МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему Кросплатформний застосунок для організації роботи репетиторів
Tutor's organizing work cross-platform application

Виконав: студент денної форми навчання
спеціальності 126 – Інформаційні системи та технології

(шифр і назва напряму підготовки, спеціальності)

Кравченко Кирило Дмитрович

(прізвище, ім'я, по-батькові)

Керівник Розновець О.І.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент к.ф.-м.н., доцент Крапівний Ю.М.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№ 11 від «10» червня 2022 р.

Завідувач кафедри

_____ _Євгеній МАЛАХОВ_

(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___

протокол № __ від «_ _» ____ 2022 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Володимир ВИЧУЖАНІН_

(підпис) (ім'я, прізвище)

Одеса – 2022

АНОТАЦІЯ

Дана дипломна робота присвячена темі «Кросплатформний застосунок для організації роботи репетиторів».

Метою дипломної роботи є створення кросплатформного застосунку, що працює на платформах iOS та Android і призначеного для приватних підприємців, які займаються репетиторською діяльністю. Цей продукт надає репетитору ряд інструментів для більш ефективного планування своєї професійної діяльності та фінансового обліку, що, у свою чергу, сприятиме підвищенню продуктивності роботи підприємця, дозволяючи економити його час та енергію.

Основні переваги створеного мобільного застосунку:

1) централізація інформації про учнів;

2) надання інтерфейсу для складання розкладу занять;

3) формування наочної статистики занять та доходів.

Застосунок спроектований з використанням архітектурного підходу REST, комплекту засобів розробки кросплатформних застосунків Flutter, платформи розробки веб-застосунків Node.js та бібліотеки Express.

ABSTRACT

This graduate work is dedicated to the topic «Tutor's organizing work cross-platform application».

The purpose of this graduate work is to create a cross-platform application that runs on iOS and Android platforms and is designed for private entrepreneurs engaged in tutoring. This product provides a tutor with a number of tools for more effective planning of his professional activities and financial accounting, which in turn will increase the productivity of the entrepreneur, allowing him to save time and energy.

The main advantages of the created mobile application are:

1) centralizing student information;

2) providing an interface for scheduling classes;

3) generating visual statistics of classes and income.

The application is designed using REST architectural approach, a set of cross-platform application development tools Flutter, web application development platform Node.js and Express library.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ФОП – фізична особа-підприємець

CPU – Central processing unit

GPU – Graphics processing unit

JOSE – JSON Object Signing and Encryption

JSON – JavaScript Object Notation

JWT – JSON Web Token

MVC – шаблон проектування Model-View-Controller

REST – Representational State Transfer (передача репрезентативного стану)

UI – User Interface (інтерфейс користувача)

ВСТУП

Ще порівняно недавно індивідуальні заняття розглядалися лише як невеликий підробіток для викладачів і можливість наздогнати шкільну програму для учнів. Але з кожним роком репетиторство продовжує набирати обертів. Збільшується попит на кваліфікованих педагогів з різних дисциплін. За даними найбільшої платформи для пошуку репетиторів, з 2017 по 2021 роки кількість викладачів, які пропонують свої послуги на сайті, виросла більш ніж у 10 разів [1].

З репетиторами займаються дошкільнята (ранній розвиток, логопед, підготовка до школи), у початковій та середній школі батьки прагнуть дати своїм дітям «міцну базу», а у одинадцятому класі гостро постає питання про успішну здачу іспитів у формі ЗНО та вступ до університету. Особлива необхідність в індивідуальному підході до освіти виникла і в умовах пандемії, коли дистанційне навчання через свої особливості не сприяло засвоєнню матеріалу учнями.

Велика кількість клієнтів різних вікових груп та завантажений робочий графік перешкоджають ефективному веденню професійної діяльності репетитора. Часто трапляється таке, що заплановані уроки переносяться або скасовуються зовсім, а якщо заняття все-таки відбудеться, репетитор має тримати в голові безліч додаткової інформації: яку тему зараз проходити з учнем, чи треба влаштовувати опитування, чи перевірено домашнє завдання.

Слід також взяти до уваги і той факт, що репетитор є фізичною особою-підприємцем та, не дивлячись на те, що до ведення бухгалтерського обліку у ФОПів не висуваються серйозні законодавчі вимоги, має потребу вести історію своїх доходів, для того щоб аналізувати та корегувати свою роботу. На усе це ще наклається й людьский фактор – інколи учні оплачують заняття наперед, інколи забувають оплатити заняття. Для вирішення цих проблем розробляється інформаційна система, що виконує функції

віртуального секретаря та надає інструменти для фіксування прибутків людям, які займаються репетиторською діяльністю.

Отже, метою метою дипломної роботи є створення кросплатформного застосунку, що працює на платформах iOS та Android і призначеного для приватних підприємців, які займаються репетиторською діяльністю. Для реалізації системи доцільно застосувати платформно незалежні технології, що дозволить використовувати створений застосунок на різних програмно-апаратних платформах, зокрема за допомогою мобільних пристроїв. Застосування кросплатформного підходу дозволяє створювати застосунки для різних платформ з єдиною кодовою базою, що заощаджує час на розробку та позбавляє зайвих зусиль.

Для досягнення поставленої мети необхідно виконати наступні задачі:

1) проаналізувати особливості роботи репетитора з урахуванням основних потреб ФОП;

2) сформулювати задачі репетитора, які потребують реалізації у застосунку;

3) обгрунтувати вибір архітектури, технологій та засобів створення кросплатформного застосунку;

4) спроектувати базу даних для зберігання інформації;

5) розробити застосунок з застосуванням обраних засобів і протестувати його роботу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення користувачів програмного продукту та їх задач

Згідно зі Статтею 50 Цивільного кодексу України «Право фізичної особи на здійснення підприємницької діяльності» фізична особа здійснює своє право на підприємницьку діяльність за умови її державної реєстрації в порядку, встановленому законом [2]. З цього випливає той факт, що потенційними користувачами створюваної системи є люди, які займаються репетиторською діяльністю та офіційно зареєстровані як ФОП, або працевлаштовані на контрактній основі у приватних навчальних центрах.

У результаті аналізу особливостей роботи репетитора виділений основний бізнес процес проведення заняття, який проілюстровано за допомогою методології BPMN – мови візуального моделювання для додатків бізнес-аналізу та визначення робочих процесів підприємства [3]. Діаграму створено за допомогою графічного редактора draw.io.

Репетитор зв'язується з учнем щодо проведення заняття (рис. 1.1). Якщо це новий учень, викладач створює профіль учня у застосунку, вносячи всю необхідну контактну інформацію (ім’я, клас, номер телефону, домашню адресу, базову вартість заняття). Після цього відбувається узгодження типу заняття (очно або онлайн), дати та часу його проведення. Коли всі деталі заняття погоджено, репетитор створює нову подію у застосунку, маючи при цьому можливість додатково вказати формат проведення заняття (очно або онлайн) та статус виконання домашнього завдання (не задано або виконано або не виконано). Після того, як заняття буде проведено, репетитор помічає його як «проведене» (або «скасоване», якщо заняття не відбулось), отримує гроші та змінює статус оплати заняття (за замовчуванням заняття має статус «не оплачено»).

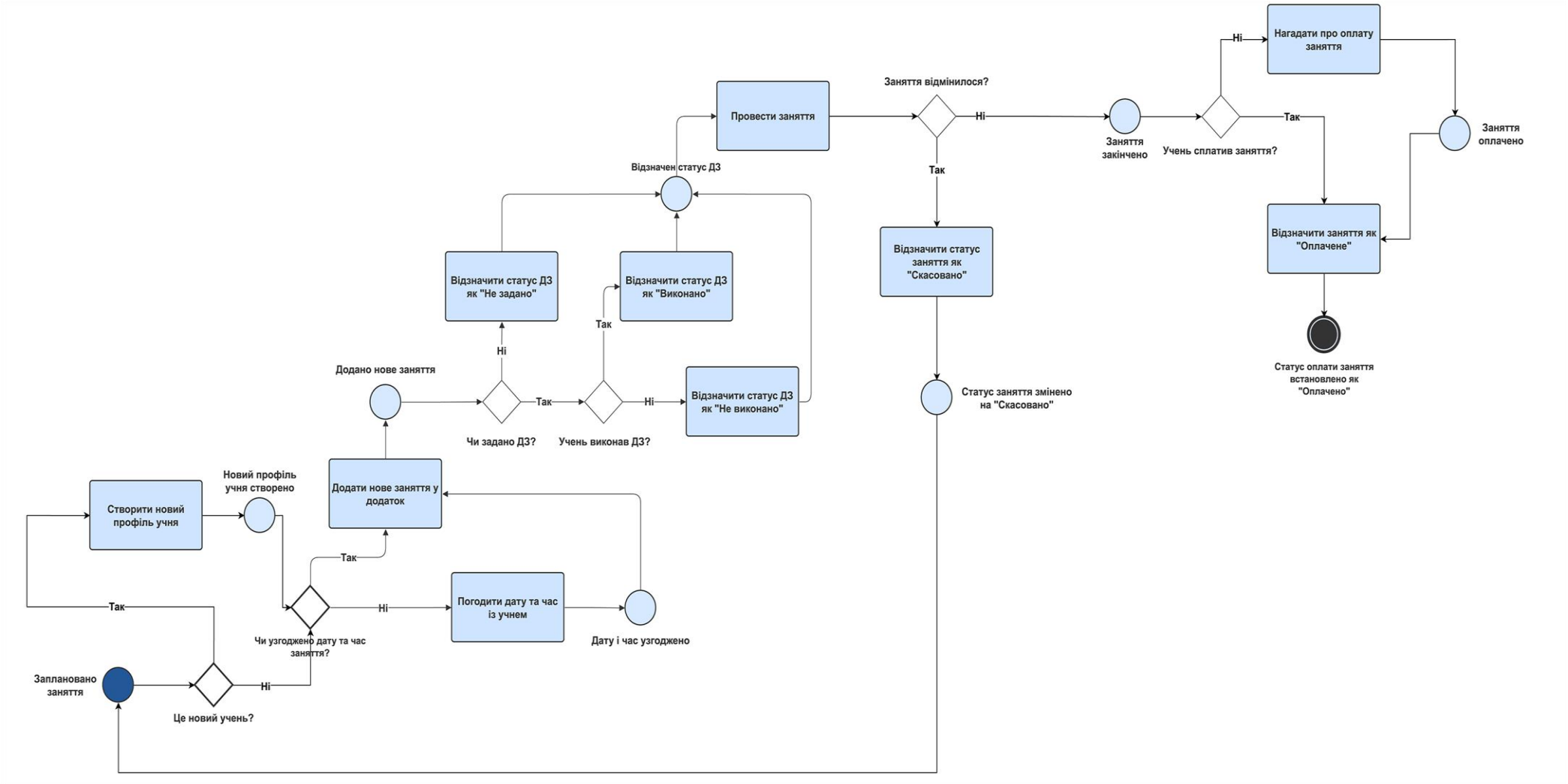


Рисунок 1.1 – Діаграма бізнес-процесу проведення заняття

На підставі розгляду даного бізнес-процесу можна виділити список задач та підзадач користувача, які перераховані в табл. 1.1, з урахуванням вхідних та вихідних даних для кожної задачі.

Таблиця 1.1 – Задачі користувачів кросплатформного застосунку для організації роботи репетиторів

Номер	Формулювання задачі	Формулювання підзадачі	Вхідні дані	Вихідні дані
1	Вести облік учнів	Додати нового учня	Ім'я, клас, телефон, адреса, базова вартість заняття, примітка, статус учня «Активний»	Новий профіль учня
		Додати учня до архіву	Учень, Статус учня «В архіві»	Неактивний учень
		Редагувати інформацію про учня	Учень, ім'я, клас, телефон, адреса, базова вартість заняття, примітка	Оновлений профіль учня
		Переглянути профіль учня	Учень	Учень, ім'я, клас, телефон, адреса, базова вартість заняття, примітка
2	Складати розклад занять	Додати нове заняття	Дата, час початку, тривалість, тип (онлайн/офлайн), актуальна вартість, статус ДЗ (не задано, не виконано, виконано), статус заняття (заплановано/не відбулося)	Нове заняття

Продовження таблиці 1.1

Номер	Формулювання задачі	Формулювання підзадачі	Вхідні дані	Вихідні дані
		Відзначити заняття проведеним	Заняття, Статус заняття «проведено»	Проведене заняття
		Редагувати інформацію про заняття	Заняття, Дата, час початку, тривалість, тип (онлайн/офлайн), актуальна вартість, статус ДЗ (не задано, не виконано, виконано)	Оновлений запис про заняття
		Переглянути інформацію про заняття	Заняття	Дата, час початку, тривалість, тип (онлайн/офлайн), актуальна вартість, статус ДЗ (не задано, не виконано, виконано)
3	Вести аналітику	Відзначити заняття оплаченим	Заняття	Статус оплати заняття «оплачено»
		Переглянути дохід за день/тиждень/ місяць/рік	Проміжок часу, Учень	Сума доходу
		Переглянути статистику учня	Проміжок часу, Учень	Кількість занять, заборгованість учня

Продовження таблиці 1.1

Номер	Формулювання задачі	Формулювання підзадачі	Вхідні дані	Вихідні дані
4	Керувати нагадуваннями про майбутні заняття	Створити нагадування	Заняття, час нагадування	Нове нагадування
		Редагувати нагадування	Нагадування, час нагадування	Оновлене нагадування
		Видалити нагадування	Нагадування	Відсутні

1.2 Порівняння конкурентних програмних продуктів

Прямими конкурентами розроблюваного застосунка є програмні продукти, що позиціонуються як застосунки для репетиторів – Light [4], TutorTask [5] і ЯРепетитор [6].

Застосунок Light, який працює на платформах iOS та Android, надає лише мінімально необхідний набір інструментів, якого, на жаль, недостатньо для задоволення потреб користувача-репетитора. Екран розкладу не наочний, немає можливості вказувати додаткову інформацію про заняття. Профілі учнів не дозволяють зберігати необхідну контактну інформацію, а відстеження фінансів зводиться до простого відображення балансу учня.

Застосунок TutorTask частково усуває ці недоліки, проте сильно програє у візуальній складовій та стабільності роботи. При цьому цей застосунок доступний лише для пристроїв, що працюють на платформі Android.

Застосунок ЯРепетитор, який працює на платформах iOS та Android, найбільш близький до того, щоб задовольнити основні потреби користувача-репетитора, однак також, як і TutorTask, не може похвалитися стабільністю роботи програми і не надає користувачеві даних про проведені заняття, що істотно зменшує можливості аналітики та коригування професійної діяльності.

Порівняння мобільних застосунків, що надають інструменти для організації роботи репетиторів, представлено у табл. 1.2.

Таблиця 1.2 – Порівняння мобільних застосунків, що надають інструменти для організації роботи репетиторів

Параметр оцінки	Застосунки			
	Light	TutorTask	Ярепетитор	Розроблюваний застосунок
Складання розкладу	+	+	+	+
Нагадування про заняття	-	-	-	+
Зберігання контактної інформації про учнів	-	+	+	+
Детальна статистика	-	+	-	+
Фінансовий облік	+	+	+	+
Стабільність роботи	+	-	-	+
Кросплатформність	+	-	+	+
Сучасний UI	+	-	+	+

1.3 Висновок до розділу 1

Аналіз існуючих застосунків для організації роботи репетиторів виявив ряд проблем, пов’язаних з використанням цих програмних продуктів. До найбільш поширених проблем відносяться відсутність таких важливих та необхідних функцій як зберігання контактної інформації про учнів, отримання статистичних даних щодо проведених занять та фінансів, відсутність системи сповіщень (нагадувань про зантяття). Також не всі застосунки здатні працювати на різних програмно-апаратних платформах.

Доцільність створення розроблюваного застосунку для організації роботи репетиторів ґрунтується на можливості вирішення вищезазначених проблем завдяки реалізації функцій, відсутніх у застосунках-конкурентах, а також завдяки застосуванню платформно незалежних технологій для створення програмного продкуту.

2 ВИБІР АРХІТЕКТУРИ ТА ШАБЛОНУ ПРОЕКТУВАННЯ КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНИЗАЦІЇ РОБОТИ РЕПЕТИТОРІВ

Ідея створення кросплатформного застосунку для організації роботи репетиторів та його проект представлені на дев'ятнадцятій всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології», тези доповіді опубліковані [7].

2.1 Вибір архітектури

Для реалізації застосунку для організації роботи репетиторів обрана трирівнева архітектура клієнт-сервер – обчислювальна модель, в якій сервер розміщує, надає та керує більшістю ресурсів та послуг, що запитуються клієнтом [8].

Трирівнева архітектура клієнт-сервер складається з клієнтського застосунку, серверу застосунків та серверу баз даних. На рис. 2.1 показано, як для трирівневої архітектури клієнт-сервер відбувається розмежування рівнів програмного забезпечення: рівень інтерфейсу користувача, рівень реалізації прикладної логіки і засобів обробки даних, рівень баз даних.

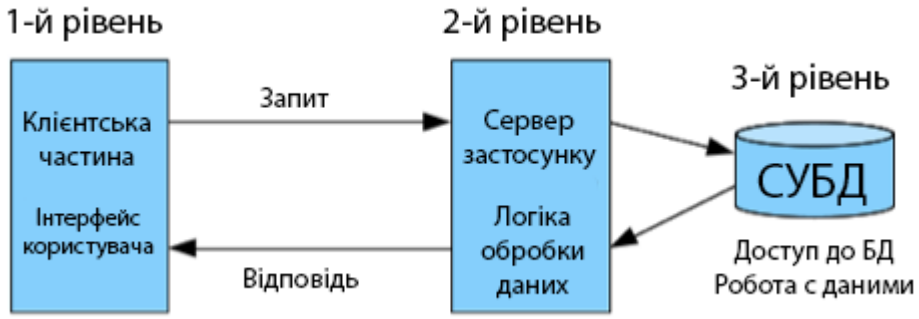


Рисунок 2.1 – Трирівнева архітектура клієнт-сервер

2.2 Вибір шаблону проектування

Для реалізації інформаційної системи обраний шаблон проектування MVC (модель – представлення – контролер), який розділяє програму на три окремі компоненти: модель даних програми, інтерфейс користувача і логіку взаємодії користувача з системою, завдяки чому модифікація одного з цих компонентів надає мінімальний вплив на інші або не робить його зовсім.

Модель містить лише чисті дані програми, у ній немає логіки, що описує, як представити ці дані користувачеві. Представлення відображає дані моделі користувачу. Представлення знає, як отримати доступ до даних моделі, але воно не знає, що ці дані означають, і що користувач може зробити для маніпулювання ними. Контролер існує між представленням та моделлю. Він слухає події, викликані представленнями (або іншим зовнішнім джерелом), і виконує відповідну реакцію на ці події. Найчастіше реакція полягає у виклику методу моделі. Оскільки представлення та модель пов'язані через механізм повідомлень, результат цієї дії автоматично відображається у представленні [9].

Використання шаблону MVC дозволяє легко модифікувати програму. Додавання та оновлення нових представлень спрощується, оскільки одна частина коду не залежить від інших частин. Таким чином, будь-які зміни в певній частині програми не впливають на інші частини. Це допомагає підвищити гнучкість та масштабованість програми.

Структура веб-застосунку, побудованого згідно концепції MVC, показана на рис. 2.2.

Запити надсилаються одним суб'єктом – агентом користувача (або проксі від його імені). Найчастіше агентом користувача є веб-браузер, але це може бути будь-що, наприклад, робот, який переглядає веб-сторінки для поповнення та підтримки індексу пошукової системи.

Клієнт формує запит на отримання інформації (або ресурсів), який має бути згенерований сервером. Коли сервер отримує цей запит, він використовує інформацію, включену в запит, для створення відповіді, що містить

інформацію, що запитується. Після створення відповідь відправляється назад клієнту у запитаному форматі для відображення користувачеві.

Шлях – це ділянка коду, яка пов'язує HTTP-метод (GET, POST, PUT, DELETE і т.д.), шаблон URL та контроллер, яки викликається для обробки цього шаблону.

Контроллер отримує дані з моделей, обробляє їх та повертає користувачеві для перегляду.

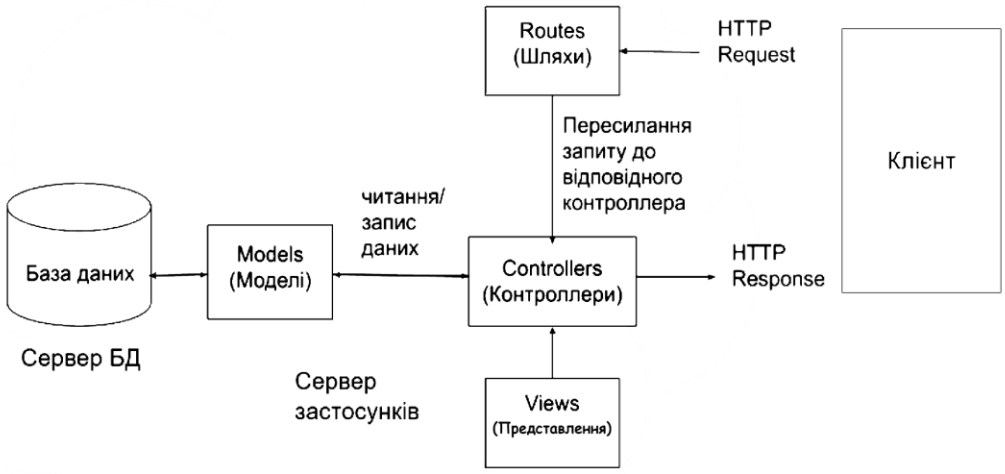


Рисунок 2.2 – Структура веб-застосунку, побудованого згідно концепції MVC

2.3 Вибір стилю взаємодії компонентів програмного забезпечення

Для взаємодії компонентів програмного забезпечення проектованої системи обраний архітектурний стиль REST [10], який забезпечує стандарти між комп'ютерними системами в мережі інтернет, що полегшує взаємодію систем одна з одною. У архітектурному стилі REST реалізація клієнта і реалізація сервера можуть здійснюватися незалежно, при цьому кожний з них не знає про іншого.

Використовуючи інтерфейс REST, різні клієнти звертаються до одних і тих же кінцевих точок REST, виконують одні й самі дії і отримують одні й самі відповіді. Запит REST зазвичай складається з [9]:

1) HTTP-метода, який визначає, яку операцію необхідно виконати;
2) заголовка, який дозволяє клієнту передати інформацію про запит;
3) шляху до ресурсу;
4) необов'язковового тіла повідомлення, що містить додаткові дані;

Однією з головних переваг використання під архітектурного стилю REST є те, що цей стиль спирається на стандарт HTTP, а це означає, що він не залежить від формату зберігання даних, що забезпечує його швидкість і легкість.

Протокол REST відокремлює зберігання даних і інтерфейс користувача від сервера. Це означає, що розробники можуть працювати над різними частинами проекту незалежно один від одного і при необхідності застосовувати різні середовища розробки.

REST можна швидко масштабувати насамперед завдяки поділу між клієнтом та сервером. Крім того, розробники можуть легко інтегрувати API REST без особливих додаткових зусиль.

3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНИЗАЦІЇ РОБОТИ РЕПЕТИТОРІВ

3.1 Вибір моделі представлення даних

База даних спроектована на основі функціональних потреб користувачів, перерахованих у табл. 1.1, незалежно від особливостей реалізації застосунку. Як модель представлення даних обрано реляційну модель. Далі описані причини, через які саме реляційна модель представлення даних заслуговує на перевагу перед іншими.

Реляційні бази даних працюють із структурованими даними. Вони підтримують транзакційну узгодженість та забезпечують різноманітні можливості для підтримки їх цілісності та несуперечності. Ключовою особливістю реляційних баз даних є те, що вони суворо визначені і добре структуровані, тому дані не дублюються і в них немає необхідності в здатності розширюватись.

Всі перераховані вище характеристики реляційних баз даних мають пряме відношення до вимог до бази даних створюваного програмного продукту. В розроблюваному застосунку необхідно оперувати сутностями, що відображають реальний світ: учні, заняття, домашні завдання, оплата тощо. Системі не потрібно розширюватись, тому що створено вичерпний список задач користувачів інформаційної системи (див. табл. 1.1).

Цілісність даних потрібно забезпечувати через те, що діяльність репетитора має елементи системи планування та журналювання, які потребують гарантованої достовірності. Крім того, оскільки операції з даними користувача застосунку не передбачають складних обчислень, для моделі предметної області буде достатньо реляційної моделі даних. Цілісність даних у даному випадку більш пріоритетна, ніж швидкість роботи.

3.2 Проектування бази даних

Основні сутності, що будуть присутні в проектованій базі даних: Користувач, Учень, Заняття, Предмет.

У табл. 3.1 описані типи зв’язків між перерахованими сутностями.

Таблиця 3.1 – Зв’язки між сутностями

Назва сутності	Тип зв’язку	Назва сутності
Користувач	Один-до-багатьох	Учень
Учень	Один-до-багатьох	Заняття
Предмет	Один-до-багатьох	Заняття

Між сутностями Користувач та Учень існує зв'язок один-до багатьох: у одного користувача може бути багато учнів; з іншого боку, один учень прив’язується до одного викладача (якщо цей же учень навчається ще у одного викладача, то він з точки зору застосунку вважається іншою персоною). Зв'язок формалізований шляхом додавання ідентифікатору користувача у сутність Учень у вигляді зовнішнього ключа.

Між сутностями Учень та Заняття існує зв’язок один-до багатьох: один учень може відвідувати багато занять; з іншого боку, на одному занятті може бути присутній лише один конкретний учень (навіть якщо репетитор проводить заняття для декількох учнів, з точки зору застосунку заняття для кожного учня вважається окремим). Зв'язок формалізований шляхом додавання ідентифікатору учня у сутніть Заняття у вигляді зовнішнього ключа.

Між сутностями Заняття та Предмет існує зв'язок один-до багатьох: один предмет може повторюватись на багатьох заняттях; з іншого боку, на одному конкретному занятті може вивчатися лише один предмет. Зв’язок формалізований шляхом додавання ідентифікатору учня у сутність Заняття у вигляді зовнішнього ключа.

Атрибути сутностей з зазначенням обмежень цілісності описані у таблицях 3.2 – 3.5.

Таблиця 3.2 – Опис сутності Користувач (User)

Назва стовпця	Призначення стовпця	Порожні значення	Обмеження атрибуту
user_id	Ідентифікатор	Ні	Первинний ключ
username	Ім’я користувача	Ні	Унікальний текст
email	Електронна адреса	Ні	Унікальний текст, повинен містити символ «@»
password	Пароль у захешованому вигляді	Ні	Текст

Таблиця 3.3 – Опис сутності Учень (Student)

Назва стовпця	Призначення стовпця	Порожні значення	Обмеження атрибуту
student_id	Ідентифікатор	Ні	Первинний ключ
student_status	Статус учня (активний / в архіві)	Ні	Текст, значення за замовченням «ACTIVE»
student_name	Ім’я учня	Ні	Текст
student_class	Клас учня	Так	Текст
phone_number	Номер телефону	Так	Текст, містить лише символ «+» на початку, та цифри від «0» до «9»
address	Адреса	Так	Текст
base_price	Базова вартість заняття за годину	Ні	Позитивне дійсне число
notes	Нотатки	Так	Текст
user_id	Ідентифікатор користувача, який займається з учнем	Ні	Зовнішній ключ для зв’язку з таблицею Користувач

Таблиця 3.4 – Опис сутності Заняття (Lesson)

Назва стовпця	Призначення стовпця	Порожні значення	Обмеження атрибуту
lesson_id	Ідентифікатор	Ні	Первинний ключ
lesson_status	Статус заняття (заплановане / проведене / скасоване)	Ні	Текст, значення за замовченням «PLANNED»

Продовження таблиці 3.4

Назва стовпця	Призначення стовпця	Порожні значення	Обмеження атрибуту
lesson_start_time	Час початку заняття	Ні	Час
lesson_duration	Тривалість заняття	Ні	Позитивне дійсне число
lesson_type	Формат проведення заняття (онлайн / офлайн)	Ні	Текст, значення за замовченням «OFFLINE»
actual_price	Модифікована вартість заняття за годину	Так	Позитивне дійсне число
homework_status	Стан домашнього завдання	Ні	Текст, значення за замовченням «NOT SET»
payment_status	Стан оплати	Ні	Текст, значення за замовченням «NOT PAID»
student_id	Ідентифікатор, що вказує на учня, з яким проводиться заняття	Ні	Зовнішній ключ для зв’язку з таблицею Учень
subject_id	Ідентифікатор, що вказує на предмет, по якому проводиться заняття	Ні	Зовнішній ключ для зв’язку з таблицею Предмет

Таблиця 3.5 – Опис сутності Предмет (Subject)

Назва стовпця	Призначення стовпця	Порожні значення	Обмеження атрибуту
user_id	Ідентифікатор	Ні	Первинний ключ
subject_id	Назва предмету	Ні	Унікальний текст

Тепер потрібно перевірити спроектовану БД на НФБК (нормальну форму Бойса-Кодда) яку ще називають «декомпозицією без затрат», згідно з якою атрибут повинен залежати від потенційного ключа. Розглянемо перевірку НФБК на прикладі найбільшої сутності Заняття (Lesson). Покажемо послідовний перехід від однієї нормальної форми до іншої.

Це відношення находиться в першій нормальній формі, бо значення кожного атрибуту не дробляться на декілька значень.

Це відношення находиться в другій нормальній формі, бо кожен неключовий атрибут функціонально повно залежить від первинного ключа – lesson_id (ідентифікатору заняття).

Це відношення находиться в третій нормальній формі, бо кожен неключовій атрибут залежить тільки від первинного ключа.

ER-діаграма бази даних кросплатформного застосунку для організації роботи репетиторів, що створена за допомогою ресурсу DBdiagram [11], зображена на рис. 3.1.

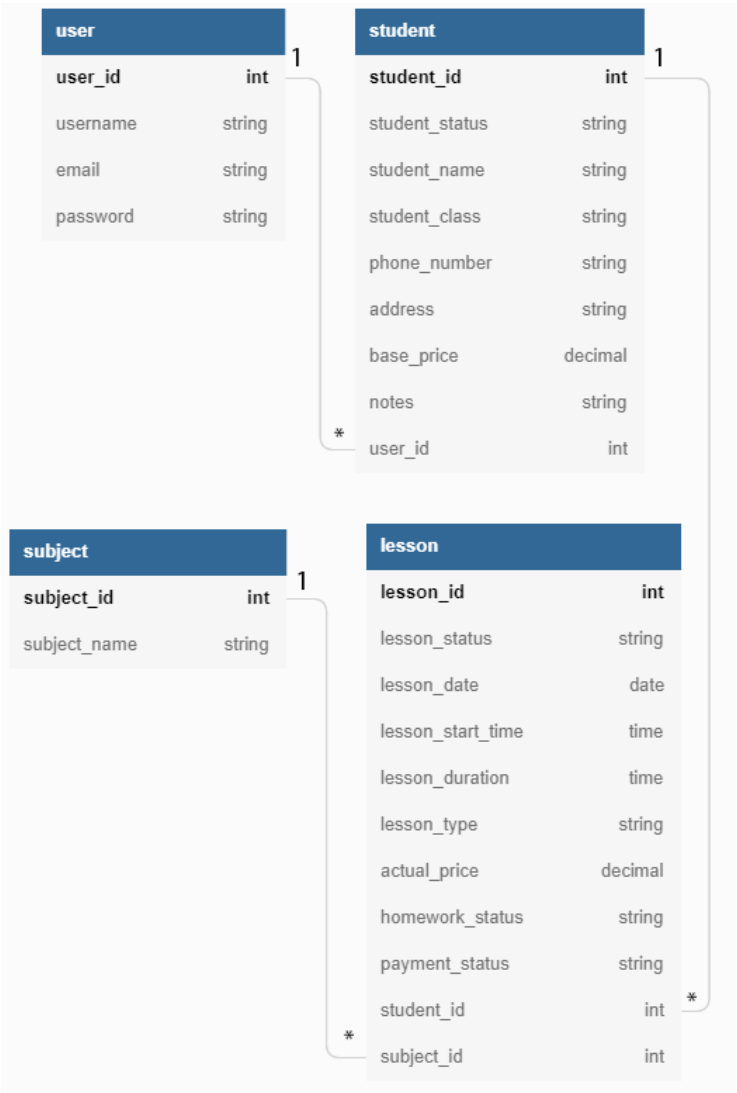


Рисунок 3.1 – ER-діаграма бази даних кросплатформного застосунку для організації роботи репетиторів

4 ЗАСОБИ РЕАЛІЗАЦІЇ КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ РЕПЕТИТОРІВ

Для написання серверної частини кросплатформного застосунку для організації роботи репетиторів обрана платформа Node.js – асинхронне орієнтоване на події середовище виконання мови JavaScript, призначене для створення масштабованих мережевих застосунків.

Node.js використовує архітектуру однопотокового циклу подій для одночасної роботи з кількома клієнтами. У моделі багатопотокового запиту-відповіді кілька клієнтів надсилають запит, а сервер обробляє кожен із них, перш ніж надіслати відповідь. Однак, для обробки одночасних запитів використовується кілька потоків. Ці потоки визначаються в пулі потоків, і кожен раз, коли надходить запит, окремий потік призначається для обробки [12].

Для взаємодії з Node.js використовується Express – допоміжний серверний фреймворк для створення веб-застосунків на основі JavaScript.

В умовах економічного розвитку та в сучасних технологічних тенденціях на сьогоднішній день мобільні пристрої є одними з основних використовуваних платформ, оскільки мобільні застосунки приносять користувачам величезну користь та переваги, полегшуючи доступ до інформації, її отримання та роботу у будь-який час та в будь-якому місці.

Тому, для того щоб охопити максимальну кількість користувачів мобільних пристроїв, в якості інструмента для реалізації інтерфейсу користувача обраний комплект засобів розробки Flutter, що дозволяє створювати нативно скомпільовані програми для мобільних пристроїв та настільних комп'ютерів, а також веб-застосунки. При цьому, незалежно від вибраного пристрою, кодова база розроблюваного продукту залишається незмінною, що значно зменшує часові та грошові витрати на розробку [13].

Однією з особливостей Flutter як технології кросплатформної розробки є функція гарячого перезавантаження, яка дозволяє швидко і легко

експериментувати, змінювати інтерфейси користувача, додавати функції і виправляти помилки. Гаряче перезавантаження працює шляхом ін'єкції оновлених файлів вихідного коду в працюючу віртуальну машину Dart (VM). Після того як віртуальна машина оновлює класи новими версіями полів та функцій, фреймворк Flutter автоматично перебудовує дерево віджетів, дозволяючи швидко побачити ефект від змін. Застосунок оновлюється, відображаючи зміни, а поточний стан програми зберігається. Програма продовжує виконуватись з того місця, де вона знаходилася до виконання команди гарячого перезавантаження. Код оновлюється, і виконання продовжується.

До всього іншого, Flutter дозволяє викликати специфічні для платформи API, доступні в коді Java або Kotlin на Android і коді Objective C або Swift на iOS. Платформоспецифічний API Flutter працює з передачею повідомлень. З Flutter надсилається повідомлення хосту на iOS або Android по каналу платформи. Хост прослуховує канал платформи та отримує повідомлення. Потім він використовує будь-які специфічні для платформи API, використовуючи нативну мову програмування, і відправляє відповідь у частину програми Flutter (рис. 4.1).

Повідомлення передаються між Flutter Code (UI) і хостом (платформою) за допомогою каналів платформи. Повідомлення та відповіді передаються асинхронно, а користувальницький інтерфейс залишається відгукчивим.

На боці Dart за допомогою MethodChannel (API) відправляється повідомлення, що відповідає виклику методу. На стороні Android MethodChannel Android (API) і на стороні iOS FlutterMessageChannel (API) використовуються для отримання викликів методів і відправки результатів. Ці класи дозволяють розробляти плагін для платформ з дуже простим кодом.

При необхідності виклики методів можна скерувати і в зворотному напрямку: платформа Android/IOS виступає в ролі клієнта, а метод реалізується в мові Dart.

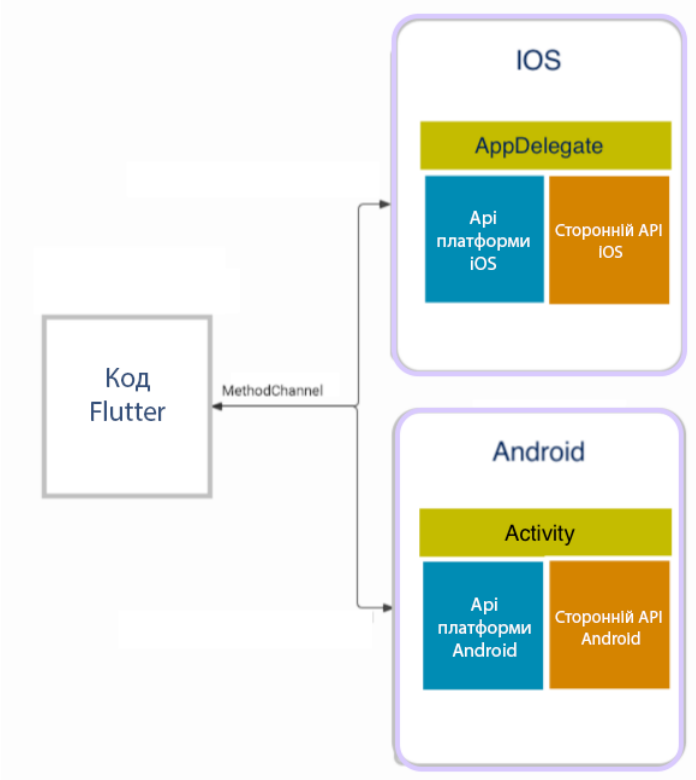


Рисунок 4.1 – Архітектура каналів платформи

Порівняно зі своїм прямим конкурентом – технологією React Native [14], Flutter має кілька переваг, які роблять цей інструмент більш кращим для розробки застосунка.

Програми, написані на React Native, працюють у 6-20 разів повільніше порівняно з нативними. У той час як у Flutter продуктивність знижується лише у 2-5 разів [14].

Основна ідея платформи Flutter базується на віджетах. Розробники можуть створити елегантний і виразний інтерфейс користувача, комбінуючи різні віджети відповідно до бізнес-моделі клієнта. Фреймворк має багаті віджети для структурних елементів, стилістичних елементів тощо. Розробники програм Flutter також можуть створювати власні віджети та бути впевненими у нативній продуктивності програми. Завдяки цьому, з Flutter застосунок має єдиний зовнішній вигляд, незважаючи на версію ОС або модель пристрою. React Native ж успадковує нативні візуальні елементи,

тому застосунок буде виглядати трохи по-різному навіть за різних версій прошивки пристрою, не кажучи вже про відмінності між iOS та Android [15].

Як СУБД використовується PostgreSQL, на її вибір вплинули наступні переваги:

1) вихідний код PostgreSQL знаходиться у вільному доступі за ліцензією з відкритим кодом, що дозволяє вільно використовувати та впроваджувати його відповідно до потреб бізнесу;

2) безліч компаній та приватних осіб роблять свій внесок у розвиток PostgreSQL: сильна спільнота гарантує, що помилки у коді СУБД виправляються без затримок;

3) існує багато технічних можливостей для масштабної експлуатації PostgreSQL, тому база даних PostgreSQL може зростати та бути настільки великою, наскільки це необхідно;

4) високостійкість PostgreSQL завдяки журналюванню з випередженням запису.

Щоб забезпечити достатній рівень захищеності системи, паролі не зберігаються в базі даних у відкритому вигляді. Завдяки адаптивній криптографічній хеш-функції bcrypt формується ключ, який використовується для захищеного зберігання паролів. Для реалізації такого шифрування використовується бібліотека node-bcrypt-js. На основі паролю користувача створюється хеш, який зберігається у базі даних. При аутентифікації цей хеш порівнюється з введеним паролем. Якщо хеш і пароль співпадають – аутентифікацію можно вважати успішною.

5 РЕАЛІЗАЦІЯ КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ РЕПЕТИТОРІВ

5.1 Створення бази даних

Створення типів даних розглядається на прикладі типу `STUDENT_STATUS`:

```
CREATE TYPE STUDENT_STATUS as ENUM ('ACTIVE', 'ARCHIVED');
```

Після ключових слів `CREATE TYPE` подається ім'я типу. Далі зазначається, що дані цього типу можуть містити значення ACTIVE або ARCHIVED.

Створення таблиць розглядається на прикладі таблиці `Student`:

```
CREATE TABLE Student(
    student_id SERIAL NOT NULL PRIMARY KEY,
    student_status STUDENT_STATUS DEFAULT 'ACTIVE',
    student_name VARCHAR NOT NULL,
    student_class VARCHAR,
    phone_number VARCHAR,
    address VARCHAR,
    base_price NUMERIC (9,2) NOT NULL CHECK base_price>0,
    notes VARCHAR,
    user_id INT NOT NULL REFERENCES User );
```

Після ключових слів `CREATE TABLE` вказується ім'я створюваної таблиці. Кожному імені поля зіставляються тип даних PostgreSQL і обмеження цілісності. Поле `student_id` є первинним ключем (`PRIMARY KEY`) і має тип `SERIAL`, що являє собою спосіб створення унікального цілочисельного значення для поля шляхом збільшення попереднього значення на 1 (за замовчуванням). Поле `student_status` має тип `STUDENT_STATUS` та обмеження `DEFAULT`, що встановлює значення поля за замовченням на ACTIVE. Поля `student_name` та `base_price` мають обмеження `NOT NULL`, тобто ці поля не можуть містити порожні значення. Поле `base_price` також має накладене обмеження `CHECK base_price>0`, і це означає, що значення цього поля повинні бути позитивними.

Запити для створення інших обєктів бази даних представлені у додатку А.

5.2 Запити до бази даних для вирішення задач користувачів

Для вирішення підзадач 1.1, 2.1 використовується запит типу `INSERT INTO` *`таблиця`*`(стовпець1, стовпець2, стовпець3) VALUES (значення1, значення2, значення3)`. Параметр значення у обох випадках передається з відповідного елемента управління інтерфейсу користувача.

Для вирішення підзадач 1.2, 1.3, 2.2, 2.3, 3.1, використовується запит типу `UPDATE таблиця SET стовпець = значення WHERE id=значення`. Значення ідентифікатора екземпляра об'єкта, що видаляється, передається з відповідного елемента управління інтерфейсу користувача.

Для вирішення підзадач 1.4, 2.4 використовуєтьс запити типу `SELECT * FROM` *`таблиця`* `WHERE id=значення`. Значення ідентифікатора екземпляра об'єкта, що видаляється, передається з відповідного елемента управління інтерфейсу користувача.

Для вирішення підзадачі 3.2 використовується запит типу `SELECT DATE_TRUNC ('month', txn_date) AS txn_month, SUM (actual_price) as monthly_sum FROM Lesson GROUP BY txn_month`. Так само отримуються значення на день та рік.

Для вирішення підзадачі 3.3 використовується запит типу `SELECT COUNT(*) FROM Lesson WHERE student_id = значення`.

5.3 Реалізація шаблону MVC у застосунку

Завдяки шаблону MVC, застосунок, що розроблюється, поділяється на наступні логічні частини:

1) Model – реалізує логіку зберігання і модифікації даних;

2) View – надає користувачеві інтерфейс для взаємодії з застосунком;

3) Controller – забезпечує обробку подій та взаємодіє з даними у БД.

Реалізація шаблону проектування MVC в розробленому програмному продукті представлена на рис 5.1.

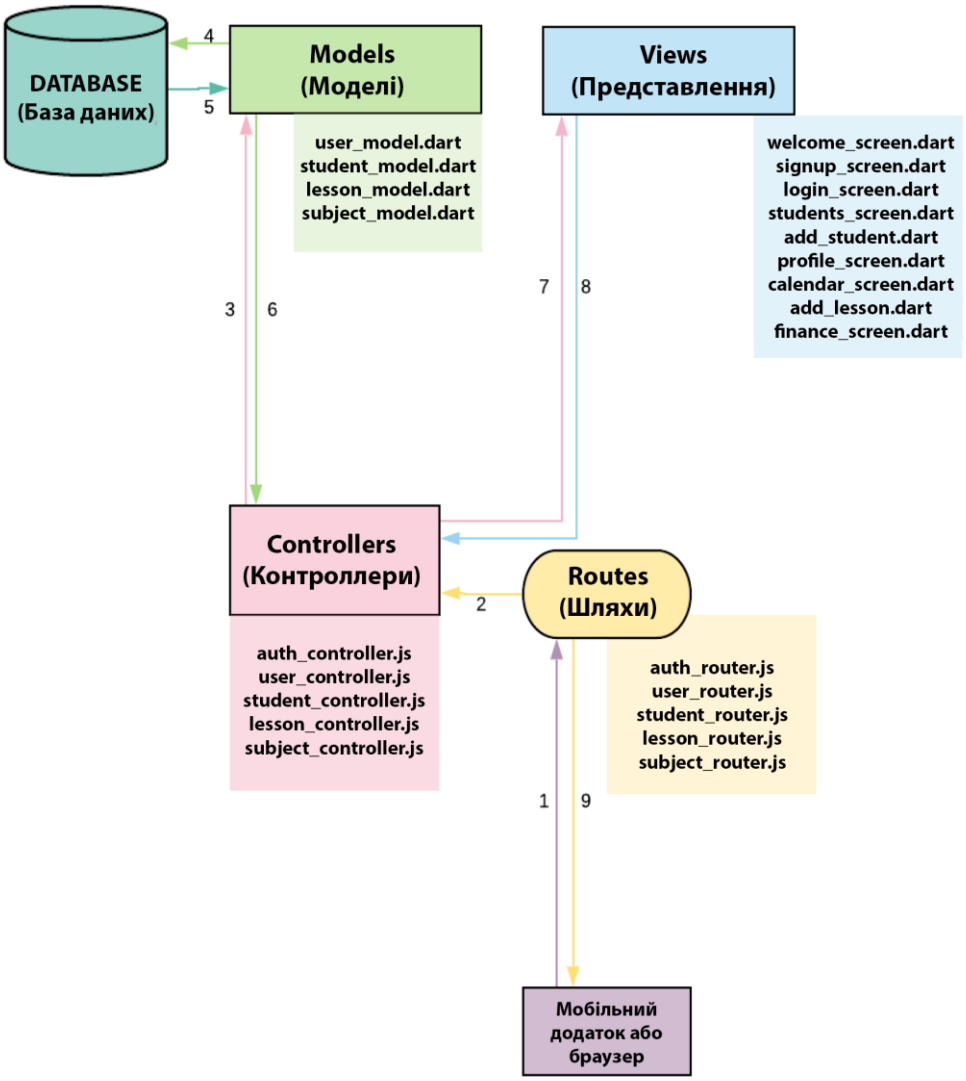


Рисунок 5.1 – Реалізація шаблону MVC в розробленому застосунку

Під кожним із складових шаблону показана його прив’язка до створених класів.

У представленні знаходяться наступні класи:

– welcome_screen – вітальний екран;
– signup_screen – екран реєстрації;
– login_screen – екран авторизації;
– students_screen – екран для перегляду учнів;
– add_student – вікно додавання нового учня;
– profile_screen – екран детальної інформації про учня;
– calendar_screen – екран з розкладом занять;

– add_lesson – вікно створення нового заняття;

– finance_screen – екран для відображення фінансової статистики.

Структура контролерів представлена наступними компонентами: Auth, User, Student, Lesson, Subject. Кожний з цих компонентів відповідає за маніпуляції зі своєю областю.

Контролери мають наступні методи:

– getAll – метод для отримання всіх записів;

– getOne – метод для отримання конкретного запису;

– editOne – метод для редагування конкретного запису;

– addOne – метод для створення нового запису.

Модель представлена наступними класами для роботи з базою даних:

– User – опис моделі користувача, що має поля username, email, password;

– Student – опис моделі учня, що має поля student_status, student_name, student_class, phone_number, address, base_price, notes, user_id;

– Lesson – опис моделі заняття, що має поля lesson_status, lesson_date, lesson_start_time, lesson_duration, lesson_type, actual_price, homework_status, payment_status, student_id, subject_id;

– Subject – опис моделі предмета, що має поле subject_name.

5.4 Безпека застосунку

Для реалізації авторизації та аутентифікації використаний стандарт JSON Web Token (JWT) [16], який визначає компактний та автономний спосіб безпечної передачі інформації між сторонами у вигляді об'єкту JSON.

JWT відрізняються від інших веб-маркерів тим, що містять набір тверджень. Твердження використовуються для передачі інформації між двома сторонами. Чим є ці твердження, залежить від конкретного випадку використання. Наприклад, формула може стверджувати, хто видав токен, як довго він дійсний або які дозволи були надані клієнту.

JWT є рядком, що складається з трьох частин, розділених точками, і серіалізований з використанням стандарту base64 (рис. 5.2). У найбільш поширеному форматі серіалізації, компактній серіалізації, JWT виглядає наступним чином: `Header.Payload.Signature`

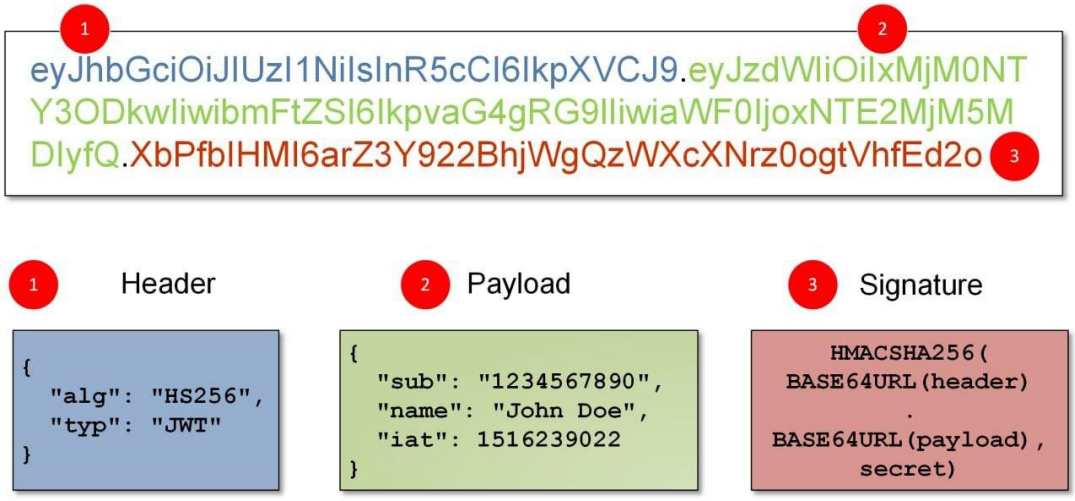


Рисунок 5.2 – Структура JWT

Після декодування отримуємо два рядки JSON: заголовок та корисне навантаження і підпис. Заголовок JOSE (JSON Object Signing and Encryption) містить тип токена (у даному випадку JWT) і алгоритм підпису. Корисне навантаження містить затвердження. Вони відображаються у вигляді рядка JSON, що зазвичай містить не більше десятка полів, щоб JWT був компактним. Ця інформація зазвичай використовується сервером для перевірки того, що користувач має дозвіл на виконання дії, що запитується. Підпис гарантує, що токен не було змінено. Сторона, що створює JWT, підписує заголовок та корисне навантаження секретом, який відомий як емітенту, так і одержувачу, або закритим ключем, відомим лише відправнику. Коли токен використовується, сторона, що приймає, перевіряє відповідність заголовка і корисного навантаження підпису.

5.5 Реалізація кросплатформності у застосунку

Головна ідея Flutter полягає в побудові кросплатформного інтерфейсу користувача за допомогою елементів, які називаються віджетами. Віджети описують, як має виглядати їхнє уявлення з урахуванням поточної конфігурації та стану. Коли стан віджета змінюється, він перебудовує свій опис, який фреймворк порівнює з попереднім описом, щоб визначити мінімальні зміни, необхідні у дереві рендерингу для переходу з одного стану до іншого. Для прикладу такого підходу розглянемо, як був реалізований екран редагування профілю учня.

Віджети в Flutter поділяються на дві групи: Stateful (зі станом) та Stateless (без стану). Якщо віджет може змінюватися, наприклад, коли користувач взаємодіє з ним, він є віджетом зі станом. Віджет без стану ніколи не змінюється. Екран редагування профілю учня є прикладом Stateful віджета, оскільки містить елементи з якими може взаємодіяти користувач.

Реалізація віджету користувача зі станом вимагає створення двох класів: перший з класів описує об'єкти (див. лістинг 5.1), які створюються при кожному малюванні. Екземпляр цього класу не створюється кожний раз заново і використовується для збереження переданих параметрів та ініціалізації стану.

```
class EditStudent extends StatefulWidget {
  const EditStudent({Key key}) : super(key: key);
  @override
  State<EditStudent> createState() => _EditStudentState();}
```

Лістинг 5.1 – Створення класу `EditStudent`

Другий клас – це клас стану. Він займається безпосередньо відмальовуванням, реагуючи на зміни (див. лістинг 5.2). Зміна стану Stateful віджетів є основним джерелом перемальовки інтерфейсу. Для цього необхідно змінити його властивості всередині виклику методу зміни стану.

Таким чином, на відміну від інших фреймворків, у Flutter немає явного відстеження стану. Будь-яка зміна властивостей віджету поза методом зміни стану не призведе до перемальовування інтерфейсу.

```
class _EditStudentState extends State<EditStudent> {
     @override
     Widget build(BuildContext context) {
     }
}
```

Лістинг 5.2 – Створення класу `_EditStudentState`

Усередині віджету стану `_EditStudentState` оголошено змінні типу `TextEditingController` (див. лістинг 5.3), які дозволяють відстежувати зміну текстових полів із відповідним контролером. Текстове поле оновлює значення, а контролер повідомляє своїх слухачів. Слухачі можуть читати властивості тексту та значення, щоб дізнатися, що запровадив користувач.

Сам віджет повертає елемент `Scaffold()`, який виконує функцію головного контейнера та відповідає за структуру екрану. У тілі `Scaffold()` знаходяться віджети `Container()` та `Column()`, які відповідають за позиціювання віджетів для введення тексту `InputField()` до яких і прив’язані текстові контролери.

```
class _EditStudentState extends State<EditStudent> {
     final studentNameController = TextEditingController();
     final basePriceController = TextEditingController();
     @override
     Widget build(BuildContext context) {
     return Scaffold(
       appBar: AppBar(
          backgroundColor: kPrimaryDarkColor,
          automaticallyImplyLeading: false,
          title: Text("Изменить профиль"), centerTitle: true, ),
       body: Container(
          padding: EdgeInsets.all(20), child: Column(
```

Лістинг 5.3 – Наповнення віджета стану

```
          children: [
            InputField(
              title: "Имя ученика", hint: "Введите имя ученика",
              controller: studentNameController,
             ),
            InputField(
               title: "Стоимость занятия", hint: "00.00",
               controller: basePriceController,
               keyboardType: TextInputType.number,
               ),    ],   ),   );
      }
}
```

Лістинг 5.3, лист 2

У традиційних Android-застосунках при малюванні елементів інтерфейсу спочатку викликається Java-код фреймворку Android. Системні бібліотеки Android надають компоненти, що відповідають за малювання, в об'єкт Canvas, який Android потім може відмалювати за допомогою Skia, графічного движка, написаного на C/C++, який викликає CPU або GPU для завершення малювання на пристрої.

Кросплатформні фреймворки зазвичай працюють шляхом створення шару абстракції над базовими бібліотеками інтерфейсу користувача Android і iOS, намагаючись згладити невідповідності в предславленні кожної платформи. Код програм часто пишеться на інтерпретованій мові, наприклад на JavaScript, яка, у свою чергу, повинна взаємодіяти з системними бібліотеками Android на Java або iOS на Objective-C для відображення інтерфейсу користувача. Все це додає накладні витрати, які можуть бути значними, особливо там, де існує багато взаємодій між інтерфейсом користувача і логікою програми. Flutter, який застосовується у даній роботі, навпаки, мінімізує ці абстракції, обходячи системні бібліотеки віджетів інтерфейсу користувача на користь власного набору віджетів. Код Dart, який малює візуальні ефекти Flutter, компілюється у нативний код, який використовує Skia для рендерингу [17].

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

Перше, що бачить користувач при запуску застосунку – це привітальне вікно (рис. 6.1), у якому йому пропонується увійти в свій обліковий запис за допомогою вказаних раніше при реєстрації електронної пошти та паролю (рис. 6.2, а), або створити новий обліковий запис, увівши адресу електронної пошти та пароль (рис 6.2, б).

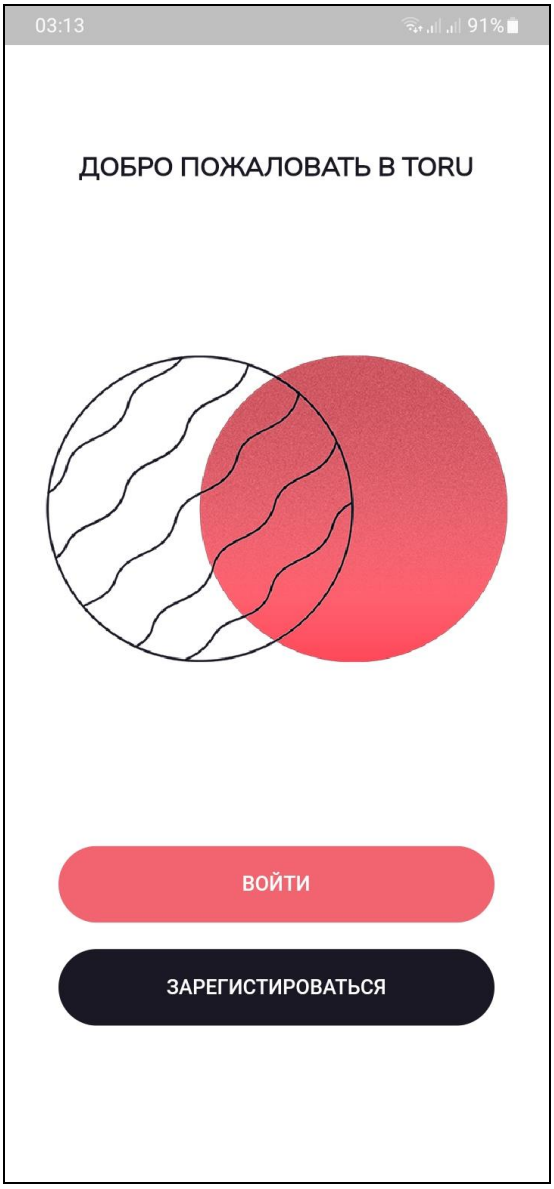
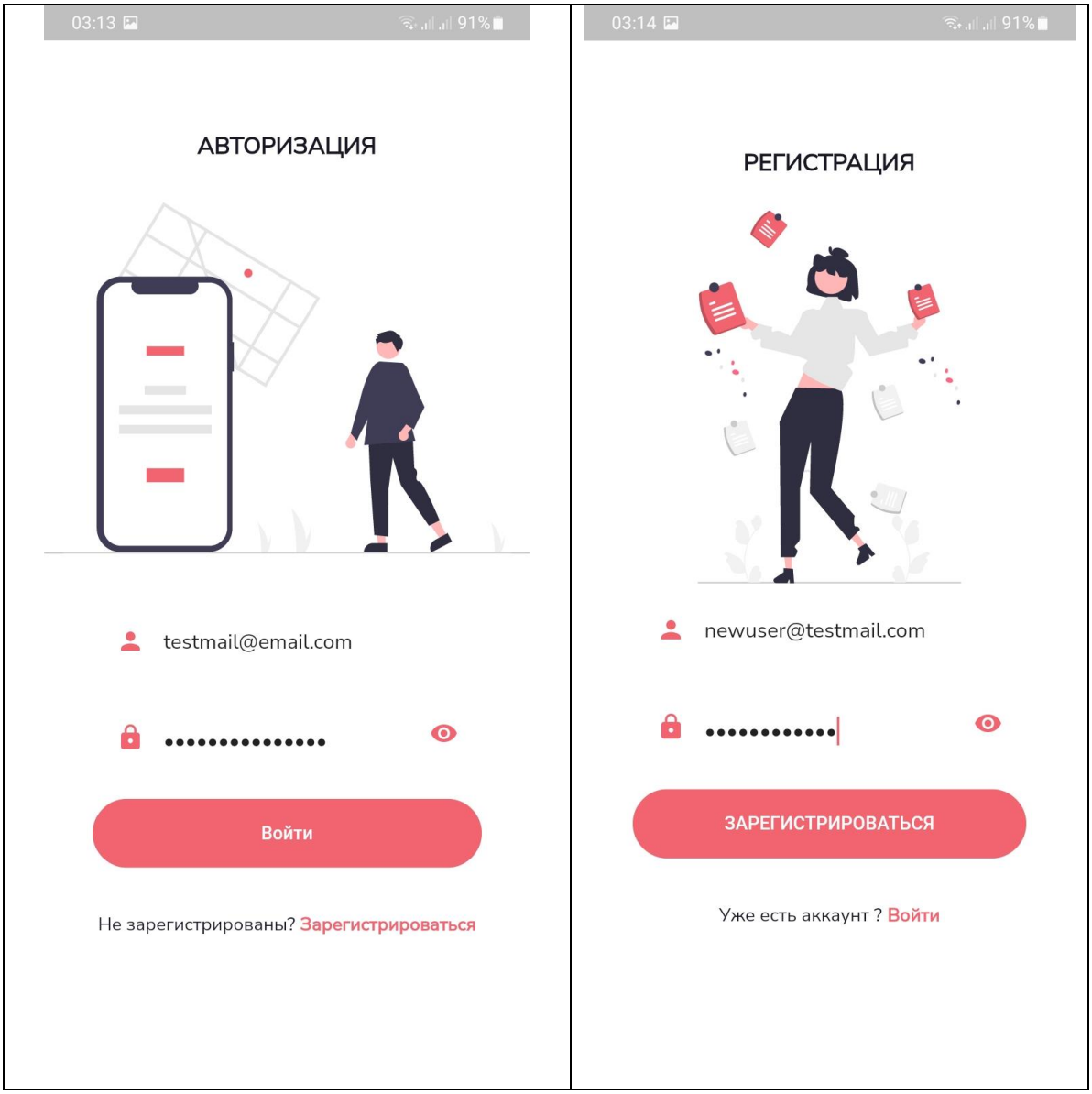


Рисунок 6.1 – Привітальний екран



а) б)

Рисунок 6.2 – Авторизація (а) та реєстрація нового користувача (б)

Після успішної авторизації користувач потрапляє на екран зі списком учнів (рис 6.3). Біля імені та прізвища кожного учня відображається вартість заняття.

Внизу екрана знаходиться меню, яке дозволяє користувачеві перемикатися між вкладками. Усього доступні три вкладки: «Учні», «Розклад» та «Фінанси».

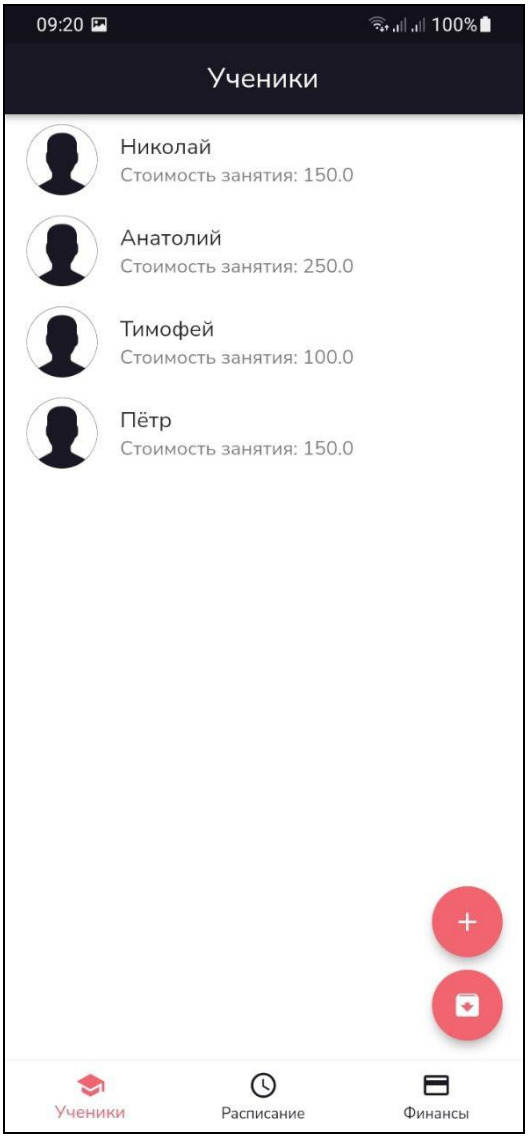
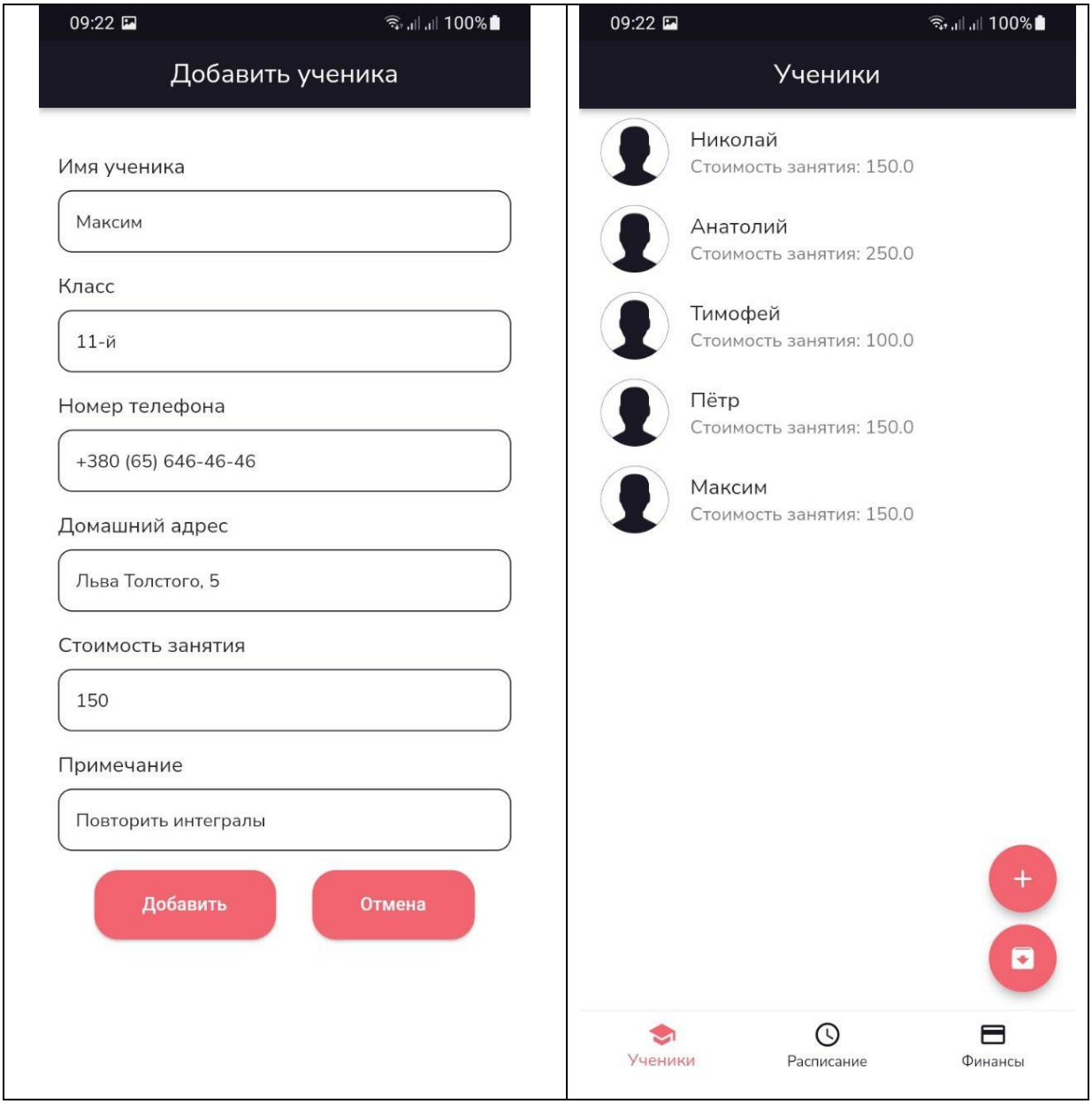


Рисунок 6.3 – Відображення списку учнів

Для того щоб створити новий профіль учня, необхідно натиснути кнопку зі знаком «плюс». Після цього з'явиться вікно створення нового профілю учня (рис 6.4, а). Необхідно ввести такі дані як ім’я та прізвище, клас, номер телефону, домашня адреса, вартість заняття та, при необхідності, додати примітку. Дані про ім’я і вартість заняття є обов’язковими для заповнення, інші поля можна залишити порожніми. Після внесення необхідної інформації (рис. 6.4, а) та натискання кнопки «Додати» буде створений новий профіль учня (рис. 6.4, б). Кнопка «Отмена» повертає користувача на екран зі списком учнів.

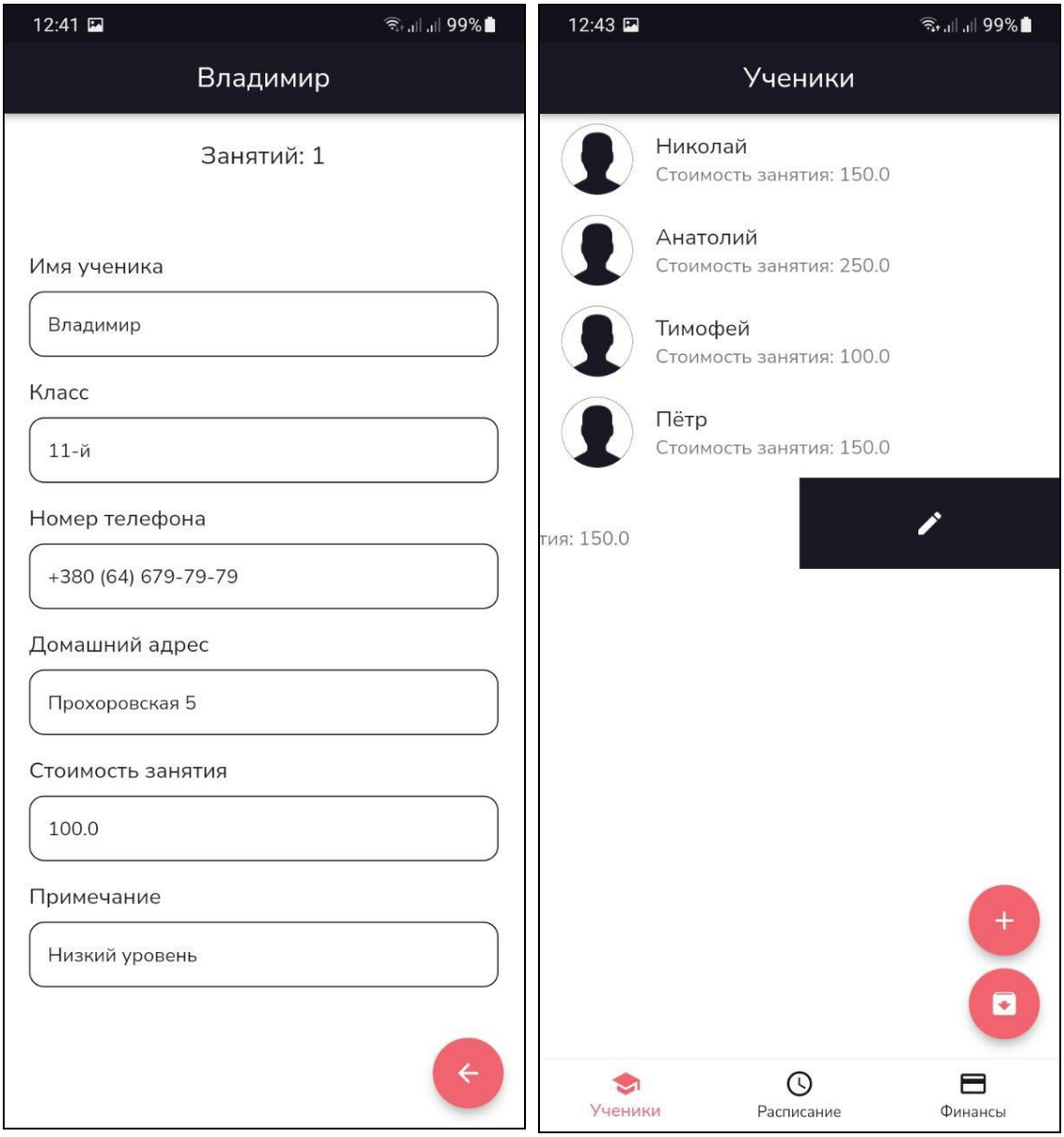


а) б)

Рисунок 6.4 – Додавання нового учня (а) та його відображення у списку (б)

Якщо натиснути на профіль учня, з'явиться екран з усією необхідною контактною інформацією і кількістю занять, які репетитор з ним провів (рис. 6.5, а). Кнопка зі стрілкою вліво дозволяє повернутися до списку усіх учнів.

Якщо провести пальцем вліво по рядку, що відповідає одному з учнів, з'явиться кнопка редагування профілю учня (рис. 6.5, б).

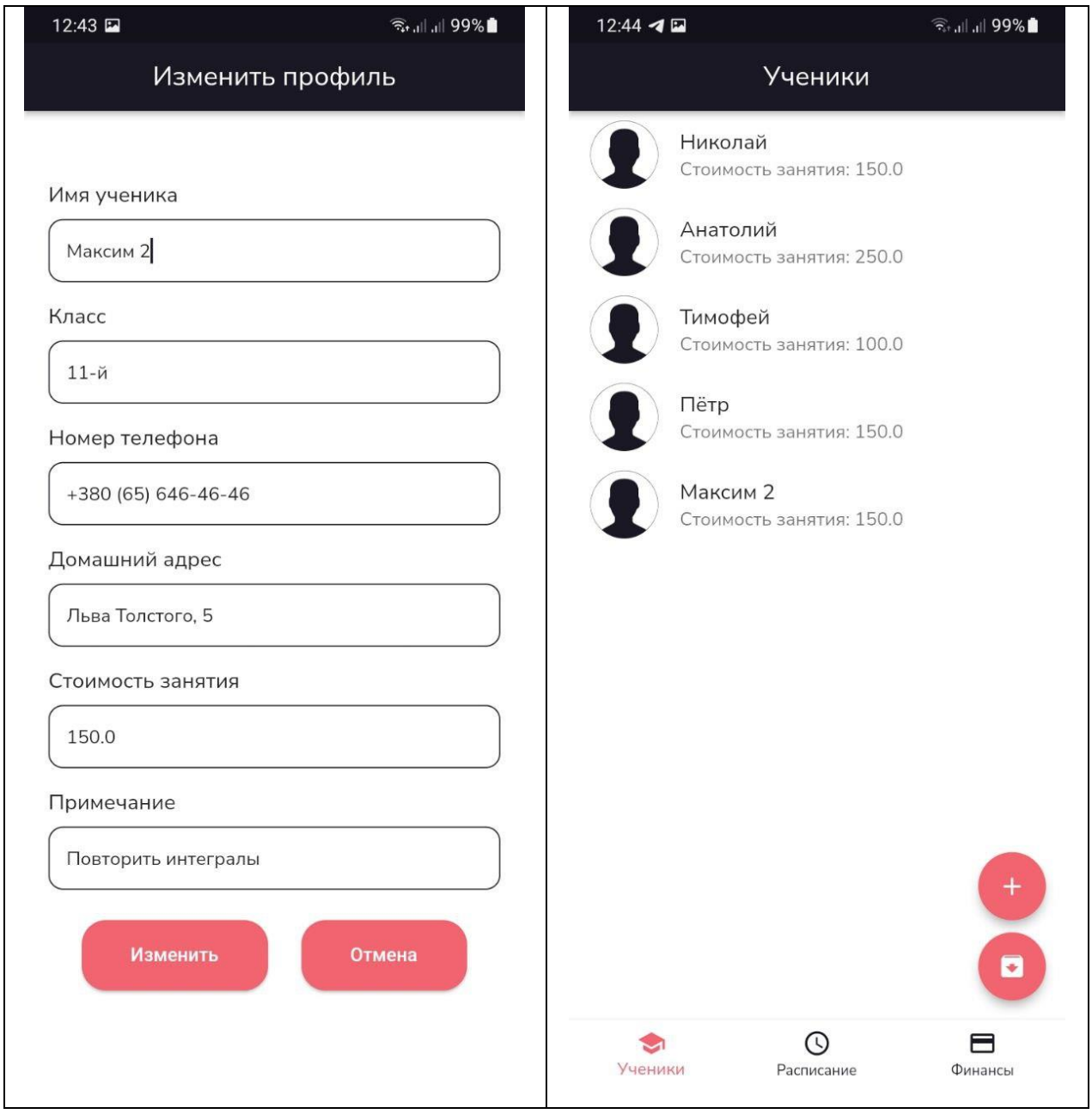


а) б)

Рисунок 6.5 – Профіль учня (а) та кнопка редагування профілю (б)

Після натискання цієї кнопки з'явиться вікно для редагування інформації (рис. 6.6, а). Відредагувати можна імя учня, клас, номер його телефону, адресу, вартість заняття та примітки.

Після зміни необхідної інформації та натискання кнопки «Змінити» користувач знов повернеться на екран з списком учнів та зможе побачити внесені зміни (рис. 6.6, б).



а) б)

Рисунок 6.6 – Редагування учня (а) та його оновлене відображення у списку (б)

Якщо провести пальцем вправо по одному з учнів, з'явиться кнопка додавання учня в архів (рис. 6.7).

Необхідно зазначити, що у застосунку не передбачається повне видалення учнів, оскільки при розрахунку фінансових даних інформація про оплату занять видаленими учнями не потрапить у розрахунок.

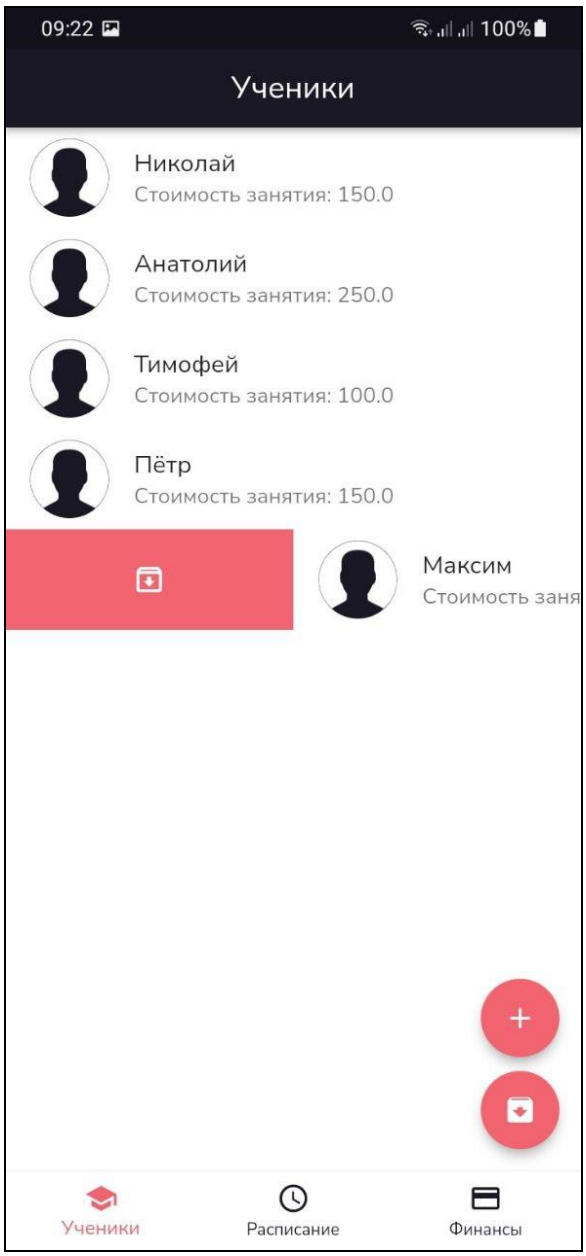
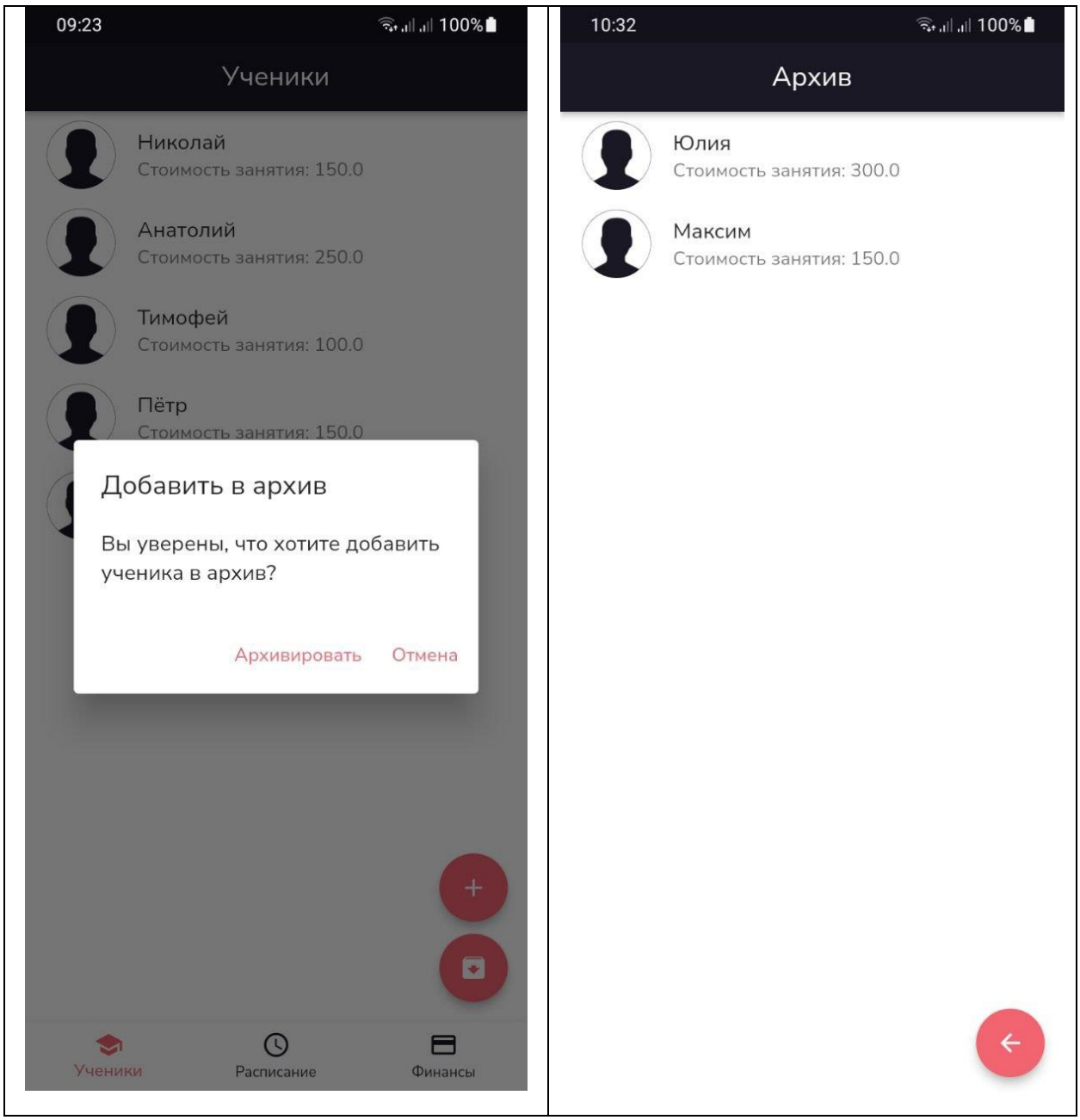


Рисунок 6.7 – Кнопка додавання учня в архів

Після натискання цієї кнопки з'явиться попереджуюче модальне вікно (рис. 6.8, а) з проханням підвердити чи відмінити додавання учня до архіву.

Після того, як учень потрапить до архіву, його інформація не видаляється, замість цього він перестає відображатися у списку учнів та потрапляє на окремий екран архівованих учнів (рис. 6.8, б).



а) б)

Рисунок 6.8 – Архівація учня

Якщо так само провести пальцем вправо по рядку, що відповідає одному з архівованих учнів, з'явиться кнопка деархівації (рис. 6.9). Після натискання цієї кнопки з'явиться попереджуюче модальне вікно. Після того, як учень потрапить до архіву, його інформація не видаляється, але він перестає відображатися у списку архівованих учнів та знов потрапляє на загальний екран учнів.

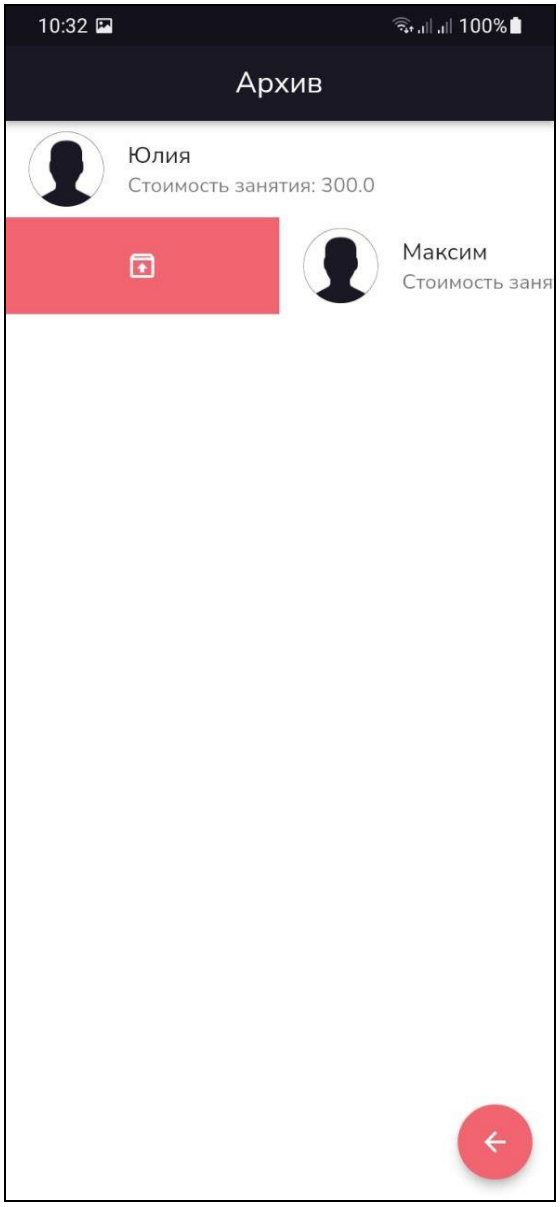
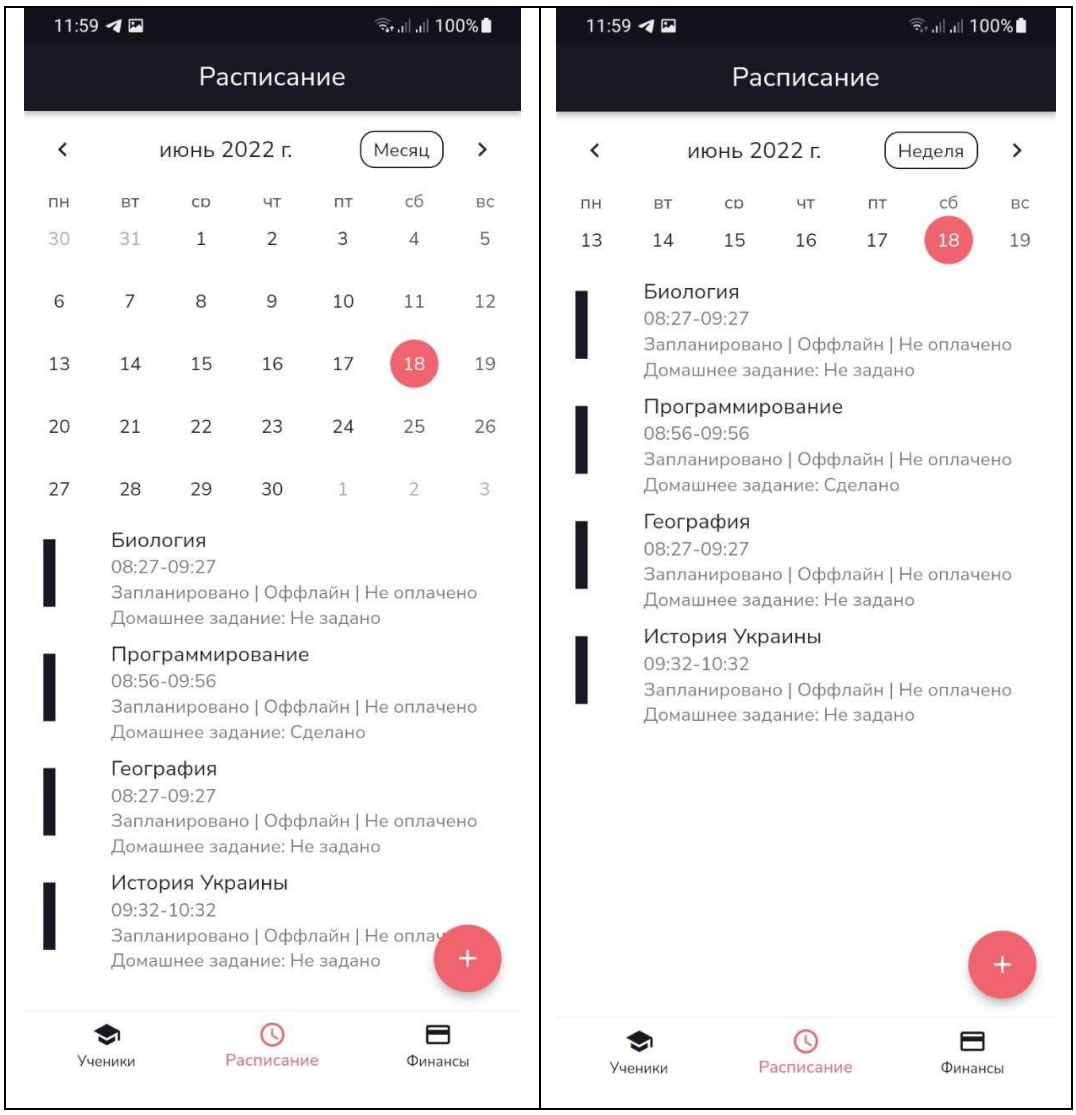


Рисунок 6.9 – Кнопка повернення учня з архіву

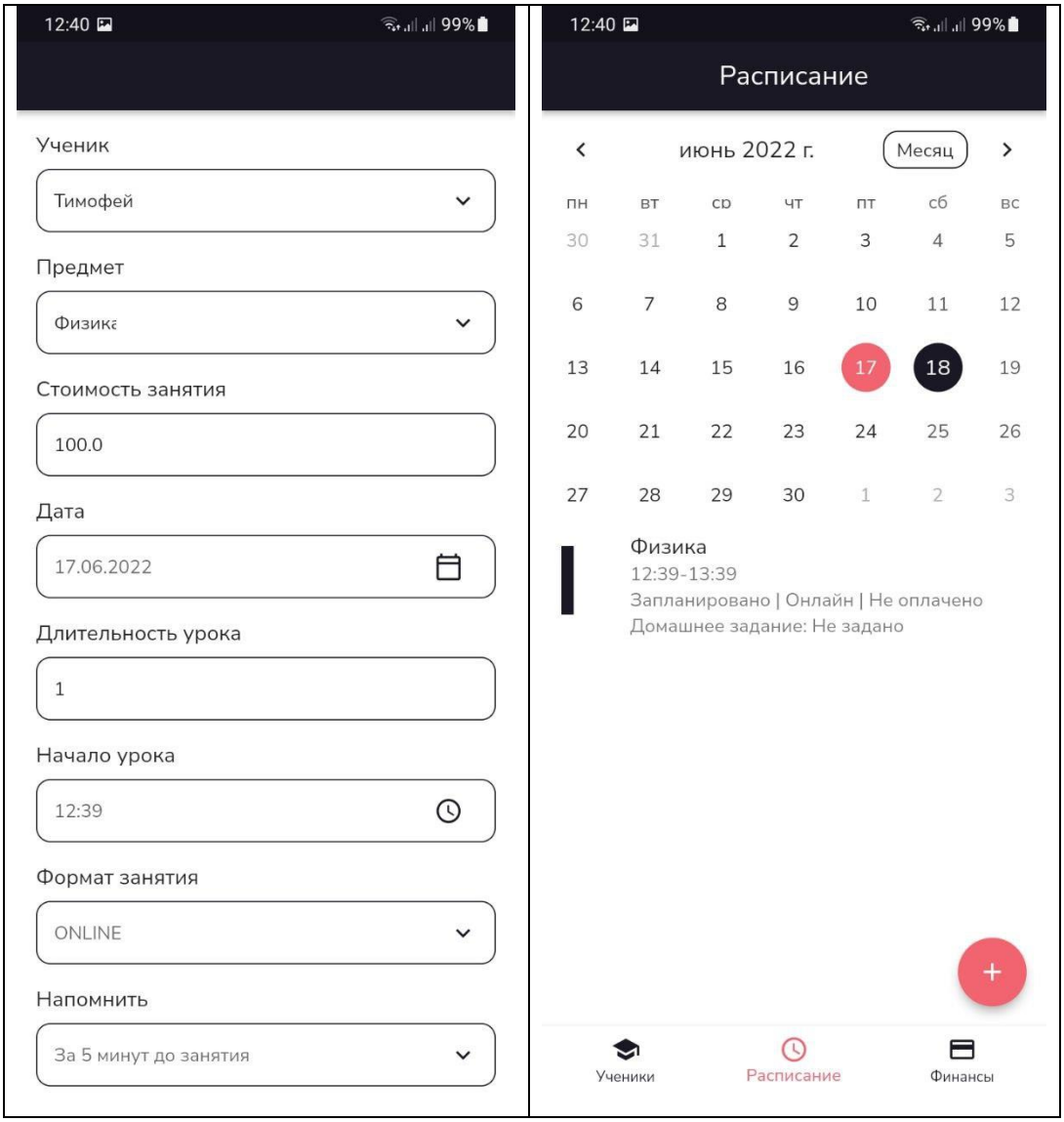
Для планування занять необхідно відкрити екран розкладу занять. На ньому відображений календар, який має два режими: «місяць» (рис. 6.10, а) та «тиждень» (рис. 6.10, б). Відповідно до вибраної дати відображається список занять, запланованих на цей день, з зазначенням часу проведення заняття, предмету, статусу домашнього завдання, форми проведення заняття та статусу оплати.



а) б)

Рисунок 6.10 – Режими календаря: «місяць» (а) та «тиждень» (б)

При натисканні на кнопку зі занком «+» відкриється екран планування нового заняття (рис. 6.11, а). Учень, з яким буде проводитися заняття, обирається зі списку існуючих учнів. Предмети обираються з готового списку, який покриває основні області надання послуг репетиторів та вже заповнений при інсталяції застосунку. Поле «Вартість заняття» за замовченням містить в собі базову вартість заняття з учнем. Якщо репетитору за якихось причин потрібно змінити ціну для цього заняття, він може ввести нове значення. Час закінчення заняття прораховується автоматично. Після заповнення інформації та натискання кнопки «Додати» створюється нове заплановане заняття (рис. 6.11, б).



а) б)

Рисунок 6.11 – Планування заняття: дані про нове заняття (а) та відображення запланованого заняття (б)

Також можна виставити час, за який перед початком заняття репетитор отримає нагадування про нього (рис. 6.12).

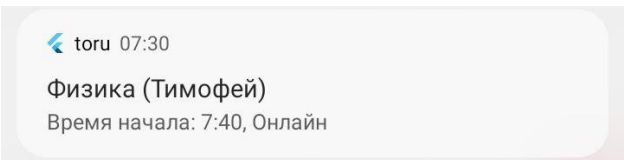
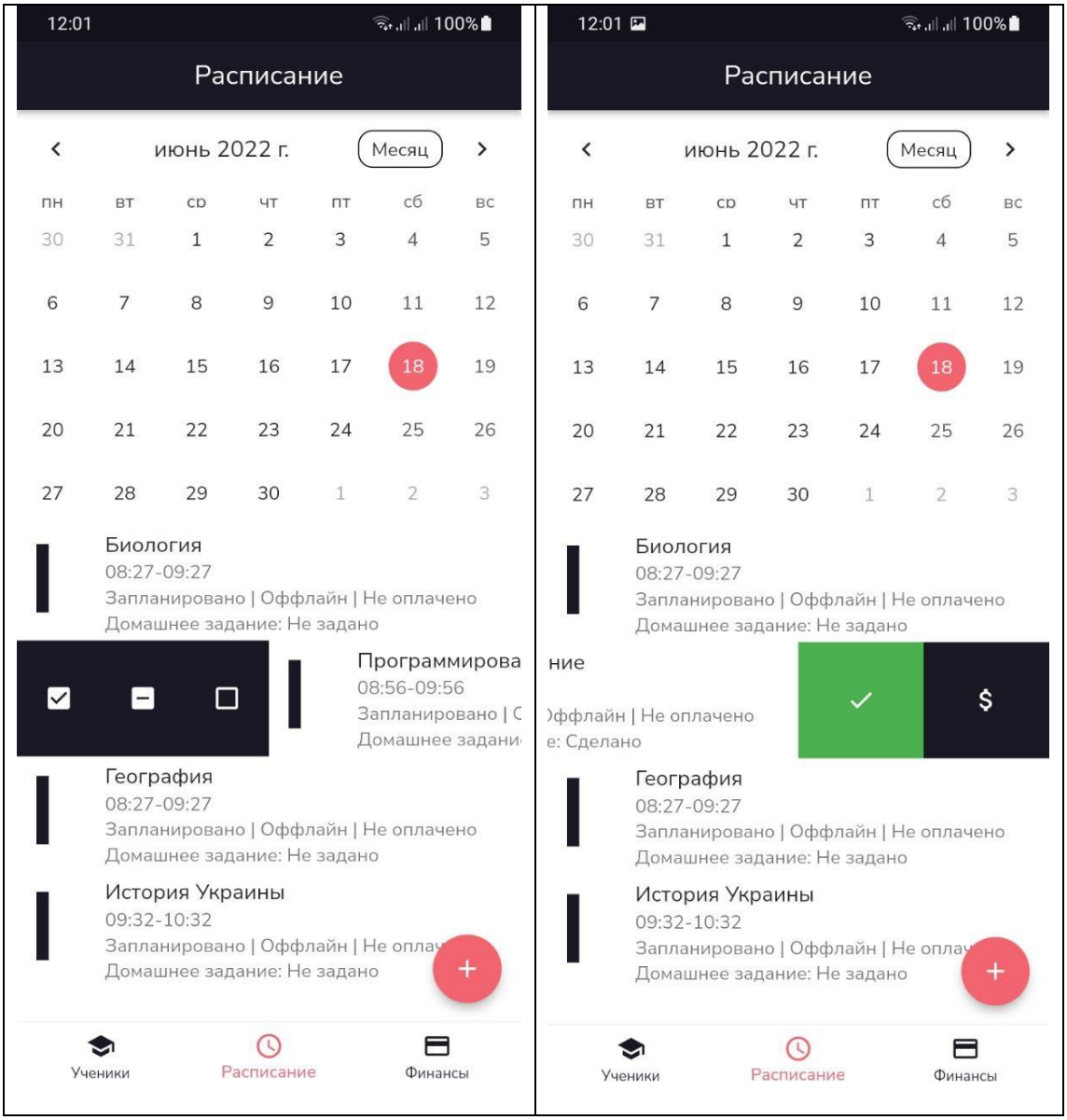


Рисунок 6.12 – Нагадування про заняття

Якщо провести пальцем вправо по одному з занять, з'являться кнопки керуванням статусом доманшнього завдання, які відображають стани «Виконано», «Не виконано», «Не задано» (рис. 6.13, а). При проведенні пальцем вліво з'являються кнопки зміни статусу заняття і статусу оплати за нього (рис. 6.13, б). Кнопка з іконкою грошей змінює статус на «Оплачено», після чого змінюється сума заборгованості учня на екрані фінансів. Кнопка з галкою змінює статус заняття на «Проведене» і воно перестає відображатись у списку запланованих занять.



а) б)

Рисунок 6.13 – Керування заняттями

На екрані фінансів відображається прибуток за певні часові рамки (тиждень, місяць, рік). Також на цьому екрані знаходиться список учнів, заняття з якими були проведені, але не були оплачені, з відображенням заборгованності по цим заняттям (рис. 6.14).

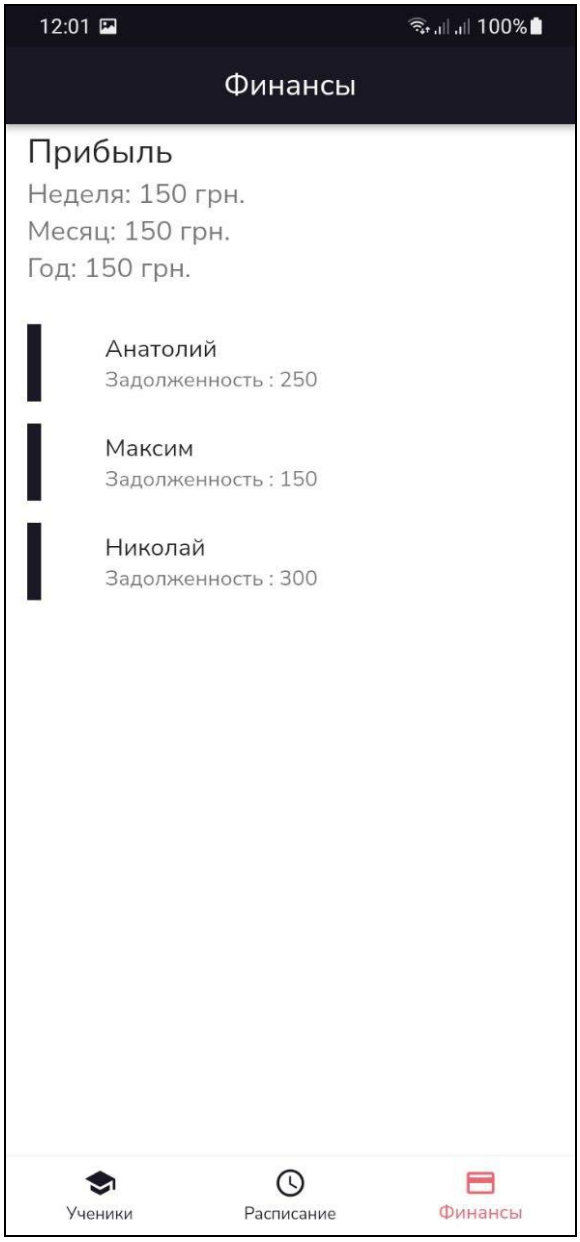


Рисунок 6.14 – Екран фінансів

ВИСНОВКИ

При виконанні дипломної роботи проаналізовані особливості роботи репетитора з урахуванням основних потреб ФОП та виділені задачі потенційних користувачів застосунка. Огляд існуючих застосунків для організації професійної діяльності репетиторів показав, що конкурентні аналоги можуть задовольнити далеко не всі потреби потенційного користувача, а деякі з них мають проблеми зі стабільностю роботи або не здатні працювати на різних програмно-апаратних платформах.

Спроектований та розроблений мобільний застосунок для iOS та Android вирішує описані проблеми шляхом реалізації функцій, відсутніх у застосунках-конкурентах, а також завдяки застосуванню платформно незалежних технологій для створення програмного продкуту. В результаті репетитор отримує набір простих та зрозумілих інструментів, завдяки яким він здатний організовувати, аналізувати та корегувати свою діяльність.

Для зберігання даних, якими оперує репетитор при своїй діяльності, спроектована та створена база даних під управлінням СУБД PostgreSQL. Для реалізації застосунку використані сучасні засоби створення кросплатформного програмного забезпечення, зокрема, трирівнева архітектура клієнт-сервер, шаблон проектування MVC та архітектурний підхід REST. Серверна частина застосунку створена за допомогою мови програмування JavaScript, платформи Node.js, фреймворка Express. Кросплатформна клієнтська частина застосунку побудована за допомогою технології Flutter.

Система дозволяє користувачу-репетитору:

1) вести облік учнів та мати своєчасний доступ до всієї необхідної контактної інформації;

2) складати розклад занять з автоматичним прорахунком часу закінчення уроку, залежно від його тривалості;

3) створювати нагадування про майбутні заняття;

4) контролювати статус виконання домашнього завдання (не задано/виконано/не виконано);

5) регулювати тип заняття, що проводиться (онлайн/оффлайн);

6) переглядати статистику щодо учнів;

7) відстежувати заборгованість з оплати занять;

8) отримувати інформацію про доходи за певні часові рамки.

Як перспектива розвитку застосунку розглядається його модернізація в повноцінну онлайн платформу для навчання, де буде реалізована система особистих кабінетів для учнів, в яких вони зможуть отримувати та відправляти домашнє завдання, проходити тестування та в один клік оплачувати індивідуальні заняття. У той же час викладачі зможуть проводити свої уроки за допомогою вбудованої інтерактивної дошки з можливістю голосового спілкування з учнем, що позбавить необхідності використання сторонніх месенджерів та програм.

Ідея створення програмного продукту та його проект представлені на дев’ятнадцятій всеукраїнській конференції студентів і молодих науковців, тези доповіді опубліковані.

Кросплатформний застосунок для організації роботи репетиторів прийнятий до використання в творчій студії «Art&life» (див. додаток Б).

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Підсумки навчального року 2020/2021 від БУКІ: як ми побороли кризу і карантин та інші досягнення в числах і фактах [Електронний ресурс] // Режим доступу: https://buki.com.ua/ru/news/pidsumky-2020-2021/
2. Глава 5. Фізична особа-підприємець. Стаття 50. Право фізичної особи на здійснення підприємницької діяльності [Електронний ресурс] // Режим доступу: https://i.factor.ua/ukr/law-54/section-296/article-5271/
3. What is BPMN? [Електронний ресурс] // Режим доступу: https://www.visual-paradigm.com/guide/bpmn/what-is-bpmn/
4. Light – приложение для репетиторов [Електронний ресурс] // Режим доступу: https://light-app.net
5. TutorTask для репетитора [Електронний ресурс] // Режим доступу: https://play.google.com/store/apps/details?id=com.tutortask.app&hl=ru&gl=US
6. ЯРепетитор [Електронний ресурс] // Режим доступу: https://yarepetitor.com
7. Кравченко К.Д, Розновець О.І. Кросплатформний застосунок для організації роботи репетиторів / Інформатика, інформаційні системи та технології: тези доповідей дев’ятнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 29 квітня 2022 р. – Одеса, 2022. – с. 98-100
8. What is Client Server Architecture? [Електронний ресурс] // Режим доступу: https://intellipaat.com/blog/what-is-client-server-architecture/
9. Benefit of using MVC [Електронний ресурс] // Режим доступу: https://www.geeksforgeeks.org/benefit-of-using-mvc/
10. What is REST? [Електронний ресурс] // Режим доступу: https://www.codecademy.com/article/what-is-rest
11. Relational Database Benefits and Limitations (Advantages & Disadvantages) [Електронний ресурс] // Режим доступу: https://databasetown.com/relational-database-benefits-and-limitations/

12. Draw Entity-Relationship Diagrams, Painlessly [Електронний ресурс] // Режим доступу: https://dbdiagram.io/
13. What Is Node.js and Why You Should Use It [Електронний ресурс] // Режим доступу: https://kinsta.com/knowledgebase/what-is-node-js/
14. Top 8 Flutter Advantages and Why You Should Try Flutter on Your Next Project [Електронний ресурс] // Режим доступу: https://relevant.software/blog/top-8-flutter-advantages-and-why-you-should-try-flutter-on-your-next-project/
15. Flutter vs Native vs React-Native: Examining performance [Електронний ресурс] // Режим доступу: https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980
16. REACT NATIVE VS. FLUTTER: WHAT IS BETTER FOR APP DEVELOPMENT IN 2021? [Електронний ресурс] // Режим доступу: https://nix-united.com/blog/flutter-vs-react-native/
17. Introduction to JSON Web Tokens [Електронний ресурс] // Режим доступу: https://jwt.io/introduction
18. Flutter architectural overview [Електронний ресурс] // Режим доступу: https://docs.flutter.dev/resources/architectural-overview#:~:text=During%20development%2C%20Flutter%20apps%20run,JavaScript%20if%20targeting%20the%20web

ДОДАТОК А

SQL-запити для створення таблиць бази даних

```
CREATE TYPE STUDENT_STATUS as ENUM ('ACTIVE', 'ARCHIVED');
CREATE TYPE LESSON_STATUS as ENUM ('PLANNED', 'COMPLETED',
'CANCELED');
CREATE TYPE LESSON_TYPE as ENUM ('OFFLINE', 'ONLINE');
CREATE TYPE HOMEWORK_STATUS as ENUM ('NOT SET','NOT DONE','DONE');
CREATE TYPE PAYMENT_STATUS as ENUM ('NOT PAID', 'PAID');

CREATE TABLE User(
    id SERIAL NOT NULL PRIMARY KEY,
    username VARCHAR NOT NULL UNIQUE,
    email VARCHAR NOT NULL UNIQUE CHECK email LIKE '%@%',
    password VARCHAR NOT NULL );

CREATE TABLE Student (
    id SERIAL NOT NULL PRIMARY KEY,
    status STUDENT_STATUS DEFAULT 'ACTIVE',
    name VARCHAR NOT NULL,
    class VARCHAR,
    phone_number VARCHAR,
    address VARCHAR,
    base_price NUMERIC (9,2) NOT NULL CHECK base_price>0,
    notes VARCHAR,
    user_id INT NOT NULL REFERENCES User );

CREATE TABLE Lesson (
    id SERIAL NOT NULL PRIMARY KEY,
    status LESSON_STATUS DEFAULT 'PLANNED',
    date DATE NOT NULL,
    start_time TIME NOT NULL,
    duration INTERVAL NOT NULL,
    lesson_type LESSON_TYPE DEFAULT 'OFFLINE',
    actual_price NUMERIC (9,2) NOT NULL CHECK actual_price>0,
    homework_status HOMEWORK_STATUS DEFAULT 'NOT SET',
    payment PAYMENT_STATUS DEFAULT 'NOT PAID'
    student_id INT NOT NULL REFERENCES Student,
    subject_id INT NOT NULL REFERENCES Subject );

CREATE TABLE Subject (
    id SERIAL NOT NULL PRIMARY KEY,
    subject_name UNIQUE VARCHAR NOT NULL );
```

ДОДАТОК Б
Довідка про впровадження

Подписано цифровой
подписью: Olga Roznovets
Дата: 2022.06.20 10:12:58 +03'00'

ДОВІДКА

про впровадження

Кросплатформний застосунок для організації роботи репетиторів, розроблений студентом Одеського національного університету імені І.І. Мечникова Кравченком Кирилом Дмитровичем під час виконання дипломної роботи бакалавра на кафедрі математичного забезпечення комп'ютерних систем, прийнятий до використання в творчій студії «Art&life».

ФОП Гончаренко Юлія Дмитрівна,
Ідентифікаційний код юридичної особи 3446001489

(підпис)