НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ В.О. РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
“___” червня 2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

по спеціальності **123 «Комп'ютерна інженерія»**

на тему: Комп’ютерна система управління розкладом дистанційного навчання

Виконав : студент IV курсу, групи КВ-71
Янечко Андрій Сергійович
(підпис)

Керівник доц. к.т.н. Павловський В.І.
(підпис)

Консультант з нормоконтролю, доц.каф. СП і СКС, к.т.н. Клятченко Я.М.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент проф. каф. ОТ, д.т.н., проф. Сімоненко В.П. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.О. РОМАНКЕВИЧ
(підпис)
«___» червня 2021 р.

ЗАВДАННЯ
на дипломний проєкт студента
Янечко Андрія Сергійовича

1. Тема проєкту «Комп'ютерна система управління розкладом дистанційного навчання»,
керівник проєкту доц. к.т.н. Павловський В. І.,
затверджені наказом по університету від «_____»_____2021р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
- Аналіз існуючих засобів порівняння;
- Проектування системи;
- Реалізація системи;
- Шляхи поліпшення;
- Висновок.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

- Алгоритм оброблення запитів мережевого додатку від користувача. Схема алгоритму;
- Схема бази даних. Схема структурна;
- Схема компонентів системи. Схема структурна;
- Схема переходів між екранами. Схема структурна;
- Презентація за темою роботи.

6. Консультанти розділів проєкту[1]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СП і СКС		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	15.11.2020	
2.	Розроблення та узгодження технічного завдання	30.11.2020	
3.	Аналіз існуючих рішень	10.01.2021	
4.	Підготовка матеріалів першого розділу дипломного проєкту	12.04.2021	
5.	Підготовка матеріалів другого розділу дипломного проєкту	20.04.2021	
6.	Підготовка матеріалів третього розділу дипломного проєкту	27.04.2021	
7.	Підготовка матеріалів четвертого розділу дипломного проєкту	10.05.2021	
8.	Підготовка графічної частини дипломного проєкту	18.05.2021	
9.	Оформлення документації дипломного проєкту	23.05.2021	
10.	Попередній огляд матеріалів диплому	26.05.2021	

Студент _____ Андрій ЯНЕЧКО
(підпис)

Керівник проєкту _____ Володимир ПАВЛОВСЬКИЙ
(підпис)

[1] Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (64 с., 30 рис., - додатки).

КОМП'ЮТЕРНА СИСТЕМА УПРАВЛІННЯ РОЗКЛАДОМ ДИСТАНЦІЙНОГО НАВЧАННЯ, KOTLIN, KTOR, JETPACK COMPOSE, FIREBASE MESSAGING, KOTLINX SERIALIZATION, JSON, MYSQL, HIBERNATE, GOOGLE DRIVE, ANDROID.

Об'єкт розробки – комп'ютерна система управління розкладом навчання.

Мета розробки - комп'ютерна система, що дозволяє динамічно змінювати розклад занять, миттєво сповіщати користувачів про зміни в навчальному модулі, групі чи конкретному занятті та завантажувати файли будь-якого типу і прив'язувати їх до заняття.

В ході розробки:

- сформульовані основні вимоги до системи;
- розроблена гнучка архітектура системи;
- розроблена структура бази даних системи;
- розроблено мережевий додаток;
- розроблено мобільний додаток під платформу Android.

Основні характеристики системи:

- можливість створювати навчальні модулі, академічні групи та заняття;
- можливість додавати та видаляти користувачів з академічних груп;
- можливість завантажувати файли та прив'язувати їх до заняття;
- можливість сповіщення користувачів про зміні в занятті, групі чи навчальному модулі.

В процесі розробки використані такі технології: мова програмування Kotlin, асинхронний фреймворк для створення мікросервісів та веб-додатків KTOR, СУБД MySQL, декларативний UI фреймворк під платформу Android Jetpack Compose, сервіс пуш повідомлень Firebase Messaging, хмарне сховище файлів Google Drive.

Поліпшення системи можливо за допомогою додавання нового функціоналу та покращення дизайну додатку.

ABSTRACT

Qualification work includes an explanation note (64 p., 30 pictures, - appendices).

COMPUTER DISTANCE LEARNING SCHEDULE MANAGEMENT SYSTEM, KOTLIN, KTOR, JETPACK COMPOSE, FIREBASE MESSAGING, KOTLINX SERIALIZATION, JSON, MYSQL, HIBERNATE, GOOGLE DRIVE, ANDROID.

The object of development is computer distance learning schedule management system.

The purpose of the development is a computer system that allows you to dynamically change the schedule of classes, which provides the ability to instantly notify users of changes in the training module, group or specific lesson also system supports downloading files of any type and linking them to the lessons.

During development:

- the basic requirements have been formulated;
- a flexible system architecture has been developed;
- a structure of the database has been developed;
- a network application has been developed
- a mobile application for the Android platform has been developed.

Main characteristics of the system:

- ability to create training modules, academic groups and classes;
- ability to add and remove users from academic groups;
- ability to download files and link them to the lesson;
- ability to notify users of changes in the lesson, group or training module.

During development such technologies have been used: Kotlin programming language, KTOR asynchronous framework for creating microservices and web applications, MySQL database, Jetpack Compose declarative UI toolkit, Firebase Messaging push notification service, Google Drive as cloud file storage.

System improvement could be achieved by adding new functionality to the system and redesign user interface of mobile application.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.002 ТЗ	Комп’ютерна система	4		
			управління розкладом			
			дистанційного навчання			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Комп’ютерна система	2		
			управління розкладом			
			дистанційного навчання			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Комп’ютерна система	50		
			управління розкладом			
			дистанційного навчання			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Блок-схема алгоритму	1		
			оброблення запитів			
			мережевого додатку від			
			користувача			
			Схема алгоритму			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Янечко А.С..			Комп’ютерна система управління дистанційним навчанням Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Павловський В.І.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-71	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.006 Д2	Схема бази даних.	1		
			Схема структурна			
	А4	ІАЛЦ. 045440.007 Д3	Схема компонентів	1		
			системи			
			Схема структурна.			
	А4	ІАЛЦ. 045440.008 Д4	Схема переходів	1		
			між екранами			
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

					ІАЛЦ.045440.001 ОА	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: «Комп’ютерна система управління розкладом дистанційного навчання».

Галузь застосування: комп’ютерна система управління розкладом дистанційного навчання ЗВО.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою цього проєкту є комп’ютерна система, що дозволяє динамічно змінювати розклад занять, миттєвого сповіщати користувачів про зміни в навчальному модулі, групі чи конкретному занятті та завантажувати файли будь-якого типу і прив’язувати їх до заняття.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- можливість створювати навчальні модулі;
- можливість створювати академічні групи, які прив’язані до навчальних модулів;

- можливість додавати та видаляти користувачів з академічних груп;
- можливість додавати та видаляти користувачів з навчальних модулів;
- можливість створювати заняття та модифікувати їх область видимості для конкретної академічної групи та навчального модуля;
- можливість додавати та модифікувати опис заняття та будь-які файлові матеріали;
- можливість повідомити користувачів про зміни в навчальному модулі, академічний групі чи конкретному занятті;
- зручний перегляд навчальних модулів, занять та академічних груп в додатку;
- відслідковування та повідомлення про зміни в навчальному модулі, занятті чи академічній групі;
- можливість назначати відповідального або академічну групу відповідальних користувачів за групу, модуль чи заняття, тобто щоб редагувати відповідні об’єкти міг не тільки власник;
- відслідковування статусу користувача стосовно навчального модуля, групи чи заняття та повідомлення про це уповноважених користувачів, наприклад новий користувач підписався на модуль, необхідно, щоб інший уповноважений користувач дав доступ до навчального модуля.
- налаштування профіля користувача;
- відображення розкладу для конкретного користувача.

5.2 Вимоги до апаратного забезпечення

- Процесор: Intel Core i5;
- Оперативна пам’ять: 4 Гб;
- Простір на диску: 512 мб.

5.3 Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Linux, MacOS.

- Мобільний телефон з встановленою операційною системою Android.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.01.2021
2.	Розроблення та узгодження технічного завдання	30.01.2021
3.	Аналіз існуючих рішень	10.02.2021
4.	Підготовка матеріалів першого розділу дипломного проекту	11.04.2021
5.	Підготовка матеріалів другого розділу дипломного проекту	15.04.2021
6.	Підготовка матеріалів третього розділу дипломного проекту	23.04.2021
7.	Підготовка матеріалів четвертого розділу дипломного проекту	10.05.2021
8.	Підготовка графічної частини дипломного проекту	20.05.2021
9.	Оформлення документації дипломного проекту	24.05.2021
10.	Попередній огляд матеріалів диплому на кафедрі	25.05.2021

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	ІАЛЦ.045440.000	Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ. 045440.001 ОА	Опис альбому	2	
3	А1	ІАЛЦ. 045440.002 ТЗ	Технічне завдання	1	
4	А1	ІАЛЦ. 045440.003 ТП	Відомість технічного проєкту	1	
5	А1	ІАЛЦ. 045440.004 ПЗ	Пояснювальна записка	50	
6	А1	ІАЛЦ. 045440.005 Д1	Алгоритм оброблення запитів мережевого додатку від користувача. Схема алгоритму	1	
7	А1	ІАЛЦ. 045440.006 Д2	Схема бази даних. Схема структурна	1	
8	А1	ІАЛЦ. 045440.007 Д3	Схема компонентів системи. Схема структурна	1	
9	А1	ІАЛЦ. 045440.008 Д4	Схема переходів між екранами. Схема структурна	1	

				ІАЛЦ.045550.003 ТП		
	ПІБ	Підп.	Дата			
Розробн.	Янечко А.С..			Відомість дипломного проєкту	Лист	Листів
Керівн.	Павловський В.І.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-71	
Н/контр.	Клятченко Я.М.					
Зав.каф.	Романкевич В.О.					

Пояснювальна записка
до дипломного проєкту

на тему: Комп’ютерна система управління розкладом дистанційного навчання

Київ – 2021 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

КС – комп’ютерна система.

УРДН – управління розкладом дистанційного навчання.

СУР – система управління розкладом.

ЗВО – заклад вищої освіти.

УР – управління розкладом.

АПІ (API) - прикладний програмний інтерфейс.

БД – база даних.

СУБД – системи управління базами даних.

ООП – об’єктно-орієнтоване програмування.

HTTP (Hyper Text Transfer Protocol) - протокол прикладного рівня передачі гіпертекстової інформації.

JRE (Java Runtime Environment) – середовище виконання Java.

JDK (Java Development Kit) – комплект розроблення додатків мовою програмування Java.

ORM (Object Relation Mapping) – технологія, що перетворює реляційну модель в класи мови об’єктно-орієнтованої мови програмування.

JPA (Java Persistence API) – стандартизований інтерфейс для Java RM фреймворків.

UI (User Interface) – інтерфейс користувача.

UX (User Experience) – зручність використання інтерфейсу користувача.

DSL (Domain Specific Language) – спеціальна мова, яка вирішує конкретну задачу.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript.

GraalVM – віртуальна машина Java та JDK.

LVVM (Low Level Virtual Machine) – універсальна система аналізу, трансформації і оптимізації програм.

IDE (Integrated development environment) – комплекс програних засобів, які

використовуються програмістами для розроблення програмного забезпечення.

MVP (Minimum viable product) – продукт, тільки з мінімально необхідним функціоналом.

ВСТУП

Наразі ми живимо в еру технологій, тому все більше і більше систем переходять у онлайн використання, адже постає вимога в миттєвому доступі до інформації та її редагуванню.

Зважаючи на ситуацію, що склалася з Covid-19 багато систем змушені створити онлайн аналоги задля зручності використання та адаптації до умов пандемії, зокрема це стосується і ЗВО, адже зараз освітній процес відбувається виключно дистанційно і постає необхідність створення системи яка дозволяє централізований та мобільний доступ до інформації, комунікацію між студентами та викладачами та зручне керування розкладом.

В ході виконання дипломного проєкту була розроблена комп'ютерна система управління розкладом дистанційного навчання, що дозволяє динамічно змінювати розклад занять, миттєво сповіщати користувачів про зміни в навчальному модулі, групі чи конкретному занятті та завантажувати файли будь-якого типу і прив'язувати їх до заняття.

1. АНАЛІЗ КОМП'ЮТЕРНИХ СИСТЕМ УПРАВЛІННЯ РОЗКЛАДОМ ДИСТАНЦІЙНОГО НАВЧАННЯ

1.1. Галузь застосування КС УРДН

Навчальний процес, як і більшість бізнес-процесів, має складну організацію та оперативне управління. Зокрема він передбачає початкове планування та оперативне коригування розкладу навчання в зв'язку зі зміною обставин.

Можна виділити декілька типів систем управління розкладом (СУР):

СУР традиційні для закладів вищої освіти (ЗВО), коли розклад складається на семестр і на протязі цього часу практично не змінюється. Вона має обмежений функціонал – статичний розклад, відсутність корегування вже існуючих записів, або відсутність сповіщення про їх зміну, також важливим фактором є цілісність системи, тобто, щоб система була єдиним джерелом інформації освітнього процесу та комунікації з викладачами.

СУР для різних тренінгів та онлайн-курсів, коли розклад може корегуватися в залежності від різних обставин чи подій, зокрема виконання означеною більшістю слухачів поставлених завдань.

На сучасному етапі для вирішення цих задач створюються комп'ютерні системи управління розкладом (КС УР)

Ситуація, що склалася з Covid-19 призвела до того, що більшість ЗВО мають переглядати свої КС УР з метою адаптації управління начальним процесом до умов пандемії, коли навчання переведено в дистанційний режим і вимагає більш детального управління взаємодією викладачів та студентів. Більша гнучкість КС УР дасть змогу зручно планувати розклад занять, підтримувати зв'язок студентів з викладачами, слугувати єдиним джерелом інформації щодо навчального процесу та всього, що з ним пов'язано, а також вчасно інформувати всіх студентів про зміни в ході навчання або інші необхідні новини.

1.2. Відомі КС УРДН та їх недоліки

Серед відомих комп'ютерних систем управління розкладом дистанційного навчання можна зазначити http://rozklad.kpi.ua/.

Можливості системи – інформація про розклад групи, розклад викладача, інформація про аудиторію, в якій буде проводитися заняття.

Недоліки системи:

- неможливість налаштування для конкретного користувача або групи користувачів, наприклад додати у розклад консультацію для певної підгрупи користувачів;
- неможливість можливості додавання додаткових матеріалів заняття, опису;
- відсутня інтеграція з засобами для проведення онлайн занять;
- відсутність контролю від автора навчального модуля;
- відсутність сповіщень про зміни розкладу в навчальному модулі, групі чи занятті.

Частково додатковим прикладом КС УРДН може слугувати «Електронний кампус». Система надає можливості для перегляду поточних оцінок, присутність студентів на заняттях, атестацію, та оголошень.

Недоліки системи:

- відсутня інтеграції з розкладом;
- відсутня сповіщення про події;
- відсутнє спілкування між викладачами та студентами.

1.3. Постановка задачі на розробку КС УРДН

З метою адаптації управління начальним процесом до умов пандемії, коли навчання переведено в дистанційний режим і вимагає більш детального управління взаємодією викладачів та студентів постає необхідність у розробці комп'ютерної системи яка б надавала змогу динамічно керувати розкладом дистанційного навчання, а також:

- повідомляти студентів про початок занять або про його зміни;

- повідомляти студентів про додаткові консультації;
- оперативної комунікації між студентами та викладачами;
- можливість завантажувати будь-які матеріали.

2. ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ РОЗКЛАДОМ ДИСТАНЦІЙНОГО НАВЧАННЯ

2.1. Основні вимоги до КС УРДН

До основних вимог КС входять:

- можливість створювати навчальні модулі;
- можливість створювати академічні групи, які прив'язані до навчальних модулів;
- можливість додавати та видаляти користувачів з академічних груп;
- можливість додавати та видаляти користувачів з навчальних модулів;
- можливість створювати заняття та модифікувати їх область видимості для конкретної академічної групи та навчального модуля;
- можливість додавати та модифікувати опис заняття та будь-які файлові матеріали;
- можливість повідомити користувачів про зміни в навчальному модулі, академічний групі чи конкретному занятті;
- зручний перегляд навчальних модулів, занять та академічних груп в додатку;
- відслідковування та повідомлення про зміни в навчальному модулі, занятті чи академічній групі;
- можливість назначати відповідального або академічну групу відповідальних користувачів за групу, модуль чи заняття, тобто щоб редагувати відповідні об'єкти міг не тільки власник;
- відслідковування статусу користувача стосовно навчального модуля, групи чи заняття та повідомлення про це уповноважених користувачів, наприклад новий користувач підписався на модуль, необхідно, щоб інший уповноважений користувач дав доступ до навчального модуля.
- налаштування профіля користувача;
- відображення розкладу для конкретного користувача.

2.2. Проєктування архітектури КС УРДН

Відповідно до поставлених вимог система має забезпечити такі можливості як:

1. Інтеграцію даних про учасників навчального процесу, поточний розклад занять;
2. Динамічне керування розкладом;
3. Підтримку файлів довільного типу з різноманітними матеріалами, що використовуються в начальному процесі;
4. Швидке сповіщення про зміни в розкладі,
5. Взаємодію з мобільними пристроями учасників навчального процесу.

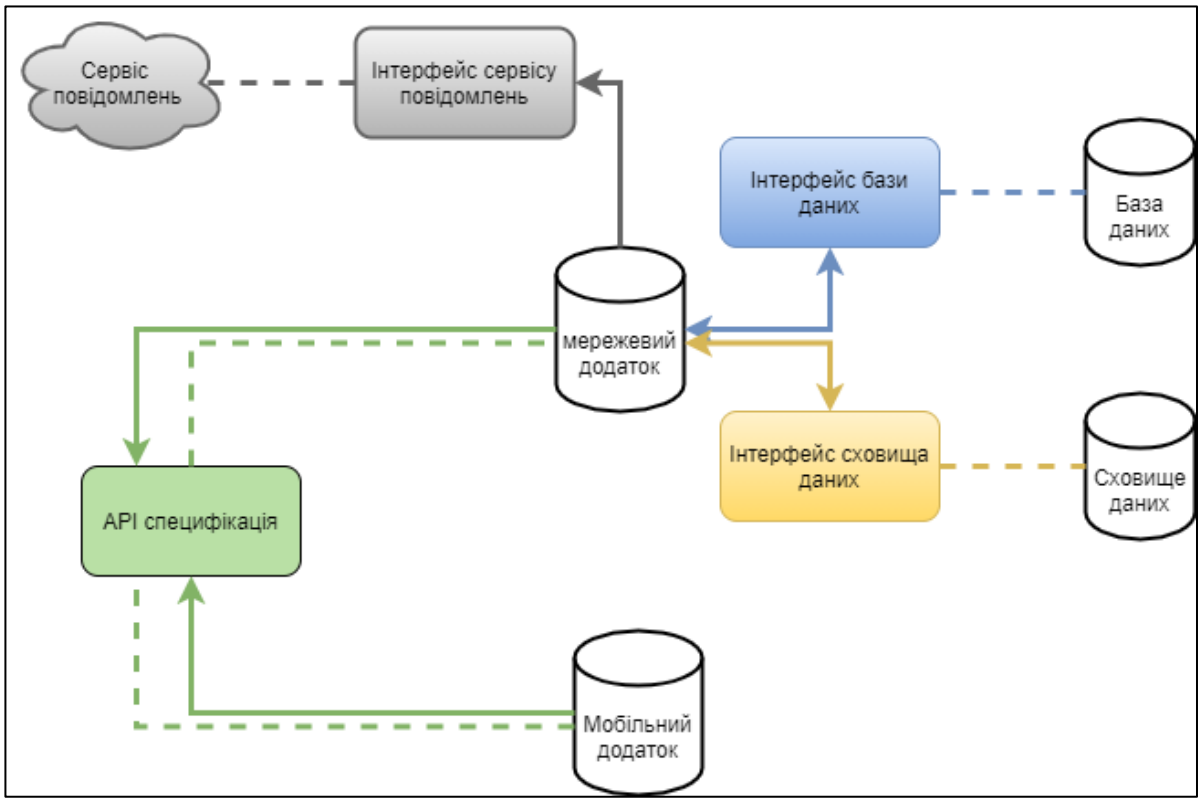


Рисунок 2.1 - Архітектура системи КС УРДН

Архітектура КС УРДН (Рисунок 2.1) складається з дев’яти основних компонентів:

1. АПІ специфікація;
2. Android додаток;
3. Мережевий додаток;
4. Інтерфейс бази даних;

5. Віддалена база даних;
6. Інтерфейс сховища файлів;
7. Віддалене сховище файлів;
8. Інтерфейс сервісу повідомлень;
9. Сервіс повідомлень.

API специфікація представляє собою контракт між сервером та клієнтом і використовується для успішної комунікації та передачі даних. Будь-яка зміна в інтерфейсі АПІ призводить до змін на серверній та на клієнтській компонентах, тому зміни повинні бути узгоджені та строго дотримані з обох сторін, на рисунку 1, цей компонент позначений як «API специфікація» і зеленими стрілками показано, що мобільний додаток надсилає запит, який визначений в специфікації та очікує на відповідь, яка в свою чергу має теж відповідати специфікації.

Мережевий додаток працює поверх HTTP сервера та приймає запити, які описані в API, обробляє їх та повертає відповідь в узгодженому форматі. Цей компонент системи є одним з головних, оскільки саме мережевий додаток містить всю логіку системи та комунікація з зовнішнім світом, наприклад базою даних або віддаленим сховищем файлів за допомогою відповідних інтерфейсів (контрактів).

Інтерфейс бази даних необхідний для того, щоб узгодити моделі, які будуть зберігатися в базі даних та оголошує необхідні методи для вибірки, пошуку та інших операцій, пов’язаних з базою даних.

Віддалена база даних слугує сховищем усіх даних системи, цей компонент системи підтримує інтерфейс бази даних.

Інтерфейс сховища файлів необхідний для того, щоб підтримувати завантаження та передачу даних до віддаленого сховища файлів, оголошує усі необхідні моделі та методи для операцій над файлами.

Віддалене сховище файлів слугує сховищем файлів будь-якого типу. Цей компонент системи підтримує завантаження, зчитування та перегляд файлів будь-якого типу та довільного розміру, надійність та безпека гарантується

стороннім сервісом, також цей компонент підтримує інтерфейс сховища файлів.

Android додаток представляє собою клієнтську частину, головна задача якого відобразити інформацію з серверу або передати її на сервер та показати результат користувачу. На відміну від мережевого додатку, цей компонент системи не має прямого доступу до бази даних або віддаленого сховища файлів, а вся комунікація з системою повинна йти через мережевий додаток і відбуватися за допомогою попередньо узгодженого API.

Інтерфейс сервісу повідомлень описує моделі та оголошує методи, які використовує мережевий додаток для передачі повідомлень в режимі реального часу.

Сервіс повідомлень – це компонент, який підтримує інтерфейс сервісу повідомлень та містить конкретну реалізацію.

Система має прив’язку тільки до інтерфейсів, тому таку систему легко розширювати, а також легко замінити будь-який компонент на інший, головне, щоб новий компонент підтримував потрібний інтерфейс.

2.3. Проєктування бази даних

База даних це один із найголовніших компонентів системи, вона зберігає у собі усі дані, які використовує комп’ютерна система управління дистанційного навчання.

Виходячи з вимог база даних повинна мати такі основні таблиці:

- таблиця користувачів (users);
- таблиця навчальних модулів (course);
- таблиця груп (course_group);
- таблиця завдання (lesson);
- таблиця зв’язку між користувачами (users_to_users_subscription);
- таблиця зв’язку між користувачами та групами (user_to_group);
- таблиця зв’язку між користувачами та навчальними модулями (user_to_course);
- таблиця зв’язку між групою та заняттям (group_to_lesson);

– таблиця зв’язку між ідентифікаторами файлів та заняттям (file_to_lesson_content).

Всього в базі даних зберігається дев’ять таблиць, їх вміст та зв’язки відображені на рисунку 2.2.

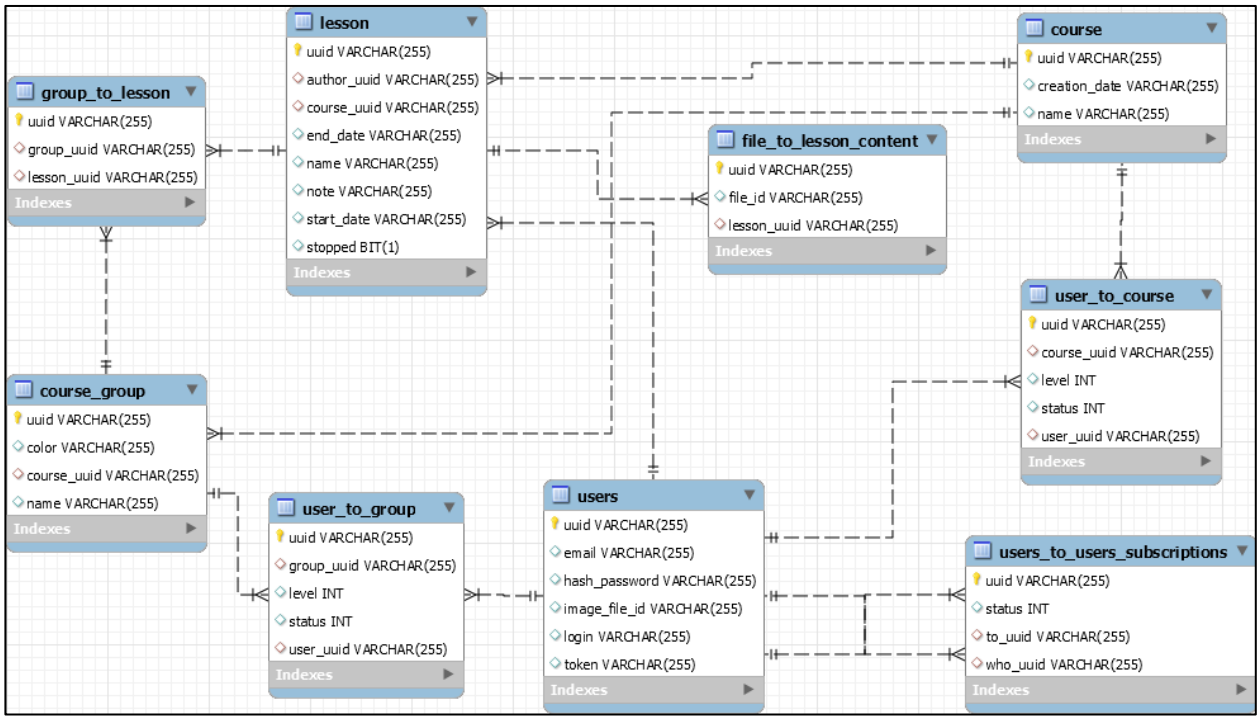


Рисунок 2.2 - Загальна схема бази даних

2.4. Проєктування інтерфейсів користувача

Інтерфейс користувача безпосередньо впливає на UI/UX, тому проектуванню інтерфейсу користувача необхідно приділяти окрему увагу.

Виходячи з вимог, поставлених до додатку на платформі Android інтерфейс повинен мати наступні компоненти, взаємодію між якими детальніше показано на рисунку 2.3:

– екран авторизації;
– екран реєстрації;
– головний екран з відображенням найважливішої інформації;
– меню додатку;
– екран зі списком навчальних модулів;
– екран зі списком користувачів системи;
– екран, шо містить детальну інформацію про користувача;

- екран, що містить детальну інформацію про начальний модуль;
- екран створення навчальних модулів;
- екран створення груп навчальних модулів;
- екран створення занять.

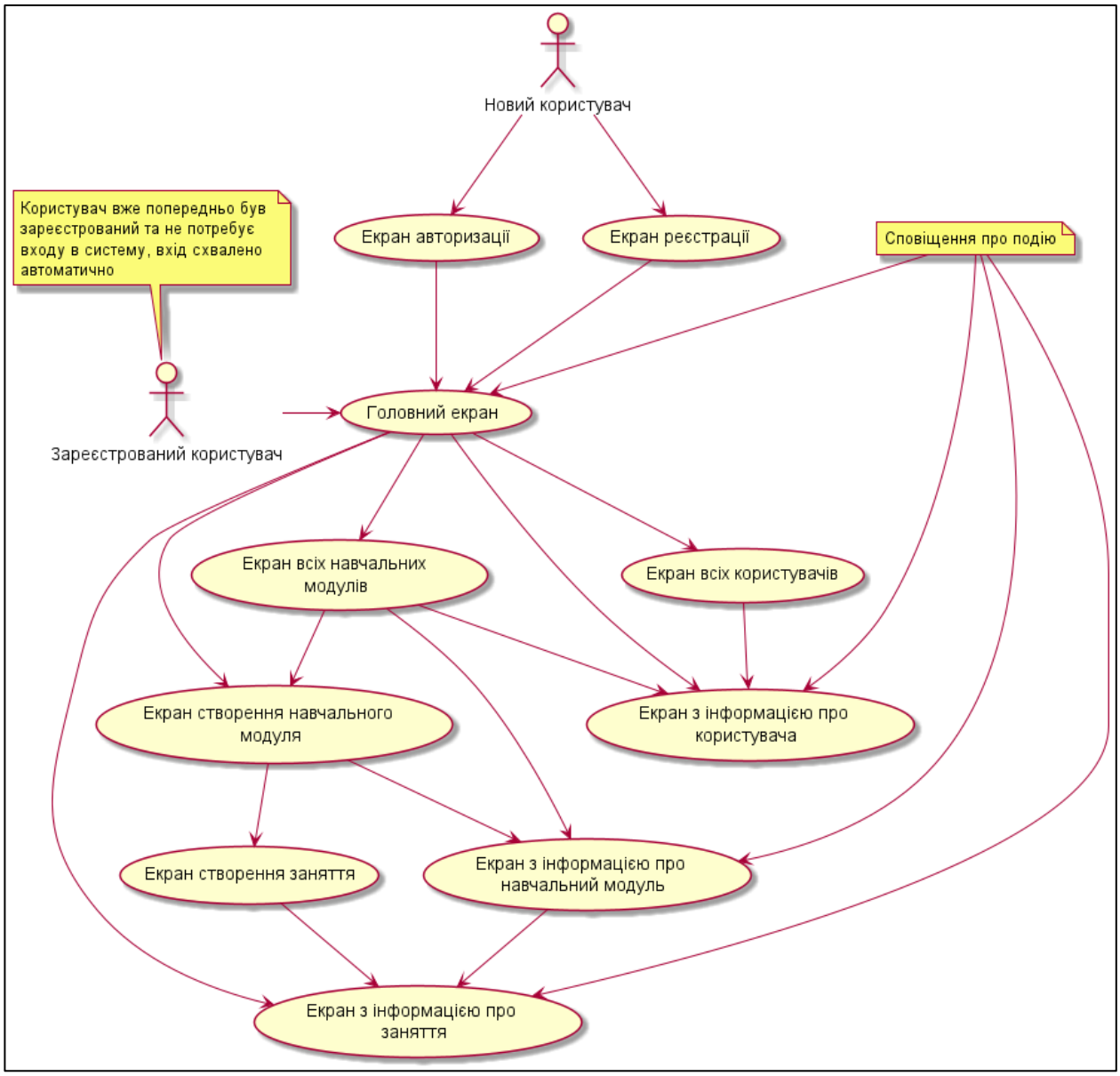


Рисунок 2.3 – Зв’язок між екранами

2.5. Проектування структури мережевого додатку

Мережевий додаток грає ключову роль в системі, оскільки саме в цьому компоненті міститься вся критична логіка системи. Також мережевий додаток має прямий доступ до віддаленої бази даних, сервісу пуш повідомлень та віддаленого сховища файлів, тобто він поєднує в собі усі компоненти, схема

комунікації зображена на рисунку 2.4.

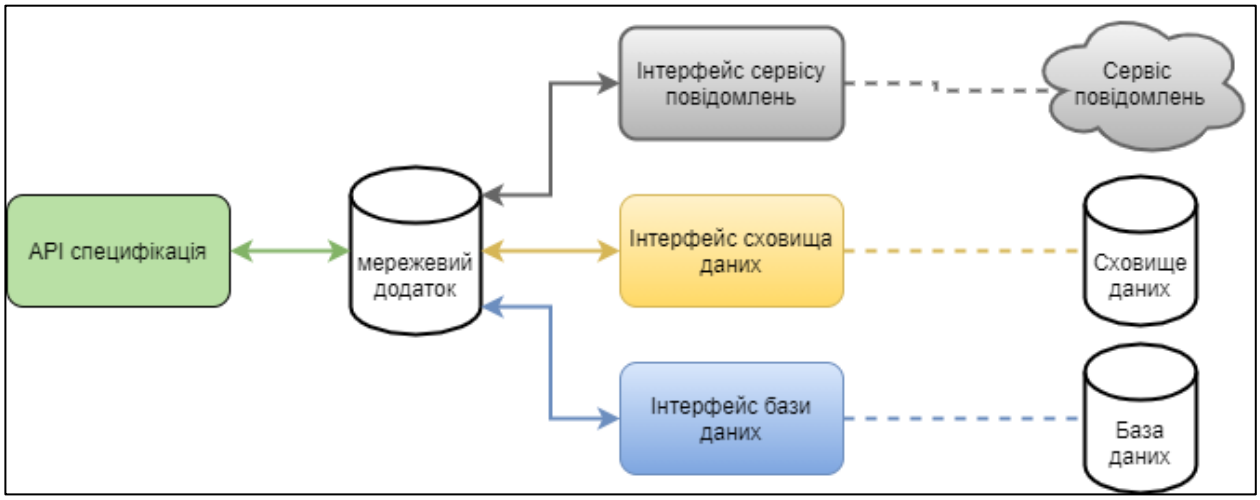


Рисунок 2.4 - Схема комунікації мережевого додатку

Проектуванню цього критичного компоненту слід приділяти особливу уваги, адже саме цей компонент регулює пропускну спроможність системи та її відмовостійкість.

Основні вимоги до структури мережевого додатку:

- відмовостійкість;
- легка та швидка заміна основних та периферійних компонентів системи;
- зручне масштабування.

Отже, для того, щоб задовольнити основні вимоги необхідно розділити реалізацію та інтерфейс, таке рішення призведе одразу до декількох позитивних наслідків:

- покращить тестування, адже все буде зав’язано не на конкретну реалізацію, а на інтерфейси і при тестуванні окремого компонента, можна просто передати на вхід тестову реалізацію з потрібною логікою;
- також залежність від інтерфейсів дозволяє легко змінити компоненти в разі необхідності, головне щоб новий компонент підтримував необхідний інтерфейс.

Після виділення необхідних інтерфейсів в окремі логічні модулі маємо наступну структуру мережевого додатку, яка зображена на рисунку 2.5.

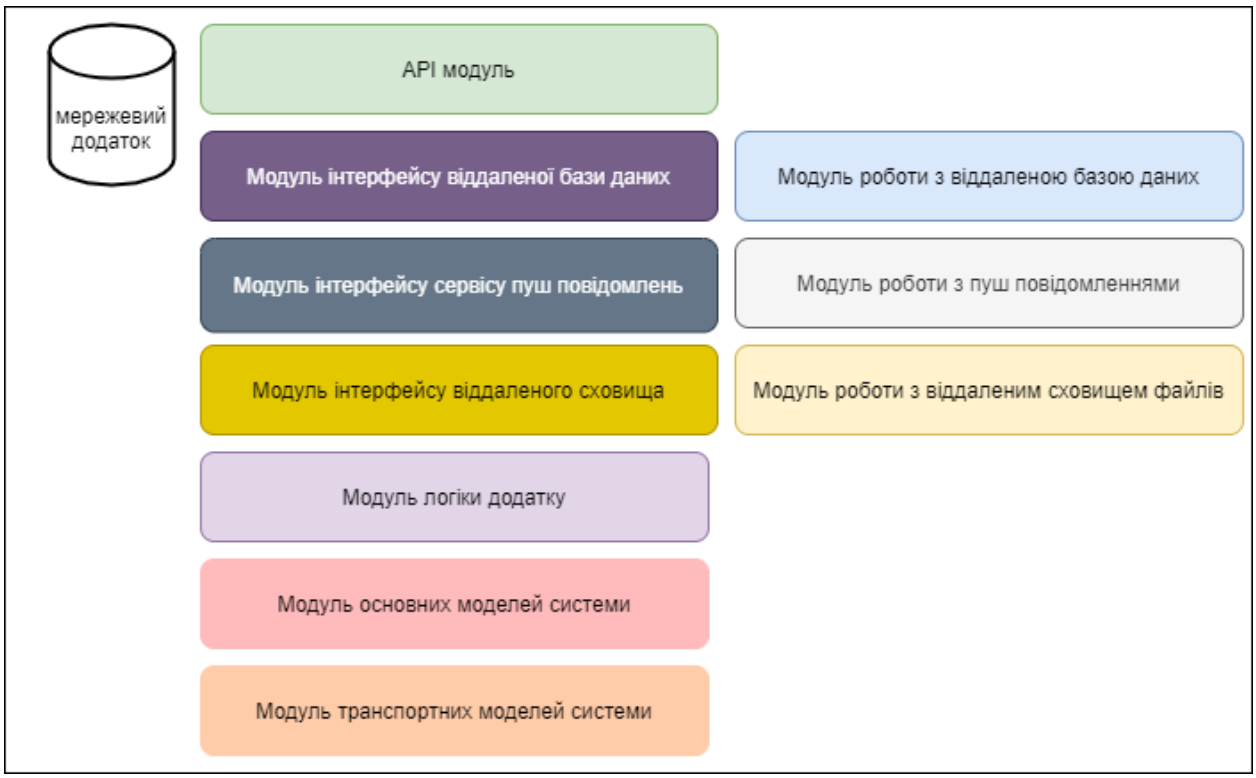


Рисунок 2.5 - Структура мережевого додатка

Отже мережевий додаток поділений на десять окремих логічних модулів, детальніше про кожний йтиметься у третьому розділі, а саме:

1. API модуль;
2. Модуль інтерфейсу віддаленої бази даних;
3. Модуль інтерфейсу сервісу пуш повідомлень;
4. Модуль інтерфейсу віддаленого сховища;
5. Модуль логіки додатку;
6. Модуль основних моделей системи;
7. Модуль транспортних моделей системи.

2.6. Проектування структури мобільного додатку

Структура мобільного додатку теж складається з декількох логічних модулів, це пов’язано з тим, що кожен модуль відповідає конкретно за свою задачу і підтримувати та тестувати такий код стає набагато легше ніж коли він був би одномодульним.

Існує декілька підходів щодо розбивання додатку на логічні модулі:

- розбиття за логічними компонентами;

- розбиття за функціональними компонентами.

Під розбиттям за функціональними компонентами мається на увазі, що окремий модуль створюється під кожну функціональність, а якщо є якась частина функціоналу, яка використовується в більше ніж одному функціональному компоненті - вона виноситься у свій окремий модуль. Плюси такої структуру додатку, в тому що вона дозволяє дуже чітко розділити границі між окремим функціональними компонентами, через що повторне використання та тестування компонентів стає дуже зручним та простим, також слід зазначити, що найголовніший плюс такої структури це можливість працювати над одним додатком великої кількості людей, адже у кожного функціонального компонента свій окремий модуль і зв’язок між компонентами відбувається завдяки попередньо визначеному інтерфейсу. Серед мінусів такого підходу це велика кількість модулів, що призводить до великої кількості коду, тобто додатки, які побудовані на основі такої структури, як правило, мають більшу кодову базу ніж додатки, які побудовані за іншої структурою.

Під розбиттям за логічними компонентами мається на увазі, що під окремий логічний компонент, наприклад: локальна база даних, клієнт зв’язку з сервером, менеджер локальних файлів, інтерфейс користувача, логіка додатку, - створюється окремий модуль. Плюси такої структури в тому, що вона дає чітку різницю між логічними компонентами, що робить код більш підтримуваним та підлеглим для тестування.

Оскільки над додатком працює лише одна людина, то доцільно вибрати розбиттям за логічними компонентами, адже вона вимагає менше коду, тобто менше часу на реалізацію.

Отже, структура мобільного додатку складається з п’яти окремих логічних модулів, яка зображена на рисунку 2.6:

1. Модуль інтерфейсу користувача;
2. Модуль логіки додатку;
3. Модуль локального сховища даних;

4. Модуль доступу до файлів;

5. Модуль роботи з мережевим додатком.

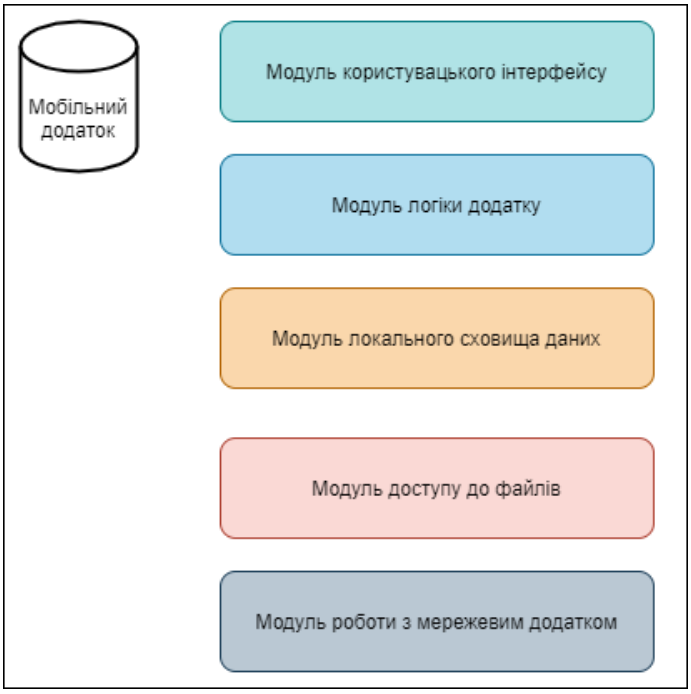


Рисунок 2.6 - Структура мобільного додатку

2.7. Вибір інструментів та технологій реалізації

2.7.1. Вибір мови програмування

Розглядаючи питання мови програмування вибір випав на мову програмування Kotlin.

Kotlin – статично типізована мова програмування, яка в більшості випадків працює поверх JVM та має повну інтеграцію з мовою програмування Java, але на відміну від інших JVM мов програмування Kotlin також компілюється в JavaScript та за допомогою LLVM компілятора може компілюватися навіть в код машинних інструкцій [9].

При розробці Kotlin розробники ставили собі за мету створити більш типо-безпечнішу мову ніж Java та легшу у використанні ніж Scala. Розробка мови почалась в 2010 році, а сирцевий код мови програмування був відкритий 2012 року. Перша версія мови вийшла в 2016 році, наразі актуальна версія мови програмування Kotlin – 1.5.0.

Починаючи з 2017 року Kotlin отримав офіційну підтримку для створення

застосунків під платформу Android, а 2019 року компанія Google визнала мову Kotlin основною мовою програмування додатків під платформу Android.

Отже одразу декілька фактів роблять мову програмування Kotlin ідеальною для проєкта:

- офіційна мова програмування застосунків під платформу Android;
- має повну інтеграцію з мовою програмування Java, а отже є можливість використовувати всі бібліотеки, фреймворки та популярні і перевірені рішення, які раніше були написані для Java;
- єдина кодова база для мережевого додатку, мобільного додатку та веб сторінки, оскільки Kotlin працює під JVM та має можливість компіляції у JavaScript;
- зручний та інтуїтивно зрозумілий синтаксис;
- зручний механізм Kotlin DSL, що дозволяє описати проблему чи якусь конкретну ситуацію за допомогою створення спеціальної мови, інколи це буває зручно, яскравим прикладом використання цього підходу – фреймворк KTOR, детальніше про який буде розказано нижче.
- має механізм під назвою extensions, доповнення, який дозволяє зручно додавати методи до класів, які закриті до модифікації, наприклад до бібліотечних класів;
- має підтримку як і ООП так і функціонального стилю програмування;
- лямбди є частиною мови;
- має підтримку функцій вищого порядку, це функція, яка приймає в якості параметра іншу функцію;
- має підтримку співпрограм (співпроцедур), яка реалізована окремою бібліотекою, але в мові присутнє ключове слово suspend, яке означає, що функція повинна бути викликана тільки з контексту співпрограми (співпроцедури) [4].

2.7.2. Вибір системи управління базою даних

База даних – найважливіший компонент системи, адже саме в цьому компоненті зберігаються усі дані системи тому потрібно приділяти особливу увагу обранню системи управління базами даних.

В проекті використовується система керування базою даних - MySQL 8.0, яка має відкритий код і підтримується корпорацією Oracle. MySQL ідеальне рішення для малих та середніх проектів, найчастіше її використовують в якості віддаленого сервера з базою даних. MySQL дуже гнучка у використанні, адже в ній є підтримка таблиць типу MyISAM, повнотекстовий пошук та таблиці типу InnoDB, які підтримують транзакції на рівні окремих записів. Оскільки це одна з найпопулярніших систем курування базою даних, то існує багато створених рішень на багатьох мовах програмування, які вирішують задачі доступу та модифікації даних в MySQL [7].

Отже, вибираючи MySQL як систему для управління базами даних, отримуємо наступні переваги:

- швидку та гнучку систему управління базами даних;
- готове рішення на мові програмування Java для під’єднання до MySQL;
- велику кількість бібліотек для спрощення операцій над даними;
- велику кількість бібліотек для представлення таблиць у вигляді класів вибраної мови програмування.

Для доступу до MySQL було використано бібліотеку Hibernate, яка має повноцінну підтримку MySQL.

Hibernate надає каркас для відображення між об’єктно-орієнтованою моделлю даних і реляційною базою даних. Одна з найпопулярніших реалізацій JPA.

2.7.3. Вибір віддаленого сховища файлів

Основними вимогами до цього компоненту системи є:

- наявність авторизації та захист від несанкціонованого доступу до файлів;
- наявність відкритого API для доступу до файлів;
- великий об’єм пам’яті;
- відмовостійкість.

Google Drive поєднує у собі усі необхідні вимоги.

Авторизація відбувається за допомогою попередньо згенерованого ключа доступу, що унеможливлює несанкціонований доступ до файлів, також Google Drive має відкриту API специфікацію також Google забезпечує бібліотеку для комунікації з сервісом Google Drive, яка має реалізацію на різних мовах програмування, зокрема на Java, яка і використовувалася в проекті.

2.7.4. Вибір інструментів створення мережевого додатку

Оскільки мережевий додаток це серце системи, тому цей компонент системи має бути вкрай надійним та мати вискоку пропускну спроможність, також не останню роль грає масштабування системи.

Отже, фреймворк мережевого додатку повинен забезпечувати:

- вільну архітектуру додатку, не прив’язану до фреймворку;
- асинхронну обробку запитів;
- високу швидкодію.

Серед фреймворків мережевих додатків слід відокремити декілька популярний: Spring Framework, Quarkus, KTOR.

1. Spring Framework – один з найпопулярніших фреймворків для створення мережевих додатків під платформу JVM. Має велику кількість рішень стосовно авторизації, захисту інформації, MVC архітектура для REST додатків, та інші [1];
2. Quarkus – популярний фреймворк для створення мережевих додатків, оптимізований під GraalVM, конкурент фреймворку Spring, серед переваг – використовує значно менше пам’яті, але при цьому не має стільки ж

готових рішень та бібліотек скільки має Spring [2];

KTOR – асинхронний фреймворк для створення мережевих додатків, написаний на Kotlin, тому при використанні цього фреймворку є можливість використовувати всю зручність синтаксису Kotlin [3]. KTOR досить простий у використанні фреймворк, має зручний механізм для вбудовування різних плагінів (таких як авторизація чи логування). На відміну від Spring Framework, KTOR не змушує використовувати якусь конкретну структуризацію додатку також він побудований за допомогою Kotlin співпрограм, що забезпечує ефективне використання потоків під час асинхронної роботи, має вбудований механізм під назвою Routing - це механізм фреймворку KTOR, який порівнює запит зі списком доступних запитів, які об’явлені в методі налаштування мережевого додатку, і в разі співпадіння визначає яким чином повинен оброблятися запит: POST, GET, PUT, PATCH чи DELETE та який метод викликати для його обробки, якщо ж співпадіння не знайдено, то буде повернута помилка з кодом 404 Not found. Основним об’єктом запиту є інтерфейс ApplicationCall саме він передається в якості параметра методу який обробляє запит після успішного визначення методу обробки. Використовуючи цей інтерфейс можна отримати доступ до усіх параметрів, хедерів запиту, до контенту запиту, в разі, якщо це файл або інші дані, а також відправити повідомлення будь-якого типу назад користувачеві.

Зважаючи на всі переваги та недоліки було вибрано фреймворк KTOR.

Для роботи з файлами типу json використовується бібліотека для мови програмування Kotlin – Kotlinx Serialization [5].

Для тестування основних класів використовувався фреймворк Junit, для тестування класів пов’язаних з базою даних використовувалась база даних в оперативній пам’яті – H2.

2.7.5. Вибір інструментів створення мобільного додатку на платформі Android

Найголовнішими вимогами до додатку під платформу Android є:

- висока продуктивність;
- швидке реагування на дії користувача;

Теперішній Android UI фреймворк був створений ще в 2009 року, і він не був розрахований на сучасні телефони, тоді перед розробниками стояла мета зробити фреймворк для відображення на пристроях які мають досить обмежені ресурси та досить малу роздільну здатність екрану, тому на сьогодні компанія Google створює новий декларативний UI фреймворк – Jetpack Compose, який прийде на заміну старому. Фреймворк зараз знаходиться на відкритій бета стадії і розробники спонукають до його використання за для того, щоб зробити його якомога зручніше для використання.

Основні переваги Jetpack Compose:

- написаний у функціональному стилі;
- декларативний, використовує мову Kotlin та власний плагін компілятора, що дозволяє писати логіку та UI однією мовою та за допомогою одних і тих самих конструкцій;
- оптимально використовує доступні ресурси пристрою.

Jetpack Compose складається з двох основних компонентів:

1. Compose;
2. UI toolkit (не офіційна назва Crane).

Compose це проект, який дозволяє ефективне управління деревами довільного типу. UI toolkit, який має не офіційну назву Crane, це реалізація UI на основі Compose, де компонентами дерева є UI елементи, які в коді мають вигляд функцій, помічених Composable анотацією. Саме Compose дає змогу ефективно використовувати можливості пристрою, адже UI функції можуть викликатися з різних потоків та оновлюватися лише частково, що економить ресурси телефона. Поєднання Compose та Crane отримало назву Jetpack

Compose.

Наразі компанія JetBrains, засновники Kotlin, всіляко підтримують проект Jetpack Compose та вже створили порт для Windows, Linux та Android під назвою Compose Desktop, його унікальність в тому, що це версія Jetpack Compose, але написана на Kotlin Native та використовує 2Д рендерер Skija для рендеренгу UI елементів, що дає змогу використовувати цей фреймворк для різних платформ, попри плюси цього порту використання має деякі обмеження, пов’язані з Kotlin Native, але проект розвивається як і сам Kotlin Native, тому згодом це буде повноцінний конкурент таким кросплатформним UI фреймворкам як: Flutter, React Native та Xamarin.

Для асинхронної роботи використовується бібліотека для мови програмування Kotlin - coroutines, вона дозволяє ефективно використовувати ресурси процесора під час виконання асинхронних операцій.

Для забезпечення зв’язку між мережевим додатком та мобільним додатком на платформі Android, доцільно використати перевірену роками бібліотека для мови програмування Java – Retrofit [6], це бібліотека, яка забезпечує доступ до мережевих додатків, останні версії бібліотеки мають вбудовану підтримку співпрограм мови Kotlin.

Для роботи з файлами типу JSON використовується бібліотека для мови програмування Kotlin – Kotlinx Serialization.

Для тестування використовувався фреймворк JUnit та Espresso.

3. РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ РОЗКЛАДОМ ДИСТАНЦІЙНОГО НАВЧАННЯ

3.1. Реалізація мережевого додатку КС УРДН

3.1.1. Модуль роботи з АПІ

Цей модуль використовує фреймворк KTOR для приймання та відправки запитів, для дотримання специфікації API, використовується бібліотека роботи з json файлами – Kotlinx Serialization.

Основна робота цього модуля полягає в тому, щоб прийняти запит, обробити всі необхідні параметри та передати його на обробку в модуль основної логіки, а той, в свою чергу, має надати відповідь, яку модуль для роботи з АПІ повинен привести до вигляду, який описаний в АПІ специфікації, та відправити назад відправнику. У випадку коли сталася помилка модуль повинен відловити її, обробити згідно специфікації та сповістити про це відправника запиту.

Модуль поєднує в собі усі модулі, адже саме цей модуль відповідальний за комунікацію мережевого додатку та інших компонентів системи, тому модуль роботи з АПІ є точкою входу та створює основні класи мережевого додатку.

Залежності модуля роботи з АПІ від інших модулів зображена на рисунку 3.1:

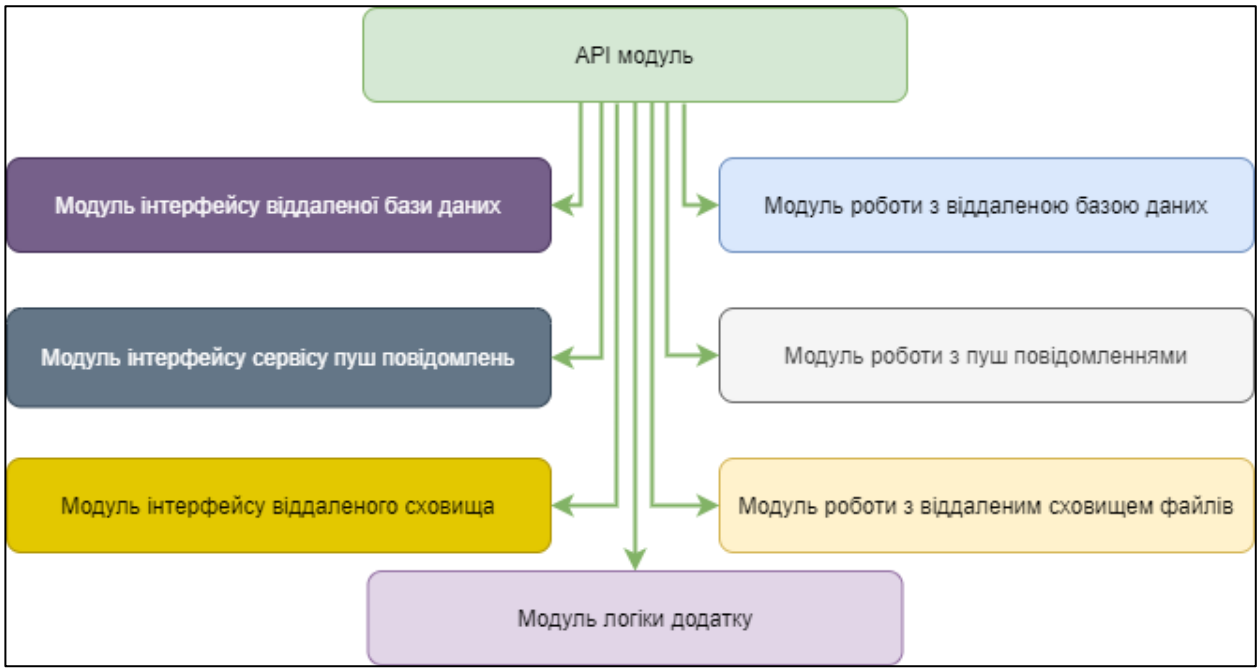


Рисунок 3.1 – Залежність модуля роботи з АПІ від інших модулів

Перелік залежностей модуля роботи з АПІ:

- Залежність від модуля інтерфейсу віддаленої бази необхідна для того, щоб опиратися в реалізації лише на інтерфейси, а не на конкретну реалізацію компонента;
- залежність від модуля роботи з віддаленої базою необхідна для підключення конкретної реалізації інтерфейсів віддаленої бази;
- залежність від інтерфейсу сервісу пуш повідомлень необхідна для того, щоб опиратися в реалізації лише на інтерфейси, а не на конкретну реалізацію компонента;
- залежність від модуля роботи пуш повідомлень необхідна для підключення конкретної реалізації інтерфейсу сервісу пуш повідомлень;
- залежність від модуля інтерфейсу віддаленого сховища необхідна для того, щоб опиратися в реалізації лише на інтерфейси, а не на конкретну реалізацію компонента;
- залежність від модуля роботи з віддаленим сховищем необхідна для відключення конкретної реалізації інтерфейсу віддаленого сховища;
- залежність від модуля логіки додатку необхідна для реалізації коректної логіки.

Схема оброблення запитів представлена на рисунку 3.2:

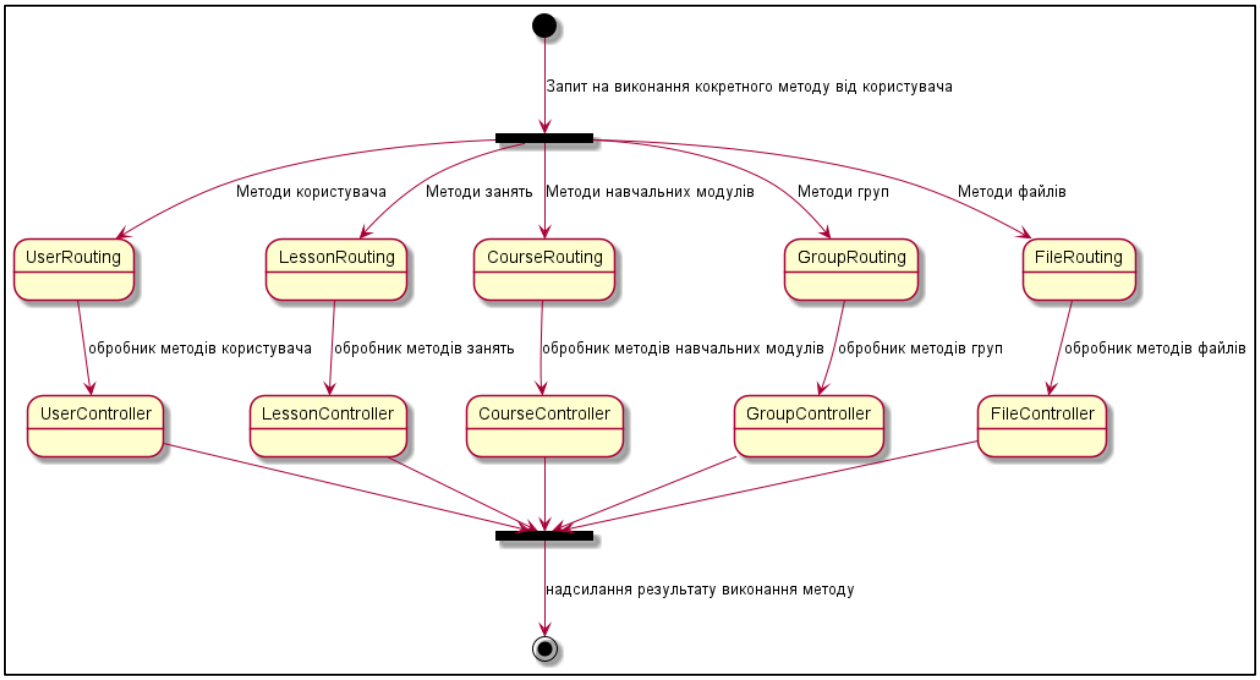


Рисунок 3.2 – Схема оброблення запитів від користувача

Оброблення запитів виконується наступним чином:

1. Користувач надсилає запит на виконання мережевим додатком якоїсь операції, методу;
2. Запит потрапляє на routing, який визначає що це за тип запиту та до якого обробника надсилати запит;
3. Запит надсилається до відповідного обробника;
4. Обробник викликає методі модуля логіки для того, щоб виконати необхідну операцію;
5. Обробник транслює відповідь модуля логіки згідно АПІ специфікації;
6. Обробник надсилає повідомлення користувачу.

Методи пов’язані з операціями над користувачами, рисунок 3.3:

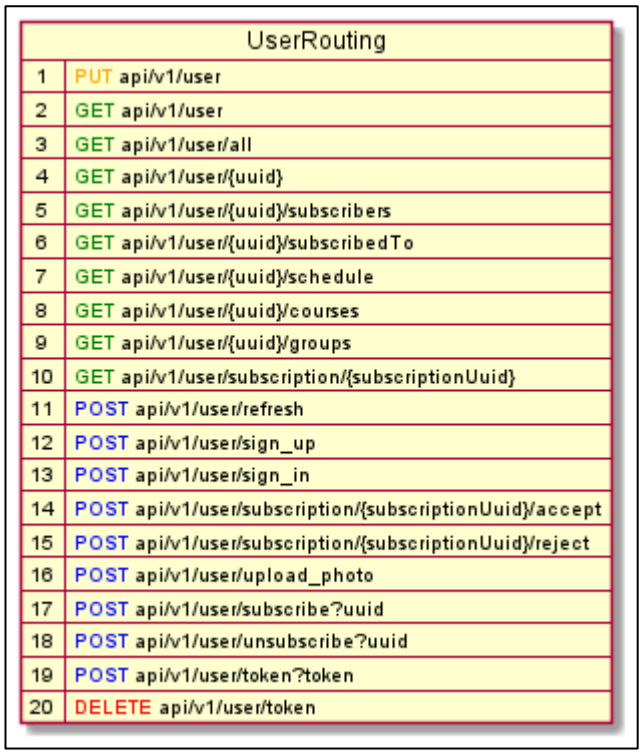


Рисунок 3.3 –методи, які доступні для операцій, які пов’язані з користувачами

Склад методів:

1. Запит, шо обробляє оновлення інформації про користувача;
2. Запит, що повертає інформацію про користувача який зробив цей запит або помилку, в разі, якщо користувач не знайдений;
3. Запит, що повертає інформацію про усіх користувачів системи;
4. Запит, що повертає інформацію про конкретного користувача чий ідентифікатор збігається з тим, який переданий як параметр або помилку якщо користувача з таким ідентифікатором не знайдено;
5. Запит, що повертає усіх користувачів, що підписані на користувача чий ідентифікатор збігається з тим, який переданий як параметр або помилку якщо користувача з таким ідентифікатором не знайдено;
6. Запит, що повертає усіх користувачів на яких підписаний користувач ідентифікатор якого збігається з тим, який переданий як параметр або помилку якщо користувача з таким ідентифікатором не знайдено;
7. Запит, що повертає розклад занять користувачеві чий ідентифікатор збігається з тим, який переданий як параметр або помилку якщо

користувача з таким ідентифікатором не знайдено;

8. Запит, що повертає усі навчальні модулі, на які підписаний користувач ідентифікатор якого збігається з тим, який переданий як параметр або помилку якщо користувача з таким ідентифікатором не знайдено;
9. Запит, що повертає усі групи, в яких приймає участь користувач ідентифікатор якого збігається з тим, який переданий як параметр або помилку якщо користувача з таким ідентифікатором не знайдено;
10. Запит, що повертає підписку користувача на іншого користувача за ідентифікатором підписки;
11. Запит, який перевіряє токен авторизації, та в разі, якщо інформація, яка передана цим токеном коректна, генерує новий токен та повертає його;
12. Запит, який реєструє користувача;
13. Запит, який робить вхід користувача в систему та повертає токен авторизації;
14. Запит, який приймає підписку іншого користувача;
15. Запит, який відхиляє підписку іншого користувача;
16. Запит загрузки фотографії користувача;
17. Запит підписки на іншого користувача;
18. Запит відміни підписки на іншого користувача;
19. Запит на додавання ідентифікатора пристрою в сервісі пуш повідомлень;
20. Запит на видалення ідентифікатора пристрою в сервісі пуш повідомлень.

Методи роботи з навчальними модулями зображені на рисунку 3.4:

CourseRouting	
1	GET api/v1/course/all
2	GET api/v1/course/{uuid}
3	GET api/v1/course/{uuid}/schedule
4	POST api/v1/course
5	POST api/v1/course/{subscriptionUuid}/accept
6	POST api/v1/course/{subscriptionUuid}/reject
7	POST api/v1/course/{uuid}/subscribe
8	POST api/v1/course/{uuid}/unsubscribe?userUuid

Рисунок 3.4 – Методи, які доступні для операцій над навчальними модулями

Склад методів:

1. Метод, що повертає усі навчальні модулі системи;
2. Метод, що повертає навчальний модуль ідентифікатор якого збігається з переданим, інакше повертає помилку про те, що навчальний модуль не знайдено;
3. Метод, що повертає розклад занять для навчального модуля;
4. Метод, що створює навчальний модуль;
5. Метод, що приймає заявку на вступ до навчального модуль;
6. Метод, що відхиляє заявку на вступ до навчального модуля;
7. Метод, який створює заявку на вступ до навчального модуля;
8. Метод, який видаляє заявку на вступ до навчального модуля.

Методи для роботи з групами в навчальних модулях, рисунок 3.5:

GroupRouting	
2	GET api/v1/group/{uuid}
3	GET api/v1/group/{uuid}/schedule
4	POST api/v1/group
5	POST api/v1/group/{subscriptionUuid}/accept
6	POST api/v1/group/{subscriptionUuid}/reject
7	POST api/v1/group/{uuid}/subscribe
8	POST api/v1/group/{uuid}/unsubscribe?userUuid

Рисунок 3.5 – Методи роботи з групами в навчальних модулях

Склад методів:

1. Метод, що повертає групу ідентифікатор якої збігається з переданим, інакше повертає помилку про те, що групу не знайдено;
2. Метод, що повертає розклад занять для конкретної групи;

3. Метод, що створює навчальний групу;
4. Метод, що приймає заявку на вступ до групи;
5. Метод, що відхиляє заявку на вступ до групи;
6. Метод, який створює заявку на вступ до групи;
7. Метод, який видаляє заявку на вступ до групи.

Запити роботи з заняттями, рисунок 3.6:

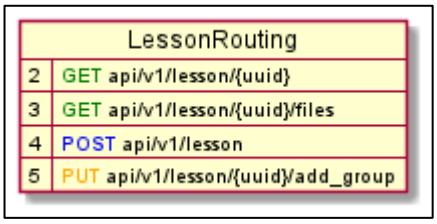


Рисунок 3.6 – Доступні методи роботи з заняттями

Склад методів:

1. Метод, що повертає заняття ідентифікатор якого збігається з переданим, інакше повертає помилку про те, що заняття не знайдено;
2. Метод, що повертає файли до заняття ідентифікатор якого збігається з переданим;
3. Метод, що створює заняття;
4. Метод, що додає групу до заняття.

Оскільки система підтримує завантаження файлів будь-якого типу, існують метод для роботи з файлами та контролер для обробки запитів, які зображені на рисунку 3.7:

Рисунок 3.7 – Метод для завантаження файлів

Склад методів:

1. Метод, який відправляє зміст файлу, ідентифікатор якого збігається з переданим.

3.1.2. Модуль основних класів системи

Цей модуль містить основні класи системи, оскільки мережевий додаток

поділений на декілька модулів, то виникає необхідність виділити окремий модуль, який містить основні логічні моделі системи і який буде підключатися до інших модулів. Таке рішення дозволить описати основні класи тільки один раз, а в інших модулях використовувати попередньо описані класи.

До основних класів системи належать (Рисунок 3.8):

1. Перелік, який визначає рівень доступу користувача
2. Перелік, який визначає рівень статусу підписки;
3. Модель користувача;
4. Модель підписки;
5. Модель навчального модуля;
6. Модель групи в навчальному модулі;
7. Модель заняття.
8. Модель підписки користувача.

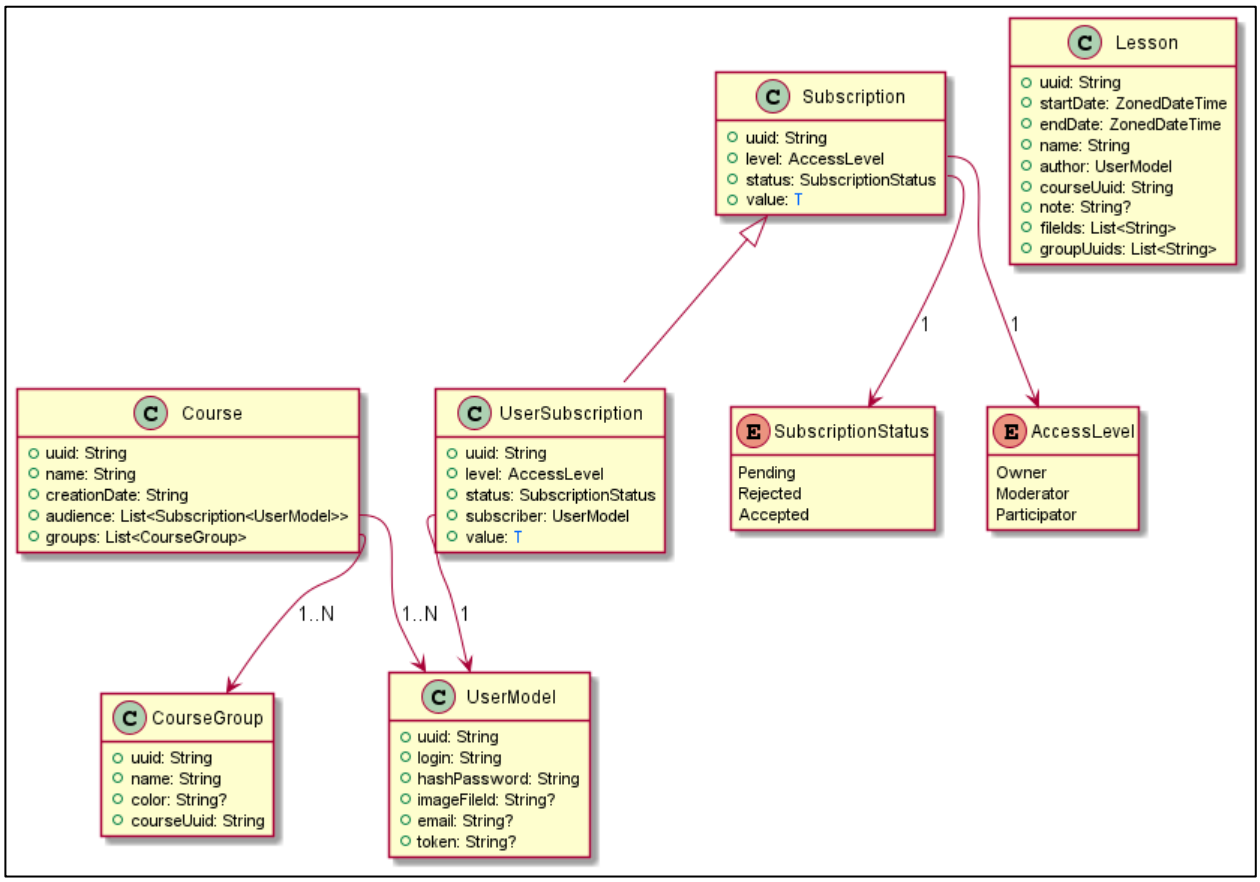


Рисунок 3.8 – Діаграма класів модуля основних класів

3.1.3. Модуль транспортних класів системи

Цей модуль необхідний для представлення класів модуля основних класів у JSON вигляді. В модулі транспортних класів системи використовується бібліотека Kotlinx Serialization, яка переводить класи мовою Kotlin у JSON файли.

Цей модуль використовується в API модулі та в модулі основної логіки.

Діаграма класів модуля зображена на рисунку 3.9.

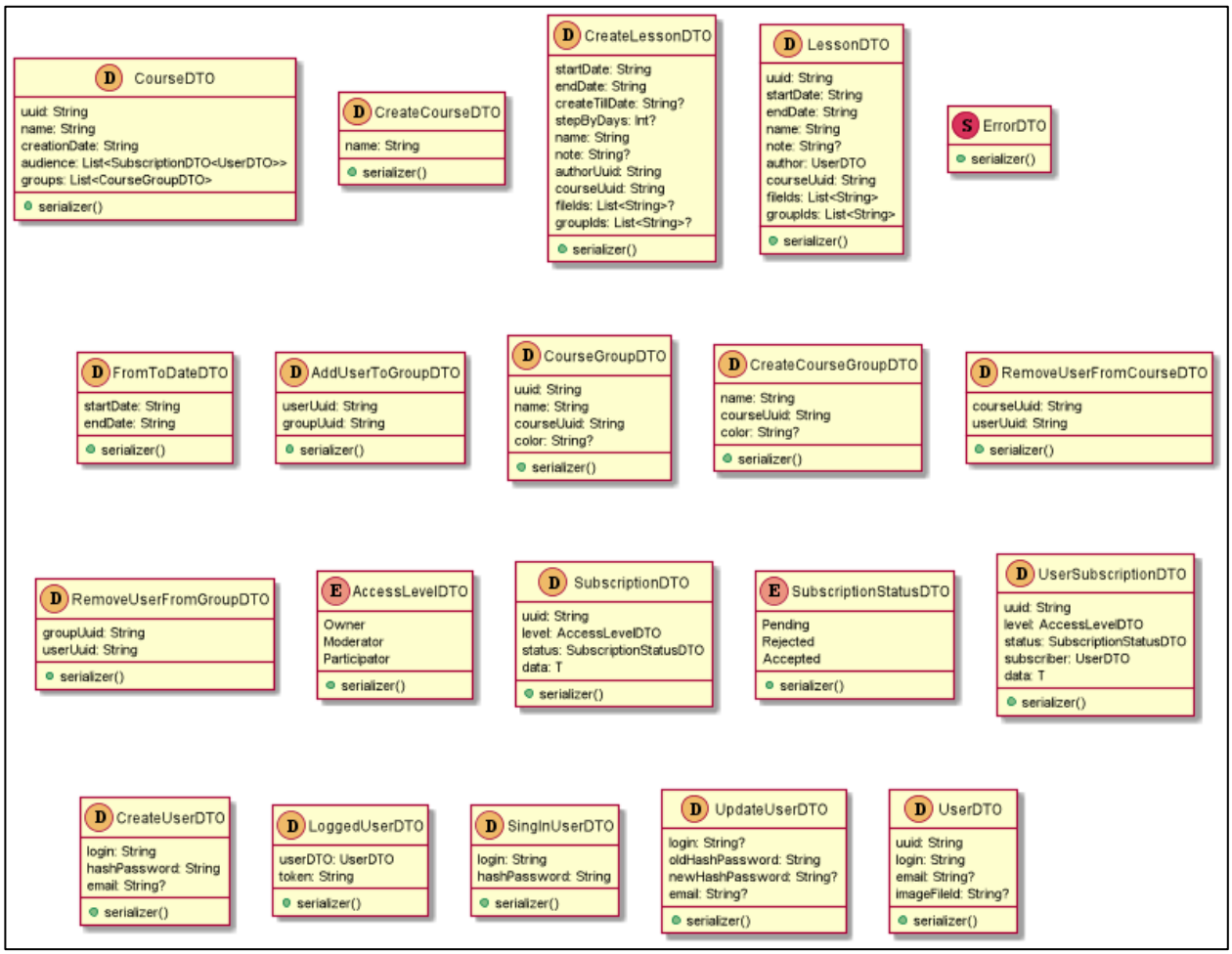


Рисунок 3.9 – Діаграма класів модуля транспортних класів системи

Як видно з рисунку 3.9 кожен клас має метод serializer, який використовується для сереалізації та десереалізації класів мови програмування Kotlin у файл типу JSON.

3.1.4. Модуль основної логіки додатку

Цей модуль складається тільки з класів та інтерфейсів мови програмування Kotlin та не залежить від ніяких сторонніх бібліотек.

В цьому модулі містяться необхідні класи для забезпечення коректної логіки додатку. Саме класи цього модуля виконують перевірку коректності даних переданих АПІ модулем, виконують необхідну логіку, перевіряючи рівень доступу користувача, та повертають відповідь назад АПІ модулю, який в свою чергу відправляє її користувачу системи.

Залежність модуля основної логіки додатку від інших модулів зображена на рисунку 3.10.

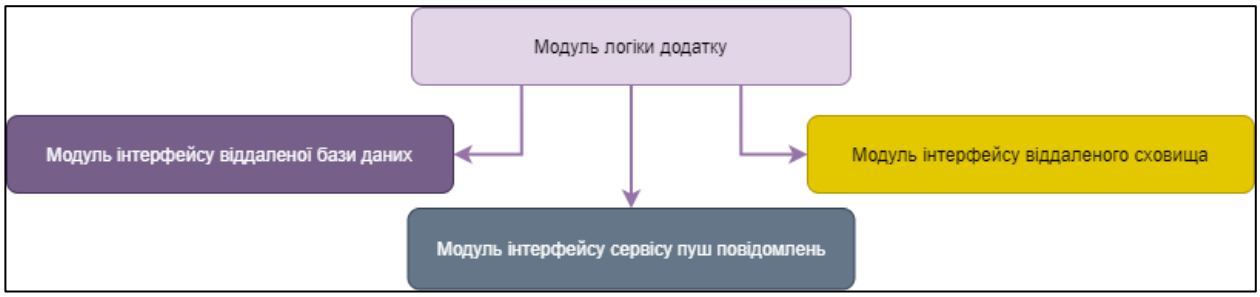


Рисунок 3.10 – Залежність модуля основної логіки від інших модулів мережевого додатку

Оскільки основна логіка додатку потребує доступу до бази даних, сервісу пуш повідомлень та віддаленого сховища, то модуль основної логіки залежить від інтерфейсів перелічених модулів.

Зв’язок основних класів цього модуля зображені на рисунку 3.11.

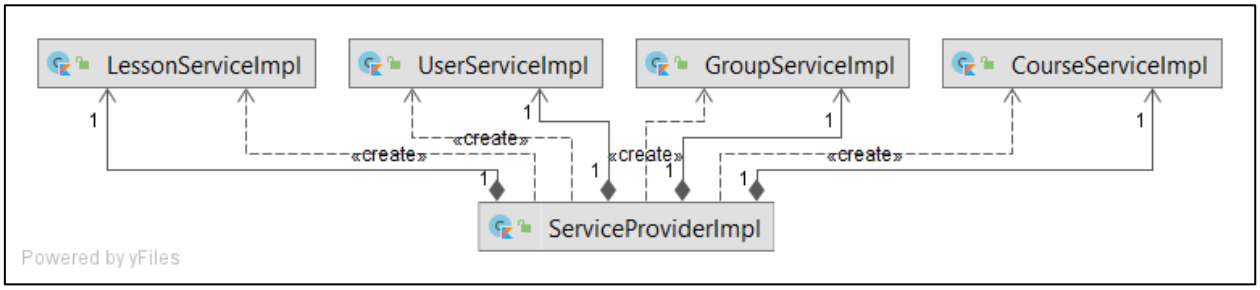


Рисунок 3.11 – Зв’язок основних класів модуля

Саме в зазначених класах закладена основна логіка система, така як:

- доступ до віддаленої бази даних через інтерфейс, підтримка модифікації даних через інтерфейс;
- логіка модифікування навчальних модулів, груп та занять;
- логіка підписування користувачів на навчальні модулі, групи та інших користувачів;

- відслідковування змін в навчальному модулі, занятті чи групі та повідомлення усіх необхідних користувачів.

3.1.5. Модуль роботи зі сховищем файлів

Цей модуль повинен забезпечити безпечний доступ до віддаленого сховищай файлів. Оскільки в якості віддаленого сховища файлів використовуєтсья Google Drive, який має свою власну бібліотеку доступу до віддаленого сховища файлів, то основні класи системи мають наступний вигляд, рисунок 3.12.

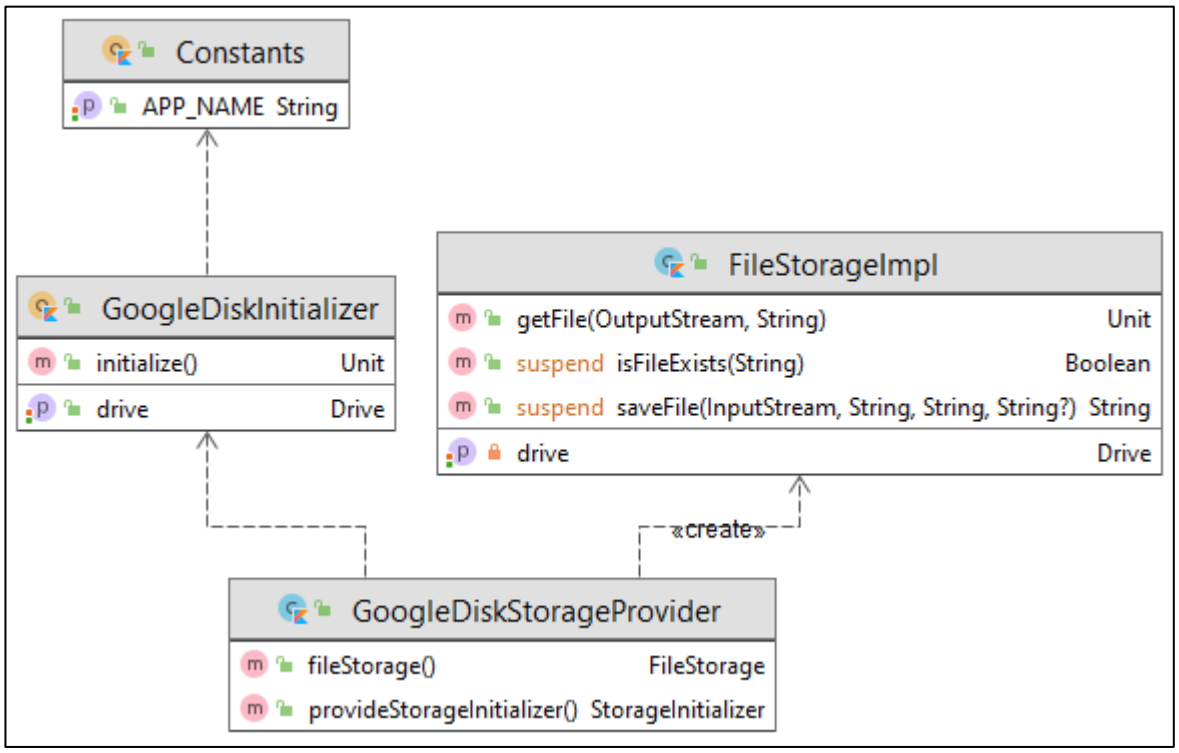


Рисунок 3.12 – Основні класи модуля роботи з віддаленим сховищем

Клас, який відповідальний за реалізацію доступу до віддаленого сховище, FileStorageImpl, реалізує доступ до сховища файлів за допомогою вище згаданої Java бібліотеки для доступу до віддаленого сховища файлів Google Drive, позначеню Drive на рисунку 3.12 та має наступні можливості:

- перегляд змісту всього диску;
- перегляд змісту окремого каталагу диску;
- перегояд змісту окремого файлу;
- модифікація файлу;

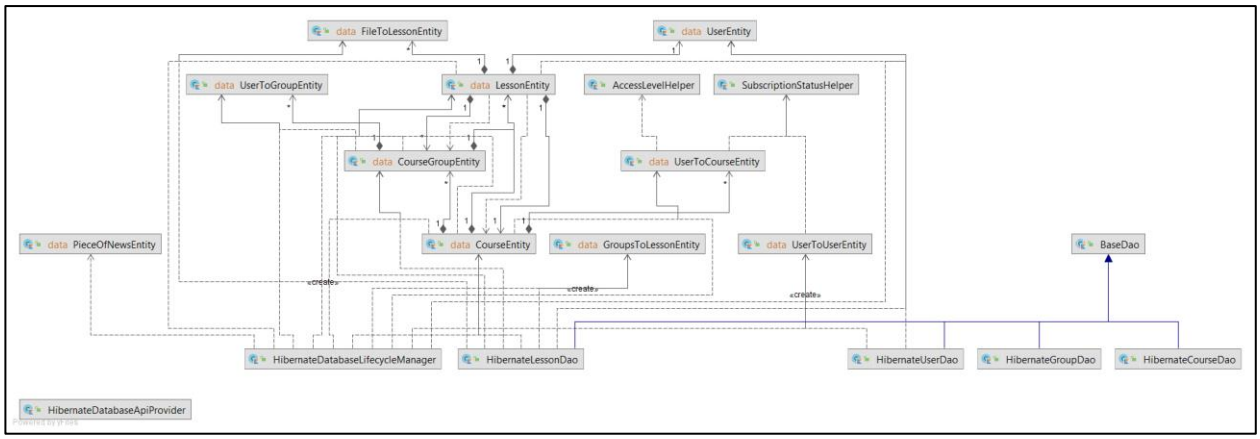
- завантаження файлу;

3.1.6. Модуль роботи з віддаленої базою даних

Саме цей модуль реалізує підключення, доступ до віддаленої бази даних та переведення таблиці у класи мовои програмування Kotlin.

Модуль не має залежностей від інших модулів, тільки від бібліотеки Hibernate.

Щоб забезпечити вище перечислені вимоги використовується бібліотека Hibernate. Ця бібліотека має дуже гнучке налаштування підключення до СУБД. Основні класи модуля та зв’язок між класами зображена на рисунку 3.13.



Риснуок 3.13 - Основні класи модуля роботи з віддаленою базою даних та зв’язок між класами

Бібліотека Hibernate є найпопульрнішою реалізацією JPA, також багато файрмворків та IDE мають вбудовану підтримку цього фреймворку для роботи з СУБД.

3.2. Реалізація мобільного додатку під платформу Android управління КС УРДН

3.2.1. Модуль користувацького інтерфейсу

Для реалізації користувацького інтерфейсу використовувася новий UI фреймворк – Jetpack Compose. Основними елементами фреймворку є composable функції, які описують інтерфейс.

Реалізовані розділи додатку:

1. Екран авторизації, рисунок 3.14.

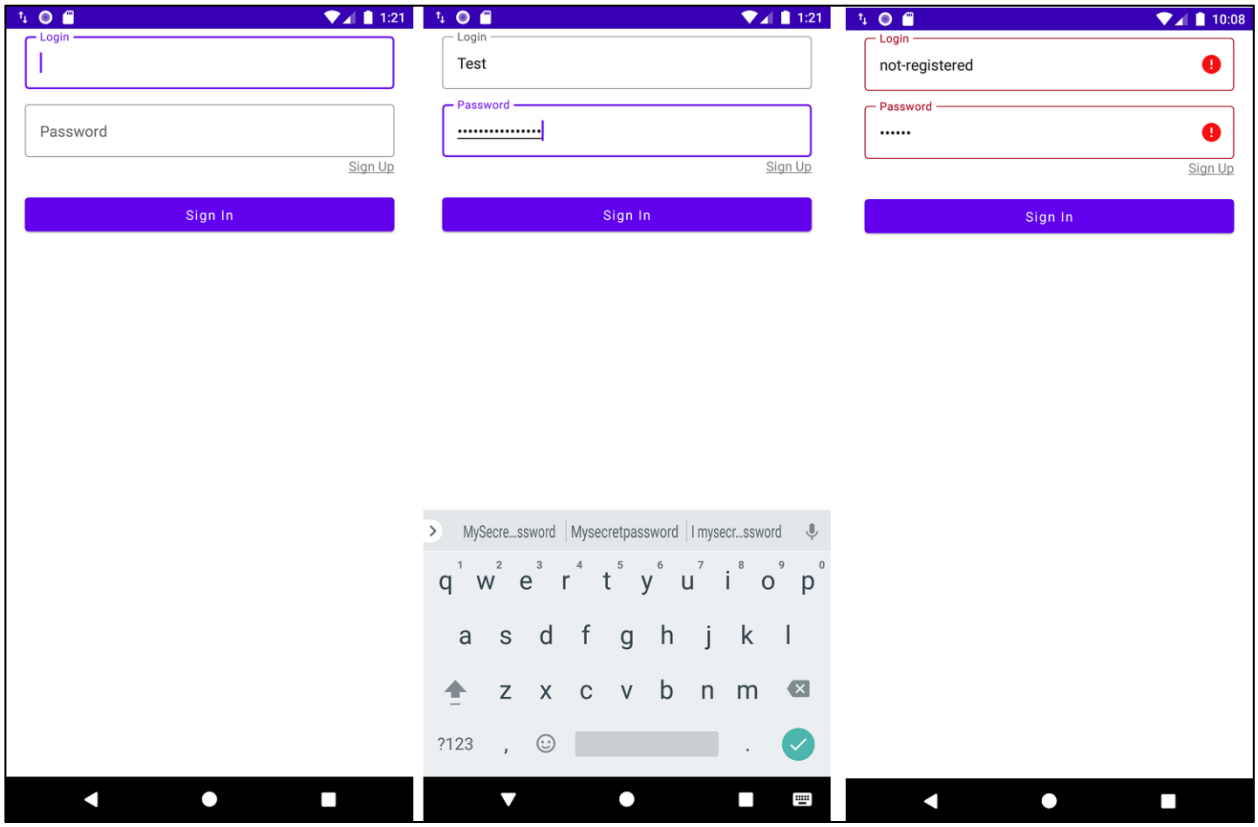


Рисунок 3.14 – Можливі стани екрану авторизації

2. Екран реєстрації, рисунок 3.15.

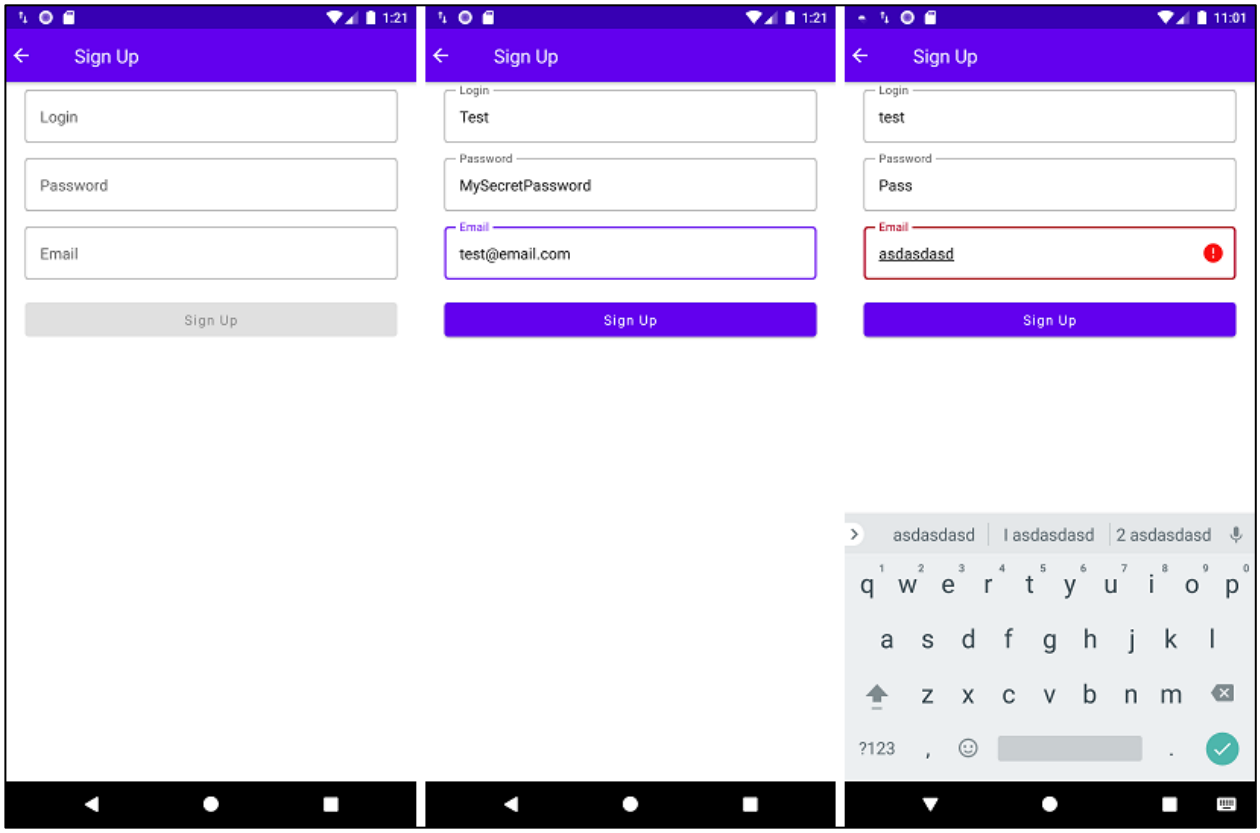


Рисунок 3.15 – Можливі стани екрану реєстрації

3. Головний екран з розкладом, рисунок 3.16.

Рисунок 3.16 – Головний екрна з розкладом додатку

4. Список навчальних модулів, рисунок 3.17.

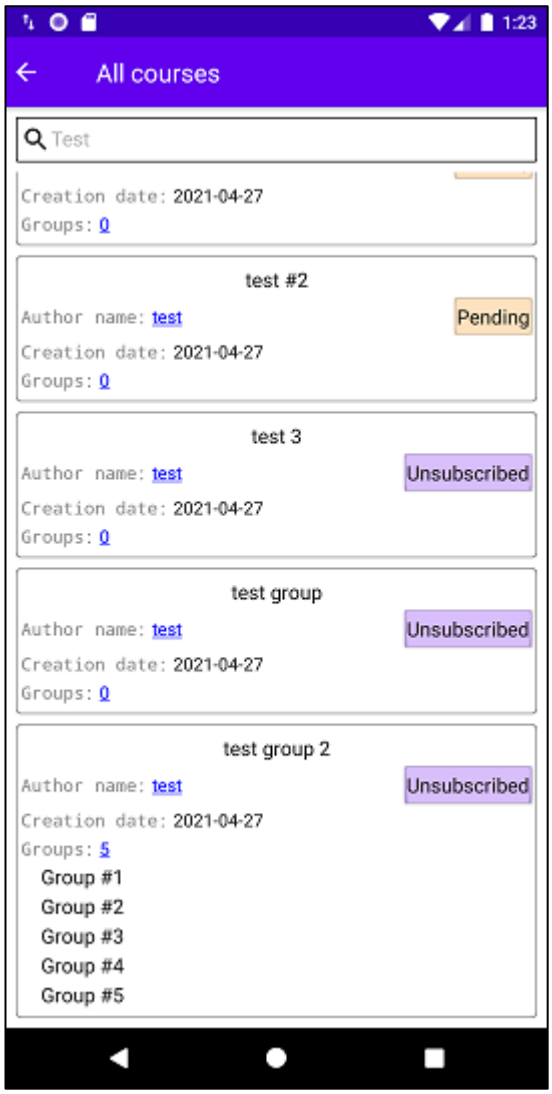


Рисунок 3.17 – Список навчальних модулів

5. Екран детальної інформації про навчальний модуль, рисунок 3.18.

Рисунок 3.18 – Екран детальної інформації про навчальний модуль

6. Ділог вибору дати та часу, рисунок 3.19.

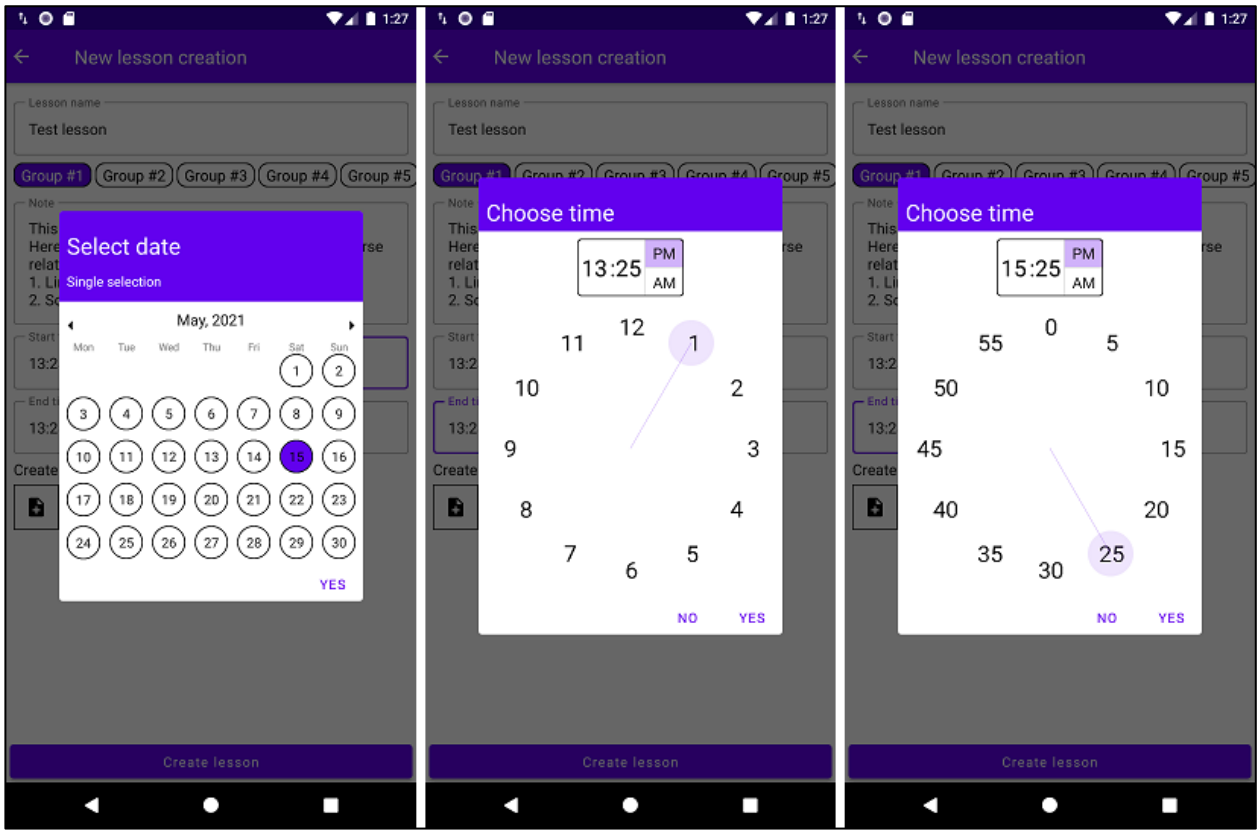


Рисунок 3.19 – Діалоги вибору дати та часу

7. Екран створення нового навчального модуля, рисунок 3.20.

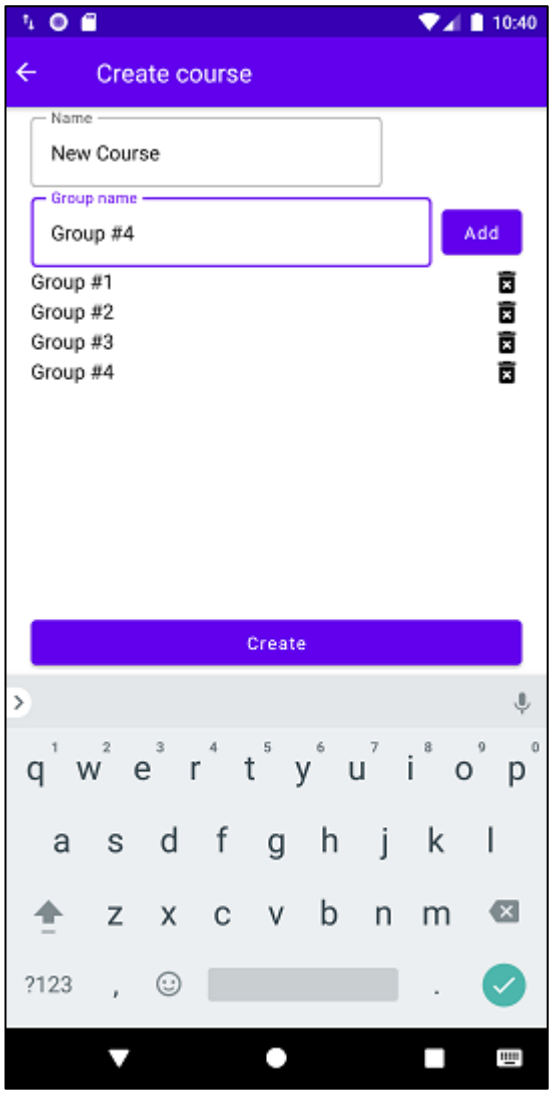


Рисунок 3.20 – Створення нового навчального модуля

8. Екран створення нового заняття, рисунок 3.21.

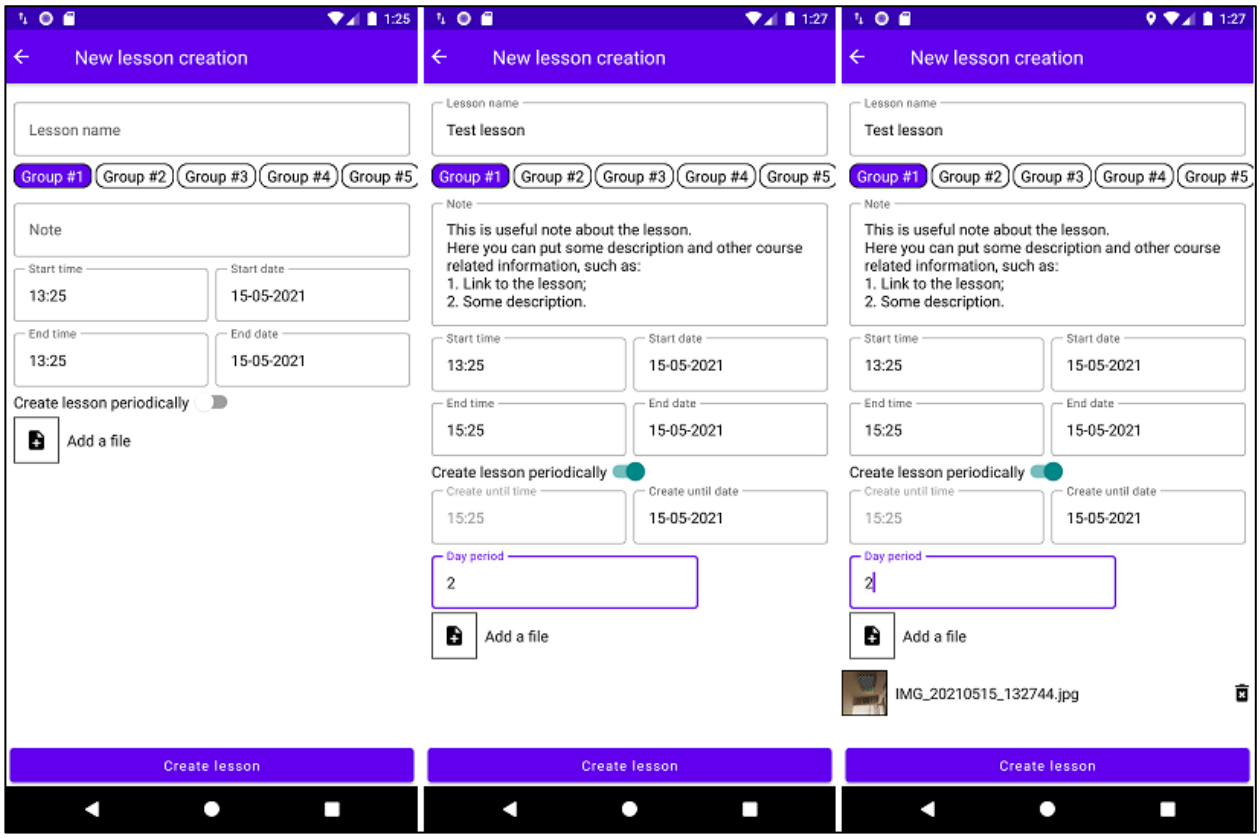


Рисунок 3.21 – Екран створення нового заняття.

9. Сповіщення про події, рисунок 3.22.

Рисунок 3.22 – Сповіщення про створення нового навчального модуля

10.Головне меню додатку, рисунок 3.23

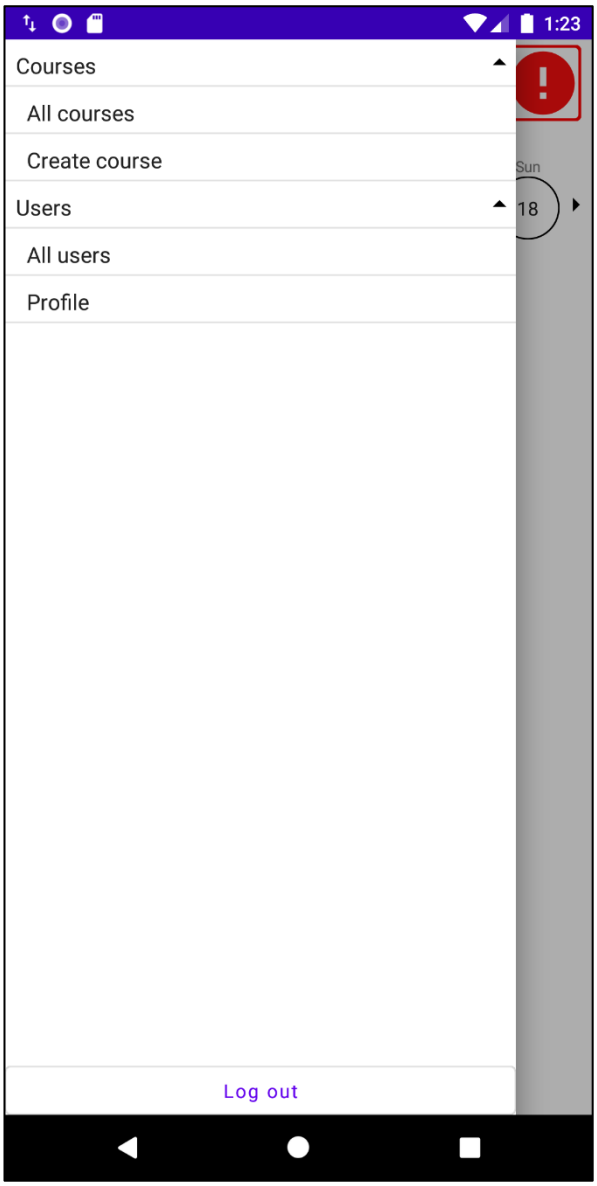


Рисунок 3.23 – Головне меню додатку

3.2.2. Модуль роботи з віддаленим сервером

Цей модуль забезпечує комунікацію між мережевим додатком та додатком на платформі Android. Для підключення до мережевого додатку використовується бібліотек для мови програмування Java – Retrofit. Містить усі методи, які описані детальніше в розділі реалізації мережевого додтку та моделі точно такі ж як і в модулі транспортних класів системи. Оскільки мережевий додаток та мобільний додаток використовують одну й ту ж мову програмування Kotlin, ці класи та методи не потрібно описувати двічі, а можна просто підключити необхідний модуль з мережевого додатку.

3.2.3. Модуль роботи з локальними файлами

Цей модуль відповідальний за зберігання та керування локальними файлами на мобільному додатку під платформу Android.

Модуль забезпечує наступний функціонал:

- завантаження файлів з прогресом та їх зберігання на диску;
- пошук файлів в локальному просторі;
- видалення файлів.

Основні класи модуля, рисунок 3.23:

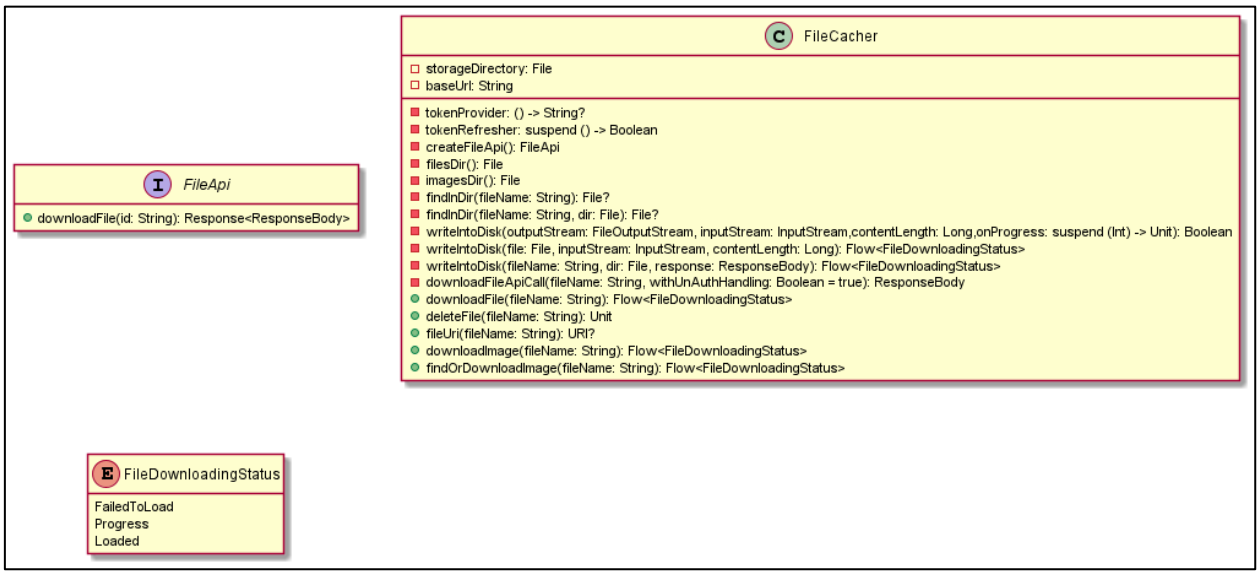


Рисунок 3.23 – Основні класи модуля роботи з локальними файлами

4. ШЛЯХИ ПОЛІПШЕННЯ

4.1. Шляхи поліпшення мережевого додатку

Оскільки розроблена система представляє собою так званий MVP, тому можливостей для покращення системи є багато, але насамперед потрібно звернути увагу саме на критичні елементи системи, такі як:

- захист інформації;
- відмовостійкість;
- максимальну пропускну спроможність;
- інтеграцію з існуючими системами.

Захист інформації можна покращити завдяки передачі даних між клієнтом (додатком під платформу Android) та мережевим додатком за допомогою HTTPS, також оскільки СУБД знаходиться на віддаленому сервері, то необхідно захистити передачу даних і між мережевим додатком та СУБД.

Відмовостійкість можна покращити за допомогою виділення окремих сервісів для кожного логічного компоненту. Тобто, переписати з монолітної архітектури на мікросервісну, такі зміни потребують часу та більше продуманої комунікації внутрішніх елементів мережевого додатку, але в перспективі це дасть доволі велику відмовостійкість, адже помилка в одному ізольованому сервісі не призведе до помилки системи в цілому. Також, розглядаючи мікросервісну архітектуру можна покращити й максимальну пропускну спроможність завдяки клонуванню критичних сервісів та розподілення трафіку між ними порівно.

Щодо інтеграції з існуючими системами мається на увазі такі системи як Zoom, Skype, Microsoft Teams, Google Meets, Slack ті інші системи, завдяки яким можна проводити онлайн зустрічі.

Як правило такі системи розвиваються за вимогами користувачів, тобто розробники проводять опитування, в якому визначають яку функціональність їм слід додати або покращити.

4.2. Шляхи поліпшення додатку на платформі Android

Серед поліпшень додатку на платформі слід виділити наступні пункти:

- підвищення конфіденційності;
- підвищення продуктивності додатку;
- покращення інтерфейсу користувача.

Оскільки додаток приймає конфіденційну інформацію, то постає необхідність забезпечити її надійний захист. Під платформу Android існують декілька готових рішень цієї проблеми.

Щодо продуктивності додатку, то збільшення цього параметру допустиме завдяки очікуванню стабільної версії Jetpack Compose, в якому розробники обіцяють приділити велику увагу продуктивності UI фреймворку та виправлення деяких відомих багів.

ВИСНОВКИ

Під час виконання дипломного проєкту була розроблена комп’ютерна система керування розкладом дистанційного навчання.

В ході виконання дипломного проєкту був проведений аналів вже існуючих рішень з метою додавання нового функціоналу та виявлення і усунення деяких існуючих недоліків.

Під час проектування було розглянуто різні підходи до реалізації, різні архітектуру, різні мови програмування та фреймворки. Детально описані усі технології, що використовувалися під час реалізації, їх плюси та мінуси та обґрунтування доцільності вибраних технологій.

Під час реалізації системи детально розглянуто функціонал, який був створений та шляхи його використання. Продемонстровано доступні методі мережевого додатку та їх опис, структурна схема бази даних та показані екрани у створеному додатку під платформу Android.

Розроблена система має місце використання не лише у ЗВО, а також і для ефективного контролю тренінгів та онлайн-курсів. Система надає змогу зручно планувати розклад занять, підтримувати зв’язок студентів з викладачами, слугувати єдиним джерелом інформації щодо навчального процесу та всього, що з ним пов’язано, а також вчасно інформувати всіх студентів про зміни в ході навчання або інші необхідні новини.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Фреймворк створення мережевих додатків Spring Framework [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Spring_Framework.
2. Фреймворк створення мережевих додатків Quarkus [Електронний ресурс]. Режим доступу: https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus.
3. Фреймворк створення мережевих додатків KTOR [Електронний ресурс]. Режим доступу: https://ktor.io/docs/welcome.html.
4. Документація співпрограм мови Kotlin - Kotlinx Coroutines [Електронний ресурс]. Режим доступу: https://github.com/Kotlin/kotlinx.coroutines
5. Документація бібліотеки роботи з JSON файлами - Kotlinx Serialization [Електронний ресурс]. Режим доступу: https://github.com/Kotlin/kotlinx.serialization
6. Документація бібліотеки роботи з мережевими додатками Retrofit [Електронний ресурс]. Режим доступу: https://square.github.io/retrofit/
7. Інформація про СУБД MySQL [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/MySQL
8. Офіційна документація розроблення додатків під платформу Android [Електронний ресурс]. Режим доступу: https://developer.android.com/
9. Інформація про мову програмування Kotlin [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Kotlin
10. Жемеров Д. Kotlin в действии / Д. Жемеров, С. Исакова., 2017. – 402 с. – (Manning Publications).
11. Dawn G. , Head First Android Development: A Brain-Friendly Guide / G. Dawn, G. David., 2016. – 734 с.