МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 1636–“С” 2024.10.29. 010 ПЗ

ЛИНЯ АНДРІЯ МИКОЛАЙОВИЧА

2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2024 р.	**____ _____ 2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему __Система обліку фінансових показників з аналітичним модулем_

Спеціальність ___122 - Комп'ютерні науки_____
(код і назва)

Освітня програма ____Інформаційно управляючі системи та технології_
(назва)

Орієнтація освітньої програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____ _____ ___Густер О.М.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____к.т.н., доцент_____ _____ _____Ткаченко О.М._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Линь А.М._____
(підпис) (ПІБ студента)

КИЇВ - 2024

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____ 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Линь Андрій Миколайович
(прізвище, ім'я, по батькові)

Спеціальність 122 - Комп'ютерні науки
(код і назва)

Освітня програма Інформаційно управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Система обліку фінансових показників з аналітичним модулем

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи отримання звітів на основі аналізу фінансових даних з біржових джерел та формування прогнозування зросту або падіння цін

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області
2. Моделювання системи
3. Розробка системи
4. Результати дослідження

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської кваліфікаційної роботи _____ Ткаченко О.М.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Линь А.М.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

ПЗ – програмне забезпечення.

UML (Unified Modeling Language) – уніфікована мова моделювання.

Інтернет-послуги – Послуги які надаються користувачам щодо забезпечення доступу до мережі, розробки і розміщення реклами, організаційного та інформаційного супроводу інтернет-ресурсів.

Браузер – Програма навігації та перегляду веб-ресурсів, дозволяє запитувати і проглядати файли в Інтернет.

Каталог – Перелік однорідних об'єктів, складений в порядку, що полегшує їх знаходження.

SQL – Structured Query Language.

CRUD – (англ. Create, Read, Update, Delete) чотири базові функції управління даними «створення, зчитування, зміна і видалення».

ОС – операційна система.

URL – (Uniform Resource Locator) – єдиний вказівник на ресурс.

PostgreSQL - це потужна об'єктно-реляційна система керування базами даних (СКБД), яка надає широкі можливості зберігання, організації та маніпулювання даними. Вона відноситься до відкритих джерел і заснована на мові SQL.

ASP .NET - це високорівневий .NET веб-фреймворк, який дозволяє швидко створювати безпечні і підтримувані веб-сайти [2].

ВСТУП

У сучасному бізнес-середовищі аналіз фінансових показників має вирішальне значення для ефективного прийняття рішень, що дозволяє компаніям оцінювати минулі результати, планувати майбутні дії та забезпечувати стійке зростання. Однак самих фінансових даних часто недостатньо без ретельного аналізу та інтерпретації. Це підкріплює потребу в комплексній системі обліку фінансової ефективності з аналітичним модулем, який поєднує функції фінансового обліку з поглибленими аналітичними можливостями. Така система допомагає організаціям перетворювати необроблені фінансові дані в практичні висновки, тісно відповідаючи сучасним тенденціям у прийнятті рішень на основі даних і цифровій трансформації.

Актуальність цього дослідження підкреслюється зростаючою роллю фінансової аналітики в організаційній стратегії. Сучасний конкурентний ринок вимагає точного та своєчасного аналізу, що дозволяє оперативно реагувати на економічні зміни та більш надійне фінансове управління. Інтеграція фінансового обліку з аналітикою дає менеджерам можливість приймати обґрунтовані рішення на основі прогнозних і приписуваних ідей.

Таким чином, розробка системи, яка автоматизує та вдосконалює аналіз фінансових показників, є своєчасною та важливою для ефективності бізнесу.

Предметом дослідження є методологія та технологія проектування та впровадження системи бухгалтерського обліку фінансової діяльності.

Об'єктом дослідження є фінансові та аналітичні процеси в організації, зосереджуючись на тому, як ці процеси можна оптимізувати за допомогою технологій.

Мета дослідження спрямоване на покращення існуючих фінансових систем по прогнозування зросту цін розробивши систему обліку фінансової ефективності з надійним аналітичним модулем, здатним забезпечити всебічне розуміння фінансового стану організації. Кінцева мета полягає в тому, щоб підвищити ефективність прийняття рішень і операцій за допомогою оптимізованої обробки даних і розширених аналітичних можливостей.

Для досягнення мети дослідження були окреслені наступні **завдання**:

- проведення системного аналізу вимог до фінансової ефективності та поточного технологічного ландшафту в бухгалтерському обліку;
- розробка концептуальної моделі інтегрованої системи фінансового обліку та аналітики;
- розробити та впровадити систему, забезпечивши функціональність як стандартної бухгалтерії, так і розширеної аналітики;
- перевірити ефективність системи, протестувавши її в контрольованому організаційному середовищі та оцінивши її продуктивність у реальних сценаріях.

Дослідження використовує кілька **методів** дослідження, включаючи системний аналіз, методи розробки програмного забезпечення, моделювання

даних і тестування прототипів. Системний аналіз дає змогу зрозуміти вимоги користувачів, а моделювання даних допомагає структурувати фінансову інформацію. Практика розробки програмного забезпечення використовується для керівництва розробкою системи, а тестування прототипу гарантує, що система відповідає функціональним і аналітичним потребам.

Наукова новизна даного дослідження полягає в інтеграції фінансового обліку та розширеної аналітики в єдину систему. На відміну від традиційного бухгалтерського програмного забезпечення, ця система включає прогнозну та приписну аналітику, що дозволяє оцінювати фінансові показники в реальному часі та прогнозувати майбутнє. Дослідження також пропонує структурований підхід до об’єднання даних бухгалтерського обліку з аналітичними процесами, вносячи нові методології в розробку фінансового програмного забезпечення.

Апробація результатів дослідження: систему тестували в різних сценаріях, щоб оцінити її функціональність, точність і зручність використання. Ці тести включали як змодельовані фінансові дані, так і реальні тематичні дослідження, демонструючи його ефективність у реальних фінансових середовищах. Результати показали, що система успішно інтегрувала функції бухгалтерського обліку з аналітикою, виявившись цінною для оцінки ефективності та стратегічного планування.

Магістерська робота має наступну **структуру**:

Системний аналіз предметної області – у цьому розділі розглядається поточний ландшафт систем фінансового обліку та визначаються ключові вимоги до інтеграції аналітичних функцій. Моделювання системи – у цьому розділі детально описано дизайн і моделювання запропонованої системи обліку фінансових результатів, охоплюючи як функціональні вимоги, так і аналітичні можливості. Розробка системи – у цьому розділі описано процес розробки, включаючи архітектуру системи, кодування та початкове

тестування. Результати досліджень – у останньому розділі представлені результати тестування та аналізу продуктивності системи, що дає уявлення про її ефективність і сфери для подальшого розвитку.

Це дослідження, завдяки поєднанню фінансового обліку та аналітики, пропонує цінний інструмент для сучасного управління фінансами, сприяючи покращенню процесу прийняття рішень та ефективності.

Магістерська робота містить 86 сторінок, в тому числі 28 рисунків та 2 додатки. Вона посилається на 31 джерело, включаючи наукові статті, книги та електронні ресурси.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

В останні роки інтеграція аналітики в системи обліку фінансової ефективності стала важливою через зростаючу складність фінансових даних і попит на більш складні інструменти підтримки прийняття рішень. Традиційні системи бухгалтерського обліку в першу чергу обробляють записи транзакцій, фінансову звітність і дотримання нормативних вимог, але часто не можуть забезпечити аналіз у реальному часі та прогнозне розуміння, що має важливе значення для сучасного динамічного бізнес-середовища. Щоб усунути цю прогалину, організації все частіше шукають передові аналітичні інструменти, які можуть перетворювати необроблені фінансові дані в практичну інформацію, сприяти прогнозуванню тенденцій і сприяти прийняттю стратегічних рішень.

Системи обліку фінансової ефективності традиційно розроблені для управління основними фінансовими функціями організації — доходами, витратами, активами, зобов’язаннями, бюджетуванням і прогнозуванням. Ці

системи в першу чергу зосереджені на попередніх показниках, що дозволяє компаніям оцінювати минулу фінансову діяльність і гарантувати дотримання нормативних стандартів. Однак сучасний бізнес потребує більш вдосконаленого підходу, який включає аналіз у реальному часі, розширені можливості прогнозування та інструменти оптимізації продуктивності, щоб залишатися конкурентоспроможними.

Аналітичний модуль, вбудований у програмне забезпечення фінансового обліку, пропонує кілька ключових переваг для організацій, таких як здатність аналізувати моделі, прогнозувати майбутні фінансові результати та покращувати загальне прийняття рішень. Завдяки прогнозній аналітиці компанії отримують можливість передбачати тенденції доходів, розуміти поведінку витрат і прогнозувати майбутні фінансові стани. Завдяки використанню аналітичних даних, система може запропонувати конкретні рекомендації, дозволяючи фінансовим менеджерам приймати рішення на основі даних, які покращують розподіл ресурсів і прибутковість. Крім того, інтеграція показників ефективності та ключових показників ефективності забезпечує постійну систему оцінювання, яка допомагає управлінським командам контролювати фінансовий стан і точно визначати області, які потребують уваги.

Еволюція технології фінансової аналітики представила низку потужних інструментів для інтеграції, аналізу та візуалізації даних. Основні технології включають сховище даних, яке централізує дані з багатьох джерел для комплексного аналізу, а також інструменти бізнес-аналітики (BI), такі як Power BI, Tableau та Qlik, які пропонують інтерактивні інформаційні панелі для аналізу в реальному часі. Алгоритми машинного навчання підтримують прогнозну та приписну аналітику, допомагаючи компаніям визначати тенденції та генерувати практичні висновки на основі історичних даних. Платформи хмарних обчислень, такі як AWS і Azure, забезпечують масштабовану інфраструктуру для великих наборів даних і спільного аналізу. Крім того, API та інструменти інтеграції даних сприяють безперешкодному

обміну даними між фінансовою системою та іншими системами ERP або CRM[1].

Незважаючи на переваги, організації стикаються з кількома проблемами в обліку фінансової діяльності. Розташовані дані, де інформація розосереджена між окремими системами, перешкоджають комплексному фінансовому аналізу. Традиційним системам також бракує можливостей реального часу, що обмежує здатність організації швидко адаптуватися до мінливих фінансових умов. Точне прогнозування залишається проблемою, часто покладаючись на ручні методи, схильні до помилок. Нарешті, масштабованість є критичною; у міру зростання організації фінансова система повинна підтримувати більші обсяги даних і більш складний аналіз без втрати ефективності.

Щоб інтегрована фінансова ефективність та аналітична система була ефективною, вона повинна консолідувати дані з багатьох джерел, що дозволяє отримати цілісне уявлення про фінансовий стан. Він також має бути масштабованим, здатним обробляти більші обсяги даних і складнішу аналітику в міру розвитку потреб організації. Удосконалені інструменти прогнозування та розуміння в реальному часі є важливими, що дозволяє системі надавати релевантні дані в темпі, необхідному для швидкого прийняття рішень. Зручні для користувача інтерфейси, включаючи інформаційні панелі та візуалізації, покращують зручність використання, забезпечуючи доступність фінансової інформації як для технічних, так і для нетехнічних користувачів. Безпека даних і відповідність нормативним вимогам є важливими, враховуючи конфіденційний характер фінансової інформації та нормативні стандарти, яким повинні відповідати організації.

Поточні рішення на ринку варіюються від систем планування ресурсів підприємства (ERP), таких як SAP і Oracle Financials, до спеціалізованих інструментів фінансової звітності. Однак багато з цих рішень або зосереджені на управлінні транзакціями, або надають обмежені аналітичні функції, що призводить до потреби в інтегрованому рішенні, яке поєднує обидві

можливості. Ідеальні системи повинні забезпечувати безперебійну взаємодію між транзакційними та аналітичними функціями, зменшуючи ручну консолідацію даних і покращуючи процес прийняття рішень. Крім того, системам загального призначення часто не вистачає прогнозної та директивної аналітики, розробленої спеціально для фінансових показників. Настроювані параметри, які можна адаптувати до конкретних організаційних вимог і нормативних стандартів, також залишаються прогалиною в багатьох існуючих інструментах[2].

Агрегатор валютних курсів — це інтернет-сервіс або програмний інструмент, що надає користувачам доступ до актуальної інформації про курси обміну валюти, отримані від різних банків, фінансових установ, біржів та інших джерел. Його головне завдання — забезпечити користувачів точними і свіжими даними щодо поточних курсів валют, що дозволяє швидко орієнтуватися на валютному ринку та приймати обґрунтовані рішення.

Система агрегатора курсу валют функціонує за рахунок збору даних із численних джерел, включаючи банки, фінансові організації, біржі та інтернет-ресурси. Зібрані дані постійно оновлюються, щоб гарантувати користувачам максимально актуальну інформацію, з можливістю оновлень як у реальному часі, так і з певною часом. Одна з важливих функцій агрегатора — це можливість порівнювати курси з різними джерелами, що дає користувачам можливість оцінити різницю між ними та обрати найвигідніший варіант обміну.

Агрегатор також може включати функції для візуалізації курсів у графіках та діаграмах, які допомагають аналізувати тенденції на валютному ринку та прогнозувати подальші коливання. Це полегшує користувачам аналіз змін і дає змогу скористатися більш обґрунтованими фінансовими рішеннями. Додатково, агрегатори часто надають можливості налаштування та сповіщення: користувачі можуть встановити сповіщення для отримання повідомлень про зміни курсу, наприклад, коли валюта досягає певного значення або перетинає заданий поріг[2].

Крім основної інформації про курси, агрегатор може забезпечувати користувачів додатковими даними, такими як новини валютного, фінансові аналітичні огляди та інші корисні відомості, які сприяють більшій обґрунтованості валютних операцій. Агрегатор валютних курсів таким чином стає зручним інструментом для відстеження та порівняння даних з різних джерел в одному місці. Це корисно для людей, які займаються валютними операціями, часто подорожчають або цікавляться фінансовими ринками.

1.2 Огляд існуючих рішень

Розглянемо існуючі рішення предметної області.

Oracle Financials — це комплексний набір додатків, розроблених для керування та оптимізації фінансових операцій на підприємствах. Це основний компонент Oracle E-Business Suite і перетворився на хмарний Oracle Fusion Cloud Financials, що забезпечує гнучкість, масштабованість та інтеграцію з іншими програмами Oracle Cloud.

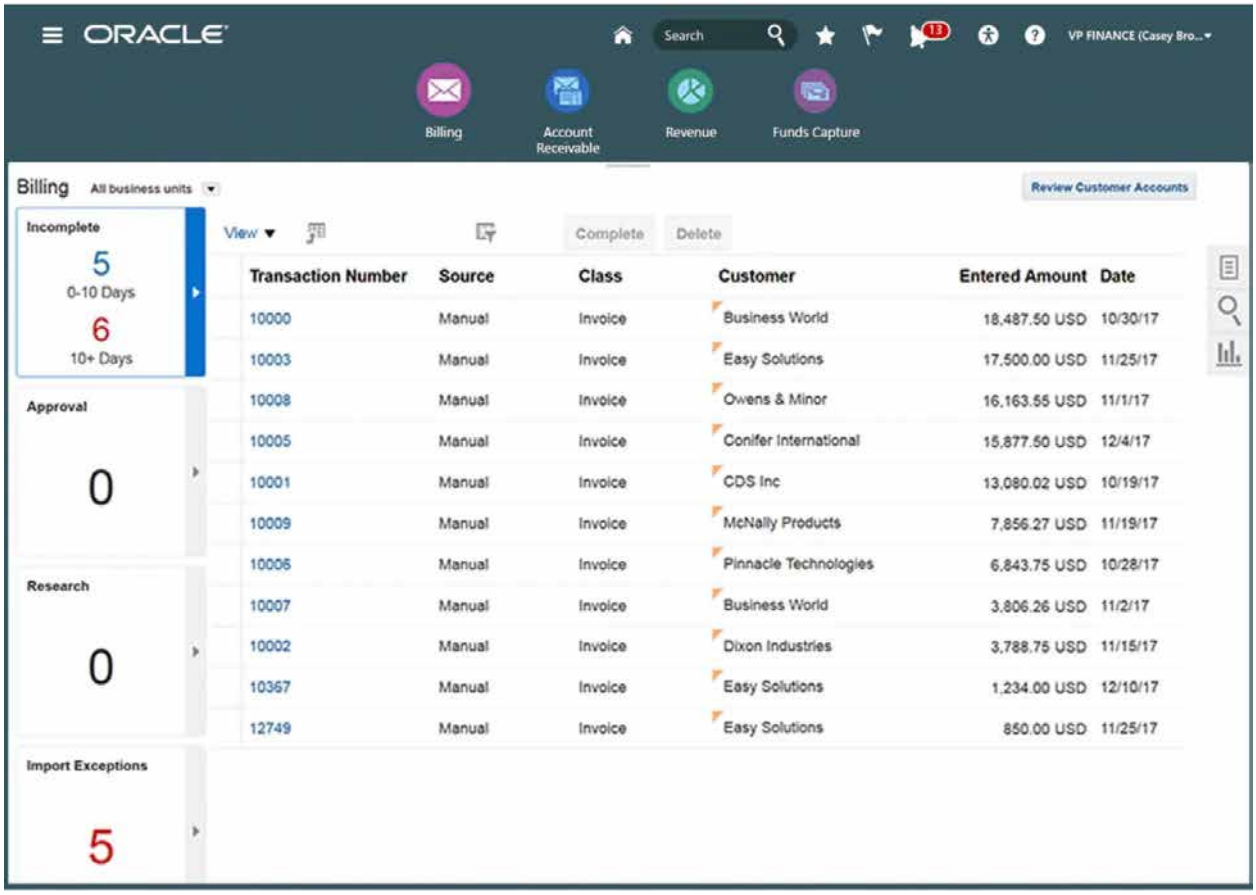

Рис. 1 Програмне забезпечення «Oracle Financials»

Oracle Financials надає надійну головну книгу, яка дозволяє організаціям ефективно керувати, записувати та аналізувати фінансові операції. Він підтримує багатовимірну звітність, яка дає змогу користувачам аналізувати фінансові дані в режимі реального часу в різних сегментах, таких як відділи, регіони або лінії продуктів. Крім того, він підтримує кілька валют, юридичних осіб і глобальні стандарти відповідності, що робить його добре придатним для багатонаціональних організацій[3].

Oracle Financials автоматизує процеси кредиторської та дебіторської заборгованості, оптимізуючи операції з виставлення рахунків, платежів, зборів і управління клієнтами. Автоматизація зменшує ймовірність ручних помилок, прискорює цикли грошових потоків і покращує видимість непогашеної дебіторської заборгованості та зобов’язань постачальників. Ці модулі також пропонують підтримку електронних платежів, забезпечуючи інтеграцію з банками та платіжними процесорами для ефективної обробки транзакцій.

Oracle Financials містить інструменти для бюджетування, фінансового планування та прогнозування, що дозволяє організаціям встановлювати фінансові цілі, створювати бюджети та відстежувати відхилення. Інтеграція з інструментами планування та бюджетування Oracle дозволяє проводити детальне фінансове моделювання, прогнозну аналітику та аналіз сценаріїв, що дає змогу фінансовим командам адаптуватися до мінливих ринкових умов і приймати проактивні рішення.

Звітність і аналітика є центральними для Oracle Financials із доступом до даних у реальному часі через Oracle Analytics Cloud Oracle. Платформа надає комплексні можливості звітності, які можна налаштувати та адаптувати до конкретних потреб, пропонуючи інформацію через інформаційні панелі, ключові показники ефективності (KPI) і фінансові показники. Машинне навчання та прогнозна аналітика допомагають користувачам аналізувати тенденції та робити прогнози на основі даних, що підтримує довгострокове планування та стратегію.

Oracle Financials підтримує відстеження та управління основними активами, від придбання до амортизації та вибуття. Цей модуль забезпечує відповідність фінансовим нормам щодо оцінки активів, допомагаючи організаціям точно реєструвати та звітувати про вартість своїх основних засобів. Він також надає інформацію про ефективність активів, що дозволяє краще приймати рішення щодо використання та обслуговування активів[3].

Можливості управління грошовими коштами в Oracle Financials допомагають організаціям контролювати грошові потоки, керувати ліквідністю та прогнозувати грошові позиції. Модуль управління казначейством керує інвестиціями, боргами та ризиками, допомагаючи організаціям підтримувати оптимальний рівень готівки та уникати проблем з ліквідністю. Ці інструменти дозволяють аналізувати надходження та відтоки грошових коштів, підтримуючи краще фінансове планування та інвестиційні стратегії.

Oracle Financials містить вбудовані інструменти для керування дотриманням глобальних фінансових норм, таких як МСФЗ, GAAP і SOX. Платформа підтримує журнали аудиту, контроль доступу та розподіл обов'язків, забезпечуючи цілісність і безпеку фінансових даних. Крім того, інструменти управління ризиками дозволяють організаціям ефективно контролювати та зменшувати фінансові ризики.

Як частина екосистеми Oracle, Oracle Financials легко інтегрується з іншими програмами Oracle, такими як Oracle Human Capital Management (HCM) і Oracle Supply Chain Management (SCM). Ця інтеграція дозволяє спрощувати обмін даними та комплексну автоматизацію процесів у всій організації. Його масштабованість робить його придатним для підприємств будь-якого розміру, від малих і середніх підприємств до великих глобальних корпорацій.

Oracle Financials можна розгорнути як локально, так і через хмарну інфраструктуру Oracle. Хмарна версія, Oracle Fusion Cloud Financials, пропонує додаткові переваги автоматичного оновлення, покращеної безпеки даних і зниження витрат на інфраструктуру. Ця гнучкість дозволяє компаніям вибирати найкращий метод розгортання для своїх потреб, водночас користуючись перевагами потужних фінансових інструментів Oracle.

Oracle Financials має зручний інтерфейс, який доступний на всіх пристроях, включаючи настільні ПК, планшети та мобільні пристрої. Завдяки сучасному дизайну UI/UX, інтерактивним інформаційним панелям і доступній візуалізації даних користувачі можуть інтуїтивно керувати платформою та бути в курсі фінансових показників, де б вони не були.

Розглянемо інший продукт.

SAP S/4HANA Finance — це розширене рішення для управління фінансами в пакеті SAP S/4HANA ERP, призначене для підтримки організацій за допомогою обробки даних у режимі реального часу, надійної аналітики та бездоганної інтеграції фінансових операцій. S/4HANA Finance, що працює на базі даних HANA в пам'яті SAP, забезпечує швидкий доступ до даних,

дозволяючи користувачам приймати обґрунтовані рішення з фінансовою інформацією в реальному часі[4].

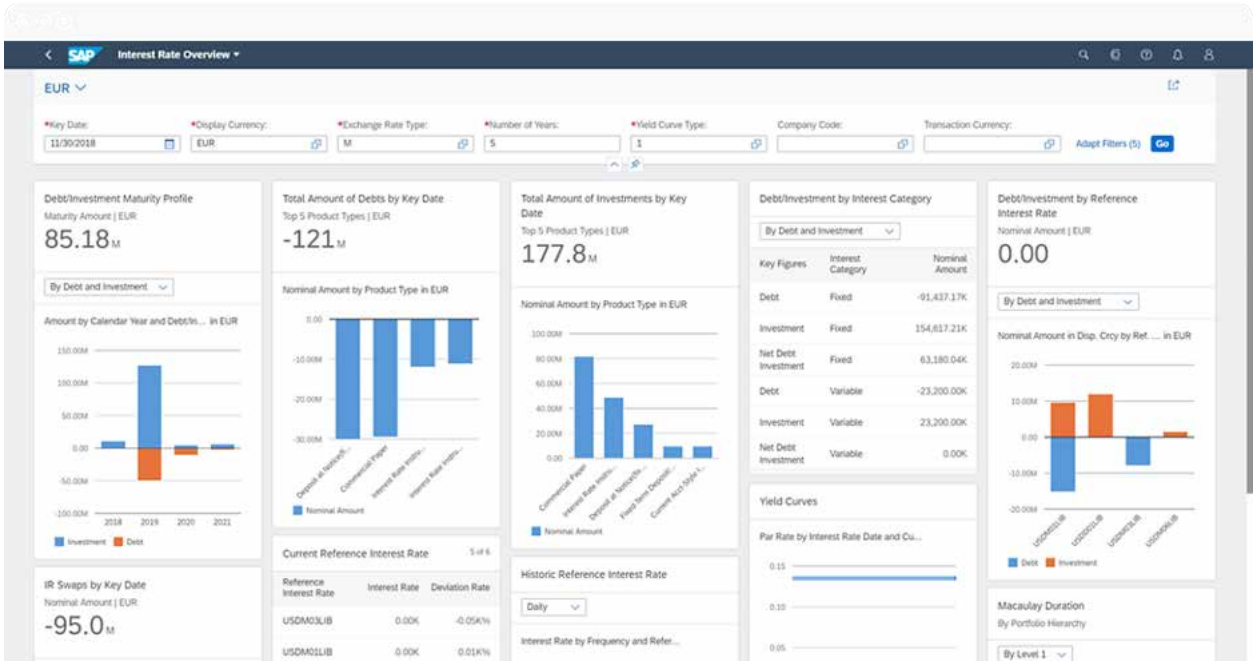

Рис. 2 Програмне забезпечення «SAP S/4HANA»

В основі SAP S/4HANA Finance — обробка даних у реальному часі за допомогою SAP HANA. Це дозволяє організаціям отримувати доступ до найновіших фінансових даних без пакетної обробки чи затримки даних. Він підтримує постійне закриття та фінансову консолідацію в режимі реального часу, що дозволяє фінансовим командам закривати книги швидше та точніше.

S/4HANA Finance використовує єдиний запис «універсального журналу» для уніфікації даних із головної книги, таблиць обліку активів, контролінгу та бухгалтерської книги матеріалів. Цей консолідований підхід спрощує структури даних, усуває надмірності та підвищує ефективність. Зменшуючи потреби в узгодженні даних, універсальний журнал дозволяє детально аналізувати фінансовий і управлінський облік в одному місці, що дає змогу швидше отримати інформацію та прийняти рішення[4].

S/4HANA Finance пропонує широкі інструменти фінансового планування та аналізу (FP&A) для бюджетування, прогнозування та планування сценаріїв. Інтеграція з SAP Analytics Cloud дозволяє користувачам

створювати динамічні фінансові моделі та досліджувати різні сценарії, підтримуючи краще стратегічне планування. Завдяки можливостям прогнозної аналітики та машинного навчання користувачі також можуть створювати прогнози на основі історичних даних і поточних тенденцій.

Модулі «Кредиторська заборгованість» (AP) і «Дебіторська заборгованість» (AR) призначені для оптимізації процесів, зменшення кількості помилок, що виникають вручну, і покращення управління грошовими потоками. Вони надають інструменти для більш ефективного управління платежами, зборами, кредитами та суперечками. Функції автоматизації, такі як автоматичне зіставлення рахунків-фактур і інтелектуальне збирання коштів, допомагають фінансовим командам заощадити час і підвищити точність обробки фінансових транзакцій[4].

SAP S/4HANA Finance містить вдосконалені інструменти управління готівкою та казначейством, які допомагають організаціям керувати ліквідністю та зменшувати фінансові ризики. Він пропонує аналіз грошових потоків у реальному часі, прогнозування готівки та комплексні казначейські функції, такі як управління інвестиціями та боргом. Модуль казначейства підтримує глобальні операції з мультивалютним керуванням, допомагаючи фінансовим командам приймати стратегічні рішення та оптимізувати грошові резерви.

Завдяки інструментам для безперервного та прискореного фінансового закриття S/4HANA Finance допомагає організаціям дотримуватися нормативних стандартів, таких як IFRS та GAAP. Звірки в режимі реального часу та журнали аудиту гарантують точність і прозорість фінансових даних, підтримуючи дотримання як внутрішнього, так і зовнішнього аудиту. Рішення також підтримує нормативну відповідність функціям електронних підписів, захисту даних і контролю доступу користувачів.

SAP S/4HANA Finance пропонує вбудовані можливості аналітики та звітності, включаючи настроювані інформаційні панелі та KPI для

моніторингу фінансової ефективності. Користувальницький інтерфейс на базі SAP Fiori забезпечує інтуїтивно зрозумілий досвід на основі ролей з інтерактивними інформаційними панелями, які дозволяють користувачам детально переглядати певні точки даних. Інтеграція з SAP Analytics Cloud додатково розширює можливості, дозволяючи користувачам використовувати розширені аналітичні засоби, такі як прогнозне моделювання та машинне навчання, щоб отримати глибше розуміння.

Модуль обліку основних засобів підтримує весь життєвий цикл основних засобів, від придбання до амортизації та вибуття. Завдяки функціям для розрахунку амортизації, переоцінки активів і звітності, це гарантує, що організації можуть контролювати та точно враховувати вартість активів з часом. Інтеграція модуля з іншими фінансовими функціями допомагає підтримувати відповідність стандартам фінансової звітності та оптимізує використання активів.

Аналіз витрат і прибутковості (CO-PA) у S/4HANA Finance дозволяє компаніям аналізувати прибутковість за клієнтами, продуктами та регіонами. Використовуючи дані в режимі реального часу, фінансові команди можуть відстежувати витрати та прибутки, допомагаючи їм приймати обґрунтовані рішення щодо ціноутворення на продукт, скорочення витрат і підвищення продуктивності. Ця функція бездоганно інтегрується з даними про продажі, ланцюжок поставок і виробництво, надаючи комплексне уявлення про прибутковість усієї організації.

SAP S/4HANA Finance — це потужне рішення для управління фінансами, яке поєднує в собі обробку даних у реальному часі, розширену аналітику та гнучке розгортання. Його спрощена структура даних і інтегровані модулі роблять його придатним для складних організацій, забезпечуючи швидше фінансове закриття, розширене планування й аналіз, а також покращену відповідність. Завдяки прогнозній аналітиці, інтуїтивно зрозумілим інтерфейсам користувача та бездоганній інтеграції в екосистему

SAP S/4HANA Finance дає можливість організаціям оптимізувати фінансову ефективність і приймати стратегічні рішення на основі даних[4].

1.3 Постановка завдання

Ціль цього проекту — надати користувачам можливість контролювати та оцінювати ключові фінансові показники, пов’язані з ефективністю їхньої організації, такі як доходи, витрати, прибутковість і грошові потоки. Розроблена програма використовує базу даних, куди користувачі можуть вводити та аналізувати свої фінансові дані, що дозволяє їм створювати персоналізовані звіти про фінансовий стан і тенденції.

Основна мета програмного забезпечення — запропонувати користувачам швидкий доступ до важливої інформації про фінансові показники їх організації, підтримуючи прийняття обґрунтованих рішень. Основні функції системи включають надання користувачам можливості швидкого доступу до статистики щодо:

- ефективність їх стратегій управління фінансами;
- тенденції доходів, витрат і прибутковості з часом;
- загальний грошовий потік і стан ліквідності.

коригування, необхідні для досягнення конкретних фінансових цілей або контрольних показників.

Програма працює в інтерактивному режимі за допомогою структурованого меню, де користувачі вибирають конкретні запити, а програма виконує відповідний аналіз і відображає відповідну фінансову інформацію на основі їх вибору. Ця структура покращує взаємодію з користувачами, проводячи їх через індивідуальний аналіз, який узгоджується з цілями організації.

1.4 Функціональні та нефункціональні вимоги

Функціональні та нефункціональні вимоги окреслюють основні можливості та основні характеристики, яким має відповідати система. Ці вимоги забезпечують корисність системи в управлінні фінансами та аналітичній обробці та визначають сферу розробки та тестування.

Функціональні вимоги:

1. управління та зберігання даних: система повинна дозволяти користувачам вводити фінансові дані, включаючи доходи, витрати, активи, зобов'язання та інші ключові показники ефективності (KPI);
2. інтеграція даних: система повинна інтегруватися із зовнішніми фінансовими системами та базами даних, такими як системи ERP або CRM, щоб отримати відповідну фінансову інформацію;
3. аналітичний модуль: має обробляти фінансові дані для створення звітів про такі показники, як прибутковість, ліквідність, платоспроможність і операційна ефективність;
4. звітність і візуалізація: користувачі повинні мати доступ до фінансових звітів, які можна створювати в режимі реального часу та на вимогу;
5. сповіщення та сповіщення користувачів: система повинна підтримувати автоматичні сповіщення та сповіщення про ключові фінансові події, такі як перевищення бюджету, низька ліквідність або проблеми з прибутковістю;
6. безпека та відповідність: система повинна гарантувати, що обробка даних відповідає фінансовим нормам, таким як GDPR і SOX, і надавати журнали аудиту для відстеження змін даних і доступу.

Нефункціональні вимоги:

1. продуктивність: система повинна забезпечувати швидку обробку даних і формування звіту, при цьому час відгуку для ключових функцій (наприклад, формування звіту) не перевищує 3 секунд;
2. надійність і доступність: система повинна підтримувати високий рівень надійності, гарантуючи, що критично важливі функції залишаються в робочому стані навіть у години пік;
3. масштабованість: система повинна масштабуватися горизонтально, щоб обробляти збільшений обсяг даних і запити користувачів у міру зростання організації;
4. юзабіліті: інтерфейс користувача має бути інтуїтивно зрозумілим, із чіткою навігацією та інструкціями, що дозволяє нетехнічним користувачам легко керувати системою;
5. безпека даних: система повинна використовувати шифрування для даних у дорозі та в спокої, забезпечуючи захист фінансової інформації від несанкціонованого доступу;
6. ремонтопридатність: системний код має бути модульним і добре задокументованим, щоб полегшити оновлення та модифікації;
7. відповідність і перевірка: система має відповідати галузевим стандартам і нормативним вимогам щодо обробки фінансових даних.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) — це мова моделювання, яка стандартизує візуалізацію, специфікацію, конструкцію та документацію як структури, так і поведінки програмних систем. Будучи основоположним інструментом об’єктно-орієнтованого програмування, UML надає схему розробки програмного забезпечення, яке допомагає розробникам і зацікавленим сторонам розуміти, аналізувати та передавати системні компоненти та зв’язки.

UML містить основні компоненти, які об’єднують різні види системи. Структурні діаграми використовуються для представлення статичних частин системи, показуючи класи, об’єкти та дані. Діаграма класів описує системні класи та їхні зв’язки, включаючи успадкування, асоціацію та залежності. Діаграми об’єктів пропонують знімок діаграм класів, щоб зафіксувати стан екземплярів у певний момент. Діаграми компонентів зосереджені на організації та залежностях програмних компонентів, таких як виконувані файли та бібліотеки. Діаграми розгортання зображують фізичний розподіл артефактів між вузлами, детально описуючи розташування серверів, конфігурації та мережеві структури[5].

Для моделювання динаміки системи UML використовує діаграми поведінки, які фіксують взаємодії, робочі процеси та стани. Діаграма варіантів використання показує, як актори (користувачі або зовнішні системи) взаємодіють з різними функціями або службами, демонструючи функціональність системи з точки зору кінцевого користувача. Діаграми послідовності ілюструють порядок повідомлень, якими обмінюються об’єкти для виконання певних завдань, висвітлюючи хід взаємодії з часом. Діаграми діяльності представляють робочі процеси або процеси, включаючи рішення та одночасні дії. Діаграми кінцевих автоматів показують, як об’єкт переходить

між станами на основі конкретних подій або умов, відстежуючи його реакції з часом. Діаграми взаємодії, як підмножина діаграм поведінки, деталізують потік керування та даних між об’єктами та зазвичай представлені діаграмами послідовності та зв’язку.

Значення UML у розробці програмного забезпечення полягає в його здатності забезпечувати ефективну візуалізацію та проектування складних систем. Надаючи стандартизовану структуру, UML сприяє узгодженому обміну даними між різними інструментами та платформами, допомагаючи командам узгоджувати дизайн системи та специфікації. Діаграми UML також служать вичерпною документацією для архітектури системи, сприяючи передачі знань, адаптації та обслуговуванню. Крім того, UML підтримує аналіз і тестування, дозволяючи розробникам виявляти потенційні проблеми на ранніх стадіях процесу розробки, що важливо для забезпечення якості.

Переваги UML виходять за межі технічних аспектів, оскільки він покращує комунікацію між зацікавленими сторонами, включаючи розробників, аналітиків і керівників проектів. Спрощуючи детальне планування, UML допомагає планувати системні компоненти та взаємодії, що призводить до більш ефективного проектування програмного забезпечення. Його модульна структура сприяє багаторазовому використанню, що дає змогу оптимізувати майбутні проекти, таким чином скорочуючи час і витрати на розробку.

Підсумовуючи, UML є цінним інструментом у розробці програмного забезпечення, який покращує процеси проектування, спілкування та документування. Він структурує складну системну документацію та підтримує надійний підхід до системного аналізу та планування, значно сприяючи ефективності та успіху програмних проектів.

Інтелектуальний аналіз даних — це процес виявлення значущих закономірностей, тенденцій і ідей у великих наборах даних за допомогою статистичних, математичних і машинного навчання. Відомий як «виявлення знань у базах даних» (KDD), він перетворює необроблені дані в цінні

відомості, які сприяють ухваленню рішень і вирішенню проблем у різноманітних сферах, як-от бізнес, охорона здоров'я, фінанси та маркетинг. Виявляючи закономірності, організації можуть створювати ефективніші стратегії, покращувати результати та адаптуватися до мінливих потреб і вимог.

Процес інтелектуального аналізу даних починається зі збору відповідних даних, які часто беруться з кількох джерел, таких як бази даних, журнали та записи транзакцій. Після збору дані проходять очищення та попередню обробку для усунення невідповідностей, заповнення відсутніх значень і стандартизації форматів, гарантуючи їх точність і придатність для аналізу. Цей крок має вирішальне значення для створення надійної основи для виявлення корисних ідей[5].

Перетворення та редукція даних також є ключовими етапами, на яких дані перетворюються у готові для аналізу формати, наприклад, шляхом кодування категоріальних змінних або масштабування числових значень. Методи зменшення розмірності, такі як аналіз головних компонентів (PCA), можуть бути застосовані для спрощення складних наборів даних, збереження важливої інформації при зменшенні обчислювального навантаження, тим самим підвищуючи ефективність аналізу та можливість інтерпретації.

Суть інтелектуального аналізу даних полягає в застосуванні алгоритмів для визначення закономірностей, кореляцій і тенденцій. Ці методи відрізняються залежно від бажаного результату. Класифікація, наприклад, призначає мітки на основі минулих прикладів і часто використовується для виявлення спаму. Кластеризація групує подібні точки даних і широко використовується в сегментації клієнтів. Вивчення правил асоціації виявляє зв'язки між змінними, як у аналізі ринкового кошика, а регресійний аналіз прогнозує числові результати, наприклад прогнози продажів.

Оцінка та інтерпретація забезпечують надійність цих шаблонів. Такі методи, як перехресна перевірка, перевіряють точність тестової моделі, тоді як інтерпретація дозволяє організаціям узгоджувати висновки з дієвими

стратегіями. Візуалізація за допомогою діаграм, графіків або інформаційних панелей робить цю інформацію доступною, полегшуючи розуміння та подальше прийняття рішень. У деяких випадках моделі можуть бути інтегровані безпосередньо в бізнес-процеси, забезпечуючи аналіз у реальному часі або автоматизовану підтримку прийняття рішень.

Інтелектуальний аналіз даних використовує різні техніки та алгоритми. Дерева рішень моделюють шляхи класифікації та регресії, пропонуючи інтуїтивно зрозумілі структури. K-Nearest Neighbors (KNN) призначає мітки на основі сусідніх точок даних і поширений у системах рекомендацій. Нейронні мережі, особливо в глибокому навчанні, є цінними для розпізнавання складних образів. Методи аналізу правил асоціації, такі як Apriori та FP-Growth, допомагають визначити закономірності у великих наборах даних, що є важливим для аналізу ринкового кошика. Алгоритми кластеризації, такі як K-середні та ієрархічна кластеризація, підтримують сегментацію клієнтів і виявлення аномалій.

Програми інтелектуального аналізу даних охоплюють управління взаємовідносинами з клієнтами, де він підтримує персоналізований маркетинг і задоволеність клієнтів, до виявлення шахрайства, діагностики охорони здоров’я, оптимізації виробництва та аналізу соціальних мереж для управління брендом. Незважаючи на свої переваги, інтелектуальний аналіз даних також стикається з проблемами, такими як керування шумними даними, масштабованість і забезпечення точності моделі. Етичні міркування, зокрема конфіденційність і безпека даних, залишаються першочерговими, оскільки інтелектуальний аналіз даних часто включає конфіденційну інформацію.

У сучасному бізнес-ландшафті інтелектуальний аналіз даних став неоціненним, дозволяючи організаціям перетворювати величезні сховища даних у стратегічні знання. Ця можливість підтримує швидше прийняття рішень на основі даних, зниження витрат і підвищення продуктивності, дозволяючи компаніям впроваджувати інновації, передбачати тенденції та

отримувати конкурентні переваги, глибоко розуміючи потреби та поведінку клієнтів.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це тип інструменту Уніфікованої мови моделювання (UML), який наочно демонструє, як користувачі та системи взаємодіють у програмному середовищі. Відображаючи основні функції, доступні користувачам, він забезпечує високорівневе уявлення про те, що пропонує система, показуючи завдання, які користувачі можуть виконати, і фіксуючи загальну поведінку системи з точки зору користувача, не зосереджуючись на технічних деталях.

Основні елементи на діаграмі варіантів використання включають акторів, варіанти використання та зв’язки. Акторами зазвичай є зовнішні сутності, такі як окремі користувачі, інші системи або пристрої, які взаємодіють із системою. Кожен учасник, представлений у вигляді фігурок, має певну роль, яка диктує їм доступ до різноманітних функцій у системі. Приклади можуть включати "Клієнт", "Адміністратор" або "Обробник платежів". Варіанти використання представляють дії або послуги, які система надає цим учасникам, і зображені у вигляді овалів, наприклад «Розмістити замовлення» або «Створити звіт». Кожен варіант використання відображає функціональні можливості системи, як їх розуміє користувач, підкреслюючи, чого досягає система, а не докладно описує задіяні процеси.

Відносини на діаграмі варіантів використання ілюструють, як різні учасники взаємодіють із певними функціями. Лінія асоціації з’єднує акторів із відповідними варіантами використання, показуючи пряму взаємодію. Деякі варіанти використання можуть включати інші як частину ширшої функціональності, показані пунктирною лінією, що вказує на зв’язок «включити». Якщо варіант використання має умовну функціональність, наприклад параметр, який застосовується лише в певних ситуаціях,

використовується зв’язок «розширення». У випадках, коли існує ієрархія, наприклад, коли спеціалізований актор або варіант використання успадковує більш загальний, застосовується зв’язок узагальнення.

Спроєктована діаграма прецедентів представлена на рис.3.

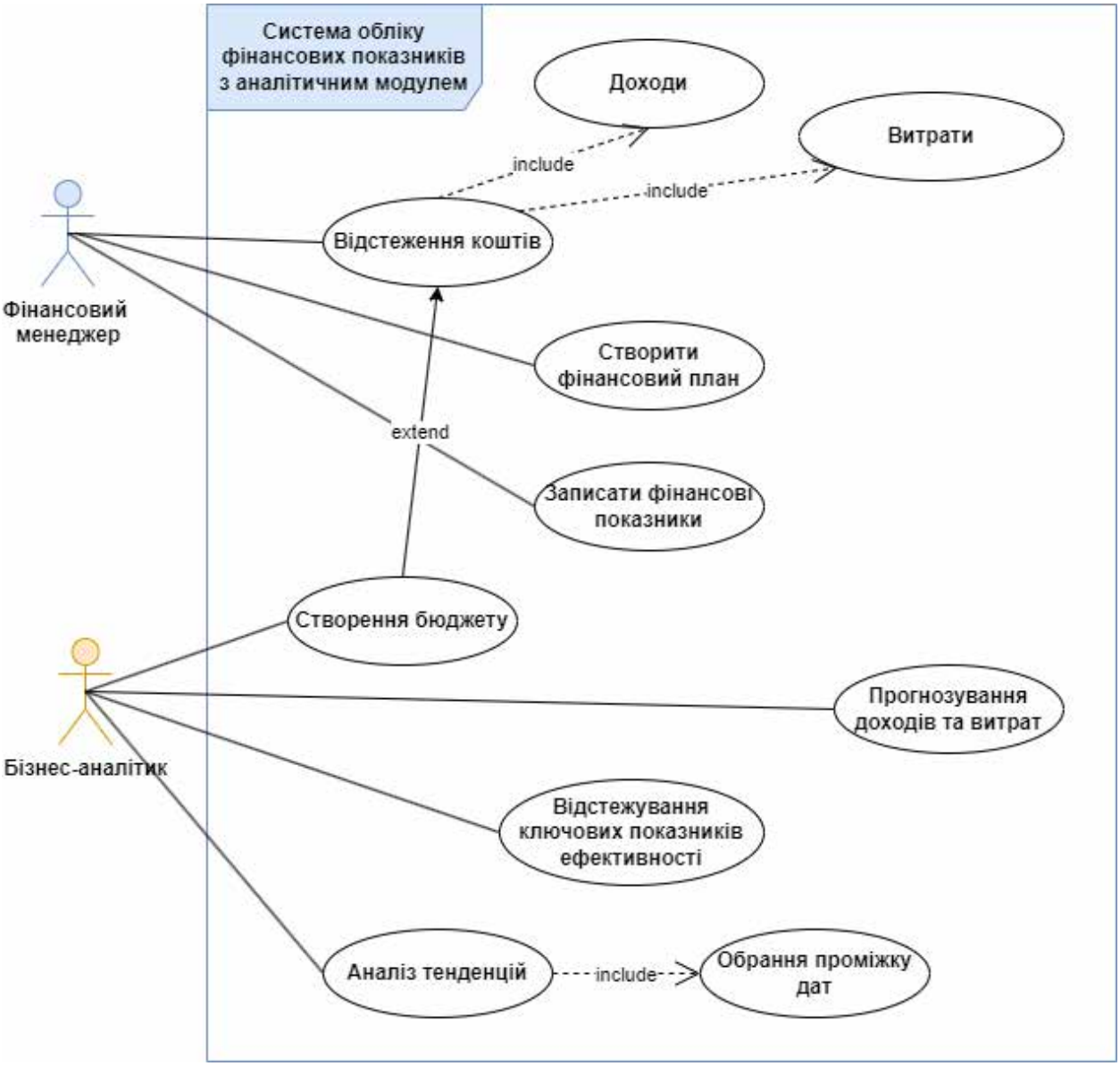

Рис. 3 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Фінансовий менеджер”;
- “Бізнес-аналітик”.

Актор «Фінансовий менеджер» включає такі прецеденти:

- відстеження коштів;
- доходи;

- витрати;
- створити фінансовий план;
- записати фінансові показники.

Актор «Бізнес-аналітик» включає такі прецеденти:

- створення бюджету;
- прогнозування доходів та витрат;
- аналіз тенденції;
- відстеження ключових показників ефективності.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Назва прецеденту: Створення фінансового плану

Актор: фінансовий менеджер

Опис: Фінансовий менеджер має на меті створити комплексний фінансовий план, який буде керувати бюджетом організації, прогнозуванням і фінансовою стратегією на майбутній фінансовий рік. Фінансовий план охоплюватиме прогнози надходжень, розподіл витрат, інвестиційні стратегії та оцінку ризиків для забезпечення фінансової стабільності та зростання.

Передумови: Фінансовий менеджер має доступ до системи обліку фінансових результатів. Відповідні історичні фінансові дані доступні та доступні в системі. Фінансовий менеджер отримав вказівки та цілі від вищого керівництва щодо процесу фінансового планування.

Основний потік:

1. фінансовий менеджер отримує доступ до системи обліку фінансових результатів, використовуючи свої облікові дані;
2. після входу менеджер обирає на головній панелі модуль «Фінансове планування»;
3. керівник натискає кнопку «Створити новий фінансовий план», яка відкриває новий шаблон фінансового плану;
4. фінансовий менеджер вводить таку основну інформацію: Назва плану: описова назва фінансового плану. Фінансовий рік: рік,

який охоплює фінансовий план. Цілі: ключові цілі, яких має досягти план (наприклад, зростання доходів, скорочення витрат);

5. система автоматично отримує відповідні історичні фінансові дані, включаючи минулі звіти про доходи, витрати та грошові потоки, щоб забезпечити контекст для нового плану;
6. менеджер вводить прогнозовані показники доходу на основі історичних тенденцій, аналізу ринку та очікуваного зростання продажів;
7. керівник також може встановлювати різні сценарії (наприклад, оптимістичний, песимістичний) для прогнозів доходу;
8. фінансовий менеджер розподіляє прогнозовані витрати для різних відділів, включаючи постійні та змінні витрати, зарплати та операційні витрати;
9. система надає розбивку минулих витрат, щоб допомогти приймати обґрунтовані рішення;
10. менеджер описує потенційні інвестиції, такі як нові проекти, придбання обладнання або маркетингові ініціативи, а також відповідні витрати та очікувану віддачу;
11. фінансовий менеджер визначає потенційні ризики, які можуть вплинути на фінансовий план, наприклад, ринкові коливання або нормативні зміни;
12. керівник окреслює стратегії пом’якшення цих ризиків;
13. після введення всіх необхідних даних фінансовий менеджер переглядає фінансовий план на точність і здійсненність;
14. система може надавати аналітику та візуалізацію, щоб допомогти оцінити життєздатність плану;
15. менеджер за потреби вносить корективи на основі інформації, наданої системою;
16. задовольнявшись фінансовим планом, менеджер зберігає

документ у системі;

17. фінансовий менеджер подає фінансовий план на затвердження вищому керівництву;
18. система повідомляє фінансового менеджера про подання, і план потрапляє в чергу на перегляд;
19. після встановленого періоду керівник отримує відгук або схвалення від вищого керівництва, що дозволяє вносити будь-які необхідні зміни.

Альтернативні потоки:

Якщо фінансовий менеджер стикається з проблемами під час отримання історичних даних, система запропонує їм усунути несправність або вручну ввести необхідні дані.

Якщо фінансовому менеджеру потрібна додаткова підтримка, він може отримати доступ до довідкових ресурсів або проконсультуватися з фінансовим аналітиком за допомогою функції обміну повідомленнями системи.

Розглянемо інший прецедент.

Назва прецеденту: Прогнозування доходів і витрат

Актор: бізнес-аналітик

Опис: Бізнес-аналітику доручено спрогнозувати доходи та витрати організації на майбутній фінансовий період. Цей процес прогнозування використовуватиме історичні фінансові дані, ринкові тенденції та внутрішню бізнес-аналітику для надання точних прогнозів, які будуть інформувати для планування бюджету та прийняття стратегічних рішень.

Передумови: Бізнес-аналітик має доступ до системи обліку фінансових результатів. Історичні фінансові дані, включно з попередніми звітами про доходи та витрати, доступні та актуальні в системі. Відповідний аналіз ринку та економічні показники доступні для ознайомлення.

Основний потік:

1. бізнес-аналітик входить у систему обліку фінансових

результатів, використовуючи свої облікові дані;

2. після входу в систему аналітик переходить до модуля «Прогнозування» з головної інформаційної панелі;
3. аналітик натискає опцію «Створити новий прогноз», яка відкриває новий шаблон прогнозування;
4. аналітик вводить важливі деталі;
5. система автоматично отримує відповідні історичні дані про доходи та витрати, щоб забезпечити основу для прогнозу;
6. аналітик переглядає ці дані на предмет тенденцій і аномалій, які можуть вплинути на майбутні прогнози;
7. бізнес-аналітик вивчає поточні ринкові умови та економічні показники, що стосуються галузі організації;
8. система може надавати інструменти для інтеграції зовнішніх ринкових даних або звітів для більш глибокого аналізу;
9. аналітик вводить прогнозовані показники доходу на основі історичних тенденцій, очікуваного зростання продажів і ринкових умов;
10. можна створити різні сценарії (наприклад, найкращий випадок, найгірший випадок) для візуалізації потенційних результатів за різних обставин;
11. бізнес-аналітик розподіляє прогнозовані витрати, враховуючи як постійні, так і змінні витрати;
12. аналітик використовує минулі звіти про витрати як орієнтир і враховує будь-які очікувані зміни, такі як нові найми, розширення або заходи щодо скорочення витрат;
13. модуль прогнозування надає аналітичні інструменти, такі як регресійний аналіз або прогнозування часових рядів, для уточнення прогнозів;
14. аналітик застосовує ці інструменти для підвищення точності прогнозів доходів і витрат;

15. аналітик переглядає прогноз на послідовність і здійсненність, забезпечуючи відповідність прогнозів стратегічним цілям організації;
16. за необхідності вносяться коригування на основі аналітичних інструментів і будь-яких останніх подій у бізнесі;
17. задовольнявшись прогнозом, аналітик зберігає документ у системі;
18. система генерує вичерпний звіт із детальним описом прогнозованих доходів і витрат, включаючи візуалізації, такі як графіки та діаграми;
19. аналітик ділиться звітом про прогнозування з відповідними зацікавленими сторонами, включаючи фінансових менеджерів і вище керівництво, для зворотного зв’язку та обговорення;
20. система може сприяти цьому, надаючи опції для експорту звіту або прямого обміну ним на платформі;
21. після перегляду прогнозу зацікавленими сторонами бізнес-аналітик отримує відгук або запити на перегляд;
22. на основі цього відгуку аналітик вносить необхідні оновлення в прогноз.

Альтернативні потоки:

Якщо бізнес-аналітик стикається з проблемами під час отримання історичних даних, система запропонує їм усунути неполадки або вручну ввести необхідні дані.

Якщо аналітику потрібна додаткова інформація, він може проконсультуватися з членами фінансової команди або отримати доступ до відповідних галузевих звітів за допомогою інтегрованих ресурсів системи.

2.2.2 Діаграма класів. Діаграма класів є важливою частиною уніфікованої мови моделювання (UML), яка візуально представляє структуру програмної системи. Він демонструє різні класи в системі,

докладно описуючи їхні атрибути, методи та зв’язки, які їх з’єднують. Ця діаграма життєво важлива для розуміння взаємодії та співпраці між різними компонентами, що робить її важливим інструментом як на етапі проектування, так і на етапі документації розробки програмного забезпечення[7].

Кожен клас на діаграмі класів зображується у вигляді прямокутника, розділеного на три частини. Верхній розділ містить назву класу, тоді як середній розділ містить список його атрибутів, які часто включають типи даних. У нижньому розділі описано методи, пов’язані з цим класом, із детальним описом його функціональних можливостей.

Діаграми класів також ілюструють кілька типів зв'язків між класами. Асоціація представляє зв’язок між двома класами, вказуючи, що один клас використовує або взаємодіє з іншим. Цей зв’язок може бути як односпрямованим (одностороннім), так і двонаправленим (двостороннім). Агрегація, більш специфічний тип асоціації, ілюструє зв’язок ціле-частина, де частина може існувати незалежно від цілого. Наприклад, хоча університет може складатися з кількох кафедр, ці кафедри можуть існувати самі по собі.

Навпаки, композиція означає більш міцні відносини власності, де частина не може вижити незалежно від цілого. Прикладом може бути клас зі студентами; якщо клас видалено, студенти не можуть існувати в цьому контексті. Спадкування показує, що один клас (підклас) отримує атрибути та методи від іншого класу (суперкласу), представленого суцільною лінією з замкнутою порожнистою стрілкою, яка вказує на суперклас. Залежність ілюструє слабший зв’язок, вказуючи на те, що один клас покладається на

інший. Зміни в залежному класі можуть вимагати змін у класі, від якого він залежить.

Створена діаграма класів представлена на Рис.4.

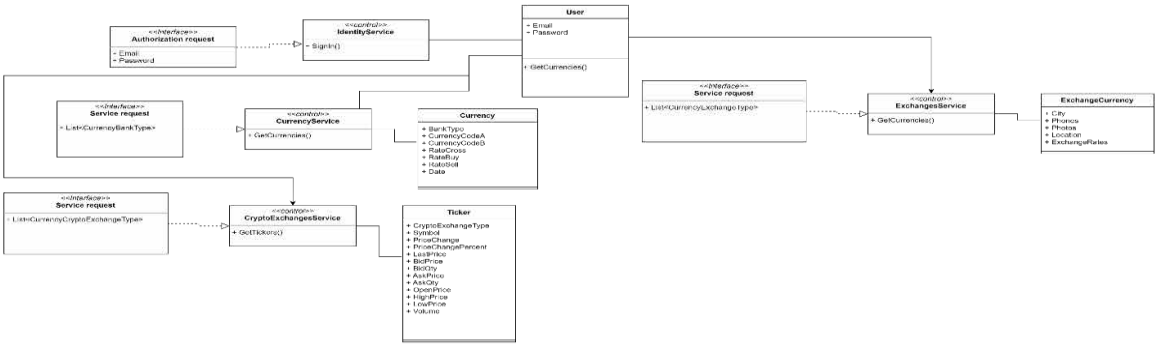

Рис. 4 Діаграма класів

2.2.3 Діаграма станів. Діаграма стану, також відома як діаграма кінцевого автомата, — це динамічна модель, яка використовується для представлення станів об’єкта та переходів між цими станами у відповідь на події чи умови. Цей тип діаграм є частиною Уніфікованої мови моделювання (UML) і особливо цінний для ілюстрації того, як об’єкт поводиться з часом, демонструючи його життєвий цикл від створення до знищення[8].

Основні компоненти діаграми стану включають стани, переходи, події та дії. Стан представляє певний стан або ситуацію протягом життєвого циклу об’єкта. Наприклад, у системі обліку фінансової ефективності стан «Очікує на затвердження» може означати, що фінансовий звіт очікує перевірки. Переходи зображуються у вигляді стрілок, що з'єднують стани, що представляють рух від одного стану до іншого. Ці переходи викликаються подіями, які можуть бути внутрішніми (наприклад,

закінчення таймера) або зовнішніми (наприклад, отримання введених даних користувачем).

Кожен перехід також може мати пов’язані з ним дії, які є завданнями, що виконуються у відповідь на подію. Наприклад, коли фінансовий звіт переходить із стану «Чернетка» в стан «Подано», дія може передбачати надсилання сповіщення фінансовому менеджеру. Діаграма також може ілюструвати початковий і кінцевий стани, вказуючи, де починається і закінчується життєвий цикл відповідно.

Діаграми станів надають кілька переваг у процесі розробки програмного забезпечення. Вони пропонують чітке та наочне уявлення про те, як поводиться об’єкт, полегшуючи розробникам і зацікавленим сторонам розуміння складних взаємодій. Ця ясність особливо важлива на етапі проектування, оскільки вона допомагає виявити потенційні проблеми та вдосконалити поведінку системи перед впровадженням.

Крім того, діаграми станів покращують спілкування в групах розробників, надаючи спільну мову для обговорення поведінки системи. Вони також можуть відігравати важливу роль у тестуванні та перевірці, гарантуючи, що всі можливі стани та переходи враховуються та функціонують належним чином.

Створена діаграма станів представлена на Рис.5.

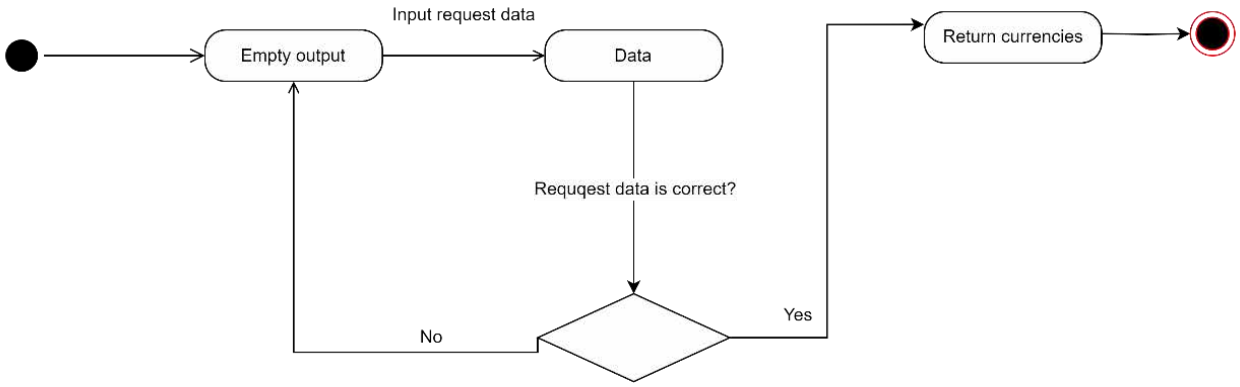

Рис. 5 Діаграма станів пошуку та замовлення товару

2.2.4 Діаграма послідовності. Діаграма послідовності — це тип діаграми взаємодії, який використовується в уніфікованій мові моделювання (UML) для представлення динамічної поведінки системи шляхом ілюстрації того, як об’єкти взаємодіють з часом. Ця діаграма зосереджена на послідовності повідомлень, якими обмінюються різні об’єкти або компоненти в конкретному сценарії, надаючи чітке уявлення про взаємодії, які відбуваються під час конкретного варіанту використання або процесу.

Ключові елементи діаграми послідовності включають лінії життя, повідомлення, блоки активації та зворотні повідомлення. Лінія життя представляє екземпляр класу або об’єкта, задіяного у взаємодії, зображеного у вигляді вертикальної пунктирної лінії, що тягнеться вниз. Кожна лінія життя пов’язана з певною сутністю, наприклад користувачем, системним компонентом або зовнішньою службою[8].

Повідомлення - це стрілки, намальовані між лініями життя, які вказують на зв'язок між об'єктами. Ці повідомлення можуть представляти виклики методів, відповіді або сигнали, і вони можуть відрізнятися за типом, включаючи синхронні виклики (де відправник очікує на відповідь) і асинхронні виклики (де відправник продовжує без очікування).

Поля активації, також відомі як фокус керування, показують тривалість часу, протягом якого об’єкт активний або обробляє повідомлення. Вони представлені у вигляді прямокутників уздовж лінії життя та допомагають уточнити, коли об’єкт бере участь у певній операції.

Діаграма послідовності часто включає зворотні повідомлення, представлені у вигляді пунктирних стрілок, що вказують назад на абонента, що вказує на те, що відповідь була отримана після відправлення

повідомлення. Це допомагає проілюструвати потік контролю та час відповідей.

Діаграми послідовності особливо корисні з кількох причин. Вони надають візуальне представлення порядку операцій, полегшуючи розуміння складних взаємодій і часу подій у системі. Ця ясність є безцінною для розробників і зацікавлених сторін під час аналізу поведінки системи, виявлення потенційних вузьких місць і забезпечення фіксації всіх необхідних взаємодій.

Наведена нижче діаграма послідовності (Рис. 6).

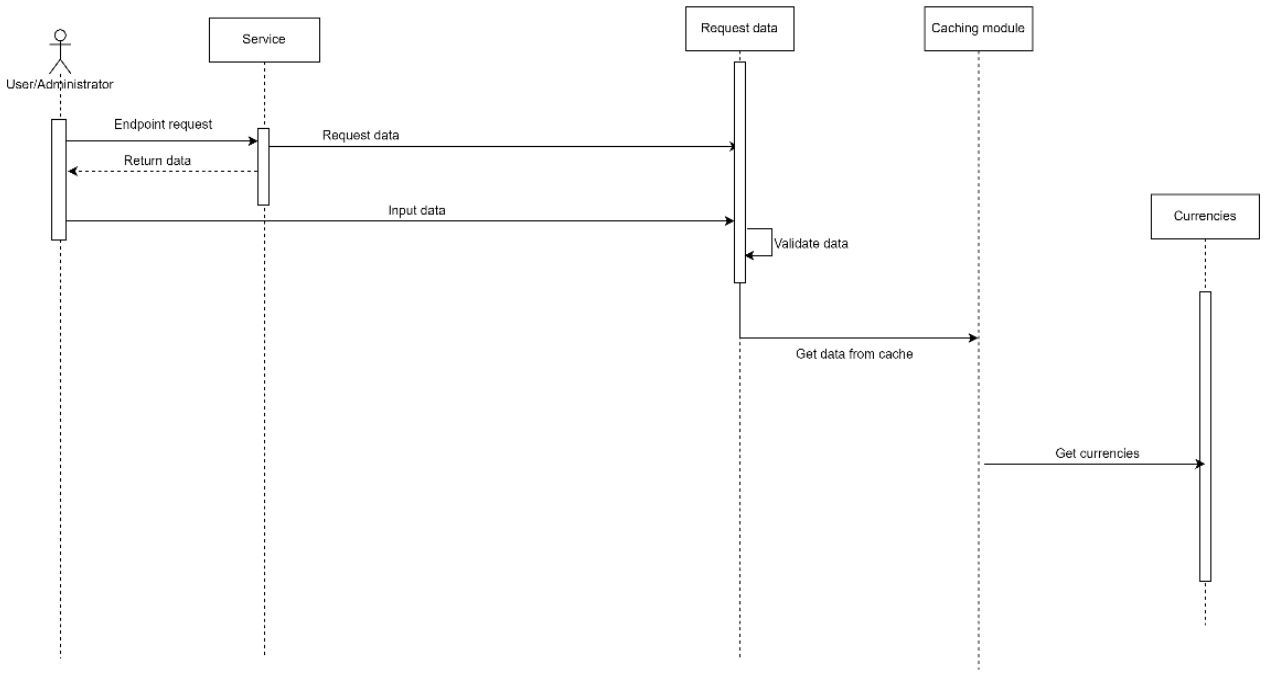

Рис. 6 Діаграма послідовності

2.3 Огляд інструментів для реалізації завдань Data Mining

У контексті реалізації завдань інтелектуального аналізу даних для системи обліку фінансової ефективності з аналітичним модулем доступні різні інструменти, які обслуговують різні аспекти інтелектуального аналізу даних, включаючи попередню обробку даних, моделювання, аналіз і візуалізацію.

Нижче наведено огляд кількох відомих інструментів і платформ, які зазвичай використовуються для аналізу даних.

RapidMiner — це наукова платформа з відкритим вихідним кодом, яка надає комплексний пакет для підготовки даних, машинного навчання та прогнозної аналітики. Він пропонує зручний інтерфейс із візуальним дизайнером робочого процесу, що дозволяє користувачам створювати моделі інтелектуального аналізу даних без глибоких знань програмування. RapidMiner підтримує широкий спектр алгоритмів для класифікації, регресії та кластеризації, що робить його придатним для завдань фінансового аналізу, таких як кредитний скоринг, оцінка ризиків і прогнозування[10].

KNIME (Konstanz Information Miner) — ще одна платформа з відкритим кодом, призначена для аналізу даних, звітності та інтеграції. Він має модульну архітектуру, яка дозволяє користувачам створювати конвеєри даних за допомогою візуального інтерфейсу. KNIME підтримує різні методи аналізу даних і може інтегруватися з R, Python та іншими мовами програмування. Його можливості включають попередню обробку даних, статистичний аналіз і машинне навчання, що робить його універсальним інструментом для аналізу фінансової ефективності.

Weka — це популярний пакет програмного забезпечення для аналізу даних із відкритим кодом, який пропонує набір алгоритмів машинного навчання для завдань аналізу даних. Він надає інструменти для попередньої обробки даних, класифікації, регресії, кластеризації та аналізу правил асоціації. Weka особливо корисна для освітніх цілей і досліджень, а її графічний інтерфейс користувача робить її доступною для користувачів, які, можливо, не знайомі з програмуванням. Його можливості можна використовувати для фінансового прогнозування та аналізу тенденцій.

SAS Enterprise Miner — це комерційний інструмент інтелектуального аналізу даних, який надає розширені аналітичні можливості, включаючи прогнозне моделювання, машинне навчання та інтелектуальний аналіз даних. Він пропонує комплексне середовище для підготовки даних, дослідження та

розробки моделей. SAS широко використовується у фінансовій галузі для таких завдань, як управління ризиками, виявлення шахрайства та сегментація клієнтів завдяки своїй надійності та масштабованості.

IBM SPSS Modeler — це потужний інструмент інтелектуального аналізу даних і прогнозної аналітики, який дозволяє користувачам створювати прогнозні моделі за допомогою візуального інтерфейсу. Він підтримує різні методи інтелектуального аналізу даних, включаючи класифікацію, регресію та кластеризацію, а також пропонує інтеграцію з великими джерелами даних і можливостями підготовки даних. Його сильна сторона полягає в здатності обробляти великі набори даних і передових інструментах статистичного аналізу, що робить його придатним для складного фінансового аналізу.

Python став домінуючою мовою в галузі науки про дані та машинного навчання завдяки своїм великим бібліотекам. Такі бібліотеки, як Pandas, полегшують маніпулювання даними та попередню обробку, а Scikit-learn пропонує широкий спектр алгоритмів машинного навчання для класифікації, регресії та кластеризації. TensorFlow і Keras — це потужні бібліотеки для глибокого навчання, які підходять для складніших завдань фінансового прогнозування та розпізнавання образів. Універсальність Python дозволяє налаштовувати та інтегрувати в існуючі фінансові системи[11].

R — це мова статистичного програмування, яка широко використовується в аналізі та інтелектуальному аналізі даних. Він пропонує численні пакети для різноманітних завдань інтелектуального аналізу даних, включаючи caret для навчання моделі, dplyr для маніпулювання даними та ggplot2 для візуалізації даних. Сильні статистичні можливості R роблять його особливо корисним для поглибленого фінансового аналізу та моделювання, дозволяючи аналітикам отримувати висновки з фінансових даних.

Незважаючи на те, що Tableau є переважно інструментом візуалізації даних, Tableau може відігравати важливу роль у видобутку даних, дозволяючи користувачам досліджувати та візуалізувати тенденції та шаблони даних в інтерактивному режимі. Його здатність підключатися до різних джерел даних

і створювати інформаційні панелі допомагає зацікавленим сторонам у фінансовому секторі розуміти ключові показники ефективності та приймати рішення на основі даних.

Вибір правильного інструменту для завдань інтелектуального аналізу даних у системі обліку фінансової ефективності залежить від кількох факторів, зокрема конкретних вимог до аналізу, обсягу та складності даних, технічного досвіду користувачів і бюджетних обмежень. Можна також розглянути комбінацію цих інструментів, щоб використати їхні відповідні сильні сторони, забезпечуючи всебічний аналіз даних і формування розуміння в контексті управління фінансовою ефективністю[12].

2.4 Структура джерела інформації для інтелектуального аналізу

При проектуванні джерела інформації для інтелектуального аналізу в системі обліку фінансових показників, яка включає аналітичний модуль, дуже важливо встановити структурований підхід, який сприяє ефективному управлінню даними, пошуку та аналізу. Це джерело інформації охоплюватиме різні типи даних, включаючи як внутрішні, так і зовнішні джерела.

Внутрішні дані укладатимуться з основних фінансових документів, таких як баланси, звіти про прибутки та збитки та звіти про рух грошових коштів, а також детальні записи про операції, пов’язані з продажами, покупками та витратами. Історичні дані про продуктивність разом із бюджетами та прогнозами також відіграватимуть важливу роль у забезпеченні контексту для аналізу. Навпаки, зовнішні джерела даних можуть включати ринкову статистику, фінансову інформацію про конкурентів, галузеві звіти та дані про дотримання нормативних вимог, збагачуючи загальні аналітичні можливості.

Важливим компонентом цієї структури є ефективна категоризація даних. Фінансові показники будуть класифіковані за кількома ключовими

областями, такими як коефіцієнти ліквідності, такі як коефіцієнт поточної ліквідності та швидкості, коефіцієнти прибутковості, коефіцієнти левериджу та коефіцієнти ефективності. Крім того, ключові показники ефективності (KPI) будуть важливими, охоплюючи такі показники, як швидкість зростання доходу, операційний дохід, EBITDA та вартість залучення клієнтів. Нефінансові показники, включно з показниками задоволеності клієнтів і плинністю кадрів, нададуть більш повне уявлення про продуктивність.

Обробка та зберігання даних є фундаментальними для забезпечення безперебійної роботи джерела інформації. Впровадження сховища даних дозволить створити централізоване сховище для зберігання як структурованих, так і неструктурованих даних, отриманих із різних платформ. Застосування процесів ETL (Extract, Transform, Load) буде необхідним для підтримки якості та цілісності даних. Для керування базами даних поєднання реляційних баз даних, таких як SQL Server або MySQL, і опцій NoSQL, таких як MongoDB, може ефективно вмістити різноманітні типи даних[13].

Аналітичний модуль міститиме набір інструментів для аналізу даних. Використання програм статистичного аналізу, як-от R або Python, разом із інструментами бізнес-аналітики, як-от Tableau та Power BI, дозволить глибоко розуміння фінансових даних. Фреймворки машинного навчання, такі як TensorFlow і Scikit-learn, сприятимуть прогнозній аналітиці, підвищуючи здатність прогнозувати тенденції та поведінку. Різні методи моделювання, включаючи аналіз часових рядів, регресійний аналіз і кластеризацію, сприятимуть прийняттю обґрунтованих рішень.

Візуалізація даних необхідна для ефективного спілкування та розуміння. Інтерактивні інформаційні панелі можуть представляти фінансові показники та KPI у зручному для користувача форматі, що дозволяє різним зацікавленим сторонам, таким як фінансові менеджери, аналітики та керівники, налаштовувати свої погляди відповідно до їхніх конкретних потреб. Крім того, можливості автоматизованого звітування оптимізують створення регулярних фінансових звітів, тоді як функції спеціального

звітування дозволять аналізувати дані на вимогу.

Безпека та відповідність залишаються першочерговими в дизайні джерела інформації. Впровадження надійних заходів контролю доступу забезпечить збереження конфіденційності даних, а керування доступом на основі ролей дозволить лише авторизованому персоналу переглядати конфіденційну інформацію. Крім того, ведення контрольних журналів забезпечить підзвітність шляхом відстеження доступу до даних і змін. Відповідність нормативним стандартам, таким як GDPR і SOX, буде критично важливою, що вимагає регулярних аудитів для підтвердження дотримання цих правил.

Отже, ретельна документація та стратегії управління знаннями підвищать зручність використання системи. Вичерпні посібники та посібники користувача детально описуватимуть функціональні можливості та процеси системи, доповнені навчальними матеріалами для адаптації та постійного навчання користувачів. Спеціальна база знань слугуватиме сховищем ідей, передового досвіду та аналітичних висновків, сприяючи культурі обміну знаннями між користувачами та надаючи тематичні дослідження для довідки.

Таким чином, цей структурований підхід до створення джерела інформації для системи обліку фінансових показників спрямований на полегшення ефективного збору, обробки та аналізу фінансових даних. Інтегруючи різні джерела даних і аналітичні інструменти, організації можуть отримати цінну інформацію, покращити процес прийняття рішень і підвищити загальну фінансову ефективність[13].

В ході роботи було створено сховище даних, що дозволить проводить аналіз в різних розрізах. Структура сховища даних представлена на рисунку 7.

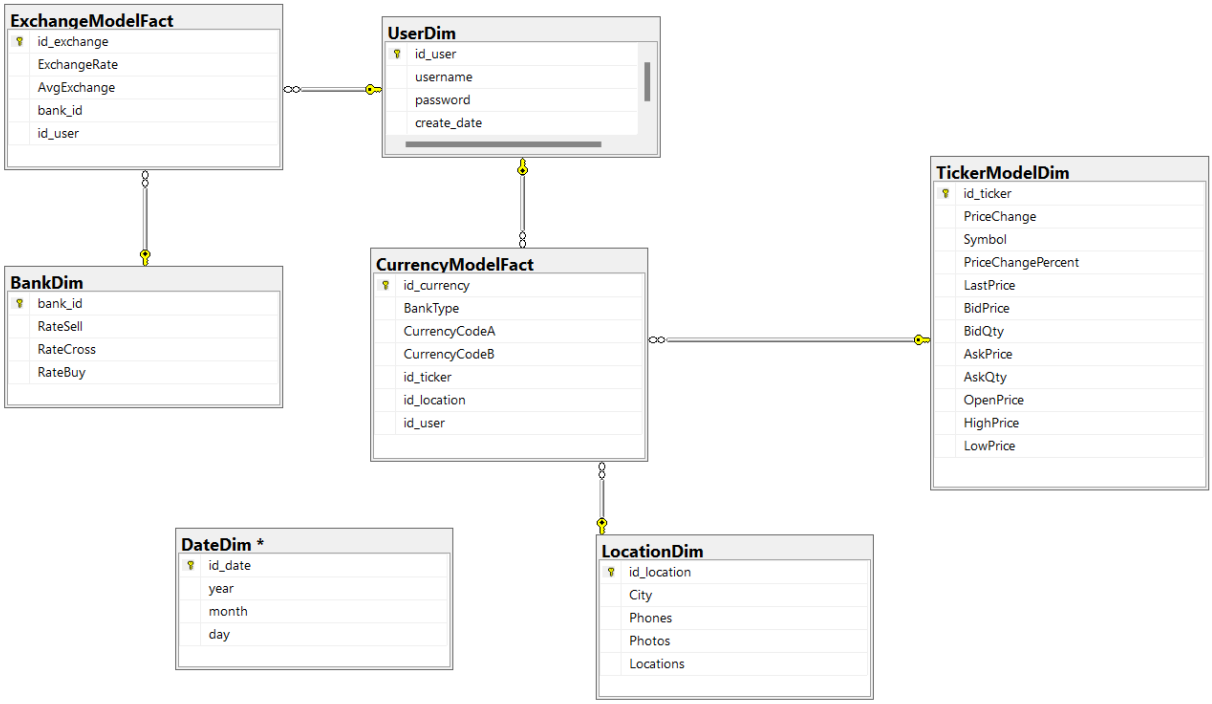

Рис. 7 Сховище даних

Вимір UserDim зберігає дані про користувачів системи.

- id_user (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор користувача;
- username (VARCHAR(50)) - ім'я користувача;
- password (VARCHAR(100)) — пароль користувача, який зазвичай зберігається в зашифрованому вигляді;
- create_date (DATETIME) – дата створення облікового запису користувача.

Вимір TickerModelDim зберігає інформацію про тикери (ціни акцій та інших фінансових інструментів).

- id_ticker (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор тикера;
- PriceChange (DECIMAL (18, 2)) - Зміна ціни;
- Symbol (VARCHAR(10)) – символ тикера (наприклад, "AAPL" для Apple);

- PriceChangePercent (DECIMAL(5, 2)) — відсоткова зміна ціни;
- LastPrice (DECIMAL(18, 2)) - остання ціна;
- BidPrice (DECIMAL (18, 2)) - ціна покупки;
- BidQty (DECIMAL(18, 2)) - кількість пропозицій на купівлю;
- AskPrice (DECIMAL(18, 2)) — ціна продажу;
- AskQty (DECIMAL(18, 2)) - кількість пропозицій на продаж;
- OpenPrice (DECIMAL (18, 2)) - ціна відкриття;
- HihgPrice (DECIMAL(18, 2)) - максимальна ціна за період;
- LowPrice (DECIMAL (18, 2)) - мінімальна ціна за період.

Таблиця фактів CurrencyModelFact зберігає дані про валютні курси.

- id_currency (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор валюти;
- CurrencyCodeA (VARCHAR(3)) — код першої валюти (наприклад, USD);
- CurrencyCodeB (VARCHAR(3)) — код другої валюти (наприклад, EUR);
- id_ticker (INT, FOREIGN KEY) - зв'язок з таблицею TickerModelDim, яка зберігає інформацію про валюту.

Вимір BankDim зберігає інформацію про курси валют у різних банках.

- bank_id (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор банку;
- RateSell (DECIMAL(18, 2)) - курс продажу валюти;
- RateCross (DECIMAL (18, 2)) - крос-курс;
- RateBuy (DECIMAL(18, 2)) - курс купівлі валюти.

Таблиця фактів ExchangeModelFact зберігає дані про обмінні курси, пов'язані з користувачами.

- id_exchange (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор обміну;
- ExchangeRate (DECIMAL (18, 2)) - обмінний курс;
- AvgExchange (DECIMAL (18, 2)) – середній курс обміну;
- id_user (INT, FOREIGN KEY) — зв'язок із таблицею UserDim, що вказує на користувача.

Вимір LocationDim зберігає дані про розташування, такі як офіси або філії.

- id_location (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор розташування;
- City (VARCHAR(50)) – місто;
- Phones (VARCHAR(100)) — телефони розташування;
- Photos (VARCHAR(MAX)) — фотографії розташування;
- Locations (VARCHAR(MAX)) — докладний опис розташування.

Вимір DateDim зберігає інформацію про дати для аналітичних цілей.

- id_date (INT, PRIMARY KEY, IDENTITY) – унікальний ідентифікатор дати;
- year (INT) – рік;
- month (INT) – місяць;
- day (INT) - день.

Запити, по створенню таблиць-вимірів та таблиць-фактів написано на SQL та представлено у додатку А.

2.5 OLAP-технології

Технології онлайн-аналітичної обробки (OLAP) є важливими компонентами систем обліку фінансової ефективності, особливо тих, що містять аналітичні модулі. Ці технології дозволяють аналізувати комплексні

дані, дозволяючи користувачам інтерактивно досліджувати й аналізувати багатовимірні дані. Такі можливості мають вирішальне значення для фінансових аналітиків і осіб, які приймають рішення, яким необхідно отримати інформацію з показників ефективності, тенденцій і прогнозів для інформування про стратегічне планування.

В основі OLAP лежить концепція багатовимірного моделювання даних, яка організовує інформацію в структуру даних, відому як куб даних. Цей куб містить різні виміри, такі як час, географія та фінансові категорії, що полегшує всебічний аналіз зв’язків і закономірностей у фінансових даних. Наприклад, куб OLAP, розроблений для фінансових показників, може включати такі параметри, як потоки доходів, категорії витрат, періоди часу та географічні регіони, що дозволяє проводити ретельний аналіз фінансового стану компанії[14].

MOLAP, або багатовимірний OLAP, зберігає дані в багатовимірному масиві, що забезпечує швидкий пошук і аналіз. Цей формат відмінно справляється зі складними агрегаціями та обчисленнями, що робить його особливо ефективним, коли швидкість має значення. Однак MOLAP може зіткнутися з проблемами в управлінні дуже великими наборами даних порівняно з аналогами.

ROLAP, або реляційний OLAP, працює безпосередньо з реляційними базами даних, перетворюючи запити OLAP в оператори SQL для доступу до базових даних. Цей підхід забезпечує підвищену масштабованість і гнучкість, що дозволяє організаціям аналізувати великі набори даних без обмежень багатовимірного зберігання. Незважаючи на свої переваги, ROLAP іноді може призводити до зниження продуктивності запитів через необхідність пошуку даних і обчислень у реальному часі.

HOLAP, або гібридний OLAP, поєднує в собі переваги MOLAP і ROLAP. Це дозволяє докладним даним зберігатися в реляційних базах даних,

використовуючи багатовимірне сховище для агрегованих даних. Ця гібридна структура надає користувачам швидкий доступ до узагальненої інформації, а також дозволяє детально аналізувати більші набори даних, що робить її універсальним вибором для оцінки фінансової ефективності.

Технології OLAP мають кілька функцій, які значно розширюють можливості аналізу даних. Наприклад, функції деталізації та згортання дозволяють користувачам ефективно переміщатися по ієрархіях даних. Деталізація дозволяє отримати більш детальний перегляд, тоді як згортання збирає дані для більш високого рівня аналізу, таким чином полегшуючи дослідження фінансових показників на різних рівнях деталізації.

Крім того, OLAP підтримує обчислені вимірювання, що дозволяє користувачам отримувати нові показники з існуючих даних. Ця функція особливо корисна для генерування спеціальних фінансових коефіцієнтів або показників ефективності, адаптованих до потреб організації[15].

Підсумовуючи, технології OLAP є невід’ємною частиною систем обліку фінансової діяльності з аналітичними модулями. Спрощуючи багатовимірний аналіз і пропонуючи функції, які сприяють поглибленому дослідженню даних, OLAP дає можливість організаціям перетворювати фінансові дані в практичну інформацію. Ця аналітична здатність необхідна для прийняття обґрунтованих рішень і стратегічного планування, що дозволяє підприємствам ефективно орієнтуватися в складних фінансових показниках.

3 РОЗРОБКА СИСТЕМИ

3.1 Логічна модель даних

ERwin — це потужний інструмент моделювання даних, який широко використовується для проектування, візуалізації та керування структурами

даних. Він відіграє вирішальну роль у розробці та управлінні базами даних, особливо в програмах корпоративного рівня. Цей інструмент полегшує створення діаграм Entity-Relationship (ER), які графічно представляють сутності даних, їхні атрибути та зв'язки між ними. Такі візуальні представлення необхідні для розуміння складних архітектур даних і забезпечення цілісності даних у системах.

Однією з ключових особливостей ERwin є його здатність підтримувати кілька платформ баз даних, включаючи Oracle, SQL Server, MySQL та інші. Ця сумісність дозволяє організаціям розробляти та керувати базами даних відповідно до їхніх конкретних потреб, незалежно від базової технології баз даних. ERwin також надає можливості зворотного проектування, дозволяючи користувачам імпортувати існуючі бази даних і створювати відповідні моделі даних. Ця функція особливо корисна для організацій, які хочуть документувати або аналізувати застарілі системи.

ERwin сприяє спільній роботі між фахівцями з обробки даних, пропонуючи контроль версій і командні функції. Користувачі можуть ділитися моделями, відстежувати зміни та співпрацювати в режимі реального часу, що підвищує продуктивність і гарантує узгодженість усіх зацікавлених сторін протягом усього процесу моделювання даних. Це середовище для спільної роботи має важливе значення для великих проектів із залученням кількох учасників[16].

Інструмент має зручний інтерфейс, який спрощує процес моделювання, дозволяючи користувачам легко створювати та змінювати моделі даних. Функція перетягування, інтуїтивно зрозумілі меню та візуальні посібники покращують роботу користувача, роблячи її доступною як для початківців, так і для досвідчених розробників моделювання даних.

ERwin підтримує розширені концепції моделювання даних, включаючи розмірне моделювання для сховищ даних і програм бізнес-

аналітики. Ця можливість дозволяє користувачам створювати схеми зірок і сніжинок, які є ключовими для організації даних таким чином, щоб оптимізувати продуктивність запитів і аналітичну обробку.

На додаток до своїх можливостей моделювання, ERwin пропонує функції для управління даними та відповідності. Організації можуть використовувати інструмент для документування походження даних, встановлення стандартів даних і забезпечення дотримання політики даних. Ця функція є важливою для підтримки якості даних і забезпечення відповідності таким нормам, як GDPR і HIPAA.

Логічна модель системи представлена на рис. 8.

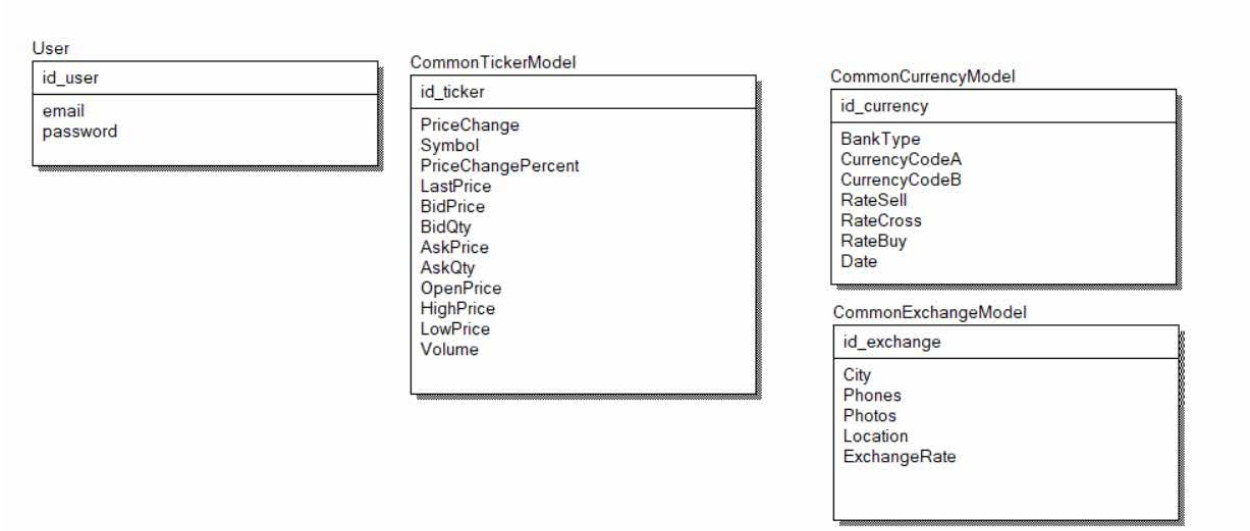

Рис. 8 ER-діаграма

3.2 Вибір системи управління базою даних та її реалізація

Система керування базами даних (СУБД) — це важливе програмне забезпечення, яке дозволяє користувачам ефективно створювати, керувати та організовувати бази даних. Діючи як посередник між користувачем і даними, СУБД спрощує процеси зберігання, пошуку та маніпулювання даними. Він широко використовується в різних секторах, включаючи бізнес, освіту та

дослідження, щоб гарантувати, що дані залишаються безпечними, доступними та точними.

СУБД можна класифікувати на кілька типів на основі їх архітектури та призначення. Системи управління реляційними базами даних (RDBMS) є одними з найпоширеніших. Вони структурують дані в таблицях, що складаються з рядків і стовпців, що дозволяє користувачам легко отримувати інформацію за допомогою мови структурованих запитів (SQL). RDBMS користується перевагою через свою надійність і здатність обробляти складні запити. Популярні приклади включають MySQL, PostgreSQL, Oracle Database і Microsoft SQL Server[17].

Інша категорія — бази даних NoSQL, призначені для керування неструктурованими або напівструктурованими даними. Ці системи пропонують гнучкість, дозволяючи зберігати дані без попередньо визначеної схеми, що робить їх особливо корисними для додатків, які мають справу з великими даними або вимагають обробки даних у реальному часі. Приклади баз даних NoSQL включають сховища документів, такі як MongoDB, сховища ключ-значення, такі як Redis, сховища сімейства стовпців, такі як Cassandra, і бази даних графів, такі як Neo4j.

Об’єктно-орієнтовані системи управління базами даних (OODBMS) зберігають дані як об’єкти, віддзеркалюючи принципи об’єктно-орієнтованого програмування. Цей підхід є перевагою для додатків, які потребують представлення складних зв’язків даних, таких як системи САПР або мультимедійні програми. Крім того, існують ієрархічні та мережеві моделі СУБД, які організовують дані у вигляді деревоподібних або графових структур відповідно. Хоча ці старіші моделі сьогодні використовуються рідше, вони заклали основу для сучасних систем баз даних.

Однією з істотних переваг СУБД є її здатність зберігати цілісність даних. Це гарантує, що інформація залишається точною та узгодженою

протягом тривалого часу, підтримується такими обмеженнями, як первинні та зовнішні ключі, які допомагають зберегти зв’язки між таблицями.

Безпека є ще одним важливим аспектом, оскільки СУБД пропонують різні механізми для захисту даних. Вони включають автентифікацію користувача, рівні авторизації та методи шифрування для захисту конфіденційної інформації від несанкціонованого доступу.

Крім того, СУБД надає функції для відновлення та резервного копіювання даних, що дозволяє користувачам відновлювати дані у разі збою або пошкодження. Ця можливість є важливою для підтримки безперервності бізнесу та захисту від втрати даних.

Крім того, СУБД підтримує одночасний доступ, що дозволяє кільком користувачам одночасно взаємодіяти з базою даних, не викликаючи конфліктів. Це особливо важливо для програм, які потребують обробки даних у реальному часі та співпраці між користувачами.

Таким чином, система керування базами даних є основоположним інструментом, який покращує керування даними шляхом оптимізації процесів, покращення цілісності та безпеки даних, а також полегшення ефективного пошуку та маніпулювання даними. Його різноманітні функціональні можливості роблять його важливим компонентом у різних галузях, зокрема в бізнесі, охороні здоров’я, фінансах і технологіях[19].

Вибір PostgreSQL, орієнтованого на систему обліку фінансових результатів з аналітичним модулем, є стратегічним рішенням, яке ґрунтується на кількох переконливих факторах.

PostgreSQL — це вдосконалена система керування реляційними базами даних із відкритим вихідним кодом, яка вирізняється своєю міцністю та надійністю. Однією з головних причин вибору PostgreSQL є його суворе дотримання стандартів SQL, що забезпечує сумісність і простоту

використання для розробників, які знайомі з реляційними базами даних. Ця сумісність дозволяє здійснювати ефективні запити та керування фінансовими даними, необхідними для системи бухгалтерського обліку, де точність і точність є найважливішими.

Багатий набір функцій PostgreSQL ще більше підвищує його привабливість. Він підтримує широкий спектр типів даних, включаючи JSON і XML, що полегшує обробку як структурованих, так і напівструктурованих даних. Ця універсальність особливо корисна для аналітичного модуля, якому може знадобитися обробка різноманітних джерел даних, що дозволяє здійснювати комплексний аналіз даних і звітність.

Крім того, PostgreSQL пропонує розширені аналітичні можливості за допомогою таких розширень, як PostGIS для аналізу геопросторових даних і вбудовану підтримку віконних функцій, які дозволяють виконувати складні обчислення в кількох рядках даних. Ці функції є безцінними для створення глибоких фінансових звітів і прогнозів, що дає можливість бізнес-аналітикам приймати рішення на основі даних.

Іншою значною перевагою PostgreSQL є акцент на цілісності та узгодженості даних. Він використовує надійну відповідність ACID (атомність, узгодженість, ізоляція, довговічність), що гарантує надійну обробку всіх транзакцій. Цей рівень захисту даних має вирішальне значення для системи обліку фінансових результатів, де будь-які розбіжності можуть призвести до серйозних проблем.

PostgreSQL також може похвалитися потужною підтримкою спільноти та великою кількістю документації, що полегшує пошук рішень потенційних проблем під час розробки. Активна спільнота сприяє постійному вдосконаленню та доступності численних плагінів і розширень, що дозволяє користувачам адаптувати систему баз даних до конкретних потреб.

Крім того, масштабованість є критично важливим фактором для будь-якої фінансової системи, особливо в умовах збільшення обсягу даних. PostgreSQL розроблено для ефективної обробки великих наборів даних, що робить його ідеальним вибором для компаній, що розвиваються. Його здатність масштабуватись по вертикалі та горизонталі гарантує, що система може працювати зі зростаючими робочими навантаженнями без шкоди для продуктивності.

Нарешті, відкритий вихідний код PostgreSQL забезпечує гнучкість у ліцензуванні та економічну ефективність, дозволяючи необмежене використання та модифікацію. Це особливо вигідно для академічних проектів, де бюджетні обмеження можуть бути фактором.

Таким чином, PostgreSQL обрано за відповідність стандартам SQL, багатий набір функцій, міцну цілісність даних, аналітичні можливості, активну підтримку спільноти, масштабованість і економічну ефективність. Ці атрибути роблять його дуже придатною системою керування базами даних для системи обліку фінансової ефективності з аналітичним модулем, гарантуючи, що проект може ефективно відповідати вимогам точного фінансового відстеження та глибокого аналізу.

3.3 Вибір інструментарію для створення програмного забезпечення

При розробці системи обліку фінансової ефективності з аналітичним модулем вибір інструментів розробки має вирішальне значення для забезпечення ефективності, масштабованості та зручності обслуговування. Наступні інструменти були обрані через їхні унікальні переваги та можливості:

.NET Framework: основа середовища розробки, .NET надає комплексну та універсальну платформу для створення надійних програм. Він

підтримує різні мови програмування та пропонує багату екосистему бібліотек і фреймворків, що сприяє швидкому розвитку та інтеграції.

ASP.NET: ця веб-платформа, частина екосистеми .NET, призначена для створення динамічних веб-додатків. Його підтримка архітектури MVC (Model-View-Controller) покращує поділ завдань, спрощуючи керування складними програмами, такими як система обліку фінансової ефективності. Вбудовані функції безпеки ASP.NET також забезпечують належний захист конфіденційних фінансових даних.

C#: як основна мова програмування для цього проекту, C# відома своєю простотою та потужністю. Він дає змогу розробникам писати чистий код, який зручно підтримувати, і бездоганно інтегрується з платформою .NET. C# підтримує принципи об’єктно-орієнтованого програмування, які необхідні для моделювання фінансових об’єктів та їх поведінки в системі.

MVC (Model-View-Controller): цей архітектурний шаблон допомагає організовувати код у три взаємопов’язані компоненти, сприяючи чіткому розподілу між інтерфейсом користувача та бізнес-логікою. Використовуючи MVC, команда розробників може створити програму, що масштабується та підтримується, яку легше тестувати та змінювати з часом.

Entity Framework Core (EF Core): як об’єктно-реляційний картограф (ORM), EF Core спрощує взаємодію з базою даних, дозволяючи розробникам працювати з даними в термінах об’єктів. Це оптимізує доступ до даних і маніпуляції, спрощуючи реалізацію функцій обліку фінансової ефективності, забезпечуючи при цьому цілісність і ефективність бази даних. Підтримка EF Core для PostgreSQL розширює можливості бази даних, спрощуючи складні запити та зв’язки даних.

PostgreSQL: цю систему керування реляційною базою даних із відкритим вихідним кодом обрано за її надійність, підтримку розширених типів даних і високу продуктивність. Дотримання PostgreSQL стандартів SQL

разом із потужними аналітичними функціями робить його ідеальним вибором для керування великою кількістю фінансових даних і забезпечення цілісності даних у системі.

Kubernetes: як платформа оркестровки контейнерів Kubernetes автоматизує розгортання, масштабування та керування контейнерними програмами. Використання Kubernetes покращує масштабованість і стійкість системи, забезпечуючи безперебійне оновлення та технічне обслуговування, ефективно обробляючи різні навантаження.

Docker: ця платформа використовується для створення, розгортання та запуску програм у контейнерах, забезпечуючи узгодженість у різних середовищах. Docker спрощує процес розробки, дозволяючи розробникам пакувати додатки та їхні залежності разом, оптимізуючи процес розгортання в різних середовищах, від розробки до виробництва.

ArgoCD як інструмент безперервної доставки для Kubernetes, ArgoCD полегшує автоматизоване розгортання програм. Він гарантує, що розгорнутий стан додатків відповідає бажаній конфігурації, дозволяючи швидкий відкат і сприяючи безперервній інтеграції та безперервному розгортанню (CI/CD).

Kustomize.io: цей інструмент використовується для більш ефективного керування конфігураціями Kubernetes. Kustomize дозволяє розробникам налаштовувати конфігурації додатків, не змінюючи вихідні файли YAML, що полегшує керування різними середовищами (такими як розробка, постановка та виробництво), зберігаючи контроль версій.

Таким чином, набір цих інструментів — .NET, ASP.NET, C#, MVC, EF Core, PostgreSQL, Kubernetes, Docker, ArgoCD і Kustomize.io — забезпечує надійний і сучасний стек для розробки для створення системи обліку фінансової ефективності. з аналітичним модулем. Разом вони гарантують, що система є масштабованою, зручною для обслуговування та здатною ефективно обробляти складні фінансові дані та потреби в аналізі.

3.4 Архітектура програмного забезпечення

Вибір архітектури мікросервісів для «Системи обліку фінансової ефективності з аналітичним модулем» надає численні переваги, які добре відповідають цілям і вимогам системи.

Однією з головних переваг є масштабованість. Мікросервіси дозволяють незалежно масштабувати різні компоненти, що особливо вигідно для аналітичного модуля. Цей модуль може вимагати більше обчислювальних ресурсів у періоди інтенсивного аналізу даних або звітності. Дозволяючи індивідуальне масштабування кожного компонента, система може більш ефективно розподіляти ресурси на основі коливань робочого навантаження.

Ще однією важливою перевагою є гнучкість у виборі технологій. Кожен мікросервіс може використовувати різні технології та мови програмування, що дозволяє розробникам вибирати найбільш підходящі інструменти для певних функцій. Наприклад, основні бухгалтерські послуги можуть бути реалізовані за допомогою технології, оптимізованої для транзакційних процесів, тоді як аналітичний модуль може використовувати спеціалізовані інфраструктури, розроблені для аналізу та візуалізації даних.

Ремонтопридатність також покращується завдяки мікросервісному підходу. Розділивши програму на менші керовані служби, розробники можуть працювати, оновлювати та розгортати кожну службу незалежно. Ця модульність не тільки спрощує поточне технічне обслуговування, але й забезпечує швидші ітерації в розробці, полегшуючи реагування на мінливі потреби бізнесу[18].

Архітектура також покращує ізоляцію несправностей. У налаштуванні мікросервісу проблеми в одній службі не обов’язково впливають на всю систему. Це означає, що якщо аналітичний модуль стикається з проблемами, основні функції бухгалтерського обліку можуть продовжувати працювати в

нормальному режимі, гарантуючи, що фінансові операції залишаться незмінними.

Також полегшено співпрацю між командами розробників. Мікросервіси сприяють децентралізованому підходу, дозволяючи різним командам одночасно працювати над різними сервісами. Це розділення може призвести до швидшого циклу розробки, оскільки команди можуть зосередитися на конкретних функціях, не втручаючись у роботу одна одної.

Безперервне розгортання — ще одна ключова перевага мікросервісів. Незалежний характер кожної служби дозволяє спрощувати методи інтеграції та розгортання. Ця можливість необхідна для ефективного впровадження оновлень у фінансову систему без необхідності повного перегляду програми.

Крім того, мікросервіси спрощують інтеграцію зовнішніх служб, що є вирішальним для фінансових систем, яким часто потрібно підключатися до різних API, таких як платіжні шлюзи або податкові служби. Ця архітектура гарантує, що кожна служба може взаємодіяти із зовнішніми системами стандартизованим способом, мінімізуючи збої в роботі інших частин програми.

Крім того, аналітичний модуль може використовувати мікросервіси для безпроблемного збору та обробки даних із різних джерел. Ця інтеграція підтримує розширену аналітику, обробку даних у режимі реального часу та навіть програми машинного навчання, що зрештою сприяє прийняттю більш обґрунтованих фінансових рішень.

Нарешті, мікросервісна архітектура підвищує безпеку. Кожну послугу можна захищати окремо, що дозволяє застосовувати індивідуальні заходи безпеки, які відображають конфіденційність даних, що обробляються. Цей цілеспрямований підхід допомагає захистити фінансову інформацію від несанкціонованого доступу та забезпечує відповідність відповідним нормам[19].

Підсумовуючи, прийняття мікросервісної архітектури для системи обліку фінансової ефективності з аналітичним модулем забезпечує надійну структуру, яка сприяє масштабованості, зручності обслуговування та гнучкості. Це забезпечує ефективну інтеграцію, розгортання та методи безпеки, в результаті чого створюється оперативна система управління фінансами, здатна адаптуватися до динамічних потреб бізнесу.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Вимоги до апаратного та програмного забезпечення

При розробці дуже важливо визначити необхідні специфікації апаратного та програмного забезпечення для забезпечення оптимальної продуктивності, можливостей обробки даних і безперебійної взаємодії з користувачем.

Для апаратного забезпечення надійна настройка сервера є важливою. Для ефективного керування завданнями одночасної обробки рекомендується багатоядерний ЦП, наприклад Intel Xeon або AMD Ryzen, щонайменше з чотирма ядрами. Сервер повинен мати принаймні 16 ГБ оперативної пам’яті, хоча для обробки інтенсивних аналітичних навантажень рекомендується 32 ГБ або більше. Не менш важливим є зберігання; SSD ємністю не менше 512 ГБ забезпечить швидкий доступ до даних, а конфігурації RAID можуть підвищити продуктивність і резервування. З’єднання Gigabit Ethernet є життєво важливим для ефективної передачі даних, особливо якщо очікується, що система керуватиме значними обсягами транзакцій або отримувати доступ до віддалених джерел даних.

На стороні клієнта необхідні сучасні багатоядерні процесори (наприклад, Intel i5/i7 або їхні еквіваленти AMD), а також мінімум 8 ГБ оперативної пам’яті для забезпечення безперебійної роботи програми. Локальне сховище має становити принаймні 256 ГБ для встановлення програмного забезпечення та кешованих даних. Для чіткої видимості фінансових даних і аналітичних результатів також рекомендується монітор із роздільною здатністю не менше 1920x1080.

З точки зору резервного копіювання та відновлення, впровадження зовнішнього рішення для резервного копіювання, такого як мережеве сховище (NAS) або хмарна служба, має вирішальне значення для регулярного резервного копіювання даних. Джерело безперебійного живлення (UPS) допоможе запобігти втраті даних під час перебоїв з електропостачанням.

Якщо говорити про вимоги до програмного забезпечення, то сервер може працювати на Windows Server 2019 або 2022 або на дистрибутиві Linux, як-от Ubuntu Server або CentOS, тоді як клієнти можуть використовувати Windows 10 або новішу версію, macOS або сумісні версії Linux. Для безпечного керування фінансовими даними необхідна система керування реляційною базою даних, наприклад Microsoft SQL Server, PostgreSQL або MySQL. За бажанням можна використовувати базу даних у пам’яті, як-от Redis або Memcached, для підвищення продуктивності шляхом кешування інформації, до якої часто звертаються.

Для розробки життєво важливий вибір відповідної мови програмування. Такі варіанти, як C#, Java або Python, можна використовувати для серверної частини, тоді як розробку фронтенду можна виконувати за допомогою HTML5, CSS3 і фреймворків JavaScript, таких як React, Angular або Vue.js. Щоб полегшити зв’язок між мікросервісами, розробники можуть створювати RESTful API за допомогою фреймворків, таких як ASP.NET Core, Flask або Spring Boot.

Також важливі аналітичні здібності. Використання бібліотек аналізу даних, таких як Pandas або Apache Spark, підтримуватиме обробку даних, а

інструменти візуалізації, такі як D3.js або Tableau, можуть ефективно представляти інформацію. Інструменти розробки повинні включати інтегровані середовища розробки (IDE), такі як Visual Studio або IntelliJ IDEA, а також Git для контролю версій і співпраці. Рішення для контейнеризації, такі як Docker, дозволяють створювати ізольовані середовища для мікросервісів.

Для ефективного керування мікросервісами потрібні такі інструменти оркестровки, як Kubernetes або Docker Compose. Заходи безпеки, включаючи веб-сервер, як-от Nginx або Apache, необхідні для захисту даних, а сертифікат SSL необхідний для безпечного зв'язку між клієнтами та серверами.

Впровадження автоматизованих інструментів тестування, таких як Postman для тестування API, разом із фреймворками модульного тестування, такими як JUnit або pytest, допоможе забезпечити надійність системи. Створення конвеєра безперервної інтеграції та розгортання (CI/CD) за допомогою таких інструментів, як Jenkins або GitHub Actions, ще більше оптимізує процес розробки.

Таким чином, ці специфікації апаратного та програмного забезпечення закладають міцну основу для створення системи обліку фінансової ефективності з аналітичним модулем, гарантуючи, що вона може ефективно обробляти фінансові транзакції, проводити комплексний аналіз і надавати цінну звітність, залишаючись безпечною та зручною для користувача.

4.2 Реалізація системи

Перш за все, я створив віртуальну машину на основі Ubuntu Server 22.04, використовуючи гіпервізор Hyper-V. Це стало початковим кроком для подальшого встановлення microk8s, що дозволяє працювати з кластером Kubernetes.

Рис. 9 Ubuntu

Hyper-V, розроблений компанією Microsoft, є потужною платформою для віртуалізації, яка дозволяє створювати та керувати віртуальними машинами на базі Windows. Вона є складовою частиною операційних систем Windows Server, а також Windows 10 Pro і Enterprise.

Ця платформа надає низку важливих функцій і можливостей. По-перше, Hyper-V підтримує віртуалізацію на серверному рівні, що дозволяє консолідувати фізичні сервери на одному пристрої. Це додатково зменшить витрати на обладнання та спростить управління ресурсами.

Крім того, Hyper-V також доступний для віртуалізації на рівнях настільного комп'ютера, що дозволяє користувачам Windows 10 Pro і Enterprise створювати й керувати віртуальними машинами на своєму робочому столі.

Крім того, Hyper-V підтримує різні операційні системи як гостьові ОС, включаючи Windows, Linux та інші. Це надає можливість використовувати різноманітні платформи віртуалізації відповідно до вимог користувача.

Управління віртуальними машинами працює за допомогою інструментів, які дозволяють створити, наслідкувати, копіювати та переміщувати віртуальні машини. Користувачі можуть контролювати ресурси, налаштування мережі, моніторинг та інші аспекти управління

віртуальними машинами через графічний інтерфейс Hyper-V або командний рядок PowerShell.

Hyper-V також забезпечує можливість міграції віртуальних машин між фізичними серверами без зупинки роботи, а налаштування кластерів Hyper-V дозволяють забезпечити високу доступність віртуальних машин. Це критично важливо для безперервної роботи додатків, навіть у разі відмови фізичного сервера[19].

Ще однією важливою особливістю Hyper-V є його інтеграція з засобами віддаленого управління, такими як System Center і Windows Admin Center. Це дозволяє адміністраторам керувати віртуальними машинами та хостами з єдиного централізованого інтерфейсу. без цього Hyper-V надає ряд можливостей для забезпечення безпеки віртуальних машин. Ізоляція та захист даних створюються за допомогою таких технологій, як Secure Boot, BitLocker і Network Isolation, які запобігають несанкціонованому доступу до віртуальних машин.

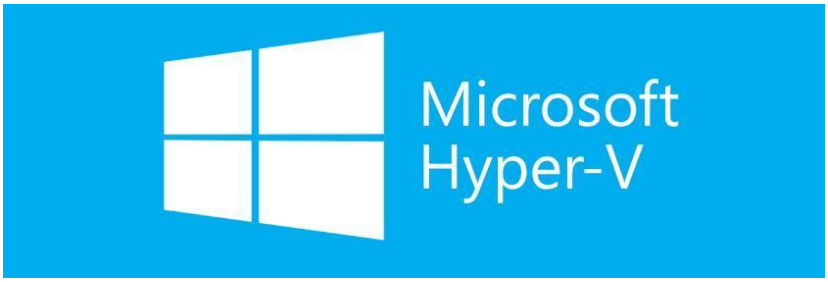

Рис. 10 Hyper-V

Microk8s є швидким і простим способом розгортання та управління Kubernetes-кластером на локальному комп'ютері або віртуальній машині. Розроблений компанією Canonical, яка також створює Ubuntu Linux, Microk8s має низку особливостей, які роблять його привабливим для розробників.

Рис. 11 MicroK8s

По-перше, Microk8s легко встановити на різні операційні системи, включаючи Linux, macOS та Windows. Його інсталяція займає лише кілька хвилин, а також вимагає мінімальних системних ресурсів, включаючи всі завантажені компоненти Kubernetes.

Цей інструмент надає можливість створення локального Kubernetes-кластера, що дозволяє користувачам імітувати робоче середовище Kubernetes без необхідності підключення до продуктивних кластерів. Microk8s працює в ізольованому середовищі, що захищає основну систему від можливих конфліктів, підвищуючи безпеку та надійність локальної розробки.

Серед додаткових переваг Microk8s – наявність вбудованих компонентів, таких як DNS-сервер, інгрес-контролер, системи зберігання даних, моніторинг та логування. Це дозволяє швидко налаштувати повноцінне середовище для розробки та тестування додатків. Користувачі також можуть легко оновлювати кластер Kubernetes, додавати чи видаляти компоненти, а також контролювати та керувати середовищем через командний рядок або графічний інтерфейс.

Microk8s виявляється надзвичайно корисним інструментом для розробки, тестування та навчання, хоча він дозволяє швидко розгорнути

Kubernetes-кластер на локальному комп’ютері та експериментувати з усіма його можливостями без використання виробничих ресурсів.

Для зручного моніторингу кластерів я підняв Kubernetes Dashboard. Це веб-інтерфейс, який забезпечує простий та інтуїтивно зрозумілий спосіб управління Kubernetes-кластерами. Адміністратори та розробники можуть легко взаємодіяти з ресурсами, не вдаючись до командного рядка.

Kubernetes Dashboard дозволяє створювати, оновлювати та видаляти різні ресурси, такі як поді, служби та розгортання, а також контролювати їхній стан. Dashbord надає інформацію про використання ресурсів, включаючи показники CPU, пам’яті та завантаження, а також журнали подій для моніторингу та відладки.

Ще важливою функцією є можливість налаштування доступу до кластера, що дозволяє адміністраторам визначати права користувачів і ролі для управління доступом до ресурсів. Крім того, Dashboard підтримує розширення через плагіни, що відкриває додаткові можливості для управління кластером.

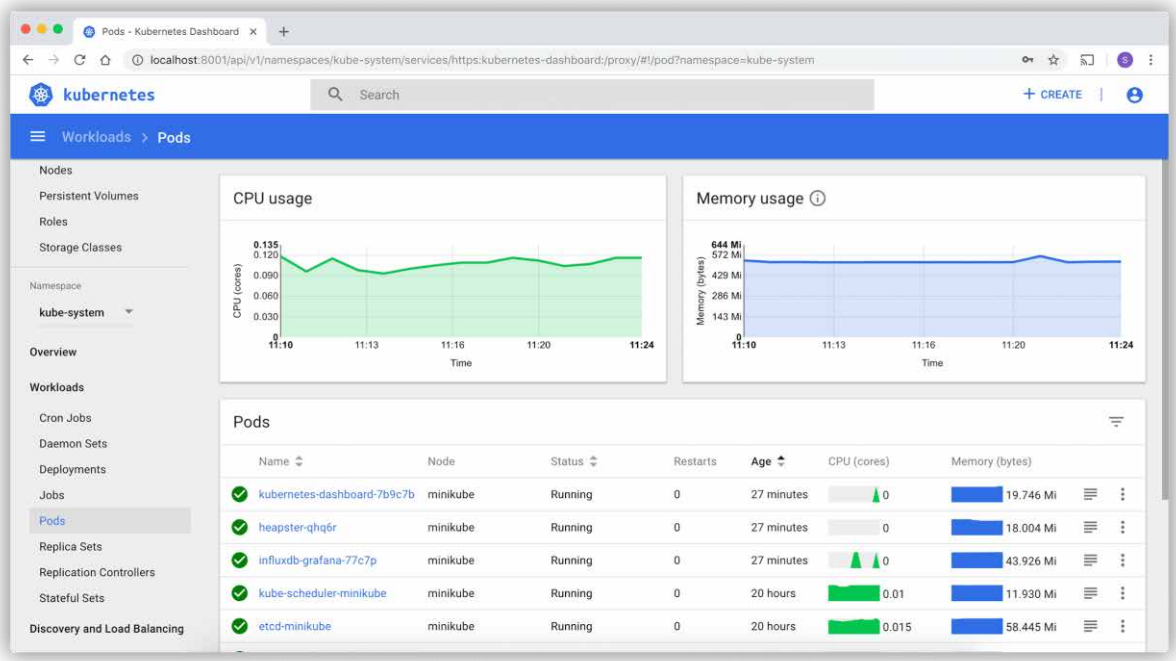

Рис. 12 Kubernetes dashboard

Argo CD був доповнений новими сервісами, які дозволяють відслідковувати зміни та автоматично їх розгортати. Коли ми завантажуємо в репозиторій, на який підписується Argo CD, він автоматично відображає зміни та швидко їх застосовує.

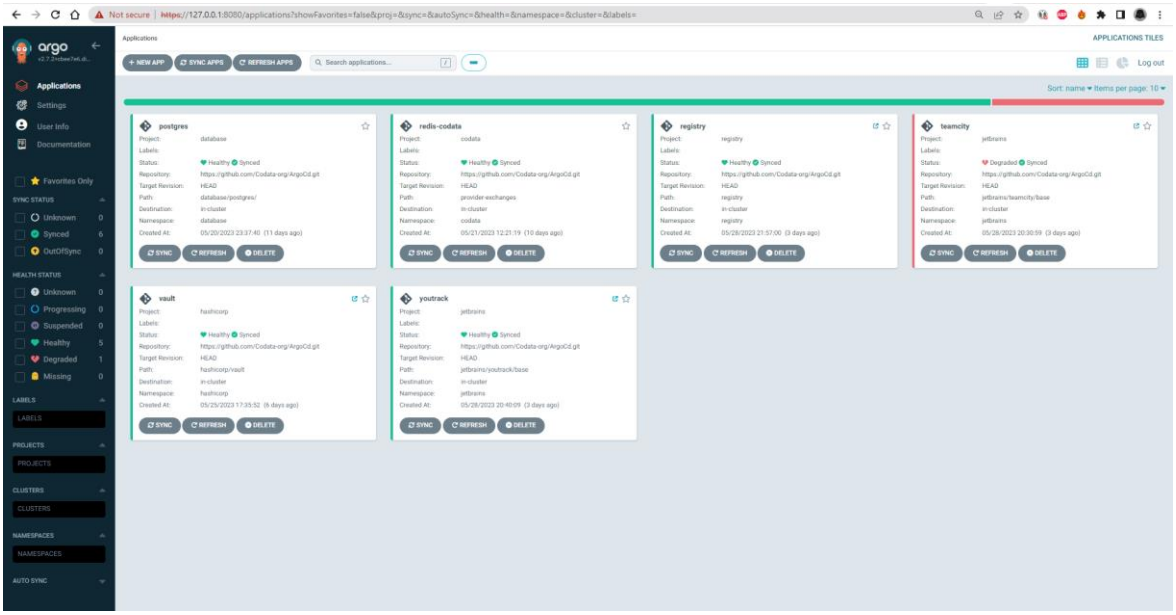

Рис. 13 ArgoCD dashboard

Серед нових функцій реалізовано сервіс Treeum.API, який видає отримувати курси валют із фізичних обмінників через REST API.

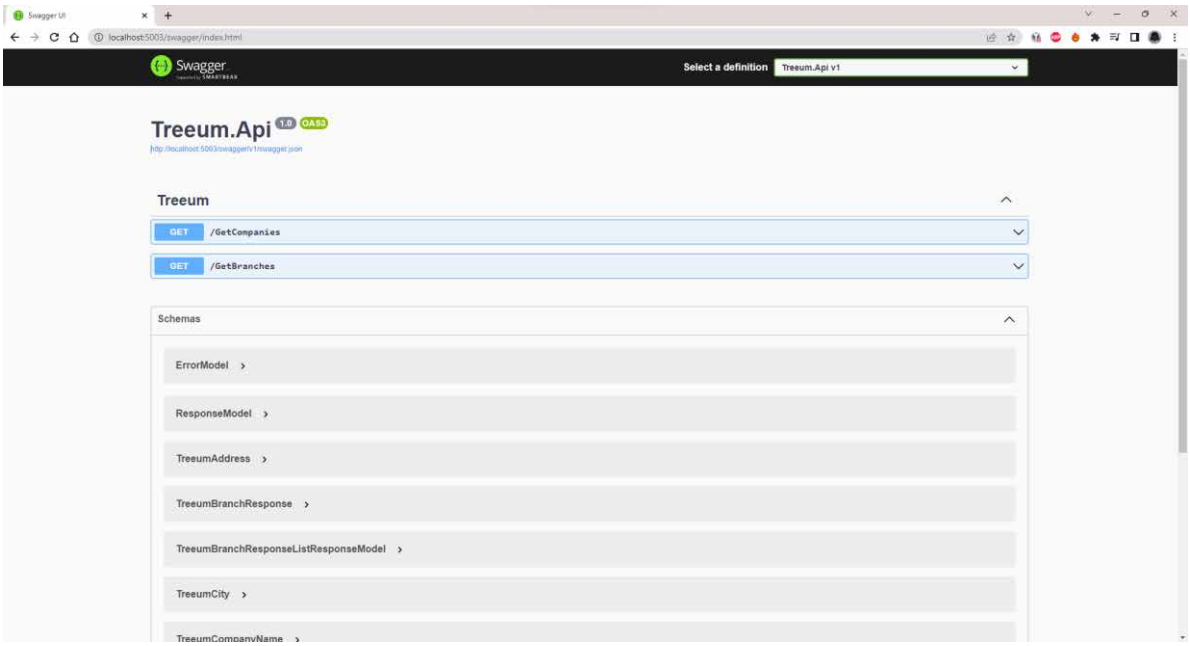

Рис. 14 сервіс Treeum.API

Ми можемо скористатися цим сервісом, щоб отримати актуальні курси валют.

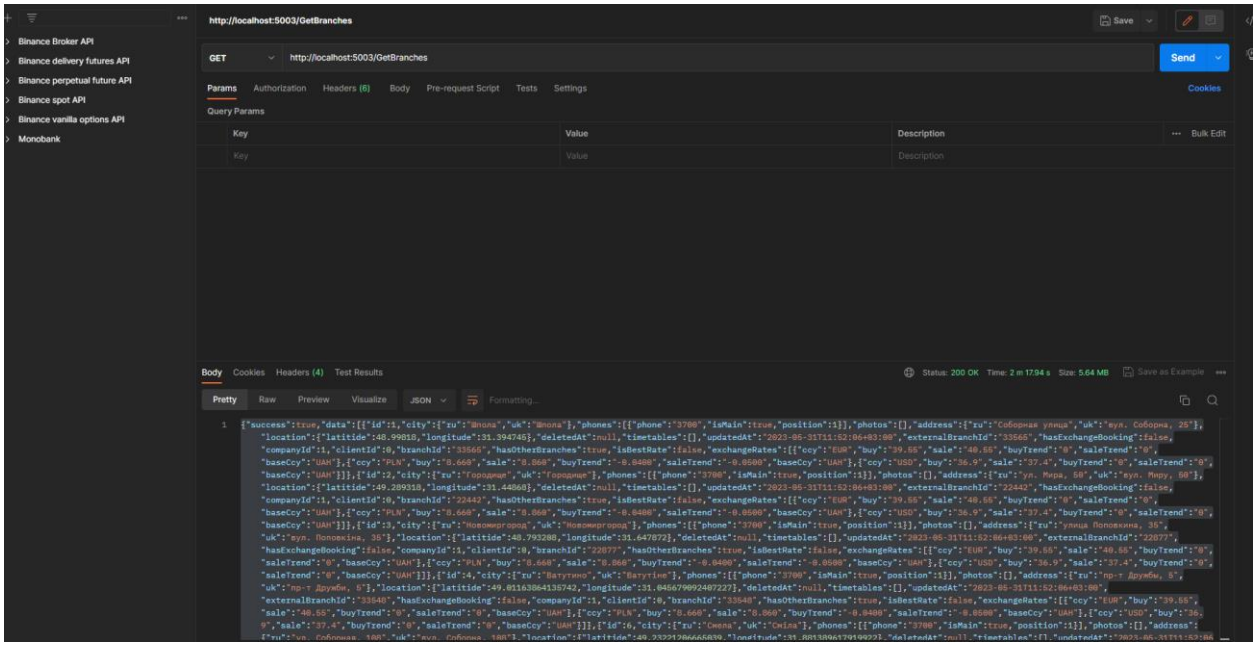

Рис. 15 відповідь від сервіса Treeum.API

Крім того, реалізовано сервіс CryptoExchanges.API, який забезпечує доступ до курсів валют з криптовалютних бірж через REST API.

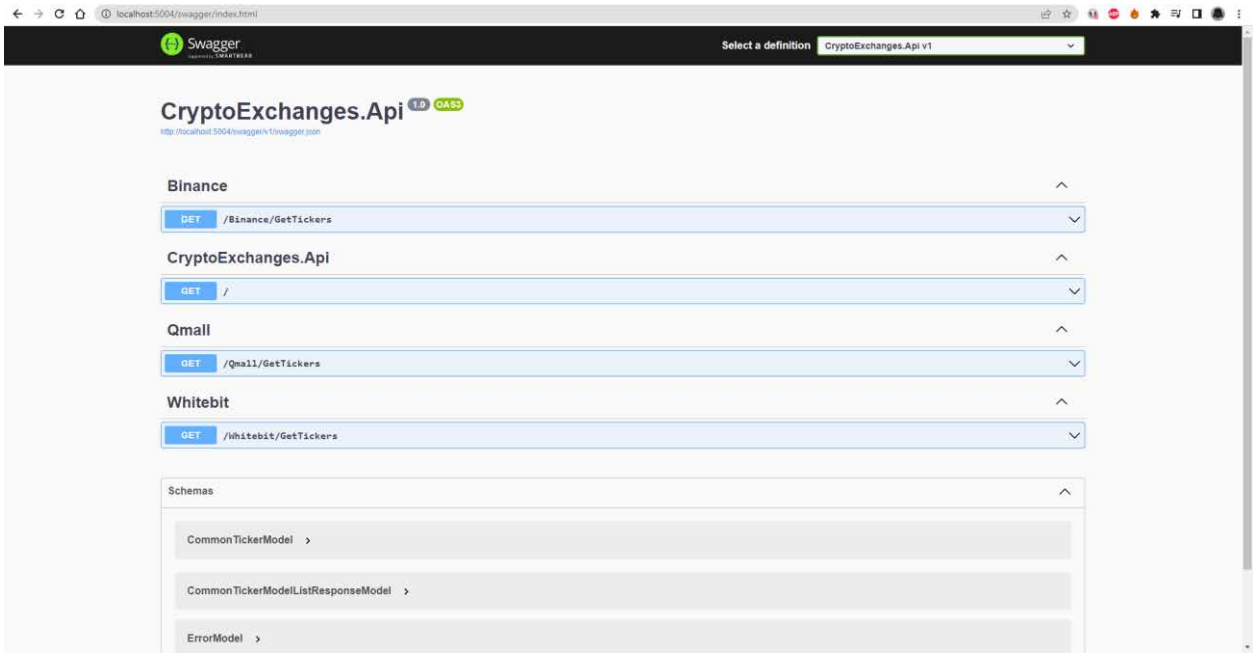

Рис. 16 сервіс CryptoExchanges.API

Цей сервіс дозволяє отримувати інформацію з бірж, таких як Binance, Whitebit та Qmall, забезпечуючи користувачів актуальними даними про криптовалюту.

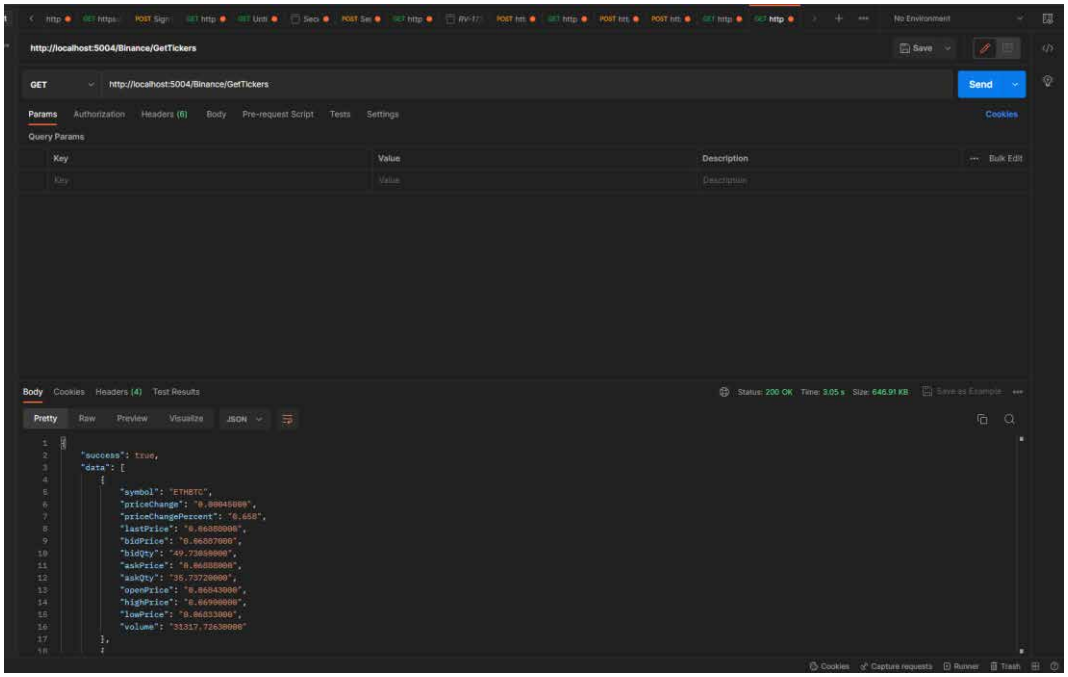

Рис. 17 відповідь від сервіса CryptoExchanges.API від біржі Binance

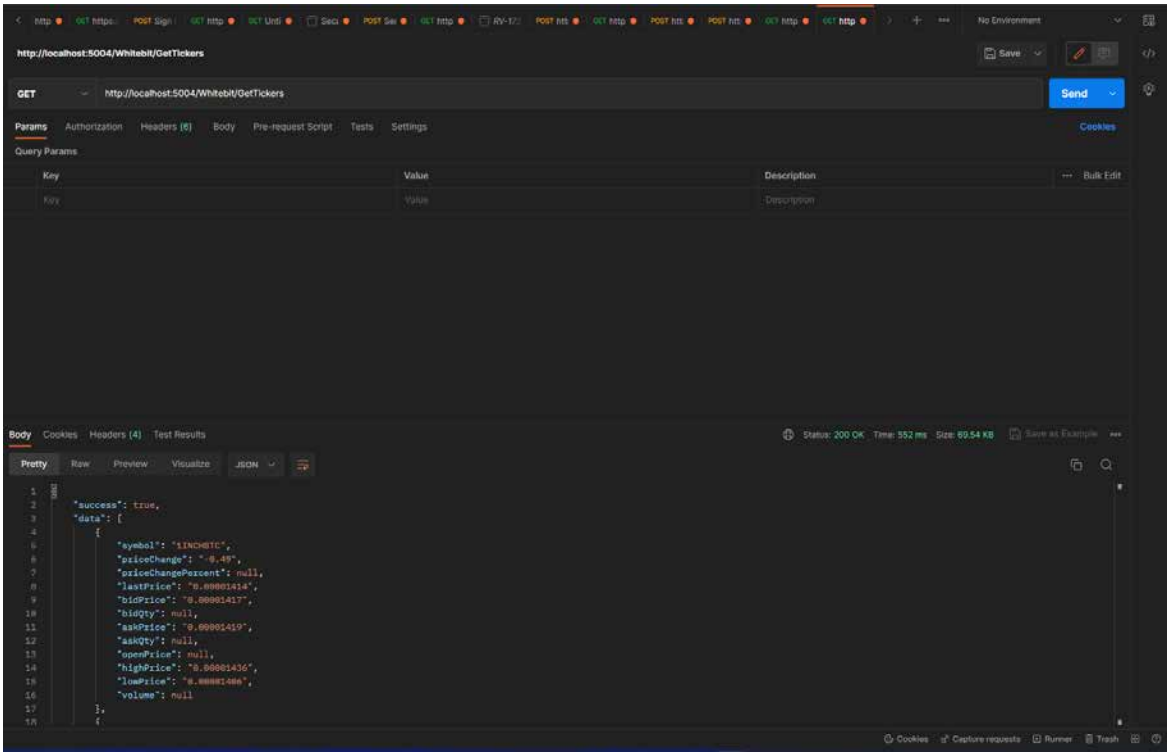

Рис. 18 відповідь від сервіса CryptoExchanges.API від біржі Whitebit

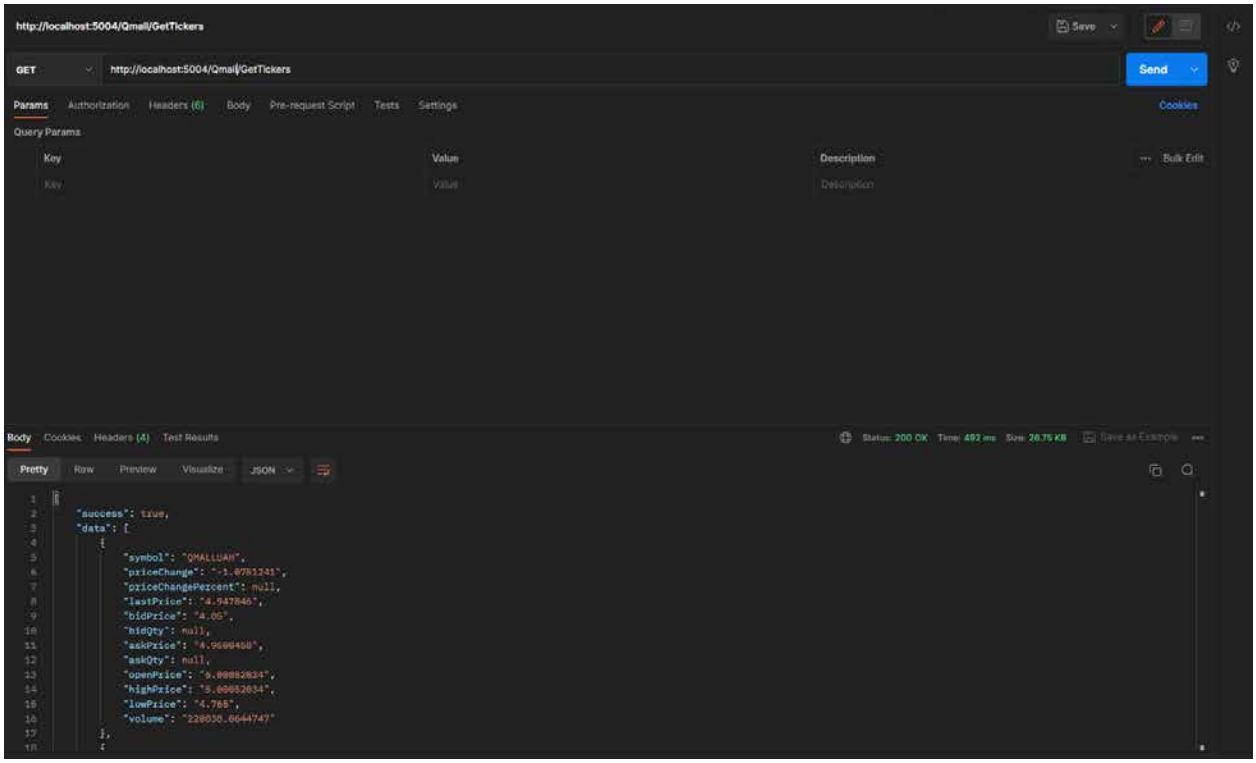

Рис. 19 відповідь від сервіса CryptoExchanges.API від біржі Qmall

Ще одним компонентом сайту є сервіс Identity.API, який використовується для управління користувачами, забезпечуючи реєстрацію та авторизацію. Сервіс успішно виконує свою функцію, додаючи або перевіряючи існуючих користувачів і шифруючи паролі для підвищення безпеки. У наведеному прикладі зображена таблиця з користувачами.

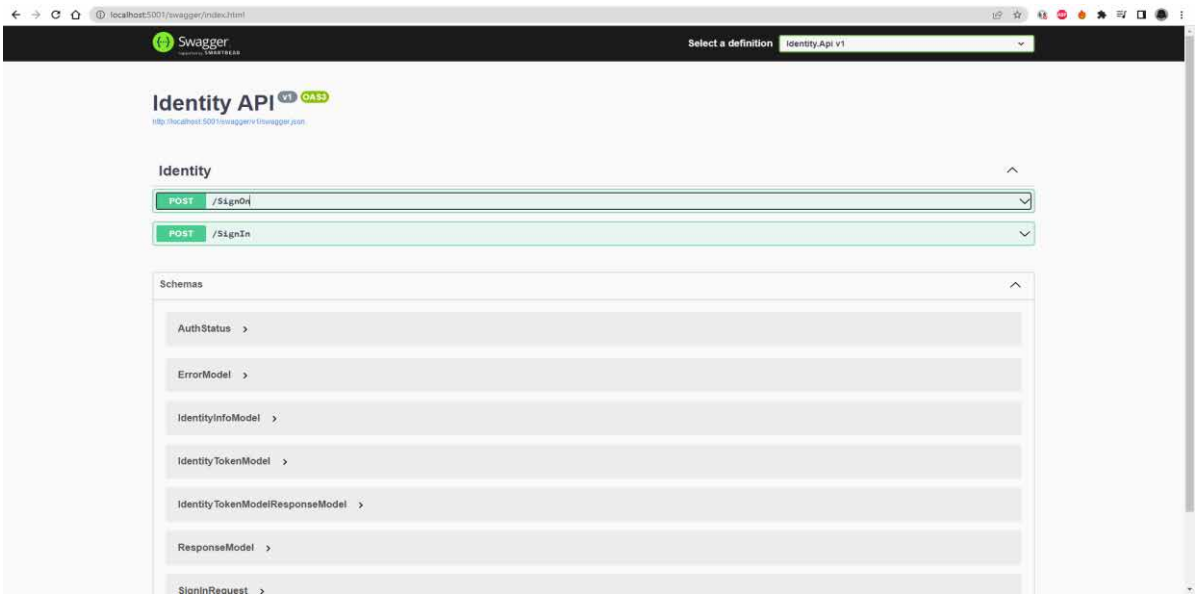

Рис. 20 сервіс Identity.API

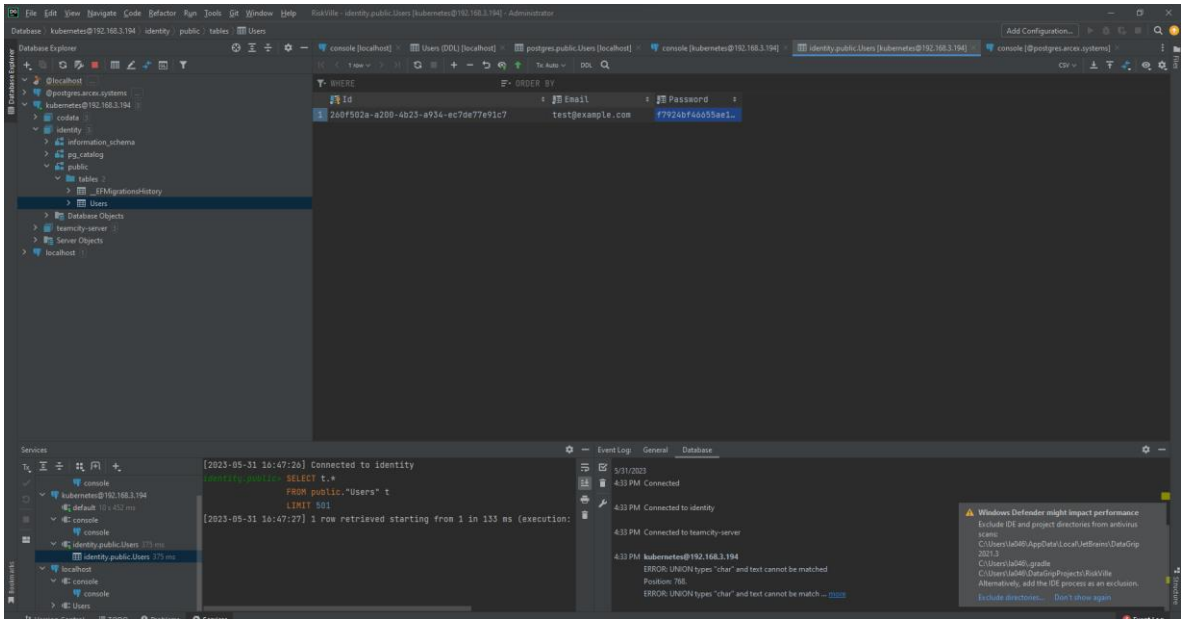

Рис. 21 Таблиця з користувачами

На наступному екрані демонструється успішна авторизація користувача, після якого видаються доступ та токени оновлення. Ці два типи токенів є критичними для автентифікації та авторизації в багатьох системах. Токен доступу служить для підтвердження ідентичності користувача при доступі до захищених ресурсів; він має обмежений термін дії. Користувач може передавати цей токен разом із запитами на сервер або API, що дає доступ до захищених даних.

Токен оновлення, з іншого боку, використовується для оновлення токена доступу без повторної авторизації. Він має довший термін дії і дозволяє отримати новий маркер доступу після закінчення терміну дії старого. Користувачі можуть використовувати токен оновлення для запиту нового токена доступу, що робить процес більш зручним і безпечним.

Використовуючи Access та Refresh токенів вашої безпеку системи, незважаючи на обмежені терміни дії Access token, ризик його зловживання зменшується. Однак важливо остаточно обробити ці токени, використовуючи захищені протоколи передачі, наприклад HTTPS, та захистити їх у безпечних місцях, щоб запобігти несанкціонованому доступу.

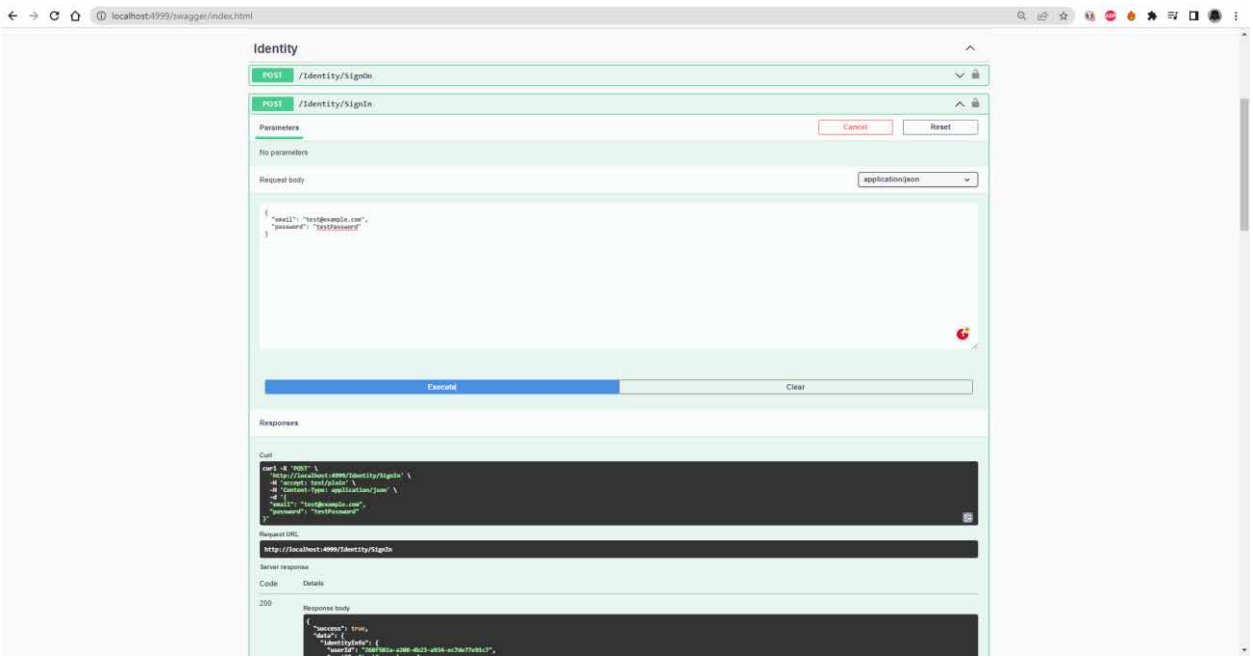

Рис. 22 Успішна авторизація

Реєстрація користувачів відбувається аналогічно, під час якого також надається доступ та токени оновлення. Нам у відповідному зображенню можна побачити успішний процес реєстрації.

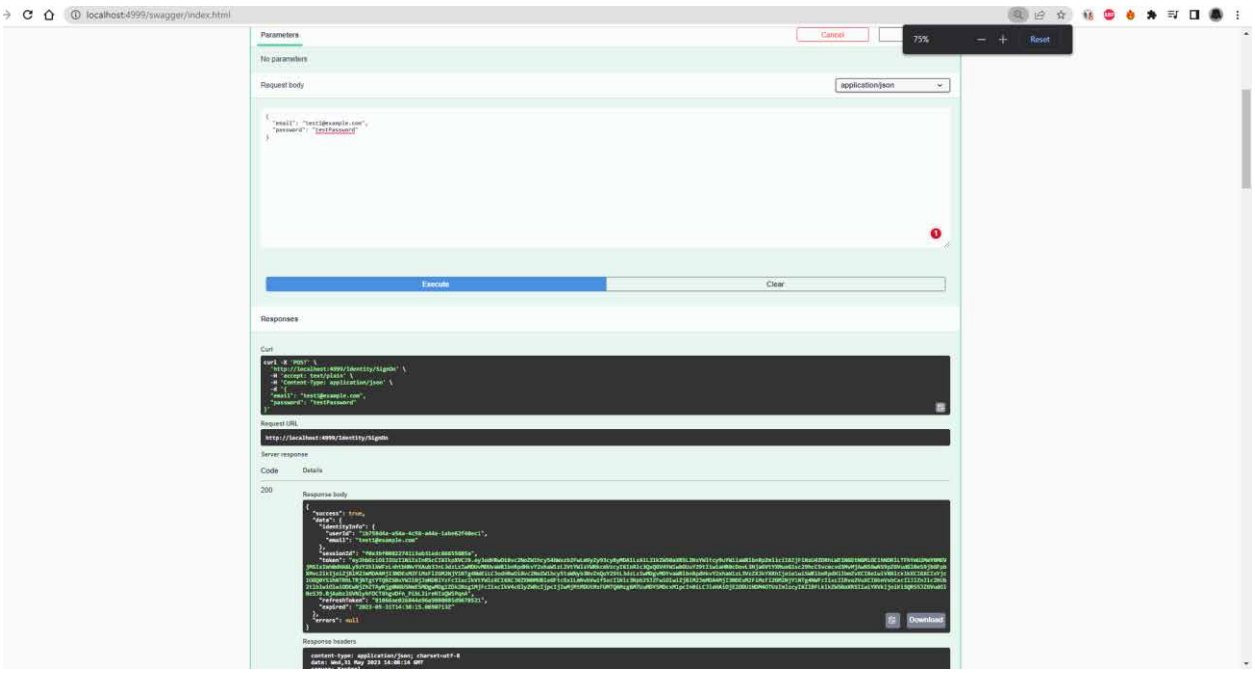

Рис. 23 Успішна реєстрація

Для оновлення інформації про курси валют використовують фонові завдання, реалізовані за допомогою Hangfire. Ці завдання потім запускаються,

оновлюють дані і кешують їх, що дозволяє припинити завантаження на бекенд. Hangfire є бібліотекою для планування і виконання фонових завдань у додатках на платформі .NET. Вона забезпечує простий механізм для створення та управління такими завданнями, які можуть виконуватися асинхронно або на регулярній основі.

Серед основних функцій Hangfire можна виділити постійне зберігання завдань у базах даних, що гарантує їх збереження навіть після перезапуску програми чи сервера. Крім того, бібліотека дозволяє легко планувати виконання завдань у визначений час або з певною тривалістю, надаючи зручний API для створення розкладів та визначення інтервалів запуску.

Hangfire підтримує асинхронне виконання завдань, що сприяє ефективнішому використанню ресурсів та запобігає блокуванню основного потоку програми. Додатково, бібліотека надає можливість створення різних черг завдань для контролю пріоритетів і пропускної здатності. Вбудований веб-інтерфейс надає можливість моніторингу виконання статусів, прогресу та статистики фонових завдань.

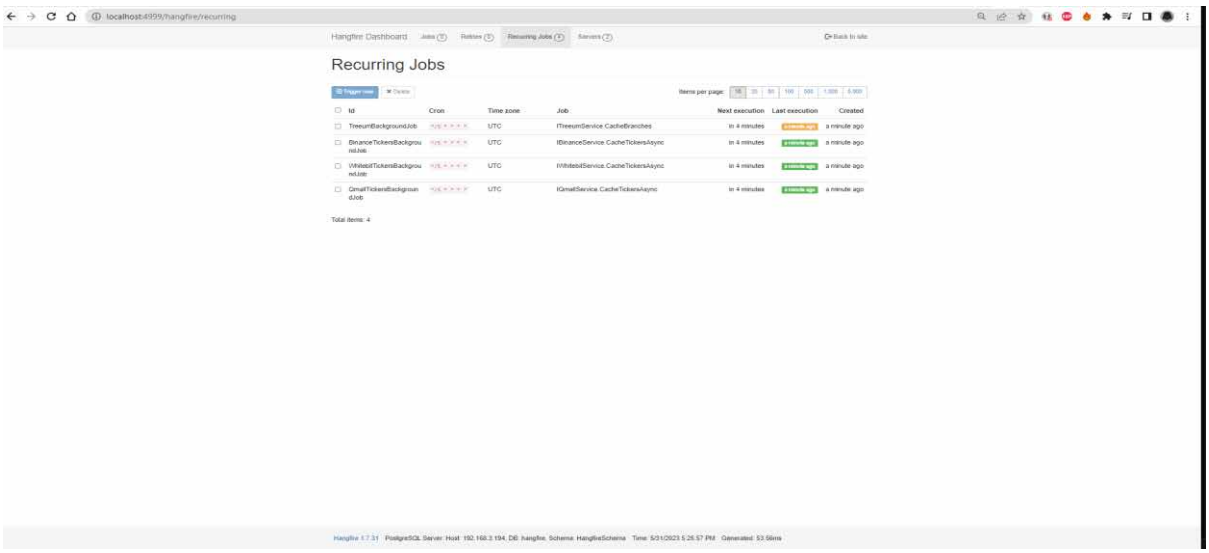

Рис. 24 Background jobs

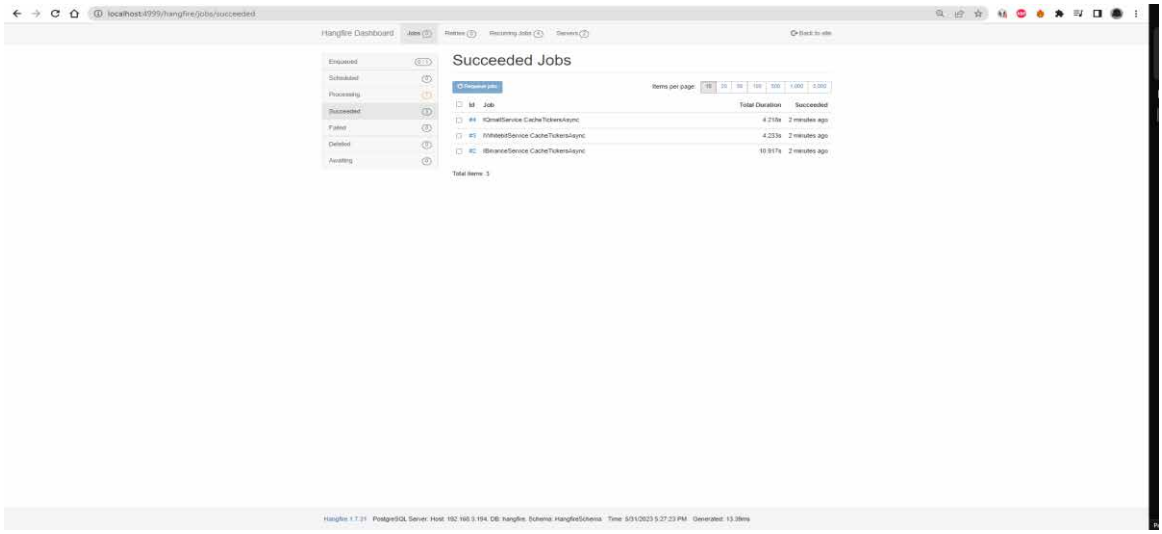

Рис. 25 Виконані Background jobs

На одному з зображень представлено виконані фонові завдання, де можна переглянути статистику та, у разі невдачі, отримати інформацію про трасування стека та інші параметри.

Дані про курси валют з криптовалютних бірж та фізичних обмінників кешуються в Redis. В одному зі скріншотів відображено, як наш Gateway отримує ці дані.

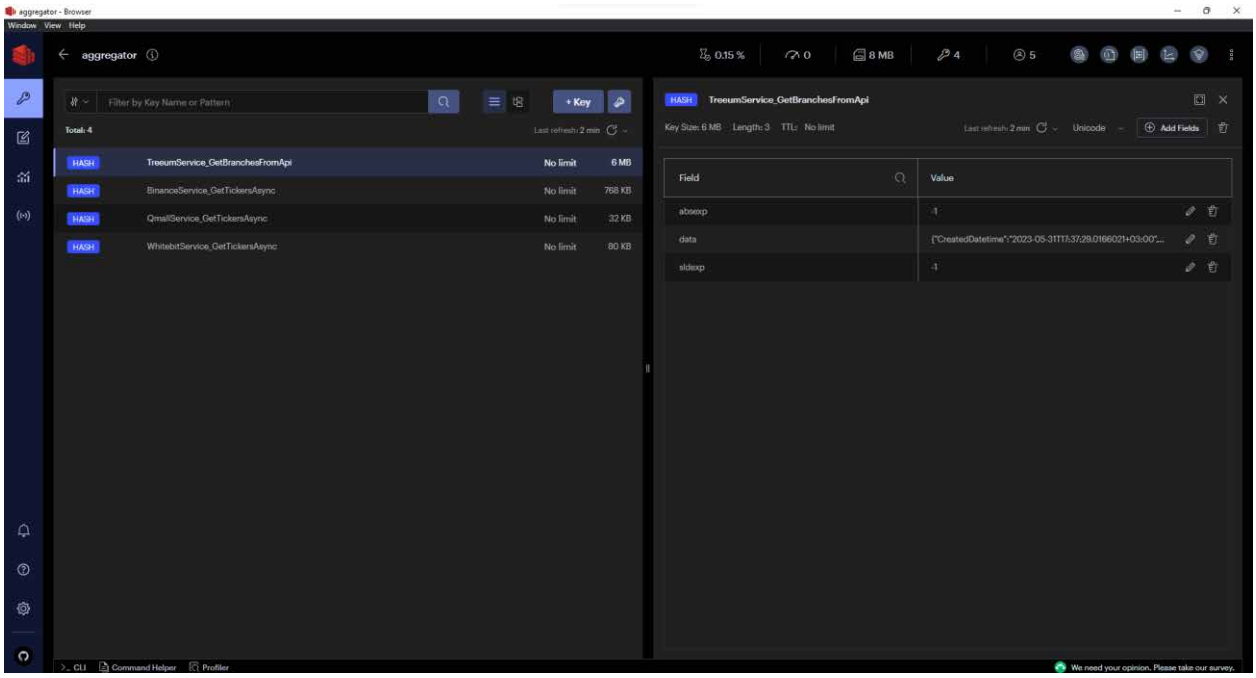

Рис. 26 Записи в Redis Cache

Для доступу до курсів валют необхідно вибрати список обмінників, представлених у формі типу enum. На ще одному зображенні показані можливості отримання інформації про курси валют від фізичних обмінників, які також підтримуються з Redis Cache.

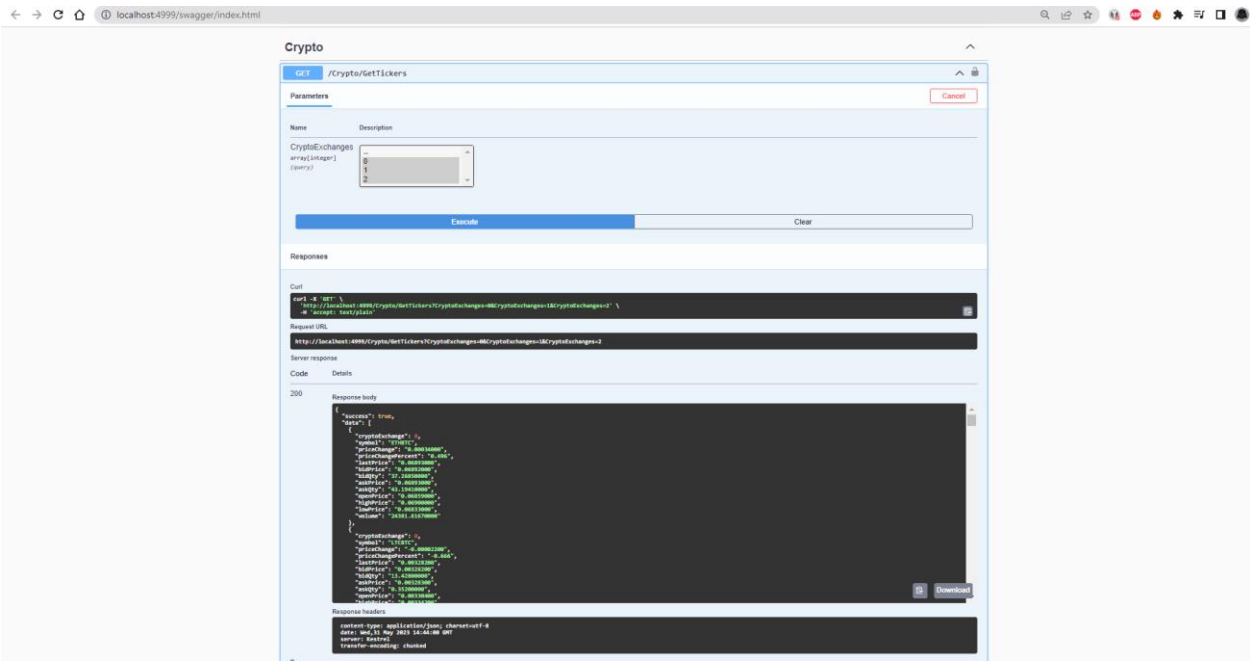

Рис. 27 Отримання інформації про курси валют з вибраних бірж

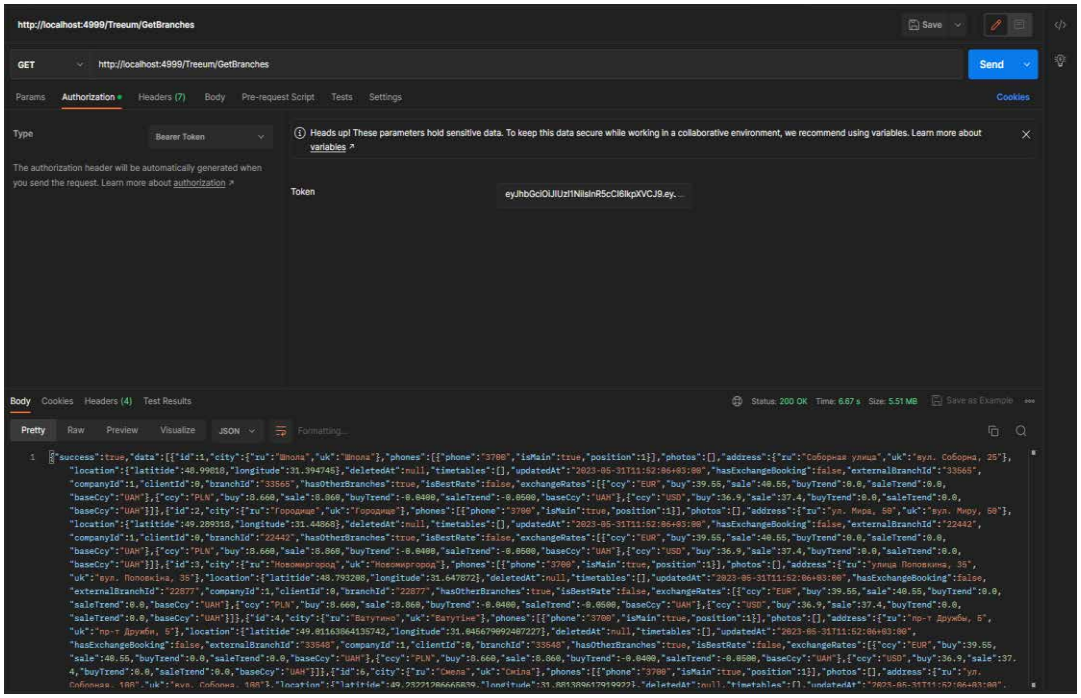

Рис. 28 Отримання інформації про курси валют з фіз обмінників

4.3 Тестування системи

У контексті розробки система обліку фінансової ефективності з аналітичним модулем комплексне тестування є важливим для забезпечення надійності, точності та продуктивності програмного забезпечення. Процес тестування включає кілька етапів, кожен з яких стосується різних аспектів системи, щоб перевірити її функціональність, зручність використання та загальну якість.

Модульне тестування передбачає тестування окремих компонентів або модулів програми окремо. Кожну функцію або метод слід перевірити, щоб переконатися, що вони працюють належним чином у різних умовах. Цього можна досягти за допомогою тестових фреймворків, таких як JUnit для Java, NUnit для .NET або pytest для Python. Модульні тести повинні охоплювати:

- перевірка введених даних;
- точність розрахунку;
- обробка винятків.

Після завершення модульних тестів необхідне інтеграційне тестування, щоб оцінити, як різні модулі взаємодіють один з одним. Цей етап зосереджений на взаємодії між системою обліку фінансових результатів та її аналітичним модулем. Основні області для тестування включають:

- потік даних між модулями;
- відповіді API та інтеграція із зовнішніми службами (наприклад, платіжними шлюзами, сторонніми постачальниками даних);
- взаємодія з базою даних для забезпечення цілісності та узгодженості даних.

Функціональне тестування оцінює програмне забезпечення відповідно до встановлених вимог. Тестові приклади слід отримувати з історій користувачів і документації вимог. Цей тип тестування повинен охоплювати:

- процеси реєстрації та автентифікації користувачів;
- введення даних і обробка транзакцій;

- формування фінансової звітності та аналітики;
- дозволи та ролі користувачів у системі.

Тестування прийнятності користувачами проводиться, щоб переконатися, що система відповідає потребам і очікуванням кінцевих користувачів. Залучення реальних користувачів до цього процесу має вирішальне значення, оскільки вони можуть надати цінний відгук про зручність використання та функціональність. UAT має зосередитися на:

- простота використання та навігації в додатку;
- чіткість звітів та аналітичних виходів;
- загальна задоволеність користувачів системою.

Тестування продуктивності оцінює, як система поводиться під різними навантаженнями та умовами. Це включає стрес-тестування, тестування навантаження та тестування масштабованості. Ключові показники для аналізу включають:

- час відгуку під час пікового використання;
- пропускна здатність;
- споживання ресурсів (ЦП, пам'ять і сховище).

Дотримуючись структурованого підходу до тестування, «Систему обліку фінансової ефективності з аналітичним модулем» можна перевірити на надійність, функціональність і продуктивність. Комплексне тестування гарантує, що система не тільки відповідає технічним вимогам, але й відповідає потребам користувачів і бізнес-цілям, що зрештою призводить до успішного розгортання та адаптації користувачами.

ВИСНОВКИ

У цій магістерській роботі ми розпочали всебічний шлях до розробки надійного програмного рішення, призначеного для вдосконалення фінансового менеджменту та процесів прийняття рішень в організаціях. Основна мета полягала в тому, щоб створити систему, яка не тільки точно відстежує фінансові показники, але й надає глибоку аналітику для прийняття стратегічних рішень.

Проект розпочався з ретельного аналізу існуючих систем фінансового обліку та виявлення прогалин та обмежень, з якими стикаються організації під час оцінки фінансової діяльності. На основі цього аналізу ми визначили чіткі цілі для нашої системи, зосередившись на зручних інтерфейсах, обробці даних у реальному часі та розширених аналітичних можливостях.

Для досягнення цілей розробки ми застосували системний підхід до розробки, який включав кілька критичних етапів.

Було співпрацювання із зацікавленими сторонами, щоб зібрати вимоги та зрозуміти конкретні потреби потенційних користувачів. Цей етап передбачав проведення інтерв’ю, опитувань і семінарів для визначення бажаних функцій і особливостей системи.

На основі зібраних вимог було розроблено архітектуру системи з використанням мікро сервісного підходу. Ця архітектура була обрана через її масштабованість, гнучкість і легкість інтеграції з іншими службами. Дизайн включав модуль фінансового обліку та аналітичний модуль, кожен з яких служив окремим, але взаємопов’язаним цілям.

Після оцінки різних технологій було обрано відповідні інструменти та фреймворки, які відповідають нашим цілям проектування. Це включало вибір системи керування реляційною базою даних для зберігання даних, серверної інфраструктури для розробки API та інтерфейсної інфраструктури для інтуїтивно зрозумілого інтерфейсу користувача.

Етап розробки включав кодування різних компонентів системи, гарантуючи, що кожен модуль був ретельно перевірений під час процесу впровадження. Ми інтегрували розширені аналітичні інструменти, щоб надавати інформацію про фінансові дані в режимі реального часу, дозволяючи користувачам без зусиль виконувати поглиблений аналіз.

Було проведено ретельне тестування на різних етапах розробки, включаючи модульне тестування, інтеграційне тестування, функціональне тестування, тестування прийнятності користувачами та тестування продуктивності. Це всебічне тестування підтвердило надійність, точність і безпеку системи, усунувши потенційні вразливості перед розгортанням.

Хоча ця розробка заклала міцну основу для системи обліку фінансових результатів, є можливості для подальшого розвитку. Майбутня робота може бути зосереджена на вдосконаленні можливостей машинного навчання для прогнозної аналітики, інтеграції додаткових джерел даних для більш повного фінансового огляду та вивченні розробки мобільних додатків для підвищення доступності.

Таким чином, ця магістерська робота є значним внеском у сферу управління фінансовою ефективністю. Систематичний підхід, який застосовувався протягом усього процесу розробки, не лише призвів до створення функціонального програмного рішення, але й дав цінну інформацію про складність фінансового обліку та аналітики. Подолаючи розрив між збором даних і прийняттям стратегічних рішень, система, розроблена в рамках цього проекту, має потенціал для розширення можливостей організацій для оптимізації їх фінансової діяльності та стимулювання зростання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Argo CD - Declarative GitOps CD for Kubernetes. Argo CD - Declarative GitOps CD for Kubernetes. [Електронний ресурс] – Режим доступу: https://argo-cd.readthedocs.io/en/stable/ (date of access: 31.05.2023).
2. ASP.NET Core load/stress testing. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/test/load-tests?view=aspnetcore-7.0 (date of access: 31.05.2023).
3. ASP.NET documentation. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0 (date of access: 31.05.2023).
4. Docker Docs: How to build, share, and run applications. Docker Documentation. [Електронний ресурс] – Режим доступу: https://docs.docker.com/ (date of access: 31.05.2023).
5. Documentation & Resources - YouTrack. JetBrains. [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/youtrack/documentation/ (date of access: 31.05.2023).
6. Getting Started with ASP.NET MVC 5 Index. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/ (date of access: 31.05.2023).
7. Git - Documentation. Git. [Електронний ресурс] – Режим доступу: https://git-scm.com/doc (date of access: 31.05.2023).
8. Integration tests in ASP.NET Core. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-7.0 (date of access: 31.05.2023).

9. Kubernetes Documentation. Kubernetes. [Електронний ресурс] – Режим доступу: https://kubernetes.io/docs/home/ (date of access: 31.05.2023).
10. Microservice Architecture with ASP.NET Core. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/shows/on-net/microservice-architecture-with-aspnet-core (date of access: 31.05.2023).
11. .NET Microservices. Architecture for Containerized .NET Applications. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/ (date of access: 31.05.2023).
12. Overview of Entity Framework Core - EF Core. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/ef/core/ (date of access: 31.05.2023).
13. PostgreSQL: Documentation. PostgreSQL: The world's most advanced open source database. [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/ (date of access: 31.05.2023).
14. Richter J. CLR via C#. 4th ed. Redmond : Microsoft Press, 2012. 863 p.
15. SIG CLI. SIG CLI. [Електронний ресурс] – Режим доступу: https://kubectl.docs.kubernetes.io/ (date of access: 31.05.2023).
16. TeamCity Documentation Home | TeamCity On-Premises. TeamCity On-Premises Help. [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/help/teamcity/teamcity-documentation.html (date of access: 31.05.2023).
17. Ахенбах, Р. та Майєр, А. (2020). Аналіз фінансових даних: методи та застосування. Лондон: Wiley. ISBN: 978-1-119-46781-1.
18. Адамс, Р. і Ходж, К. (2021). Управління фінансовою діяльністю: теорія і практика. Нью-Йорк: Routledge. ISBN: 978-0-367-49678-9.
19. Датта, С. і Лалл, А. (2019). Системи вимірювання продуктивності: теоретичні основи та практичне застосування. Лондон: Palgrave Macmillan. ISBN: 978-3-319-79490-6.

20. Езамель, М. та Сяо, Дж. (2020). Управлінський облік і фінансова ефективність: емпіричне дослідження. Нью-Йорк: Springer. ISBN: 978-3-030-47388-6.
21. Хван Ю. та Шин Дж. (2022). Розширена фінансова аналітика: прийняття рішень на основі даних. Кембридж: Cambridge University Press. ISBN: 978-1-108-85515-7.
22. Іттнер, К. та Ларкер, Д. (2019). Інновації в вимірюванні ефективності: теорія і практика. Бостон: Harvard Business Review Press. ISBN: 978-1-63369-492-7.
23. Коллер, Т., Гедхарт, М. і Весселс, Д. (2020). Оцінка: вимірювання та управління вартістю компаній. Хобокен: Уайлі. ISBN: 978-1-119-69196-2.
24. Ларрінага, К. та Марімон, Ф. (2021). Екологічний облік: роль звітності у фінансовій діяльності. Лондон: Routledge. ISBN: 978-0-367-49600-6.
25. Лі, Т. та Лім, Дж. (2021). Аналітика даних у вимірюванні фінансової ефективності. Нью-Йорк: Springer. ISBN: 978-3-030-69112-7.
26. Нісім, Д. та Пенман, С. (2019). Вимірювання фінансової ефективності: аналітичний підхід. Нью-Йорк: Oxford University Press. ISBN: 978-0-19-884280-9.
27. Отлі, Д. та Бродбент, Дж. (2021). Управління продуктивністю: комплексний підхід. Лондон: Видавництво CIMA. ISBN: 978-1-85964-516-7.
28. Парментер, Д. (2020). Ключові показники ефективності: розробка, впровадження та використання виграшних KPI. Хобокен: Уайлі. ISBN: 978-1-119-58850-0.
29. Портер, М. (2022). Конкурентна перевага: створення та підтримка чудової продуктивності. Нью-Йорк: Вільна преса. ISBN: 978-1-5011-0501-1.
30. Ріахі-Белькауї, А. (2019). Теорія бухгалтерського обліку: концептуальний та інституційний підхід. Нью-Йорк: Routledge. ISBN: 978-0-367-14152-8.
31. Шалтеггер, С. і Берріт, Р. (2020). Бізнес-кейс для сталого розвитку: роль вимірювання ефективності. Нью-Йорк: Springer. ISBN: 978-3-030-41853-7.