Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____*Комп'ютерних наук*_____
(повна назва)
Кафедра_____*Системотехніки*_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Розробка компонентів системи нетворкінгу пошуку студентів-кофаундерів для заснування стартапів
(тема)

Виконав:
студент _2__ курсу, групи _*ІТПм-22-2*
_____*Нероба Ю. Р.*_____
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки* __
(код і повна назва спеціальності)
Тип програми __освітньо-професійна_____
Освітня програма _«*Інформаційні технології проектування»*_____
(повна назва освітньої програми)

Керівник _____*проф. Міщеряков Ю. Р.*_____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ __*Гребеннік І.В.*__
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет_____*Комп'ютерних наук*_____

Кафедра_____*Системотехніки*_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність_____*122 Комп'ютерні науки*_____
(код і повна назва)

Тип програми _____освітньо-професійна_____

Освітня програма_____«*Інформаційні технології проектування»*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«___» _____20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____*Неробі Юлії Русланівні*_____
(прізвище, ім'я, по батькові)

1. Тема роботи: *Розробка компонентів системи нетворкінгу пошуку студентів-кофаундерів для заснування стартапів*
затверджена наказом по університету від ___*20.11.*____2023 р. №____*1373Ст*____
2. Термін подання студентом роботи до екзаменаційної комісії: 23.02.2024 р
3.Вихідні дані до роботи: *Розробити компоненти системи нетворкінгу пошуку студентів-кофаундерів для заснування стартапів. Інтегрувати AI для пошуку студентів-кофаундерів для заснування стартапів*

4. Перелік питань, що потрібно опрацювати в роботі: *Вступ 1 Аналіз предметної області 1.1 Опис предметної області 1.2 Дослідження впливу технологій 1.3 Аналіз аналогічних систем 1.4 Бачення та сфера застосування 1.4.1 Бізнес вимоги 1.4.2 Бачення рішення 1.4.3 Сфера застосування та обмеження 2 Постановка задачі 2.1 Архітектура додатку 2.2 Вимоги до зберігання даних 2.3 Вимоги до користувацького інтерфейсу 3 Визначення алгоритмів розроблюваної системи 3.1 Опис алгоритмів головних бізнес-процесів розроблюваного веб-застосунку4 Програмна реалізація 4.1 Архітектура застосунку 4.2 Імплементація безпеки застосунку 4.3 Інтеграція з Iubenda 4.4 Інтеграція з Дія 4.5 Інтеграція з хмарним 4.6 Інтеграція з Open AI 4.6.1 Експериментальні дослідження 4.7 Реалізація бекенду застосунку 4.7.1 Реалізація сервісу odyssey 4.7.2 Реалізація сервісу beacon 4.7.3 Реалізація сервісу* lore *4.7.4 Реалізація сервісу forge 4.8 Розробка дизайну 4.9 Реалізація фронтенду застосунку 4.10 Взаємодія з БД Висновки*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій: _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання на виконання роботи*	*22.09.2023*	
2	*Формування бізнес-вимог та бізнес ризиків*	*23.09.2023-1.10.2023*	
3	*Юридичне дослідження*	*2.10.2023 – 10.10.2023*	
4	*Розробка дизайну*	*11.10.2023 – 1.11.2023*	
5	*Вибір технологій та інтеграцій*	*2.11.2023 – 18.11.2023*	
6	*Розробка серверної частини з необхідними інтеграціями*	*19.11.2023 – 10.12.2023*	
7	*Розробка клієнтської частини*	*11.12.23 – 31.12.23*	
8	*Написання кваліфікаційної роботи*	*01.01.2024 - 22.01.2024*	
9	*Представлення кваліфікаційної роботи в ДЕК*	*23.01.2024*	

Дата видачі завдання 22.09.2023 р.

Студент _____ Нероба Ю. Р.
(підпис)

Керівник роботи _____ проф Міщеряков Ю. Р.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Атестаційна робота: 112 с., 41 рис., 28 джерел фнформації, 5 додатків.

НЕТВОРКІНГ, СТАРТАПИ, SPRING FRAMEWORK, ДІЯ, GCP, JS, AI

Об’єкт дослідження – система нетворкінгу пошуку студентів-кофаундерів для заснування стартапів. Предметом дослідження є впровадження інтеграцій сторонніх існуючих систем в розроблюваний застосунок.

Мета роботи – створити платформу, яка дозволяє користувачам не тільки знайти потенційних партнерів для стартапів, але й набути практичний досвід у командній роботі, управлінні проектами та реалізації бізнес-ідей.

До методів дослідження належать вивчення та аналіз технічної документації й теоретичних відомостей. Окрім того, використана методика аналогій, котра включає аналіз існуючих систем нетворкінгу та пошуку кофаундерів для визначення їх сильних та слабких сторін, що допоможе виявити потенційні можливості для покращення та унікальні особливості нової системи.

В результаті розробки отримана система має надавати змогу кваліфікованим фахівцям не тільки знайти партнерів для стартапів, але й розвивати свої бізнес-ідеї, перетворюючи їх на реальні проекти.

ABSTRACT

Explanatory note to the certification work of the master's degree contains: 112 pages, 41 pictures, 28 sources of information and 5 additional sections.

NETWORKING, START UP, SPRING FRAMEWORK, DIIA, GCP, JS, AI

The object of study is a networking system for finding student co-founders for establishing startups. The subject of the study involves the implementation of integrations.

The goal of the work is to create a platform that allows users to not only find potential partners for startups but also to gain practical experience in teamwork, project management, and the realization of their business ideas.

The research methods include the study and analysis of technical documentation and theoretical information. Additionally, the methodology of analogies is used, which involves the analysis of existing networking and co-founder search systems to identify their strengths and weaknesses. This will help to discover potential opportunities for improvement and unique features of the new system.

As a result of the development, the system should enable qualified professionals to not only find partners for startups but also to develop their business ideas, turning them into real projects.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HR (Human Resources) - Управління людськими ресурсами

API (Application Programming Interface) – Прикладний програмний інтерфейс

ЄДЕБО - Єдина державна електронна база з питань освітиIT (Information Technology) – Інформаційні технології

MVP (Minimum Viable Product) - Мінімально життєздатний продукт RBAC (Role Based Access Control) - Керування доступом на основі ролей

EE (Enterprise Edition) – Корпоративний Варіант

БД – База даних

HTTP (HyperText Transfer Protocol) – Протокол передачі гіпертекстуUI (User Interface) – користувацький інтерфейс

OAuth (Open Authorization) - Відкритий стандарт авторизації

OWASP (Open Web Application Security Project) - Відкритий проєкт з безпеки вебзастосунків

JSON (JavaScript Object Notation) - запис об'єктів JavaScript

JWT (JSON Web Token) – JSON веб токен

ВСТУП

Яка дійсна цінність вищої освіти на теренах України? Як часто питають на співбесідах диплом, а не фактичний досвід? У скількох з випускників вишів запитується документ про його закінчення? Справедливості заради варто зауважити, що в юриспруденції, медицині, інженерії, освіті та для держслужбовців диплом є обов'язковим і вимагається на співбесідах. Але як щодо інших спеціальностей?

Отже, окрім посередньої якості освіти студенти в Україні стикаються ще з проблемами того, що їх дипломи не мають ваги; тим, що вони не можуть влаштуватись на першу роботу за причиною відсутності досвіду та необхідних навичок на комерційних проектах.

Об'єктивно, розробкою одного веб-ресурсу не вирішити настільки багатогранну проблему, однак можливо надати інструмент, який можливо використати для здобуття досвіду роботи на реальних проектах, які будуть створені самими ж студентами в кооперації з іншими такими ж спеціалістами на веб-платформі. 0.year.exp має стати cost-free веб-ресурсом, спрямованим об'єднувати людей з різних сфер діяльності, таких як IT, медицина, бізнес, наука, мистецтво, юриспруденція навколо спільних ідей, які потенційно можуть стати стартапами. Така платформа здатна сприяти обміну знаннями, збагаченню портфоліо чи резюме набуття навичок командної співпраці, поглиблення hard- та soft- навичок і розширення мережі контактів, що є есенційним для розвитку кар'єри.

Таким чином задля успіху ініціативи і з точки зору принципів сталості проектів при розробці MVP 0.year.exp необхідно дотримуватись принципів соціальної відповідальності, адаптованості, надійності та прозорості. Крім того, надаючи послуги, що безпосередньо пов'язані зконфіденційною інформацією при проектуванні важливо врахувати обов'язковість захисту інтелектуальної власності користувачів, їх особистих даних та забезпечення верифікації публічних даних з метою запобігання шахрайства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Проблемою винесеною на дослідження в даній атестаційній роботі – є складнощі працевлаштування за відсутності досвіду після чи підчас отримання вищої освіти, розгляду та пошуку вирішення своєрідного парадоксу: неможливості отримати роботу без досвіду і відсутності можливості отримати досвід без роботи.

Серед варіантів подолання даної дилеми може бути:

- заснування власних стартапів;
- волонтерство;
- стажування;
- проходження додаткових курсів з «гарантованим»працевлаштуванням;
- фріланс;
- професійні нетворкінги.

При детальному розгляді кожного з вищеперерахованих варіантів, справедливим буде твердження, що вони є досить ризикованими, а часом навіть неможливими для деяких сфер діяльності:

- волонтерство в Україні особливо на момент написання даної кваліфікаційної роботи є вузькоспеціалізованим і переважно спрямованим на допомогу військовим, переселенцям чи постраждалим від бойових дій;
- проходження курсів з «гарантованим» працевлаштуванням ніколи не означає, що по завершенню курсів вам знайдуть вакансію або наддадуть роботу в компанії, що проводила курси;
- стажування, здебільшого є нагодою, що не перекриває кількість пропозицій на ринку праці за умов фінансових криз, інфляції та періодів масових звільнень;
- знаходження вакансій за допомогою нетворкінгів та фріланс залежні від глобальної економіки і не є надто надійними джерелами пошуку прибутку;

– стартапи вимагають пошуків інвестицій, команди і є доволі ненадійними, але можуть бути вартими того, щоб почати.

Для ідеї поданої на атестаційну роботу є створення проекту, що поєднує методи розробки стартапів та нетворкінгу.

З перспективи нетворкінгу розроблюваний в рамках даної атестастаційної роботи проєкт 0.year.exp дозволяє об’єднання з фахівцями з необхідної галузі; пошук спеціалістів за схожими напрямами досліджень; перевірку освіти співрозмовника; модерацію публічних даних ініціатив з метою запобігання нелегального контенту на веб-ресурсі.

З перспективи платформи для стартапів 0.year.exp не надає фінансування чи створення грантів, проте допомагає в підборі команди та підписанні цифрових угод таких як Co-Founder Agreement.

Важливою приміткою є те, що жодна команда не зобов’язана робити зі свого проекту стартап і проект може бути лише тестовим середовищем для набуття та поглиблення знань. Генерування та підписання co founder agreement угоди виключно на 0.year.exp також є опціональним.

Головним критерієм вибору даного методу є можливість надати інструмент для втілення власних мрій, пошуку нових можливостей, розвитку навичок, спроможність створити власну кар’єру.

Науково-технічна задача даної атестаційної роботи полягає у створенні системи, що здатна стати одним з інструментів самореалізації студентів. Основна увага в проекті буде зосереджена на розробці веб-платформи залучення до проектної діяльності, обміну досвідом та знаннями, розвитку професійних зв'язків. Як внесок в здобутки наукової спільноти кваліфікаційна робота передбачає розгляд інтеграції штучного інтелекту та державних сервісів при пошуку команди.

1.2 Дослідження впливу технологій

Інтернет, смарт-девайси стали невід’ємною частиною побуту сучасної людини. Навчання, споживання суспільно-політичного чи розважального

контенту, банківські операції або навіть звичайне спілкування відбувається переважно онлайн. До цього списку також можна віднести зовнішній чи внутрішній рекрутинг, робочі комунікації тощо. Така цифрова свобода розширює можливості в пошуках справи життя чи способів самовираження.

Розглядаючи проблематику рекрутингу, що постає головною, в розроблюваній системі, можнавиокремити наступні аспекти, що вирішуються за допомогою технологій:

- комунікація;
- пошук;
- довіра;
- саморозвиток та розвиток власних ідей.

Загалом, реалізацію комунікації та пошуку здатні забезпечити соціальні мережі, довіру - сервіси перевірки публічних даних громадян.

Соціальні мережі - це онлайн-платформи та веб-ресурси, які дозволяють користувачам спілкуватися, обмінюватися інформацією та створювати контент. Відомими прикладами цих мереж можуть бути такі платформи, як Facebook, LinkedIn, Twitter, Instagram, і багато інших. Вони створюють спільноти користувачів, які можуть об'єднуватися навколо різних інтересів, тем та сфер професійної діяльності.

HR можуть використовувати соціальні мережі для пошуку потенційних кандидатів на вакансії. Вони можуть шукати кандидатів за ключовими словами в публікаціях чи описах профілів, професійними навичками, аналізувати їхні навички та досвід, та надсилати запрошення на співбесіду. Компанії можуть використовувати соціальні мережі для побудови свого бренду, поширення досягнень, своєї культури та цінностей, що може привернути потенційних кандидатів. Спільні проекти та ідеї можуть бути просунуті через соціальні мережі, де багато людей знаходяться онлайн. Користувачі можуть ділитися інформацією про свої ідеї та залучати інших до участі.

Сервіси перевірки публічних даних громадян – це веб-ресурси, що доступні кожному і які дозволяють пересвідчитись в правдивості публічних даних розміщених в Інтернеті з метою попередження шахрайства та іншої

незаконної діяльності. Серед тих, що необхідні в рамках написання 0.year.exp знаходяться Дія та Єдина державна електронна база з питань освіти.

Єдина державна електронна база з питань освіти є автоматизованою системою збирання, реєстрації, оброблення, зберігання та захисту відомостей та даних з питань освіти. Власником Єдиної державної електронної бази з питань освіти є держава, розпорядником – Міністерство освіти і науки України, а технічним адміністратором – державне підприємство «Інфоресурс». ЄДЕБО надає можливість отримувати доступ до документів, пов'язаних з освітою, таких як свідоцтва про освіту, академічні документи, дипломи тощо.

За допомогою цієї електронної бази даних, всі бажаючі мають можливість отримувати доступ до важливої інформації та проводити детальні дослідження в галузі освіти, що допомагає виявляти прозорість та проблеми в цій сфері.

Дія [1] - це цифрова платформа, яка надає можливість громадянам перевіряти свої особисті дані, взаємодіяти з державними органами та користуватися різними державними послугами онлайн. Реалізацією проєкту займається Міністерство цифрової трансформації. Метою Дія є подолання бюрократії, цифровізація документообігу та спрощення взаємодії між громадянами, державою та бізнесами.

Основні сервіси та функції, які надаються за допомогою «Дія», включають:

– Перевірка інформації;
– Звернення до державних органів;
– Реєстрація ФОП чи впровадження Дія.City;

Якщо розглядати перелік сервісів, які допомагають в досягненні кар’єрного успіху, то не можна назвати один, що вирішить проблеми всіх і кожного. Саморозвиток не забезпечить ніхто і ніщо, окрім безпосередньо людини, що його бажає. Технології в даному випадку можуть слугувати лише інструментами в досягненні обраної мети, а їх вибір залежить виключно від потреб конкретної людини.

1.3 Аналіз аналогічних систем

Ідея висунута при проектуванні 0.year.exp не є новою й аспектно реалізована багатьма іншими вже існуючими веб-ресурсами, серед яких:

– Djinni;
– LinkedIn;
– Work.ua;
– Behance;
– Upwork;
– Freelance.ua;
– Sketchfab,;
– Fiverr;
– Tinder.

Інтерфейс Djinni наведений на рисунку 1.1.

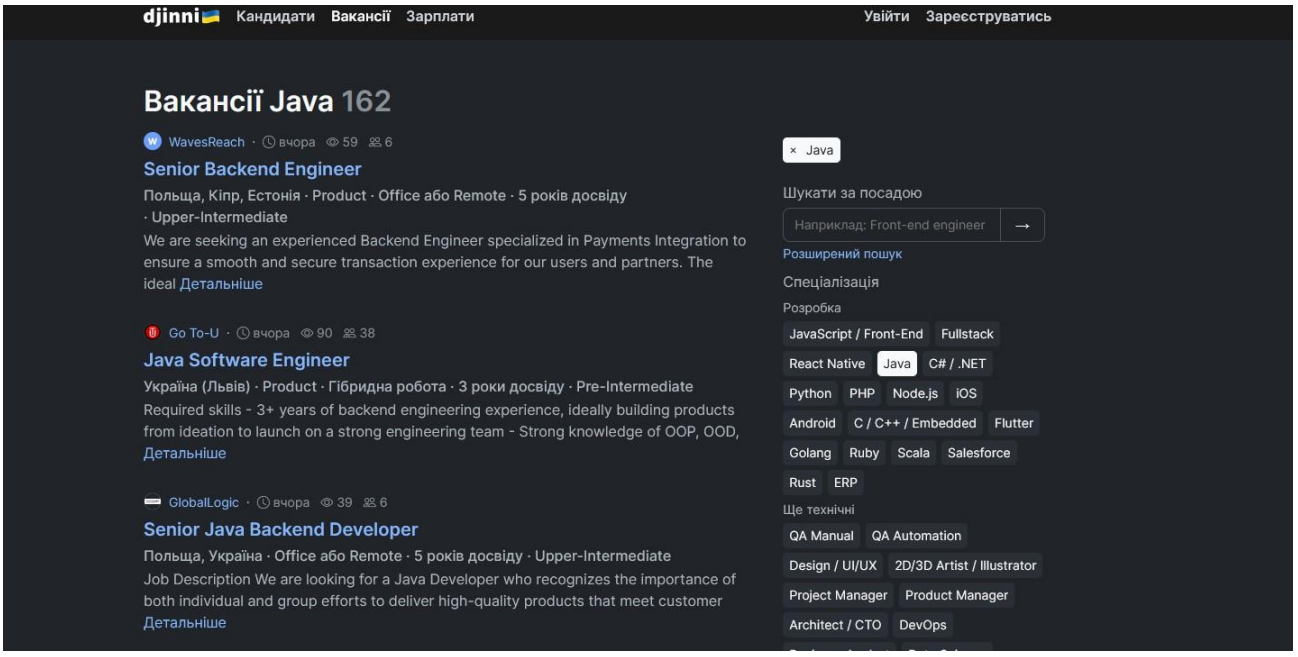

Рисунок 1.1 - Інтерфейс Djinni

Djinni - це український сервіс для пошуку роботи, який фокусується на сфері IT і технологій. Він відрізняється тим, що забезпечує приватність для

кандидатів: їхні профілі не є публічними, і тільки вони можуть ініціювати контакт із роботодавцями. Контакт з роботодавцями відбувається напряму, що сприяє більш відкритому та ефективному діалогу між сторонами і дозволяє обговорювати всі деталі працевлаштування без посередників. Ця платформа є дуже популярною серед IT-фахівців, таких як розробники, дизайнери, QA інженери та проектні менеджери, оскільки вона дозволяє їм знаходити роботу, яка точно відповідає їх навичкам та інтересам.

LinkedIn є світовою професійною соціальною мережею, яка дозволяє користувачам будувати професійні зв'язки та спільноти. Він допомагає створити професійний інтернет-профіль, демонструвати свій досвід та здібності, а також знаходити роботу та спільноти, де можна отримати досвід та навички. З точки зору нетворкінгу LinkedIn дозволяє знаходити і додавати професійні контакти до вашої мережі. Ви можете підтримувати зв'язки з колегами, роботодавцями та іншими фахівцями відповідно до вашої галузі або інтересів. Інтерфейс LinkedIn наведений на рисунку 1.2.

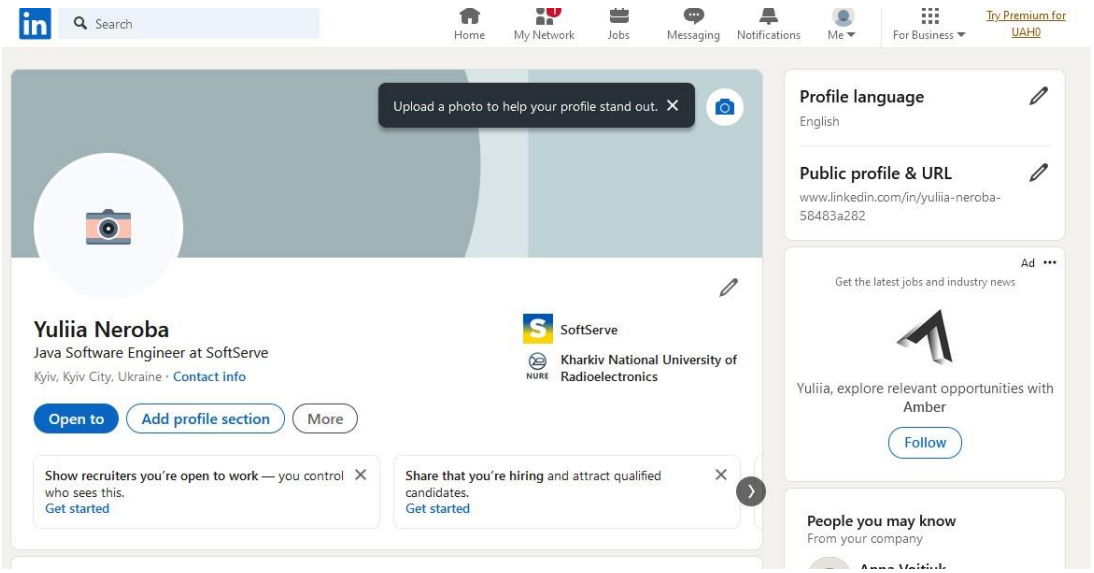

Рисунок 1.2 - Інтерфейс LinkedIn

Behance - це платформа для креативних фахівців, таких як дизайнери, фотографи, художники тощо. На даному веб-ресурсі можна створити власне дизайнерське портфоліо та демонструвати свої роботи, що є необхідним для здобуття клієнтів і отримання замовлень.

Upwork - це платформа для фрілансу, де можна пропонувати різні послуги, від веб-розробки до копірайтингу. Upwork є однією з платформ, яка реалізує модель гіг-економіки. У гіг-економіці акцент робиться на короткостроковій індивідуальній праці, і Upwork надає інфраструктуру для такої роботи. Вона дозволяє фрілансерам працювати над різноманітними завданнями та проектами, залучаючи їхні унікальні навички та досвід. Розглядаючи цільову аудиторію предметної області, варто зауважити, що сервістакож дозволяє заробляти гроші, виконуючи проекти з усього світу, навіть без значного практичного досвіду. Інтерфейс Upwork наведений на рисунку 1.3.

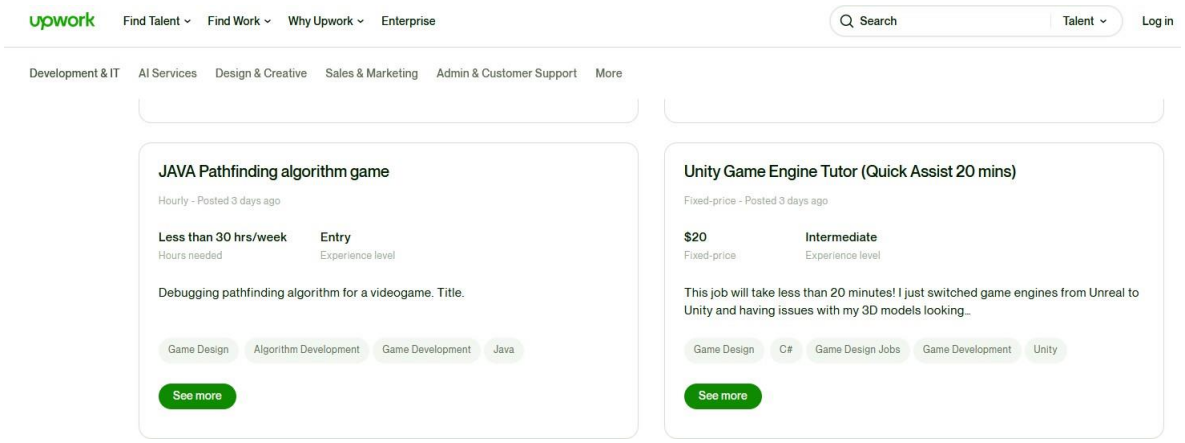

Рисунок 1.3 – Інтерфейс Upwork

Іншими альтернативними майданчиками для фрілансу подібними до Upwork є Freelance.ua та Fiverr.

Sketchfab - це платформа для 3D дизайнерів, що є корисною за бажання демонструвати свої роботи та знайомитися з потенційними клієнтами або роботодавцями. Цей майданчик подібно до Behance має на меті презентувати портфоліо та роботи фрілансерів, продавати їх, однак зі своєю спеціалізацією - 3D.

Tinder - це додаток, спрямований на знайомства та створення романтичних відносин, і використання його для працевлаштування не є стандартною практикою. Однак деякі користувачі можуть використовуватицей додаток для знайомства з потенційними співробітниками або бізнес-партнерами.

Платформи для працевлаштування та кар'єрного розвитку стали важливими інструментами в сучасному світі праці, надаючи вузькоспеціалізовані рішення для осіб, які прагнуть покращити свої професійні можливості. Кожна з таких платформ має свої унікальні характеристики та специфічні функції, спрямовані на вирішення конкретних завдань у сфері працевлаштування та кар'єрного розвитку. Однією з таких спеціалізованих платформ є 0.year.exp, яка відзначається своєю спеціалізацією та спрямованістю на розв'язання конкретної проблеми працевлаштування для даної цільової аудиторії.

1.4 Бачення та сфера застосування

1.4.1 Бізнес вимоги

Для визначення бізнес-вимог необхідно означити бізнес-цілі, критерії успіху, потреби клієнтів та ринку і які передбачаються бізнес-ризики при впровадженні проєкту. Зведені разом дані підпункти і створюють вимоги бізнесу до проекту, що був обраний в рамках даної атестаційної роботи.

До бізнес цілей проекту належать:

– розвиток інноваційних ідей та можливість молодих фахівців створювати власні стартапи та проекти;

– популяризація освіти та навчання серед молоді для постійного самовдосконалення.

– Проект можна вважати успішним за умов досягнення таких факторів як:

– кількість створених стартапів та проектів завдяки спільним зусиллям;

– кількість взаємодій та співпраці між молодими фахівцями різних галузей на платформі;

– достатня кількість користувачів для покриття витрат на нетворкінг через прибуток за рахунок реклами.

При цьому не менш важливим є задоволення потреб майбутніх клієнтів та ринку, що й визначає популярність проекту, а отже й прибуток з нього. Серед потреб клієнтів та ринку підчас етапу аналізу було визначено, що:

– більшість молодих фахівців прагнуть постійно вдосконалюватисята розвивати свої професійні компетенції. Вони активно шукають можливості для навчання, співпраці та обміну досвідом з однодумцями;

– значна частина молодих фахівців мріють про створення власних стартапів або проектів, але їм часто бракує підтримки, знань та ресурсів для реалізації своїх ідей;

– є потреба в створенні можливостей для взаємодії, співпраці та обміну знаннями між молодими фахівцями з різних галузей, що призведе до сприяння інноваційному розвитку держави.

Окрім необхідності задоволення вищеперерахованих умов задля успішного запуску проекту також необхідно враховувати ризики, що можуть виникнути як при супроводі, так і на самому початку впровадження 0.year.exp. До таких ризиків відносяться:

– «ринок праці з недостатньою кількістю робочої сили»;

– недостатня кількість користувачів, щоб покрити базові потреби для існування нетворкінгу;

– відмова info.edbo.gov.ua та академій, що викладають курси, в інтеграції;

– економічна криза та унеможливлення отримання грантів на стартапи;

– поширення скаму та іншого незаконного контенту;

– зловживання службовим становищем серед модераторів;

– підробка особи користувачами;

– відмова в співпраці від команди Дія|Впровадження.

1.4.2 Бачення рішення

Візія розроблюваного проекту полягає в розробці осередку об'єднання молодих талантів, незалежно від їхньої галузі та досвіду. Прагнення створити середовище, де молоді українські обдаровані особистості можуть знайти одне одного, спільно вдосконалювати свої навички та знаходити спільні проекти для реалізації.

1.4.3 Сфера застосування та обмеження

В якості початкової версії нетворкінгу планується представити веб-версію продукту з найбільш важливими функціями створення профілю, можливість ручної перевірки достовірності даних модераторами, додавання ініціатив та залучення до них

В наступних релізах додатку передбачається, що має бути реалізовано:

- інтеграція з іншими ресурсами задля автоматизації перевірки наданих користувачами даних про освіту та особу;
- покращений алгоритм для пошуку фахівців не тільки за спеціальністю, а й напрямами створених за період навчання наукових робіт, таких як курсові проекти, дипломні роботи тощо.

Серед недоліків 0.year.exp є обмеження, що не є реалістичним виправити на ранніх релізах. До таких обмежень належать:

- відсутність пошуку менторів;
- відсутність фінансової підтримки ініціатив від безпосередньонетворкінгу;
- доступ лише в Україні;
- відсутність спонсорства.

2 ПОСТАНОВКА ЗАДАЧІ

Об’єктом дослідження є нетворкінг пошуку студентів-кофаундерів для заснування стартапів, який має бути реалізований у вигляді веб-додатку. До основних можливостей, що мають бути реалізовані при розробці належать:

- верифікація освіти;
- створення ініціатив;
- пошук ініціатив та фахівців;
- верифікація особи.

2.1 Архітектура додатку

Створення онлайн-платформи для нетворкінгу вимагає ретельного планування та проектування архітектури, щоб забезпечити надійність, масштабованість та ефективну взаємодію користувачів, а отже вимоги до архітектури щонайменше мають включати:

- гнучкість та легкість масштабування;
- балансування навантаження;
- висока доступність та стійкість до збоїв;
- можливість інтеграцій з іншими сервісами;

Гнучкість архітектури передбачає її здатність адаптуватися до змін у технологічному ландшафті, а також до еволюції бізнес-потреб. Це означає, що архітектура має бути створена таким чином, щоб нові компоненти, сервіси чи функціонал могли бути легко інтегровані без необхідності переписування існуючого коду. Модульність та використання мікросервісів може бути ефективним підходом у цьому контексті, оскільки вони дозволяють ізолювати різні функції застосунку та спрощують процес оновлення та внесення змін.

Легкість масштабування - стосується здатності застосунку витримувати зростаюче навантаження, збільшуючи свої ресурси без значних затрат часу та коштів. Масштабування може бути як горизонтальним з доданням більшої кількості реплік сервісів, так і вертикальним, що передбачає додання ресурсів

до сервісів. Веб-застосунок повинен бути спроектований так, щоб легко підтримувати обидва види масштабування.

Балансування навантаження в контексті веб-архітектури полягає у розподілі запитів чи робочого навантаження між кількома серверами або ресурсами з метою оптимізації використання ресурсів, мінімізації часу відповіді та запобігання перевантаження одного сервера. Це особливо важливо для веб-застосунків, які мають велику кількість користувачів або високе навантаження, оскільки дозволяє забезпечити стабільність та доступність сервісу. Правильно розроблений механізм балансування забезпечує здатність системи адаптуватися до коливань в навантаженні, розподіляючи запити між серверами для оптимізації продуктивності та зниження ризику відмови.

Висока доступність в контексті архітектурних вимог означає, що веб-застосунок повинен бути доступним для користувачів максимально можливий час. Це досягається через розробку архітектури, яка здатна швидко відновлюватися після збоїв, а також має здатність до автоматичного перерозподілу ресурсів у разі виходу з ладу окремих компонентів.

Врахування можливості інтеграцій в архітектурі веб-застосунку означає створення такої структури, яка може легко спілкуватися та обмінюватися даними з зовнішніми системами, сервісами або API. Це включає використання стандартів відкритих інтерфейсів, таких як REST або GraphQL, що сприяє гнучкості та спрощує процес інтеграції з різноманітними зовнішніми джерелами.

Отже, архітектура нетворкінгу має відповідати балансу відкритості і захищеності даних, гнучкості, але не надмірної декомпозиції, високої доступності але не за надлишкових ресурсів.

2.2 Вимоги до зберігання даних

В базі даних мають зберігатись дані про користувачів, ініціативи, освіту та участь користувачів в ініціативах. Дані користувачів мають включати:

- id – ідентифікатор користувача;

– email – пошта користувача, що має слугувати додатковими індексом та бути унікальною;

– photoLink – посилання на фото користувача, що має зберігатись в gcp cloud storage;

– name – ім’я користувача;

– surname – прізвище користувача;

– fatherName – по-батькові користувача;

– about – опис, користувача, який він надає про себе;

– educations – освітні дані користувача;

– role – роль в системі користувача;

– keyWords – ключові слова робіт користувача для пошуку по ним;

– contacts – контактні дані користувача;

– position – бажана позиція користувача.

Дані про освіту обов’язково мають включати userId та тип. Опційними значеннями мають бути number – номер диплому, serialCode – серійний код, university - університет, degree - ступінь, major - спеціальність якщо це дані диплому; resource – компанія курсів, dataToCheck – дані для перевірки, result – курс якщо це дані сертифікату курсів; number – номер студентського, university – університет якщо це студентський квиток. Ці дані мають зберігатись в БД лише для модераторів з метою перевірки і після занесення до даних користувачів видалятись.

Дані запрошень мають враховувати власний унікальний ідентифікатор, ідентифікатор користувача, ідентифікатор ініціативи та тип участі.

Дані ініціатив мають складатись з

– id - унікальний, первинний ключ;

– name – назва ініціативи;

– description – опис ініціативи;

– privateLink (рядок) – посилання на приватний чат;

– status – статус ініціативи.

Фотографії та документи мають зберігатись в хмарному сховищі, в БД мають бути лише посилання на розташування файлів в сховищі.

2.3 Вимоги до користувацького інтерфейсу

Користувацький інтерфейс повинен бути легким у використанні та інтуїтивно зрозумілим. Елементи керування повинні бути детермінованими та логічними. Схожі елементи керування мають бути згруповані задля полегшення навігації. Має бути забезпечено достатню контрастність та чіткість текстових та графічних елементів. Між макетами різних сторінок має прослідковуватись послідовність. Стиль користувацького інтерфейсу має бути мінімалістичним та витриманим в одній кольоровій гаммі.

Всі зображення та шрифти мають бути створені самостійно або взяті з відкритих джерел. Логотип має бути розроблений самостійно, його стиль має бути мінімалістичний, однак зі зрозумілою знаковою частиною.

3 ВИЗНАЧЕННЯ АЛГОРИТМІВ РОЗРОБЛЮВАНОЇ СИСТЕМИ

3.1 Опис алгоритмів головних бізнес-процесів розроблюваного веб-застосунку

Бізнес-процес у контексті веб-додатків включає в себе аналіз та розробку серії кроків або діяльностей, які необхідні для досягнення конкретної бізнес-мети в рамках веб-додатка. Кінцевою метою користувачів 0.year.exp з точки зору розробки має бути заснування стартапу зі знайденою командою кваліфікованих спеціалістів. Бізнес процеси, що мають сприяти досягненню цієї мети – це:

- реєстрація;
- перевірка освіти;
- створення ініціативи;
- пошук команди;
- підписання co-founder agreement.

Процеси реєстрації та перевірки освіти продемонстровано на рисунку 3.1. Тобто необхідними даними при реєстрації є ПІБ та Дія штрих-код, що можна знайти на звороті документів в застосунку Дія. Перевірка особи є автоматичною, перевірка освіти вимагає втручання модератора за причиною поки що відсутньої інтеграції з ЄДБО та компаніями-видавцями курсів.

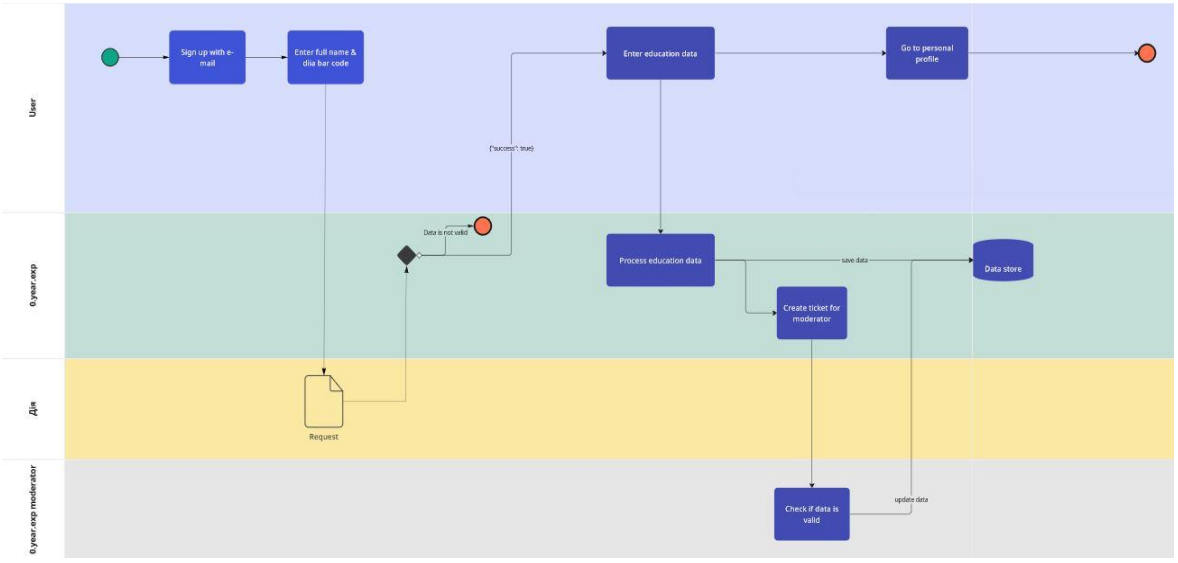

Рисунок 3.1 – Діаграма bpmn 2.0 реєстрації та перевірки освіти

За замовчуванням після реєстрації кожен користувач володіє лише роллю USER. Для отримання прав модерації, інший модератор має надати ці права користувачу через систему 0.year.exp. Доступ до сторінок здійснюється на основі ролі користувача, що реалізовує підхід до безпеки - RBAC.

Продемонстрований на рисунку 3.1 процес виконує одразу дві вимоги другого розділу, а саме – можливість верифікації особи та верифікації освіти.

Процес створення ініціативи наведено на рисунку 3.2. Даний процес реалізовує вимогу «можливість створення ініціатив», котра зазначена в другому розділі. Важливо зауважити, що кожна ініціатива має завчасно визначений перелік допустимих статусів. При створенні ініціативи за замовчуванням ставиться статус - JUST_CREATED. Повний перелік статусів включає:

– JUST_CREATED;
– IN_SEARCH_OF_MEMBERS;
– IN_DEV;
– READY_TO_BE_A_STARTUP.

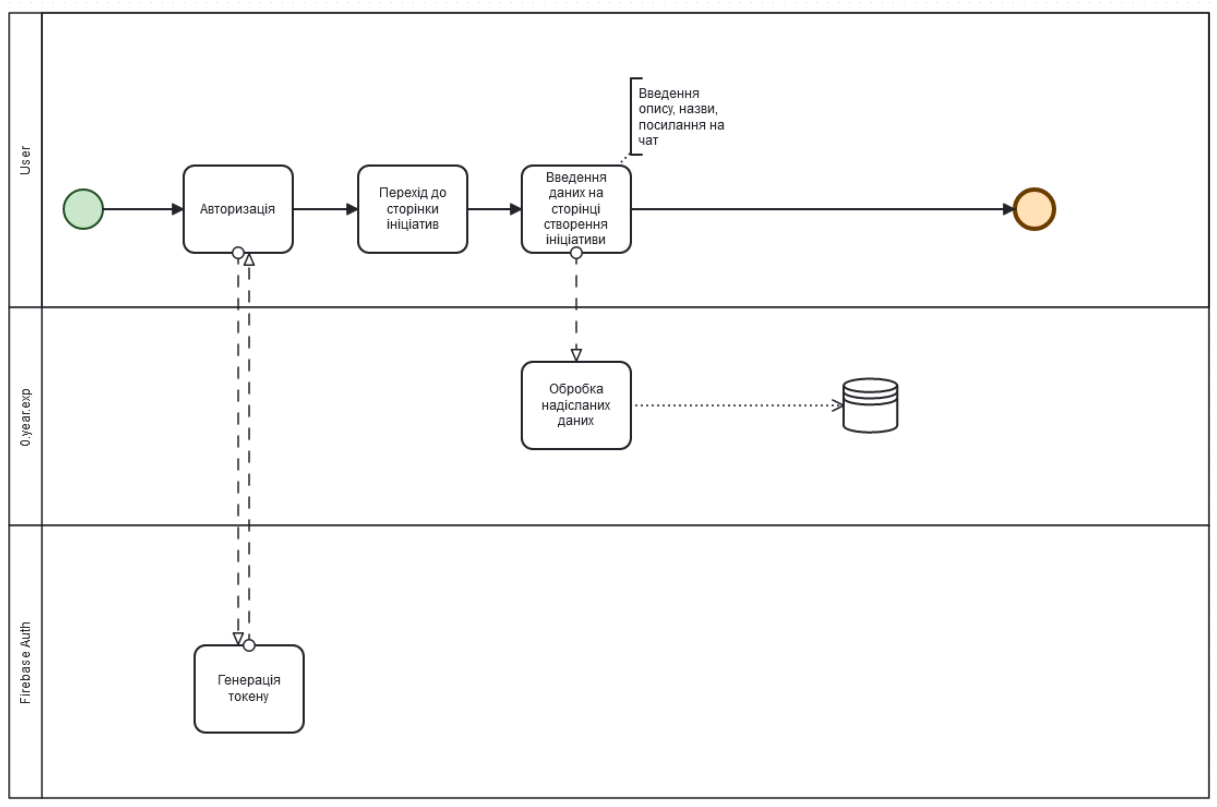

Рисунок 3.2 – Діаграма bpmn 2.0 створення ініціативи

Бізнес-процес пошуку команди є доступним тільки для ініціатив з статусом IN_SEARCH_OF_MEMBERS. Пошук учасників доступний двома шляхами:

– надсиланням запрошень користувачам, які ті зможуть переглянути, схвалити або відхилити;

– прийняття учасників, котрі подали заявки на ініціативу або відхилення їх заявки з огляду на доступний перегляд профілю кандидата.

Для участі в ініціативі участь має бути погоджена з обох сторін. Користувач знаходиться в ініціативі лише коли його участь має статус APPROVED. Повний перелік статусів участі включає:

– JUST_ADDED;

– APPROVED;

– APPLIED.

– Алгоритм підбору кандидатів з позиції учасника та кандидат зображено на рисунку 3.3-3.4. На цих діаграмах фактично зазначається як має бути реалізована поставлена вимога можливості пошуку ініціатив та фахівців.

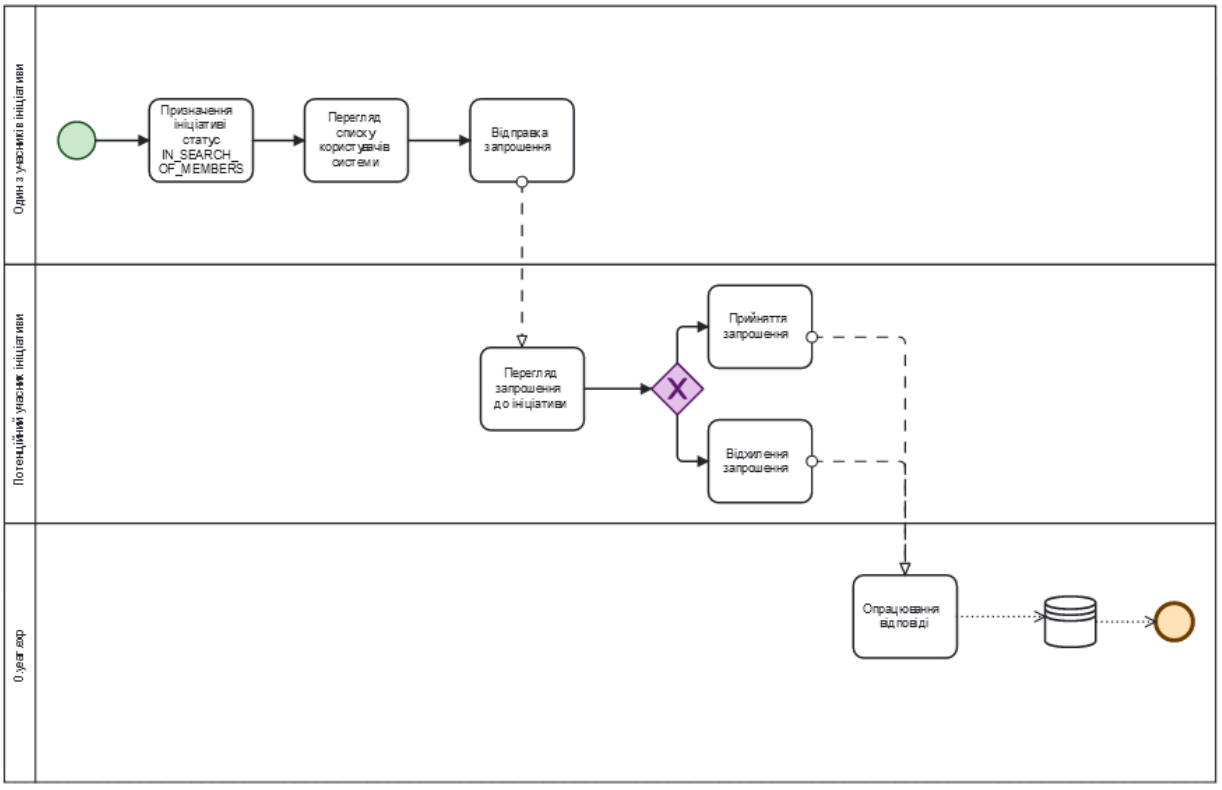

Рисунок 3.3 – Діаграма bpmn 2.0 підбору кандидатів з позиції учасника ініціативи

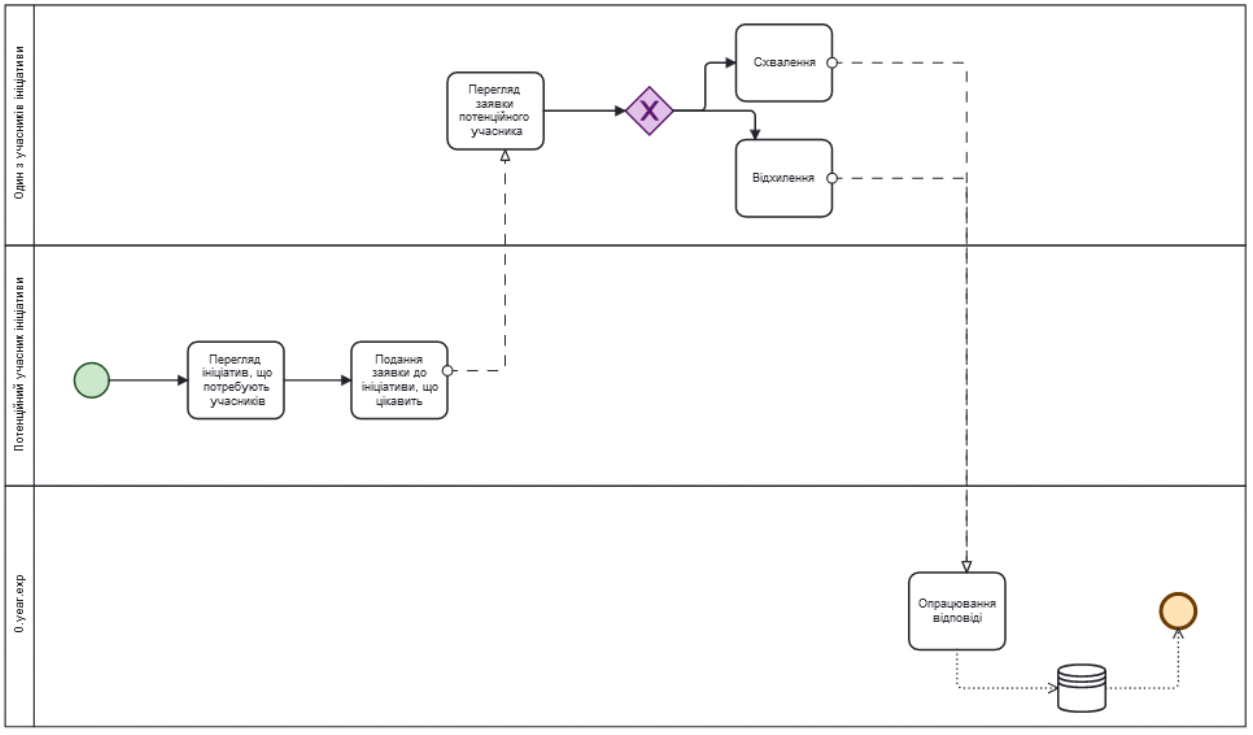

Рисунок 3.4 – Діаграма bpmn 2.0 вибору ініціатив

Бізнес-процес формування угоди наведений на рисунку 3.5. Даний процес вимагає статусу ініціативи READY_TO_BE_A_STARTUP.

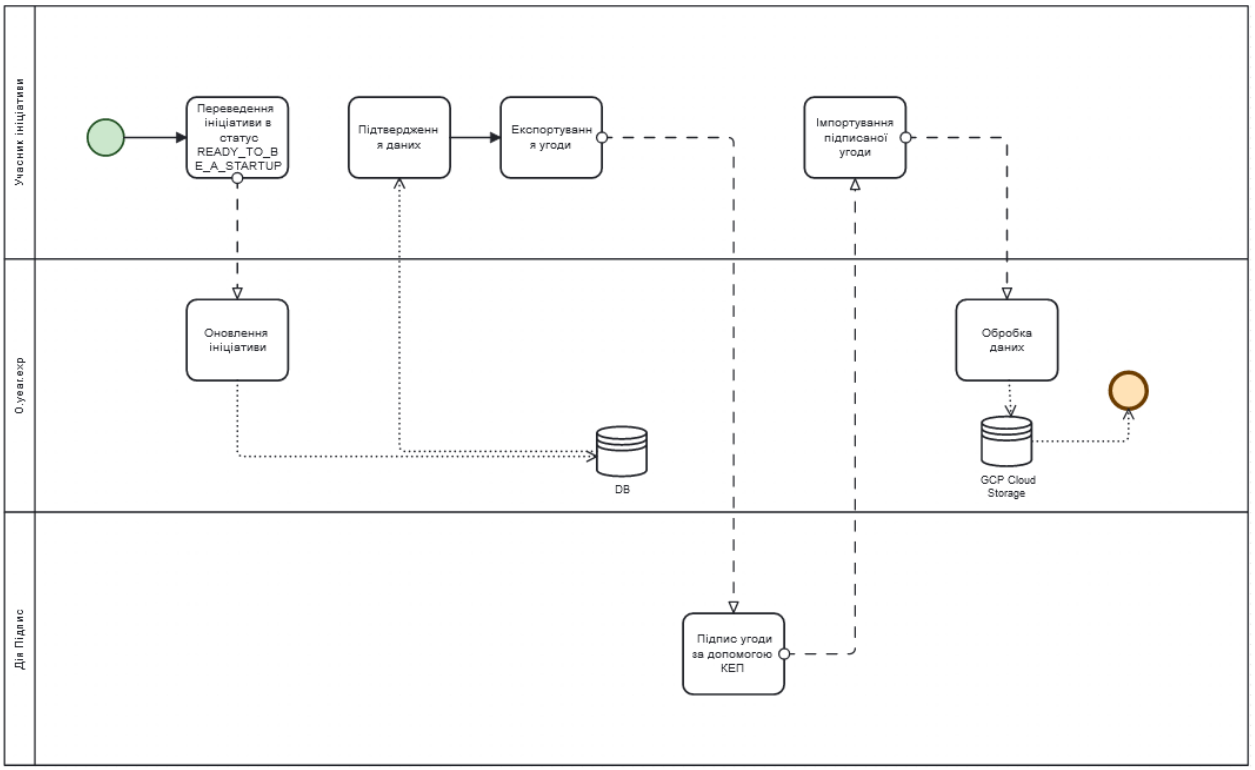

Рисунок 3.5 – Діаграма bpmn 2.0 підписання угоди

Угода, як зазначено на діаграмі, зберігається на google cloud storage, подальший доступ з якого можливий через підпис посилання в бакеті. Користувачем експортована угода з сервісу має бути підписана через Дія підпис.

Такий підпис угоди є безкоштовним, виконується приблизно за хвилини 3 і вимагає лише особистого електронного підпису. Важливо зауважити, що користувач має підписати документ та імпортувати до системи у форматі .p7s для накладання декількох підписів.

Метою введення даного функціоналу є бізнес-ризик викрадення інтелектуальної праці недоброчесними учасниками команди. Ця угода може слугувати доказом у суді відповідно до ст. 8 ЗУ «Про електронні документи та електронний документообіг» та ЗУ «Про електронні довірчі послуги».

Шаблон угоди наведений в додатку А.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Архітектура застосунку

З огляду на вимоги до архітектури, що включають гнучкість, масштабованість, балансування навантаження, доцільним є реалізувати мікросервісну архітектуру для застосунку.

Основна ідея мікросервісної архітектури полягає в тому, що замість створення одного великого монолітного застосунку, який може бути складним у розробці та підтримці, розробники створюють багато маленьких сервісів, кожен з яких відповідає за конкретний аспект функціональності.

За такого підходу розробники не обмежені ні одною мовою, ні однією базою даних. Взаємодія між сервісами обмежується лише обраними підходами комунікацій: HTTP, gRPC, GraphQL, через повідомлення абощо.

З огляду на поставлені бізнес задачі та загальновизнані підходи до побудови мікросервісної архітектури можна визначити наступні сервіси для застосунку:

- odyssey – API Gateway сервіс;
- lore – мікросервіс, що відповідає за формування документів та їх завантаження в хмарне сховище;
- caryatid – UI сервіс, що слугує для взаємодії користувача з системою через браузер;
- forge – мікросервіс, що відповідає за обробку користувацьких даних та їх ініціатив;
- beacon – Euruka сервер, що відповідає за знаходження всіх інших сервісів.

Odyssey та beacon – сервіси, що необхідні для реалізації двох патернів мікросервісної архітектури – Gateway та Discovery.

Головна ідея Gateway Pattern полягає в тому, щоб забезпечити єдину точку входу до системи мікросервісів. Клієнтські запити спочатку надходять до

Gateway, який потім направляє ці запити до відповідних мікросервісів. Це не тільки спрощує архітектуру з точки зору клієнта, але й дозволяє централізовано управляти безпекою, логуванням, балансуванням навантаження та іншими перехресними аспектами.

Не менш важливим фактором впровадження даного архітектурного патерну - забезпечення рівня абстракції між клієнтськими програмами та сервісами застосунку. Це означає, що клієнтським застосункам немає необхідності знати внутрішню роботу сервісів, їх точки входу, порти, хости і тд. Натомість вони можуть покладатися на шлюз API для обробки запитів і відповідей.

Це також полегшує масштабування системи, оскільки додавання чи зміна мікросервісів не вимагає змін у клієнтському додатку. Однак, есенційним мінусом даного підходу є потенційна централізована точка збою.

На рисунку 4.1.1 наведено типовий вигляд архітектури за такого підходу. Зображення взято з відкритого джерела інформації - https://microservices.io.

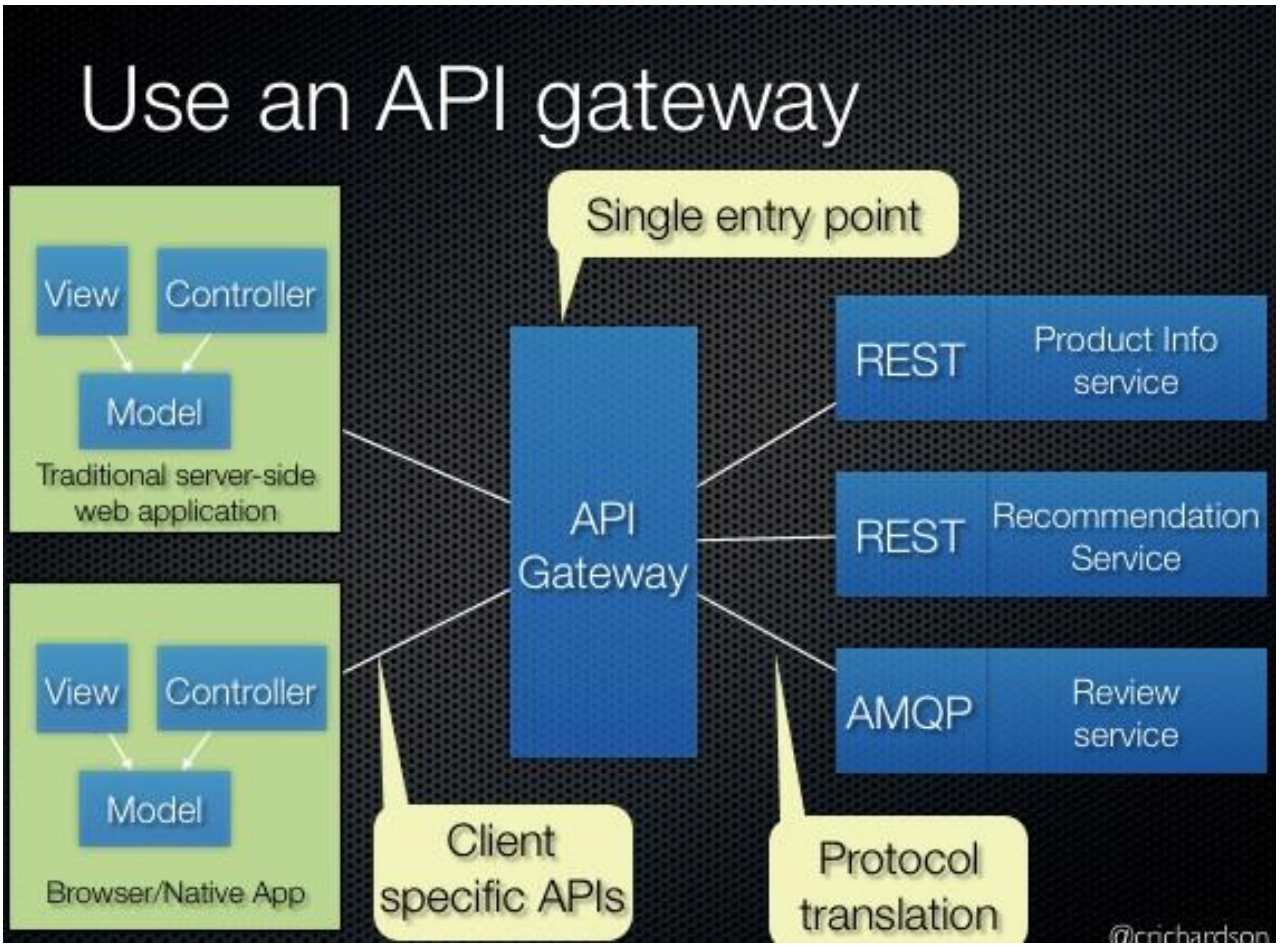

Рисунок 4.1.1 – Типова архітектура з використанням API Gateway

Другим задіяним архітектурним патерном в розроблюваному застосунку є Service Discovery Pattern. Service Discovery дозволяє сервісам знаходити та спілкуватися один з одним без необхідності знати точну локацію чи IP-адреси інших сервісів. В основі цього патерну лежить реєстрація кожного сервісу в спеціальному реєстрі сервісів при його запуску і видалення з цього реєстру при зупинці сервісу. Даний підхід також має недолік у вигляді потенційної точки збою. Типова архітектура за впровадження Service Discovery Pattern виглядає так як продемонстровано на рисунку 4.1.2. Зображення взято з відкритого джерела інформації - https://microservices.io.

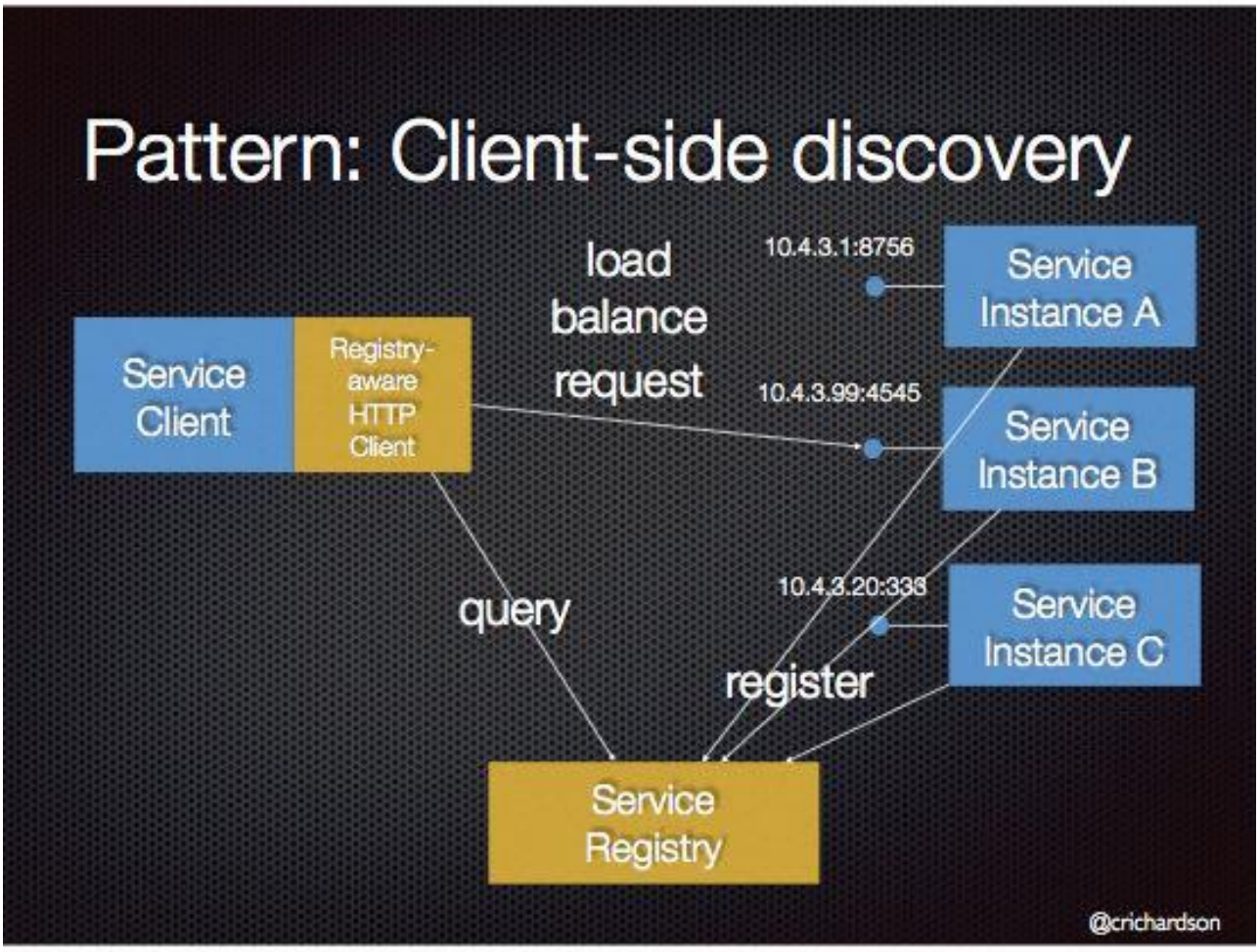

Рисунок 4.1.2 – Типова архітектура з використанням Service Discovery Pattern

Робота Service Discovery в системі 0.year.exp продемонстрована на рисунку 4.1.3. На даному скріншоті можна побачити підняті та зареєстровані сервіси odyssey, lore та forge. Наразі вони мають по одному інстансу та розгорнуті локально, однак в майбутньому при розгортанні в хмарі або на hostiq це буде змінено.

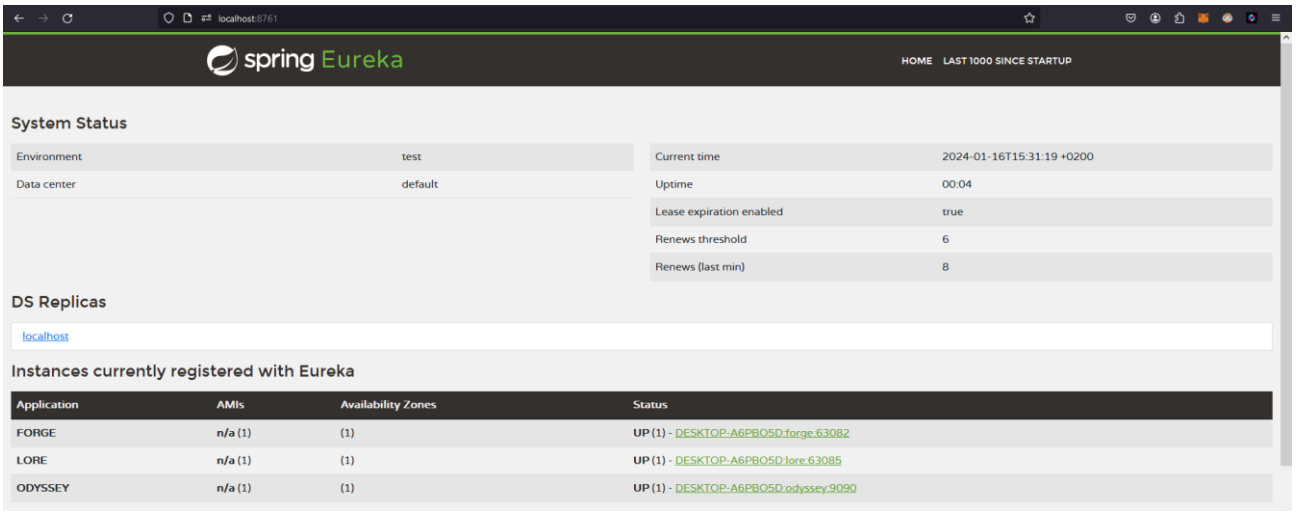

Рисунок 4.1.3 – Робота service discovery

З огляду на спільні переваги патернів, може здатись що вони можуть бути взаємозамінними, однак вони натомість лише доповнюють один одного. Gateway Pattern спрощує взаємодію між клієнтами та системою, пропонуючи єдиний канал для управління запитами. Service Discovery Pattern підвищує гнучкість та ефективність внутрішньої комунікації між сервісами, дозволяючи їм легко знаходити та спілкуватися один з одним. Використання цих патернів разом у мікросервісній архітектурі дозволяє створити більш гнучкі, масштабовані та відмовостійкі системи.

4.2 Імплементація безпеки застосунку

Як визначено, архітектура додатку – мікросервісна, а отже кожен з сервісів є REST API. При побудові REST API важливо дотримуватись основних принципів їх реалізації. До цих принципів належать:

– stateless - кожен HTTP-запит, надісланий на сервер, містить усі необхідні дані для його обробки. Сервер не зберігає інформацію про попередні запити від клієнта.

– client-server – поділ клієнтської та серверної частин.

– єдині інтерфейси для ресурсів.

– cacheable.

– layered архітектура.

Виходячи з перших двох принципів, очевидно, що реалізацій сесій на стороні бекенду не може і передбачатись. Отже, сесіями користувачів має займатись окремий сервіс автентифікації і його відповідь має зберігатись на клієнтській частині і передаватись разом з запитом на серверну частину.

В якості сервісу автентифікації було обрано Firebase Auth. Firebase Authentication від Google надає розробникам повний набір інструментів для реалізації систем аутентифікації в їхніх додатках. Цей сервіс є частиною більшого набору інструментів Firebase, який дозволяє легко інтегрувати різноманітні функції у мобільні та веб-додатки. До зазначених розробниками переваг належать:

- доступні інтеграції з соціальними мережами Google, Facebook, Twitter, GitHub;
- підтримки варіацій методів аутентифікації;
- безпека впровадження та взаємодії;
- масштабованість;
- можливість кастомізації.

Однак окрім цих переваг сервіс також є безкоштовним, легким у використанні та зі зрозумілою документацією, що надається компанією Google.

Причиною відмови написання власного сервісу аутентифікації на користь Firebase Auth є мета заощадити час та вже гарантований високий рівень безпеки.

З метою інтеграції в консолі Firebase було створено власний проект з назвою 0-year-exp, що можна побачити на рисунку 4.2.1.

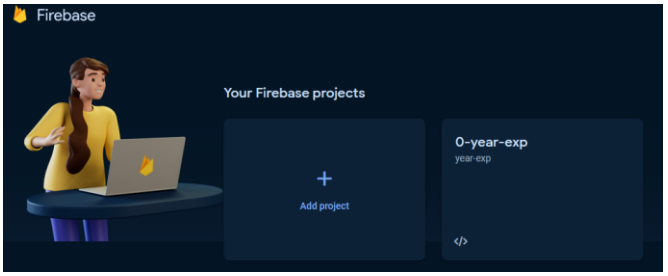

Рисунок 4.2.1 – Проект в консолі Firebase

З метою спрощення взаємодії з користувачем, до Firebase Auth під’єднано google провайдер через який проходить аутентифікація. При цьому не потрібно

створювати новий логін та пароль, а також користувачі можуть бути впевнені, що система не отримає доступ до їхнього пароля від Google.

З’єднання з google провайдером є можливим через OAuth 2.0. OAuth 2.0 в даному контексті є стандартом, який дозволяє користувачам надавати делегований доступ до своїх облікових записів на інших сервісах. У випадку Google provider, OAuth 2.0 використовується для того, щоб дозволити користувачам входити у систему через їхній обліковий запис Google. Процес під’єднання продемонстровано на рисунку 4.2.2, загальні налаштування – рисунок 4.2.3.

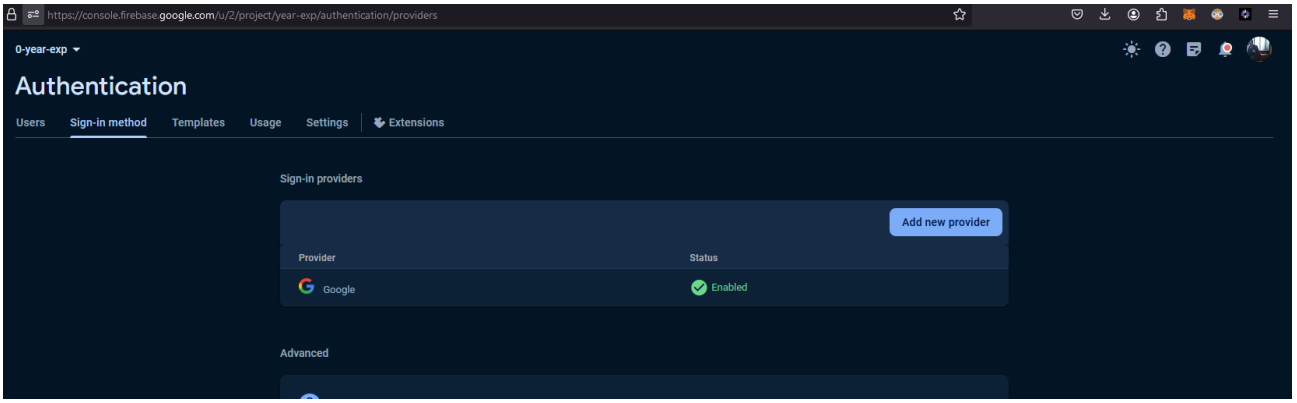

Рисунок 4.2.2 – Включення гугл провайдера до методів аутентифікацій

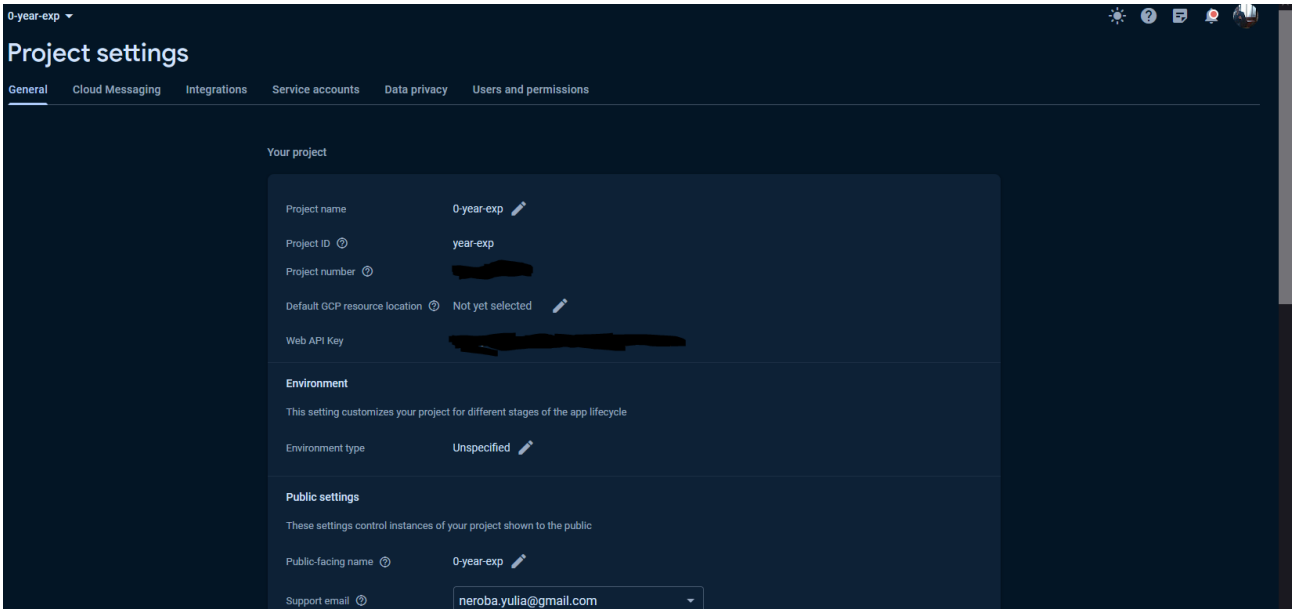

Рисунок 4.2.3 – Налаштування проекту в Firebase Auth

Виклик з метою аутентифікації відбувається з UI сервісу caryatid. Демонстрація роботи інтеграції в проекті наведена на рисунку 4.2.4.

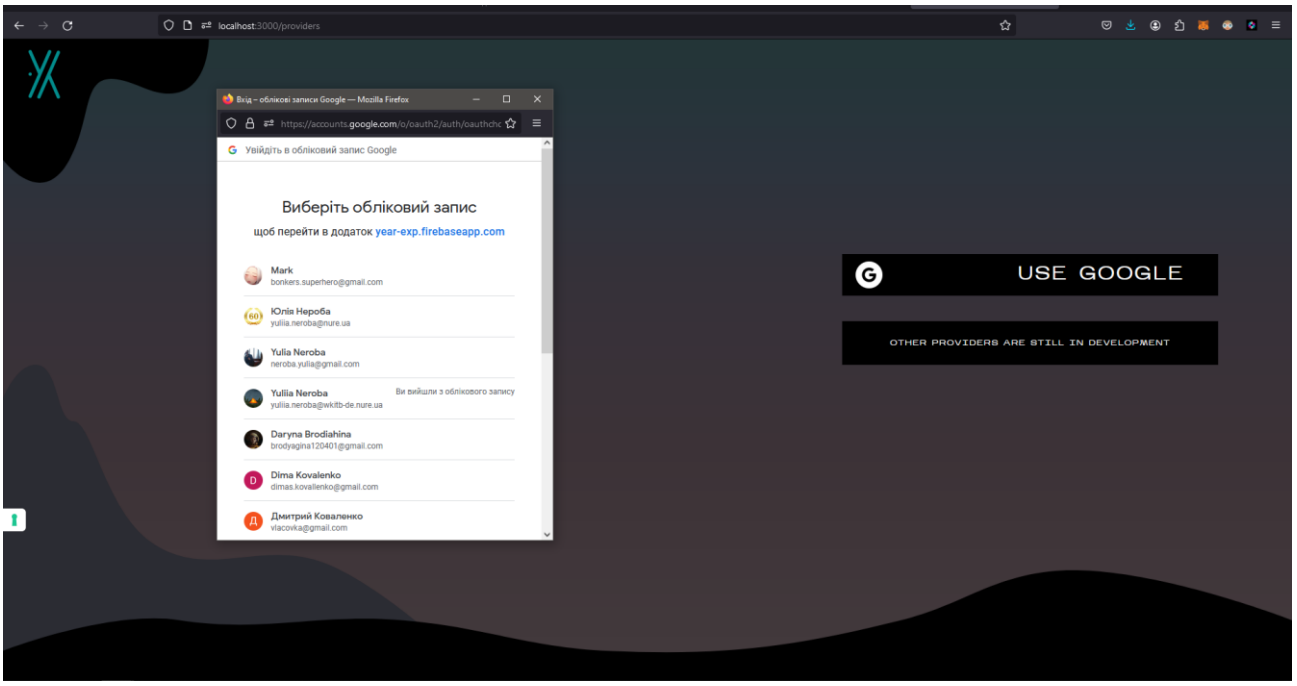

Рисунок 4.2.4 – Аутентифікація в 0.year.exp

Після успішного логіну в систему, в сервісі caryatid зберігається JWT токен.

JWT токен - це компактний, безпечний спосіб передачі інформації між двома сторонами у форматі JSON. Ця інформація може бути перевірена та довірена, оскільки вона підписана. JWT може бути підписаним за допомогою секретного ключа або використовуючи пару ключів публічного/приватного ключа. JWT складається з трьох частин:

– заголовок, який зазвичай містить два поля: тип токену, який є jwt, та використовуваний алгоритм підпису;

– навантаження, яке містить інформацію про користувача та інші дані;

– підпис, який формується з закодованого заголовку та навантаження і підписується алгоритмом, що вказаний в заголовку.

В подальшому на бекенд такий токен надсилається разом з заголовком Authorization, де перевіряється чи токен справжній і чи варто надавати дані за запитом.

Приклад такого токену наведений на рисунку 4.2.5.

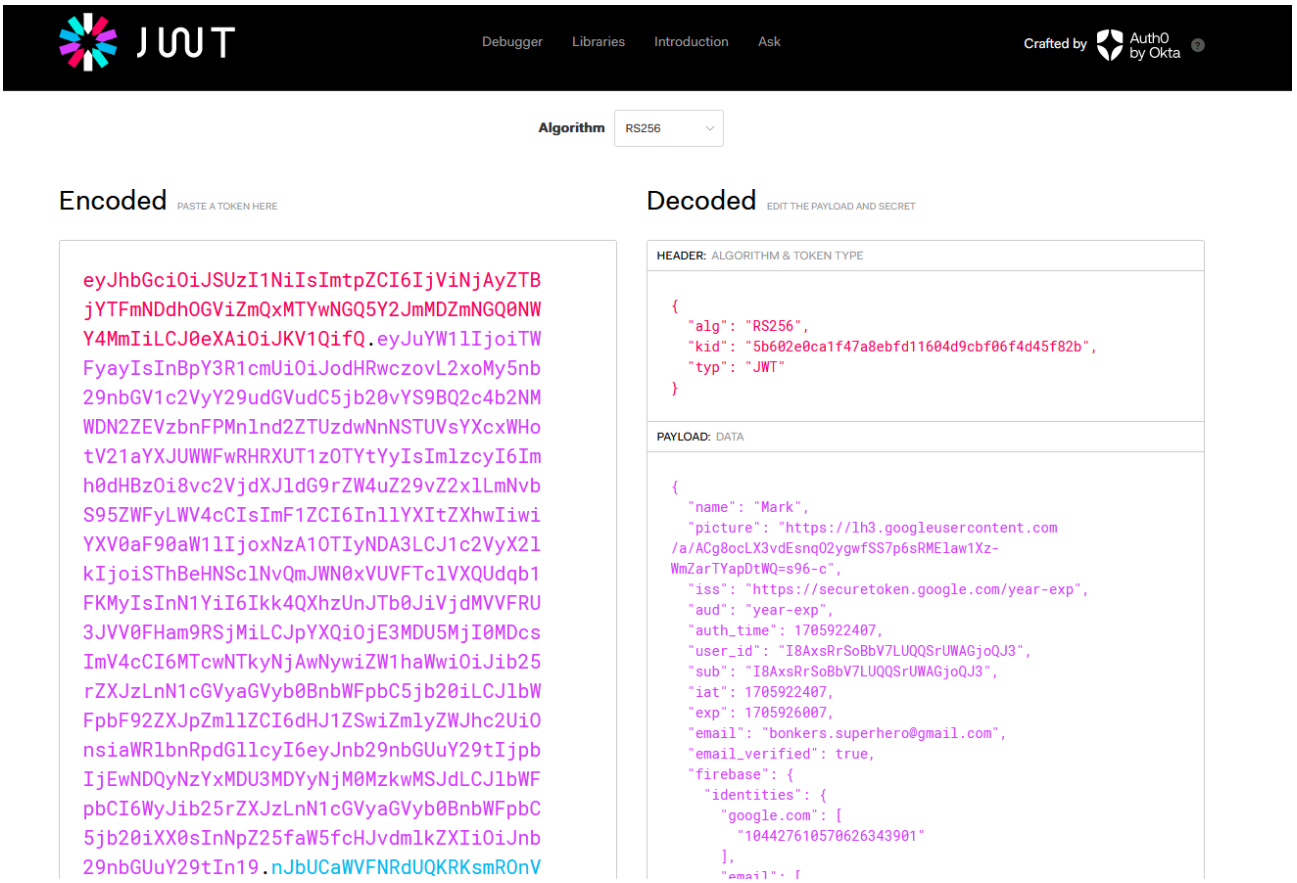

Рисунок 4.2.5 – Токен, що згенерований для користувача системи 0.year.exp

Фрагмент сервісу з запитом, що надсилається з caryatid для отримання токену наведено на лістингу коду 4.2.1, обробка даних на бекенді – 4.2.2.

```
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);

const UserService = {
  // Метод для входу через Google акаунт
  login() {
    const provider = new GoogleAuthProvider();
    return signInWithPopup(auth, provider);
  },
```

Лістинг коду 4.2.1 – Запит в caryatid для логіну

```
package year.exp.forge.config;


import com.google.firebase.auth.FirebaseAuth;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.HttpStatus;
```

```
import org.springframework.web.server.ServerWebExchange;
import org.springframework.web.server.WebFilter;

public class FirebaseAppCheckFilter implements WebFilter {

    private static final Logger logger =
LoggerFactory.getLogger(FirebaseAppCheckFilter.class);

    @Override
    public Mono<Void> filter(ServerWebExchange exchange, WebFilterChain chain) {

        String authorizationHeader =
exchange.getRequest().getHeaders().getFirst("Authorization");

        if (authorizationHeader != null &&
authorizationHeader.startsWith("Bearer ")) {
            String idToken = authorizationHeader.substring(7);
            try {
                FirebaseAuth.getInstance().verifyIdToken(idToken);
                return chain.filter(exchange);
            } catch (Exception e) {
                exchange.getResponse().setStatusCode(HttpStatus.FORBIDDEN);
                return exchange.getResponse().setComplete();
            }
        } else {
            logger.info("In request there is no authorizationHeader");
            exchange.getResponse().setStatusCode(HttpStatus.UNAUTHORIZED);
            return exchange.getResponse().setComplete();
        }
    }
}
```

Лістинг коду 4.2.2 – Клас перевірки токену на бекенді

Для можливості перевірки токену на бекенді використовується клас FirebaseInitializer, який ініціалізує FirebaseApp, креденшиали для який було взято з налаштувань проекту в Firebase console. Лістинг коду ініціалізації – 4.2.3.

```
@Bean
public void initialize() {
    try {
        FileInputStream serviceAccount = new
```

```
FileInputStream("src/main/resources/firebase-service-credentials.json");

        FirebaseOptions options = FirebaseOptions.builder()
                .setCredentials(GoogleCredentials.fromStream(serviceAccount))
                .build();

        FirebaseApp.initializeApp(options);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Лістинг коду 4.2.3 – Бін ініціалізації FirebaseApp

Також з метою безпеки додатку, коли необхідно отримати дані користувача використовується клас HttpServletRequest, з якого отримується токен та email.

4.3 Інтеграція з Iubenda

Iubenda - це сервіс, який допомагає розробникам веб-сайтів та мобільних додатків генерувати юридичні документи, такі як Політика конфіденційності та Умови користування, а також дотримуватися вимог законодавства щодо захисту даних, таких як GDPR. Iubenda допомагає забезпечити, що веб-додаток відповідає національним та міжнародним юридичним вимогам, надаючи зрозумілі та професійно сформульовані Політики конфіденційності, Умови користування та інші необхідні юридичні документи.

Демонстрація інтеграції наведена на рисунку 4.3.1, лістинг коду впровадження – 4.3.1.

```
<script type="text/javascript">
      var _iub = _iub || [];
      _iub.csConfiguration =
{"askConsentAtCookiePolicyUpdate":true,"cookiePolicyInOtherWindow":true,"enableFadp":true,"enable
Lgpd":true,"fadpApplies":true,"floatingPreferencesButtonCaptionColor":"#FFFFFF2E","floatingPrefer
encesButtonColor":"#FFFFFF00","floatingPreferencesButtonDisplay":"anchored-center-
left","lang":"en","perPurposeConsent":true,"siteId":3443926,"whitelabel":false,"cookiePolicyId":8
2968126, "banner":{
"acceptButtonDisplay":true,"closeButtonDisplay":false,"customizeButtonDisplay":true,"explicitWith
```

```
drawal":true,"listPurposes":true,"position":"float-top-
center","rejectButtonDisplay":true,"showTitle":false }};
      </script>
      <script type="text/javascript"
src="https://cs.iubenda.com/autoblocking/3443926.js"></script>
      <script type="text/javascript" src="//cdn.iubenda.com/cs/iubenda_cs.js" charset="UTF-8"
async></script>
```

Лістинг коду 4.3.1 – Інтеграція Iubenda в коді застосунку

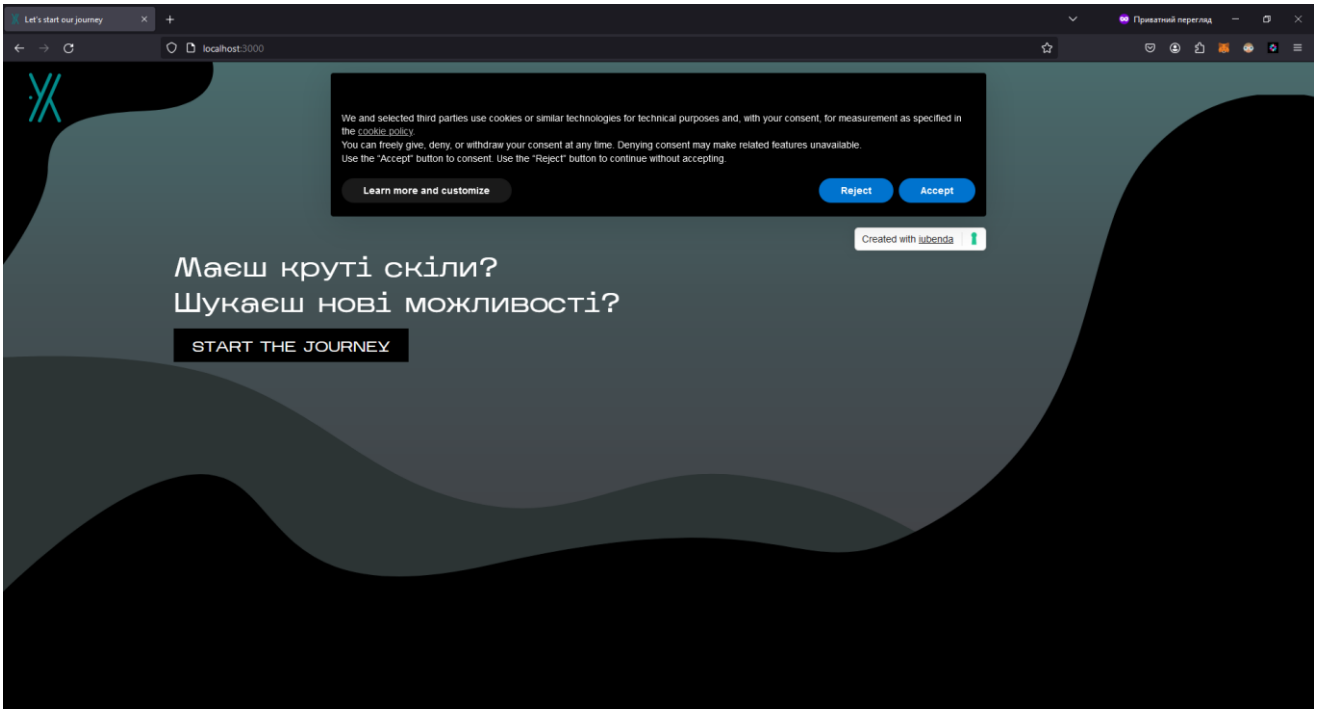

Рисунок 4.3.1 – Відображення інтеграції з Iubenda

4.4 Інтеграція з Дія

Застосунок Дія дає можливість отримувати копії цифрових документів, а також перевіряти дійсність цифрових документів, вік чи ПІБ користувача. Інтегруватися з Дією можна у 2 способи:

– валідація;

– шеринг.

Шеринг дозволяє користувачу надсилати дані за QR на пошту власні дані і цей сценарій не є валідним у випадку необхідності підтвердити лише ПІБ. Валідація - перевірка дійсності цифрового документа або відповідності даних користувача заданим параметрам. Наприклад, перевірка віку або ПІБ. Така валідація є доступною по API та сканером у застосунку Дія.

З метою впровадження даної інтеграції автором кваліфікаційної роботи було зконтактовано з командою Дія | Впровадження, надано дані , що включають:

– ім'я;

– назву університету;

– тему роботи;

– контактний номер телефону.

В чаті команди Мінцифри після заповнення заявки з даними застосунку, отримано документацію та доступ до тестового середовища Дія.

Тестове середовище включає можливість встановлення Diia Stage з застосунку TestFlight. Вигляд тестової версії Дія з тестовим користувачем – Дія Надія Володимирівна наведено на рисунку 4.4.1.

TestFlight дозволяє розробникам запрошувати користувачів для участі в бета-тестуванні додатків. Розробники можуть надіслати запрошення через електронну пошту, а користувачі можуть встановити та використовувати бета-версії додатків, переглядаючи їх через додаток TestFlight на своєму iOS пристрої.

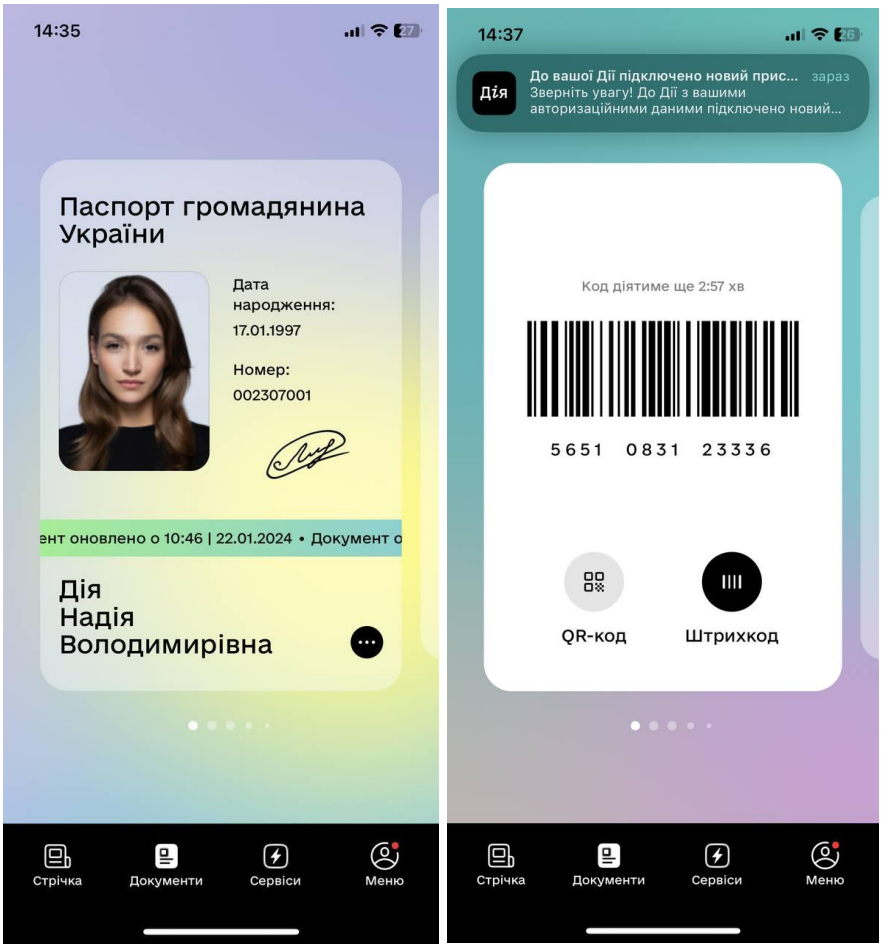

Рисунок 4.4.1 – Тестовий застосунок Дія

Для створення запиту на перевірку ПІБ необхідно мати acquirer_token та auth_acquirer_token, що надається командою Дія | Впровадження. На основі цих двох токенів отримується сесійний токен, з яким необхідно створити гілку з даними наведеними на лістингу 4.4.1.

```
"name": "year-exp",
    "deliveryTypes": ["api"],
    "offerRequestType": "dynamic",
    "scopes": {
        "sharing": [
            "passport", "internal-passport", "foreign-passport"
        ],
        "identification": [ ],
        "documentIdentification": [
            "internal-passport", "foreign-passport"
        ]
    }
```

Лістинг коду 4.4.1 – Запит для створення гілки проекту

Валідувати ПІБ можна лише маючи ідентифікатор гілки, сесійний токен та дані користувача включно з штрихкодом з Дія. Лістинг коду, що виконує запит на перевірку – 4.4.2.

```
private final RestTemplate restTemplate;

@Setter(onMethod_ = {@Autowired})
RedisService redisService;

public DiiaService(){
    restTemplate = new RestTemplate();
}

public Boolean checkName(User dto){
    String token = getSessionToken();
    IdentificationRequest identification = new IdentificationRequest();
    identification.setBranchId(BRANCH);
    identification.setBarcode(dto.getDiiaCode());
    identification.setPerson(person);
    try {
        restTemplate.setInterceptors(
                Collections.singletonList(
                        new CustomHeaderInterceptor("Authorization", "Bearer " +
```

```
token)));

        IdentificationResult result =
restTemplate.postForObject(DOCUMENT_IDENTIFICATION, identification,
IdentificationResult.class);
        assert result != null;
        return result.isSuccess();
    } catch (Exception ex) {
        logger.warn(ex.getMessage());
        return false;
    }
}

private String getSessionToken() {
    var sessionToken = redisService.getValue("diia_token");
    if (sessionToken == null) {
        sessionToken = getDiiaSessionToken();
        redisService.cacheValue("diia_token", sessionToken, 1, TimeUnit.HOURS);
    }
    return sessionToken.toString();
}

private String getDiiaSessionToken() {
    restTemplate.setInterceptors(
            Collections.singletonList(
                    new CustomHeaderInterceptor("Authorization", "Basic " +
AUTH_ACQUIRER_TOKEN)));
    DiiaToken dto = restTemplate.getForObject(SESSION_TOKEN_PATH+ACQUIRER_TOKEN,
DiiaToken.class);
    assert dto != null;
    return dto.getToken();
}
```

Лістинг коду 4.4.2 – Запит для перевірки ПІБ через Дія

Дані наведені на обох лістингах не є повними, що зроблено з метою конфіденційності даних автора роботи та не уточненим питанням про можливість поширення документації від команди Дія. Оскільки життя сесійного токену декілька годин і було б не доцільно щораз викликати API для його генерування, до сервісу додано кешування. База даних для кешу – Redis, про який детальніше знаходиться інформація в розділі 4.9.

4.5 Інтеграція з хмарним сховищем

Хмарне сховище - це модель зберігання даних, в якій дані зберігаються, керуються і обробляються на віддалених серверах, які доступні через інтернет. Ці сервери зазвичай належать і управляються сторонніми провайдерами хмарних сервісів. Хмарне сховище дозволяє користувачам і компаніям зберігати свої дані в безпечному, віддаленому місці, що забезпечує легкий доступ до даних з будь-якого місця, де є інтернет-з'єднання.

Популярні хмарні сховища включають Amazon S3, Google Cloud Storage, Microsoft Azure Blob Storage та інші.

Вартість S3 - $0.023 per GB, GCS - $0.020 per GB, Microsoft Azure Blob Storage - $0.018 per GB. Доступну та зрозумілу документацію мають S3 та GCS. Отже, враховуючи ціну, зрозумілість документації та вже реалізовані інтеграції з Google, рішенням для хмарного сховища обрано GCS.

Google Cloud Storage (GCS) - це хмарне сховище, яке пропонується компанією Google в рамках її обширного портфоліо хмарних сервісів. Це масштабоване, надійне та високо доступне сховище, призначене для зберігання великих обсягів даних.

В рамках проекту GCS використовується для зберігання фото та угод. Щоб взаємодіяти з ним необхідні:

– підключення відповідних бібліотек на серверній частині;

– Google креденшиали;

– попередньо створений бакет на GCS.

У контексті Google Cloud Storage (GCS), "бакет" є основною одиницею зберігання. Це аналогічно до каталогу на файловій системі, але на рівні хмари. Бакет проекту 0.year.exp наведений на рисунку 4.5.1.

Лістинг коду для завантаження файлу – 4.5.1.

```
public String upload(MultipartFile file, String folder) throws IOException {
    String fileName = folder+"/"+ UUID.randomUUID()+file.getOriginalFilename();

    BlobId blobId = BlobId.of(bucketName, fileName);
```

```
    BlobInfo blobInfo = BlobInfo.newBuilder(blobId).
            setContentType(file.getContentType()).build();

    storage.create(blobInfo, file.getBytes());
    return fileName;
}
```

Лістинг коду 4.5.1 – Завантаження файлу в GCS

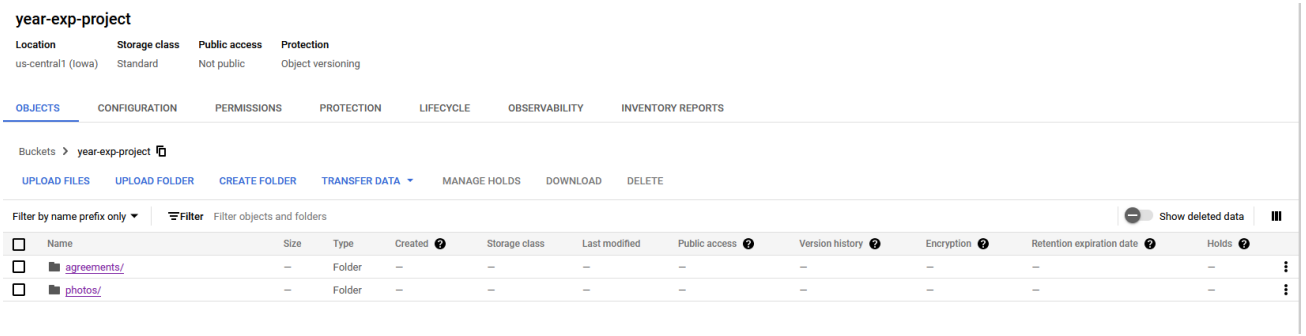

Рисунок 4.5.1 – GCS для 0.year.exp

Лістинг коду для отримання файлу з бакету – 4.5.2. При отриманні повертається не безпосередньо файл, а підписаний URL. Підписування файлів допомагає забезпечити, що лише авторизовані користувачі мають доступ до цих файлів. Це запобігає несанкціонованому доступу або розповсюдженню файлів. Підписані URL-адреси надають тимчасовий доступ до приватних файлів, збережених у бакеті. Використовуючи підписані URL-адреси для доступу до файлів, можна зменшити ризики, пов'язані з постійним відкритим доступом, які можуть включати витоки даних або ненавмисні пошкодження даних.

```
public URL signFile(String fileName) {
    BlobId blobId = BlobId.of(bucketName, fileName);
    Blob blob = storage.get(blobId);

    if (blob == null) return null;

    BlobInfo blobInfo = blob.asBlobInfo();
    return storage.signUrl(blobInfo, 12l, TimeUnit.HOURS,
SignUrlOption.withV4Signature());
}
```

Лістинг коду 4.5.2 – Отримання файлу з GCS

4.6 Інтеграція з Open AI

OpenAI - це дослідницька організація, областю досліджень яких є штучний інтелект. Open AI відома своїми проривами в галузі глибинного навчання та машинного навчання та зосереджена на фундаментальних дослідженнях у галузі штучного інтелекту, працюючи над створенням безпечних та корисних систем ШІ.

Безпосередня інтеграція додатку для пошуку та аналізу даних відбувається з Open AI API, а точніше з асистентами, яких можна створити через Open AI API. Цей API може використовуватися майже для будь-якого завдання, пов'язаного з обробкою тексту, і пропонує моделі з різними можливостями налаштування. Асистенти можуть виконувати завдання для користувачів, використовуючи інструкції, моделі, інструменти та знання. Наразі API підтримує три типи інструментів:

– code interpreter – інструмент, який дозволяє асистенту виконувати код.

– retrieval – інструмент, який дозволяє асистенту витягувати або отримувати інформацію з різних джерел. Наприклад, файлів.

– function calling – інструмент, який дозволяє асистенту виконувати певні задані функції на основі запитів користувача.

Для надсилання запиту до асистентів необхідно:

– створити асистента в api - визначити його набір інструкцій та вибір моделі, увімкнути такі інструменти, як code interpreter, retrieval і function calling за потреби;

– створити потік;

– додати повідомлення до потоку;

– запустити асистента на потоці;

В рамках розробки проекту, було створено 2 асистенти: для пошуку користувачів за запитом та обробки файлів і отримання з них ключових слів щодо досліджень користувачів. Обидва асистенти наведені на рисунку 4.6.1.

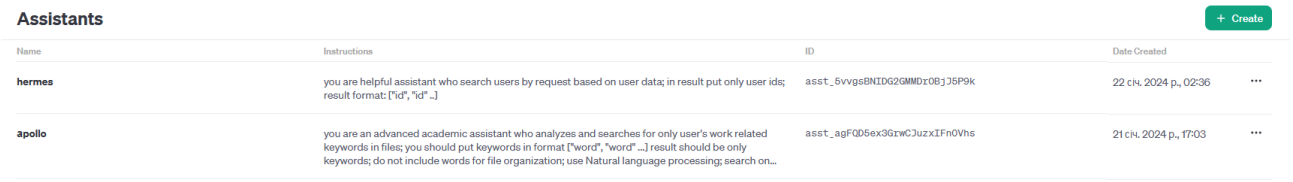

Рисунок 4.6.1 – Створені асистенти

Лістинг коду для взаємодії з асистентом пошуку користувачів наведено на код лістингу 4.6.1.

```
@SneakyThrows
public List<Registered> search(String request) {
    var users = userRepository.findByRoleInUser();

    var candidates = users.stream().map(this::convert).toList();
    Gson gson = new Gson();
    String userData = gson.toJson(candidates);

    String apiKey = "№№№";
    OpenAiService service = new OpenAiService(apiKey);
    CreateThreadAndRunRequest createThreadAndRunRequest = new
CreateThreadAndRunRequest();

    createThreadAndRunRequest.setAssistantId("asst_5vvgsBNIDG2GMMDr0BjJ5P9k");

    ThreadRequest threadRequest = new ThreadRequest();

    MessageRequest messageRequest = new MessageRequest();
    messageRequest.setRole("user");
    messageRequest.setContent("request: " + request + " user data: " +
userData);
    threadRequest.setMessages(List.of(messageRequest));

    createThreadAndRunRequest.setThread(threadRequest);

    Run run = service.createThreadAndRun(createThreadAndRunRequest);
    while (!run.getStatus().equals("completed")) {
        Thread.sleep(600l);
        run = service.retrieveRun(run.getThreadId(), run.getId());
        if (run.getStatus().equals("failed") ||
run.getStatus().equals("cancelled") || run.getStatus().equals("expired")) {
            break;
```

```
        }
    }

    OpenAiResponse<Message> openAiResponse =
service.listMessages(run.getThreadId());
    String array =
openAiResponse.getData().get(0).getContent().get(0).getText().getValue();
    List<String> ids = gson.fromJson(array, List.class);
    return ids.stream().map(
            id -> {
                year.exp.forge.domain.User user = getUser(id);
                return modelMapper.map(user, Registered.class);
            }
    ).toList();
}
```

Лістинг коду 4.6.1 – Лістинг коду пошуку користувачів

Результат виконання запиту продемонстрований на рисунку 4.6.2.

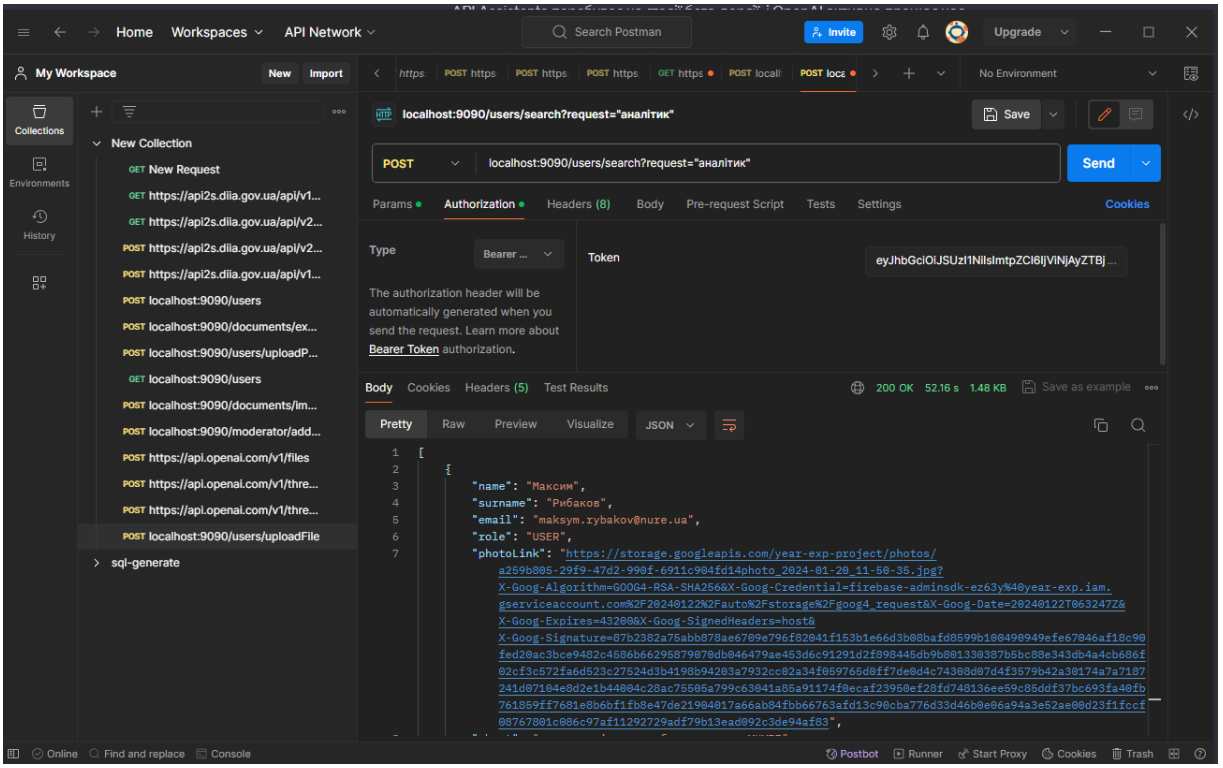

Рисунок 4.6.2 – Результат пошуку за запитом «аналітик»

Налаштування асистента наведені на рисунку 4.6.3.

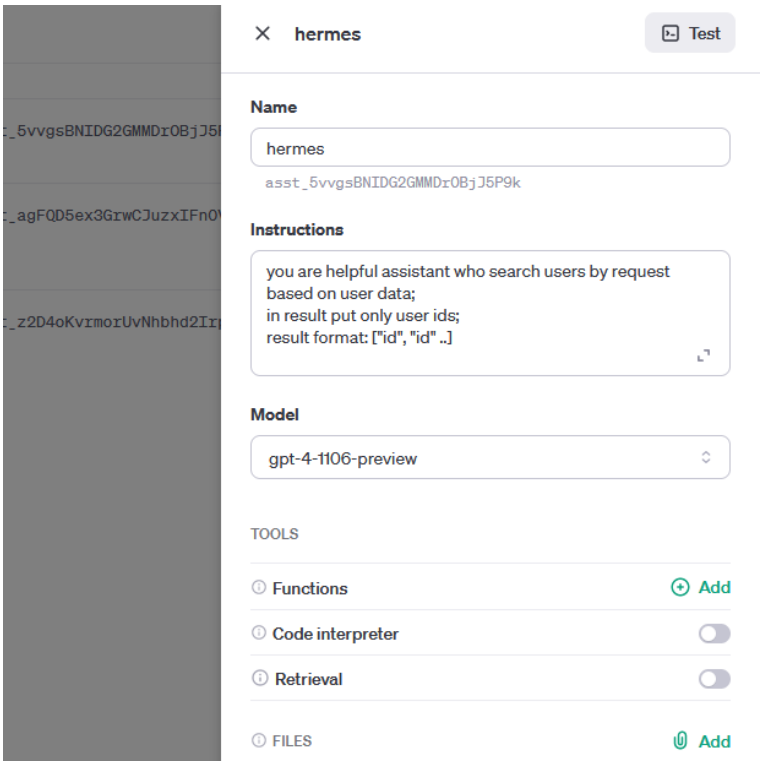

Рисунок 4.6.3 – Налаштування асистенту пошуку

Налаштування асистента обробки файлів наведені на рисунку 4.6.4.

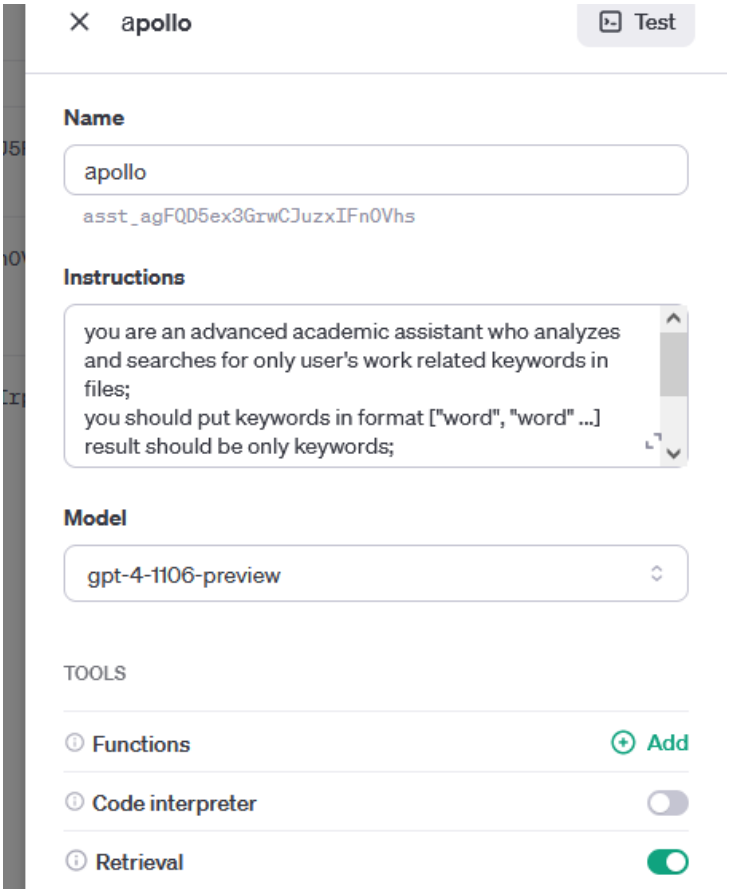

Рисунок 4.6.4 – Налаштування асистенту обробки файлів

Лістинг коду для взаємодії з асистентом аналізу файлів та виділення ключових слів наведено на код лістингу 4.6.2. Окрім того, на лістингу продемонстрована відправка файлу в сховище Open AI та взаємодія з отриманим fileId. Результат обробки файлу продемонстровано на рисунку 4.6.5, завантаження файлу згідно документації в сховище Open AI більш детально показано на рисунку 4.6.6.


```
_id: ObjectId('659b63c7dea3266a1b62cf36')
email: "yuliia.neroba@nure.ua"
photoLink: "photos/ae9d00e9-8a05-40cc-b907-e4a49da85aa53записати6.PNG"
name: "Василь"
surname: "Чарівник"
fatherName: "Гнатович"
about: "розробник і хороша людина (they/them)"
▸ educations: Array (2)
role: "USER"
keyWords: "["нечіткі множини", "операції над нечіткими множинами", "мета роботи",…"
▸ contacts: Object
position: "back-end developer"
_class: "year.exp.forge.domain.User"
```


Рисунок 4.6.5 – Результат обробки файлу

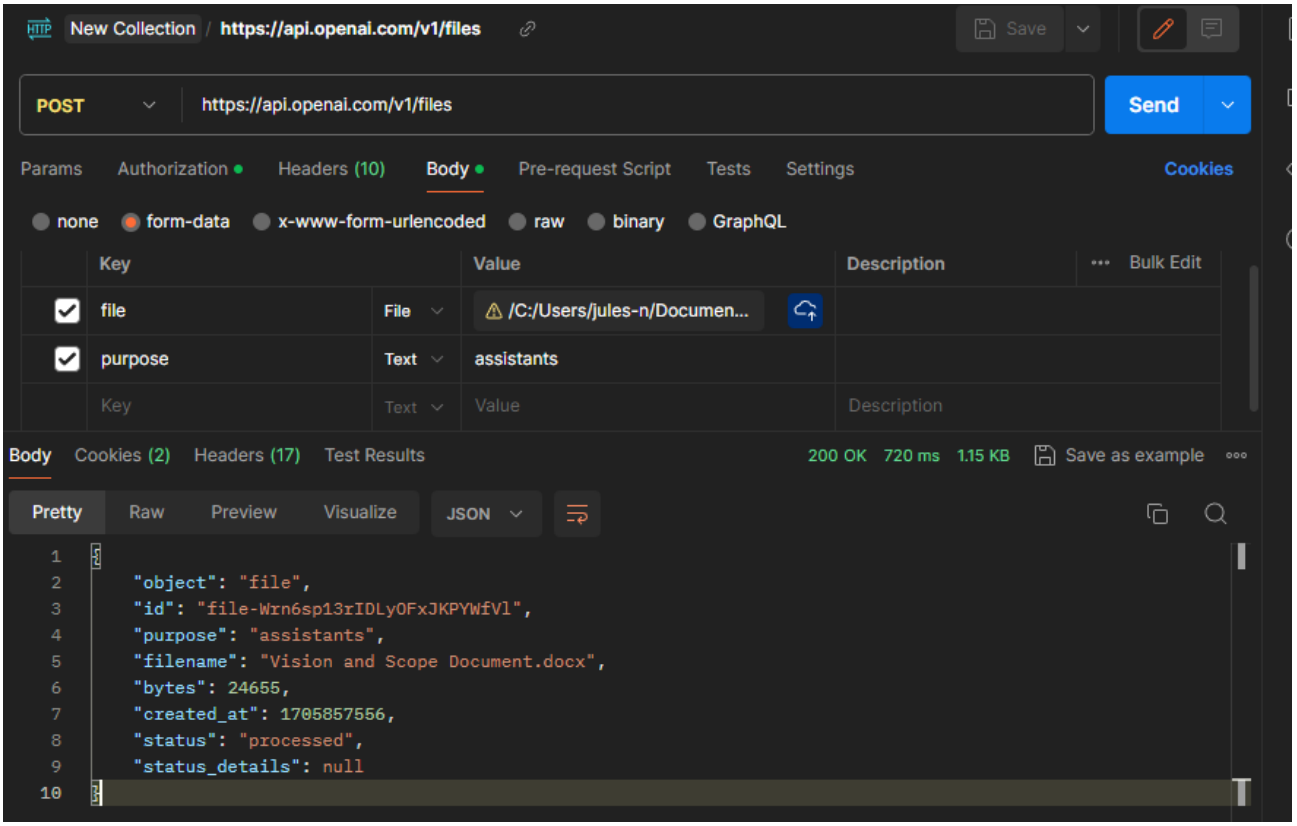

Рисунок 4.6.6 – Завантаження файлу в сховище Open AI відповідно до документації

```
public void uploadFile(HttpServletRequest request, MultipartFile file) throws
IOException, InterruptedException {
    String url = "https://api.openai.com/v1/files";
    String apiKey = "№№№";
    String fileId = null;
    try (CloseableHttpClient client = HttpClients.createDefault()) {
        HttpPost httpPost = new HttpPost(url);

        HttpEntity entity = MultipartEntityBuilder.create()
                .addBinaryBody("file", file.getInputStream(),
ContentType.DEFAULT_BINARY, file.getOriginalFilename())
                .addTextBody("purpose", "assistants")
                .build();

        httpPost.setEntity(entity);
        httpPost.setHeader("Authorization", "Bearer " + apiKey);

        String json = null;
        try (CloseableHttpResponse response = client.execute(httpPost)) {
            json = EntityUtils.toString(response.getEntity());
        }
        if (json == null) throw new RuntimeException("failed to upload file to
open ai");
        ObjectMapper mapper = new ObjectMapper();
        Map<String, Object> map = mapper.readValue(json, Map.class);
        fileId = map.get("id").toString();
    } catch (IOException e) {
        e.printStackTrace();
    }

    OpenAiService service = new OpenAiService(apiKey);
    CreateThreadAndRunRequest createThreadAndRunRequest = new
CreateThreadAndRunRequest();
    createThreadAndRunRequest.setAssistantId("asst_agFQD5ex3GrwCJuzxIFnOVhs");
    ThreadRequest threadRequest = new ThreadRequest();
    MessageRequest messageRequest = new MessageRequest();
    messageRequest.setRole("user");
    messageRequest.setContent("analyze and retrieve keywords from file with id "
+ fileId);
    messageRequest.setFileIds(List.of(fileId));
    threadRequest.setMessages(List.of(messageRequest));
    createThreadAndRunRequest.setThread(threadRequest);
    Run run = service.createThreadAndRun(createThreadAndRunRequest);
    while (!run.getStatus().equals("completed")) {
```

```
        Thread.sleep(6001);
        run = service.retrieveRun(run.getThreadId(), run.getId());
        if (run.getStatus().equals("failed") ||
run.getStatus().equals("cancelled") || run.getStatus().equals("expired")) {
            break;
        }
    }

    OpenAiResponse<Message> openAiResponse =
service.listMessages(run.getThreadId());
    String keyWords =
openAiResponse.getData().get(0).getContent().get(0).getText().getValue();

    year.exp.forge.domain.User user = getUser(request);
    user.setKeyWords(keyWords);
    userRepository.save(user);
}
```

Лістинг коду 4.6.2 – Обробка файлу з допомогою асистента Open AI

4.6.1 Експериментальні дослідження

Інструкції, що задаються для асистентів є фактично промтами для AI. В якості експерименту було проведено дослідження з поліпшенням промтів і визначення точності відповіді даних залежно від оновлених промтів.

Документ для тестування – звіт з лабораторної з предмету CALS. Бажаний результат: C language, Arduino, Atmel, Мікроконтролер, CALS. Кількість ітерацій – 20. Загальний огляд результатів експериментів наведено на рисунку 4.6.1.1, графік – 4.6.1.2.

instruction	result	precision
you are an assistant who searches for keywords in files	робота №1) from Kharkiv National University of Radio Electronics (ХАРКІВСЬКИЙ	0,8
you are an advanced academic assistant who searches for keywords in files; you sh	No. 1 (Звіт з лабораторної роботи №1) related to the course on CALS-	0,6
you should put keywords in csv format;	мікроконтролер,Atmel Studio 7,світлодіоди,програмування,PORTD,регістр порту	0,6
you are an advanced academic assistant who searches for only essential keywords	Архітектура,Програмування,Мікроконтролер,AVR,ATMEGA128,Розробка,Світлод	0,4
keywords in files	Studio 7 interface", "PORTD", "logical zero", "logical one"	0,4
user's work related keywords in files;	"cals-технології,мікроконтролери,avr,atmega128,світлодіоди,atmel studio 7,прог	0,8

Рисунок 4.6.1.1 – Огляд результатів

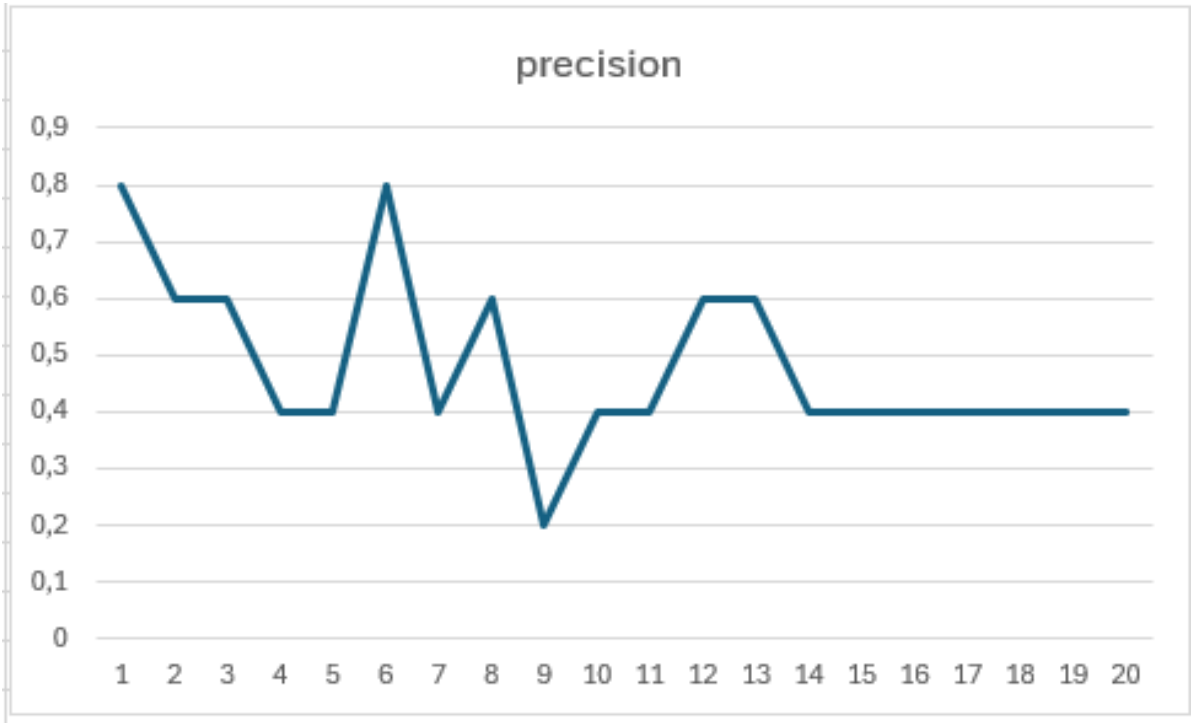

Рисунок 4.6.1.2 – Графік зміни точності залежно від промтів

Отже, необхідний промт, який мав найбільшу точність – 0.8. Інструкції даного промту: you are an advanced academic assistant who analyzes and searches for only; user's work related keywords in files; you should put keywords in format ["word", "word" ...]; result should be only keywords; do not include words for file organization; use Natural language processing; search on what and how exactly user worked.

4.7 Реалізація бекенду застосунку

Як вже зрозуміло з лістингів коду в попередніх розділах, мовою написання бекенду є Java, фреймворком – Spring Framework.

Java – кросплатформна мова програмування розроблена Sun Microsystem. Java є однією з найпопулярніших мов для розробки складних мережевих додатків і систем, які потребують одночасного виконання декількох завдань.

Spring Framework – java фреймворк для написання EE застосунків. Фреймворк надає комплексні рішення для побудови розширюваних та ефективних сервісів. Spring розроблений так, щоб спростити розробку та

підтримку Java-додатків, зокрема додатків на основі принципів інверсії керування та вставки залежностей. До обов'язкових Spring залежностей, що впроваджені та використані в мікросервісах належать:

– Spring Security;

– Spring Euruka;

– Spring OAuth;

– Spring Gateway.

4.7.1 Реалізація сервісу odyssey

Назва сервісу має посилання на давньогрецьку міфологію, призначення сервісу – слугувати централізованим входом у систему.

Сервіс включає бібліотеки:

– org.springframework.cloud:spring-cloud-starter-gateway;

– org.springframework.boot:spring-boot-starter-web;

– org.springframework.cloud:spring-cloud-starter-netflix-eureka-client;

– org.springframework.boot:spring-boot-starter-actuator.

Залежність org.springframework.cloud:spring-cloud-starter-netflix-eureka-client є геть необов'язковою, та саме зв'язок з discovery сервісом дозволяє не прив'язуватись до хостів та портів сервісів. Загальна структура сервісу наведена на рисунку 4.7.1.1, конфігурація шлюзу - 4.7.1.2. Повний код знаходиться в додатку Б.

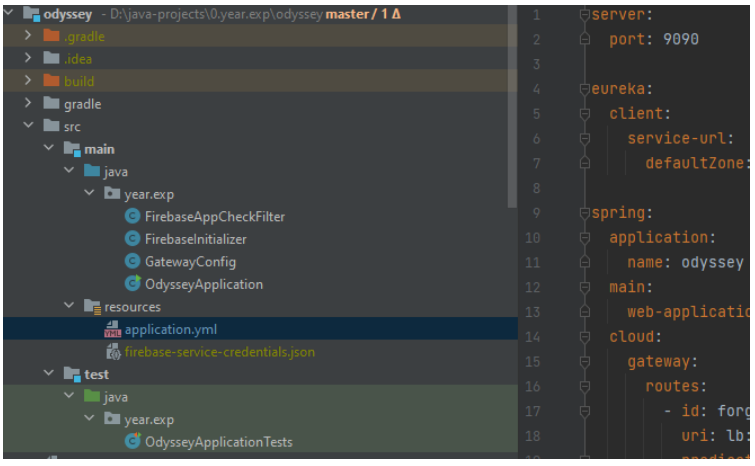

Рисунок 4.7.1.1 – Структура проекту

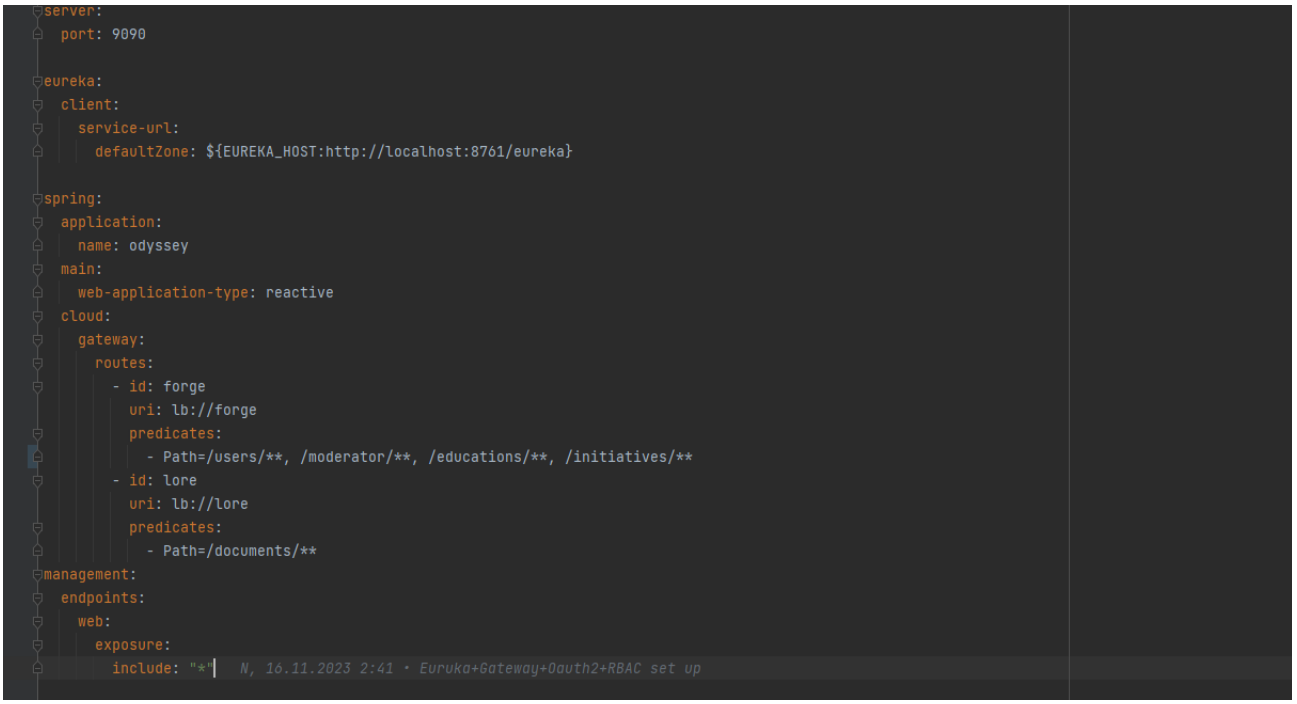

Рисунок 4.7.1.2 – Конфігурація шлюзу

4.7.2 Реалізація сервісу beacon

Beacon – service discovery, який розроблений на Spring Framework і потребує лише 2 бібліотеки для запуску:

– org.springframework.cloud:spring-cloud-starter-netflix-eureka-server;

– org.springframework.boot:spring-boot-starter-test.

Конфігурація сервісу наведена на рисунку 4.7.1.3. Обов’язковим також є додання анотації @EnableEurekaServer для класу, що містить метод main, котрий запускає застосунок.

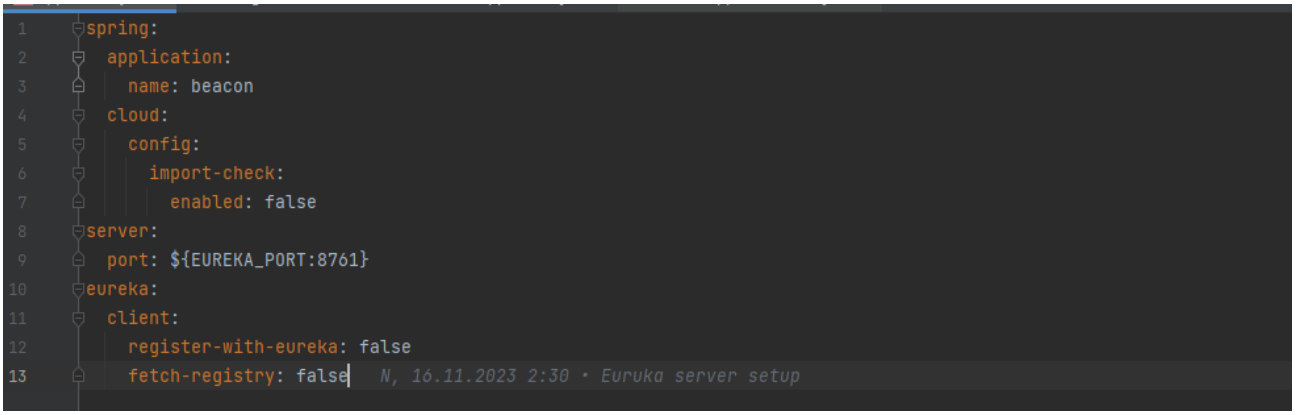

Рисунок 4.7.2.1 – Конфігурація сервісу

4.7.3 Реалізація сервісу lore

Lore – сервіс, основна мета даного сервісу – генерація та зберігання в бакеті google cloud storage угод. Сервіс має лише контролер – DocumentController з можливістю імпорту та експорту угод, що продемонстровано на рисунку 4.7.3.1. Сервіс не має БД, проте має зв'язок з сховищем даних.

Реалізація основної бізнес-задачі – генерація файлів – продемонстрована на код лістингу 4.7.3.1. Даний код додає в таблиці кофаундерів, їх рівні частки прибутку, дату генерації та назву стартапу. Результат генерації документу наведений на рисунках 4.7.3.2 – 4.7.3.4.

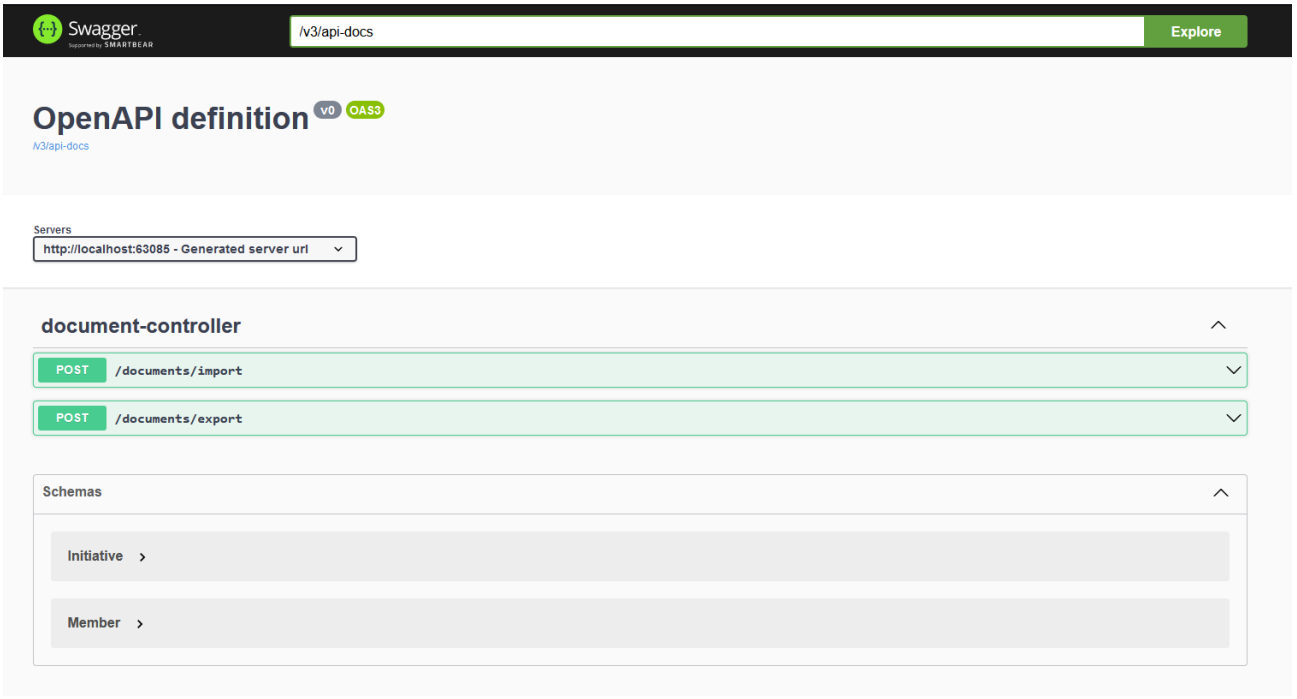

Рисунок 4.7.3.1 – Документація по контролерах сервісу lore

```
@Service
public class DocumentService {

    public String export(Initiative dto) throws IOException {
        String filePath = "src/main/resources/founders-agreement-template.docx";
        FileInputStream fis = new FileInputStream(filePath);
        XWPFDocument document = new XWPFDocument(fis);
        textReplace(document, dto.getName());
        addFounders(document, dto.getMembers());
        addShares(document, dto.getMembers());
```

```
        String path = "src/main/resources/tmp/"+dto.getId()+".docx";
        try (FileOutputStream out = new FileOutputStream(path)) {
            document.write(out);
        }
        document.close();
        return path;
    }
    private void textReplace(XWPFDocument document, String name) {
        for (XWPFParagraph paragraph : document.getParagraphs()) {
            List<XWPFRun> runs = paragraph.getRuns();
            if (runs != null) {
                for (XWPFRun run : runs) {
                    String text = run.getText(0);
                    if (text != null && text.contains("_")) {
                        String date =
LocalDate.now().format(DateTimeFormatter.ofPattern("dd.MM.yyyy"));
                        text = text.replace("_", date);
                        run.setText(text, 0);
                    }
                    if (text != null && text.contains("=")) {
                        text = text.replace("=", name);
                        run.setText(text, 0);
                    }
                }
            }
        }
    }

    private void addFounders(XWPFDocument document, List<Member> members) {
        if (!document.getTables().isEmpty()) {
            XWPFTable table = document.getTables().get(0);
            for (int i = 1; i <= members.size(); i++) {
                XWPFTableRow newRow = table.createRow();
                Member current = members.get(i-1);
                newRow.getCell(0).setText(current.toString());}}}

    private void addShares(XWPFDocument document, List<Member> members) {
        float shares = (float) 100/members.size();
        if (!document.getTables().isEmpty()) {
            XWPFTable table = document.getTables().get(1);
            for (int i = 1; i <= members.size(); i++) {
                XWPFTableRow newRow = table.createRow();
                Member current = members.get(i-1);
                newRow.getCell(0).setText(current.toString());
```

```
                newRow.getCell(1).setText(shares+"%");
            }}}}
```

Лістинг коду 4.7.3.1 – Сервіс генерації документів

FOUNDERS AGREEMENT

PARTIES

- This Founders Agreement (hereinafter referred to as the **"Agreement"**) is entered into on 13.01.2024 (the **"Effective Date"**), by and between the following:

Founder's name
Міщкевич Гриць Васильович
Міщкевич Валерія Василівна
Небийдіда Олег Євгенович

(collectively referred to as the **"Founders"**).

Рисунок 4.7.3.2 – Додані до угоди дата та кофаундери

BUSINESS VENTURE

- The Founders agree that Гаражі купувати has been created as the business venture (hereinafter, referred to as "**the Company**") located in Ukraine.

Рисунок 4.7.3.3 – Додана назва ініціативи

Founder's name	Shares
Міщкевич Гриць Васильович	33.333332%
Міщкевич Валерія Василівна	33.333332%
Небийдіда Олег Євгенович	33.333332%

MANAGEMENT STRUCTURE

Рисунок 4.7.3.4 – Додані частки прибутку та кофаундери

4.7.4 Реалізація сервісу forge

Forge – сервіс, що займається обробкою даних користувачів, ініціатив, освіт та запрошень. Попри комплексну задачу винести будь-яку бізнес задачу неможливо без уникнення імплементації антипатерну надмірної декомпозиції. Forge має декілька контролерів, що продемонстровані на рисунку 4.7.4.1.

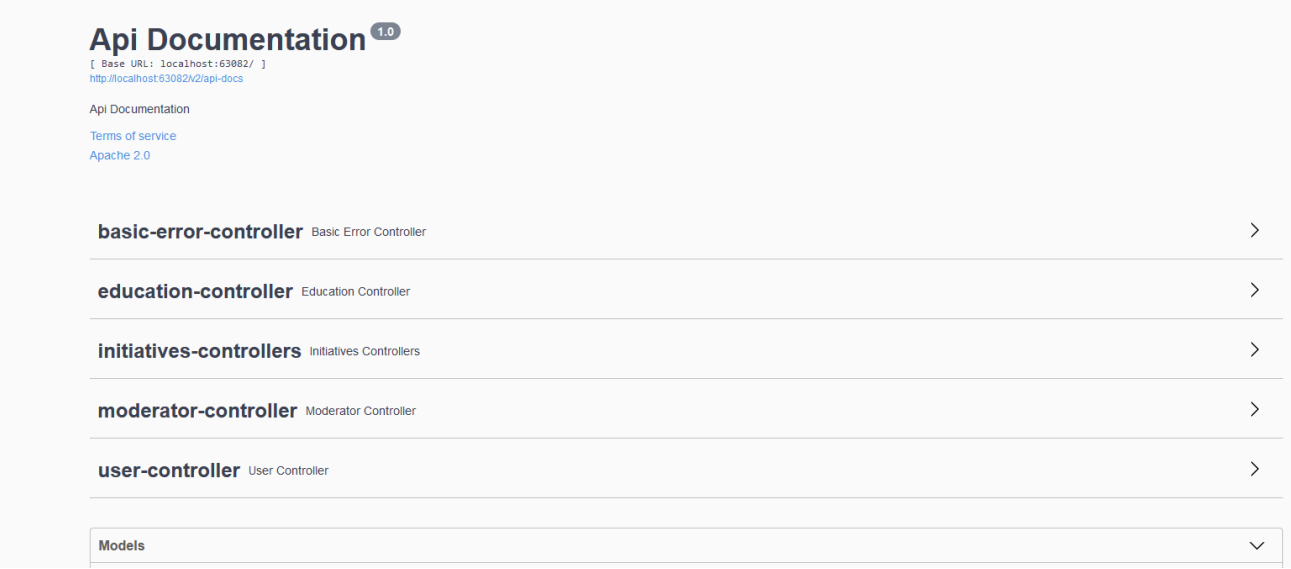

Рисунок 4.7.4.1 – Контролери forge

Контролер освіти містить ендпоінти, що необхідні для маніпуляцій з академічними даними користувачів. Перелік ендпоінтів наведено на рисунку 4.7.4.2.

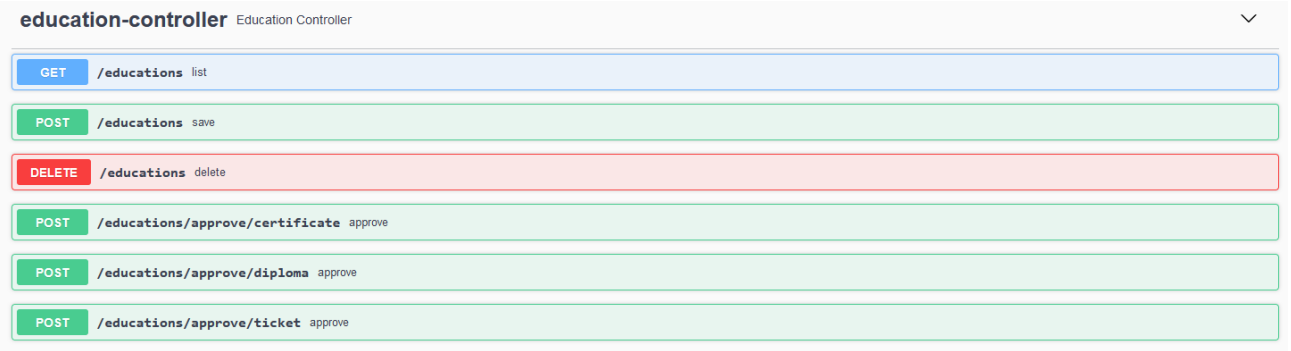

Рисунок 4.7.4.2 – Ендпоінти контролеру освіти

Повний перелік ендпоінтів інших контролерів та модельних класів містяться в додатку В. Оскільки, інтеграції з іншими сервісами вже були розглянуті, а в іншому сервіс лише обробляє дані та зберігає в базі даних, то даний код може бути винесений в додатки і там власне й знаходиться. Додаток з кодом застосунку – додаток Г.

4.8 Розробка дизайну

Прототипи розроблюваного дизайну представлені в додатку Д, загальний огляд наведено на рисунку 4.8.2. Дизайн був розроблений в сервісі Figma, котрий представляє собою інструмент для дизайну інтерфейсів, що нині належить компанії Adobe. Його особливістю є те, що він працює повністю в браузері, дозволяючи командам легко співпрацювати у реальному часі.

Використаний шрифт – KharkivTone, що був розроблений @kattte_drozd.

Створений логотип наведений на рисунку 4.8.1. В знаковій частині логотип поєднує абревіатуру назви проекту.

Рисунок 4.8.1 – Логотип проекту 0.year.exp

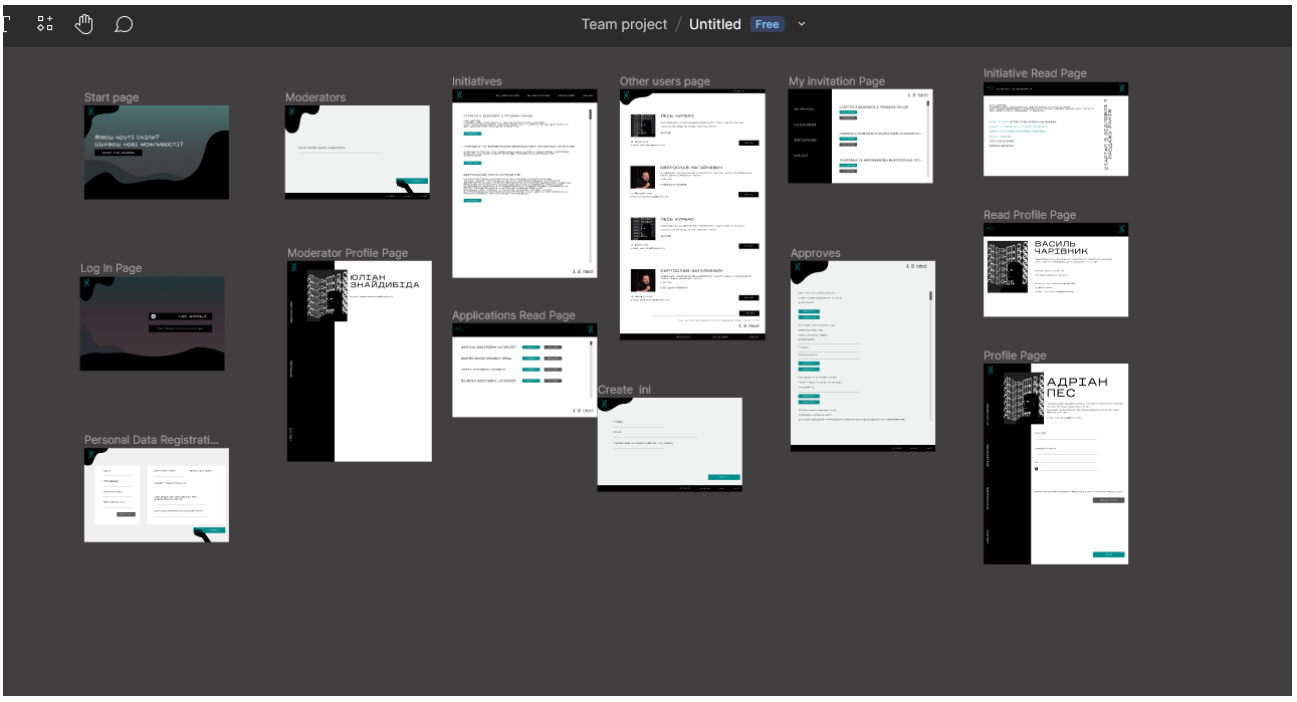

Рисунок 4.8.2 – Загальний огляд макетів для фронтенду застосунку 0.year.exp

4.9 Реалізація фронтенду застосунку

Як є очевидним з розділу 4.2 в якості мови написання фронтенду обрано React.js. React.js - це бібліотека JS, котра призначена для побудови ефективних та динамічних веб-додатків, які можуть легко взаємодіяти з користувачем та автоматично оновлювати відображення при зміні даних. Структура caryatid наведена на рисунку 4.9.

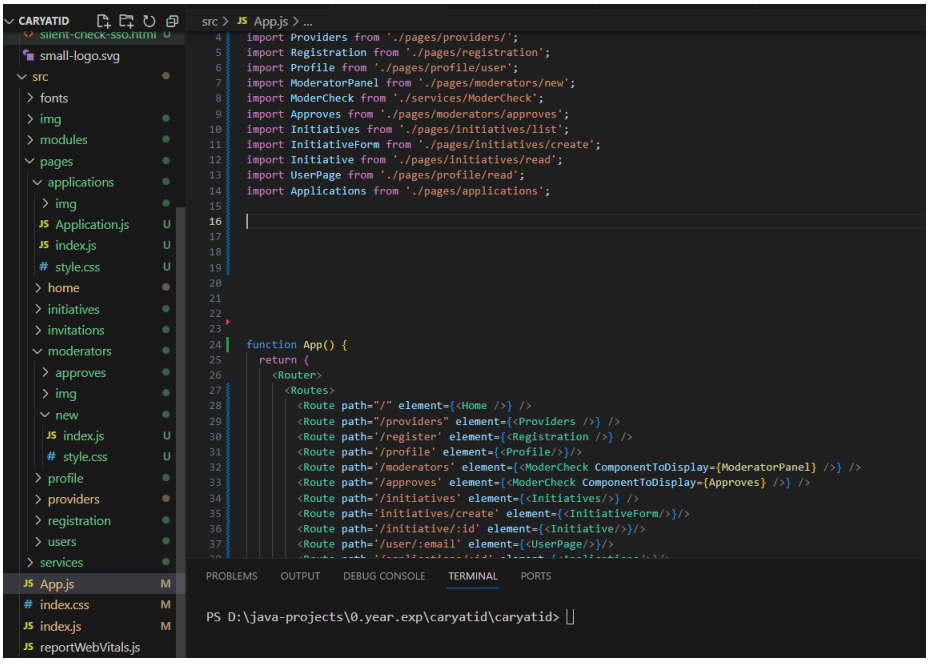

Рисунок 4.9.1 – Структура сервісу caryatid

Застосунок містить 3 сервіси:

– UserService – необхідний для зв’язку з Firebase Auth.

– HttpService – сервіс, що використовується для відправки запитів на серверну частину.

– ModerCheck – компонент, що попереджує доступ звичайних користувачів до сторінок, що мають бути доступні лише модераторам.

Частково лістинг коду UserService вже продемонстровано в частині 4.2. Код компоненту ModerCheck наведено в лістингу 4.9.1.

```
import React, { useState, useEffect } from 'react';
import {HttpService} from './HttpService';
import Verum from '../img/kavyarnya-prykoliv.gif';
```

```
const ModerCheck = ({ ComponentToDisplay }) => {
  const [isAdmin, setIsAdmin] = useState(false);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    const checkAdminStatus = async () => {
      setIsAdmin(false);
      try {
        const jwtToken = sessionStorage.getItem('token');
        if (jwtToken !== null || jwtToken !== undefined) {
          HttpService.setJwt(jwtToken);
          const response = await HttpService.get('/users/role');
          if (response.data === "MODERATOR") {
            setIsAdmin(true);
          }
        }
      } catch (error) {
        console.error('Error fetching data:', error);
      } finally {
        setIsLoading(false);
      }
    };

    checkAdminStatus();
  });

  if (isLoading) {
    return <div><img src={Verum} alt="Loading" /></div>;
  }

  return isAdmin ? <ComponentToDisplay /> : null;
};

export default ModerCheck;
```

Лістинг коду 4.9.1 – Компонент ModerCheck

Код сервісу HttpService наведено на лістингу коду 4.9.2. Фактично даний сервіс є обгорткою для axios. Axios - це JavaScript бібліотека, яка використовується для здійснення HTTP-запитів у Node.js або у браузерах. Axios дозволяє налаштовувати заголовки запитів, тайм-аути, параметри відповіді та інші аспекти HTTP-запитів та виконувати базові CRUD операції.

```
import axios from 'axios';

const HttpService = axios.create({
  headers: {
    'Content-Type': 'application/json',
    'Access-Control-Allow-Origin': '*'
  }
});

// Функція для встановлення токену
```

```
HttpService.setJwt = function(jwt) {
  this.defaults.headers.common['Authorization'] = `Bearer ${jwt}`;
};

export { HttpService };
```

Лістинг коду 4.9.2 – Сервіс HttpService

Інші сервіси представляють собою компоненти для відображення сторінок, код яких наведений в додатку Е.

4.10 Взаємодія з БД

Виходячи з неоднорідності даних та необхідної швидкодії, рішенням, що чудово підходить за цими параметрами є MongoDB, котра представляє собою документ-орієнтовану систему керування базами даних, яка використовує BSON документи для зберігання даних. Відсутність завчасно визначеної схеми є ключовою перевагою даної БД. Це означає, що документи в одній і тій же колекції можуть мати різні поля.

MongoDB широко використовується для створення масштабованих додатків, завдяки своїй гнучкості, продуктивності та легкості у використанні. Завдяки оптимізованим індексам та можливості зберігання пов'язаних даних в одному документі, MongoDB може надавати високу продуктивність для операцій читання та запису.

Перелік колекцій наведено на рисунку 4.10.1.

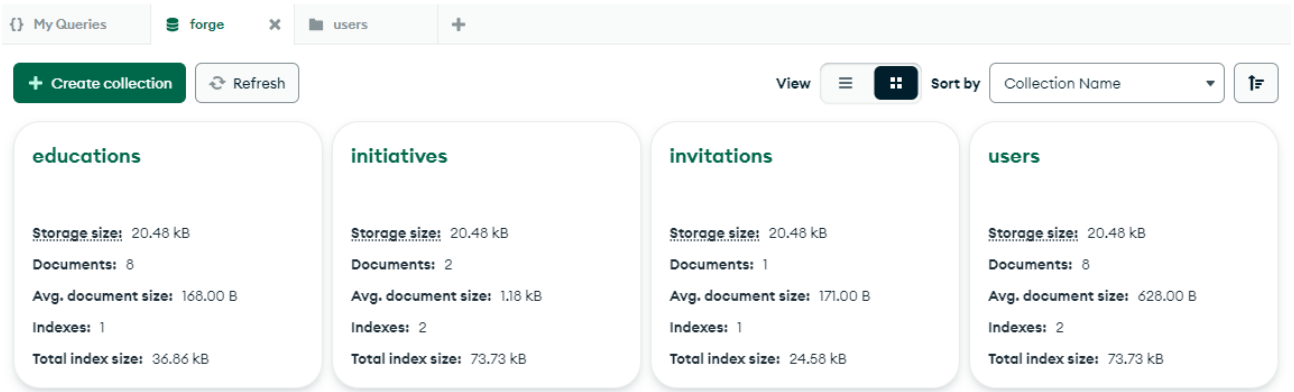

Рисунок 4.10.1 – Перелік колекцій

Іншою нереляційною БД, що використовується в застосунку є Redis. Redis є відкритим, ключ-значення сховищем даних в пам'яті, яке часто використовується як база даних, кеш і брокер повідомлень. Оскільки Redis зберігає дані в оперативній пам'яті (RAM), він забезпечує дуже швидкий доступ до даних, з тисячами операцій на секунду, що робить його ідеальним для використання в якості системи кешування.

Запуск Redis здійснюється з Docker, що продемонстровано на рисунку 4.10.2.

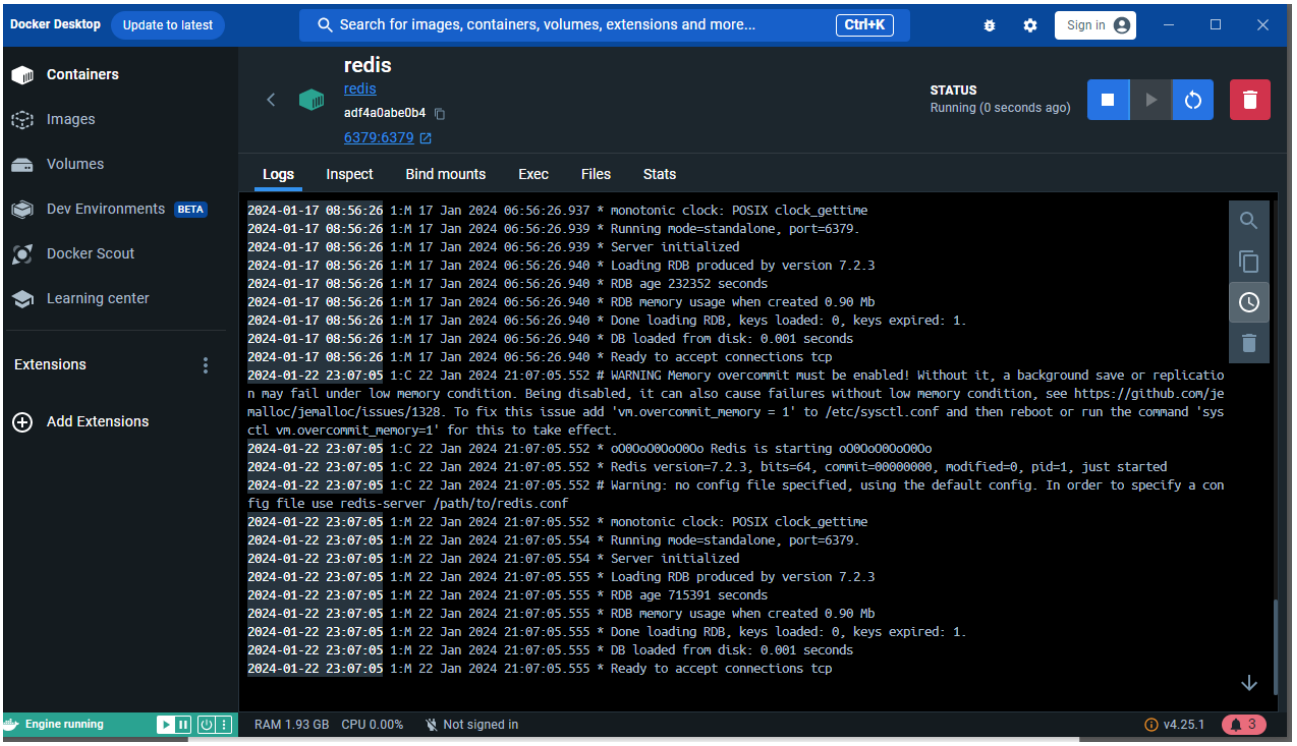

Рисунок 4.10.2 – Робота Redis

Налаштування для зв’язку з базами даних продемонстровано на лістингу коду 4.10.1.

```
redis:
  host: localhost
  port: 6379
application:
  name: forge
data:
  mongodb:
```

```
  host: ${MONGODB_HOST:localhost}
  port: ${MONGODB_PORT:27017}
  database: ${MONGODB_DATABASE:forge}
  autoIndexCreation: true
```

Лістинг коду 4.10.1 – Дані для підключення до БД

Отже, в результаті додаток має архітектуру, що продемонстрована на рисунку 4.1, реалізовані заявлені вимоги щодо дизайну, безпеки та функціоналу.

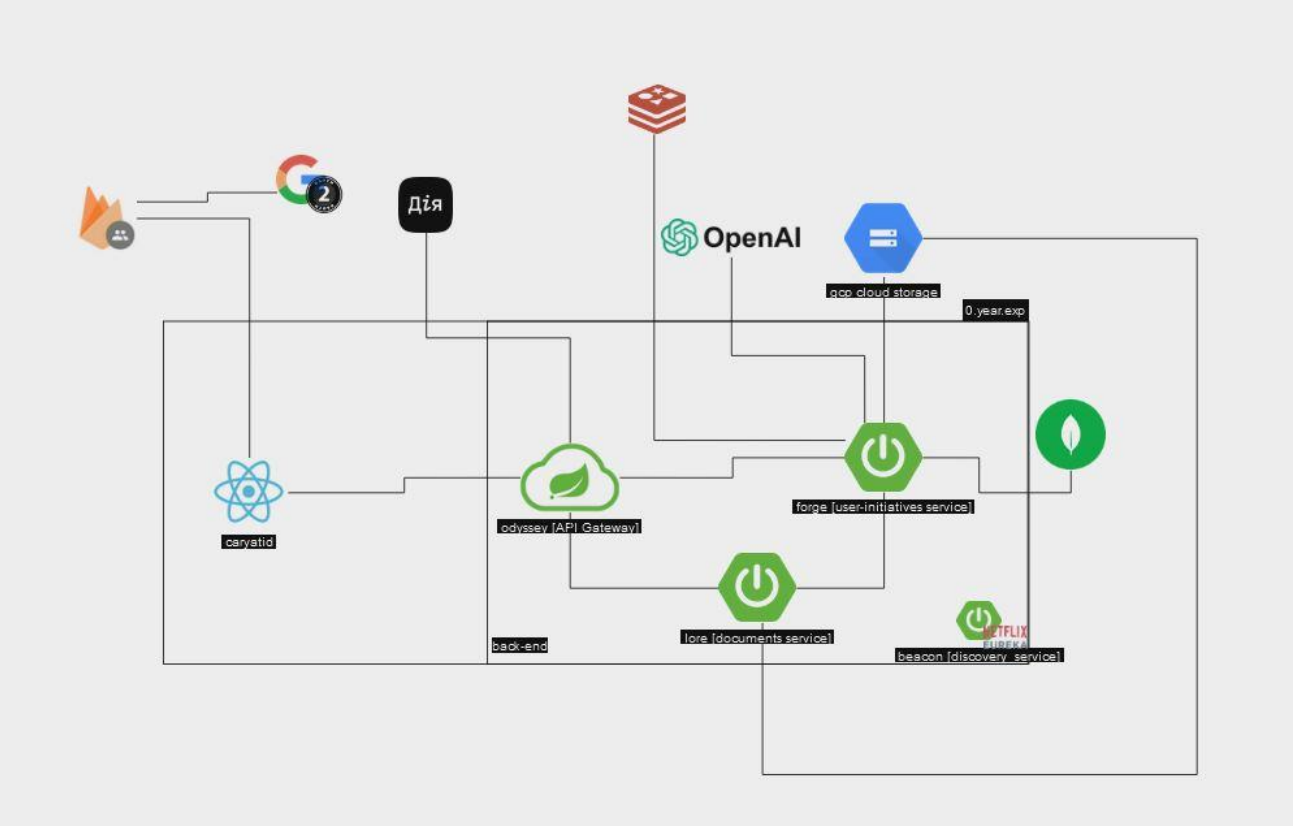

Рисунок 4.1 - Архітектура веб-сервісу

ВИСНОВКИ

Проблематика працевлаштування була і буде залишатись актуальною для суспільства з вільною ринковою економікою. Розроблений застосунок є лише інструментом, що може стати в нагоді початківцям або людям, що змінюють напрям роботи.

При розробці було вивчено та проаналізовано технічну документацію фреймворків, сторонніх API й теоретичних відомостей про них, було проаналізовано аналогічні системи, виділено їх переваги та недоліки й побудовано на основі цього власну систему-нетворкінг. Програмно дана система є гнучкою, здатною до масштабування, безпечною та ефективною. При її написанні використано Java, Spring, MongoDB, Redis та React.js.

Експерементальною частиною дослідження стало тренування вже існуючої AI моделі та взаємодія з нею. На основі експерементальних даних визначено інфструкції за яких асистент найбільш точно виконує запит користувача.

Результатом дослідження є реалізована MVP версія нетворкінгу для пошуку студентів кофаундерів для заснування стартапів з допомогою AI при пошуку потенційних учасників ініціатив.

При тестуванні перевірено, що платформа є працездатною і оброблювані дані є коректними та відповідають очікуванням.

Розроблена платформа може бути вдосконалена в майбутньому за рахунок:

- розширення функціоналу з додаванням листувань;
- розсилок;
- автоматизованих інтеграцій перевірки освіти;
- покращення натренованості AI;
- поліпшення безпеки;
- включення нових провайдерів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. «Spring Security in Action» [Електронне видання] / Автор: Laurentiu Spilka – 2020. – 560с.

2. «MongoDB in Action: Covers MongoDB Version 3.0» [Електронне видання] / Автор.: Peter Baccum, Kyle Banker, Sean Ferh, Doug Garrett– 2016. – 480с.

3. «Філософія Java четверте видання» [Електронне видання] /Автор: Еккель Б. – Пітер, 2016. – 1168 с.

4. «Learning UML 2.0: A Pragmatic Introduction to UML» [Електронне видання] / Автори: Кім Гамільтон, Расс Майлз - 2006. – 286 с.

5. UI/UX Starter Guide [Електронний ресурс] URL: https://www.figma.com/community/file/1019904134655257869 (Дата звернення: 08.01.2024)

6. «Екстремальне програмування: розробка через тестування» [Електронне видання] / Автор: Кент Бек – 2019. – 224с.

7. «Чиста архітектура. Мистецтво розробки програмного забезпечення» [Електронне видання] / Автор: Роберт Мартін – 2018. – 352 с.

8. «Lean Analytics: Use Data to Build a Better Startup Faster» [Електронне видання] / Автор: Алістер Кролл, Бенджамін Йосковіц – 2013. – 439с.

10. «Microservices Patterns: With examples in Java 1st Edition» [Електронне видання] / Автор: Chris Richardson – 2018. – 520с.

11. MongoDB docs [Електронний ресурс] URL: https://www.mongodb.com/docs/ (Дата звернення: 01.01.2024)

12. Microservice architecture patterns [Електронний ресурс] URL: https://microservices.io/patterns/microservices.html Дата звернення (06.01.2024)

13. Java docs [Електронний ресурс] URL: https://docs.oracle.com/en/ (Дата звернення: 15.01.2024)

14. React docs [Електронний ресурс] URL: https://uk.reactjs.org/docs/getting-started.html (Дата звернення: 15.01.2024)

15. «Modern Java in Action: Lambdas, streams, functional and reactive

programming» [Електронне видання] / Автори: Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft – 2018. - 592с.

16. Axios docs [Електронний ресурс] URL: https://axios-http.com/uk/docs/intro (Дата звернення: 01.01.2024)

17. GitHub docs [Електронний ресурс] URL: https://docs.github.com/en/get-started/quickstart/hello-world (Дата звернення: 01.01.2024)

19. SpringFox Swagger Docs [Електронний ресурс] URL: https://springfox.github.io/springfox/docs/current/ (Дата звернення: 01.12.2023)

20. Gradle Docs [Електронний ресурс] URL: https://docs.gradle.org/current/userguide/userguide.html (Дата звернення: 01.11.2023)

22. Spring Security article [Електронний ресурс] URL: https://www.baeldung.com/spring-security-method-security (Дата звернення: 9.11.2023)

23. Redis Docs [Електронний ресурс] URL: https://redis.io/docs/ (Дата звернення: 01.01.2024)

24. «Spring Microservices in Action 1st Edition» [Електронне видання] / Автор: John Carnell - 2017. – 384с.

25. «Head First Object-Oriented Analysis and Design 1st Edition» [Електронне видання] / Автори: Brett D. McLaughlin , Gary Pollice , Dave West - 2006. – 636с.

26. Open AI Docs [Електронний ресурс] URL: https://platform.openai.com/docs/api-reference (Дата звернення: 01.01.2024)

27. Diia Docs [Електронний ресурс] URL: https://diia.gov.ua/faq/23 (Дата звернення: 01.01.2024)

28. Google Docs [Електронний ресурс] URL: https://cloud.google.com/docs (Дата звернення: 01.01.2024)