

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

В.о. завідувача кафедри

Михайло НОВОТАРСЬКИЙ

(підпис)

“__” _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Сервіс для створення та проведення соціальних опитувань

Виконав : студент 4 курсу, групи ІО-15
(шифр групи)

Жердєв Данило Олександрович

(прізвище, ім’я, по батькові)

(підпис)

Керівник асистент, Гордієнко Н. Ю.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант (нормоконтроль) ас. Пономаренко А.М.

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Рецензент асистент Соколовський Владислав Володимирович

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

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

Студент _____

(підпис)

Київ – 2025 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

Михайло НОВОТАРСЬКИЙ

(підпис)

“ ” _____ 2025 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Жердева Данила Олександровича

1. Тема проєкту Сервіс для створення та проведення соціальних опитувань
керівник проєкту Гордієнко Нікіта Юрійович, асистент,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 23 травня 2025 року №1705-с

2. Термін здачі студентом закінченого проєкту 3 червня 2025 р.

3. Вихідні дані до проєкту технічна документація, теоретичні дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1.Огляд існуючих рішень для соціальних опитувань.

Розділ 2. Огляд та вибір інструментів розробки.

Розділ 3. Деталі розробки сервісу.

Розділ 4. Дослідження та аналіз розробленого застосунку.

5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Пономаренко А. М.		

7. Дата видачі завдання «7» жовтня 2024 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	7.10.2024-22.01.2025	
2.	<i>Вивчення та аналіз завдання</i>	23.01.2025-16.02.2025	
3.	<i>Розробка архітектури та загальної структури системи</i>	17.02.2025-10.03.2025	
4.	<i>Розробка структур окремих підсистем</i>	11.03.2025-04.04.2025	
5.	<i>Програмна реалізація системи</i>	10.05.2025-25.05.2025	
6.	<i>Оформлення пояснювальної записки</i>	26.05.2025-01.06.2025	
7.	<i>Передзахист</i>	9.06.2025	
8.	<i>Захист</i>	21.06.2025	

Студент-дипломник _____ Данило ЖЕРДСВ
(підпис)

Керівник проєкту _____ Нікіта ГОРДІЄНКО
(підпис)

АНОТАЦІЯ

У даній бакалаврській роботі було розроблено веб-застосунок для створення та проведення соціальних опитувань. Основною метою застосунку є надання інструменту для збору думок користувачів в онлайн форматі. Враховуючи важливість безпеки для такої системи в роботі, особливу увагу було приділено інструментам захисту від зовнішніх втручань, а також перевірку чи є користувач ботом.

В застосунку реалізовані механізми автентифікації, авторизації, шифрування критичних даних, а також захисту API від різних видів кібератак.

Для створення веб-застосунку для серверної частини використовувалася мова програмування Java та її фреймворк Spring, що дає можливість створити масштабований та безпечний REST-сервіс. Для клієнтської частини було використано фреймворк Angular, а в якості бази даних було використано PostgreSQL.

ANNOTATION

In this bachelor's thesis, a web application for creating and conducting social surveys was developed.

The main purpose of the application is to provide a tool for collecting user opinions in an online format. Given the importance of security for such a system, special attention was paid to protection tools against external interference, as well as bot detection.

The application implements authentication, authorization, encryption of critical data, and protection of the API against various types of cyberattacks.

For the server-side development, the Java programming language was used along with the Spring framework, enabling the creation of a scalable and secure REST service. For the client-side, the Angular framework was used, and PostgreSQL was chosen as the database.

справки	Формат	Значення			Найменування	Кіл. листів	№ екземпля	Додаток
					Документація загальна			
					Знову розроблена			
	A4	ІАЛЦ.467200.002 ТЗ			Сервіс для створення та	3		
					проведення соціальних			
					опитувань			
					Технічне завдання			
	A4	ІАЛЦ.467200.003 ПЗ			Сервіс для створення та	82		
					проведення соціальних			
					опитувань			
					Пояснювальна записка			
	A4	ІАЛЦ.467200.004 Д1			Сервіс для створення та	1		
					проведення соціальних			
					опитувань			
					Структурна схема			
					системи			
	A4	ІАЛЦ.4672008.005 Д2			Сервіс для створення та	1		
					проведення соціальних			
					опитувань			
					Функціональна схема			
					(діаграма бази даних)			
	A4	ІАЛЦ.467200.006 Д3			Сервіс для створення та	1		
					проведення соціальних			
					опитувань			
					Алгоритм дій			
					веб-застосунку			
	A4	ІАЛЦ.467200.007 Д4			Сервіс для створення та	57		
					проведення соціальних			
					опитувань			
					Текст програмного коду			
					ІАЛЦ.467200.001 ОА			
Зм	Лист	№ докум.	Підп	Дата				
Розроб		Жердев Д.О			Сервіс для створення та проведення соціальних опитувань Опис альбому		Літ.	Арку ш
Перев		Гордієнко Н.Ю.						Аркушів
					НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15			

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «Сервіс для створення та проведення соціальних опитувань»

Київ – 2025

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	2
2 ПІДСТАВИ ДЛЯ РОЗРОБКИ	2
3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4 ДЖЕРЕЛА РОЗРОБКИ.....	2
5 ТЕХНІЧНІ ВИМОГИ.....	3
5.1 Вимоги до розробленого продукту	3
5.2 Вимоги до програмного забезпечення	3
6 ЕТАПИ РОЗРОБКИ	3

					ІАЛЦ.467200.002 ТЗ			
		№ докум.	Підпис	Дата				
Розробив	Жердев Д. О.				Сервіс для створення та проведення соціальних опитувань Технічне завдання	Літ.	Аркуш	Аркушів
Перевірив	Гордієнко Н. Ю.						1	3
Реценз.	Соколовський В.В.					НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15		
Н. Контр.	Пономаренко А. М							
Затвердив								

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на дипломний проєкт бакалавра за темою “Сервіс для створення та проведення соціальних опитувань”. Застосунок може бути практично використаний для отримання інформації про думку користувачів стосовно якогось питання.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання роботи кваліфікаційно-освітнього рівня бакалавр спеціальності 123 “Комп’ютерна інженерія” по освітньо-професійній програмі “Комп’ютерні системи та мережі”, який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут ім.Ігоря Сікорського”

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробки веб-застосунку для створення та проведення соціальних опитувань, з забезпеченням високих стандартів захисту від кібератак та збереженням конфіденційних даних користувачів

4 ДЖЕРЕЛА РОЗРОБКИ

Основними джерелами інформації для розробки дипломного проєкту є статті в мережі Інтернет, наукова-технічна література, інформація отримана з пройдених дисциплін та загальнодоступна документація існуючих програм.

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Простий і інтуїтивно-зрозумілий інтерфейс системи
- Надати можливість проходити користувачу реєстрацію та авторизацію
- Надати можливість створювати, редагувати, видаляти опитування
- Надати можливість користувачу проходити створене опитування
- Забезпечити безпеку особистих даних користувачів

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux
- IntelliJ IDEA
- Visual Studio Code
- Java Development Kit (JDK) 17 версії
- Node.js версії 18.10.0
- Angular 16 версії

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	07.10.2024-22.01.2025
Вивчення та аналіз завдання	23.01.2025-16.02.2025
Розробка архітектури та загальної структури системи	17.02.2025-10.03.2025
Розробка структур окремих частин системи	11.03.2025-04.04.2025
Програмна реалізація системи	10.05.2025-01.06.2025
Оформлення пояснювальної записки	10.05.2025-03.06.2025

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Сервіс для створення та проведення соціальних опитувань»

Київ – 2025

3MICT

ПЕРЕЛІК СКОРОЧЕНЬ.....	4
ВСТУП	5
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	7
1.1 Еволюція інструментів соціальних опитувань	7
1.1.1 Історичний огляд традиційних методів опитування	7
1.1.2 Сучасні проблеми та вимоги до онлайн сервісів створення та проведення опитувань	8
1.2 Аналіз існуючих рішень	9
ВИСНОВОК ДО РОЗДІЛУ 1	17
РОЗДІЛ 2 ОГЛЯД ТА ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ.....	18
2.1 Архітектурні підходи до розробки веб-сервесів.....	18
2.1.1 Монолітна архітектура	19
2.1.2 Мікросервісна архітектура.....	21
2.1.3 Server-less архітектура.....	23
2.2 Огляд сучасних фреймворків для розробки	25
2.2.1 Back-end фреймворки	26
2.2.2 Front-end фреймворки.....	30
2.3 Аналіз систем керування базами даних.....	34
2.3.1 Реляційні бази даних та системи керування ними	35
2.3.2 Нереляційні бази даних та їх особливості.....	36
2.4 Вибір технологічного стеку	38
ВИСНОВОК ДО РОЗДІЛУ 2	39
РОЗДІЛ 3 ДЕТАЛІ РОЗРОБКИ СЕРВІСУ	40
3.1 Розробка серверної частини застосунку	40
3.1.1 Початкова конфігурація	40
3.1.2 Створення моделі даних.....	44

					ІАЛЦ.467200.003 ПЗ				
Зм.	Арк.	№ докум.	Підпис	Дата	Сервіс для створення та проведення соціальних опитувань Пояснювальна записка	Літ.	Аркуш	Аркушів	
Розробив		Жердев Д.О.						1	82
Перевірив		Гордієнко Н.Ю.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15			
Реценз.		Соколовський В.В.							
Н. Контр.		Пономаренко А.М.							
Затвердив									

3.1.3 Реалізація бізнес логіки застосунку	47
3.1.4 Розробка контролерів	49
3.1.5 Створення Security рівня	50
3.1 Розробка клієнтської частини	55
3.2.1 Ініціалізація Angular проєкту	55
3.2.2 Створення компонента реєстрації та автентифікації	57
3.2.3 Розробка сервісів	58
3.2.3 Створення компонентів	59
3.2.3 Розробка шаблонів	61
ВИСНОВОК ДО РОЗДІЛУ 3	63
РОЗДІЛ 4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ	64
4.1 Огляд головної сторінки застосунку	64
4.2 Тестування реєстрації та автентифікації	66
4.3 Тестування створення опитування	68
4.4 Тестування можливостей редагування опитування та генерації результатів проходження	71
4.5 Тестування функціоналу проходження опитування	74
ВИСНОВОК ДО РОЗДІЛУ 4	77
ВИСНОВКИ	78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	80

ПЕРЕЛІК СКОРОЧЕНЬ

API	(Application Programming Interface) Інтерфейс програмного забезпечення
JSON	(JavaScript Object Notation) Нотація об'єктів JavaScript
ORM	(Object-Relational Mapping) Об'єктно-реляційне відображення
SQL	(Structured Query Language) Структурована мова запитів
HTML	(HyperText Markup Language) Мова гіпертекстової розмітки
CSRF	(Cross-Site Request Forgery) Підроблення міжсайтового запиту
URL	(Uniform Resource Locator) Уніфікований локатор ресурсу
SPA	(Single Page Application) Односторінковий додаток
SSR	(Server-Side Rendering) Рендеринг на стороні сервера
SSG	(Static Site Generation) Генерація статичного сайту
JWT	(JSON Web Token) JSON-веб токен
UI	(User Interface) Інтерфейс користувача
UX	(User Experience) Досвід користувача
СКБД	Система Керування Базою Даних
СКРБД	Система Керування Реляційною Базою Даних
БД	База Даних

ВСТУП

У наш час думка громадян є важливим інструментом для формування політик у різних сферах життя, починаючи від державного управління й закінчуючи маркетинговими стратегіями. Соціальні опитування є одним з основних інструментів для отримання громадської позиції стосовно певного питання. Завдяки переходу таких опитувань у онлайн-формат у користувачів з'явилася можливість зручно, без прив'язки до часу та місця брати в них участь, відповідно збільшення аудиторії дає більш точне розуміння, що необхідно користувачам. Та на відміну від опитувань в офлайн форматі, такі сервіси спрощують обробку та аналіз результатів опитування.

Проте з появою переваг в онлайн сервісах проведення опитувань, також виникають нові виклики, пов'язані з захистом таких систем, а саме захист персональних даних, гарантування анонімності респондентів, а також протидія маніпуляціям із результатами опитувань. Такі аспекти є особливо критичними, коли мова йде про дослідження соціально важливих тем, вибори або формування публічної політики.

Після проведення аналізу існуючих рішень на ринку, можна побачити, що більшість популярних платформ хоч і надають базовий функціонал для створення опитувань, але не завжди відповідають вимогам щодо конфіденційності, самостійного контролю над зібраною інформацією, або стійкості до атак та маніпуляцій над результатами опитувань. В свою чергу open-source рішення вимагають складної конфігурації, та не завжди здатні до масштабування для реальних умов застосування

Саме тому виникає необхідність у створенні власного веб-застосунку, який зможе поєднати простоту використання з високим рівнем захисту даних. Він має забезпечити гнучке налаштування опитувань, захист від

					ІАЛЦ.467200.003 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

автоматизованих ботів, шифрування критичних даних й автентифікацію користувачів.

					ІАЛЦ.467200.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Еволюція інструментів соціальних опитувань

1.1.1 Історичний огляд традиційних методів опитування

Ще з давніх часів соціальні опитування є основним інструментом збору думок населення. До початку цифрової епохи для проведення опитувань основними методами залишалися паперові анкети, телефонні дзвінки, інтерв'ю “обличчям до обличчя” або поштової розсилки. Завдяки таким способам дослідники все таки могли отримувати цінну інформацію, хоча обмеження в охопленні та ефективності на той час були критичні. Такі процеси, як збирання, транспортування, обробка та аналіз даних були дуже трудомісткими та напругу залежали від людського фактору. Крім того, в вищесказаних варіаціях проведення опитувань було майже неможливо забезпечити анонімність та достовірність відповідей.

З початком цифровізації ситуація доволі сильно змінилася. Використання доступу до мережі інтернет та зростання кількості цифрових пристроїв у звичайних громадян створило підґрунтя для активного впровадження онлайн опитувань. В цей період часу компаніями були створені зручні та функціональні платформи, такі як Google Forms, Typeform, SurveyMonkey та інші. Вони дали можливість користувачам самостійно створювати анкети, та поширювати їх за допомогою електронної пошти або соціальних мереж, а також доволі оперативно отримувати результати. Використання таких платформ дозволило зменшити витрати на проведення досліджень, прискорило аналіз відповідей, а також збільшило можливу кількість учасників опитування. З розвитком можливостей пристроїв, та можливостей веб-застосунків відкрилися

					ІАЛЦ.467200.003 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

можливості для візуалізації результатів, інтеграції з іншими веб-сервісами та автоматизації звітності.

1.1.2 Сучасні проблеми та вимоги до онлайн сервісів створення та проведення опитувань

Проте, як це часто і буває, те що має переваги, те буде мати і свої недоліки. Стрімка цифровізація соціальних досліджень породила нові проблеми, які потребують комплексного підходу на стику технологій, права та соціальної психології.

Сучасні платформи для збору даних прибирають межу “анонімним” та “персоніфікованим” опитуванням. Багато досліджень показують, що більша частина популярних сервісів фіксують метадані, в які може входити геолокація, цифрові відбитки браузера(fingerprinting) та поведінковий паттерн(час проведення на кожному питанні). Таким чином такі платформи створюють такий собі парадокс, в той час як користувач має ілюзію анонімності, такі системи збирають достатньо даних для деанонімізації.

Також окрім недоліків збору інформації стосовно особистості самими платформами завжди є можливість для кібератак, або зламу сервісу, яке може спричинити, як витік конфіденційних даних користувачів, так і маніпуляцією над результатами опитування. До таких атак можуть відноситися накрутка відповідей за допомогою ботів, масове проходження опитувань з підробленими обліковими записами, атаки типу Ddos, а також ін'єкції шкідливого коду через форми введення.

Тож відповідно до вищесказаного можна зрозуміти, що сучасні системи опитувань повинні відповідати технологічним та соціальним вимогам. Вони мають надавати, як зрозумілий інтерфейс та простоту використання, для забезпечення можливості розуміння використання сервісом людиною будь-

					ІАЛЦ.467200.003 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

якого віку, так і високий стандарт безпеки даних користувачів та результатів опитування.

Також важливо зазначити, що в багатьох сферах, таких як державне управління, охорона здоров'я, освіта є підвищений інтерес до інтеграції такої системи з іншими ІТ-інфраструктурами, але це висуває додаткові вимоги до захищеності API, підтримка автентифікації через державні платформи(наприклад “Дія”)

1.2 Аналіз існуючих рішень

Перед розробкою власного застосунку необхідно зробити аналіз вже існуючих рішень, оглянути їх переваги та недоліки, для розуміння, як має виглядати та функціонувати застосунок, та при цьому спробувати позбутися негативних моментів в популярних сервісах для створення опитувань.

Google Forms

Google forms є, напевно, найпопулярнішим інструментом для проведення онлайн-опитувань. Він доволі активно використовується в освітніх закладах, бізнес-середовищі та в громадських організаціях. Його перевага полягає у простоті використання та можливості взаємодії з іншими сервісами компанії Google, що дозволяє автоматизувати збір і обробку даних. Наприклад можна зберігати результати у вигляді таблиці, надсилати повідомлення про відповіді на електронну пошту gmail, а також для автоматизації є можливість використовувати Google Apps Script.

					ІАЛЦ.467200.003 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

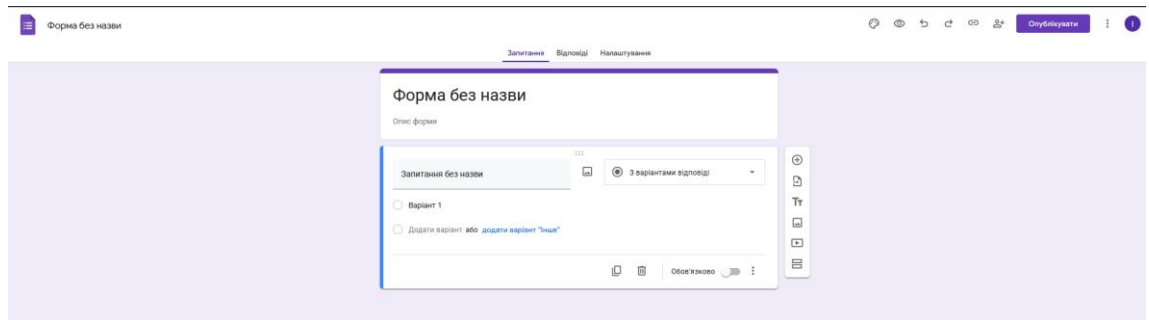


Рисунок 1.1 - Інтерфейс сторінки створення опитування Google Forms

Однак, попри простоту та зручність роботи сервісу, google forms має деякі обмеження та недоліки, які пов'язані з захистом даних та конфіденційності:

- Обмежена анонімність
- Залежність від екосистеми Google
- Нежостатній рівень захист

Це проявляється в тому, що навіть при включеному режимі “анонімного опитування” сервіс все одно фіксує метадані, такі як ір-адресу користувача, дані про пристрій(тип браузера, ОС), час проходження опитування.

Також враховуючи те, що всі дані зберігаються на серверах Google це створює потенційні ризики, а саме можливість витоку інформації через зовнішні атаки.

Через те, що сервіс не орієнтований на створення опитувань, котрі будуть мати важливі дані, або використовуватися для досліджень, Google forms не має інструментів для шифрування відповідей, а також не надає інструментів для повного видалення метаданих та захисту від Ddos-атак або масового надсилання спам-відповідей.[1]

Microsoft Forms

Ще один сервіс однієї з найбільших корпорацій, що представляє собою сучасне хмарне рішення для створення та проведення опитувань, яке є частиною екосистеми Microsoft 365.

					ІАЛЦ.467200.003 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

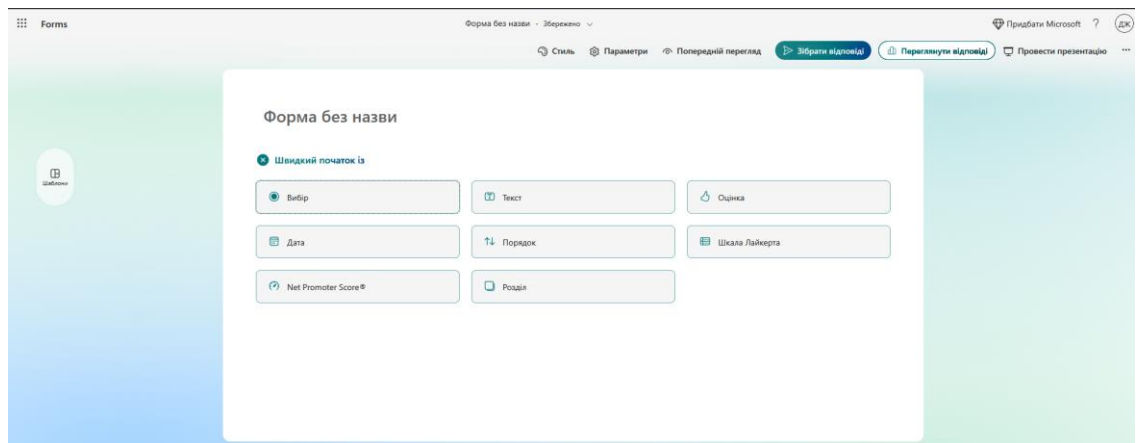


Рисунок 1.2 - Інтерфейс сторінки створення опитування Microsoft Forms

Якраз таки завдяки інтеграції з іншими продуктами компанії Microsoft, зокрема з Teams, Excel, SharePoint та інші, сервіс отримав свою аудиторію серед освітніх установ, або корпоративних клієнтів.

З переваг можна віднести відносно простий та зрозумілий інтерфейс, він незначно важчий, порівняно з Google Forms, але це обумовлено більшою кількістю функціоналу, та можливостей, котрий може використати користувач.

В формах наявна достатня кількість типів відповідей, котрі покривають та забезпечують користувача необхідним для створення повноцінного опитування. Як вже було сказано, важливою частиною застосунку є можливість взаємодіяти з іншими продуктами компанії, наприклад, користувачі можуть автоматично експортувати результати опитувань у формат Excel, що значно спрощує подальший аналіз отриманих даних. Для освітнього середовища платформа пропонує спеціалізовані функції, такі як створення тестів з автоматичною перевіркою відповідей та системою оцінювання, що робить її дуже корисною в навчальних закладах.[2]

До обмежень та недоліків такої платформи можна віднести:

- Недостатня кількість типів питань
- Обмежена логіка гілкування

Проблема в тому, що навіть незважаючи на те, що сервіс надає можливість використовувати різні типи відповідей, які можуть покривати потреби

звичайного користувача, однак відсутність типів, таких як матриці, табличні питання, вкладені списки обмежує користувача для більш складного та комплексного дослідження.

Microsoft Forms підтримує можливість змінювати питання в залежності від попередніх відповідей, проте порівняно з іншими сервісами, наприклад з Typeform, що буде описаний нижче, сервіс не дає змоги створювати вкладені гілки та додавати багаторівневі сценарії, а також унеможлиблює зміну балів та змінних питань.

Typeform

Є дуже багатофункціональним інструментом для проведення онлайн-опитувань, який значно відрізняється від традиційних рішень своїм інтерфейсом, який показує по одному питання на екран, даючи можливість більш персоналізовано створити кожне питання.

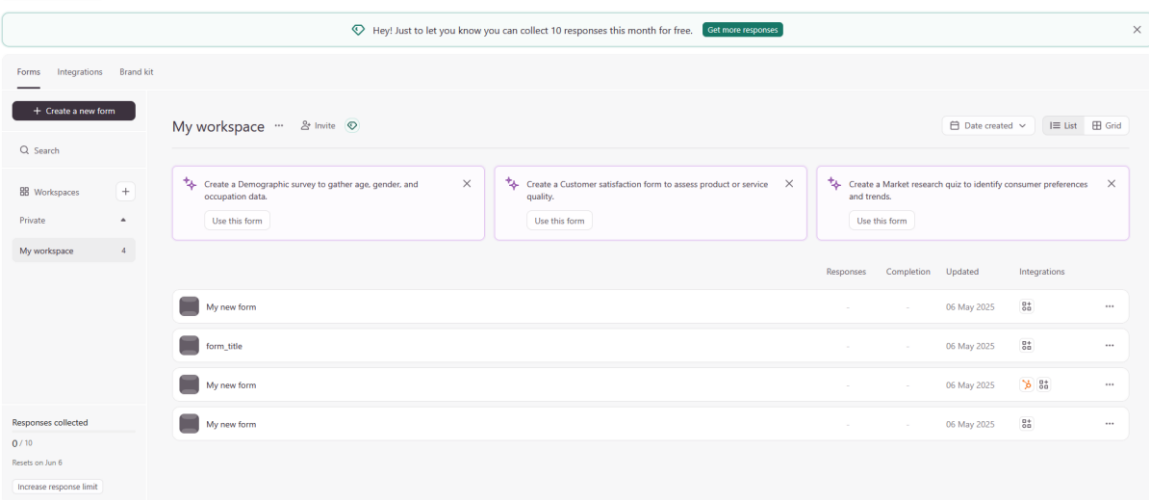


Рисунок 1.3 - Інтерфейс сервісу Typeform

Також сервіс надає можливість переходити від питання до питання залежно від відповідей користувача, що відкриває нові можливості для проведення опитувань.

Наявність великої кількості готових шаблонів на різні теми може доволі сильно скоротити час на створення опитувань вручну, а наявність можливості додавати мультимедійний матеріал, котрий буде відтворюватися прямо в

сервісі, на відміну від, наприклад google forms, покращує досвід роботи користувача. Що також важливо відмітити, це те, що платформа не стоїть на місці, а слідує та адаптується за трендами сучасного вебу, було інтегровано штучний інтелект, завдяки якому можна створювати питання для опитування.

До недоліків платформи можна віднести такі пункти:

- Обмеження безкоштовної версії
- Питання безпеки
- Анонімність і збирання метаданих

Для залучення великої кількості аудиторії часом дуже важливо, щоб сервіс надавав достатній безкоштовний функціонал, проте нажаль більша частина інструментів цього сервісу є платною, і обмеження для безкоштовної підписки доволі суттєві. У безкоштовному тарифі сервіс обмежує загальну кількість відповідей, яку можна зібрати за місяць на всі форми, це значення складає лише 10, що й неймовірно малою кількістю, для складання комплексної статистики. Але кількість відповідей не єдине обмеження, також безкоштовна версія дозволяє мати тільки декілька активних форм, на які користувач може давати свої відповіді, а також недоступна функція, котра дозволяє динамічно змінювати структуру опитування в залежності від відповідей користувача.

Щодо безпеки платформа не використовує капчу під час реєстрації, що робить її уразливою до заповнення форм ботами, також невідомо чи використовується шифрування під час збереження даних в таблицю.

Проблема анонімності полягає в тому, що лише одному з платних тарифів є можливість використати опцію анонімізації, але для інших користувачів сервіс буде фіксувати ір-адресу. А також буде зберігати цифрові сліди у вигляді cookie-файлів та local storage.[3]

SurveyMonkey

Ця платформа вважається одною з найстаріших та найвідоміших для створення онлайн-опитувань. Сервіс є дуже популярним для використання у

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		13

корпоративному сектору, маркетингових дослідженнях, а також в освіті та соціології. Широкий спектр функцій. Підтримка логічних переходів, зручний конструктор форм, наявність можливості використання штучного інтелекту, все це обумовлює його популярність.

Також платформа підтримує інтеграцію з великою кількістю інших сервісів, має автоматичну аналітику, в яку входить побудова графіків, діаграм, або ж просто текстових звітів.

Платформа має як безкоштовну версію, так і платний варіант з більш розширеним функціоналом. Використовуючи платну версію можна зв'язатися з підтримкою сервісу та також отримати повний доступ до Restful API, завдяки котрому можна взаємодіяти з опитуваннями.

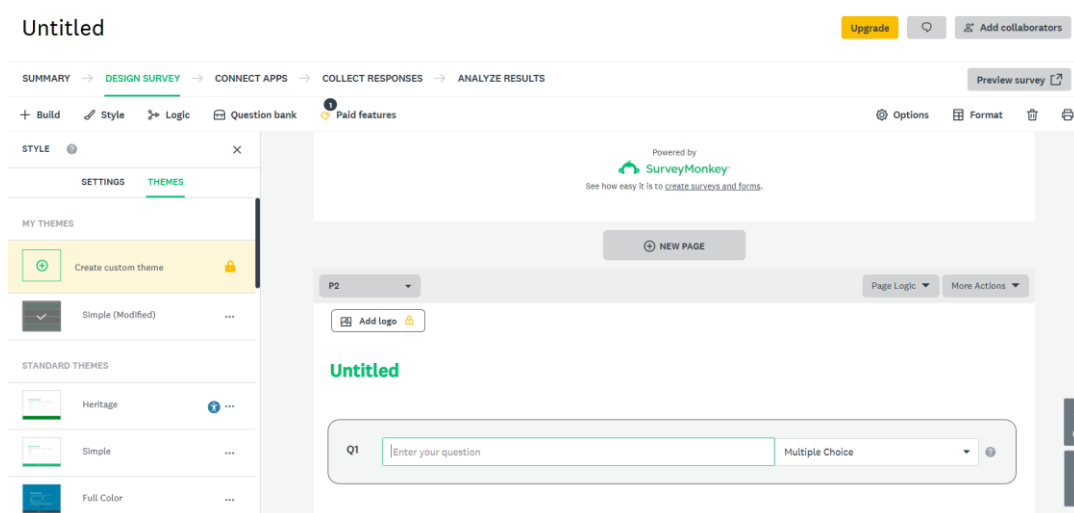


Рисунок 1.4 - Інтерфейс сервісу SurveyMonkey

До недоліків можна віднести такі пункти:

- Обмеження безкоштовної версії
- Платний API

Як і у випадку з сервісом TypeForm ця платформа має обмежений функціонал в безкоштовній версії. Кількість відповідей обмежено до 40 на анкету, закритий повний доступ до аналітики, та заборонено використовувати деякі типи питань.

					ІАЛЦ.467200.003 ПЗ	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

Хоч це і не суттєвий недолік для такої системи, проте через те, що без підписки на дорогі тарифи неможливо використовувати API для інтеграції з іншими системами, виникає обмеження автоматизації збору та обробки даних, що може вплинути на якість аналізу у масштабних дослідженнях або.[4]

LimeSurvey

Це open-source платформа для створення опитувань, яка орієнтована на установи, що цінують повний контроль над даними. Вона дає можливість, як у випадку з іншими корпоративними сервісами створювати складні опитування з гнучкою логікою. Завдяки відкритому коду сервіс можуть використовувати в державному секторі, університетах і серед it-компаній, як потребують прозорості, адаптивності, а також контролю над захищеності даних.

Відкритий код сервісу доступний на GitHub, завдяки чому можна маніпулювати над сервісом, впроваджувати нові модулі або інший необхідний функціонал, та розгорнути на власному сервері, що дозволяє не переживати за конфіденційні дані користувачів, бо вони будуть під вашим контролем.

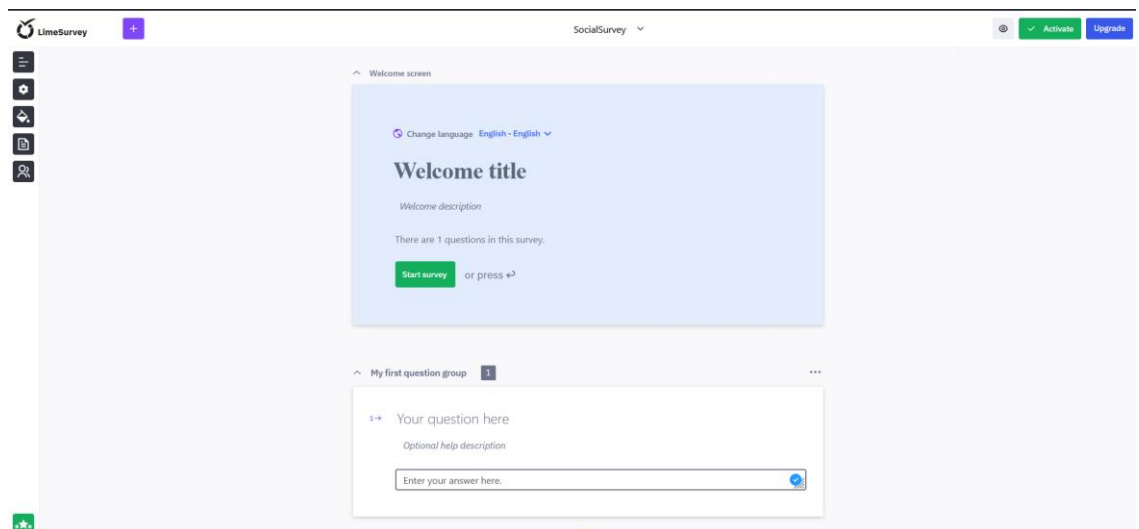


Рисунок 1.5 - Інтерфейс сторінки створення опитування LimeSurvey

Порівняно з корпоративними сервісами опитування, LimeSurvey завдяки можливості самостійно розгорнути на своєму сервері не викликає переживань, щодо безпеки даних, та до недоліків можна віднести лише незначно складніший

інтерфейс порівняно з іншими платформами, а також для самостійного розгортання недоліком є складність зі встановленням та налаштуванням.[5]

За результатами огляду та аналізу існуючих готових рішень для створення та проведення опитувань систематизуємо ці дані у вигляді таблиці 1.1, де будуть представлені переваги та недоліки оглянутих платформ.

Ці дані будуть важливими для формування фінального розуміння, сервіс з якими характеристиками та перевагами має вийти, а також яких недоліків вже існуючих платформи можна позбутися в процесі розробки власної програмного забезпечення.

Таблиця 1.1 – Характеристика оглянутих платформ

Критерій	Google Forms	Microsoft Forms	TypeForm	SurveyMonkey	LimeSurvey
Безкоштовна версія	Так	Так	Так (обмежена)	Так (обмежена)	Так
API	Ні	Ні	Так	Так	Так
Анонімність	Низький рівень	Низький рівень	Середній рівень	Середній рівень	Високий рівень
Захист від ботів	Ні	Частково	Ні	Частково	Так
Самохостинг	Ні	Ні	Ні	Ні	Так
Гнучкість логіки	Низький рівень	Низький рівень	Середній рівень	Високий рівень	Високий рівень
Кастомізація дизайну	Низький рівень	Низький рівень	Високий рівень	Середній рівень	Середній рівень

ВИСНОВОК ДО РОЗДІЛУ 1

В даному розділі ми зробили комплексний огляд еволюції інструментів соціологічного опитування, який бере свій початок від паперових анкет до сучасних веб-застосунків. Прагнення до масштабованості, автоматизації та зменшення витрат на проведення опитування обумовлює перехід від застарілих засобів до сучасних цифрових форматів проведення опитувань. Однак така цифровізація несла з собою, як позитивні моменти, які вже були описані, так і відкрила нові виклики: уразливості в сфері захисту персональних даних, ризики втрати анонімності та зростання впливу сторонніх сервісів на збереження та обробку інформації.

Також в розділі особливу увагу аналізу вже існуючих рішень. В результаті аналізу таких п'яти сервісів, як Google Forms, Microsoft Forms, Typeform, SurveyMonkey та LimeSurvey було виявлено, що попри сучасний дизайн та широкий спектр можливостей більшість комерційних платформ мають недоліки, котрі часто пов'язані з захистом конфіденційної інформації, а також не завжди відповідають очікуванням щодо гнучкості налаштувань та масштабованість, як особливо актуальні для користувачів без платної підписки. У свою чергу open-source рішення LimeSurvey дає можливість самохостингу, що забезпечує високий рівень контролю за даними, а також дозволяє користувачу адаптувати систему під індивідуальні потреби, хоча це і може потребувати високих технічних знань, що мають бути неактуальні для великої частини аудиторії.

В результаті роботи над першим розділом з'явилось розуміння того, які характеристики необхідно мати майбутньому застосунку. Він має поєднати переваги сучасних платформ – зручність, масштабованість і гнучкість, а також вирішити проблеми корпоративних рішень – забезпечити захист даних користувачів та відкриту архітектуру.

					ІАЛЦ.467200.003 ПЗ	Арк.
						17
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2

ОГЛЯД ТА ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ

На даному етапі необхідно оглянути та проаналізувати основні підходи до розробки, які можна буде застосувати при проектування та розробці веб-застосунку. Успіх та коректна робота сервісу згідно до вимог, встановленими в першому розділі, також залежить не тільки від правильної розробки застосунку, але також від вибору технологій, які допоможуть раціонально, не витрачаючи час на зайву конфігурацію, реалізувати необхідний функціонал, забезпечуючи при цьому високий рівень якості розробленого додатку.

2.1 Архітектурні підходи до розробки веб-сервесів

Архітектура застосунку є таким собі планом, в якому встановлено яким чином компоненти розробленої системи взаємодіють або між собою, або з користувачем. Визначається в якому порядку відбувається передача даних, та коли саме вони мають відправлятися та отримувати зворотній результат

Це безпосередньо дуже важливо під час проектування системи, адже саме правильно обрана архітектура є одним з стовпів правильно функціонуючого додатку. Обрана архітектура напряду впливає на можливості до масштабованості, безпеки, гнучкості, тестованості та витрат на підтримку продукту. Звичайно що, враховуючи, що існують багато підходів до розробки, можна зробити висновок, що всі вони мають свої плюси та мінуси та для кожного проекту необхідно обрати свою архітектуру, яка забезпечить розробників раціонально підтримувати та розширювати функціонал додатку, не руйнуючи при цьому досвід роботи користувачів з платформою.

В той же час у разі невдалої обраної архітектури, підтримка проекту та додавання нових фіч може стати справжньою проблемою. Тому необхідність

					ІАЛЦ.467200.003 ПЗ	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

огляду та вибору архітектури є важливим процесом на рівні важливості вибору стеку технологій для програмування додатку.

Найпоширенішими підходами до побудови веб-застосунків на вважається монолітна архітектура, мікросервісна архітектура та Serverless-модель. Саме ці три види архітектури будуть оглянуті, їх переваги, недоліки та модель функціонування систем з подібними моделями взаємодії компонентів

2.1.1 Монолітна архітектура

Монолітна архітектура отримала свою назву на основі великого геологічно утворення, простіше кажучи великого каменя, що може мати на увазі, великий об'єкт. Так от системи на основі монолітної архітектури представляють собою, можна сказати такий собі великий цілісним “камінь”, котрий суміщає в собі всі компоненти застосунку. До цього відноситься і сервіси доступу до бази даних, бізнес логіка, в якій відбувається обробка даних отриманих з бази даних та передача даних до користувацького інтерфейсу, котрий так само є компонентом в цьому моноліті.[6]

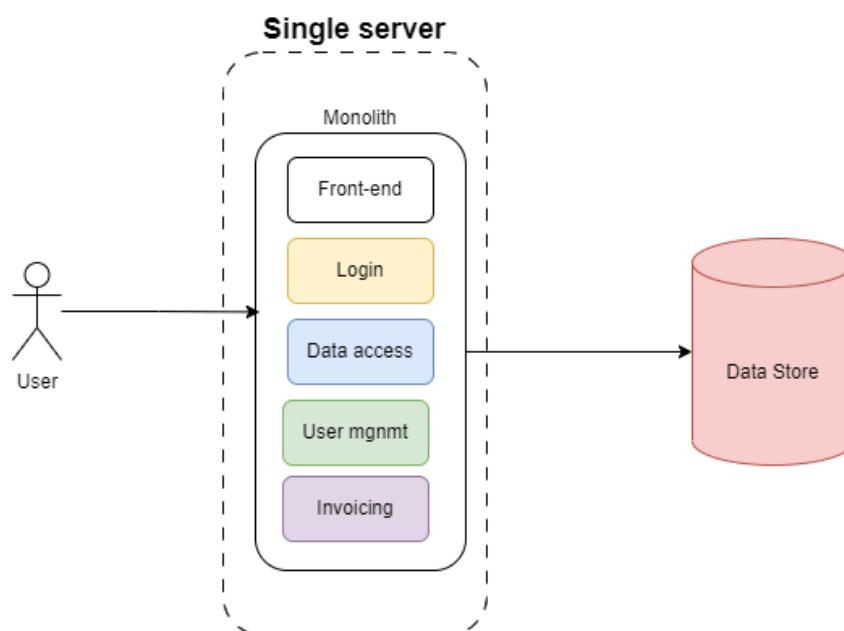


Рисунок 2.1 - Принцип мікросервісної архітектури

Розглянемо плюси використання такої архітектури.

Завдяки тому, що в монолітній архітектурі для всіх компонентів системи використовуються один і той самий стек технологій, це звичайно що спрощую розробку такої системи.

Системи побудовані на основі такої архітектури працюють з одним виконуємим файлом або каталогом, котрі порівняно з іншими архітектурами значно спрощують процес розгортання застосунку на сервері.

Також, якщо моноліт, ще не має дуже великих розмірів, то є шанс того, що тестування такого сервісу може відбуватися швидше порівняно з розподіленими сервісами. Розмір моноліту буде впливати не тільки на тестування, а й на можливість розширення функціоналу, та виправлення багів під час розробки та роботи системи, якщо моноліт залишається невеликих розмірів, враховуючи те, що весь код сконцентрований в одному місці, і всі компоненти взаємодіють між собою всередині додатку, проводити виправлення багів може бути легше.

Щодо мінусів таких систем хотілося на початку сказати, що раніше, тільки на початку цифровізації та розвитку вебу саме ця архітектура була основною, і перевага, котрі були зазначені вище справді були дуже актуальні, але за час розвитку вебу розмір таких сервісів збільшувався, і важкість підтримувати такі системи все зростає та зростає, відповідно до цього, проаналізуємо недоліки такої архітектури.

І почнемо з того, що поки моноліт невеликий то розробка може йти доволі вдало, проте зі зростанням проекту, якщо це ще й корпоративна система, котра складається з сотень компонентів, що взаємодіють між собою і реалізують велику кількість функціоналу, підтримка та розширення таких проектів стає складною працею.

Відповідно до попереднього недоліку, виправлення багів в системі також залежить від розмірів застосунки, бо під час розширення

					ІАЛЦ.467200.003 ПЗ	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

функціоналу застосунку часто відбувається так, що при додаванні нової функції, з'являється проблема у взаємодії з існуючим функціоналом, і поки буде вирішуватися цей баг, може з'явитися баг, через те, що всі компоненти такої системи пов'язані між собою, і дуже важко щось розробляти, нічого при цьому не ламаючи.

Зрозуміло, що, через ці проблеми з взаємодією компонентів між собою та появі багів під час розробки, час на розробку нового функціоналу або навіть видалення застарілого може займати велику кількість часу.

Ще одним недоліком таких систем є те, що кожен раз під час додавання нового функціоналу, необхідно робити деплой не тільки окремого компонента, в якому були зроблені зміни, а всього застосунку, що буде займати велику кількість часу.

Враховуючи розміри системи та скільки ресурсів було витрачено компанією на її розробку, зрозуміло, що таку систему, написану на монолітній архітектурі майже ніколи не переписують, та не змінюють стек технологій, через те, що це неймовірно дорого та невигідно для бізнесу, через це, зазвичай в таких системах використовуються застарілі технології, котрі використовували розробники десятки років тому.[7]

2.1.2 Мікросервісна архітектура

На відміну від монолітних систем, платформи зроблені на мікросервісній архітектурі мають розподілені сервіси, кожен з яких працює окремо від інших. Такі сервіси можуть мати різні стеки технологій, використовувати різні підходи до програмування та все одно мати можливість працювати один з одним передаючи дані один до одного для обробки у форматі JSON.

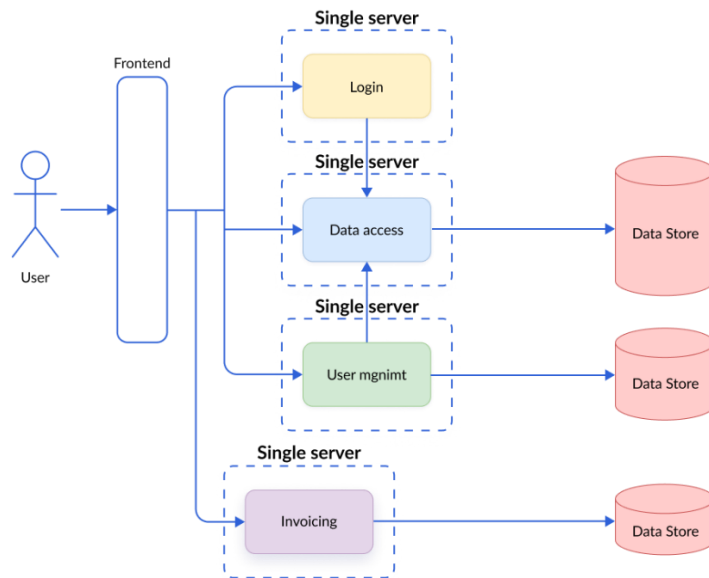


Рисунок 2.2 - Принцип мікросервісної архітектури

Розглянемо переваги таких систем.

Можливість збільшення розмірів та додавання нового функціоналу не потребує таких зусиль порівняно наприклад з монолітним застосунком, враховуючи що сервіси не дуже великі між собою, та кількість компонентів, що взаємодіють між собою не така велика як в монолітах, шанс натрапити на ланцюг багів значно менший.

У випадку з мікросервісною архітектурою з кожним сервісом може працювати невелика команда, завдяки такій гнучкості, робота над застосунком спрощується, особливо у випадку звільнення або найма нового розробника в команду, в такому випадку людині набагато легше розібратися в функціонування окремого маленького мікросервісу, та почати розробляти нові функції.

Також враховуючи те, що кожен сервіс не залежить від інших, окрім даних, що до цього поступають, це дає можливість обирати різні стеки технологій, котрі будуть підходити або команді, або бізнесу через якісь особливості окремих інструментів.

Що стосується розгортання застосунку, можливо те, що перше розгортання всіх сервісів займе час, але під час додавання нового функціоналу

не потрібно робити розгортання всієї системи, а лише тих сервісів в яких відбулися зміни. А також після розгортання, навіть якщо відбудуться якісь помилки на окремому сервісі, або не буде працювати сервер на якому знаходиться сервіс, це не перерве роботу усієї системи, і не потрібно буде перезапускати увесь застосунок, в цей час лише деякі її функції будуть недоступні, і після виправлення необхідно буде запустити лише відключений компонент, для відновлення повного функціоналу застосунку.[8]

Незважаючи на велику кількість позитивних моментів в розподілених сервісах, все одно існують проблеми котрі можуть бути створені такою організацією взаємодії компонентів.

Під час збільшення таких систем може виникати проблема узгодженості даних, котрі циркулюють між сервісами, з додаванням нового функціоналу, може з'явитися потреба відправляти нові дані до іншого сервісу, котрий треба налаштувати на обробку цих даних.

Через передачі даних між серверами з використанням мережі, в якій можуть відбуватися затримки, а не всередині одного сервера, може виникати зниження продуктивності порівняно з монолітом.

Час та складність проведення тестування може бути доволі довгим та важким процесом, через те, що потрібно тестувати функції сервісу спочатку в ізоляції від інших компонентів, а потім ще перевіряти, чи правильно відбувається взаємодія з іншими сервісами.[9]

2.1.3 Server-less архітектура

Попри свою назву, додаток розроблений на такій архітектурі все одно має сервер. Мається на увазі про модель обчислень, за якою розробники не керують власними сервера безпосередньо, а покладаються на хмарну інфраструктуру, котра і буде підтримувати сервер, розгортати його й обслуговувати функції, котрі як раз таки і має написати розробник, це будуть фрагменти коду котрі

					ІАЛЦ.467200.003 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

будуть виконуватися під час виклику, котрий у веб-сервісі може бути наприклад результатом на HTTP запит. До платформ що реалізують функціонал подібного шаблону, а саме Function as a Service(FaaS) відносяться AWS Lambda, Google Cloud Functions, Azure Functions

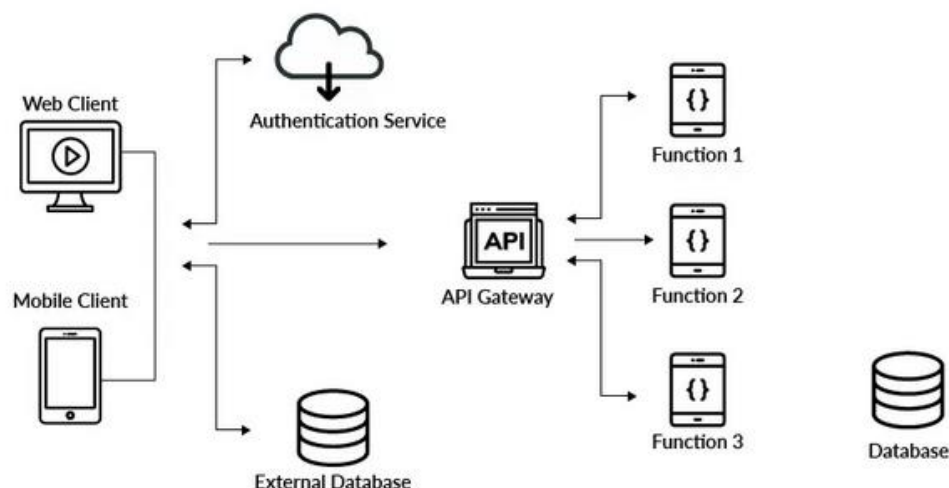


Рисунок 2.3 - Принцип Serverless архітектури

Що стосується плюсів використання такої архітектури, то це те, що системи побудовані на Serverless архітектурі можуть бути дуже привабливими в плані оплати, тому що хмарні сервіси беруть гроші не за час оренди сервера, а за фактичне використання обчислювальної потужності, тобто розробник платить лише за час, коли сервіс був використаний на відміну від розгортання додатків інших архітектур на звичайних серверах.

Ще однією перевагою таких систем є швидкість розгортання застосунку, розробнику достатньо лише написати необхідну функцію та завантажити її у відповідну хмарну платформу

Порівняно з іншими архітектурами легка конфігурація інфраструктури, балансування навантаження та масштабування, котрі автоматично виконуються на стороні платформи на якій і розробляється сервіс

До головних недоліків систем такої архітектури, виступає затримка виклику, важкість повноцінного тестування та залежність від провайдеру.

Через те, що для економії ресурсів, коли не має виклику функції, платформа зупиняє її дію, то для першого виклику користувачем буде потребуватися якийсь час на те, щоб платформа виконала цю функцію.

Реалізувати повноцінне інтеграційне тестування, що проаналізувати як саме взаємодіють backend та frontend частини у таких системах важко, хоча розробник все ще може проводити окремі тестування написаних функцій.

Як вже було сказано недоліком таким систем є залежність від місця розробки, тобто після розробки сервісу під якогось конкретного провайдера, мігрувати на іншу платформу без переписування логіки неможливо. Під час розробки serverless застосунків використовуються специфічні бібліотеки, котрі взаємодіють з хмарним провайдером та підходять виключно для нього, тобто зміна платформи вимагатиме зміни коду функцій. [10]

2.2 Огляд сучасних фреймворків для розробки

До якогось періоду часу для розробки як серверної так і клієнтської частини застосунку розробники використовували мови програмування включно з бібліотеками, та конфігурували все вручну, але з часом стало зрозуміло, що такі налаштування займають велику кількість часу, хоча в своїй основі кожного разу представляють собою комбінацію майже одних і тих самих налаштувань. Саме таким чином були створені фреймворки, котрі спрощують розробку застосунку завдяки вбудованих інструментів, котрі роблять час, який витрачається на конфігурацію, мінімальним.

Вибір фреймворку є важливим аспектом для розробки застосунку, саме він буде визначати, які інструменти будуть використані. Кожен фреймворк має

					ІАЛЦ.467200.003 ПЗ	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

свої плюси та мінуси, тому необхідно проаналізувати кожну популярну технологію розробки, котра може бути використана для розробки веб додатку.

2.2.1 Back-end фреймворки

Фреймворки для розробки серверної частини застосунку надають розробнику зручні інструменти для обробки ННТР запитів, надають можливість без зайвої конфігурації працювати з базами даних, реалізовувати логіку авторизації, автентифікації та маршрутизації, а також керувати сесіями та безпекою. То ж у цьому підрозділі будуть оглянуті відомі фреймворки, від яких напряду буде залежати продуктивність розробки.

Spring framework

Побудований на мові програмування Java цей фреймворк зарекомендував себе, як дуже вдалий варіант для розробки веб застосунків. З застосуванням принципу ООП, Spring надає зручні інструменти для розробки застосунків за цим принципом, завдяки використанню принципу інверсії управління та ін'єкцію залежностей, за яким фреймворк контролює життєвий цикл бінів, які є звичайними Java об'єктами, контрольованими Spring.

Фреймворк має в своєму арсеналі велику кількість бібліотек для зручної роботи з веб, доступу до бази даних, інструменти для автентифікації, авторизації та тестування і навіть інтеграції з штучним інтелектом.[11]

До ключових переваг цього фреймворки віднести: швидкість розробки, завдяки використанню бібліотек Spring, час який необхідно використати на конфігурацію зводиться до мінімуму, якщо звичайно що, не враховувати кастомну кастомізацію.

Spring вважається одним із самих надійних інструментів для захисту даних. Саме через це, цей фреймворк дуже часто використовується великими корпораціями, для яких критично важливий захист даних користувачів та

					ІАЛЦ.467200.003 ПЗ	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

безпека від зламу, зазвичай це банкові системи, Е-commerce, а також державні портали.

Попри те, що фреймворк надає інструменти для більшої частини проблем, які потрібно вирішити розробнику, це не забороняє використання сторонніх бібліотек, котрі також добре інтегруються в Spring застосунки. Це ж стосується і гнучкості конфігурації, розробник може обрати варіант конфігурації який може бути як в xml файлі, так і Java кодом з використанням Spring анотацій.

Завдяки використанню ін'єкції залежності та використання бібліотеки тестування, процес тестування систем написаних з використанням цього фреймворку стає набагато легше.

До недоліків фреймворку можна віднести контроль під час розробки, через автоматичний контроль бінів фреймворком, доволі важко слідкувати яким саме чином, та в якому порядку вони інтегруються в інші класи, через це іноді можуть виникати баги, які можуть потребувати у великих застосунках багато часу на вирішення.

Враховуючи кількість інструментів та особливість розробки систем з використанням Spring, які мають враховувати використання ООП, розуміння роботи з бінами та контейнером бінів, а також вміння контролювати коректну ін'єкцію об'єктів, на вивчення цього необхідно витратити багато часу.[12]

Django

На сьогодні відсоток застосунків, які використовують мову програмування Python та бібліотек написаних на цій мові дуже високий. Цю мову використовують буквально в кожній сфері, де можна застосувати інформаційні технології для автоматизації, або ще для якихось розрахунків. Тому не дивно, що фреймворк написаний на цій мові для розробки веб серверів стане гарним рішенням, яке підійде великій кількості розробників.

У фреймворк інтегровано багато інструментів, які в інших фреймворках необхідно було підключати, використовуючи додаткові бібліотеки. До таких інструментів відноситься ORM, що дозволяє будувати таблиці бази даних не

					ІАЛЦ.467200.003 ПЗ	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

використовуючи SQL, а шляхом прив'язки полів класа до стовбців таблиці, генерувати таблиці та взаємодіяти з ними за допомогою коду на Python. Також Django має інструменти роботи з автентифікацією та авторизацією, маршрутизацією, а також має вбудований шаблонізатор, котрий дозволяє створювати динамічні HTML-сторінки з частинками Python коду.

Головною перевагою Django є звичайно що спільнота, кожен день розробники працюють над покращенням фреймворку, а також завдяки популярності python постійно створюється велика кількість бібліотек, які можна використовувати для побудови додатку з багатим функціоналом.[13]

Попри велику кількість переваг, фреймворк, як і інші, має свої недоліки, Django має великий набір вбудованих функцій, та компонентів які не потрібні для побудови малих сервісів, також для опанування необхідної для побудови системи більшого рівня необхідно витратити значну кількість часу та зусиль.

А також, як відомо мова python має доволі низьку обчислювальну швидкість, відповідно до цього і фреймворк написана на ній та система, що використовує Django може працювати доволі повільно, погіршуючи досвід користувача.

Express.js

Фреймворк є каркасом для розробки серверної розробки веб та мобільних застосунків для Node.js. Порівняно з громіздкими фреймворками котрі були розглянуті до цього, а саме Spring та Django, цей фреймворк набагато легший і не має такої кількості вбудованих функцій. Завдяки цій легкості фреймворк часто використовують для розробки маленьких додатків, котрі не потребують зайвої конфігурації, завдяки мінімалізму фреймворку час необхідний для розробки простого API нижчий ніж у великих фреймворках, достатньо налаштувати сервер, маршрутизацію та зробити обробку HTTP запитів.

Тобто головною особливістю Express.js є його гнучкість, відсутність жорсткої структури та мала кількість вбудованих інструментів, фреймворк дозволяє розробнику самому обрати в якому стилі організовувати проект та

					ІАЛЦ.467200.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

обрати ті інструменти, котрі розробники вважають кращими для вирішення проблеми.

Проте, саме через таку особливість, фреймворк має значні недоліки.

Саме через легкість та відсутність необхідної структури такий фреймворк не підходить для проектів середнього та великого об'єму, через хаотичність, підтримка та розвиток проектів буде займати багато часу, котрий буде витрачатися на розуміння, що відбувається в проекті.

Порівняно з іншими великим інструментами, що були розглянуті до цього, фреймворк не має такої великої кількості функціоналу, Express.js надає базовий інструментарій для розробки арі або односторінкових сайтів, проте для створення повноцінної серверної частини, наприклад для ORM, налаштування автентифікації, валідації введених даних і тд., необхідно використання сторонніх бібліотек.[14]

ASP.NET

Фреймворк, написаний на мові C#, та відповідно до цього належить компанії Microsoft.ASP.Net є дуже масивним інструментом для розробки систем будь якого типу, не важливо чи це веб додаток, десктопний застосунок або мобільний додаток, а також може використовуватися, не тільки для розробки Windows та Android застосунків, як це вже було сказано, а також для IOS, хоч це і не є раціональним варіантом.

Завдяки тому, що фреймворк реалізован на мові C#, котра є доволі швидкою порівняно з іншими мовами, особливо динамічної типізації, застосунки розроблені з використанням .Net можуть похизуватися високою продуктивністю, що може бути дуже важливою в деяких системах.

Взагалі враховуючи інструментарій котрий надає цей фреймворк, його можна порівняти з Spring framework, навіть якщо подивитися на сфери в яких застосовується сервіси, розроблені на рішенні компанії Microsoft та Spring, вони будуть приблизно однаковими, це корпоративні системи, інтернет магазини,

					ІАЛЦ.467200.003 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

державні установи. Тобто зрозуміло, що сервіси надають схожі інструменти та виконують однакові проблеми.

Проте все одно більшість компаній обирає саме Spring для розробки своїх систем. Це пов'язано з тим, що як системи написані на Spring так і на Aps.Net зазвичай це великі системи, що можуть складатися з сотень компонентів, а переписувати вже існуючі системи на нову технологію дуже не вигідно з точки зору бізнесу, а Java стала стандартом enterprise розробки ще на початку цифровізації, і зайняли цю нішу, саме тому фреймворк поступається Spring у популярності, проте все ще надає зручні та багатofункціональні інструменти для створення веб застосунків, до цього відноситься інструменти вбудовані засоби безпеки для захисту від найбільш популярних видів атак(XSS,CSRF,SQL Injection), а також засоби автентифікації та авторизації, легка взаємодія з базами даних та підтримка ORM.

Проте, ASP.Net має і свою недоліки, котрі схожі з іншими рішенням, наприклад, як і у випадку з Spring, фреймворк надає велику кількість інструментів та їх можливості, призводять до того, що не повне розуміння їх принципу роботи, може призвести до некоректної роботи застосунку, або взагалі неможливість правильно використовувати їх для розробки певного функціоналу.

А також в порівнянні з Java та Python на яких написані фреймворки про які було описано вище, C# має в своєму арсеналі меншу кількість open-source бібліотек, які можуть дуже сильно спростити процес розробки та зменшити час на імплементацію нового функціоналу.

2.2.2 Front-end фреймворки

На відміну від серверної частини, для написання клієнтської частини переважно використовується одна мова програмування JavaScript, та її похідні, наприклад строго типізований варіант TypeScript. Використання фреймворків у

					ІАЛЦ.467200.003 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

фронтенді допомагає побудувати структуровану архітектуру, що допомагає скоротити час розробки.

Angular

Фреймворк був розроблений в 2010 році компанією Google, та з того часу постійно розвивається, враховуючи постійні зміни трендів у клієнтській частині застосунків. Для опису платформи можна використати порівняння з Spring, бо Angular по своєму опису це такий собі Spring зі світу фронтенду, він такий же громіздкий, має строгу архітектуру, з коробки має велику кількість функціоналу, на відміну від інших фреймворків написан на строго типізованому TypeScript, все це робить Angular дуже привабливим для використання в великих проектах, де критично важливо підтримувати структурованість та стиль проекту для коректної підтримки та розвитку. Саме тому цей фреймворк часто йде поруч з Spring або ASP.Net у великих корпоративних додатків.

В цілому Angular був побудований для створення односторінкових застосунків, де контент буде змінюватися не через змінення url, а відповідно до налаштованій маршрутизації динамічно змінюватися на той самій сторінці, а така технологія як двостороннє зв'язування даних, яке автоматично оновлює клієнтський інтерфейс відповідно до змін значення в моделі та навпака, сильно допомагає в реалізації SPA.[15]

Також варто сказати про інструмент Angular CLI, що по суті є командним рядком з додатковими функціями фреймворку. Завдяки цьому можна створити проект, згенерувати код або створити окремий компонент системи відповідно до задуманої архітектури.

До недоліків можна віднести те, що порівняно з іншими менш громіздкими фреймворками, в дуже великих системах написаних на Angular використання двосторонньої прив'язки може викликати затримки у відображенні інформації.

					ІАЛЦ.467200.003 ПЗ	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

А також, як вже було сказано на відміну від серверної частини, фронтенд не такий постійний, через постійний розвиток та появи нових можливостей у вебi, залишатися на старих версіях не завжди є гарним варіантом, так як це може знизити інтерес до застосунку з боку клієнта в бік більш інноваційного додатку, через це є бажання переходити на нові версії, і у випадку з Angular це може бути проблематично, враховуючи, що може з'явитися необхідність переписувати вже існуючий код, що є дуже важкою задачею у великих проектах.

Також до недоліку можна віднести те, що через великий розмір систем на Angular, час на перше відкриття сайту може зайняти більше часу для користувача, що не є привабливим для нього.[16]

Next.js

Розроблений компанією Vercel цей фреймворк реалізує інструменти бібліотеки React, котра використовується для створення інтерактивних користувацьких інтерфейсів. Перш ніж перейти до аналізу Next.js, треба оглянути бібліотеку, яка по суті є серцем цього фреймворку - React.

Розробники котрі займаються створенням систем з використанням React використовують компонентний підхід, котрий дозволяє розбивати користувацький інтерфейс на компоненти, які подібні до функцій або класів в мовах програмування для серверної частини, завдяки цьому кількість коду значно зменшується, забезпечуючи кращу читаність.

На відміну від Angular, React не має двосторонньої прив'язки даних, а використовує односпрямований потік даних.

Ще однією особливістю React є його кросплатформеність, використовуючи React Native можна розробляти не тільки користувацький інтерфейс для веб застосунків, або десктопних застосунків, а також для мобільної розробки.[17]

Next.js же у свою чергу використовує інструменти, які надає React та доповнює його для більш комплексного програмування та підтримці веб-застосунків.

					ІАЛЦ.467200.003 ПЗ	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

За замовчуванням React виконує клієнтський рендеринг, тобто весь JavaScript код завантажується в браузер і вже потім рендериться сторінка, під час такого варіанту рендеру зрозуміло, що перший запуск додатку, особливо на слабких пристроях буде займати якийсь час. Використання next.js вирішує цю проблему шляхом зміни клієнтського рендерингу на SSR, тобто сторінки будуть генеруватися на сервера та відправлятися до користувача вже готовими, або ще простіший варіант це SSG, або ж статична генерація, при якій сторінки створюються під час збірки проекту, це може бути дуже зручний варіантом для деяких випадків, наприклад для блогів або каталогів.[18]

Ще доволі цікавою особливістю цього фреймворку для розробки клієнтської частини є можливість створювати API Routes та підключатися до бази даних, тобто по суті для створення невеликих застосунків не обов'язково писати серверну частину окремо на іншій технології, використовуючи Next.js є можливість повністю побудувати як frontend так і backend.

До недоліків Next.js можна віднести вищу ціну для розгорки клієнтської частини, порівняно з іншими frontend фреймворками, бо через те що Next.js використовує SSR, то для розгортки необхідний окремий сервер, на відміну від інших фреймворків, котрі можуть використовувати статичний хостинг.

Vue.js

Розроблений у 2014 році цей фреймворк для розробки користувацьких інтерфейсів та односторінкових веб-додатків зарекомендував себе завдяки відносно до других платформ по типу Angular, легкості та гнучкості. Розробник Vue.js був членом команди в при розробці Angular, тому і Vue.js став продуктом котрий об'єднує найкращі сторони Angular і використовує функціонал бібліотеки React.

Порівняно з React або Angular цей фреймворк вважається щось на кшталт золотої середини. Він не такий громіздкий та важкий у вивченні як Angular, але і не такий малофункціональний з коробки як React. Коли як Angular орієнтований на корпоративні застосунки, і потребує глибоко розуміння роботи

					ІАЛЦ.467200.003 ПЗ	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

функціоналу, а React не має без сторонніх рішень багатьох важливих інструментів, Vue не є таким громіздким та не потребує жорсткої структури, але при цьому має вбудовані інструменти для покриття більшої частини проблем з якими може зустрітися розробник. Завдяки цієї легкості та гнучкості в архітектурі Vue стає гарним рішенням для розробки невеликих або середніх застосунків.

Також важливою особливістю роботи фреймворка є здатність миттєво відображати в інтерфейсі будь-які зміни у моделі без потреби вручну керувати DOM, як це необхідно робити при роботі з чистим JS.[19]

Враховуючи відсутність чіткою структури, зрозуміло, що недоліком фреймворку буде виникнення потенційних труднощів під час збільшення функціоналу системи, а також через відсутність чітких правил в якому стилі має виглядати структура не тільки самого проекту, але і коду, це може також ускладнювати підтримку коду, через різні підходи до розробки у людей або команд.

2.3 Аналіз систем керування базами даних

Дані в нашому світі є однією з найважливіших одиницею, і місце для їх зберігання, а саме бази даних є невід'ємною частиною майже всіх систем. Застосунки, незалежно від галузі, зберігають, обробляють і аналізують великі обсяги даних, і наявність такого інструменту як база даних, що забезпечує централізоване, надійне та ефективне зберігання цих даних, критично допомагає у функціонуванні додатків.

Вибір типу бази даних і системи управління доволі сильно впливає на систему, та на процес програмування, організації даних та доступ до них з backend частини. Саме тому в цьому підрозділі буде оглянуто тип бази даних та систему керування ними.

					ІАЛЦ.467200.003 ПЗ	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

2.3.1 Реляційні бази даних та системи керування ними

Реляційна база даних – це тип бази даних, в якій всі дані зберігаються у вигляді таблиць. Основна ідея – це структуровано зберігати дані, поділяючи великі кластери даних на багато таблиць та пов'язуючи їх між собою первинними ключами. Завдяки використанню такого підходу кількість однакових даних у таблиць зменшується, відповідно і час, який потребується на обробку запиту до таблиці також буде менше. [20]

Проте база даних не може існувати без системи керування базою даних. Такі інструменти забезпечують створення, керування, оновлення, захист та видалення реляційної бази даних. Найбільш популярними СКРБД на сьогодні є Oracle Database, PostgreSQL, MySQL.

Oracle Database

Комерційна система керування базами даних від компанії Oracle, що вважається однією з найпотужніших у світі. Завдяки наявності великої кількості інструментів для безпеки, аналітики, масштабування, а також високий рівень довіри до компанії Oracle, робить цю СКРБД важливою у великих корпоративних системах, банках та державних установах.

Не дивлячись на те, що Oracle Database є однією з найстаріших рішень, вона все ще залишається найбільш функціонально насиченою, пропонуючи не тільки велику кількість інструментарію, про який вже було сказано, а також надаючи високу продуктивність і надійність у критично важливих системах.

Через свою функціональність та в якомусь сенсі в деяких моментах застарілі порівняно з іншими СКБД варіанти реалізації якихось можливостей, Oracle Database є важкою для вивчення та використання. Разом з цим ліцензія продукту для використання малими проектами є не вигідна та недоречна, саме тому ця база даних не така популярна серед не комерційних розробників.

PostgreSQL

					ІАЛЦ.467200.003 ПЗ	Арк.
						35
Зм.	Арк.	№ докум.	Підпис	Дата		

Була розроблена в 1996, ця СКБД з відкритим кодом підтримує повний набір SQL-функцій, а також можливість створювати нові типи даних, агрегати, індекси тощо.

Завдяки своїй орієнтованості на надійність і точність PostgreSQL використовують у наукових проектах, або в аналітичних системах, де важлива складна логіка обробки даних. Проте СКРБД також може використовуватися і в комерційних застосунках, або в банках замість Oracle Database, та на відміну від продукту компанії Oracle, використовується в системах малих та середніх розмірів, завдяки можливості безкоштовного використання.

MySql

Розроблена в 1995 році та наразі є продуктом компанії Oracle. Вважається найпопулярнішим рішенням у веб-розробці невеликих і середніх проєктів із помірною складністю.

Завдяки простоті встановлення та порівняно з іншими СКБД легшу конфігурацію, робить MySql більш привабливою для людей, які не ставлять собі на меті глибоко розібратися в управлінні базою даних.

Мінусом цієї СКБД є те, що вважається, що вона є не такою ефективною з роботою з великими даними, та не підтримує весь можливий функціонал без платної комерційної версії.

2.3.2 Нереляційні бази даних та їх особливості

На відміну від реляційних баз даних нереляційні не використовують таблиці для зберігання даних, замість цього вони можуть зберігати дані в документах, де об'єкти також мають первинні ключі. Завдяки відсутності необхідності зберігати все в таблиці з жорсткою структурою, нереляційні бази даних можуть гнучко адаптуватися до змін у структурі даних, без потреби в зміні таблиць, що вже використовуються.

					ІАЛЦ.467200.003 ПЗ	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

Головною перевагою нереляційних баз є їх масштабованість. Завдяки спроможності розподіляти дані на кілька серверів і обробляти великі обсяги запитів одночасно з'являється можливість створювати високонавантажені системи, які будуть при цьому мати високі показники продуктивності.

До найбільш популярних NoSQL баз даних можна віднести MongoDB, яка завжди потрапляє в топ 5 найчастіше використовуваних СКБД серед всіх типів баз даних.

MongoDB є документоорієнтованою системою, тобто дані зберігаються у вигляді документів у форматі JSON. Такі документи будуть частиною колекції, котра чимось схожа на концепцію таблиці в реляційній базі даних. Для отримання інформації з документів використовується особлива мова запитів MongoDB, котра також представлена у форматі JSON документів

Представником іншого типу системи керування нереляційної бази даних є Redis, який є другим по популярності після MongoDB.[21]

На відміну від документоорієнтованого MongoDB, Redis – ключ-значення сховище, яке часто використовується в якості системи кешування або бази даних для обчислень у реальному часі, це тому що Redis працює в оперативній пам'яті, що не дає можливості збереження великої кількості даних, проте надає надзвичайно швидкий доступ до даних.[22]

В цілому можна сказати, що нереляційні бази даних є корисною альтернативою для реляційних бд, і можуть бути використані в специфічних випадках.

Щодо мінусів можна сказати, що такі бд не завжди забезпечують повну підтримку транзакцій та складних запитів, а також відсутня перевірка цілісності даних на рівні системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						37
Зм.	Арк.	№ докум.	Підпис	Дата		

2.4 Вибір технологічного стеку

В результаті аналізу існуючих технологій для розробки веб-застосунків, а також врахування особистих знань та досвіду використання оглянутих інструментів в якості фреймворку для розробки серверної частини було обрано фреймворк Spring з мовою програмування Java, котрий славиться своїми інструментами забезпечення безпеки даних, що є критично важливим в моєму застосунку. Для клієнтської частини був обраний фреймворк Angular, в цілому можна сказати, що для побудови невеликого сервісу, яким можна вважати і мій сервіс, цей фреймворк може бути не найкращім вибором, проте через гарну структурованість, яку надає фреймворку, в майбутньому я буду мати можливість розширювати можливості веб застосунку, не плутаючись в існуючому коді. В якості СКБД була обрана PostgreSQL, через те, що вона має всі інструменти необхідні для роботи з даними.

Також окрім використання основних інструментів для розробки застосунку, я буду використовувати такі ситеми, як Git, що є системою контролю версій, яка дозволить мені зберігати історію змін, та в разі виникнення помилок при розробці нового функціоналу, повертатися до старих версії, а також дозволить використовувати середовище для збереження коду GitHub для подальшого деплою на сервер. Для тестування розробленого API, буду використовувати інструмент Postman котрий дозволять симулювати запити від користувачів, та відслідковувати відповіді, які ми отримуємо від сервера.

					ІАЛЦ.467200.003 ПЗ	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 2

В другому розділі було зроблено аналіз основних технологій розробки backend та frontend частини застосунку, а також інструменти для збереження та керування даними проекту.

В результаті огляду існуючих інструментів було зроблено висновок, що кожна технологія має свої переваги та недоліки в залежності від того, в якому проекті їх використовувати.

Тому враховуючи вимоги до мого застосунку для серверної частини було обрано мову програмування Java та її фреймворк Spring, котрий надасть необхідні інструменти для забезпечення безпеки даних користувача. Для розробки клієнтської частини було обрано фреймворк на основі мови TypeScript Angular. Він вважається потужним інструментом, який має жорстку структуру, що дозволить створити проект з подальшою перспективою для його розширення. В якості СКБД було обрано PostgreSQL.

					ІАЛЦ.467200.003 ПЗ	Арк.
						39
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3

ДЕТАЛІ РОЗРОБКИ СЕРВІСУ

3.1 Розробка серверної частини застосунку

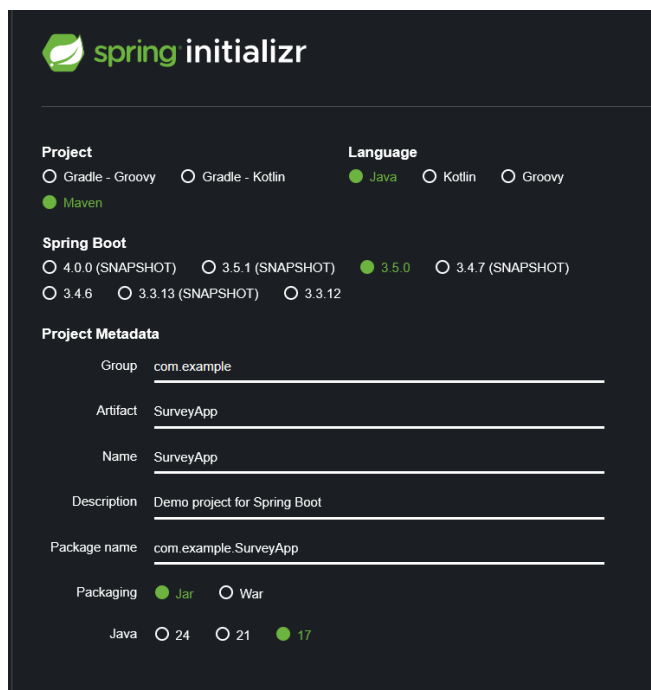
Для розробки серверної частини застосунку було обрано мову програмування Java та фреймворк котрий використовує цю мову – Spring. Головними частинами розробки серверної частини нашого застосунку є створення можливості автентифікації та авторизації для забезпечення закритого або відкритого доступу до інформації відповідно до потреб, також звичайно що, необхідно створити методи для взаємодії з підключеною базою даних, скбд якої в нашому проекті буде виступати PostgreSQL, та рівень бізнес логіки, де відповідно на запит від клієнтської частини застосунку будуть оброблюватися дані, що прийшли з клієнтської частини застосунку.

3.1.1 Початкова конфігурація

Перед тим як безпосередньою перейти к реалізації логіки серверної частини застосунку, треба ініціалізувати проект та підключити необхідні бібліотеки, необхідні для розробки.

Для ініціалізації Spring застосунку будемо використовувати ресурс Spring Initializr, в якому можна буде налаштувати проект, обравши версію Spring, мову програмування, назву, та інструмент зборки проекту.

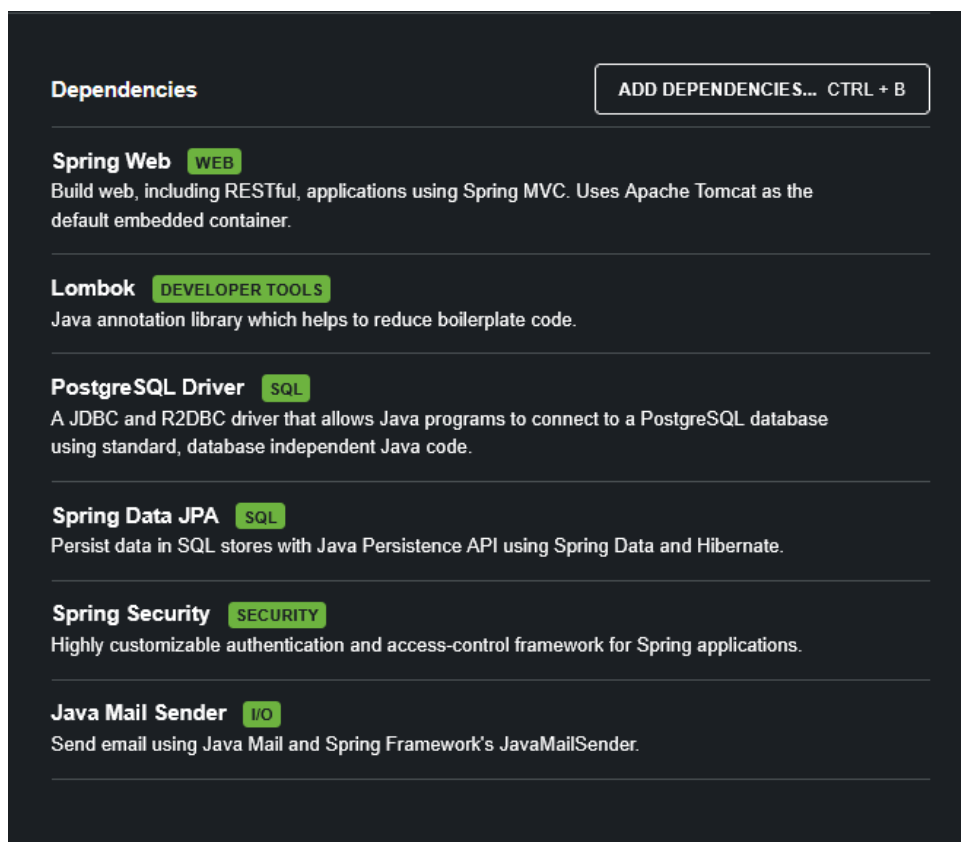
					ІАЛЦ.467200.003 ПЗ	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		



The image shows the Spring Initializr web interface for configuring a new project. It includes sections for Project (Gradle - Groovy, Gradle - Kotlin, Maven), Language (Java, Kotlin, Groovy), Spring Boot (4.0.0, 3.5.1, 3.5.0, 3.4.7, 3.4.6, 3.3.13, 3.3.12), Project Metadata (Group, Artifact, Name, Description, Package name), Packaging (Jar, War), and Java (24, 21, 17).

Рисунок 3.1 - Ініціалізація проекту

А також, що найбільш головне, на сервісі можна додати необхідні бібліотеки, які називаються залежностями.



The image shows the 'Dependencies' section of the Spring Initializr interface. It lists several dependencies with their categories and descriptions: Spring Web (WEB), Lombok (DEVELOPER TOOLS), PostgreSQL Driver (SQL), Spring Data JPA (SQL), Spring Security (SECURITY), and Java Mail Sender (I/O). A button 'ADD DEPENDENCIES... CTRL + B' is visible at the top right.

Рисунок 3.2 - Обрані залежності

В результаті до застосунку було додано 5 бібліотек:

Spring web – необхідний для створення REST api, саме завдяки цій залежності можна створити контролери, котрі будуть реагувати на запити з клієнтської частини застосунку, та викликати методи бізнес логіки для обробки даних та повернення на фронтенд нових даних.

Spring Data Jpa – бібліотека, що дозволяє високорівневно взаємодіяти з базою даних завдяки роботі Hibernate, котрий дозволяє використовувати ORM, що значно спрощує роботу з даними в таблицях.

Spring Security – залежність надає інструменти для розробки Security частини застосунку, це і реєстрація, автентифікація, авторизація.

Java Mail Sender – необхідна бібліотека для відправки повідомлення користувача на пошту, буде використовуватися для перевірки існування пошти та з її використанням активації акаунта для подальшої автентифікації.

Lombok – маленька бібліотека, котра зменшує кількість коду, це і сеттери і геттери, конструктори з параметрами, конструктори без параметрів та ін.

PostgreSql driver – інструмент необхідний для роботи Spring з PostgreSQL

Також ще однією бібліотекою, котру необхідно підключити є бібліотека для роботи з JWT, котрий буде в команді з Spring Security використовуватися для автентифікації та авторизації у додатку. Проте цієї бібліотеки немає на Spring Initializr, тому для імпорту цієї залежності перейдемо на MavenRepository, де можна знайти велику кількість існуючих бібліотек написаних на Java.

Окрім бібліотеки JWT, для автоматичного шифрування та дешифрування даних з бази даних буде використовуватися бібліотека jasypt, залежність якої, треба також знайти на MvnRepository.

					ІАЛЦ.467200.003 ПЗ	Арк.
						42
Зм.	Арк.	№ докум.	Підпис	Дата		

MVN

REPOSITORY

Search for groups, artifacts, categories

Indexed Artifacts (57.9M)

58.0M

43.0M

28.0M

14.0M

0

2006

2011

2016

2021

2025

Popular Categories

Testing Frameworks & Tools

Android Packages

JVM Languages

Logging Frameworks

Java Specifications

JSON Libraries

Core Utilities

Mocking

Web Assets

Annotation Libraries

Language Runtime

HTTP Clients

Logging Bridges

Dependency Injection

XML Processing

Web Frameworks

Android Platform

Home » io.jsonwebtoken » jjwt-api » 0.12.3

io.jsonwebtoken

jjwt-api

0.12.3

JWT :: API » 0.12.3

The public API for the JWT library, used to create and consume JSON Web Tokens in Java.

License

Apache 2.0

Categories

JWT Libraries

Tags

security json authentication jwt api

Date

Oct 15, 2023

Files

pom (1 KB)

jar (136 KB)

View All

Repositories

Central

Jenkins Public

Mulesoft

WSO2 Public

Ranking

#746 in MvnRepository (See Top Artifacts)

#5 in JWT Libraries

Used By

764 artifacts

Note: There is a new version for this artifact

New Version

0.12.6

Maven

Gradle

SBT

Mill

Ivy

Grape

Leiningen

Buildr

Scope: Compile

<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-api -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.12.3</version>

</dependency>

Рисунок 3.3 - JJwt бібліотека в MavenRepository

Після встановлення необхідних бібліотек перед початком програмування, потрібно додати конфігурацію для підключення до бази даних, а також параметри для формування jwt токена.

```

1  spring.application.name=surveyapp
2
3  spring.jpa.hibernate.ddl-auto=create
4  spring.datasource.url=jdbc:postgresql://localhost:5432/survey_database
5  spring.datasource.username=postgres
6  spring.datasource.password=13052004
7  spring.jpa.show-sql=true
8  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
9
10 jwt.secret=4gyu3242g34gu23g4u4h2y3yg4u2y4g423u4t7i6
11 jwt.lifetime=259200000
12
13 recaptcha.secret=6Ld0j1krAAAAACjy0Lydi9dpuT1pf7YnIwwE9mbv
14
15 app.encrypt.key=SecretKey
16
17 spring.mail.host=smtp.gmail.com
18 spring.mail.port=587
19 spring.mail.username=surveyappmailsender99@gmail.com
20 spring.mail.password=buhtpwdihggdbnhh
21 spring.mail.properties.mail.smtp.auth=true
22 spring.mail.properties.mail.smtp.starttls.enable=true

```

Рисунок 3.4 - Конфігурація в application.properties

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		43

Для підключення до вже створеної PostgreSQL бази даних використовуємо параметри посилання на базу даних, а також для підтвердження володінням базою використовуємо username та password.

Встановлюємо параметр create для hibernate.ddl-auto, таким чином під час запуску застосунку буде створені нові таблиці, у разі, якщо є база даних не перестворювати таблицю кожен раз під час запуску є варіант обрати update.

Для формування jwt будемо використовувати два параметри secret, котрий є може бути випадковим набором значень, а lifetime, котрий буде часом зберігання токена.

Для використання в сервісі такого інструменту, як CAPTCHA, на сайті компанії Google, що відповідає за інтеграцію в проєкти такого інструменту було сформовано два токени, один з яких знаходиться на фронтенд частині, і другий, що записаний в application.properties на сервері. Ці токени використовуються для підключення до Google API, та для генерації нових токенів, котрі в собі мають інформацію про проходження капчі.

Поле encrypt.key є одним з параметрів для шифрування даних бібліотекою Jasypt.

Щоб використовувати Java Mail Sender та відправляти повідомлення користувачам на пошту для активації акаунта, було прописано конфігурацію, в якій вказано, який тип пошти буде відправляти повідомлення, а також вказана пошта та пароль для автентифікації.

3.1.2 Створення моделі даних

Завдяки використанню бібліотеки Spring Data Jpa, котра в свою чергу є модифікацією для Hibernate, котрий відомий завдяки ORM.

ORM – технологія, що пов'язую класи з таблицями база даних. Таким чином для генерації таблиць та їх стовбців в базі даних необхідно створити

					ІАЛЦ.467200.003 ПЗ	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

відповідні класи-сутності, з об'єктами котрих в подальшому ми будемо взаємодіяти, і зберігати, оновлювати, отримувати або видаляти дані з таблиці.

Проте перед тим як почати створення сутностей, треба зробити декомпозицію. В нашому застосунку буде присутня реєстрація та автентифікація, тому зрозуміло, що нам знадобиться таблиця User, також є наші опитування, котрі також будуть мати свою таблицю, а опитування мають питання, та питання можуть мати варіанти відповіді. Окрім цього ще має бути таблицю, де будуть зберігатися відповіді на опитування користувачів, а також таблицю для збереження токенів активації акаунта, котрі будуть брати участь у генерації відповідного посилання на клієнську частину, з котрої буде відправлятися запит на серверну для встановлення параметра `enabled=true` у користувача.

Таким чином в результаті декомпозиції вийшли такі сутності

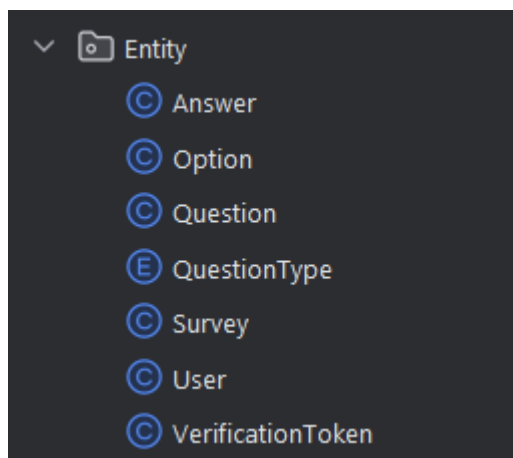


Рисунок 3.5 - Класи-сутності

Де клас User буде мати такі поля, які відповідно будуть стовпцями таблиці

- Id – унікальний ключ користувача
- Username – логін користувача, котрий буде використовуватися для реєстрації, та подальшої автентифікації в системи
- Email – поле, яке також необхідно вказати під час реєстрації, буде використовуватися для активації акаунта

- Password – також частина для Security шару, як і username буде використовуватися для генерації jwt токена, за яким буде відбуватися авторизація в системі
- Enabled – поле, яке використовується для перевірки, чи активований акаунт користувача, якщо параметр дорівнює false, користувачу буде відмовлено в автентифікації, якщо ж користувач перейшов за посиланням на пошті, і токен активації коректний, то значення встановлюється як true, і можна робити вхід в систему

Клас Survey має такі поля:

- Id – унікальний ключ
- Title – назва опитування
- Description – опис опитування
- requireAuth – в застосунку при створенні опитування можна буде вказати, чи потрібно для проходження цього опитування автентифікація в системі, чи ні, саме завдяки цьому полю, ми будемо реалізовувати цю логіку
- isActive – це поле дозволяє контролювати стан опитування, чи доступне воно ще, чи, той хто створив його вже закінчив
- isPublic – поле дозволяє контролювати які опитування є публічними, котрі будуть видні всім автентифікованим користувачам, а які приватними, тобто доступними за посиланням
- activeFrom – час коли опитування стане активним
- activeUntil – час коли опитування перестане бути активним, в застосунку користувач може вказувати ці параметри, і в разі, якщо час пройшов, опитування автоматично, без потреба вручну його закінчувати, перейде в isActive = false
- User creator – цим полем ми пов'язуємо таблиці Survey та User

Клас Question має такі поля:

					ІАЛЦ.467200.003 ПЗ	Арк.
						46
Зм.	Арк.	№ докум.	Підпис	Дата		

- Id - Унікальний ключ
- Survey survey – пов’язуємо таблиці survey та question, таким чином робимо зв’язку, що питання обов’язково мають опитування, до якого вони належать
- Text – текст питання
- questionType – це поле Enum(котре має три типи:single_choice, multiple_choice,text) – завдяки чому можна питанню на момент створення буде встановлено один з трьох цих типів,
- position – визначає яким по черзі в опитування стоїть питання

Клас Option

- id – унікальний ключ
- question – поєднує таблиці option та question
- text – текст варіанту
- position – позиція варіанта в питання

Клас VerificationToken

- id – унікальний ключ
- token – безпосередньо згенерований токен, який як параметр надходить у посиланні на пошту, для переходу та активації акаунта
- User user – пов’язує таблицю з таблицею user, з відношенням один до одного, тобто кожен користувач може мати тільки один токен активації
- expiryDate – час коли надісланий токен перестане буде дійсним.

3.1.3 Реалізація бізнес логіки застосунку

Бізнес логіка реалізована в сервісах застосунку, котрі обробляють дані отримані з клієнтської частини та модифікують поля бази даних

					ІАЛЦ.467200.003 ПЗ	Арк.
						47
Зм.	Арк.	№ докум.	Підпис	Дата		

використовуючи методи репозиторіїв, котрі є інструментом Spring Jpa і дозволяють робити запити до бази даних без потреби вручну писати SQL-запити.

```

9  @Repository
   public interface QuestionRepository extends JpaRepository<Question, Long> {
      2 usages  Danylo Zherdiev
10      List<Question> findBySurveyIdOrderByPositionAsc(Long surveyId);
11
      2 usages  Danylo Zherdiev
12      List<Question> findBySurveyId(Long surveyId);
13
14  }
15

```

Рисунок 3.6 - Приклад вигляду класу-репозиторію для таблиці Question

На прикладі методу createSurvey в SurveyService можна побачити за яким принципом будується бізнес логіка застосунку. Тобто до методу подається DTO, котре приходить з клієнської частини у форматі json, ці дані оброблюються та завдяки методам репозиторіям зберігають ці дані до таблиць.

Цей метод відповідно до його назви дозволяє зберігати нові створені опитування до бази. Параметрами методу є CreateSurveyRequest що є dto, котре має такі ж поля як і сутність опитування, та ім'я користувача

Спершу перевіряється чи існує взагалі користувач з таким ім'ям, якщо не існує, то викидається відповідна помилка.

Після цього відбувається підготовка об'єкта сутності Survey до збереження в таблицю, тобто з dto приймаються значення. А також окремо від збереження опитування, зберігаємо питання, котрим перед цим встановлено опитування, до якого вони відносяться, і зберігаються варіанти, яким встановлено до яких питань вони відносяться.

```

1 usage  ▲ Danylo Zherdiev
43 public Survey createSurvey(CreateSurveyRequest createSurveyRequest, String username) {
44     User creator = userRepository.findByUsername(username)
45     .orElseThrow(() -> new EntityNotFoundException("User not found"));
46     System.out.println("❖ createSurveyRequest - public" + createSurveyRequest.isPublic());
47
48     Survey survey = Survey.builder()
49         .title(createSurveyRequest.getTitle())
50         .description(createSurveyRequest.getDescription())
51         .requiresAuth(createSurveyRequest.isRequiresAuth())
52         .activeFrom(createSurveyRequest.getActiveFrom())
53         .activeUntil(createSurveyRequest.getActiveUntil())
54         .creator(creator)
55         .isActive(true)
56         .isPublic(createSurveyRequest.isPublic())
57         .build();
58     System.out.println("❖ saved survey - public" + survey.isPublic());
59     survey = surveyRepository.save(survey);
60
61     for (QuestionUpdateDto qRequest : createSurveyRequest.getQuestions()) {
62         Question question = Question.builder()
63             .survey(survey)
64             .text(qRequest.getText())
65             .questionType(qRequest.getType())
66             .position(qRequest.getPosition())
67             .build();
68
69         questionRepository.save(question);
70
71         if (qRequest.getType() == QuestionType.SINGLE_CHOICE || qRequest.getType() == QuestionType.MULTIPLE_CHOICE) {
72             List<Option> options = new ArrayList<>();
73             for (CreateOptionRequest opt : qRequest.getOptions()) {
74                 options.add(Option.builder()
75                     .question(question)
76                     .text(opt.getText())
77                     .position(opt.getPosition())
78                     .build());
79             }
80             optionRepository.saveAll(options);
81             question.setOptions(options);
82         }
83     }
84 }

```

Рисунок 3.7 - Метод createSurvey

3.1.4 Розробка контролерів

Контролери є ключовим компонентом веб застосунку, що відповідають за прийом і обробку HTTP-запитів, а також повернення відповідей клієнтській частині системи.

Розглянемо на прикладі SurveyController, котрий обробляє запити пов'язані з маніпуляціями над даними таблиці Survey.

```

▲ Danylo Zherdiev
22 ▼ @RestController
23 @RequestMapping("/api/surveys")
24 public class SurveyController {
25     17 usages
26     private final SurveyService surveyService;
27     2 usages
28     private final UserRepository userRepository;
29     1 usage
30     private final JwtService jwtService;
31
32     ▲ Danylo Zherdiev
33     public SurveyController(SurveyService surveyService, UserRepository userRepository, JwtService jwtService) {
34         this.surveyService = surveyService;
35         this.userRepository = userRepository;
36         this.jwtService = jwtService;
37     }
38
39     ▲ Danylo Zherdiev
40     @PostMapping
41     public ResponseEntity<Survey> createSurvey(@RequestBody CreateSurveyRequest createSurveyRequest, HttpServletRequest httpServletRequest) {
42         String username = surveyService.extractUsername(httpServletRequest);
43         if (username == null) return ResponseEntity.status(401).build();
44         Survey createdSurvey = surveyService.createSurvey(createSurveyRequest, username);
45         return ResponseEntity.ok(createdSurvey);
46     }
47 }

```

Рисунок 3.8 - Вигляд контролеру SurveyController

Контролер позначається анотацією RequestMapping, що вказує на яку адресу треба звертатися клієнту, щоб взаємодіяти з цим контролером та його методами, котрі можуть містити додатковий шлях

На прикладі метода createSurvey можна побачити, що він обробляє POST Http запит з клієнтської частини, отримуючи json та тіло запиту httpServletRequest, з котрого завдяки використанню методу extractUsername() з Header Authorization ми отримуємо jwt токен, та конвертуємо його в ім'я користувача.

Ми перевіряємо отримане ім'я, і якщо воно не знайдено, то повертає помилку, якщо ж ім'я існує, то автентифікацію користувача підтверджено і виконується метод createSurvey, котрий був розглянутий до цього, в нього ми передаємо дані з запиту та username

3.1.5 Створення Security рівня

Для забезпечення рівня безпеки під час автентифікації та авторизації, і також захисту даних та арі-ендпоінтів був створений security рівень, котрий складається з інструментів бібліотеки Spring Security, а також Jwt токен для забезпечення безстанної автентифікації.

Використовуючи базовий інтерфейс SecurityFilterChain зробимо конфігурацію Spring Security, в котрому буде вказано які ендпоінти потрібно захистити, а які залишити відкритими, наприклад для здійснення реєстрації та авторизації, вказуємо в якості параметрі методу requestMatchers().permitAll, що дозволяє неавтентифікованим користувачам відправляти запити на цю адерсу. Також вказуємо відповідний cors, який дозволяє надсилати клієнтській частині запити на серверну, а також через те, що ми використовуємо jwt встановлюємо значення фільтру, що буде перевіряти токени перед стандартним фільтром Spring Security

```

9  @Bean
10 @
11 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
12     return http
13         .csrf(csrf -> csrf.disable())
14         .cors(Customizer.withDefaults())
15         .authorizeHttpRequests(auth -> auth
16             .requestMatchers("/api/auth/**").permitAll()
17             .requestMatchers(HttpMethod.GET, "/api/surveys/{id}").permitAll()
18             .requestMatchers("/api/submit").permitAll()
19             .anyRequest().authenticated()
20         )
21         .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
22         .authenticationProvider(authenticationProvider())
23         .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class)
24         .build();
25 }

```

Рисунок 3.9 - Метод конфігурації Spring Security

Для створення токена, перевірки чи дійсний він ще, а також отримання імені користувача з jwt були створенні відповідні методи в JwtService, котрі в подальшому будуть потрібні в JwtAuthenticationFilter.

```

15 @Service
16 public class JwtService {
17
18     @Value("${jwt.secret}")
19     private String jwtSecret;
20
21     @Value("${jwt.expiration}")
22     private long jwtExpiration;
23
24     4 usages
25     private SecretKey secretKey;
26
27     Danylo Zherdiev
28     @PostConstruct
29     public void init() { this.secretKey = Keys.hmacShaKeyFor(jwtSecret.getBytes()); }
30
31     2 usages Danylo Zherdiev
32     public String generateToken(String username, Map<String, Object> extraClaims) {
33         return Jwts.builder()
34             .subject(username)
35             .claims(extraClaims)
36             .issuedAt(new Date())
37             .expiration(new Date(System.currentTimeMillis() + jwtExpiration))
38             .signWith(secretKey)
39             .compact();
40     }
41
42     3 usages Danylo Zherdiev
43     public String extractUsername(String token) {
44         return Jwts.parser().jwtParserBuilder()
45             .verifyWith(secretKey)
46             .build().jwtParser()
47             .parseSignedClaims(token).jwtClaims()
48             .getPayload().claims()
49             .getSubject();
50     }
51 }

```

Рисунок 3.10 - Частина коду класу JwtService

Основна перевірка на відповідність токену до даних зареєстрованого користувача з бази даних відбувається в методі `doFilterChain`, що виконує автоматичну автентифікацію користувачів на кожен запит, виключаючи тільки запити, що були вказані, як доступні, в конфігурації.

Тобто відповідно до всіх розроблених фільтрів, що б мати можливість використовувати більшу частину функціоналу застосунку, необхідно мати коректно згенерований токен. Для того щоб його згенерувати були створені методи `register` та `authenticate` в `AuthenticationService`

```
1 usage  Danylo Zherdiev
@ public AuthenticationResponse register(RegisterRequest request) {
    User user = new User();
    user.setUsername(request.getUsername());
    user.setEmail(request.getEmail());
    user.setPassword(passwordEncoder.encode(request.getPassword()));
    userRepository.save(user);

    String jwtToken = jwtService.generateToken(user.getUsername(), Map.of());
    return new AuthenticationResponse(jwtToken);
}
```

Рисунок 3.11 - Метод для реєстрації користувача

Для реєстрації з клієнтської частини отримуємо `dto` котре має ім'я користувача, його пошту та пароль. Завдяки використанню методів репозиторію ми зберігаємо ці дані, додатково при цьому використовуючи `passwordEncoder` зашифровуємо пароль, та генеруємо токен використовуючи метод з сервісу.

В методі автентифікації спершу йде перевірка чи правильний логін та пароль, якщо в результаті перевірки метод не прокинув виняток `BadCredentialsException`, генеруємо `jwt` токен перед цим отримавши юзера з бази.

```

1 usage  Danylo Zherdiev
@
public AuthenticationResponse authenticate(AuthenticationRequest request) {
    authManager.authenticate(
        new UsernamePasswordAuthenticationToken(
            request.getUsername(), request.getPassword()
        )
    );

    User user = userRepository.findByUsername(request.getUsername())
        .orElseThrow();

    String jwtToken = jwtService.generateToken(user.getUsername(), Map.of());
    return new AuthenticationResponse(jwtToken);
}
}

```

Рисунок 3.12 - Метод для автентифікації користувача

Ці методи викликаються коли з клієнтської частини на AuthController надається відповідний запит, та у відповідь повертається токен, котрий в подальшому на фронтенді буде зберігатися локально у користувача, і використовуватися в кожному запиті на ендпоінти серверної частини

```

Danylo Zherdiev
@PostMapping("@v"/"register")
public ResponseEntity<?> register(@RequestBody RegisterRequest request) {
    if (!recaptchaService.verify(request.getRecaptchaToken())){
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(Map.of( k1: "error", v1: "Невірна reCAPTCHA"));
    }
    authenticationService.register(request);
    return ResponseEntity.ok(Map.of( k1: "message", v1: "Перевірте пошту"));
}

Danylo Zherdiev
@PostMapping("@v"/"login")
public ResponseEntity<AuthenticationResponse> login(@RequestBody AuthenticationRequest request) {
    User user = userRepository.findByUsername(request.getUsername()).get();
    if (!user.isEnabled()){
        System.out.println("Акаунт користувача не активований");
        throw new DisabledException("Потрібна активація через пошту");
    }else {
        return ResponseEntity.ok(authenticationService.authenticate(request));
    }
}
}

```

Рисунок 3.13 – Методи проходження реєстрації та автентифікації

Окрім використання jwt та Spring Security з фільтрами, ще одним шаром для проходження реєстрації та автентифікації для подальшої генерації jwt токена є активації акаунта користувачем, шляхом переходу за посиланням надісланим на пошту. В класі Email Service з використання інтерфейсу

JavaMailSender відповідної підключеної бібліотеки, відбувається генерація та відправка повідомлення користувачу.

```

7  @Service
8  public class EmailService {
9
10     2 usages
11     private final JavaMailSender javaMailSender;
12
13     1 usage
14     private String activationBaseUrl = "http://localhost:4200";
15
16     1 usage
17     public EmailService(JavaMailSender javaMailSender) { this.javaMailSender = javaMailSender; }
18
19     1 usage
20     public void sendActivationEmail(String to, String token) {
21         String link = activationBaseUrl + "/activate?token=" + token;
22         String subject = "Активация аккаунта";
23         String body =
24             """
25             Вітаємо.
26             Перейдіть за наданим посиланням для активації аккаунта, після цього ви можете пройти аутентифікацію в системі.
27             """ + link;
28
29         SimpleMailMessage message = new SimpleMailMessage();
30         message.setTo(to);
31         message.setSubject(subject);
32         message.setText(body);
33         javaMailSender.send(message);
34     }

```

Рисунок 3.14 – Сервіс для створення та відправки повідомлення на email

В результаті чого користувач отримує посилання на шаблон компонента клієнтської частини, котрий робить http запит на ендпоінт сервера за шляхом /api/auth/activate в котрому проходить перевірка, чи співпадає токен з запиту та токен, що знаходиться в таблиці VerificationToken та відповідає відповідному користувачу.

```

1  Danylo Zherdiev
2  @GetMapping("/{token}")
3  public ResponseEntity<String> activate(@RequestParam String token) {
4      var verifToken = verificationTokenRepository.findByToken(token);
5      if (verifToken.isEmpty()) {
6          return ResponseEntity.badRequest().body("Невірний токен");
7      }
8      VerificationToken vtok = verifToken.get();
9      if (vtok.getExpiryDate().isBefore(Instant.now())) {
10         return ResponseEntity.badRequest().body("Термін дії токена сплив");
11     }
12     User user = vtok.getUser();
13     user.setEnabled(true);
14     userRepository.save(user);
15     return ResponseEntity.ok().body("Акаунт успішно активовано");
16 }

```

Рисунок 3.15 – ендпоінт для активації акаунта користувача

3.2 Розробка клієнтської частини

Клієнтська частина буде надавати користувачу шаблони, взаємодія з якими буде призводити до відправки відповідних запитів до серверної частини для отримання оброблених даних, або виконання якихось модифікацій в базі. Для розробки клієнтської частини буде використано мову програмування TypeScript, яка є строго типізованим варіантом JavaScript, та фреймворк, який використовує цю мову Angular

3.2.1 Ініціалізація Angular проєкту

Перед тим як почати роботу з Angular необхідно встановити Angular CLI командою `npm install -g @angular/cli@16.2.16`, це інструмент, що дозволить використовуючи командний рядок, зручно створювати компоненти застосунку.

Після цього використовуючи команду `ng new survey-app` згенеруємо проєкт Angular, котрий за замовчуванням буде мати необхідні бібліотеки, налаштування для зборки застосунку, роутінг і тд.

Також для стилізації шаблонів буде використаний фреймворк Bootstrap, котрий також потрібно встановити командою `npm install bootstrap@5 --save`, та додати його до стилів в конфігураційний файл `angular.json`

```
{
  "src/assets"
},
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
```

Рисунок 3.16 - Стили в angular.json

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		55

Закінчивши з встановленням необхідних інструментів та їх налагодженням, використовуючи angular cli сформуємо відповідну структуру застосунку шляхом створення папок з компонентами та їхніми шаблонами

```
PS E:\AngularProjects\survey-frontend\survey-app> ng generate component features/create-survey
```

Рисунок 3.17 - Приклад команди для створення компонента

В результаті генерація було сформовані компоненти, котрі завдяки взаємодії через налаштований роутінг будуть утворювати повноцінний функціональний застосунок

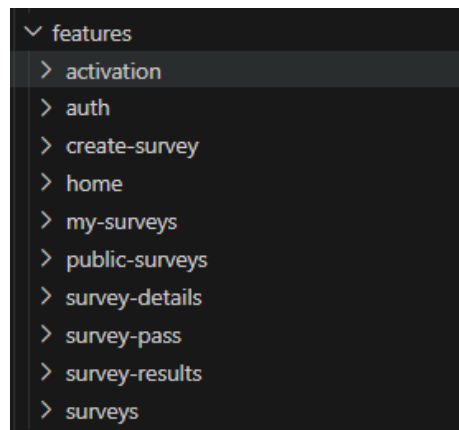


Рисунок 3.18 - Згенеровані компоненти

Для навігації по застосунку було визначено маршрути в роутері застосунку

```
const routes: Routes = [
  {path: 'login', component: LoginComponent},
  {path: '', component: HomeComponent},
  {path: 'register', component: RegisterComponent},
  {path: 'activate', component: ActivationComponent},

  {path: 'my-surveys', component: MySurveysComponent, canActivate: [AuthGuard]},
  {path: 'create-survey', component: CreateSurveyComponent, canActivate: [AuthGuard]},
  {path: 'survey-results/:id', component: SurveyResultsComponent, canActivate: [AuthGuard]},
  {path: 'surveys/public', component: PublicSurveysComponent, canActivate: [AuthGuard]},

  {path: 'survey/:id', component: SurveyPassComponent},
  {path: 'survey-details/:id', component: SurveyDetailsComponent, canActivate: [AuthGuard]},

  {path: '**', redirectTo: ''}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Рисунок 3.19 - Роутінг в застосунку

3.2.2 Створення компонента реєстрації та автентифікації

Для доступу до функціоналу застосунку необхідно бути зареєстрованим та автентифікованим, для цього потрібно отримати та зберегти на клієнті в localStorage jwt токен, отриманий від серверної частини за встановленим ендпоінтом. Для виконання запитів на ендпоінти AuthController був створений сервіс котрий використовуючи Http Post запити звертається до нашого бекенду

```
export class AuthService {  
    private baseUrl = 'http://localhost:8080/api/auth';  
    constructor(private http: HttpClient) { }  
  
    login(request: LoginRequest): Observable<any> {  
        return this.http.post(`${this.baseUrl}/login`, request);  
    }  
    register(request: RegisterRequest): Observable<any> {  
        return this.http.post(`${this.baseUrl}/register`, request);  
    }  
  
    saveToken(token: string): void {  
        localStorage.setItem('token', token);  
    }  
  
    getToken(): string | null {  
        return localStorage.getItem('token');  
    }  
  
    logout(): void {  
        localStorage.removeItem('token');  
    }  
  
    isLoggedIn(): boolean {  
        return !!this.getToken();  
    }  
  
    activate(token: string): Observable<string> {  
        return this.http.get(`${this.baseUrl}/activate`, {  
            params: { token },  
            responseType: 'text'  
        });  
    }  
}
```

Рисунок 3.20 - Сервіс AuthService

Ці методи будуть використовуватися в компонентах login.component.ts та register.component.ts

Завдяки двосторонньому зв'язуванню ми отримуємо в компонент змінні з шаблону, котрі ввів користувач і в методах компонентів викликаємо методи з сервісу authService відправляючи запит на серверну частину.

В разі успіху викладаємо метод `saveToken` котрий збереже токен в `localStorage`, і при спробі запиту на сервесі буде діставатися звідти та використовуватися для авторизації.

```
export class LoginComponent {
  username='';
  password='';
  error='';

  constructor(private authService: AuthService, private router: Router){}

  login():void{
    this.authService.login({username:this.username,password:this.password})
      .subscribe({
        next: (res) => {
          this.authService.saveToken(res.token);
          this.router.navigate([''])
        },
        error: (err)=>{
          this.error = 'Неправильні дані для входу';
          console.error(err);
        }
      })
  }
}
```

Рисунок 3.21 - Метод для автентифікації

3.2.3 Розробка сервісів

Сервіси – це класи, які можуть реалізовувати бізнес-логіку та надавати доступ до зовнішніх ресурсів, наприклад до розробленої серверної частини. Головною перевагою сервісів є можливість використовувати розроблені функції в компонентах застосунку, що робить сервіси необхідними для покращення структури коду застосунку, прибираючи дублювання коду в компонентах.

Для створення запитів до розробленої серверної частини, в компонентах буде використовуватися методи розроблених сервісів, котрі будуть створювати необхідні http запити з адресою на відповідні до цілей запиту, ендпоінти, на яких функціоную сервер

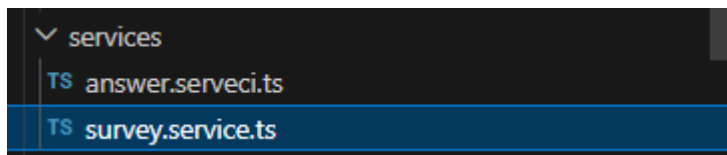


Рисунок 3.22 - Розроблені сервіси

Структурно сервіси схожі на компоненти, проте головна відмінність, що їх відрізняє це наявність декоратору `@Injectable`, що дозволяє імпортувати цей сервіс та використовувати його функції в будь якому компоненті сервісу.

```
@Injectable({
  providedIn: 'root'
})
export class SurveyService {
  private baseUrl = 'http://localhost:8080/api/surveys';

  constructor(private http: HttpClient) {}

  getMySurveys(): Observable<Survey[]> {
    return this.http.get<Survey[]>(`${this.baseUrl}`);
  }

  createSurvey(request: CreateSurveyRequest): Observable<any> {
    return this.http.post(`${this.baseUrl}`, request);
  }

  getSurveyById(id:number): Observable<Survey>{
    return this.http.get<Survey>(`${this.baseUrl}/${id}`);
  }

  getSurveyDetailsById(id:number): Observable<Survey>[]
  return this.http.get<Survey>(`${this.baseUrl}/survey-details/${id}`)
  ]

  updateQuestion(questionId:number, data:QuestionRequest):Observable<any>{
    return this.http.put<any>(`http://localhost:8080/api/questions/${questionId}`,data);
  }

  addQuestion(surveyId: number, question: any): Observable<any> {
    return this.http.post<any>(`${this.baseUrl}/${surveyId}/questions`, question);
  }
}
```

Рисунок 3.23 - Частина коду SurveyService

3.2.4 Створення компонентів

На прикладі компоненту `CreateSurveyComponent` можемо визначити, що компонент на відміну від сервісів, які надають методи всьому проєкту, реалізує інкапсульований функціонал для свого конкретного шаблону, позначеному в

декораторі @Component параметром templateUrl, шляхом надання методів для виконання певних дій у відповідь на дії користувача на сторінці.

Цей компонент надає функції для створення опитування, це як і додавання питання на сторінці або варіанта відповіді так і їх видалення, встановлення назви опитування, його опису. А також вибір типу питання, котрі підтримуються в серверній частині, це текстове питання, питання з однією відповіддю, або питання з декількома відповідями. Також шаблон має декілька Boolean змінних для встановлення користувачем, чи буде опитування публічним та чи буде воно активним. А також для автоматичного контролю доступу опитування користувачам за часом, користувач може обрати часовий проміжок коли опитування розпочнеться і коли закінчиться.

В методі submit, що підтверджує створення опитування, викликається в момент натискання кнопки на сторінці, викликається метод createSurvey в surveyService, котрий робить post запит серверну частину передаючи json з встановленими параметрами, котрі сервер має прийняти як dto та обробити в сервісному методі

```
submit(): void {
  const isoFrom = this.activeFromDate && this.activeFromTime ? `${this.activeFromDate}T${this.activeFromTime}:00Z` : undefined;
  const isoUntil = this.activeUntilDate && this.activeUntilTime ? `${this.activeUntilDate}T${this.activeUntilTime}:00Z` : undefined;
  const request: CreateSurveyRequest = {
    title: this.title,
    description: this.description,
    requireAuth: this.requireAuth,
    isPublic: this.isPublic,
    questions: this.questions,

    activeFrom: isoFrom,
    activeUntil: isoUntil
  };
  console.log('>>> payload:', request);
  this.surveyService.createSurvey(request).subscribe({
    next: () => this.router.navigate(['/my-surveys']),
    error: (err) => console.error('Помилка при створенні опитування:', err)
  });
}
```

Рисунок 3.24 - Метод для підтвердження створення опитування

В параметр до методу ми передаємо dto CreateSurveyRequest котра має відповідати dto. що буде приймати ці дані на сервері

					ІАЛЦ.467200.003 ПЗ	Арк.
						60
Зм.	Арк.	№ докум.	Підпис	Дата		

```
createSurvey(request: CreateSurveyRequest): Observable<any> {
  return this.http.post(`${this.baseUrl}`, request);
}
```

Рисунок 3.25 - Метод для відправки запиту на сервер для створення опитування

3.2.4 Розробка шаблонів

Для роботи зі змінними та методами компонентів Angular надає двосторонню прив'язку, котра реалізується за допомогою використання директиви ngModel, тож коли користувач вводить в поле якусь інформацію паралельно ця інформація записується в змінну на фронтенді, і використовується для відправлення на серверну частину. Також angular надає інструменти для динамічного формування шаблонів, відповідно до даних.

Розглянемо частину коді з шаблону компонента для створення опитувань, використовуючи інструмент *ngIf можемо динамічно змінювати шаблон відповідно до стану, на прикладі ми перевіряємо параметр тип питання в q, і якщо тип станове не питання з текстовою відповіддю, то на сторінці з'являється новий блок, котрий використовує інструмент *ngFor для динамічного формування списку, в цьому коді він буде шукати та виводити всі варіанти відповіді на питання.

```

132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
<div *ngIf="q.type !== 'TEXT'" class="mb-2">
  <div *ngFor="let opt of q.options; let j = index" class="input-group mb-2">
    <input
      type="text"
      class="form-control"
      [(ngModel)]="q.options![j].text"
      name="option{{i}}-{{j}}"
      placeholder="Варіант відповіді"
      required
    />
    <button
      type="button"
      class="btn btn-outline-danger"
      (click)="removeOption(i, j)"
    >
      Видалити
    </button>
  </div>
  <button
    type="button"
    class="btn btn-outline-primary"
    (click)="addOption(i)"
  >
    Додати варіант
  </button>
</div>

```

Рисунок 3.26 - Приклад використання директив Angular в шаблоні компоненту createSurvey

Для забезпечення комфорту користувача у використанні веб застосунку необхідний сучасний зручний інтерфейс застосунку. Для цього в проекті для стилізації компонентів був використаний фреймворк bootstrap 5, завдяки ньому ми встановлюємо на яких місцях в сторінках будуть знаходитися блоки шаблону, а також використовуємо інші класи для покращення UI/UX дизайну застосунку

```

<div class="row g-3 mb-4">
  <div class="col-md-6">

```

Рисунок 3.27 - Клас Bootstrap фреймворку для розташування елементів на сторінці

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділі був оглянутий процес розробки нашого застосунку. Було окремо розглянуто етапи розробки серверної частини, котра включала в собі ініціалізацію та встановлення необхідних бібліотек для програмування застосунку. Для взаємодії з таблицями бази даних були створені сутності даних та використані методи бібліотеки Data Jpa для створення запитів до бази, без використання SQL мови. Використовуючи ці методи була розроблена бізнес логіка застосунку, яка буде працювати з отриманими з клієнтської частини або з таблиці даними.

Після цього була розроблена клієнтська частина на Angular, котра надає користувачу коректне представлення даних отриманих з серверної частини, та робить необхідні запити на неї для маніпуляції з даними.

В результаті розробки був створений сервіс, у вигляді web-застосунку, для створення та проведення соціальних опитувань.

					ІАЛЦ.467200.003 ПЗ	Арк.
						63
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4

ТЕСТУВАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Важливим етапом розробки програмного забезпечення є його тестування, щоб подивитися та впевнитися, що клієнтська та серверна частини взаємодіють правильно, і не виникають непередбачуваних помилок під час роботи застосунку.

Для тестування запусимо backend та frontend на портах 8080 та 4200, відповідно

```
2025-06-01T13:33:37.679+03:00 INFO 5888 --- [surveyapp] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-06-01T13:33:37.686+03:00 INFO 5888 --- [surveyapp] [ restartedMain] c.e.surveyapp.SurveyappApplication : Started SurveyappApplication in 4.74 seconds (process running)
```

Рисунок 4.1 - Запуск серверної частини

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
runtime.js          | runtime | 6.50 kB |
4 unchanged chunks

Build at: 2025-06-01T10:36:27.047Z - Hash: 2c0080be4d9233e4 - Time: 8701ms

✓ Compiled successfully.
```

Рисунок 4.2 - Запуск клієнтської частини

4.1 Огляд головної сторінки застосунку

Після запуску застосунку та переходу за пустим шляхом ми опиняємося на головній сторінці, котра має дві форми, до автентифікації та після.

До проходження реєстрації та автентифікації головна сторінка має такий вигляд, головна ціль, це запропонувати користувачу пройти автентифікацію, щоб користувач зміг отримати доступ до більшої частини функціоналу застосунку.

					ІАЛЦ.467200.003 ПЗ	Арк.
						64
Зм.	Арк.	№ докум.	Підпис	Дата		

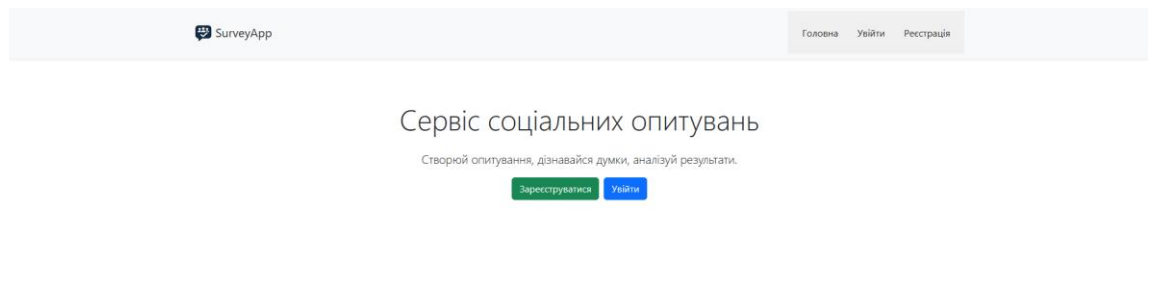


Рисунок 4.3 - Вигляд головної сторінки до автентифікації

Після того як користувач заїде в систему, отримає та збереже jwt токен в браузері, йому буде доступний новий вигляд головної сторінки, котра в навігаційному меню має такі компоненти як Головна, Публічні, Мої, Створити, Вийти.

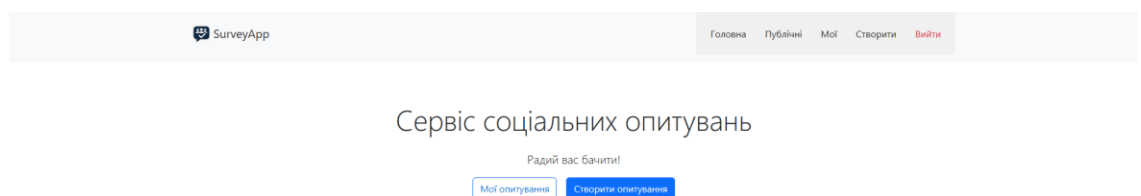


Рисунок 4.4 - Вигляд головної сторінки після автентифікації

Проаналізуємо кожен з компонентів навігації

Головна – користувач може повернутися на головну сторінку

Публічні – при створенні соціальних опитувань користувач може обрати, чи буде це опитування публічним, якщо так, то список таких опитувань буде знаходитися за цим шляхом

Мої – Знаходяться створені користувачем опитування, можна подивитися які параметри має опитування, відредагувати опитування, та отримати результати проведення опитування

Створити – сторінки для створення нових опитувань

					ІАЛЦ.467200.003 ПЗ	Арк.
						65
Зм.	Арк.	№ докум.	Підпис	Дата		

Вийти – користувач робить вихід з системи, його jwt токен видаляється з локального сховища.

4.2 Тестування реєстрації та автентифікації

Для проходження реєстрації та автентифікації необхідно обрати відповідні варіанти в меню навігації головної сторінки, у разі вибору реєстрації буде запропоновано форму, для вводу логіну, пошти та паролю, а також необхідно буде пройти капчу.

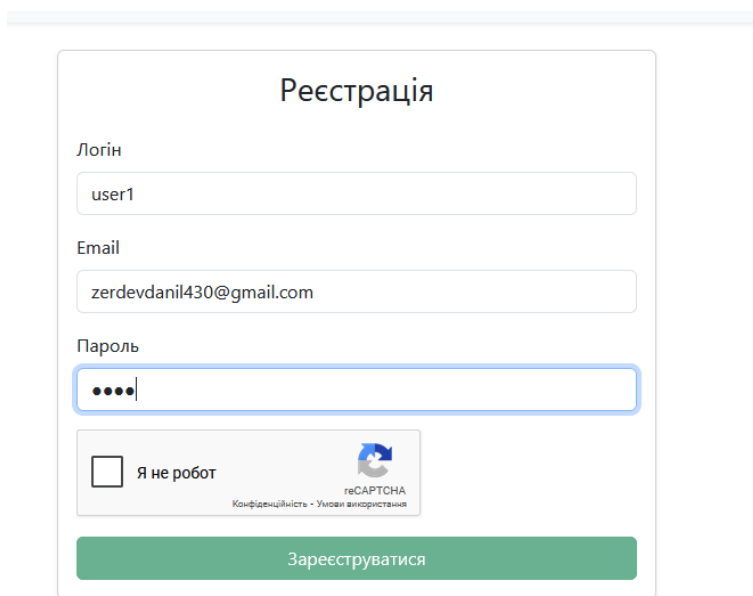
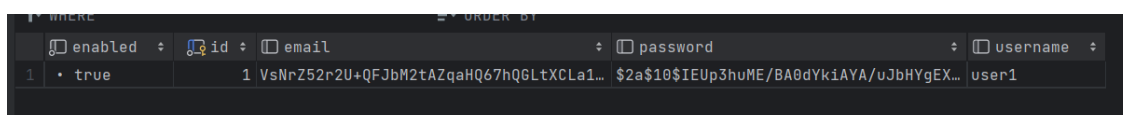


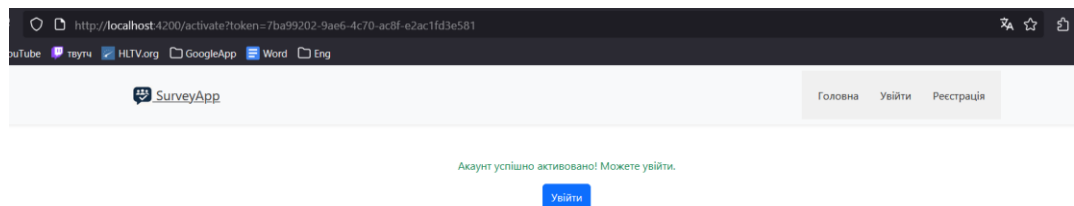
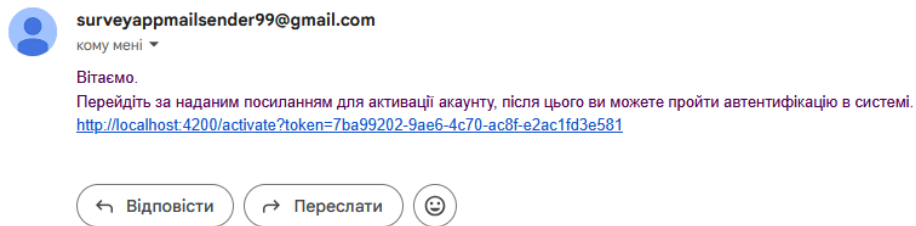
Рисунок 4.5 - форма для реєстрації

Після проходження процесу реєстрації, дані відправляються на сервер, де вони зберігаються в таблицю і користувача редиректить на сторінку входу в сервіс, проте перед тим як перейти до автентифікації, на пошту було відправлене повідомлення для активації акаунта.



enabled	id	email	password	username
1	1	VsNrZ52r2U+QFJbm2tAZqaHQ67hQGltXCLa1...	\$2a\$10\$IEUp3huME/BA0dYkiAYA/uJbHYgEX...	user1

Рисунок 4.6 - Збережені дані в таблиці user



Після відкриття форми для автентифікації користувач має ввести свій логін та пароль, на основі котрих буде сформований та збережений jwt токен. У разі некоректних введених даних, форма видасть помилку.

Рисунок 4.9 - Результат неправильно введенных данных

В результаті коректної автентифікації, користувача перекине на головну сторінку.

Рисунок 4.10 - Збережений токен в результаті проходження автентифікації

					ІАЛЦ.467200.003 ПЗ	Арк.
						67
Зм.	Арк.	№ докум.	Підпис	Дата		

Таким чином після збереження токена, він буде використаний в Заголовках запиту для авторизації користувача.

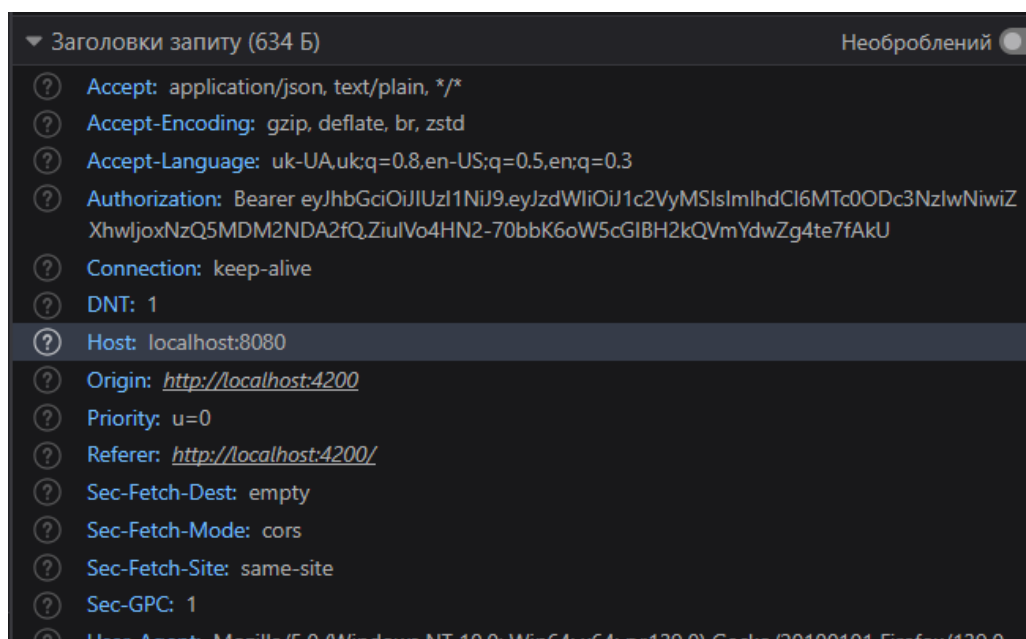


Рисунок 4.11 - Приклад запиту на захищений маршрут

4.3 Тестування створення опитування

При переході на маршрут create-survey відривається шаблон компоненту CreateSurvey.

На відкритій сторінці користувач обирає назву опитування його опис, чи буде воно анонімним, тобто потребувати авторизації, і чи буде воно публічним, або буде доступним тільки за наданим автором посилання. Для автоматичного контролю за доступністю для користувачів, можна встановити часові пороги, котрі будуть використовуватися для перевірки, чи активне ще опитування, чи ні. Та безпосередньо присутній функціонал для додавання питань, в яких вказується назва питання, його тип, та в залежності від типу пропонується додавання варіантів опитування, якщо питання потребує не текстової відповіді

Для тестування роботи було створене тестове опитування, котре має всі заповнені параметри, та три питання кожного з можливих типів.

Створення опитування

Назва опитування

Дистанційна робота: вплив на життя та психічне здоров'я

☐ Анонімне опитування
☒ Публічне опитування

Опис опитування

дистанційний формат позначається на вашому розпорядку, продуктивності й емоційному стані. Будь ласка, відповідайте щиро — ваше бачення допоможе краще зрозуміти потреби працівників і створити комфортні умови в майбутньому.

Початок активності

Кінець активності

01.06.2025

15:00

03.06.2025

22:00

Питання 1

Як давно ви працюєте повністю або частково дистанційно

[Видалити](#)

Текстове

Питання 2

Як дистанційний режим позначається на вашій продуктивності

[Видалити](#)

Один варіант

Став більш ефективним

[Видалити](#)

Жодних змін

[Видалити](#)

негативно

[Видалити](#)

Додати варіант

Питання 3

Які переваги дистанційної роботи на вашу думку

[Видалити](#)

Кілька варіантів

Економія часу на дорогу

[Видалити](#)

Комфортне середовище

[Видалити](#)

Краще співвідношення робота/особисте життя

[Видалити](#)

Додати варіант

Додати питання

[Створити опитування](#)

Рисунок 4.12 - Створення опитування

Після створення опитування шляхом натискання відповідної кнопки, відбувається запит на ендпоінт серверної частини, для збереження цього опитування в базу даних, перед цим пройшовши шифрування завдяки використанню бібліотеку Jasypt та класу, що його реалізує StringEncryptConverter, котрий шифрує всі рядкові значення в таблиці.

```

Danylo Zherdiev
@Converter(autoApply = true)
public class StringEncryptConverter implements AttributeConverter<String,String> {
    5 usages
    private StandardPBEStrngEncryptor encryptor;

no usages  Danylo Zherdiev
    public StringEncryptConverter(){
        this.encryptor = new StandardPBEStrngEncryptor();
        this.encryptor.setAlgorithm("PBEWithMD5AndDES");
        this.encryptor.setPassword("SecretKey");
    }

Danylo Zherdiev
    @Override
    public String convertToDatabaseColumn(String attribute) {
        if (attribute == null) return null;
        return encryptor.encrypt(attribute);
    }

Danylo Zherdiev
    @Override
    public String convertToEntityAttribute(String dbData) {
        if (dbData == null) return null;
        return encryptor.decrypt(dbData);
    }
}

```

Рисунок 4.13 – Клас, що шифрує та дешифрує String поля в таблицях

В результаті створення та збереження опитування, а також його шифрування для забезпечення конфіденційності опитування в разі витoku інформації з бази даних, маємо такі дані в таблицях.

id	creator_id	is_active	is_public	require_auth	title	description
1	1	true	*	false	Http://l2ocF2M7ng-NfKqV5XUMH7-mvT9McAgAf1fo80UzPvN2WZbvr7Iouq380fvcN1tdmnp3E...	TaPuEcJ80vz3z3J3J3rphsnt3IBtX3L1/vi@WbJ0KtpJQlnrFFE-MbC30bW5y3Xmisp7F4f+U3/qUp...

Рисунок 4.14 - Збережене в таблиці опитування

					ІАЛЦ.467200.003 ПЗ	Арк.
						70
Зм.	Арк.	№ докум.	Підпис	Дата		

	position	id	survey_id	question_type	text
1	0	1	1	TEXT	Xqx/5b7Wi538X8SmWLhQYbHPDX8wjaQTnCzzZDIlaJI0tah...
2	1	2	1	SINGLE_CHOICE	JW4D3guyqj7KPwnf92nyET1MPEMtJVwReQH/I8EmWzrTAKD...
3	2	3	1	MULTIPLE_CHOICE	N3aHNPhIsnkvysYyn9Gp65M++e0Ekoeb7hVCdMxcpc08fL3...

Рисунок 4.15- Збережені в таблиці питання, пов’язані в опитуванням за його id

	position	id	question_id	text
1	0	1	2	2JWNe1Vh6N5yiAKFuGxewpG6Va3/YxAnqSjoEfYh7sbbnoR...
2	1	2	2	jaDty6tJ6xy3SbQxMastwc+DLEhIuxbkfP/ee4x7mdU=
3	2	3	2	+Y1Iw6Pq/rYrYC2cQy7RjDZ2Aeuwxv9mQUFyzjcjFMnc=
4	0	4	3	yAsYxbG7Cxp1qubL/h6+rZK2uWxVIMP/SbEfMNDbcgkZ4RM...
5	1	5	3	dKWam9oBrL0bN3L6kEvsVQkkupw/ajam+VhTdW5HZYKFwJC...
6	2	6	3	ZsuvD9vAhzXwQq5SAH6h7ue+yUFnMbIinhHmKILC3BZd4Qn...

Рисунок 4.16 - Збережені в таблиці варіанти відповіді, пов’язані з питанням, за його id

4.4 Тестування можливостей редагування опитування та генерації результатів проходження

Для редагування опитування або перевірки його результатів необхідно перейти на вкладку Мої з меню навігації, саме там будуть знаходитися створені опитування.

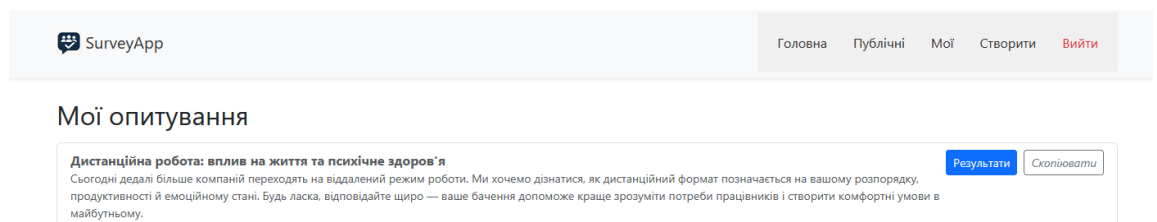


Рисунок 4.17 - Вигляд сторінки “Мої”

При переході на створене до цього опитування, відкриється можливість для його редагування, або автор також має можливість вручну закінчити

опитування, тобто зробити його неактивним. Змінимо в Питання , третій варіант “негативно” на “дуже негативно” для перевірки можливості редагування.

В результаті перевіримо дані таблиці options, для підтвердження змін, де можна побачити зміну в значеннях ідентифікаторів та значень зашифрованих варіантів.

	position	id	question_id	text
1	0	4	3	yAsYxb67Cxp1qubL/h6+rZK2uWxVIMP/SbEfMNDbcgkZ4RM...
2	1	5	3	dKWam9oBrL0bN3L6kEvsVQkkupw/ajam+VhTdW5HZYKFwJC...
3	2	6	3	ZsuvD9vAhzXwQq5SAH6h7ue+yUFnMbIinhHmKILC3Bzd4Qn...
4	0	7	2	aw8DejNUR7YJcn2mLk7+QzzCo5guoDKqiQTLlk/If4KtEzz...
5	1	8	2	eA2BSdspzCPfkTBu09epP6zSI2wtNLVqNZRt97JQK2Y=
6	2	9	2	RuJlS3Ub/BmSE9KzRtisdnjXf9LxRnEKsMXUiRvTN910CCC...

Рисунок 4.18 - Результат редагування варіанту “негативно” на “дуже негативно”

Також варто зазначити, що для того щоб запобігти підміну питань та варіантів, після проходження опитування якимось користувачем, заборонено редагувати опитування.

Дистанційна робота: вплив на життя та психічне здоров'я

Сьогодні дедалі більше компаній переходять на віддалений режим роботи. Ми хочемо дізнатися, як дистанційний формат позначається на вашому розпорядку, продуктивності й емоційному стані. Будь ласка, відповідайте щиро — ваше бачення допоможе краще зрозуміти потреби працівників і створити комфортні умови в майбутньому.

☐ Публічне опитування

☒ Потребує авторизації

Питання 1:

Як давно ви працюєте повністю або частково дистанційно?

TEXT

Редагувати

Неможливо редагувати: на питання вже були надані відповіді.

Питання 2:

Як дистанційний режим позначився на вашій продуктивності?

SINGLE CHOICE

Стає більш ефективним

Жодних змін

Дуже негативно

Редагувати

Неможливо редагувати: на питання вже були надані відповіді.

Питання 3:

Які переваги дистанційної роботи ви відчуваєте найчастіше?

MULTIPLE CHOICE

Економія часу на дорогу

Гнучкий графік

Комфортне середовище

Краще співвідношення роботи/особисте життя

Редагувати

Неможливо редагувати: на питання вже були надані відповіді.

Додати питання

Зберегти зміни

Завершити опитування

Рисунок 4.19 - Спроба редагування після проходження користувачем опитування

Якщо ж вручну завершити опитування, то при спробі користувача пройти опитування він отримає таку сторінку.

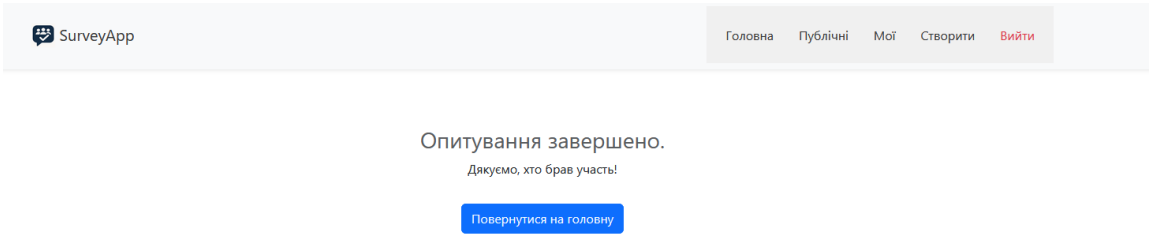


Рисунок 4.20 - Сторінка опитування після його закриття автором, або після закінчення терміну дії

Для того щоб отримати результати, необхідно повернутися на мої опитування, та натиснувши на клавішу результати можна побачити обрані користувачами варіант як в текстовому виді, так і у виді діаграми.

Аналітика: Дистанційна робота: вплив на життя та психічне здоров'я

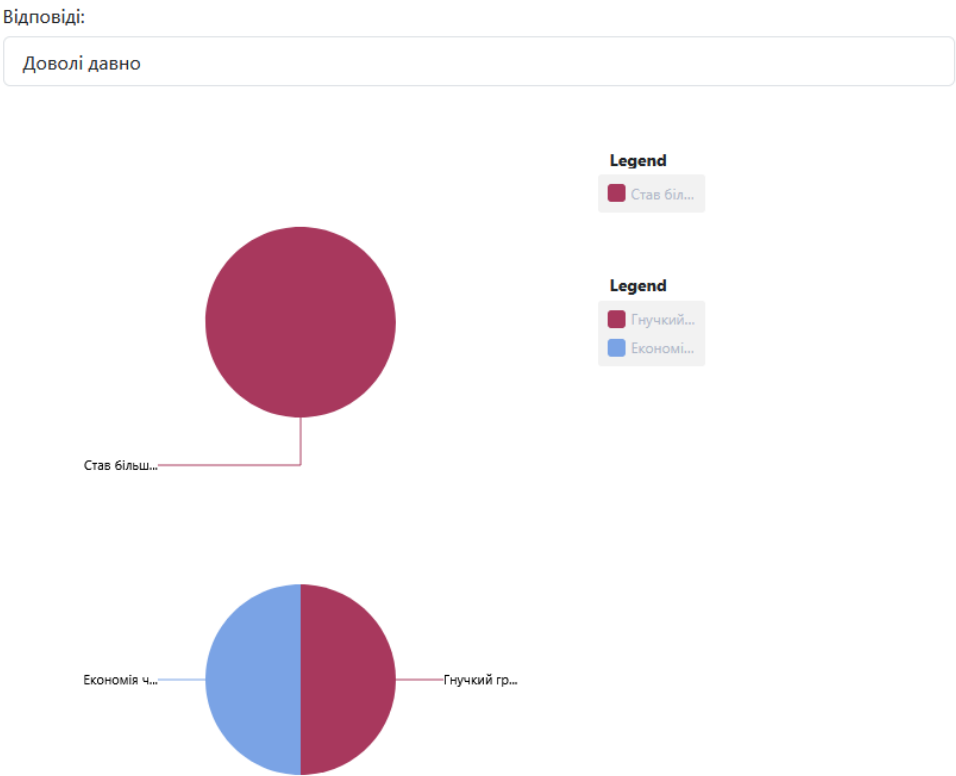


Рисунок 4.21 - Сторінка з результатами опитування

4.5 Тестування функціоналу проходження опитування

Для тестування проходження опитування, враховуючи, що ми створили публічне опитування, котре може бачити будь який автентифікований користувач, необхідно перейти на вкладку Публічні в меню навігації, та відкрити цікаве для нас опитування.

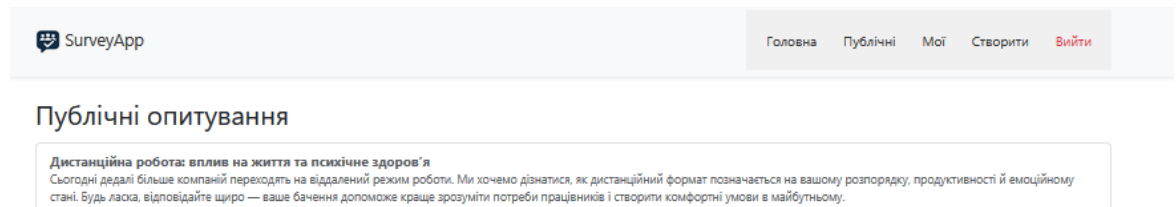


Рисунок 4.22 - Вигляд сторінки “Публічні”

Для успішного проходження опитування потрібно обов’язково вказати варіанти відповіді на всі питання в опитуванні, в разі не виконання цього, кнопка “Надіслати відповідь” буде недоступна.

Дистанційна робота: вплив на життя та психічне здоров'я

Сьогодні дедалі більше компаній переходять на віддалений режим роботи. Ми хочемо дізнатися, як дистанційний формат позначається на вашому розпорядку, продуктивності й емоційному стані. Будь ласка, відповідайте щиро — ваше бачення допоможе краще зрозуміти потреби працівників і створити комфортні умови в майбутньому.

1. Як давно ви працюєте повністю або частково дистанційно?

2. Як дистанційний режим позначився на вашій продуктивності?

☐ Став більш ефективним
☐ Жодних змін
☐ дуже негативно

Будь ласка, оберіть варіант.

3. Які переваги дистанційної роботи ви відчуваєте найчастіше?

☒ Економія часу на дорогу
☒ Гнучкий графік
☐ Комфортне середовище
☐ Краще співвідношення робота/особисте життя

Рисунок 4. 23 - Вигляд форми для проходження опитування

					ІАЛЦ.467200.003 ПЗ	Арк.
						74
Зм.	Арк.	№ докум.	Підпис	Дата		

Після проходження опитування дані про відповіді користувача будуть збережені в таблицю, і користувачу буде заборонено повторно проходити це опитування.

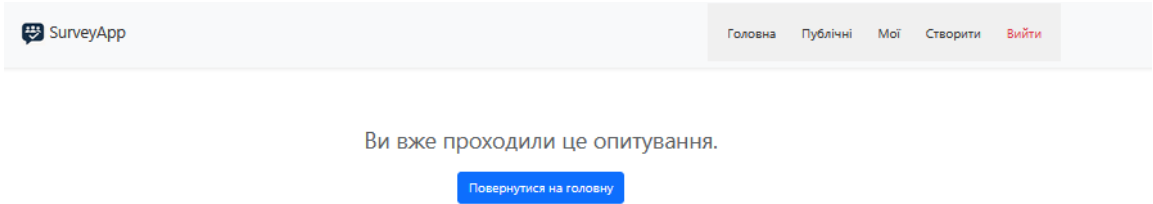


Рисунок 4.24 - Спроба повторного проходження опитування

4	4	1	1	1749452170094	Tntt12QhsnFDfRp7jKoSR92TyL4LHDKGq2y6Z3IWgXY=
5	5	2	1	1749452170311	<null>
6	6	3	1	1749452170320	<null>

Рисунок 4.25 - Збереження відповідей користувача

	answer_id	option_id
1	6	4
2	6	5
3	5	8

Рисунок 4.26 - Збереження варіантів відповіді на питання з варіантами

Якщо під час створення опитування користувачем було зазначено, що це опитування анонімне, тобто не потребує автентифікації для респондентів, в такому випадку користувач може скористатися кнопкою “Скопіювати” на сторінці “Мої” для копіювання посилання на опитування в буфер обміну, та надати його респонденту.

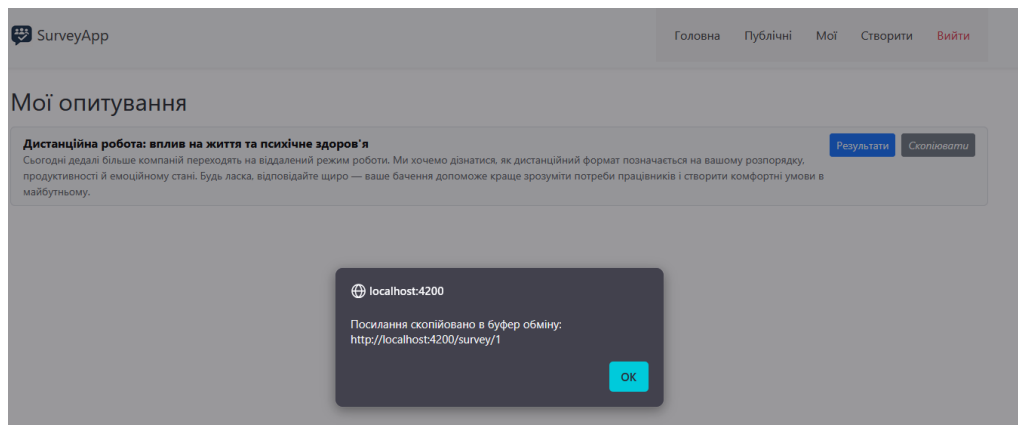


Рисунок 4.27 – Результат натискання кнопки “Скопіювати”

Рисунок 4.28 – Сторінка проходження опитування для неавтентифікованого користувача

В разі проходження опитування анонімом, значення respondent_id буде дорівнювати null, тобто пустому значенню, проте відповіді будуть враховуватися для створення аналітичної діаграми.

	id	question_id	respondent_id	timestamp	text_answer
1	1	1	<null>	1749451944756	dQFr3H6o9ZWh1fQp1QVv0qU3a1FXQLASmdrC/HShvYo=
2	2	2	<null>	1749451944761	<null>
3	3	3	<null>	1749451944862	<null>

Рисунок 4.29 – Результат проходження опитування неавтентифікованим респондентом

ВИСНОВОК ДО РОЗДІЛУ 4

У даному розділі ми детально розглянули можливості розробленого сервісу, та протестували його на правильність роботи, перевірили які запити відправляється з клієнтської частини та які відповіді на них отримуємо з серверної частини. Перевірили правильність збереження даних, редагування та отримання з таблиць бази даних.

Для тестування застосунку було пройдено всі етапи від реєстрації та автентифікації до проходження опитування та отримання результатів у виді діаграми. Під час тестування роботи застосунку проблем не виникло, тому можна враховувати, що розроблений застосунок працює коректно і виконує поставлені задачі.

					ІАЛЦ.467200.003 ПЗ	Арк.
						77
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання дипломного проєкту мною був створений сервіс у вигляді веб-застосунку для створення та проведення соціальних опитувань, котрий надає користувачам можливість створювати опитування з питаннями різних типів відповіді, та встановлює чи буде опитування потребувати автентифікації, або зробивши їх відкритими, таким чином залишив респондентів анонімними.

У першому розділі був здійснен аналіз предметної області, а також проведено порівняння існуючих готових рішень, та визначені їхні плюси та мінуси, в результаті чого було сформоване розуміння того, яким приблизно має бути застосунок.

У другому розділі було приділено увагу огляду існуючих інструментів для розробки серверної та клієнтської частин застосунку. Враховуючи потреби та особисті знання, були обрані необхідні фреймворки, бібліотеки та систему керування базою даних.

У третьому розділі описано реалізацію backend та frontend частин обраними інструментами розробки. В результаті розробки був сформований застосунок, що діє за схемою, коли з клієнтської частини на контролер серверної частини надходить Http запит, котрий викладає метод бізнес логіки застосунку, в якій проходить обробка даних, в результаті якої відбувається редагування даних таблиць бази даних.

У четвертому розділі для гарантування коректності роботи застосунку, було проведено його тестування. Результат перевірки роботи всіх можливих сценарії поведінки користувача на сервісі показав, що веб-застосунок працює без непередбачуваних проблем.

					ІАЛЦ.467200.003 ПЗ	Арк.
						78
Зм.	Арк.	№ докум.	Підпис	Дата		

Розроблений застосунок має зрозумілий для користувача інтерфейс, при цьому надає достатній набір функцій для створення та проведення опитувань, а також забезпечує цілісність та безпеку даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Forms:що це та як створити [Електронний ресурс] – 2024 – Режим доступу до ресурсу: <https://wedex.com.ua/blog/google-forms-shho-cze-ta-yak-stvoryty/>
2. 12 Valuable Microsoft Forms Features You May Not Know [Електронний ресурс] -2024 - Режим доступу до ресурсу: <https://www.gravityunion.com/blog/12-valuable-microsoft-forms-features>
3. Typetform – just another solution on the block [Електронний ресурс] – 2023 - Режим доступу до ресурсу: <https://www.pandadoc.com/blog/typeform-review/>
4. What is SurveyMonkey[Електронний ресурс] – 2023 – Режим доступу до ресурсу: <https://www.genroe.com/blog/what-is-surveymonkey/7979>
5. LimeSurvey: The world`s #1 open source survey platform [Електронний ресурс] - <https://github.com/LimeSurvey/LimeSurvey>
6. Monolithic vs. Microservices Architecture[Електронний ресурс] – 2025 – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/>
7. Microservicies vs. monolithic architecture[Електронний ресурс] – 2022 – Режим доступу до ресурсу: <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>
8. Microservice architecture style[Електронний ресурс] – 2025 – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>

					ІАЛЦ.467200.003 ПЗ	Арк.
						80
Зм.	Арк.	№ докум.	Підпис	Дата		

9. What are Advantages and Disadvantages of Microservices Architecture - [2024] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-microservices-architecture/>
10. Serverless Architecture Overview [Електронний ресурс] – 2021 – Режим доступу до ресурсу: <https://www.datadoghq.com/knowledge-center/serverless-architecture/>
11. Spring Framework (Spring) [Електронний ресурс] – 2024 – Режим доступу до ресурсу: <https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework>
12. What are Java Spring framework advantages and disadvantages [Електронний ресурс] – 2021 – Режим доступу до ресурсу: <https://how.dev/answers/what-are-java-spring-framework-advantages-and-disadvantages>
13. What is Django [Електронний ресурс] – 2021 – Режим доступу до ресурсу: <https://www.ibm.com/think/topics/django>
14. Express.js Tutorial [Електронний ресурс] – 2025 – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/express-js/>
15. What is Angular? A Guide to the Angular Framework [Електронний ресурс] – 2025 – Режим доступу до ресурсу: <https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular>
16. Pros and Cons of Angular Development [Електронний ресурс] – 2023 – Режим доступу до ресурсу: <https://www.agiliway.com/pros-and-cons-of-angular-development/>
17. Key features of React.js: Building Dynamic User Interfaces [Електронний ресурс] – 2023 – Режим доступу до ресурсу:

<https://medium.com/@elightwalk/key-features-of-react-js-building-dynamic-user-interfaces-d572e318ac88>

18.Next.js Key Features and Its Advantages[Електронний ресурс] – 2025 –

Режим доступу до ресурсу: <https://www.xenonstack.com/blog/next.js-features>

19.Top 7 Frontend Frameworks to Use in 2025[Електронний ресурс] – 2025 –

Режим доступу до ресурсу: <https://roadmap.sh/frontend/frameworks>

20.RDBMS Full Form – Relation Database Management Systems[Електронний ресурс] – 2025 – Режим доступу до ресурсу:

<https://www.geeksforgeeks.org/rdbms-full-form/>

21.Why Use MongoDB and When to Use it[Електронний ресурс] – 2020 –

Режим доступу до ресурсу:

<https://www.mongodb.com/resources/products/fundamentals/why-use-mongodb>

22.Redis:What it is, What it does,and Why You Should Care[Електронний

ресурс] – 2022 – Режим доступу до ресурсу: <https://backendless.com/redis-what-it-is-what-it-does-and-why-you-should-care/>

					ІАЛЦ.467200.003 ПЗ	Арк.
						82
Зм.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК 1

Сервіс для створення та проведення соціальних опитувань

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2025 р

ДОДАТОК 2

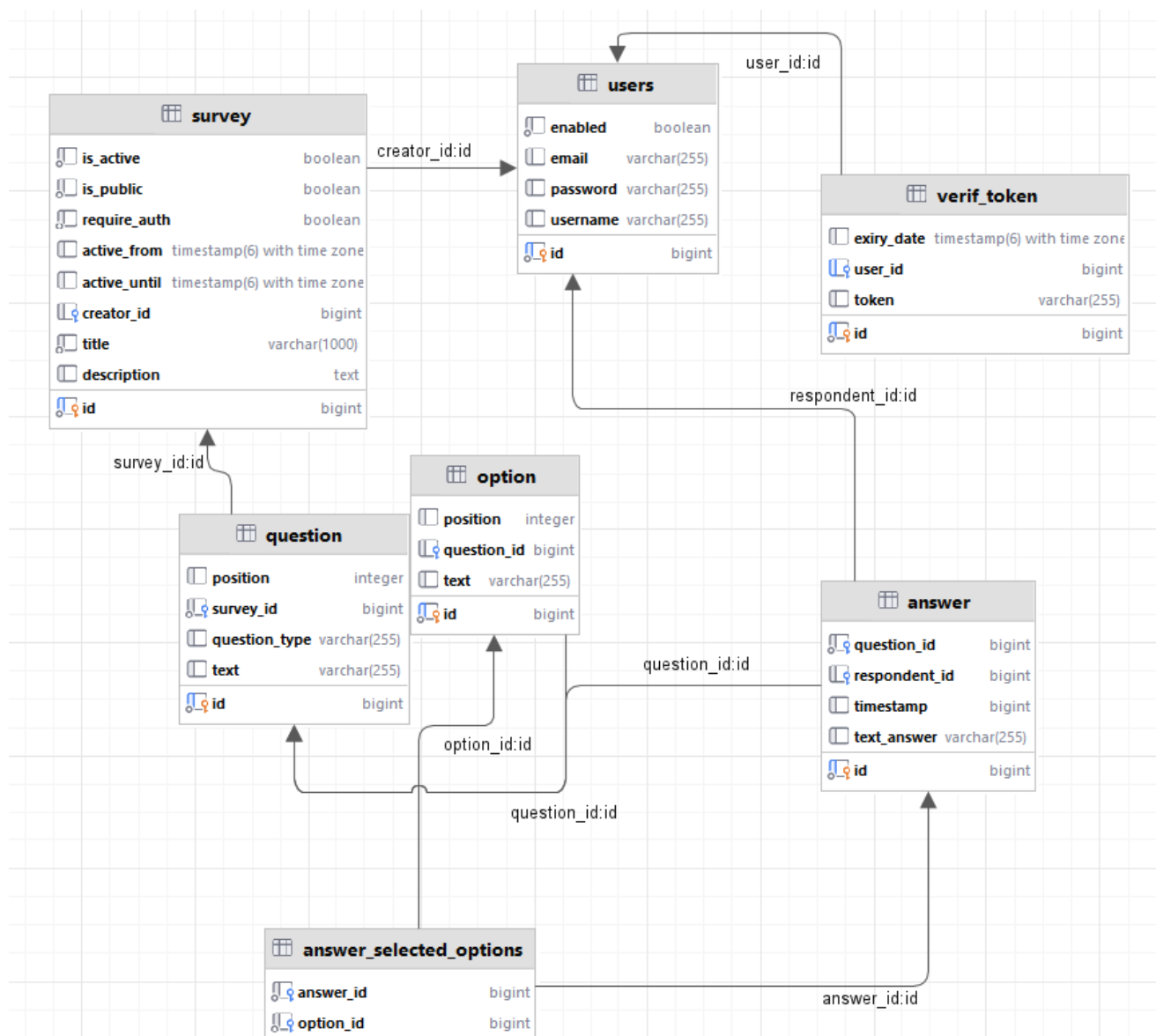
Сервіс для створення та проведення соціальних опитувань

Функціональна схема (діаграма бази даних)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2025 р



					ІАЛЦ.467200.005 Д2		
		№ докум.	Підпис	Дата			
Розробив	Жердев Д. О.				Сервіс для створення та проведення соціальних опитувань Функціональна схема (діаграма бази даних)		
Перевірив	Гордієнко Н. Ю.						
Н. Контр.	Пономаренко А.М.						
Затвердив							
					Літ.	Аркуш	Аркушів
						1	1
					НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15		

ДОДАТОК 3

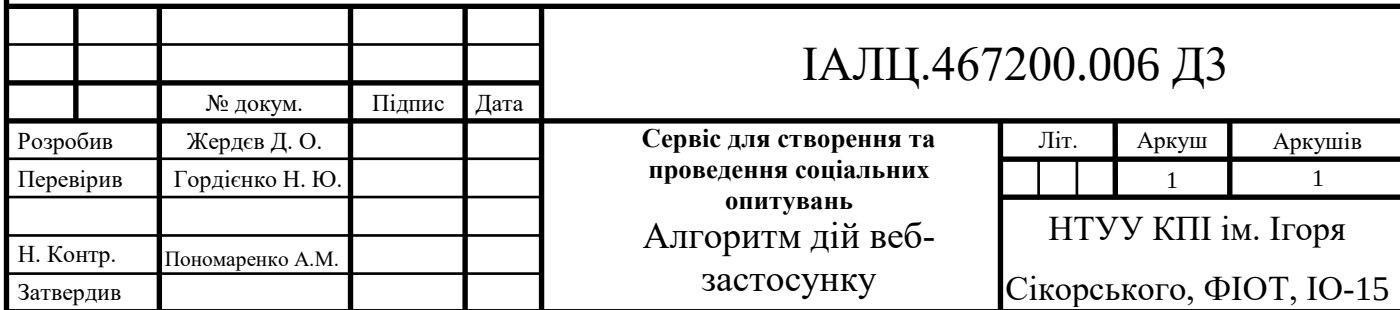
Сервіс для створення та проведення соціальних опитувань

Алгоритм дій веб-застосунку

ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ 2025 р



ДОДАТОК 4

Еволюційні алгоритми глобальної пошукової оптимізації

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 57

Київ 2025 р

Серверна частина

GitHub репозиторій

https://github.com/ZherdevDanil/SurveyApp_Backend

Клієнтська частина

GitHub репозиторій

https://github.com/ZherdevDanil/SurveyApp_Frontend

SurveyApplication.java

```
package com.example.surveyapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SurveyappApplication {

    public static void main(String[] args) {
        SpringApplication.run(SurveyappApplication.class, args);
    }

}
```

User.java

```
package com.example.surveyapp.Entity;

import jakarta.persistence.*;
import lombok.*;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.time.LocalDateTime;
import java.util.Collection;
import java.util.List;

@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Table(name = "users")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

					ІАЛЦ.467200.007 Д4						
		№ докум.	Підпис	Дата							
Розробив	Жердев Д. О.				Сервіс для створення та проведення соціальних опитувань Текс програмного коду			Літ.	Аркуш	Аркушів	
Перевірів	Гордієнко Н. Ю.								1	57	
Н. Контр.	Пономаренко А.М.										
Затвердив											
					НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-15						

```

private String username;
private String email;

private String password;

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return List.of();
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return true;
}
}

```

Survey.java

```

package com.example.surveyapp.Entity;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.Instant;
import java.time.LocalDateTime;

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Survey {

```

					ІАЛЦ.467200.007 Д4	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

```

@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@Column(nullable = false, length = 1000)
private String title;
@Column(columnDefinition = "TEXT")
private String description;
private boolean requireAuth;
private boolean isActive;
private boolean isPublic;

private Instant activeFrom;
private Instant activeUntil;

@ManyToOne
@JoinColumn(name = "creator_id")
private User creator;
}

```

Question.java

```

package com.example.surveyapp.Entity;

import com.fasterxml.jackson.annotation.JsonManagedReference;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;

import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Question {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "survey_id", nullable = false)
    private Survey survey;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

```

private String text;

@Enumerated(EnumType.STRING)
private QuestionType questionType;

private Integer position;

@OneToMany(mappedBy = "question", cascade = CascadeType.ALL, orphanRemoval = true)
@JsonManagedReference
private List<Option> options;
}

```

Option.java

```

package com.example.surveyapp.Entity;

import com.fasterxml.jackson.annotation.JsonBackReference;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Option {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "question_id")
    @JsonBackReference
    private Question question;

    private String text;
    private Integer position;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

Answer.java

```
package com.example.surveyapp.Entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;

import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Answer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    private User respondent;

    @ManyToOne(optional = false)
    private Question question;

    private String textAnswer;

    @ManyToMany
    @JoinTable(
        name = "answer_selected_options",
        joinColumns = @JoinColumn(name = "answer_id"),
        inverseJoinColumns = @JoinColumn(name = "option_id")
    )
    private List<Option> selectedOptions;

    private Long timestamp;
}
```

AuthenticationService.java

```
package com.example.surveyapp.Security;
```

```

import com.example.surveyapp.dto.AuthenticationRequest;
import com.example.surveyapp.dto.AuthenticationResponse;
import com.example.surveyapp.dto.RegisterRequest;
import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Repository.UserRepository;
import com.example.surveyapp.Security.JwtService;
import lombok.RequiredArgsConstructor;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Map;

@Service
@RequiredArgsConstructor
public class AuthenticationService {

    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;
    private final JwtService jwtService;
    private final AuthenticationManager authManager;

    public AuthenticationResponse register(RegisterRequest request) {
        User user = new User();
        user.setUsername(request.getUsername());
        user.setEmail(request.getEmail());
        user.setPassword(passwordEncoder.encode(request.getPassword()));
        userRepository.save(user);

        String jwtToken = jwtService.generateToken(user.getUsername(), Map.of());
        return new AuthenticationResponse(jwtToken);
    }

    public AuthenticationResponse authenticate(AuthenticationRequest request) {
        authManager.authenticate(
            new UsernamePasswordAuthenticationToken(
                request.getUsername(), request.getPassword()
            )
        );

        User user = userRepository.findByUsername(request.getUsername())
            .orElseThrow();
    }

```

					ІАЛЦ.467200.007 Д4	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        String jwtToken = jwtService.generateToken(user.getUsername(), Map.of());
        return new AuthenticationResponse(jwtToken);
    }
}

```

JwtAuthenticationFilter.java

```

package com.example.surveyapp.Security;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtService jwtService;
    private final UserDetailsService userDetailsService;

    public JwtAuthenticationFilter(JwtService jwtService, UserDetailsService
userDetailsService) {
        this.jwtService = jwtService;
        this.userDetailsService = userDetailsService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException,
IOException {

        final String authHeader = request.getHeader("Authorization");
        final String jwt;
        final String username;

```

					ІАЛЦ.467200.007 Д4	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return;
        }

        jwt = authHeader.substring(7);
        username = jwtService.extractUsername(jwt);

        if (username != null && SecurityContextHolder.getContext().getAuthentication() ==
null) {
            var userDetails = userDetailsService.loadUserByUsername(username);

            if (jwtService.isTokenValid(jwt, userDetails.getUsername())) {
                var authToken = new UsernamePasswordAuthenticationToken(
                    userDetails, null, userDetails.getAuthorities()
                );
                authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
                SecurityContextHolder.getContext().setAuthentication(authToken);
            }
        }

        filterChain.doFilter(request, response);
    }
}

```

JwtService.java

```

package com.example.surveyapp.Security;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.Jws;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import jakarta.annotation.PostConstruct;
import javax.crypto.SecretKey;
import java.util.Date;
import java.util.Map;

@Service
public class JwtService {

```

					ІАЛЦ.467200.007 Д4	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

```

@Value("${jwt.secret}")
private String jwtSecret;

@Value("${jwt.lifetime}")
private long jwtExpiration;

private SecretKey secretKey;

@PostConstruct
public void init() {
    this.secretKey = Keys.hmacShaKeyFor(jwtSecret.getBytes());
}

public String generateToken(String username, Map<String, Object> extraClaims) {
    return Jwts.builder()
        .subject(username)
        .claims(extraClaims)
        .issuedAt(new Date())
        .expiration(new Date(System.currentTimeMillis() + jwtExpiration))
        .signWith(secretKey)
        .compact();
}

public String extractUsername(String token) {
    return Jwts.parser()
        .verifyWith(secretKey)
        .build()
        .parseSignedClaims(token)
        .getPayload()
        .getSubject();
}

public boolean isTokenValid(String token, String expectedUsername) {
    try {
        String actualUsername = extractUsername(token);
        return actualUsername.equals(expectedUsername) && !isTokenExpired(token);
    } catch (Exception e) {
        return false;
    }
}

public boolean isTokenExpired(String token) {

```

					ІАЛЦ.467200.007 Д4	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        Date expiration = Jwts.parser()
            .verifyWith(secretKey)
            .build()
            .parseSignedClaims(token)
            .getPayload()
            .getExpiration();
        return expiration.before(new Date());
    }
}

```

SecurityConfig.java

```

package com.example.surveyapp.Security;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.Customizer;
import
org.springframework.security.config.annotation.authentication.configuration.AuthenticationCo
nfiguration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
public class SecurityConfig {

    private final JwtAuthenticationFilter jwtAuthFilter;
    private final UserDetailsService userDetailsService;

    public SecurityConfig(JwtAuthenticationFilter jwtAuthFilter, UserDetailsService
userDetailsService) {
        this.jwtAuthFilter = jwtAuthFilter;
        this.userDetailsService = userDetailsService;
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

```

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    return http
        .csrf(csrf -> csrf.disable())
        .cors(Customizer.withDefaults())
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/api/auth/**").permitAll()
            .requestMatchers(HttpMethod.GET, "/api/surveys/{id}").permitAll()
            .requestMatchers("/api/submit").permitAll()
            .anyRequest().authenticated()
        )
        .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authenticationProvider(authenticationProvider())
        .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class)
        .build();
}

@Bean
public AuthenticationProvider authenticationProvider() {
    var authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userDetailsService);
    authProvider.setPasswordEncoder(passwordEncoder());
    return authProvider;
}

@Bean
public AuthenticationManager authenticationManager(AuthenticationConfiguration config)
throws Exception {
    return config.getAuthenticationManager();
}

@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
}

```

UserDetailsServiceImpl.java

```

package com.example.surveyapp.Security;

import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Repository.UserRepository;

```

					ІАЛЦ.467200.007 Д4	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    private final UserRepository userRepository;

    public UserDetailsServiceImpl(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return userRepository.findByUsername(username)
            .orElseThrow(() -> new UsernameNotFoundException("User not found"));
    }
}

```

AnalyticsService.java

```

package com.example.surveyapp.Service;

import com.example.surveyapp.Entity.*;
import com.example.surveyapp.Repository.AnswerRepository;
import com.example.surveyapp.Repository.QuestionRepository;
import com.example.surveyapp.Repository.SurveyRepository;
import com.example.surveyapp.dto.OptionStats;
import com.example.surveyapp.dto.QuestionResultResponse;
import com.example.surveyapp.dto.SurveyResultsResponse;
import jakarta.persistence.EntityNotFoundException;
import org.springframework.stereotype.Service;

import java.nio.file.AccessDeniedException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.stream.Collectors;

```

					ІАЛЦ.467200.007 Д4	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

```

@Service
public class AnalyticsService {

    private final SurveyRepository surveyRepository;
    private final QuestionRepository questionRepository;
    private final AnswerRepository answerRepository;

    public AnalyticsService(SurveyRepository surveyRepository, QuestionRepository
questionRepository, AnswerRepository answerRepository) {
        this.surveyRepository = surveyRepository;
        this.questionRepository = questionRepository;
        this.answerRepository = answerRepository;
    }

    public SurveyResultsResponse getResultsForSurvey(Long surveyId,String username) throws
AccessDeniedException {
        Survey survey = surveyRepository.findById(surveyId).orElseThrow(()-> new
EntityNotFoundException("Survey not found"));

        if (!survey.getCreator().getUsername().equals(username)){
            throw new AccessDeniedException("Not allowed to view analytics of this survey");
        }

        List<Question> questions =
questionRepository.findByIdOrderByPositionAsc(surveyId);
        List<Answer> allAnswers = answerRepository.findById(surveyId);

        SurveyResultsResponse response = new SurveyResultsResponse();
        response.setSurveyId(survey.getId());
        response.setTitle(survey.getTitle());

        List<QuestionResultResponse> questionResultResponses = new ArrayList<>();
        for (Question question: questions){
            QuestionResultResponse questionResultResponse = new QuestionResultResponse();
            questionResultResponse.setQuestionId(question.getId());
            questionResultResponse.setText(questionResultResponse.getText());
            questionResultResponse.setQuestionType(question.getQuestionType());

            List<Answer> questionAnswers = allAnswers.stream()
                .filter(ans -> ans.getQuestion().getId().equals(question.getId()))
                .toList();

            if (question.getQuestionType() == QuestionType.TEXT) {

```

					ІАЛЦ.467200.007 Д4	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        List<String> texts = questionAnswers.stream()
            .map(Answer::getTextAnswer)
            .filter(Objects::nonNull)
            .toList();
        questionResultResponse.setTextAnswers(texts);

    } else {
        Map<Option, Long> optionCounts = questionAnswers.stream()
            .flatMap(ans -> ans.getSelectedOptions().stream())
            .collect(Collectors.groupingBy(opt -> opt, Collectors.counting()));

        List<OptionStats> stats = optionCounts.entrySet().stream()
            .map(e -> {
                OptionStats os = new OptionStats();
                os.setOptionId(e.getKey().getId());
                os.setText(e.getKey().getText());
                os.setCount(e.getValue());
                return os;
            })
            .toList();

        questionResultResponse.setOptionStats(stats);
    }

    questionResultResponses.add(questionResultResponse);
}

response.setQuestionResultResponses(questionResultResponses);
return response;
}
}

```

AnswerService.java

```

package com.example.surveyapp.Service;

import com.example.surveyapp.Entity.Answer;
import com.example.surveyapp.Entity.Option;
import com.example.surveyapp.Entity.Question;
import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Repository.AnswerRepository;
import com.example.surveyapp.Repository.OptionRepository;
import com.example.surveyapp.Repository.QuestionRepository;
import com.example.surveyapp.Repository.UserRepository;

```

					ІАЛЦ.467200.007 Д4	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import com.example.surveyapp.dto.AnswerRequest;
import jakarta.persistence.EntityNotFoundException;
import org.springframework.stereotype.Service;

import java.nio.file.AccessDeniedException;
import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

@Service
public class AnswerService {

    public final AnswerRepository answerRepository;
    public final UserRepository userRepository;

    private final QuestionRepository questionRepository;
    private final OptionRepository optionRepository;

    public AnswerService(AnswerRepository answerRepository, UserRepository userRepository,
QuestionRepository questionRepository, OptionRepository optionRepository) {
        this.answerRepository = answerRepository;
        this.userRepository = userRepository;
        this.questionRepository = questionRepository;
        this.optionRepository = optionRepository;
    }

    public Answer submitAnswer( AnswerRequest answerRequest, String username) throws
AccessDeniedException {
        Question question = questionRepository.findById(answerRequest.getQuestionId()).orElseThrow(()->new
EntityNotFoundException("Question not found"));

        User user = new User();

        if (username != null){
            user = userRepository.findByUsername(username).orElseThrow(()->new
EntityNotFoundException("User not found"));
        }

        if (user != null &&
answerRepository.findByRespondentAndQuestion(user,question).isPresent()){
            throw new AccessDeniedException("You already answered to this question");
        }
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    }

    Answer answer = Answer.builder()
        .question(question)
        .respondent(user)
        .timestamp(Instant.now().toEpochMilli())
        .build();

    switch (question.getQuestionType()){
        case TEXT -> {
            answer.setTextAnswer(answerRequest.getTextAnswer());
        }
        case SINGLE_CHOICE, MULTIPLE_CHOICE -> {
            if (answerRequest.getSelectedOptionId() == null ||
answerRequest.getSelectedOptionId().isEmpty()){
                throw new IllegalArgumentException("Options must be provided");
            }
            List<Option> selectedOptions =
optionRepository.findAllById(answerRequest.getSelectedOptionId());
            answer.setSelectedOptions(selectedOptions);
        }
    }
    return answerRepository.save(answer);
}

public List<Answer> submitAnswers(List<AnswerRequest> answerRequests, String username)
throws AccessDeniedException {
    List<Answer> answers = new ArrayList<>();

    for (AnswerRequest answerRequest : answerRequests){
        answers.add(submitAnswer(answerRequest,username));
    }
    return answers;
}

public boolean hasUserSubmitted(Long surveyId, String username) {
    Question question = questionRepository.findBySurveyId(surveyId).get(0);
    if (username != null) {
        return
answerRepository.existsByQuestionIdAndRespondent_Username(question.getId(), username);
    } else {
        return false;
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						16
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    }
}
}

```

OptionService.java

```

package com.example.surveyapp.Service;

import com.example.surveyapp.Entity.Option;
import com.example.surveyapp.Entity.Question;
import com.example.surveyapp.Entity.QuestionType;
import com.example.surveyapp.Repository.OptionRepository;
import com.example.surveyapp.Repository.QuestionRepository;
import com.example.surveyapp.dto.CreateOptionRequest;
import jakarta.persistence.EntityNotFoundException;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class OptionService {

    private final OptionRepository optionRepository;
    private final QuestionRepository questionRepository;
    private final UserService userService;

    public OptionService(OptionRepository optionRepository, QuestionService questionService,
QuestionRepository questionRepository, UserService userService) {
        this.optionRepository = optionRepository;
        this.questionRepository = questionRepository;
        this.userService = userService;
    }

    public Option addOption(Long questionId, CreateOptionRequest createOptionRequest){
        Question question = questionRepository.findById(questionId)
            .orElseThrow(() -> new EntityNotFoundException("Question not found"));

        if (question.getQuestionType() == QuestionType.TEXT) {
            throw new IllegalArgumentException("Cannot add options to text-type questions");
        }
    }

```

					ІАЛЦ.467200.007 Д4	Арк.
						17
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        Option option = Option.builder()
            .question(question)
            .text(createOptionRequest.getText())
            .position(createOptionRequest.getPosition())
            .build();

        return optionRepository.save(option);
    }

    public List<Option> getOptionsForQuestion(Long questionId) {
        if (!questionRepository.existsById(questionId)) {
            throw new EntityNotFoundException("Question not found");
        }

        return optionRepository.findByIdOrderByPositionAsc(questionId);
    }

    public Option updateOption(Long optionId, CreateOptionRequest createOptionRequest) {
        Option option = optionRepository.findById(optionId)
            .orElseThrow(() -> new EntityNotFoundException("Option not found"));

        option.setText(createOptionRequest.getText());
        option.setPosition(createOptionRequest.getPosition());

        return optionRepository.save(option);
    }

    public void deleteOption(Long optionId) {
        Option option = optionRepository.findById(optionId)
            .orElseThrow(() -> new EntityNotFoundException("Option not found"));

        optionRepository.delete(option);
    }
}

```

QuestionService.java

```

package com.example.surveyapp.Service;

import com.example.surveyapp.Entity.Option;
import com.example.surveyapp.Entity.Question;
import com.example.surveyapp.Entity.QuestionType;
import com.example.surveyapp.Entity.Survey;
import com.example.surveyapp.Repository.AnswerRepository;

```

					ІАЛЦ.467200.007 Д4	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import com.example.surveyapp.Repository.OptionRepository;
import com.example.surveyapp.Repository.QuestionRepository;
import com.example.surveyapp.Repository.SurveyRepository;
import com.example.surveyapp.dto.CreateOptionRequest;
import com.example.surveyapp.dto.QuestionUpdateDto;
import jakarta.persistence.EntityNotFoundException;
import jakarta.transaction.Transactional;
import org.springframework.stereotype.Service;

import java.nio.file.AccessDeniedException;
import java.util.List;

@Service
public class QuestionService {

    private final QuestionRepository questionRepository;
    private final OptionRepository optionRepository;
    private final SurveyRepository surveyRepository;

    private final UserService userService;

    private final AnswerRepository answerRepository;

    public QuestionService(QuestionRepository questionRepository, OptionRepository
optionRepository, SurveyRepository surveyRepository, UserService userService,
AnswerRepository answerRepository) {
        this.questionRepository = questionRepository;
        this.optionRepository = optionRepository;
        this.surveyRepository = surveyRepository;
        this.userService = userService;
        this.answerRepository = answerRepository;
    }

    public Question addQuestionToSurvey(Long surveyId, Question question){
        Survey survey = surveyRepository.findById(surveyId).orElseThrow(()->new
RuntimeException("Survey not found"));
        question.setSurvey(survey);
        return questionRepository.save(question);
    }

    public Question createQuestion(Long surveyId, QuestionUpdateDto request, String username)
throws AccessDeniedException {

```

					ІАЛЦ.467200.007 Д4	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

```

Survey survey = surveyRepository.findById(surveyId)
    .orElseThrow(() -> new EntityNotFoundException("Survey not found"));

if (!survey.getCreator().getUsername().equals(username)) {
    throw new AccessDeniedException("You are not owner of this survey");
}

Question question = Question.builder()
    .survey(survey)
    .text(request.getText())
    .questionType(request.getType())
    .position(request.getPosition())
    .build();

Question savedQuestion = questionRepository.save(question);

if (request.getType() == QuestionType.SINGLE_CHOICE || request.getType() ==
QuestionType.MULTIPLE_CHOICE) {
    List<Option> options = request.getOptions().stream()
        .map(o -> Option.builder()
            .question(savedQuestion)
            .text(o.getText())
            .position(o.getPosition())
            .build())
        .toList();

    optionRepository.saveAll(options);
    savedQuestion.setOptions(options);
}

return savedQuestion;
}

public List<Question> getQuestionsBySurvey(Long surveyId,String username) throws
AccessDeniedException {
    Survey survey = surveyRepository.findById(surveyId).orElseThrow(()->new
EntityNotFoundException("Survey not found"));

    if (survey.isRequireAuth()){
        if (!survey.getCreator().getUsername().equals(username)){

```

					ІАЛЦ.467200.007 Д4	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        throw new AccessDeniedException("You are not allowed to view questions");
    }
}
return questionRepository.findBySurveyIdOrderByPositionAsc(surveyId);
}

public void deleteQuestion(Long questionId,String username) throws AccessDeniedException
{
    Question question = questionRepository.findById(questionId).orElseThrow(()->new
EntityNotFoundException("Question not found"));

    if (!question.getSurvey().getCreator().getUsername().equals(username)){
        throw new AccessDeniedException("You are not allowed to delete this question");
    }
    questionRepository.delete(question);
}

@Transactional
public Question updateQuestion(Long questionId, QuestionUpdateDto questionUpdateDto,
String username) throws AccessDeniedException {
    boolean hasAnswers = answerRepository.existsByQuestionId(questionId);
    if (hasAnswers){
        throw new IllegalArgumentException("This question already have answers, edit is
not allowed ");
    }
    Question question = questionRepository.findById(questionId)
        .orElseThrow(() -> new EntityNotFoundException("Question not found"));

    if (!question.getSurvey().getCreator().getUsername().equals(username)) {
        throw new AccessDeniedException("You are not allowed to change question");
    }
    question.setText(questionUpdateDto.getText());
    question.setQuestionType(questionUpdateDto.getType());
    question.setPosition(questionUpdateDto.getPosition());

    if      (questionUpdateDto.getType()      ==      QuestionType.SINGLE_CHOICE      ||
questionUpdateDto.getType() == QuestionType.MULTIPLE_CHOICE) {
        question.getOptions().clear();

        int index = 0;
        for ( CreateOptionRequest createOptionRequest : questionUpdateDto.getOptions()) {
            Option option = Option.builder()

```

					ІАЛЦ.467200.007 Д4	Арк.
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        .question(question)
        .text(createOptionRequest.getText())
        .position(index++)
        .build();
    question.getOptions().add(option);
}

} else {

    question.getOptions().clear();
}

return questionRepository.save(question);
}
}

```

SurveyService.java

```

package com.example.surveyapp.Service;

import com.example.surveyapp.Entity.*;
import com.example.surveyapp.Repository.*;
import com.example.surveyapp.Security.JwtService;
import com.example.surveyapp.dto.*;
import jakarta.persistence.EntityNotFoundException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.transaction.Transactional;
import org.springframework.stereotype.Service;

import java.nio.file.AccessDeniedException;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class SurveyService {
    public final SurveyRepository surveyRepository;
    public final UserRepository userRepository;

    public final JwtService jwtService;

    private final QuestionRepository questionRepository;

```

					ІАЛЦ.467200.007 Д4	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

```

private final OptionRepository optionRepository;
private final AnswerRepository answerRepository;

public SurveyService(SurveyRepository surveyRepository, UserRepository userRepository,
JwtService jwtService, QuestionRepository questionRepository, OptionRepository
optionRepository, AnswerRepository answerRepository){
    this.surveyRepository=surveyRepository;
    this.userRepository = userRepository;
    this.jwtService = jwtService;
    this.questionRepository = questionRepository;
    this.optionRepository = optionRepository;
    this.answerRepository = answerRepository;
}

public List<Survey> getSurveys(Long id){
    List<Survey> surveys = surveyRepository.findByCreatorId(id);
    return surveys;
}

public Survey createSurvey(CreateSurveyRequest createSurveyRequest, String username) {
    User creator = userRepository.findByUsername(username)
        .orElseThrow(() -> new EntityNotFoundException("User not found"));
    System.out.println("Чи createSurveyRequest - public" +
createSurveyRequest.isPublic());

    Survey survey = Survey.builder()
        .title(createSurveyRequest.getTitle())
        .description(createSurveyRequest.getDescription())
        .requireAuth(createSurveyRequest.isRequireAuth())
        .activeFrom(createSurveyRequest.getActiveFrom())
        .activeUntil(createSurveyRequest.getActiveUntil())
        .creator(creator)
        .isActive(true)
        .isPublic(createSurveyRequest.isPublic())
        .build();
    System.out.println("Чи saved survey - public"+survey.isPublic());
    survey = surveyRepository.save(survey);

    for (QuestionUpdateDto qRequest : createSurveyRequest.getQuestions()) {
        Question question = Question.builder()
            .survey(survey)
            .text(qRequest.getText())

```

					ІАЛЦ.467200.007 Д4	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        .questionType(qRequest.getType())
        .position(qRequest.getPosition())
        .build();

    question = questionRepository.save(question);

    if (qRequest.getType() == QuestionType.SINGLE_CHOICE || qRequest.getType() ==
QuestionType.MULTIPLE_CHOICE) {
        List<Option> options = new ArrayList<>();
        for (CreateOptionRequest opt : qRequest.getOptions()) {
            options.add(Option.builder()
                .question(question)
                .text(opt.getText())
                .position(opt.getPosition())
                .build());
        }
        optionRepository.saveAll(options);
        question.setOptions(options);
    }
}

return survey;
}

public List<PublicSurveysResponse> getAllPublicSurveys(){
    List<Survey> allSurveys = surveyRepository.findAll();
    List<PublicSurveysResponse> publicSurveys = new ArrayList<>();

    for (Survey survey : allSurveys){
        if (survey.isPublic()){
            PublicSurveysResponse publicSurveysResponse = PublicSurveysResponse.builder()
                .id(survey.getId())
                .title(survey.getTitle())
                .description(survey.getDescription())
                .build();
            publicSurveys.add(publicSurveysResponse);
        }
    }
    return publicSurveys;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

```

public UpdateSurveyRequest getSurveyDetails(Long id) {
    Survey survey = surveyRepository.findById(id)
        .orElseThrow(() -> new EntityNotFoundException("Survey not found"));

    List<Question> questions = questionRepository.findBySurveyId(survey.getId());

    List<QuestionUpdateDto> questionDtos = questions.stream()
        .map(q -> {
            List<CreateOptionRequest> optionDtos = q.getOptions().stream()
                .map(opt -> CreateOptionRequest.builder()
                    .id(opt.getId())
                    .text(opt.getText())
                    .position(opt.getPosition())
                    .build())
                .toList();

            return QuestionUpdateDto.builder()
                .id(q.getId())
                .text(q.getText())
                .type(q.getQuestionType())
                .position(q.getPosition())
                .canEdit(!answerRepository.existsByQuestionId(q.getId()))
                .options(optionDtos)
                .build();
        })
        .toList();

    return UpdateSurveyRequest.builder()
        .id(survey.getId())
        .title(survey.getTitle())
        .description(survey.getDescription())
        .requireAuth(survey.isRequireAuth())
        .isActive(survey.isActive())
        .questions(questionDtos)
        .build();
}

```

```

@Transactional
public Survey finishSurvey(Long surveyId, String username) throws AccessDeniedException {
    Survey survey = surveyRepository.findById(surveyId).orElseThrow(() -> new
EntityNotFoundException());
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        if (!survey.getCreator().getUsername().equals(username)){
            throw new AccessDeniedException("You are not allowed");
        }
        survey.setActive(false);
        return surveyRepository.save(survey);
    }

    public Optional<Survey> findById(Long id){
        return surveyRepository.findById(id);
    }

    public List<Survey> findByUser(User user){
        List<Survey> surveys = surveyRepository.findByCreatorId(user.getId());
        return surveys;
    }

    public void deleteSurvey(Long id){
        surveyRepository.deleteById(id);
    }

    public Survey save(Survey survey){
        return surveyRepository.save(survey);
    }

    public String extractUsername(HttpServletRequest request){
        final String authHeader = request.getHeader("Authorization");
        if(authHeader == null || !authHeader.startsWith("Bearer ")){
            return null;
        }
        String jwt = authHeader.substring(7);
        return jwtService.extractUsername(jwt);
    }
}

```

UserService.java

```

package com.example.surveyapp.Service;

import com.example.surveyapp.Entity.Option;
import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Repository.UserRepository;

```

					ІАЛЦ.467200.007 Д4	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {
    public final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public User register(User user){
        return userRepository.save(user);
    }

    public Optional<User> findById(Long id){
        return userRepository.findById(id);
    }

    public Optional<User> findByUsername(String username){
        return userRepository.findByUsername(username);
    }
}

```

CorsConfig.java

```

package com.example.surveyapp.Config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;

import java.util.List;

@Configuration
public class CorsConfig {
    @Bean
    public CorsFilter corsFilter(){
        CorsConfiguration corsConfiguration = new CorsConfiguration();

```

					ІАЛЦ.467200.007 Д4	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        corsConfiguration.setAllowedOrigins(List.of("http://localhost:4200"));
        corsConfiguration.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE"));
        corsConfiguration.setAllowedHeaders(List.of("Authorization", "Content-type"));
        corsConfiguration.setAllowCredentials(true);

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/*", corsConfiguration);
        return new CorsFilter(source);

    }

}

```

AnalyticsController.java

```

package com.example.surveyapp.Controller;

import com.example.surveyapp.Entity.Survey;
import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Service.AnalyticsService;
import com.example.surveyapp.Service.SurveyService;
import com.example.surveyapp.dto.SurveyResultsResponse;
import jakarta.persistence.EntityNotFoundException;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.nio.file.AccessDeniedException;
import java.util.Optional;

@RestController
@RequestMapping("/api/analytics")
public class AnalyticsController {
    private final AnalyticsService analyticsService;

    private final SurveyService surveyService;

    public AnalyticsController(AnalyticsService analyticsService, SurveyService
surveyService) {
        this.analyticsService = analyticsService;
    }

```

					ІАЛЦ.467200.007 Д4	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        this.surveyService = surveyService;
    }
    @GetMapping("/{surveyId}/results")
    public ResponseEntity<SurveyResultsResponse> getSurveyResults(@PathVariable Long
surveyId, HttpServletRequest httpRequest) throws AccessDeniedException {
        Survey survey = surveyService.findById(surveyId).orElseThrow(()->new
EntityNotFoundException("Survey not found"));
        String username = surveyService.extractUsername(httpServletRequest);
        if (!survey.getCreator().getUsername().equals(username)){
            throw new AccessDeniedException("You are not allowed to see result survey
results");
        }
        SurveyResultsResponse response =
analyticsService.getResultsForSurvey(surveyId,username);
        return ResponseEntity.ok(response);
    }
}

```

AnswerController.java

```

package com.example.surveyapp.Controller;

import com.example.surveyapp.Entity.Answer;
import com.example.surveyapp.Security.JwtService;
import com.example.surveyapp.Service.AnswerService;
import com.example.surveyapp.Service.SurveyService;
import com.example.surveyapp.dto.AnswerRequest;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.nio.file.AccessDeniedException;
import java.util.List;

@RestController
@RequestMapping("/api/answers")
public class AnswerController {
    private final AnswerService answerService;
    private final JwtService jwtService;

    private final SurveyService surveyService;

```

					ІАЛЦ.467200.007 Д4	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    public AnswerController(AnswerService answerService, JwtService jwtService, SurveyService
surveyService) {
        this.answerService = answerService;
        this.jwtService = jwtService;
        this.surveyService = surveyService;
    }
    @PostMapping
    public ResponseEntity<?> submitAnswer(@RequestBody List<AnswerRequest> answerRequests,
HttpServletRequest httpServletRequest) throws AccessDeniedException {
        String username = null;
        if (httpServletRequest.getHeader("Authorization")!=null &&
httpServletRequest.getHeader("Authorization").startsWith("Bearer ")){
            username = surveyService.extractUsername(httpServletRequest);
        }
        List<Answer> answer = answerService.submitAnswers(answerRequests,username);
        return ResponseEntity.ok(answer);
    }

    @GetMapping("/check")
    public ResponseEntity<Boolean> hasUserSubmitted(@RequestParam Long
surveyId,HttpServletRequest httpServletRequest){
        String username = null;
        if (httpServletRequest.getHeader("Authorization")!=null &&
httpServletRequest.getHeader("Authorization").startsWith("Bearer ")){
            username = surveyService.extractUsername(httpServletRequest);
        }
        boolean isAlreadySubmitted = answerService.hasUserSubmitted(surveyId,username);
        return ResponseEntity.ok(isAlreadySubmitted);
    }
}

```

AuthController.java

```

package com.example.surveyapp.Controller;

import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Security.AuthenticationService;
import com.example.surveyapp.Service.UserService;
import com.example.surveyapp.dto.AuthenticationRequest;
import com.example.surveyapp.dto.AuthenticationResponse;
import com.example.surveyapp.dto.RegisterRequest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;

```

					ІАЛЦ.467200.007 Д4	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import static org.springframework.http.ResponseEntity.ok;

@RestController
@RequestMapping("/api/auth")
public class AuthController {
    private final AuthenticationService authenticationService;

    public AuthController( AuthenticationService authenticationService) {
        this.authenticationService = authenticationService;
    }

    @PostMapping("/register")
    public ResponseEntity<AuthenticationResponse> register(@RequestBody RegisterRequest request) {
        return ResponseEntity.ok(authenticationService.register(request));
    }

    @PostMapping("/login")
    public ResponseEntity<AuthenticationResponse> login(@RequestBody AuthenticationRequest request) {
        return ResponseEntity.ok(authenticationService.authenticate(request));
    }
}

```

QuestionController.java

```

package com.example.surveyapp.Controller;

import com.example.surveyapp.Entity.Question;
import com.example.surveyapp.Entity.Survey;
import com.example.surveyapp.Service.QuestionService;
import com.example.surveyapp.Service.SurveyService;
import com.example.surveyapp.dto.CreateQuestionRequest;
import com.example.surveyapp.dto.QuestionUpdateDto;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.nio.file.AccessDeniedException;
import java.util.List;

```

					ІАЛЦ.467200.007 Д4	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import java.util.Map;

@RestController
@RequestMapping("/api")
public class QuestionController {
    private final QuestionService questionService;

    private final SurveyService surveyService
        ;

    public QuestionController(QuestionService questionService, SurveyService surveyService) {
        this.questionService = questionService;
        this.surveyService = surveyService;
    }

    @PostMapping("/surveys/{surveyId}/questions")
    public ResponseEntity<Question> createQuestion(@PathVariable Long surveyId, @RequestBody
    QuestionUpdateDto questionUpdateDto/*CreateQuestionRequest*/, HttpServletRequest
    httpServletRequest) throws AccessDeniedException {
        String username = surveyService.extractUsername(httpServletRequest);
        Question created =
    questionService.createQuestion(surveyId,questionUpdateDto,username);
        return ResponseEntity.ok(created);
    }

    @GetMapping("/surveys/{surveyId}/questions")
    public ResponseEntity<List<Question>> getQuestionsBySurvey(@PathVariable Long
    surveyId,HttpServletRequest httpServletRequest) throws AccessDeniedException {
        String username = surveyService.extractUsername(httpServletRequest);
        List<Question> questions = questionService.getQuestionsBySurvey(surveyId,username);
        return ResponseEntity.ok(questions);
    }

    @PutMapping("/questions/{questionId}")
    public ResponseEntity<?> updateQuestion(@PathVariable Long questionId, @RequestBody
    QuestionUpdateDto questionUpdateDto/*CreateQuestionRequest createQuestionRequest*/,
    HttpServletRequest httpServletRequest) throws AccessDeniedException {
        try{
            String username = surveyService.extractUsername(httpServletRequest);
            Question updatedQuestion =
    questionService.updateQuestion(questionId,questionUpdateDto,username);
            return ResponseEntity.ok(updatedQuestion);}
        catch (IllegalArgumentException ex){

```

					ІАЛЦ.467200.007 Д4	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        return
        ResponseEntity.status(HttpStatus.BAD_REQUEST).body(Map.of("error", ex.getMessage()));
    }
}

@DeleteMapping("/questions/{questionId}")
public      ResponseEntity<Void>      deleteQuestion(@PathVariable      Long
questionId,HttpServletRequest httpServletRequest) throws AccessDeniedException {
    String username = surveyService.extractUsername(httpServletRequest);
    questionService.deleteQuestion(questionId,username);
    return new ResponseEntity<>(HttpStatus.NO_CONTENT);
}
}

```

SurveyController.java

```

package com.example.surveyapp.Controller;

import com.example.surveyapp.Entity.Survey;
import com.example.surveyapp.Entity.User;
import com.example.surveyapp.Repository.UserRepository;
import com.example.surveyapp.Security.JwtService;
import com.example.surveyapp.Service.SurveyService;
import com.example.surveyapp.dto.CreateSurveyRequest;
import com.example.surveyapp.dto.PublicSurveysResponse;
import com.example.surveyapp.dto.UpdateSurveyRequest;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.nio.channels.AcceptPendingException;
import java.nio.file.AccessDeniedException;
import java.time.Instant;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/surveys")
public class SurveyController {
    private final SurveyService surveyService;
    private final UserRepository userRepository;
    private final JwtService jwtService;

```

					ІАЛЦ.467200.007 Д4	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    public SurveyController(SurveyService surveyService, UserRepository userRepository,
JwtService jwtService) {
        this.surveyService = surveyService;
        this.userRepository = userRepository;
        this.jwtService = jwtService;
    }

    @PostMapping
    public ResponseEntity<Survey> createSurvey(@RequestBody CreateSurveyRequest
createSurveyRequest, HttpServletRequest httpServletRequest) {
        String username = surveyService.extractUsername(httpServletRequest);
        if (username == null ) return ResponseEntity.status(401).build();
        Survey createdSurvey = surveyService.createSurvey(createSurveyRequest,username);
        return ResponseEntity.ok(createdSurvey);
    }

    @GetMapping("/public")
    public ResponseEntity<List<PublicSurveysResponse>> getAllPublicSurveys(){
        List<PublicSurveysResponse> publicSurveysResponse =
surveyService.getAllPublicSurveys();
        return ResponseEntity.ok(publicSurveysResponse);
    }

    @GetMapping("/{id}")
    public ResponseEntity<UpdateSurveyRequest> getSurveyById(@PathVariable Long id) throws
AccessDeniedException {
        Survey survey = surveyService.findById(id).get();
        /*if(!survey.isActive() || (survey.getActiveFrom() != null &&
Instant.now().isBefore(survey.getActiveFrom()))
        || (survey.getActiveUntil()!=null &&
Instant.now().isAfter(survey.getActiveUntil()))){
            throw new AccessDeniedException("Опитування не активне");
        }*/

        return ResponseEntity.ok(surveyService.getSurveyDetails(id));
    }

    @GetMapping("/survey-details/{id}")
    public ResponseEntity<UpdateSurveyRequest> getSurveyDetailsById(@PathVariable Long id)
throws AccessDeniedException {
        return ResponseEntity.ok(surveyService.getSurveyDetails(id));
    }

```

					ІАЛЦ.467200.007 Д4	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    @GetMapping
    public ResponseEntity<List<Survey>> getAllUserSurveys(HttpServletRequest httpServletRequest) {
        String username = surveyService.extractUsername(httpServletRequest);
        if (username == null) return ResponseEntity.status(401).build();

        User user = userRepository.findByUsername(username).orElseThrow();
        List<Survey> surveys = surveyService.findByUser(user);
        return ResponseEntity.ok(surveys);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteSurvey(@PathVariable Long id, HttpServletRequest httpServletRequest) throws AccessDeniedException {
        String username = surveyService.extractUsername(httpServletRequest);
        if (username == null) return ResponseEntity.status(401).build();

        Survey survey = surveyService.findById(id).orElseThrow();

        if (!survey.getCreator().getUsername().equals(username)){
            throw new AccessDeniedException("You can delete only your own surveys");
        }

        surveyService.deleteSurvey(id);
        return ResponseEntity.noContent().build();
    }

    @PutMapping("/{id}")
    public ResponseEntity<Survey> updateSurvey(@PathVariable Long id, @RequestBody Survey updatedSurvey, HttpServletRequest httpServletRequest ) throws AccessDeniedException {
        String username = surveyService.extractUsername(httpServletRequest);
        if (username == null) return ResponseEntity.status(401).build();

        Survey survey = surveyService.findById(id).orElseThrow();
        if (!survey.getCreator().getUsername().equals(username)){
            throw new AccessDeniedException("You can update only your surveys");
        }
        survey.setTitle(updatedSurvey.getTitle());
        survey.setDescription(updatedSurvey.getDescription());
        survey.setRequireAuth(updatedSurvey.isRequireAuth());
        survey.setPublic(updatedSurvey.isPublic());
    }

```

					ІАЛЦ.467200.007 Д4	Арк.
						35
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        return ResponseEntity.ok(surveyService.save(survey));
    }

    @PutMapping("/{id}/finish")
    public ResponseEntity<Survey> finish(@PathVariable Long id , HttpServletRequest
HttpServletRequest) throws AccessDeniedException {
        String username = surveyService.extractUsername(httpServletRequest);
        Survey finishedSurvey = surveyService.finishSurvey(id,username);
        return ResponseEntity.ok(finishedSurvey);
    }
}

```

Auth.service.ts

```

import { Injectable } from '@angular/core';
import {HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { tick } from '@angular/core/testing';
import { LoginRequest} from '../models/login-request.model';
import { RegisterRequest } from '../models/register-request.model';
@Injectable({
    providedIn: 'root'
})
export class AuthService {

    private baseUrl = 'http://localhost:8080/api/auth';

    constructor(private http: HttpClient) { }

    login(request: LoginRequest):Observable<any>{
        return this.http.post(`${this.baseUrl}/login`,request);
    }

    register(request: RegisterRequest): Observable<any> {
        return this.http.post(`${this.baseUrl}/register`, request);
    }

    saveToken(token: string): void {
        localStorage.setItem('token', token);
    }

    getToken(): string | null {
        return localStorage.getItem('token');
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

```

}

logout(): void {
  localStorage.removeItem('token');
}

isLoggedIn(): boolean {
  return !!this.getToken();
}
}

```

Answer.service.ts

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({ providedIn: 'root' })
export class AnswerService {
  private baseUrl = 'http://localhost:8080/api/answers';

  constructor(private http: HttpClient) {}

  submitAnswers(answers: any[]) {
    return this.http.post(this.baseUrl, answers);
  }
}

```

Survey.service.ts

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from '../auth/auth.service';
import { Survey } from '../models/survey.model';
import { AnswerRequest } from '../models/answer-request.model';
import { SurveyResultsResponse } from '../models/survey-result.model';

/*export interface Survey {
  id: number;
  title: string;
  description: string;
  requireAuth: boolean;
  createdAt: string;
}*/

```

					ІАЛЦ.467200.007 Д4	Арк.
						37
Зм.	Арк.	№ докум.	Підпис	Дата		

```

export interface CreateSurveyRequest {
    title: string;
    description:string;
    requireAuth: boolean;
    isPublic: boolean;
    questions: QuestionRequest[],
    activeFrom?: string,
    activeUntil?:string
}

export interface QuestionRequest {
    text: string;
    type: 'TEXT' | 'SINGLE_CHOICE' | 'MULTIPLE_CHOICE';
    position: number;
    options?: {text: string, position:number}[];
}

@Injectables({
    providedIn: 'root'
}))
export class SurveyService {
    private baseUrl = 'http://localhost:8080/api/surveys';

    constructor(private http: HttpClient) {}

    getMySurveys(): Observable<Survey[]> {
        return this.http.get<Survey[]>(`${this.baseUrl}`);
    }

    createSurvey(request: CreateSurveyRequest): Observable<any> {
        return this.http.post(`${this.baseUrl}`, request);
    }

    getSurveyById(id:number): Observable<Survey>{
        return this.http.get<Survey>(`${this.baseUrl}/${id}`);
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		

```

getSurveyDetailsById(id:number): Observable<Survey>{
  return this.http.get<Survey>(`${this.baseUrl}/survey-details/${id}`)
}

updateQuestion(questionId:number, data:QuestionRequest):Observable<any>{
  return this.http.put<any>(`http://localhost:8080/api/questions/${questionId}`,data);
}

addQuestion(surveyId: number, question: any): Observable<any> {
  return this.http.post<any>(`${this.baseUrl}/${surveyId}/questions`, question);
}

updateSurvey(surveyId:number, data:any): Observable<any>{
  return this.http.put<any>(`${this.baseUrl}/${surveyId}`,data)
}

checkIfAlreadySubmitted(surveyId:number): Observable<boolean>{
  return this.http.get<boolean>(`http://localhost:8080/api/answers/check`,{
    params:{ surveyId }
  });
}

submitAnswers(answers: AnswerRequest[]): Observable<any>{
  return this.http.post(`http://localhost:8080/api/answers`,answers);
}

getSurveyAnalytics(surveyId: number) {
  return
this.http.get<SurveyResultsResponse>(`http://localhost:8080/api/analytics/${surveyId}/results`);
}

getPublicSurveys():Observable<any[]>{
  return this.http.get<any>(`${this.baseUrl}/public`)
}

finishSurvey(surveyId:number):Observable<any>{
  return this.http.put<any>(`${this.baseUrl}/${surveyId}/finish`,{});
}
}

```

Login.component.ts

```
import { Component } from '@angular/core';
```

					ІАЛЦ.467200.007 Д4	Арк.
						39
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import { AuthService } from 'src/app/core/auth/auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {
  username='';
  password='';
  error='';

  constructor(private authService: AuthService, private router: Router){}

  login():void{
    this.authService.login({username:this.username,password:this.password
    }).subscribe({
      next: (res) => {
        this.authService.saveToken(res.token);
        this.router.navigate([''])
      },
      error: (err)=>{
        this.error = 'Неправильні дані для входу';
        console.error(err);
      }
    })
  }
}

```

Register.component.ts

```

import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/core/auth/auth.service';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})

```

					ІАЛЦ.467200.007 Д4	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		

```

export class RegisterComponent {
  username: string = '';
  password: string = '';
  email: string = '';

  constructor(
    private authService: AuthService,
    private router: Router
  ) {}

  register(): void {
    this.authService.register({
      username: this.username,
      password: this.password,
      email: this.email
    }).subscribe({
      next: () => {
        alert('Реєстрація успішна!');
        this.router.navigate(['/login']);
      },
      error: (error) => {
        console.error('Помилка при реєстрації:', error);
        alert('Помилка реєстрації. Спробуйте ще раз.');
      }
    });
  }
}

```

Create-survey.component.ts

```

import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { QuestionRequest, SurveyService, CreateSurveyRequest } from
'src/app/core/services/survey.service';

interface CreateQuestionRequest {
  text: string;
  type: 'TEXT' | 'SINGLE_CHOICE' | 'MULTIPLE_CHOICE';
  position: number;
  options?: string[];
}

interface CreateFullSurveyRequest {
  title: string;

```

					ІАЛЦ.467200.007 Д4	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

```

description: string;
requireAuth: boolean;
isPublic: boolean;
questions: CreateQuestionRequest[];
}

@Component({
  selector: 'app-create-survey',
  templateUrl: './create-survey.component.html',
  styleUrls: ['./create-survey.component.css']
})
export class CreateSurveyComponent {
  title: string = '';
  description: string = '';
  requireAuth: boolean = false;
  isPublic: boolean = false;
  questions: QuestionRequest[] = [];
  activeFromDate!:string;
  activeFromTime!:string;
  activeUntilDate!:string;
  activeUntilTime!:string;

  constructor(private surveyService: SurveyService, private router:Router){}

  addQuestion():void{
    this.questions.push({
      text:'',
      type: 'TEXT',
      position: this.questions.length,
      options:[]
    });
  }

  removeQuestion(index: number):void{
    this.questions.splice(index,1);
  }

  addOption(questionIndex: number):void{
    this.questions[questionIndex].options = this.questions[questionIndex].options || [];
    this.questions[questionIndex].options!.push({
      text: '',

```

					ІАЛЦ.467200.007 Д4	Арк.
						42
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        position: this.questions[questionIndex].options!.length
    });
}

removeOption(questionIndex: number, optionIndex:number):void{
    this.questions[questionIndex].options!.splice(optionIndex,1);
}

submit(): void {
    const isoFrom = this.activeFromDate && this.activeFromTime ?
`${this.activeFromDate}T${this.activeFromTime}:00Z` : undefined;
    const isoUntil = this.activeUntilDate && this.activeUntilTime ?
`${this.activeUntilDate}T${this.activeUntilTime}:00Z` : undefined;
    const request: CreateSurveyRequest = {
        title: this.title,
        description: this.description,
        requireAuth: this.requireAuth,
        isPublic: this.isPublic,
        questions: this.questions,

        activeFrom: isoFrom,
        activeUntil: isoUntil
    };
    console.log('>>> payload:', request);
    this.surveyService.createSurvey(request).subscribe({
        next: () => this.router.navigate(['/my-surveys']),
        error: (err) => console.error('Помилка при створенні опитування:', err)
    });
}
}

```

Home.component.ts

```

import { Component } from '@angular/core';
import { AuthService } from 'src/app/core/auth/auth.service';

@Component({
    selector: 'app-home',
    templateUrl: './home.component.html',

```

					ІАЛЦ.467200.007 Д4	Арк.
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    styleUrls: ['./home.component.css']
  })
  export class HomeComponent {
    constructor(public authService:AuthService){}
  }

```

My-surveys.component.ts

```

import { Component, OnInit } from '@angular/core';
import { SurveyService } from 'src/app/core/services/survey.service';
import { Router } from '@angular/router';
import { Survey } from 'src/app/core/models/survey.model';
@Component({
  selector: 'app-my-surveys',
  templateUrl: './my-surveys.component.html',
  styleUrls: ['./my-surveys.component.css']
})
export class MySurveysComponent implements OnInit{
  surveys: Survey[] = [];
  constructor(private surveyService: SurveyService,private router:Router){}

  viewDetails(id:number){
    this.router.navigate(['/survey-details',id])
  }

  ngOnInit(): void {
    this.surveyService.getMySurveys().subscribe({
      next: (data) => this.surveys=data,
      error: (err) => console.error('Помилка завантаження опитувань',err)
    });
  }
}

```

Public-surveys.component.ts

```

import { Component } from '@angular/core';
import { SurveyService } from 'src/app/core/services/survey.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-public-surveys',
  templateUrl: './public-surveys.component.html',
  styleUrls: ['./public-surveys.component.css']
})
export class PublicSurveysComponent {
  surveys: any[] = [];

```

					ІАЛЦ.467200.007 Д4	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

```

constructor(private surveyService: SurveyService, private router: Router) {}

ngOnInit(): void {
  this.surveyService.getPublicSurveys().subscribe({
    next: data => this.surveys = data,
    error: err => console.error('Помилка отримання публічних опитувань:', err)
  });
}

goToSurvey(id: number): void {
  this.router.navigate(['/survey', id]);
}
}

```

Survey-details.component.ts

```

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { CreateSurveyRequest, SurveyService } from 'src/app/core/services/survey.service';
//import { Survey } from 'src/app/core/services/survey.service';
import { Survey } from 'src/app/core/models/survey.model';
import { Title } from '@angular/platform-browser';

```

```

@Component({
  selector: 'app-survey-details',
  templateUrl: './survey-details.component.html',
  styleUrls: ['./survey-details.component.css']
})

```

```

export class SurveyDetailsComponent implements OnInit{
  survey?: Survey;
  editedIndex: number | null = null;
  editableQuestion: any = null;
  newQuestionVisible: boolean = false;
  newDescription: string = '';
  newQuestion: any = {
    text: '',
    type: 'TEXT',
    options: []
  };
}

```

```

constructor(private route: ActivatedRoute, private surveyService: SurveyService, private
router: Router) {}

```

					ІАЛЦ.467200.007 Д4	Арк.
						45
Зм.	Арк.	№ докум.	Підпис	Дата		

```

ngOnInit(): void {
  const id = Number(this.route.snapshot.paramMap.get('id'));
  this.surveyService.getSurveyDetailsById(id).subscribe({
    next: data => this.survey = data,
    error: err => console.error('Помилка завантаження опитування', err)
  });
}

editQuestion(index: number): void {
  this.editedIndex = index;
  const original = this.survey!.questions[index];
  this.editableQuestion = {
    text: original.text,
    type: original.type,
    options: original.options?.map((o: any) => ({ text: o.text, position: o.position })) ||
[]
  };
}

cancelEdit(): void {
  this.editedIndex = null;
  this.editableQuestion = null;
}

addOption(): void {
  this.editableQuestion.options.push({
    text: '',
    position:
this.editableQuestion.options.length });
}

removeOption(index: number): void {
  this.editableQuestion.options.splice(index, 1);
}

saveQuestion(index: number, questionId: number): void {
  const request = {
    text: this.editableQuestion.text,
    type: this.editableQuestion.type,
    position: index,
    options: this.editableQuestion.type === 'TEXT' ? [] : this.editableQuestion.options
  };

  this.surveyService.updateQuestion(questionId, request).subscribe({

```

					ІАЛЦ.467200.007 Д4	Арк.
						46
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    next: () => {
      this.survey!.questions[index].text = request.text;
      this.survey!.questions[index].type = request.type;
      this.survey!.questions[index].options = request.options;
      this.cancelEdit();
    },
    error: err => console.error('Помилка збереження питання', err)
  });
}

showNewQuestionForm(): void {
  this.newQuestionVisible = true;
  this.newQuestion = {
    text: '',
    type: 'TEXT',
    options: []
  };
}

addNewOption(): void {
  this.newQuestion.options.push({ text: '', position: this.newQuestion.options.length });
}

removeNewOption(index: number): void {
  this.newQuestion.options.splice(index, 1);
}

saveNewQuestion(): void {
  const request = {
    text: this.newQuestion.text,
    type: this.newQuestion.type,
    position: this.survey!.questions.length,
    options: this.newQuestion.type === 'TEXT' ? [] : this.newQuestion.options
  };

  this.surveyService.addQuestion(this.survey!.id, request).subscribe({
    next: (newQ) => {
      this.survey!.questions.push(newQ);
      this.newQuestionVisible = false;
    },
    error: err => console.error('Помилка додавання питання', err)
  });
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						47
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    });
}

saveSurveyChanges():void{
    if (!this.survey) return;
    const updatedSurvey = {
        title: this.survey.title,
        description: this.survey.description,
        requireAuth: this.survey.requireAuth,
        public: this.survey.public,
    };
    this.surveyService.updateSurvey(this.survey.id,updatedSurvey).subscribe({
        next: ()=>{
            alert('Зміни пройшли успішно')
            this.router.navigate(['/my-surveys'])
        },error: err=>{
            if(err.status === 400 || err.status === 409){
                alert('неможливо редагувати питання, на яке вже надано відповідь')
            }
            alert('Помилки при збереженні відредагованого опитування')
        }
    })
}

finishSurvey():void{
    this.surveyService.finishSurvey(this.survey!.id)
    .subscribe(updated => this.survey = updated);
}
}

```

Survey-pass.component.ts

```

import { Component } from '@angular/core';
import { SurveyService } from 'src/app/core/services/survey.service';
import { ActivatedRoute } from '@angular/router';
import { AnswerService } from 'src/app/core/services/answer.serveci';
import { Survey } from 'src/app/core/models/survey.model';
import { AnswerRequest } from 'src/app/core/models/answer-request.model';
import { Router } from '@angular/router';
@Component({
    selector: 'app-survey-pass',
    templateUrl: './survey-pass.component.html',
    styleUrls: ['./survey-pass.component.css']

```

					ІАЛЦ.467200.007 Д4	Арк.
						48
Зм.	Арк.	№ докум.	Підпис	Дата		

```

}))
export class SurveyPassComponent {

  surveyId!: number;
  survey: Survey | undefined;
  alreadySubmitted = false;
  loading = true;
  answers: { [questionId: number]: any } = {};
  surveyClosed = false;

  constructor(
    private route: ActivatedRoute,
    private surveyService: SurveyService,
    private answerService: AnswerService,
    public router : Router
  ) {}

  ngOnInit(): void {
    this.surveyId = Number(this.route.snapshot.paramMap.get('id'));
    this.loadSurveyAndCheck();

    /*this.surveyService.checkIfAlreadySubmitted(this.surveyId).subscribe({
      next: submitted => {
        this.alreadySubmitted = submitted;
        if (!submitted) {
          this.loadSurvey();
        } else {
          this.loading = false;
        }
      },
      error: err => {
        console.error('Помилка перевірки проходження:', err);
        this.loading = false;
      }
    });*/
  }

  private loadSurveyAndCheck(): void {
    this.surveyService.getSurveyById(this.surveyId).subscribe({
      next: data => {
        this.survey = data;
        if (!this.isSurveyActive(data)) {
          this.surveyClosed = true;

```

					ІАЛЦ.467200.007 Д4	Арк.
						49
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        this.loading = false;
    } else {
        this.checkIfSubmitted();
    }
},
error: err => {
    console.error('Помилка завантаження опитування:', err);
    this.loading = false;
}
});
}

```

```

private checkIfSubmitted(): void {
    this.surveyService.checkIfAlreadySubmitted(this.surveyId).subscribe({
        next: submitted => {
            this.alreadySubmitted = submitted;
            this.loading = false;
        },
        error: err => {
            console.error('Помилка перевірки проходження:', err);
            this.loading = false;
        }
    });
}

```

```

private isSurveyActive(s: Survey): boolean {
    const now = new Date();
    if (!s.isActive) {
        return false;
    }
    if (s.activeFrom && new Date(s.activeFrom) > now) {
        return false;
    }
    if (s.activeUntil && new Date(s.activeUntil) < now) {
        return false;
    }
    return true;
}

```

```

loadSurvey(): void {
    this.surveyService.getSurveyById(this.surveyId).subscribe({

```

					ІАЛЦ.467200.007 Д4	Арк.
						50
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    next: data => {
      this.survey = data;
      this.loading = false;
    },
    error: err => {
      console.error('Помилка завантаження опитування:', err);
      this.loading = false;
    }
  });
}

onCheckboxChange(questionId: number, optionId: number, checked: boolean): void {
  if (!this.answers[questionId]) {
    this.answers[questionId] = [];
  }
  if (checked) {
    this.answers[questionId].push(optionId);
  } else {
    this.answers[questionId] =
      this.answers[questionId].filter((id: number) => id !== optionId);
  }
}

onCheckboxChangeEvent(
  qId: number,
  optId: number,
  event: Event
): void {
  const input = event.target as HTMLInputElement;
  this.onCheckboxChange(qId, optId, input.checked);
}

/*submitAnswers(): void {
  if (!this.survey) return;

  const answerRequests: AnswerRequest[] = this.survey.questions.map(q => {
    const ans = this.answers[q.id];

    if (q.type === 'TEXT') {
      return {
        questionId: q.id,
        textAnswer: ans || ''
      };
    }
  });
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						51
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    }

    if (q.type === 'SINGLE_CHOICE') {
        const selected = ans ? [ans as number] : [];
        return {
            questionId: q.id,
            selectedOptionId: selected
        };
    }

    return {
        questionId: q.id,
        selectedOptionId: ans as number[]
    };
});

this.surveyService.submitAnswers(answerRequests).subscribe({
    next: () => (this.alreadySubmitted = true),
    error: err => console.error('Помилка надсилання відповідей:', err)
});
}*/

submitAnswers(): void {
    if (!this.survey) {
        return;
    }

    const answerRequests: AnswerRequest[] = this.survey.questions.map(q => {
        const ans = this.answers[q.id];

        // Для питань із варіантами (радіо та чекбокси) приводимо у масив чисел
        const selected: number[] = ans != null
            ? Array.isArray(ans)
                ? ans // вже масив для MULTIPLE_CHOICE
                : [ans] // обгортаємо у масив для SINGLE_CHOICE
            : [];

        if (q.type === 'TEXT') {
            return {
                questionId: q.id,
                textAnswer: (ans as string) || ''
                // selectedOptionId можна опустити, якщо DTO допускає undefined
            };
        }
    });
}

```

					ІАЛЩ.467200.007 Д4	Арк.
						52
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    };
}

// SINGLE_CHOICE або MULTIPLE_CHOICE
return {
    questionId: q.id,
    selectedOptionId: selected
    // textAnswer можна опустити або поставити null/undefined
};
});

this.answerService.submitAnswers(answerRequests).subscribe({
    next: () => {
        this.alreadySubmitted = true;
    },
    error: err => {
        console.error('Помилка надсилання відповідей:', err);
    }
});
}

```

```

/*canSubmit():boolean{
    if(!this.survey){
        return false;
    }
    for (const q of this.survey.questions) {
        if (q.type !== 'TEXT') {
            const ans = this.answers[q.id];
            if (!Array.isArray(ans) || ans.length === 0) {
                return false;
            }
        }
    }
    return true;
}*/

```

```

    canSubmit(): boolean {
    if (!this.survey) {
        return false;
    }

```

					ІАЛЦ.467200.007 Д4	Арк.
						53
Зм.	Арк.	№ докум.	Підпис	Дата		

```

for (const q of this.survey.questions) {
  if (q.type === 'TEXT') {
    // Для текстових питань ми нічого не перевіряємо
    continue;
  }

  const ans = this.answers[q.id];
  // Якщо відповіді взагалі немає → не можна сабмітити
  if (ans == null) {
    return false;
  }

  if (q.type === 'MULTIPLE_CHOICE') {
    // Для мультивибору треба, щоб це був масив і не порожній
    if (!Array.isArray(ans) || ans.length === 0) {
      return false;
    }
  }
  // Для SINGLE_CHOICE достатньо просто перевірити ans != null,
  // бо ans – одне число, і воно truthy
}

return true;
}
}

```

Survey-results.component.ts

```

import { Component, OnInit } from '@angular/core';
import { SurveyService } from 'src/app/core/services/survey.service';
import { ActivatedRoute } from '@angular/router';
import { SurveyResultsResponse } from 'src/app/core/models/survey-result.model';

@Component({
  selector: 'app-survey-results',
  templateUrl: './survey-results.component.html',
  styleUrls: ['./survey-results.component.css']
})
export class SurveyResultsComponent implements OnInit {
  surveyId!: number;
  results?: SurveyResultsResponse;
  loading = true;
  error: string | null = null;

```

					ІАЛЦ.467200.007 Д4	Арк.
						54
Зм.	Арк.	№ докум.	Підпис	Дата		

```

constructor(
  private route: ActivatedRoute,
  private surveyService: SurveyService
) {}

ngOnInit(): void {
  this.surveyId = Number(this.route.snapshot.paramMap.get('id'));
  this.surveyService.getSurveyAnalytics(this.surveyId).subscribe({
    next: res => {
      this.results = res;
      this.loading = false;
    },
    error: err => {
      this.error = 'Не вдалося завантажити аналітику';
      console.error(err);
      this.loading = false;
    }
  });
}

formatStatsForChart(optionStats: any[]): { name: string, value: number }[] {
  return optionStats.map(opt => ({
    name: opt.text,
    value: opt.count
  }));
}
}

```

App-routing.module.ts

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from '../features/auth/login/login.component';
import { RegisterComponent } from '../features/auth/register/register.component';
import { AuthGuard } from '../core/guards/auth.guard';
import { SurveyResultsComponent } from '../features/survey-results/survey-results.component';
import { MySurveysComponent } from '../features/my-surveys/my-surveys.component';
import { CreateSurveyComponent } from '../features/create-survey/create-survey.component';
import { SurveyPassComponent } from '../features/survey-pass/survey-pass.component';
import { HomeComponent } from '../features/home/home.component';

```

					ІАЛЦ.467200.007 Д4	Арк.
						55
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import { SurveyDetailsComponent } from './features/survey-details/survey-details.component';
import { PublicSurveysComponent } from './features/public-surveys/public-surveys.component';

const routes: Routes = [
  {path:'login', component: LoginComponent},
  {path:'', component: HomeComponent},
  {path:'register', component: RegisterComponent},

  {path:'my-surveys', component: MySurveysComponent, canActivate:[AuthGuard]},
  {path:'create-survey', component: CreateSurveyComponent, canActivate:[AuthGuard]},
  {path:'survey-results/:id', component: SurveyResultsComponent, canActivate:[AuthGuard]},
  {path:'surveys/public', component: PublicSurveysComponent, canActivate:[AuthGuard]},

  {path:'survey/:id', component: SurveyPassComponent},
  {path:'survey-details/:id', component: SurveyDetailsComponent, canActivate:[AuthGuard]},
  {path:'**', redirectTo:''}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

import { Component } from '@angular/core';
import { AuthService } from '../core/auth/auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'survey-app';

  constructor(public authService: AuthService, private router: Router){}

  logout():void{
    this.authService.logout();
    this.router.navigate(['/login'])
  }
}

```

App.module.ts

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

```

					ІАЛЦ.467200.007 Д4	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

```

import { AppRoutingModuleModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { AuthModule } from './features/auth/auth.module';
import { SurveysModule } from './features/surveys/surveys.module';
import { HTTP_INTERCEPTORS, HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';
import { AuthInterceptor } from './core/interceptors/auth.interceptor';
import { HomeComponent } from './features/home/home.component';
import { MySurveysComponent } from './features/my-surveys/my-surveys.component';
import { CreateSurveyComponent } from './features/create-survey/create-survey.component';
import { SurveyPassComponent } from './features/survey-pass/survey-pass.component';
import { SurveyResultsComponent } from './features/survey-results/survey-results.component';
import { SurveyDetailsComponent } from './features/survey-details/survey-details.component';
import { NgxChartsModule } from '@swimlane/ngx-charts';
import { PublicSurveysComponent } from './features/public-surveys/public-surveys.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    MySurveysComponent,
    CreateSurveyComponent,
    SurveyPassComponent,
    SurveyResultsComponent,
    SurveyDetailsComponent,
    PublicSurveysComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModuleModule,
    AuthModule,
    SurveysModule,
    HttpClientModule,
    FormsModule,
    NgxChartsModule
  ],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: AuthInterceptor,
      multi: true,
    },
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

					ІАЛЦ.467200.007 Д4	Арк.
						57
Зм.	Арк.	№ докум.	Підпис	Дата		