НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмне забезпечення для Peer-To-Peer Lending платформи»

Виконав:
студент IV курсу, групи КП-51
Тимошенко Владислав Анатолійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Олещенко Л.М. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри АУТС ФІОТ, к.т.н.,
Полторак В.П. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення» («Програмне забезпечення комп'ютерних та інформаційно-пошукових систем»)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Тимошенку Владиславу Анатолійовичу

1. Тема проекту «Програмне забезпечення для Peer-To-Peer Lending платформи», керівник проекту Олещенко Любов Михайлівна, к.т.н., доцент, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту «16» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих програмних рішень;
   - обґрунтування вибору засобів реалізації;
   - розроблення архітектури та програмних модулів;
   - аналіз розробленого програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - процес інвестування (креслення);
   - зв' язок модулів (креслення);
   - схема бази даних (плакат);
   - UML - діаграма. Варіанти використання (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Розроблення структури програмного забезпечення	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення дизайну програмного забезпечення	03.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
7.	Розроблення програмного забезпечення	10.03.2019	
8.	Тестування програмного забезпечення	17.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
11.	Підготовка графічної частини дипломного проекту	21.04.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студент В.А. Тимошенко

Керівник проекту Л.М. Олещенко

АНОТАЦІЯ

Даний дипломний проект присвячений розробленню програмного забезпечення для Peer-To-Peer Lending платформи.

Розроблене програмне забезпечення являє собою сайт, який надає змогу розміщувати інвестиції за принципом Peer-To-Peer, прогнозувати очікуваний прибуток, автоматично генерувати інвестицію, створювати інвестицію з окремих кредитів та виводити детальну інформацію про кожен з кредитів.

У даному дипломному проекті було розроблено архітектуру, реалізовано функції автоматичного та самостійного інвестування, модуль особистого кабінету, модуль перегляду створених інвестицій, модуль перегляду транзакцій користувача, модулі для введення та виведення коштів, модуль для зміни мови, модуль для завантаження документів користувача, розроблено дизайн проекту. Для збільшення продуктивності клієнтської частини, було використано серверний рендеринг. Реалізовано захист персональних даних шляхом хешування персональної інформації користувача в базі даних.

ABSTRACT

This diploma project deals with the development of software for the Peer-To-Peer Lending platform.

Developed software is a site that allows you to invest in Peer-To-Peer, predict expected return, automatically generate an investment, create an investment from individual loans, and display detailed information about each loan.

In this diploma project, an architecture was developed, automatic and independent investment functions were implemented, a personal cabinet module, a revised investment view module, a user transaction view module, input and output modules, a language modulator, a user upload module, a design platform. Server-side rendering was used to increase the client's performance. The protection of personal data has been realized by hashing the user's personal information in the database.

ДП.045440-01-90 Програмне забезпечення для Peer-To-Peer Lending платформи.
Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмне забезпечення	4	
	для Peer-To-Peer Lending		
	платформи. Технічне		
	завдання		
ДП.045440-03-81	Програмне забезпечення	70	
	для Peer-To-Peer Lending		
	платформи.		
	Пояснювальна записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для Peer-To-Peer Lending		
	платформи. Програма		
	та методика тестування		
ДП.045440-05-34	Програмне забезпечення	11	
	для Peer-To-Peer Lending		
	платформи. Керівництво		
	користувача		
ДП.045440-06-99	Програмне забезпечення	1	
	для Peer-To-Peer Lending		
	платформи. Зв’язок		
	модулів. Діаграма		
	компонентів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Програмне забезпечення	1	
	для Peer-To-Peer Lending		
	платформи. Процес		
	інвестування. Діаграма		
	діяльності		
ДП.045440-08-98	Програмне забезпечення	1	
	для Peer-To-Peer Lending		
	платформи. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ PEER-TO-PEER LENDING ПЛАТФОРМИ

Технічне завдання

ДП.045430-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.М. Олещенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ В.А. Тимошенко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для Peer-To-Peer Lending платформи.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості платформи для Peer-To-Peer інвестування.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинно забезпечувати такі функціональні можливості:

1) можливість автоматичного генерування інвестиції;
2) можливість інвестування окремі кредити;
3) можливість авторизації, реєстрації користувачів;
4) можливість вводу та виводу коштів;
5) можливість перегляду своїх інвестицій;
6) можливість перегляду детальної статистики по інвестиції;
7) можливість зміни мови;
8) можливість перегляду транзакцій;
9) можливість верифікації персональної інформації.

Розробку виконано з використанням мови Java і фреймворку Spring Boot для розроблення серверної частини, JavaScript і фреймворку Angular для розроблення клієнтської частини та PostgreSQL – для бази даних.

Додаткові вимоги:

1) динамічне оновлення даних;

2) зміна мови інтерфейсу користувача.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

– «Зв'язок модулів. Діаграма компонентів»;

– «Процес інвестування. Діаграма діяльності».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту……..14.11.2018

Розроблення та узгодження технічного завдання……. 28.11.2018

Розроблення структури програмного забезпечення…..15.12.2018

Розроблення дизайну інтерфейсу……03.02.2019

Програмна реалізація програмного забезпечення…….10.03.2019

Тестування програмного забезпечення…….. 17.03.2019

Підготовка матеріалів графічної частини проекту……21.04.2019

Оформлення технічної документації проекту……... 26.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до "Програми та методики тестування".

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ PEER-TO-PEER LENDING ПЛАТФОРМИ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.М. Олещенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ В.А. Тимошенко

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

СКБД – система керування базою даних

P2P (Peer-To-Peer) інвестування – спосіб займу грошей жодним чином не пов'язаним між собою особам або «рівноправним сторонам» без залучення традиційного фінансового посередника, наприклад, банку.

UI (User interface) – графічний інтерфейс користувача, який забезпечує передачу інформації між користувачем і програмно–апаратним компонентом комп'ютерної системи.

SPA (single page application) – веб–додаток або веб–сайт, який взаємодіє з користувачем, динамічно переписуючи поточну сторінку, а не завантажуючи усі нові сторінки з сервера.

MVC (model-view-controller) – архітектурний шаблон, який використовується для проектування та розробки програмного забезпечення.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем.

JSON (JavaScript Object Notation) – формат обміну даними між комп’ютерами.

ACID (Atomicity, Consistency, Isolation, Durability) – атомарність, узгодженість, ізольованість, довговічність, це набір властивостей, що гарантують надійну роботу транзакцій бази даних.

HTTP (HyperText Transfer Protocol) – протокол прикладного рівня для передачі даних у мережі Інтернет.

TCP/IP (Transmission Control Protocol/Internet Protocol) – концептуальна модель і набір комунікаційних протоколів транспортного рівня, що використовуються в Інтернеті та подібних комп'ютерних мережах.

HTTPS (HyperText Transfer Protocol Secure) – розширення протоколу HTTP, що підтримує шифрування за допомогою криптографічних протоколів SSL і TLS.

SEO (Search engine optimization) – процес підвищення якості та кількості трафіку веб–сайту, збільшення відвідуваності веб-сайту або веб-сторінки користувачам веб-пошукової системи.

JVM (Java Virtual Machine) – віртуальна машина, яка дозволяє комп'ютеру запускати програми Java, а також програми, написані на інших мовах, які також компілюються в байт-код Java.

SSL (Secure Sockets Layer) – стандартна технологія для встановлення безпечного зашифрованого зв'язку між веб-сервером і браузером

TLS (Transport Layer Security) – протокол, що забезпечує безпечний зв'язок між клієнт-серверними програмами, які взаємодіють одна з одною через мережу Інтернет.

ВСТУП

У наш час багато людей інвестують свої гроші з метою отримання прибутку, але у банках або інших фінансових установах їх можуть не влаштовувати ставки по депозитам, або ж взагалі банківська система, саме тому все більш популярними стають P2P-інвестування.

На сьогодні P2P -інвестування є одним з найбільш швидкозростаючих сегментів світового кредитного ринку з щорічним приростом на рівні 100%. Відповідно до цієї моделі кредитування, видача і отримання позик фізичним особам здійснюється безпосередньо, без залучення банків або кредитних організацій, що вигідно і позичальнику, і інвестору. Але, не дивлячись на значні успіхи P2P-інвестування на світовому ринку, в Україні цей сегмент ще недостатньо розвинений, багато в чому через недосконалість правової системи, а також через слабке розуміння принципів роботи прямого інвестування [1].

Для здійснення P2P-інвестування в мережі Інтернет програмне забезпечення повинне надавати можливості формування кредитного портфелю власноруч; формування кредитного портфелю автоматично; перегляду необхідної інформації для інвестування по кредиту; перегляду інформації по рухам коштів на рахунку клієнта; прогнозування очікуваного прибутку; можливість динамічного оновлення даних у разі зміни інформації по кредиту; введення та виведення коштів та ідентифікацію користувачів.

У дипломному проекті проаналізовано існуючі на даний час програмні рішення та запропоновано програмне забезпечення з урахуванням недоліків існуючих розробок. Дане програмне забезпечення не потребує від користувача спеціальних фінансових знань та навичок програмування, що робить його доступним для кожного.

1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Загальний опис проблеми інвестування грошей у мережі Інтернет

Інвестування грошей – доволі складний і кропіткий процес, так як інвестор зацікавлений в отриманні максимального прибутку зі своїх інвестицій, а також в гарантії збереження власних коштів. У наш час найпопулярніший спосіб інвестувати гроші – покласти їх на депозит у банк. Проте банк надає відносно невеликі відсотки і це не влаштовує багатьох інвесторів. Альтернативою банку виступають P2P-платформи, які також гарантують збереження власних коштів, проте надають більші відсотки.

P2P-інвестування – це пряме фінансування від інвестора до позичальника без участі банків, кредитних організацій або фінансових установ. В P2P бере участь лише платформа, задача якої – зробити процес інвестування прозорим, зрозумілим і безпечним. На рис. 1 зображено процес роботи P2P-платформи.

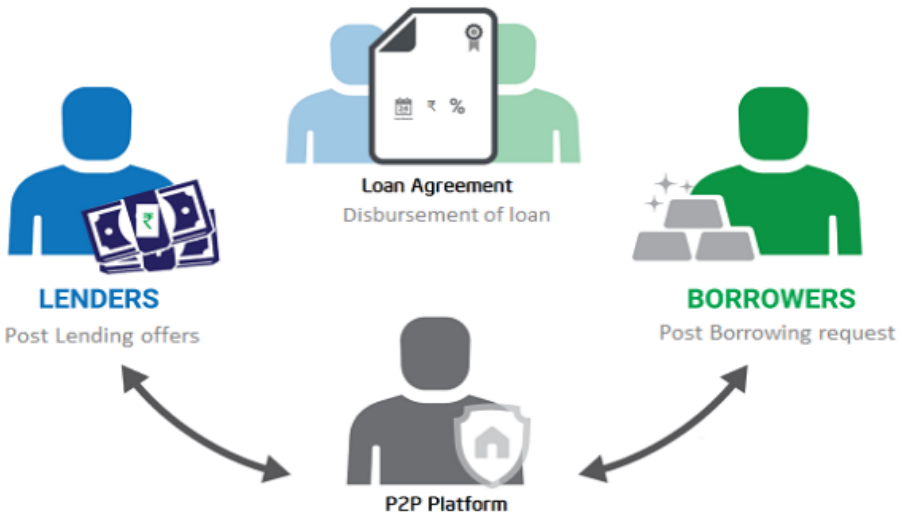


Рис. 1. Процес роботи P2P-платформи [2]

Перша така платформа розпочала роботу в 2005 році у Великобританії. У 2006 подібні платформи почали з’являтися у США.

Згодом ідея P2P – платформ розповсюдилась у всьому світі, і сьогодні вони впевнено займають свою долю фінансового ринку.

Протягом перших років існування таких платформ виникали великі труднощі через значну кількість випадків неповернення грошей позичальниками. Згодом були прийняті заходи зі зниження ризику неповернення кредитів позичальниками, частка неповернутих кредитів суттєво зменшилась і інвестори почали отримувати обіцяний прибуток.

Також велику роль у розвитку P2P-інвестицій зіграла фінансова криза 2008 року, оскільки багатьом позичальникам було відмовлено у видачі кредитів від банків, а також інвестори втратили довіру до банків. Інвестори і позичальники почали шукати альтернативу банкам. Так на фінансовому ринку з'явилась вільна ніша, яку зайняли P2P-платформи. Також розвитку цього виду платформ сприяла їх допомога у кредитуванні малого бізнесу, в той час, як банки не охоче брались за даний вид кредитування. Так, наприклад, у травні 2012 року британський уряд пообіцяв інвестувати £ 100 мільйонів у малий бізнес за допомогою альтернативних каналів кредитування, в тому числі P2P-платформ в надії обійти банки, які не хотіли видавати подібні кредити.

P2P, щорічний приріст якого становить понад 100%, є однією з областей інвестування, що швидко розвиваються. Процентні ставки варіюються від 5,6% до 35,8% в залежності від терміну позики і рейтингу позичальника. Для сумнівних позичальників вводяться штрафні ставки, що знаходяться в діапазоні від 1,5% до 10%. Керівники традиційних фінансових інститутів все частіше приєднується до компаній P2P у якості членів правління, кредиторів і інвесторів, а це є свідченням того, що нова фінансова модель займає свою власну нішу в рамках основної тенденції [3].

Одним з аспектів, на які треба звернути увагу, є законодавство Європейського союзу для P2P-інвестування. Вимоги до розкриття інформації подібні до вимог банку. На відміну від банку, P2P-платформи мають право оприлюднювати інформацію про кредит, але не персональні

дані позичальника, такі як ім’я, телефон тощо [4]. Також необхідно обов’язково проводити ідентифікацію користувачів в зв’язку з законом про запобігання легалізації доходів від злочинної діяльності та фінансування тероризму [5].

Найважливіший елемент роботи системи P2P – диверсифікація ризику неповернення, або ж прострочення позики позичальником. Ризик неповернення, або ж прострочення займу знижується через те, що кредитний портфель інвестора ділиться на невеликі за розміром позики великій кількості позичальників. Коли дехто з позичальників не повертає або ж прострочує позику, то це не несе критичних наслідків для усього кредитного портфеля інвестора і гарантує отримання очікуваного прибутку [6]. На рис. 2 зображено процес диверсифікації ризику неповернення.

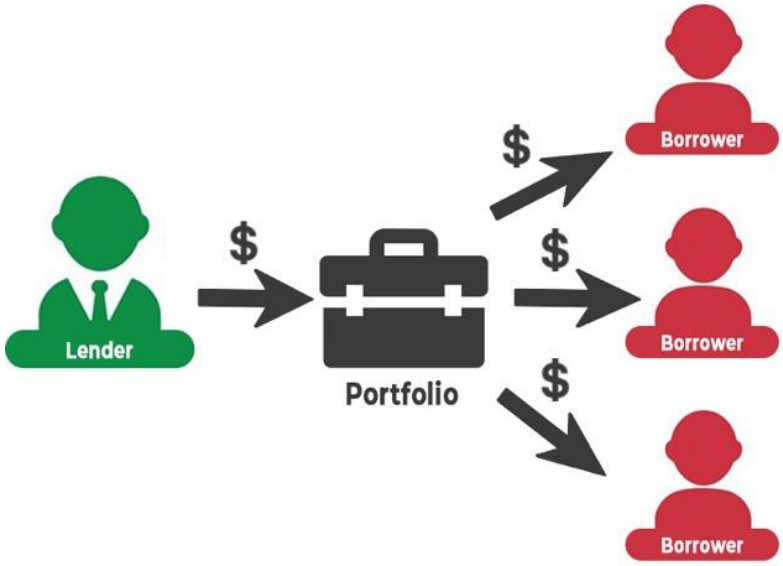


Рис. 2. Процес диверсифікації ризику неповернення коштів

В результаті автоматизації багатьох послуг накладні витрати знижуються, і обслуговування обходиться дешевше, ніж надання послуг звичайними фінансовими установами. Внаслідок цього інвестори

отримують більш високі доходи. Для прикладу наведемо офіційні дані «Wall Fargo Bank» (банк) [7] і Prosper Loans (P2P платформа) [8].

– Wall Fargo Bank: 270 000 співробітників і 9000 офісних приміщень;
– Prosper Loans: 240 співробітників і 2 офісні приміщення.

Основні переваги P2P-платформи:

– альтернатива вибору – інвестор може самостійно зібрати свій портфель з кредитів і вийти на ту проценту ставку, яка йому необхідна, або сформувати свій портфель автоматично, скориставшись можливостями платформи. Для допомоги в самостійному виборі кредитів, платформа надає інвестору всю необхідну інформацію про позичальника;
– короткий шлях грошей від інвестора до позичальника;
– мінімізація ризику через диверсифікацію ризику неповернення;
– зручність – усі операції проходять онлайн, тобто отримати позику, або інвестувати гроші можна, не виходячи з дому, достатньо лише зареєструватися у системі та надати всю необхідну інформацію, витративши на цей процес кілька хвилин;
– фінансова вигода – P2P надає вищі процентні ставки ніж у інших фінансових установах;
– платформа автоматично генерує договір по кожній з позик інвестора;
– максимально прозора система – інвестор бачить рух коштів по кожному з своїх кредитів, а також в допомогу йому, платформа генерує графіки очікуваного прибутку відповідно до його портфеля;
– страхування інвестицій – платформа бере на себе увесь ризик, пов’язаний з ризиком інвестування;
– інвестор може отримати прибуток відразу, як тільки позичальник погасить свою позику.

Згідно дослідження PricewaterhouseCoopers, вісім з десяти банків розраховують у найближчі 3–5 років створити стратегічні партнерства з P2P

сервісами та цифровими платформами грошових переказів [9]. За прогнозами Foundation Capital [10], до 2025 року обсяг світового ринку P2P-кредитування досягне $ 1 трлн.

Отже, на даний момент P2P – це признаний фінансовий інструмент з великим потенціалом розвитку.

1.2. Аналіз існуючих програмних рішень

Розглянемо основні недоліки існуючих програмних рішень для P2P- інвестування. Для того, щоб розібратися в особливостях проектування ПЗ для подібних задач, необхідно витратити багато часу, а у деяких випадках без спеціальних фінансових знань не обійтись. Нижче представлені найбільш поширені P2P-платформи, а також їх переваги та недоліки.

1.2.1. *P2P платформа Mintos*

Mintos – одна з найпопулярніших платформ у Європі для P2P-інвестування. Цільова аудиторія даної платформи – інвестори, які хочуть інвестувати гроші в кредити позичальників (рис. 3).

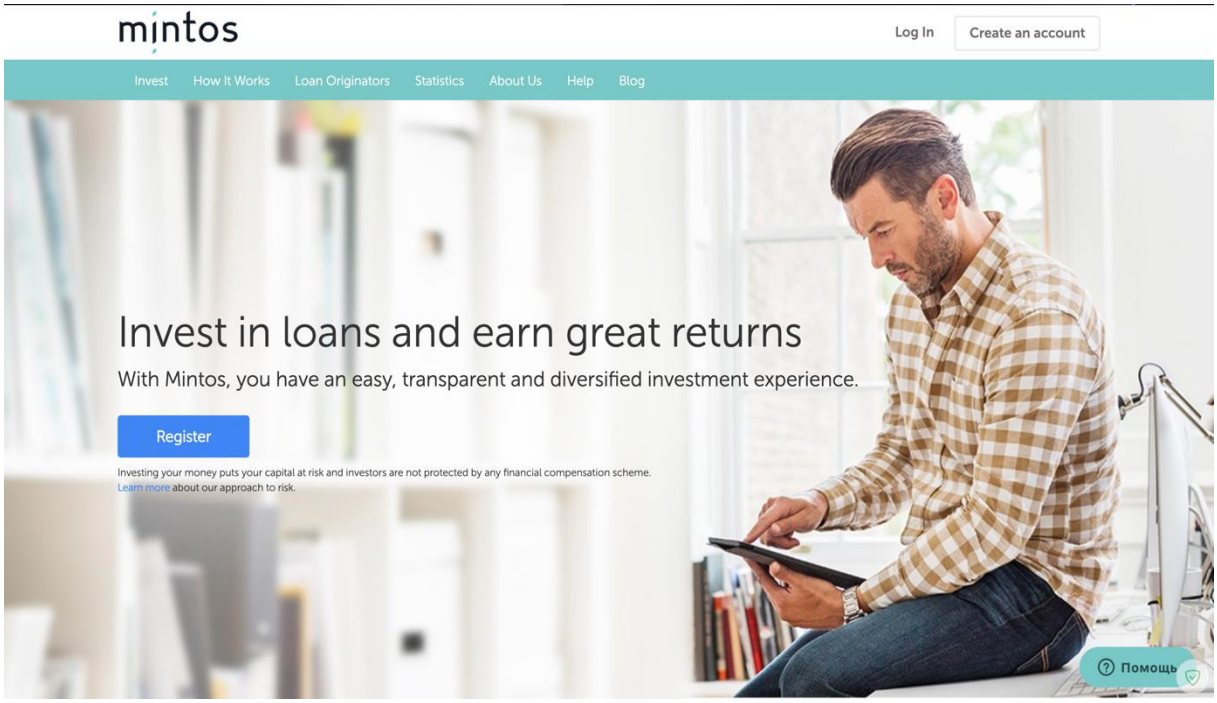


Рис. 3. P2P-платформа Mintos [6]

Основна задача платформи – створити зв’язок між інвестором та позичальником та забезпечити їм комфортні умови співпраці і гарантувати прибуток інвестору. Платформа надає можливість сформувати інвестору власний кредитний портфель, у якому він може вибрати самостійно в які кредити він буде інвестувати. В цілому платформа має зручний графічний інтерфейс користувача, але через відсутність SPA-архітектури в клієнтській частині, при переході з одного місця платформи в інше, користувач бачить лише білий екран, а при невеликій швидкості Інтернет – з’єднання це критично. Також у інвестора немає можливості сформувати портфель автоматично, скориставшись можливостями платформи, що відлякує частину інвесторів, які хочуть спробувати даний вид інвестування [11].

Переваги програмного забезпечення:

– наявність декількох мов на платформі;
– візуалізація даних за допомогою графіків;
– продумана система диверсифікації ризику для інвестора;
– можливість сформувати свій кредитний портфель власноруч;
– зручний графічний інтерфейс користувача;
– наявність всієї необхідної інформації про позичальника;
– велика кількість доступних кредитів для інвестування.

Недоліки програмного забезпечення:

– відсутність SPA-архітектури в клієнтській частині;
– відсутність можливості сформувати кредитний портфель автоматично;
– відсутність динамічного оновлення даних.

1.2.2. *P2P – платформа PeerBerry*

PeerBerry – P2P-платформа, орієнтована на Європейський фінансовий ринок. PeerBerry дозволяє користувачам інвестувати в кредити, надані від небанківських фінансових установ (кредитори) (рис. 4).

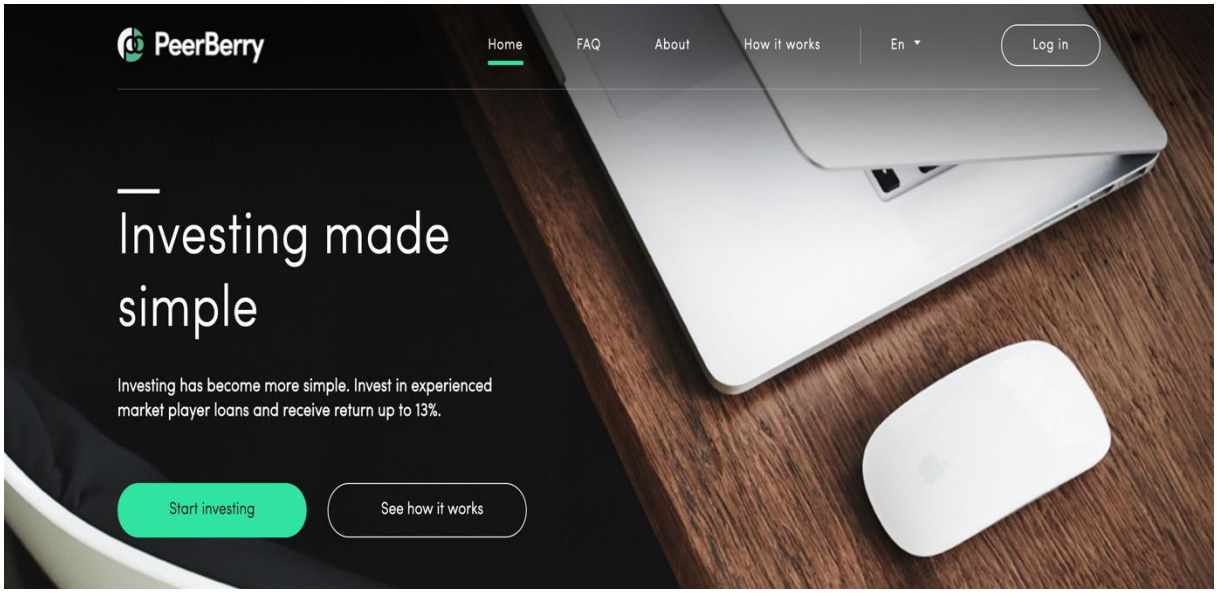


Рис. 4. P2P-платформа PeerBerry [7]

Більшість позичальників кредитів мають гарантію повернення коштів у сфері споживчих кредитів, кредитів на нерухомість та кредитів на авто. Спеціалізується як на короткострокову, так і на довгострокову співпрацю з інвестором [12].

Переваги програмного забезпечення:

– наявність декількох мов;

– система диверсифікації ризику;

– SPA-архітектура у клієнтській частині;

– зручний графічний інтерфейс користувача;

– наявність короткострокового і довгострокового інвестування.

Недоліки програмного забезпечення:

– невелика кількість доступних кредитів;

– відсутність можливості сформувати кредитний портфель власноруч;

– відсутність динамічного оновлення даних;

– відсутність можливості візуалізації даних;

– недостатньо інформації про позичальника;

– відсутність детальної інформації про кожен кредит.

1.2.3. *P2P платформа Bondora*

Bondora – P2P-платформа, орієнтована на східний Європейський фінансовий ринок (рис. 5).

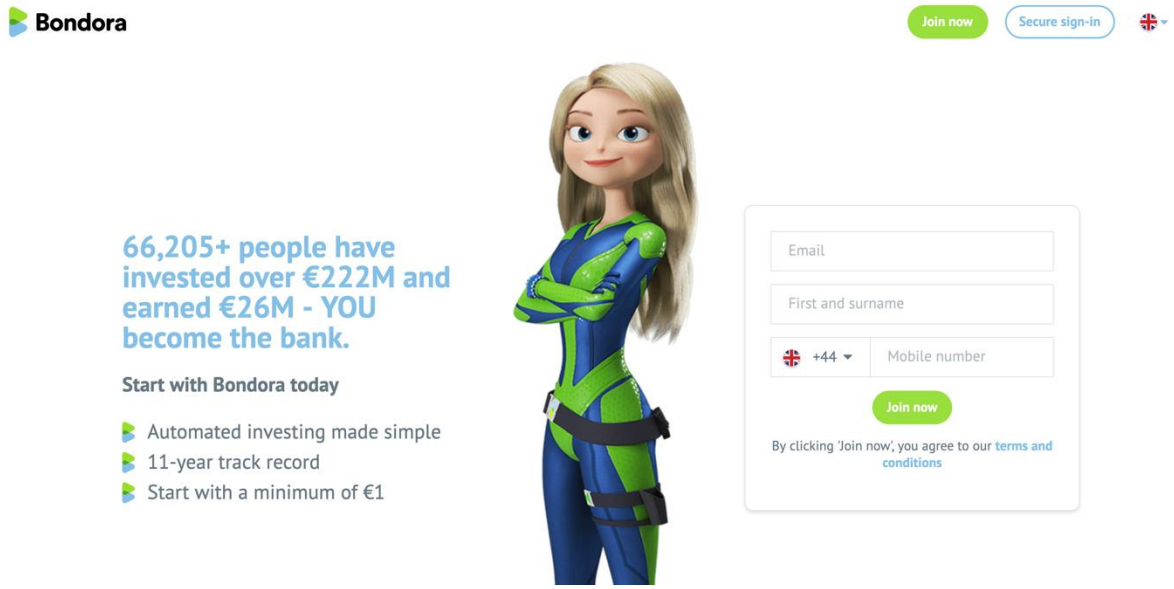


Рис. 5. P2P-платформа Bondora [8]

Bondora спеціалізується на довгострокову співпрацю з інвестором, мінімальний період інвестування варіюється в межах від 1 до 5 років. Платформа надає можливості для автоматизованого формування кредитного портфеля інвестора [13].

Переваги програмного забезпечення:

– можливість сформувати кредитний портфель автоматично;

– наявність декількох мов;

– продумана система диверсифікації ризику для інвестора;

– візуалізація даних за допомогою графіків;

– велика кількість доступних кредитів для інвестування.

Недоліки програмного забезпечення:

– відсутність можливості сформувати кредитний портфель власноруч;

– відсутність SPA-архітектури в клієнтській частині;

– відсутність динамічного оновлення даних;

– відсутня можливість короткострокового інвестування;

– недостатньо інформації про позичальника;

– відсутність детальної інформації про кожен кредит.

У таблиці 1 наведено порівняння P2P-платформ Mintos, PeerBerry, Bondora згідно вище сформульованих вимог для P2P-інвестування коштів у мережі Інтернет.

Таблиця 1

Порівняння P2P – платформ Mintos, PeerBerry, Bondora

Назва	Mintos	PeerBerry	Bondora
Опис	Одна з найпопулярніших P2P-платформ в Європі.	P2P-платформа, орієнтована на Європейський фінансовий ринок.	P2P-платформа, орієнтована на східний Європейський фінансовий ринок.
Система диверсифікації ризику	Так	Так	Так
Наявність декількох мов	Так	Так	Так
Візуалізація даних	Так	Ні	Так
Формування кредитного портфелю автоматично	Ні	Ні	Так
SPA-архітектура	Ні	Так	Ні
Динамічне оновлення даних	Ні	Ні	Ні

1.3. Постановка задачі

У результаті проведеного дослідження можна сформулювати такі вимоги до програмного забезпечення для Peer-To-Peer Lending платформи:

– можливість автоматичного генерування інвестиції;

– надання усієї необхідної інформації для збору свого кредитного портфеля, якщо інвестор захоче зібрати свій кредитний портфель власноруч;

– SPA-архітектура у клієнтській частині;
– максимально простий і зручний UI, щоб інвестор без спеціальних фінансових знань міг легко розібратись в системі;
– особистий кабінет користувача, у якому інвестор може продивитись інформацію про себе, а також надати всі необхідні документи;
– модуль для генерування договорів по кожній інвестиції;
– модуль для введення грошей на платформу;
– модуль для виведення грошей з платформи на банківський рахунок інвестора;
– модуль для динамічного оновлення інформації, реалізований через WebSockets;
– модуль для відображення руху коштів по кожній з інвестицій;
– модуль для зміни валюти інвестування;
– модуль для зміни мови;
– модуль для відображення кредитного портфеля інвестора та всю необхідну інформацію, пов’язану з ним;
– модуль, який буде відповідати за безпеку обміну даних між Front end частиною та Back end частиною платформи;
– модуль для забезпечення диверсифікації ризику у разі, якщо інвестор захоче зібрати свій кредитний портфель сам;
– модуль для серверного рендерингу.

1.4. Висновки до розділу

У даному розділі описано особливості процесу P2P-інвестування в мережі Інтернет, проаналізовано існуючі P2P lending платформи, виявлено їх недоліки та переваги.

Основними перевагами існуючих P2P lending платформ є наявність декількох мов, система диверсифікації ризику, візуалізація даних.

Основними недоліками розглянутих програмних платформ є відсутність SPA-архітектури в клієнтській частині, відсутність динамічного

оновлення даних та відсутність поєднання автоматичного інвестування з інвестуванням в окремі кредити.

На основі виділених недоліків існуючих програмних рішень сформульовано основні вимоги для розроблення програмного забезпечення.

Розроблюваний програмний продукт має забезпечувати можливість формування користувачем кредитного портфелю власноруч та автоматично, динамічно оновлювати дані, використовувати SPA-архітектуру та надавати можливість короткострокового та довгострокового інвестування.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування

У цьому підрозділі було розглянуто найбільш популярні мови програмування для розроблення програмного забезпечення P2P- інвестування у мережі Інтернет.

2.1.1. *Огляд мови програмування Java*

Java – об’єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems і випущена 23 травня 1995 року. Спочатку мова створювалася для побутових приладів, але згодом почала використовуватись для створення динамічних веб-сторінок – servlet [14].

На сьогоднішній день, Java – найпопулярніша мова програмування (18% від українського ринку), на якій створюють мобільні додатки для ОС Android, веб-додатки, веб-сервіси, десктопні програми, а також графічні додатки (Java FX). Код, написаний на Java, транслюється в байт-код, за допомогою JVM, яка оброблює байт код і передає команди, як інтерпретатор, що забезпечує роботу програми на будь-якій з сучасних платформ, на яку можна встановити віртуальну машину Java [15].

Розглянемо переваги мови програмування Java:

– Безпека. Функціонування обмежується віртуальною машиною Java. Перевірка достовірності відбувається на основі шифрування з відкритим ключем.
– Надійність. Java усуває помилки в різних ситуаціях, спираючись в основному на час компіляції та перевірку помилок під час виконання.
– Високопродуктивний. Just-in-time компілятор забезпечує високу продуктивність.
– Об'єктно-орієнтованість. В Java все є об'єктом, саме тому код можна легко розширювати.

– Простота. Для того щоб розуміти Java, потрібно розуміти концепції об'єктно-орієнтованого програмування.
– Кросплатформеність. Байт код поширюється через інтернет і інтерпретується в Java Virtual Machine, на якій він працює.
– Розповсюдженість. Java займає найбільшу частку на ринку в світу і впевнено її тримає протягом багатьох років.
– Архітектурно-нейтральний. Компілятор генерує архітектурно-нейтральні об'єкти формату файлу, що робить скомпільований код виконуваним на багатьох процесорах при наявності в системі Java Runtime.
– Багатопоточність. Java дозволяє писати програми, які виконують декілька завдань одночасно, що дозволяє створювати інтерактивно-налагоджені додатки.
– Динамічний. Класи завантажуються тільки на вимогу, тому не має необхідності мати знання про всі файли, або файлову систему.
– Garbage Collector. Garbage Collector керує управлінням пам’яті, і в разі необхідності звільнює її.
– Самодокументованість коду. Java може автоматично згенерувати документацію проекту, на основі коментарів в коді.
– Наявність багатьох бібліотек. В Java існує багато бібліотек, які вирішують різноманітні нетривіальні задач [16].

Розглянемо недоліки мови програмування Java:

– Платне комерційне використання. Oracle (компанія, яка володіє правами власності на Java) оголосила, що з 2019 року почне стягувати плату за використання Java Standard Edition 8 в «комерційних цілях». За все нові оновлення та виправлення помилок доведеться заплатити.
– Багатослівність. Довгі, надмірно складні речення ускладнюють читання і перегляд коду.

– Відсутність нативного дизайну. Для створення графічного інтерфейсу користувача у розробників немає нативних інструментів.
– Швидкість написання коду. Для того, щоб написати просту програму на Java, потрібно витратити більше часу ніж, на інших мовах програмування, через ускладнений синтаксис [17].

2.1.2. *Огляд мови програмування Python*

Python – високорівнева, об'єктно-орієнтована скриптова мова програмування, що інтерпретується. Python використовує ключові слова з англійської мови, що надає простоту читання коду, написаного на python [18].

Розробка python почалась в кінці 1980-х років. В 1991 році був опублікований код python і він почав впевнено набувати популярність в мережі Інтернет. Згодом була випущена версія python 2.0., якою зацікавилась велика кількість людей, і в розробці python з'явився елемент open-source – багато розробників прийняли участь у подальшій розробці мови програмування.

Мова постійно розвивається і випускаються нові версії python (найбільш розповсюдженні версії – це python 2.7 і python 3.x.). На сьогоднішній день python займає 12% від українського ринку [19].

Розглянемо переваги мови Python:

– Легкий для вивчення. У python проста структура і чітко визначений синтаксис. Завдяки цьому мову можна вивчити за достатньо короткий час і python широко використовується багатьма викладачами в якості мови навчання для студентів.
– Простота читання. Блоки коду виділяються за допомогою відступів, що значно спрощує читання коду.
– Простота обслуговування. Так як мова має просту структуру і чітко визначений синтаксис його легко підтримувати в подальшому.

– Кросплатформеність. В Python присутня широка крос-платформна бібліотека, що дозволяє мові запускатися на різних платформах і при цьому зберігати однаковий інтерфейс.
– Наявність інтерактивного режиму. Дозволяє "на льоту" тестувати потрібні ділянки коду.
– Розширюваність. В Python можна впроваджувати низькорівневі модулі, які написані на інших мовах програмування для найбільш гнучкого вирішення поставлених задач.
– Робота з різними базами даних. В стандартній бібліотеці Python наявні модулі для роботи з більшістю баз даних.
– Створення GUI. Python дозволяє створювати GUI додатки, які будуть працювати незалежно від типу платформи.
– Швидкість написання коду. Мова має широку підтримку бібліотек і чисті об'єктно-орієнтовані конструкції, які збільшують продуктивність програміста в 2-10 разів при використанні мов Java, C, C ++ і C # [20].

Розглянемо основні недоліки мови Python:

– Продуктивність. Python швидкий для багатьох веб-додатків, але він виконується за допомогою інтерпретатора замість компілятора, що сповільнює його, оскільки компіляція і виконання допомагають йому нормально працювати.
– Помилки під час виконання. Код на Python вимагає більше часу для тестування, оскільки деякі помилки, можна побачити, лише коли програми остаточно виконуються [21].

2.1.3. *Огляд мови програмування JavaScript*

JavaScript – об’єктно-орієнтована, динамічна, прототипна мова програмування, що інтерпретується [22].

JavaScript був створений Бренданом Айком (Brandan Eich) у 1995 році. У 1996 році мова JavaScript була стандартизована і отримала офіційне ім'я

ECMAScript, потім з’явилася ECMAScript 2 в 1998 і ECMAScript 3 в 1999 році. Це перетворилося в мову JavaScript, яка тепер працює не тільки в різних браузерах, але також на різних пристроях, включаючи мобільні та настільні комп'ютери.

Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, змінювати структуру та зовнішній вигляд веб-сторінки, роблячи її “більш живою”, асинхронно обмінюватися даними з сервером, та керувати браузером. Мова постійно розвивається і з кожним роком виходить нове версія стандарту – ECMAScript. З кожним роком популярність JavaScript зростає, так як все більше сервісів переходить в мережу Інтернет. На сьогоднішній день частка JavaScript займає 16% від українського ринку і з кожним роком зростає [23].

Розглянемо основні переваги JavaScript:

– Повна інтеграція з HTML і CSS. Повна інтеграція з кодом і оформленням сторінки (html і css) робить цю мову незамінною у веб-розробці.
– Гнучкість мови. Гнучкість мови дозволяє використовувати безліч шаблонів програмування відповідно до конкретних умов.
– Garbage Collector. Garbage Collector керує управлінням пам’яті, і в разі необхідності звільнює її.
– Мультипарадигмальна мова програмування. JavaScript об'єднує об'єктно-орієнтований і функціональний підходи.
– Підтримка. JavaScript підтримується всіма сучасними веб-браузерами.
– Стабільність роботи. JavaScript має високу стабільність роботу, так як при наявності не критичних помилок, код буде продовжувати виконуватися.
– Анімація. Мова має багато способів роботи з анімацією без використання сторонніх бібліотек [24].

Розглянемо основні недоліки JavaScript:

– Відсутність багатопоточності. У JavaScript код виконується лише в одному потоці.
– Реалізація наслідування. Наслідування у JavaScript ґрунтується на прототипах і відувається безпосередньо на об'єктах, на відміну від інших мов програмування.
– Відсутність статичної типізації. JavaScript підтримує тільки динамічну типізацію [25].

Таблиця 2

Порівняння мов програмування Java, Python, JavaScript

Назва	Java	Python	JavaScript
Опис	Об’єктно-орієнтована мова програмування	Високорівнева, об’єктно-орієнтована скриптова мова програмування, що інтерпретується	Об’єктно-орієнтована, динамічна, прототипна мова програмування, що інтерпретується
Об'єктно-орієнтованість	Так	Так	Так
Garbage Collector	Так	Так	Так
Гнучкість мови	Ні	Так	Так
Багатопоточність	Так	Так	Ні
Анімація	Ні	Ні	Так
Інтеграція з HTML і CSS	Ні	Ні	Так
Статична типізація	Так	Ні	Ні

2.2. Вибір системи керування бази даних

2.2.1. *PostgreSQL*

PostgreSQL – об'єктно-реляційна система керування базами даних (СКБД) з відкритим кодом. Основа PostgreSQL – серверний процес бази даних. Вона виконується на одному сервері. Такий поділ клієнтів і сервера дозволяє побудувати розподілену систему. Можна відокремити клієнтів від сервера за допомогою мережі і розробляти клієнтські програми в середовищі, яка буде зручною для користувача.

Розроблення СУБД PostgreSQL розпочалось в 1986 році. Перша версія, що мала назву Postgres, вийшла в 1988 році і почала активно використовуватись для реалізації багатьох різних досліджень і створення програмного забезпечення, а саме: системи аналізу фінансових даних, пакет моніторингу продуктивності реактивних двигунів, база даних переміщення астероїдів, база даних медичної інформації та декілька географічних інформаційних систем. Також СУБД використовувалась як засіб навчання декількох університетах. У 1994 році в СУБД додали інтерпретатор мови SQL, і вона отримала нову назву Postgres95. При розробці Postgres95 акцент ставився на виявлення і розуміння існуючих проблем у коді сервера. У 1996 році вийшла нова версія, яка отримала сучасну назву – PostgreSQL. У ній акцент розробки змістився на розширення можливостей і сумісності при продовженні роботи у всіх інших областях [26].

Розглянемо наступні переваги PostgreSQL:

– Модель даних. PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Фундаментальна характеристика об'єктно-реляційної бази даних – це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить PostgreSQL неймовірно гнучким і надійним.
– Структури і типи даних. Існує великий список типів даних, які підтримує PostgreSQL. Крім числових, з плаваючою точкою,

текстових, булевих і інших очікуваних типів даних (а також багато їх варіацій), PostgreSQL може похвалитися підтримкою uuid, грошового, геометричного, бінарного типів, мережевих адрес, бітових рядків, текстового пошуку, xml, json, масивів, композитних типів і діапазонів, а також деяких внутрішніх типів для ідентифікації об'єктів і логів.

– Створення нового типу. Якщо раптом великого списку типів даних PostgreSQL вам виявиться недостатньо, можна використовувати команду CREATE TYPE, щоб створити нові типи даних.
– Розміри даних. Зазвичай обсяг даних обмежується максимальним розміром файлів операційної системи. Оскільки PostgreSQL може зберігати табличні дані у файлах меншого розміру, він може обійти це обмеження.
– Цілісність даних. PostgreSQL прагне відповідати стандарту ANSI-SQL: 2008, відповідає вимогам ACID і відомий своєю транзакційною цілісністю.
– Якість написання вихідного коду. PostgreSQL також відома через свою якість написання вихідного коду: у автоматизовану досліженні в вихідному коді PostgreSQL на предмет помилок, було виявлено 20 помилок на 775 000 рядків вихідного коду (в середньому – це одна помилка на 39 000 рядків коду). Якщо порівнювати з іншими СКБД, то MySQL – 97 помилок на 8 000 рядків коду, FreeBSD – 306 помилок на 2 500 рядків коду. Це свідчить про велику надійність PostgreSQL.
– Відкрита для загального доступу. Порівнюючи PostgreSQL з іншими базами даних, з відкритим кодом (MySQL, Apache або ж FreeBSD), PostgreSQL не контролюється однією компанією, і її розробка відбувається людьми, та компаніями, які використовують цю СКБД [27].

Недоліками PostgreSQL є наступні:

– Продуктивність. При простих операціях читання PostgreSQL може значно уповільнити сервер і бути повільніше своїх конкурентів, таких як MySQL.
– Популярність. PostgreSQL не користується великою популярністю, хоча і має достатньо велику спільноту.
– Хостинг. Іноді досить складно знайти хостинг з підтримкою PostgreSQL. [28].

2.2.2. *MySQL*

MySQL – вільна, реляційна, серверна СКБД. MySQL функціональна і успішно працює з веб-додатками.

MySQL винайшов працівник компанії TcX Михайло Віденіус. У 1979 році він розробив UNIREG – прототип MySQL. Пізніше UNIREG була розширена для підтримки великих баз даних і була переписана на декількох мовах. У 1994 році компанія TcX стала розробляти програми для роботи в мережі Інтернет з використанням UNIREG. Згодом TcX переписали UNIREG, використовуючи утиліти сторонніх розробників для mSQL,написали API для своєї нової системи. Пізніше, в травні 1995 року, з'явилась MySQL 1.0.

MySQL пропонує багато інструментів для розробки додатків, незважаючи на те, що в ній не реалізована вся SQL функціональність. Так як це серверна СУБД, додатки для доступу до даних працюють зі службами MySQL [29].

Розглянемо основні переваги MySQL:

– Простота. Встановити MySQL досить просто, існує багато різноманітних програм, які облегшують роботу з БД.
– Реплікація. MySQL підтримує ряд різних типів реплікації – це практика обміну інформацією між двома або більше хостами, щоб

допомогти підвищити надійність, доступність і відмовостійкість. Це корисно для налаштування резервного копіювання бази даних або горизонтального масштабування однієї бази даних.

– Комплексна підтримка транзакцій. MySQL очолює список надійних транзакційних баз даних, доступних на ринку. Завдяки таким функціям, як повна атомарна, узгоджена, ізольована, довгострокова підтримка транзакцій, підтримка декількох версій транзакцій і необмежене блокування на рівні рядків, це ідеальне рішення для повної цілісності даних. Вона гарантує миттєву ідентифікацію тупикових ситуацій.
– Безпека. СУБД MySQL є надзвичайно безпечною, тому що вона пропонує функції, які забезпечують повний захист даних. Вона захищає сценарії доступу до даних за допомогою паролів. Паролі зберігаються в розширеній і складній формі шифрування. Вона також пропонує підтримку SSH і SSL для забезпечення безпеки підключень. Утиліти відновлення та резервного копіювання забезпечують фізичне і логічне резервне копіювання в додаток, до повного і моментального відновлення.
– Висока продуктивність. MySQL має відмінну структуру механізму зберігання, яка допомагає налаштовувати сервер бази даних MySQL для бездоганної продуктивності. Будь то веб-сайт електронної комерції, який отримує мільйон запитів щодня, або високошвидкісна система транзакційної обробки, MySQL призначена для задоволення найбільш вимогливих додатків, забезпечуючи оптимальну швидкість, повнотекстові індекси і унікальні кеші пам'яті для підвищення продуктивності [30].

Розглянемо основні недоліки MySQL:

– Обмеження. База даних не є повністю сумісною з SQL та обмежена в деяких областях, таких як: зберігання даних, відмовостійкість та діагностика продуктивності.

– Надійність. Іноді поступається іншим СУБД по надійності через деякі способи обробки даних MySQL (зв'язки, транзакції, аудити).
– Швидкість розробки. Існує багато скарг на процес розробки при використанні MySQL.
– Масштабування. MySQL не призначена для масштабування. Для того, щоб виконати масштабування, потрібні серйозні інженерні зусилля.
– Уповільнений розвиток. Оскільки СУБД MySQL була придбана компанією Sun Microsystems у 2008 році, а пізніше корпорацією Oracle у 2009 році, з'явилися скарги від користувачів, що процес розробки для СУБД значно сповільнився, оскільки співтовариство більше не має агентства для швидкого реагування на проблеми і внесення змін [31].

2.2.3. *SQLite*

SQLite – полегшена реляційна СКБД, яка реалізована у вигляді бібліотеки і базується на стандарті SQL-92.

Особливість SQLite – СУБД працює безпосередньо з файлами, в яких зберігаються дані, а не з сокетами і портами, як у мережевих СУБД. Ця особливість забезпечує швидкість роботи. SQLite має багато допоміжних бібліотек, які значно облегшують роботу з нею [32].

Розглянемо основні переваги SQLite:

– Зручний для користувача. SQLite іноді описується як база даних з нульовою конфігурацією, яка готова до використання з коробки. SQLite не запускається як серверний процес, а це означає, що його ніколи не потрібно зупиняти, запускати або перезапускати, а також не має жодних конфігураційних файлів, якими потрібно керувати. Ці функції допомагають упорядкувати шлях від інсталяції SQLite до його інтеграції з додатком.

– Портативність. На відміну від інших СКБД, які зазвичай зберігають дані як велику партію окремих файлів, вся база даних SQLite зберігається в одному файлі. Цей файл може бути розташований в будь-якому місці в ієрархії каталогів і може бути розділений за допомогою змінного носія або протоколу передачі файлів.
– Стандарти. На перший погляд може здатися, що ця СУБД дуже проста, але вона використовує SQL. Деякі особливості опущені (RIGHT OUTER JOIN або FOR EACH STATEMENT), але основні підтримуються.
– Зручність при розробленні та тестуванні. SQLite легко масштабується через свою структуру – один файл і бібліотека на C.
– Тестування. У SQLite є режим пам'яті, який можна використовувати для швидкого запуску тестів без накладних витрат на фактичні операції з базою даних, що робить його ідеальним вибором для тестування [33].

Основними недоліками SQLite є наступні:

– Обмежена паралельність. Хоча кілька процесів можуть одночасно отримувати доступ до бази даних SQLite і запитувати їх, тільки один процес може вносити зміни до бази даних у будь-який момент часу. Це означає, що SQLite підтримує більшу паралельність, ніж більшість інших вбудованих СКБД, але не так, як MySQL або PostgreSQL.
– Немає управління користувачами. Оскільки SQLite читає та записує безпосередньо до звичайного файлу диска, доступними є лише типові права доступу для операційної системи. Це робить SQLite поганим вибором для програм, які вимагають декількох користувачів із спеціальними правами доступу.
– Низька продуктивність для великих систем. Внаслідок свого проектування цю СУБД досить складно використовувати на великих проектах.

– Робота з великою кількістю даних. SQLite може технічно підтримувати базу даних розміром до 140 ТБ, доки диск і файлова система також підтримують вимоги до розміру бази даних. Однак, веб-сайт SQLite рекомендує, щоб будь-яка база даних, що наближається до 1 Тб, розміщувалася в централізованій базі даних клієнт-сервер, оскільки базу даних SQLite такого розміру або більше було б важко керувати [34].

Таблиця 3

Порівняння СКБД PostgreSQL, MySQL, SQLite

Назва	SQLite	MySQL	PostgreSQL
Архітектура	Файлова (вбудована)	Клієнт-сервер	Клієнт-сервер
ОС сервера	немає	FreeBSD, Linux, OS X, Solaris, Windows	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows
Реплікація	немає	Master-Slave, Master-Master	Master-Slave
Мова програмування (базовий код)	C	C, C++	C
Підтримувані мови програмування	ActionScript, Ada, Basic, C, C#, C++, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MATLAB, PHP, PL/SQL	C, C++, Delphi, Perl, Java, Lua, .NET, Node.js	NET, C, C++, Delphi, Java, Perl, PHP, Python, Tcl

2.3. Вибір середовища розроблення програмного забезпечення

Для розроблення ПЗ було обрано IDE IntelliJ IDEA. IntelliJ IDEA є спеціальним середовищем програмування або інтегрованим середовищем розробки (IDE), багато в чому призначеним для Java. Це середовище використовується особливо для розроблення програм.

IntelliJ IDEA була розроблена компанією JetBrains, раніше відомою як IntelliJ. Вперше вона була випущена в 2001 році, і стала дуже популярною через вдосконалену кодову навігацію і можливість рефакторингу коду.

IntelliJ IDEA навіть отримала відзнаку та була визнана кращим інструментом програмування на основі Java в 2010 році, обійшовши своїх конкурентів, таких як NetBeans, Eclipse і JDeveloper. Середовище розробки з відкритим вихідним кодом для Android, випущене Google у 2014 році, також базується на IntelliJ IDEA. IDE підтримує багато інших мов програмування, таких як Python, Lua і Scala, JavaScript. [35]

IntelliJ IDEA вважається одним з кращих засобів програмування на основі Java, через її допоміжні можливості, що робить її зручною у використанні і робить програми, створені в ній, добре продуманими. Вона також має розширені функції перевірки помилок, що дозволяє швидше і простіше перевіряти помилки. Що відрізняє IntelliJ IDEA від своїх аналогів – це простота використання, гнучкість і продуманий дизайн.

Також однією з основних переваг IntelliJ IDEA є те, що вона підтримує як Back end, так і Front end фреймворки – універсальна IDE.

2.4. Додаткові технології та бібліотеки

2.4.1. *Angular*

Angular – це платформа і основа для побудови клієнтських додатків у HTML і TypeScript. Angular написаний на TypeScript. Він реалізує основні та додаткові функції як набір бібліотек TypeScript, які імпортуються у програми.

Основними будівельними блоками програми Angular є NgModules, які забезпечують контекст компіляції для компонентів. NgModules збирають відповідний код у функціональні набори; Angular – додаток визначається набором NgModules. Додаток завжди має принаймні кореневий модуль, який дозволяє завантажувати початкову програму, і зазвичай має багато інших модулів.

Компоненти визначають види, які є набором елементів екрану, які Angular може вибрати і змінити відповідно до логіки програми та даних.

Компоненти використовують сервіси, які забезпечують певну функціональність, безпосередньо не пов'язану з переглядами. Сервіси можуть бути введені в компоненти як залежності, що робить код модульним, багаторазовим і ефективним.

Як компоненти, так і служби – це класи, де декоратори позначають їх тип і надають метадані, які повідомляють Angular, як їх використовувати.

Метадані для класу компонентів асоціюють його з шаблоном, який визначає подання. Шаблон поєднує звичайний HTML з директивами Angular та розміткою, що дозволяє Angular змінити HTML перед його відображенням.

Метадані для класу сервісу надають інформацію Angular, яка повинна зробити її доступною для компонентів через ін'єкцію залежностей (DI) [36].

Компоненти програми зазвичай визначають багато представленнь, організованих ієрархічно. Angular надає послугу маршрутизатора, щоб допомогти визначити шляхи навігації серед переглядів. Маршрутизатор забезпечує складні навігаційні можливості браузера. На рис. 6. Зображена структура Angular.

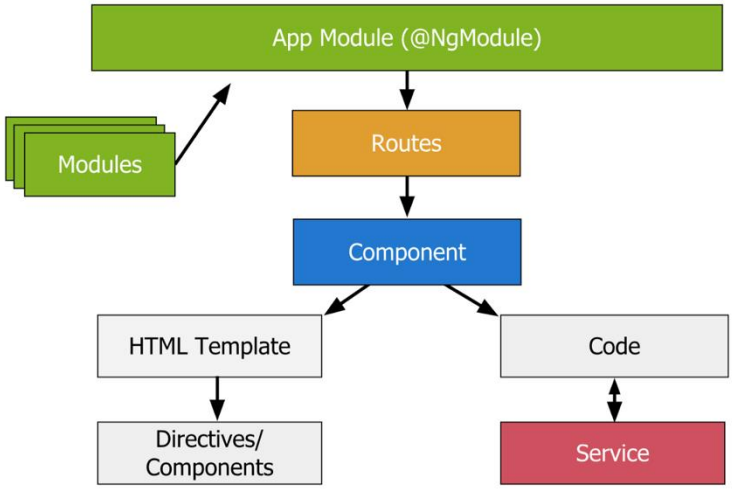


Рис. 6. Структура Angular [37]

2.4.2. *Spring Boot*

Spring Boot є відкритим вихідним кодом на основі Java, що використовується для створення мікросервісу. Він розроблений Pivotal Team і використовується для побудови автономних і готових до виготовлення Spring-додатків.

Мікросервіс – це архітектура, яка дозволяє розробникам розробляти та розгортати послуги самостійно. Кожна служба, що працює, має свій власний процес, і це забезпечує легку модель для підтримки бізнес-додатків.

Мікро-сервіси надають своїм розробникам такі переваги:

– простота розгортання;

– проста масштабованість;

– сумісність з контейнерами;

– мінімальна конфігурація;

– менший час виробництва.

Spring Boot забезпечує розробникам Java гарну платформу для розроблення автономної Spring-програми, яку можна легко запустити. Можна розпочати роботу з мінімальними конфігураціями без необхідності встановлення всієї конфігурації Spring [38].

Spring Boot пропонує своїм розробникам такі переваги:

– розроблення власних Spring-програм;

– висока продуктивність;

– менший час для розроблення ПЗ.

Spring Boot розроблено з наступними цілями:

– для уникнення складної XML-конфігурації;

– для розроблення готових додатків Spring у більш простий спосіб;

– для скорочення часу розроблення і запуску програми самостійно;

– для простішого початку роботи з програмою.

Spring Boot автоматично налаштовує програму на основі залежностей, які додає розробник до проекту за допомогою анотації @EnableAutoConfiguration. Наприклад, якщо база даних MySQL знаходиться на шляху до класу, але ви не налаштували будь-яке підключення до бази даних, то Spring Boot автоматично налаштовує базу даних в пам'яті.

Точкою входу програми завантаження Spring є клас, що містить анотацію @SpringBootApplication і основний метод. Spring Boot автоматично сканує усі компоненти, включені в проект, за допомогою анотації @ComponentScan.

2.5. Висновки до розділу

У даному розділі розглянуто перелік інструментів, які можна використовувати для розроблення високонавантаженої платформи. Систематизувавши отриману інформацію, для подальшої роботи були визначені конкретні інструменти, які будуть використовуватись при розробленні програмної платформи для P2P інвестицій.

Для розроблення back end частини було обрано мову програмування Java, оскільки вона є архітектурно-нейтральною, безпечною і високопродуктивною, а також гарно справляється з багатопоточністю.

В якості back end фрейморку був вибраний Spring Boot, оскільки він автоматично налаштовує програму на основі анотацій, які додаються до проекту, і забезпечує легку модель для підтримки бізнес-вимог.

Для розробки front end частини було обрано мову програмування JavaScript, оскільки вона забезпечує повну інтеграцію з HTML і CSS, і роботу з анімацією.

В якості front end фреймворку був вибраний Angular, оскільки він використовує MVC шаблон архітектури і має свою ієрархічну структуру.

В якості СКБД було вибрано PostgreSQL, оскільки PostgreSQL – об'єктно-реляційна СУБД, підтримує великий список типів даних, має високу якість написання вихідного коду і підтримує самостійне створення нових типів.

В якості середовища для розроблення програмного забезпечення було обрано IntelliJ IDEA, оскільки ця IDE підтримує back end і front end фреймворки – можливість написання front end і back end частини в одній IDE, має продуманий інтерфейс та систему пошуку помилок і вважається найкращою IDE для Java.

3. РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА ПРОГРАМНИХ МОДУЛІВ

3.1. Загальний опис архітектури

Peer-To-Peer Lending платформа – це веб-орієнтована платформа, тому для розробки платформи була вибрана клієнт-серверна архітектура.

Клієнт-серверна архітектура – це структура розподілених додатків, яка розділяє завдання і навантаження між серверами і клієнтами.

Клієнт-серверна архітектура визнає такі типи компонентів:

– клієнт використовує сервіси, що надаються сервером;

– сервер надає інформацію для клієнтів;

– мережа – через мережу відбувається взаємодія клієнта і сервера.

Клієнти і сервери спілкуються через комп'ютерні мережі. Клієнт і сервер незалежні один від одного і знаходяться на окремому апаратному забезпеченні, але для роботи вони повинні залишатися в одній системі. На хості сервера може виконуватись одна або більше програм і вони діляться своїми ресурсами з клієнтами, які до них звертаються. Клієнт не ділиться своїми ресурсами, натомість він робить запити на сервер для отримання необхідної для нього інформації. Клієнт – це сторона, яка ініціює спілкування з сервером, а сервер чекає на вхідний запит. Клієнтом може бути будь-який пристрій, який має доступ до мережі інтернет [39]. На рис. 4 зображена клієнт-серверна архітектура.

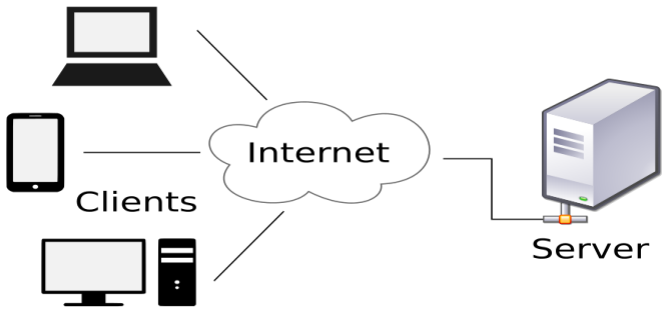


Рис. 7. Клієнт-серверна архітектура [39]

3.2. Опис модулів програмного забезпечення

У структурі розроблюваного програмного забезпечення можна чітко виділити такі модулі:

– модуль авторизації, реєстрації, особистий кабінет;

– модуль для верифікації інформації про клієнта;

– модуль зміни мови платформи;

– модуль серверного рендерингу;

– модуль портфоліо;

– модуль статистики;

– модуль ринку кредитів та корзина;

– модуль транзакцій;

– модуль для введення коштів на платформу;

– модуль для виведення коштів з платформи;

– модуль автоматичного створення інвестицій;

– модуль безпеки.

3.2.1. *Модуль авторизації та реєстрації*

Сутність користувача (рис. 8) складається з таких полів:

– agreed_to_conditions – користувач згоден з правилами користування платформою;

– account_verification – статус верифікації банківського рахунку користувача;

– birthday – день народження користувача;

– communication_language – комунікаційна мова користувача;

– country_code – телефонний код країни користувача;

– created_time – час створення користувача;

– email – електронна пошта користувача;

– first_name – ім’я користувача;

– gender – стать користувача;

– last_name – прізвище користувача;

– phone – мобільний телефон користувача;

– profilestatus – статус перевірки ідентифікаційних документів користувача;

– updatedtime – час обновлення персональної інформації користувача

– email_verified – верифікація електронної пошти користувача

– address – адреса користувача.

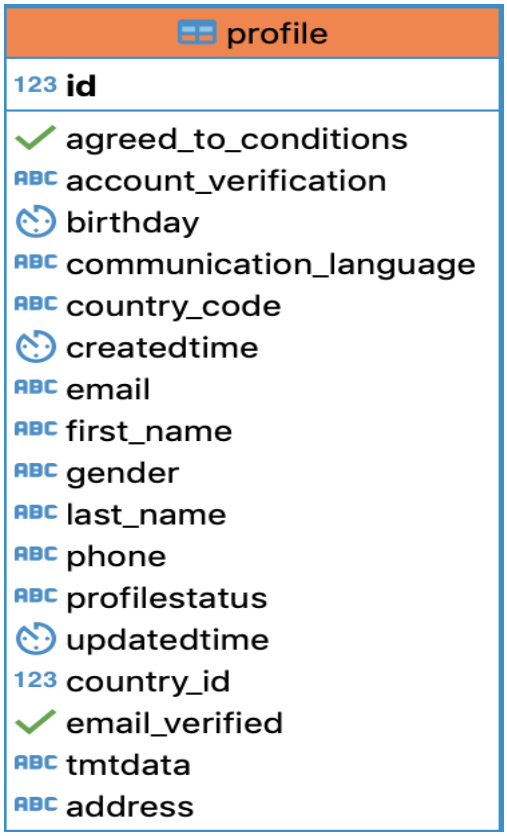


Рис. 8. Сутність користувача

Сутність користувача також пов’язана з іншою сутністю – user_credentials (рис. 9)

В таблиці user_credentials зберігається наступні поля:

– login – його логін, в якості логіну використовується email;

– login_type – тип логіну – за допомогою email;

– password – пароль користувача;

– profile_id – унікальний id користувача.

Сутність user_credentials використовується для авторизації користувача. Лише після успішної авторизації клієнт отримує свою персональну інформацію з таблиці profile.

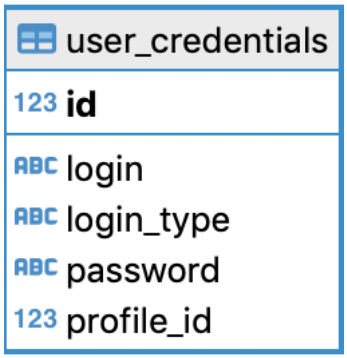


Рис. 9. Сутність user_credentials

При реєстрації користувач вказує свої дані – електронну пошту, ім’я, прізвище, країну, пароль і дає згоду з правилами користуванням платформою. При відправленні даних з клієнта на сервер, пароль шифрується публічним ключом, який знаходиться на стороні клієнта. Сервер отримує дані, розшифровує пароль, хешує його і після цього записує його в базу даних. Завдяки цьому забезпечується захист даних, адже навіть отримавши повний доступ до бази даних, не можливо дізнатись пароль користувача, а у випадку перехоплення запиту на передачу даних від клієнта до сервера, пароль зашифрований, і розшифрувати його зможе тільки той, хто знає приватний ключ. Після цього на його електронну пошту відправляється лист з підтвердженням його реєстрації. Користувач підтверджує свою електронну пошту і відразу отримує токен користувача. Токен користувача – це засіб авторизації для кожного запиту від клієнта до сервера, токен зберігається в браузері і має свій час дії – 12 годин. Після цього користувач потравляє в свій особистий кабінет. В своєму особистому кабінеті, користувач може додати про себе інформацію, таку як телефон, адрес, стать і мову комунікації.

При авторизації користувач повинен ввести свою електронну пошту та пароль. Клієнт шифрує пароль, способом описаним вище і дані відправляються на сервер. Сервер перевіряє відповідність отриманих даних з даними, які зберігаються в базі даних і після цього повертає токен авторизації, якщо дані валідні, або помилку якщо дані не валідні.

3.2.2. *Модуль для верифікації інформації про клієнта*

Для роботи з програмною платформою користувачу необхідно підтвердити свою особистість. Для цього йому необхідно надати документи – ID карта або паспорт.

Сутність документу (рис. 10) складається з таких полів:

– comment – коментарій до документа, використовується, якщо документ не валідний;
– create_date – дата відправки документа;
– document_type – тип документа (ID карта або паспорт);
– valid_from – дата видачі документа;
– valid_to – дата закінчення дії документа;
– path – власне сам документ у бінарному форматі;
– verification_status – статус документа;
– profile_id – id користувача, який надав документи.

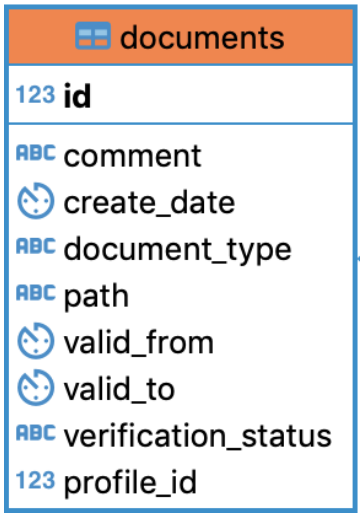


Рис. 10. Сутність документа

Користувач спочатку завантажує свої документи, у одному з форматів: '.jpeg', '.jpg', '.pdf'., '.png', розмір файлу не повинен перевищувати 5 мегабайт. Якщо користувач завантажує декілька документів, їх сумарний розмір не повинен перевищувати 20 мегабайт. Потім його статус верифікації змінюється на IN_VERIFICATION. Якщо документи валідні статус верифікації змінюється на VERIFY. Якщо документи не валідні, статус верифікації змінюється на NOT_VERIFY. У випадку якщо строк дії документів закінчився, статус верифікації користувача змінюється на NOT_VERIFY і йому необхідно надати документи заново.

3.2.3. *Модуль зміни мови*

Зміна мови платформи відбувається динамічно на стороні клієнта. Для цього був використаний модуль i18n – модуль перекладу з динамічним сховищем json [40]. В front end частині платформи містяться файли у форматі json. Формат JSON файлу дуже простий (ключ: переклад). Коли користувач змінює мову, front end просто змінює файл з мовою, і замінює контент відповідно до ключів. Для коректної роботи необхідно, щоб структура файлів з перекладами, а саме ключі були однаковими, а відрізнялись лише значеннями. Цей спосіб динамічної зміни мови надає великі переваги, так як пошукові системи, можуть бачити контент сайту відповідно до мови запиту.

3.2.4. *Модуль серверного рендерингу*

Архітектура front end частини розроблена за принципом SPA. Тобто контент динамічно завантажується при роботі. Незважаючи на всі переваги даного підходу, у нього є деякі недоліки:

– пошукові системі не індексують контент веб-додатку, оскільки для того, щоб пошуковий робот зміг проаналізувати контент, йому потрібно запустити веб-додаток, але робот не робить цього і в

результаті отримує порожню сторінку. Через це важко отримати найвищу позицію у пошукових системах.

– якщо у користувача слабке з’єднання у мережі, веб-додаток буде довго завантажуватися [41].

Для ліквідації даних недоліків впроваджують серверний рендеринг. При серверному рендерингу перед тим, як користувач почне взаємодію з додатком, на сервері генерується готова веб-сторінка, яку користувач просто отримує, перед тим, як він почне завантажувати усі інші частини, необхідні для роботи додатку. Ця веб-сторінка – це точна копія сторінки, на якій знаходиться користувач, за винятком того, що користувач не зможе взаємодіяти з нею, поки не завантажаться усі необхідні частини. Даний підхід дозволяє пошуковому роботу побачити зміст веб-сторінки, не запускаючи веб-додаток. У випадку слабкого з’єднання користувач відразу побачить контент сайту, а не буде спостерігати екран завантаження або пусту сторінку. Проте у даного підходу є незначний недолік – загальний час завантаження веб-додатку збільшується на декілька сотих секунди [42].

3.2.5. *Модуль портфоліо*

Портфоліо – місце, де користувач може отримати актуальний стан своїх інвестицій.

Сутність інвестиції (рис. 11) складається з таких полів:

– deal_type – тип інвестиції;

– activate_datetime – дата активації інвестиції;

– amount – сума грошей вкладених у інвестицію;

– cancelled_principal_amount – сума на яку був здійснений достроковий вихід з інвестиції;

– closure_date – дата закінчення інвестиції;

– create_datetime – дата створення інвестиції;

– current_day – день на якому знаходиться інвестиція;

– penalty_amount – штраф за достроковий вихід з інвестиції;

– planned_closure_date – дата планового закриття інвестиції;

– status – статус інвестиції;

– term – період на який розміщується інвестиція;

– max_profitability – максимальна дохідність з інвестиції;

– min_profitability – мінімальна дохідність з інвестиції;

– profile_id – id користувача, який розмістив інвестицію;

– offer_id – id інвестиції.

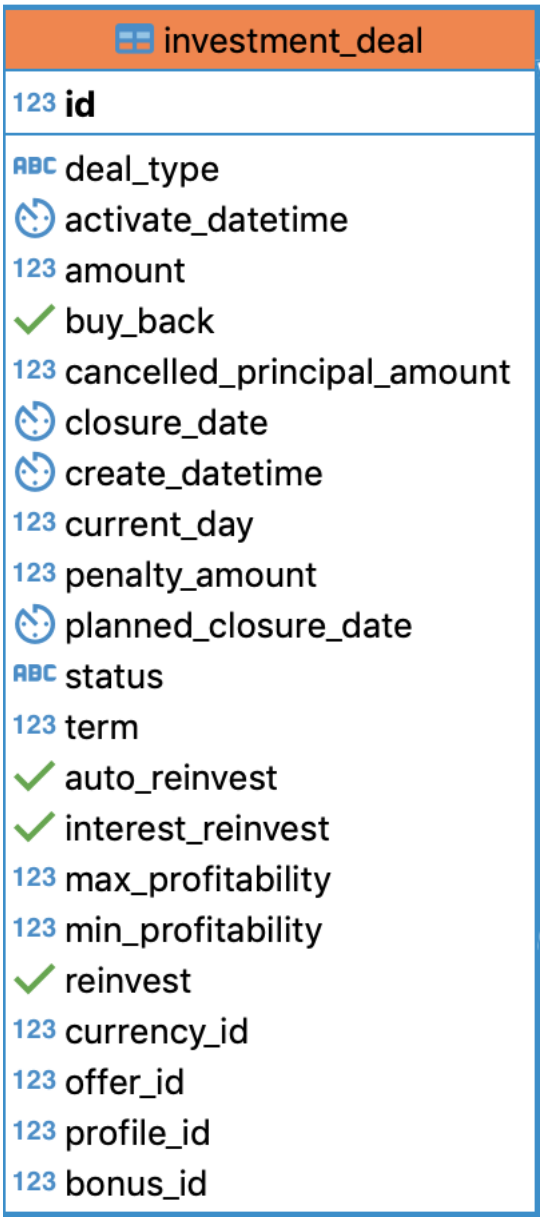


Рис. 11. Сутність інвестиції

В портфоліо користувач може продивитись загальну інформацію про інвестицію. Інвестиції бувають двох видів: зроблені автоматично за допомогою системи, або ж користувач сам вибрав кредити, в які він захотів інвестувати.

У випадку, якщо користувач створює інвестицію автоматично за допомогою системи, то йому надаються наступні додаткові функціональності:

– можливість створення інвестиції без коштів. Користувач має змогу створити інвестицію без вкладання грошей в неї – ознайомлення користувача з принципом роботи платформи. В такому випадку інвестиція буде створена з відкладеною датою активації. Користувачу надається 24 години для того, щоб він вніс кошти на платформу. В разі внесення достатньої суми на платформу для розміщення інвестиції, інвестиція автоматично розміщується. У випадку невнесення коштів через 24 години інвестиція автоматично відміняється;
– можливість виходу з інвестиції без втрати коштів. Після активації інвестиції користувачу надається 24 години, протягом яких він зможе відмінити інвестицію, не сплачуючи комісію за послуги платформи. В такому випадку кошти повертаються на рахунок користувача в платформі без комісії;
– можливість виходу з інвестиції. Якщо користувач захоче відмінити інвестицію після 24 годин з моменту активації інвестиції, він повинен буде сплатити комісію – 1% від суми інвестиції на користь платформи, решта коштів буде повернута йому на його рахунок в платформі.

Якщо користувач власноруч створював інвестицію, то у портфоліо він зможе тільки переглядати інформацію про актуальний стан його інвестиції.

3.2.6. *Модуль статистики*

В статистиці користувач має змогу переглядати детальну інформацію про інвестицію.

Сутність кредиту (рис. 12) складається з таких полів:

– calculated_yield – очікуваний прибуток;

– loan_id – id кредиту;

– overdue – прострочка кредиту;

– purpose – ціль кредиту;

– risk – ризик кредиту;

– risk_type – класифікація ризику кредиту;

– status – статус кредиту;

– country_id – id країни, в якій видано кредит;

– currency_id – id валюти, в якій видано кредит.

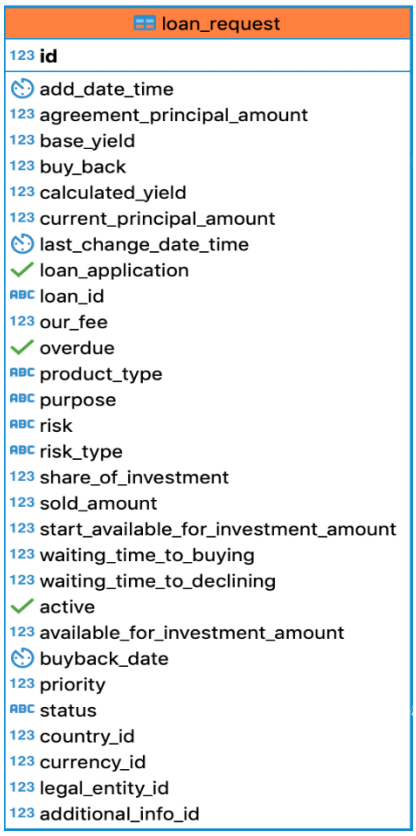


Рис. 12. Сутність кредиту

Також в сутність кредиту пов’язана з сутністю додаткової інформації по кредиту. На рис. 13 Зображена сутність додаткової інформації про кредит.

Сутність додаткової інформації про кредит складається з таких полів:

– client_age – вік людини, якій видано кредит;

– client_gender – стать людини, якій видано кредит;

– loan_start_date – дата видачі кредиту;

– redemption_number – кількість платежів, які залишились до повного погашення боргу по кредиту;

– next_redemprion_amount – сума наступного погашення по кредиту;

– next_redemprion_date – дата наступного погашення по кредиту;

– overdue_start_date – дата початку прострочення кредиту;

– planned_closure_date – планова дата виплати кредиту;

– principal_overdue_amount – сума прострочки кредиту.

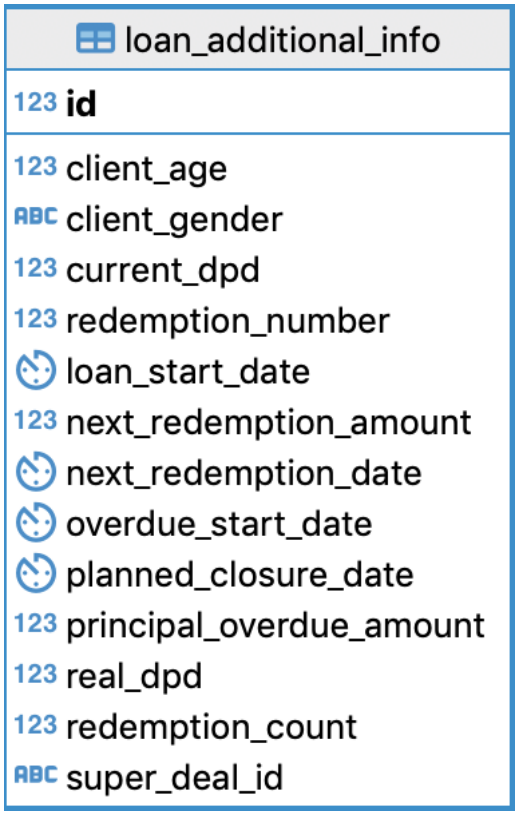


Рис. 13. Додаткова інформація про кредит

У статистиці користувач може детально розглянути кожен кредит, в який він інвестував, дізнатись актуальний стан кожного кредиту – чи знаходиться кредит у прострочці, скільки платежів залишилось виплатити позичальнику, дату наступного платежу, ризик кредиту, очікуваний прибуток від даного кредиту, країну, в якій видано кредит, валюту, в якій видано кредит, скільки було вкладено його коштів в даний кредит, класифікацію кредиту, а також кількість кредитів, які входять в його інвестицію.

3.2.7. *Модуль ринку кредитів та корзина*

Ринок кредитів – місце, де користувач може створити інвестицію власноруч з доступних кредитів на платформі. Ринок складається з сутності кредиту (рис. 12), сутності додаткової інформації про кредит (рис. 13), сутності корзини (рис. 14), в якій знаходять кредити, в які хоче інвестувати користувач, сутності кредиту, який знаходиться в корзині (рис. 15) і сутності резервації коштів в кредиті (рис. 16).

Сутність корзини складається з таких полів:

– delete_date_time – час, в який корзина буде анульована;

– currency_id – id валюти корзини;

– profile_id – id користувача, який створив корзину.

Рис. 14. Сутність корзини

Сутність кредиту, який знаходиться в корзині, складається з таких полів:

– amount – сума, яку користувач вклав в даний кредит;
– amount_in_loan_currency – сума, яку користувач вклав в даний кредит, в перерахунку на валюту, в якій був виданий кредит;
– available – чи доступний кредит для інвестування;
– locked – чи сума, яку захотів вкласти користувач в кредит, заблокована за ним;
– shopping_card_id – id корзини, в яку входить кредит;
– loan_request_id – id резервації коштів в кредиті – для зв'язку сутності кредиту, який знаходиться в корзині з сутністю резервації коштів у кредиті;

Рис. 15. Сутність кредиту, який знаходиться в корзині

Сутність резервації коштів в кредиті складається з таких полів:

– amount – сума, на яку відбулася резервація;
– loan_request_id – id резервації коштів в кредиті;

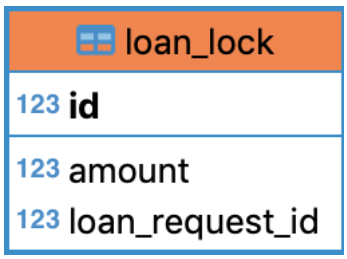


Рис. 16. Сутність резервації коштів в кредиті

За допомогою ринку кредитів користувач має змогу сам створити інвестицію і вибрати лише ті кредити, в які він хоче інвестувати свої кошти.

Для зручності користування ринком кредитів реалізовані фільтри:

– валюта;

– кредитна організація, яка видала кредит;

– рейтинг ризику кредиту;

– країна, в якій видано кредит;

– дата видачі кредиту;

– дата закінчення кредиту.

Так як доступних кредитів багато, то вони завантажуються частинами – по 20, 40, або 60 кредитів на одній сторінці, в залежності від того, скільки користувач вказав у налаштуванні.

Користувач вибирає кредит, вводить суму, яку він хоче інвестувати в даний кредит і після цього кредит потрапляє в корзину. В випадку, якщо коштів достатньо для здійснення купівлі даного кредиту, сума, яку користувач хоче інвестувати в кредит, резервується за даним користувачем протягом однієї години. В момент резервування у інших користувачів динамічно оновлюється інформація по даному кредиту – для того, щоб інші користувачі не змогли інвестувати в кредит суму більшу, ніж та, яка є доступною для інвестування в даному кредиті. Динамічне оновлення реалізовано за допомогою websockets. Websockets дозволяють клієнту динамічно отримувати інформацію про зміну даних на сервері. Після цього користувач може далі додавати кредити в корзину протягом години, і у випадку, якщо коштів вистачає, кредити будуть резервуватися. Через

годину корзина автоматично видаляється і резерв по кредитам користувача відміняється. Якщо в корзину потрапляє кредит, на якого в користувача не вистачає коштів, резервування коштів у кредиті не відбувається і можливість покупки корзини зникає і користувачу потрібно або редагувати суму, яку він хоче інвестувати в даний кредит або видалити цей кредит з корзини, можливість купівлі корзини зникає до вирішення користувачем проблеми з доступними коштами. Після того, як користувач остаточно сформував корзину, йому потрібно підтвердити, що він бажає здійснити інвестування в дані кредити і він отримує загальну інформацію про корзину – загальну суму і валюту корзини і кількість кредитів, які знаходяться в корзині і йому необхідно ввести свій пароль для підтвердження. Після цього користувач зможе переглянути свою інвестицію у портфоліо.

3.2.8. *Модуль транзакцій*

Користувач може продивитись усі транзакції, які були виконані в системі з його коштами. Користувачу необхідно ввести дату початку періоду і дату кінця періоду, за який він хоче переглянути транзакції, валюту, по якій відбудеться фільтрація транзакцій, тип транзакцій, і інвестицію по якій він хоче переглянути транзакції. Також після формування транзакцій користувач може завантажити їх у форматі excel.

Сутність транзакції (рис. 17) складається з таких полів:

– amount – сума транзакції;

– purpose – тип транзакції;

– from_id – id рахунку, з якого виходить транзакція;

– to_id – id рахунку, на який приходить транзакція;

– operation_id – id операції;

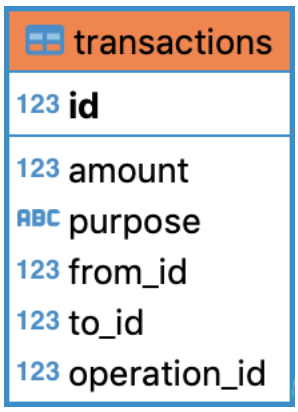


Рис. 17. Сутність транзакції

3.2.9. *Модуль для введення коштів на платформу*

Модуль для введення коштів на платформу складається з банківських реквізитів для кожної валюти, на які користувачу необхідно перевести кошти і вказати в цілі призначення платежу ID користувача в системі.

Після отримання коштів на банківський рахунок, кошти вводять в платформу в тому ж обсязі, в якому вони надійшли на банківський рахунок платформи. Якщо користувач переслав кошти на банківський рахунок, але не пройшов верифікацію, кошти будуть повернені йому назад. Банківський рахунок, з якого прийшли кошти буде приписаний до користувача і вивести кошти з платформи він зможе тільки на цей же рахунок.

3.2.10. *Модуль для виведення коштів з платформи*

Сутність запиту на виведення коштів (рис. 18) скаладається з таких полів:

– amount – сума, яку хоче вивести користувач;

– error_message – повідомлення про можливу помилку;

– status – статус обробки запиту;

– profile_id – id клієнта, який зробит запит;

– bank_account_id – id банківського акаунта користувача;

– currency_id – id валюти, в якій був запит на виведення коштів;

Рис. 18. Сутність запиту на виведення коштів

Для виведення коштів з платформи користувачу необхідно вказати суму, яку він хоче вивести з платформи та валюту і ввести свій пароль. Користувач не може використовувати одну валюту в ході роботи з платформою, а вивести в іншій. Після цього кошти переводяться на банківський рахунок користувача, з якого раніше були переведені кошти.

3.2.11. *Модуль автоматичного створення інвестицій*

Користувач може створити інвестицію автоматично, вказавши суму, валюту і термін дії інвестиції. Термін дії інвестиції може бути 1, 3, 6, 12 місяців. Потім йому необхідно вибрати одну з двох стратегій інвестування:

– Безпечне інвестування – процента ставка гарантується;

– Ризикове інвестування – процентна ставка є плаваючою і в результаті користувач може отримати або більший прибуток ніж при безпечному інвестування або менший.

Після цього користувач ознайомлюється з детальною інформацією по цій інвестиції, і для створення інвестиції йому необхідно ввести пароль. Відбувається створення інвестиції і користувач може переглянути інформацію про інвестицію у портфоліо.

3.2.12. *Модуль безпеки*

Для передачі даних від клієнта до сервера використовується протокол HTTPS. Усі основні дані, які передаються через HTTPS, шифруються: URL-запити, включаючи шлях та назву ресурсу (сторінки), параметри запиту, заголовки та cookie, які містять ідентифікаційні дані про користувача (токен авторизації). Не шифруються: назва або адреса хоста (веб-сайту) та порт, оскільки вони використовуються транспортним протоколом TCP/IP для встановлення з'єднання [43].

Також додатково відбувається шифрування паролю при передачі його від клієнта до сервера, потім пароль хешується в СКБД, також відбувається хешування приватних даних користувача в СКБД.

3.3. Висновки до розділу

У цьому розділі було представлено загальний опис архітектури та основних програмних модулів розробленої P2P Lending платформи.

Створено модуль авторизації, модуль реєстрації, модуль зміни мови, модуль портфоліо, модуль статистики, модуль ринку кредитів, модуль корзини, модуль автоматичного інвестування, модулі для введення та виведення коштів, модуль транзакцій, модуль для верифікації інформації про клієнта, модуль серверного рендерингу, модуль безпеки.

Також було описано взаємозв’язок програмних модулів та наведено ключові сутності для роботи з кожним модулем.

Для розроблення програмного забезпечення було використано такі технології, як серверний рендеринг, i18n, фреймворки Spring Boot та Angular для забезпечення оптимальної продуктивності роботи P2P Lending платформи.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості тестування програмного забезпечення

У даному розділі буде проаналізовано методи тестування програмного забезпечення.

4.1.1. *Методи тестування програмного забезпечення*

Тестування програмного забезпечення – процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином [44].

Може оцінюватись:

– перевірка описаних функціональних можливостей;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з програмним забезпеченням та операційними системами;

– тестування верстки;

– тестування кросбраузерності;

– usability тестування;

– тестування продуктивності.

4.1.2. *Димове тестування*

Димове тестування ПЗ означає мінімальній набір тестів на явні помилки. Це швидкі тести, які перевірять функціональність програми. Програму, яка не справилась з димовим тестуванням, немає сенсу тестувати далі [45].

На цьому етапі тестування необхідно перевірити всю описану функціональні можливості модулів. Для цього виконуємо наступні кроки:

– реєструємось в системі (рис. 19);

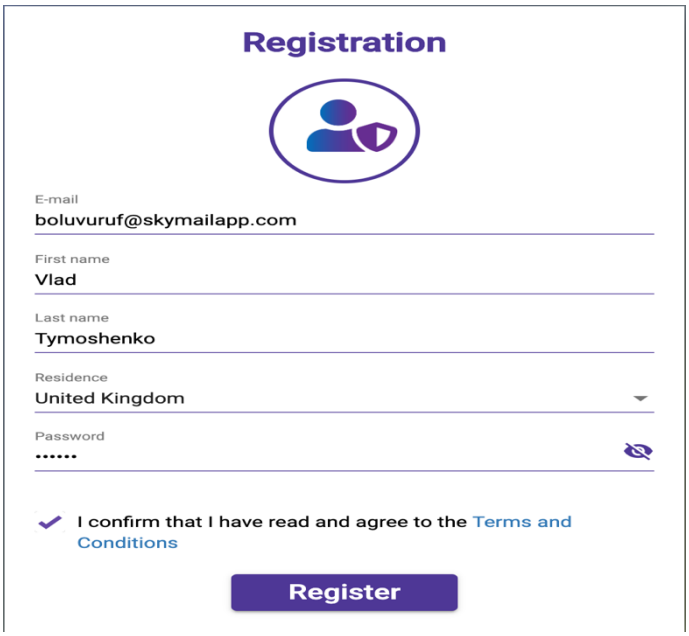


Рис. 19. Вікно реєстрації

– завантажуємо ідентифікаційні документи (рис. 20);

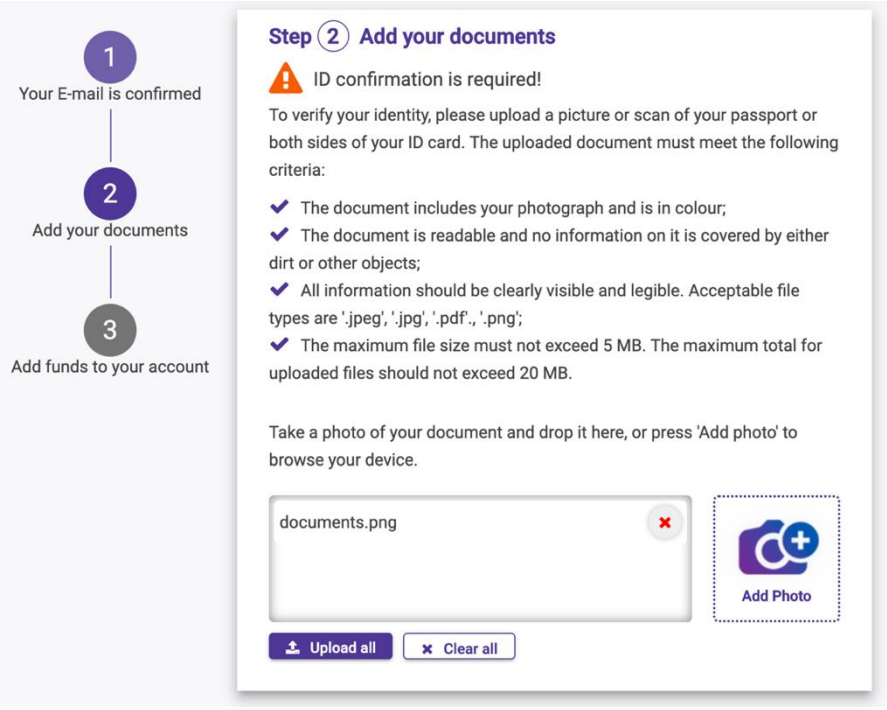


Рис. 20. Завантаження ідентифікаційних документів

– вводимо кошти на платформу (рис. 21);

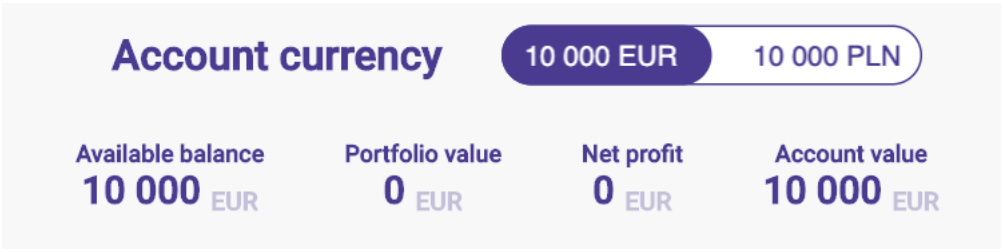


Рис. 21. Введення коштів на платформу

– створюємо інвестицію автоматично, з стратегією безпечного інвестування (рис. 22);

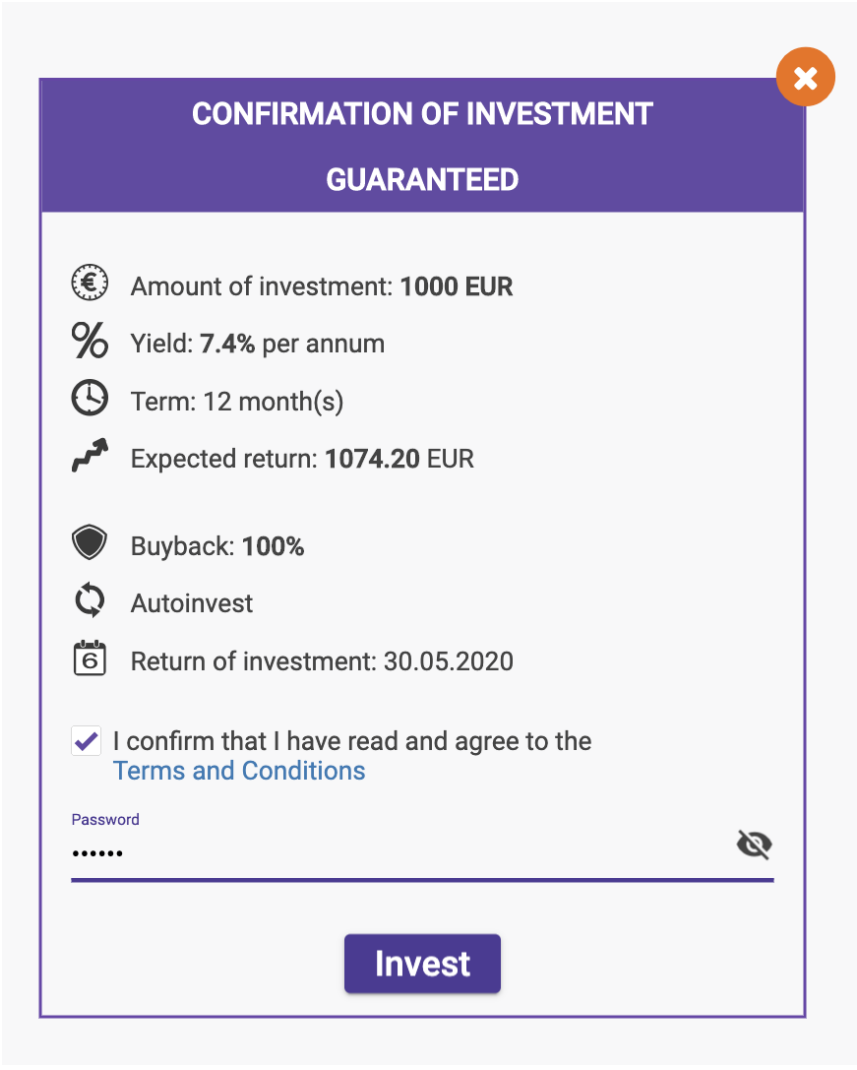


Рис. 22. Створення інвестиції автоматично

– переглядаємо створену інвестицію в портфоліо (рис. 23);

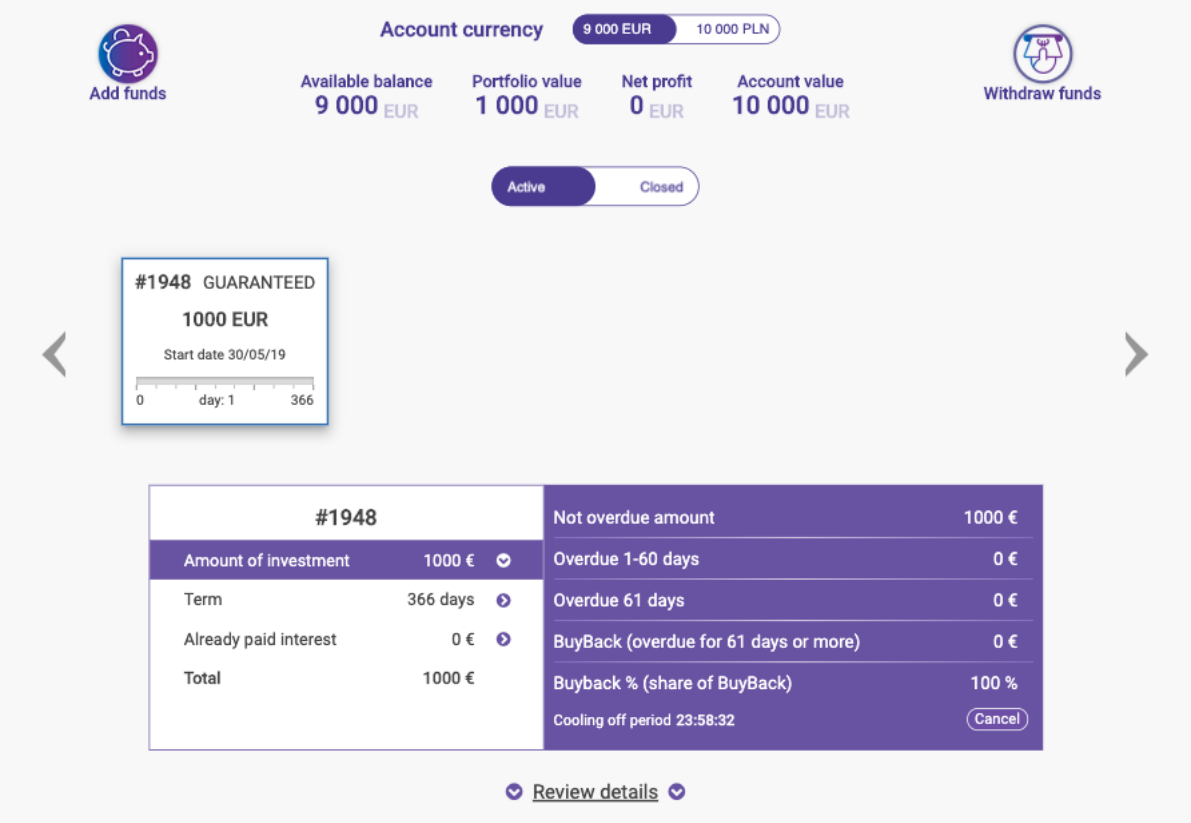
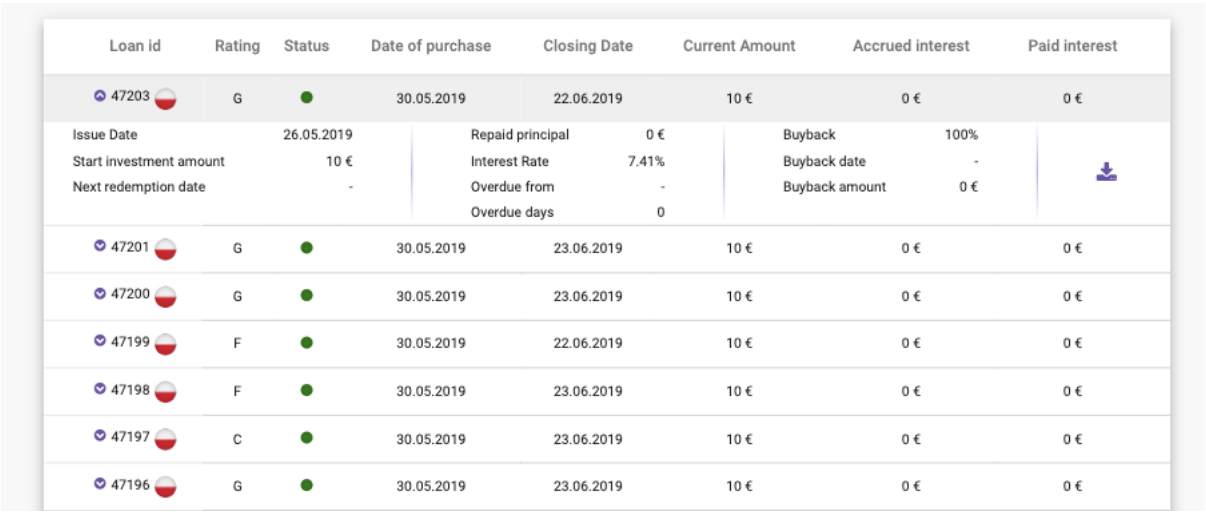


Рис. 23. Портфоліо з створеною інвестицією

– переглядаємо статистику створеної інвестиції (рис. 24);



Loan id	Rating	Status	Date of purchase	Closing Date	Current Amount	Accrued interest	Paid interest
47203	G	●	30.05.2019	22.06.2019	10 €	0 €	0 €
47201	G	●	30.05.2019	23.06.2019	10 €	0 €	0 €
47200	G	●	30.05.2019	23.06.2019	10 €	0 €	0 €
47199	F	●	30.05.2019	22.06.2019	10 €	0 €	0 €
47198	F	●	30.05.2019	23.06.2019	10 €	0 €	0 €
47197	C	●	30.05.2019	23.06.2019	10 €	0 €	0 €
47196	G	●	30.05.2019	23.06.2019	10 €	0 €	0 €

Issue Date 26.05.2019
Start investment amount 10 €
Next redemption date -
Repaid principal 0 €
Interest Rate 7.41%
Overdue from -
Overdue days 0
Buyback 100%
Buyback date -
Buyback amount 0 €

Рис. 24. Статистика інвестиції

– перевіряємо транзакції, які відбулись по створеній інвестиції (рис. 25);

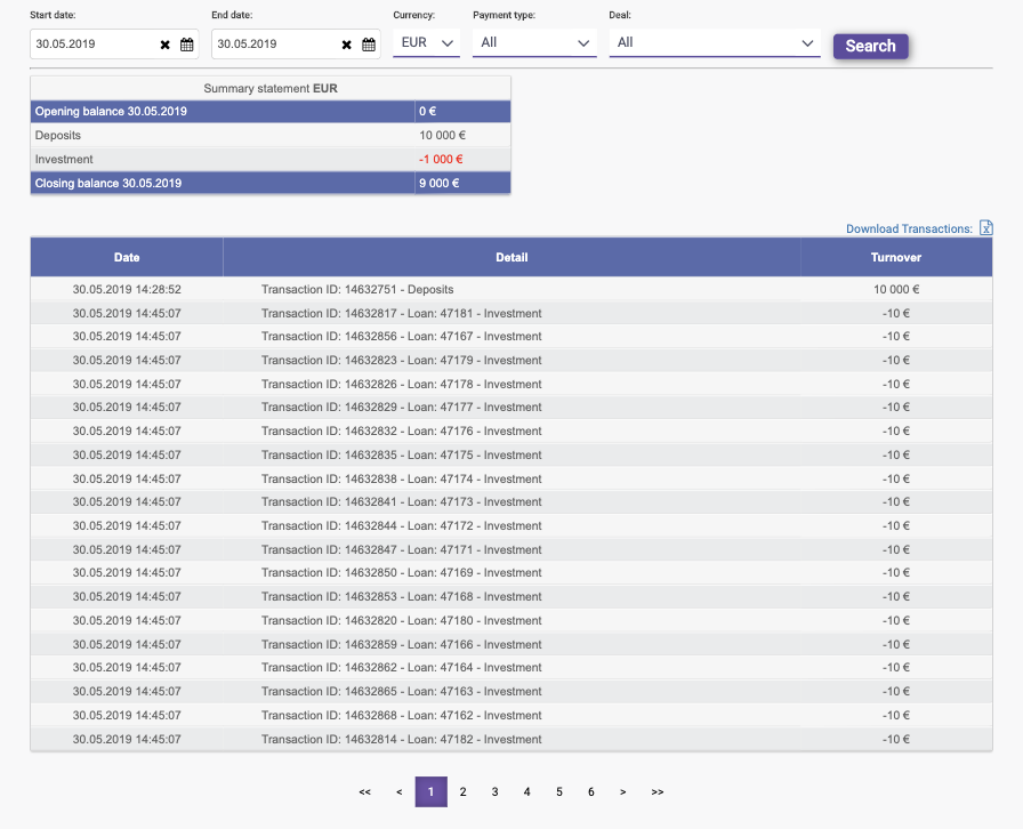


Рис. 25. Транзакції по інвестиції

– створюємо інвестицію, користуючись ринком кредитів (рис. 26);

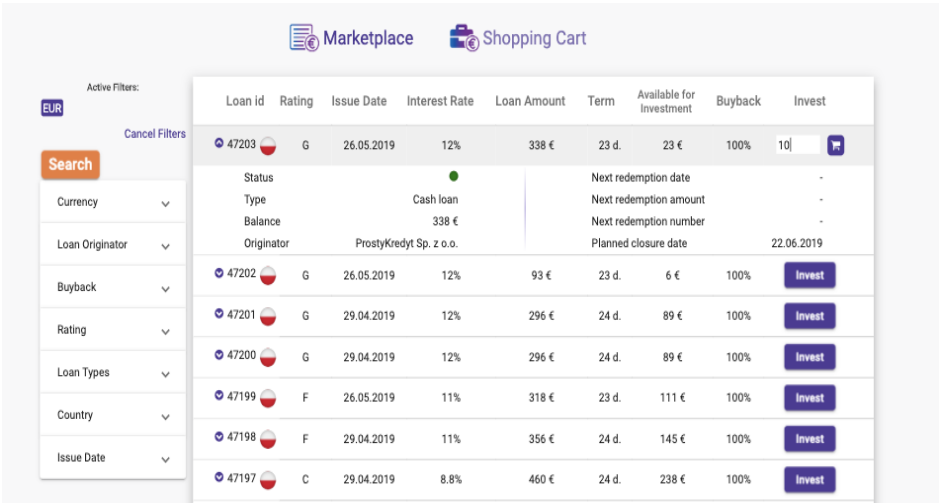


Рис. 26. Створення інвестиції за допомогою ринку кредитів

– переходимо в корзину і підтверджуємо покупку кредиту (рис. 27);

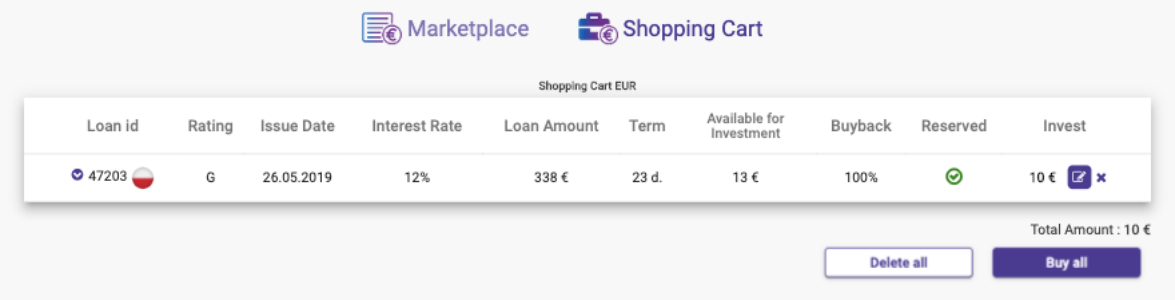


Рис. 27. Корзина

– підтверджуємо покупку кредиту через ринок кредитів (рис. 28);

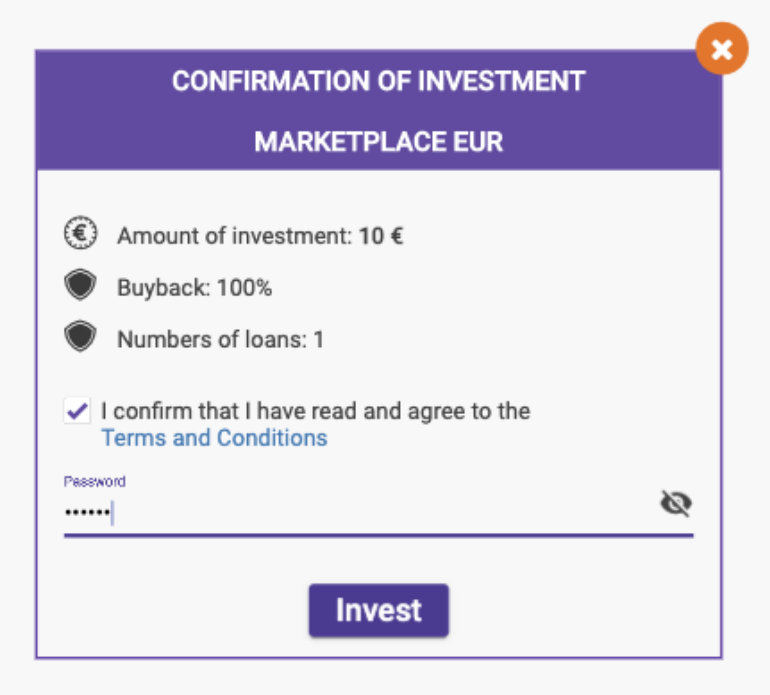


Рис. 28. Підтвердження покупки кредиту через ринок кредитів

– перевіряємо портфоліо (рис. 29);

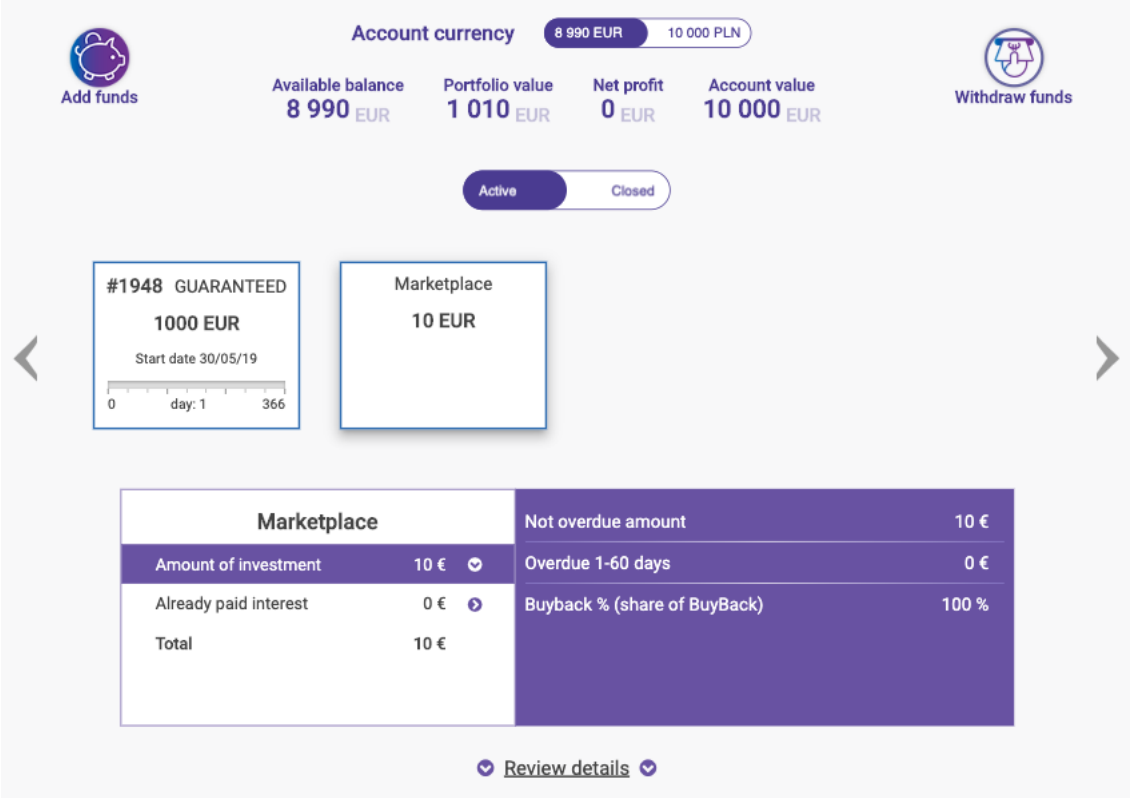


Рис. 29. Перевірка портфоліо

– виведення коштів з платформи (рис. 30);

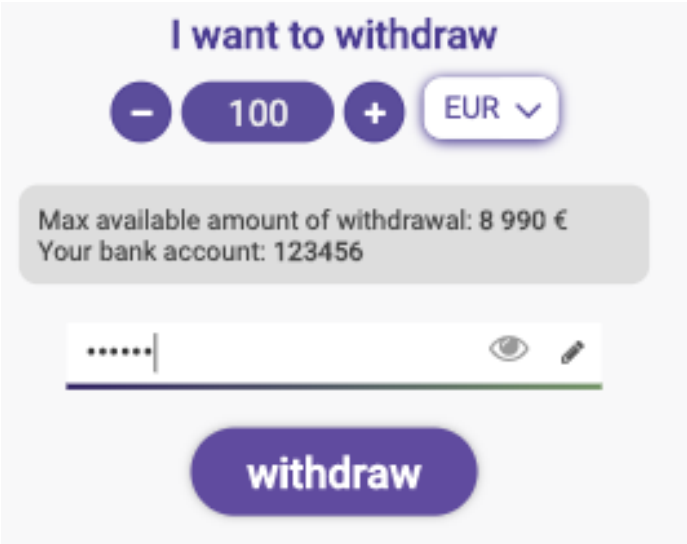


Рис. 30. Виведення коштів з платформи

– перевірка балансу після виведення коштів з платформи (рис. 31).

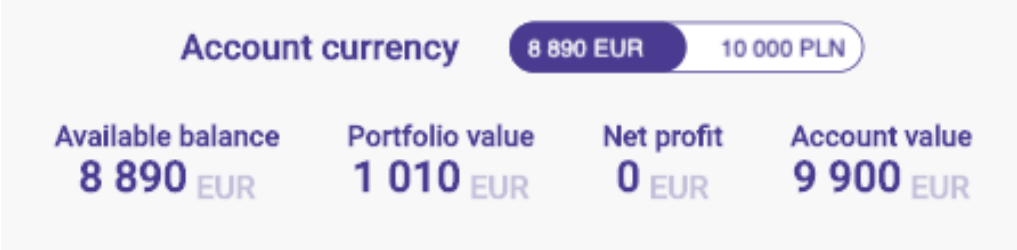


Рис. 31. Перевірка балансу після виведення коштів з платформи

Димове тестування основної функціональності розробленої програмної платформи пройшло успішно.

4.1.3. *Тестування продуктивності*

Для тестування продуктивності платформи використаємо розширення Chrome Lighthouse.

Lighthouse – це автоматизований інструмент з відкритим вихідним кодом для поліпшення якості веб-сторінок. Його можна запускати на будь-якій веб-сторінці, відкритій або такій, яка вимагає автентифікації. Він має аудит продуктивності, доступності, SEO тощо [46].

Тестування буде відбуватись для двох випадків:

– найкращий випадок – у якості девайсу вибираємо комп’ютер, без симуляції 3G інтернету і без уповільнення процесора. (рис. 32);

– найгірший випадок – в якості девайсу вибираємо мобільний телефон, симуляція 3G інтернету, уповільнення процесора в 4 рази. (рис. 33).

Ці випадки тестування повністю покривають тестування продуктивності, адже вони показують результати тестування в найгіршому і найкращому випадку. Отже, середній результат продуктивності буде знаходитись в діапазоні між найгіршим і найкращим випадком.

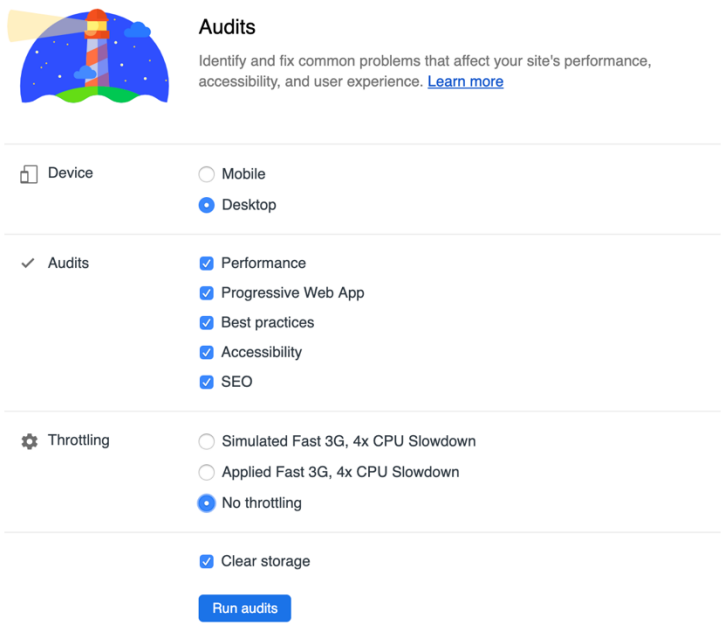


Рис. 32. Перевірка продуктивності платформи (вхідні дані – комп'ютер, без симуляції 3G інтернету і без уповільнення процесора)

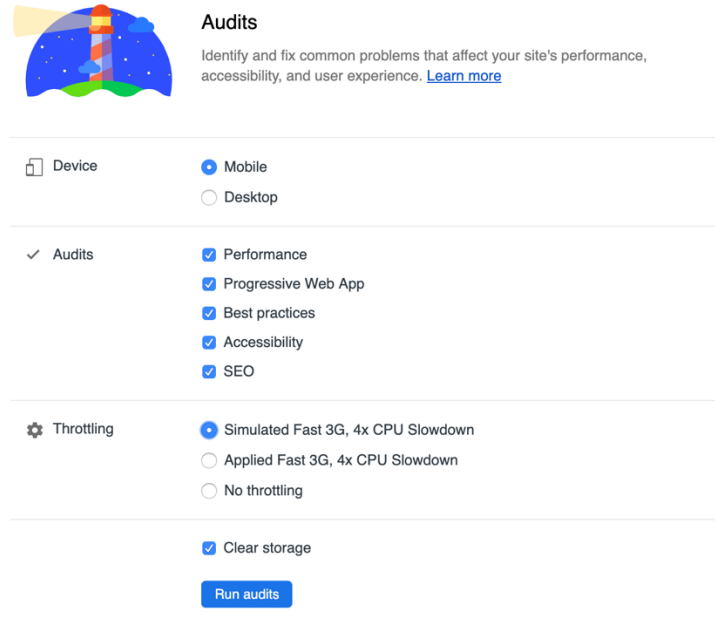


Рис. 33. Перевірка продуктивності платформи (вхідні дані – мобільний телефон, симуляція 3G інтернету, уповільнення процесора в 4 рази)

У найкращому випадку ми отримали результат продуктивності в 100%. (рис. 34)

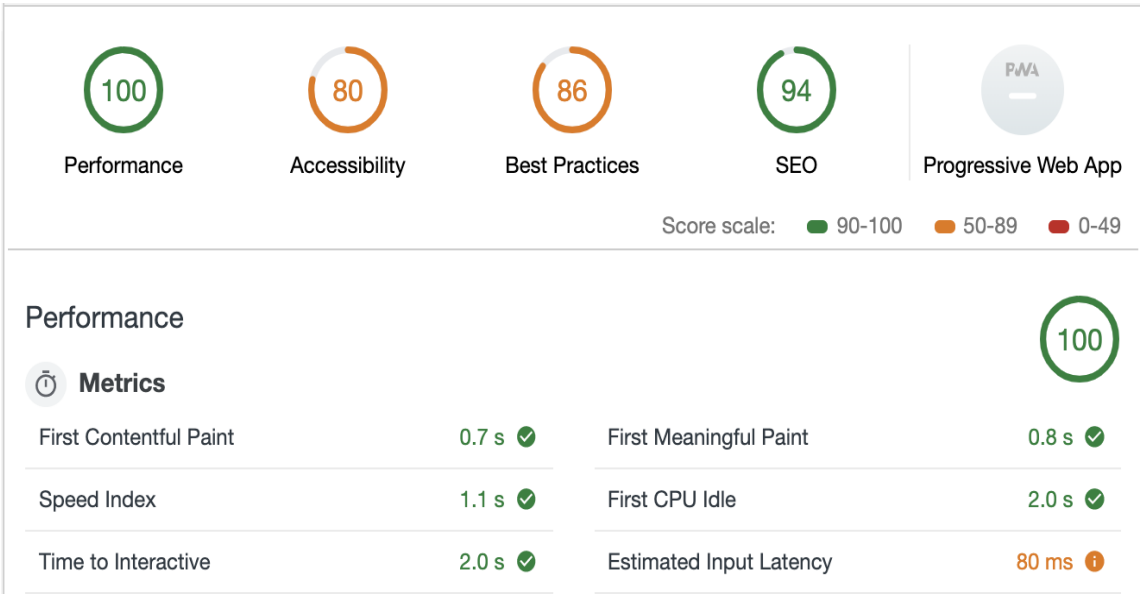


Рис. 34. Результат перевірки продуктивності платформи для вхідних даних – комп'ютер, без симуляції 3G інтернету і без уповільнення процесора

У найгіршому випадку ми отримали результат продуктивності в 67%. (рис.35)

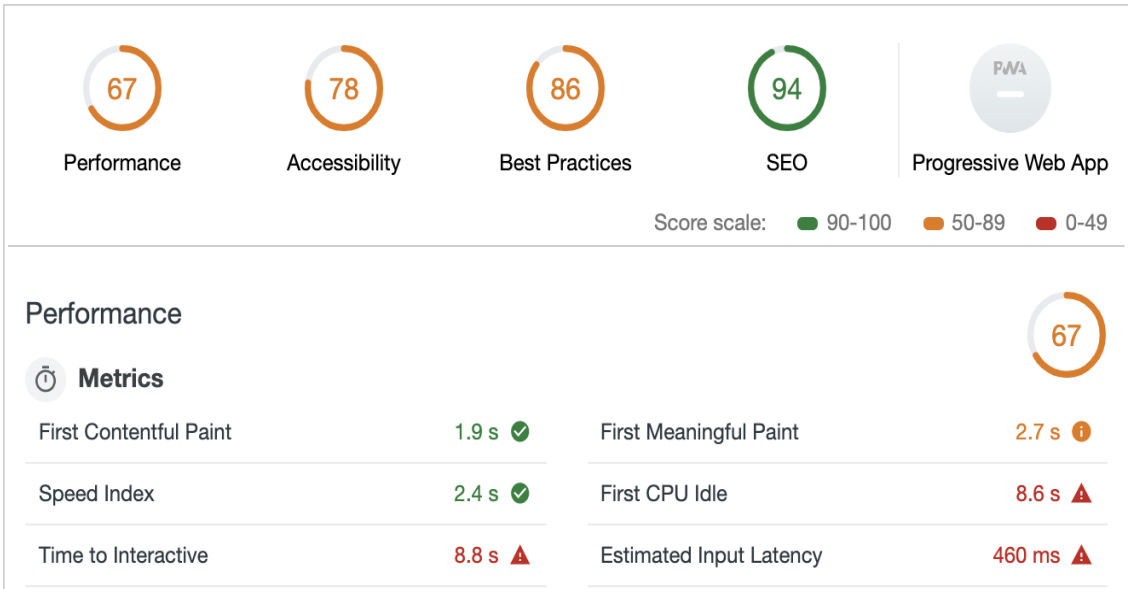


Рис. 35. Результат перевірки продуктивності платформи для вхідних даних – мобільний телефон, симуляція 3G інтернету, уповільнення процесора в 4 рази

Отже, середня продуктивність запропонованого ПЗ буде знаходитись у діапазоні 67 – 100% (табл.4).

Таблиця 4

Порівняння продуктивності розробленого ПЗ з існуючими рішеннями

Назва	Запропоноване програмне забезпечення	Mintos	PeerBerry	Bondora
Найкращий випадок	100%	98%	99%	100%
Найгірший випадок	67%	43%	10%	60%

4.2. Порівняння розробки з існуючими аналогами

Під час підготовки до розроблення програмної платформи було досліджено існуючі аналоги, виявлені їх основні переваги та недоліки, вони детально описані в розділі 1.

Розроблена платформа має такі переваги як:

– захист персональних даних;

– продуману систему диверсифікації ризику для інвестора;

– наявність декількох мов.

При реалізації платформи були усунуті основні недоліки існуючих аналогів, а саме:

– платформа надає всю необхідну інформацію для інвестування в позики і оновлює її в режимі реального часу;

– використовується SPA - архітектура у клієнтській частині;

– має більшу продуктивність клієнтської частини в порівнянні з аналогами;

– надається можливість вибирати стратегію інвестування автоматично, або ж створювати інвестицію власноруч.

4.3. Рекомендації для подальшого вдосконалення

Подальшими напрямками удосконалення програмного забезпечення для розробленої P2P lending платформи можна вважати наступні:

– можливість додавання декількох мов на платформу;
– підвищення продуктивності роботи клієнтської частини на мобільних пристроях шляхом перетворення розробленої P2P lending платформи в Progressive Web App – гібриду веб та мобільного додатку;
– додавання можливості візуалізації даних для більш наочного зображення руху коштів по інвестиціям, або ж візуалізація очікуваного прибутку;
– підвищення індексу SEO, наразі він 94%, враховуючи рекомендації Lighthouse;
– створити мобільні додатки для Android та IOS, для більш зручного використання розробленого програмного забезпечення.

4.4. Висновки до розділу

У цьому розділі було розглянуто тестування розробленого програмного забезпечення для P2P Lending платформи.

Розглянуто основні види тестування програмного забезпечення, проведено димове тестування.

Порівняно продуктивність клієнтської частини розробленого програмного забезпечення з існуючим аналогами засобами Lighthouse. У ході тестування не було виявлено помилок. Розроблене програмне забезпечення показало кращі результати при тестування продуктивності в Lighthouse, а також покрило усі недоліки аналогів і при цьому зберегло їх переваги.

Запропоновано рекомендації для подальшого вдосконалення програмного забезпечення для P2P Lending платформи.

ВИСНОВКИ

Головною метою дипломного проекту було створення Peer-To-Peer Lending платформи для здійснення P2P - інвестицій коштів у мережі Інтернет.

У першому розділі проаналізовано переваги та недоліки існуючих Peer-To-Peer Lending платформ для здійсненння онлайн – інвестування коштів.

На основі цього аналізу в другому розділі для реалізації back end частини обрано мову програмування Java, фреймворк Spring boot, для реалізації front end частини було обрано мову програмування JavaScript, фреймворк Angular. Для використаних мов програмування використано бібліотеки, які значно спрощують розроблення програмного забезпечення. Використано додаткові технології, такі як i18n – для перекладу контенту платформи на інші мови, Angular universal – для забезпечення серверного рендерингу, RxJS – для роботи з асинхронністю.

У третьому розділі описано ключові модулі системи та їх зв’язок один з одним, наведено пов’язані з ними сутності бази даних. Реалізовано функції створення інвестиції автоматично та власноруч; функції перегляду створених інвестицій у портфоліо користувача; додавання власної персональної інформації в особистому кабінеті; виведення та введення коштів; перегляд транзакцій по кожній з інвестицій; можливість зміни валюти інвестування. Застосовано серверний рендеринг. Реалізовано динамічне оновлення даних.

У четвертому розділі проведено тестування розробленого програмного забезпечення та виконано порівняння продуктивності з існуючими аналогами. Завдяки гнучкій архітектурі проект можна розвивати у майбутньому. Одним з майбутніх розширень є розроблення мобільних додатків для операційних систем IOS та Android та покращення індексу SEO.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. P2P-інвестування [Електронний ресурс]. – Режим доступу: https://is.gd/urW6BT. Дата доступу: березень 2019. Назва з екрану.
2. Робота P2P-платформи [Електронний ресурс]. – Режим доступу: https://is.gd/YmeppB. Дата доступу: березень 2019. Назва з екрану.
3. P2P-платформи [Електронний ресурс]. – Режим доступу: http://bit.do/eTZSn. Дата доступу: березень 2019. Назва з екрану.
4. Розкриття інформації P2P-платформами [Електронний ресурс]. – Режим доступу: https://is.gd/R2ehEn. Дата доступу: березень 2019. Назва з екрану.
5. Запобігання легалізації доходів від злочинної діяльності та фінансування тероризму платформами [Електронний ресурс]. – Режим доступу: https://is.gd/pQ2Dbo. Дата доступу: березень 2019. Назва з екрану.
6. Диверсифікація ризику [Електронний ресурс]. – Режим доступу: https://is.gd/HQk6sF. Дата доступу: березень 2019. Назва з екрану.
7. Wall Fargo Bank [Електронний ресурс]. – Режим доступу: https://is.gd/c49ui7. Дата доступу: березень 2019. Назва з екрану.
8. Prosper Loans [Електронний ресурс]. – Режим доступу: http://bit.do/eTZVf. Дата доступу: березень 2019. Назва з екрану.
9. PricewaterhouseCoopers [Електронний ресурс]. – Режим доступу: https://is.gd/IZF9ER. Дата доступу: березень 2019. Назва з екрану.
10. Foundation Capital [Електронний ресурс]. – Режим доступу: https://is.gd/YtFxW4. Дата доступу: березень 2019. Назва з екрану.
11. Mintos [Електронний ресурс]. – Режим доступу: https://is.gd/3DxOgF. Дата доступу: березень 2019. Назва з екрану.
12. PeerBerry [Електронний ресурс]. – Режим доступу: https://is.gd/tD2qR1. Дата доступу: березень 2019. Назва з екрану.

13. Bondora [Електронний ресурс]. – Режим доступу: https://is.gd/RGpjnh. Дата доступу: березень 2019. Назва з екрану.
14. Історія Java [Електронний ресурс]. – Режим доступу: https://is.gd/fxNDOU. Дата доступу: квітень 2019. Назва з екрану.
15. Java [Електронний ресурс]. – Режим доступу: https://is.gd/7V7QgZ. Дата доступу: квітень 2019. Назва з екрану.
16. Переваги Java [Електронний ресурс]. – Режим доступу: https://is.gd/6RZeOx. Дата доступу: квітень 2019. Назва з екрану.
17. Недоліки Java [Електронний ресурс]. – Режим доступу: https://is.gd/Oplqzw. Дата доступу: квітень 2019. Назва з екрану.
18. Python [Електронний ресурс]. – Режим доступу: https://is.gd/zM5JxW. Дата доступу: квітень 2019. Назва з екрану.
19. Історія Python [Електронний ресурс]. – Режим доступу: https://is.gd/jVKFSd. Дата доступу: квітень 2019. Назва з екрану.
20. Переваги Python [Електронний ресурс]. – Режим доступу: https://is.gd/254GUf. Дата доступу: квітень 2019. Назва з екрану.
21. Недоліки Python [Електронний ресурс]. – Режим доступу: https://is.gd/GfE3Fw. Дата доступу: квітень 2019. Назва з екрану.
22. JavaScript [Електронний ресурс]. – Режим доступу: https://is.gd/ALzAZp. Дата доступу: квітень 2019. Назва з екрану.
23. Історія JavaScript [Електронний ресурс]. – Режим доступу: https://is.gd/OOuf6r. Дата доступу: квітень 2019. Назва з екрану.
24. Переваги JavaScript [Електронний ресурс]. – Режим доступу: https://is.gd/hI6cvO. Дата доступу: квітень 2019. Назва з екрану.
25. Недоліки JavaScript [Електронний ресурс]. – Режим доступу: https://is.gd/hFHIUv. Дата доступу: квітень 2019. Назва з екрану.
26. PostgreSQL [Електронний ресурс]. – Режим доступу: https://is.gd/XaX8TL. Дата доступу: квітень 2019. Назва з екрану.
27. Переваги PostgreSQL [Електронний ресурс]. – Режим доступу: https://is.gd/TmB29q. Дата доступу: квітень 2019. Назва з екрану.

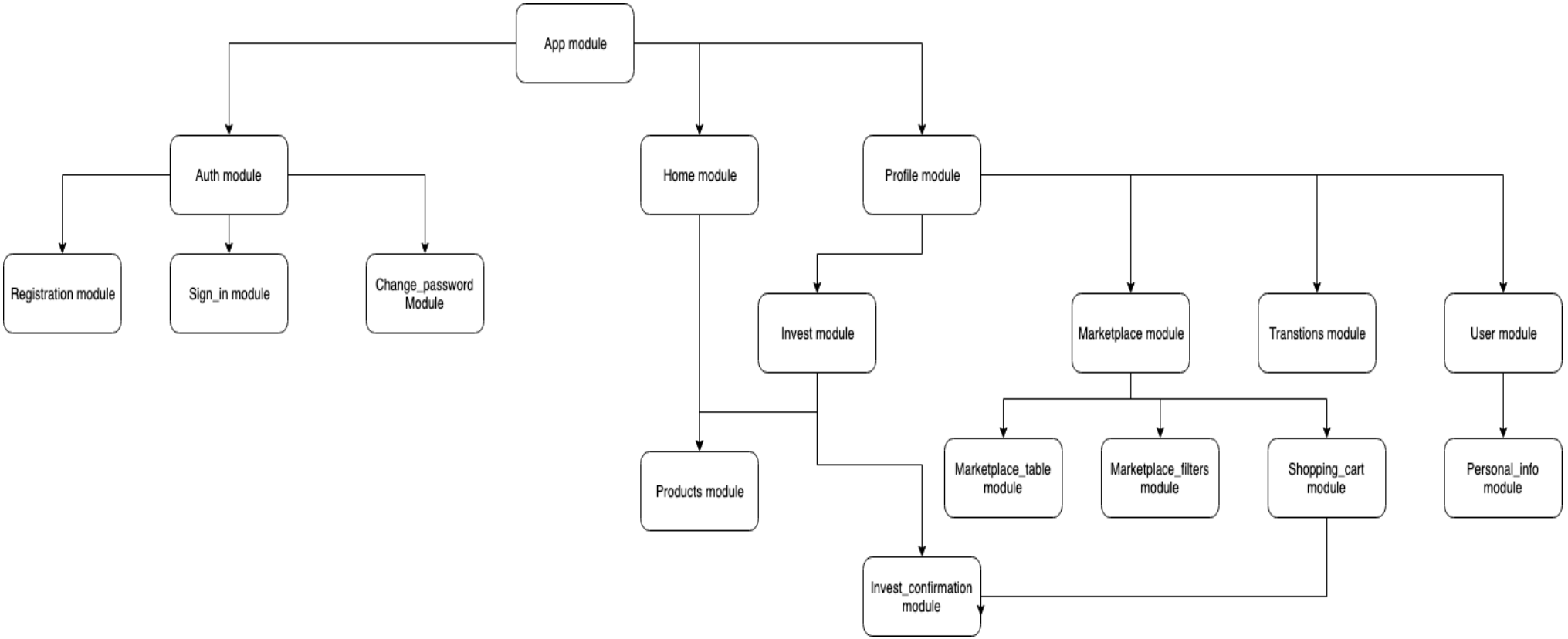
28. Недоліки PostgreSQL [Електронний ресурс]. – Режим доступу: https://is.gd/W80jQ8. Дата доступу: квітень 2019. Назва з екрану.
29. MySQL [Електронний ресурс]. – Режим доступу: https://is.gd/T61opJ. Дата доступу: квітень 2019. Назва з екрану.
30. Переваги MySQL [Електронний ресурс]. – Режим доступу: https://is.gd/dcY7WJ. Дата доступу: квітень 2019. Назва з екрану.
31. Недоліки MySQL [Електронний ресурс]. – Режим доступу: https://is.gd/8zrp9U. Дата доступу: квітень 2019. Назва з екрану.
32. SQLite [Електронний ресурс]. – Режим доступу: https://is.gd/TCv9DF. Дата доступу: квітень 2019. Назва з екрану.
33. Переваги SQLite [Електронний ресурс]. – Режим доступу: https://is.gd/vRu9Bw. Дата доступу: квітень 2019. Назва з екрану.
34. Недоліки SQLite [Електронний ресурс]. – Режим доступу: https://is.gd/ntI4Fh. Дата доступу: квітень 2019. Назва з екрану.
35. IntelliJ IDEA [Електронний ресурс]. – Режим доступу: https://is.gd/gSLQSa. Дата доступу: квітень 2019. Назва з екрану.
36. Angular [Електронний ресурс]. – Режим доступу: https://is.gd/ldnBwa. Дата доступу: квітень 2019. Назва з екрану.
37. Архітектура Angular [Електронний ресурс]. – Режим доступу: https://is.gd/c2cRf7. Дата доступу: квітень 2019. Назва з екрану.
38. Spring Boot [Електронний ресурс]. – Режим доступу: https://is.gd/TeV9Vw. Дата доступу: квітень 2019. Назва з екрану.
39. Клієнт–серверна архітектура [Електронний ресурс]. – Режим доступу: https://is.gd/noYeT4. Дата доступу: квітень 2019. Назва з екрану.
40. I18n [Електронний ресурс]. – Режим доступу: https://is.gd/vmfc7v. Дата доступу: квітень 2019. Назва з екрану.
41. Серверний рендеринг [Електронний ресурс]. – Режим доступу: https://is.gd/pN21En. Дата доступу: квітень 2019. Назва з екрану.
42. Angular Universal [Електронний ресурс]. – Режим доступу: https://is.gd/FXISTG. Дата доступу: квітень 2019. Назва з екрану.

43. HTTPS [Електронний ресурс]. – Режим доступу: https://is.gd/EcLY3r. Дата доступу: квітень 2019. Назва з екрану.
44. Тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: http://bit.do/eTKBY. Дата доступу: травень 2019. Назва з екрану.
45. Димове тестування [Електронний ресурс]. – Режим доступу: http://bit.do/eTKS7. Дата доступу: травень 2019. Назва з екрану.
46. Lighthouse [Електронний ресурс]. – Режим доступу: http://bit.do/eTLwH. Дата доступу: травень 2019. Назва з екрану.

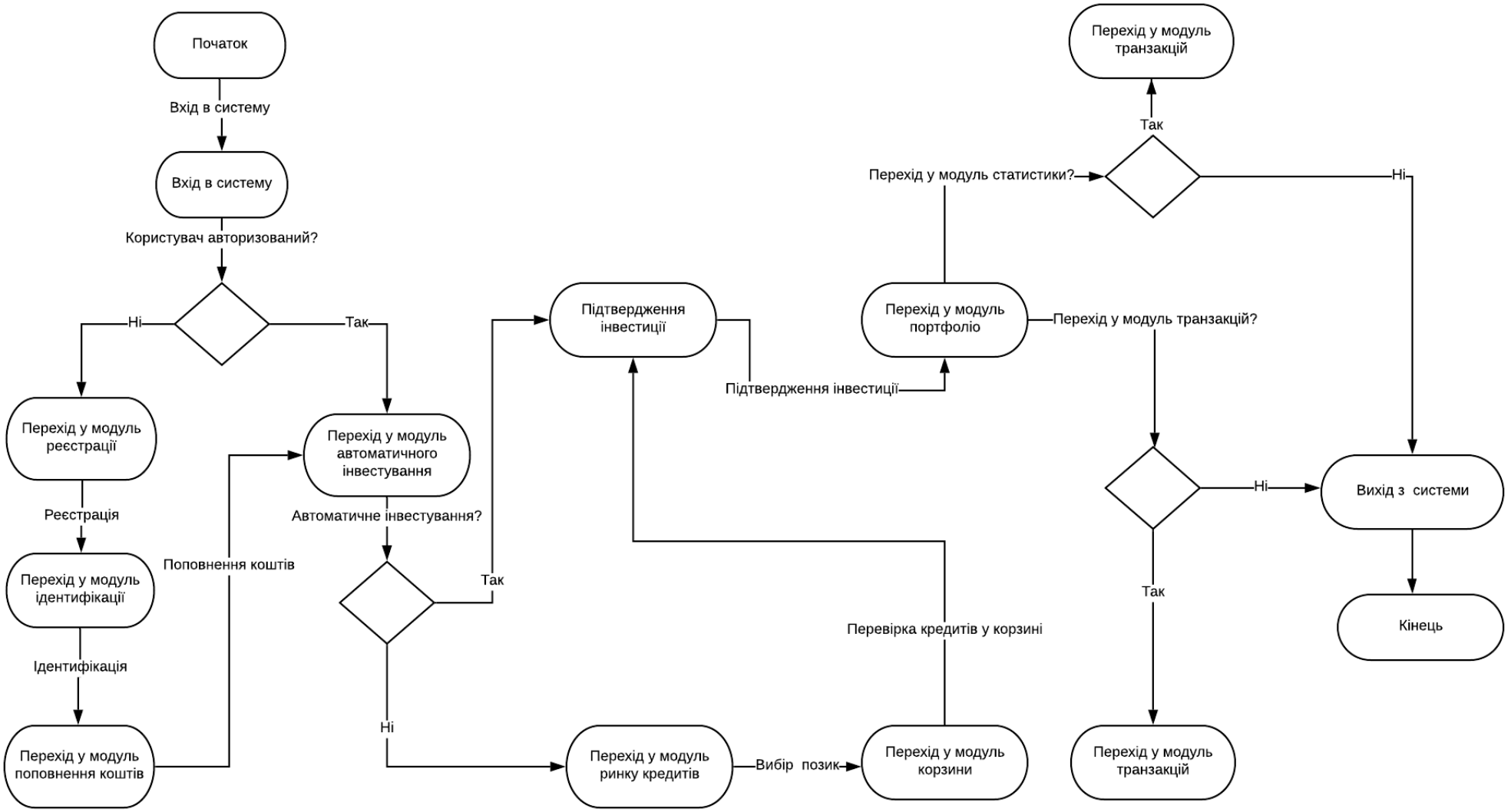
ДОДАТКИ

Додаток 1

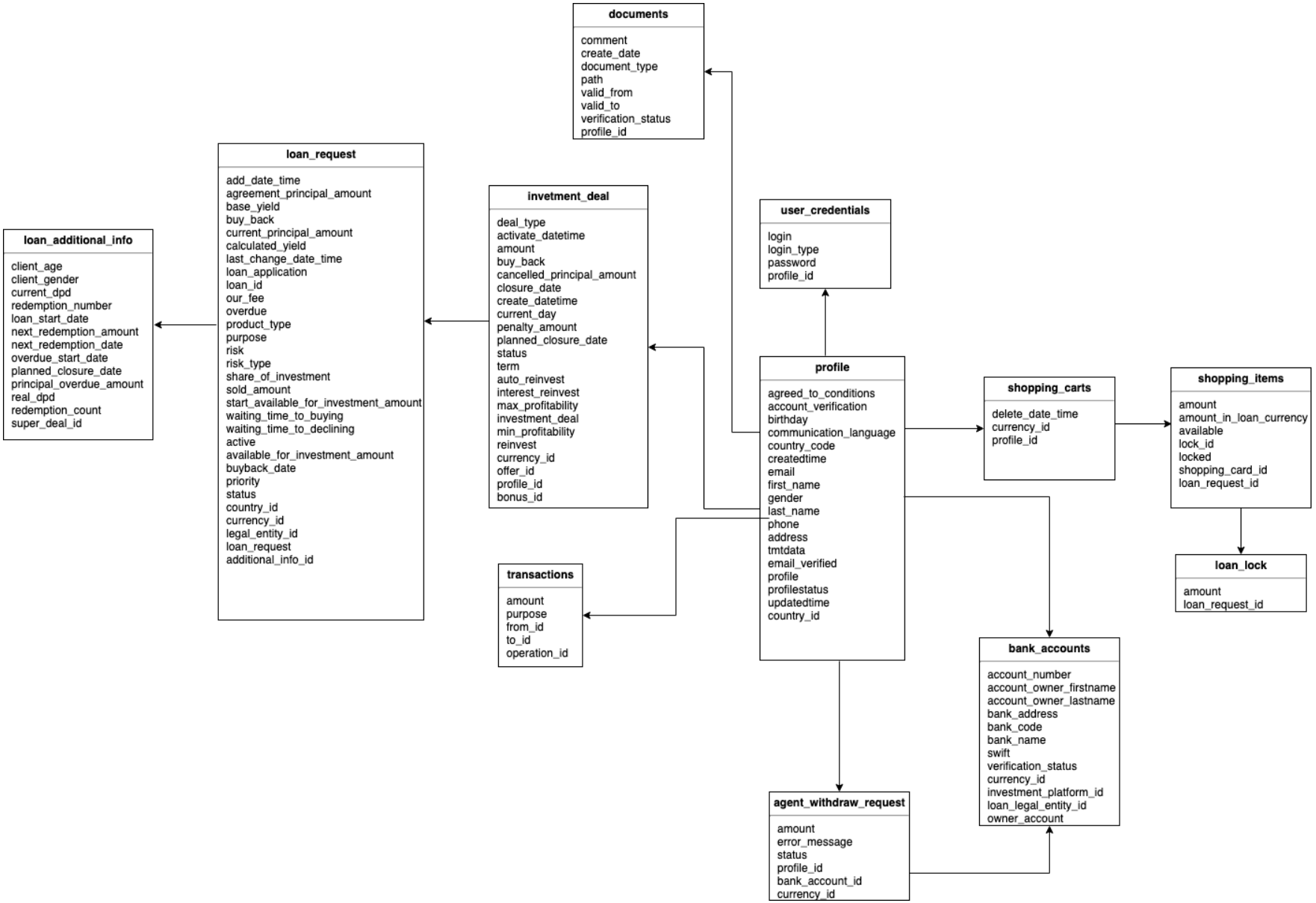
Копії графічних матеріалів



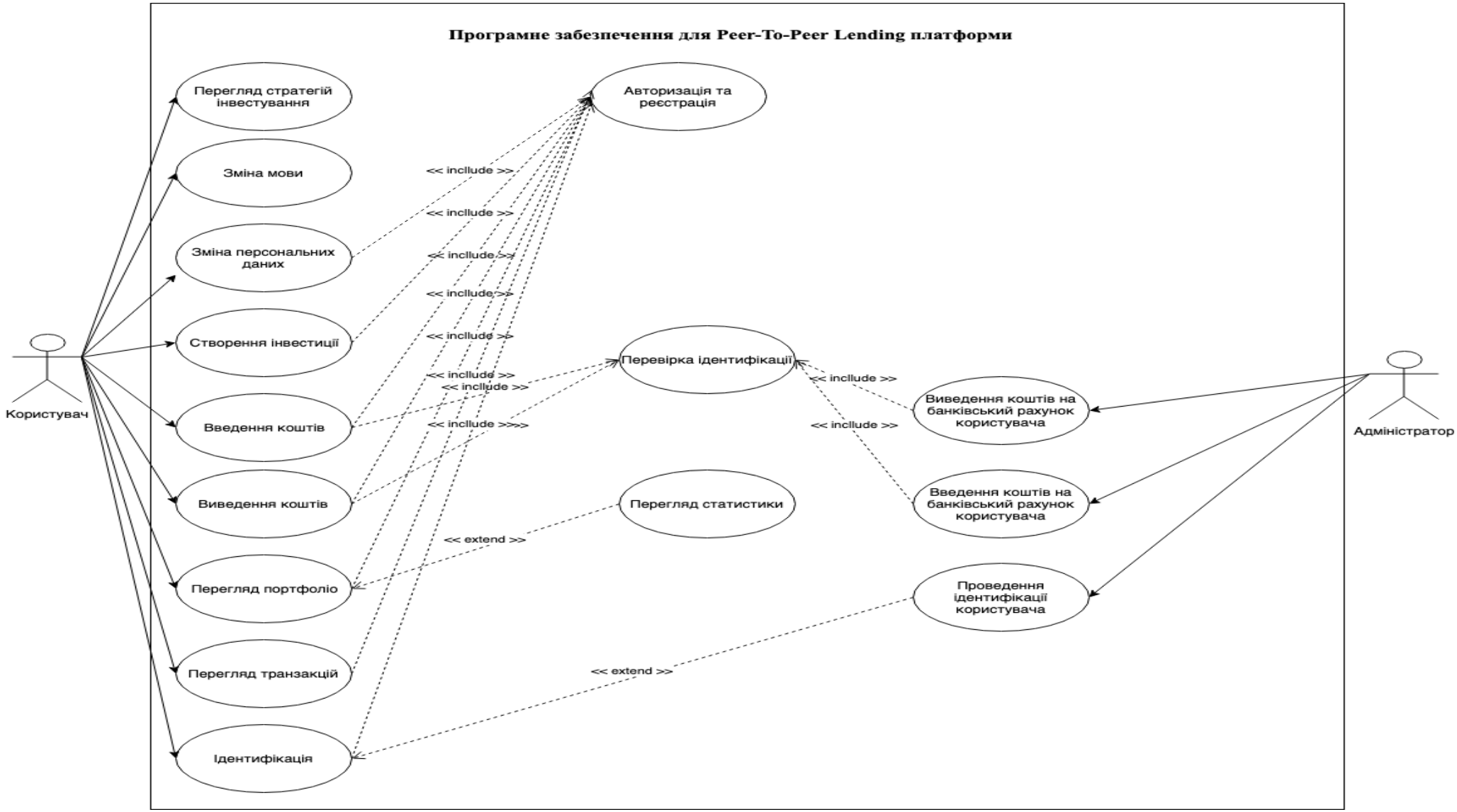
ДП.045430-06-99
Програмне забезпечення Peer-To-Peer Lending платформи. Зв’язок модулів. Діаграма компонентів



ДП.045430-07-99
Програмне забезпечення для Peer-To-Peer Lending платформи. Процес інвестування. Діаграма діяльності



Тимошенко Владислав, група КП-51



Тимошенко Владислав, група КП-51

Додаток 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Програмне забезпечення для Peer-To-Peer Lending платформи

Виконав: Тимошенко Владислав Анатолійович

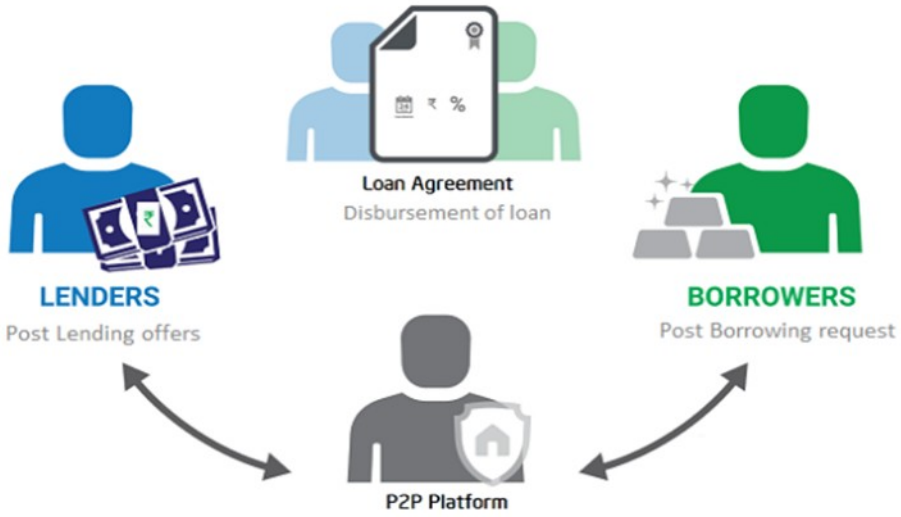
Керівник дипломного проекту: доцент, к.т.н., Олещенко Любов Михайлівна

Київ – 2019

АКТУАЛЬНІСТЬ

Переваги P2P Lending платформ

- Альтернатива банкам та іншим фінансовим установам;
- Вищі процентні ставки на інвестування грошей у порівнянні з іншими фінансовими установами;
- Економія часу – усі операції виконуються онлайн;
- Максимально прозора і зрозуміла система.



Мета проекту

розробити високопродуктивну Peer-To-Peer lending платформу, яка дозволятиме формувати кредитний портфель власноруч та автоматично, динамічно оновлювати дані та матиме можливість короткострокового та довгострокового інвестування коштів у мережі Інтернет.

Задачі дипломного проекту

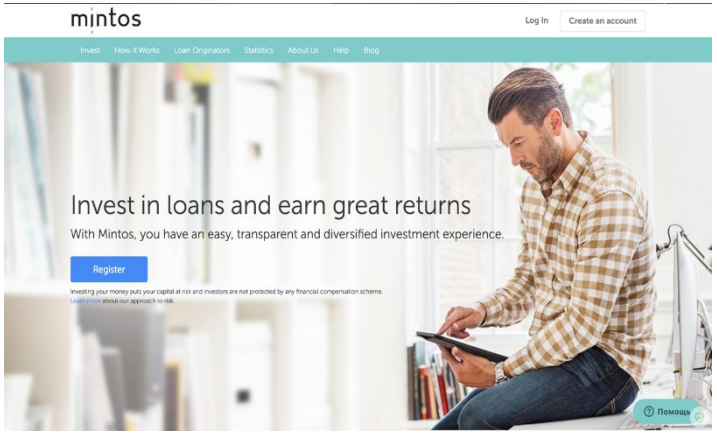
1. Здійснити аналіз існуючих програмних рішень для інвестування коштів у мережі Інтернет.

2. На основі виділених недоліків існуючих рішень обрати стек технологій для проектування програмного забезпечення.

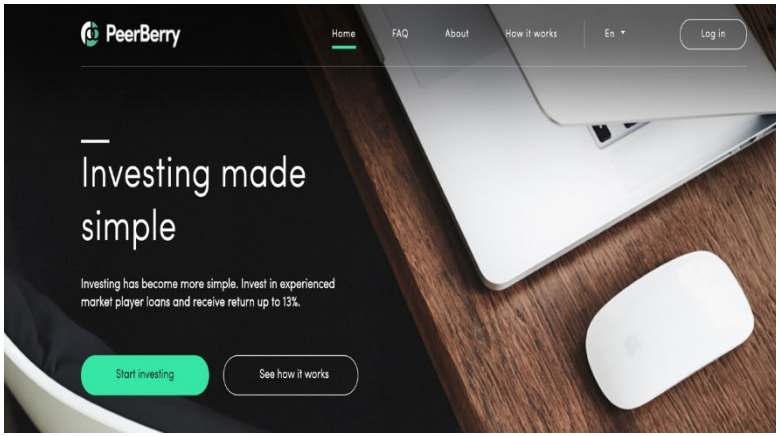
3. Розробити програмне забезпечення для P2P Lending платформи, яке матиме функціональні переваги та переваги продуктивності його використання в мережі Інтернет у порівнянні з існуючими рішеннями.

4. Здійснити тестування розробленого програмного забезпечення. Порівняти розроблене програмне забезпечення з існуючими аналогами.

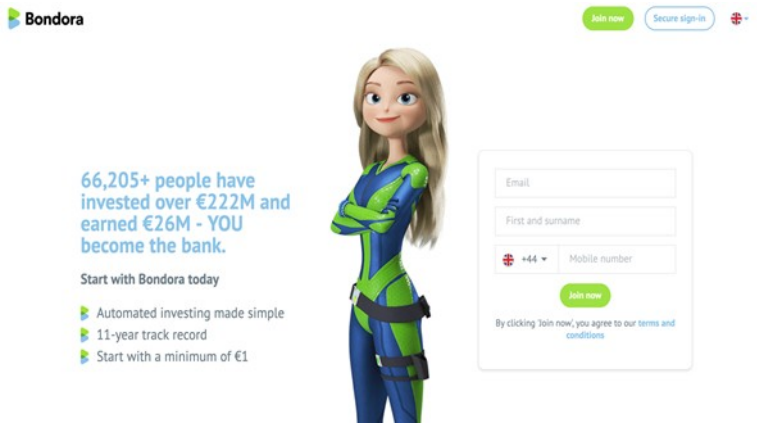
Аналіз існуючих рішень



Mintos



PeerBerry



Bondora

Переваги:

- Система диверсифікації ризику
- Наявність декількох мов
- Короткострокове і довгострокове інвестування

Недоліки існуючих рішень

- Відсутність динамічного оновлення даних
- Відсутність SPA-архітектури
- Відсутність стратегій інвестування або інвестування в окремі кредити
- Недостатньо інформації для інвестування

Основні вимоги до системи

- Клієнт-серверна архітектура системи;
- Реєстрація та авторизація користувачів;
- Можливість створити інвестицію автоматично;
- Можливість створити інвестицію шляхом вибору окремих кредитів;
- Можливість зміни мови;
- Можливість завантажувати документи;
- SPA – архітектура у клієнтській частині;
- Динамічне оновлення даних.

Використані технології

Back end:

Java

Spring Boot

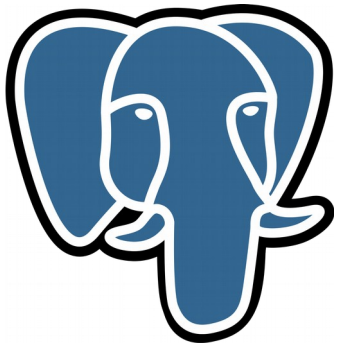
Front End:

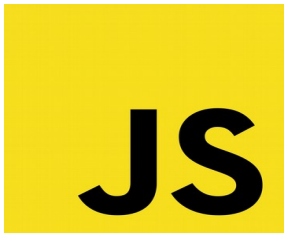
JavaScript

Angular

БД:

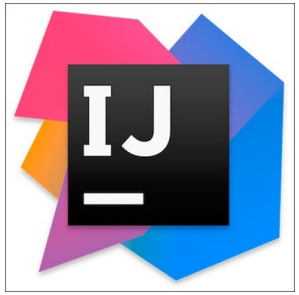
PostgreSQL





Середовище розробки:

Intellij idea



Додаткові технології та бібліотеки

I18n

Angular-Universal

RxJS

Серверний рендеринг

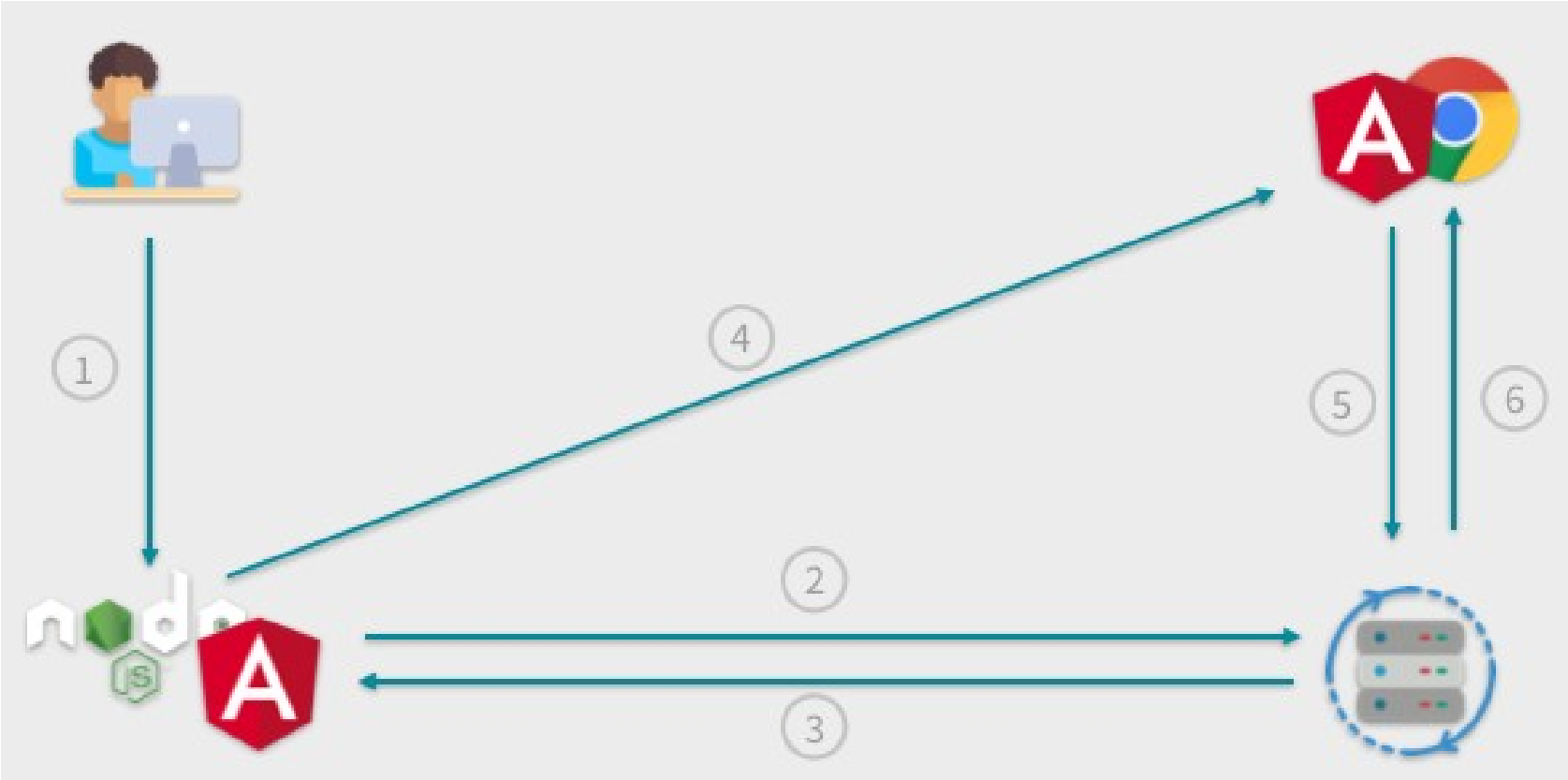
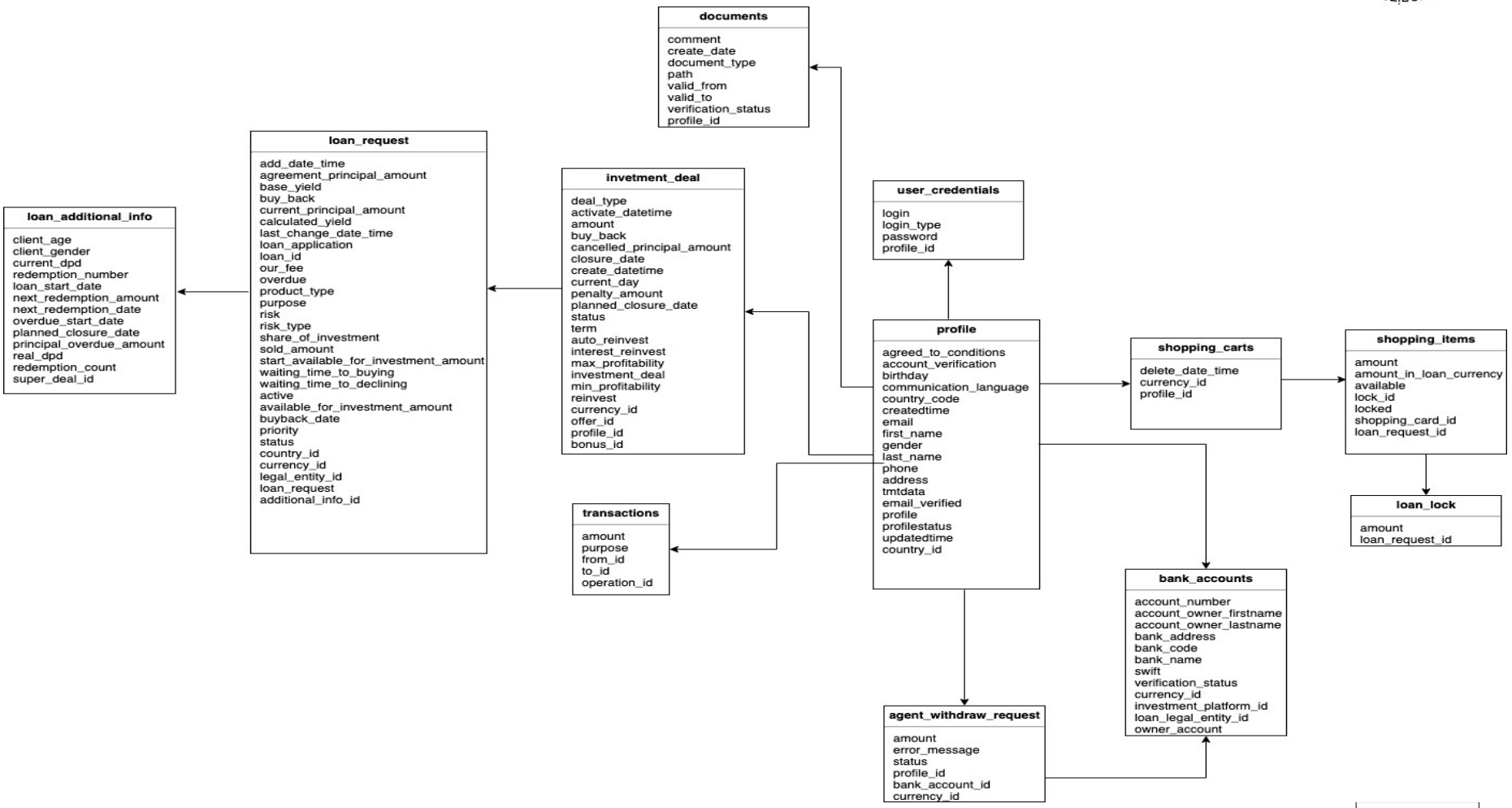
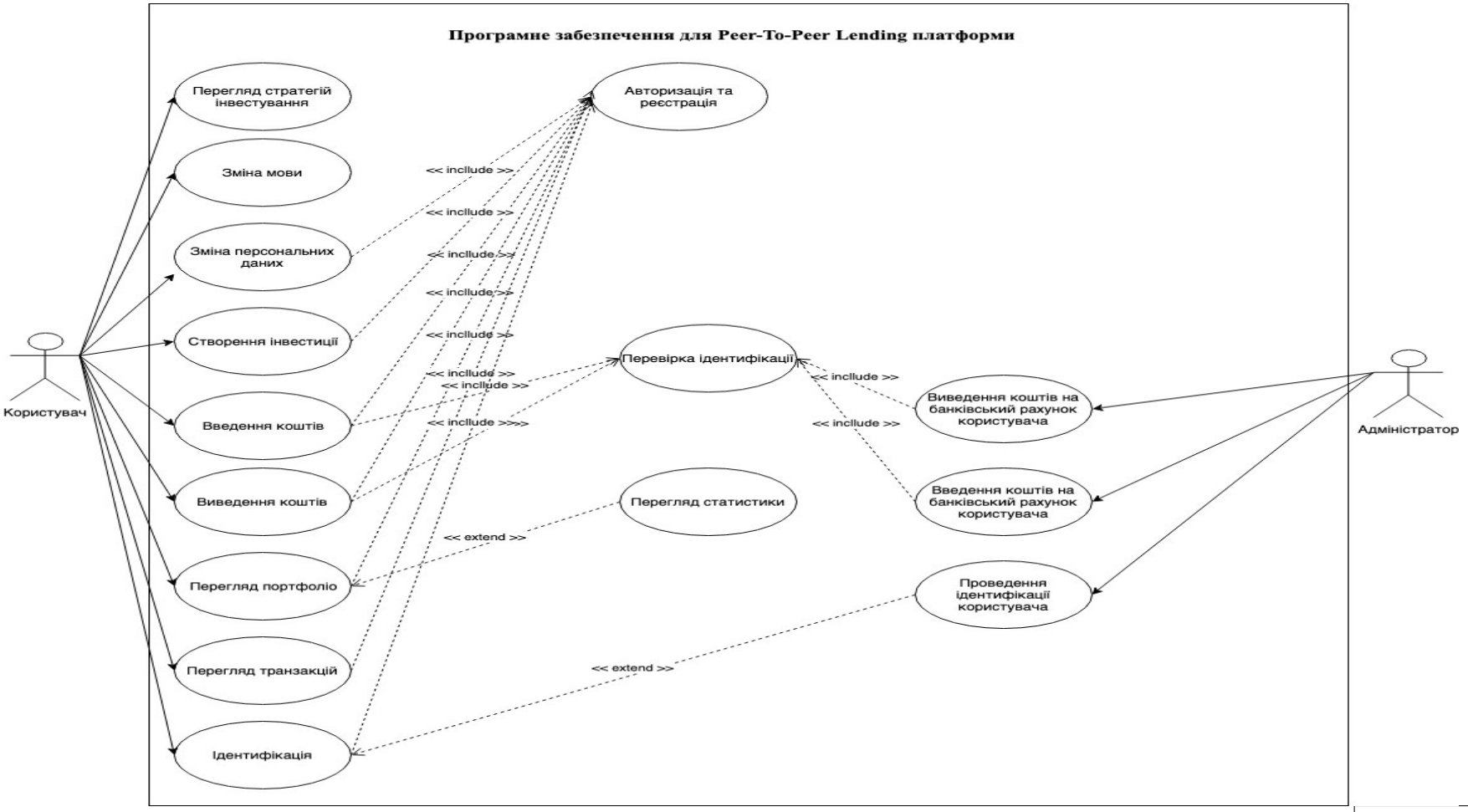


Схема бази даних



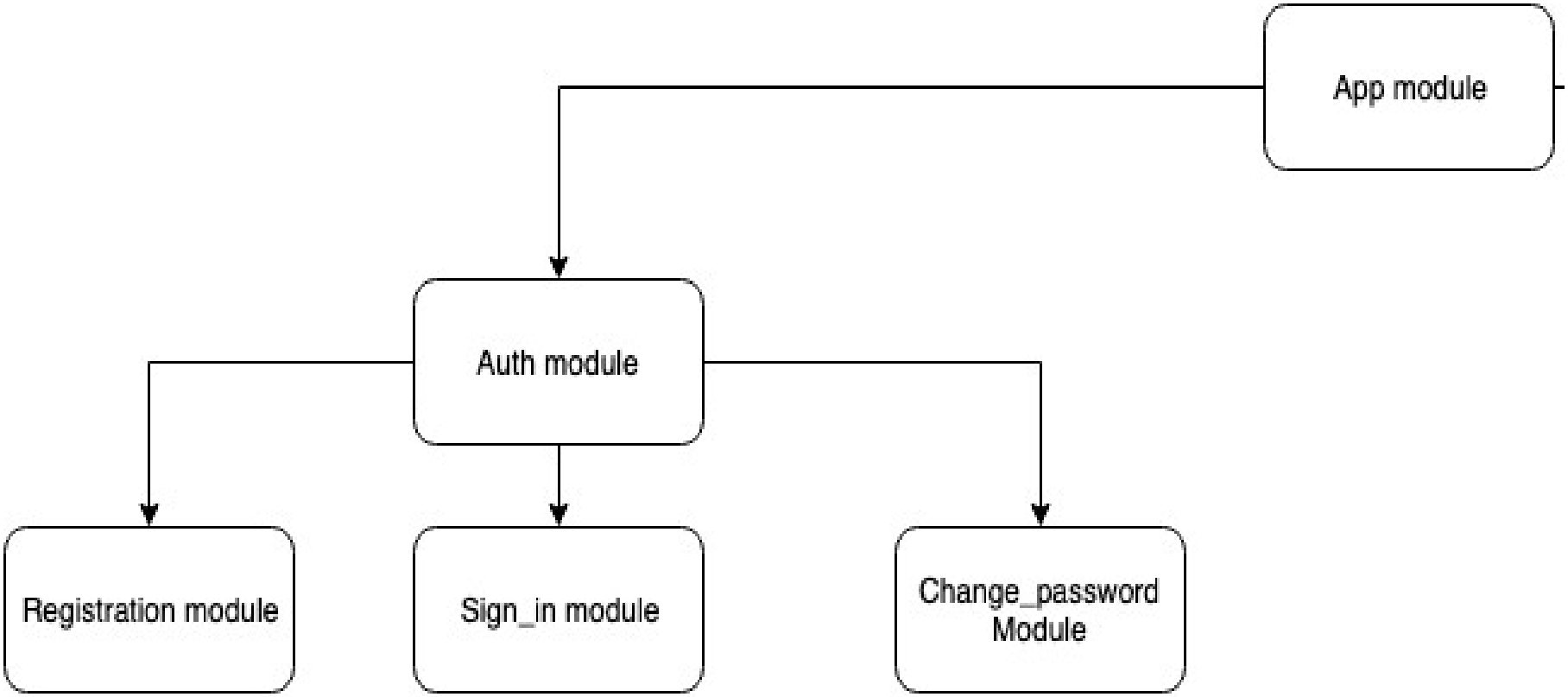
UML - діаграма варіантів використання



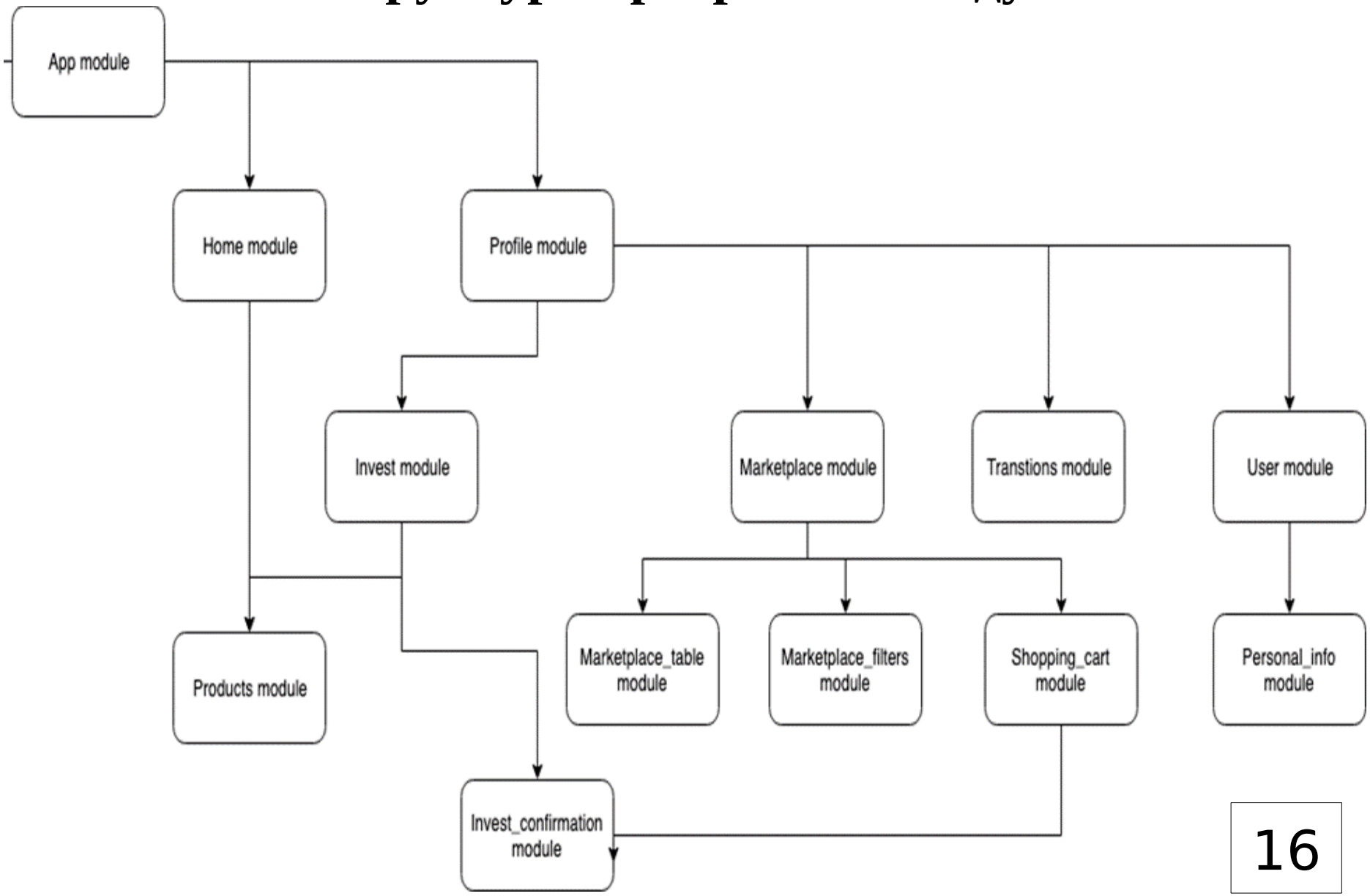
Основні модулі

- модуль авторизації, реєстрації, особистий кабінет;
- модуль для верифікації інформації про клієнта;
- модуль зміни мови платформи;
- модуль портфоліо;
- модуль статистики;
- модуль ринку кредитів та корзина;
- модуль транзакцій;
- модуль для введення коштів на платформу;
- модуль для виведення коштів з платформи;
- модуль автоматичного створення інвестицій.

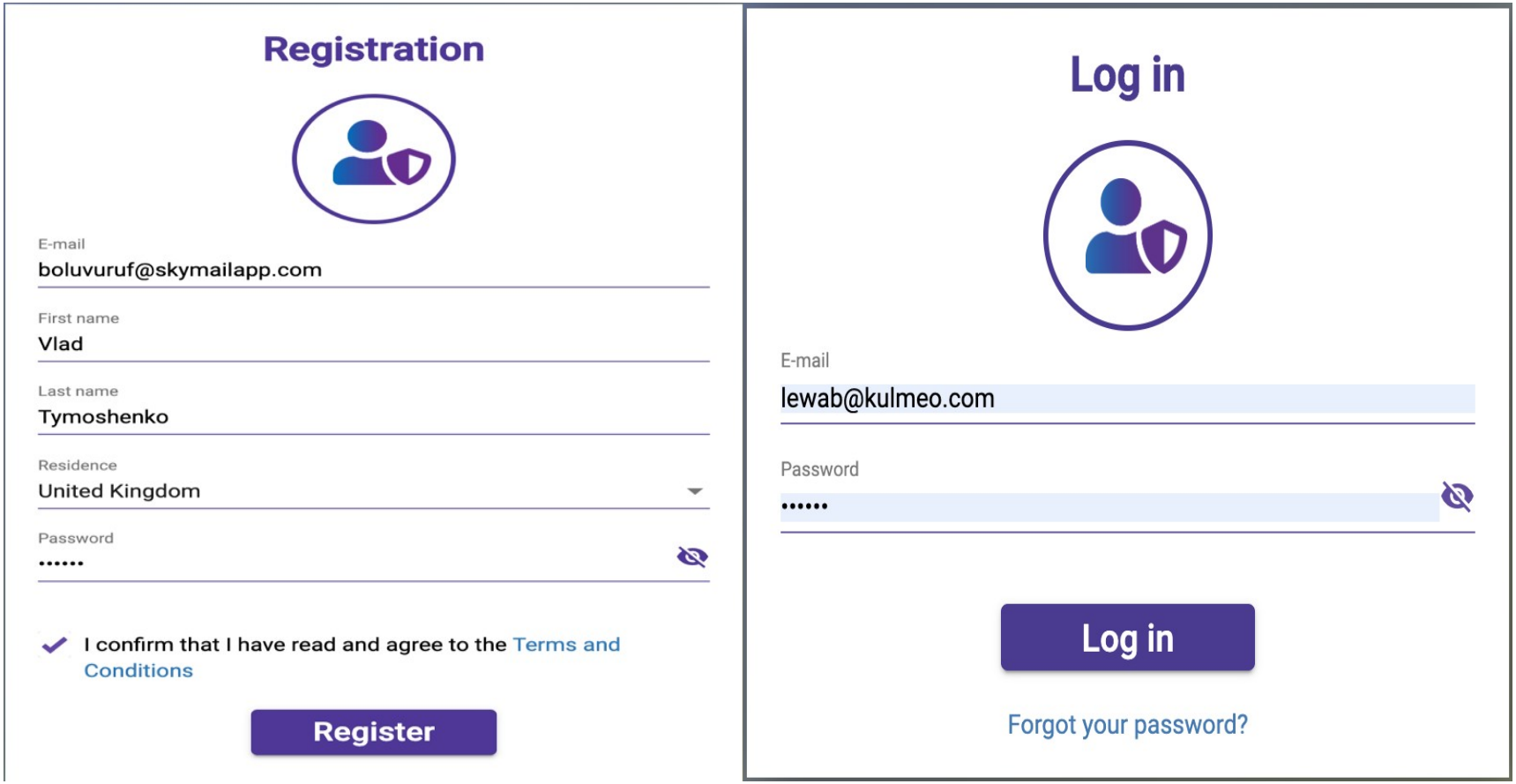
Структура програмних модулів



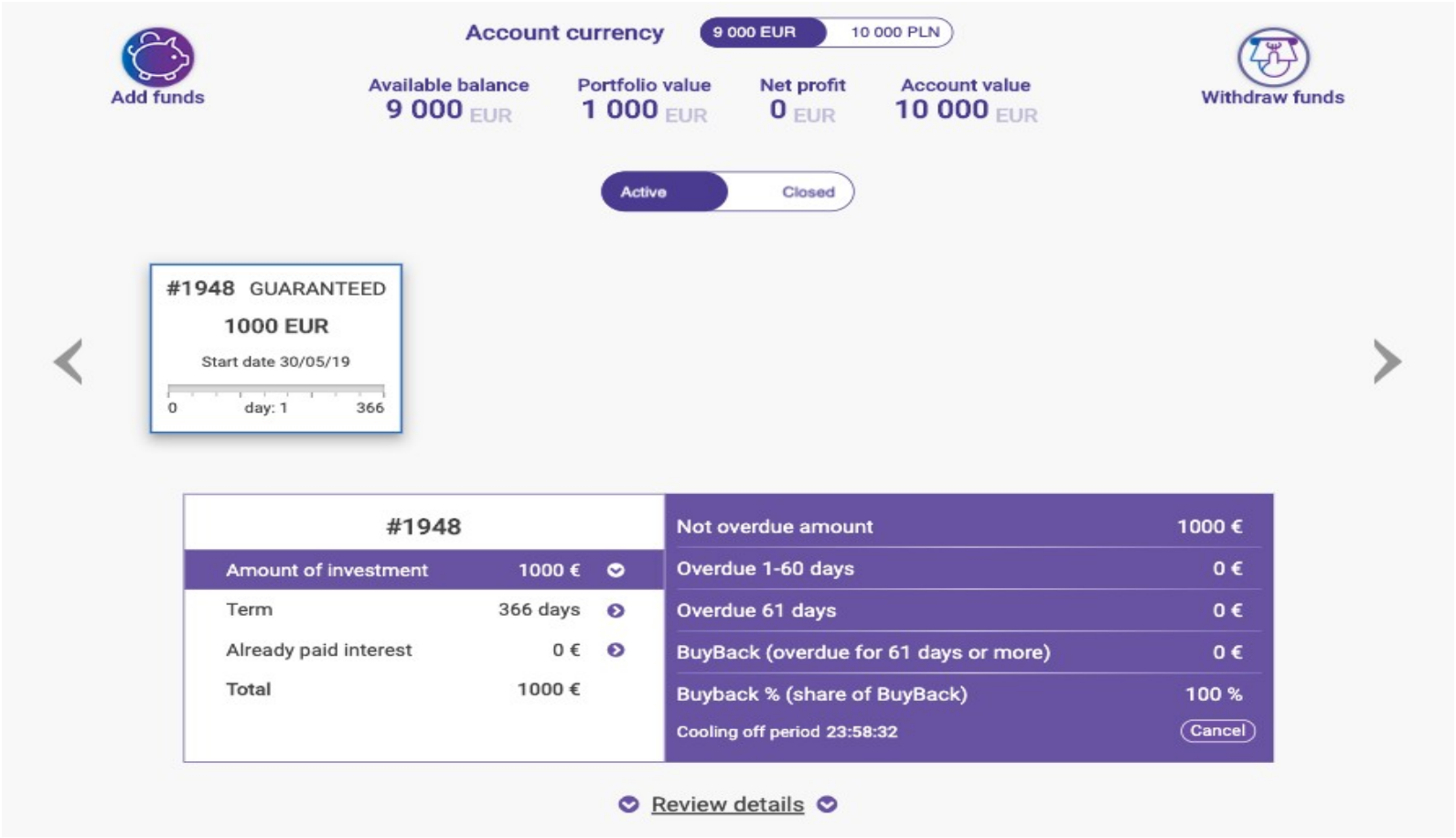
Структура програмних модулів



Розроблене програмне забезпечення. Модуль реєстрації / авторизації



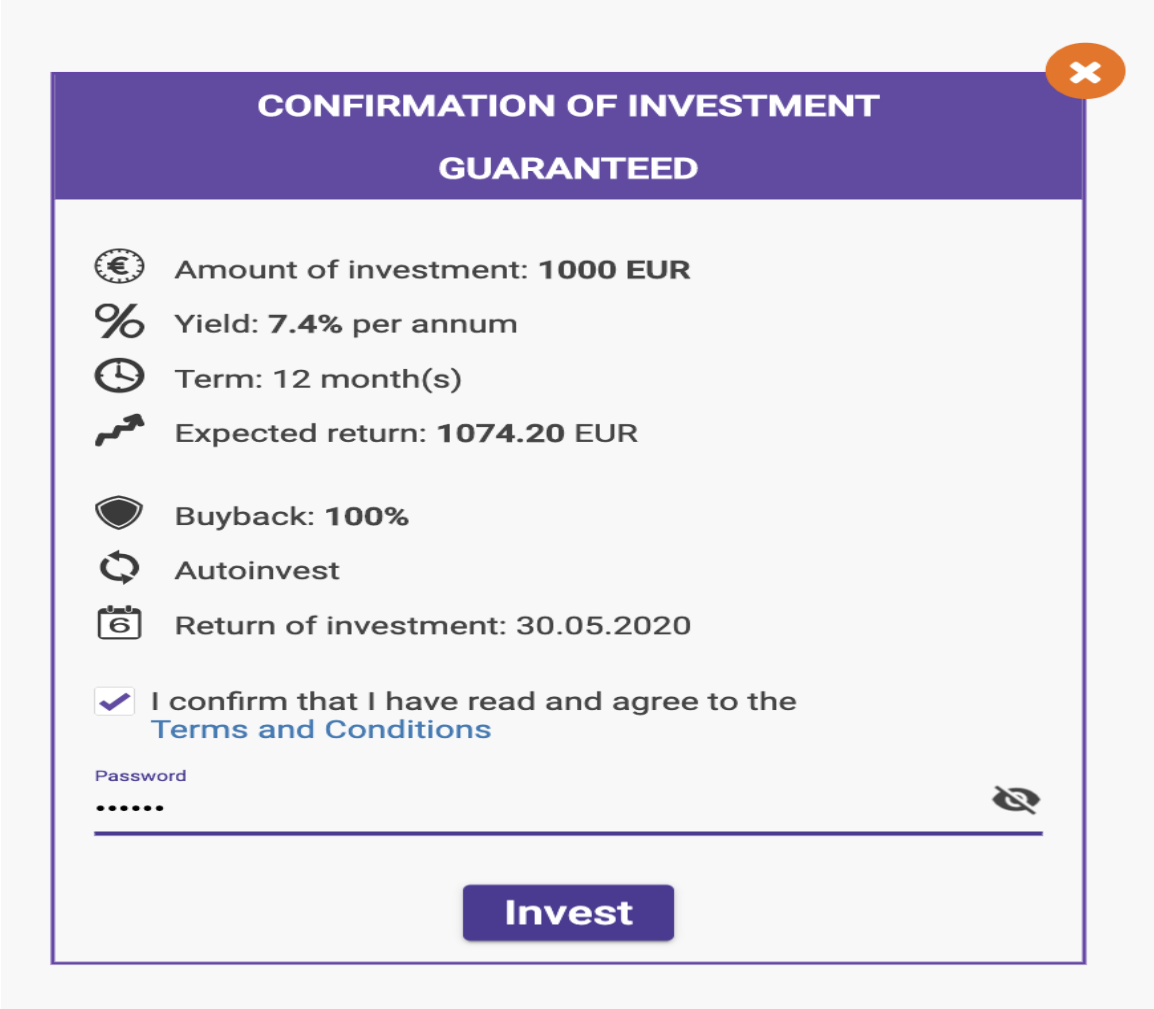
Розроблене програмне забезпечення. Модуль портфоліо



Розроблене програмне забезпечення. Модуль транзакцій

Date	Detail	Turnover
10.06.2019 03:52:02	Transaction ID: 15359837 - Deposits	10 000 €
10.06.2019 03:56:03	Transaction ID: 15359855 - Loan: 47152 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359846 - Loan: 47156 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359858 - Loan: 47151 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359843 - Loan: 47163 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359840 - Loan: 47164 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359861 - Loan: 47150 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359864 - Loan: 47149 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359849 - Loan: 47155 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359867 - Loan: 47146 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359870 - Loan: 47145 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359852 - Loan: 47154 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359879 - Loan: 47141 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359876 - Loan: 47142 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359894 - Loan: 47131 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359891 - Loan: 47135 - Investment	-40 €
10.06.2019 03:56:03	Transaction ID: 15359888 - Loan: 47136 - Investment	-40 €

Розроблене програмне забезпечення. Модуль підтвердження інвестиції

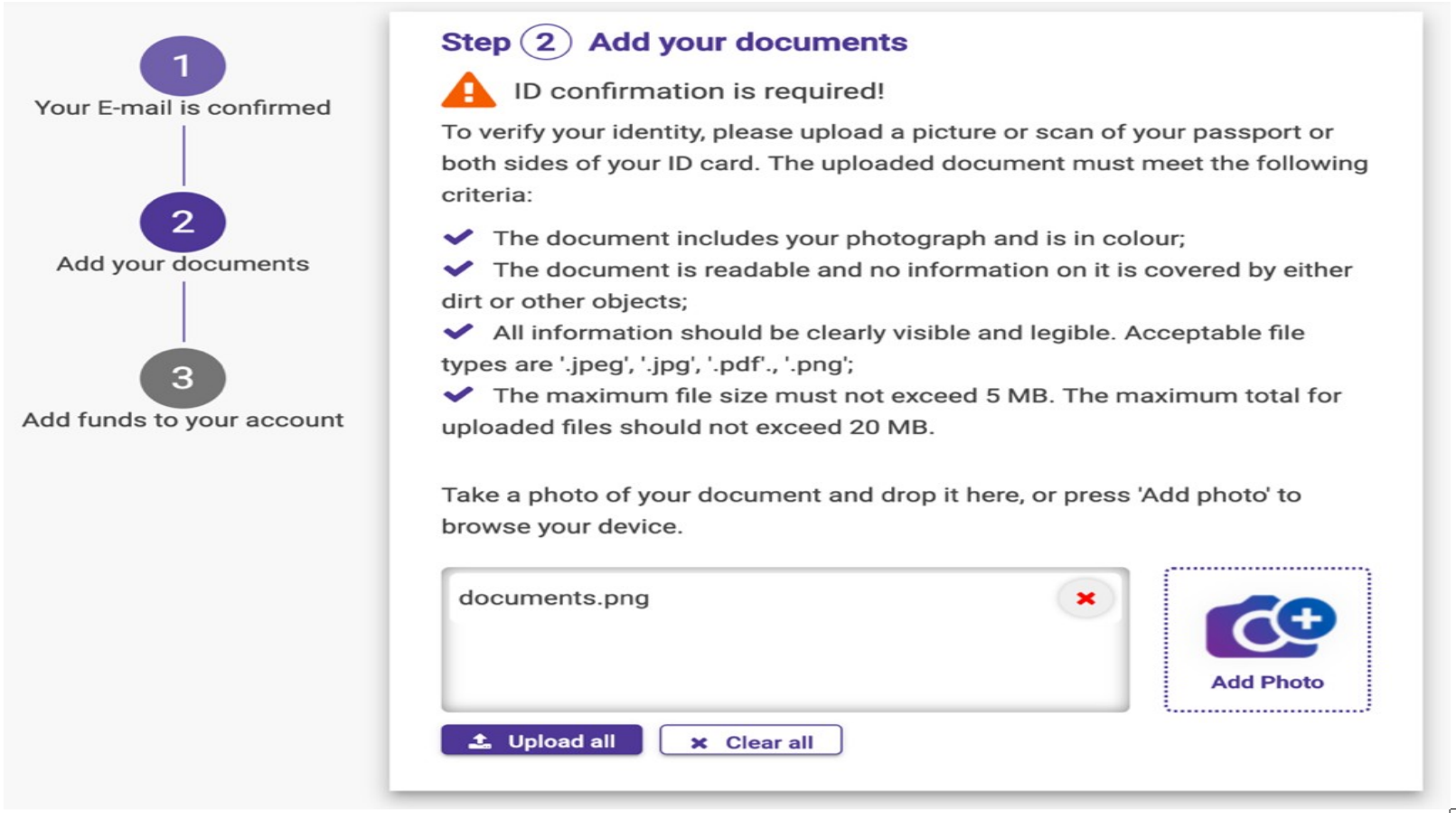


Розроблене програмне забезпечення. Модуль статистики

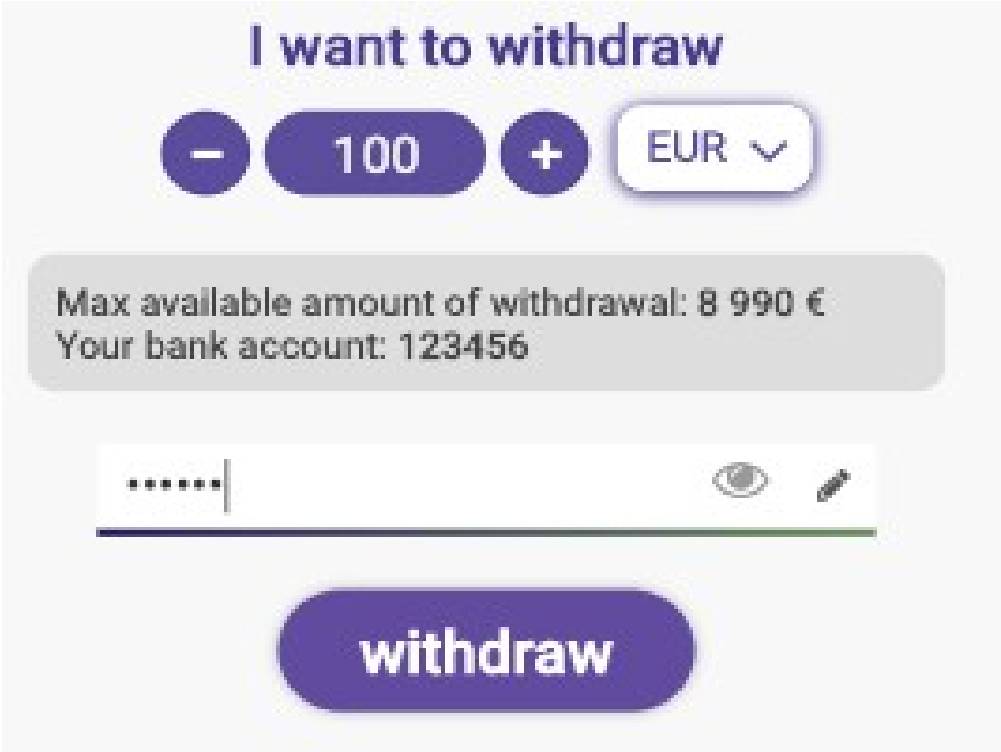
Loan id	Rating	Status	Date of purchase	Closing Date	Current Amount	Accrued interest	Paid interest
47203	G		30.05.2019	22.06.2019	10 €	0 €	0 €
47201	G		30.05.2019	23.06.2019	10 €	0 €	0 €
47200	G		30.05.2019	23.06.2019	10 €	0 €	0 €
47199	F		30.05.2019	22.06.2019	10 €	0 €	0 €
47198	F		30.05.2019	23.06.2019	10 €	0 €	0 €
47197	C		30.05.2019	23.06.2019	10 €	0 €	0 €
47196	G		30.05.2019	23.06.2019	10 €	0 €	0 €

Issue Date	26.05.2019	Repaid principal	0 €	Buyback	100%
Start investment amount	10 €	Interest Rate	7.41%	Buyback date	-
Next redemption date	-	Overdue from	-	Buyback amount	0 €
		Overdue days	0		

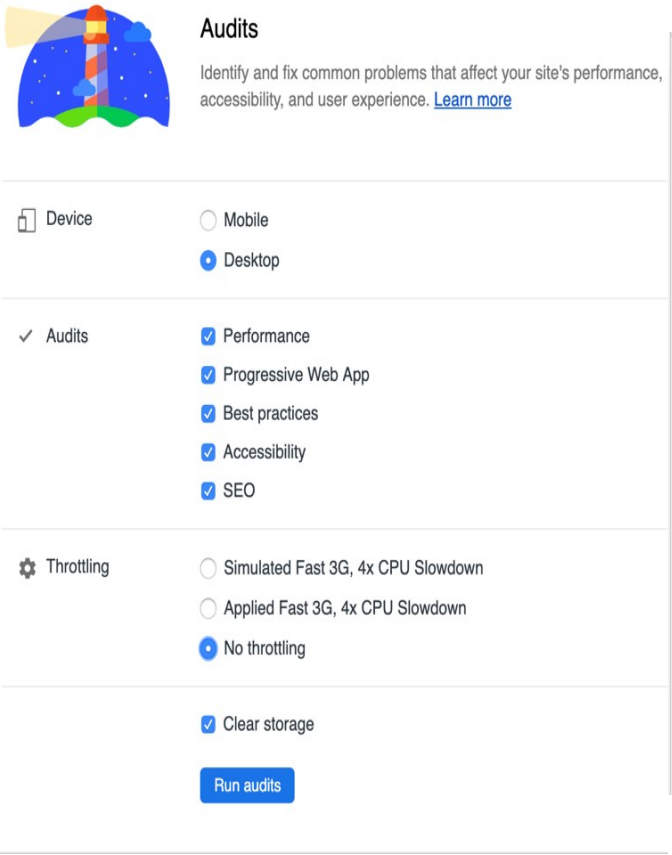
Розроблене програмне забезпечення. Модуль ідентифікації

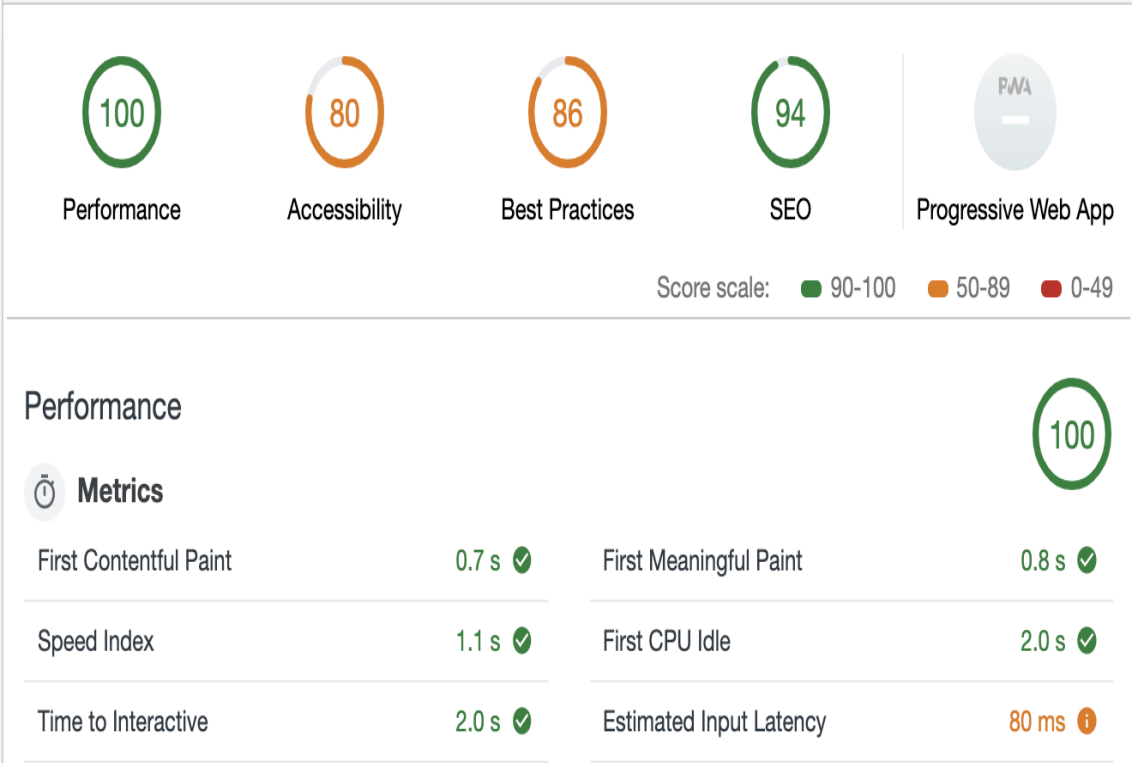


Розроблене програмне забезпечення. Модуль виведення коштів

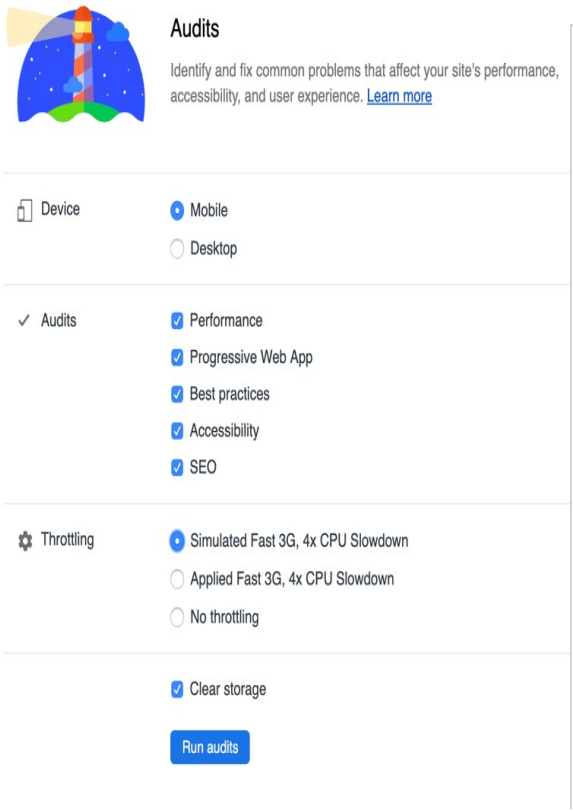


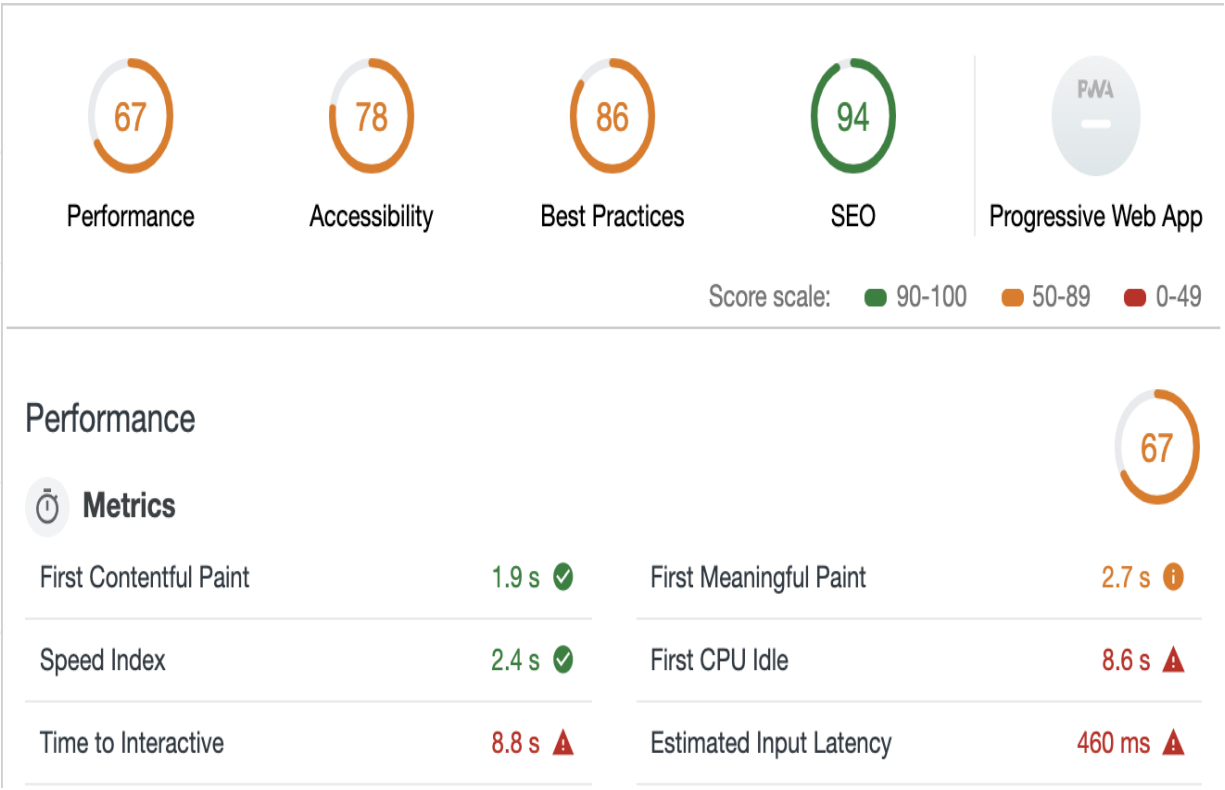
Тестування розробленого програмного забезпечення в Lighthouse





Тестування розробленого програмного забезпечення в Lighthouse





Порівняння запропонованого програмного забезпечення з існуючими аналогами

Результат перевірки продуктивності платформи	Запропоно-ване програмне забезпе-чення	Mintos	PeerBerry	Bondora
Найкращий випадок (використовується комп'ютер, без симуляції 3G інтернету, без уповільнення процесора)	100%	98%	99%	100%
Найгірший випадок (використовується мобільний телефон, симуляція 3G інтернету, уповільнення процесора в 4 рази)	67%	43%	10%	60%

Висновки

1. Здійснено аналіз існуючих програмних рішень для інвестування коштів у мережі Інтернет та виділено їх основні недоліки.
2. На основі виділених недоліків існуючих рішень обрано стек технологій для проектування програмного забезпечення.
3. Розроблено архітектуру програмного забезпечення для P2P Lending платформи. Розроблене програмне забезпечення має додаткові можливості для автоматичного інвестування та інвестування в окремі кредити, має декілька мов, вбудовану систему диверсифікації ризику та можливість комбінування стратегій інвестування. Для досягнення високої продуктивності програмного забезпечення застосовано технологію серверного рендерингу.
4. Здійснено тестування розробленого ПЗ та порівняно результати з існуючими аналогами. Згідно проведеного тестування, запропоноване програмне забезпечення має вищу продуктивність у порівнянні з аналогами.

Дякую за увагу!

Додаток 3

Лістинг модуля ринку кредитів

```
<div class="example-container mat-elevation-z8">
  <div class="example-loading-shade"
       *ngIf="pending">
    <mat-spinner></mat-spinner>
  </div>
  <div class="example-table-container" [ngClass]="{'hide': !loans || !loans.length || pending}">
    <table mat-table [dataSource]="loans" class="example-table" matSort multiTemplateDataRows>

      <!-- Loan Id Column -->
      <ng-container matColumnDef="loanId">
        <th mat-header-cell *matHeaderCellDef>{{'marketplace.marketplace-table.loan-id' | translate}}</th>
        <td mat-cell *matCellDef="let row">
          <div class="layout inline">
            <ng-container *ngIf="expandedElement === row; else arrowDown">
              <i class="fa fa-chevron-circle-up m-r-5" aria-hidden="true"></i>
            </ng-container>
            <ng-template #arrowDown>
              <i class="fa fa-chevron-circle-down m-r-5" aria-hidden="true"></i>
            </ng-template>
            {{row.innerLoanId}}
            <ng-container [ngSwitch]="row.country">
              <span *ngSwitchCase="'DE'" class="lang-icon de"></span>
              <app-tooltip *ngSwitchCase="'Poland'" class="layout inline" [description]="'misc.tooltip.statistic-table-
country' | translate" [className]="'statistick'">
                <span  class="lang-icon pl"></span>
              </app-tooltip>
              <span *ngSwitchDefault class="lang-icon en"></span>
            </ng-container>
          </div>
        </td>
      </ng-container>

      <!-- Rating Column -->
      <ng-container matColumnDef="rating">
        <th mat-header-cell *matHeaderCellDef mat-sort-header><app-tooltip [description]="'misc.tooltip.statistic-
rating' | translate" [className]="'statistic-header'">{{'profile.statistics.header.rating' | translate}}</app-
tooltip></th>
        <td mat-cell *matCellDef="let row">
          <ng-container [ngSwitch]="row.rating">
            <app-tooltip *ngSwitchCase="'AA'" [description]="'misc.tooltip.statistic-rating-AA' | translate"
[className]="'statistick'">
              {{row.rating}}
            </app-tooltip>
            <app-tooltip *ngSwitchCase="'A'" [description]="'misc.tooltip.statistic-rating-A' | translate"
[className]="'statistick'">
              {{row.rating}}
            </app-tooltip>
            <app-tooltip *ngSwitchCase="'B'" [description]="'misc.tooltip.statistic-rating-B' | translate"
[className]="'statistick'">
              {{row.rating}}
            </app-tooltip>
            <app-tooltip *ngSwitchCase="'C'" [description]="'misc.tooltip.statistic-rating-C' | translate"
[className]="'statistick'">
              {{row.rating}}
            </app-tooltip>
            <app-tooltip *ngSwitchCase="'D'" [description]="'misc.tooltip.statistic-rating-D' | translate"
[className]="'statistick'">
              {{row.rating}}
            </app-tooltip>
            <app-tooltip *ngSwitchCase="'E'" [description]="'misc.tooltip.statistic-rating-E' | translate"
[className]="'statistick'">
              {{row.rating}}
            </app-tooltip>
```

```
          <app-tooltip *ngSwitchCase="'F'" [description]="'misc.tooltip.statistic-rating-F' | translate"
[className]="'statistick'">
            {{row.rating}}
          </app-tooltip>
          <app-tooltip *ngSwitchCase="'G'" [description]="'misc.tooltip.statistic-rating-G' | translate"
[className]="'statistick'">
            {{row.rating}}
          </app-tooltip>
        </ng-container>
      </td>
    </ng-container>

    <!-- Issue Date Column -->
    <ng-container matColumnDef="issueDate">
      <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'marketplace.marketplace-table.issue-date' |
translate}}</th>
      <td mat-cell *matCellDef="let row">{{row.issueDate | date:'dd.MM.yyyy'}}</td>
    </ng-container>

    <!-- Interest Rate Column -->
    <ng-container matColumnDef="interestRate">
      <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'marketplace.marketplace-table.interest-rate' |
translate}}</th>
      <td mat-cell *matCellDef="let row">{{row.interestRate}}%</td>
    </ng-container>

    <!-- Amount Column -->
    <ng-container matColumnDef="loanAmount">
      <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'marketplace.marketplace-table.amount' |
translate}}</th>
      <td mat-cell *matCellDef="let row">{{row.loanAmount}} {{row.shownCurrency}}</td>
    </ng-container>

    <!-- Term Column -->
    <ng-container matColumnDef="term">
      <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'marketplace.marketplace-table.term' |
translate}}</th>
      <td mat-cell *matCellDef="let row">{{row.term}} d.</td>
    </ng-container>

    <!-- Available For Invest Column -->
    <ng-container matColumnDef="availableForInvestment">
      <th mat-header-cell *matHeaderCellDef mat-sort-header
class="availableForInvestment">{{'marketplace.marketplace-table.available-for-investment' | translate}}</th>
      <td mat-cell *matCellDef="let row">{{row.availableForInvestment | digitsAfterDot:'2'}}
{{row.shownCurrency}}</td>
    </ng-container>

    <!-- Buyback Column -->
    <ng-container matColumnDef="buyback">
      <th mat-header-cell *matHeaderCellDef>{{'marketplace.marketplace-table.buy-back' | translate}}</th>
      <td mat-cell *matCellDef="let row">{{row.buyback}}%</td>
    </ng-container>

    <!-- Invest Column -->
    <ng-container matColumnDef="invest">
      <th mat-header-cell *matHeaderCellDef class="invest">{{'marketplace.marketplace-table.invest' |
translate}}</th>
      <td mat-cell *matCellDef="let row" class="invest" (click)="$event.stopPropagation()">
        <!--apply contentx to template - ctx: this-->
        <ng-template [ngTemplateOutlet]="invest" [ngTemplateOutletContext]="{row: row, ctx: this}"></ng-
template>
      </td>
```

```
    </ng-container>

    <ng-container matColumnDef="expandedDetail">
      <td mat-cell *matCellDef="let row" [attr.colspan]="displayedColumns.length">
        <div class="example-element-detail"
            [@detailExpand]="row == expandedElement ? 'expanded' : 'collapsed'">
          <div class="layout table-expanded-row">
            <div class="layout row space-around">
              <div class="layout column">
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.status' | translate}}</div>
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.type' | translate}}</div>
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.balance' |
translate}}</div>
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.originator' |
translate}}</div>
              </div>

              <div class="layout column">
                <div class="layout row p-v-5 flex-end">
                  <ng-container [ngSwitch]="row.status">
                    <div *ngSwitchCase="'Active'"><app-tooltip [description]="'misc.tooltip.statistic-status-active' |
translate" [className]="'statistick'"><span class="overdue-status" [ngClass]="'active'"></span></app-
tooltip></div>
                    <div *ngSwitchCase="'Overdue'"><app-tooltip [description]="'misc.tooltip.statistic-status-
overdue' | translate" [className]="'statistick'"><span class="overdue-status"></span></app-tooltip></div>
                  </ng-container>
                </div>
                <div class="layout row p-v-5 flex-end">{{row.loanType}}</div>
                <div class="layout row p-v-5 flex-end">{{row.loanBalance}} {{row.shownCurrency}}</div>
                <div class="layout row p-v-5 flex-end">{{row.loanOriginator}}</div>
              </div>
              <div class="additional-border"></div>
              <div class="layout column">
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.expanded.next-
redemption-date' | translate}}</div>
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.expanded.next-
redemption-amount' | translate}}</div>
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.expanded.next-
redemption-number' | translate}}</div>
                <div class="layout row p-v-5 flex-start">{{'marketplace.marketplace-table.expanded.planned-
closure-date' | translate}}</div>
              </div>

              <div class="layout column">
                <ng-container *ngIf="row.nextRedemptionDate; else noData">
                  <div class="layout row p-v-5 flex-end">{{row.nextRedemptionDate | date:'dd.MM.yyyy'}}</div>
                </ng-container>
                <ng-container *ngIf="row.nextRedemptionAmount; else noData">
                  <div class="layout row p-v-5 flex-end">{{row.nextRedemptionAmount | digitsAfterDot:'2'}}
{{row.shownCurrency}}</div>
                </ng-container>
                <ng-container *ngIf="row.redemptionNumber && row.redemptionCount; else noData">
                  <div class="layout row p-v-5 flex-
end">{{row.redemptionNumber}}/{{row.redemptionCount}}</div>
                </ng-container>
                <ng-container *ngIf="row.plannedClosureDate; else noData">
                  <div class="layout row p-v-5 flex-end">{{row.plannedClosureDate | date:'dd.MM.yyyy'}}</div>
                </ng-container>
              </div>
              <ng-template #noData>
                <div class="layout row p-v-5 flex-end">-</div>
              </ng-template>
              <!--<div class=""-->
```

```
          </div>
        </div>
      </div>
    </td>
  </ng-container>

  <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
  <tr mat-row *matRowDef="let element; columns: displayedColumns;"
      class="example-element-row"
      [class.example-expanded-row]="expandedElement === element"
      (click)="expandedElement = expandedElement === element ? null : element">
  </tr>
  <tr mat-row *matRowDef="let row; columns: ['expandedDetail']" class="example-detail-row"></tr>
 </table>
 <div class="mobile-container">
  <ng-container *ngFor="let loan of loans">
   <div class="layout column mobile-margin">
    <article class="layout row space-between border-article">

     <div class="layout column flex-1">
      <div class="p-v-5 border-left">
       <ng-container [ngSwitch]="loan.country">
        <span *ngSwitchCase="'DE'"><span class="lang-icon lang-
de"></span>{{loan.innerLoanId}}</span>
        <span *ngSwitchCase="'UA'"><span class="lang-icon lang-
ru"></span>{{loan.innerLoanId}}</span>
        <span *ngSwitchCase="'Poland'"><span class="lang-icon pl"></span>{{loan.innerLoanId}}</span>
       </ng-container>
      </div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.originator' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.issue-date' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.interest-rate' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.buy-back' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.loan-amount' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.loan-balance' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.rating' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.status' | translate}}</div>
      <div class="p-5 border-left">{{'marketplace.marketplace-table.term' | translate}}</div>
      <div class="p-5 available-column">{{'marketplace.marketplace-table.available-for-investment' |
translate}}</div>

     </div>

     <div class="layout column flex-2">

      <div class="p-5 layout row flex-end border-right">{{loan.loanType}}</div>
      <div class="p-5 layout row flex-end border-right">{{loan.loanOriginator}}</div>
      <div class="p-5 layout row flex-end border-right">{{loan.issueDate | date:'dd.MM.yyyy'}}</div>
      <div class="p-5 layout row flex-end border-right">{{loan.interestRate}}%</div>
      <div class="p-5 layout row flex-end border-right">{{loan.buyback}}%</div>
      <div class="p-5 layout row flex-end border-right">{{loan.loanAmount}}
{{loan.shownCurrency}}</div>
      <div class="p-5 layout row flex-end border-right">{{loan.loanBalance}}
{{loan.shownCurrency}}</div>
      <div class="p-5 layout row flex-end border-right">{{loan.rating}}</div>
      <div class="p-5 layout row flex-end border-right">
       <ng-container [ngSwitch]="loan.status">
        <ng-container *ngSwitchCase="'Active'"><span class="overdue-status"
[ngClass]="'active'"></span></ng-container>
        <ng-container *ngSwitchCase="'Overdue'"><span class="overdue-status"></span></ng-container>
       </ng-container>
      </div>
      <div class="p-5 layout row flex-end border-right">{{loan.term}} d.</div>
```

```
          <div class="p-5 layout row flex-end available-column center">{{loan.availableForInvestment | digitsAfterDot:'2'}} {{loan.shownCurrency}}</div>

        </div>
       </article>

       <div class="layout row center-center">
        <ng-template [ngTemplateOutlet]="invest" [ngTemplateOutletContext]="{row: loan, ctx: this}"></ng-template>
       </div>

      </div>
     </ng-container>
   </div>
   <mat-paginator [length]="total" [pageSizeOptions]="[20,40,60]" showFirstLastButtons></mat-paginator>
  </div>
  <div class="no-loans" [ngClass]="{'show': (!loans || !loans.length) && !pending}">{{'marketplace.marketplace-table.no-loans' | translate}}</div>
</div>



<ng-template #invest let-loan="row" let-ctx="ctx">

  <ng-container *ngIf="!loan.onSC; else shoppingCartInput">

    <ng-container *ngIf="!loan.showInvestInput; else showInvestInput">
     <div @animateForm class="mobile-border-button">
      <button mat-raised-button color="primary" (click)="ctx.invest(loan)">{{'marketplace.marketplace-table.invest' | translate}}</button>
     </div>
    </ng-container>

    <ng-template #showInvestInput>
     <form  @animateForm [formGroup]="formInvest" (ngSubmit)="ctx.onInvestSend()" class="layout row inline">
      <app-input-amount-tooltip [min]="1" [max]="loan.maxAvailableSum" [currency]="loan.currency" [amount]="loan.amount" formControlName="invest" (onHitEnter)="ctx.onInvestSend(loan)"></app-input-amount-tooltip>
      <i class="fa fa-shopping-cart" aria-hidden="true" (click)="ctx.onInvestSend(loan)"></i>
     </form>
    </ng-template>

  </ng-container>

  <ng-template #shoppingCartInput>

    <ng-container *ngIf="!loan.edit; else editForm">

     <div @animateForm class="layout row inline mobile-end">
      <div class="sum_amount"><strong>{{loan.amount}} {{loan.shownCurrency}}</strong></div>
      <i class="fa fa-pencil-square-o" aria-hidden="true" (click)="ctx.showEditForm(loan)"></i>
      <i class="fa fa-times" aria-hidden="true" (click)="ctx.deleteLoan(loan)"></i>
     </div>

    </ng-container>

    <ng-template #editForm>

      <form @animateForm [formGroup]="formEdit" class="layout row inline" (ngSubmit)="ctx.editValue(loan)">
```

```
        <app-input-amount-tooltip [min]="1" [max]="loan.maxAvailableSum" [currency]="loan.currency"
[amount]="loan.amount" formControlName="edit" (keyup.enter)="ctx.editValue(loan)"></app-input-amount-
tooltip>
        <i class="fa fa-check-square" aria-hidden="true" (click)="ctx.editValue(loan)"></i>
        <i class="fa fa-times" aria-hidden="true" (click)="ctx.deleteLoan(loan)"></i>
       </form>

    </ng-template>

  </ng-template>

</ng-template>

:host {display: flex; flex: 1 0 100%; position: relative; }
.pl {background:url(../../../../assets/img/pl.svg);}
.hide {display: none; }
.no-loans {display: none; text-align: center;}
.show {display: block; }
.example-container {width: 100%; }
.example-loading-shade {
  position: absolute;
  top: 0;
  left: 0;
  bottom: 56px;
  right: 0;
  background: rgba(0, 0, 0, 0.15);
  z-index: 1;
  display: flex;
  align-items: center;
  justify-content: center;
}
.mobile-container{ display: none;}
table { display: table; width: 100%;
  th, td {text-align: center;}
}
th {font-size: 16px;}
th.availableForInvestment {font-size: 14px; max-width: 100px; }
.invest { width: 150px; }
.lang-icon {
  margin-left: 5px; box-shadow:0 0 5px rgba(0,0,0,.3) inset; border-radius:50%;
  width:22px; height:22px; line-height:22px; vertical-align:middle; display:inline-block;
}
.mat-raised-button.mat-primary {height: 30px; }
.invest-form{ display: inline-flex; }
.fa-chevron-circle-up, .fa-chevron-circle-down {color: #4e3796; }
.fa-shopping-cart, .fa-pencil-square-o, .fa-times, .fa-check-square {padding: 5px; margin: auto; color: white;
background-color: #4e3796; border-radius: 5px;}
.fa-times {background-color: unset; color: #4e3796;}
.show_sum {display: inline-flex; }
.sum_amount{width: 50px; text-align: center; margin: auto; }
tr.example-detail-row {height: 0; }
tr.example-element-row:not(.example-expanded-row):hover {background: #f5f5f5; }
tr.example-element-row:not(.example-expanded-row):active {background: #efefef; }
tr.example-expanded-row {background: #efefef;}
.example-element-row td {border-bottom-width: 0; }
.example-element-detail {overflow: hidden; display: flex; }
.table-expanded-row {width: 100%; }
.overdue-status {border-radius:50%; width:12px; height:12px; display:inline-block; background-color:red; box-
shadow:0 0 5px rgba(0,0,0,.3) inset;
  &.active {background-color:green;}
  &.closed {background-color: grey;}
}
//.animation-form {transition: 225ms cubic-bezier(0.4, 0.0, 0.2, 1); }
```

```
.additional-border {width: 1px;height: 100px;background: linear-gradient( 180deg, rgb(255,255,255) 0%, rgb(143,121,208)  50% , rgb(255,255,255)  100%);}
@media screen and (max-width: 991px) {
  th {font-size: 12px;}
  th.availableForInvestment {font-size: 12px;}
  .lang-icon {
    width: 14px;
    height: 14px;
    line-height: 14px;
  }
}
@media screen and (max-width: 767px){
  table { display: none; }
  .mobile-container{display: block; }
  .lang-icon { width: 16px; height: 16px; line-height: 16px; margin: 0 5px; }
  .mat-raised-button.mat-primary {width: 100%; }
  .mobile-border-button {border: 2px solid #a7a3a3; border-top: none; padding: 10px; width: 100%; }
  form, app-input-amount-tooltip {width: 100%; }
  form {border: 2px solid #a7a3a3; border-top: none; padding: 5px;}
  .fa { margin: 0 10px; padding: 7px; font-size: 20px; }
  .mobile-end {justify-content: flex-end; width: 100%; border: 2px solid #a7a3a3; border-top: none; padding: 5px; }
  .sum_amount { margin: 5px; font-size: 18px; }
  .mobile-margin {margin: 10px 0!important; }
  .mat-elevation-z8 {box-shadow: none; }
  .border-article {border: 2px solid #a7a3a3; }
  .border-left {border-right: none; border-bottom: 2px solid #a7a3a3; height: 30px;}
  .border-right {border-left: none; border-bottom: 2px solid #a7a3a3; height: 30px;}
  .overdue-status {width: 16px; height: 16px; }
  .available-column{height: 40px;}
}
@media screen and (max-width: 480px) { ::ng-deep.mat-paginator {.mat-paginator-range-label { margin: 0!important; } }}

import {AfterViewInit, Component, OnDestroy, OnInit, ViewChild} from '@angular/core';
import {FormBuilder, FormControl, FormGroup} from '@angular/forms';
import {MarketplaceService} from '@app/marketplace/services/marketplace.service';
import {delay, take, takeUntil} from 'rxjs/operators';
import {Store} from '@ngrx/store';
import * as fromCore from '@core/reducers';
import * as Actions from '../../actions/marketplace.actions';
import * as fromMarketplace from '../../reducers';
import {combineLatest, Subject} from 'rxjs';
import {MatPaginator, MatSort} from '@angular/material';
import {animate, state, style, transition, trigger} from '@angular/animations';
import {environment} from '../../../../environments/environment';

@Component({
  selector: 'app-marketplace-table',
  templateUrl: './marketplace-table.component.html',
  styleUrls: ['./marketplace-table.component.scss'],
  animations: [
    trigger('detailExpand', [
      state('collapsed', style({height: '0px', minHeight: '0', display: 'none'})),
      state('expanded', style({height: '*'})),
      transition('expanded <=> collapsed', animate('400ms cubic-bezier(0.4, 0.0, 0.2, 1)')),
    ]),
    trigger('animateForm', [
      transition(':enter', [
        style({ opacity: 0 }),
        animate('500ms cubic-bezier(0.4, 0.0, 0.2, 1)', style({ opacity: 1 })),
      ]),
    ])
```

```
  ]
})
export class MarketplaceTableComponent implements OnDestroy, AfterViewInit, OnInit {
    public pending: boolean = true;
    private unsubscribe$ = new Subject<void>();
    public total: number;
    activeFilters;
    loans;
    formInvest: FormGroup;
    formEdit: FormGroup;
    sortDirection: string;
    sortBy: string;
    currentPage: number;
    shoppingCart;
    loanTrackerSub;
    // public pageSize: Array<number> = [20, 40, 60];
    public displayedColumns: string[] = [
      'loanId',
      'rating',
      'issueDate',
      'interestRate',
      'loanAmount',
      'term',
      'availableForInvestment',
      'buyback',
      'invest'
    ];
    @ViewChild(MatPaginator) paginator: MatPaginator;
    @ViewChild(MatSort) sort: MatSort;
    constructor(
        private fb: FormBuilder,
        private marketplaceService: MarketplaceService,
        private store: Store<fromCore.State>
    ) {
        // this.store.select(fromMarketplace.selectSortArray).pipe(takeUntil(this.unsubscribe$)).subscribe(res => {
        //     if (res.length) {
        //         this.sortBy = res[0];
        //         this.sortDirection = res[1];
        //     } else {
        //         this.sortBy = undefined;
        //         this.sortDirection = 'desc';
        //     }
        // });
        // this.store.select(fromMarketplace.selectCurrentPage)
        //   .pipe(takeUntil(this.unsubscribe$))
        //   .subscribe(res => this.currentPage = res);
        this.store.select(fromMarketplace.selectActiveFilters).pipe(takeUntil(this.unsubscribe$)).subscribe(filters
=> {
          console.log('FILTERS');
            console.log(filters);
            if (filters.length) {
                this.pending = true;
                this.activeFilters = filters;
                console.log(filters);
                this.selectLoans();
            } else {
                this.loans = [];
                this.pending = false;
            }
        });
        this.store.select(fromMarketplace.selectShoppingCart).pipe(takeUntil(this.unsubscribe$)).subscribe(SC =>
{
          // if (!this.shoppingCart) {
```

```
        //   this.shoppingCart = SC;
        // }
        console.log('SHOPPING CART');
          this.shoppingCart = SC;
          if (this.shoppingCart.length) {
            this.selectLoans();
          }
        });

this.store.select(fromMarketplace.selectWebsocketSubscription).pipe(takeUntil(this.unsubscribe$)).subscribe(fla
g => {
          console.log(flag);
          if (flag) {
             this.loanTrackerSub = this.marketplaceService.loanTracker.subscribe( res => {
                if (this.shoppingCart.some(elem => elem.loanId === res['eur'].loanId)) {
                   this.store.dispatch(new Actions.LoadShoppingCart());
                }
                console.log(this.loans[0].currency);
                console.log(res);
                const index = this.loans.findIndex(elem => elem.loanId ===
res[elem.currency.toLowerCase()].loanId);
                console.log(index);
                console.log(this.loans[index]);
                this.loans[index].term = res[this.loans[index].currency.toLowerCase()].term;
                this.loans[index].loanAmount = res[this.loans[index].currency.toLowerCase()].loanAmount;
                this.loans[index].loanBalance = res[this.loans[index].currency.toLowerCase()].loanBalance;
                this.loans[index].availableForInvestment =
res[this.loans[index].currency.toLowerCase()].availableForInvestment;
                if (this.loans[index].availableForInvestment > 100) {
                   this.loans[index].maxAvailableSum = 100;
                } else {
                   this.loans[index].maxAvailableSum = this.loans[index].availableForInvestment;
                }
                console.log(this.loans[index]);
             });
          } else {
             this.loanTrackerSub.unsubscribe();
          }
        });
        this.formInvest = fb.group({invest: new FormControl()});
        this.formEdit = fb.group({edit: new FormControl()});
    }
    ngOnInit() {
      combineLatest(
        this.store.select(fromMarketplace.selectCurrentPage),
        this.store.select(fromMarketplace.selectSortArray),
        this.store.select(fromMarketplace.selectPageSize)
      )
      .pipe(takeUntil(this.unsubscribe$))
      .subscribe(combined => {
        console.log('COMBINED PAGINATOR', combined);
        [this.paginator.pageIndex, [this.sort.active, this.sort.direction], this.paginator.pageSize] = combined;
      });
      // this.store.select(fromMarketplace.selectCurrentPage)
      //   .pipe(take(1))
      //   .subscribe(res => this.paginator.pageIndex = res);
      // this.store.select(fromMarketplace.selectSortArray)
      //   .pipe(take(1))
      //   .subscribe(res => {
      //     this.sort.active = res[0];
      //     this.sort.direction = res[1];
      //   });
    }
```

```
ngAfterViewInit() {
  this.sort.sortChange.subscribe(() => {
    this.paginator.pageIndex = 1;
    console.log(this.sort.active, this.sort.direction);
    if (!this.sort.active || !this.sort.direction) {
      this.store.dispatch(new Actions.SetSortArray([]));
      console.log('Paginator', this.paginator);
      this.store.dispatch(new Actions.SetCurrentPage(0));
      this.selectLoans();
      return;
    }
    const sortArr = [this.sort.active, this.sort.direction];
    this.store.dispatch(new Actions.SetSortArray(sortArr));
    this.store.dispatch(new Actions.SetCurrentPage(0));
    console.log('SORT1');
    this.selectLoans();
  });
  this.paginator.page
    .pipe(delay(0), takeUntil(this.unsubscribe$))
    .subscribe(_ => {console.log('Paginator1', this.paginator); this.setPage(this.paginator); });
  // combineLatest(
  //   this.store.select(fromMarketplace.selectActiveFilters),
  //   this.store.select(fromMarketplace.selectShoppingCart)
  // ).pipe(takeUntil(this.unsubscribe$))
  //   .subscribe(combined => {
  //     console.log('Combined', combined);
  //     const [filters, SC]: any[] = combined;
  //     if (this.activeFilters === filters && this.shoppingCart === SC) { return; }
  //     if (filters.length) {
  //       console.log('SELECT LOANS');
  //       this.pending = true;
  //       this.activeFilters = filters;
  //       console.log(filters);
  //       this.shoppingCart = SC;
  //       this.selectLoans();
  //     }
  //     // if (this.shoppingCart.length) {
  //     //   this.selectLoans();
  //     // }
  //   });
}
get edit() {return this.formEdit.get('edit').value; }
onSetPage(pager) {
    this.pending = true;
    this.currentPage = pager.currentPage;
    console.log('current Page: ', pager.currentPage);
    this.store.dispatch(new Actions.SetCurrentPage(pager.currentPage));
    this.selectLoans();
}
setPage(paginator) {
  // this.currentPage = pageIndex;
  if (!environment.isServer) {window.scrollTo(0, 0); }
  console.log('Set Page', paginator);
  this.store.dispatch(new Actions.SetCurrentPage(paginator.pageIndex));
  this.store.dispatch(new Actions.SetPageSize(paginator.pageSize));
  this.selectLoans();
}
 mapLoansForView(data) {
  console.log('Map Loans', data.total);
   this.total = data.total;
   this.loans = data.loans;
   this.loans.forEach(res => {
       res.show = false;
```

```
      res.showInvestInput = false;
      res.showFastBuyInput = false;
      res.edit = false;
      if (res.availableForInvestment > 100) {
         res.maxAvailableSum = 100;
      } else {
         res.maxAvailableSum = res.availableForInvestment;
      }
      switch (res.currency) {
        case 'EUR': res.shownCurrency = '€'; break;
        case 'PLN': res.shownCurrency = 'zł'; break;
      }
      if (this.shoppingCart.length) {
         this.shoppingCart.forEach(elemSC => {
            if (elemSC.loanId === res.loanId && elemSC.currency === res.currency) {
               res.onSC = true;
               res.amount = elemSC.amount;
               // if(elemSC.amount + res.availableForInvestment > 100) {
               //     res.maxAvailableSum = 100;
               // } else {
               //     res.maxAvailableSum = elemSC.amount + res.availableForInvestment;
               // }
               res.changeValue = elemSC.amount;
               res.maxAvailableSum = elemSC.maxAvailableSum;
               res.available = elemSC.available;
               res.locked = elemSC.locked;
            }
         });
      }
    });
    // this.loans = new MatTableDataSource(this.loans);
    // this.loans.paginator = this.paginator;
  }
changeSortDirection(field_name) {
    this.sortBy = field_name;
    (this.sortDirection === 'desc') ? this.sortDirection = 'asc' : this.sortDirection = 'desc';
    const sortArr = [field_name, this.sortDirection];
    this.store.dispatch(new Actions.SetSortArray(sortArr));
    this.selectLoans();
}

selectLoans() {
  if (this.activeFilters) {
    this.marketplaceService.setParamsForFilteredLoans(this.activeFilters).pipe(take(1)).subscribe(data => {
      console.log('select Loans');
        this.total = data.total;
        this.mapLoansForView(data);
        this.pending = false;
    });
  }
}


ngOnDestroy() {
    this.unsubscribe$.next();
    this.unsubscribe$.complete();
    // this.subsArray.forEach(res => res.unsubscribe());
    this.loanTrackerSub.unsubscribe();
    // this.subscribe.unsubscribe();
    // this.currentPageSubscribe.unsubscribe();
}

showDetail(elem) {
```

```
      if (elem.show) {
        elem.show = false;
      } else {
        this.loans.forEach((i) => i.show = false);
        elem.show = true;

      }
    }
    hideTooltip(elem) {
      console.log('Click outside');
      console.log(elem.showTooltip);
      elem.showTooltip = false;
    }
    invest(elem) {
     console.log(elem);
     this.loans.forEach(res => {
      res.showInvestInput = false;
      res.showFastBuyInput = false;
     });
     elem.showInvestInput = true;
     this.formInvest.setValue({invest : ''});
    }
    // fastBuy(elem) {
    //     this.loans.forEach(res => res.showInvestInput = false);
    //     this.loans.forEach( res => res.showFastBuyInput = false);
    //     elem.showFastBuyInput = true;
    //     this.formFastBuy.setValue({fastBuy: ''});
    // }
    closeInvestForm(elem) {
      elem.showInvestInput = false;
      this.formInvest.setValue({invest : ''});
    }
    onInvestSend(elem) {
      console.log(this.formInvest.controls.invest.value);
      if (this.formInvest.controls.invest.value !== '') {
        if (this.formInvest.controls.invest.value > elem.maxAvailableSum) {
          this.formInvest.setValue({invest : elem.maxAvailableSum});
        }
        this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(false));
        console.log(this.formInvest.controls.invest.value);
        elem.showInvestInput = false;
        this.pending = true;
        this.marketplaceService.addLoanToShoppingCart(elem.loanId, this.formInvest.controls.invest.value,
elem.currency)
          .pipe(take(1))
          .subscribe(res => {
            if (res === 200) {
              this.store.dispatch(new Actions.LoadShoppingCart());
            } else {
              this.pending = false;
              this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(true));
            }
        });
      }
    }
    showEditForm(elem) {
      this.loans.forEach( res => {
        res.edit = false;
        res.showFastBuyInput = false;
      });
      elem.edit = true;
      this.formEdit.setValue({edit: elem.amount});
    }
```

```
editValue(elem) {
    console.log(this.edit);
    if (!elem.available) {
        console.log('Unavailable');
        console.log(this.edit, elem.maxAvailableSum);
        console.log(elem);
        if (+this.edit > elem.maxAvailableSum) {
            console.log('1 Step');
            elem.amount = elem.maxAvailableSum;
            this.formEdit.setValue({edit: elem.amount});
            return;
        }
        if (+this.edit === elem.amount) {
            console.log('2 Step');
            this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(false));
            this.marketplaceService.addLoanToShoppingCart(elem.loanId, +this.edit, elem.currency)
              .pipe(take(1))
              .subscribe(res => {
                if (res === 200) {
                    this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(false));
                    this.store.dispatch(new Actions.LoadShoppingCart());
                } else {
                    this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(true));
                }
            });
            return;
        }
    }
    if (this.edit === elem.amount) {elem.edit = false; }
    if (this.edit !== elem.amount) {
        this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(false));
        this.pending = true;
        if (this.edit === '') {
            this.marketplaceService.addLoanToShoppingCart(elem.loanId, 1, elem.currency)
              .pipe(take(1))
              .subscribe(res => {
                if (res === 200) {
                    this.store.dispatch(new Actions.LoadShoppingCart());
                } else {
                    this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(true));
                }
            });
        } else {
            this.marketplaceService.addLoanToShoppingCart(elem.loanId, +this.edit, elem.currency)
              .pipe(take(1))
              .subscribe(res => {
                if (res === 200) {
                    this.store.dispatch(new Actions.LoadShoppingCart());
                } else {
                    this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(true));
                }
            });
        }
    }
}
deleteLoan(elem) {
    this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(false));
    this.pending = true;
    this.marketplaceService.deleteLoanFromShoppingCart(elem.loanId, elem.currency)
      .pipe(take(1))
      .subscribe(res => {
        console.log(res);
        if (res === 200) {
```

```
            this.store.dispatch(new Actions.LoadShoppingCart());
            this.selectLoans();
          } else {
            this.pending = false;
            this.store.dispatch(new Actions.SetWebSocketSubscriptionFlag(true));
          }
        });
    }
}
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ PEER-TO-PEER LENDING ПЛАТФОРМИ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.М. Олещенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ В.А. Тимошенко

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для Peer-To-Peer Lending платформи, яке являє собою web-сайт, створений з використанням технологій Spring Boot та Angular, серверного рендерингу, i18n.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) забезпечення належного рівня безпеки даних;
2) коректність роботи з базою даних;
2) тестування верстки;
4) тестування кросбраузеронсті;
5) usability тестування;
6) тестування продуктивності;
7) відповідність вимогам технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) димове тестування (Smoke testing);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність програмного забезпечення перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування кросбраузерності в різних web-браузерах;
5) тестування верстки;
6) тестування зручності використання;
7) тестування інтерфейсу;
8) тестування продуктивності
9) тестування стабільності роботи при різних умовах;

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ PEER-TO-PEER LENDING ПЛАТФОРМИ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.М. Олещенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ В.А. Тимошенко

2019

ЗМІСТ

1. Опис структури програмного забезпечення

Програмне забезпечення для Peer-To-Peer Lending платформи складається з набору модулів. Основне призначення – створення інвестицій у споживчі кредити. Структура програмного забезпечення складається з таких модулів:

- модуль реєстрації та авторизації;
- модуль автоматичного створення інвестиції;
- модуль ринку кредитів;
- модуль підтвердження створення інвестиції;
- модуль портфоліо;
- модуль транзакцій;
- модуль ідентифікації;
- модуль персонального кабінету користувача.

2. Опис роботи з модулями програмного забезпечення

Модуль реєстрації (рис.1) складається з набору полів, які користувачу необхідно заповнити:

- поле електронної адреси;
- поле імені;
- поле прізвища;
- поле країни в якій проживає користувач;
- поле паролю.

Якщо користувач вводить не правильну інформацію, то поле буде виділено червоним, і з’явиться повідомлення про помилку. Після того як користувач заповнив всі поля він натискає на кнопку реєстрації і йому відправляється лист на електронну пошту. Для того щоб закінчити процес реєстрації користувачу необхідно підтвердити свою електронну адресу, натиснувши на посилання в листі.

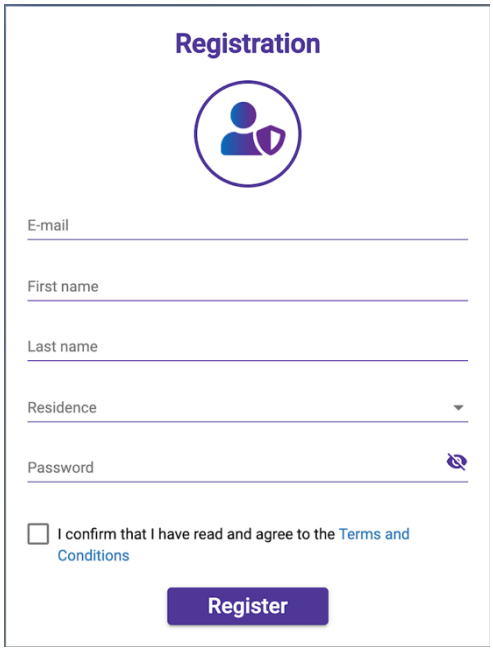


Рис. 1. Модуль реєстрації

Після успішної реєстрації користувач потрапляє у модуль ідентифікації(рис. 2).

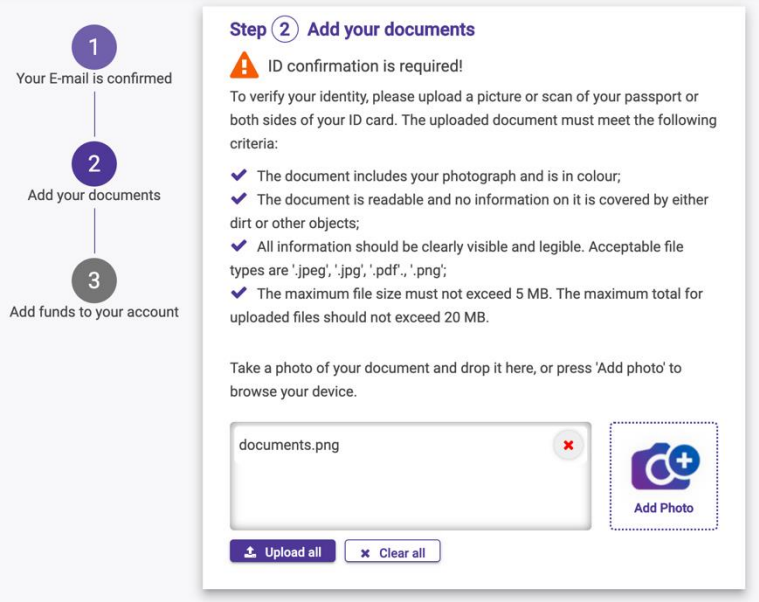


Рис. 2. Модуль ідентифікації

Для того щоб пройти ідентифікацію йому необхідно надіслати свої документи у одному з форматів: '.jpeg', '.jpg', '.pdf', '.png', розмір файлу не повинен перевищувати 5 мегабайт. Якщо користувач завантажує декілька

документів, їх сумарний розмір не повинен перевищувати 20 мегабайт. Після завантаження документів у модулі ідентифікації користувач зможе побачити свої документи, які він завантажив. Потім користувач потрапляє у модуль введення коштів. Для введення коштів користувачу необхідно здійснити переведення коштів на банківські реквізити, обов'язково вказавши свій id в системі. Банківський рахунок з якого прийшли кошти буде закріплений за користувачем. Виведення коштів з платформи буде можливим лише на цей банківський рахунок, бо цього вимагає європейське законодавство.

Після введення коштів користувач потрапляє в модуль автоматичного інвестування (рис. 3).

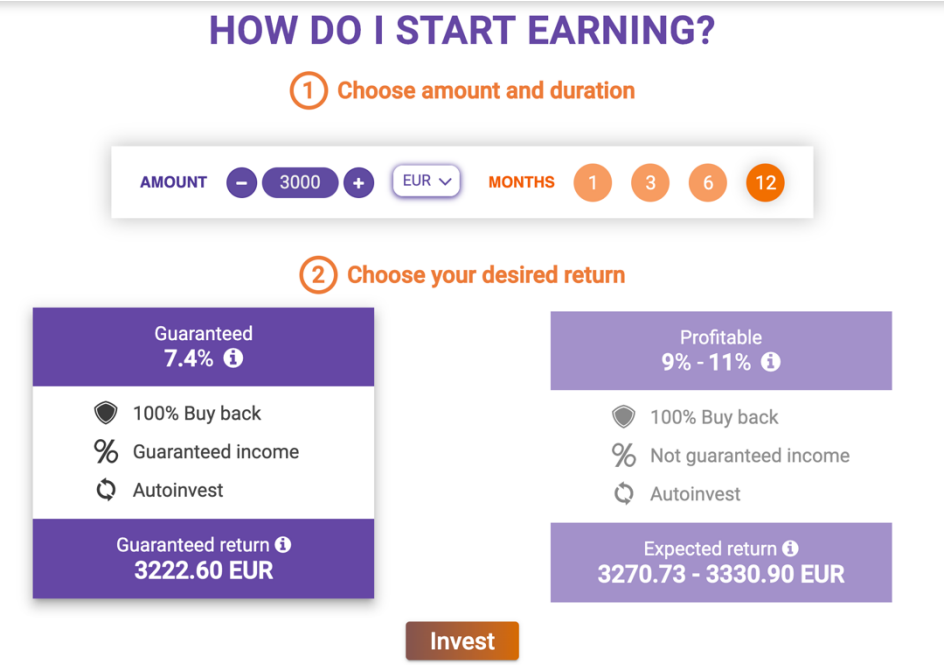


Рис. 3. Модуль автоматичного інвестування

У цьому модулі користувач може вибрати одну з двох стратегій інвестування, суму інвестиції, а також термін на який він хоче інвестувати кошти. На основі цих параметрів відбувається прогнозування очікуваного прибутку. Коли користувач натискає на кнопку інвестування, він переходить до підтвердження інвестиції (рис. 4). У цьому модулі користувач

має змогу ще раз перевірити умови інвестування і ввести свій пароль, якщо він хоче інвестувати кошти. Також у модулі відбувається перевірка доступних коштів користувача і суми на яку він хоче інвестувати. Якщо сума доступних коштів менша від суми на яку користувач хоче інвестувати, то він отримає попередження, що у нього недостатньо коштів для створення інвестиції, і якщо він створити таку інвестицію, то у нього буде 24 години для того щоб внести необхідну суму на платформу, в іншому разі інвестиція буде скасована.

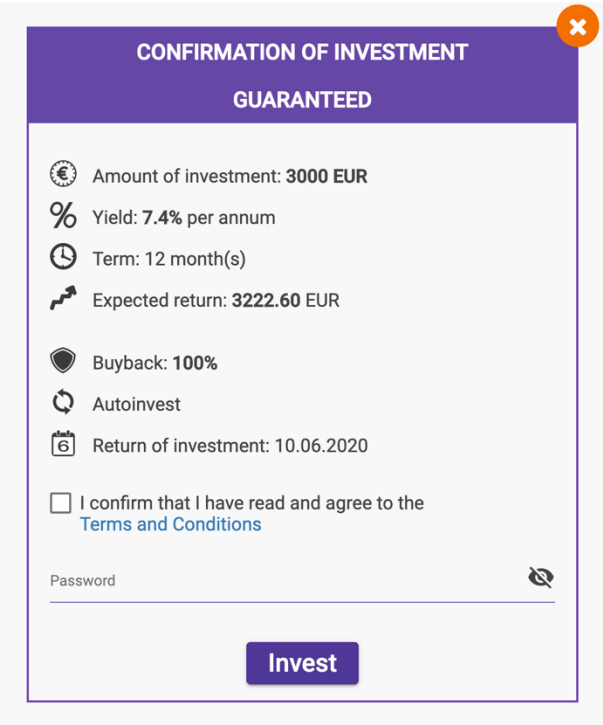


Рис. 4. Модуль підтвердження інвестиції

Після створення інвестиції користувач потрапляє в портфоліо, де він може переглянути свої інвестиції (рис. 5). Вся інформація поділена по трьом категоріям:

- категорія загальної інформації;
- категорія періоду інвестиції;
- категорія суми інвестиції.

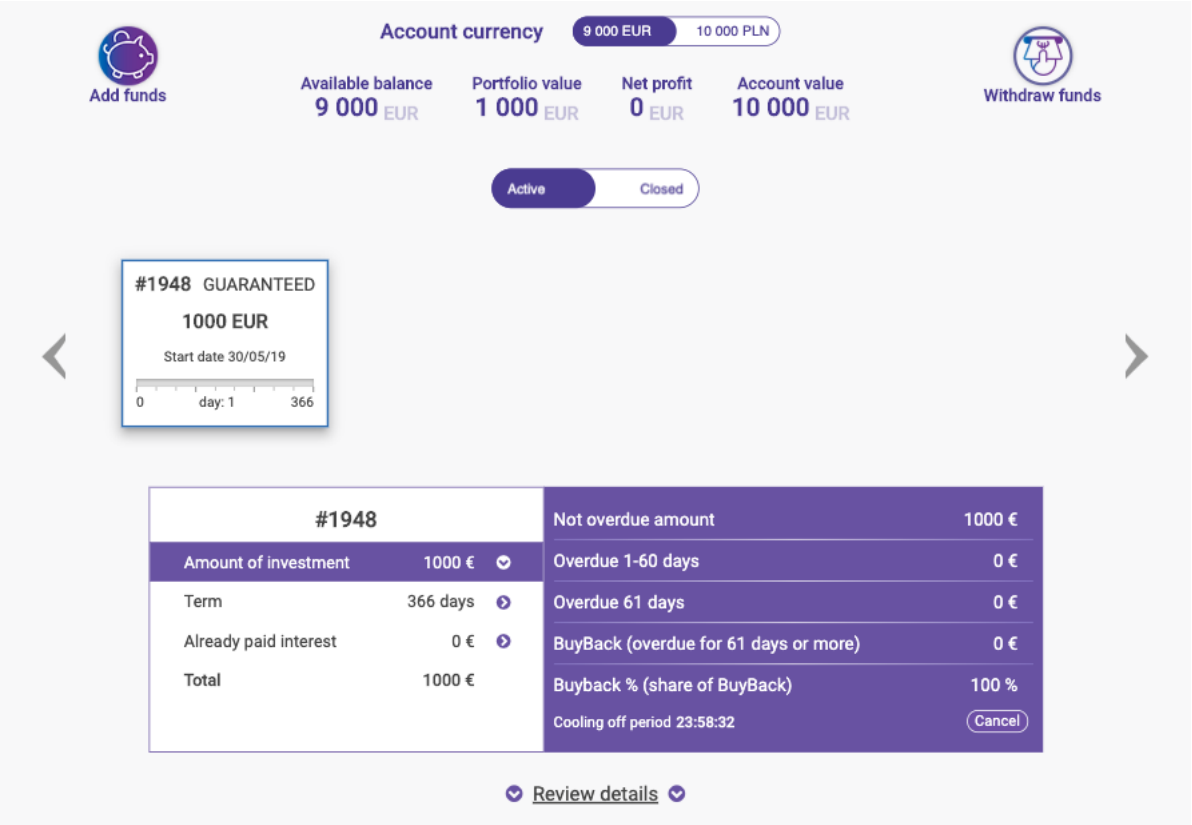


Рис. 5. Модуль портфоліо

У категорія періоду інвестиції користувач може закрити достроково інвестицію. У цьому разі 1% від суми інвестиції буде утриманий на користь платформи. Для підтвердження виходу з інвестиції користувач повинен ввести свій пароль. У разі якщо інвестиція була розміщена менше 24 годин тому, користувач має змогу закрити інвестицію безкоштовно. Кнопка відміни інвестиції, якій менше 24 годин розташована в категорії загальної інформації, а також біля неї іде таймер 24 годин з моменту створення інвестиції.

Користувач може створити інвестицію скориставшись ринком кредитів (рис. 6). Для цього йому необхідно самостійно вибрати кредити в які він хоче інвестувати. Так як присутня велика кількість кредитів, то для спрощення пошуку кредитів, у які користувач захоче інвестувати, йому доступні фільтри:

- валюта;

- ким виданий кредит;
- дата видачі кредиту;
- дата виплати кредиту;
- країна в якій видано кредит;
- рейтинг позичальника;
- відсоток очікуваного прибутку.

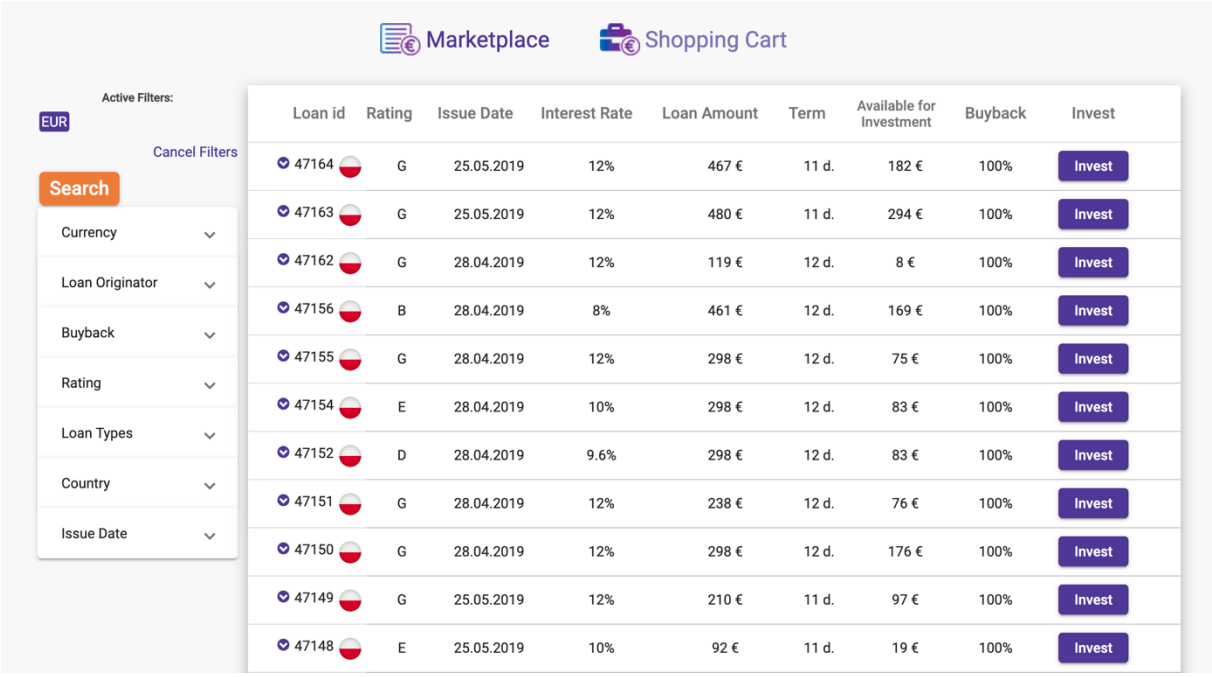


Рис. 6. Модуль ринку кредитів

Користувач може інвестувати в один кредит не більше ніж 100 євро. Це робить з метою диверсифікації ризику неповернення. Після вибору кредитів в які користувач захотів інвестувати, він потрапляє в корзину (рис. 7), де знаходяться всі кредити, в які він захотів інвестувати кошти. В корзині користувач має змогу ще раз перевірити кожний кредит, а також відредагувати суму інвестиції по кожному з кредитів.

Коли користувач перевірив всю інформацію, він натискає на кнопку покупки, і переходить в модуль підтвердження інвестиції. Після

підтвердження інвестиції користувач потрапляє в портфоліо. Так як користувач сам створював інвестицію, то він не може достроково її закрити.

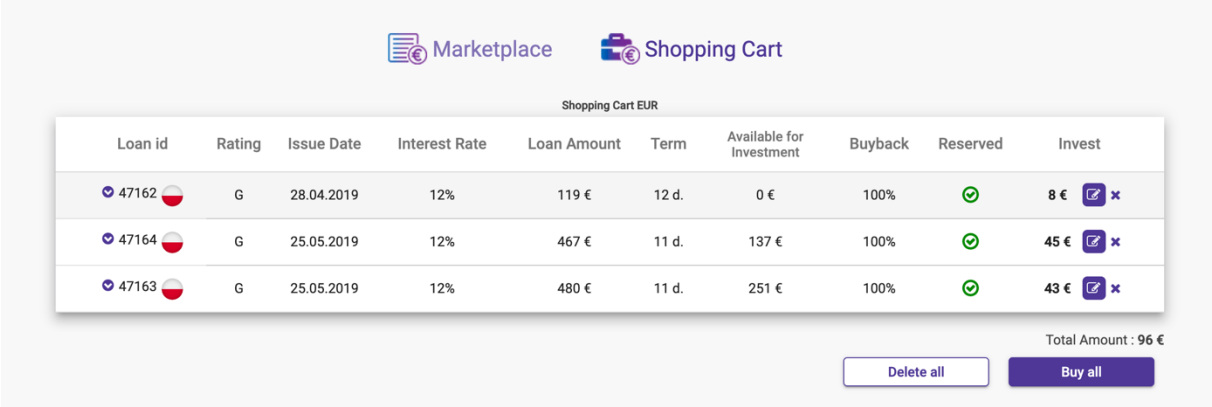


Рис. 7. Модуль корзини

Знаходячись в портфоліо користувач має змогу переглянути детальну інформацію по інвестиції – для цьому він переходить в модуль статистики (рис. 8). У цьому модулі користувач бачить детальну інформацію по кожному з кредитів, які знаходяться в інвестиції.

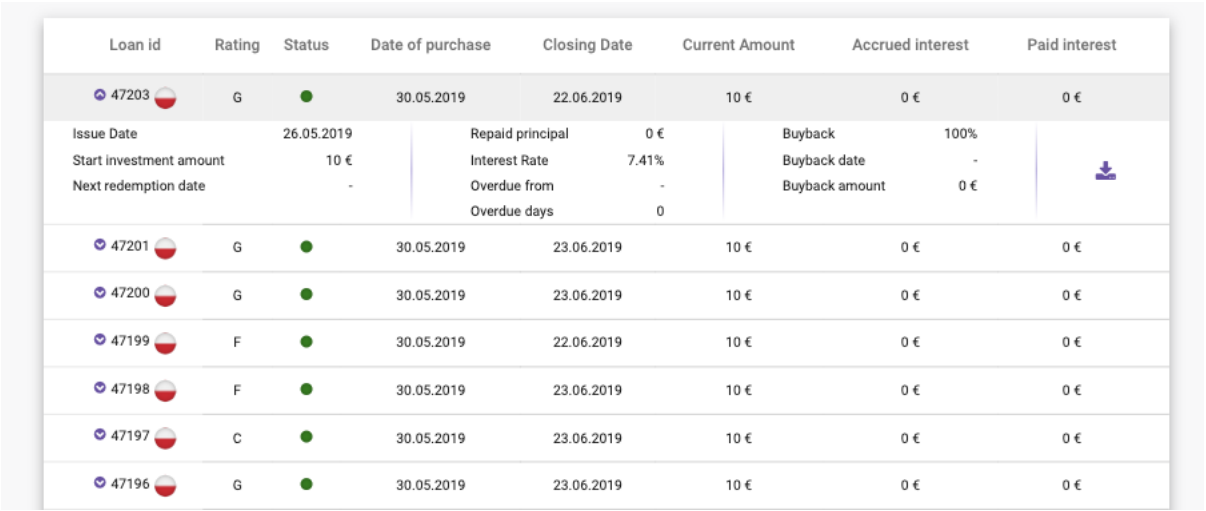


Рис. 8. Модуль статистики

Користувач може переглянути свої транзакції, для цього йому необхідно перейти в модуль транзакцій (рис. 9), і вибрати період за який він

хоче переглянути транзакції. Для зручності перегляду транзакцій йому доступні додаткові фільтри:

– фільтер валюти;

– фільтер типу транзакції;

– фідьтер по інвестиціям.

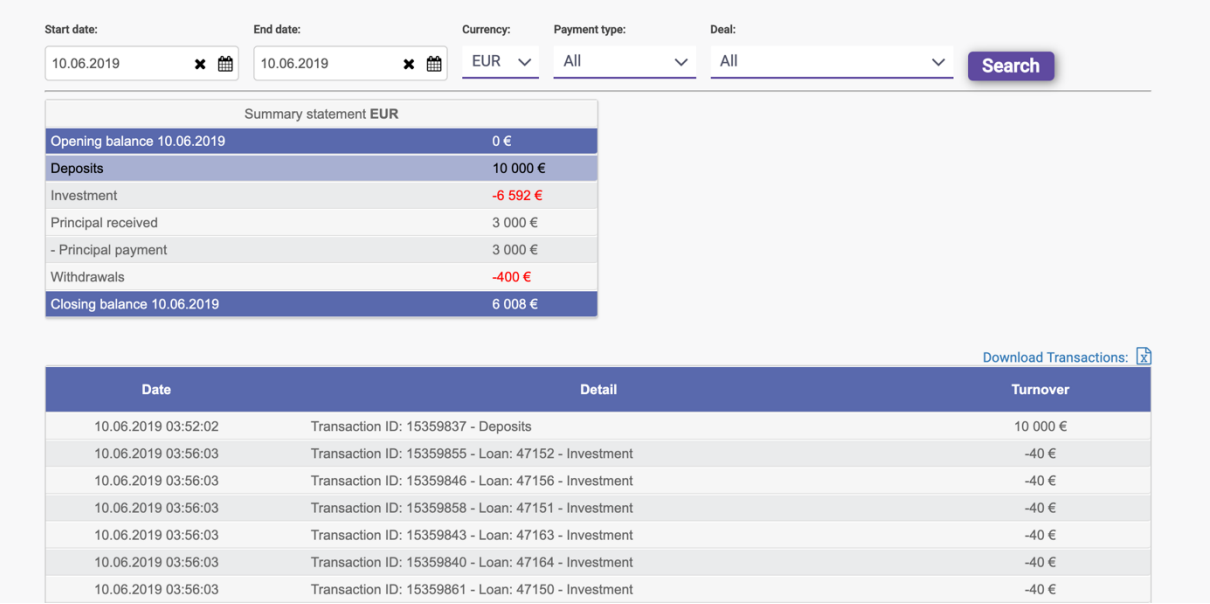


Рис. 9. Модуль транзкацій

Після введення вибору періоду і налаштування фільтрів користувачеві відображаються його транзакції, загальна інформація по транзакціям за вибраний період, а також користувач має змогу завантажити список транзакцій.

Для редагування персональної інформації, користувачеві необхідно перейти у модуль особистого кабінету (рис. 10). В особистому кабінеті користувач має редагувати песональну інформацію, змінити пароль, а також переглянути свої документи, які він надсилав для ідентифікації.

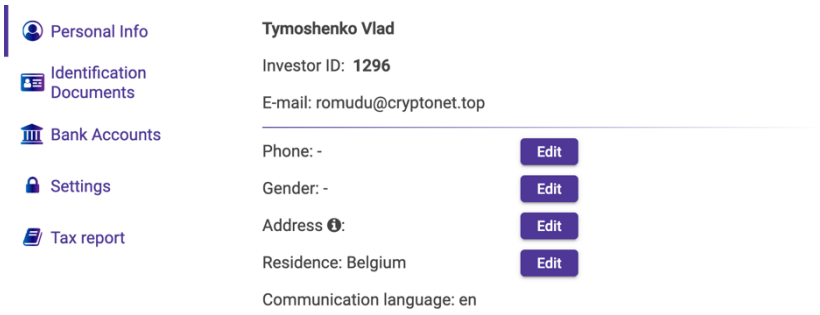


Рис. 10. Модуль особистого кабінету

Коли користувач захоче закінчити роботу з системою, його необхідно натиснути на кнопку виходу з системи (рис. 11), після цього він потрапить на головну сторінку.

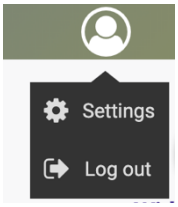


Рис. 11. Кнопка виходу з системи

Якщо користувач має акаунт в системі, то для авторизації, йому необхідно перейти у модуль авторизації (рис. 12), де йому необхідно ввести свою електронну пошту та свій пароль. Якщо користувач вводить не правильну інформацію, то йому буде показано помилку. В разі успішної авторизації користувач переходить до модулю автоматичного інвестування.

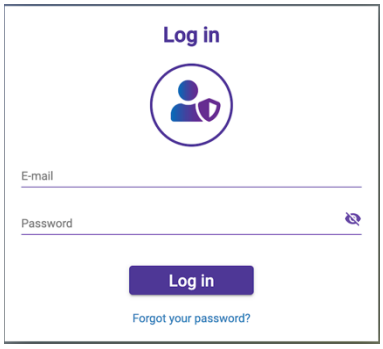


Рис. 12. Модуль авторизації