МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ГНАТЮК МАКСИМ АНДРІЙОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____Т. В. Нескородєва
_____
«____»_____ 20__р.

СИСТЕМА ТЕСТУВАННЯ ЗНАНЬ ПІД ПЛАТФОРМУ ANDROID

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (бакалаврська) робота

Науковий керівник:
Антонов Ю.С., доцент кафедри
інформаційних технологій
к. фіз.-мат. н., доцент

Оцінка: ___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця - 2021

Гнатюк М.А. Система тестування знань під платформу Android. Спеціальність 122 «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У даній кваліфікаційній роботі розглянуто створення системи оцінювання та тестування знань. Так як популярність мобільних пристроїв стрімко зростає, а пандемія диктує свої правила дистанційного навчання, то освітлена тема є актуальною.

У вступі наведено актуальність розробки під мобільну платформу Android. Наведені ключові переваги у відношенні інших мобільних платформ.

У другому розділі розглянуто інструменти та технології, які полегшили створення даної системи.

У третьому розділі проводиться опис користувацького інтерфейсу, реалізації системи зі сторони серверу та зі сторони клієнта, платформи Android.

Ключові слова: система оцінювання, Android, Kotlin, мобільний додаток, Kotlin Server Side.

Gnatyuk M.A. Knowledge testing system for the Android platform. Specialty 122 "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia, 2021.

In this qualification work the creation of a system of assessment and testing of knowledge is considered. As the popularity of mobile devices grows rapidly, and the pandemic dictates its own rules of distance learning, the covered topic is relevant.

The introduction shows the relevance of development for the Android mobile platform. The key advantages over other mobile platforms are given.

The second section discusses the tools and technologies that have facilitated the creation of this system.

The third section describes the user interface, system implementation on the server side and on the client side, the Android platform.

Keywords: evaluation system, Android, Kotlin, mobile application, Kotlin Server Side.

ЗМІСТ

ВСТУП

Сьогодні неможливо уявити людину, яка б не користувалась мобільним телефоном. Ще 25 років тому ці простої були надзвичайною рідкістю, натомість на даний момент являються головним засобом комунікації між людьми. Коротко кажучи, за такий невеликий проміжок часу смартфони стали своєрідною мініатюрною копією комп'ютера, який можна завтра мати при собі [1].

Більшість смартфонів, комунікаторів, планшетних ПК і інших видів пристроїв на стан 2021 року випускаються на базі ОС Android [2]. Давайте розглянемо основні причини популярності даної операційної системи:

1. Підтримка великої кількості пристроїв різних виробників
2. Відкритий код системи, що дає змогу на вдосконалення та розширення функціоналу
3. Доступність. Ринок Android пристроїв досить великий та різноманітний, що дозволяє обрати девайс саме під потреби користувача.
4. Спільнота користувачів. Android має безліч абонентів по всьому світу, в наслідок чого існує багато форумів, де ви можете знайти потрібно інформацію у разі виникнення запитань чи проблем у використанні.

У рамках курсової роботи ціль полягає у наступному:

1. Поглиблене вивчення екосистеми пристроїв на базі Android та застосування на практиці пакету Android SDK.
2. Огляді існуючих Android додатків за схожою тематикою, виявлення їх переваг та недоліків.
3. Написання власної серверної частини та її запуск на віддаленому пристрої.
4. Безпосередньо написання системи оцінювання та тестування знань.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Ринок мобільних додатків зростає із неймовірними темпами. Пустопово сфера мобільної розробки витісняє веб платформу. Різко збільшилась кількість розробників мобільних додатків, б'є рекорди кількість самих додатків. Дохід, який створюється індустрією мобільних додатків, досяг позахмарних показників.

З початку 2020 року епідемія коронавіруса захопила весь світ, в тому числі і Україну [5]. Пандемія вплинула на всі сфери життя, зуміла внести свої корективи у світову економіку, бізнес, туризм, навчання. Більшість офісних працівників, учнів, студентів були вимушені перейти на дистанційну форму роботи та навчання відповідно. Якщо раніше була потреба перевірити знання працівників чи учнів, то це можна було зробити в аудиторії за допомогою бланків з питаннями, але зараз як наслідок дистанційної форми роботи та навчання стало актуальним питання тестування та оцінювання знань у віддаленому режимі. Це спричинило небувалий ріст відвідування навчальних онлайн-платформ а також, відповідно, складнощі, як з організацією роботи великої кількості користувачівю. Тому використання освітніх он-лайн платформ у різних напрямках буде відбуватися й надалі.

Нижче приведений графік росту зареєстрованих користувачів в однієї із систем дистанційного навчання одного із українських вузів. Як ми можемо бачити, аналізуючи графік, небувалий ріст відбувався саме в процесі пандемії і все свідчить про те, що цей процес буде тривати й надалі [6].

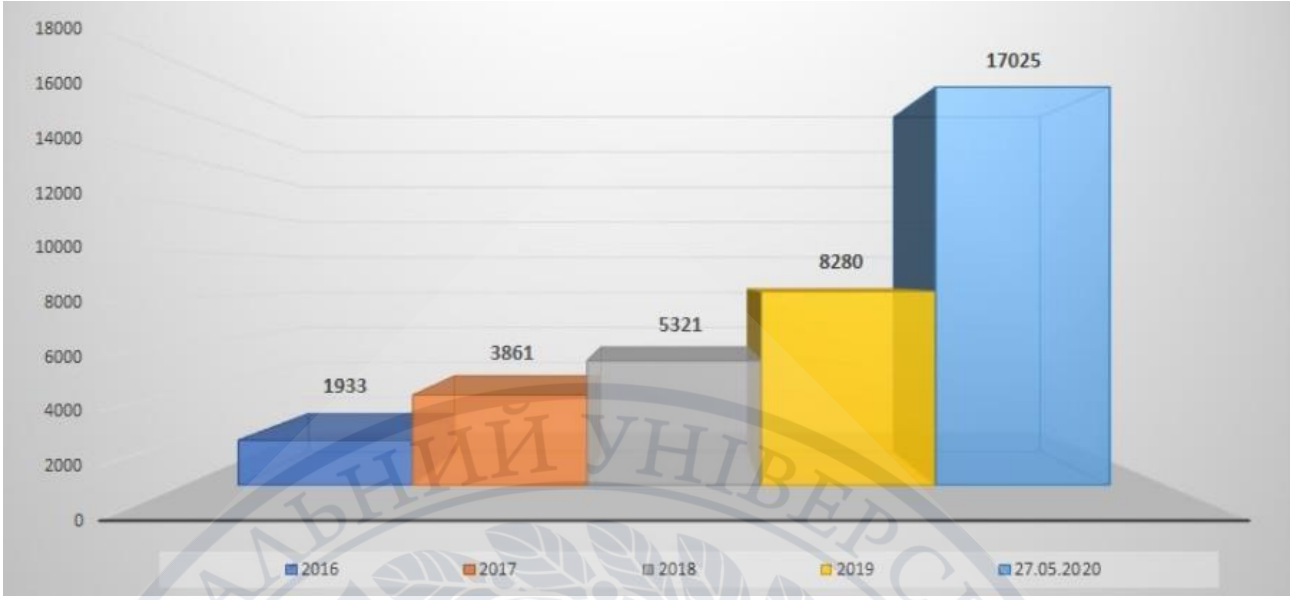

Рисунок 1.1 – Графік зростання популярності мобільної платформи [6]

1.2 Постановка задачі

Необхідно реалізувати систему тестування та оцінювання знань під управлінням системи Android [7]. Додаток повинен виконувати наступні функції:

1. Реєстрація. Відбувається за допомогою імені користувача, електронної пошти та паролю. Ім’я користувача буде використовуватись у якості імені авто конкретного тесту.
2. Авторизація. Відбувається за допомогою електронної пошти та паролю.
3. Створення тесту з наступними можливими типами питань:
    a. З однією правильною відповіддю. Кількість правильних відповідей необмежена.
    b. З декількома правильними відповідями. Кількість правильних відповідей необмежена.
    c. З відкритою відповіддю
4. Створення тесту у три етапи:
    a. Назва тесту та його опис (коротка інформація для користувача)
    b. Створення запитань та відповідей у довільній кількості
    c. Кількість правильних запитань аби вважати

5. Можливість додавати ілюстрації до кожного запитання
6. Пошук тестів за назвою або ключовим кодом. Автор тесту має можливість зробити його загальнодоступним або ж ні. У випадку закрити тесту його доступ буде проводитись лише за допомогою ключового коду.
7. Пошук тесту за допомогою QR-коду. Автор тесту має можливість поширити QR-код іншим користувачам.
8. Перегляд раніше створених тестів. Користувач, який раніше створив тест має змогу переглянути всю інформацію про проходження даного тесту іншими користувачами, а саме:
    a. Загальна статистика щодо конкретного тесту (кількість учасників, середній бал, кількість вподобань)
    b. Детальний перегляд результатів кожного з учасника, який пройшов тест (правильні та неправильні відповіді)
9. Проходження тесту. Після кожного відповіді користувач буде бачити її коректність. Після останньої відповіді з’явиться діалог із остаточним результатом склав/не склав відповідно до правил тесту.

1.3 Аналіз існуючих рішень

Системи оцінювання та тестування знань не є чимось інноваційним, тому їх наявність на ринку не є чимось дивним

Розглянемо найбільш популярні схожі за призначенням додатки під пристрої систему Android, наприклад, як [3]:

1. **Moodle [8]**

    Moodle – одна з найбільших систем навчання із можливістю тестування та оцінювання знань. Має хорошу підтримку та широкі можливості. До недоліків можна віднести не зовсім простий інтерфейс, який вимагає ознайомлення та деякі функції системи не є безкоштовними.

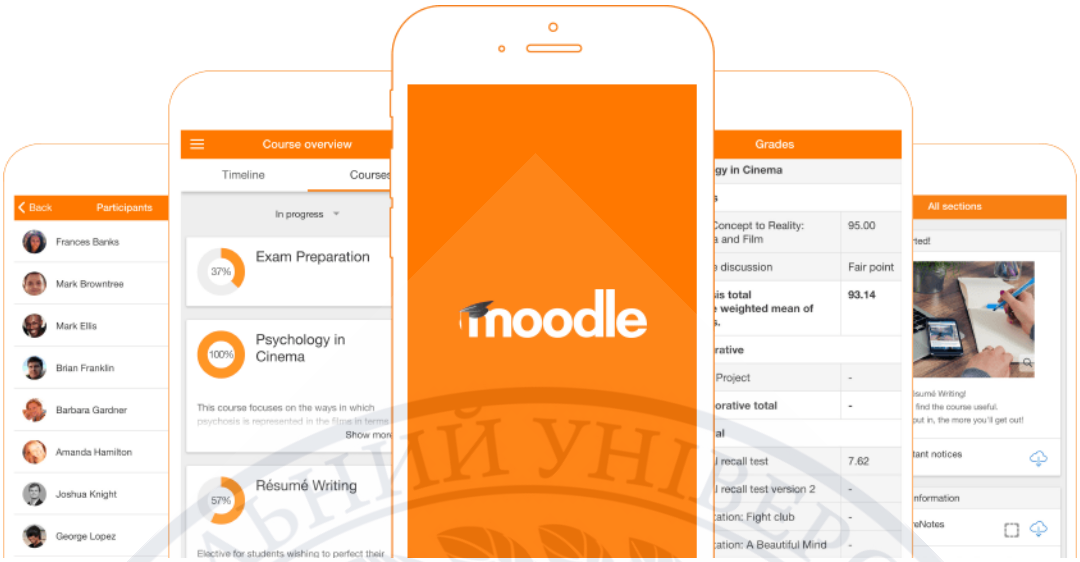

Рисунок 1.2 – Мобільний додаток Moodle

2. Kahoot [9]

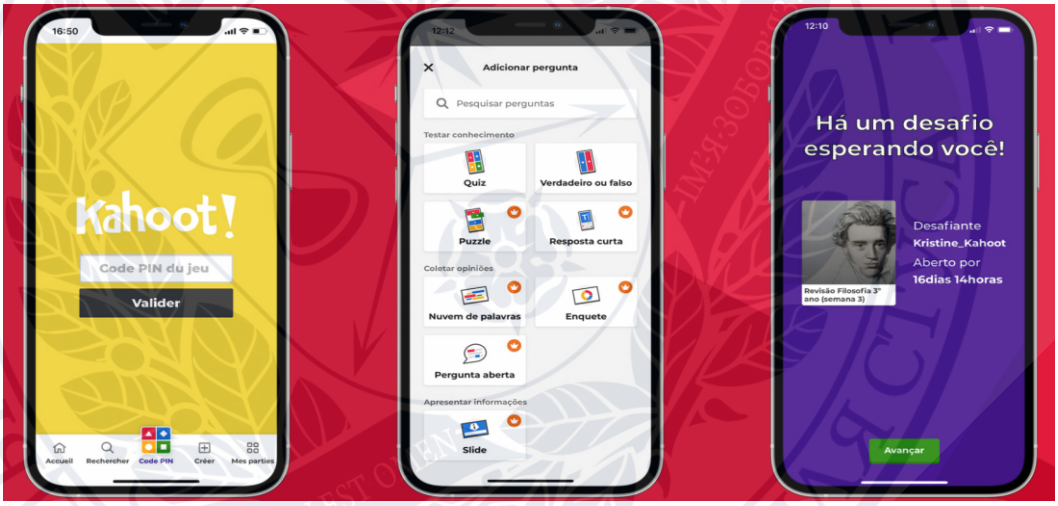

Рисунок 1.3 – Мобільний додаток Kahoot

Kahoot - це популярна навчальна платформа для проведення вікторин, створення тестів і освітніх ігор. У системи є як і веб версія так і мобільний додаток. До переваг можна віднести легкий та зрозумілий дизайн, функціональність, доступність, але відсутність можливості перегляду пройдених тестів.

3. General Knowledge Quiz [10]

General Knowledge Quiz – ще один додаток із найбільш популярних в Play Market. До переваг можна віднести простий та зрозумілий дизайн, доступність. Немає можливості створювати власні тести.

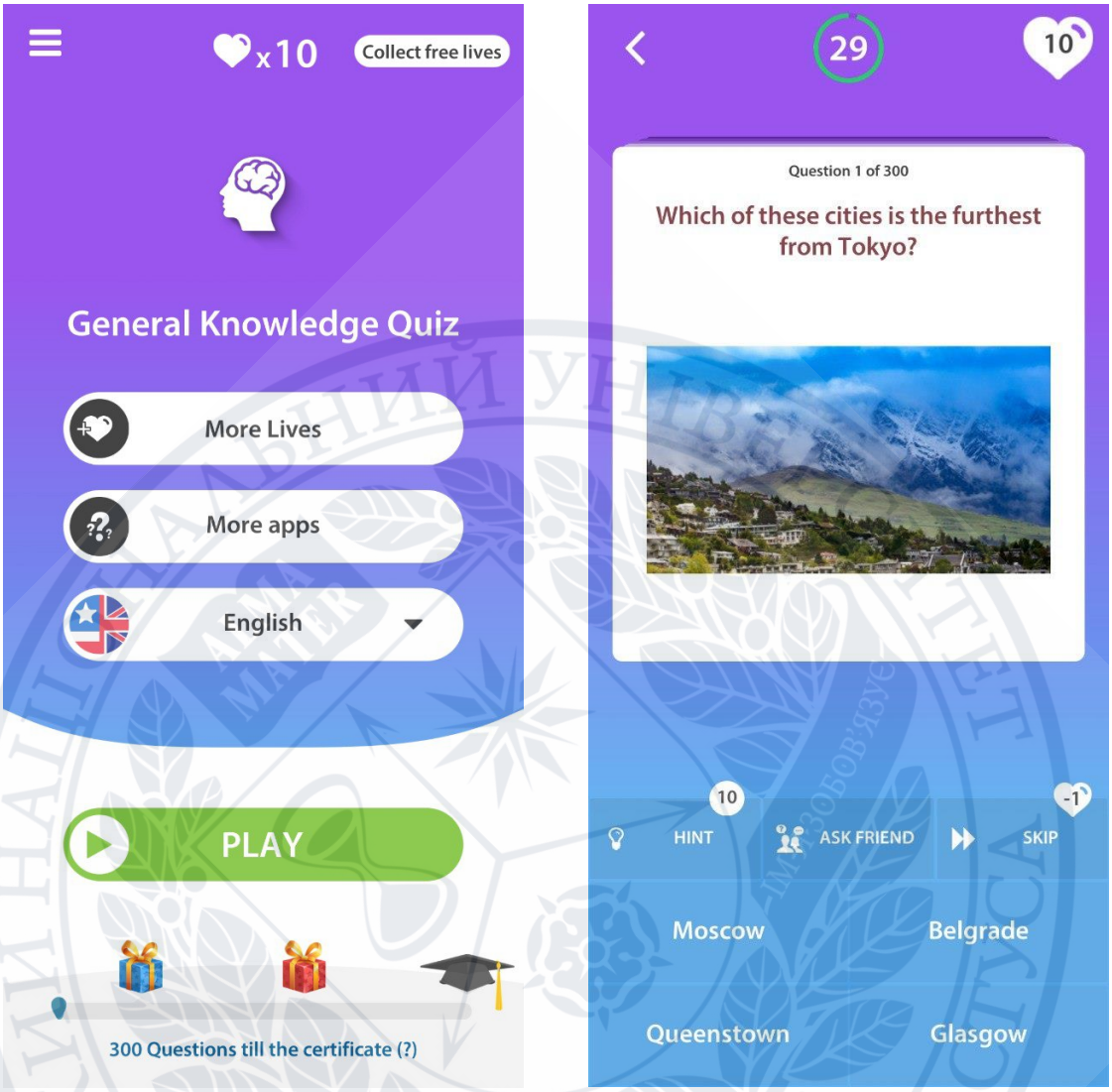

Рисунок 1.4 – Мобільний додаток General Knowledge Quiz

4. OpenTest

OpenTest – відкрита система тестування, де кожен бажаючий може розмістити свій тест, додати до нього питання, зареєструватись користувачів, переглянути результати тестування. Найбільшим недоліком є відсутність мобільного клієнта.

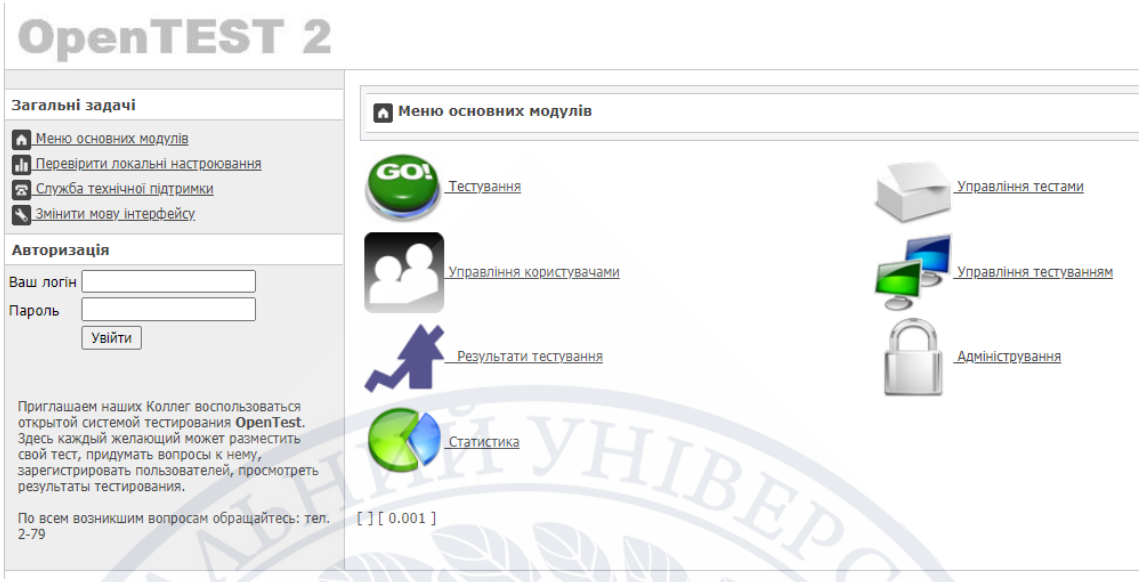

Рисунок 1.5 – Система тестування OpenTest

5. A.S.T.S [12]

Antonov Students Test System - інтелектуальна система тестування, яка на основі даних відповідей буде аналізувати їх та пропонувати додаткові, аби дати точніший результат. Основними недоліками є застарілий дизайн та відсутність мобільного клієнту.

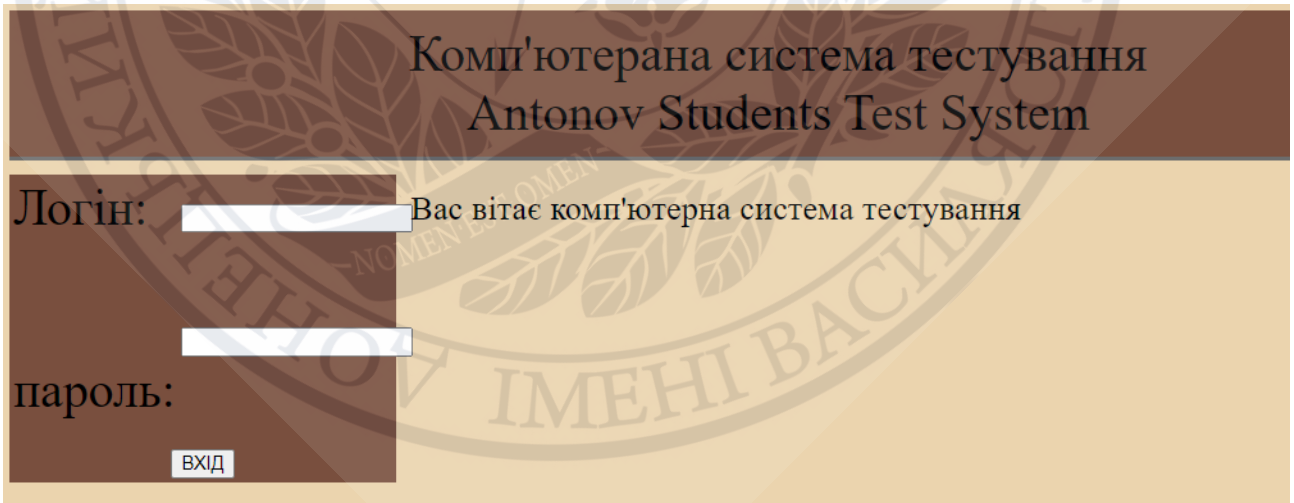

Рисунок 1.6 – Система тестування Antonov Students Test System

Розглянувши альтернативи, ми можемо скласти таблицю для порівняння та встановлення переваг та недоліків даних програм:

Таблиця 1.1 – Порівняння існуючих аналогів

	Moodle [8]	Kahoot [9]	General Quiz Knowledge [10]	OpenTest [11]	A.S.T.S. [13]
Відкритість	Так	Так	Ні	Так	Ні
Наявність мобільного додатку	Так	Так	Так	Ні	Ні
Безкоштовність	Ні	Так	Так	Так	Ні
Сучасний зрозумілий дизайн	Так	Так	Так	Так	Ні
Відкриті питання	Так	Ні	Ні	Так	Так
Закриті питання	Так	Так	Так	Так	Так
Обрання правильної послідовності	Так	Ні	Ні	Так	Так
Обрання правильної відповідності	Так	Ні	Ні	Так	Так
Можливість створювати тести довільному користувачу	Так	Так	Ні	Так	Так
Можливість перегляду пройдених тестів	Так	Ні	Ні	Так	Так
Розширені можливості аналізу відповідей	Ні	Ні	Ні	Ні	Так
Наявність експертної системи	Ні	Ні	Ні	Ні	Так

Із таблиці (1.1) ми можемо бачити, що деякі із додатків не є безкоштовними, не є відкритими та не мають можливості створення власних тестів та перегляду результатів пройдених. Деякі з додатків існують лише у веб версії.

1.3 Інструменти для написання серверної частини

Android пропонує розробникам масу можливостей: це універсальна, відкрита платформа, яка використовується мільйонами користувачів по всьому світу. На щастя, існує безліч інструментів для Android-розробників, які допоможуть швидко почати роботу. Але ще краще те, що з кожним роком інструментів стає все більше, а їх ефективність постійно підвищується.

Android Studio [14]

Android Studio – це програмний продукт призначений для розробки додатків під платформу Android. Android Studio має багато влаштованих інструментів для полегшення та пришвидшення розробки. Це середовище розробки безкоштовним та є крос-платформенним, що дає змогу користуватись ним на більшості сучасних ОС.

Android Studio побудоване на основі Intellije Idea, призначений як і для програмування самостійно так і для великих команд, завдяки таким вбудованим інструментам як Git, що спрощує взаємодію між розробниками.

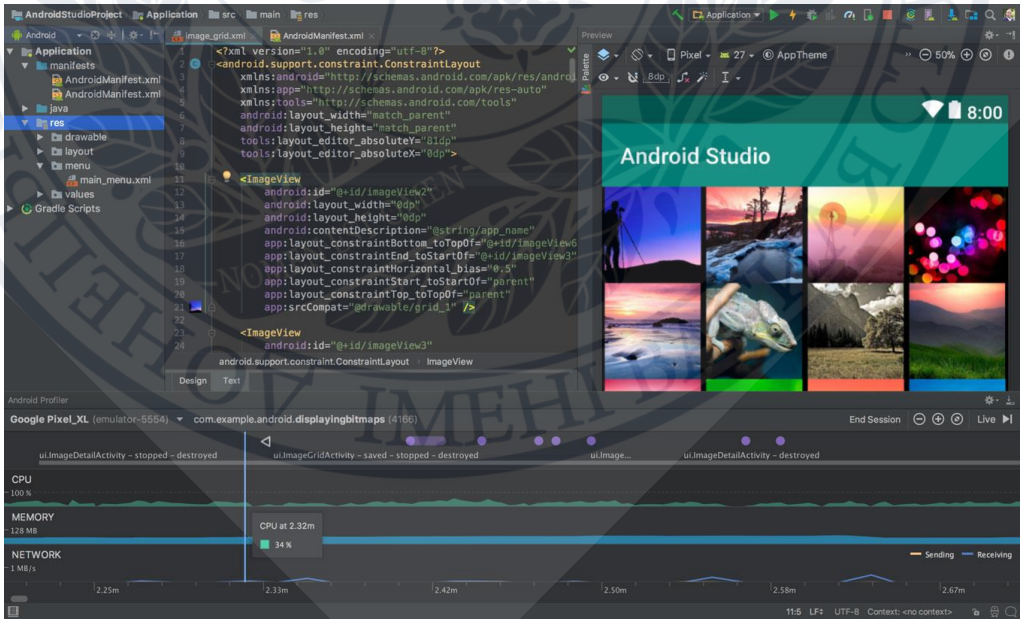

Рисунок 1.7 – графічний редактор Android Studio IDE

AVD Manager

AVD Manager – інструмент, який уже влаштований в Android Studio. Даний засіб допомагає розробникам запускати їх додатки навіть без наявності фізичних пристроїв. Це є дуже корисним, у випадку, коли є потреба перевірити додаток в різних розширенням, з різною кількістю оперативної пам’яті та іншими технічними характеристиками.

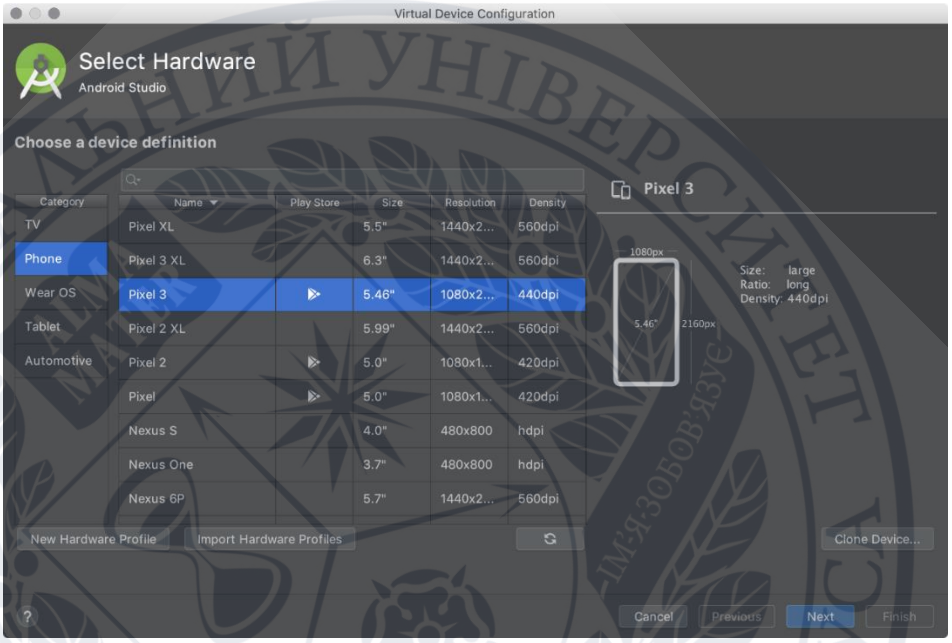

Рисунок 1.8 – конфігурація віртуального девайса в Android Studio

Android Device Monitor

Внутрішній інструмент Android Studio, який дає змогу відслідковувати всі процеси віртуального та фізичного пристрою навіть в момент його праці. Які можливості має Android Device Monitor:

- Відслідковування мережевих запитів, параметрів, результатів і т.д.
- Проводити моніторинг стану пам’яті пристрою
- Визначати втрати пам’яті в конкретний момент
- Визначати завантаженість на пристрій в залежності від встановленого додатку

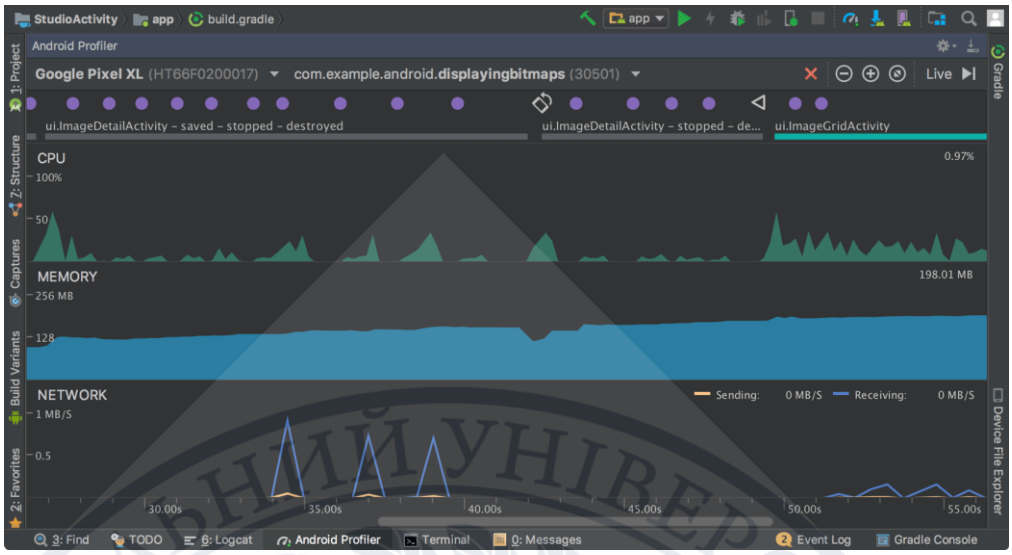

Рисунок 1.9 – моніторинг ресурсів запущеного додатку

Android Debug Bridge

ADB - консольний додаток для ПК, за допомогою якого проводиться налагодження Android пристроїв, в тому числі і емуляторів. Працює за принципом клієнт-сервер. При першому запуску ADB з будь-якою командою створюється сервер у вигляді системної служби (демона), яка буде прослуховувати всі команди, що посилаються на порт 5037. Він йде в комплекті з Android Studio і, здебільшого, вам не доведеться його встановлювати самостійно.

Переваги Android Studio [15]

- офіційна IDE для розробки під Android;
- підтримка декількох мов програмування без необхідності використання декількох IDE
- підтримка різних пристроїв окрім смартфонів (планшети, Android TV, Android Wear)
- зручна робота із системами версії контролю;
- зручний та функціональний редактор коду;
- підказки до коду та автозавершення;
- багатофункціональний графічний редактор, який дає можливість створювати сторінки для девайсів із різними характеристиками;

- велика кількість додаткових модулів, які роблять розробку швидшою;
- підтримка усіх нововведень;
- зручний інтерфейс та робота із структурою проекту.
- безліч «гарячих клавіш» для швидкості роботи;

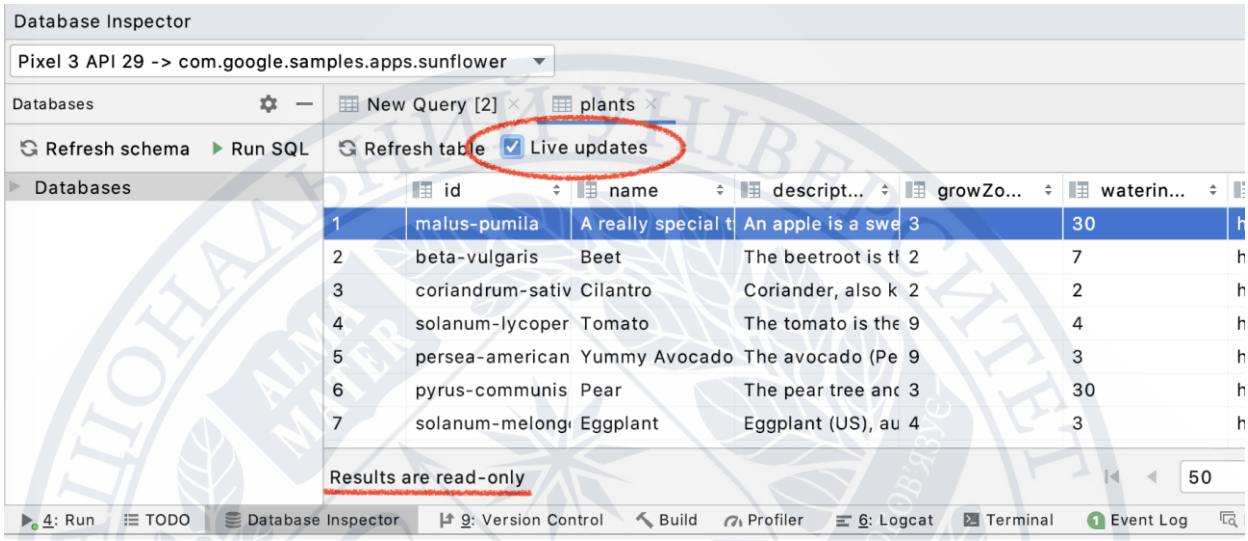

Рисунок 1.10 – Database Inspector. Дає можливість переглядати локальну базу даних в реальному часі

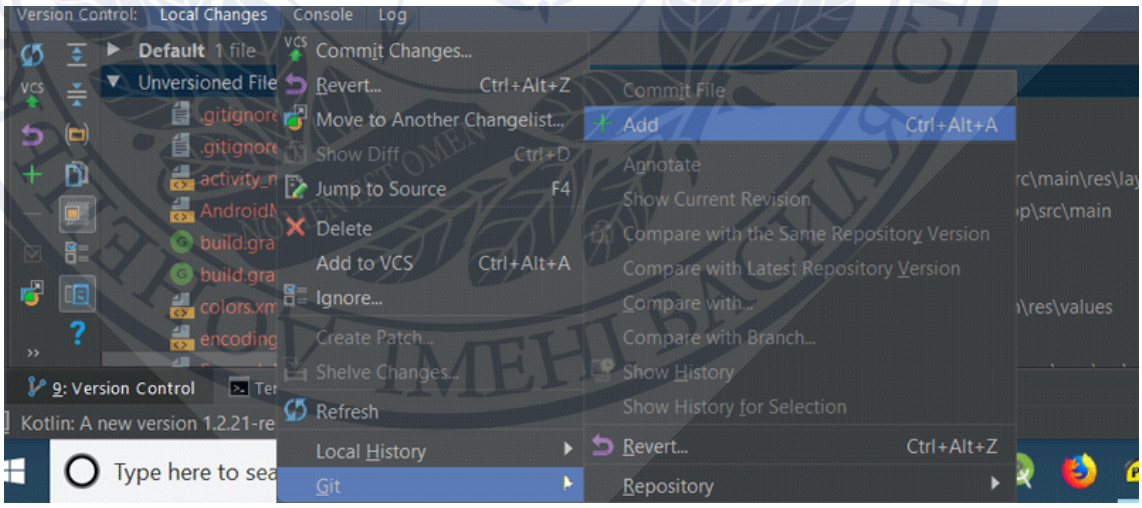

Рисунок 1.11 – робота із Git в Android Studio

1.4 Інструменти для написання клієнстької частини

Серверна частина також реалізована за допомогою мови програмування Kotlin, аби не працювати з двома цілковито різними мовами програмування.

Якщо говорити про Kotlin Server Side, то однозначним вибором IDE є IntelliJ Idea.

IntelliJ IDEA – програмний продукт, який є одним із найбільш популярних у виборі середовища розробки для Java, Kotlin та інших JVM мов програмування. Основними перевагами даної IDE є:

1. Зручний та лаконічний інтерфейс
2. Можливість керувати версіє контролю прямо з IDE
3. Розумне автодоповнення коду
4. Можливість створювати власні гарячі клавіші
5. Безліч плагінів

Висновок до розділу 1

У цьому розділі було розглянуто актуальність мобільних додатків, потребу систем оцінювання та тестування знань у теперішній ситуації в сфері навчання. Також, були описані необхідні функції програми, її альтернативи на даний момент та їх переваги та недоліки.

РОЗДІЛ 2

ОПИС ТЕХНОЛОГІЙ ВИКОРИСТАНИХ У РОЗРОБЦІ

2.1 Технології використані у розробці серверної частини

Обравши Kotlin [17] для розробки на Android [18] , була можливість використання деякого коду і на серверній частині, тому було обрано фреймворк Ktor.

Ktor [16]

Ktor - це платформа з відкритим вихідним кодом для створення асинхронних серверів і клієнтів в підключених системах з використанням потужного мови програмування Kotlin [19]. Даний фреймворк допомагає розробникам із Android швидко створювати REST [20] API із використанням уже відомих технологій. Незважаючи на переносних залежні клієнти, якщо ви спробуєте навчитися використовувати клієнт Ktor на своїй рідній платформі, ви легко зможете працювати з Ktor на інших платформах. У своїй реалізації Ktor використовує співпрограми для асинхронного програмування, щоб код залишався читаним і чистим. На додаток до цього, у Ktor є безліч бібліотек Kotlinx, які допомагають розробникам з такими завданнями, як синтаксичний аналіз відповіді Використання мережевої бібліотеки, яку можна інтегрувати в кілька платформ, є кращим рішенням, ніж вибір бібліотек для конкретної платформи. У міру його розвитку JetBrains матиме більш просунуті функції і довгострокову підтримку. Додаток Ktor [21] - це програма, яка прослуховує один або кілька портів і містить модулі з встановленими функціями; в даному випадку це Netty [22].

PostgreSql [23, 24]

Для зберігання даних використовується PostgreSQL. Це - об'єктно-реляційна система керування базами даних (СУБД) [24]. Наведена СКБД виступає альтернатива комерційним СКБД (Oracle Database, Microsoft SQL Server), так и СУБД з відкрітім кодом (MySQL, Firebird, SQLite).
Порівняно з іншімі проектами з відкритим кодом,

PostgreSQL базується на мові SQL і підтримує численні можливості. Основними перевагами даної СУБД є [25]:

1. підтримка БД необмеженого розміру;
2. потужні і надійні механізми транзакцій і реплікації;
3. спадкування;
4. легка розширюваність.

Netty [26]

Це framework для створення програмного забезпечення, що взаємодіють по мережі. Це може бути і балансувальник навантаження, і мережевий павук, і практично будь-яка інша архітектура, заснована на понятті client / server. В основі лежить java.io і java.nio. Вибір базового механізму здійснюється шляхом використання конкретних реалізацій ключових компонентів. Підхід до їх використання прихований під абстракцією і інтерфейсами, так що перейти від одного до іншого не складе труднощів на відміну від того, як би це було реалізовано в Java. Всі мережеві операції асинхронні. Управління асинхронним взаємодією здійснюється на основі подієвої моделі або з використанням Future об'єктів подібних Future об'єктів в Java. [27]

Ktor Locations

Функція Ktor, яка дозволяє обробляти та будувати маршрути набранимспособом. Даний код буде співпадати із **/list/movies/page/10.**

```
1 @Location("/list/{name}/page/{page}")
2 data class Listing(val name: String, val page: Int
```

Ktor JWT [28]

Ktor підтримує JWT (веб-токени JSON), який є механізмом для автентифікації корисних навантажень, кодованих JSON [29]. Корисно створювати автентифіковані API без станів стандартним способом, оскільки для цього існує клієнтська бібліотека на безлічі мов.

Exposed

Для керування PostgresSQL вибір впав на Exposed. Це бібліотека з відкритим кодом (ліцензія Apache), розроблена JetBrains, яка забезпечує ідіоматичний API Kotlin для деяких реалізацій реляційних баз даних, одночасно згладжуючи відмінності між постачальниками баз даних. Exposed може використовуватися як DSL високого рівня над SQL, так і як полегшений ORM (об'єктно-реляційне відображення).

Основними перевагами бібліотеки є:

1. Багато влаштованих функцій для зручної роботи
2. Хороша документація
3. Підтримка Kotlin DSL [30]
4. Є легковісною
5. Можливість працювати з різними типами баз даних

```
object Cities : Table() {
   val id = integer("id").autoIncrement() // Column<Int>
   val name = varchar("name", 50) // Column<String>
   override val primaryKey = PrimaryKey(id, name = "PK_Cities_ID")
}
val saintPetersburgId = Cities.insert {
   it[name] = "St. Petersburg"
} get Cities.id
```

Приклад створення таблиці та запису

2.2 Технології використані у розробці клієнтської частини

Ще не так давно Android розробка асоціювалась із мовою програмування Java, але прогрес не стоїть на місці та з’являються більш зручні, функціональні мови програмування. Одною із таких стала програмування мова Kotlin [31]. У 2017 році **Kotlin**, розроблений петербурзькою компанією **JetBrains**, став

офіційною мовою розробки під Android. Про це офіційно оголосили на конференції **Google I/O.**

Розглянемо основні переваги Kotlin над Java:

1. Kotlin Corounites [32] – робота із асинхроними задачами [33];
2. Data classes – спрощена робота із моделями даних;
3. Extensions – змога розширювати закриті частини програми;
4. Null Safety – обробка можливих помилок, пов’язаних із всім відомою NPE у Java [34];
5. Короткий синтаксис – у Kotlin є широкий влаштовані можливості, щоб зробити ваш код коротшим та зрозумілішим [35].

Kotlin Coroutines [36]

Деякі API ініціюють довго протікають операції (такі як мережевий введення-виведення, файловий ввід-висновок, інтенсивна обробка на CPU або GPU і ін.), Які вимагають блокування викликає коду в очікуванні завершення операцій. Співпрограми забезпечують можливість уникнути блокування що виконується потоку шляхом використання більш дешевої і керованої операції: припинення (suspend) співпрограми. Співпрограми допомагають розробникам писати асинхронний код у вигляді синхронного, використовуючи кінцевий автомат у своїй реалізації.

Navigation Components [37]

Навігація в мобільних додатках безсумнівно є однією із найбільш важливих речей на яку в першу чергу звертає увагу користувач. Jetpack Navigation Components це набір бібліотек та інструментів, які допомагають зробити навігацію простою та дають можливість масштабування.

Навігація управляє переміщенням між destinations. Destinations зазвичай представлені фрагментами, проте підтримують activities і інші призначені для користувача destinations. За допомогою Navigation Architecture Component можна з легкістю реалізувати складну навігацію в android. Він надає набір компонентів навігації, таких як Fragment transactions, Up and Back navigation, які

обробляють велику частину інформації. Крім цього, надаються шаблони Navigation UI, такі як navigation drawers, bottom navigation і багато інших, що містять мінімальну кількість коду.

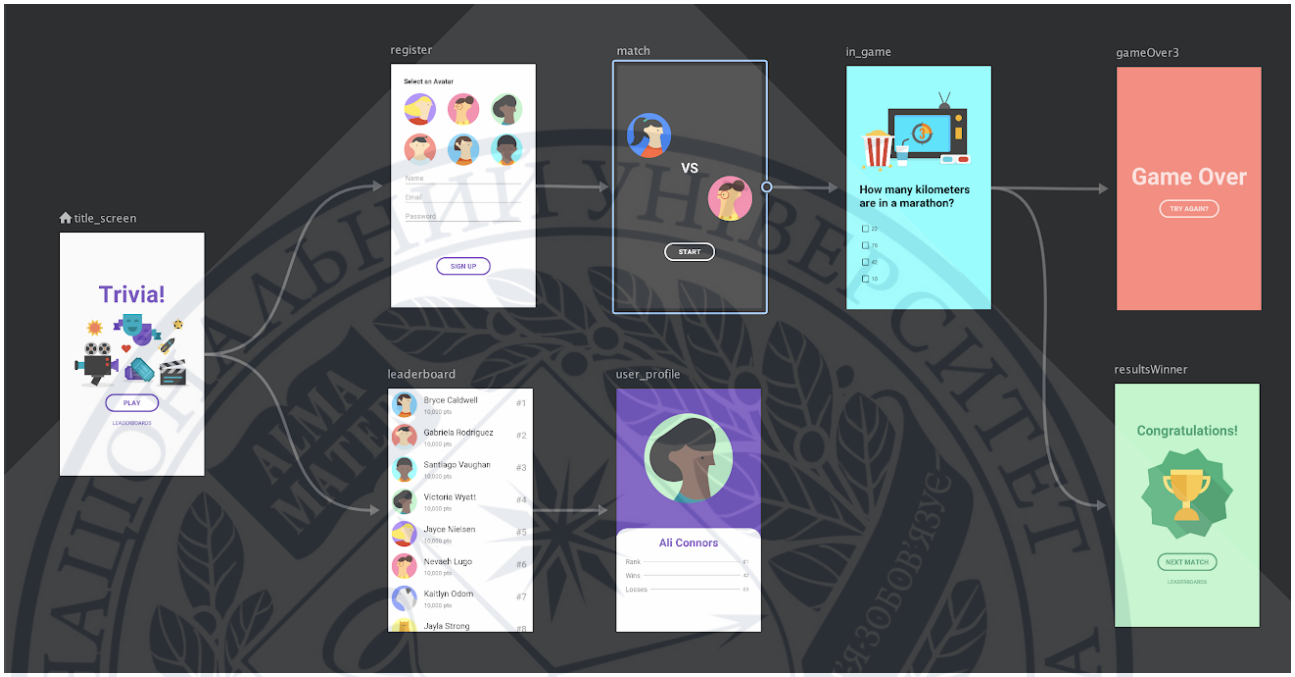

Рисунок 2.1 - Основною перевагою є візуалізація списка екранів та переходів між ними

Hilt [38, 39]

Впровадження залежностей - це архітектурний підхід, при якому один об'єкт надає залежність іншого об'єкта. Залежністю вважають об’єкт, який необхідний клієнту для коректної роботи. Передайте залежність клієнтам, замість того, щоб дозволити їм створити його власноруч - базова потреба шаблону проектування «Впровадження залежностей» (DI) [15].

Іншими словами, впровадження залежностей грунтується на концепції інверсії контролю, яка говорить про те, що клас повинен отримувати свої залежності ззовні. Говорячи просто, жоден клас не повинен створювати екземпляр іншого класу, а повинен отримувати всі екземпляри з класу конфігурації.

Переважну частину часу в Android розробці для впровадження залежностей використовували Dagger. Але нещодавно Google представила

новий набір бібліотек та інструменів під назвою Hilt, який використовує Dagger всередині та спрощує взаємодію з ним.

Основними перевагами Hilt є:

1. Зменшення boilerplate коду
2. Спрощена конфігурація
3. Покращене тестування
4. Стандартизація компонентів

Рисунок 2.2 - Основні компоненти Dependency Injection в Dagger

Retrofit [40]

Так як наший додаток побудований з міркувань клієнт-серверної архітектури, то цілком логічно, що нам буде потрібно реалізовувати http запити. Для цього в більшості Android додатків використовують Retrofit.

Retrofit – асинхронна бібліотека для мережевих запитів. Він дає змогу за допомогою опису інтерфейсу завантажувати дані із віддалених сервісів. У своїй реалізації потребує OkHttp клієнт та конвертер, який буде використовуватись для перетворення отриманої відповіді у об’єкти необхідного типу.

Основними перевагами бібліотеки є:

1. Внутрішня підтримка OkHttp за замовченням
2. Лаконічне і зрозуміле API
3. Інтеграція з іншими бібліотеками багатопоточності
4. Можливість розширення різними мережевими перехоплювачами
5. Можливість додати будь-яку бібліотеку серіалізації

```
1 interface AuthService {
2    @POST("$PUB/auth/login")
3    suspend fun login(@Body login: LoginRequestBody):
  AuthResponse
4    @POST("$PUB/auth/register")
5    suspend fun register(@Body register:
  RegisterRequestBody): AuthResponse
6 }
```

Приклад використання API Retrofit

Android KTX [41]

Android KTX - це набір Android-бібліотек з розширеннями Kotlin для спрощення кодування при роботі з Jetpack і іншими бібліотеками Android. Розширення KTX використовують можливості Kotlin, щоб надати розробникам короткий підхід при роботі з Jetpack і іншими API.

Кожна бібліотека або API має окрему бібліотеку Kotlin KTX з розширеннями Kotlin, що відносяться до цього сегменту. У цій статті ми зосередимося на п'яти розширеннях, які спростять розробку Android при роботі з MVVM, LiveData, Firebase і базою даних Room.

Нижче наведено декілька прикладів використання бібліотеки:

Перетворити рядок в URI

Зазвичай parse(uriString), Але Android KTX додасть до рядка функцію розширення, щоб рядок природніше перетворювалася в URI.

```
1 // Kotlin
2 val uri = Uri.parse(uriString)
3 // android KTX
4 val uri = uriString.toUri()
```

SharedPreferences

SharedPreferences також часто іспользуется.После використання Android KTX з'являється багато коротких кодів.

```
1 // kotlin
2 sharedPreferences.edit()
3    .putBoolean("key", value)
4    .apply()
5 // KTX
6 sharedPreferences.edit {
7    putBoolean("key", value)
8 }
```

Path

Відстань між двома шляхами змінилося на 100 пікселів.

Recycler View [42]

```
1.// kotlin
2.val pathDifference = Path(myPath1).apply {
3. op(myPath2, Path.Op.DIFFERENCE)
4.}
5.canvas.apply {
6. val checkpoint = save()
7. translate(0F, 100F)
8. drawPath(pathDifference, myPaint)
9. restoreToCount(checkpoint)
10.   }
```

```
1.// KTX
2.val pathDifference = myPath1 - myPath2
3.canvas.withTranslation(y = 100F) {
4.drawPath(pathDifference, myPaint)
5.}
```

Recycler View – графічний компонент Android, який дає змогу розробникам створювати списки даних різної складності. Він прийшов на заміну ListView і

має досить багато переваг для того, щоб розробники назавжди відмовились від ListView. Recycler View дозволяє розділяти відповідальність між встановленням даних та їх інтерфейсом. Для реалізації списку за допомогю RecyclerView, нам потрібно задіяти такі компоненти:

1. RecyclerView, який ми повинні додати в layout нашого екрану;
2. Layout для кожного рядка списку;
3. Адаптер, який містить дані і пов'язує їх зі списком.

Однією із основних переваг RecyclerView є LayoutManager. LayoutManager це удосконалення, що з'явилося в RecyclerView. У ListView єдиним доступним видом є вертикальний ListView. Використовуючи RecyclerView ми маємо наступні допоміжні класи для розстановки елементів:

1. LinearLayoutManager [43] – горизонтальні та вертикальні списки за допомогою лише одного параметру (LinearLayoutManager.VERTICAL та LinearLayoutManager.HORIZONTAL)
2. StaggeredLayoutManager - який підтримує Pinterest як шахові списки,
3. GridLayoutManager [44] - який підтримує відображення грід в додатках Gallery.

І найкраще, що ми можемо робити все це динамічно, як ми хочемо.

До інших переваг ми можемо віднести також:

1. Можливість анімації елементів списка
2. Можливість додавати прослуховувачів на дотик кожного елемента
3. Покращення в роботі, перевикористання елементів
4. Шаблон ViewHolder [45]

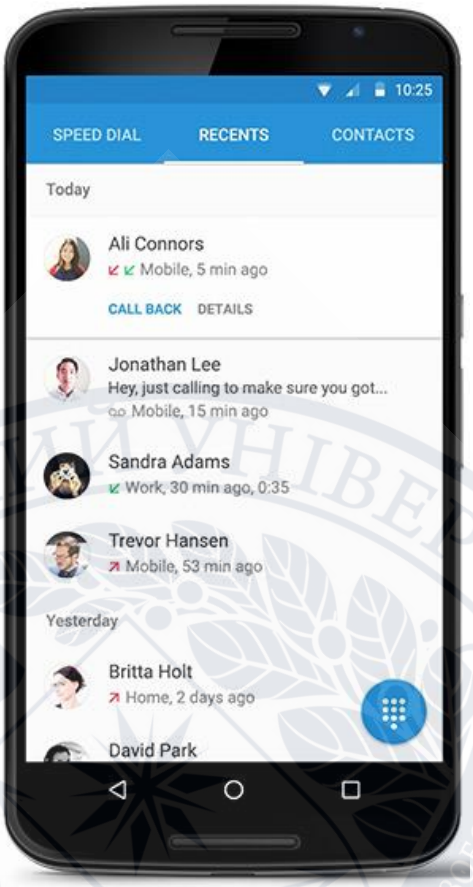

Рисунок 2.3 – Приклад створення досить складного списку за допомогою RecyclerView [46]

Висновок до розділу 2

У даному розділі розглянуті основні інструменти та технології для швидкої та комфортною розробки мобільних додатків. Розглянуті основні бібліотеки для побудови багатофункціонального додатку.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКА

3.1 Опис призначеного для користувача інтерфейсу

Всі дизайн макети виконані у додатку Figma. Це орієнтований на браузер UI/UX дизайнерський програмний продукт із лаконічним інтерфейсом та інструментарієм, що дає змогу швидко розробляти нові прототипи та вносити зміни уже в існуючі.

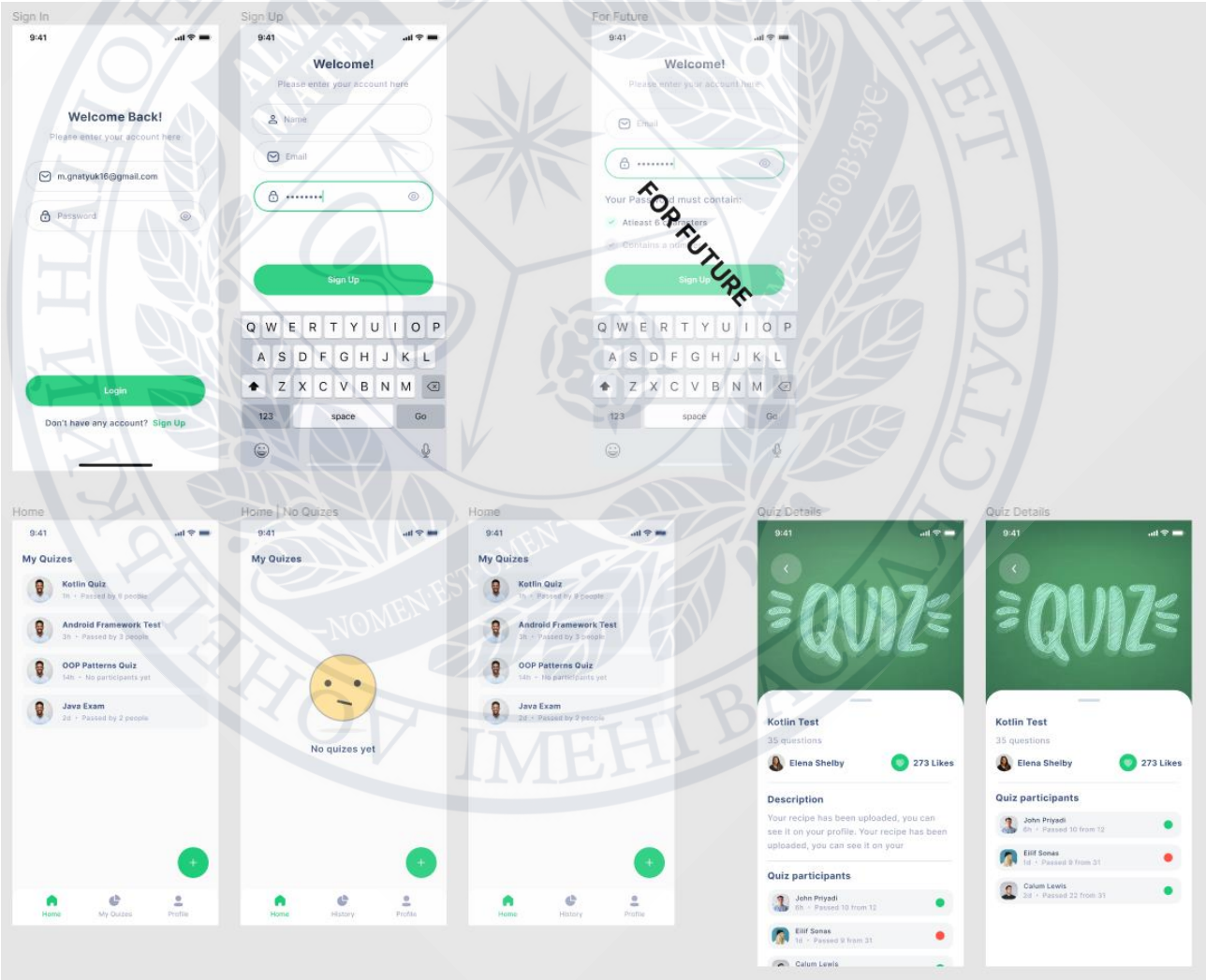

Рисунок 3.1 – Макет дизайну

Екран авторизації

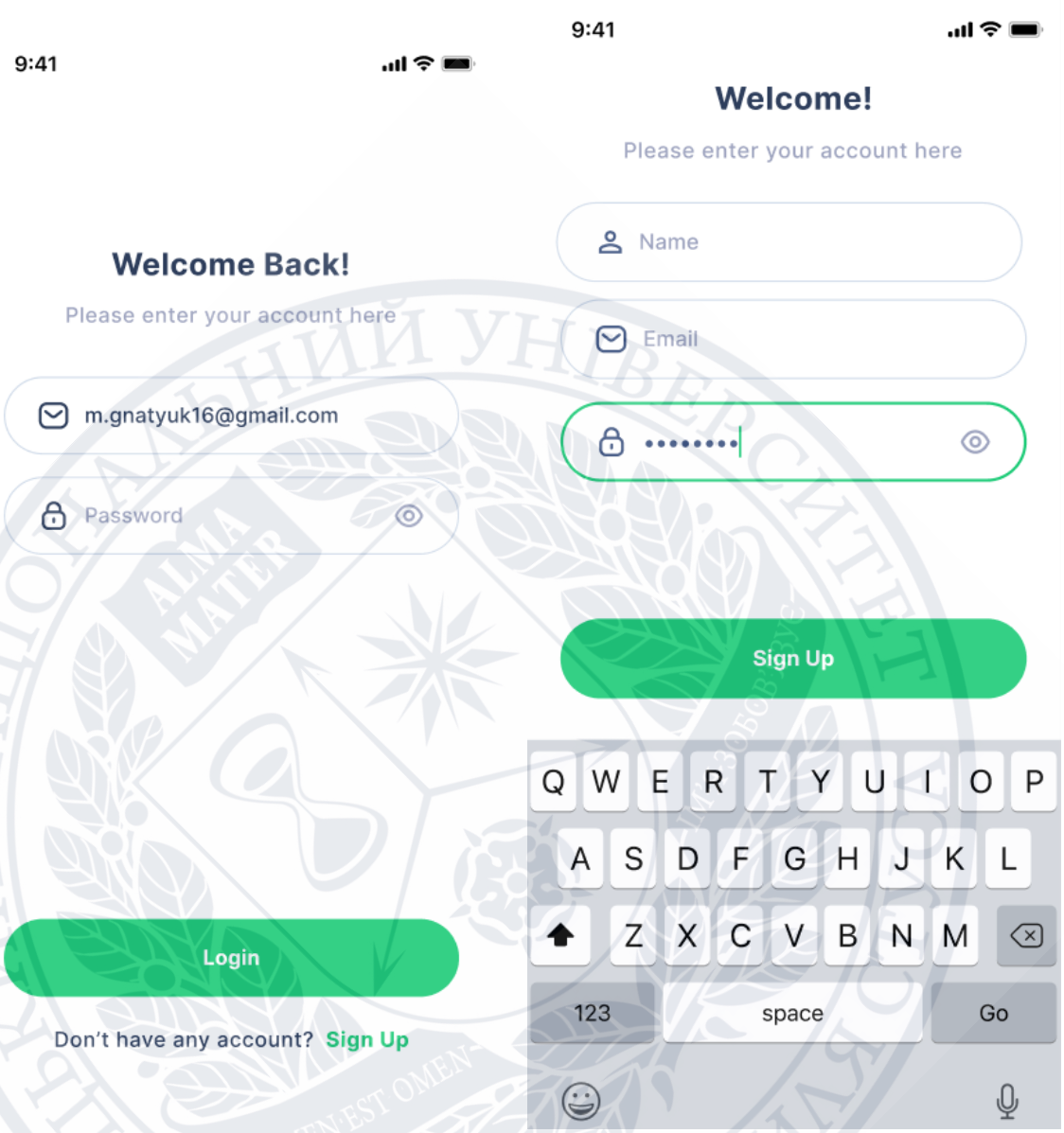

Рисунок. 3.2 – Екран авторизації

Екран входу є початковим екраном у випадку, коли користувач не виконував вхід в додаток раніше. Вхід відбувається за допомогою електронної пошти та паролю. Після успішного входу на наступний запуск додатку користувач одразу потрапить на головну сторінку, так як данні про користувача зберігаються.

На екран реєстрації ми можемо потрапити з екрану входу у випадку відсутності аккаунта. Ми можемо зареєструватись заповнивши три поля:

- Ім’я (це ім’я буде використуватись в якості автора тесту, які будуть шукати тести створені користувачем)
- Електронна пошта
- Пароль (має містити цифри та заголовні букви)

Після успішної реєстрації дані користувача будуть збережені та після наступного запуску додатку буде відкрита одразу головна сторінка.

Головна сторінка

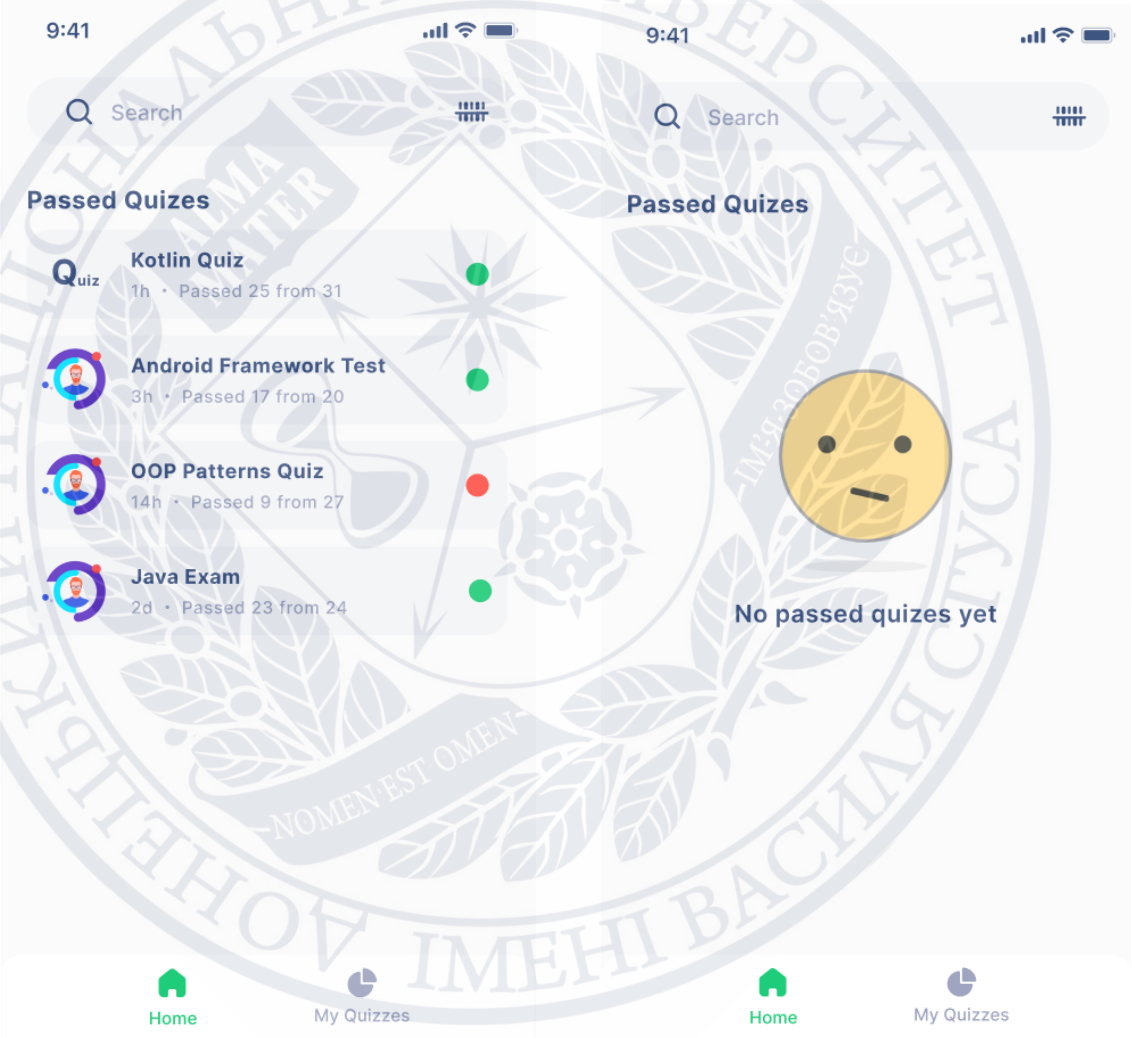

Рисунок 3.3 – Екран головної сторінки

Головна сторінка є початковою у випадку авторизованого попередньо користувача.

Даний екран дає змогу за допомогою назви або коду знайти потрібний тест. Містить інформацію про пройдені тести. Елемент списку пройдених тестів має наступну інформацію:

- Логотип
- Назва
- Дата проходження
- Статус (склав/не склав)
- Кількість правильних відповідей

У випадку відсутності пройдених тестів буде стан екрану на рис. (Головна сторінка, порожній список).

Проходження тесту

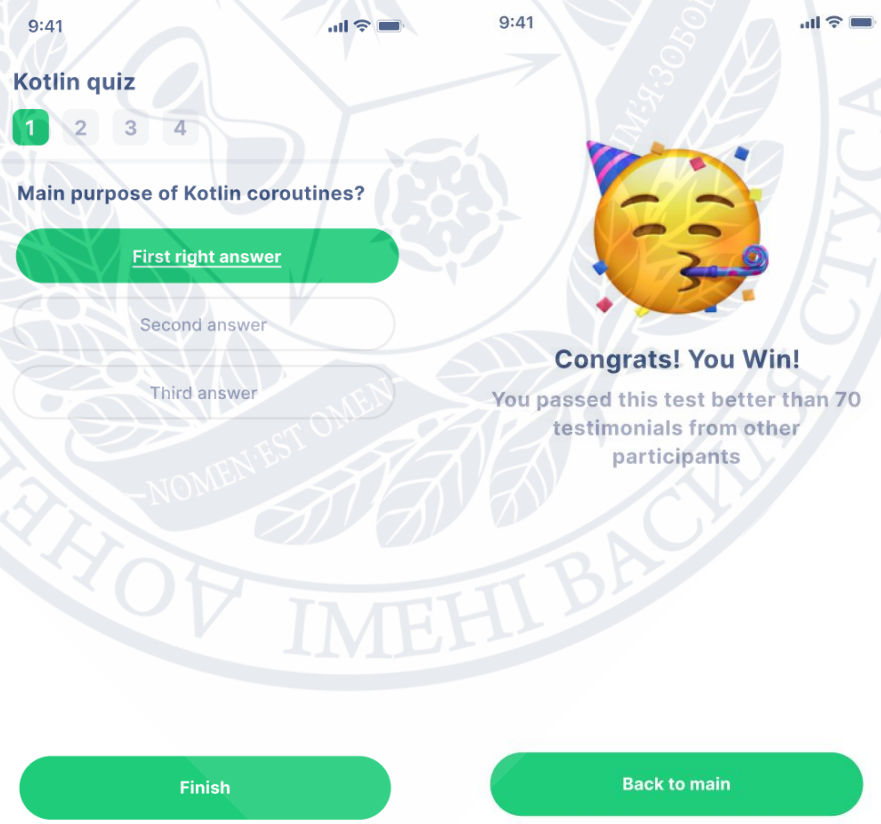

Рисунок 3.4 – Екран проходження тесту

Екран проходження тесту дає нам можливість перевірити наші знання у вибраному тесті. Одразу після вибору відповіді ми отримаємо результа –

вірно/невірно. У разі невірної відповіді ми отримаємо інформацію про відповідь, яка була вірною. Після проходження останнього питання ми отримаємо результат проходження – склав/не склав.

Мої тести

Рисунок 3.5 – Екран моїх тестів

Екран моїх тестів містить інформацію про раніше створені нами тести. Кожен елемент списку має інформацію:

- Логотип
- Назву тести

- дату створення
- кількість людей, які намагались пройти тест.

Рис. (пустий список) містить логотип та надпис, що повідомляють нам про відстуність тестів, створених нами.

Створення тесту

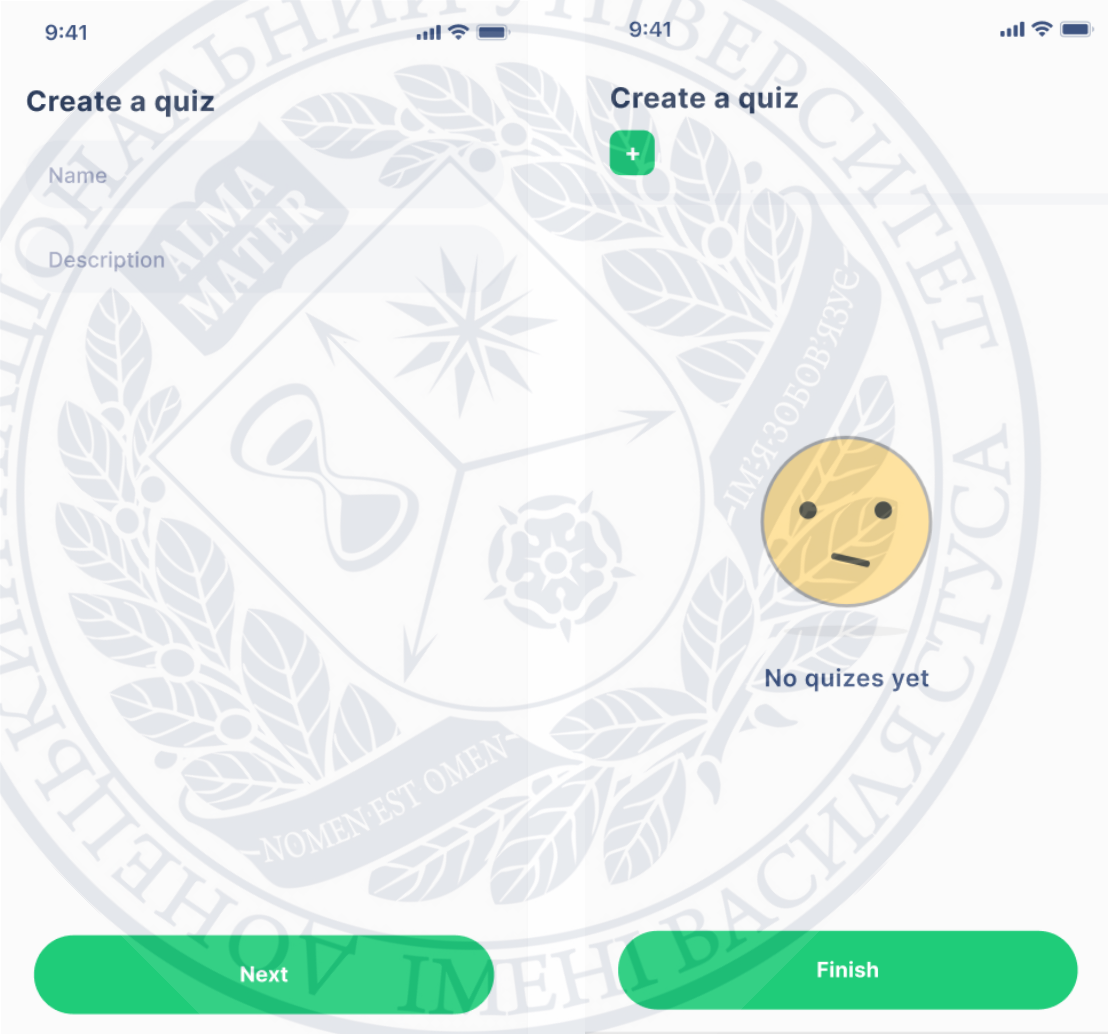

Рисунок 3.6 – Екран створення тесту, крок 1-2

Екран створення тесту на кроці 1 дає нам змогу ввести початкові дані про тест, такі як назву та опис. Після заповнення необхідених полів можемо перейти до кроку 2.

Створення тесту на кроці 2 дає на змогу створювати запитання із відповіддями. На рис. (Створення тесту. Крок 2) зображений стан порожнього списку без запитань.

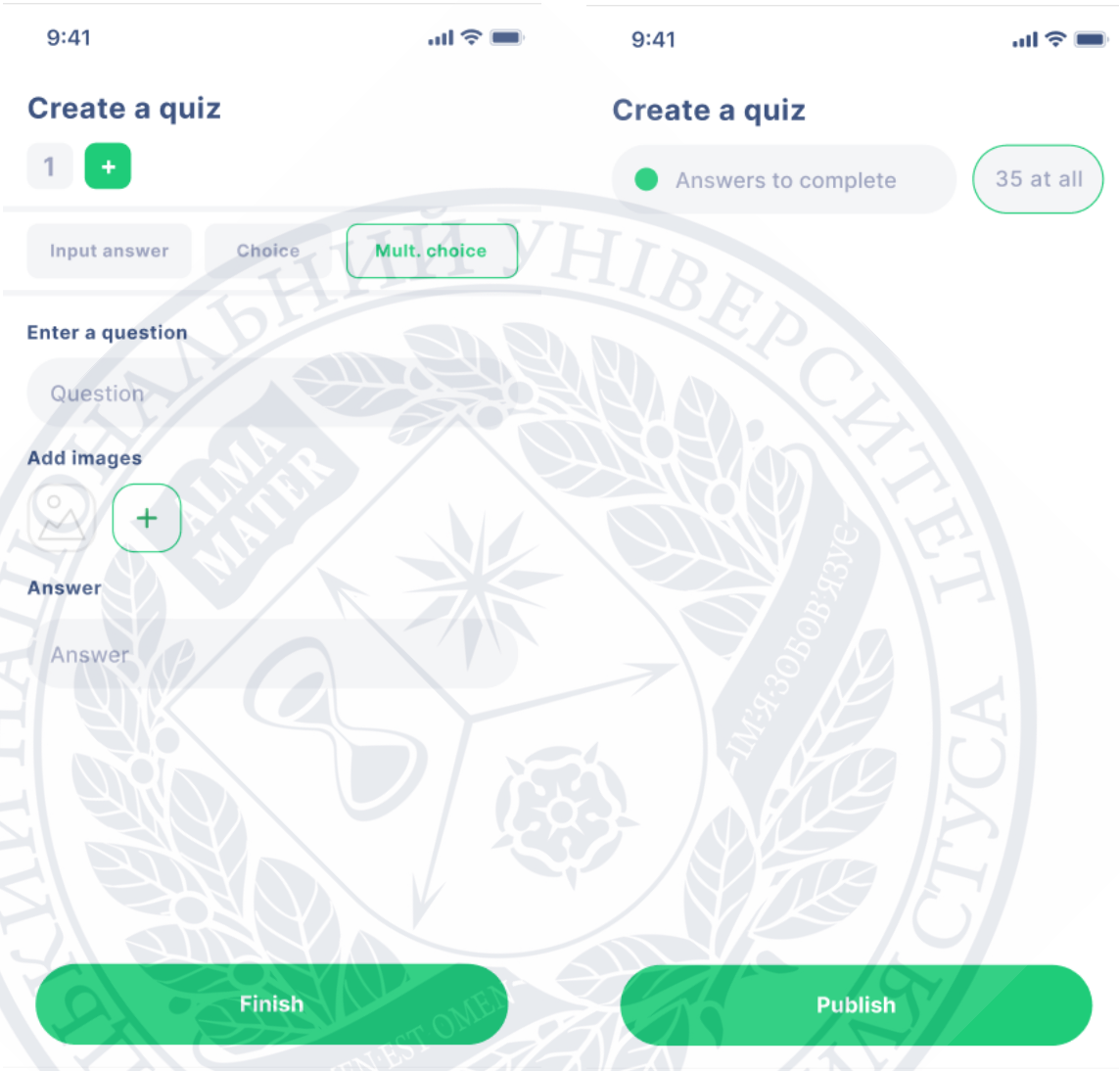

Рисунок 3.7 – Екран створення тесту, крок 2-3

Після натискання на кнопку «додати запитання» з’явиться екран із можливістю додавання запитання. На ньому ми маємо можливість ввести саме питання, додати фото, обрати тип відповіді та саму відповідь.

Створення тесту на кроці 3 нам дає можливість ввести кількість правильних відповідей, потрібних для успішного проходження

Екран детального огляду створеного нами тесту

Рисунок 3.8 – Екран детального огляду

Даний екран містить детальну інформацію про тест, таку як:

- збільний логотип
- назву
- автора
- кількість вподобань
- кількість питань
- список учасників, які брали участь у проходженні тесту

Елемент списку учасників, які брали участь у проходження тесту містить таку інформацію:

- Ім’я учасника
- Дату проходження
- Статус – склав/не склав
- Кількість правильних відповідей

4.2 Реалізація серверної частини

Структура серверної частини поділена на модулі відповідно свого призначення.

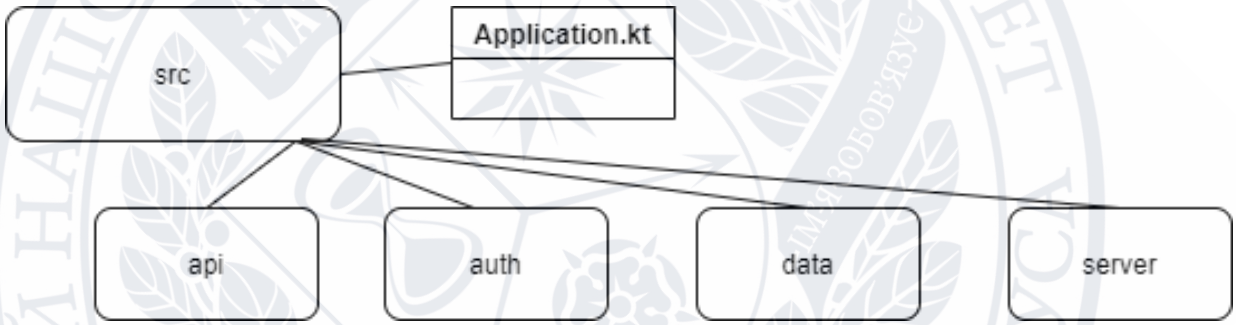

Рисунок 3.9 – Структура програми серверної частини

Api модуль

Даний модуль відповідає за взаємодію користувачів із REST [24] сервером.

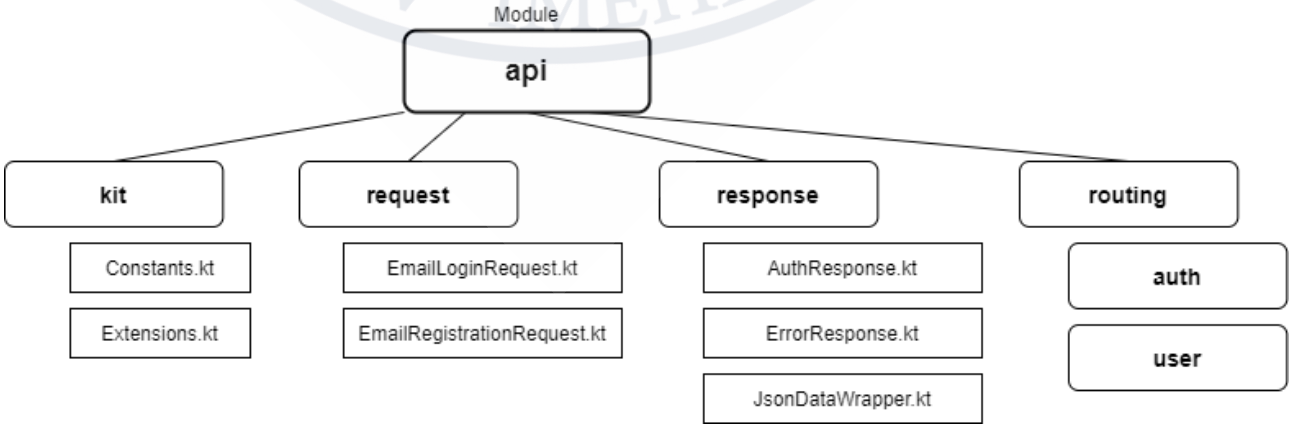

Рисунок 3.10 – API модуль серверної частини

- Kit – містить константи, необхідні для роботи з помилками, функції для конвертації String в Json потрібного формату
- Request – об’єкти, які ми будемо приймати в тілах запитів.
- Response – об’єкти, які ми будемо відправляти у відповідь в разу успішного виконання запиту
- Routing – функції, які призначені для обробки конкретних запитів

```
1 @Location(USER_ENDPOINT)
2 class User {
3    @Location("/quizzes")
4    class Quizzes(val page: Int, val size: Int)
5 }
```

Приклад організації оброки запитів

Наприклад, функція вище буде відповідати за обробку запиту за адресою /api/me/quizzes. Поля в класі Quizzes page та size нам потрібні для того, щоб реалізувати пагінацію.

```
1 getQuizzesByUserId(page: Int, size: Int, userId: Int)
  = dbQuery {
        i. Quizzes.select { Quizzes.userId eq userId }
              1..limit(n = size, offset = page * size)
2 .orderBy(Quizzes.createdAt to SortOrder.ASC)
```

Фрагмент коду реалізація пагінації за допомогою Exposed

Пагінація реалізована за допомогою функцій розширень бібліотеки Exposed, де в транзакції ми можемо вказати кількість елементів необіхдних для повернення (size в параметрі класу Quizzes), та offset, який обраховується за формулою

Offset = page * size,

Поле size нам також приходить як властивість класу Quizzes.

Authentication модуль

Містить в собі реалізацію алгоритму хешування пароля та додаткові функції в класі JwtService для генерації jwt токенів [26].

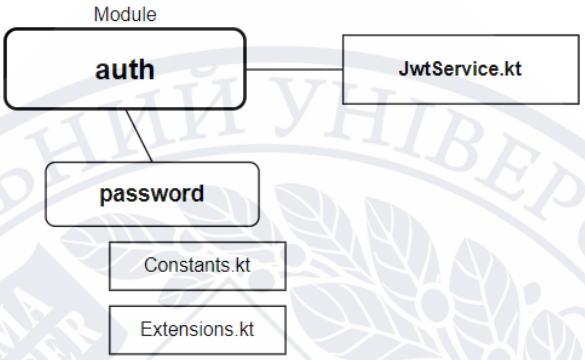

Рисунок 3.11 – Authentication модуль серверної частини

Server модуль

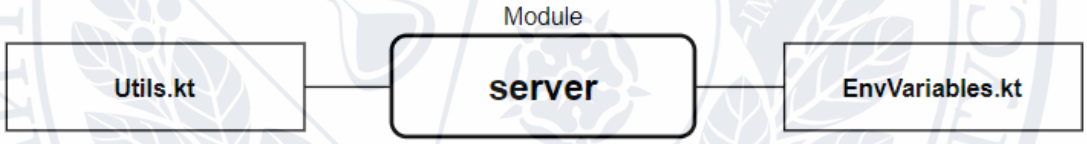

Рисунок 3.12 – Server модуль серверної частини

Містить в собі назви конфігураційних змінних для бази даних, jwt ключа та інші. Також містить допоміжні функції для роботи із конфігураційними змінними як на тестовому так і на віддаленому сервері.

Data модуль

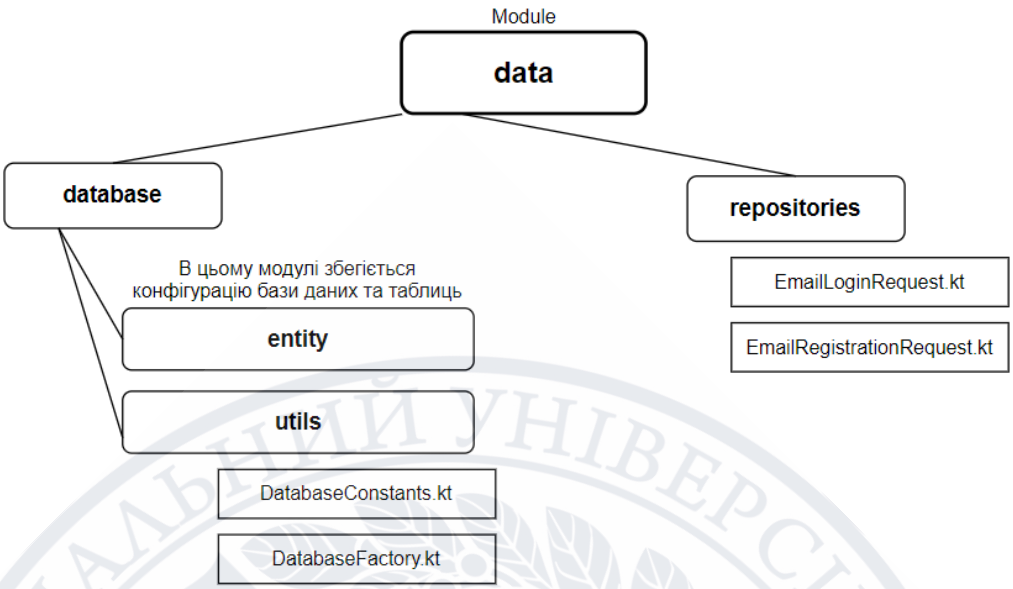

Рисунок 3.13 – Data модуль серверної частини

Містить в собі реалізацію репозиторіїв, необхідних для обробки користувацьких запитів. Також містить налаштування бази даних PostgreSQL та опис таблиць бази даних.

Розглянемо таблиці більше детально

Quizzes (Тести**)**

Таблиця призначена для зберігання даних про тести. Містить наступну інформацію:

1. userId – id користувача, яким тест був створений
2. name – назва тесту
3. description – опис тесту, коротка інформація про вміст
4. image – фото тесту
5. code – код тесту, представлений у форматі шести латинських літер з цифрами, наприклад (**f3A6g1**)
6. createdAt – дата створення тесту, в подальшому для сортування за часом створення і повернення коректної відповіді зі сторони сервера

Структура бази даних [47]

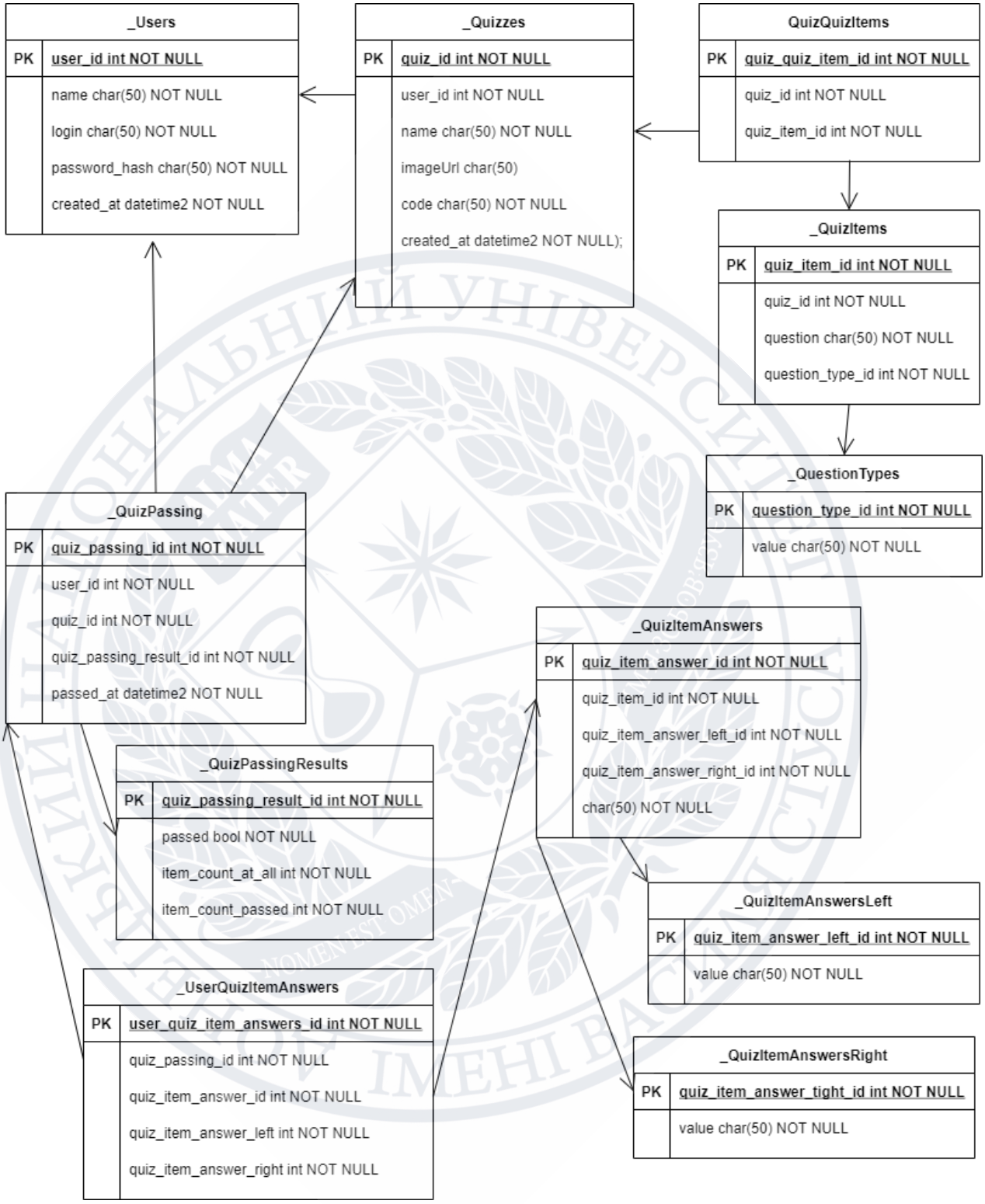

Рисунок 3.14 – Схема бази даних

Users (Користувачі**)**

Таблиця призначена для зберігання даних про користувачів. Містить таку інформацію як:

1. Імя користувача
2. Електронну пошту
3. Посилання на фотографію (буде добавлено в мабутньому)
4. Дату створення профілю
5. Хеш паролю (пароль не зберігається так як він є, а лише його хеш у цілях безпеки)

QuestionTypes

Таблиця-довідник призначена для зберігання обмеженої кількості елементів типу питання. Містить поле value – назва типу запитання. На момент написання дипломної роботи містить три значення:

- SINGLE_CHOICE – закрите питання із однією правильною відповіддю
- MULTIPLE_CHOICE – закрите питання із декількома правильними відповідями
- INPUT_ANSWER – питання із відкритою відповіддю. Може бути декілька правильних відповідей.

QuizItem (Запитання до тесту)

Таблиця містить інформацію про питання. Значення наступних полів такі:

1. Question – текст запитання
2. Image – фото до запитання, є опціональним
3. quizItemTypeId – id типу запитання

RealtionQuizQuizItems

Таблиця відношень для зберігання зв’язку між елементами таблиць Quiz та QuziItem. Одне запитання може відносити до декількох тестів.

- quizId – id елемента таблиці Quiz
- quizItemId – id елемента таблиці QuizItem

QuizItemAnswer

Інформація про відповіді до запитань міститься в трьох таблицях. Розбиття на три таблиці було необхідне для того, щоб ми мали можливість зберігати запитання різних типів і при додаванні нового структура нашої бази даних не змінювалась. Розглянемо приклад зберігання. Маємо запитання із декількома правильними відповідями: Виберіть правильні твердження стосовно мови програмування Kotlin:

1. Kotlin є строготипізованою мовою програмування (+)
2. Kotlin є виключено функціональною мовою програмування (-)
3. Kotlin має інструменти для забезпечення null-safety (+)

Як бачимо, ми маємо дві правильні відповіді. Таке відповіді до запитання будуть зберігатись у наступному вигляді:

QuizItemAnswerLeft

Таблиця 3.1 – Записи у таблиці QuizItemAnswerLeft

id	value
1	Kotlin є строготипізованою мовою програмування
2	Kotlin є виключно функціональною мовою програмування
3	Kotlin має інструменти для забезпечення null-safety

QuizItemAnsweRight

Таблиця 3.2 – Записи у таблиці QuizItemAnswerRight

id	Value
1	TRUE
2	TRUE
3	FALSE

RelationItemAnswers

Таблиця 3.3 – Записи у таблиці RelationItemAnswers

id	quizItemAnswerLeftId	quizItemAnswerRightId
1	1	1
2	2	3
3	3	2

Таким чином, у таблиці (3.3) відповіді зберігаються у вигляді відповідностей, де у випадку із типом запитання з декількома правильними відповідями права частина відповідності є просто значення вірно/невірно.

QuizzesPassing (Історія проходження тестів)
Дана таблиця містить інформацію про пройдені тести. Поля мають наступне значення:

- userId – id користувача, який проходив тест
- quizId – id тесту, який проходив користувач
- passedAt – дата проходження
- quizPassingResultId – id результату проходження

QuizPassingResults (Результати проходження)
Дана таблиця містить інформацію про результат проходження тесту. Поля мають таке значення:

- passed – логічне значення (склав/не склав)
- itemCount – кількість запитань тесту
- itemCountPassed – кількість запитань тесту, на які користувач дав правильну відповідь

UserQuizItemAnswers (Збереження відповідей користувача)
Таблиця зберігає інформацію про відповіді користувача на кожне запитання. Поля мають такі значення:

1. quizItemAnswerId – id попередньо вірного співвідношення відповідей
2. quizItemAnswerLeft – ліва частина відповіді яку обрав користувач
3. quizItemAnswerRight – права частина відповіді яку обрав користувач

3.3 Реалізація клієнтської частини

Існує досить багато підходів для побудови складних систем з правильною архітектурою. Усі вони мають багато відмінностей, але натомість і багато спільного. Визначимо основні принципи яким має слідувати кожна архітектура:

1. Незалежність від сторонніх бібліотек та фреймворків. Цей принцип нам говорить про те, що кожна архітектура має бути якомога гнучкішою та зручною для розширення. Використання великої кількості сторонніх бібліотек заставляє розробників слідувати реалізаціям бібліотек, що протирічить цьому принципу.
2. Тестування. Цей принцип говорить про те, що бізнес логіка додатку має покриватись тестами для більшої надійності розробки та швидкості знаходження несправностей.
3. Незалежність від реалізацій. Це принцип описує підхід, в якому ми повинні залежати від інтерфейсів (контрактів), а не від конкретних реалізацій. Ми повинні мати можливість швидко змінити реалізацію доступу до бази даних чи віддаленого серверу без зміни бізнес логіку додатку.

Clean Architecture Android

Clean Architecture – це концепція організації коду, запропонована Робертом Мартіном [48], основною ідеєю якої є розділення обов’язків між компонентів задля спрощення подальшого ускладнення системи із як найменшою затратою часу.

1. Шар даних (Data Layer)
2. Шар бізнес-логіки (Domain Layer)
3. Шар подання (Presentation Layer)

Задля забезпечення цієї концепції кожен шар використовує свою модель даних. Схема цих шарів виглядає наступним чином: [49]

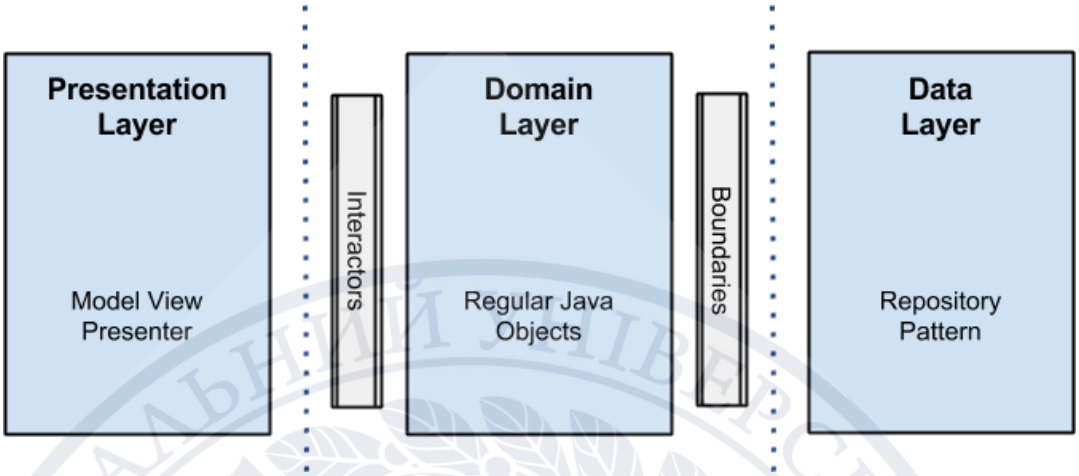

Рисунок 3.15 – шари архітектури в Clean Architecture

Окрім підходу до організації коду, розподілення класів по задачам, слід звертати увагу на архітектурні підходи реалізації та взаємодії об’єктів. У даній задачі був використаний MVVM паттерн проектування системи.

Model-View-ViewModel (тобто MVVM) [50] – архітектурний шаблон, автором якого є Джон Госсман. Даний підхід є альтернативою MVC та MVP. Він описує підхід, в якому шар подання відділений від бізнес логіки та відповідає лише за відображення даних. Які переваги в даного підходу:

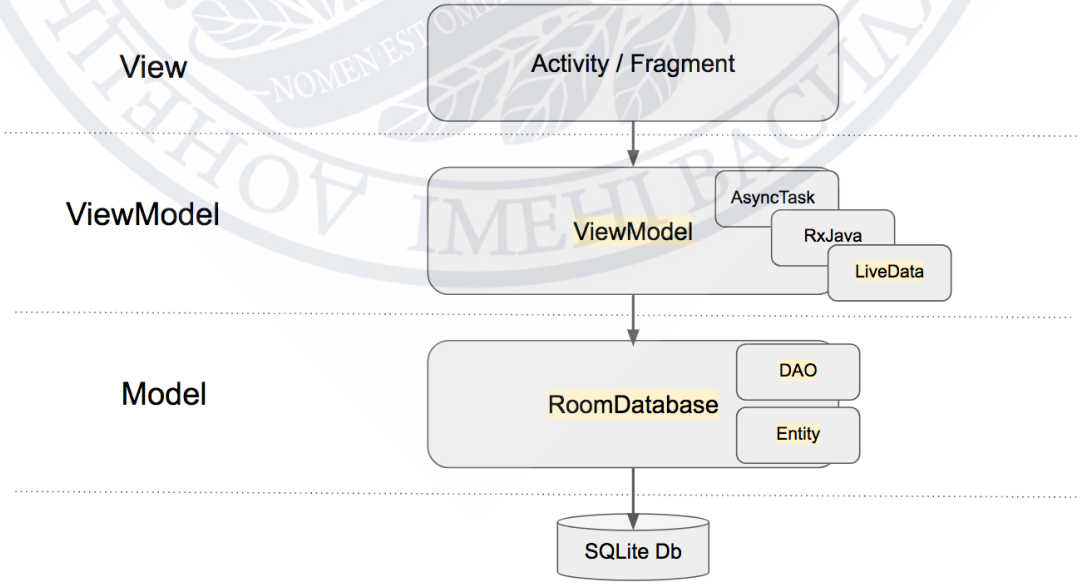

Рисунок 3.16 – Відношення компонентів у архітектурі MVVM

- Швидке розширення уже наявного функціоналу. Робота над компонентами може бути розділена між декількома розробниками;
- Можливість тестування. Розділення на компоненти дає змогу протестувати кожен компонент окремо;
- Розділення бізнес логіки. Кожен компонент відповідальний лише за свою конкретну задачу.

Розглянемо реалізацію даних підходів у розрізі клієнтського додатку.

Data модуль

Даний модуль є відповідальним за отримання інформації про користувачів, їхні тести та результати цих тестів

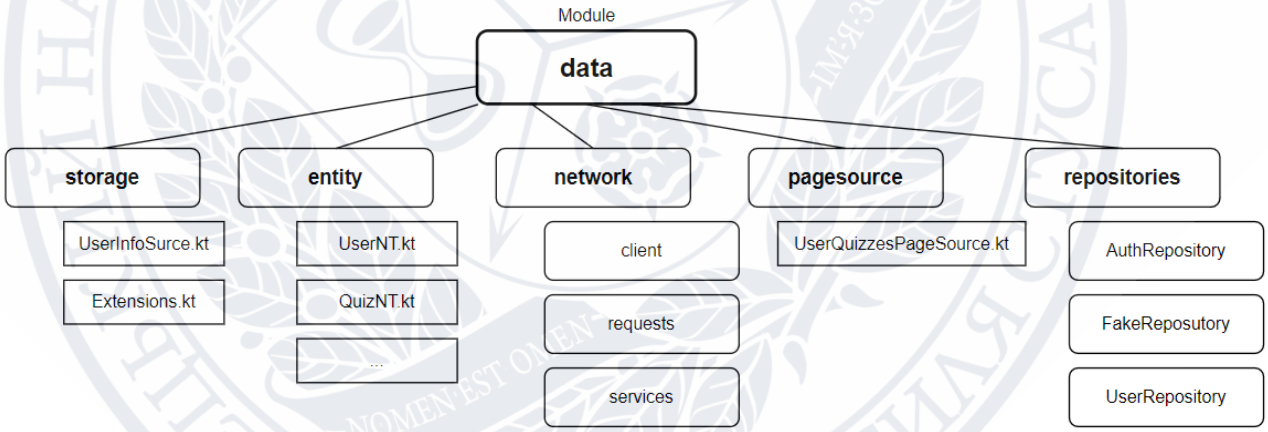

Рисунок 3.17 – Data модуль клієнтської частини

- Storage – містить реалізації зберігання інформації на локальному пристрої
- Entity – містить об’єкти, моделі даних, необхідні для коректного отримання інформації з віддаленого серверу
- Network – компоненти, необхідні для роботи із мережевими запитами.
- PageSource - містить реалізації джерела для отримання тестів із можливістю пагінації

- Repositories – компоненти, які об’єднують отримання інформації із віддалених серверів та її зберігання на локальному пристрої

Domain модуль

Модуль призначення для компонентів, які описують головні моделі додатку та інтерфейси репозиторіїв для їх подальшої реалізації

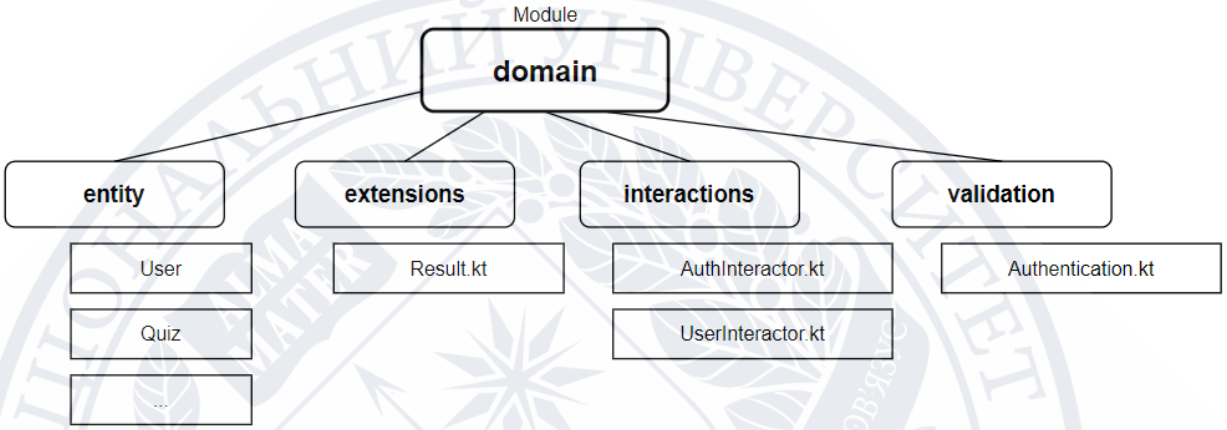

Рисунок 3.18 – Domain модуль клієнтської частини

- Entity – містить опис об’єктів, які безпосередньо відповідають за логіку додатку
- Extensions – містить функції, які допомагають у обробці отримання результатів
- Interactions – компоненти, які об’єднують у собі логіку роботи якогось конкретного функціоналу
- Validation – компоненти, які допомагають перевіряти дані на валідність

Presentation модуль

Даний модуль відповідає за логіку відображення отриманих даних

- Application – містить компонент, який відповідає за головну точку входу в додаток. Є відповідальним за ініціалізацію усіх необхідних компонентів

- Navigation – містить компоненти для навігації (переміщення між сторінками додатку)
- Pages – безпосередньо реалізація кожної сторінки
- Utils – допоміжні класи із функціями, які об’єднані для спільного використання

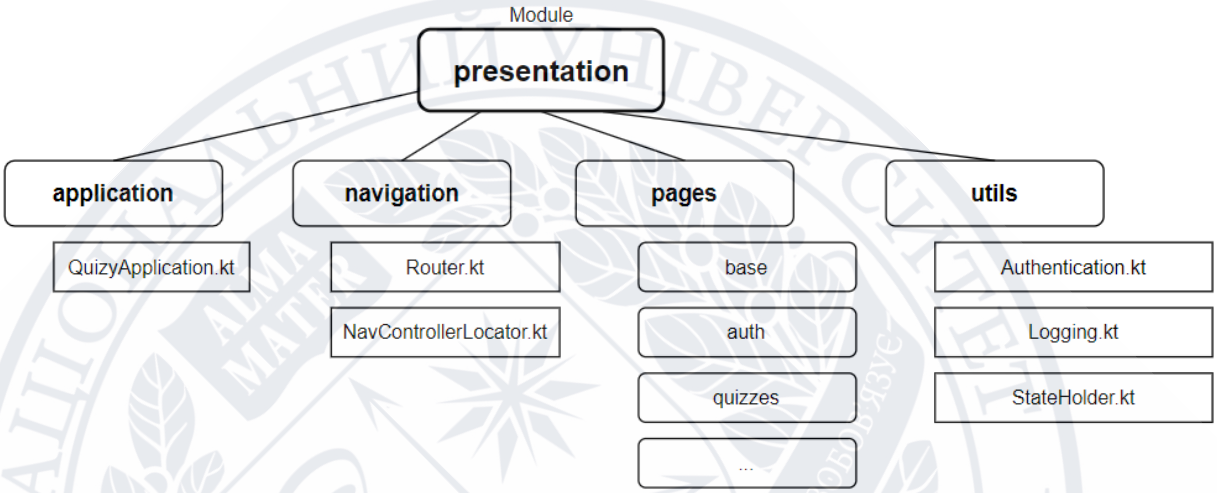

Рисунок 3.19 – Presentation модуль клієнтської частини

ВИСНОВКИ

У рамках даної роботи була розглянута актуальність мобільних додатків на сьогоднішній день, їх темпи розвитку порівняно із додатками на інших платформах.

Були проаналізовані сучасні інструменти розробки Android додатків, їх функціональні можливості та переваги.

Створений та протестований дизайн відповідно до UI/UX принципів user-friendly дизайн. Реалізована серверна частина згідно із REST концепцією. Набуті навички по впровадженні новітніх архітектурних підходів та патернів у реалізації клієнтського додатку під платформу Android.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Еволюція мобільних телефонів URL: https://tech.informator.ua/2018/11/25/evolyutsiya-mobilnyh-telefonov-chem-udivlyali-razrabotchiki/ (дата звернення: 31.05.2021)
2. Android URL: https://en.wikipedia.org/wiki/Android_(operating_system) (дата звернення: 31.05.2021)
3. Apple URL: https://en.wikipedia.org/wiki/Apple_Inc. (дата звернення: 31.05.2021)
4. Android vs Apple URL: https://www.businessnewsdaily.com/9906-ios-vs-android-business.html (дата звернення: 31.05.2021)
5. Covid-2019 URL: https://en.wikipedia.org/wiki/COVID-19 (дата звернення: 31.05.2021)
6. Mobile vs Web URL: https://www.uzhnu.edu.ua/uk/news/-yak-pratsyuye-distantsijne-navchannya-v-uzhnu.htm (дата звернення: 31.05.2021)
7. Система тестування та оцінювання знань URL: http://ap.uu.edu.ua/article/279 (дата звернення: 31.05.2021)
8. Moodle URL: https://moodle.org/ (дата звернення: 31.05.2021)
9. Kahoot URL: https://kahoot.com/ (дата звернення: 31.05.2021)
10. Generale Knowledge Quiz URL: https://play.google.com/store/apps/details?id=com.generalknowledgequiz.triviagames&hl=en_US&gl=US (дата звернення: 31.05.2021)
11. OpenTest URL: http://univer.nuczu.edu.ua/opentest2/ (дата звернення: 31.05.2021)

12. Антонов Ю.С. Оцінка повноти відповідей в автоматизованих системах контролю знань. Наукові праці Донецького національного технічного університету. Сер.: Інформатика, кібернетика та обчислювальна техніка. – 2012. № 15. С. 113–117.
13. Антонов Ю.С., Космінська О.М. Методика аналізу тестових завдань на основі отриманих результатів тестування. Інформаційні технології і засоби навчання. 2009. № 4. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/81/. https://doi.org/10.33407/itlt.v12i4.81.
14. Android Studio URL: https://developer.android.com/studio (дата звернення: 31.05.2021)
15. Android Studio Features URL: https://developer.android.com/studio/features (дата звернення: 31.05.2021)
16. Ktor URL: https://ktor.io/ (дата звернення: 31.05.2021)
17. Elements Of Kotlin, CommonsWare, Mark L. Murphy, 2021, 174p.
18. Head First Android Development, 3rd Edition, O'Reilly Media, Inc, Dawn Griffiths, David Griffiths, 2020, 452p.
19. Kotlin Programming for Beginners: An Introduction to Learn the Kotlin Programming Language with Tutorials and Hands-On Examples, Lightbulb Publishing, Nathan Metzler, 2020, 23p.
20. REST URL: https://docs.marklogic.com/guide/rest-dev.pdf (дата звернення: 31.05.2021)
21. Kotlin Apprentice (3rd Edition), Razeware LLC, Irina Galata, Joe Howard, Ellen Shapiro, 235
22. Programming Kotlin Applications: Building Mobile and Server-Side Applications with Kotlin, Wiley, Brett McLaughli, 2021, 145p.
23. PostgresSql URL: https://en.wikipedia.org/wiki/PostgreSQL (дата звернення: 31.05.2021)

24. PostgresSql URL: http://sd.blackball.lv/library/PostgreSQL_Server_Programming_Second_Edition_(2015).pdf (дата звернення: 31.05.2021)
25. СУБД для программиста URL: https://andpop.ru/courses/db_books/Tarasov.pdf (дата звернення: 31.05.2021)
26. Kotlin: программирование на примерах, СПб: БХВ-Петербург, Аделекан И., 2020, 124 ст.
27. Java to Kotlin (Early Release), O'Reilly Media, Duncan McGregor, Nat Pryce, 2021, 313.
28. JWT Handbook URL: https://assets.ctfassets.net/2ntc334xpx65/o5J4X472PQUI4ai6cAcqg/13a2611de03b2c8edbd09c3ca14ae86b/jwt-handbook-v0_14_1 (дата звернення: 31.05.2021)
29. Json URL: https://en.wikipedia.org/wiki/JSON#:~:text=JSON%20(JavaScript%20Object%20Notation%2C%20pronounced,(or%20other%20serializable%20values) (дата звернення: 31.05.2021)
30. Programming DSLs in Kotlin: Design Expressive and Robust Special Purpose Code, Pragmatic Programmers, LLC, Venkat Subramaniam, 2020, 130p.
31. Kotlin URL: https://en.wikipedia.org/wiki/Kotlin_(programming_language) (дата звернення: 31.05.2021)
32. Kotlin Coroutines URL: https://kotlinlang.org/docs/coroutines-overview.html (дата звернення: 31.05.2021)
33. Elements Of Kotlin Coroutines 0.3, CommonsWare, Mark L. Murphy, 2021, 174p.
34. Programming Android with Kotlin: Achieving Structured Concurrency with Coroutines (Early Release), O'Reilly Media, Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, Mike Dunn, 2021, 524p.
35. On Java 8, Leanpub, Bruce Eckel, 1241p
36. Kotlin at a Glance: Use of Lambdas and higher-order functions to write more concise, clean, reusable, and simple code, BPB Publication, Swati Saxena, 97p

37. Navigation Components URL: https://developer.android.com/guide/navigation (дата звернення: 31.05.2021)
38. Dependency Injection URL: https://www.programmer-books.com/wp-content/uploads/2019/01/Dependency-Injection (дата звернення: 31.05.2021)
39. Dependency Injection - Hilt URL: https://developer.android.com/training/dependency-injection/hilt-android (дата звернення: 31.05.2021)
40. Retrofit URL: https://square.github.io/retrofit/ (дата звернення: 31.05.2021)
41. Android KTX URL: https://developer.android.com/kotlin/ktx (дата звернення: 31.05.2021)
42. Recycler View URL: https://developer.android.com/jetpack/androidx/releases/recyclerview (дата звернення: 31.05.2021)
43. Android Studio Game Development: Concepts and Design, Apress, Jerome DiMarzio, 23p
44. Learn Android App Development: A step-by-step guide for non coders, Amazon.com Services LLC, Soyombo Soyinka, 124p
45. Exploring Android 2.0, CommonsWare, Mark L. Murphy, 523p
46. Df Android Studio 4.2 Development Essentials - Kotlin Edition: Developing Android Apps Using Android Studio 4.2, Kotlin and Android Jetpack, Payload Media, Neil Smyth, 2021, 125p.
47. Database Design URL: https://resources.saylor.org/wwwresources/archived/site/wp-content/uploads/2014/12/CS403-1.10-Database-Design-2nd-Edition-CCBY (дата звернення: 31.05.2021)
48. Clean Architecture URL: http://prof.mau.ac.ir/images/Uploaded_files/Clean%20Architecture_%20A%20Craftsman%E2%80%99s%20Guide%20to%20Software%20Structure%20and%20Design-Pearson%20Education%20(2018)[7615523] (дата звернення: 31.05.2021)

49. Architecting Android.The clean way URL: https://github.com/android10/Android-CleanArchitecture (дата звернення: 31.05.2021)
50. MVVM, MVP, MVC URL: https://pure.tue.nl/ws/files/48628529/Lou_2016 (дата звернення: 31.05.2021)