МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ПРИСІЧ АРТЕМ ВОЛОДИМИРОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____ Т. В. Нескородєва
«____» _____ 2021р.

РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ДЛЯ НАВЧАЛЬНОЇ СИСТЕМИ

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:
Римар П. В., старший викладач
кафедри інформаційних технологій

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2021

АНОТАЦІЯ

Присіч А.В. «Розробка серверної частини для навчальної системи» Спеціальність 122 «Комп'ютерні науки» Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній (бакалаврській) роботі досліджено технологію розробки серверної частини мобільного додатку для навчальної системи. За допомогою мови програмування Dart та фреймворка Flutter було розроблено мобільний додаток «Get Simple», який дозволяє користувачеві проходити онлайн-навчання за різноманітними курсами. Додаток зроблено як для викладача, так і для студента. Відповідно, можна реєструвати нові курси або бачити перелік курсів, на які вже записано зі статистикою проходження.

Ключові слова: мобільний додаток, Flutter, GetX, API, моделі даних, запит на сервер, Firebase.

45с., 21 рис., 20 джерел.

ANOTATION

Prisich Artem «Development of the server part for the educational system» Specialty122 «Computers Science» Vasyl Stus Donetsk National University, Vinnytsia, 2021.

In the qualification (bachelor's) work the technology of development of the server part of the mobile application for the educational system is investigated. Using the Dart programming language and the Flutter framework, the Get Simple mobile application has been developed, which allows the user to take online training in a variety of courses. The application is made for both teacher and student. Accordingly, you can register new courses or see a list of courses for which you have already registered with the statistics of passing.

Key words: mobile add-on, Flutter, GetX, API, models of data, fed to the server, Firebase.

45p., 21 pic., 20 sources.

ЗМІСТ

ВСТУП

Смартфони стрімко увійшли в наше життя, зробивше його більш комфортним, зручнішим та сучаснішим. З розвитком IT-технологій постійно розвивається і індустрія мобільних додатків. З більш ніж 2,96 млн додатків (і їх кількість зростає) в магазині Google Play, Android з моменту своєї появи набрав обертів і випередив конкурентів з часткою ринку більше 85%. Кожен рік Android продовжує розвиватися, пропонувати нові ідеї, інновації, методи і інструменти в розробці мобільних додатків. Такі шалені темпи розвитку цього напрямку (додатків для Android) є багатообіцяючі та дуже конкурентоспроможні.

Завдяки покращеному інтерфейсу впровадження Android зростало семимильними кроками. Щоб залишатися конкурентоспроможними в постійно мінливій індустрії розробки мобільних додатків, вкрай важливо завжди бути на крок попереду та втілювати в собі тенденції завтрашнього дня.

Перш ніж вирішите найняти професійного Android-розробника мобільних додатків, необхідно краще зрозуміти останні тенденції розробки додатків саме для цього напрямку.

Що дають мобільні додатки бізнесу:

- Допомагає організувати продажі і стимулювати повторні покупки.
- Підвищує лояльності покупців і клієнтів.
- Автоматизує бізнес-процеси.
- Дає можливість аналізувати аудиторії на основі різних даних.
- Прийом платежів.
- Надання сервісу нового рівня і т.і.

Умови сьогодення, а саме, пандемія COVID-19 та жорсткий карантин, рекордно збільшили використання мобільних додатків. Місяці в ізоляції і перехід бізнесу та навчання в онлайн формат сильно вплинули на спосіб життя та звички людини. Люди частіше купують в інтернеті, замовляють їжу онлайн, більше проводять часу у смартфонах, витрачають більше часу на ігри. Навіть управління державою переходить в формат «Держава в смартфоні». Все це призвело до

зростання попиту на різні мобільні додатки. Щомісячний час, що витрачається на мобільні додатки, зріс на 40 % у 2020 році, порівняно з 2019 роком. Така тенденція зберігається і у 2021 році. Відбулося зростання завантажень додатків для мобільних ігор (43%), для бізнесу (105%) і доставки їжі (73%). 2020-й також був роком відеоконференцій: кількість завантажень Zoom зросла на 2000% порівняно з 2019 роком. Додатки перетворилися в найбільшу споживчу систему на планеті: за прогнозами, у 2021 року загальносвітова виручка від додатків досягне $6,3 трлн.[1]

Створення платформи для навчання під час COVID-19 є не просто актуальною темою, а в загалі – обоязковою для нормального існування в сьогоднішніх умовах. За допомогою цієї платформи одні люди, не виходячи з дому, проходять курси для саморозвитку, поліпшенню професійних навиків, а інші люди навпаки надають корисну інформацію шляхом публікування своїх курсів на платформі. Майже все спілкування перейшло в онлайн-формат.

Переваги онлайн освіти:

- Доступ до програм найкращих викладачів світу: онлайн-освіта дає можливість навчатися скрізь, де є інтернет.
- Найновіша інформація, технології, теорії: матеріал онлайн-курсів оновлюється динамічніше, ніж офлайнові програми.
- Навчання безкоштовне або доступніше за ціною, ніж денне навчання в університеті.
- Можливість навчатись будь-де й будь-коли.
- Можливість поєднувати різні формати та підходи до навчання.

Недоліки онлайн освіти:

- Відсутність «живого» контакту з викладачем та групою.
- Ускладнення самомотивації та самоорганізації. Як наслідок, низький відсоток проходження курсів до кінця (у світі 7%, в Україні – 15%).

- Більшість матеріалів створені для вступного рівня, щоб охопити якомога більшу аудиторію. кількість спеціалізованих матеріалів вищого рівня складності дуже низька.
- Закінчення курсів не гарантує наявності потрібних знань і навичок.

Актуальність теми дослідження. Останні роки остаточно довели актуальність мобільних додатків**.** Сьогодні – це найпопулярніша тенденція IT-ринку. Сучасний мобільний додаток має декілька вирішальних переваг: швидкість та надзвичайна зручність для користувача. Створення такого додатку є завданням не з простих і має багато варіацій реалізації, оскільки кількість технологій стрімко зростає. Тому ця тема сьогодні є найактуальнішою.

Мета дослідження. Розробка мобільного додатку-платформи для навчання «Get Simple» з можливістю створення курсів, набором групи, отриманням сповіщень, повторним переглядом всіх курсів, відправкою запита на проходження курсу (все залежить від обраного функціоналу при реєстрації). Додаток має буди візуально приємним, інтуїтивно зрозумілим та багато функціональним.

Завдання дослідження:

- Вивчити та з’ясувати, як працюють Figma, Dart, Flutter SDK, Git, GitHub, Android Studio, Android Emulator, Firebase, Firebase Authentication, Firebase Firestore, Firebase Storage.
- З’ясувати, як розроблюються мобільні додатки для Android та iOS, застосовуючи SDK Flutter.
- Створити мобільний додаток «Get Simple».

Об’єкт дослідження. Методи проектування та технології створення мобільних додатків для ОС Android та IOS.

Предмет дослідження. Популярні інструменти, бібліотеки та фреймворки для створення мобільного додатку для ОС Android та IOS.

Досягнення поставленої мети потребує вирішення таких **задач дослідження**:

- Розгляд поняття мобільного додатку.

- Вивчення та обрання інструментів для створення мобільного додатку.
- Розробка дизайну та прототипу мобільного додатку.
- Створення моделі даних та інтегрування бізнес-логікі у проект.
- Створення мобільного додатку для ОС Android та IOS, що буде задовольняти меті роботи.

Робота складається зі вступу, трьох розділів зі своїми підрозділами, висновку та списку літературних джерел кількістю 20 найменувань та 21 рисунку. Загальний об’єм роботи 45 сторінок.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ МОБІЛЬНОГО ДОДАТКУ

Даний розділ присвячено опису поняття серверної розробки, наведено основні її характеристики.

1.1 Поняття серверної розробки

Backend-розробка – це набір апаратно-програмних засобів, за допомогою яких реалізована логіка роботи сайту. Попросту кажучи, це те, що приховано від очей користувача і відбувається поза його браузером і комп'ютером.

Наприклад, коли ви вводите запит на сторінці пошуковика і тиснете клавішу Enter, frontend закінчується і починається backend. Ваш запит відправляється на сервер Google чи іншої пошукової системи, де розташовані алгоритми пошуку. Саме там трапляється все «диво». Як тільки на моніторі з'явилася інформація, яку ви шукали – знову відбувається повернення в зону frontend.

За великим рахунком, сервер – це той же комп'ютер, тільки більш потужний. Він зберігає дані і відповідає на запити користувачів.

Backend-розробник застосовує ті інструменти, що доступні на його сервері. Він має право вибрати будь-яку з універсальних мов програмування, наприклад, Ruby, PHP, Python, Java. Все залежить від конкретного проекту і завдання замовника.

Також для backend-розробки використовуються системи управління базами даних:

1 MySQL.
2 PostgreSQL.
3 SQLite.
4 MongoDB.

Залежно від продукту, обов'язки backend-розробника сильно змінюються. Такий фахівець може створювати та інтегрувати бази даних, забезпечувати безпеку або налаштовувати технології резервного копіювання та відновлення.

Взаємодія frontend і backend відбувається по колу:

- frontend відправляє призначену для користувача інформацію в backend;
- інформація обробляється;
- і повертається назад, прийнявши зрозумілу форму.

Над цим працюють різні фахівці, але кожному з них має розуміти принципи, за якими працюють його колеги. Навіть дизайнеру інтерфейсів (UI-дизайнер) важливо хоча б в загальних рисах знати, що являє собою backend проекту, яким він займається. Це допоможе адекватно оцінити технічні можливості сайту або програми, та краще виконати свою задачу.

Взаємодія клієнта із сервером створюється за допомогою HTTP-запита, який безпосередньо відправляється на сервер, сервер шукає інформацію, вбудовує її в шаблон і повертає у вигляді HTML-сторінки.

Обов'язки frontend- і backend-розробників, як правило, розділені, але бувають моменти, коли програміст вирішує проблеми як на стороні сервера, так і в клієнтської частини. Обидва види розробки мають на увазі і технічні, і творчі компоненти. Нерідко на ринку зустрічаються фахівці, які впевнено почуваються як у frontend, так і в backend і можуть поєднувати їх.

1.2 Що таке API

API (application programming interface) – це інтерфейс прикладного програмування, набір методів, класів, бібліотек, функцій, що забезпечують взаємодію програм між собою.

Ця розробка була створена для того, щоб спростити роботу програмістів. Якщо спробувати дати визначення простими словами, то API – це зручний інструмент, який представляє собою набір класів, функцій, процедур, стандартів, які дозволяють додаткам ефективно взаємодіяти між собою. Програмісти використовують цей механізм при створенні самих різних систем.

Де використовують API?

Швидка реєстрація в додатках через акаунти в соціальних мережах. За допомогою спеціального API-протоколу в соціальній мережі Facebook та на інших платформах користувач отримує можливість використовувати спрощений доступ до продуктів компанії, пройшовши швидку реєстрацію на сайті через авторизацію свого аккаунта.

Google. Ця система використовує інтерфейс програмування для надання розробникам різних додатків доступу і можливості інтеграції інформації на різних сервісах. Наприклад, можна знайти і переглянути відеоролик з платформи YouTube прямо в додатку.

Надання API у вигляді готового продукту. Розробники пропонують доступ до свого додатком для отримання оперативних даних по метеорологічним зведеннями в будь-якій точці земної кулі і ін.

Основний функціонал API

Популярність API обумовлена простотою застосування і функціональністю. Програмісту не обов’язково вивчати внутрішні механізми дії цього інтерфейсу програмування, досить просто застосовувати його для об’єднання додатків в єдину систему.

Принцип роботи механізму API полягає в організації багаторівневої ієрархії, в якій підлеглі компоненти створюються з однаковою структурою. Вибудовується стандартна мережева модель OSI з певною кількістю ступенів (не менше 7). Внутрішні рівні класифікуються, виділяють додатки HTTP, IMAP, фізичні рівні трансляції та ін. Така побудова дозволяє використовувати в інтерфейсі функціонал нижніх API для роботи верхніх.

Для ефективної організації роботи створюються бібліотеки функцій і класів з описом сигнатур і семантики. Сигнатура в даному випадку є частиною оголошення функції, яка ідентифікує елемент. Її можна представити за допомогою різних мов програмування і визначити можливості перезавантаження. Описуючи мови виклику, фахівці поділяють сигнатури виклику і реалізації заданих функцій. Сигнатуру визначають, враховуючи

область видимості і послідовностей фактичних типів аргументів. Такі компоненти дозволяють компілятору розпізнавати функції при роботі з мовою C++. Якщо це метод певного класу, сигнатуру включають в ім’я даного класу.

Семантика функції описує її дію і принципи роботи. Вона описує результат обчислень і характеристики, від яких залежить його отримання. Тобто в таких моделях результат залежить не тільки від аргументів, а й від реального стану. При цьому не так і важливо, що API-з’єднання дає можливість отримувати інформацію.

Бібліотеки використовуються при написанні програм і додатків, створення сервісів для обслуговування клієнтів і багато чого іншого.

Типи API

Коли розробляються сайт з API або інші продукти, підбираються типи інтерфейсів, які підходять для вирішення тих чи інших завдань. Програмні інтерфейси класифікуються за переліком функцій, призначенню, виконуваних завдань і можливостям. Є стандартні продукти і альтернативні рішення, за допомогою яких можна вирішити ті ж проблеми іншими методами.

Виділяють глобальні продукти з окремими мовами програмування, призначені для вирішення локальних завдань. Є також продукти для управління різними графічними компонентами модулів програм. Саме графічні API призначені для поліпшення яскравості та чіткості зображень в комп’ютерних іграх, різних додатках з якісною візуалізацією. Чим складніше система інтерфейсу, тим більша ймовірність виникнення технічних складнощів при роботі з додатком. Які проблеми можуть виникати:

- складності портирування кодів програм при зміні API в системі. Такі труднощі можуть виникати при перенесенні модулів в іншу ОС;
- звуження переліку функцій продукту при застосуванні в іншому додатку (наприклад, при переході в систему з більш високим рівнем управління). Полегшуються типові завдання певного класу, але губляться деякі можливості (доступ до контролю над іншими регуляторами та ін.). Тобто управління

базовими елементами стає зручнішим і легким, але частина опцій залишається недоступною.

Крім того, існує і політика випуску API, яка визначає ступінь доступності технології різним користувачам. Так, виділяють політики:

- Private – API призначений тільки для внутрішнього використання;
- «Партнер» -API призначений для надання доступ до технології тільки окремим діловим партнерам;
- Public – API є публічним і відкритим всім.

API пошукових систем та веб-спеціалістів.

Фахівці, які займаються програмуванням і оформленням сайтів, а також їх просуванням, використовують спеціальний Web API. Це інтерфейси, які включають комплект певних HTTP-запитів. При отриманні такого роду запитів модуль генерує HTTP-відповіді певної структури. Щоб передавати інформацію, між ними застосовуються формати XML або JSON. У цій сфері застосування Web API – це своєрідна веб-служба з певними програмами та відповідними інтерфейсами. Щоб отримати доступ до даних модулів, потрібно пройти процедуру ідентифікації в Інтернеті за онлайн-адресою. Тобто при необхідності передачі даних на сервер потрібно використовувати серверний модуль взаємодії з API.

Для спрощення підбору розробники намагаються вкладати в назву інтерфейсу його призначення і ключовий функціонал (наприклад, API з ім'ям syngestureapisampleapp application створено для роботи «єдиний користувач»).

При підборі оптимального додатки веб-фахівцям потрібно враховувати зміни, які спостерігаються після масового застосування стандартів Web 2.0. Нововведення стосувалися протоколів обміну структурованих даних в SOAP (розподіл в обчислювальному середовищі, що стосується доступу до об'єктів). Ці протоколи були приведені до спрощеного архітектурному стилю. Це дало можливість прискорити процеси виконання заданих дій для інтернет-магазинів і інших онлайн-ресурсів з великим об'ємом інформації. Таким чином, розробник

при підборі додатки визначає, який саме інтерфейс необхідно застосувати для автоматизації всіх основних процесів.

Окремі компоненти системи взаємодіють між собою за аналогією зв’язків серверів і користувачів мережі Інтернет. Незважаючи на відсутність єдиних стандартів, системи на базі архітектури REST реалізуються із застосуванням класичних моделей HTTP, URL, JSON і XML. Такий підхід забезпечує можливість доповнень і розширень функціональності додатків.

Висновок до розділу 1

В цьому розділі було розглянуто загальне поняття серверної розробки. Визначено що таке API , де воно використовується , основний функціонал та його типи. Наступний розділ буде присвячено постановці задачі та інструментам для її розв’язку.

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ ТА ІНСТРУМЕНТИ

Даний розділ присвячено постановці задачі дослідження, а також опису необхідних інструментів для її реалізації.

2.1 Постановка задачі

Необхідно розробити макет та прототип мобільного додатку. Розробити моделі даних, інтегрувати бізнес-логіку та реалізувати логіку спілкування додатку із сервером, застосовуючи мову програмування Dart та фреймворк Flutter.

Завданням даної роботи є розробка клієнтської частини мобільного додатку та реалізація бізнес логіки навчальної системи «Get Simple». Мобільний додаток буде новою навчальною платформою, яка зможе надавати можливість користувачам створювати курси, купувати або відвідувати безкоштовні курсі, знаходити матеріал для саморозвитку і все це в комфортних умовах віддаленого користування.

Завдяки цьому додатку люди отримують можливість, не виходячи з дому , що на даний момент є актуальним та безпечним, підвищувати кваліфікаційні навички, навчатися чомусь новому, а особи, що володіють знаннями або ж цікавим матеріалом, зможуть викладати свої курси для її поширення.

2.2 Огляд аналогів

Coursera – Онлайн-платформа «Coursera», яка розпочала свою роботу в 2012 році, вже в цьому ж році в розділі «Освіта» рейтингу веб-сайтів журналу «Time» Coursera посіла перше місце. Перш за все, завдяки динамічному розвитку даний майданчик забезпечив собі лідируючу позицію: постійно розширювався список організацій-партнерів, стрімко зростала кількість курсів, а з ним і студентів . З Coursera почали співпрацювати не лише освітні установи США (Стенфорд, Прінстон, Каліфорнійський, Колумбійський університети),

університети окремих штатів (Теннессі, Нью-Йорк, Небраска), але і провідні культурні організації (наприклад, Музей природознавства США).

Udemy – Сайт MOOC-платформи «Udemy» має російськомовну версію, що робить навігацію по порталу максимально зрозумілою та комфортною. Освітні проекти представлені великім різноманіттям тематики. Вони розбиті на шістнадцять категорій, серед яких є комп'ютерні, гуманітарні дисципліни, а також хобі та рукоділля, мистецтво і фотозйомка. Курси Udemy читають фахівці з різних галузей, інструктори, які мають практичний досвід у бізнесі, менеджменті, фінансах, технологіях. Один з найпопулярніших курсів розроблений Джеком Уелчем, генеральним директором General Electric, топ-менеджером з колосальним управлінським досвідом.

Canvas Network – Проект Canvas Network відрізняється великою різноманітністю курсів, які проводять абсолютно різні за рівнем підготовки та напрямку діяльності люди: доктора наук, менеджери, письменники. Даний проект є дуже демократичним і легким. Курси не мають єдиного підходу до викладання. Матеріал можуть пояснювати виключно в коротких відеолекціях, доповнювати можливістю обговорювати прослуханий на форумі з викладачем та іншими студентами. Після закінчення курсу може видаватися сертифікат.

Проаналізувавши різні аналогічні навчальні платформи було вирішено створити власну платформу із більш сучасним дизайном та схожим функціоналом для надання можливості у період COVID-19, карантину та інших несприятливих умовах, здобувати освіту маючи лише телефон з інтернетом, що робить отримання таких можливостей більш зручним, комфортним та ефективним.

2.3 Огляд дизайнерських інструментів

Figma (Фігма) – це живий графічний онлайн-редактор для спільної роботи декількох розробників одночасно. У ньому можна створити прототип сайту, інтерфейс програми та обговорити правки з колегами в режимі реального часу.

Чому була обрана саме Figma:

1. Екосистема – під час створення веб-сайту або мобільного додатку, дизайнерам часто доводиться довантажувати файли на хмарні сховища, пересилати їх поштою тощо, що забирає багато часу та створює незручності під час роботи. Крім того, файли займають місце в пам'яті комп'ютера. За допомогою Figma ці проблеми успішно вирішуються, оскільки робочі файли знаходяться на власній хмарі, дизайнер може просто відправити посилання на файл клієнтові або верстальщику. Крім того, якщо на проекті змінюється фахівець, не виникне проблем з тим, де знаходяться вихідні дані.
2. Спільне редагування – групова робота над документами полегшує комунікацію і прискорює результат. Це прекрасно довели Google Docs. За допомогою Figm'и дизайнери, проектні менеджери і клієнти можуть одночасно коментувати, ставити запитання і правити макети, що дозволяє робити спільний проект більш живим та зручним у виконанні.
3. Багатозадачність – розробники часто скаржилися на продуктивність Sketch або Photoshop, коли потрібно було працювати одночасно з декількома робочими областями. Figma ж дозволяє працювати з більш ніж десятьма файлами одночасно, і продуктивність залишаємся такою ж високою.
4. Прототипування мобайл-проектів - найкраще переваги цього продукту розкриваються при розробці дизайну мобільних додатків. Коли проект складається з більш, ніж 30 екранів, стає досить складно переключатися з одного на інший. Figma дає можливість зібрати всі екрани в одному місці і ефективно управляти ними. Figma корисна і для програмістів, тому що дає можливість прямо в програмі подивитися інформацію про об'єкти, яка потрібні для коду [2].

2.4 Мова програмування Dart

Dart – це мова програмування загального призначення від компанії Google, яка націлена перш за все на розробку веб-додатків (як на стороні клієнта, так і

на стороні сервера) і мобільних додатків. Це також означає, що одну і ту ж програму на Dart можна компілювати під різні платформи - Windows (x86 / 64), Android, iOS [3]. Dart – об'єктно-орієнтована мова. Всі значення, які використовуються в програмі на Dart, представляють об'єкти [4]. У своєму розвитку Dart зазнав впливу більш ранніх мов, таких як Smalltak, Java, JavaScript. Його синтаксис схожий на синтаксис інших С-подібних мов. Це також мова від Google, але значно поліпшена в останні роки [6].

2.5 Інструменти для розробки

Flutter – безкоштовний і відкритий набір засобів розробки мобільного користувальницького інтерфейсу, створений компанією Google. Простіше кажучи, за допомогою Flutter можливо створити власне мобільний додаток з одним масивом коду. Це означає, що для створення двох додатків (IOS і Android) можна використовувати єдину мову програмування і одну базу коду[5].

Flutter націлений на дві важливі речі:

- SDK (Software Development Kit): набір інструментів, який допоможе у розробці додатків. Також він містить можливості для компіляції коду в нативному машинному коді (код для IOS і Android).
- Framework (Бібліотека призначеного для користувача інтерфейсу на основі віджетів): Колекція функціональних елементів призначеного для користувача інтерфейсу (кнопок, текстових вводів, повзунків і т.д.), які можна персоналізувати під особисті переваги.

Для розробки з Flutter використовується мова програмування Dart, тому що фокусується на розвитку верстки веб-сторінок; його можна з легкістю використовувати для створення мобільних і веб-додатків.

4 основні причини використовувати Flutter для MVP:

- **Це дешево**: для розроблення мобільного додатку за допомогою Flutter не потрібно створювати і підтримувати два мобільних додатки (один для IOS і Android).
- **Це просто**: для створення MVP досить одного розробника.

- **Це ефективно:** неможливо помітити різницю між нативним додатком і додатком Flutter.
- **Це красиво**: можна легко використовувати віджети, що надаються Flutter, і персоналізувати їх для створення оригінального призначеного для користувача інтерфейсу для ваших клієнтів.

GetX – це надлегке і потужне рішення для Flutter. Воно поєднує в собі інтелектуальне впровадження залежностей, високопродуктивне управління станом, управління маршрутами швидким і практичним способом.

GetX має наступні базові принципи, які є пріоритетом для всіх ресурсів в бібліотеці[5]:

- GetX при мінімальному споживанні ресурсів сфокусований на максимальну продуктивності .
- GetX використовує простий і приємний синтаксис. Не має значення, що розробник хоче зробити, завжди є більш легкий спосіб з GetX. Це заощадить час розробки та забезпечить максимальну продуктивність, яку може забезпечити програма.
- GetX дозволяє повністю розділити уявлення, логіку уявлення, бізнес-логіку, впровадження залежностей і навігацію.
- GetX НЕ роздутий. Він дуже функціональний, завдяки чому можливо розпочати програмувати, ні про що не турбуючись, тому що кожна з необхідних функцій знаходиться в окремих контейнерах і запускається тільки після використання. Якщо використовується тільки управління станом, то буде скомпільовано тільки управління станом. Якщо використовується маршрутизація, то нічого з управління станом не буде скомпільовано. Таким чином кожне рішення GetX було спроектовано так, щоб бути надлегким у використанні. Також в цьому є і заслуга Flutter, який вміє усувати невживані ресурси, як жоден інший фреймворк [7].

Ця архітектура має три головні стовпи :

- Керування станом.

- Керування маршрутом.
- Керування залежностями.

Саме завдяки останньому пункту у списку в додатку реалізовано впровадження залежностей або Dependency Injection.

Dependency Injection – це поняття вперше використано в статті Мартіна Фаулера «Inversion of Control Containers and the Dependency Injection Pattern». Інверсія управління – концепція, що лежить в основі впровадження залежності. Це означає, що клас не повинен конфігурувати свої залежності статистично, а повинен бути налаштований іншим класом ззовні.

Clean Architecture – як правило, в Clean Architecture код розділений на кілька рівнів з одним правилом залежності: внутрішній рівень не повинен залежати від будь-яких зовнішніх рівнів. Це означає, що залежності повинні вказуватися всередині кожного рівня, щоб не було залежностей між рівнями (шарами) [11].

Clean Architecture, робить код:

- Незалежних від фреймворків.
- Тестованим.
- Незалежних від UI.
- Незалежних від Бази даних.

Незалежним від будь-якого зовнішнього впливу

Stacked Themes - ця бібліотека, яка являє собою набір віджетів та класів, що допомагають керувати кількома темами або функцією темної / світлової теми в додатку Flutter. Цей пакет містить кілька базових класів, що полегшують управління темами при створенні вашого додатка з управління темами. Stacked Themes надає основну функціональність заміни даних ThemeData, з наданих розробленого додатку, до яких можна отримати доступ за допомогою Theme.of(context). Крім того, він також надає низку допоміжних функції, за допомогою яких можна змінити будь який колір.[13]

Бібліотека використовує Theme Manager для управління всіма функціоналами теми. Для початку потрібно викликати функцію ініціалізації

Theme Manager перед запуском програми. Потрібно змінити основну функцію на фьючер функцію, і зробити її асинхронною щоб можна було чекати ініціалізацію виклику статичної функції на Theme Manager. Починати треба з обгортання нашого Material App або будь-якого іншого віджета Theme Builder. Функція конструктора повертає контекст, regular Theme, dark Theme та theme Mode. Це відновить будь-яку програму та передасть нову тему через контекст, який може бути використаний у розробляємому додатку.

Theme Builder має властивість під назвою themes, де можна знайти список Theme Data. Зазвичай це лише 2 або 3, світла, темна, нейтральна, але можна зберігати їх у списку, щоб мати їх скільки завгодно. Щоб підтримувати «чистоту», необхідно повернутися до списку тем із функції, створеній в іншому файлі (theme_setup). Це досить проста функція, але вона може почати виглядати досить великою, враховуючи, скільки властивостей теми будете використовуватися у темах. Саме тому рекомендують все це зберігати в окремому файлі. Спосіб обміну темами відбувається шляхом встановлення індексу теми, яку можливо буде використовувати.

Firebase – це платформа розробки мобільних додатків, з дуже величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих кросплатформених додатків. Головна перевага платформи в тому, що вона дозволяє розробнику не відволікатися на створення бекенд, тобто прихованої від користувача програмної частини проекту, наприклад, серверного коду. І це спрощує і прискорює створення мобільних додатків, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді. Саме зв'язка Firebase з фреймворком Flutter дозволяє програмістам компанії AVADA MEDIA створювати швидкі програми для Android і iOS, що дозволяють швидко і ефективно вирішувати найрізноманітніші завдання .[14]

Firebase – це одне з BaaS-рішень (Backend as a Service), яке дає розробнику масу можливостей у його роботі. Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає

розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі. Додаток підключається до бази даних через WebSocket, який відповідає за синхронізацію даних протягом усього сеансу. Це дуже зручно та економить робочий час.

Також Firebase виступає в якості сховища файлів. Firebase Storage забезпечує надійне завантаження і вивантаження файлів для додатка. Хмарне зберігання файлів відео, аудіо або будь-якого іншого типу підтримується Google Cloud Storage. Вміст хмарного сховища надійно захищене. Надійність та зручність – це підтвердження функціональності Firebase.

Firebase надає такі функції, як аналітика, бази даних, обмін повідомленнями та звіти про аварійне, тому можна швидко працювати і зосередитися на своїх користувачів [15].

GitLab - веб-додаток і система управління репозиторіями програмного коду для Git.

GitLab пропонує рішення для зберігання коду та спільної розробки масштабних програмних проектів. Репозиторій включає в себе систему контролю версій для розміщення різних ланцюжків розробки та гілок, дозволяючи розробникам перевіряти код і відкочуватися до стабільної версії софта в разі непередбачених проблем.[12]

GitLab є конкурентом GitHub, в якому серед багатьох інших проектів розміщується розробка ядра Linux. Оскільки GitLab розробляється на тій же основі управління версіями (Git), принцип їх роботи однаковий. GitLab підтримує як публічні, так і необмежену кількість приватних гілок розробки[9].

Можливості GitLab діляться на наступні категорії:

- управління (Manage),
- планування (Plan),
- створення (Create),
- перевірка (Verify),
- упаковка (Package),
- безпеку (Secure),

- релізи (Release),
- конфігурація (Configure),
- моніторинг (Monitoring),
- захист (Defend).

Управління

- Аутентифікація і авторизація. Двофакторна аутентифікація, інтеграція до призначених для користувача каталогами (AD / LDAP), гранулярний доступ до об’єктів в GitLab, підтримка токенів і SSO.
- Аналітика. Аналітика продуктивності розробників, трекінг виконання завдань групами користувачів.

Планування

- Відстеження проблем. Контроль за статусом проблем, підтримка дискусій користувачів для обговорення проблеми, можливість прикріплення файлів.
- Контроль часу. Звіти по витраченому часу і оцінка часу, що залишився на виконання завдань.

Створення

- Управління вихідним кодом. Графік комітів, запити на злиття гілок розробки, інтеграція з Jira.
- Веб-консоль для редагування коду. Веб-уявлення коду в інтерфейсі, редагування коду, синхронізація файлів з вихідним кодом.

Перевірка

- Підтримка процесу Continuous Integration (CI). Вбудовані інструменти CI / CD, інтеграція з Github, перегляд Пайплайн розробки, онлайн-візуалізація HTML-артефактів.
- Перевірка якості коду і тестування. Звіти за якістю коду, юніт-тестів, тестування навантаження, тести на доступність і юзабіліті.

Упаковка

- Управління репозиторіями. Підтримка репозиторіїв C / C ++, Maven (Java), NPM, NuGet (.NET), Composer (PHP), PyPi (Python) та інших.
- Управління контейнерами. Підтримка роботи з Docker, управління репозиторієм через API і вебхукі, приватних контейнерних репозиторіїв.

Безпека

- Підтримка SAST і DAST. Робота з Static Application Security Testing і Dynamic Application Security Testing включаючи можливості звітності.
- Сканування залежностей і управління уразливими. Gitlab підтримує автоматизоване виявлення залежностей в коді і дозволяє будувати звіти по можливим вразливостей.

Релізи

- Підтримка процесу Continuous Delivery (CD). Можливість запуску CI / CD в різних середовищах (Windows, Mac, Linux), підтримка канаркових релізів, забезпечення безпеки Пайплайн.
- Оркестрації релізів. Відстеження релізів, асоціація релізів з етапами, управління доступом до захищених оточенням.

Конфігурація

- Управління Kubernetes. Підтримка роботи з декількома кластерами Kubernetes, розгортання в кластері Kubernetes, управління змінними в залежності від оточення.
- ChatOps і бессерверной обчислення. Розгортання та інші операції з чату і підтримка виконання функцій через Knative.

Моніторинг

- Метрики. Моніторинг продуктивності додатків, кластерів kubernetes і самого Gitlab з можливістю відправлення повідомлень.
- Управління інцидентами і логування. Автоматичне створення інцидентів в разі перевищення порогів і відправка логів в зовнішні системи.

Захист

- Web Application Firewall і безпеку контейнерів. Блокування атак на веб-інтерфейс і відстеження життєвого циклу контейнерів.
- Мережева безпека. Підтримка мікросегментації контейнерів для ізоляції потенційно небезпечних контейнерів і застосування політик безпеки.

Android Studio – інтегроване середовище розробки (IDE) для роботи з платформою Android. Дана IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіна Android Development Tools (ADT) для Eclipse [16].

Android Studio заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains – це офіційний засіб розробки Android додатків. Дане середовище розробки доступне для Windows, macOS і Linux. 17 травня 2017 на щорічній конференції Google I/O, Google анонсував підтримку мови Kotlin, що використовувалась в AndroidStudio, як офіційної мови програмування для платформи Android на додаток до Java і C++.

З кожною новою версією Android Studio з'являються все нові і нові функції – більш сучасні та більш прогресивні. На даний момент доступні наступні функції:

- Розширений редактор макетів: WYSIWYG, функція попереднього перегляду макета на декількох конфігураціях екрану, здатність працювати з UI компонентами за допомогою Drag-and-Drop,
- Збірка додатків, заснована на Gradle.
- генерація кількох .apk файлів і різні види збірок
- Рефакторинг коду.
- Статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій і інше.
- Вбудований ProGuard і утиліта для підписування додатків.
- Шаблони основних макетів і компонентів Android.

- Підтримка розробки додатків для Android Wear і Android TV.
- Вбудована підтримка Google Cloud Platform, яка включає в себе інтеграцію з сервісами Google Cloud Messaging і AppEngine.
- Android Studio 2.1 підтримує Android N Preview SDK, а це значить, що розробники зможуть почати роботу зі створення програми для нової програмної платформи.
- Нова версія AndroidStudio 2.1 здатна працювати з оновленим компілятором Jack, а також отримала покращену підтримку Java 8 і вдосконалену функцію InstantRun.
- Починаючи з Platform-tools 23.1.0 для Linux виключно 64-розрядна.
- В AndroidStudio 3.0 по стандарту включені інструменти мови Kotlin засновані на JetBrains IDE.

Висновок до розділу 2

В даному розділі було розглянуто аналоги платформ для навчання, проаналізовані їх сильні сторони, а також всі інструменти, які обрані для розробки. Обрання правильних інструментів значно спрощує розробку продукту та робить програмний код легко розширюваним та зрозумілим , що дуже важливо для підтримання готового, робочого продукту.

РОЗДІЛ 3
РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ МОБІЛЬНОГО ДОДАТКУ

В даному розділі описано процес створення серверної частини мобільного додатку. Наведено екрани взаємодії користувача з додатком.

3.1 Створення прототипу

Першим кроком у створенні мобільного додатку був прототип. Він потрібен для того , щоб розробник міг чітко розуміти як має працювати програма, зробити так, щоб усі кроки у додатку були інтуїтивно зрозумілі користувачеві. Прототип було створено для двох різних функціоналів.

Дві найпотужніші функцій Figma – це компоненти та стилі. Вони дозволяють вам повторно використовувати об’єкти і атрибути інтерфейсу, щоб систематично підтримувати дизайн в масштабі. Коли вам потрібно щось змінити, наприклад, колір посилання вашого бренду або домашню іконку. Ви можете внести зміну один раз – в вихідний основний компонент або стиль і подивитися, як він оновиться в усіх ваших проектах.[17]

Також можна використовувати компоненти та стилі:

- В межах одного окремого файлу в безкоштовній версії Figma
- Для різних файлів і проектів в пакеті Figma Professional
- Між командами в пакеті Figma Organization

Для більш зручнішого користування готовим продуктом, при створенні найменування та управлінні компонентами і стилями необхідно було враховувати нескінченну кількість нюансів, тому було написано спеціальне керівництво, яке допоможе користувачу.

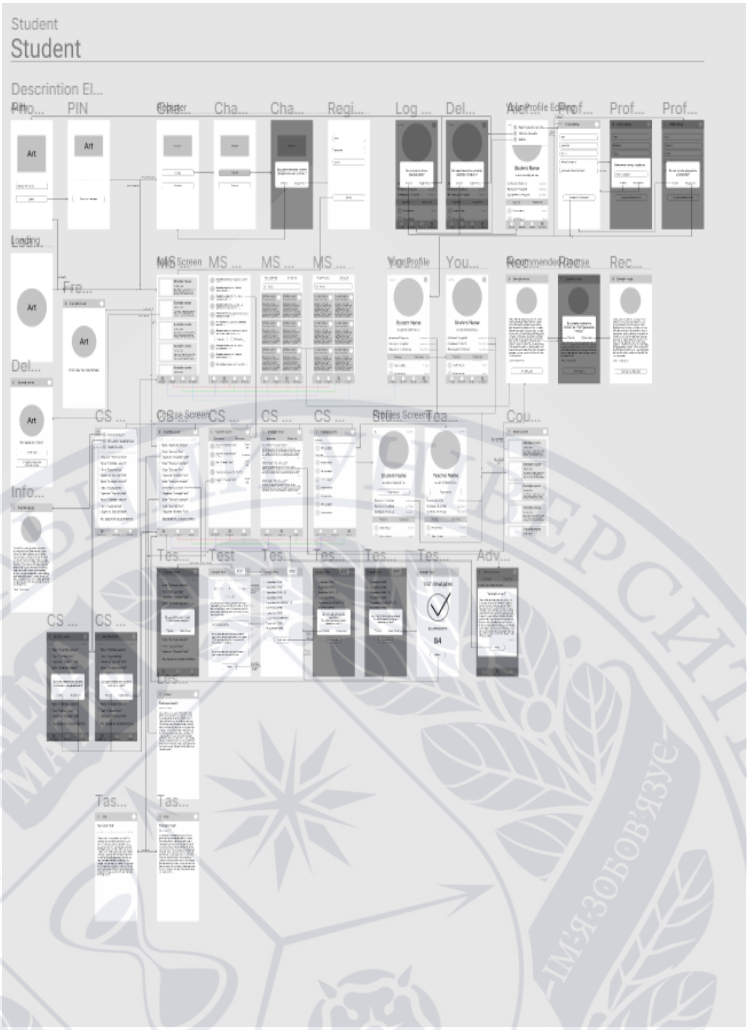

Рисунок 3.1 – Скріншот прототипу для Студента

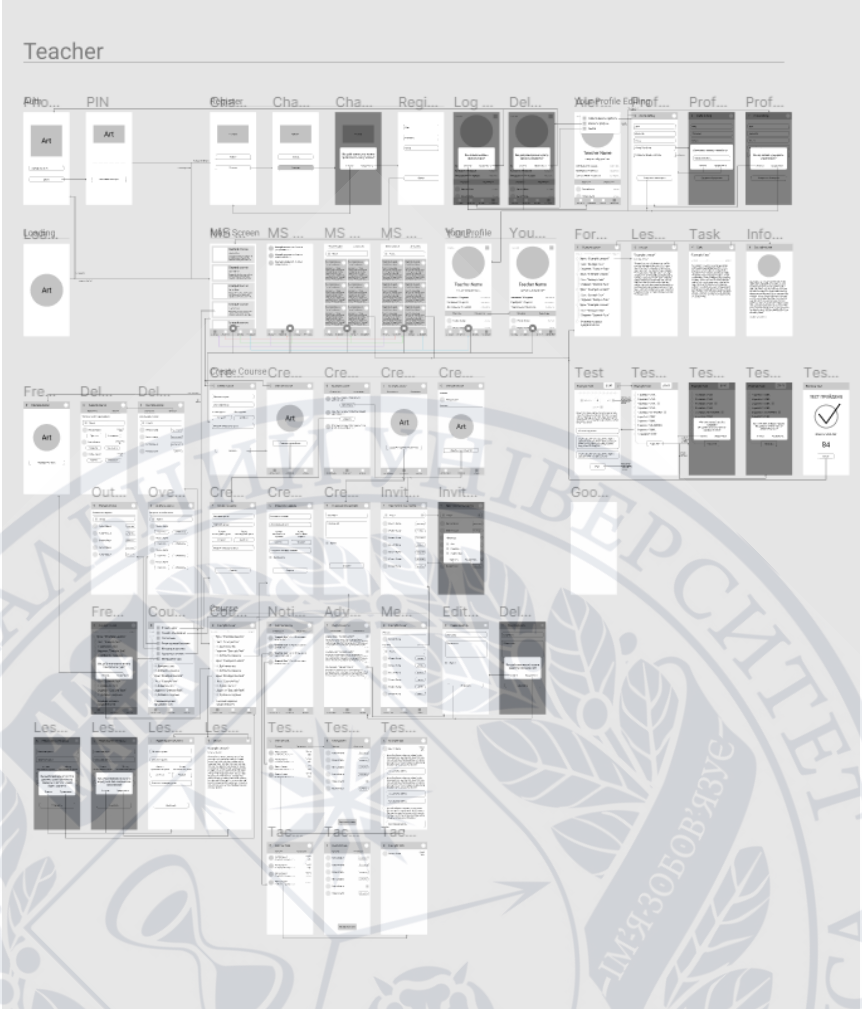

Рисунок 3.2 – Скріншот прототипу для Викладача

Було розроблено максимально зрозумілу логіку користування додатком для юзера.

3.2 Створення архітектури проекту

Архітектуру було обрано GetX з розмежуванням на модулі відповідальності як було описано в Clean Architecture

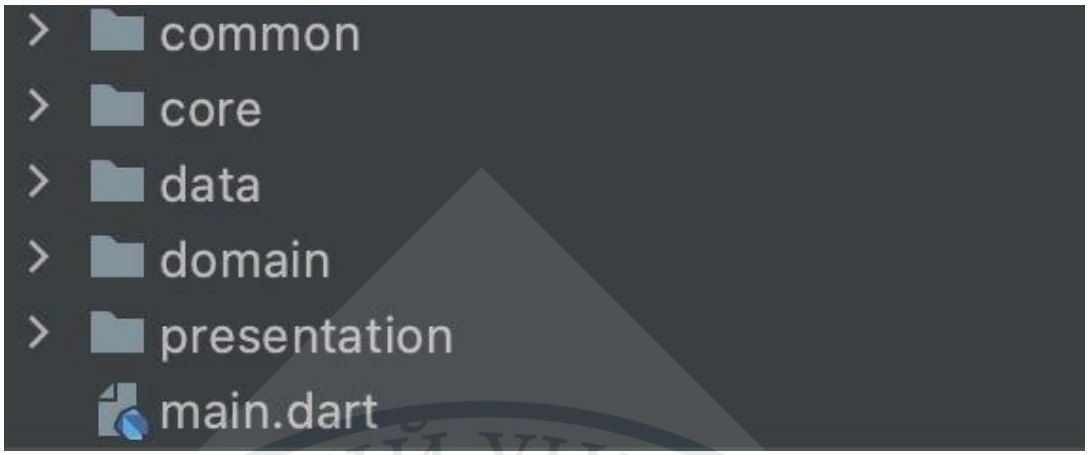

Рисунок 3.3 – Скріншот архітектури додатку

3.3 Створення моделей даних

Створення моделей даних для коректного спілкування сервера із користувачем . У додатку є всього 3 моделі даних: модель юзера, модель курсу та модель завдання.

У моделі User є поле тип, яке дає зрозуміти, який функціонал обрав користувач і які доступи, права, та функціональність можна йому надавати.

У моделі Course є поле файлу, який можна завантажити на сервер та поширити серед студентів. Також є поле з uid-ом користувача, що створив даний документ.

У моделі Task є списки даних із питаннями до пройденого курсу, uid курса, до якого відноситься дана модель та правильні варіанти відповідей.

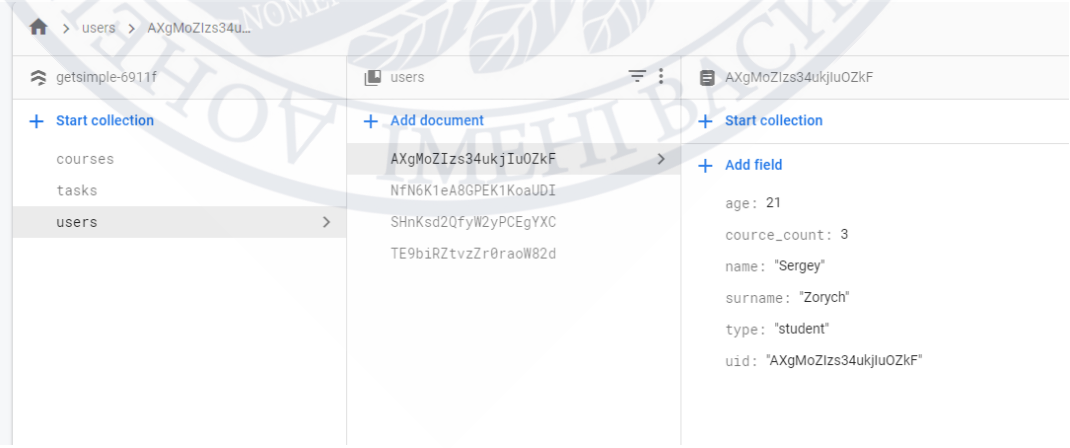

Рисунок 3.6 – Структура моделей на прикладі моделі Юзера

Було обрано саме Cloud Firestore тому що:

- Кращі запити і більш структуровані дані – в той час, як база даних реального часу – це просто гігантське дерево JSON. Cloud Firestore трохи більше структурований. Всі ваші дані складаються з документів (які в основному є сховищами ключових значень) і колекцій (які є колекціями документів). Документи також часто вказують на вкладені колекції, які містять інші документи, які самі можуть містити інші документи і т.д. Ці структуровані дані допомагають вам двома способами. По-перше, всі запити є неглибокими, що означає, що ви можете запросити документ, не захоплюючи все дані під ним. Це означає, що ви можете зберігати свої дані ієрархічно таким чином, щоб це мало для вас більше сенсу, не турбуючись про те, щоб ваша база даних залишалася неглибокою. По-друге, у вас є більш потужні запити. Наприклад, тепер ви можете виконувати запити по декількох полях без необхідності створювати ті поля «combo», які об'єднують (і денормалізують) дані з інших частин вашої бази даних. У деяких випадках Cloud Firestore просто запускає ці запити безпосередньо, а в інших випадках він автоматично створює і підтримує індекси для вас.[18]
- Розроблений для масштабування Cloud Firestore зможе масштабуватися краще, ніж база даних реального часу. Важливо відзначити, що запити масштабування відповідно до розміру результуючого набору, а не набору даних. Таким чином, пошук буде залишатися швидким, незалежно від того, наскільки великим може стати ваш набір даних.
- Більш проста ручна вибірка даних – як і в базі даних реального часу, ви можете налаштувати прослуховувачі в Cloud Firestore для потокової передачі змін в режимі реального часу.
- Підтримка декількох регіонів - це в основному означає більшу надійність, оскільки дані спільно використовуються відразу декількома центрами обробки даних. Але існує сильна узгодженість, тобто завжди можна зробити запит та отримати останню версію своїх даних.

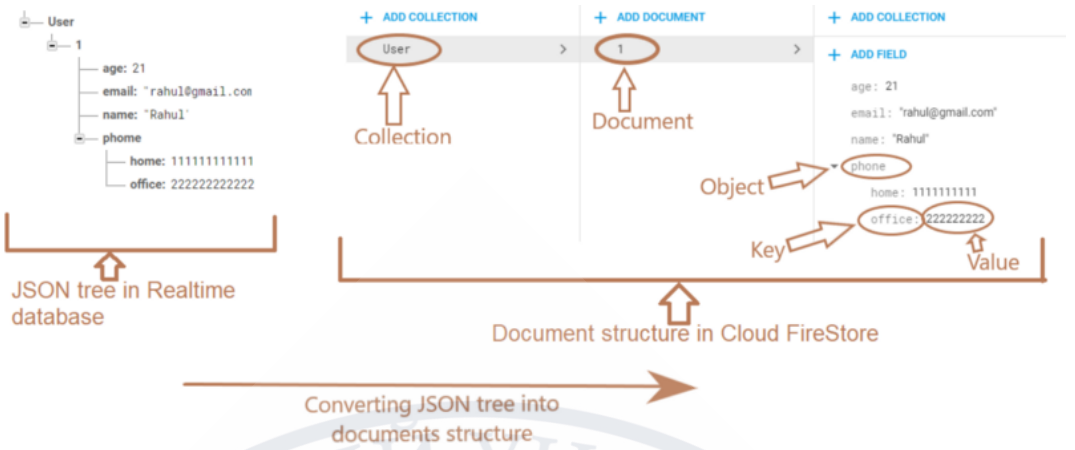

Рисунок 3.7 - Різниця між Cloud Firestore і Realtime Database

3.4 Інтегрування бізнес-логіки

Призначена для користувача аутентифікація є важливою вимогою для більшості додатків Android сьогодні. Маючи можливість надійно аутентифікувати своїх користувачів і, таким чином, унікально ідентифікувати їх, ви можете запропонувати їм індивідуальний досвід на основі їхніх інтересів і переваг. Ви також можете переконатися, що у них немає проблем з доступом до своїх особистих даних при використанні вашого додатку з декількох пристроїв.

За замовчуванням Firebase не дозволяє аутентифікацію користувача. Таким чином, ви повинні вручну включити аутентифікацію користувача на основі пароля в консолі Firebase. Для цього перейдіть в розділ Auth і натисніть кнопку «Налаштування входу в систему». Тепер ви побачите список всіх доступних вхідних провайдерів.

Після успішної реєстрації користувач буде автоматично залогінений. Користувач буде залогінений, навіть якщо ваш додаток перезавантажиться.

Рисунок 3.8 – Скріни аутентифікації юзера через мобільний телефон

Після підтвердження номера телефона кодом , що був відправлений після введення, юзер потрапляє на екрани реєстрації своїх даних для зберігання їх в базі та вибір функціонала, який його цікавить.

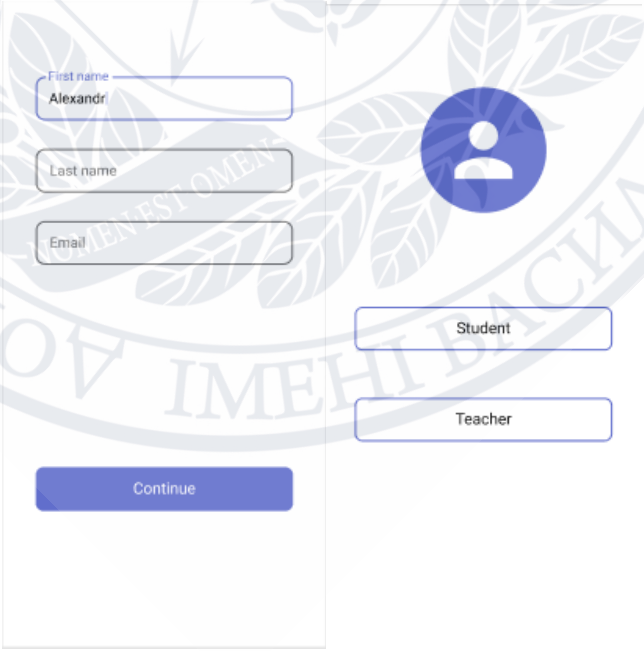

Рисунок 3.9 - Скріншоти з додатку введення юзером персональних даних та вибора функціоналу

Після чого юзеру показується головний екран із запитом на отримання усіх курсів на які підписався юзер з бази даних

Рисунок 3.10 – Скріншот головного екрану, таба «Мої курси»

В залежності від обраного функціоналу на цьому екрані будуть показуватися або створені вчителем курси або ж курси, у яких приймає участь студент.

Наступна вкладка – це сповіщення. Також, в залежності від функціоналу, на цьому екрані будуть відображені різні сповіщення.

Рисунок 3.11 – Скріншот головного екрану , таба «Сповіщення»

Як можна побачити на скріншотах, у цій вкладці юзер може побачити інформацію з курсів, в яких він приймає участь, запрошення на курси, статус запрошення.

Наступний екран – всі курси. На цьому екрані при потраплянні юзера на нього виконується запит до бази для вивантаження усіх курсів з бази даних із пагінацією в 10 елементів аби не перезавантажувати додаток . Поки йдуть запити на сервер, користувач може побачити прогрес-бар, який сповіщає його про те, що щось вивантажується з бази.

На екрані під переключенням табів знаходиться пошук, який реалізується за допомогою запита на сервер і запит відправляється лише після того, як юзер введе 3 або більше символів у пошукове поле. Після чого, як тільки він перестає писати, має пройти 600 ms для того, щоб відправився запит, щоб не відправляти запити кожен раз після зміни текстового поля.

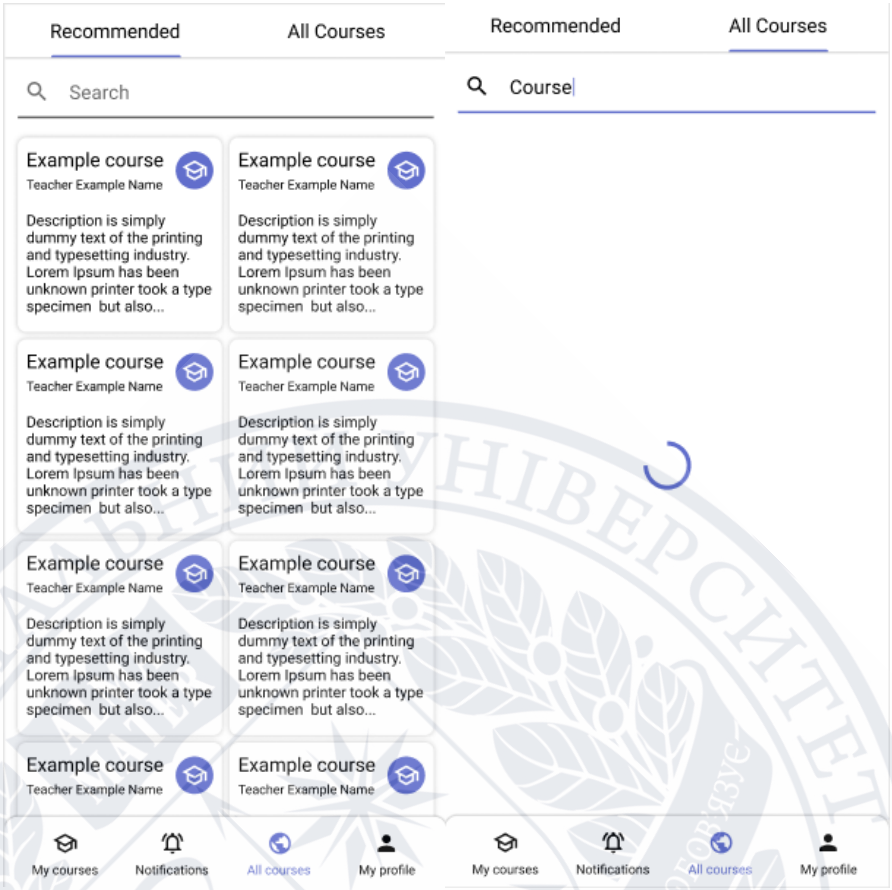

Рисунок 3.12 – Скріншот головного екрану, таба «Всі курси»

Наступна вкладка – аккаунт. Після запуску додатку, якщо юзер вже існує, робиться запит на профіль юзера, де зберігається вся інформація про юзера. Після запиту вся інформація записується у тимчасове сховище Shared Preferences і поки юзер не вийшов з аккаунта ми маємо повний доступ до його інформації і можемо користуватися нею у всіх місцях додатку.

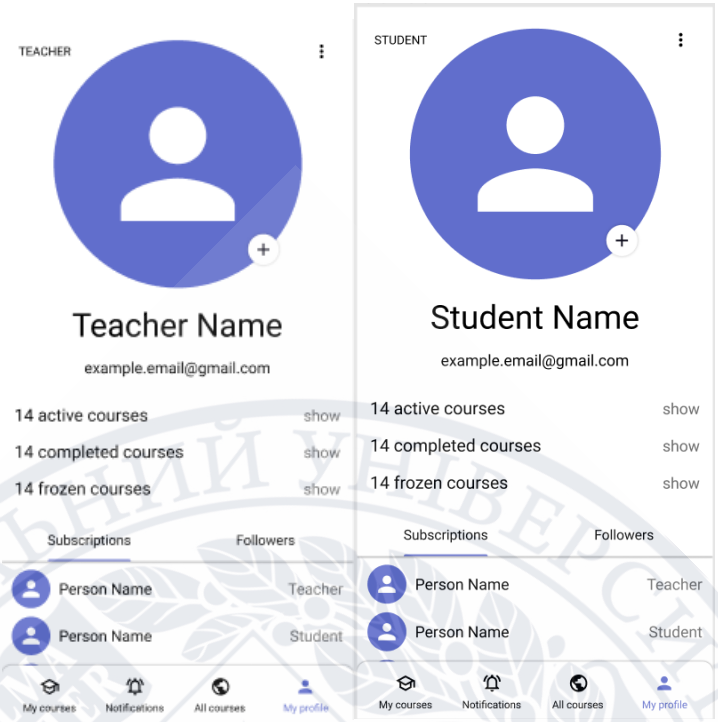

Рисунок 3.13 – Скріншот головного екрану , таба «Мій профіль»

При натисканні на три крапки зверху, юзеру показується контекстне меню, за допомогою якого він може редагувати профіль, відключити ПУШ сповіщення, або ж вийти з додатку.

Натискаючи на редагування, юзер може змінити неключові поля і відправити нові поля запитом до бази даних, якщо поля валідні.

Наступне – сповіщення. По замовченню сповіщення увімкненні. Якщо юзер хоче відключити їх, він натискає на пункт налаштувань «Сповіщення» і відправляє запит на відключення сповіщення до серверу.

І останнє – вихід з свого аккаунту у додатку. Якщо користувач натискає на цей пункт, відправляється запит до сервера на закінчення сесії аутентифікації і юзера перекидає на екран логіна.

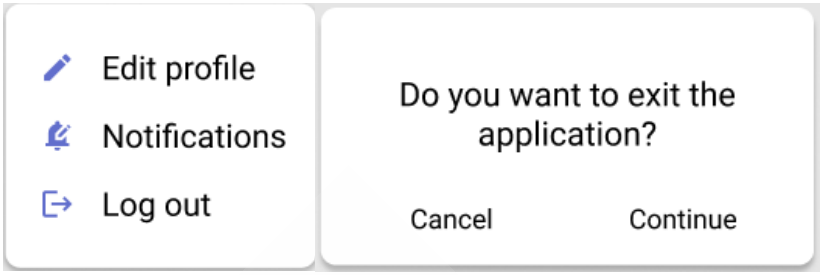

Рисунок 3.14 – Скріншот контекстного меню та алерта про підтвердження операції завершення сесії

Рисунок 3.15 – Скріншот екрану редагування інформації користувача

Коли юзер натискає на курс, що його зацікавить, йому відкривається екран детально про курс, на якому він бачить скорочену інформацію про курс, та має змогу відправити реквест (запит) на учать у вибраному курсі

Рисунок 3.16 – Скріншот екрану курсу детально

Якщо запит прийнятий або відхилений на екрані курсу буде показано статус

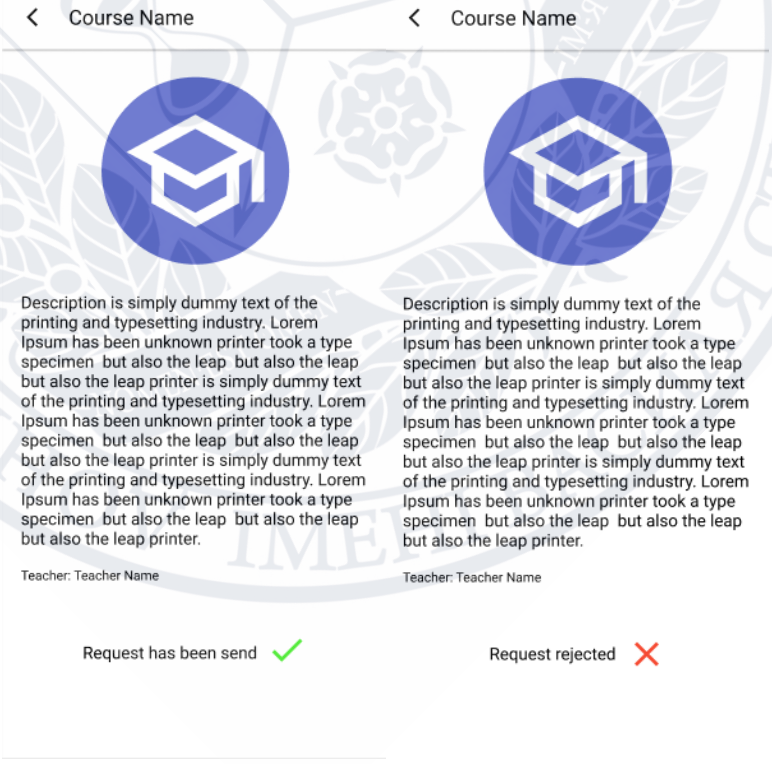

Рисунок 3.17 – Скріншот екрану курсу детально із різними статусами

Після проходження курсу студенту надається тест, який потрібно пройти для того, щоб показати на скільки був засвоєний матеріал. По конкретному курсу йде запит на тест і після проходження відправляється результат.

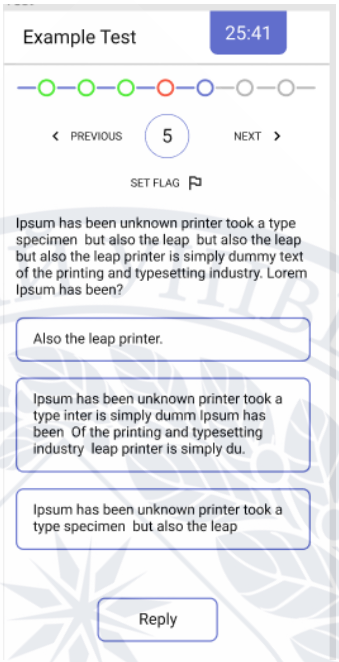

Рисунок 3.18 – Скріншот екрану проходження тестування

Рисунок 3.19 – Скріншот екрану проходження тестування з вибраним варіантом відповіді

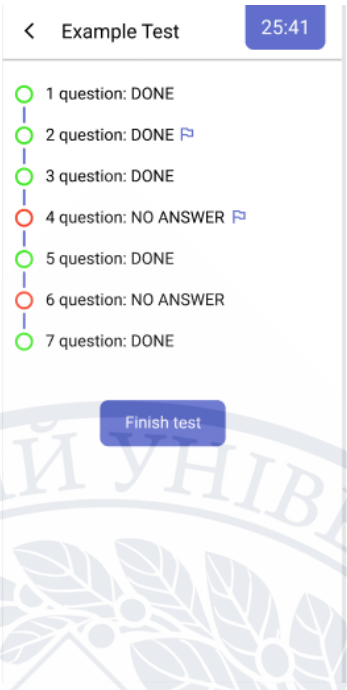

Рисунок 3.20 – Скріншот екрану відправки пройденого тесту

Після відправки запиту про завершення тесту, перевіряється кількість правильних відповідей і робиться запит на отримання результату тесту.

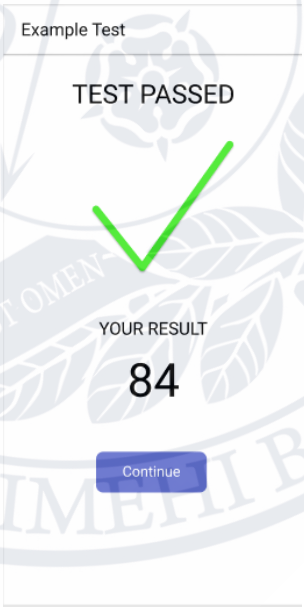

Рисунок 3.21 – Скріншот екрану відправки пройденого тесту

Висновок до розділу 3

Серверна частина додатоку «Get Simple» була розроблена за допомогою Flutter SDK, та найзручніших і найтехнологічніших інструментів, які доступні на

сьогодні в pub.dev та Firebase. Завдяки технологіям Firebase додаток включає в себе весь необхідний функціонал із потужним рівнем захисту даних від розробників платформи Google. Ця платформа надала змогу швидко інтегрувати функціонал аутентифікації, хмарової бази даних, і відкинула необхідність розкривати свій власний сервер для поширення додатку у мережі інтернет.

ВИСНОВКИ

З кожним роком зростає кількість мобільних пристроїв та їх популярність. Вони стають засобами першої необхідності. Відповідно, зростає кількість необхідних програм, які встановлюються на ці пристрої. Також збільшується кількість програмістів, які займаються розробкою додатків під мобільні пристрої.

Метою бакалаврської роботи було розробити мобільний додаток для онлайн-навчання. Дана робота була присвячена розробці серверної частини мобільного додатку.

Для того, щоб розпочати роботу над додатком «Get Simpe», було проаналізовано актуальність даного питання. Платформа для навчання в період пандемії буде користуватися великим попитом для можливості працювати, навчатися не виходячи з дому. Взагалі, онлайн навчання дає людині можливість отримувати різноманітні знання легко, доступно, мобільно та якісно.

Для дотатку «Get Simple» була обрана архітектура GetX. Щоб реалізувати цю архітектуру, у Flutter-додатку застосовується бібліотека GetX. Для реалізації серверної частини було обрано Firebase, сервіс від компанії Google, який надає змогу користуватися хмарним сервером, базою даних у хмарині та розробити повноцінний back-end із авторизацією та аналітикою.

Було досліджено, вивчено і обрано багато інструментів для розробки сучасного мобільного додатку, що надали змогу реалізувати всі необхідні функції. Було вирішено розробити мобільний додаток для ОС Android версії 7.0, застосовуючи SDK – Flutter, як один з найперспективніших інструментів. Мова програмування Dart - це мова, створена Google, яка позиціонується, як альтернатива JavaScript.

Для створення макета і прототипу майбутнього мобільного додатку була обрана програма Figma. Розробляючи UX/UI дизайн мобільного додатку враховано ефект «естетика-юзабіліті» та підібрано кольори і графічні

компоненти таким чином, щоб вони привертали увагу, а дизайн у цілому був інтуїтивно зрозумілий користувачу.

СПИСОК ЛІТЕРАТУРИ

1. Mobile App Usage Surged 40% During COVID-19 Pandemic. URL: https://www.appannie.com/en/insights/market-data/mobile-app-usage-surged-40-during-covid-19-pandemic/
2. 8 причин, щоб полюбити Figma. URL: https://artjoker.ua/ru/blog/8-prichin-chtoby-polyubit-figma-tak-zhe-silno-kak-my/
3. Введення в мову програмування Dart. URL: https://metanit.com/dart/tutorial/1.1.php
4. Dart documentation. URL: https://dart.dev/guides
5. Що таке Flutter. URL: https://habr.com/ru/post/481326/
6. Flutter - Beautiful native apps in record time. URL: https://flutter.dev/
7. GetX. URL: https://github.com/jonataslaw/getx/blob/master/README.ru.md
8. Dependency Injection. URL: https://habr.com/ru/post/350068/
9. Dependency Injection In Flutter. URL: https://medium.com/flutter-community/dependency-injection-in-flutter-f19fb66a0740
10. Dio. URL: https://pub.dev/packages/dio
11. Що таке Clean Architecture. URL: https://proglib.io/p/clean-architecture-android-apps/
12. Flutter Clean Architecture. URL: https://pub.dev/packages/flutter_clean_architecture
13. Stacked Themes. URL: https://pub.dev/packages/stacked_themes
14. Що таке Firebase. URL: https://avada-media.ua/services/firebase/
15. Firebase helps you build and run successful apps. URL: https://firebase.google.com/
16. Android Studio. URL: https://developer.android.com/studio
17. Кращі практики Figma , URL: https://ux.pub/luchshie-praktiki-figma-komponenty-stili-i-obschie-biblioteki/
18. Різниця між Cloud Firestore та Realtime Database ,URL: https://coderoad.ru/46549766/%D0%92-%D1%87%D0%B5%D0%BC-

%D1%80%D0%B0%D0%B7%D0%BD%D0%B8%D1%86%D0%B0-%D0%BC%D0%B5%D0%B6%D0%B4%D1%83-Cloud-Firestore-%D0%B8-Firebase-Realtime-Database

19. Початок роботи з Firebase Authenticate URL : https://code.tutsplus.com/ru/tutorials/get-started-with-firebase-authentication-for-ios--cms-29227

Декларація щодо унікальності текстів роботи
та невикористання матеріалів інших авторів без посилань

Присіч Артем Володимирович
Прізвище, ім'я, по батькові

Інформаційних і прикладних технологій
Факультет

122 «Комп'ютерні науки»
Шифр і назва спеціальності

«Сучасні інформаційні технології та програмування»
Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Розробка серверної частини для навчальної системи» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

дата підпис