МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії

програмного забезпечення

Євген ДАВИДЕНКО

«19» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ПЛАТФОРМА ІНВЕСТУВАННЯ В ІТ-СТАРТАПИ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Дмитро КРАВЕЦЬ**

«19» червня 2026 р.

Керівник роботи _____ **Світлана БОРОВЛЬОВА**

ст. викладачка «19» червня 2026 р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп’ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
Євген ДАВИДЕНКО
«01» лютого 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Кравець Дмитро

1. Тема кваліфікаційної роботи Платформа інвестування в IT-стартапи затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від 26 грудня 2025 р.
2. Строк представлення кваліфікаційної роботи _____ 2026 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Результатом роботи є розроблений вебзастосунок, що реалізує функціонал інвестиційної платформи, включаючи реєстрацію користувачів, створення та управління проєктами, обробку платежів та взаємодію між користувачами.

Перелік питань, що підлягають розробці:

- аналіз предметної області інвестиційних платформ;
- формування технічного завдання та вимог до системи;

- розробка архітектури програмного забезпечення;
- проєктування бази даних;
- реалізація серверної частини (backend);
- розробка клієнтської частини (frontend);
- реалізація системи авторизації та автентифікації;
- інтеграція платіжних систем;
- тестування програмного продукту;
- оцінка якості програмного забезпечення.

4. Перелік графічних матеріалів: Презентація.
5. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «_03_» ___січня___ 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Платформа інвестування в ІТ-стартапи**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	03.02.2026	07.02.2026	Виконано
2.	Огляд літератури за темою роботи	08.02.2026	20.02.2026	Виконано
3.	Складання календарного плану КБР	10.02.2026	12.02.2026	Виконано
4.	Аналіз предметної області	15.02.2026	30.02.2026	Виконано
5.	Розробка проєктних рішень	06.03.2026	20.03.2026	Виконано
6.	Моделювання та конструювання ПЗ	21.03.2026	10.04.2026	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	11.04.2026	10.05.2026	Виконано
8.	Відгук керівника КБР	12.05.2026	12.05.2026	Виконано
9.	Оформлення КБР та презентації	13.05.2026	25.05.2026	Виконано
10.	Попередній захист	26.05.2026	26.05.2026	Виконано
11.	Завершення оформлення КБР та презентації	08.06.2026	12.06.2026	Виконано
12.	Рецензування	12.06.2026	14.06.2026	Виконано
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Дмитро КРАВЕЦЬ**

«__» _____ 2026 р.

Керівник роботи _____ **Світлана БОРОВЛЬОВА**

ст. викладачка

«__» _____ 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Платформа інвестування в ІТ-стартапи

Здобувач 408 гр.: Кравець Дмитро

Керівник: ст. викладачка Боровльова Світлана

Актуальність даної роботи зумовлена необхідністю підвищення ефективності організації інвестиційних процесів у цифровому середовищі, забезпечення надійної та масштабованої роботи вебплатформ, а також удосконалення механізмів комунікації та взаємодії між учасниками інвестиційної діяльності.

Об’єктом роботи є процеси організації інвестиційної діяльності та взаємодії користувачів у вебплатформах інвестування в ІТ-стартапи

Предметом роботи є методи та програмні засоби проєктування і реалізації вебзастосунку інвестування в ІТ-стартапи.

Метою роботи є розробка вебзастосунку інвестиційної платформи, що створює умови для ефективної взаємодії користувачів, оптимізації інвестиційних процесів та підтримки масштабованості системи.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність теми, визначено мету, завдання, об’єкт і предмет кваліфікаційної роботи.

У першому розділі проведено аналіз аналогів інвестиційних платформ та предметної області кваліфікаційної роботи.

У другому розділі визначено функціональні та нефункціональні вимоги до системи, а також спроєктовано загальну структуру мікросервісної архітектури.

У третьому розділі побудовані діаграми, а також обґрунтовано вибір технологій реалізації та підходів до організації взаємодії між сервісами.

У четвертому розділі показано програмну реалізацію та проведено аналіз результатіт обчислення та тестування. Також, зроблено керівництво користувача.

У висновках підведено підсумки виконаної роботи та визначено перспективи подальшого розвитку системи.

Результатом роботи є створення застосунку, що реалізує реєстрацію користувачів, управління профілями, створення та перегляд стартап-проєктів, здійснення інвестицій та обмін інформацією між учасниками системи.

Кваліфікаційна робота викладена на 78 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 24 найменувань та 3 додатків. Праця містить 16 таблиць та 22 рисунків.

Ключові слова: інвестиційна платформа, вебзастосунок, мікросервісна архітектура, стартап, REST API, gRPC, Golang, база даних.

ABSTRACT

to the qualifying bachelor's thesis

Platform for investing in IT startups

Student of 408 group: Kravets Dmytro

Supervisor: Senior Lecturer Borovlyova Svitlana

The relevance of this work is due to the need to increase the efficiency of organizing investment processes in a digital environment, ensuring reliable and scalable operation of web platforms, as well as improving communication and interaction mechanisms between participants in investment activities.

The object of the work is the processes of organizing investment activities and user interaction in web platforms for investing in IT startups

The subject of the work is methods and software tools for designing and implementing a web application for investing in IT startups.

The purpose of the work is to develop a web application for an investment platform that ensures effective user interaction, optimization of investment processes and support for system scalability.

The qualification work consists of an introduction, 4 sections, conclusions and a list of reference sources.

The introduction substantiates the relevance of the topic, defines the goal, objectives, object and subject of the qualification work.

The first section analyzes analogues of investment platforms and the subject area of the qualification work.

The second section defines functional and non-functional requirements for the system, and also designs the general structure of the microservice architecture.

The third section builds diagrams, and justifies the choice of implementation technologies and approaches to organizing interaction between services.

The fourth section shows the software implementation and analyzes the results of calculations and testing. Also, a user manual is made.

The conclusions summarize the work done and determine the prospects for further development of the system.

The result of the work is the creation of an application that implements user registration, profile management, creation and viewing of startup projects, investment and information exchange between system participants.

The qualification work is presented on 78 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 24 titles and 3 appendices. The work contains 16 tables and 22 figures.

Keywords: investment platform, web application, microservices architecture, startup, REST API, gRPC, Golang, database.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface.

E2E – End-to-End.

gRPC – Google Remote Procedure Call.

JWT – JSON Web Token.

OAuth – Open Authorization.

RBAC – Role-Based Access Control.

UML – Unified Modeling Language.

ВСТУП

Розвиток цифрової економіки та стрімке зростання кількості IT-стартапів формують попит на сучасні інформаційні системи, здатні спростити залучення інвестицій та підвищити ефективність бізнес-процесів. У зв’язку з цим значного поширення набули онлайн-платформи для інвестування, які об’єднують інвесторів та засновників стартапів у межах єдиного інформаційного середовища.

Подібні платформи використовуються для пошуку перспективних проєктів, створення інвестиційних кампаній та проведення фінансових операцій. Збільшення кількості користувачів і обсягів даних призвело до зростання навантаження на програмне забезпечення. За таких умов особливої актуальності набули питання масштабованості, відмовостійкості та підтримки стабільної роботи систем. Використання монолітної архітектури ускладнює модернізацію програмного забезпечення та обмежує можливості його подальшого розвитку.

Одним із підходів до розв’язання даної проблеми стала мікросервісна архітектура, що передбачає розподіл системи на незалежні сервіси з окремими зонами відповідальності. Така організація програмного забезпечення спрощує масштабування, оновлення окремих компонентів та супроводження системи.

Серед найбільш відомих платформ онлайн-інвестування можна виділити Wefunder, Crowdcube, Republic та StartEngine. Дані системи реалізують широкий набір функцій для взаємодії між інвесторами та засновниками стартапів і використовують сучасні технології розробки. Разом із цим окремі рішення мають обмеження, пов’язані зі складністю подальшого розвитку та масштабування.

Актуальність кваліфікаційної роботи зумовлена необхідністю забезпечення надійної та масштабованої роботи вебплатформ, а також удосконалення механізмів комунікації та взаємодії між учасниками інвестиційної діяльності.

Практичне значення кваліфікаційної роботи полягає у створенні архітектурної та програмної основи для розробки сучасної інвестиційної

платформи, яка може бути використана як база для створення реального програмного продукту.

Метою кваліфікаційної роботи є розробка вебзастосунку інвестиційної платформи для забезпечення ефективної взаємодії користувачів, оптимізації інвестиційних процесів та підтримки масштабованості системи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) проаналізувати сучасні підходи до створення онлайн-платформ для інвестування;

2) проаналізувати предметну область інвестування в IT-стартапи;

3) проаналізувати процеси групової взаємодії та комунікації між користувачами системи;

4) визначити функціональні вимоги до вебзастосунку;

5) визначити нефункціональні вимоги (продуктивність, безпека, масштабованість);

6) спроєктувати загальну структуру мікросервісної архітектури;

7) визначити склад та відповідальність окремих мікросервісів.

Об’єктом роботи є процеси організації інвестиційної діяльності та взаємодії користувачів у вебплатформах інвестування в IT-стартапи.

Предметом роботи є методи та програмні засоби проєктування і реалізації вебзастосунку інвестування в IT-стартапи.

Основними проєктними рішеннями є використання мікросервісної архітектури, REST API та gRPC для взаємодії сервісів, контейнеризації Docker, брокера повідомлень Kafka та системи оркестрації асинхронних процесів Temporal.

Результати кваліфікаційної роботи можуть бути використані у сфері розробки вебплатформ для інвестування, фінансових інформаційних систем та інших програмних продуктів, що потребують масштабованої мікросервісної архітектури, підтримки великої кількості користувачів та інтеграції з зовнішніми сервісами.

1 АНАЛІТИЧНА ЧАСТИНА

У даному розділі проведено системний аналіз предметної області інвестування в ІТ-стартапи, на основі чого, було сформульована постановка завдань до кваліфікаційної роботи. У цьому розділі розкрито об’єкт та предмет кваліфікаційної роботи.

Також, описано та проаналізовано структурні та функціональні особливості об’єкта кваліфікаційної роботи. На основі аналізу було обґрунтовано вибір мікросервісної архітектури та сформовано постановку завдання на розробку програмного забезпечення [12].

1.1 Аналіз предметної області інвестування в ІТ-стартапи

Розвиток цифрової економіки та стрімке зростання кількості ІТ-стартапів формують нові підходило до організації інвестиційної діяльності [6]. У сучасних умовах застосунки потребують гнучкої масштабованості, а користувачі – ефективний інструмент для інвестування в перспективні ІТ-стартапи [8]. Також, вебплатформи для інвестування набули велику популярність на сьогоднішній день, що сприяє необхідності бути адаптивними, бо з сьогоднішньою тенденцією будь яка людина може з легкістю зробити інвестицію або створити новий стартап, внаслідок чого, навантаження на вебплатформи постійно зростає.

ІТ-стартапи є одним із найбільш перспективних напрямів інвестування, оскільки, зараз ІТ розвивається дуже стрімко, і кожен день створюються перспективні програмні продукти. На відміну від традиційного бізнесу, стартапи характеризуються високим рівнем інноваційності, швидкими темпами розвитку та значним потенціалом масштабування.

Традиційні механізми інвестування часто мають низку обмежень, серед яких складність пошуку інвесторів, тривалі процедури перевірки проєктів, недостатня прозорість фінансових операцій та відсутність ефективних інструментів

комунікації між учасниками інвестиційного процесу [22]. У зв’язку з цим онлайн платформи для інвестування набули велику популярність [8].

Основними учасниками систем онлайн-інвестування є інвестори, засновники стартапів та адміністратори платформи. Інвестори здійснюють пошук перспективних проєктів, аналізують інформацію про компанії та приймають рішення щодо інвестування. Засновники стартапів створюють інвестиційні кампанії, публікують відомості про власні проєкти та взаємодіють із потенційними інвесторами. Адміністратори відповідають за модерацію контенту, контроль роботи системи та підтримання належного рівня безпеки платформи.

Важливе місце у сучасних системах онлайн-інвестування займають засоби комунікації між користувачами. Механізми обміну повідомленнями, системи сповіщень, публікація новин і звітності підвищують прозорість інвестиційного процесу та формують довіру між інвесторами й засновниками стартапів. Подібні інструменти мають особливе значення для проєктів, що перебувають на ранніх стадіях розвитку та потребують постійного інформування потенційних інвесторів.

Сучасні інвестиційні платформи містять засоби аналізу та обробки великих обсягів даних. Використання аналітичних інструментів спрощує оцінювання популярності проєктів, відстеження динаміки залучення інвестицій та формування статистичних звітів для користувачів і адміністраторів системи. На основі отриманих даних учасники платформи можуть приймати більш обґрунтовані рішення, що позитивно впливає на розвиток інвестиційної екосистеми.

Важливими характеристиками сучасних інвестиційних платформ є безпека, масштабованість та довіра між учасниками системи. Для цього використовуються механізми автентифікації, KYC/AML-перевірки, системи модерації контенту та журналювання дій користувачів [1]. До складу сучасних вебплатформ входять засоби інтеграції із зовнішніми сервісами, платіжними системами та аналітичними інструментами [22].

Розвиток цифрових технологій та постійне зростання ринку IT-стартапів обумовлюють необхідність створення сучасних вебзастосунків для онлайн-

інвестування. Такі системи повинні поєднувати зручність використання, високий рівень безпеки, гнучкість масштабування та ефективні засоби взаємодії між усіма учасниками інвестиційного процесу. Це визначає актуальність розробки програмного забезпечення для підтримки інвестування в IT-стартапи.

1.2 Аналіз структурних і функціональних особливостей систем онлайн-інвестування

Системи онлайн-інвестування є складними багатокомпонентними програмними комплексами, призначеними для автоматизації взаємодії між інвесторами, засновниками стартапів та адміністраторами платформи. Основною метою таких систем є формування єдиного цифрового середовища, у межах якого виконуються публікація інвестиційних кампаній, пошук перспективних проєктів, проведення фінансових операцій та обмін інформацією між учасниками інвестиційного процесу.

Сучасні платформи онлайн-інвестування мають багаторівневу структуру та включають велику кількість взаємопов’язаних компонентів. Як правило, система складається з клієнтської частини, серверної частини, бази даних, модулів інтеграції із зовнішніми сервісами та інфраструктурних компонентів [19].

Клієнтська частина використовується для взаємодії користувача з платформою через вебінтерфейс. На неї покладено відображення інформації, обробку дій користувачів та обмін даними із серверною частиною через API [9].

Серверна частина містить основну бізнес-логіку системи та виконує обробку запитів користувачів. У сучасних системах онлайн-інвестування дедалі частіше застосовується мікросервісний підхід, у межах якого функціональні модулі представлені незалежними сервісами з чітко визначеними зонами відповідальності. Подібна організація програмного забезпечення спрощує масштабування системи, супроводження окремих компонентів та впровадження нового функціоналу.

До основних структурних компонентів систем онлайн-інвестування належать:

1) модуль автентифікації та авторизації користувачів;
2) модуль управління профілями користувачів;
3) сервіс управління інвестиційними компаніями;
4) модуль платіжної інтеграції;
5) система сповіщень;
6) сервіс аналітики та формування звітності;
7) система модерації контенту;
8) система зберігання документів;
9) брокер повідомлень для асинхронної взаємодії між сервісами;
10) база даних та інфраструктурні сервіси.

Важливою структурною особливістю сучасних систем є використання API для взаємодії між компонентами. Для цього можуть використовуватися REST API та gRPC [16]. REST API сприяє взаємодії між сервісами через HTTP-запити, а gRPC реалізує високопродуктивний обмін даними між компонентами системи.

У розподілених системах важливе місце займає асинхронна взаємодія між сервісами. Для обміну повідомленнями використовуються брокери повідомлень, зокрема Apache Kafka, що застосовуються для реалізації event-driven architecture та підтримання стабільної роботи системи за умов великої кількості одночасних запитів. Для контейнеризації та розгортання сервісів використовуються сучасні технології, зокрема Docker [19]. Контейнеризація передбачає розміщення окремих компонентів у незалежних контейнерах, що спрощує процес розгортання та супроводження програмного забезпечення. Такий підхід полегшує масштабування окремих сервісів і зменшує вплив змін в одному компоненті на роботу інших частин системи.

Функціональні можливості сучасних платформ онлайн-інвестування включають:

1) реєстрацію та вхід користувачів за допомогою email і пароля;
2) OAuth-автентифікацію через зовнішні сервіси;
3) підтвердження email користувача;

4) управління профілем інвестора;
5) управління профілем стартапу;
6) підтримку ролей та прав доступу (RBAC);
7) каталог стартапів;
8) систему пошуку та фільтрації проєктів;
9) сторінку стартапу з інформацією про компанію;
10) завантаження та зберігання документів;
11) створення інвестиційної компанії;
12) модерацію компаній адміністратором;
13) створення заявки на інвестування;
14) інтеграцію платіжних сервісів;
15) управління інвестиційним портфелем;
16) дашборд стартапу;
17) підтримку статусів інвестиційних угод;
18) систему email та in-app сповіщень;
19) публікацію оновлень і звітності;
20) систему коментарів та питань-відповідей;
21) аналітику та формування звітів;
22) журналювання дій користувачів (audit log).

Однією з характерних особливостей систем онлайн-інвестування є підтримка різних ролей користувачів. Інвестори виконують пошук стартапів, здійснюють інвестиції та керують власним інвестиційним портфелем. Засновники стартапів створюють інвестиційні кампанії, взаємодіють з інвесторами та публікують відомості про розвиток проєкту. Адміністратори відповідають за модерацію контенту, перевірку користувачів та контроль роботи системи [15].

Важливе місце у сучасних платформах займають механізми безпеки та прозорість фінансових операцій. З цією метою використовуються KYC/AML-перевірки, системи журналювання дій користувачів, засоби шифрування даних та механізми контролю доступу до окремих функцій системи.

Таким чином, системи онлайн-інвестування характеризуються складною структурою, значною кількістю функціональних модулів та високими вимогами до масштабованості, безпеки й стабільності взаємодії між компонентами. Поєднання великої кількості сервісів та користувачів вимагає використання сучасних архітектурних підходів і технологій, здатних підтримувати надійне функціонування програмного забезпечення за умов постійного зростання навантаження.

1.3 Аналіз процесів взаємодії та комунікації користувачів у системі інвестування

Процеси взаємодії та комунікації між користувачами є одним із ключових елементів систем онлайн-інвестування. Від ефективності обміну інформацією залежить рівень довіри між учасниками інвестиційної діяльності, швидкість ухвалення рішень та успішність фінансування стартап-проєктів.

Основним процесом взаємодії є створення інвестиційної компанії засновником стартапу. Після проходження реєстрації та створення профілю користувач отримує доступ до публікації інформації про стартап та можу вказати фінансові цілі, умови інвестування та завантажити супровідні документи.

Інвестори взаємодіють із системою через каталог стартапів, використовуючи механізми пошуку та фільтрації. Після аналізу інформації про компанію користувач може здійснити інвестицію та відстежувати стан власного інвестиційного портфеля.

Важливим аспектом взаємодії користувачів є забезпечення прозорості інвестиційного процесу. Для цього система повинна надавати доступ до актуальної інформації про стан інвестиційних кампаній, результати залучення коштів та зміни статусів проєктів. Наявність такої інформації значно підвищує рівень довіри між учасниками та сприяє прийняттю обґрунтованих рішень.

Важливу роль у системі відіграє комунікація між учасниками. Для цього використовуються механізми повідомлень, email-сповіщень та системи оновлень

компаній. Засновники стартапів можуть публікувати новини, фінансові звіти та інші повідомлення для інвесторів.

Значна увага приділяється процесам модерації та перевірки користувачів. Для забезпечення безпеки та прозорості системи використовуються механізми KYC/AML. Ці інструменти перевіряють особу користувачів та мінімізувати ризики шахрайства.

У сучасних системах онлайн-інвестування важливе значення мають асинхронні механізми взаємодії між компонентами системи. Для цього можуть використовуватися брокери повідомлень та системи обробки подій, що підвищують продуктивність та забезпечують масштабованість програмного забезпечення.

1.4 Огляд та аналіз сучасних програмних рішень у сфері онлайн-інвестування

У сучасних умовах цифрової економіки існує значна кількість платформ для онлайн-інвестування в стартапи та інноваційні проєкти. Такі системи забезпечують взаємодію між інвесторами та засновниками стартапів, підтримують процеси залучення інвестицій, управління компаніями та виконання фінансових операцій.

Огляд та аналіз Wefunder

Wefunder є однією з найбільш відомих платформ для інвестування у стартапи на ранніх етапах розвитку. Розробником системи є компанія Wefunder Inc [22]. Платформа побудована з використанням трирівневої вебархітектури (3-tier web application), яка включає клієнтську частину, серверну частину та базу даних.

Для реалізації системи використовуються JavaScript для frontend-частини, а також Node.js і Django (Python) для backend-частини застосунку.

Основні функціональні компоненти платформи включають:

- реєстрацію інвесторів та стартапів;
- створення та подання інвестиційних компаній;
- перегляд інформації про стартапи;

- систему пошуку та фільтрації компаній;
- підтримку інвестування за моделями Reg CF та Reg A+.

Основною перевагою платформи є велика база інвесторів і стартапів, що формує активну взаємодію між учасниками системи. Ще однією перевагою є система, що підтримує різні інструменти інвестування та залучення фінансування.

До недоліків платформи можна віднести відсутність гнучкої системи персоналізації рекомендацій для інвесторів, а також перевантажений інтерфейс із великою кількістю інформації, що ускладнює швидке прийняття рішень користувачами. У розроблюваному вебзастосунку дані проблеми вирішуються шляхом реалізації персоналізованих рекомендацій на основі інтересів користувача, а також створення спрощеного та інтуїтивного інтерфейсу.

Огляд та аналіз Crowdcube

Crowdcube є однією з найбільш популярних платформ онлайн-інвестування у Європі. Розробником системи є компанія Crowdcube Ltd [8]. Платформа реалізована за принципом трирівневої вебархітектури. Frontend-частина системи побудована з використанням TypeScript, React, Redux та Styled Components. Для взаємодії між компонентами використовується REST-based backend.

Основні функції системи:

- створення інвестиційних компаній;
- підтримка мінімального інвестиційного внеску;
- інформаційні сторінки компаній;
- перегляд фінансових даних та документів;
- підтримка інструментів залучення інвесторів.

Основною перевагою Crowdcube є наявність детальної інформації про компанії та документів, необхідних для аналізу інвестиційних ризиків.

До недоліків системи належить недостатня гнучкість фільтрації стартапів за галуззю, рівнем ризику або стадією розвитку, а також відсутність гнучкої системи ролей і персоналізації. У власному вебзастосунку дані проблеми вирішуються

шляхом реалізації розширеної системи пошуку та фільтрації, а також підтримки гнучкої системи ролей користувачів.

Огляд та аналіз StartEngine

Платформа StartEngine розроблена компанією StartEngine Crowdfunding, Inc [21]. Система використовує трирівневу вебархітектуру з елементами мікросервісного підходу. Для реалізації backend-частини використовуються Python та Java. Інфраструктура платформи побудована із застосуванням мікросервісної архітектури, Docker та Apache Kafka.

Основні функції платформи:

- публікація компаній стартапів;
- підтримка моделей Reg CF та Reg A+;
- вторинний ринок для часток;
- панелі управління для інвесторів і стартапів;
- аналітичні інструменти та система сповіщень.

Перевагою StartEngine є широкий функціонал для управління інвестиціями та підтримка масштабованої інфраструктури. Недоліком системи є перевантажений інтерфейс та ускладнений доступ до ключових функцій платформи. У власному вебзастосунку передбачено реалізацію спрощеного доступу до основних функцій системи. Це надає користувачам без досвіду покращення навігації, з чим часто зіштовхуються нові клієнти застосунків.

Огляд та аналіз Republic

Republic є сучасною платформою для інвестування у стартапи, криптовалютні активи та інші цифрові продукти. Розробником системи є Republic Group [20]. Платформа має трирівневу вебархітектуру з елементами FinTech та Web3-рішень.

Основні функції платформи:

- інвестування у стартапи та токенізовані активи;
- підтримка Reg CF;
- робота з криптовалютними активами;

- профілі компаній;
- аналітика та система сповіщень.

Перевагами системи є підтримка різних типів активів та велика кількість користувачів. Основним недоліком Republic є перевантажений інтерфейс із великою кількістю інформації, що ускладнює роботу користувача із системою.

Порівняльний аналіз сучасних платформ онлайн-інвестування

Проведений аналіз сучасних платформ онлайн-інвестування показав, що більшість існуючих рішень забезпечують широкий функціонал для взаємодії між інвесторами та стартапами, підтримують фінансові операції, модулі аналітики та систему сповіщень.

Таблиця 1.1 - Порівняльний аналіз сучасних платформ онлайн-інвестування

Характеристики	Wefunder	Crowdcube	StartEngine	Republic
Архітектура	Трирівнева вебархітектура	Трирівнева вебархітектура	Трирівнева архітектура з елементами мікросервісного підходу	Трирівнева архітектура з FinTech-компонентами
Технології	JavaScript, Node.js, Django	React, TypeScript, Redux	Python, Java, Docker, Kafka	Власні FinTech та Web3-рішення
Основне призначення	Інвестування у стартапи ранніх стадій	Краудінвестинг у компанії	Залучення інвестицій та вторинний ринок часток	Інвестування у стартапи та цифрові активи
Масштабованість	Обмежена централізованою архітектурою	Залежить від backend-компонентів	Висока завдяки мікросервісам	Середня
Основні недоліки	Перевантажений інтерфейс, недостатня персоналізація	Обмежені можливості фільтрації та ролей	Складний інтерфейс	Велика кількість інформації та складність навігації

Таким чином, проведений аналіз сучасних програмних рішень підтверджує актуальність розробки власної платформи онлайн-інвестування, орієнтованої на покращення користувацького досвіду, забезпечення масштабованості та підтримку сучасних механізмів взаємодії між сервісами.

1.5 Аналіз сучасних архітектурних підходів до розробки вебзастосунків

Одним із традиційних підходів до розробки вебзастосунків є монолітна архітектура, у якій усі компоненти системи реалізуються як єдиний програмний модуль. У такій архітектурі клієнтська частина, бізнес-логіка та робота з базою даних тісно пов’язані між собою та розгортаються як один застосунок.

Основними перевагами монолітної архітектури є [18]:

1) простота розробки на початкових етапах;
2) простота розгортання та тестування;
3) централізоване управління компонентами системи;
4) менша складність інфраструктури.

Разом із тим монолітна архітектура має низку суттєвих недоліків:

1) складність масштабування окремих компонентів;
2) залежність усіх модулів системи один від одного;
3) ускладнення підтримки при збільшенні функціональності;
4) складність оновлення та внесення змін;
5) зниження відмовостійкості системи;
6) складність роботи великих команд розробників.

У сучасних високонавантажених системах вже значного розповсюдження набула мікросервісна архітектура. Даний підхід передбачає поділ системи на набір незалежних сервісів, кожен із яких виконує окрему функцію та має власну зону відповідальності.

Основними перевагами мікросервісної архітектури є:

1) незалежне масштабування сервісів;
2) гнучкість розробки та підтримки;

3) незалежне оновлення компонентів;
4) підвищення відмовостійкості системи;
5) зручність роботи великих команд розробників;
6) підтримка різних технологій у межах однієї системи;
7) спрощення повторного використання компонентів;
8) використання підходу event-driven architecture.

Недоліками мікросервісної архітектури є:

1) складність проєктування системи;
2) ускладнення взаємодії між сервісами;
3) необхідність використання додаткової інфраструктури;
4) складність моніторингу та тестування;
5) збільшення кількості мережевих запитів;
6) складність забезпечення узгодженості даних між сервісами.

Для взаємодії між мікросервісами використовуються REST API та gRPC. REST API спрощує HTTP-взаємодію між компонентами системи та широко використовується для інтеграції вебзастосунків. gRPC є сучасним високопродуктивним протоколом взаємодії між сервісами, який оптимізує обмін даними та ефективну роботу розподілених систем завдяки своєму алгоритму.

У сучасних розподілених системах важливу роль відіграє event-driven architecture, яка базується на асинхронному обміні повідомленнями між сервісами. Для реалізації такого підходу використовуються брокери повідомлень, такі, як Apache Kafka. Використання Kafka призначене для підвищення масштабованості системи, зменшення залежність між сервісами та надає стабільну обробку великої кількості подій.

Для контейнеризації компонентів системи використовується технологія Docker. Контейнеризація ізолює сервіси, спрощує процес розгортання та створює умови для незалежного масштабування окремих компонентів системи. Ще однією сильною стороною Docker є те, що він значно спрощує підтримку програмного забезпечення та гарантує стабільність середовища виконання на будь якому

програмному забезпеченні, так як у світі розповсюджене використання різних систем.

Таким чином, проведений аналіз сучасних архітектурних підходів показав, що використання мікросервісної архітектури є доцільним для створення сучасних масштабованих вебзастосунків у сфері онлайн-інвестування. Використання даних технологій надає гнучкість, масштабованість, відмовостійкість та ефективну взаємодію між компонентами системи.

Висновки до розділу 1

У першому розділі було проведено аналіз предметної області інвестування в IT-стартапи, досліджено структурні та функціональні особливості систем онлайн-інвестування, а також проаналізовано процеси взаємодії та комунікації користувачів у системі.

Було виконано огляд сучасних платформ онлайн-інвестування та визначено їх основні переваги й недоліки. Проведений аналіз показав, що сучасні системи повинні забезпечувати високий рівень масштабованості, безпеки та підтримки великої кількості користувачів.

У розділі, проведений аналіз, на результатах якого визначено основні вимоги до програмного забезпечення та сформувати концепцію майбутньої системи. Отримані результати стали основою для подальшого моделювання предметної області, розробки специфікації вимог та проєктування архітектури вебзастосунку інвестиційної платформи. Це стало підґрунтям для переходу до наступних етапів розробки та реалізації функціоналу системи.

Також було розглянуто сучасні архітектурні підходи до розробки вебзастосунків та обґрунтовано доцільність використання мікросервісної архітектури для реалізації інвестиційної платформи.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі виконується моделювання предметної області вебзастосунку для онлайн-інвестування в IT-стартапи, визначаються функціональні та нефункціональні вимоги до програмного забезпечення, а також описуються методи, технології та засоби, що використовуються для реалізації системи.

2.1 Моделювання предметної області вебзастосунку

Моделювання предметної області є важливим етапом проєктування програмного забезпечення, оскільки від цього залежить хід майбутньої розробки продукту. Це допомагає розробникам визначити основні компоненти системи, взаємозв’язки між ними та логіку роботи вебзастосунку до того, як почнеться розробка. Для моделювання предметної області системи онлайн-інвестування було обрано UML-нотації, що є сучасним підходом до аналізу програмного забезпечення.

Предметна область вебзастосунку охоплює процеси взаємодії між інвесторами, засновниками стартапів та адміністраторами системи. Основними об’єктами системи є користувачі, інвестиційні компанії, фінансові операції, повідомлення, документи та аналітичні дані.

У системі передбачено використання ролей користувачів:

- інвестор;
- засновник стартапу;
- адміністратор.

Інвестори можуть переглядати каталог стартапів, аналізувати інформацію про компанії. На основі отриманої інформації здійснюється інвестування та управління власним інвестиційним портфелем. Засновники стартапів можуть створювати інвестиційні компанії, завантажувати документи, публікувати оновлення та взаємодіяти з інвесторами, завдяки функції питань. Адміністратори

відповідають за модерацію контенту, управління користувачами та контроль роботи системи.

Для моделювання предметної області обрано UML-діаграми, зокрема обрано такі типи, діаграми варіантів використання, класів, компонентів, послідовності та розгортання. Використання UML спрощує формалізування структуру системи та описання взаємодію між компонентами ще до того, як почнеться робота над проєктом.

Система будується на основі мікросервісної архітектури. Основними сервісами системи є auth-service, iam-service, payments-service, notifications-service, activities-service, blockchain-gateway-service та athanor-service. Взаємодія між сервісами здійснюється через REST API, gRPC та Apache Kafka [16].

Для контейнеризації сервісів використовується Docker, а для організації асинхронних процесів – Temporal [17]. Зберігання даних здійснює PostgreSQL [14].

2.2 Опис методів та технологій

Для реалізації вебзастосунку використовуються сучасні методи проєктування програмного забезпечення, технології побудови розподілених систем та засоби моделювання архітектури.

Основним підходом до побудови системи було обрано використання мікросервісної архітектури. Такий підхід розділяє систему на незалежні сервіси з окремими зонами відповідальності, що орієнтовано на масштабованість, гнучкість, що спрощує підтримку програмного забезпечення та її розробку, на більш пізньому етапі.

Для взаємодії між сервісами використовуються такі протоколи, як REST API та gRPC. REST API спрощує HTTP-взаємодію між різними компонентами системи, а gRPC використовується для високопродуктивного обміну даними між сервісами, завдяки більш оптимізованому алгоритму передачі даних [9].

У системі використовується event-driven architecture із брокером повідомлень Apache Kafka. Даний підхід реалізує асинхронний обмін повідомленнями між сервісами та гарантує стабільну роботу системи при великій кількості одночасних запитів.

Для контейнеризації сервісів використовується Docker. Контейнеризація ізолює компоненти системи та спрощує процес розгортання програмного забезпечення на будь якому програмному забезпеченні.

Для зберігання даних використовується PostgreSQL [4]. Цей інструмент є стандартом для розробки подібних систем і вже зарекомендував себе, як надійна технологія. Система підтримує зберігання інформації про користувачів, інвестиційні компанії, фінансові операції та інші об’єкти предметної області.

Методи проєктування програмного забезпечення

Під час розробки системи використовуються сучасні методи проєктування програмного забезпечення, які вже зарекомендували себе.

Основним методом проєктування є підхід, з використанням мікросервісної архітектури. Система розділяється на незалежні функціональні сервіси. Кожен сервіс реалізує окрему бізнес-логіку та може розроблятися незалежно від інших компонентів системи.

При проєктуванні системи використовуються принципи:

- модульності;
- слабкої зв’язаності компонентів;
- незалежного масштабування сервісів;
- розподіленої обробки даних.

Технології взаємодії між сервісами

Для забезпечення взаємодії між компонентами системи використовуються REST API, gRPC та Apache Kafka.

Протокол REST API використовується для взаємодії між клієнтською частиною та backend-сервісами через HTTP-запити. Основною перевагою REST API є простота інтеграції та підтримка великої кількості клієнтських платформ.

Для високопродуктивної взаємодії між мікросервісами використовується gRPC. Даний протокол підтримує швидкий обмін даними та підтримує двосторонню комунікацію між сервісами.

Для реалізації асинхронної взаємодії між сервісами використовується Apache Kafka. Брокер повідомлень реалізує event-driven architecture та забезпечити стабільну обробку подій у розподіленій системі.

Засоби контейнеризації та оркестрації

Для контейнеризації програмного забезпечення використовується Docker. Цей інструмент надає незалежне розміщення окремих компонентів системи, що сприяє стабільності середовища виконання та спрощує процес розгортання сервісів.

Кожен мікросервіс системи розгортається у власному контейнері. Такий підхід має декілька переваг, дозволяє незалежне оновлення, масштабування окремих компоненти системи та підвищує відмовостійкість, оскільки замість падіння усього застосунку у разі критичної помилку, перестане працювати лише один сервіс.

Для організації workflow та управління асинхронними процесами використовується Temporal. Даний інструмент реалізовує механізми обробки довготривалих операцій та захищає програму від припинення функціонування, коли виникає помилка.

2.3 Специфікація вимог до програмного забезпечення

Призначення системи

Вебзастосунок призначений для автоматизації процесів онлайн-інвестування в IT-стартапи та забезпечення взаємодії між інвесторами, засновниками стартапів і адміністраторами системи. Система виконує публікацію інвестиційних компаній, управління профілями користувачів, проведення фінансових операцій, аналітику та підтримку комунікації між учасниками платформи.

Погодження, ухвалені в програмній документації

Під час розробки програмного забезпечення прийнято такі погодження:

- система реалізується як вебзастосунок;
- використовується мікросервісна архітектура;
- взаємодія між сервісами здійснюється через REST API, gRPC та Apache Kafka;
- система підтримує контейнеризацію за допомогою Docker;
- зберігання даних здійснюється у PostgreSQL.

Межі проєкту

Проєкт передбачає розробку вебзастосунку для онлайн-інвестування в IT-стартапи. У межах проєкту реалізуються функції реєстрації користувачів, створення компаній, інвестування, сповіщень, аналітики та управління користувачами.

Загальний опис

Сфера застосування

Система використовується у сфері онлайн-інвестування та призначена для автоматизації процесів взаємодії між інвесторами та засновниками стартапів. Вебзастосунок включає підтримку процесів створення інвестиційних кампаній, управління профілями користувачів, проведення фінансових операцій та обміну інформацією між учасниками системи.

Характеристики користувачів

Основними користувачами системи є:

- інвестори;
- засновники стартапів;
- адміністратори системи.

Користувачі повинні мати базові навички роботи з вебзастосунками та доступ до мережі Інтернет. Інвестори використовують систему для пошуку перспективних проєктів та здійснення інвестицій, засновники стартапів – для створення та управління інвестиційними кампаніями, а адміністратори відповідають за модерацію контенту та контроль функціонування платформи. Для

ефективної роботи із системою не вимагається спеціальної технічної підготовки, оскільки інтерфейс орієнтований на широке коло користувачів.

Загальна структура і склад системи

Система складається з:

- клієнтської частини;
- backend-сервісів;
- бази даних;
- системи автентифікації;
- системи сповіщень;
- брокера повідомлень Apache Kafka;
- системи управління workflow Temporal.

Загальні обмеження

Система потребує постійного підключення до мережі Інтернет. Продуктивність системи залежить від серверної інфраструктури та навантаження на сервіси. Коректна робота вебзастосунку також залежить від доступності зовнішніх сервісів та стабільності мережевого з’єднання. У разі виникнення збоїв або перевищення допустимого навантаження окремі функції системи можуть бути тимчасово недоступними.

Функції системи

Реєстрація та авторизація користувачів

Опис функції

Функція надає функціонал реєстрації користувачів, вхід до системи, OAuth-автентифікацію та підтвердження email. Вона є основою механізму контролю доступу до ресурсів платформи та призначена для ідентифікації користувачів. Ще однією функцією є підтримка механізму перевірки облікових записів та розмежування прав доступу відповідно до ролі користувача.

Вхідна і вихідна інформація

Вхідна інформація:

- email;

- пароль;
- OAuth-дані.

Вихідна інформація:

- JWT-токен;
- інформація про користувача;
- статус авторизації.

Функціональні вимоги:

- реєстрація користувачів;
- вхід за email/паролем;
- OAuth-автентифікація;
- підтвердження email;
- підтримка JWT;
- підтримка ролей користувачів.

Каталог та пошук стартапів

Опис функції

Функція виконує створення, перегляд та редагування профілів користувачів системи. Цей функціонал зберігає персональні дані інвесторів і засновників стартапів, а також підтримує механізми розмежування прав доступу та перевірки KYC-статусу.

Вхідна і вихідна інформація

Вхідна інформація:

- параметри пошуку;
- фільтри.

Вихідна інформація:

- список стартапів;
- інформація про компанії.

Функціональні вимоги:

- перегляд каталогу стартапів;
- пошук стартапів;

- фільтрація за галуззю;
- фільтрація за стадією розвитку;
- фільтрація за країною;
- фільтрація за рівнем ризику;
- сортування результатів.

Управління інвестиційними компаніями

Опис функції

Функція створює та модерирує інвестиційних компаній. Вона дозволяє засновникам стартапів публікувати інформацію про власні проєкти, визначати умови залучення інвестицій та керувати життєвим циклом кампанії. Також, функція підтримує процес перевірки та затвердження кампаній адміністраторами системи.

Вхідна і вихідна інформація

Вхідна інформація:

- назва компанії;
- опис;
- фінансова ціль;
- мінімальний чек;
- дедлайн;
- документи.

Вихідна інформація:

- статус компанії;
- ID компанії.

Функціональні вимоги:

- створення компанії;
- редагування компанії;
- завантаження документів;
- approve/reject компаній;
- підтримка статусів:

- draft;
- review;
- active;
- funded;
- failed;
- closed.

Здійснення інвестицій

Опис функції

Функція дозволяє створення інвестиційних заявок та виконання фінансових операцій. Вона надає змогу інвесторам здійснювати вкладення коштів у вибрані стартапи та відстежувати результати виконаних операцій. Ще одним функціоналом є забезпечення контролю над станом інвестицій та збереженням інформації про проведені транзакції.

Вхідна і вихідна інформація

Вхідна інформація:

- ID компанії;
- сума інвестиції;
- платіжні дані.

Вихідна інформація:

- статус транзакції;
- інформація про інвестицію.

Функціональні вимоги:

- створення заявки на інвестицію;
- підтвердження умов інвестування;
- платіжна інтеграція;
- підтримка історії транзакцій;
- перевірка KYC.

Система сповіщень та комунікації

Опис функції

Функція охоплює надсилання повідомлень та взаємодію між користувачами. Підсистема комунікації використовується для інформування користувачів про зміни стану інвестиційних кампаній, результати виконання операцій та інші важливі події. Також у системі реалізовано механізми обміну інформацією між інвесторами та засновниками стартапів, що підтримує постійний зв’язок між учасниками інвестиційної діяльності.

Вхідна і вихідна інформація

Вхідна інформація:

- тип події;
- ID користувача;
- текст повідомлення.

Вихідна інформація:

- email-повідомлення;
- in-app повідомлення.

Функціональні вимоги:

- email-сповіщення;
- in-app сповіщення;
- публікація оновлень стартапу;
- коментарі;
- система Q&A;
- історія повідомлень.

Вимоги до інформаційного забезпечення

Джерела і зміст вхідної інформації

Джерелами вхідної інформації є дані користувачів, інформація про стартапи, фінансові дані, документи компаній та повідомлення системи. Також джерелами даних можуть виступати результати роботи окремих мікросервісів та зовнішніх сервісів, інтегрованих із платформою. Уся вхідна інформація проходить перевірку на коректність перед її подальшою обробкою та збереженням.

Нормативно-довідкова інформація

У системі використовуються довідники ролей користувачів, класифікатори статусів компаній, типів повідомлень, довідники статусів інвестицій, типів транзакцій та параметрів системних сповіщень. Нормативно-довідкова інформація сприяє уніфікації даних та спрощенню обробки інформації в межах системи.

Вимоги до способів організації, збереження та ведення інформації

Зберігання інформації здійснюється у PostgreSQL. У системі реалізовано механізми резервного копіювання та захисту конфіденційних даних. Для підтримання цілісності інформації застосовуються транзакції та обмеження цілісності бази даних. У випадку виникнення збоїв або відмов окремих компонентів передбачено процедури відновлення даних та повернення системи до нормального режиму роботи.

Вимоги до технічного забезпечення

Для роботи системи необхідні серверна інфраструктура, Docker, мережеве підключення та сервер баз даних PostgreSQL.

Апаратне забезпечення повинно забезпечувати стабільну роботу мікросервісів та бази даних. Конфігурація серверної інфраструктури може змінюватися залежно від кількості користувачів та рівня навантаження.

Вимоги до програмного забезпечення

Архітектура програмної системи

Система побудована на основі мікросервісного підходу. На відміну від монолітної архітектури, окремі функції розподілені між незалежними сервісами, що спрощує підтримку та модернізацію програмного забезпечення. Такий підхід також створює умови для незалежного масштабування компонентів, зменшує вплив збоїв окремих сервісів на роботу системи та спрощує інтеграцію нових функцій.

Системне програмне забезпечення:

- Docker;
- Nginx.

Docker використовується для контейнеризації окремих сервісів та спрощення процесу розгортання програмного забезпечення. Nginx виконує функції вебсервера та зворотного проксі, забезпечуючи маршрутизацію запитів і балансування навантаження між компонентами системи.

Мережне програмне забезпечення:

- HTTPS;
- WebSocket.

Використання HTTPS гарантує захищений обмін даними між клієнтською та серверною частинами системи. Протокол WebSocket використовується для організації двосторонньої взаємодії в режимі реального часу та реалізації механізмів сповіщень.

Програмне забезпечення ведення інформаційної бази

Для роботи з даними використовується PostgreSQL.

Мова і технологія розробки

Frontend:

- React;
- TypeScript;
- Tailwind CSS;
- React Router;
- Axios.

Backend:

- Golang;
- REST API;
- gRPC;
- JWT-auth;
- entgo.

Інфраструктура:

- Docker;
- Apache Kafka;

- Temporal.

Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

Інтерфейс повинен забезпечувати зручну навігацію та швидкий доступ до основних функцій системи. Також передбачається підтримка адаптивного відображення на різних типах пристроїв.

Апаратний інтерфейс

Система орієнтована на роботу на персональних комп'ютерах і мобільних пристроях. Для її використання не потрібне спеціалізоване обладнання. Доступ до вебзастосунку здійснюється за допомогою будь-якого сучасного браузера, який підтримує актуальні вебтехнології.

Програмний інтерфейс

Для взаємодії між сервісами використовуються REST API та gRPC API. Програмні інтерфейси застосовуються для обміну даними між компонентами системи та інтеграції із зовнішніми сервісами. API формують єдиний механізм доступу до функцій програмного забезпечення та підтримують стандартизований формат взаємодії між окремими компонентами системи.

Комунікаційний протокол

Для взаємодії між компонентами системи використовуються HTTPS, HTTP/2 та Kafka Protocol. Використання зазначених протоколів включає надійний та безпечний обмін даними між мікросервісами. Також, підтримка асинхронної взаємодії через Apache Kafka сприяє підвищенню продуктивності та масштабованості системи.

Властивості програмного забезпечення

Супроводжуваність

Архітектура системи повинна забезпечувати простоту оновлення та підтримки компонентів.

Використання мікросервісної архітектури створює умови для виконання незалежного оновлення окремих компонентів системи. Це спрощує подальший розвиток та модернізацію програмного забезпечення.

Надійність

Система повинна зберігати працездатність у разі відмови окремих компонентів та відновлювати роботу після виникнення помилок. Для цього використовуються механізми журналювання та оркестрації, які підтримують стабільність функціонування програмного забезпечення.

Безпека

Для захисту інформації користувачів використовуються механізми шифрування та автентифікації. Також передбачено ведення журналів подій для контролю дій користувачів та адміністраторів.

Висновки до розділу 2

У другому розділі було виконано моделювання предметної області вебзастосунку та сформовано специфікацію вимог до програмного забезпечення. Визначено функціональні та нефункціональні вимоги до системи, а також вимоги до програмного, технічного та мережного забезпечення.

Було визначено основні функції системи, зокрема реєстрацію користувачів, створення інвестиційних компаній, здійснення інвестицій та систему сповіщень. Обґрунтовано використання мікросервісної архітектури, REST API, gRPC, Apache Kafka та Docker для реалізації програмного забезпечення.

Також у межах другого розділу було виконано моделювання структури системи та визначено основні компоненти мікросервісної архітектури вебзастосунку. Проведене моделювання полегшує формування основи для подальшого проєктування програмного забезпечення, розробки UML-діаграм та реалізації взаємодії між окремими сервісами системи.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі представлено результати проєктування архітектури програмного забезпечення вебзастосунку інвестиційної платформи, моделювання функцій системи та опис основних компонентів програмного забезпечення. Також у розділі наведено UML-діаграми, що відображають структуру, взаємодію та процеси роботи системи.

Під час проєктування програмного забезпечення використовувались принципи мікросервісної архітектури, модульності та незалежності компонентів системи. Для опису структури та поведінки програмного забезпечення використовувались UML-нотації та CASE-засоби моделювання.

3.1 Розробка архітектури програмного забезпечення

Архітектура програмного забезпечення базується на мікросервісній архітектурі, яка розділяє систему на окремі сервіси з різними рівнями відповідальності. Кожен сервіс має бізнес-функцію і взаємодіє з рештою системи через REST API, gRPC та брокер повідомлень Apache Kafka.

Основними компонентами системи є:

- gateway-service;
- auth-service;
- users-service;
- startups-service;
- campaigns-service;
- investments-service;
- documents-service;
- notifications-service;
- communications-service;
- payments-service;
- analytics-service.

Кожен сервіс має свою зону відповідальності і може бути розгорнутий незалежно від інших компонентів системи. Docker використовується для контейнеризації сервісів. Взаємодія сервісів здійснюється за допомогою REST API та gRPC, а Apache Kafka використовується для асинхронного обміну повідомленнями.

PostgreSQL використовується для зберігання даних, причому кожен мікросервіс має власну базу даних. Для управління робочими процесами та виконання довготривалих операцій застосовується Temporal. Така організація програмного забезпечення спрощує подальший розвиток системи та інтеграцію нових сервісів без значного впливу на вже реалізовані компоненти. Порівняно з монолітною архітектурою, мікросервісний підхід характеризується більшою гнучкістю та кращою пристосованістю до масштабування.

3.2 Розробка UML-діаграм програмного забезпечення

Для моделювання структури та поведінки програмного забезпечення було розроблено декілька UML-діаграм.

3.2.1 CASE-засоби та UML-нотації

CASE-засоби полегшують процес проєктування систему, побудову UML-діаграм та документування архітектури програмного забезпечення.

У межах кваліфікаційної роботи використовуються:

- UML-діаграма варіантів використання;
- UML-діаграма класів;
- UML-діаграма послідовності;
- UML-діаграма розгортання.

Використання UML-нотацій перед та під час розробки формалізує структуру системи, описує взаємодію між компонентами та спрощую процес подальшої реалізації програмного забезпечення, що значно полегшує розробку та дає розробникам уяву, як має виглядати кінцева версія продукту.

3.2.2 UML-діаграма варіантів використання

Діаграма варіантів використання відображає взаємодію користувачів із системою та основні функції вебзастосунку.

На діаграмі представлені такі актори:

– інвестор;

– засновник стартапу;

– адміністратор.

На рисунку зображено UML-діаграму варіантів використання. Діаграма відображає основних користувачів системи, їх взаємодію з функціями вебзастосунку та зв’язки між окремими варіантами використання. Вона використовується для опису основних сценаріїв роботи системи та розподілу функцій між різними категоріями користувачів.

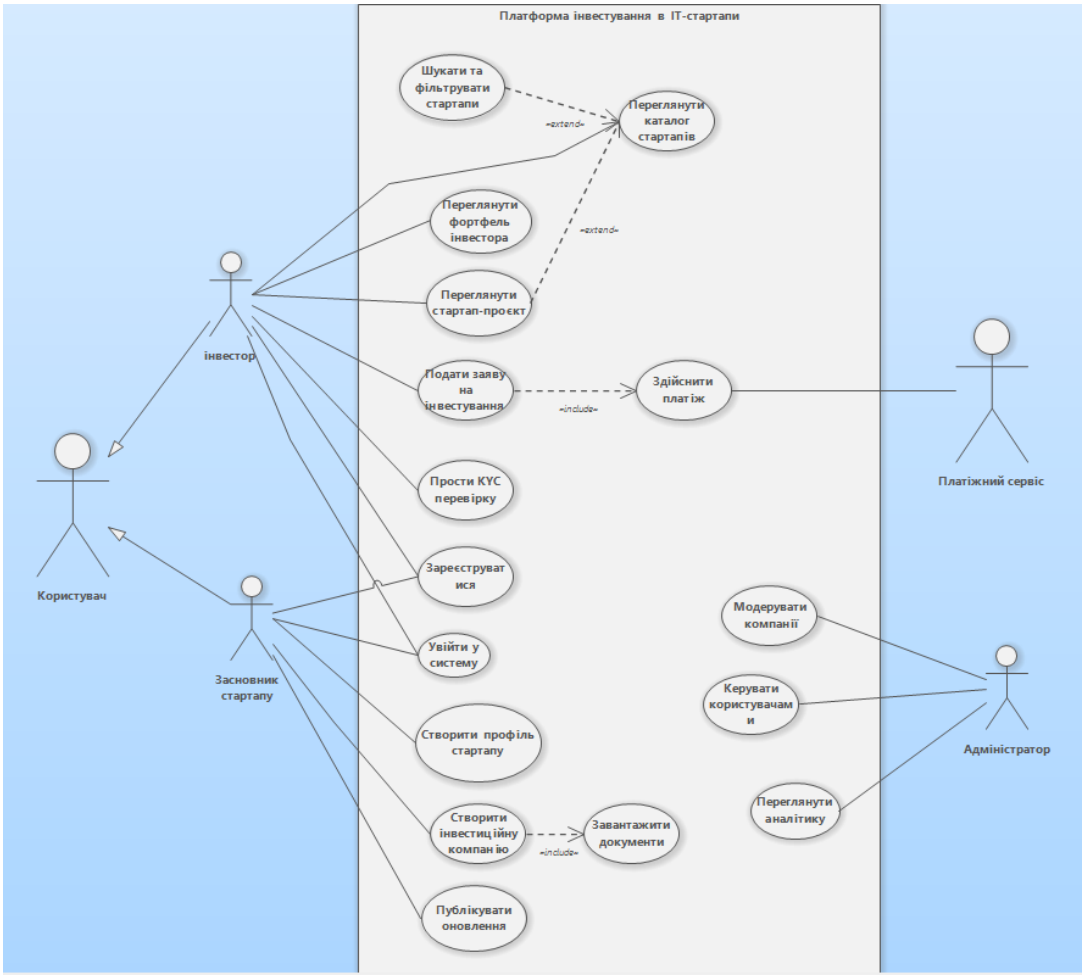

Рисунок 3.1 – UML-діаграма варіантів використання

Основними акторами системи є користувач, від якого успадковуються ролі інвестора та засновника стартапу. Окремими акторами виступають адміністратор системи та платіжний сервіс, що бере участь у виконанні фінансових операцій. Інвестор може переглядати каталог стартапів, виконувати пошук і фільтрацію проєктів, ознайомлюватися з інформацією про стартапи та станом власного інвестиційного портфеля, проходити KYC-перевірку, а також подавати заявки на інвестування з подальшим проведенням платежів. До функцій інвестора належать перегляд історії інвестицій, отримання сповіщень та управління власною інвестиційною діяльністю.

3.2.3 UML-діаграма класів

Діаграма класів описує структуру програмного забезпечення та взаємозв’язки між основними класами системи.

На діаграмі представлені класи:

- User;
- InvestorProfile;
- StartupProfile;
- Campaign;
- Investment;
- Transaction;
- Notification.

Клас User є базовим класом системи та містить основну інформацію про користувача, зокрема email, пароль і роль. Класи InvestorProfile та StartupProfile розширюють дані користувача відповідно до типу облікового запису. Клас Campaign описує інвестиційну кампанію стартапу, включаючи ціль збору коштів, статус і дедлайн.

Клас Investment використовується для збереження інформації про інвестиції користувачів, а Transaction – для обробки платіжних операцій. Клас Notification реалізує систему сповіщень про події платформи.

Між класами існують асоціативні зв’язки: один користувач може створювати декілька кампаній та інвестицій, а одна кампанія може містити багато інвестицій від різних користувачів.

На рисунку зображено UML-діаграму класів програмного забезпечення платформи інвестування в IT-стартапи. Діаграма відображає структуру основних класів системи, їх атрибути, методи та взаємозв’язки між окремими компонентами програмного забезпечення.

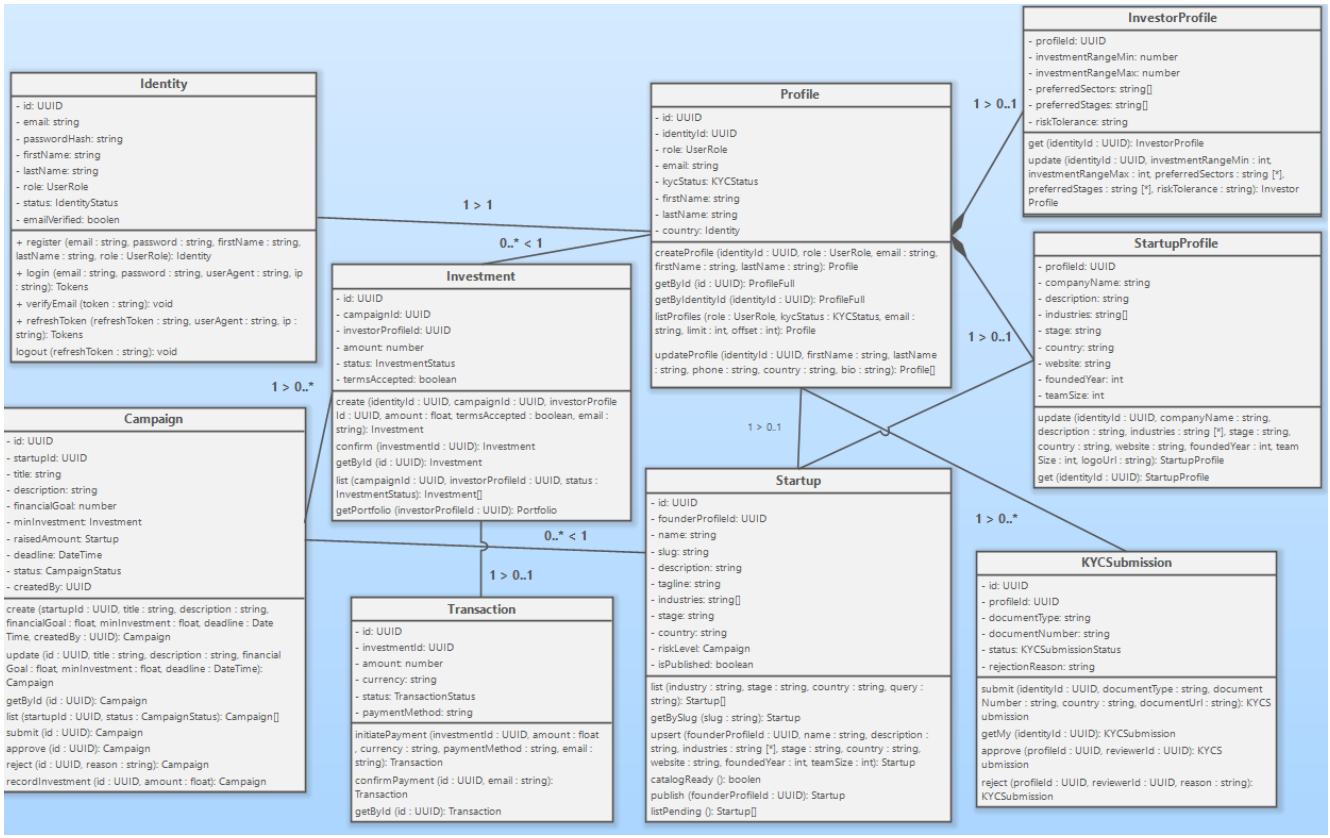

Рисунок 3.2 – UML-діаграма класів

На діаграмі зображено основні сутності системи, зокрема користувачі, профілі, стартапи, інвестиційні кампанії, платежі, заявки, повідомлення та інші компоненти, що забезпечують функціонування вебзастосунку. Для кожного класу визначено набір полів, які зберігають дані, а також методи, що реалізують бізнес-логіку системи.

3.2.4 UML-діаграма послідовності

Діаграма послідовності демонструє порядок взаємодії між компонентами системи під час виконання сценарію інвестування.

На рисунку зображено UML-діаграму послідовності процесу інвестування в IT-стартапи. Діаграма відображає послідовність взаємодії між користувачем системи та основними компонентами програмного забезпечення під час виконання операції інвестування.

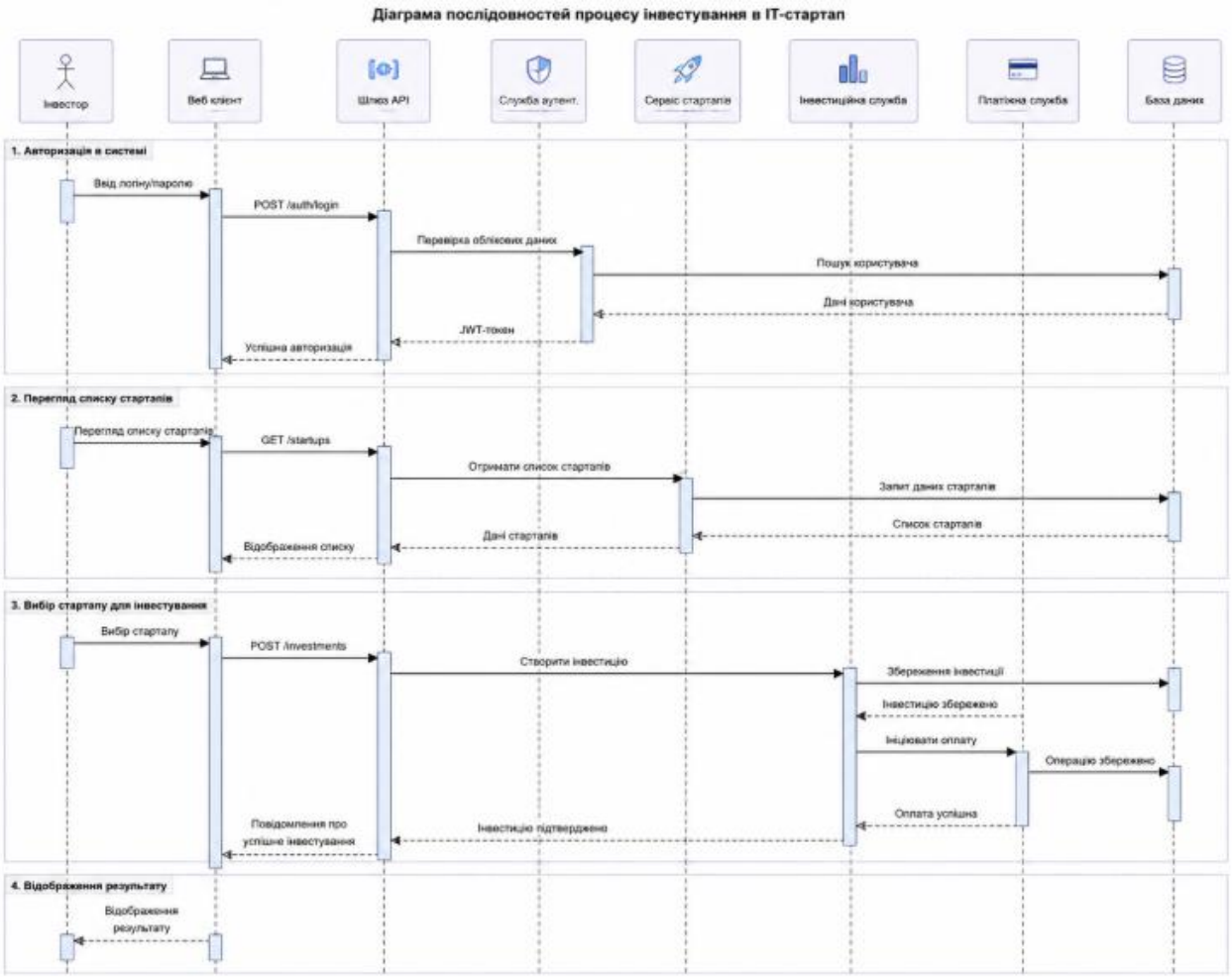

Рисунок 3.3 – UML-діаграма послідовності процесу інвестування

Діаграма демонструє послідовність обміну повідомленнями між інвестором, вебклієнтом, API Gateway, сервісами системи та базою даних. Представлена модель наочно показаує логіку роботи програмного забезпечення та взаємодію мікросервісів у процесі обробки інвестиційних операцій.

3.2.5 UML-діаграма розгортання

На рисунку зображено UML-діаграму розгортання системи платформи інвестування в IT-стартапи. Діаграма відображає фізичну структуру програмного забезпечення, взаємодію між основними сервісами та зовнішніми компонентами системи.

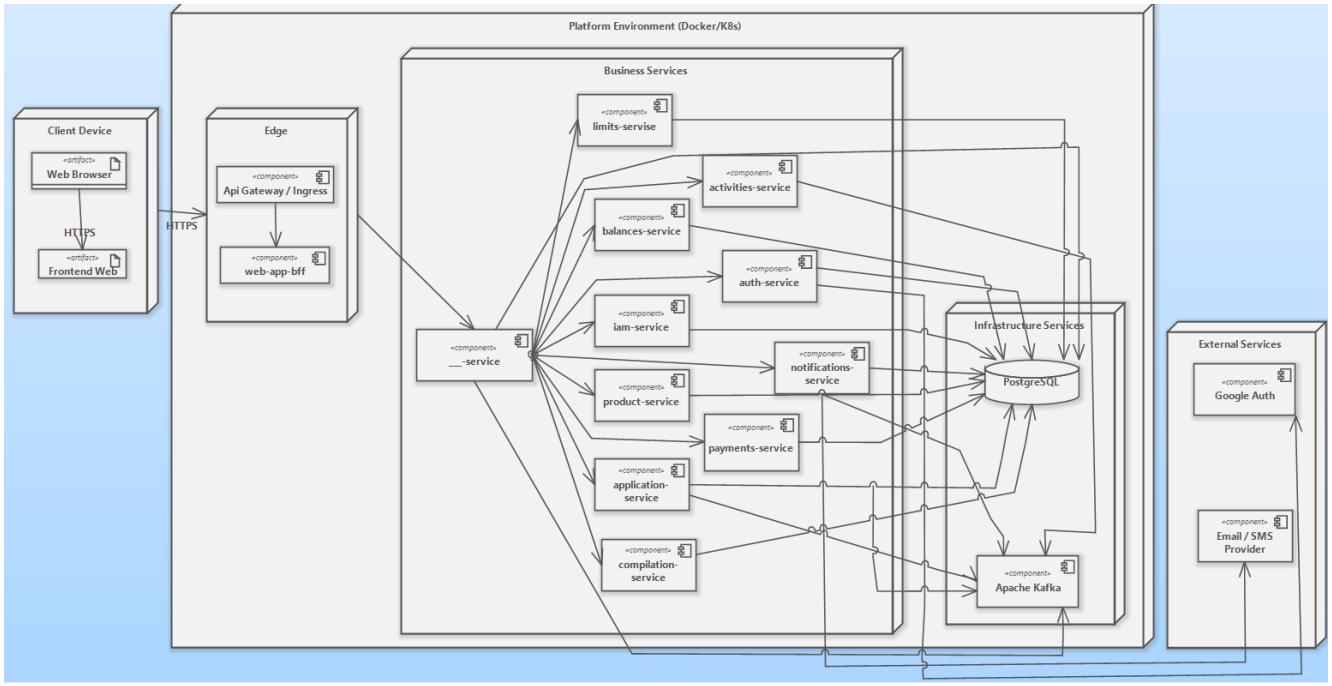

Рисунок 3.4 – UML-діаграма розгортання системи

На діаграмі показано взаємодію користувацького браузера з frontend-застосунком та gateway-service, через який здійснюється маршрутизація запитів до окремих мікросервісів системи. Також, діаграма відображає взаємодію сервісів із PostgreSQL, Redis, Apache Kafka та Temporal, що використовуються для збереження даних, кешування, асинхронного обміну повідомленнями та керування workflow-процесами.

3.2.6 UML-діаграма бази даних

На рисунку зображено UML-діаграму бази даних системи платформи інвестування в IT-стартапи. Діаграма відображає структуру основних сутностей бази даних, їх атрибути та взаємозв’язки між окремими компонентами системи.

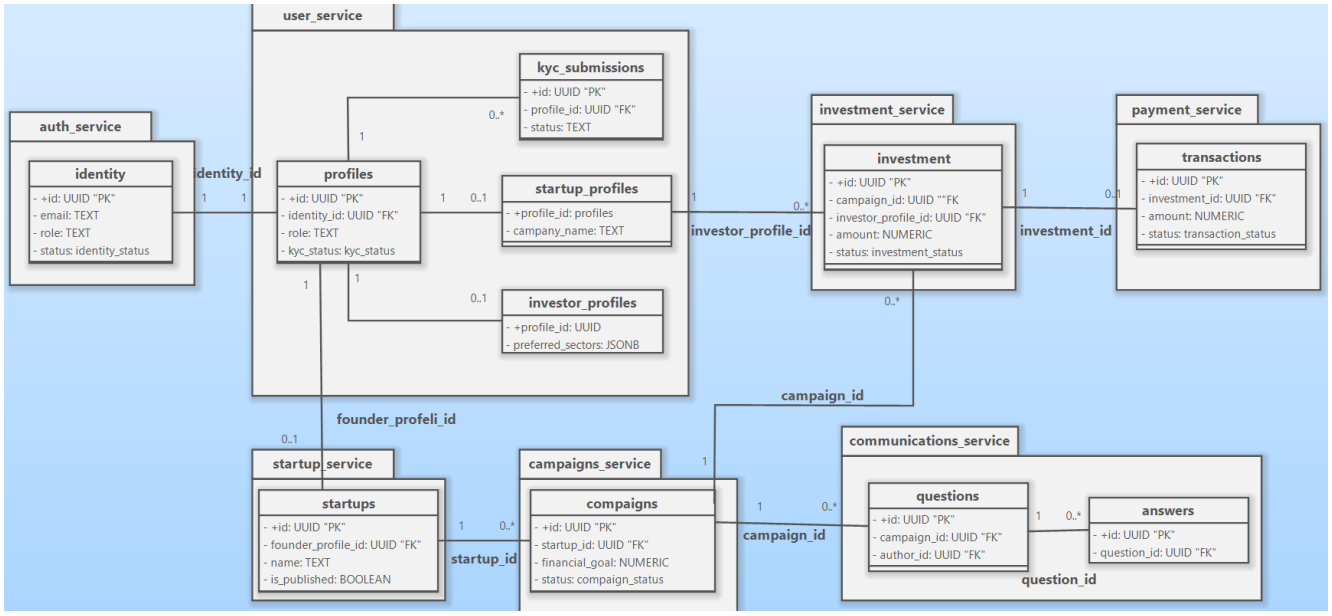

Рисунок 3.5 – UML-діаграма бази даних

На UML-діаграмі бази даних відображено реалізацію патерну Database per Service: для кожного мікросервісу створено окрему базу PostgreSQL (auth_service, users_service, startups_service, campaigns_service, investments_service, payments_service, documents_service, communications_service, notifications_service, analytics_service).

3.3 Вибір технологій та компонентів програмного забезпечення

Для реалізації вебзастосунку було обрано сучасні технології та компоненти програмного забезпечення.

Frontend-частина системи реалізується за допомогою:

- React;
- TypeScript.

Backend-частина системи реалізується мовою програмування Golang.

Для взаємодії між сервісами використовуються:

- REST API;
- gRPC;
- Apache Kafka.

Для контейнеризації використовується Docker, а для управління workflow – Temporal.

Основною системою керування базами даних є PostgreSQL.

Для реалізації безпеки системи використовуються:

- JWT;
- OAuth;
- RBAC.

Висновки до розділу 3

У третьому розділі було виконано проєктування архітектури програмного забезпечення вебзастосунку інвестиційної платформи. Розроблено структуру мікросервісної системи, визначено основні сервіси та механізми взаємодії між ними.

Також побудовано UML-діаграми, що моделюють структуру, поведінку та процеси роботи програмного забезпечення. Для реалізації системи обрано сучасні технології, такі як Golang, React, PostgreSQL, Docker, Apache Kafka та Temporal.

У межах розділу також було сформовано структуру бази даних, визначено взаємозв'язки між основними компонентами системи та описано принципи їх взаємодії. Проведене проєктування сформувало основу для подальшої реалізації програмного забезпечення та забезпечити відповідність функціональним і нефункціональним вимогам системи.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У четвертому розділі представлено результати практичної реалізації програмного забезпечення платформи інвестування в IT-стартапи, опис основних програмних компонентів, проведене тестування та аналіз його результатів.

4.1 Реалізація програмного забезпечення

Розроблена система реалізована як набір мікросервісів мовою програмування Go з використанням фреймворку Echo для HTTP API, gRPC для міжсервісної взаємодії, PostgreSQL як системи керування базами даних та Docker Compose для локального розгортання. Зовнішній доступ до системи здійснюється через єдину точку входу – API Gateway, який працює на порту 8080.

Таблиця 4.2 – Об'єкти та сутності програмного забезпечення

Сутність	Сервіс	Поля
Identity	auth-service	id, email, password_hash, role, status, email_verified, google_id
Profile	users-service	id, identity_id, role, email, kyc_status, first_name, last_name
InvestorProfile	users-service	profile_id, investment_range_min/max, preferred_sectors, risk_tolerance
StartupProfile	users-service	profile_id, company_name, industry, stage, country
Startup	startups-service	id, founder_profile_id, name, slug, industry, is_published
Campaign	campaigns-service	id, startup_id, title, financial_goal, status, deadline
Investment	investments-service	id, campaign_id, investor_profile_id, amount, status

Кінець таблиці 2

Сутність	Сервіс	Поля
Document	documents-service	id, owner_id, entity_type, entity_id, filename
Notification	notifications-service	id, user_id, type, channel, title, body, is_read
Comment / Question / Update	communications-service	id, author_id, entity/campaign/startup_id, content
AuditLog	analytics-service	id, user_id, action, entity_type, entity_id, metadata

У системі передбачено три основні ролі користувачів: INVESTOR, FOUNDER, ADMIN. Для інвестиційних компаній використовуються статуси draft, review, active, funded, failed, closed. Для облікових записів користувачів використовуються статуси ACTIVE, BLOCKED, PENDING_EMAIL.

4.1.1 Реалізація серверної частини

Серверна частина системи складається з 10 мікросервісів та спільної бібліотеки golib. Кожен мікросервіс має однакову внутрішню структуру:

- cmd/ – точка входу;
- internal/config/ – конфігурація;
- internal/app/ – бізнес-логіка;
- internal/repo/pq/ – робота з PostgreSQL;
- internal/handlers/http/ – HTTP handlers;
- internal/handlers/grpc/ – gRPC handlers;
- migrations/ – SQL-міграції.

Таблиця 4.1 – Реалізовані сервіси

Сервіс	HTTP-порт	Призначення
gateway-service	8080	Reverse proxy, JWT-автентифікація, Swagger
auth-service	8085	Реєстрація, login, JWT, refresh, verify email
users-service	8081	Профілі користувачів, RBAC
startups-service	8082	Каталог стартапів, пошук, фільтрація
campaigns-service	8083	CRUD компаній, submit, approve/reject
investments-service	8084	Заявки на інвестиції, портфель, dashboard
documents-service	8086	Завантаження та зберігання документів
notifications-service	8091	In-app та email-сповіщення
communications-service	8092	Коментарі, Q&A, оновлення стартапів
analytics-service	8093	Метрики, audit log

Для кожного сервісу створено окрему базу даних PostgreSQL, що відповідає підходу Database per Service. Ініціалізація баз виконується скриптом scripts/init-db.sql. Міграції застосовуються автоматично при старті контейнера через docker/local-entrypoint.sh.

Залежності між компонентами сервісів ін’єктуються через Uber FX. Для логування використовується structured logging на основі zap. Моніторинг сервісів здійснюється через Prometheus metrics на infra-порті 7777 кожного сервісу.

4.1.2 Реалізація клієнтської частини

Клієнтська частина системи реалізована як односторінковий застосунок (SPA) на базі React 18 і TypeScript. Збірка та локальна розробка виконуються за допомогою Vite; стилізація – Tailwind CSS. Маршрутизація здійснюється через

React Router, взаємодія з backend – через REST API (бібліотека Axios). Підтримуються дві мови інтерфейсу (українська та англійська) за допомогою i18next.

Застосунок розгортається окремо від мікросервісів і звертається до gateway-service за адресою http://localhost:8080/api/v1. У Docker-середовищі основний frontend доступний за адресою http://localhost:3000, адміністративна панель – http://localhost:3001.

Архітектура frontend-застосунку

Таблиця 4.2 – Структура фронтенду

Каталог/ модуль	**Призначення**
src/pages/	Сторінки публічної зони, кабінету інвестора та засновника
src/components/	Перевикористані UI-компоненти, графіки, layout
src/api/	HTTP-клієнт і функції доступу до REST API
src/context/	Глобальний стан автентифікації (AuthContext)
src/routes/	Захист маршрутів за роллю (ProtectedRoute)
src/hooks/	Кастомні хуки (membership, subscriptions)
src/i18n/	Локалізація (uk/en)
src/constants/	Довідники (галузі, етапи, країни)

Окремо реалізовано admin-frontend/ – спрощений застосунок для адміністратора з модерацією кампаній, KYC, користувачів, аналітики та audit log. Частина admin-сторінок використовує спільні компоненти з основного frontend.

Реалізовані інтерфейси

Таблиця 4.3 – Основні розділи клієнтського застосунку

Розділ	Маршрут	Роль	Призначення
Головна сторінка	/	PUBLIC	Огляд платформи та рекомендації
Проєкти	/projects	PUBLIC	Каталог стартапів

Кінець таблиці 5

Розділ	Маршрут	Роль	Призначення
Сторінка стартапу	/startups/:slug	PUBLIC / INVESTOR	Деталі проєкту та інвестування
Реєстрація / вхід	/register, /login	PUBLIC	Автентифікація користувачів
Панель інвестора	/app/investor/dashboard	INVESTOR	Аналітика портфеля
Інвестиції	/app/investor/investments	INVESTOR	Список інвестицій
KYC статус	/app/investor/kyc	INVESTOR	Верифікація інвестора
Панель засновника	/app/founder/dashboard	FOUNDER	Метрики кампанії
Компанії	/app/founder/campaigns	FOUNDER	Керування кампаніями
Розділ	Маршрут	Роль	Призначення
Стартап	/app/founder/startup	FOUNDER	Профіль компанії
Запитання	/app/founder/questions	FOUNDER	Відповіді інвесторам
Адмін панель	http://localhost:3001 (/campaigns, /kyc)	ADMIN	Модерація платформи

Взаємодія з API

Усі запити проходять через єдиний HTTP-клієнт apiClient (Axios). JWT access token зберігається в localStorage і додається до заголовка Authorization. При отриманні відповіді 401 застосунок автоматично оновлює токен через endpoint /auth/refresh.

API-модулі відповідають backend-сервісам:

- auth.ts – реєстрація, login, logout, refresh;
- profiles.ts – профіль, KYC, membership, subscriptions;
- startups.ts, campaigns.ts – каталог і кампанії;
- investments.ts – інвестиції, портфель, dashboard;
- communications.ts – питання/відповіді, коментарі, оновлення;

– documents.ts, notifications.ts, analytics.ts – документи, сповіщення, метрики.

Типи даних описані централізовано в src/api/types.ts, що гарантує типобезпеку на рівні TypeScript.

Автентифікація та авторизація

Стан користувача зберігається в AuthContext: роль (INVESTOR, FOUNDER, ADMIN), профіль і email. Компонент ProtectedRoute обмежує доступ до /app/* за роллю. Після входу користувач перенаправляється:

– FOUNDER – /app/founder/dashboard;

– INVESTOR – /app/investor/dashboard;

– ADMIN – окрема admin-панель (localhost:3001).

Layout кабінету (DashboardLayout) динамічно формує бокове меню залежно від ролі.

Ключові функції UI:

1) публічний каталог – перегляд стартапів, фільтрація за галуззю, рекомендації на основі підписок;

2) інвестування – форма на сторінці стартапу: сума, згода з умовами, перевірка KYC;

3) progress bar кампанії оновлюється після оплати;

4) Q&A – інвестор ставить питання засновнику на сторінці інвестування; засновник відповідає в розділі «Запитання»;

5) панель інвестора – KPI (інвестовано, поточна вартість, прибуток, ROI) і графіки (donut, bar, comparison);

6) панель засновника – зібрані кошти, список інвесторів (хто, скільки, коли, кампанія), детальний профіль компанії;

7) membership pass – відображення підписки засновника/інвестора;

8) локалізація – перемикач UK/EN у header і кабінеті.

Приклад реалізації HTTP-клієнта з автоматичним refresh токена

```
apiClient.interceptors.request.use((config) => {
```

```
  const token = getAccessToken();
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
apiClient.interceptors.response.use(
  (response) => response,
  async (error) => {
    if (error.response?.status === 401 && !error.config._retry) {
      error.config._retry = true;
      await refreshAccessToken();
      return apiClient(error.config);
    }
    return Promise.reject(error);
  },
);
```

Interceptor додає JWT до кожного запиту. При простроченні access token виконується оновлення через refresh token без повторного входу користувача.

Приклад реалізації сторінки інвестування

На сторінці StartupDetailPage завантажуються дані стартапу та активна кампанія; для інвестора відображаються форма інвестування та блок питань (CampaignQuestions). Після успішного POST /investments виконується повторне завантаження кампанії для оновлення progress bar.

```
useEffect(() => {
  if (!slug) return;
  void getStartupBySlug(slug)
    .then((s) => {
      setStartup(s);
      void trackStartupView(s.id, profile?.identity_id);
      return loadActiveCampaign(s.id);
    })
    .catch(() => {});
}, [slug, profile?.identity_id]);
async function loadActiveCampaign(startupId: string) {
  const c = await listCampaigns({ startup_id: startupId, status: 'active' });
  setCampaign(c.items[0] ?? null);
  return c.items[0] ?? null;
}
```

Викликаються GET /startups/:slug (через getStartupBySlug) та GET /campaigns?startup_id=...&status=active (через listCampaigns).

Після POST /investments кампанія перезавантажується з retry – raised_amount оновлюється асинхронно після оплати.

Код наведено у Додаток Б

Тіло запиту: campaign_id, investor_profile_id, amount, terms_accepted. JWT додається interceptor'ом у apiClient.

4.1.3 Реалізація взаємодії між мікросервісами

Взаємодія між компонентами системи здійснюється двома основними способами: синхронною взаємодією клієнта з API через Gateway та синхронною міжсервісною взаємодією через gRPC.

Клієнт, Swagger UI або frontend звертаються до системи через адресу:

http://localhost:8080/api/v1/...

Gateway-service виконує такі функції:

- перевіряє JWT-токен для захищених маршрутів;
- додає заголовки X-Identity-Id, X-User-Email, X-User-Role;
- проксіює запит до відповідного мікросервісу.

Таблиця 4.4 – Взаємодія мікросервісів системи

Виклик	Від	До	Призначення
CreateProfile	auth-service	users-service	Створення профілю після реєстрації
UpsertStartup	users-service	startups-service	Синхронізація каталогу після оновлення профілю founder
ValidateToken	gateway-service	auth-service	Валідація JWT на gateway

Маршрутизація Gateway та JWT middleware:

Код наведено у Додаток В

4.2 Опис основних класів та компонентів системи

Основні компоненти системи згруповані за мікросервісами. Спільна бібліотека golib використовується для повторного використання загальних механізмів, зокрема контрактів HTTP-помилок, Swagger UI, RBAC та міграцій.

Таблиця 4.5 – Основні компоненти бібліотеки golib

Компонент	Призначення
golib/http/contracts	Стандартні HTTP-коди помилок
golib/http/swagger	Монтування Swagger UI
golib/rbac	Ролі та дозволи
golib/migrations	Підключення до PostgreSQL, runner SQL-міграцій

Клас rbac.Role є типом-аліасом string. Константи RoleInvestor, RoleFounder, RoleAdmin визначають ролі користувачів. Метод IsValid() перевіряє допустимість значення ролі. Мапа rolePermissions визначає дозволи для кожної ролі.

Таблиця 4.6 – Основні компоненти auth-service

Компонент	Призначення	Основні поля / методи
app.Identity	Обліковий запис	ID, Email, PasswordHash, Role, Status, EmailVerified
app.Tokens	Пара JWT-токенів	AccessToken, RefreshToken, ExpiresAt
usecase.UseCase	Бізнес-логіка	Register, Login, VerifyEmail, RefreshTokens
repo.Repo	Інтерфейс БД	SaveIdentity, GetIdentity, SaveRefreshToken
clients/users.Client	gRPC-клієнт	CreateProfile
handlers/http.Handler	REST endpoints	Register, Login, GetIdentity

Основні компоненти users-service забезпечують управління профілями користувачів системи. Сервіс підтримує роботу з профілями інвесторів та

засновників стартапів, а також реалізує механізм RBAC для розмежування прав доступу.

Таблиця 4.7 – Основні компоненти users-service

Компонент	Призначення
app.Profile	Базовий профіль користувача
app.InvestorProfile	Розширений профіль інвестора
app.StartupProfile	Профіль засновника / стартапу
usecase.UseCase	CreateProfile, UpdateProfile, UpdateStartupProfile
clients/startups.Client	SyncStartupProfile через gRPC

Основні компоненти startups-service відповідають за роботу каталогу стартапів, пошук, фільтрацію та отримання інформації про стартапи. Сервіс призначений для публікації профілів стартапів у каталозі після синхронізації з users-service.

Таблиця 4.8 – Основні компоненти startups-service

Компонент	Призначення
app.Startup	Сутність каталогу
repo.Repo	ListStartups, GetStartupBySlug, UpsertStartup
usecase.UseCase	UpsertStartup за founder_profile_id

Основні компоненти campaigns-service реалізують логіку управління інвестиційними компаніями. Життєвий цикл компанії включає такі статуси: draft, review, active, funded, failed, closed. Після відправлення компанії на модерацію адміністратор може підтвердити або відхилити її.

Сервіс підтримує операції створення, оновлення та отримання інформації про стартапи, а також враховує взаємодію з users-service для синхронізації даних профілю засновника.

Таблиця 4.9 – Основні компоненти campaigns-service

Компонент	Призначення
app.Campaign	Інвестиційна компанія
usecase.UseCase	CreateCampaign, SubmitCampaign, ApproveCampaign, RejectCampaign
Статуси	draft – review – active / draft після reject

Основні компоненти investments-service забезпечують роботу із заявками на інвестування, формування портфеля інвестора та перегляд історії інвестицій.

Таблиця 4.10 – Основні компоненти investments-service

Компонент	Призначення
app.Investment	Заявка на інвестицію
usecase.UseCase	CreateInvestment, ListInvestments, GetPortfolio

Основні компоненти gateway-service забезпечують маршрутизацію HTTP-запитів між клієнтом та мікросервісами системи. Сервіс виконує роль єдиної точки входу до платформи та реалізує механізми JWT-автентифікації, перевірки доступу та передачі інформації про користувача до інших сервісів.

Таблиця 4.11 – Основні компоненти gateway-service

Компонент	Призначення
handlers/http.Handler	Маршрутизація, auth middleware
auth.Client	gRPC ValidateToken
proxy.ReverseProxy	HTTP reverse proxy до мікросервісів

Константи RBAC:

```
const (
    RoleInvestor Role = "INVESTOR"
    RoleFounder  Role = "FOUNDER"
    RoleAdmin    Role = "ADMIN"
)
const (
    PermCampaignCreate   Permission = "campaign:create"
    PermCampaignModerate Permission = "campaign:moderate"
```

```
    PermInvestmentCreate Permission = "investment:create"
    PermProfileManage    Permission = "profile:manage"
)
```

4.3 Тестування програмного забезпечення

Для перевірки працездатності розробленого програмного забезпечення було проведено інтеграційне, ручне та API-тестування.

Методи тестування

Для перевірки програмного забезпечення застосовано такі методи тестування:

- інтеграційне E2E-тестування API за допомогою автоматизованого сценарію scripts/e2e-test.ps1;
- ручне тестування окремих endpoints через Swagger UI.

Таблиця 4.12 – Охоплення E2E-сценарій повного бізнес-циклу

№	Фаза	Кроки	Очікуваний результат
1	Auth	Register founder, investor, admin; verify email; login; identity; refresh; audit	HTTP 201/200, JWT-токени
2	Profiles	GET/PUT profiles/me, startup, investor; public profile; admin list	HTTP 200, profile_id
3	Startups	List by founder_profile_id, get by slug	Стартап у каталозі після sync
4	Campaigns	Create, update, submit, approve, list active	status = active
5	Investments	Create, list, get, portfolio, dashboard	HTTP 201, investment_id
6	Documents	Upload, list, get, download	HTTP 201
7	Notifications	Send, list, mark read	HTTP 201/200
8	Communications	Questions, answers, comments, updates	HTTP 201
9	Analytics	Popularity, conversion, audit logs, logout	HTTP 200/204

Тестові облікові записи створюються динамічно з унікальним timestamp, наприклад founder-{ts}@test.com.

Приклад тестового запиту реєстрації:

POST /api/v1/auth/register

Content-Type: application/json

{

"email": "investor@test.com",

"password": "TestPass123!",

"first_name": "Ivan",

"last_name": "Petrov",

"role": "INVESTOR"

}

Очікувана відповідь:

HTTP 201

{

"id": "...",

"status": "PENDING_EMAIL"

}

Swagger

На рисунку зображено інтерфейс Swagger UI, який використовується для тестування REST API та перегляду документації програмного забезпечення.

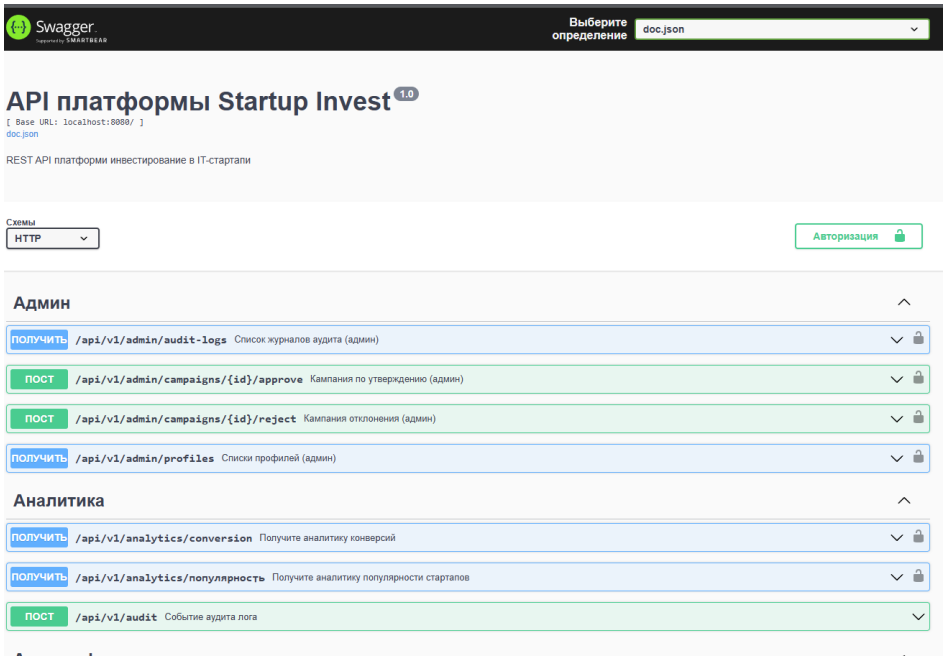

Рисунок 4.1 – Інтерфейс Swagger UI для тестування REST API

Використання Swagger UI зручний та надійний інструмент для тестування REST API, перевірку коректності роботи сервісів та спрощує процес розробки і супроводження програмного забезпечення. Також, даний інструмент використовується для контролю над взаємодією між клієнтською та серверною частинами системи.

Аналіз результатів тестування

Під час тестування оцінювалася працездатність основних функцій системи, правильність обробки запитів та взаємодія між окремими компонентами мікросервісної архітектури. Результати останнього прогону автоматизованих E2E-тестів наведено в таблиці.

Таблиця 4.13 – Результати останнього прогону автоматизованих E2E-тестів

Показник	Значення
Всього перевірок	45
Успішних	45
Невдалих	0
Час виконання	Приблизно 3-4 с

Під час тестування було перевірено:

- реєстрацію, верифікацію email, login, refresh token, logout;
- CRUD профілів founder, investor, admin;
- синхронізацію профілю founder з каталогом стартапів через gRPC UpsertStartup;
- життєвий цикл компанії draft – review – active;
- створення інвестиційної заявки;
- портфель інвестора;
- dashboard;
- завантаження та отримання документів;

- in-app сповіщення;
- комунікації, зокрема Q&A, коментарі та оновлення.

Таблиця 4.14 – У процесі розробки були виявлені та усунені такі дефекти

Проблема	Причина	Рішення
INTERNAL_ERROR при реєстрації	sqlx NamedExec без db-тегів	Додано row-структури з db-тегами в repo
404 на GET /startups, /campaigns	Gateway маршрути тільки з /*	Додано mount для exact paths
Refresh token – 401	Endpoint був за auth middleware	Винесено /auth/refresh з middleware
Профіль не створювався	auth не викликав users-service	Додано gRPC CreateProfile при Register
Каталог порожній	Не було sync profile – catalog	Додано gRPC UpsertStartup при UpdateStartupProfile

Результати тестування наведено у додатку А

4.4 Керівництво користувача

Загальні відомості

Програмний застосунок TechIT призначений для взаємодії інвесторів та засновників IT-стартапів: перегляд проєктів, інвестування в кампанії збору коштів, ведення профілю компанії та модерації заявок.

Адреси доступу:

- основний застосунок – http://localhost:3000;
- панель адміністратора – http://localhost:3001;
- перегляд листів підтвердження email (MailHog) – http://localhost:8025.

Ролі користувачів:

- інвестор – перегляд каталогу, інвестування, KYC, портфель;
- засновник – профіль стартапу, кампанії, робота з інвесторами та запитаннями;

– адміністратор – модерація кампаній, KYC, користувачів (окремий застосунок).

Мова інтерфейсу перемикається кнопками UK / EN у верхній панелі.

Початок роботи

Реєстрація облікового запису:

1) відкрийте головну сторінку застосунку;
2) натисніть «увійти» або перейдіть за посиланням «зареєструватися»;
3) заповніть форму: ім’я, прізвище, email, пароль;
4) оберіть роль: інвестор або засновник;
5) натисніть «створити обліковий запис».

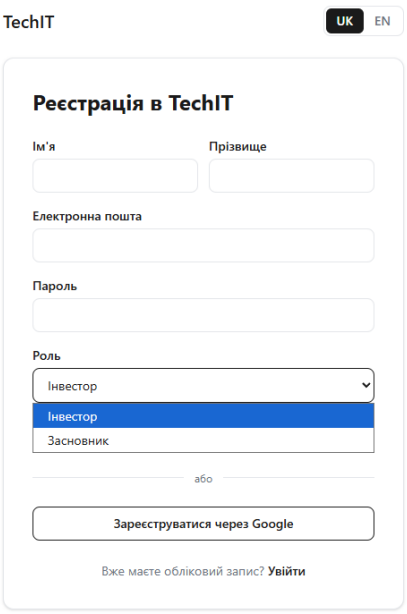

Рисунок 4.2 – Форма реєстрації користувача з вибором ролі

Після успішної реєстрації на email надсилається лист для підтвердження. У середовищі розробки його можна переглянути в MailHog.

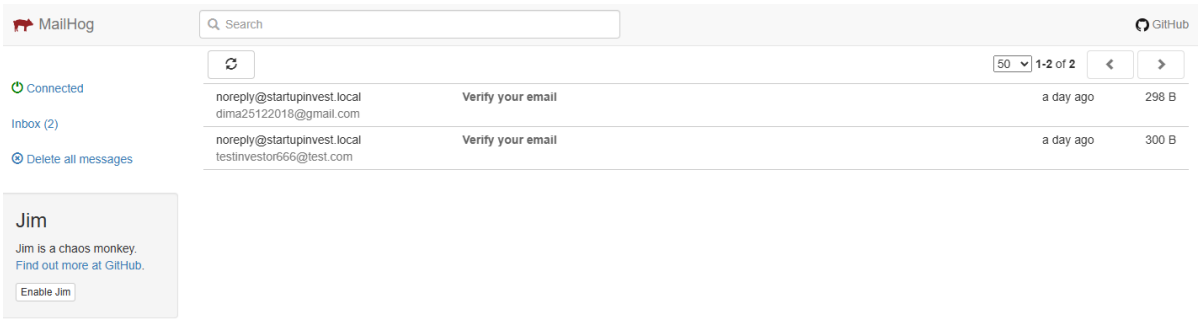

Рисунок 4.3 – Лист підтвердження email у MailHog

Вхід у систему:

1) перейдіть на сторінку «увійти»;
2) введіть email і пароль;
3) натисніть «увійти».

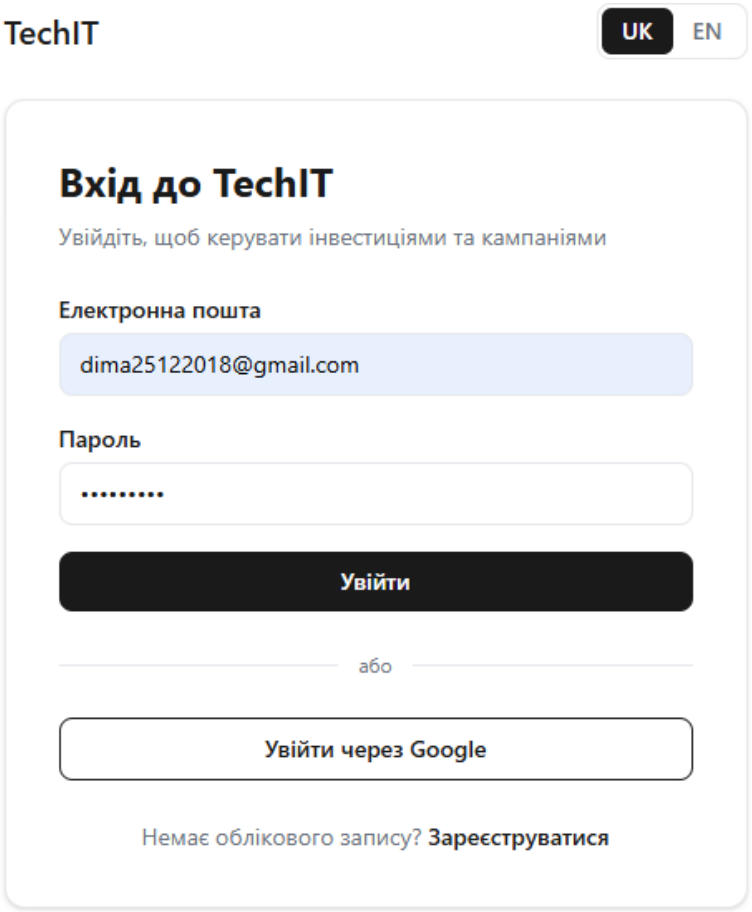

Рисунок 4.4 – Сторінка входу в систему

Після входу користувач автоматично перенаправляється:

– інвестор – на «панель» інвестора;

– засновник – на «панель» засновника.

Робота інвестора

Перегляд головної сторінки та каталогу проєктів:

1) на головній сторінці відображаються рекомендовані стартапи та категорії;

2) для повного списку натисніть «проєкти» у меню або перейдіть на /projects;

3) за потреби оберіть фільтр за галуззю або скористайтеся пошуком.

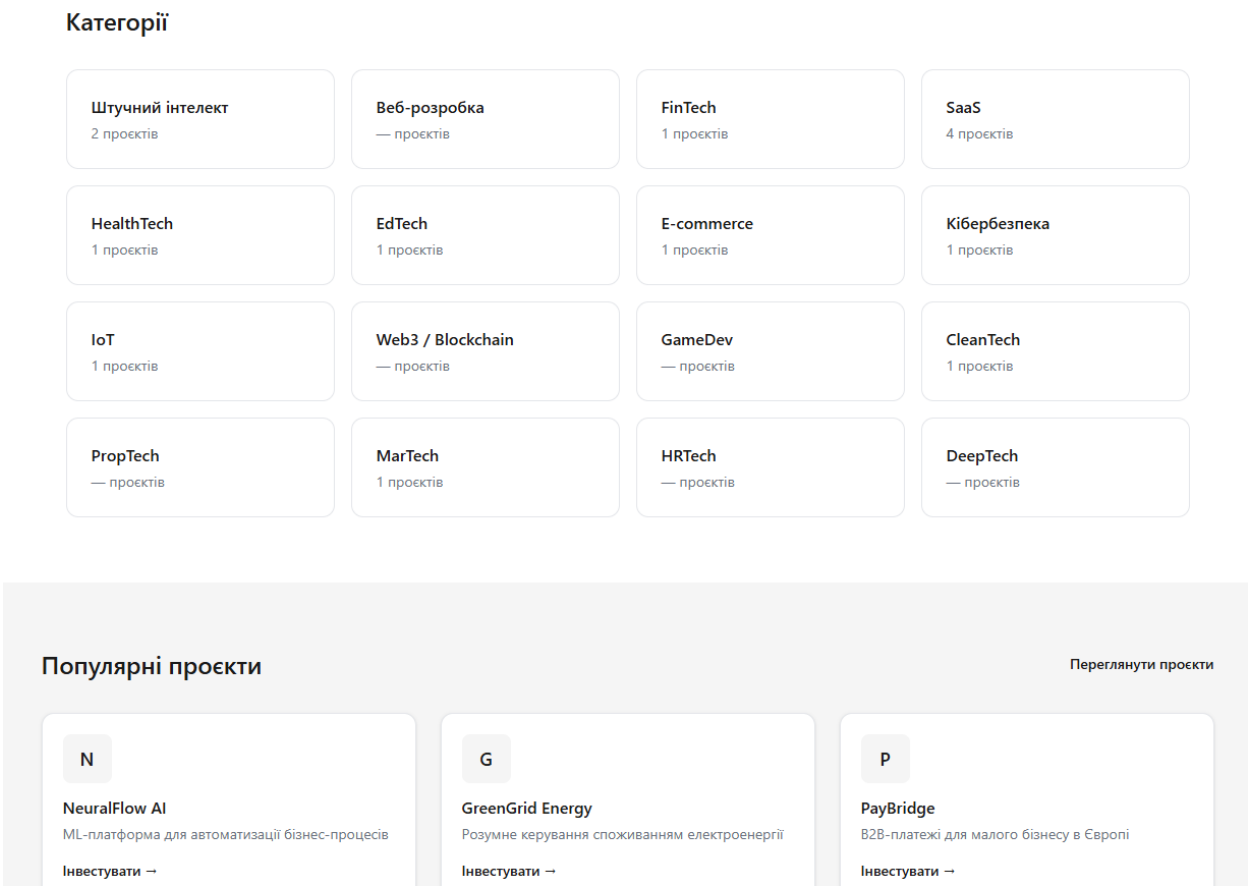

Рисунок 4.5 – Головна сторінка з категоріями та рекомендаціями

Проходження KYC (обов’язково перед інвестуванням):

1) у боковому меню оберіть «KYC»;

2) заповніть форму: тип документа, номер, країна (за потреби – посилання на документ);

3) надішліть заявку на перевірку;

4) дочекайтеся схвалення адміністратором. До схвалення кнопка «Інвестувати» поверне повідомлення про необхідність KYC.

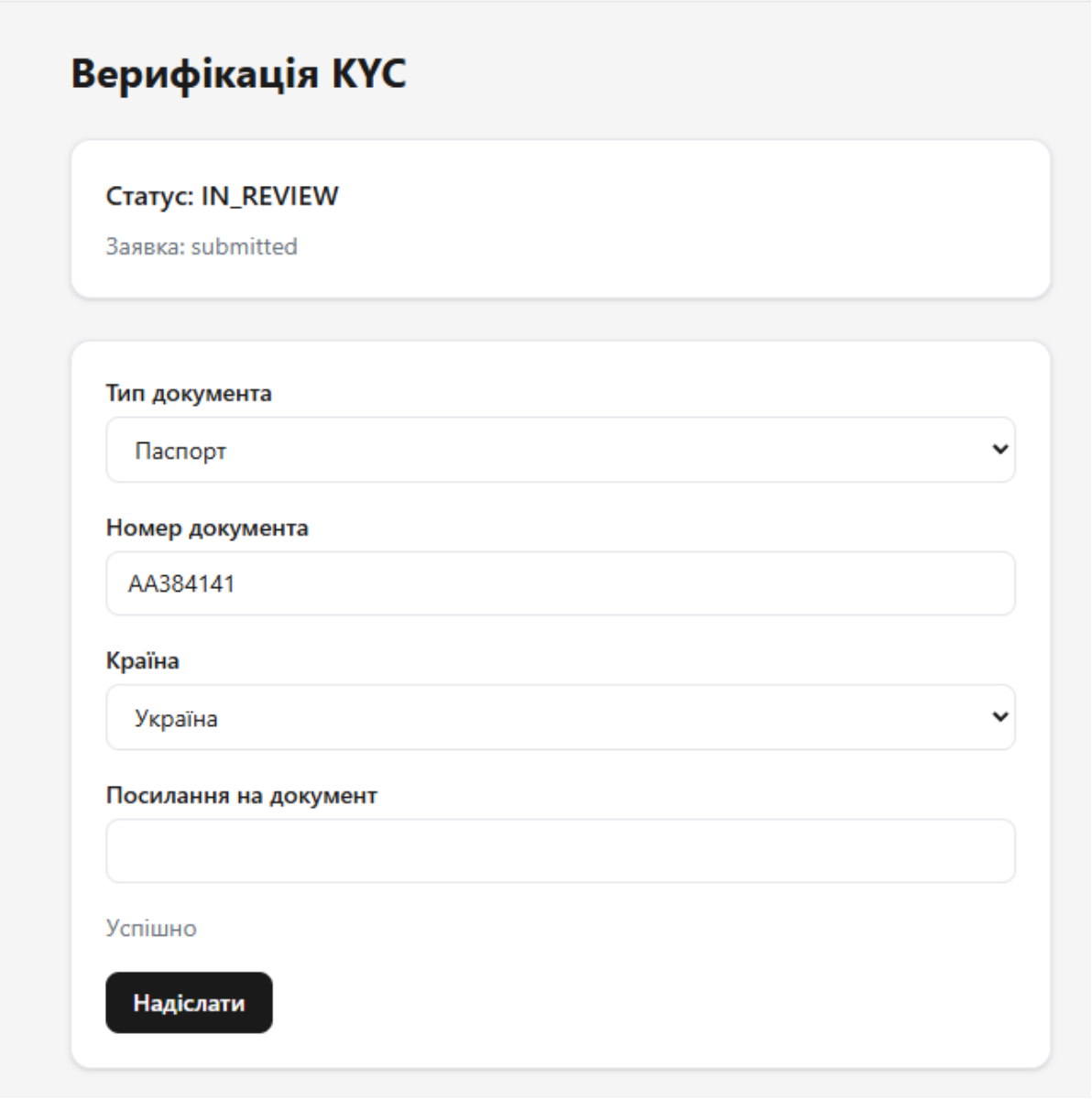

Рисунок 4.6 – Форма подання KYC-заявки

Перегляд сторінки стартапу:

1) відкрийте сторінку стартапу з активною кампанією;

2) у блоці кампанії введіть суму;

3) поставте галочку згоди з умовами;

4) натисніть «Інвестувати» ;

5) після успішної оплати progress bar оновлюється.

Після підтвердження операції система створює запис про інвестицію та оновлює інформацію про поточний стан кампанії. Дані про виконану транзакцію зберігаються в базі даних і відображаються в особистому кабінеті інвестора, що надає користувачам функціонал відстежувати історію вкладень та контролювати стан інвестиційного портфеля.

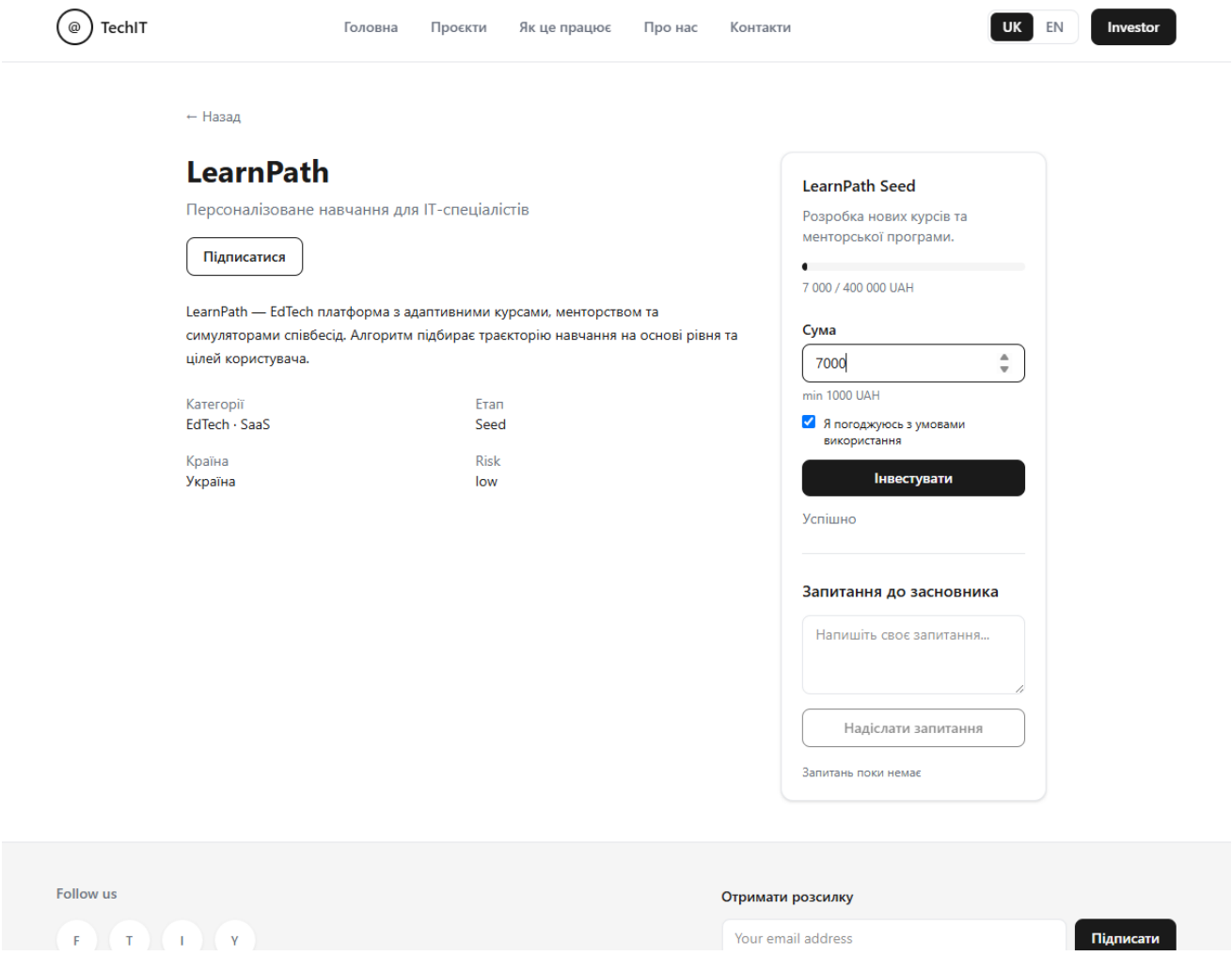

Рисунок 4.7 – Форма інвестування на сторінці стартапу

Панель інвестора та портфель:

1) у меню оберіть «панель»;

2) переглядайте показники: інвестовано, поточна вартість, прибуток, ROI;

3) нижче – графіки розподілу інвестицій і порівняння вкладеного vs вартості;

4) розділ «інвестиції» – список усіх операцій;

5) «обране» – стартапи, додані до списку спостереження;

6) «сповіщення» – системні повідомлення.

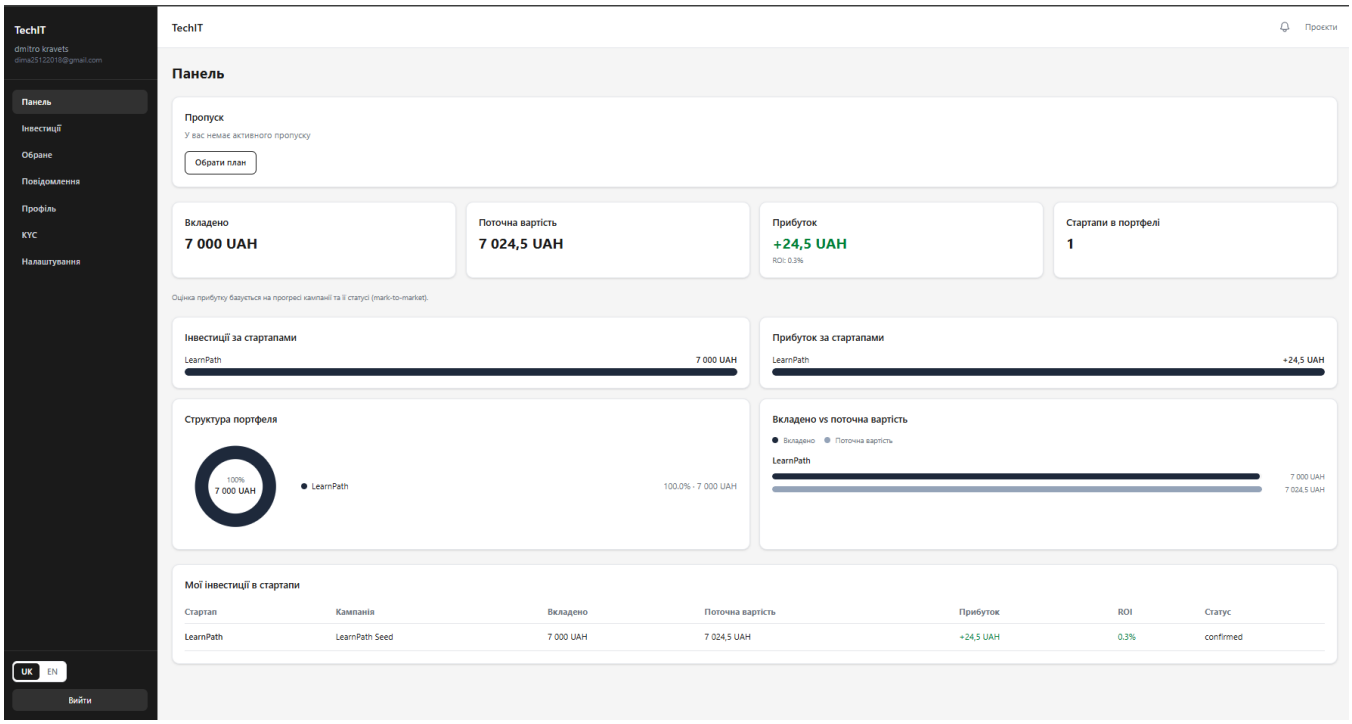

Рисунок 4.8 – Панель інвестора з KPI та графіками

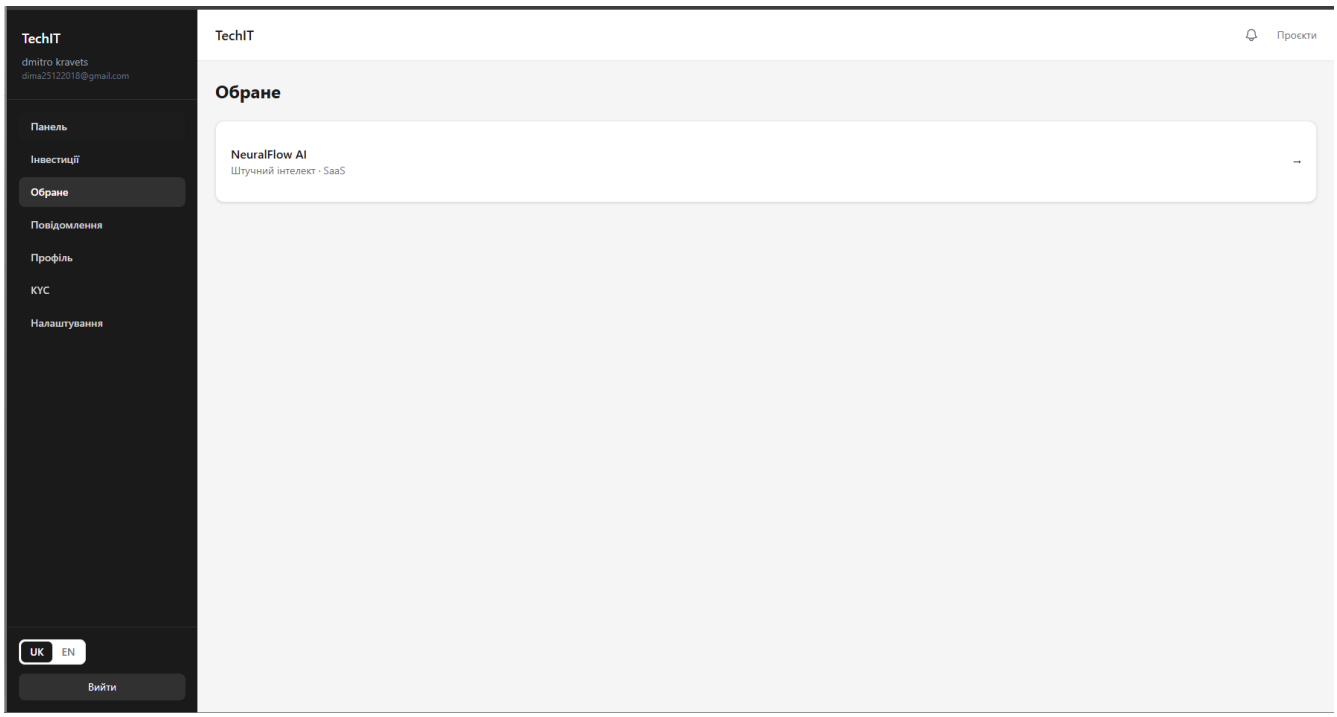

Рисунок 4.9 – Сторінка обраних стартапів

Профіль інвестора:

1) перейдіть у «профіль»;

2) заповніть або змініть: галузі інтересу, етапи, толерантність до ризику тощо;

3) збережіть зміни.

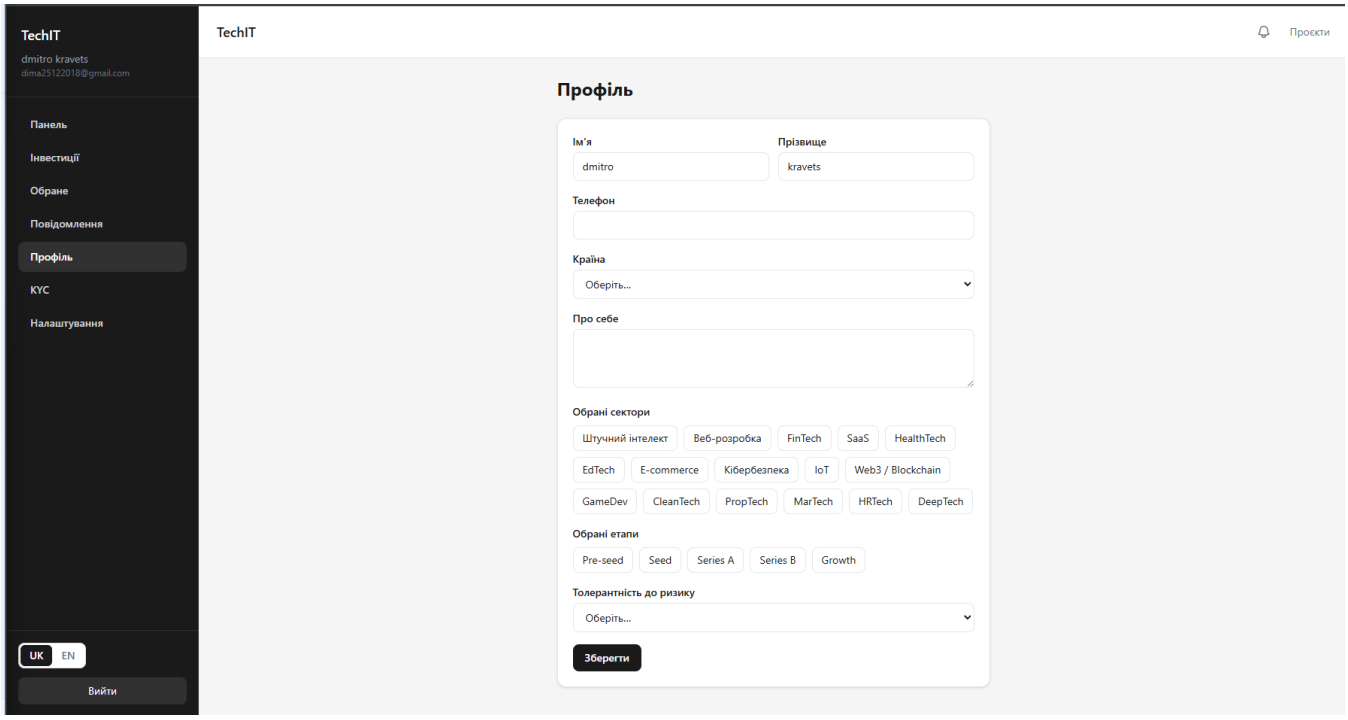

Рисунок 4.10 – Редагування профілю інвестора

Робота засновника

Перший вхід і профіль стартапу:

1) після входу відкривається «панель» засновника;

2) у меню оберіть «стартап»;

3) заповніть: назва компанії, опис, галузі, етап, країна, сайт, рік заснування, розмір команди;

4) натисніть «зберегти».

Після збереження змін оновлена інформація про користувача відображається в особистому профілі та використовується для формування персоналізованих рекомендацій. Налаштування профілю має функціонал, який враховувати

інвестиційні інтереси користувача та забезпечують більш ефективний пошук перспективних стартапів.

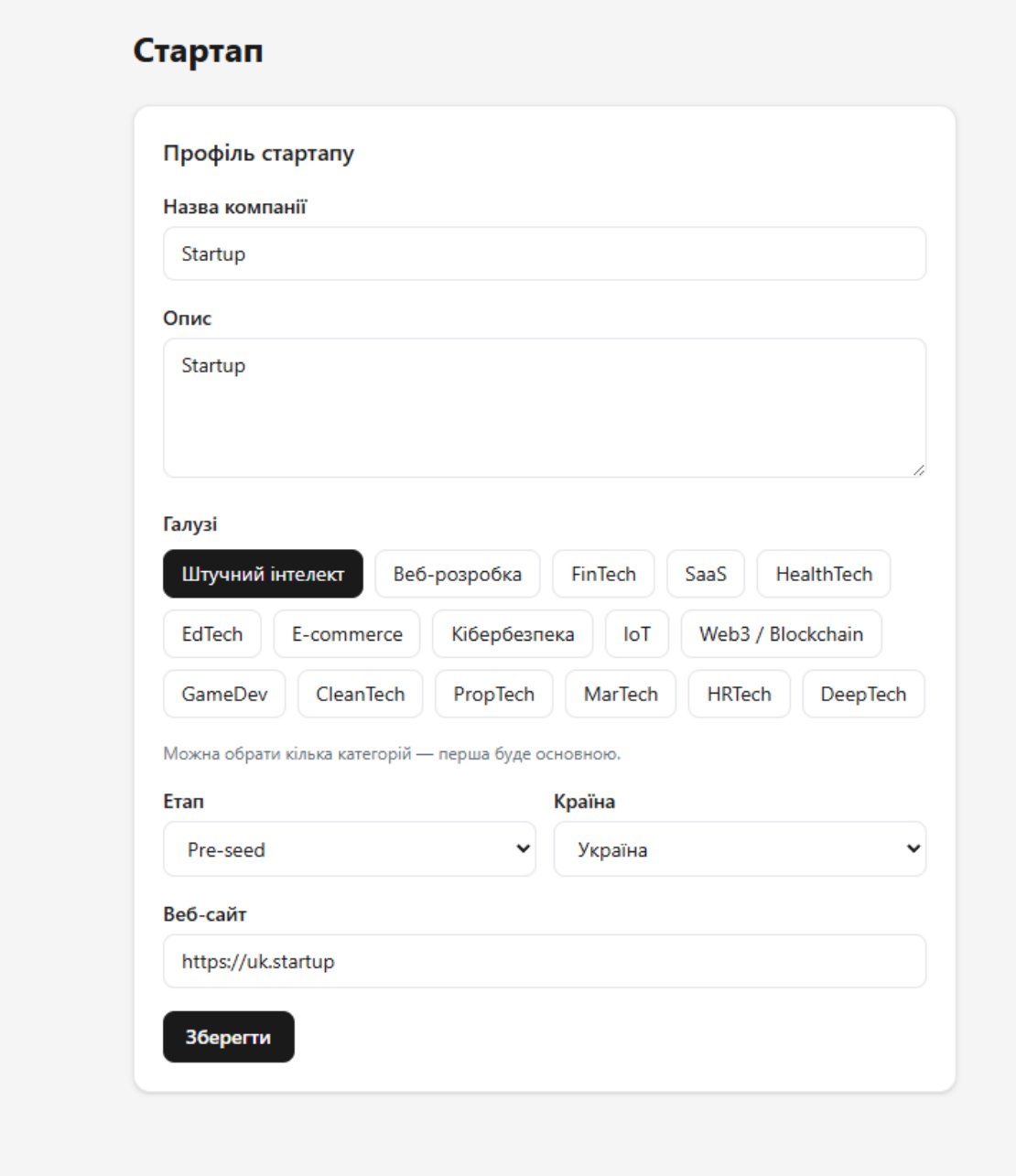

Рисунок 4.11 – Форма профілю стартапу з вибором галузей

На «панелі» засновника відображаються:

– зібрано – загальна сума по кампаніях;
– інвестори – кількість (клік відкриває детальний список);
– компанії – перехід до розділу кампаній.

Представлена форма дозволяє засновнику вказати основну інформацію про компанію, обрати галузі діяльності, стадію розвитку та країну розташування стартапу. Заповнені дані використовуються для формування публічного профілю компанії та подальшого відображення стартапу в каталозі проєктів, доступному для інвесторів.

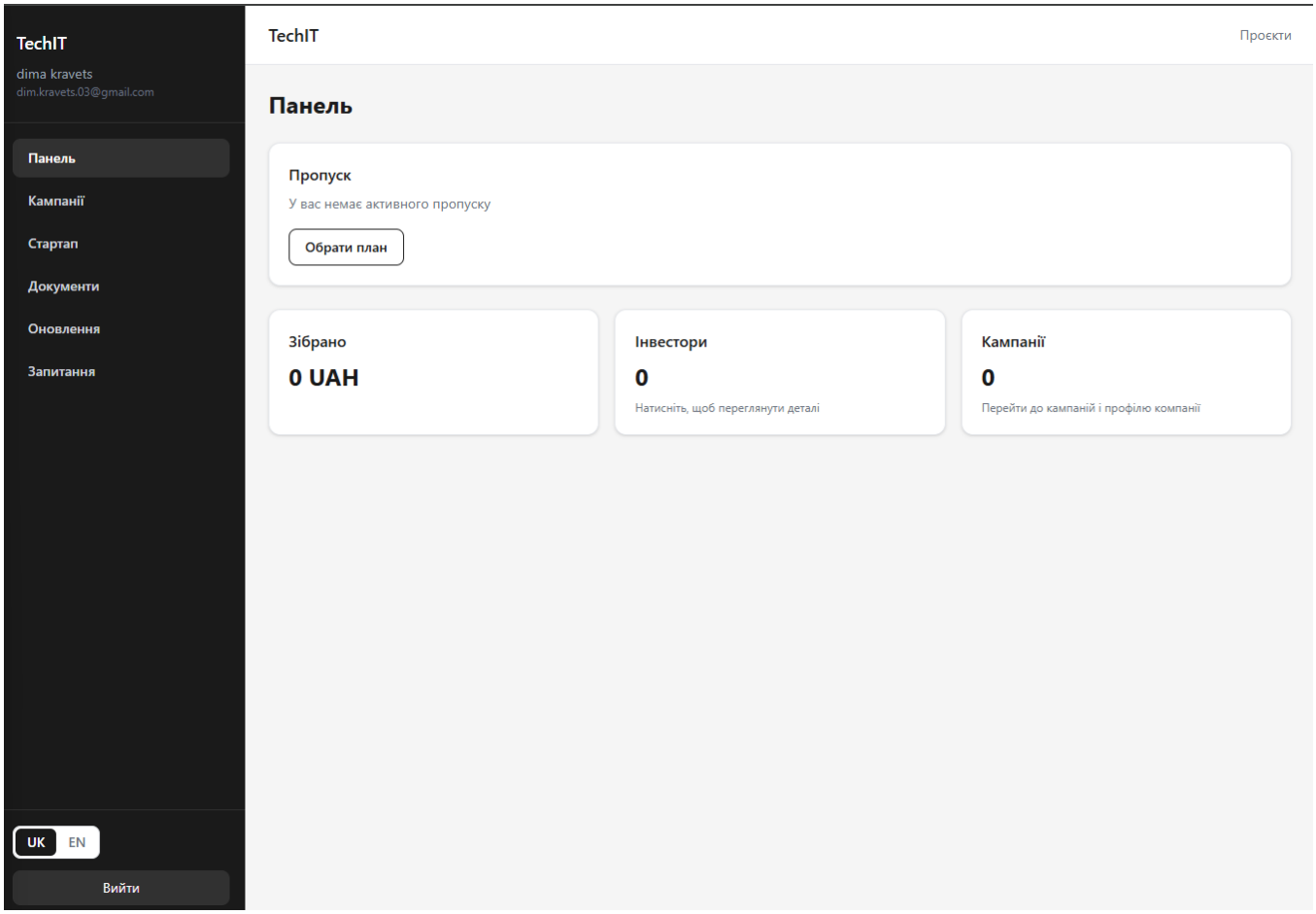

Рисунок 4.12 – Панель засновника зі зведеною статистикою

Кампанії збору коштів:

1) перейдіть у «кампанії»;
2) перегляньте картку компанії з повною інформацією про стартап;
3) натисніть «створити кампанію»;
4) заповніть: назва, опис, ціль (UAH), мінімальна інвестиція, дедлайн.
5) збережіть кампанію;

6) для нової кампанії натисніть «на модерацію», щоб адміністратор її переглянув і схвалив;

7) після схвалення кампанія стає active; progress bar показує прогрес збору.

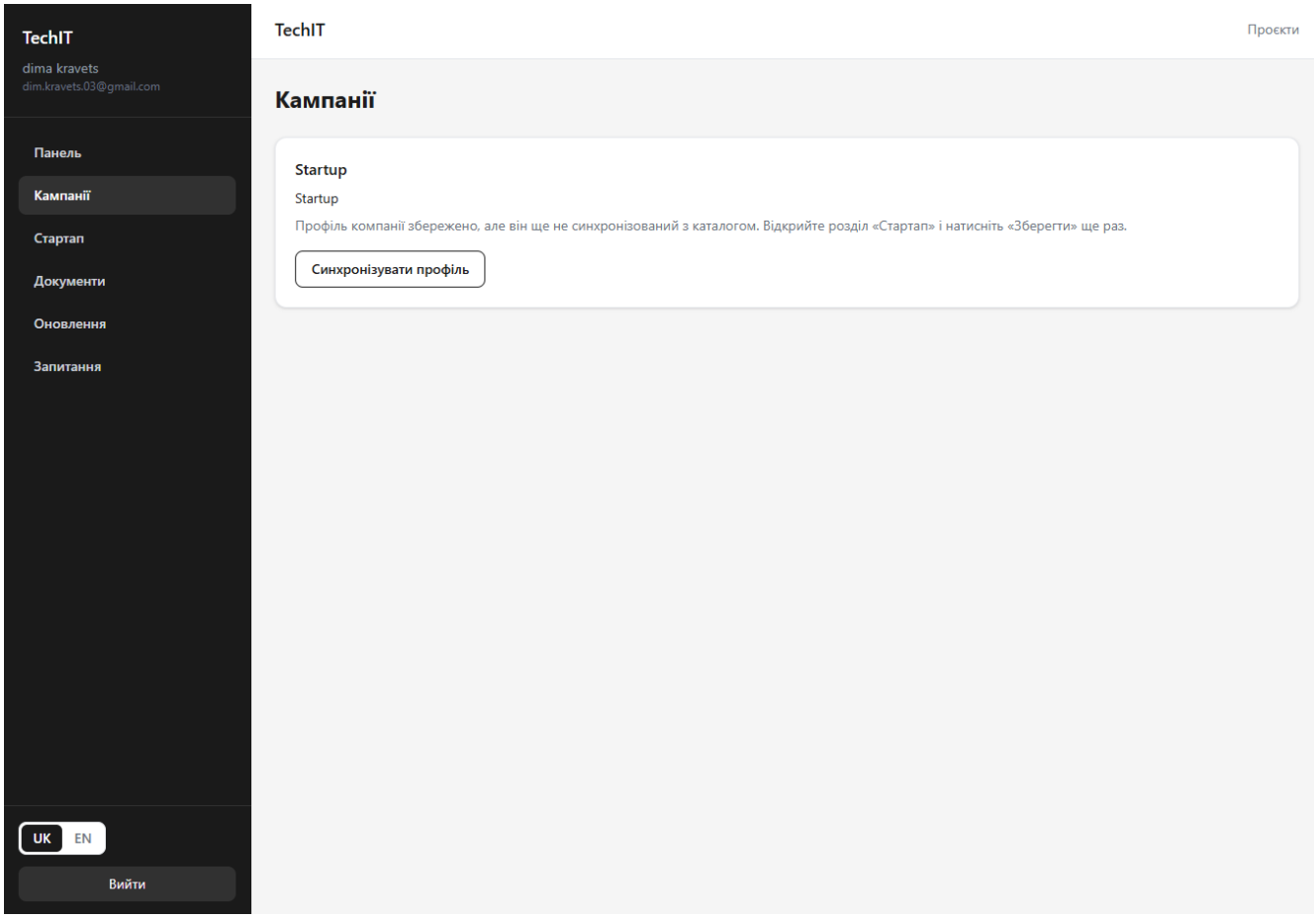

Рисунок 4.13 – Сторінка кампаній з профілем компанії

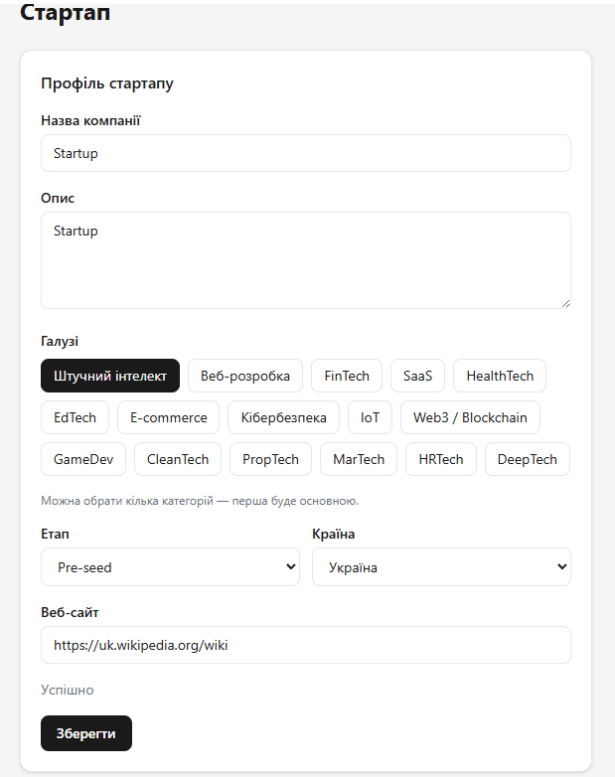

Рисунок 4.14 – Форма створення кампанії

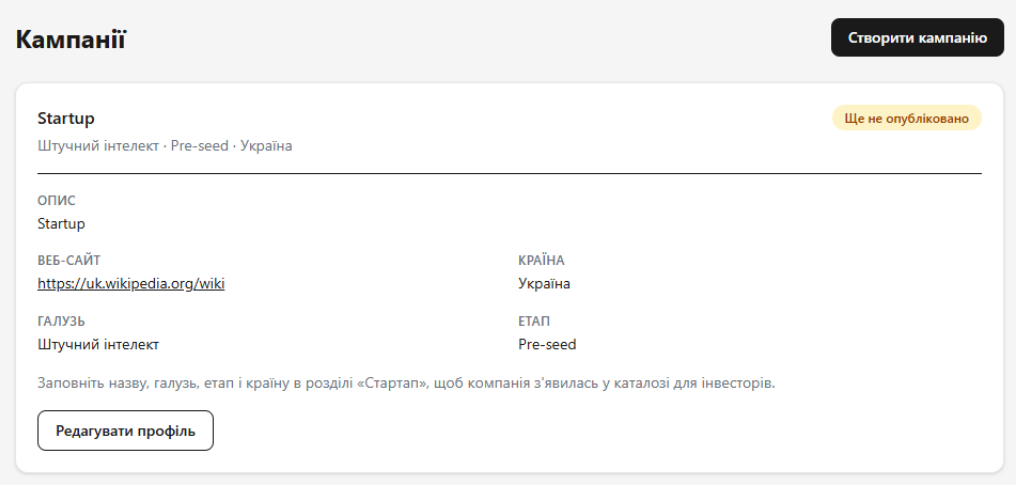

Рисунок 4.15 – Кампанія в статусі draft

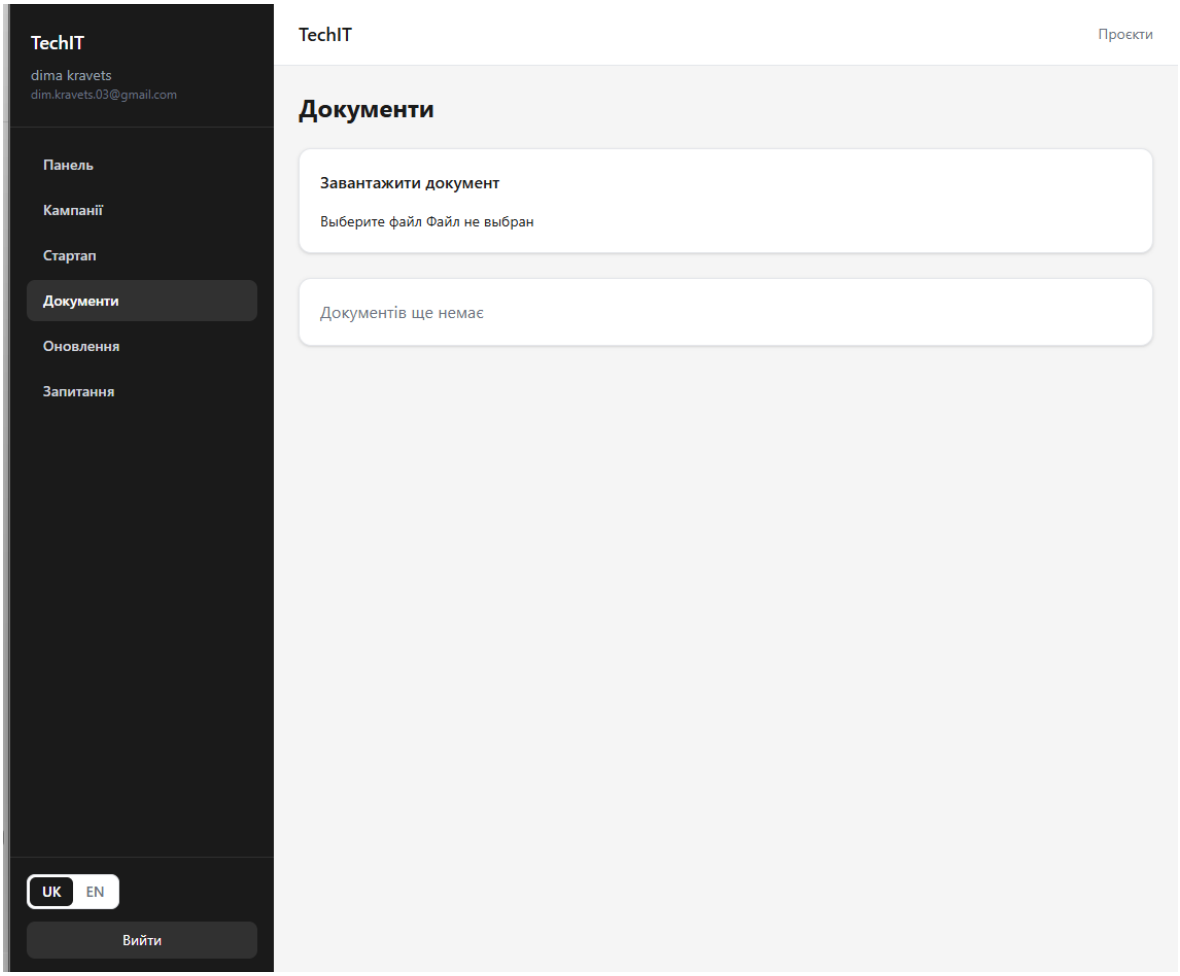

Рисунок 4.16 – Завантаження документів стартапу

Вихід із системи

У боковому меню кабінету натисніть «вийти».

Сесія завершується, токени видаляються з браузера.

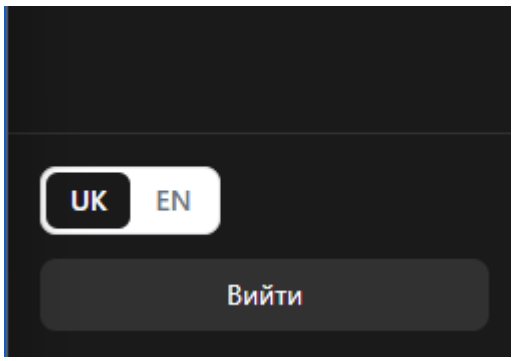

Рисунок 4.17 – Кнопка виходу з облікового запису

Висновки до розділу 4

У четвертому розділі було реалізовано серверну та клієнтську частини вебзастосунку інвестиційної платформи. Реалізовано REST API, JWT-автентифікацію, RBAC, систему управління кампаніями та взаємодію між мікросервісами. Було створено основні сервіси системи, зокрема сервіси авторизації, управління профілями користувачів, каталогу стартапів, інвестиційних кампаній, інвестицій, документів, сповіщень та аналітики.

У процесі реалізації програмного забезпечення використано мікросервісну архітектуру, Docker-контейнеризацію, PostgreSQL та gRPC-взаємодію між сервісами. Також було реалізовано механізми маршрутизації запитів через gateway-service, систему ролей користувачів та механізми захисту API на основі JWT-токенів.

Було виконано опис основних класів та компонентів системи, а також наведено приклади реалізації ключових методів програмного забезпечення. Для тестування REST API використовувались Swagger UI, Postman та автоматизовані E2E-сценарії.

Проведене тестування програмного забезпечення підтвердило працездатність основних компонентів системи, коректність взаємодії між мікросервісами та стабільність роботи мікросервісної архітектури.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено проєкт вебзастосунку інвестиційної платформи для інвестування в IT-стартапи на основі мікросервісної архітектури. Актуальність обраної теми підтверджується активним розвитком цифрової економіки, збільшенням кількості IT-стартапів та необхідністю створення сучасних інформаційних систем для організації інвестиційної діяльності.

У процесі виконання кваліфікаційної роботи було проаналізовано сучасні підходи до створення онлайн-платформ для інвестування та досліджено предметну область інвестування в IT-стартапи. Також було проаналізовано процеси групової взаємодії та комунікації між користувачами системи.

У межах роботи було визначено функціональні вимоги до вебзастосунку, зокрема вимоги до реєстрації користувачів, управління профілями, створення інвестиційних кампаній, інвестування, модерації та системи сповіщень. Визначено нефункціональні вимоги до програмного забезпечення, пов’язані з продуктивністю, безпекою, масштабованістю та надійністю системи.

У кваліфікаційній роботі спроєктовано загальну структуру мікросервісної архітектури та визначено склад і відповідальність окремих мікросервісів платформи. Для реалізації системи було обрано сучасні технології та засоби розробки, зокрема Golang, React, PostgreSQL, Docker, Apache Kafka, REST API та gRPC.

Також у межах роботи було розроблено UML-діаграми, мокапи інтерфейсу користувача та описано основні компоненти програмного забезпечення. Проведене тестування підтвердило працездатність основних функцій системи та коректність взаємодії між мікросервісами.

Отримані результати можуть бути використані як основа для подальшої розробки та впровадження сучасної платформи онлайн-інвестування в IT-стартапи з підтримкою масштабованої мікросервісної архітектури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Almeida M. G. de, Canedo E. D. Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. Applied Sciences (Switzerland). 2022. Vol. 12, № 6. DOI: https://doi.org/10.3390/app12063023

2. Andrzej Jarzyna, Samir Amzani RESTful API Design Patterns and Best Practices: Master REST API design with real-world patterns, lifecycle management, and OpenAPI practices. 2025. 406 P. URL: https://www.mendeley.com/catalogue/73023dc0-be49-3092-bad9-3948a8b88549/ (Accessed: 24.05.2026).

3. Aydemir F., Başçiftçi F. Performance and Availability Analysis of API Design Techniques for API Gateways. Arabian Journal for Science and Engineering. 2025. Vol. 50, № 15. P. 11485–11498. DOI: https://doi.org/10.1007/s13369-024-09474-9

4. Balsam S., Mishra D. Web application testing–Challenges and opportunities. Journal of Systems and Software. 2025. Vol. 219. DOI: https://doi.org/ 10.1016/j.jss.2024.112186

5. Chaves A. J., Martín C., Díaz M. The orchestration of Machine Learning frameworks with data streams and GPU acceleration in Kafka-ML: A deep-learning performance comparative. Expert Systems. 2024. Vol. 41, № 2. DOI: https://doi.org/ 10.1111/exsy.13287

6. Chaves A. J., Martín C., Díaz M., Thalor M., Allur S. R., Bhende V. S., Shao X. How enterprise architecture improves the quality of IT investment decisions. Journal of Systems and Software. 2019. Vol. 152. P. 134–150. URL: https://www.sciencedirect.com/science/article/abs/pii/S0164121219300433 (Accessed: 24.05.2026).

8. Crowdcube – Startup Investing Platform. URL: https://www.crowdcube.com/ (Accessed: 24.05.2026).

9. Gowda P., Gowda A. N. Best Practices in REST API Design for Enhanced Scalability and Security. Journal of Artificial Intelligence, Machine Learning and Data Science. 2024. Vol. 2, № 1. P. 827–830. DOI: https://doi.org/ 10.51219/jaimld/priyanka-

gowda/202

10. Ileana M., Petrov P., Milev V. Optimizing Customer Experience by Exploiting Real-Time Data Generated by IoT and Leveraging Distributed Web Systems in CRM Systems. Internet of Things. 2025. Vol. 6, № 2. DOI: https://doi.org/ 10.3390/iot6020024

11. Kazanavičius J., Mažeika D. EVALUATION OF MICROSERVICE COMMUNICATION WHILE DECOMPOSING MONOLITHS. Computing and Informatics. 2023. Vol. 42, № 1. P. 1–36. DOI: https://doi.org/ 10.31577/cai_2023_1_1

12. Martín C., Langendoerfer P., Zarrin P. S., Díaz M., Rubio B. Kafka-ML: Connecting the data stream with ML/AI frameworks. Future Generation Computer Systems. 2022. Vol. 126. P. 15–33. DOI: https://doi.org/ 10.1016/j.future.2021.07.037

13. Mohamed Ali A. Exploring Database design patterns of Microservices. Journal of Artificial Intelligence, Machine Learning and Data Science. 2024. Vol. 2, № 1. P. 1732–1735. DOI: https://doi.org/ 10.51219/jaimld/azra-jabeen-mohamed-ali/376

14. Niswar M., Safruddin R. A., Bustamin A., Aswad I. Performance evaluation of microservices communication with REST, GraphQL, and gRPC. International Journal of Electronics and Telecommunications. 2024. Vol. 70, № 2. P. 429–436. DOI: https://doi.org/10.24425/ijet.2024.149562

15. Owen A. Microservices Architecture and API Management: A Comprehensive Study of Integration, Scalability, and Best Practices. Researchgate. 2025. № January. P. 3. URL: https://surl.li/ddokaq (Accessed: 24.05.2026).

16. Oyekunle Claudius Oyeniran, Adebunmi Okechukwu Adewusi, Adams Gbolahan Adeleke, Lucy Anthony Akwawa, Chidimma Francisca Azubuko. Microservices architecture in cloud-native applications: Design patterns and scalability. Computer Science & IT Research Journal. 2024. Vol. 5, № 9. P. 2107–2124. URL: https://www.ijisae.org (Accessed: 24.05.2026).

17. Pourmajidi W., Zhang L., Steinbacher J., Erwin T., Miranskyy A. A Reference Architecture for Governance of Cloud Native Applications. IEEE Transactions on Cloud Computing. 2025. Vol. 13, № 3. P. 935–952. DOI: https://doi.org/10.1109/TCC.2025.3578557

18. Republic – Invest in Startups, Crypto and More. URL: https://republic.com/ (Accessed: 24.05.2026).

19. StartEngine – Equity Crowdfunding Platform. URL: https://www.startengine.com/ (Accessed: 24.05.2026).

20. Thalor M., Allur S. R., Bhende V. S., Chavan A. Analysis of Monolithic and Microservices System Architectures for an E-Commerce Web Application. International Journal of Intelligent Systems and Applications in Engineering. 2024. Vol. 12, № 4. P. 2400–2406. URL: https://www.ijisae.org (Accessed: 24.05.2026).

21. Weerawarna R., Miah S. J., Shao X. Emerging advances of blockchain technology in finance: a content analysis. Personal and Ubiquitous Computing. 2023. Vol. 27, № 4. P. 1495–1508. DOI: https://doi.org/10.1007/s00779-023-01712-5

22. Wefunder – Invest in Startups. URL: https://wefunder.com/ (Accessed: 24.05.2026).

23. Zhong W., Zhao L., Dou Q. The Implementation Strategy of Cost Control and the Construction of a Guarantee Model of Financial BPO in the Cloud Computing Environment. International Journal of Information System Modeling and Design. 2025. Vol. 16, № 1. DOI: https://doi.org/10.4018/IJISMD.367278

24. Zhou X. Blockchain+big data: Smart contract design for decentralized financial sharing platform. Edelweiss Applied Science and Technology. 2025. Vol. 9, № 4. P. 1942–1961. DOI: https://doi.org/10.55214/25768484.v9i4.6430

ДОДАТОК А

Результати E2E-тестування

Start-Sleep -Seconds 8; cd C:\Users\dima2\GolandProjects\diplom; powershell -File scripts/e2e-test.ps1 2>&1 | Tee-Object -FilePath docs/e2e-results-output.txt

=== E2E API Tests ===

Base URL: http://127.0.0.1:8080/api/v1

[PASS] #1 POST /auth/register -> 201

[PASS] #2 POST /auth/register -> 201

[PASS] #3 POST /auth/register -> 201

[PASS] #4 GET /auth/verify-email?token=4c3ab20253b00c77b78228b2ca6bba1df7aef82ee93d8746e8fc06215a66ff9d -> 200

[PASS] #5 GET /auth/verify-email?token=8afc99d73cb9dcb9ed3618bb9cfdd1342ce15db465c050c7944d9107f9b2ace2 -> 200

[PASS] #6 GET /auth/verify-email?token=e86179c65bcad36c07413d6ccfa8d12ba572abfd7ae33e423b694834d24bcfed -> 200

[PASS] #7 POST /auth/login -> 200

[PASS] #8 POST /auth/login -> 200

[PASS] #9 POST /auth/login -> 200

[PASS] #10 GET /identity -> 200

OK Status Data

-- ------ ----

True 200 @{status=verified}

True 200 @{status=verified}

True 200 @{status=verified}

True 200 @{created_at=2026-05-24T16:03:16.502179Z; email=founder-1779638596@test.com; email_verified=True; first_...

[PASS] #11 POST /auth/refresh -> 200

True 200 @{access_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImZvdW5kZXItMTc3OTYzODU5NkB0ZXN0LmNvbSIs...

[PASS] #12 POST /audit -> 201

True 201 @{action=e2e_test; created_at=2026-05-24T16:03:18.516250874Z; entity_id=7b76054d-1bcc-436d-9b03-527c4b49...

[PASS] #13 GET /auth/google -> 503

True 503 @{code=OAUTH_NOT_CONFIGURED}

[PASS] #14 GET /profiles/me -> 200

[PASS] #15 PUT /profiles/me -> 200

True 200 @{bio=Founder bio; country=UA; created_at=2026-05-24T16:03:16.588143Z; email=founder-1779638596@test.com...

[PASS] #16 PUT /profiles/startup -> 200

True 200 @{company_name=E2E Startup 1779638596; country=UA; description=AI platform; founded_year=2024; industry=...

[PASS] #17 GET /profiles/startup -> 200

True 200 @{company_name=E2E Startup 1779638596; country=UA; description=AI platform; founded_year=2024; industry=...

[PASS] #18 GET /profiles/me -> 200

[PASS] #19 PUT /profiles/investor -> 200

True 200 @{investment_range_max=50000; investment_range_min=1000; preferred_sectors=System.Object[]; preferred_st...

[PASS] #20 GET /profiles/investor -> 200

True 200 @{investment_range_max=50000; investment_range_min=1000; preferred_sectors=System.Object[]; preferred_st...

[PASS] #21 GET /profiles/93db7a06-5fdb-43b0-99b4-f486ad573e2d -> 200

True 200 @{bio=Founder bio; country=UA; first_name=Found; id=93db7a06-5fdb-43b0-99b4-f486ad573e2d; kyc_status=PEN...

[PASS] #22 GET /admin/profiles?limit=10 -> 200

True 200 @{items=System.Object[]; total=3}

[PASS] #23 GET /startups?founder_profile_id=93db7a06-5fdb-43b0-99b4-f486ad573e2d -> 200

Catalog startup id: 0d3a4909-1b63-4c6b-8498-fcfeaf772a31 slug: e2e-startup-1779638596-93db7a06

[PASS] #24 GET /startups -> 200

True 200 @{items=System.Object[]}

[PASS] #25 GET /startups/e2e-startup-1779638596-93db7a06 -> 200

True 200 @{country=UA; created_at=2026-05-24T16:03:18.644754Z; description=AI platform; founded_year=2024; founde...

[PASS] #26 POST /campaigns -> 201

[PASS] #27 GET /campaigns?startup_id=0d3a4909-1b63-4c6b-8498-fcfeaf772a31 -> 200

True 200 @{items=System.Object[]}

[PASS] #28 PUT /campaigns/8a7e8587-f6ea-48f0-9cda-9a13ea0f7626 -> 200

True 200 @{created_at=2026-05-24T16:03:18.874557Z; created_by=93db7a06-5fdb-43b0-99b4-f486ad573e2d; deadline=2026...

[PASS] #29 POST /campaigns/8a7e8587-f6ea-48f0-9cda-9a13ea0f7626/submit -> 200

True 200 @{created_at=2026-05-24T16:03:18.874557Z; created_by=93db7a06-5fdb-43b0-99b4-f486ad573e2d; deadline=2026...

[PASS] #30 POST /admin/campaigns/8a7e8587-f6ea-48f0-9cda-9a13ea0f7626/approve -> 200

True 200 @{created_at=2026-05-24T16:03:18.874557Z; created_by=93db7a06-5fdb-43b0-99b4-f486ad573e2d; deadline=2026...

[PASS] #31 GET /campaigns?status=active -> 200

True 200 @{items=System.Object[]}

[PASS] #32 POST /investments -> 201

[PASS] #33 GET /investments?investor_profile_id=343d26cc-cb8f-4004-901d-c3a3c77ff4a5 -> 200

True 200 @{items=System.Object[]}

[PASS] #34 GET /investments/0661e256-29a3-4813-bd9b-bd6d6ee3d97e -> 200

True 200 @{amount=5000; campaign_id=8a7e8587-f6ea-48f0-9cda-9a13ea0f7626; created_at=2026-05-24T16:03:19.006299Z;...

[PASS] #35 GET /portfolio?investor_profile_id=343d26cc-cb8f-4004-901d-c3a3c77ff4a5 -> 200

True 200 @{active=System.Object[]; completed=System.Object[]}

[PASS] #36 GET /dashboard/startup/0d3a4909-1b63-4c6b-8498-fcfeaf772a31?campaign_ids=8a7e8587-f6ea-48f0-9cda-9a13ea0f7626 -> 200

True 200 @{campaigns=System.Object[]; investor_count=0; startup_id=0d3a4909-1b63-4c6b-8498-fcfeaf772a31; total_ra...

[PASS] #37 GET /documents/689fd521-3c31-4506-8b01-bb4557e64f25 -> 200

True 200 @{content_type=text/plain; created_at=2026-05-24T16:03:19.14122Z; entity_id=8a7e8587-f6ea-48f0-9cda-9a13...

[PASS] #38 GET /documents/689fd521-3c31-4506-8b01-bb4557e64f25?download=true -> 200

True 200 E2E test document content

[PASS] #39 POST /notifications/send -> 201

[PASS] #40 GET /notifications?user_id=343d26cc-cb8f-4004-901d-c3a3c77ff4a5 -> 200

True 200 @{items=System.Object[]}

[PASS] #41 PATCH /notifications/0b12be3d-bcf1-472b-8b99-e74ad3de36d4/read -> 200

True 200 @{body=Your investment was recorded; channel=in_app; created_at=2026-05-24T16:03:19.246069Z; id=0b12be3d...

[PASS] #42 POST /questions -> 201

[PASS] #43 GET /questions?campaign_id=8a7e8587-f6ea-48f0-9cda-9a13ea0f7626 -> 200

True 200 @{items=System.Object[]}

[PASS] #44 POST /questions/95550a21-fea9-4332-9174-3d366b89c3fb/answers -> 201

True 201 @{author_id=93db7a06-5fdb-43b0-99b4-f486ad573e2d; content=About 50k/month; created_at=2026-05-24T16:03:1...

[PASS] #45 POST /comments -> 201

True 201 @{author_id=343d26cc-cb8f-4004-901d-c3a3c77ff4a5; content=Great pitch!; created_at=2026-05-24T16:03:19.3...

=== SUMMARY ===

Passed: 45 / 45

Failed: 0

ДОДАТОК Б

реалізації сторінки інвестування

```
async function handleInvest(e: React.FormEvent) {
  e.preventDefault();
  if (!campaign || !profile) return;
  setLoading(true);
  setMsg('');
  try {
    await createInvestment({
      campaign_id: campaign.id,
      investor_profile_id: profile.id,
      amount: Number(amount),
      terms_accepted: terms,
    });
    setMsg(t('common.success'));
    setAmount('');
    void reloadCampaignAfterPayment(startup!.id);
  } catch (err) {
    if (axios.isAxiosError(err) && err.response?.data?.code === 'KYC_NOT_APPROVED') {
      setMsg(t('kyc.notApproved'));
    } else {
      setMsg(t('common.error'));
    }
  } finally {
    setLoading(false);
  }
}
export async function createInvestment(data: CreateInvestmentRequest): Promise<Investment>
{
  const { data: investment } = await apiClient.post<Investment>('/investments', data);
  return investment;
}
```

ДОДАТОК В

Маршрутизація Gateway та JWT middleware

```
func (h *Handler) Start(_ context.Context) error {
    api := h.router.Group("/api/v1")
    api.Any("/auth/*", h.forward("auth"))
    h.mount(api, "startups", "/startups")
    h.mount(api, "campaigns", "/campaigns")
    h.mount(api, "investments", "/investments")
    api.Any("/admin/campaigns/*", h.forward("campaigns"))
    api.Any("/admin/*", h.forward("users"))
}

func (h *Handler) authMiddleware() echo.MiddlewareFunc {
    return func(next echo.HandlerFunc) echo.HandlerFunc {
        return func(c echo.Context) error {
            if isPublicRoute(c.Request().Method, c.Request().URL.Path) {
                return next(c)
            }
            claims, err := h.authClient.ValidateToken(ctx, token)
            c.Request().Header.Set("X-Identity-Id", claims.IdentityID.String())
            c.Request().Header.Set("X-User-Role", claims.Role)
            return next(c)
        }
    }
}
```