МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
18 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційно-аналітична технологія підтримки соціальних опитувань»
здобувача групи ІН.м – 24 Войтовича Назара Валерійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Назар Войтович
(підпис)

Керівник,
доцент кафедри кібербезпеки,
к.т.н. Віктор ОБОДЯК
(підпис)

Суми – 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-24 Войтовича Назара Валерійовича

1. Тема роботи: «Інформаційна технологія прогнозування курсу валют»
затверджую наказом по СумДУ від *«06» грудня 2023 р. № 1412-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 18 грудня 2023 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми в соціальній сфері та постановка завдань дослідження. 2) Огляд технологій у проведенні соціальних опитувань. 3) Розробка дизайну системи проведення соціальних опитувань. 4) Розробка інтелектуальної системи для проведення соціальних опитувань. 5) Аналіз отриманих результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми в соціальній сфері та постановка завдань дослідження*		
2	*Огляд технологій у проведенні соціальних опитувань*		
3	*Розробка дизайну системи проведення соціальних опитувань*		
4	*Розробка інтелектуальної системи для проведення соціальних опитувань*		
5	*Аналіз отриманих результатів*		
6	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 62 стр., 33 рис. , 2 додатки, 17 використаних джерел.

Обґрунтування актуальності теми роботи – система кваліфікаційної роботи є належним чином актуальною, оскільки вона присвячена вирішенню важливої соціальної задачі, а саме створенню системи проведення соціальних опитувань з використанням новітніх технологій.

Об’єкт дослідження — процес проведення соціальних опитувань з використанням інформаційних технологій.

Мета роботи — розробити імплементацію системи для зручного та ефективного проведення соціальних опитувань

Методи дослідження — аналіз вимог до соціальних опитувань, проектування структури бази даних, розробка серверної та клієнтської частин.

Результати — створено функціональну та ефективну систему для проведення соціальних опитувань, яка забезпечує зручний інтерфейс для користувачів, швидку обробку опитувальних даних та надійне зберігання результатів в базі даних PostgreSQL.

ІНФОРМАЦІЙНА СИСТЕМА, СУСПІЛЬНІ ОПИТУВАННЯ, RUST, SWIFT, XCODE, POSTGRESQL.

ЗМІСТ

ВСТУП

Актуальність. У сучасних умовах глобальних трансформацій соціальна робота стає ключовою професією для стійкого розвитку суспільства [1]. Здатність враховувати думку громади стає важливим аспектом вирішення загальних проблем у політиці, бізнесі та громадських організаціях.

Об'єкт дослідження. Думка громади, виявлена через соціальні опитування, виступ орієнтиром у виборі стратегій вирішення суспільних проблем.

Предмет дослідження. Використання методу соціального опитування в суспільстві, чітке визначення громадської думки, залучення громади, аналіз результатів, соціальна аналітика та планування, ініціація міжособистісної взаємодії.

Гіпотеза. Використання методу соціального опитування в суспільстві має численні переваги, такі як чітке визначення громадської думки, залучення громади, аналіз результатів, соціальна аналітика та планування, ініціація міжособистісної взаємодії.

Новизна. Робота прагне надати ефективний і доступний спосіб для широкого кола людей залучати громадську думку в суспільні справи, використовуючи сучасні технології. Мобільна та стаціонарна платформи допоможуть максимізувати охоплення аудиторії та підвищити точність вимірювання даних.

Структура. Робота включає в себе вступ, аналітичний огляд, постановку завдання, вибір методу для вирішення поставленої задачі, розгляд програмного забезпечення інформаційної системи, висновки, перелік використаних джерел та додатків.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз властивостей системи проведення соціальних опитувань

Для того щоб система для проведення соціальних опитувань мала попит серед користувачів їй необхідно відповідати низці умов, які б надавали їй перевагу у порівнянні з існуючими аналогами на ринку, наприклад:

- Безпека - усі дані за можливості мають зберігати свою конфіденційність та буди захищеними від зовнішнього доступу у найвищій доступній мірі. Це надає користувачам упевненість у власній безпеці та убезпечить повноту свободи вибору.
- UI/UX - з точки зору користувацького інтерфейсу, такі дії, як участь у опитуваннях або їх створення, не маються викликати труднощів у людей. Усі потенційні взаємодії з додатком мають бути інтуїтивно зрозумілими та швидкими у реалізації, для цього необхідна продуманість та виваженість.
- Аналіз - результати опитування повинні мати зручні та візуальні інструменти для їх подальшого аналізу, що допоможе робити відповідні висновки на їх основі. [2]
- Кросплатформенність - чим більше можливостей для участі, тим більше потенційний обсяг учасників, що, в свою чергу, збільшує релевантність отриманих результатів.
- Підтримка - кожна система на початку свого існування не може вважатися ідеальною. На початку необхідна постійна робота над помилками та покращення продукту, звертаючи увагу на відгуки користувачів.

1.2 Огляд існуючих аналогів

Найкращий спосіб зробити якийсь продукт добре, це подивитись на існуючі схожі аналоги, якщо вони існують звісно, і перейняти усі їх недоліки

та переваги. Це допоможе уникнути банальних проблем на початку життєдіяльності проекту і зорієнтує подальший вектор його розвитку у правильному напрямку.

1.2.1 Платформа “SurveyMonkey”

SurveyMonkey показує себе як гарний приклад додатку для проведення доступних соціальних опитувань.

Пропонується високий рівень візуалізації опитування, надаючи змогу змінювати як типи питань, так і сам процес його проходження зі сторони опитуваного. Надається також можливість впливати на візуальне відображення опитування, що робить можливим прирівнювати тематику питань з відображенням на екрані, покращуючи візуальне сприйняття продукту.

Функціонал аналізу результатів опитування у вигляді гарних графіків робить набагато легшим процес обробки результатів для тих хто його проводить.

Багате різноманіття каналів відправки, таких як пошта або соціальні мережі, забезпечуються високий рівень забезпечення опитувань людьми котрі беруть в них участь, збільшуючи релевантність фінальної вибірки опитуваних.

Є гарним прикладом конкурента з якого можна виділити позитивні риси та впровадити їх у власному варіанті реалізації системи проведення опитувань, що збільшить аудиторію.

Процес опитування у додатку наведено на рис. 1.1.

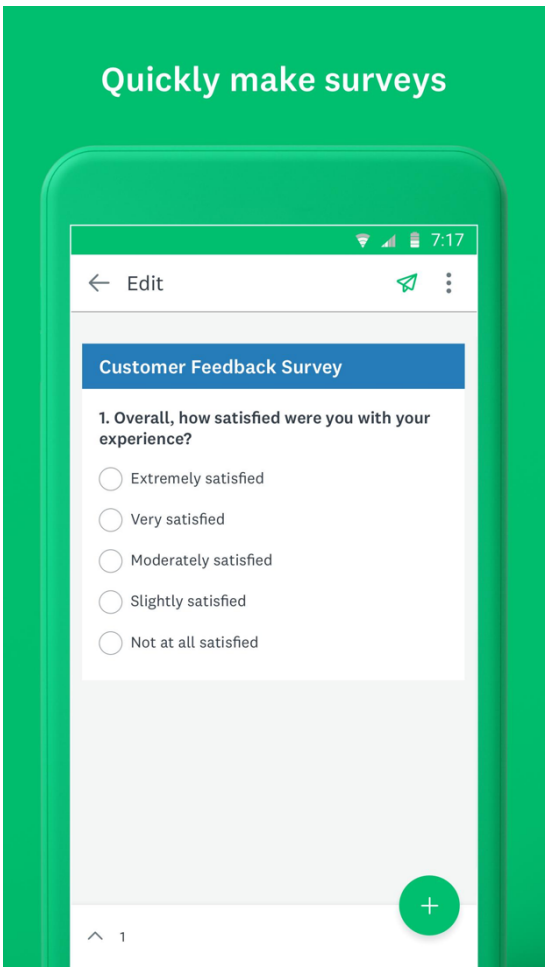


Рисунок 1.1 – Екран опитування додатку «SurveyMonkey» [3]

Надає розширені засоби аналізу даних, включаючи графіки та звіти, які допомагають користувачам зрозуміти результати опитувань.

Також платформа дозволяє створювати опитування для різних масштабів від невеликих опитувань до великих досліджень, що робить можливим вибір опції, яка краще за всіх підходить саме на даний конкретний момент.

Головним недоліком даного продукту можна окреслити його неповноцінність у можливостях безкоштовної його версії. Поки користувач не отримає за відповідну плату повну версію, він буде обмежений у функціоналі у вигляді ліміту на кількість питань та недоступності деяких варіантів аналізу даних.

Цінова політика додатку наведена на рис. 1.2

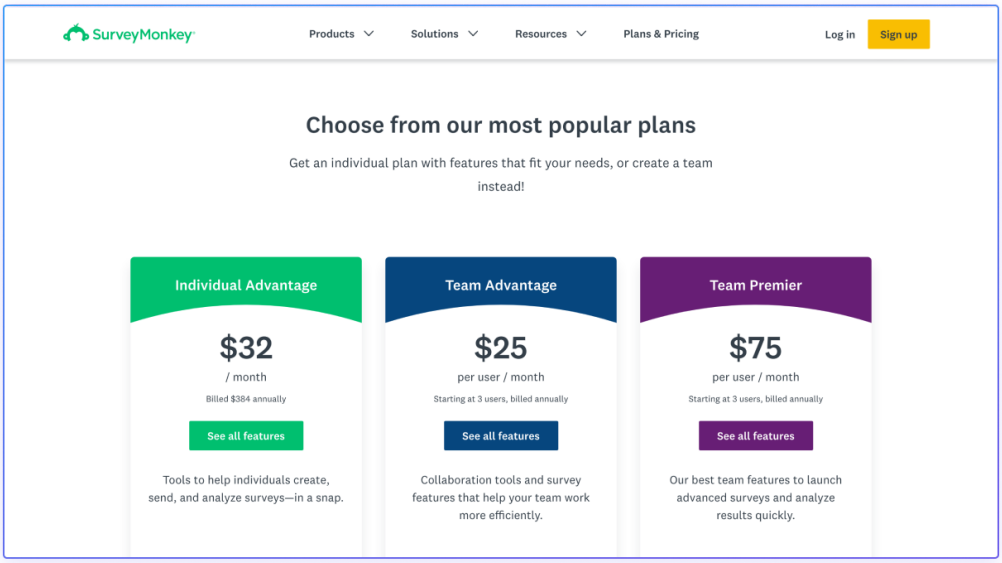


Рисунок 1.2 – Цінова політика додатку «SurveyMonkey» [3]

1.2.2 Платформа “Typeform”

Typeform пропонує зручний у використанні інтерфейс як для створення опитувань, так і участі у них.

Можливість відкривати опитування у браузері збільшує обхват аудиторії, полегшуючи залучення нових людей, адже сам процес у них буде займати меншу кількість часу.

Доступні зручні інструменти для моніторингу опитувань та подальшого їх аналізу. Різноманітні графіки та діаграми полегшують сприйняття інформації та допомагають робити відповідні висновки за візуальним сприйняттям.

Дозволяє створювати динамічні форми з логічним переходом між питаннями, що поліпшує взаємодію та знижує ризик втрати учасників, відповідно і якість вибірки.

Деякі засоби аналітики та звітності доступні тільки у платних планах, що може обмежити можливості аналізу результатів для користувачів безкоштовної версії.

Процес опитування у додатку наведено на рис. 1.3.

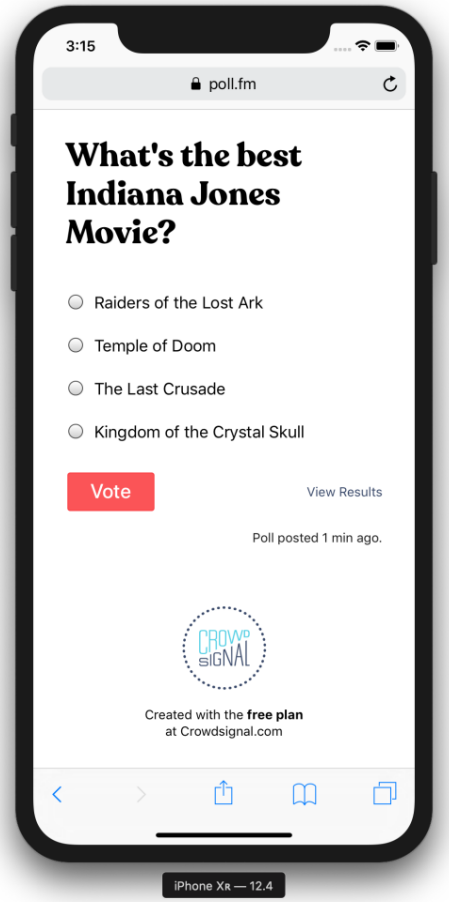


Рисунок 1.3– Екран опитування додатку «Typeform » [4]

Так як і у попередньому додатку, “Typeform” потребує додаткової платні для розблокування усього наявного у ньому функціоналу. Так для безкоштовної версії додаток лімітує кількість опитувань які можливо створити, а також число відповідей, які користувачі можу залишати до них. Якщо планується мати широкий обсяг аудиторії, ці обмеження зроблять неможливим адекватний збір даних, оскільки вибірка відповідей буде у значній мірі обмежена.

Версії Typeform можуть бути відносно дорогими, що може впливати на доступність для користувачів із обмеженим бюджетом, це потрібно врахувати у власній реалізації, адже великі ціни відлякують потенційних користувачів. Можливо розглянути варіант без оплати.

Цінова політика додатку наведена на рис. 1.4

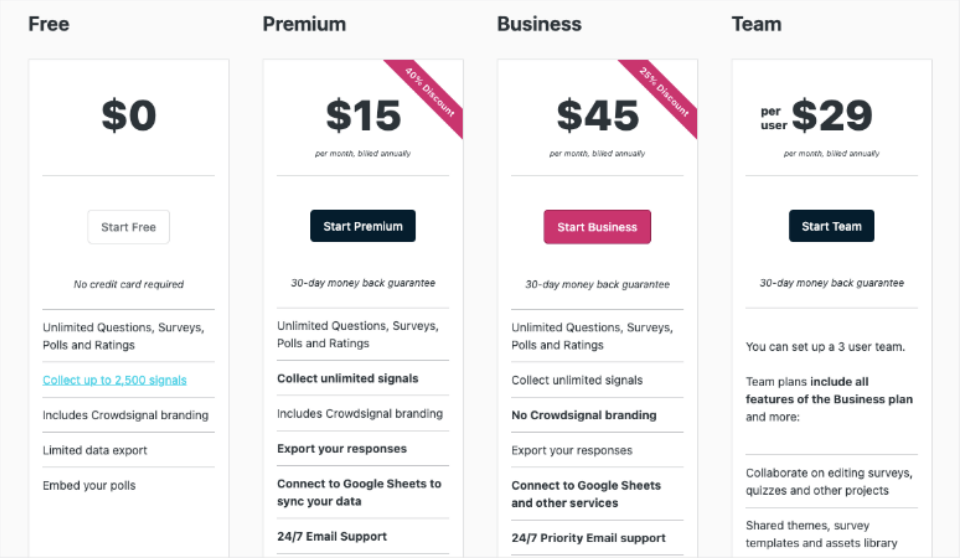


Рисунок 1.4– Цінова політика додатку «Typeform » [4]

1.3 Постановка задачі

Сучасний світ потребує передових засобів для збору та аналізу даних у соціальній сфері. Одним із головних аспектів цього процесу є проведення загальних опитувань, які потребують не лише збору ефективних, а й простих відповідей. У цьому розділі оголошуються вимоги до майбутньої системи проведення соціальних опитувань щоб означити аспекти розробки.

Функціональні вимоги:

1) Необхідно надати користувачу функціонал для створення, моніторингу, розповсюдження та аналізу різного роду опитувань.
2) Дані які збирає додаток мають бути конфіденційними та захищеними, доступ до них надається лише чітко окресленим ролям користувачів.
3) Дані опитувань мають бути зібрані до купи та представлені у зручному для розуміння людини форматі.
4) Має бути створений механізм для зручного процесу розповсюдження та рекламування опитувань зі сторони користувача який його створив, можливість перенаправлення потенційного рецензента до самого процесу відповіді на опитування.

5) Користувачі які проходять опитування повинні мати змогу вносити відомості про себе, такі як вік або стать, щоб покращити можливості аналізу фінальних даних по окремим вибіркам.

1.4 План розробки системи

Для планомірної розробки системи, щоб не ставалося ніяких неочікуваних перешкод, важливо розробити поетапний план її розробки, щоб компоненти вищого рівня не потрапляли у розробку коли компонент низького рівня, від яких вони залежать, ще не були зроблені.

Для майбутньої системи був створений наступний план розробки:

1. Вибір типу та проектування бази даних. Необхідно визначитися з окремим видом бази даних для зберігання опитувань та їх результатів, створити відповідні таблиці, проаналізувати майбутні запити до них.
2. Для зручного користування необхідно розробити зручний користувацький інтерфейс з інтуїтивно зрозумілими наборами дій, які користувач в теорії буде мати бажання зробити.
3. Розробка “backend” складової системи. Тут буде реалізовано логіку опитування та зберігання у базі даних необхідної інформації, з можливість у майбутньому знову отримати до неї доступ.
4. Розробка мобільного додатку на основі розробленого раніше інтерфейсу користувача, а також встановлення його взаємодії з “backend” складовою системи.
5. Розгортання системи та її тестування.
6. Підтримка та обслуговування системи.

2. ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

Перед початком реалізації плану необхідно обрати перелік технологій які бажаємо бачити у власній реалізації системи проведення соціальних опитувань. Вони, в свою чергу, мають максимально підходити до специфікації проекту, та у повній мірі виконувати ту задачу, для якої вони були відібрані.

У сучасному світі для кожної задачі існує декілька інструментів, які допоможуть з нею розібратися, і кожен з них має свої певні відмінності, що надають йому тих чи інших переваг. Отже, задача полягає в тому щоб проаналізувати наявні варіанти, зважити їхні особливості, та набрати ті які найбільше піді йдуть для реалізації даного продукту.

2.1 Вибір системи управління базою даних

Система проведення опитувань являє собою набір певного роду даних, які необхідно десь зберігати на постійній основі.

Саме для цього нам і потрібна база даних, воно допоможе нам зберігати усю інформацію про опитування, про людей які їх пройшли, а також відповіді які вони вибирали, після чого повернути назад у потрібний час для використання у клієнтських частинах, де дані у форматованому вигляді буде спостерігати та використовувати у власних потребах кінцевий користувач.

Напряму клієнти не взаємодіятимуть з базою даних, для цього будуть надані спеціальні сервіси, які будуть мати спеціальний інтерфейс для доступу до певних частин інформації, в залежності від ролі того, хто звертається, та його потреб.

Основні критерії, за якими буде здійснюватися вибір, можна окреслити наступним чином:

- Спосіб збереження інформації
- Структура збереженої інформації
- За рівнем швидкодії

Одразу необхідно врахувати той факт, що система у майбутньому повинна мити змогу розширятися, тому гарним варіантом для розгляду будуть саме реляційні моделі бази даних. Це надасть нам змогу відмовитись від постійного підвищення пропускних здібностей за рахунок підвищення кількості серверів, а натомість дозволить виходити із ситуації шляхом покращення апаратної складової вже наявних машин.

Найпопулярнішим стандартом мов виконання запитів у таких моделях є SQL. Його використання буде доречним і в даному випадку, оскільки він покриває усі необхідні вимоги.

2.1.1 MySQL

MySQL - це система управління реляційними базами даних. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається виграш у швидкості й гнучкості.

Таблиці зв'язуються між собою за допомогою системи відносності, завдяки чому забезпечується можливість поєднувати при виконанні запиту дані з декількох таблиць. SQL, які система MySQL може охарактеризувати за допомогою мови структурованих запитів, що використовується для доступу до бази даних [5].

SQL має зрозумілий синтаксис, зручність у роботі та інтуїтивно зрозумілий підхід, що робить розробку не такою важкою і, відповідно, швидкою.

На ринку наявна велика кількість досвідчених людей, яких завжди можна долучити до розробки для розширення можливостей функціоналу системи.

Використовується великою кількістю компаній, що вказує на те, що продукт є надійним, вивірений часом, що робить працездатність системи більш прогнозованою.

Приклад запиту MySQL наведено на рис. 2.1.

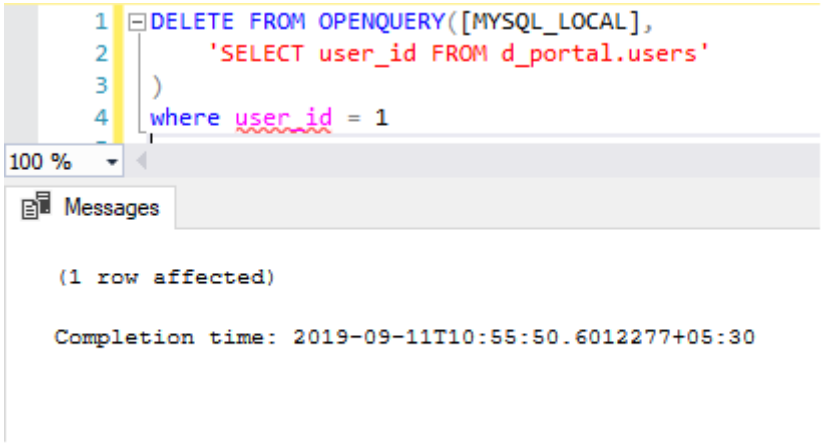


Рисунок 2.1– MySQL запит [4]

MySQL має легкий у освоєнні та функціональний інтерфейс що значно зменшує час витрачений на виконання завдань базового рівня. Є одним з лідерів коли мова йде про швидкість читання, особливо коли справа стосується великого обсягу даних.

У відкритому доступі наявна велика кількість документації та допоміжних статей, що стануть у нагоді при освоєнні інструменту. Також, при потребі, є можливість звернутися до спільноти розробників.

Проте, у порівняння з конкурентами MySQL не вистачає декілька ключових особливостей та розширених можливостей, як наприклад робота з гео-даними, яка підтримується PostgresQL.

2.1.2 PostgresQL

PostgreSQL – це система управління реляційною базою даних (СУБД) з відкритим вихідним кодом, яка використовує та розширює мову запитів SQL для роботи з даними [6]. Потужна об'єктно-реляційна система управління базами даних з відкритим вихідним кодом, яка відзначається високою надійністю, розширюваністю та розширеним набором функцій.

На відміну від MySQL, даний інструмент має підтримку небагато більшого спектру можливостей, наприклад, є можливість створювати власні

типи даних або функції, хоча за це і доводиться розплачуватися дещо сповільненим режимом роботи, особливо під час процесу читання.

Підтримка SQL стандарту дає змогу пере використовувати SQL-запити у разі потреби, та у разі міграції значно спростить процес переходу на інші можливі системи управління базами даних.

Приклад запиту PostgreSQL наведено на рис. 2.2.

```
/* Show the last time at which every table and view was vacuumed and analyzed */
SELECT relname, last_vacuum, last_autovacuum, last_analyze, last_autoanalyze
FROM pg_stat_all_tables  d_d
WHERE schemaname = 'public';

/* See which queries are most popular */
SELECT * FROM pg_stat_statements where calls >=1000;
select pg_stat_statements_reset();

/* Kill a specific query */
select pg_terminate_backend(16390);

/* Get the size of the database */
select pg_size_pretty(pg_database_size('cityscape'));

/* Kill queries that are taking too long */
SELECT query, pg_terminate_backend(pid)
    FROM pg_stat_activity
    WHERE datname = 'cityscape'
      AND pid <> pg_backend_pid()
      AND (state = 'active'
      OR state = 'idle')
      AND state_change < current_timestamp - INTERVAL '15' MINUTE;
```

Рисунок 2.2– PostgresQL запит [7]

Однією з головних переваг якими володіє PostgresQL можна назвати надійність та стабільність. При необхідності можна використати інструменти для синхронізації даних, забезпечення їх цілісності, контроль за цим процесом, та у разі збою, застосувати механізми відновлення. Це дає відповідне право на похибку, що в ключовий момент може зберегти систему в робочому стані.

Надійність є ключовим фактором для бізнесу, тому PostgresQL має стати гарним вибором для безперебійної роботи майбутньої системи для проведення соціальних опитувань.

Наявність великої кількості кваліфікованих людей на ринку праці забезпечить допомого у разі необхідності.

2.2 Серіалізація даних

У системі проведення соціальних опитувань на постійній основі будуть функціонувати велика кількість різноманітних даних, які, в свою чергу, повинні мати змогу безперебійно комунікувати один з одним, відправляючи або отримуючи різноманітні дані.

Для полегшення цього процесу дані необхідно звести до єдиного, зрозумілого усім учасникам, формату. Цей процес і називається серіалізацією даних, і до його вибору треба підходити виважено, адже неправильно обраний варіант може призвести до неочікуваних проблем та затримок, що негативно вплине на бізнес складову.

Існує велика кількість форматів, і кожен має свої переваги та недоліки.

2.2.1 JSON

JSON, або JavaScript Object Notation, можна без перебільшення назвати найпопулярнішим з існуючих форматів. Він дає змогу у текстовій формі представити об’єкт та його властивості, для подальшої передачі цієї інформації через мережу.

Об’єкт представлений у JSON форматі наведено на рис. 2.3.

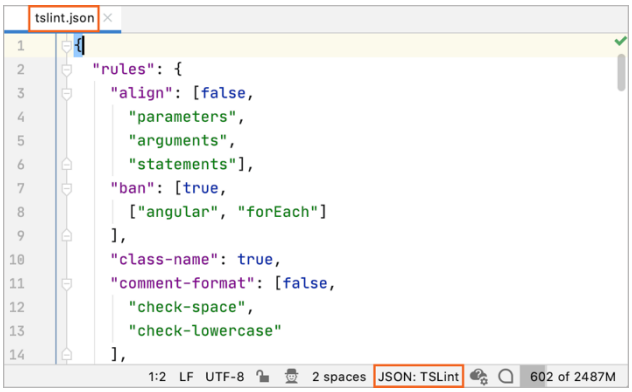


Рисунок 2.3– JSON об’єкт [8]

Він на протязі десятиліть заслуговував свій статус, будучи простим в розумінні, маючи зрозумілий людському оку синтаксис, форматом серіалізації даних.

Його популярність породжує певні переваги, як, наприклад, велика підтримка спільноти та обширний обсяг документації, що спрощує процес інтеграції та підтримки.

Проте, він не позбавлений і недоліків, як наприклад:

1. Обмежений набір типів даних.
2. Немає схематичного відображення об’єкту.
3. Посередній процес десеріалізації.
4. Дані легко читаються, а отже можуть бути легко компрометовані.

Фактори швидкодії та безпечності є ключовими, що робить доцільним продовжити пошук підходящого майбутній системі формату серіалізації даних.

2.2.2 Protobuf

Protobuf (Protocol Buffers) — це комбінація мови програмування Proto та компілятора, яка базується на схемі у вигляді окремих повідомлень, описаних цією мовою у файлах з розширенням «.proto». Ці буфери генерують код, який унікальний для кожної мови програмування або платформи.

Він схожий на JSON, але менший і швидший, а також генерує прив’язки рідною мовою. Необхідно один раз визначити, як структурувати дані, а потім використовувати спеціальний інструмент для генерації вихідного коду під різні мови програмування [9].

Приклад схеми наведено на рис. 2.4.

```
syntax = "proto3";
option csharp_namespace = "CalculatorService.Generated";
message Person
{
    int32 id = 1;
    string name = 2;
    Address address = 3;
}

message Address
{
    int32 id = 1;
    string addressLine1 = 2;
    string city = 3;
}
```

Рисунок 2.4– Protobuf структури [8]

Protocol Buffers — це комбінація мови Proto та компілятора, яка базується на схемі у вигляді окремих повідомлень, описаних цією мовою у файлах з розширенням «.proto». Ці буфери генерують код, який унікальний для кожної мови програмування або платформи.

При роботі з парсингом даних помилки мінімізуються завдяки чіткій схемі та коду, який компілятор уже створив для його обробки.

У зв’язку з тим, що формат представлення даних є бінарним, ми не можемо читати отримані дані без попередньої обробки та створення схеми Proto. Тим не менш, ми отримуємо набагато більшу швидкість обробки отриманих об’єктів програмою, що значно скорочує час очікування користувача при великих об’ємах, що покращує враження користувача від використання програми.

Компілятор і схеми розширення «.proto» дозволяють підтримувати формат даних між різними частинами системи незалежно від мови написання, що робить їхній наступний вибір більш автономним.

Підсумовуючи все це, форматом серіалізації даних може бути Protocol Buffers. Це пов’язано з тим, що його переваги у швидкодії та можливості одночасно підтримувати кілька мов програмування за допомогою схем позитивно вплинуть на можливість подальшої підтримки кодової бази системи та її загальної продуктивності.

2.3 Інструменти серверної частини

Вибір таких аспектів як мова програмування чи фреймворки є гарячою темою для обговорень, кожні з них мають свої переваги та недоліки, а, отже, свої поціновувачів та ненависників.

Для вирішення проблем даної системи була обрана мова програмування Rust. Вона є доволі новою, перший стабільна версія була представлена лише у 2015 році, проте вона вже встигла завоювати свій сегмент, довівши свою надійність та змогу відносно легко виконувати поставлені задачі.

Приклад коду написаного на мові програмування Rust наведено на рис. 2.5.

```rust
pub fn run(&mut self) {

    let mut last_tick = std::time::SystemTime::now();

    loop {
        // takes user input and move the shape accordingly
        self.take_directions();

        // check if we reached time to tick again, then tick
        if last_tick.elapsed().unwrap().as_millis() >= self.speed {
            self.game_over(); // checks if game is over
            self.tick();
            self.clear_full_rows(); //
            last_tick = std::time::SystemTime::now();
        }
    }
}
```

Рисунок 2.5– Функція написана мовою програмування Rust

Доволі зрозумілий те легко читаємий приклад синтаксису.

Rust довів свою здатність забезпечити безпеку та надійність, оскільки має декілька властивостей, що допомагають убезпечити розробника від типових помилок.

Багато людей розглядають мову програмування Rust як альтернативу іншим мовам системного програмування, таким як C або C++. Найбільшою перевагою, яку може забезпечити Rust порівняно з цими мовами, є borrow checker. Вона убезпечує від банальних помилок пов’язаних з управлінням пам`яті, таких як витоки чи неполадки, що робить кінцевий продукт більш надійним.

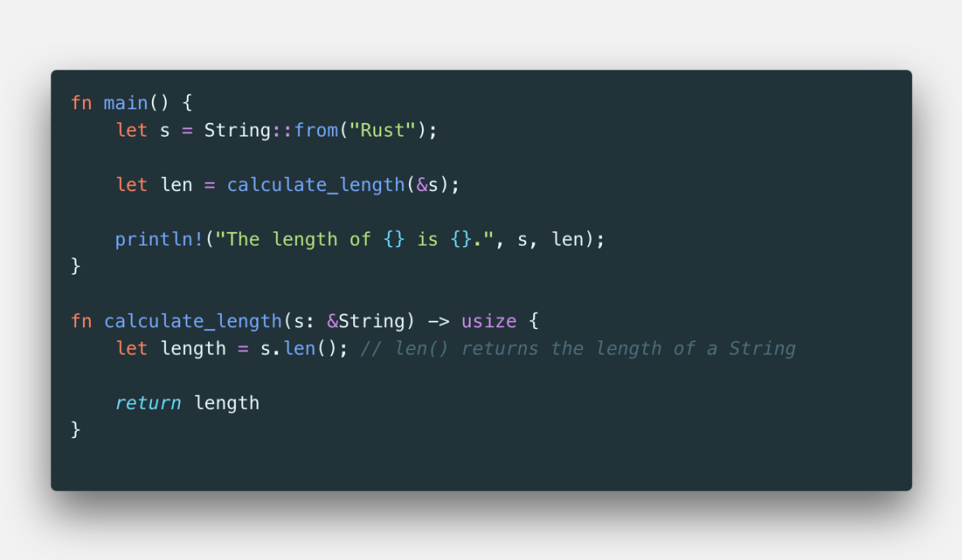
Усі об`єкти мови програмування Rust мають свою “власність” над певною ділянкою пам`яті, яка була виділена для додатку, і лише цей “власник” має змогу змінювати дані всередині цього проміжку. Це дає змогу убезпечити від стану “race condition” усередині додатку, коли два процеси намагаються одночасно маніпулювати однією ділянкою пам`яті, призводячи до неочікуваних результатів. Також це дає змогу самому компілятору аналізувати код, та завчасно виявляти потенційно небезпечні ділянки та повідомляти про них.

Щоб постійно не змінювати власника об`єкту, Rust дає змогу тимчасово взяти “власність” у борг, для подальшої маніпуляції з пам`яттю, проте зобов`язую повернути її назад коли усі необхідні дії були виконані.

Також варто зазначити що таки система володіння дає змогу уникнути необхідності додавати спеціальні механізми так званого “збирання сміття”, для своєчасного звільнення пам`яті, коли це необхідно. Натомість Rust’у лише необхідно очікувати моменту, коли власник лінки вже не буде використовуватись, та одразу звільнити те, чим він володів. Це має гарний вплив на загальну продуктивність кінцевого додатку, бо звільняє його від додаткового навантаження.

Ще однією значущою особливістю мови програмування Rust є вбудована система конкурентності, яка дозволяє ефективно працювати з паралельним виконанням коду та легко вирішувати проблеми забезпечення взаємо виключення в багатозадачних середовищах.

Приклад роботи з пам`яттю Rust наведено на рис. 2.6.



```rust
fn main() {
    let s = String::from("Rust");

    let len = calculate_length(&s);

    println!("The length of {} is {}.", s, len);
}

fn calculate_length(s: &String) -> usize {
    let length = s.len(); // len() returns the length of a String

    return length
}
```

Рисунок 2.6– Передача у борг стрічки s

Активна спільнота розробників робить можливим звернення за допомогою у разі потреби, або отримання документації за інтересуючою тематикою. Велика кількість статей та відео-уроків забезпечує плавний процес навчання, що при необхідності дозволяє виростити спеціалістів з нуля для подальшої підтримки та розробки системи.

2.3.1 Cargo

Ще однією перевагою мови програмування Rust є вбудований інструмент для керування залежностями та збіркою проекту Cargo.

Він дозволяє завчасно описати структуру з залежностями проекту, та інтуїтивно просто, за допомогою інтерфейсу командного рядка, тобто терміналу, побудувати проект, забезпечивши всі необхідні етапи компіляції проекту.

Зручний інструмент який використовується повсюдно до купи з самою мовою програмування, уникаючи необхідності залучати додаткові залежності до проекту.

Приклад роботи з Cargo наведено на рис. 2.7.

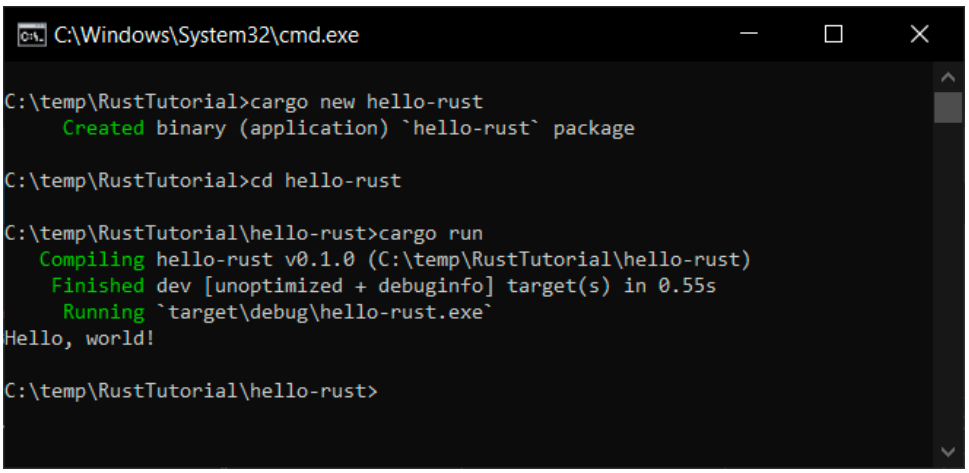


Рисунок 2.7– Робота з Cargo

Уся структура проекту, інформація про нього, залежності та модулі описуються у спеціальних Cargo.toml файлах. Це допомагаю зручно розділяти кодову базу на модулі, що допомагає забезпечити легку читабельність та відносно просту розповсюджуваність.

Приклад Cargo.toml файлу наведено на рис. 2.8.

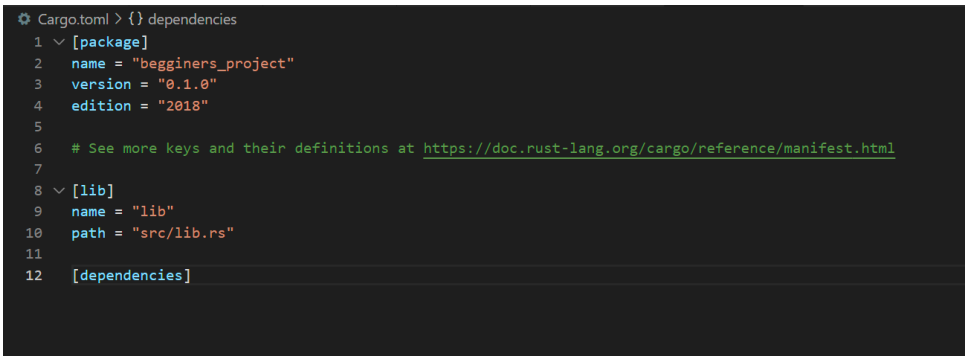


Рисунок 2.8– Опис залежностей усередині файлу Cargo.toml

Цей інструмент, який дозволяє пакетам Rust декларувати різні залежності та гарантувати, що завжди отримуватиметься повторювана збірка. Щоб досягти цієї мети, Cargo робить чотири речі [10]:

- Представляє файли метаданих з різними бітами інформації про пакет.
- Отримує та створює залежності вашого пакета.
- Викликає rustc або інший інструмент збирання з правильними параметрами для створення вашого пакета.

- Представляє конвенції, які полегшують роботу з пакетами Rust.

2.3.2 Фреймворк Actix

Actix — це платформа з серверною візуалізацією. Це означає, що процес створення та обслуговування додатків здійснюється з сервера, коли сторінка запитується користувачем. Архітектура заснована на дуже потужній системі акторів Rust. Вона добре підходить для написів сервісів, які мають логіку більш високого рівня складності.

Асинхронна архітектура, яка закладена в середину даного фреймворку, дозволяє досягати високого рівня продуктивності за рахунок розпалалелювання навантаження між різними потоками.

Така особливість робить його гарним вибором для систем, які очікують на велику кількість одночасних з`єднань від користувачів сервісу, забезпечивши гарну швидкість відгуку та чуйний досвід користування на клієнтській стороні.

Приклад роботи з Actix наведено на рис. 2.9.

```
HttpServer::new(move || {
    let auth = HttpAuthentication::bearer(validate_user);
    App::new()
        .data(data.clone())
        .wrap_fn(|req, srv| {
            sentry_middleware.start(req);
            srv.call(req).map(|res| {
                sentry_middleware.response(req, res);
                sentry_middleware.finish(req, res);
                res
            })
        })
```

Рисунок 2.9– Опис http серверу за допомогою Actix

Також Actix надає велику кількість допоміжних інструментів які мають намір пришвидшити розробку, такі як вбудований маршрутизатор або окреме середовище для обробки вхідних запитів.

Приклад налаштування маршрутизації наведено на рис. 2.10.

```rust
use actix_web::{web, App, HttpRequest, HttpServer, Responder};

async fn greet(req: HttpRequest) -> impl Responder {
    let name = req.match_info().get("name").unwrap_or("World");
    format!("Hello {}!", &name)
}

#[actix_rt::main]
async fn main() -> std::io::Result<()> {
    HttpServer::new(|| {
        App::new()
            .route("/", web::get().to(greet))
            .route("/{name}", web::get().to(greet))
    })
    .bind("127.0.0.1:8000")?
    .run()
    .await
}
```

Рисунок 2.10– Маршрутизація за допомогою Actix

2.3.3 Висновок

В результаті були обрані підходящі інструменти.

Rust у купі з фреймворком Actix вже зарекомендували себе на ринку розробки програмного Вони гарантують такі аспекти системи як надійність, безпечність та масштабованість.

Ці особливості, у купі з перевагами самої мови програмування Rust, роблять цей дует відмінним вибором для розробки backend складової майбутньої системи для проведення соціальних опитувань, дозволивши у короткий термін розробити безпечний та продуктивний веб додаток.

2.4 Інструменти мобільного iOS додатку

Ледве не кожна людина на землі знає що таке “Айфон” і яку операційну систему він має. Кожного року безліч людей чекає на презентацію нової морежі мобільного телефону він компанії Apple.

Не дивно що така велика компанія не захтіла мати свій власний ринок мобільних додатків під свою платформу, зі своїм середовищем розробки та власною мовою програмування.

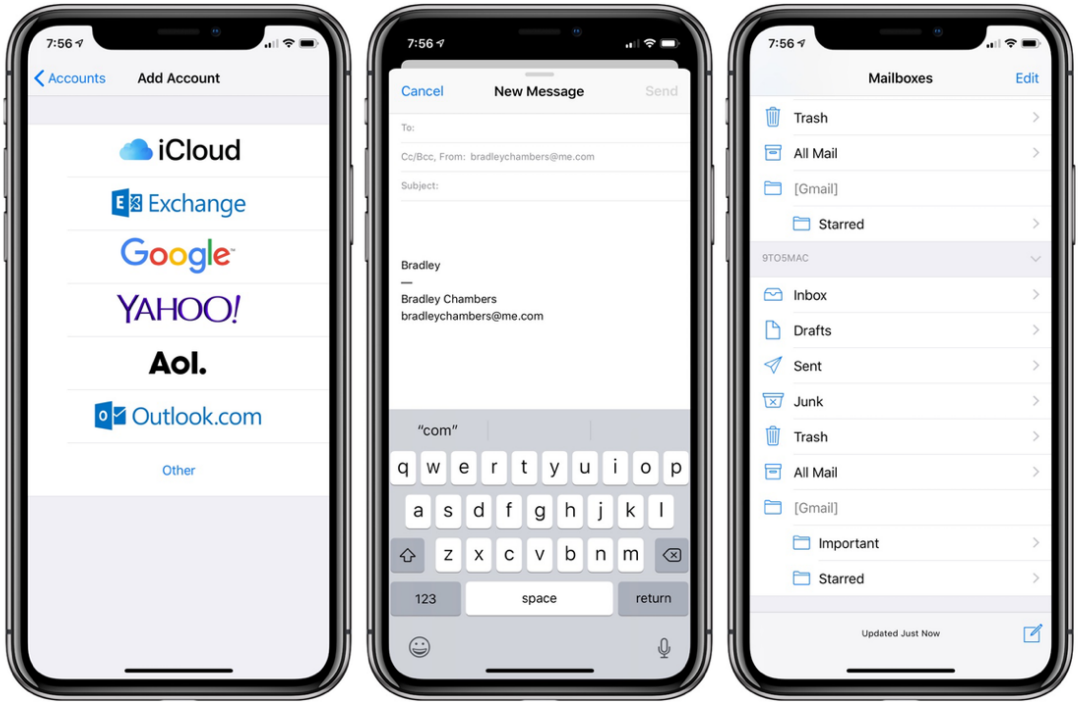


Рисунок 2.11– iPhone з увімкненим додатком ”Mail”

Цей розділ буде описувати основні інструменти які буду задіяні у проектуванні та безпосередній розробці мобільного додатку для проведення соціальний опитувань під платформу iOS.

Майбутній додаток буде тісно пов`язаний з серверною складовою системи, взаємодіючий з нею через доступний інтерфейс, викачуючи або оновлюючи дані, які зберігаються у віддаленій базі даних, для подальшої їх візуальної демонстрації безпосередньому користувачу у найбільш зручному вигляді.

2.4.1 Xcode

Xcode являє собою застосунок який суміщає в собі весь спектр необхідних інструментів для розробки різноманітних застосунків під різні платформи Apple, починаючи з iPhone і. закінчуючи Apple Watch.

Цей продукт був спеціально розроблений компанією Apple, щоб надати стороннім розробникам можливість без зайвих проблем проходити процес від безпосереднього написання коду, і до фінального розміщення та випуску готового продукту на площадках.

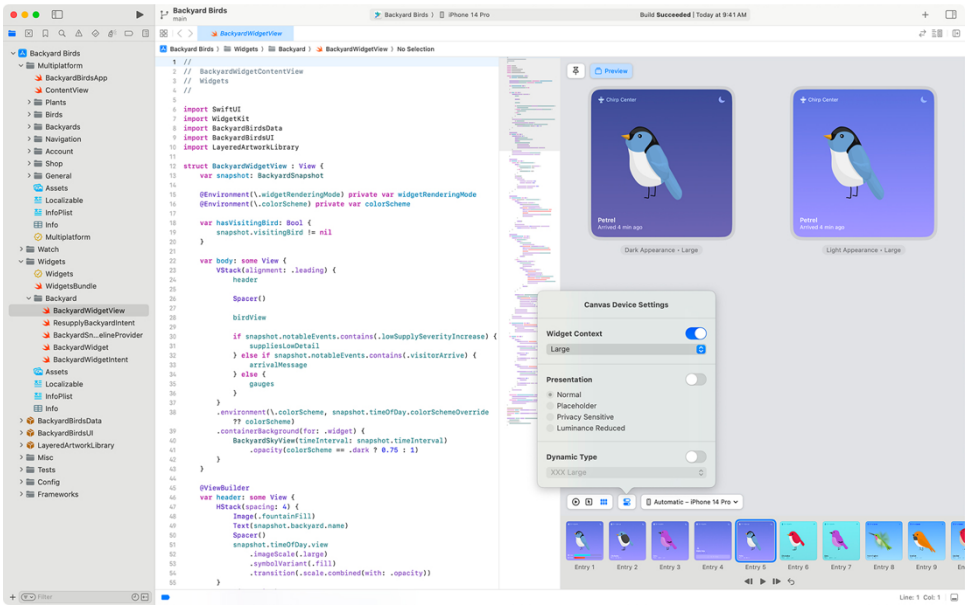


Рисунок 2.11– Інтерфейс Xcode

З моменту свого першого випуску в 2003 році Xcode пройшов довгий шлях. Xcode підтримує багато мов програмування. Серед них є C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez) та Swift, а також різноманітні моделі програмування, такі як Cocoa та Java.

Основні мовами програмування під iOS є вже застаріваючий Objective-C, розроблений на основі неймовірно популярного C, та Swift - інноваційний продукт компанії розроблений з повного нуля з метою покриття недоліків попередника.

Вбудований інтерактивний дизайнер інтерфейсу дозволяє швидко створювати графічний інтерфейс та підлаштовувати його під потреби, з відображенням змін в реальному часі, що можна використовувати як в середині коду, так і взагалі без нього.

Наявність вбудованого емулятора дозволяє протестувати додаток із приближеним до реальної ситуації сценарієм, що дає змогу що програмне забезпечення працює як треба та не трапляється ніяких непередбачуваних ситуацій.

2.4.2 Swift

Swift — це нова мова програмування, розроблена у 2014 році. Має простий синтаксис, що робить його легко читаємий та зрозумілим.

Розробники додали в нього автоматичний механізм підрахунку посилань на об’єкти, який є «збирачем сміття», схожим на той, що є в мові програмування Java, щоб зробити процес розробки більш зручним. Об’єкт вважається потрібним, поки він використовується і лічильник посилань на нього не опускається до нуля. Як тільки це відбувається, об’єкт починає вважатися непотрібним і буде видалено з пам’яті при першій можливості для звільнення місця в пам`яті під подальші потреби.

Однак цей механізм має свої недоліки. Може виникнути ситуація, коли два об'єкти мають взаємні посилання один на одного, створюючи так звану циклічну залежність. У цьому випадку вони залишаються в пам'яті пристрою назавжди, доки програма не буде припинена.

Також був впроваджений механізм для зручного використання, відомий як "optional". Він вирішує проблему покажчиків, які можуть вказувати на неіснуючий об'єкт. Для того щоб вказати компілятору, що значення даної змінної може бути відсутнім у певний момент часу, необхідно додати додатковий символ "?" в кінці типу змінної, наприклад, "Int?".

Приклад коду написаного на мові програмування Swift наведено на рис. 2.12.

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var myLabel: UILabel!

    @IBAction func myButton(sender: AnyObject) {

        myLabel.text = "It worked!"
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

        print("Hello world!")
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

Рисунок 2.12– Код написаний мовою програмування Swift

Swift є мовою програмування з відкритим вихідним кодом, що дозволяє розглядати внутрішні механізми його функціонування та вдосконалювати навички роботи з нею в разі потреби.

Важливо відзначити, що у Swift є можливість використовувати код, написаний на мові програмування Objective-C. Іншими словами, компілятор Swift може обробляти функціонал, створений на Objective-C, і доступ до цього функціоналу можна отримати через власний синтаксис Swift. Важливо зауважити, що така сумісність здійснюється тільки з Objective-C; для інших мов програмування Swift не має такої можливості.

Основною трудністю в Swift є відсутність оберненої сумісності з його попередніми версіями. Це означає, що компілятор Swift версії 3, наприклад, не може успішно зібрати проект, написаний для версії 2. Також не можна продовжувати використовувати старі версії Swift, оскільки нові версії iOS вимагають змін, які вносяться в Swift поступово. Процес переходу кодової бази, особливо якщо вона досить велика, є складним, і це може призвести до виявлення нових помилок, які можуть залишитися непоміченими під час розробки.

Незважаючи на це, дана мова програмування має стати підходящим варіантом для реалізації мобільного додатку під платформу iOS для системи проведення соціальних опитувань, забезпечення належний рівень надійності та подальшої підтримки.

3. РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ МОБІЛЬНОГО ДОДАТКУ

Для того, щоб додаток мав гарний відгук у кінцевого користувача, конче необхідно щоб він мав гарно продуманий дизайн, яким буде зручно користуватися.

Усі дії мають бути послідовні, кроки мають бути прогнозованими та легко доступними. Це забезпечить позитивний відгук у людей котрі будуть користуватися додатком.

Ця продуманість повинна бути викладена у дизайні інтерфейсу, котрий має стати візитною карточкою продукту. Має бути присутність відповідності до сучасних стандартів вигляду та бути привабливим у зовнішньому вигляді.

Для досягнення цієї мети необхідно обрати відповідний інструмент, який забезпечить відповідний рівень можливостей для здійснення ідей у вигляді кінцевого дизайну інтерфейсу.

3.1 Графічний редактор Figma

Figma - це хмарний багатоплатформовий сервіс для дизайнерів інтерфейсів і web-розробників, з яким можна працювати безпосередньо в браузері. І це лише одне з важливих переваг платформи [11].

Надзвичайно популярний інструмент по всьому світу, який використовується багатьма найбільшими компаніями світу, які тільки існують. Це допомагає забезпечитися в його надійності та майбутній підтримці.

Figma виходить за межі звичайного інструменту для створення статичних макетів, надаючи розширені можливості для розробки прототипів та взаємодій у веб-додатках.

Інтерфейс Figma наведено на рис. 3.1.

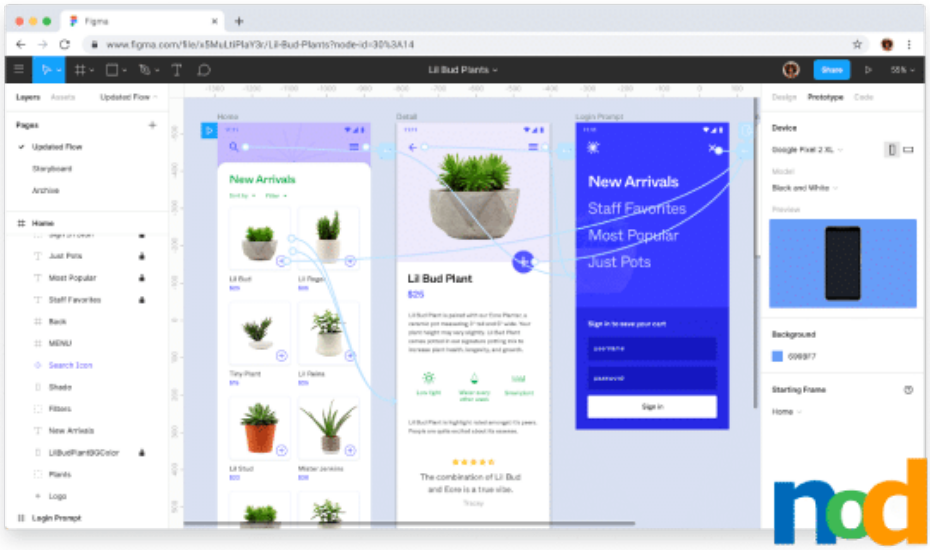


Рисунок 3.1– Інтерфейс Figma

Figma дозволяє в реальному часі працювати декільком людям над одним і тим самим проектом, дозволяючи миттєво бачити зміни, які були внесені іншими. Це сприяє гарній співпраці у середині команди, що прискорює термін готовності продукту.

Зручним інструментом зі сторони розробника є можливість згенерувати готовий код із дизайну лише в 1 клік. Є підтримка різних мов програмування, зокрема необхідної в даному випадку Swift.

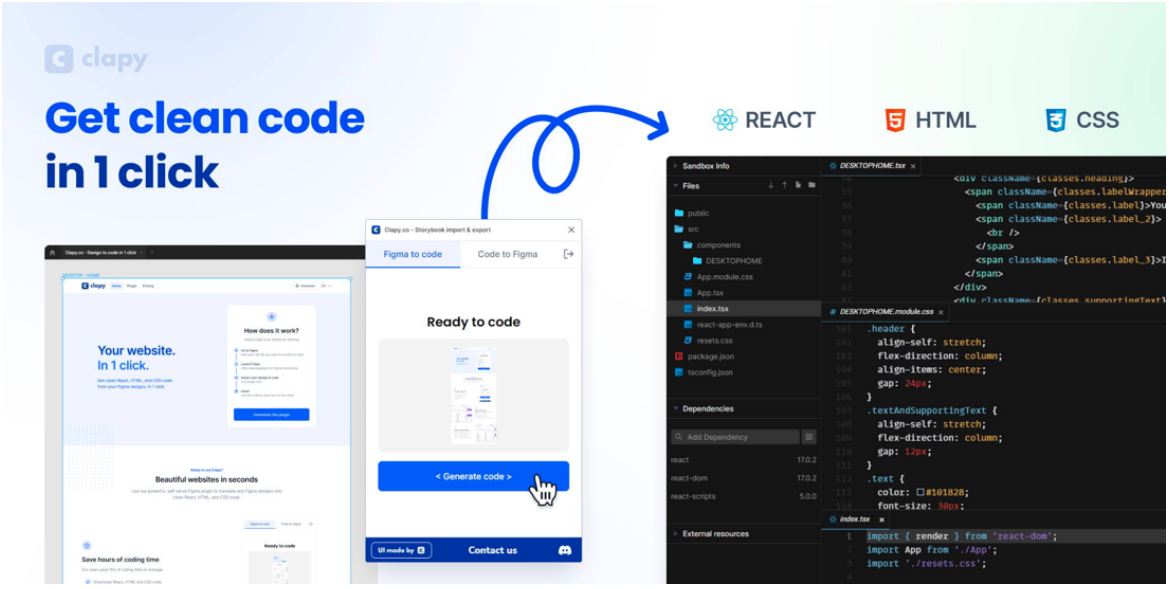


Рисунок 3.2– Figma генерація коду

Можливість протитипування дозволяє створювати системний дизайн з компонентами, які можна пере використовувати, і, відповідно, вносити зміни лише в одному місті, завіюючи їх по всьому проекту. Це робить весь наявний дизайн відповідним теперішньому часу, що пришвидшує загальний термін розробки.

Є в наявності інструменти для зручної розробки різного роду анімацій, не завіюючи при цьому сторонніх програм. Це дозволяє дизайнеру в повній мірі і не відволікаючись втілювати у документі усі свої видумки та побажання.

Гарантується безпека синхронізації роботи та безперебійний обмін даними. Наявна повна історія внесених змін з можливістю повернення до попередніх версій, що зменшує ризики втрати актуальних даних та полегшує загальну ревізію проекту.

Таким чином, Figma надає зручний і надійний спосіб розробки користувальницького інтерфейсу майбутніх додатків для iOS, які повністю відповідають потребам вирішення цієї проблеми.

3.2 Вибір назви додатку

У сучасному цифровому світі мобільні застосунки стали необхідним інструментом для розвитку бізнесу та полегшення нашого повсякденного життя. Як наслідок, конкуренція на цьому ринку надзвичайно велика. Якщо застосунок лишити малопомітним, він просто загубиться серед незліченних альтернатив [12].

Назва має відображати дух додатку, його наміри та те, що він пропонує. Шляхом довгих роздумів був обраний варіант, який найбільше відповідає цим критеріям - “weVote”.

Ця назва відображає концепцію ідеї спільної справи до об`єднання задля її досягнення, підкреслюючи важливість кожного голосу. Вона створює враження групової дії та заохочує користувачів брати участь у процесах прийняття рішень разом, збільшуючи важливість кожного голосу в спільному контексті.

3.3 Розробка користувацького інтерфейсу iOS додатку

Важливим аспектом дизайну є естетичний вигляд аплікації та відповідність її елементів фірмовому стилю. Використання відповідних кольорів, шрифтів і графічних елементів може допомогти підвищити пізнаваність бренду. Забезпечення узгодженості в усій програмі, від екрана привітання до окремих блоків функцій, допомагає створити цілісне та професійне враження.

Також важливим аспектом дизайну є естетичний вигляд додатку та відповідність його елементів фірмовому стилю. Використання відповідних кольорів, шрифтів і графічних елементів може допомогти підвищити пізнаваність бренду. Забезпечення узгодженості в усій програмі, від екрана привітання до окремих блоків функцій, допомагає створити цілісне та професійне враження.

Під час розробки дизайну інтерфейсу використовуються гайдлайни iOS та Android. iOS дизайн відомий своєю естетикою та мінімалізмом. Чисті лінії, прості форми та обмежені кольори сприяють вигляду, який є сучасним і привабливим для користувачів. Кожна операційна система пропонує власні гайдлайни – документи з рекомендаціями для розробників, які займаються проектуванням дизайну додатків. Естетика та мінімалізм є визначальними рисами дизайну в операційній системі iOS. Простота форм, чистота ліній та обмеженість кольорів створюють сучасний і привабливий вигляд, який привертає користувачів. Дотримуючись цих порад, вирішуємо одразу дві задачі: забезпечуємо зручність для користувачів та гарантуємо відповідність застосунку вимогам обраної платформи – все це позитивно впливає на подальше просування [13].

Приклад розробленого дизайну екрану для входу користувача зображено на рис. 3.3.

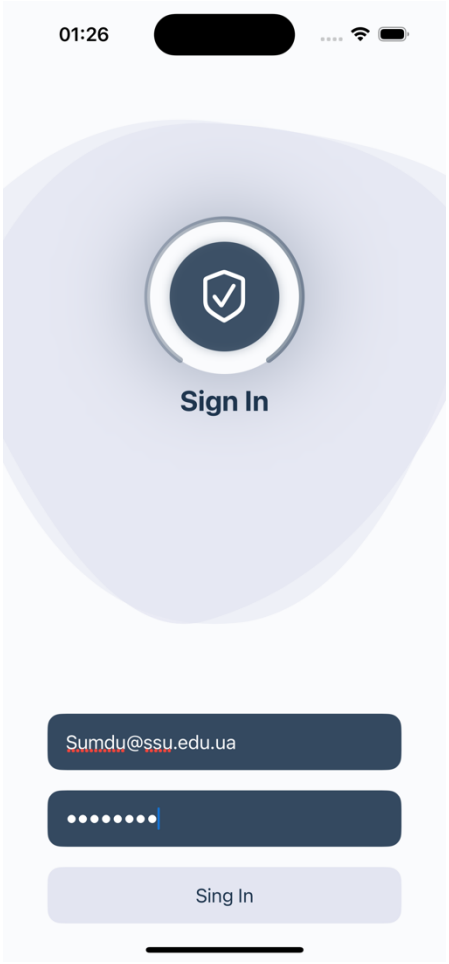


Рисунок 3.3– Екран входу додатку “weVote”

В загалі був розроблений дизайн для наступних екранів:

- Екран для входу користувача
- Екран для реєстрації користувача
- Головний екран зі списком активних, майбутніх або пройдешніх опитувань, з можливістю створи нове
- Екран для створення нового опитування, щоб отримувати результати
- Детальний екран опитування, з відображенням результатів на даний момент
- Екран з детальним відсотковим аналізом вибору користувачів по окремому варіанту опитування

Після формулювання послідовного плану дій розпочалась робота зі створення користувацького інтерфейсу. У результаті цього процесу були розроблені всі необхідні елементи дизайну для належного функціонування додатку.

Дизайн екрану з загальними результатами опитування наведено на рис. 3.3.

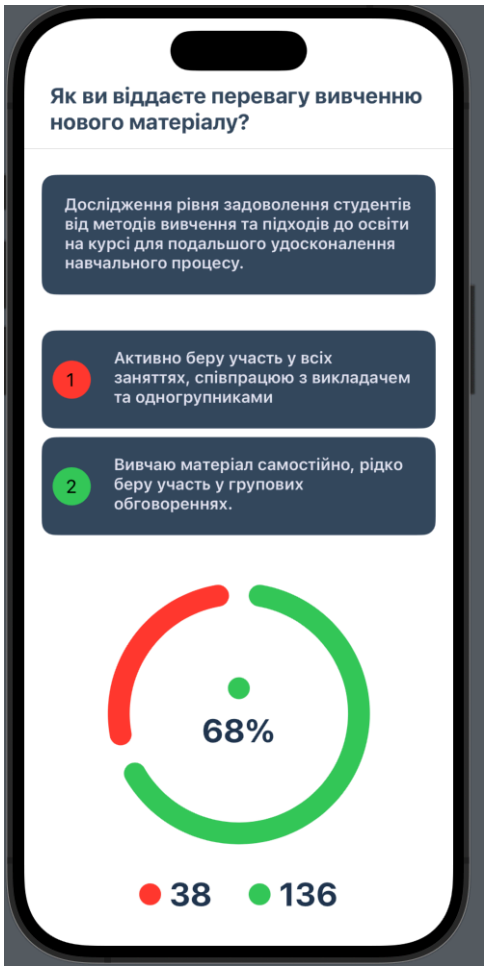


Рисунок 3.3– Екран з загальними результатами опитування додатку “weVote”

4. ТЕОРЕТИЧНІ ЗАСАДИ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Необхідно сегментувати користувачів за типом доступу, щоб правильно розрізняти функціональні можливості програми, доступні кожному користувачеві. Оскільки поточна версія системного клієнта не має таких ролей, як «модератор» або «платна підписка», наразі існує лише два типи користувачів, а саме:

- Гостьовий користувач;
- Авторизований користувач;

Роль користувача визначається тим, чи пройшов він або вона процес авторизації з моменту встановлення програми на його або її персональному пристрої. Якщо програма не може визначити наявні авторизації, вона переходить у режим роботи «Гість» і відображає початкове вікно привітання.

Можливості користувачів у режимі «Гість» наведено на рисунку 4.1.

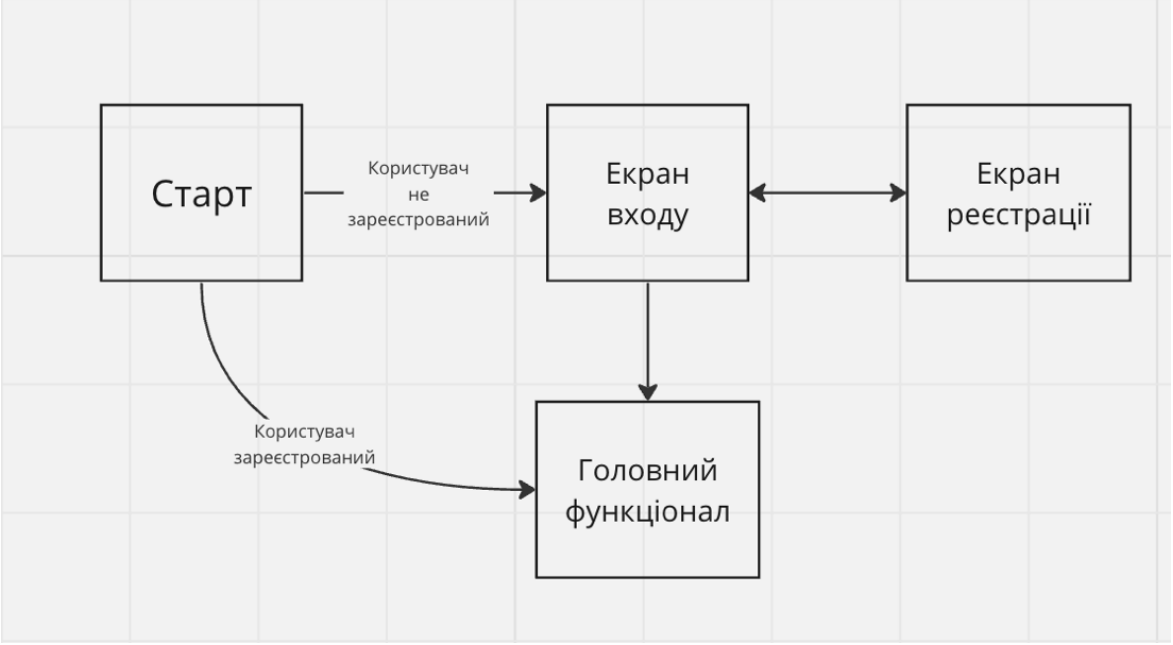


Рисунок 4.1– Навігація користувача при відкритті додатку

Як видно із зображення вище, користувачі з такими правами доступу не можуть отримати доступ до основних функцій системи. Щоб розширити

пропоновані можливості, необхідно скористатися одним із двох доступних способів проходження авторизації.

Можливості користувача в режимі «авторизації» наведено на рис. 4.2.

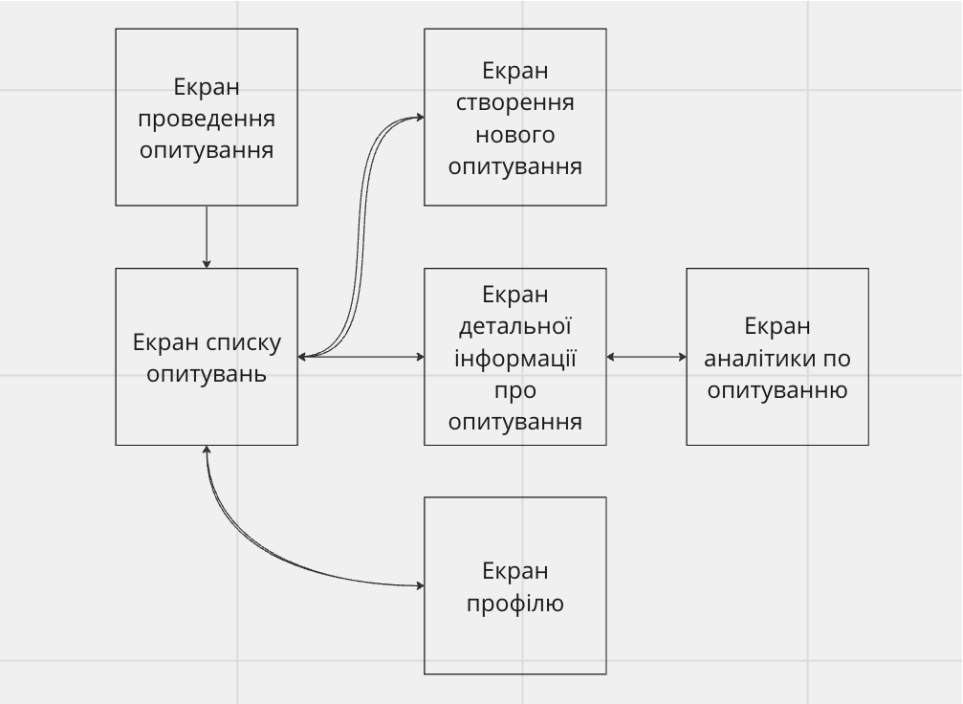


Рисунок 4.2 – Навігація залогованого користувача

4.1 Перенаправлення трафіку на створене опитування

Для зручного перенаправлення користувачів напряму до необхідного опитування на перший час було вирішено використовувати технологію deep link. Це дозволить тим хто створив опитування розміщати у мережі інтернет посилання, яку буде вести потенційного рецензента на сторінку в додатку, де він зможе обрити відповідь на питання.

Що таке deep link та глибоке посилання? Deep link — це посилання на підсторінку, а дослівно кажучи, це глибоке посилання. Розміщується партнерське посилання, і користувач після натискання переміщається на конкретну підсторінку в додатку, уникаючи, таким чином, проходження через головну сторінку, сторінку пошукової системи тощо [14].

Приклад deep link на опитування в додатку weVote наведено на рис. 4.3.

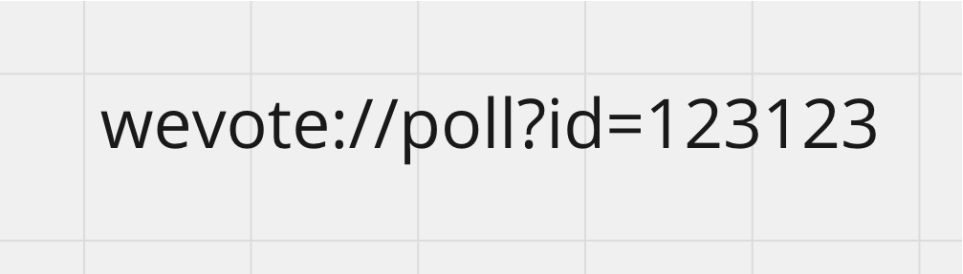


Рисунок 4.3– Внутрішнє посилання на опитування застосунку weVote

4.2 Проектування бази даних

Обрана системою модель «клієнт-сервер» робить клієнта незалежним від конкретного місця розташування бази даних, тобто він може бути розташований як локально, так і віддалено. В результаті конкретного запиту від клієнта всі пов'язані з ним операції відбуваються на стороні сервера. Клієнт також обробляє відповіді, надіслані базою даних, і форматує їх у певний спосіб, який очікує клієнт.

Технологія клієнт-сервер означає такий спосіб взаємодії програмних компонентів, при якому вони утворюють єдину систему. Як видно із назви, існує клієнтський процес, що вимагає певних ресурсів, а також серверний процес, що ці ресурси надає. Зовсім необов'язково, щоб вони перебували на одному комп'ютері. Звичайно прийнято розміщувати сервер на одному з вузлів локальної мережі, а клієнтів – на інших вузлах [15].

Результатом проектування є реалізована база даних, що представляє модель «Сутність – Комунікація». Сутність — це предмет, місце, річ, подія або поняття, що містить інформацію. Точніше, сутність — це сукупність (комбінація) об’єктів, які називаються екземплярами. Кожен екземпляр сутності має набір характеристик. У логічній моделі бази даних усі ці характеристики називаються атрибутами сутності [16].

4.3 Таблиці бази даних

Сутності бази даних weVote:

1. Користувачі.

2. Опитування.
3. Питання.
4. Варіант відповіді на питання.
5. Відповідь користувача у опитуванні.

Ці сутності являють собою набір певних особливостей, які являють собою той чи інший об`єкт. Кожна така сутність має свій певний унікальний ключ, за яким її можна відрізнити від усіх інших, і називають його “первинним”. Такі поля завжди незмінні, оскільки б це призвело до втрати доступу до об`єкта зі сторони клієнта.

Зазвичай такі поля бувають числовими, де кожна наступна сутність має значення більше на одиницю, чим у попередника. Це виключає сценарії коли клієнт на запит за одним ключем може отримати декілька варіантів на вибір, що значно спрощує логіку імплементації зі сторони розробника.

Перелік усіх доступних у межах бази даних сутностей наведено у таблиці 4.1.

Таблиця 4.1 – Список всіх сутностей БД системи «MoveJoin»

Users	Користувачі системи
Surveys	Створені опитування
Questions	Питання до опитувань
Answers	Відповіді до питань опитування
UserAnswers	Відповіді користувачів на опитування

Далі будуть описані усі наявні у системі проведення соціальних опитувань “weVote” таблиць, з повним їх описом, включаючи інформацію про поля, їх типи та загальне призначення. Якщо поля у різних таблицях мають однакову назву, то це означає що вони є зовнішніми ключами, і існує певний зв`язок між цими таблицями.

Зовнішній ключ (FOREIGN KEY) потрібен для того, щоб зв’язати дві різні таблиці між собою. Зовнішній ключ може посилатися на будь-який стовпець у батьківській таблиці. Проте загальноприйнятою практикою є посилання зовнішнього ключа на первинний ключ (primary key) батьківської таблиці [17].

Відомості про таблиці бази даних системи “weVote” наведено на у вигляді таблиць (табл.4.2-4.7.).

Таблиця 4.2 - Users

Поле	**Атрибут**	**Тип**
user_id	Ідентифікатор	SERIAL
username	Назва	VARCHAR(50)
email	Назва	VARCHAR(100)
password_hash	Назва	VARCHAR(100)
first_name	Назва	VARCHAR(50)
last_name	Назва	VARCHAR(50)
gender	Назва	VARCHAR(10)
date_of_birth	Назва	DATE
registration_date	Тип	TIMESTAMP

Таблиця 4.3 - Surveys

Поле	**Атрибут**	**Тип**
survey_id	Ідентифікатор	SERIAL
title	Назва	VARCHAR(255)
description	Тип	TEXT
creation_date	Тип	TIMESTAMP
creator_user_id	Ідентифікатор	INT

Таблиця 4.4 - Questions

Поле	**Атрибут**	**Тип**

question_id	Ідентифікатор	SERIAL
survey_id	Ідентифікатор	INT
question_text	Тип	TEXT
question_type	Тип	VARCHAR(50)

Таблиця 4.5 - Answers

Поле	**Атрибут**	**Тип**
answer_id	Ідентифікатор	SERIAL
question_id	Ідентифікатор	INT
answer_text	Тип	TEXT

Таблиця 4.5 - UsersAnswers

Поле	**Атрибут**	**Тип**
user_answer_id	Ідентифікатор	SERIAL
user_id	Ідентифікатор	INT
survey_id	Ідентифікатор	INT
question_id	Ідентифікатор	INT
answer_id	Ідентифікатор	INT
submission_date	Тип	TIMESTAMP

Дані таблиці задовольняють усім потребам системи для проведення соціальних опитувань. Вони дозволяють почати розробку серверної частини яка забезпечить усю логіку зберігання та отримання інформації з них.

Зв`язок між таблицями наведено на рис. 4.4.

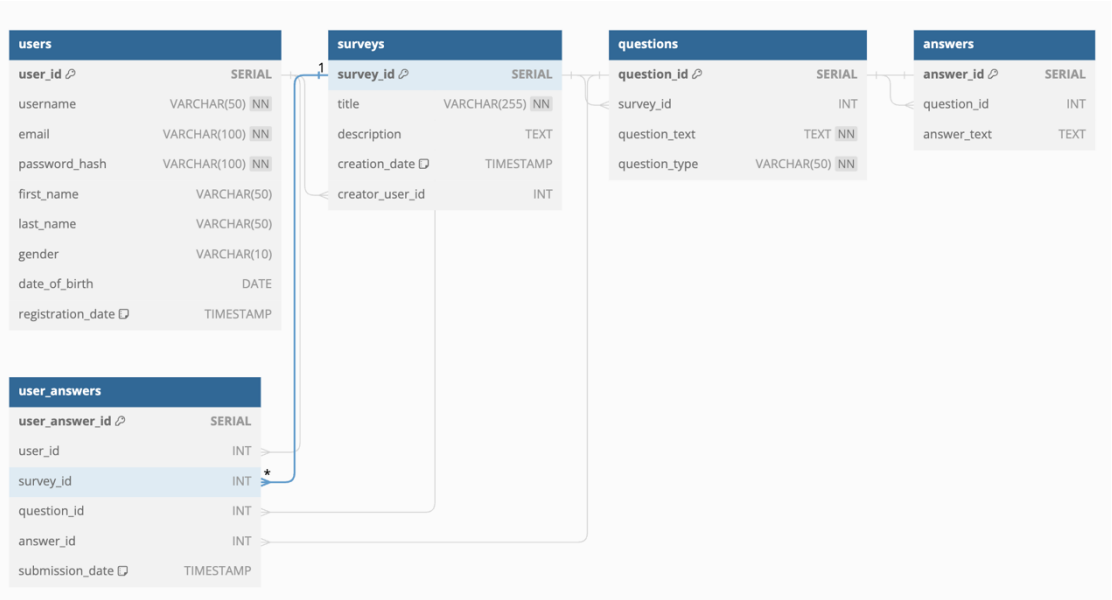


Рисунок 4.4– Зв`язок між таблицями

Рисунок надає викладений огляд структури бази даних, розробленої для системи проведення соціальних опитувань.

5. ПРОГРАМНА РЕАЛІЗАЦІЯ

5.1 Принцип роботи системи

Оскільки наша система є клієнт-серверною, то кожна її частина відповідає за свої певні задачі відведені для неї.

При налаштуванні серверної частини, вона встановлює зв`язок з базою даних для безперебійного потоку передачі даних. Отримавши запит від клієнта, сервер обробляє та авторизує його перевіряючи чи виконані всі умови, для надання йому доступу, та виконує певний набір дій повертаючи відповідну відповідь назад до користувача.

Задача ж клієнта, це зробити необхідні запити до сервера, та, в залежності від результату, відобразити необхідну інформацію на екрані користувача. Також необхідно чітко обробляти вхідні команди від нього, розуміючи які дії необхідно виконати, в результаті на натискання в тому чи іншому місці додатку, та завжди надавати зворотню відповідь, вказуючи на те, чи дія відбулась успішно.

5.2 Навігація всередині iOS додатку

При першому відкритті додатку користувач не є авторизованим, тому ми не можемо надали йому доступ до наявних соціальних опитувань. Для того щоб надати йому змогу авторизуватися, пропонується ввести дані від його існуючого акаунту, або створити новий. Все це здійснюється за допомогою екрану реєстрації.

Є одним з ключових етапів навігації необхідних для подальшого користування додатком.

Екран реєстрації наведено на рис. 5.1.

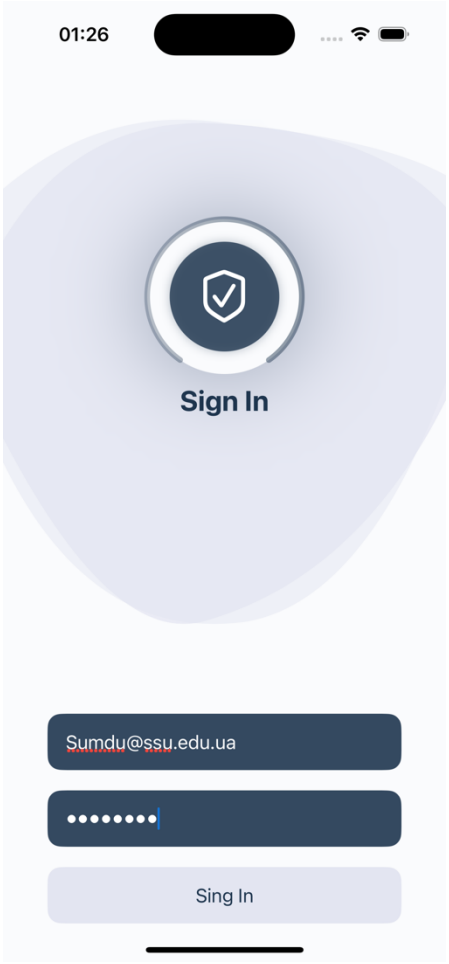


Рисунок 5.1– Екран реєстрації додатку weVote

5.2.1 Екран зі списком створених опитувань

Якщо користувач успішно пройшов цей етап, то його буде направлено далі у основний функціонал додатку, а саме на екран з переліком опитувань, які він створив, що для нового користувача завжди буде порожній, адже він ще не встиг їх створити.

У самому горі екран наявна кнопка для створення нового опитування. Її задача полягає в тому, що при натисканні воно перенаправляє на новий екран для створення опитування.

Нижче наявний перемикач між різними типами опитування, а саме за їх датою початку та кінця. За замовчуванням вибрана опція “Сьогодні”. В цьому випадку у списку будуть відображені лише ті опитування, які або починаються сьогодні, або вже йдуть. Вибравши опцію “У майбутньому” користувач побачить лише ті варіанти, які ще не почались, тобто неактивні. “Закінчені” ж

є свого роду архів опитувань, щоб можна було завжди переглянути результати минулих опитувань, та порівняти їх з теперішніми за потреби.

У центрі екрану наявний головний його елемент - список з опитуваннями відповідно до вибраної часової опції. Сам список закладається з окремих блоків, кожен з яких відображає відповідне опитування.

З корисної інформації на цих блоках можна побачити назву опитування, час його закінчення, частину детального опису, яка вмістилася, та кількість людей, які вже встигли взяти участь.

Приклад екрану зі створеними опитуваннями наведено на рис. 5.2.

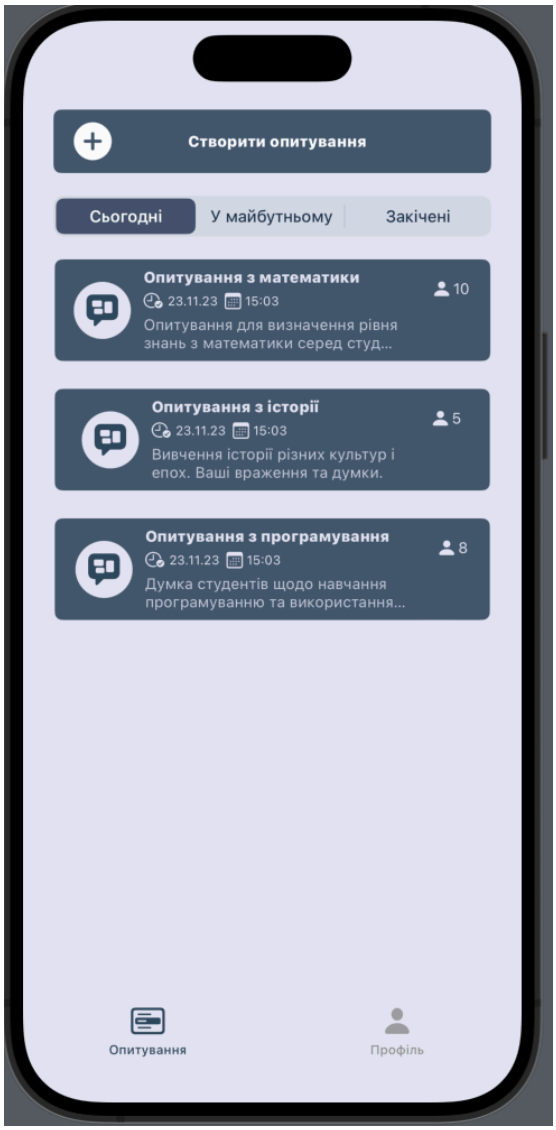


Рисунок 5.2– Екран з опитуваннями додатку weVote

5.2.2 Екран створення опитування

Цей екран відповідає за створення нового опитування. Для вирішення поставленої задачі він надає декілька текстових полів для того, щоб користувач міг надати власну інформацію.

Для початку пропонується ввести текст самого опитування, та, за потреби, короткий опис, який має на меті пояснити що саме мається на увазі у питанні, або проявляючи світло на окремі його аспекти.

Далі є можливість надати необмежену кількість варіантів відповідей. Натиснення на кнопку з іконкою плюса призводить до появи нового текстового блоку, де користувач може ввести довільний текст.

Останній блок відповідає за введення часових обмежень, а саме дат початку та кінця опитування.

Приклад екрану створення опитування наведено на рис. 5.3.

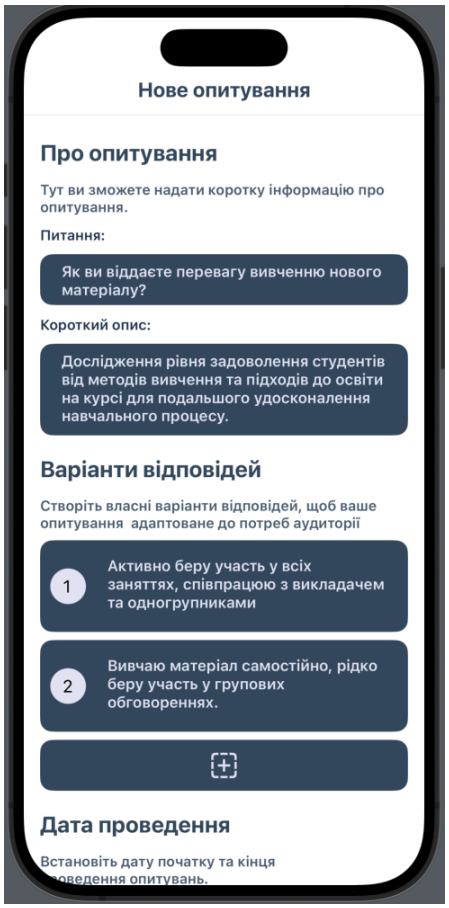


Рисунок 5.3– Екран створення опитування weVote

5.2.3 Поширення deep link

Після успішного створення опитування йому буде присвоєно ідентифікаційний номер, та, за його допомогою, згенеровано посилання deep link.

Приклад такої deep link наведено на рис. 5.4.

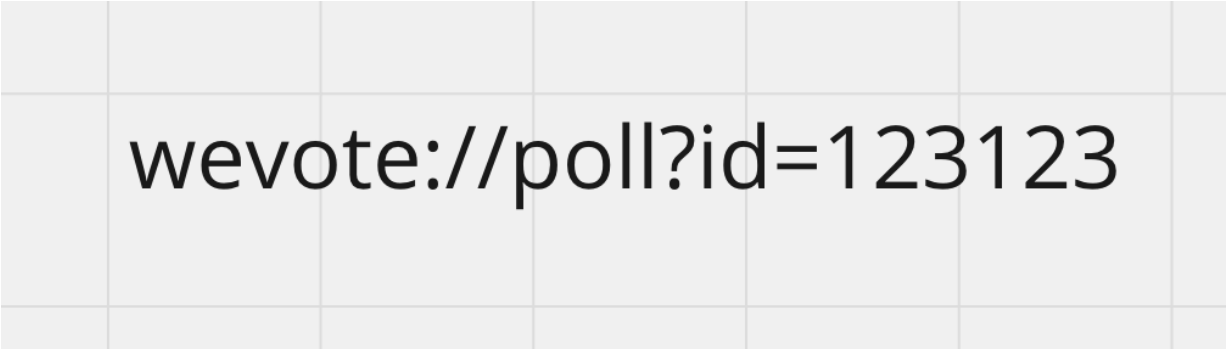


Рисунок 5.4– Deep link на опитування weVote

Воно буде автоматично скопійовано до буферу обміну на пристрої, про що користувача буде проінформовано за допомогою спливаючого вікна.

5.2.4 Екран проведення опитування

Перейшовши за deep link посиланням користувач опиняється на екрані проведення опитування.

Там він може побачити усю основну інформацію про саме опитування, та наявні варіанти вибору. За замовчування жодне з них не вибране, і поки людина не натисне на одне з них, кнопка вибору не буде активна, роблячи неможливою подальшу навігацію.

Обравши варіант користувач побачить спливаюче вікно, яке доповість що голос був врахований, та перенаправить на головну сторінку додатку.

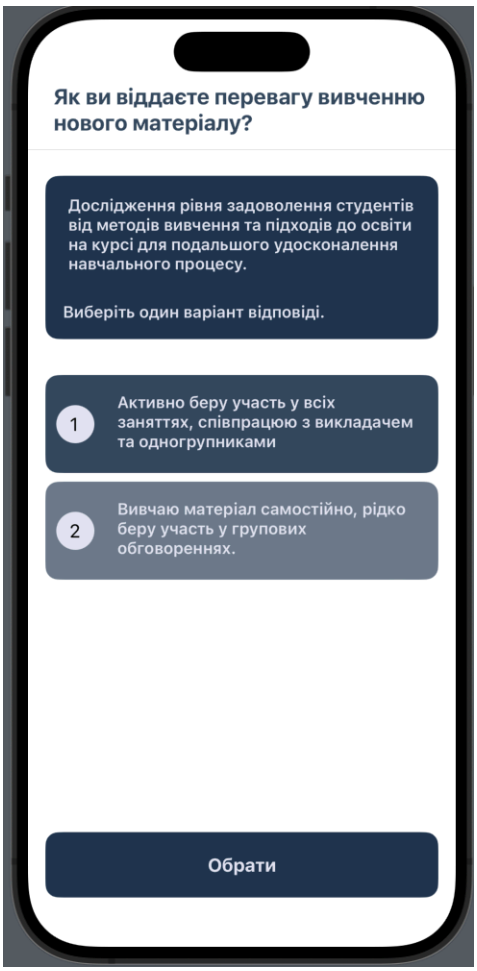


Рисунок 5.5– Екран проведення опитування weVote

5.2.5 Детальний екран опитування

Натиснувши на будь яку опитування на головному екрані можна потрапити на наступний етап, де детально описано хід проведення.

Кожному варіанту відповіді надається свій певний колір. Він відображається на номері, відповідно до кожного з наявних варіантів. Це робиться для того, щоб можливо було побудувати візуально сприятливий графік, де у відсотковому відношенні показується результат проведення опитування.

В центрі кола завжди варіант з найбільшою кількістю виборів.

В низу екрану надається точна інформація про те, скільки людей обрало той чи інший варіант відповіді.

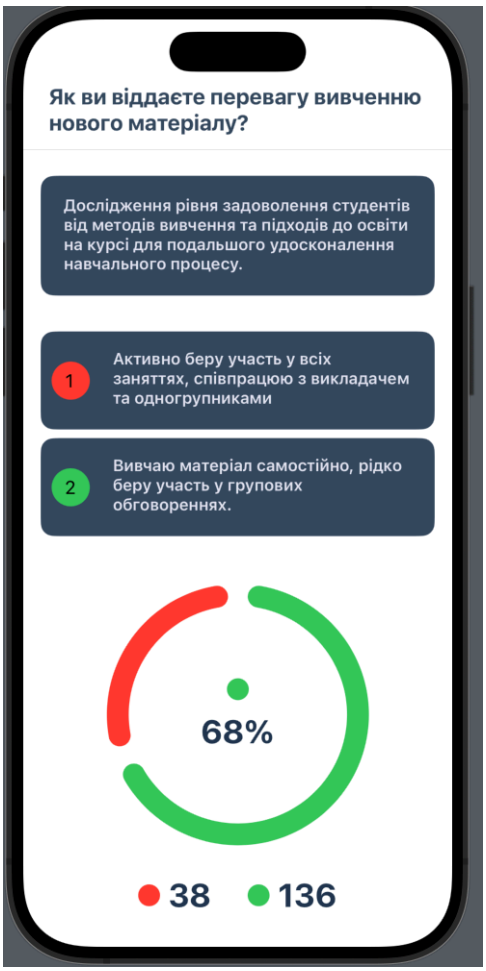


Рисунок 5.6– Екран деталей опитування weVote

5.2.6 Екран розширеної аналітики опитування

Останнім можливим кроком для користувача є можливість перейти на екран розширеної аналітики, де візуально відображається результат проведеного опитування.

Це можна здійснити натиснувши на один з варіантів вибору на попередньому екрані, що призведене до перенаправлення на екран з детальною аналітикою по вибору.

На даний момент представлена лише вибірка по статі та віковій категорії опитуваних. Спочатку в графічному вигляді наводиться відсоткове співвідношення, а в низу чітке кількісне відображення по кожній з наявних груп.

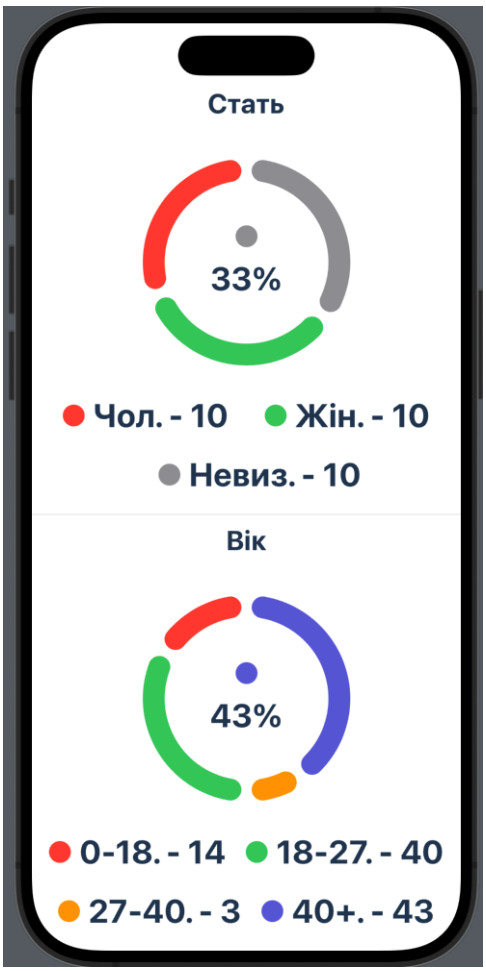


Рисунок 5.7– Екран розширеної аналітики weVote

Це останній з наявних на даний момент у додатку екранів, до яких має доступ користувач.

ВИСНОВКИ

Під час виконання переддипломної практики були виконані наступні етапи розробки системи для пошуку та організації колективних фізичних заходів:

1. Сформульовано основну тематику роботи та обґрунтовано її необхідність та відповідність завданню.

2. Здійснено літературний аналіз, спрямований на вивчення актуальних джерел за темою роботи, результати якого були включені в висновки.

3. Обрано необхідні інструменти для розробки кожного компонента системи, враховуючи їх ефективність та відповідність поставленим завданням.

4. Розроблено інтерфейс користувача для iOS додатку з використанням редактора Figma, надаючи візуальне відображення структури та елементів додатку.

5. Проведено проектування бази даних системи, визначено структуру та взаємозв'язки між її складовими частинами.

6. Реалізовано програмне забезпечення для всіх компонентів системи, забезпечуючи їх взаємодію та відповідність визначеним функціональним вимогам.

Цей підхід дозволив систематично та поетапно розвивати проект, враховуючи ключові аспекти його реалізації та забезпечуючи повноцінну функціональність розробленої системи.

СПИСОК ЛІТЕРАТУРИ

1. МЕТОДИ ДОСЛІДЖЕНЬ У СОЦІАЛЬНІЙ РОБОТІ [Електронний ресурс] – Режим доступу до ресурсу: https://www.uzhnu.edu.ua/uk/infocentre/get/42750.

2. Основні методи проведення соціологічного дослідження. Соціологічний аналіз та використання його результатів. [Електронний ресурс] – Режим доступу до ресурсу: http://nkkep.com/wp-content/uploads/2020/11/P-31P-32-Sots-25.11.pdf.

3. SurveyMonkey [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/url?sa=i&url=https%3A%2F%2Fsoftvn.vn%2Fsan-pham%2Fban-quyen-phan-mem-surveymonkey%2F&psig=AOvVaw3okWwaKVidz-8cLXpY6zWX&ust=1701084745290000&source=images&cd=vfe&opi=89978449&ved=0CBMQjhxqFwoTCJDJuNLI4YIDFQAAAAAdAAAAABAV

4. Redesigned and Responsive Polls at Poll.fm [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/url?sa=i&url=https%3A%2F%2Fcrowdsignal.com%2F2019%2F08%2F29%2Fredesigned-and-responsive-polls-at-poll-fm%2F&psig=AOvVaw2qOCGQMbYRvWQsOkccpraD&ust=1701086246689000&source=images&cd=vfe&opi=89978449&ved=0CBMQjhxqFwoTCPjWn57O4YIDFQAAAAAdAAAAABAE

5. PHP + MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://sites.znu.edu.ua/webprog/lect/1222.ukr.html

6. БАЗИ ДАНИХ. POSTGRESQL [Електронний ресурс] – Режим доступу до ресурсу: https://edu.cbsystematics.com/ua/courses/postgresql

7. This is my process to improve the performance of my PostgreSQL database and queries [Електронний ресурс] – Режим доступу до ресурсу: https://www.stevencanplan.com/2020/08/this-is-my-process-to-improve-the-performance-of-my-postgresql-database-and-queries/

8. JSON [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/help/objc/json.html

9. Protobuf [Електронний ресурс] – Режим доступу до ресурсу: https://protobuf.dev/overview/

10. The Cargo Book [Електронний ресурс] – Режим доступу до ресурсу: https://doc.rust-lang.org/cargo/guide/why-cargo-exists.html

11. Що таке Figma: функції, інструменти та переваги [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.academy/ua/chto-takoe-figma-funktsii-instrumenty-ipreimuschestva/

12. Просування мобільних додатків в 2023 — Як розкрутити мобільний застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://netpeak.net/uk/blog/prosuvannya-mobil-nikh-zastosunkiv-v-2023-yak-rozkrutiti-mobil-niy-zastosunok/

13. ДИЗАЙН МОБІЛЬНИХ ДОДАТКІВ [Електронний ресурс] – Режим доступу до ресурсу: https://kitapp.pro/uk/rozrobka-dizajnu-dodatkiv/

14. ПАРТНЕРСЬКІ ПОСИЛАННЯ ТА DEEPLINK. ВІД САМОГО ПОЧАТКУ. ЩО ЦЕ? [Електронний ресурс] – Режим доступу до ресурсу: https://mylead.global/uk/blog/what-is-deep-link-ua

15. Сучасні клієнт-серверні технології та їх застосування при вивченні систем управління базами даних [Електронний ресурс] – Режим доступу до ресурсу: https://fi.npu.edu.ua/files/Zbirnik_KOSN/12/29.pdf

16. Основи автоматизованого проектування складних об’єктів і систем [Електронний ресурс] – Режим доступу до ресурсу: http://biblio.umsf.dp.ua/xmlui/bitstream/handle/123456789/4330/Навч_пос_ОАП_10_2018.pdf?sequence=1&isAllowed=y

17. Зовнішній ключ (FOREIGN KEY) в SQL [Електронний ресурс] – Режим доступу до ресурсу: https://acode.com.ua/foreign-key-sql/

ДОДАТОК А ГРАФІЧНЕ ЗОБРАЖЕННЯ БАЗИ ДАНИХ СИСТЕМИ

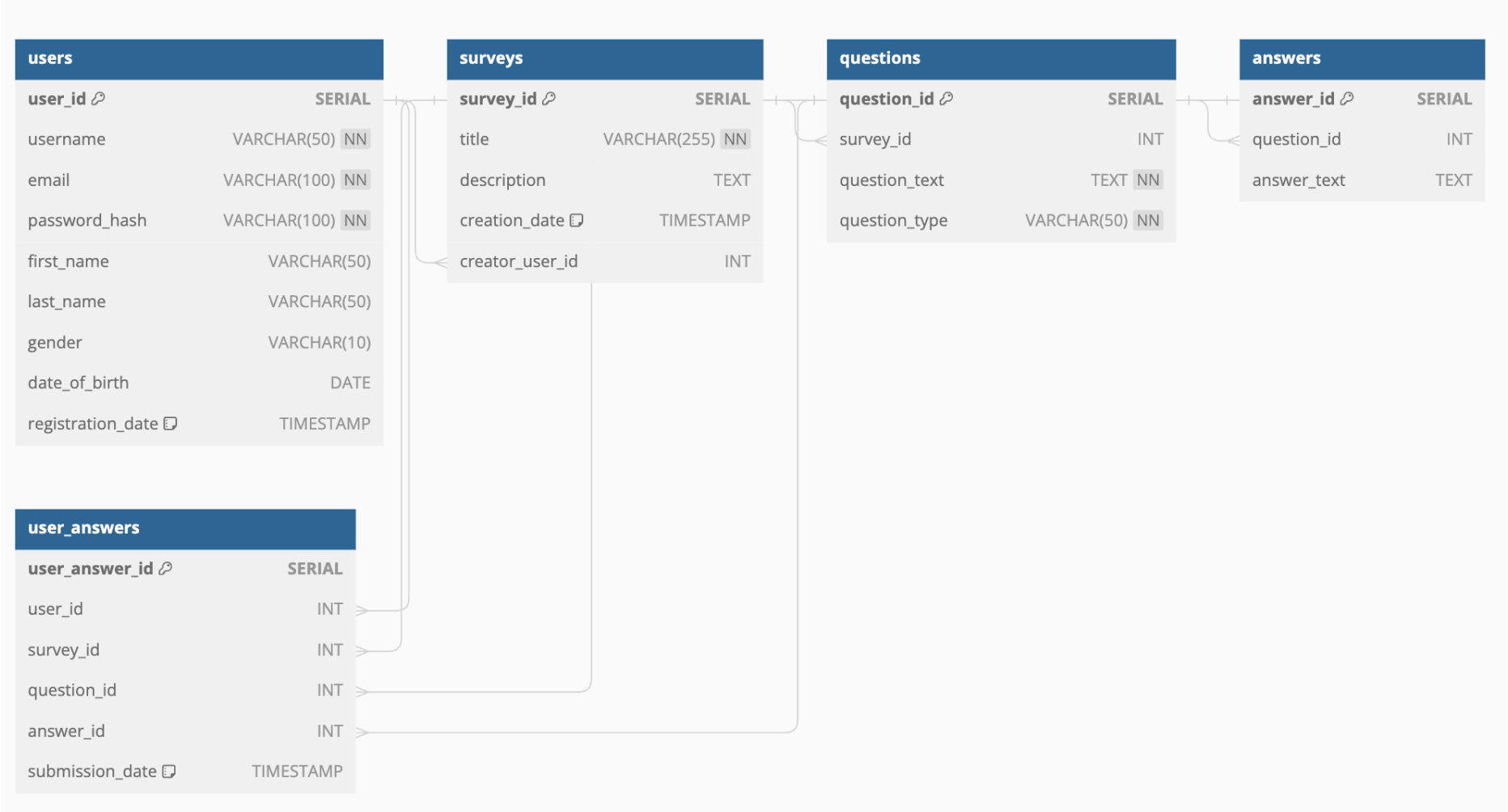


Рисунок А.1 – Відношення таблиць бази даних системи «weVote»

ДОДАТОК Б ОСНОВНІ КОДИ ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ

```
// SurveyListViewController.swift

import UIKit

class SurveyListViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    let surveys: [Survey] = []
    let tableView = UITableView()

    init(surveys: [Survey]) {
        self.surveys = surveys
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // Налаштування UITableView
        tableView.delegate = self
        tableView.dataSource = self
        tableView.register(SurveyTableViewCell.self, forCellReuseIdentifier: "SurveyCell")

        // Додавання UITableView на екран
        view.addSubview(tableView)

        // Налаштування констрейнтів
        tableView.translatesAutoresizingMaskIntoConstraints = false
        NSLayoutConstraint.activate([
            tableView.topAnchor.constraint(equalTo: view.topAnchor),
            tableView.leadingAnchor.constraint(equalTo: view.leadingAnchor),
            tableView.trailingAnchor.constraint(equalTo: view.trailingAnchor),
            tableView.bottomAnchor.constraint(equalTo: view.bottomAnchor)
        ])
    }

    // MARK: - UITableViewDataSource
```

```
    func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        return surveys.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
-> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "SurveyCell",
for: indexPath) as! SurveyTableViewCell
        let survey = surveys[indexPath.row]
        cell.configure(with: survey)
        return cell
    }
}

// Клас для представлення даних опитувань в UITableViewCell
class SurveyTableViewCell: UITableViewCell {

    let titleLabel = UILabel()
    let descriptionLabel = UILabel()
    let participantsLabel = UILabel()
    let endDateLabel = UILabel()

    override init(style: UITableViewCell.CellStyle, reuseIdentifier: String?)
{
        super.init(style: style, reuseIdentifier: reuseIdentifier)

        // Налаштування UI елементів
        titleLabel.font = UIFont.boldSystemFont(ofSize: 17)
        descriptionLabel.textColor = UIColor.darkGray
        participantsLabel.textColor = UIColor.gray
        endDateLabel.textColor = UIColor.gray

        // Додавання UI елементів на content view
        contentView.addSubview(titleLabel)
        contentView.addSubview(descriptionLabel)
        contentView.addSubview(participantsLabel)
        contentView.addSubview(endDateLabel)

        // Налаштування констрейнтів
        titleLabel.translatesAutoresizingMaskIntoConstraints = false
        descriptionLabel.translatesAutoresizingMaskIntoConstraints = false
        participantsLabel.translatesAutoresizingMaskIntoConstraints = false
```

```
        endDateLabel.translatesAutoresizingMaskIntoConstraints = false

        NSLayoutConstraint.activate([
            titleLabel.topAnchor.constraint(equalTo:    contentView.topAnchor,
constant: 8),
            titleLabel.leadingAnchor.constraint(equalTo:
contentView.leadingAnchor, constant: 16),

            descriptionLabel.topAnchor.constraint(equalTo:
titleLabel.bottomAnchor, constant: 4),
            descriptionLabel.leadingAnchor.constraint(equalTo:
contentView.leadingAnchor, constant: 16),

            participantsLabel.topAnchor.constraint(equalTo:
descriptionLabel.bottomAnchor, constant: 4),
            participantsLabel.leadingAnchor.constraint(equalTo:
contentView.leadingAnchor, constant: 16),

            endDateLabel.topAnchor.constraint(equalTo:
participantsLabel.bottomAnchor, constant: 4),
            endDateLabel.leadingAnchor.constraint(equalTo:
contentView.leadingAnchor, constant: 16),
            endDateLabel.bottomAnchor.constraint(equalTo:
contentView.bottomAnchor, constant: -8)
        ])
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    func configure(with survey: Survey) {
        titleLabel.text = survey.title
        descriptionLabel.text = survey.description
        participantsLabel.text = "Учасників: \(survey.participants)"

        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "dd.MM.yyyy"
        endDateLabel.text     =     "Кінець:     \(dateFormatter.string(from:
survey.endDate))"
    }
}
```

```
// Клас для представлення даних опитувань
struct Survey {
    let title: String
    let description: String
    let endDate: Date
    let participants: Int
}

// App.rs

// src/db.rs
use sqlx::postgres::{PgPool, PgPoolOptions};
use sqlx::Result;
use crate::models::Survey;

pub async fn get_surveys(pool: &PgPool) -> Result<Vec<Survey>> {
    let surveys = sqlx::query_as::<_, Survey>("SELECT * FROM survey")
        .fetch_all(pool)
        .await?;

    Ok(surveys)
}

// src/main.rs
use actix_web::{get, web, App, HttpServer, Responder};
use sqlx::postgres::PgPoolOptions;

mod db;
mod models;

#[get("/surveys")]
async fn get_surveys(db_pool: web::Data<sqlx::PgPool>) -> impl Responder {
    let surveys = db::get_surveys(&db_pool).await;

    match surveys {
        Ok(surveys) => web::Json(surveys),
        Err(_) => web::HttpResponse::InternalServerError().finish(),
    }
}

#[actix_web::main]
async fn main() -> std::io::Result<()> {
    let db_url = "postgres://username:password@localhost/database_name";
```

```
    let db_pool = PgPoolOptions::new()
        .max_connections(5)
        .connect(&db_url)
        .await
        .expect("Failed to connect to the database");

    HttpServer::new(move || {
        App::new()
            .data(db_pool.clone())
            .service(get_surveys)
    })
    .bind("127.0.0.1:8080")?
    .run()
    .await
}
```