
        await   this.productsCollection.insertOne({   category,
timestamp: date, products });
}

async function saveAvgPrice(category, products, sourceId) {
```

```
    const date = new Date();

     const avgPrice = Math.round(products.reduce((acc, val) =>
acc + val.price, 0) / products.length);

    await this.statsCollection.insertOne({ avgPrice, category,
date, source: sourceId });
}

async function init() {
    const db = await getMongoDB();
                    const       products       =       await
getProductsByCategoryAndStoreId('bakery', '48201070');

      const  saveProductsResult  =  await  saveProducts.call({
productsCollection:  db.collection('products')  },  'bakery',
products);
             await    saveAvgPrice.call({    statsCollection:
db.collection('stats')       },       'bakery',       products,
saveProductsResult.insertedId );
}

init();
```

4.4 Сервер побудови аналітичних запитів та провадження API для юзера

Цей сервер також побудований на базі event-driven моделі з використанням Node. js. Оскільки даний сервер використовується для обробки запитів з UI, то на передній план виходить вимога щодо швидкого реагування та неблокуючого I/O сервера.

За основу було взято REST архітектурні принципи. Цей підхід широко використовується для побудови публічних API та достатньо підходить для подібних запитів з UI.

Опис HTTP запитів:

Request:

Method: GET

Path: /api/products

Query params: category, vendor, startDate, endDate

Response:

Body:

```
[
    {
        id: string,
        category: string,
        vendor: string,
        time: string,
        title: string
        description: string,
        price: number,
        currency: string,
        url: string
    },
    …
]
```

Request:

Method: GET

Path: /api/stats

Query params: startDate, endDate, vendor, category, operation

Response:

Body:

```
[
    {
        id: string,
        category: string,
        vendor: string,
        time: string,
        value: number,
        currency: string
    }
]
```

Після отримання запиту зі сторони клієнта, сервер формує запит до БД з метою отримання аналітичної інформації. Запити надсилаються на Secondary вузол репліка сету, таким чином вдається зняти навантаження з Primary вузла, який працює на запис даних.

Сервер запроваджує основні два HTTP ендпоінти, один застосовується для вибірки самих продуктів без аналітичних даних (клієнт таким чином може сам сформувати аналітику), та ендпоінт з аналітичними даними, які вже сформовані цим сервером.

```
const { MongoClient } = require('mongodb');
const express = require('express')
const cors = require("cors");
const app = express()
const port = 3000;
async function getMongoDB() {
```

```
                        const client = new
MongoClient('mongodb://127.0.0.1:27017/goods');

    try {
        // Connect to the MongoDB cluster
        await client.connect();
    } catch (e) {
        console.error(e);
    }

    return client.db();
}

function initRouter(db) {
    console.log(db)
    const router = new express.Router();

    router.get('/api/products/', async (req, res) => {
            const { category, vendor, startDate, endDate } =
req.query;

        const date = new Date(startDate);
        const date1 = new Date(endDate);

        console.log(date, date1)
                        const productDocuments = await
db.collection('products').find({
            //time: {$gte: date, $lte: date1},
            category,
            // vendor
        }).toArray();
```

```
            const products = productDocuments.reduce((acc,
productDocument) => [...acc, ...productDocument.products],
[]);

        return res.send(products)
    });

    router.get('/api/stats/', async (req, res) => {
            const { category, vendor, startDate, endDate } =
req.query;

        const date = new Date(startDate);
        const date1 = new Date(endDate);

        const stats = await db.collection('stats').find({
            //time: {$gte: date, $lte: date1},
            category,
            // vendor
        }).toArray();

        return res.send(stats)
    });

    return router;
}

async function init() {
    const db = await getMongoDB();
    const routes = initRouter(db);
```

```
    app.use(cors({origin: '*'}))

    app.use(routes);

    app.listen(port);
}

init();
```

4.5 Браузерний застосунок для репрезентації даних

Браузерний застосунок побудований з використанням бібліотеки Chart.js для візуалізації аналітичних даних у вигляді графіків. Застосунок надсилає запити на сервер описаний в розділі 4.4, та відображає отримані дані в залежності від категорії продукту, торгової мережі, або в цілому за ринком.

Нижче наведено фрагмент коду з браузерного застосунку:

```
async function getProducts() {
                                                    return
fetch('http://localhost:3000/api/stats?category=bakery')
}

(async function() {
    const response = await getProducts();
    const data = await response.json()
    console.log(data)

    new Chart(
        document.getElementById('price_chart'),
        {
            type: 'bar',
```

```
            data: {
                labels: data.map(row => row.date),
                datasets: [
                    {
                        label: 'Average price',
                        data: data.map(row => row.avgPrice)
                    }
                ]
            }
        }
    );
})();
```

На рисунку 4.13 показано screenshot результату роботи системи. Представлено графік моніторингу товарів групи хлібобулочних виробів. Як видно, середня ціна коливається в діапазоні 40-45 грн.

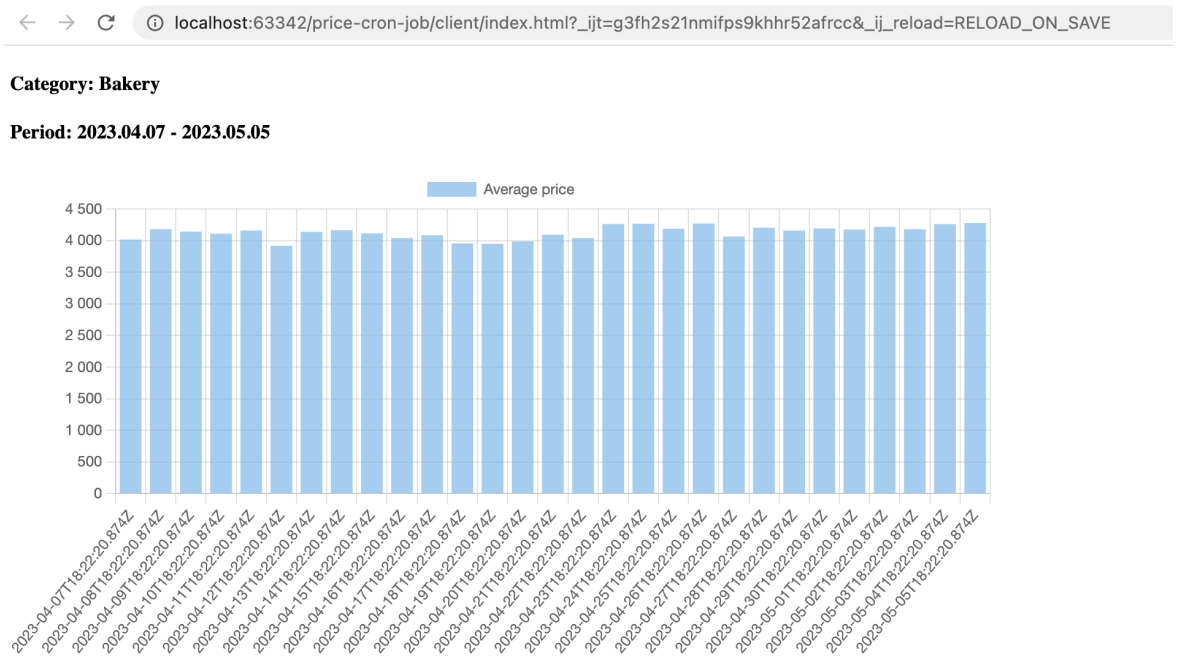


Рис. 4.13. Презентація аналітичних даних у браузерному застосунку

4.6 Розгортання в хмарі

За провайдера хмарних послуг, було взято Amazon Web Services [8]. Оскільки даний провайдер славиться своєю надійністю, гнучкістю рішень та популярністю. Через це не мають виникнути проблеми з підтримкою рішення. Для розгортання серверів взято продукт EC2, цей продукт дозволяє утримувати сервер у заздалегідь зібраному контейнері, масштабувати та відстежувати навантаження/логи/збої. Образи для контейнерів зберігаються у ще одному продукту від AWS - Elastic Container Registry. Він дозволяє проводити версіонування образів та швидко збирати контейнер для EC2.

На рисунку 4.14 представлено зв'язки та компоненти AWS середовища для обробки клієнтських запитів. На малюнку показано межі AWS середовища, як видно, майже всі сервіси, орім MongoDB, знаходяться в мережі AWS. Гарною практикою вважається розподіляти обчислювальні потужності за декількома Availability Zones, так як у разі, наприклад, стихійного лиха, дата центри в одній AZ можуть бути виведені з ладу, тому потрібні резервні потужності. Availability zone - це мережа дата центрів, які пов'язані між собою швидкісними каналами та розташовані неподалік один від одного, в межах одного регіону. В свою чергу, регіон в AWS може містити декілька AZ. У даному випадку використовується дві AZ в одному регіоні. Обрано регіон eu-central-2, потужності якого розташовані в Цюріху, це достатньо близько до нашої країни, тож швидкість з'єднання між сервером та потенційними користувачами повинна бути високою.

В кожній AZ було створено публічну та приватну мережу. Публічна мережа зазвичай містить обчислювальні ресурси, які обробляють запит з інтернету, тому у даному випадку ця мережа містить EC2 інстанси для сервісу Public API та Authorization service. Ці інстанси містять контейнери з операційною системою та програмним сервером. Приватна мережа містить інстанси Amazon RDS, на базі якого працює MySQL сервер з базою даних

юзерів для Authorization service. Ця мережа має обмежений доступ, відкрита тільки для з'єднань з Authorization service. Це зроблено для того, щоб унеможливити зловмисне втручання з інтернету.

Для EC2 інстансів налаштоване масштабування, вони знаходяться у так званій Auto Scaling Group. Для цієї групи налаштовані правила за якими буде автоматично розгорнуто або виключено один/декілька інстансів. Це дозволить системі автоматично підлаштовуватися під зміну навантаження. Як видно на малюнку, запроваджено дві Auto Scaling Group для Public API та Authorization service. Також обов'язковою практикою вважається балансування потужностей. Застосовано AWS Load balancer для кожної Auto Scaling Group. Load balancer використовує математичні алгоритми для розподілення навантаження між EC2 інстансами. Тобто запит юзера проходячи через API Gateway, в залежності від path url потрапляє у релевантний Load balancer, а далі останній вже вирішує який саме інстанс обробить цей запит. Відповідь йде зворотним шляхом EC2 -> Load balancer -> API Gateway -> User.

Підсумовуючи варто перелічити основні інфраструктурні компоненти:

— Загальна мережа.
— Availability zones.
— Subnets.
— EC2.
— Amazon RDS.
— DynamoDB.
— AWS Lambda.
— AWS S3.
— Atlas Cluster.
— ElastiCache.
— Amazon API Gateway.

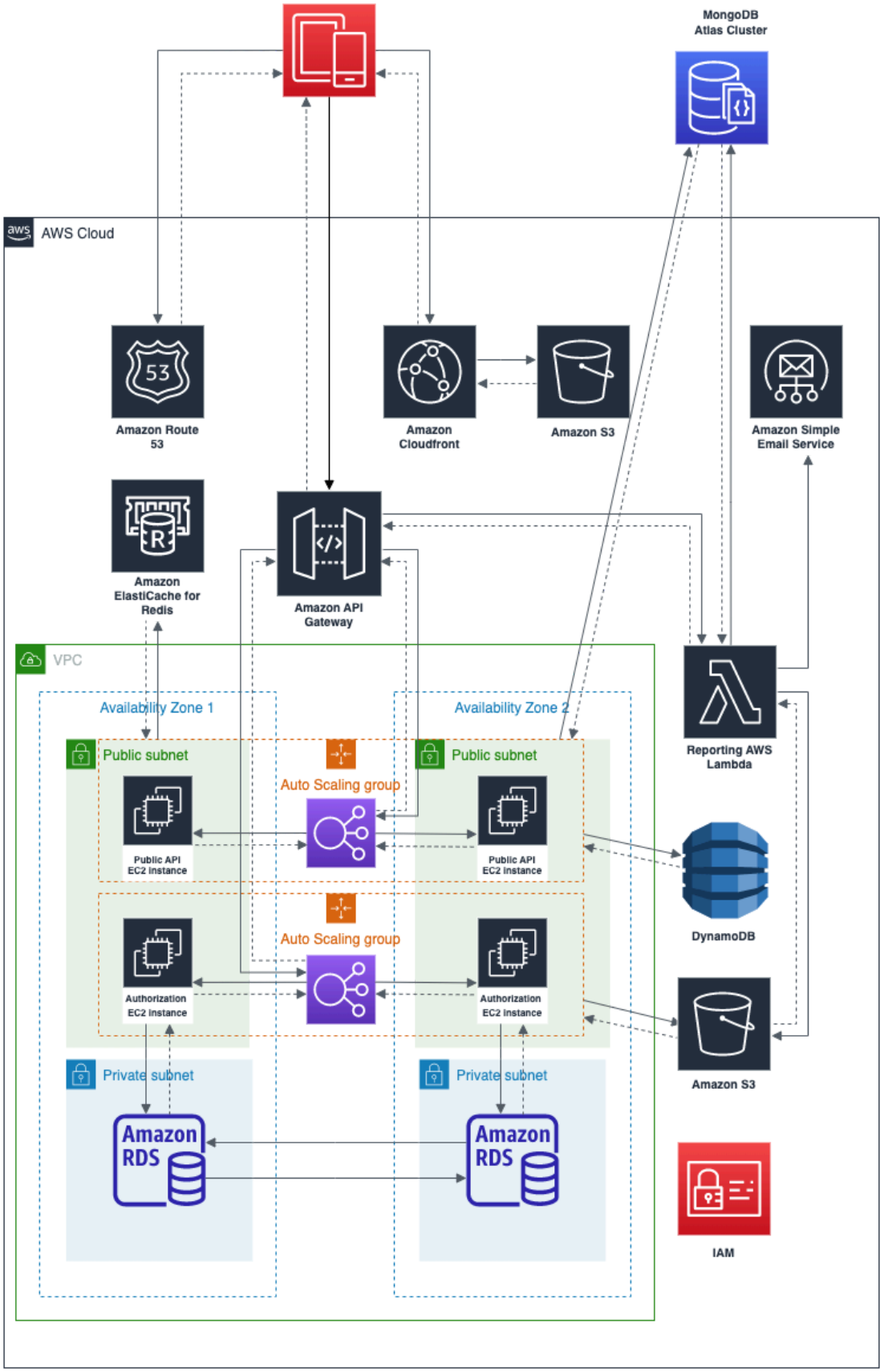


Рис. 4.14. Public API в середовищі AWS

Public API приймаючи запит спочатку перевіряє чи є потрібна інформація в кеші. Для кешу використовується Serverless технологія Amazon ElastiCache з варіацією для Reddis БД. Цей сервіс гарантує високу швидкість, надійність та масштабованість кешу. Після перевірки кешу, якщо потрібна інформація не знайдена, Public API запитує MongoDB Atlas Cluster. Цей кластер знаходиться поза AWS середовищем, тобто запит йде через інтернет, але у зашифрованому вигляді. Atlas [9] запроваджує всі функції для адміністрування, масштабування, відстежування, а також всі опції що вимагає MongoDB. Як зазначалося раніше, була розгорнута конфігурація репліка сету з двома вузлами. На перший направляються пишучі запити, а на другий тільки запити зчитування (рис. 4.15).

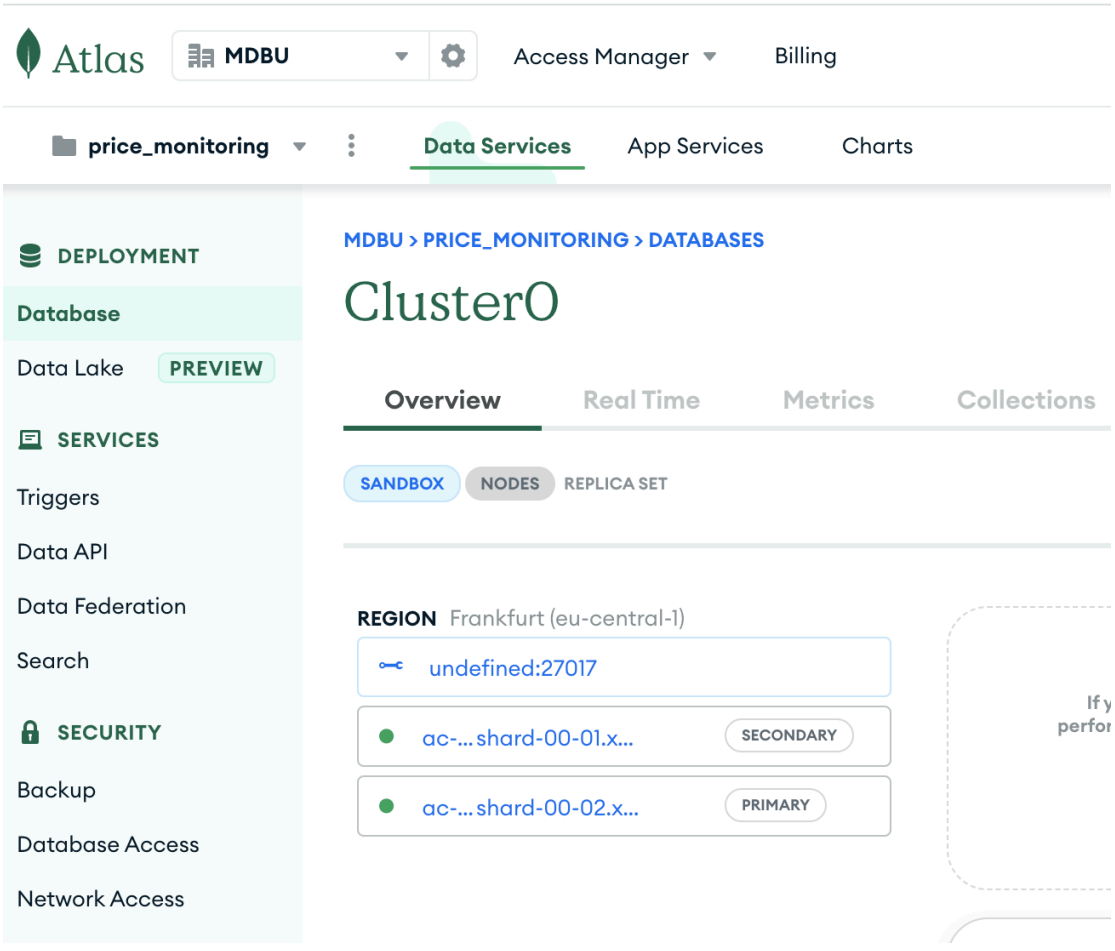


Рис. 4.15. MongoDB replica set розгорнутий в Atlas

При виникненні проблем з Atlas кластером, це можуть бути проблеми зі з’єднанням, неповнота даних, відмова кластеру тощо; Public API може звернутися до DynamoDB. Ця база даних, як і ElastiCache, базується на serverless технології, тобто інженеру не потрібно думати про апаратну та мережеву складову, клауд-провайдер самостійно забезпечить надійну безпеку, масштабування, апдейти пакетів ПЗ тощо. З мінусів такого рішення можна виділити високу ціну (але треба рахувати для кожного окремого кейсу) та слабку контрольованість ресурсів з боку інженера. Клауд-провайдер може розподіляти інформацію з serverless рішення не обмежуючись тільки певними AZ/Regions. Тобто фізично дані можуть потрапити до країни де їх знаходження буде небажаним з боку власника. Але дана інформаційна система не має дуже чутливих даних, тому враховуючи ціну, можна обмежено користуватись даною БД.

В проєкті існує ще декілька serverless технологій, одна з них - AWS Lambda. Це один з видів обчислювальних потужностей в AWS, її задача проводити певні обчислення після спрацювання тригеру. Цим тригером може бути якась подія у S3 Bucket, HTTP-запит, повідомлення з мессенджеру (SNS) тощо. Лямбда функція запускається за певним триггером, проводить обчислення, зазвичай короткотривалий час, та надає результат. Вона запускається тільки після спрацювання триггеру та зникає після завершення обчислення. AWS може запускати безліч функцій в залежності від навантаження, а користувач сплачує тільки за час виконання. Тому такий підхід доречний коли ми точно не можемо спрогнозувати частоту спрацювання тригерів. Тож цю технологію використано для генерування звітності. Юзер через клієнтський інтерфейс задає певні параметри звіту, далі цей запит проходить через API Gateway, який перенаправляє реквест до Lambda функції. Функція робить запит у базу даних та створює звіт потрібного формату, далі зберігає його в S3 Bucket або надсилає емейл через AWS Simple Email Service. У

відповідь на HTTP-запит, функція надсилає лінку для завантаження згенерованого звіту з S3 Bucket або відправляє повідомлення, що звіт надійде юзеру на пошту.

AWS S3 Bucket також є serverless технологією, яка працює за патерном Object storage, тобто клієнт не може модифікувати частину файлу, йому потрібно замінити весь файл у такому разі. AWS гарантує, що збережені файли не пропадуть та не пошкодяться. Це досягається за рахунок redundancy, тобто відбувається надлишкове резервне копіювання. Провайдер пропонує декілька класів, які можна обрати для бакету. Вони переважно відрізняються швидкістю сховища, кількістю запитів до сховища та ціною. Для даного проєкту доцільно використати стандартний клас, так як бакет використовується для збереження юзерських файлів, наприклад, аватарок, та для зберігання звітів.

Розглядаючи користувацький запит з самого початку, варто більш детально торкнутись системи розгортання статичного клієнтського застосунку. Після того, як користувач ввів DNS ім’я веб-сайту у баразузерну строку, це ім’я резолвиться на DNS сервері та потрапляє до AWS Route53. Цей сервіс потрібен для поєднання юзерського запиту з застосунками, які працюють у AWS середовищі. Він вважається надійним та відмовостійким. Далі Route53 вертає IP-адрес, який прив’язаний до запитуємого доменного ім’я. Тепер браузер вже має адрес куди слати запити. Запитуючи отриманий IP-адрес, запит потрапляє до AWS CloudFront - ще один servelsess сервіс, який часто використовується для клієнтських застосунків. Він дозволяє зменшити час завантаження сайту, така швидкість забезпечується за рахунок того, що сервіс розгорнутий у широкій мережі по всьому світу, що забезпечує близькість до кінцевого користувача. Також він може кешувати статичні файли, що також пришвидшує відгук сайту. З боку безпекового аспекту, CloudFront дозволяє захиститись від DDoS-атак. CloudFront працює у зв’язці з S3 Bucket, який спеціально настроєний у режим обслуговування статичних файлів веб-сайту. Тож кінцева html-сторінка або

JavaScript код надходить до браузеру саме з S3 Bucket. Тобто оновлюючи веб-застосунок, ми повинні завантажити його у S3 Bucket.

Node.js crone job

Як зазначалося раніше, у системі присутній ще один центральний вузол з обробки інформації. Це node.js cron job, яка опитує сторонні продуктові API. Його топологія показана на рисунку 4.16.

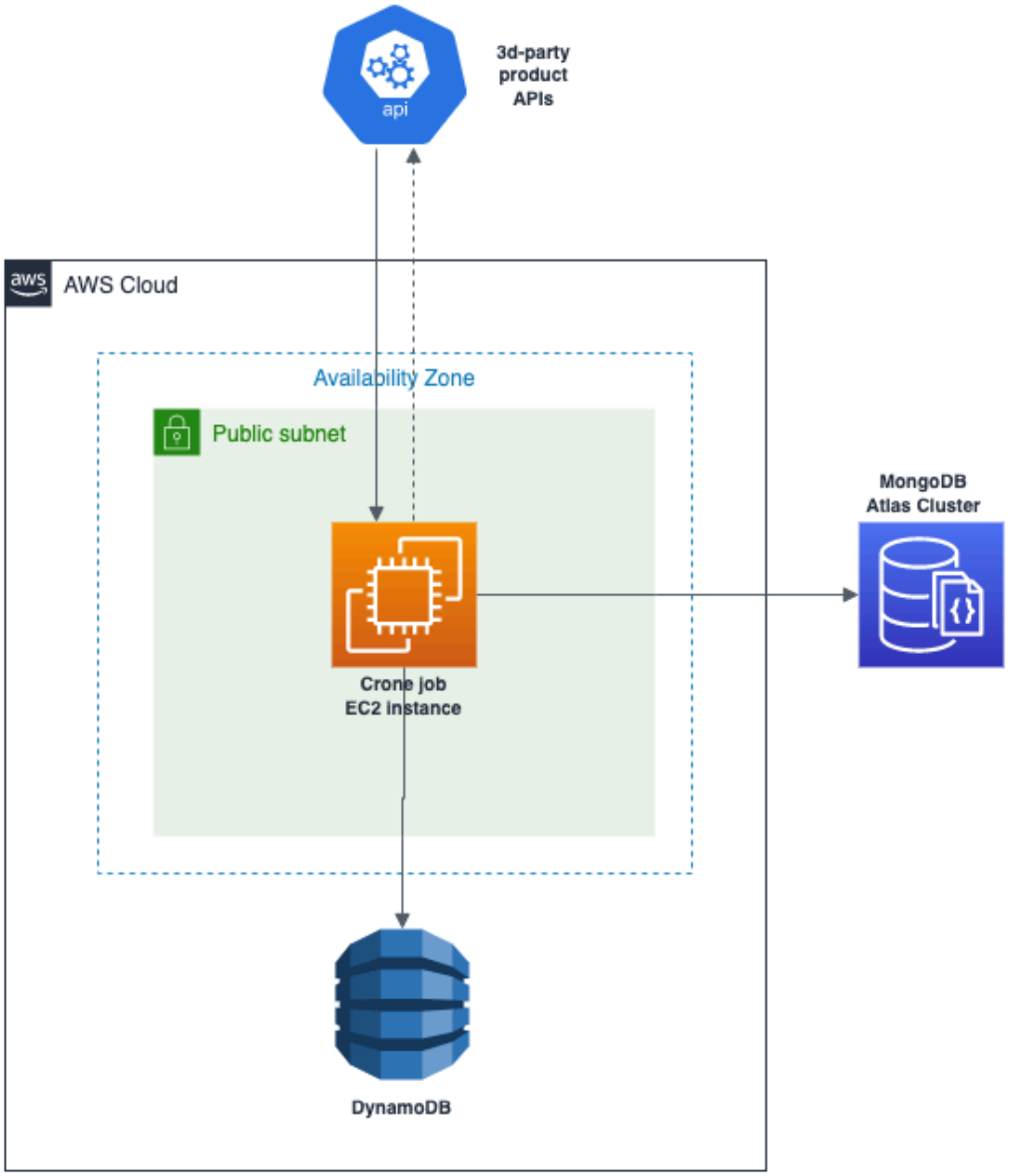


Рис. 4.16. AWS оточення для Node.js cron job

Цей сервіс також розгорнуто на базі EC2 інстанса з використанням контейнеризації. Він знаходиться в одній Availability zone та в публічній мережі. У даному випадку вже не потрібне автоматичне масштабування (Auto scaling group), як це було імплементовано для Public API, оскільки навантаженість інстансу прогнозоване. Тобто в налаштуваннях сервісу заздалегідь задається кількість опитуваних API та прогнозується навантаження на сервіс. Якщо навантаження перевищує ресурси одного інстанса, то можна заздалегедь розгорнути ще один. Як видно на рис. 4.16, сервіс робить запит за межі AWS середовища з метою опитування продуктових API. Також для збереження отриманих даних використовується запит до Atlas Cluster через інтернет. Після успішного збереження, сервіс дублює дані у DynamoDb, яка вже знаходиться у мережі AWS.

Моніторинг всієї інфраструктури відбувається за допомогою AWS CloudWatch. Цей продукт дозволяє відслідковувати логи та навантаження. Для сервісів встановлені допустимі порогові значення, при перевищені яких, відправляється повідомлення на пошту. Наприклад, якщо навантаження на EC2 instance Public API перевищує 60%, то відправляється лист на пошту та Auto scaling піднімає новий інстанс у групі. У випадку зниження навантаження - зайвий інстанс виводиться з роботи і також відправляється повідомлення про виконану операцію. Усі помилки гарно видно у CloudWatch консолі, кожен сервіс, навіть AWS Lambda, відправляють туди свої логи, тому не виникає труднощів при дебагінгу того чи іншого сервісу.

ВИСНОВКИ

Дослідивши предметну галузь інформаційних систем з автоматичного моніторингу цін, можна підсумувати, що в теперішній час IT технологій, коли все автоматизовано та диджиталізовано, автоматичний моніторинг цін стає вже не чимось на вістрі технологій, а інструментом, який обов'язково повинна мати кожна компанія, яка володіє широким асортиментом товарів та веде цінові війни з конкурентами. Адже висока насиченість ринків конкурентами спонукає компанії серйозніше ставитись до політики ціноутворення задля утримання своїх позицій в галузі. До того ж макроекономічна нестабільність, особливо валютних ринків, стимулює попит на подібні системи. Органи державного регулювання також зацікавлені в отриманні якісної і своєчасної інформації про динаміку цін на групи товарів, оскільки це дає змогу вчасно приймати рішення стосовно регуляції цін або прожиткового мінімуму.

Дослідження ринку інформаційних систем з моніторингу цін показало, що на ринку вже є гравці з багаторічним досвідом, це свідчить про те, що попит на дані системи є. Аналіз конкурентів показав, що представлені на ринку системи не позбавлені недоліків, що робить реалістичним вивід нового продукту на ринок. До того ж дослідження об'єму ринку підтвердило чималий платоспроможний попит, так показник SOM сягає 530 млн. грн. Проведені SWOT та STEP аналізи надали обґрунтування можливості реалізації проєкту.

Наступним етапом була побудована концепція проєкту. Інформаційну систему було розглянуто з точки зору системного аналізу. Були визначені контрольні віхи, так проєкт має тривалість 8 місяців та дату завершення 26.11.24. Також були описані персони користувачів, визначені вимоги до продукту та складений паспорт проєкту. Дослідження економічної моделі

показало, що проєкт є життєздатним - термін окупності складає 7,7 місяця, а NPV дорівнює 493 тис. грн.

Однією з задач було дослідження та моделювання оптимізації сховища даних, оскільки система дуже чутлива до швидкодії сховища. Експериментальним шляхом було виявлено, що без методів оптимізації, вже на другий місяць збору даних система перевищить допустимі показники затримки відповіді. Тож експериментально було перевірено ефективність патернів оптимізації сховища та надано математичні моделі для подальшого прогнозування масштабування.

Згідно з проведеного дослідження організаційної структури при розробці розподілених систем де використовується підхід Domain Driven Design, було виявлено, що дана організація розробки має чимало недоліків, серед яких: високий рівень непередбачуваності при внесенні змін, незлагодженість системи, ризики втрати знань з певного модуля, слабка направленість проєкту на досягнення цілей. Було висунуто гіпотезу, яка потім перевірялася опитуванням. Результатом дослідження стало запровадження інтеграційної команди, яка побудована на принципах, деякі з яких суперечать цінностям Agile.

Складено план управління проєктом за змістом, часом, якістю та ризиками. Так основний обсяг робіт припадає на пакет з реалізації проєкту. Трудові ресурси якого, були оцінені в 1,3 млн. грн. Запропонована модель управління якістю складається з піраміди тестів, кожний шар тестів піраміди має свій визначений об’єм та задачі. Описана модель управління ризиків, яка побудована на базі стандартів PMBOK та MSF. Спочатку ризики ідентифікуються, проводиться їх аналіз та окреслюються плани реагування. Далі, завдяки ітеративним підходам MSF, ризики обробляються на кожній ітерації. Також була описана технологія управління проєктом, так пропонується організувати роботу доменних команд за технологією Scrum, інтеграційна

команда має використовувати принципи Kanban дошки, також початкові та завершальні етапи проєкту варто організовувати за каскадною моделлю.

Була спроєктована інформаційна система, властивістю якої є підвищена надійність та здатність витримувати високі навантаження. Для цього були використані інструменти масштабування, контролю, патерни оптимізації, патерни безпеки та розподіленості. Однією з основних задач проєктування системи, було наближення її до всіх трьох властивостей CAP теореми, для цього було застосовано дві СУБД які працюють синхронно. Описано архітектуру системи, хмарну інфраструктуру, представлено основні шматки коду, моделі БД, декларація інтерфейсів та приклад роботи MVP.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Frank Frohmann.Digital Pricing, A Guide to Strategic Pricing for the Digital Economy. Springer Nature Switzerland AG 2023 C.305-317
2. Agostino Capponi, Ruizhe Jia. Blockchain Private Pools and Price Discovery.AEA Papers and Proceedings 2023
3. Radoslav Fasuga; Pavel Stoklasa; Martin Němec. The method of automated monitoring of product prices and market position determination in relation to competition quotes: Monitoring of product prices and marketability development with continuous assessment of market position in on-line sales. 2014 11th International Conference on e-Business (ICE-B)
4. Puyun Bi. Research on distributed price monitoring system based on Multi-Agent. 2015 International Symposium on Computers and Informatics
5. Aleksandr Kovalev, Lyubov' Orlova, Pavel Domkin, Sergey Sokolov. Price dialectics and the concept of creating a unified system for monitoring pricing processes in the economy. May 2021
6. R. Steinegger, P. Giessler, B. Hippchen, Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications. April 2017
7. Building with Patterns: A Summary веб-сайт URL: https://www.mongodb.com/blog/post/building-with-patterns-a-summary (дата звернення 16.03.2024)
8. Amazon Elastic Compute Cloud Documentation веб-сайт URL: https://docs.aws.amazon.com/ec2/index.html (дата звернення 16.03.2024)
9. What is MongoDB Atlas? веб-сайт URL: https://www.mongodb.com/docs/atlas/ (дата звернення 16.03.2024)
10. Price2Spy веб-сайт URL: https://www.price2spy.com/ (дата звернення 16.03.2024)

11. Prisync веб-сайт URL: https://prisync.com/ (дата звернення 16.03.2024)

12. Jungle Scout веб-сайт URL: https://www.junglescout.com/ (дата звернення 16.03.2024)

13. SEMrush веб-сайт URL: https://www.semrush.com/ (дата звернення 16.03.2024)

14. PriceRunner веб-сайт URL: https://www.pricerunner.com/ (дата звернення 16.03.2024)

15. Feature comparison – Price2Spy веб-сайт URL: https://www.price2spy.com/feature-comparison.html (дата звернення 16.03.2024)

16. Price plan comparison – Price2Spy: веб-сайт URL: https://www.price2spy.com/pricing/comparison.html (дата звернення 16.03.2024)

17. Price2Spy Reviews 2024. Verified Reviews, Pros & Cons - Capterra: веб-сайт URL: https://www.capterra.com/p/154047/Price2Spy/reviews/ (дата звернення 16.03.2024)

18. Working at WEBCentric (Serbia) | Glassdoor: веб-сайт URL: https://www.glassdoor.com/Overview/Working-at-WEBCentric-Serbia-EI_IE255943 1.11,28.htm (дата звернення 16.03.2024)

19. Price2Spy - Overview, News & Similar companies | ZoomInfo.com: веб-сайт URL: https://www.zoominfo.com/c/price2spy/356480991 (дата звернення 16.03.2024)

20. Prisync - Overview, News & Similar companies | ZoomInfo.com: веб-сайт URL: https://www.zoominfo.com/c/prisync/357297980 (дата звернення 16.03.2024)

21. Prisync Reviews: What Is It Like to Work At Prisync? | Glassdoor : веб-сайт URL: https://www.glassdoor.com/Reviews/Prisync-Reviews-E4168437.htm (дата звернення 16.03.2024)

22. Prisync Reviews 2024. Verified Reviews, Pros & Cons - Capterra: веб-сайт URL: https://www.capterra.com/p/153451/Prisync/reviews/ (дата звернення 16.03.2024)

23. COMPETERA - Overview, News & Similar companies | ZoomInfo.com: веб-сайт URL: https://www.zoominfo.com/c/competera-ltd/368542250 (дата звернення 16.03.2024)

24. Компания Competera разработала систему управления ценами. После карантина ее завалили заказами — Forbes.ua : веб-сайт URL: https://forbes.ua/ru/news/tsina-mae-znachennya-30102020-216 (дата звернення 16.03.2024)

25. Ukrainian-founded Competera secures $3M in a seed round – AIN.Capital: веб-сайт URL: https://ain.capital/2024/01/30/ukrainian-founded-competera-secures-3m-in-a-seed-round/ (дата звернення 16.03.2024)

26. Working at Competera | Glassdoor веб-сайт: URL: https://www.glassdoor.com/Overview/Working-at-Competera-EI_IE1798912.11,20.htm (дата звернення 16.03.2024)

27. Competera: веб-сайт URL: https://competera.net/ (дата звернення 16.03.2024)

28. Competera Reviews 2024. Verified Reviews, Pros & Cons - Capterra: веб-сайт URL: https://www.capterra.com/p/136974/Competera-Pricing-Platform/reviews/ (дата звернення 16.03.2024)

29. Закон Конвея: веб-сайт: URL: https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D1%8F (Дата звернення: 5.03.24)

30. S. Newman, Building Microservices, First Edition, 2015

31. Neal Ford, Building Evolutionary Architectures: Support Constant Change 1st Edition. 2017

32. Mike Cohn, Agile Estimating and Planning , 2005

33. Principles behind the Agile Manifesto URL: https://agilemanifesto.org/principles.html, (Дата звернення: 5.03.24)

34. Roy Osherove, The Art of Unit Testing, Second Edition, 2013

35. Vlad Khononov, Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy 1st Edition, 2021

36. CAP Theorem Explained — Consistency, Availability & Partition Tolerance: веб-сайт: https://medium.com/@kmrpushkar09/cap-theorem-explained-consistency-availability-partition-tolerance-19d151db54f3 (Дата звернення: 5.03.24)

ДОДАТОК А

Оцінка експертами факторів STEP-аналізу

Таблиця А.1

Експертне оцінювання політичних факторів

Фактор	Хар-р.	Експ 1	Експ 2	Експ 3	Експ 4	Ср. бал
Збільшення податкового навантаження	-	3	2	3	2	-2,5
Прагнення влади до контролю галузі	-	1	1	2	2	-1,5
Захист інтелектуальної власності	+	2	3	3	2	+2,5
Бюрократизація і рівень корупції	-	1	1	1	2	-1,25
Стабільність уряду	+	1	1	1	1	+1

Таблиця А.2

Експертне оцінювання економічних факторів

Фактор	Хар-р.	Експ 1	Експ 2	Експ 3	Експ 4	Ср. бал
Розмір заробітної плати	-	2	2	3	3	-2,5
Вартість та доступність кредитування	+	3	3	3	2	+2,75
Динаміка розвитку економіки	+	1	1	2	1	+1,25
Курс валют	+	1	2	2	1	+1,5
Рівень інфляції	-	1	1	1	1	-1
Валютні обмеження, обмеження пересування капіталу	-	1	2	2	1	-1,5

Таблиця А.3

Експертне оцінювання соціальних факторів

Фактор	Хар-р.	Експ 1	Експ 2	Експ 3	Експ 4	Ср. бал
Міграція населення	-	1	2	2	3	-2
Середній вік	-	1	1	1	1	-1
Баланс праці та відпочинку	+	2	2	1	2	+1,75
Розмір і структура сім'ї	+	1	1	2	1	+1,25
Ставлення до IT-галузі	+	2	1	2	1	+1,5
Темпи споживання	+	1	1	1	2	+1,25

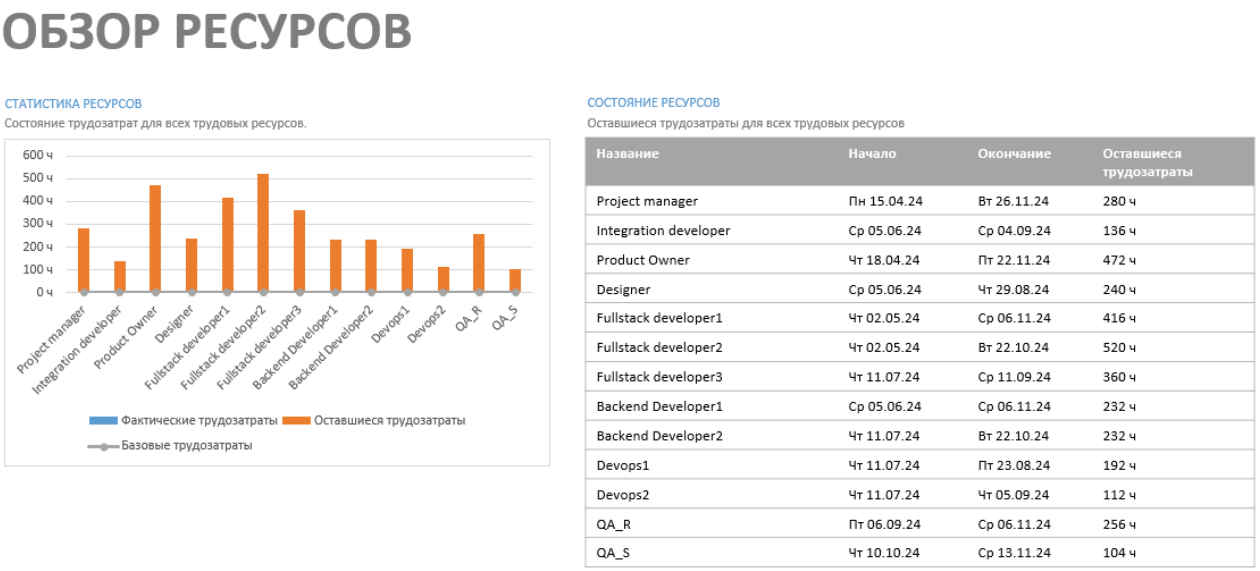
Таблиця А.4

Експертне оцінювання технічних факторів

Фактор	Хар-р.	Експ 1	Експ 2	Експ 3	Експ 4	Ср. бал
Доступність та швидкість інтернету	+	3	3	3	3	+3
Доступ до нових технологій	+	3	2	3	2	+2,5
Рівень інновації та технічного розвитку	+	1	2	2	2	+2,75
Державна технічна політика	+	1	1	1	1	+1

ДОДАТОК Б

Огляд ресурсів



Название	Начало	Окончание	Оставшиеся трудозатраты
Project manager	Пн 15.04.24	Вт 26.11.24	280 ч
Integration developer	Ср 05.06.24	Ср 04.09.24	136 ч
Product Owner	Чт 18.04.24	Пт 22.11.24	472 ч
Designer	Ср 05.06.24	Чт 29.08.24	240 ч
Fullstack developer1	Чт 02.05.24	Ср 06.11.24	416 ч
Fullstack developer2	Чт 02.05.24	Вт 22.10.24	520 ч
Fullstack developer3	Чт 11.07.24	Ср 11.09.24	360 ч
Backend Developer1	Ср 05.06.24	Ср 06.11.24	232 ч
Backend Developer2	Чт 11.07.24	Вт 22.10.24	232 ч
Devops1	Чт 11.07.24	Пт 23.08.24	192 ч
Devops2	Чт 11.07.24	Чт 05.09.24	112 ч
QA_R	Пт 06.09.24	Ср 06.11.24	256 ч
QA_S	Чт 10.10.24	Ср 13.11.24	104 ч

Рис. Б.1. Огляд ресурсів проєкту у MS Project

ДОДАТОК В

Вартісна оцінка проєкту

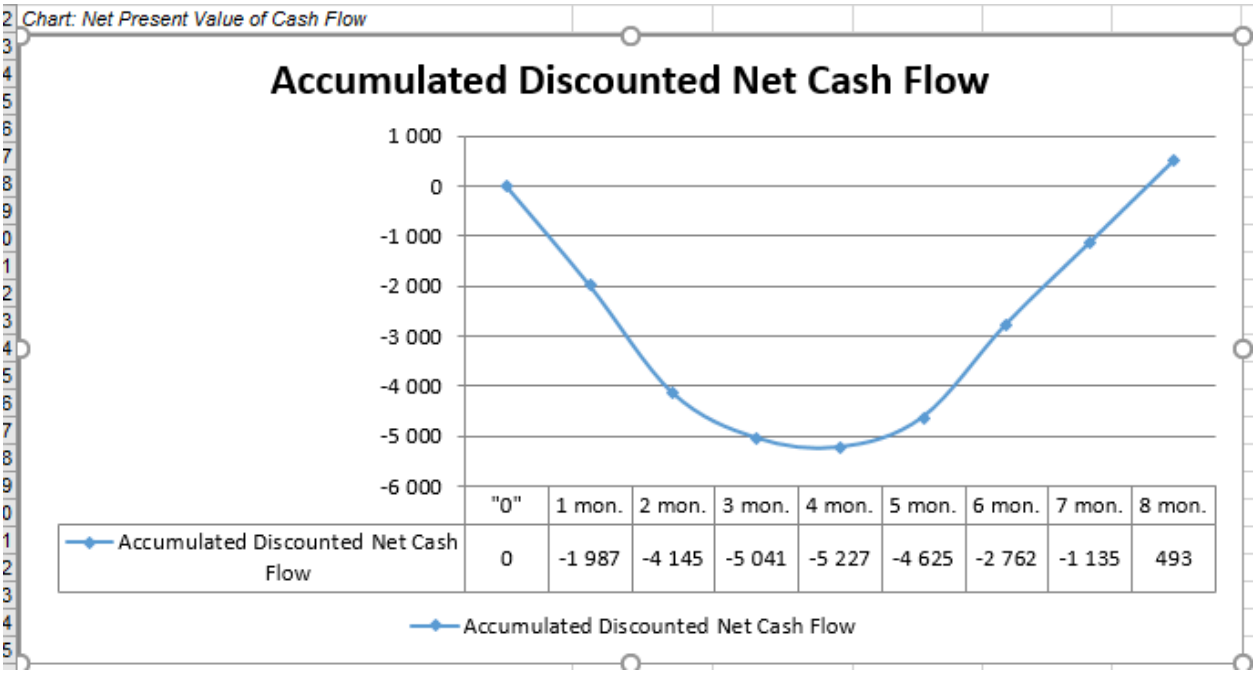


Рис. В.1. Графік окупності проєкту

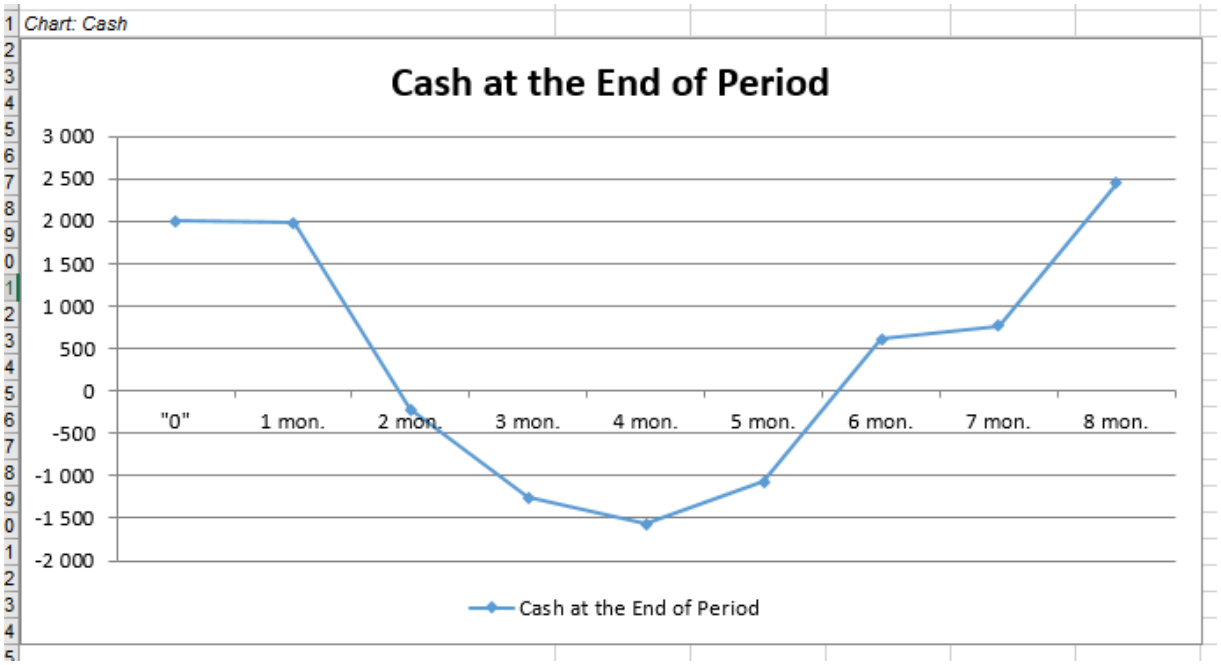


Рис. В.2. Графік залишку грошей на рахунку

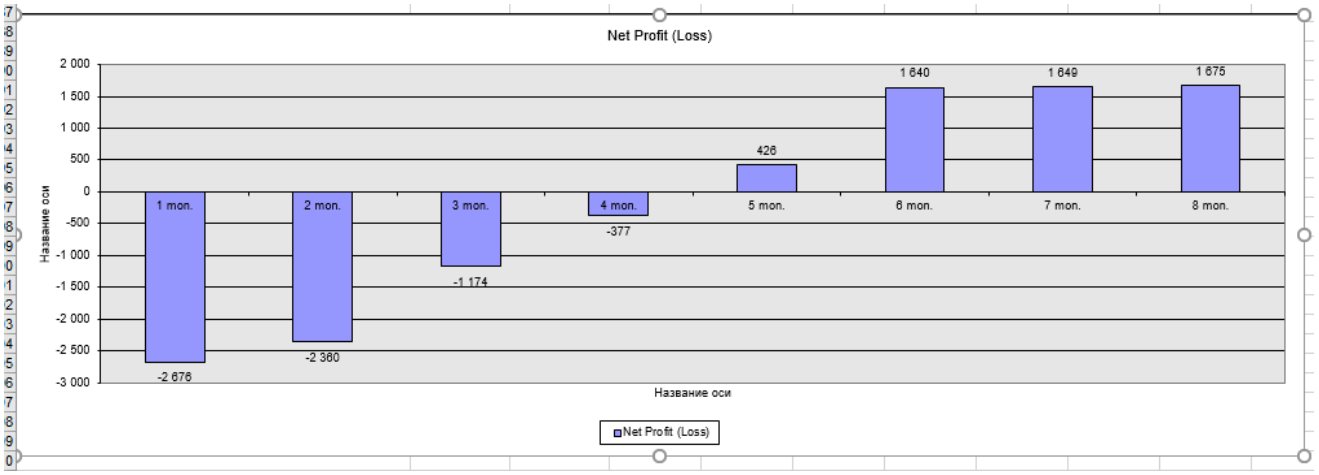


Рис. В.3. Графік чистого прибутку/збитку

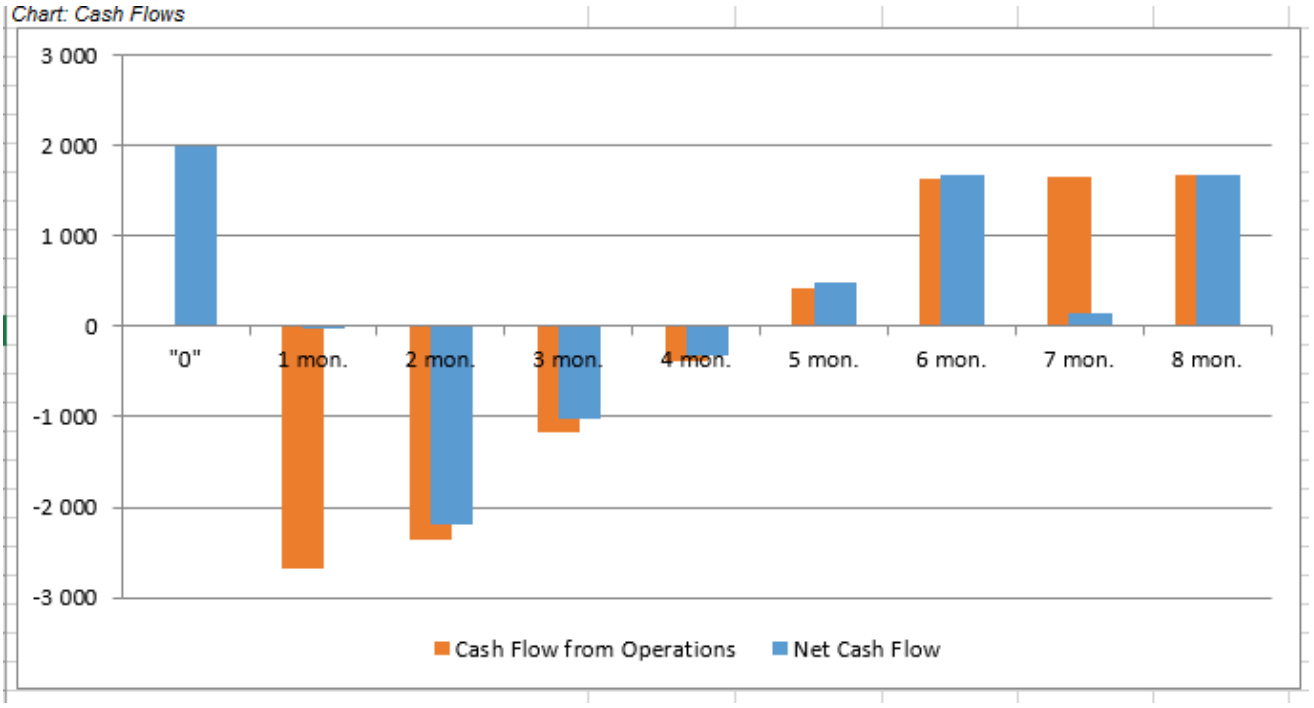


Рис. В.4. Графік руху коштів

SUMMARY OF INVESTMENTS			"0"	1 mon.	2 mon.	3 mon.	4 mon.	5 mon.	6 mon.	7 mon.	8 mon.	TOTAL
Required Investments:		gr. 000	**0**	**-656**	**-159**	**-242**	**-164**	**-166**	**-248**	**-7**	**-7**	***-1 649***
Construction Expenses Paid		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Purchase of Equipment and Other Property		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Deferred Expenses Paid		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Investments in Working Capital		gr. 000	0	-656	-159	-242	-164	-166	-248	-7	-7	*-1 649*
Source of Financing:		gr. 000	**2 000**	**2 000**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	***4 000***
Proceeds from Issue Of Share Capital		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Grants and similar investments		gr. 000	0	2 000	0	0	0	0	0	0	0	*2 000*
Proceeds from Debt		gr. 000	2 000	0	0	0	0	0	0	0	0	*2 000*
Leased Equipment		gr. 000										*0*
Payments for the Capital		gr. 000	**0**	**25**	**25**	**125**	**124**	**123**	**221**	**1 519**	**0**	***2 161***
Repayment of Debt		gr. 000	0	0	0	100	100	100	200	1 500	0	*2 000*
Interest Paid		gr. 000	0	25	25	25	24	23	21	19	0	*161*
Lease Payments		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Dividends Paid		gr. 000	0	0	0	0	0	0	0	0	0	*0*
Info: Cash at the end of period		gr. 000	*2 000*	*1 980*	*-221*	*-1 253*	*-1 565*	*-1 073*	*614*	*770*	*2 451*	
Minimum cash at the end of period	*-1 565*	gr. 000										

Рис. В.5. Зведений звіт про інвестиції в проєкт