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

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему: Система гнучкого управління розкладом заходів

Виконав: студент 4 курсу, групи__ТВ-з51

_____Олійник Богдан Степанович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _к.т.н., доц. Смаковський Денис Сергійович_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший , бакалаврський
Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ О.В. Коваль
(підпис)
”___”_____2018р.

ЗАВДАННЯ
на дипломну роботу студенту
_____Олійнику Богдану Степановичу_____
(прізвище, ім’я, по батькові)

1.Тема роботи Система гнучкого управління розкладом заходів
керівник роботи Смаковський Денис Сергійович, к.т.н., доц.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201р. №__

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи: персональний комп`ютер під керуванням операційної системи Ubuntu 16.04, мова програмування Kotlin та JavaScript
4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)проаналізувати задачу створення системи гнучкого управління розкладом заходів, визначити переваги та недоліки подібних систем, розробити серверну частину та клієнтські додатки для системи гнучкого управління розкладом заходів.

5. Перелік ілюстративного матеріалу: схеми архітектури додатку, знімки екранних форм, діаграма прецедентів системи, діаграма структури системи, зразки розробленого інтерфейсу додатку.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ” ”_____201 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Програмна реалізація системи		
5.	Оформлення пояснювальної записки		
6.	1 Захист програмного продукту		
7.	2 Передзахист		
8.	Захист		

Студент _____ Олійник Б.С.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Смаковський Д.С.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці системи гнучкого управління розкладом заходів.

Метою роботи є аналіз існуючих аналогів такої системи і впровадження власного рішення на основі переваг і недоліків розглянутих систем.

Для досягнення мети були вирішені наступні задачі:

1. Провести аналіз існуючих систем.

2. Дослідити інструменти для створення системи орієнтованої на мобільні пристрої.

3. Спроектувати та впровадити серверну частину системи.

4. Забезпечити користувачів даної системи клієнтськими додатками на бази мобільних технологій.

В роботі було використано архітектурну платформу Firebase, для якої кодування здійснювалось на мові програмування JavaScript. Клієнтські додатки були створені для мобільної платформи Android з використанням мови Kotlin та Android SDK.

Обсяг роботи 67 сторінок, 20 ілюстрацій, 7 таблиць, 11 використаних джерел, 3 додатки.

Ключові слова: розклад, android, firebase, kotlin, адміністратор, захід.

ABSTRACT

This thesis is devoted to the development of a system of flexible control of schedule events.

The purpose of the work is to analyze the existing analogues of such a system and implement its own solution based on the advantages and disadvantages of the systems under consideration.

To achieve the goal, the following tasks were solved:

1. Analyze existing systems.

2. Explore tools for creating a mobile-oriented system.

3. Design and implement the server part of the system.

4. Provide users of this system with client applications to the base of mobile technologies.

In this work, the architectural platform Firebase, for which encoding was performed in the programming language JavaScript, was used. Client apps have been created for the Android mobile platform using the Kotlin language and Android SDK.

Expanatory note: 67 p., 20 fig., 7 appendixes, 11 references.

Keywords: schedule, android, firebase, kotlin, admin, event.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ	Програмне забезпечення
API	Application Programming Interface
MVC	Model-View-Controller
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
MVP	Model-View-Presenter
MVVM	Model-View-ViewModel
DSL	Domain-specific language
JDK	Java Development Kit
ORM	Object-relational mapping
SQL	Structured query language
UI	User interface
SDK	Software development kit

ВСТУП

Одним з найважливіших на сьогодні питань людства є розвиток. Адже він ніколи не стоїть на місці, і це в наші дні стосується всіх сфер діяльності, не тільки комп'ютерних технологій. Одними із найпопулярніших способів розвитку в наші дні є конференції, тренінги або курси лекцій.

Кількість гостей таких заходів коливається від десятків до сотень, є необхідність забезпечення всіх відвідувачів графіком лекцій, який вони зазвичай отримують в паперовому вигляді. Окрім проблеми з переглядом розкладу на заходах з плаваючим графіком лекції виникає потреба в інформуванні відвідувачів про зміни в розкладі, перенос лекцій абощо.

На допомогу в таких випадках приходять спеціальні сервіси та застосунки, які допомагають відфільтрувати та систематизувати інформацію. За допомогою таких додатків користувач може значно спростити процес пошуку та обробки потрібної інформації. Використання таких застосунків, що позиціонуються як помічник для людини в галузі планування графіку дозволяє зробити свій графік більш систематизованим, а також бути добре проінформованим про вірогідні зміни в цьому графіку. Окрім того важливою задачею постає проблема комунікації між адміністратором певного заходу і відвідувачами, кількість яких може бути доволі великою.

Поширеність мобільних телефонів в нас час дуже важко переоцінити, з високою вірогідністю більшість відвідувачів будуть мати при собі мобільних пристрій, який призначений для того щоб допомагати людині, беручи частину її обов'язків на себе.

Результати роботи пропонуються до використання сервісам, що агрегують в собі заходи, курси лекцій, практикуми. Кінцевими користувачами системи будуть відвідувачі заходів, адміністратори конкретних заходів, а також головний адмінстратор, що буде мати змогу керувати заходами. Впровадження запропонованої системи дасть можливість описаним вище користувачам зручніше співпрацювати один з одним.

Тому, було запропоновано дослідити можливість створення системи для гнучкого управління розкладом орієнтованої на мобільні пристрої та реалізувати розглянуту систему в число якої входить серверна частина та два клієнтських додатки.

1. АНАЛІЗ ВИМОГ З РОЗРОБКИ СИСТЕМИ ГНУЧКОГО УПРАВЛІННЯ РОЗКЛАДОМ ЗАХОДІВ

1.1. Аналіз вимог та постановка задачі

Метою роботи є аналіз та розробка системи гнучкого управління розкладом, яка складається з серверної частини та двох клієнтських додатків для адміністратора і відвідувача заходу.

В ході роботи було визначено наступні задачі:

– обрання архітектурної платформи для забезпечення зв`язку між двома клієнтськими додатками, що базуються на основі мобільних технологій;

– проектування та розробка серверної частини системи;

– проектування та розробка клієнтського додатку для відвідувача заходу;

– проектування та розробка клієнтського додатку для адміністратора заходу.

Для вирішення цих задач було застосовано принцип декомпозиції і визначені наступні підзадачі:

– провести дослідження сучасних методів розробки серверної частини системи орієнтованої на мобільні пристрої;

– провести порівняльний аналіз існуючих інструментів та методів розробки клієнтських додатків під мобільні платформи;

– обрати набір бібліотек для розробки серверної та клієнської частини системи;

– обрати фреймворк для реалізації клієнтських додатків;

– проаналізувати та обрати доступні види баз даних для серверної частини системи;

– спроектувати та налаштувати базу даних на сервері;

– спроектувати та налаштувати хмарне середовище не сервері для зберігання картинок;

– налаштувати систему сповіщень між клієнськими додатками;

– створити клієнтський додаток для відвідувача заходу;

– створити клієнтський додаток для адміністратора заходу.

Вихідними даними системи є список заходів, який включає в себе розклад занять. Усіма заходами в системі керує головний адміністратор головним завданням якого є підтримання актуальної інформації про заходи та їх адміністраторів на серверній частині системи. За актуальність розкладу, сповіщення про зміни, та завантаження мапи заходу відповідає адміністратор заходу.

Вимоги до розроблюваної системи:

– система гнучкого управління розкладом складається з наступних частин: серверна частина, клієнстський додаток адміністратора, клієнський додаток відвідувача заходу;

– надання можливості керування заходами, призначення для них адміністраторів;

– забезпечення надійної системи авторизації адміністратора заходу із застосування найнадійніших на сьогодні методів;

– забезпечення адміністратора заходу інструментами, що дадуть змогу зручно керувати розкладом та мапою заходу;

– забезпечення відвідувача заходу інструментами, що дадуть змогу зручно переглядати розклад та мапу заходу;

– забезпечення відвідувача заходу вчасними сповіщення про зміни в розкладі чи зміну місця проведення лекції.

1.2. Загальні положення щодо взаємодії клієнських додатків із сервером

Оскільки однією з найважливіших частин системи є взаємодія буде доцільним розглянути принципи роботи мобільних додатків із серверною частиною.

Клієнт – в обчислювальній системі це програмний компонент, який надсилає запити на сервер.

Програма-клієнт взаємодіє із сервером, використовуючи спеціальний протокол. Вона може считувати з сервера дані, змінювати їх безпосередньо на сервері, запускати на сервері нові процеси і так далі. Отримані від сервера дані програма може дати користувачу або використати в будь-яких інших цілях, це залежить від того для чого призначена програма. Програма-клієнт і програма-сервер можуть працювати як на одному компоненті, так і на різних. У другому випадку для обміну інформацією використовується мережеве з'єднання. Клієнт це частина клієнт-серверної архітектури [2].

Архітектура клієнт-серверу - є одним із архітектурних шаблонів програмного забезпечення та є важливою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію, які звертаються до них;
– набір клієнтів, які використовують сервіси, що надаються серверами;
– мережа, яка забезпечує взаємодію між клієнтами та серверами.

На рисунку 1.1 зображено базове представлення клієнт-серверної архітектури

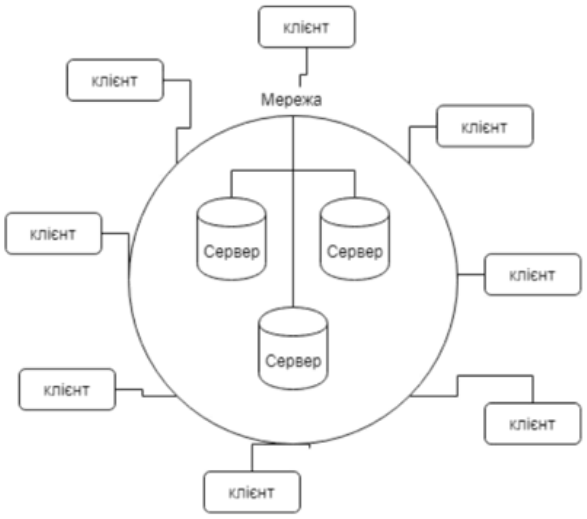

Рисунок 2.1 – Клієнт-серверна архітектура

REST – це підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. В основі REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP. Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад, HTML, XML, JSON) [6] . Будь-який REST протокол (HTTP в тому числі) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь» [8]. Стверджується, що такий підхід забезпечує масштабовність системи і дозволяє їй еволюціонувати з новими вимогами [1].

Клієнт-серверні додатки є найпоширенішими і в той же час найскладнішими в розробці. Дії, які можна виконувати над ресурсом, визначаються повідомленнями, які визначено стандартним протоколом. В системі WWW цей протокол — HTTP, але існують REST-архітектури на основі й інших протоколів [6]. Найчастіше використовують 4 тип запитів:

– GET – отримати ресурс;
– POST – створити новий ресурс;
– PUT – замінити стан поточного ресурсу;
– DELETE – видалити ресурс.

На рисунку 1.2 зображено взаємодію клієнта та сервера за допомогою REST API.

Рисунок 1.2 – Методи REST API

Також для взаємодії з сервером буде використано технологію push-сповіщень. За допомогою неї сервер може відправляти повідомлення на клієнт без його запиту.

Створення застосунків на подібній архітектурі є частим явищем. Клієнт серверна архітектура є актуальною начасі.

2. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

Існує безліч програм які дозволяють користувачам керувати свої графіком планувати та переносити зустрічі, як мінімум кожна велика корпорація випустила свою версію календаря. Також існує безліч систем в яких розклад витупає як додатковий функціонал, система, що призначена для управління розкладом подій для конференцій чи тренінгів відрізняється від інших певною системою прав користувачів та їх ролей.

Після вивчення ринку технологій і пошуку схожого програмного забезпечення для управління розкладом заходів чи подій були розглянуті наступні програми: Google Calendar, KPI Weeks, ClassUp.

2.1. Google Calendar

Одним з розглянутих продуктів був Google Calendar. Google Calendar - сервіс для планування зустрічей, подій, розроблений компанією Google. На сьогоднішній день важко собі уявити хоч одну велику операційну систему чи корпорацію в якої не було б написаної власної програми календаря. Найвідомішою з таких програм і є Google Calendar. Її використовують і вважають найзручнішою більшість користувачів Android та всієї екосистеми Google загалом.

Цей програмний продукт було взято до уваги через те, що його прямим завданням є показ розкладу користувачам. В розроблюваній системі основним функціоналом теж вважається розклад, тому за основу відображення календарного розкладу в додатку адміністратора та відвідувача заходу було взято ідеї основні принципи роботи із одного з найпопулярніших додатків на сьогодні.

Цей аналог можна розглядати скоріше як приклад того, як має виглядати і працювати програма, що ставить собі за ціль показати користувачу розклад його подій.

Як повний аналог розроблюваній системі ми можемо розглянути цей додаток наступним чином: програма Google Calendar надає своєму користувачеві можливість дуже зручно та просто керувати своїм розкладом, створювати події, а також додавати інших користувачі в свої події. Це повністю відрізняється від концепції системи гнучкого управління розкладом заходів тим, що в нашій системі визначена чітка рольова система, при якій користувач не може змінювати свій розклад, лише переглядати. А за зміну розкладу відповідає адміністратор.

Окрім того в програмному продукті висунутому на розгляд відсутній додатковий функціонал, який необхідний саме для організації заходів та подій, а саме завантаження та перегляд мапи заходу і також елемент двухфакторної авторизації користувача в залежності від його ролі.

На Рисунку 2.1 зображено інтерфейс програми Google Calendar

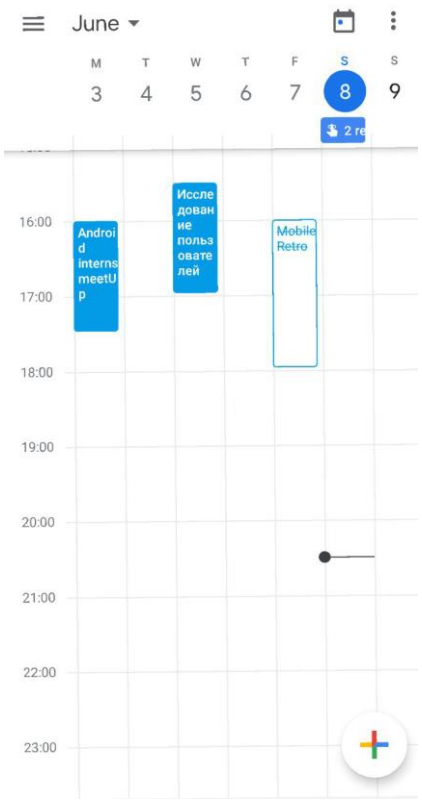

Рисунок 2.1 – Користувацький інтерфейс Google Calendar

2.2. KPI Weeks

Наступним аналогом системи гнучкого управління графіком розглянемомо мобільний додаток KPI Weeks, призначений для перегляду розкладу та орієнтований на аудиторію студентів КПІ.

Цей додаток має багато спільних елементів із розроблюваною системою, таких як перегляд розкладу. У цьому додатку як і в попередньому Google Calendar в шапці додатку ми можемо бачити блок в якому знаходить навігація по дням (Рисунок 2.2)

Нижче розміщений розклад на обраний в верхньому модулі день, лекції виділені різними кольорами для кращого сприйняття. Також в шапці ми можемо побачити елемент, що відповідає за перемикання поточного тижню з першого на другий, дні тижня представлені лише з понеділка по п`ятницю.

Додаток Kpi Weeks має багато чого спільного з розроблюваною системою, і тому при розробці буде важливо розглянути наступне: навігація по дням тижня в шапці додатку, формат виводу лекцій на день та виділення лекцій кольором залежно від порядку лекції.

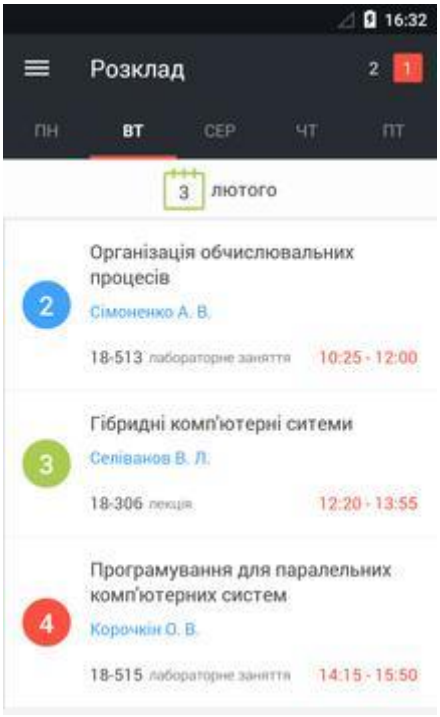

Рисунок 2.2 – Користувацький інтерфейс KPI Weeks

Загалом, цей додаток не є прямим конкурентом розроблюваному, але має багато спільного функціоналу. KPI Weeks орієнтований лише на вузьку аудиторію студентів КПІ та виконує свою базову функцію – показ розкладу.

2.3. ClassUp

Останньою розглянутою системою для реалізації системи гнучкого управління розкладом є додаток ClassUp. Це універсальний додаток для розкладу в університетах.

Він дозволяє користувачеві як перегляди розклади так і створювати свої до яких потім зможуть долучитись його одногрупники. Загалом функціонал додатку повністю копіює попередній додаток KPI Weeks, але позиціонує він себе як універсальний додаток для розкладів в університеті і дозволяє будь-якому члену класу вносити зміни в розклад, а також створювати свої розклади чи лекції. Інтерфейс для створення лекції в додатку ClassUp можемо бачити на Рисунку 2.3

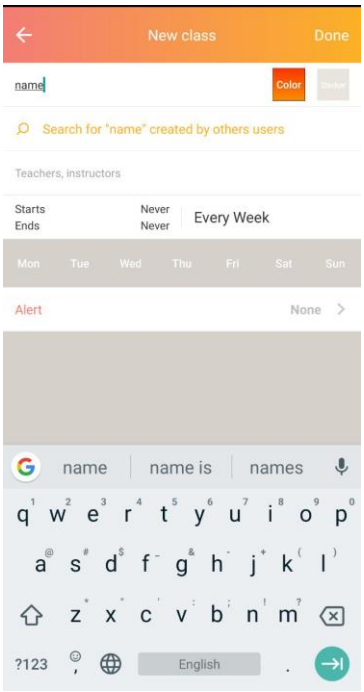

Рисунок 2.3 – Клієнський інтерфейс ClassUp

Розглянутий аналог максимально наближений до розроблюваний системи серед інших аналогів. Відмінність полягає в наступному: розроблювана система орієнтована на заходи тренінги абощо, розклад яких може змінювати, а також такі заходи потребують одного адміністратора, який буде нести відповідальність за правильність та актуальність розкладу. Додаток ClassUp же в свою чергу орієнтований суто на студенську аудиторію, де розклад заповюється самими ж студентами без перевірок та аунтефікації.

2.4. Висновки до розділу

Таким чином було розглянуто три основних й найпопулярніших системи для реалізації системи гнучкого управління розкладом.

Кожна з них має свої переваги та недоліки.

Google Calendar хороший для ніші самоорганізації та надає максимальні можливості для управління власним часом із зручними інструментами, синхронізацією. Але щодо проблеми розкладу для заходів такий інструментів місця мати не буде, адже налаштований лише під одного користувача.

KPI Weeks має приємний зрозумілий інтерфейс в якому не склад розібратись. Додаток чітко займає свою нішу, та працює для студентів КПІ. Цей додаток обмежений лише одним розкладом, і також не надає ніяких інструментів для внесення правок в розклад.

Останній додаток ClassUp найбільш наблище до ідеї розроблюваної системи. Додаток надає можливість як переглядати так і вно ити зміни в розклад, але дозволяє робити це всім користувачам, незалежно від ролі, в цьому його недолік для системи гнучкого управління заходами, а не університетським розкладом.

Тому питання доцільності розробки системи гнучкого управління розкладом, яка буде реалізовувати всі переваги та не включати всі недоліки вище перерахованих систем, стає ще більш пріоритетним.

3. ЗАСОБИ РЕАЛІЗАЦІЇ СИСТЕМИ ГНУЧКОГО УПРАВЛІННЯ РОЗКЛАДОМ

Одним із найважливіших завдань при розробці програмних продуктів є вибір таких засобів, які б полегшили роботу програміста, надавши всі необхідні інструменти для реалізації поставленого завдання, і дали б змогу отримати результат, який повністю задовольняє користувача.

Для реалізації клієнтських додатків було використано Android Studio, яка заснована на популярному сьогодні середовищі розробки Intelij Idea, та текстовий редактор Sublime Text 3. Скрипти для серверної частини, такі як тригери були написані на мові програмування JavaScript на фреймворку Node.js. Клієнтські додатки були написані на мови програмування Kotlin, та з використанням збірника проектів Gradle.

3.1. Програмне середовище клієнтських додатків

Для подальшого написання клієнтських додатків системи гнучкого управління розкладом було обрано програмне середовище Android Studio.

Android Studio – це редактор вихідних кодів, розроблений компанією Google для Windows, Linux та macOS. Вона включає в себе підтримку налагодження, вбудоване керування Git, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду. Це також налаштовується, так що користувачі можуть змінити редактор, поєднання клавіш і перевагу. Це безкоштовне та відкрите джерело, хоча офіційне завантаження знаходиться під фірмовою ліцензією [4].

Середовище розробки адаптоване для виконання завдань, що вирішуються в процесі розробки додатків для платформи Android. У тому числі у середовище

включені засоби для тестування програм на сумісність з різними версіями пристроїв та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільної здатності (планшети, смартфони, ноутбуки, годинники, окуляри тощо)[5]. Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції [6].

Графічний інтерфейс програмного середовища Android Studio можемо побачити на Рисунку 3.1.

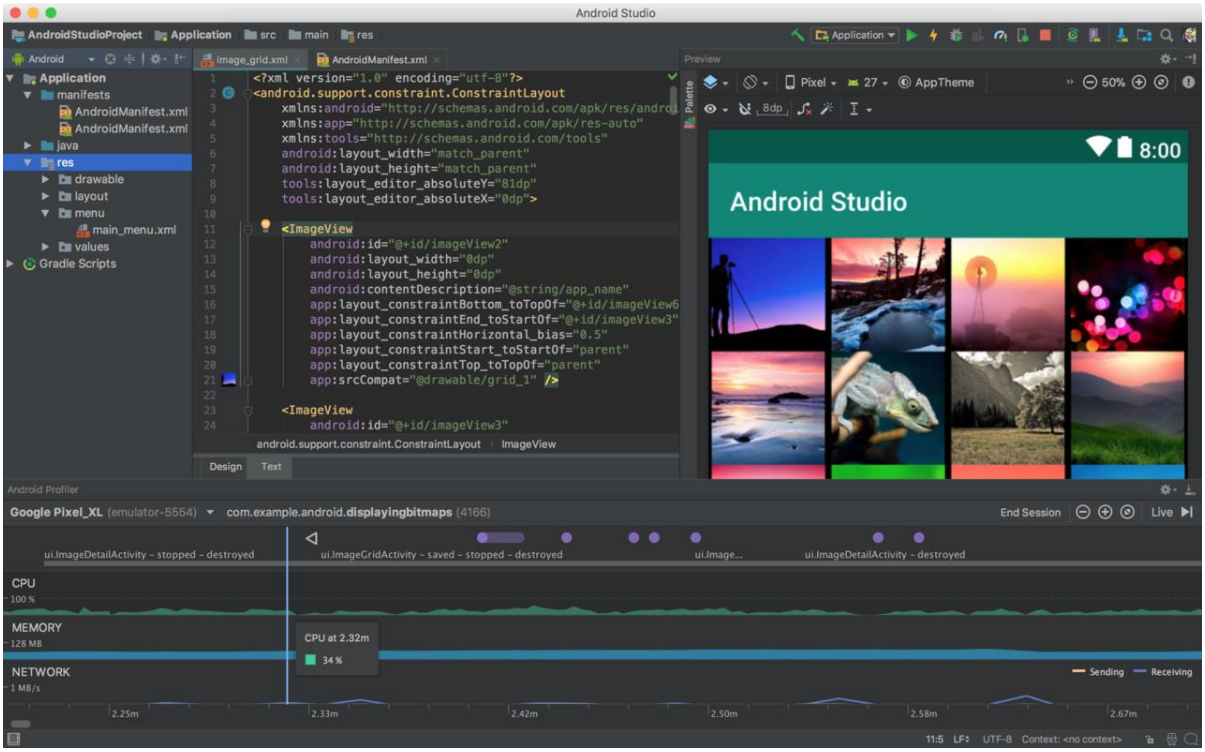

Рисунок 3.1 – Інтерфейс Android Studio

Описане вище програмне середовище використовувалось для розробки двох клієнстських додатків для адміністратора заходу і для відвідувача заходу.

Додаткові переваги обраного програмного середовища наступні:

- живі макети (layout) — подання (rendering) програми в реальному часі;
- консоль: підказки щодо оптимізації, допомога з перекладом, стеження за

напрямком, агітації та акції;

- бета релізів та покрокові релізи;
- Android-орієнтований рефакторинг коду та швидкі поправки коду;
- утиліти Lint для продуктивності, юзабіліті, сумісності версій та інших проблем;
- використання ProGuard та підписів до програм;
- шаблони для створення розширених Android компонентів;
- багатий редактор макетів, що дозволяє користувачам перетягнути і покласти компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети на різних конфігураціях екранів [7].

Для написання трігерів для серверної частини використовувався текстовий редактор Sublime Text 3. Користувацький інтерфейс цього редатору представлений на Рисунку 3.2.

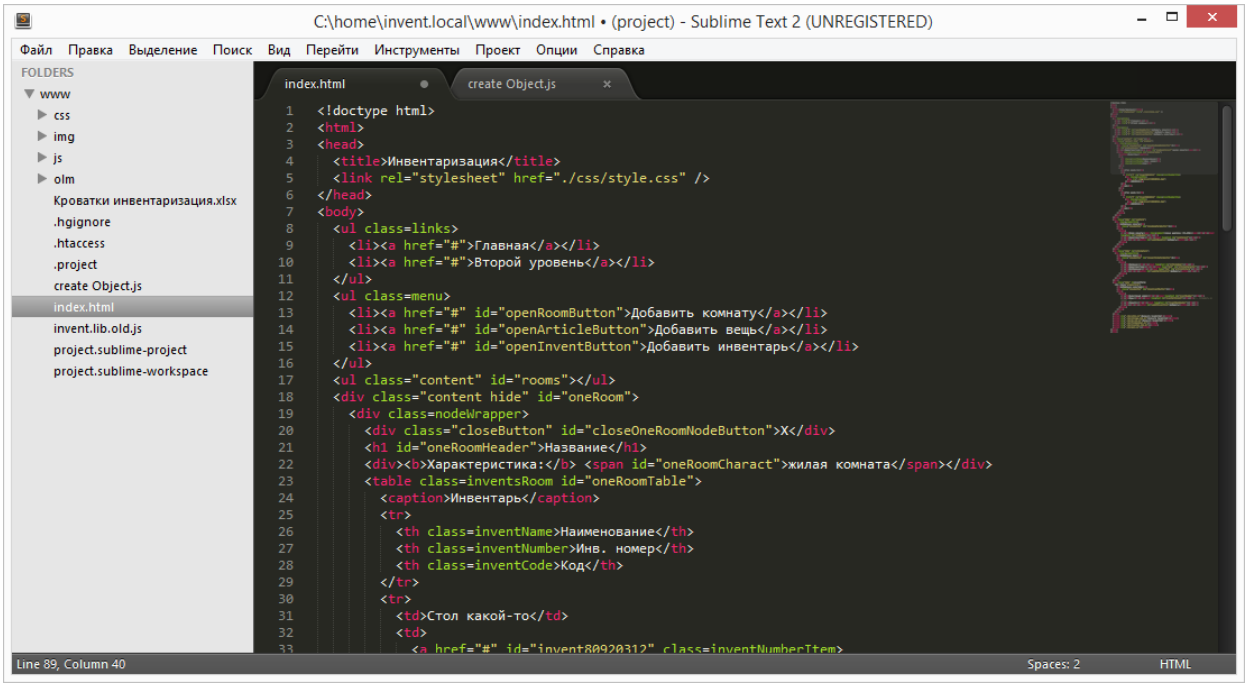

Рисунок 3.2 – Користувацький інтерфейс Sublime Text 3

- Goto Anything - швидка навігація до файлів, символів або рядків
- Палітра команд використовує адаптивне узгодження для швидкого виклику клавіатури довільних команд
- Одночасне редагування: одночасно виконайте такі ж інтерактивні зміни в

декількох вибраних областях

- API плагіна на основі Python
- Широке налаштування за допомогою файлів налаштувань JSON, включаючи налаштування, специфічні для конкретного проекту та платформи
- Крос-платформенний (Windows, MacOS, Linux) і підтримуючі плагіни для крос-платформних
- Сумісність з багатьма граматиками мови від TextMate

3.2. Вибір архітектури клієнстських додатків

Вибір архітектури це дуже важлива тема, тому що колись, налагоджуючи величезний клас з десятками різних методів і властивостей, можна опинитися не в змозі знайти та виправити в ньому помилки. Природно, такий клас важко тримати в голові як єдине ціле, тому завжди будуть втрачатися з поля зору ті чи інші важливі деталі. Якщо вже така ситуація склалась, то досить імовірно, що:

- цей клас – спадкоємець Activity;
- дані зберігаються прямо в Activity;
- View-підкласи ні за що не відповідають.

Визначимо ознаки гарної архітектури додатку:

- Збалансований розподіл обов'язків між сутностями з жорсткими ролями. Розподіл зменшує навантаження на мозок, коли ми намагаємося з'ясувати, як працює та чи інша сутність. Чим більше людина розвивається, то тим краще мозок буде адаптуватися до розуміння складних концепцій. Але у всього є межа, і вона досягається досить швидко. Таким чином, найпростіший спосіб зменшити складність полягає в розподілі обов'язків між декількома сутностями за принципом єдиної відповідальності.

– Можливість тестування архітектури визначає, наскільки легко можна написати юніт-тести, а найчастіше – чи можна їх написати взагалі. А чи варто тестувати взагалі? Як правило, це не питання для тих, у кого не пройшли юніт-тести після додавання нового функціоналу або після рефакторингу якихось тонкощів класу. Це означає, що тести вберегли розробників від виявлення проблеми після випуску нової версії.

– Простота використання та низька вартість обслуговування. Варто відзначити, що кращий код – це код, який ніколи не був написаний. Чим менше коду, тим менше помилок. Тому бажання писати менше коду зовсім не говорить про те, що розробник лінується. А вибираючи найрозумніше рішення, потрібно завжди враховувати вартість його підтримки.

Для реалізації клієнтського застосунку було використано фреймворк, в основу якого було покладено шаблон проектування MVVM (рисунок 3.2).

MVVM полегшує відокремлення розробки інтерфейсу від розробки бізнес логіки (бек-енд), відомої як модель. Модель представлення це частина, яка відповідає за перетворення даних, їх подальшої підтримки та використання [4]. З цієї точки зору, модель представлення скоріше схожа на модель, ніж на представлення і виконує більшість, якщо не усю, логіку відображення даних. Модель представлення може реалізовувати медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримані представленням.

MVVM зручно використовувати замість класичного архітектурного паттерну MVC та йому подібних у тих випадках, коли на платформі, де ведеться розробка, присутнє “зв'язування даних”.

Рисунок 3.3 – Схема роботи MVVM шаблону

Шаблон MVVM ділиться на три частини:

– Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку;

– Представлення (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо;

– Модель вигляду (ViewModel, що означає “Model of View”) з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись.

3.3. Опис інструментів розробки

Програмний комплекс побудований за на основі платформи для розробки мобільних додатків Firebase. Головним критерієм при виборі платформи була орієнтованість на використання додатку на мобільних платформах, і так як Firebase задовольняє більшість вимог і функцій які необхідні для розробки системи орієнтованої на мобільні пристрої було обрано саме цю платформу. На Рисунку 3.4 зображена план-схема завантаження картинки на сервер Firebase.

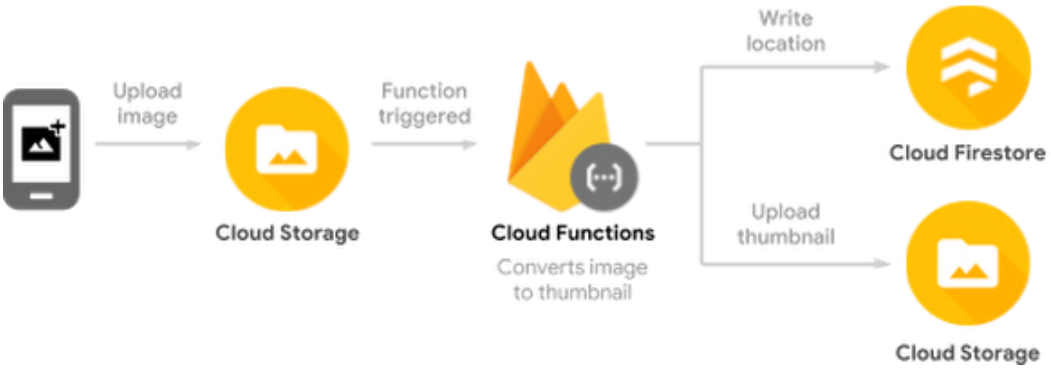

Рисунок 3.4 – Завантаження картинки на сервер Firebase

Для клієнтського рівня було використано такий набір технологій: мова програмування Kotlin та Java, фреймворк для побудови android-додатків Android

SDK, а також такі додаткові бібліотеки як Room, Android Jetpack, Koin, Android Arch. В якості IDE використовувалась Android Studio.

Kotlin – статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains.

Kotlin прекрасно підходить для розробки додатків для Android, приносячи всі переваги сучасної мови на платформу Android без введення нових обмежень:

– Сумісність: Kotlin повністю сумісна з JDK 6, гарантуючи, що програми Kotlin можуть працювати на старих пристроях Android без проблем;

– Продуктивність: Програма Kotlin працює так само швидко, як еквівалентна на Java, завдяки дуже подібній структурі байт-коду. З підтримкою вбудованих функцій, код, що використовує лямбди, часто працює навіть швидше, ніж той самий код, написаний на Java;

– Час компіляції: Kotlin підтримує ефективну інкрементальну компіляцію, тому, хоча є додаткові накладні витрати для чистих збірок, інкрементні збірки зазвичай швидші ніж з Java.

3.4. Обґрунтування вибору програмної реалізації

При проектуванні системи було вивчено та проаналізовано предметну область та вимоги замовника. Після ретельного аналізу було вирішено розроблювати програмний продукт, який заснований на мобільних-технологіях для використання за допомогою мобільних пристроїв.

Технології, які використовуються на сервері, були обрані за принципом зручності у використанні, відкритості вихідних кодів, актуальності в наш час та можливістю виконання на будь-якій операційній системі. Платформа Firebase надає змогу створювати програми на мовах, зокрема мова Kotlin, на будь-якій операційній системі. Фреймворк Android SDK надає величезні можливості для створення мобільних-сервісів будь-якої складності, забезпечуючи при цьому велику швидкодію

та надійність. Також перевагою використання цих технологій є те, що їх можна запустити у емуляторах контейнерах та побудувати на цьому швидке та надійне розгортання тестування з подальшим масштабуванням без створення при цьому спеціальної архітектури програмного забезпечення. Для доступу до бази даних було використано ORM Room через його зручність та універсальність. За допомогою цього фреймворку можливо абстрагуватися від конкретної реалізації бази даних і створити відповідні моделі даних. Це надає змогу зосередити всю увагу на створенні якісної та протестованої бізнес-логіки.

Базою даних було обрано NoSQL через те, що це проект з відкритим вихідним кодом і надзвичайно великою підтримкою з боку розробників. Також важливою властивістю є велика швидкість роботи, надійність та кількість вбудованих можливостей. Цю базу даних можливо розгорнути на будь-якій сучасній операційній системі.

Дані технології в сукупності дають змогу збудувати якісний та надійний продукт, який захищений від патентних позовів з боку розробників, бо всі ці технології покриті ліцензіями, які виключають таку можливість і надають доступ до вихідних кодів даних проектів.

3.5. Висновки до розділу

В даному розділі було розглянуто інструменти та технології неохідні для створення системи гнучкого управління розкладом заходів.

Було розглянуто Android Framework та функціонал який він надає. Було детально розглянуто середовище розробки клієнтських додатків Android Studio та його допоміжні інструменти.

Наведено аргументацію по вибору архітектури для побудови додатку MVVM, та порівняння з класичним MVC та MVP. Було наведено переваги щодо використання

мови програмування Kotlin як основної мови для розробки клієнтських додатків.

У цьому розділі було обрано архітектурну платформу для побудови розроблюваної системи. Вирішено будувати систему на базі платформи Firebase. Обгрунтовано використання саме цієї платформи через деталі реалізації клієнтських додатків і загальну орієнтованість системи на мобільні платформи. Розглянуто переваги бази даних NoSQL.

Окрім того було розглянуто середовище для розробки трігерів на базі архітектурної платформи Firebase. Обрано такі інструменти як Node.js, та вирішено зробити реалізацію серверних трігерів на мові програмування JavaScript.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ГНУЧКОГО УПРАВЛІННЯ ЗАХОДАМИ

Аналізуючи поставлену задачу та методи її вирішення, було вирішено розроблювати програмний комплекс на основі мобільних-технологій. Головною перевагою мобільного-застосунку перед іншими варіантами є його зручність і простота у використанні, швидкість роботи, а також доступність, ви можете робити всі дії на ходу. У мобільних пристроях однією із переваг є зручність поширення програм, в офіційних магазинах додатків.

Система гнучкого управління розкладом буде складатися з чотирьох модулів, кожен з яких буде розбиватися на різну кількість функціональних підблоків. Головним модулем буде кабінет користувача, який одночасно буде елементом компонування системи. Така структура, дозволить легко та інтуїтивно користуватися системою за рахунок того, що модулі розташовуються у порядку, який зазвичай використовують для вирішення схожих задач.

4.1. Опис функціональності системи для відвідувача заходу

Програмний застосунок для виділення скелету та розпізнавання пози людини на відеопотоці містить у собі одного головного актора – користувач системи.
На рисунку 4.1 представлена діаграма прецедентів, яка описує функції та дії відвідувача заходу у системі.

Відвідувач заходу має наступні функції:

– авторизація, для того щоб зайти в додаток користувач має спочатку знайти потрібний захід за його назвою, ввівши його в тестове поле, а потім обравши потрібний серед переліченого;

– сканування квитка. Після того, як захід обрано користувач повинен відсканувати свій квиток, в даному випадку це QR код;

– перегляд розкладу;

– перегляд деталей лекції в першу чергу включає в себе обрання потрібної лекції для перегляду;

– перегляд деталей обраної лекції, що включає в себе назву, місце проведення, імя викладача;

– про зміни в розкладі користувач дізнається з шторки сповіщень, змінюватись може дата проведення або місце;

– переглядання сповіщення включає в себе можливість відмітити сповіщення як прочитане;

– показати відсканований квиток при вході до заходу.

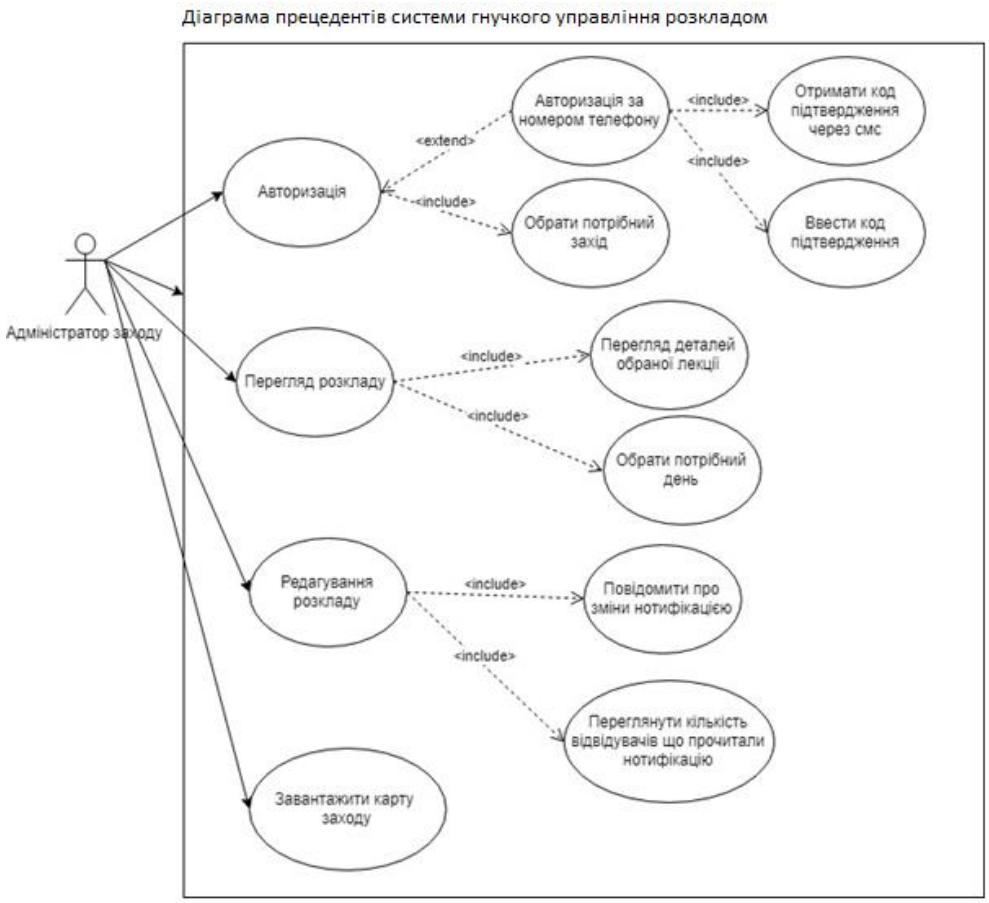

Рисунок 4.1 — Діаграма прецедентів системи гнучкого управління рокзладом

4.2. Опис функціональності системи для адміністратора заходу

Далі ми будемо розглядати роль аміністратора заходу в системі. Ця роль є найголовнішою в такій системі. Роль адміністратора поляє в тому щоб забезпечити всіх відвідувачів заходу актуальною інформацією про захід, про розклад, місце проведення.

На Рисунку 4.2 ми можемо бачити діаграму прецендентів для ролі адміністратор заходу.

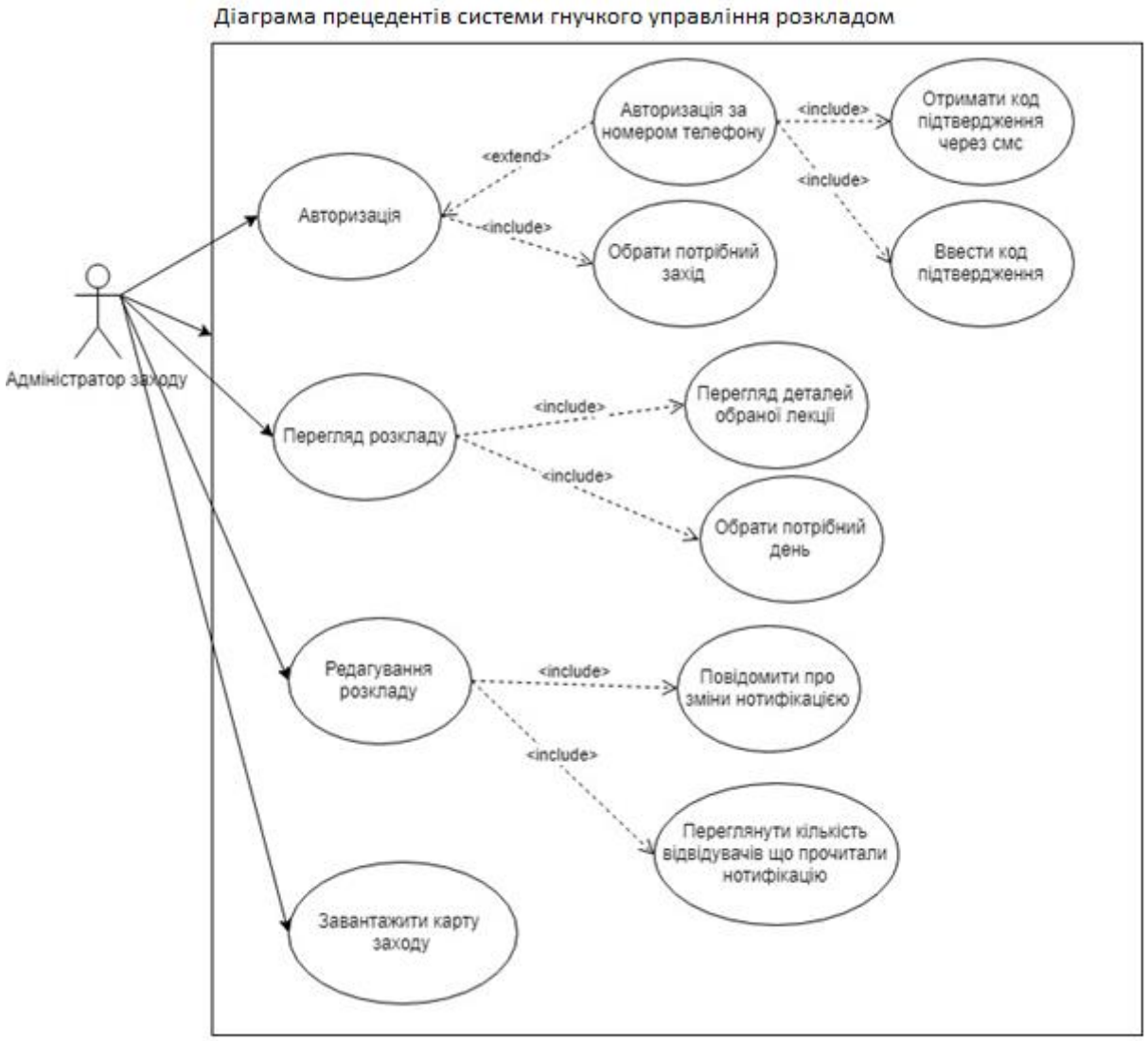

Рисунок 4.2 – Діаграма прецедентів системи гнучкого управління розкладом

На даній діаграмі ми можемо бачити детальніше ролі адміністратора системи. Його функції наступні:

– авторизація в системі, що має в собі кілька додаткових кроків;

– обрати потрібний захід, із запропонованого списку здійсьнюючи пошук за назвою

– пройти аунтенфікацію за допомогою номеру телефону, якщо теж включає в себе кілька додаткових пунктів;

– отримати смс повідомлення з одноразовим кодом;

– ввести отриманий код підтвердження;

– перегляд розкладу;

– обрання потрібної лекції для перегляду;

– перегляд деталей обраної лекції;

– редагування розкладу, що здійсьнюється на окремому екрані та включає в себе зміну таких полів як назва заходу, місце його проведення, зміну викладача чи зміну графіку;

– сповістити всіх відвідувачів лекції про зміни в будь-якому полі;

– переглянути кількість користувачів, що прочитали нотифікацію, віносно загальної кількості користувачі, які відвідають захід та зареєструвались в системі;

– завантаження фотографії, що ілюструє план-схему заходу.

4.3. Опис таблиць бази даних

Перш за все потрібно розібратись в необхідних даних, в їх оптимальній структурі та зв’язками між ними. А саме як вони влаштовані на рівні бази даних. Для доступу до даних із бази даних для кожної таблиці створюється звичайний клас на мові Kotlin з публічними властивостям, який є об’єктним відображенням таблиці в базі даних.

DbContext забезпечує об'єктно-орієнтований інтерфейс для доступу і маніпулювання даними, що зберігаються в базах даних. Клас DbSet відповідає таблиці в базі даних, поле моделі являє собою значення окремого стовпця рядка.

Також використання об'єктно-реляційної проекції дозволяє зручно отримувати данні з таблиць, які мають зв'язок “багато-до-багатьох” та “один-до-багатьох”.

База даних системи реалізована за допомогою ORM Room. Розглянемо більш детально структури кожної із таблиць системи гнчкого управління рокзладом.

Детальна інформація про їх структури (ім’я, тип і розмір поля, опис поля) приведена у таблицях 4.1 — 4.7.

Таблиця 4.1. Структура таблиці “Адміністратор”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
UserName	nvarchar(100)	Імя користувача
Email	nvarchar(100)	Адреса електронної пошти
Phone	nvarchar(100)	Номер мобільного телефону

Таблиця 4.2. Структура таблиці “Користувач”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
UserName	nvarchar(100)	Імя користувача
QRCode	nvarchar(100)	Номер квитка
Phone	nvarchar(100)	Номер мобільного телефону
EventID	nvarchar(100)	ІД відуваного заходу

Ці таблиці представляють собою коирстувачів системи: адмінстратора заходу та відвідувача заходу, але оскільки способи авторизації для них в системі відрізняються то деякі поля, наприклад поле Phone для відвідувача заходу є

опціональним.

Таблиця 4.3. Структура таблиці “Захід”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
Name	nvarchar(255)	Імя
AdminID	nvarchar(450)	Ід адмінстратора
ImageUrl	nvarchar(450)	Посилання на картинку пами заходу
Postponed	bool	Перенесена чи ні
VisitorCount	Integer	Кількість зареєстрованих відвідувачів

Таблиця 4.4. Структура таблиці “Зареєстровані користувачі”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
EventId	nvarchar(255)	Ід заходу
UserId	nvarchar(450)	Ід юзера

Таблиця 4.4 слугує допоміжною таблицею між заходами та відвідувачами заходів. Це необхідно для того, щоб розуміти хто із відвідувачів підписний на той чи інший захід, щоб в подальшому розібрати цим користувачам сповіщення про зміни.

Таблиця 4.5. Структура таблиці “Лекція”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
EventId	nvarchar(255)	Ід заходу
Place	nvarchar(450)	Локація
Time	nvarchar(450)	Час
TeacherID	nvarchar(450)	Id лектора

Таблиця 4.6. Структура таблиці “Сповіщення”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
EventId	nvarchar(255)	Ід заходу
LessonId	nvarchar(450)	Ід лекції
Title	nvarchar(450)	Заголовок сповіщення
Body	nvarchar(450)	Текст сповіщення

Таблиця 4.7. Структура таблиці “Прочитані сповіщення”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
NotificationID	nvarchar(255)	Ід сповіщення
UserId	nvarchar(450)	Ід відвідувача
ReadStatus	bool	Прочитано

Таблиця 4.7 слугує для того, щоб відобразити кількість користувачів, що прочитали сповіщення, загальна кількість користувачі, що відвідають захід береться із Таблиці 4.3 де зберігаються заходи, із поля VisitorsCount.

4.4. Програмна реалізація взаємодії з базою даних

Після розробки схеми бази даних було розпочато розробку функціоналу для роботи бази на основі раніше описаних схем. В якості ORM було використано бібліотеку, створену компанією Google спеціально для додатків, як будуть працювати в операційній системі Android. Розглянемо класи які були створені у ході розробки:

AppDatabase

Даний клас відповідає за роботу з базою даних, а саме автоматично створює файл бази даних, генерує таблиці потрібні для правильної роботи системи. Також обєкт AppDatabase надає DAO для роботи з певними таблицями бази даних. Цей клас дає абстракцію за всією базою даних. Для коректної генерації потрібно вказати анотації Database список моделей, що повинні бути в згенерованій базі даних, а також версії, для коректного версіювання. Також необхідно описати методи для отримання DAO об’єктів, що необхідні для роботи з даними.

UserDao

Цей клас несе відповідальність за роботу з таблицею користувачів в базі даних. Він містить в собі код для коректного додання нового користувача в таблицю та його видалення.

AdministatorDao

Схожий на попередній клас тільки описує методи роботи бази даних із адміністратором заходу.

EventDao

Даний клас робить можливим отримання списку заходів їхнього адміністратора. А також дозволяє записувати нові заходи, змінювати існуючі чи міняти адміністратора..

Також були створені класи сутностей які реалізовують створенні в минулому

підрозділі схеми таблиць. Для їх створення було використано спеціальні анотації бібліотеки Room, а саме @Entity. При компіляції Room сама створить таблиці по написаним моделям сутностей позначених даною анотацією.

В ході створення класів сутностей були створені такі класи:

- Event – клас описує модель таблиці заходу описану раніше;
- Lesson – клас описує модель таблиці лекції;
- User – клас описує модель таблиці користувача системи;
- Notification – клас описує модель таблиці списку сповіщень описану раніше.

4.5. Опис архітектурної платформи

Для розробки даної системи гнучкого управління графіком було використано таку платформу як Firebase. Дана платформа орієнтована на те, що клієнтнськими додатками будуть мобільні додатки, але також можливо написати клієнт і на web-платформу.

Firebase надає інструменти для зручного перегляду бази даних, налаштування ролей та прав для бази даних.

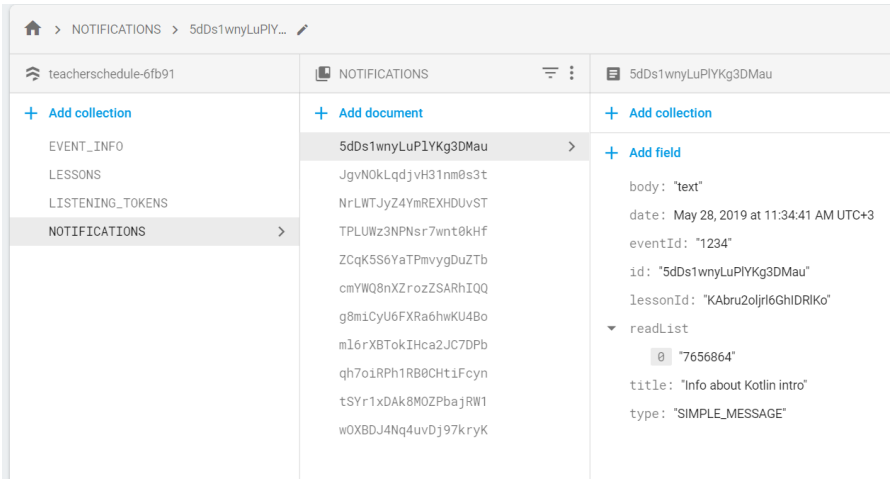

Рисунок 4.3 – Інтерфейс бази даних на платформі Firebase

Окрім бази даних Firebase надає нам в користування і хмарне середовище в якому можна зберігати файли, що завантажує клієнт. Користувацький інтерфейс головної адмін-панелі платформи Firebase ви можете бачити на Рисунку 4.4

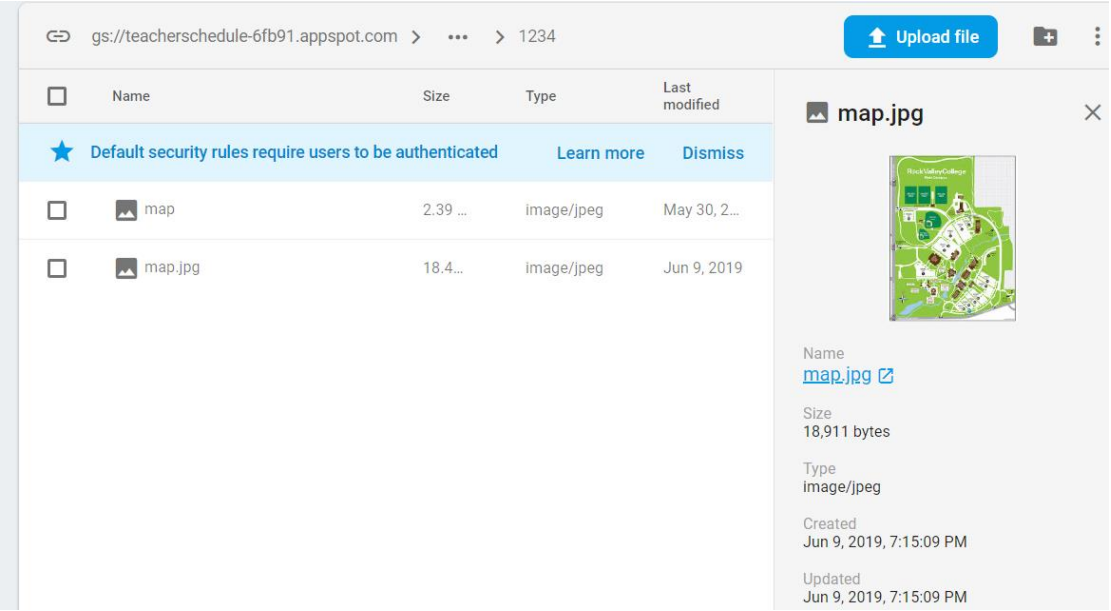

Рисунок 4.4 – Адмін-панель хмарного середовища платформи Firebase

Зберігання картинок в середовище здійсьнюється за наступною схемою: Images/{eventId}/map.jpg де змінна eventId дорівнює ід заходу. Для кожного заходу адміністратор може завантажити одну план-схему заходу.

4.6. Висновки до розділу

Таким чином було реалізовано і протестовану систему гнучкого управління розкладом заходів, що складається із 3 частин: серверна частина побудована на базі архітектурної платформи Firebase, клієнтських додаток для відвідувача заходу і клієнтський додаток для адміністратора заходу .

Серверна частина реалізована за допомогою мови програмування JavaScript з використання технології Node.js.

Клієнтські додатки реалізовані за допомогою середовища розробки Android Studio на основі Android SDK. Написані додатки на офіційній мові програмування для платформи Android Kotlin. При розробці бази даних для кешування на стороні клієнта було використано ORM Room, що в свою чергу заснований на базі даних SQLite.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблена програмний комплекс розроблений з використанням мобільних-технологій і тому може адмініструватись з браузера, а також працює на системі Android починаючи з версії API 16 Android 4.1 або вище.

5.1. Інсталяція та системні вимоги

Адміністратор події повинен мати в своєму розпорядженні мобільний пристрій на базі ОС Android версії 4.1 або вище, на даний момент таких девайсів на ринку більше ніж 98% що дозволяє в повній мірі покрити аудиторію користувачів.

5.2. Інструкція з використання програмного продукту

При вході на клієнтський додаток, користувачеві необхідно вказати назву події. На рисунку 5.1 зображена форма входу.

При вході в додаток користувачеві буде панель з управлінням вкладками в нижній частині екрану, на першому екрані знаходить головне вікно програми розкладу, в шапці ми можемо побачити вкладки які відповідають за дати, показано 7 днів поточного тижня, обрана вкладка виділяється білим кольором, інші вкладення залишаються стандартними, сьогоднішній день позначається на вкладенні невеликою жовтою круглою позначкою. Коли бажаний день обрано на основному екрані з’являється розклад на цей день, з кольоровими помітками які сигналізують про порядковість занять, на кожному із пунктів також присутня кнопка для того, щоб

перенести заняття. Це зображено на Рисунку 5.2

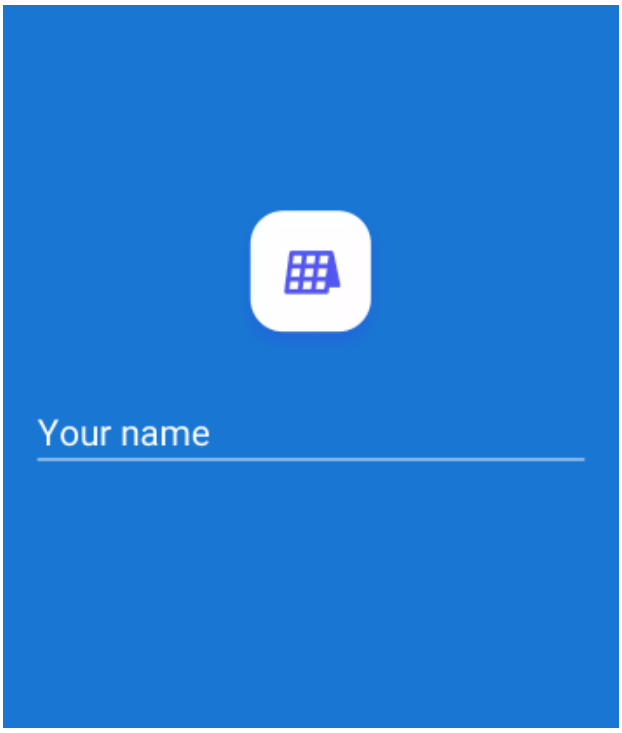

Рисунок 5.1 — Форма авторизації

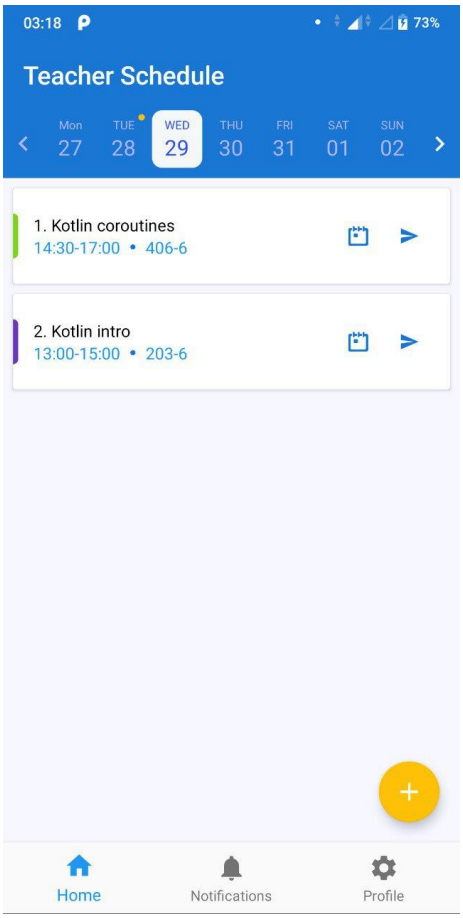

Рисунок 5.2 — Головне вікно програми

Якщо користувач нажимає на кнопку відкласти заняття що знаходиться зправа від основної інформації про лекцію йому відкриється модальне вікно з вибором опцій для того щоб повідомити про перенесення заняття, доступні наступні опції: Вікласти на 5, 15, 30 хвилин або відмінити заняття взагалі і також можна вказати коментар щодо причини перенесення чи відміни заняття (Рисунок 5.3)

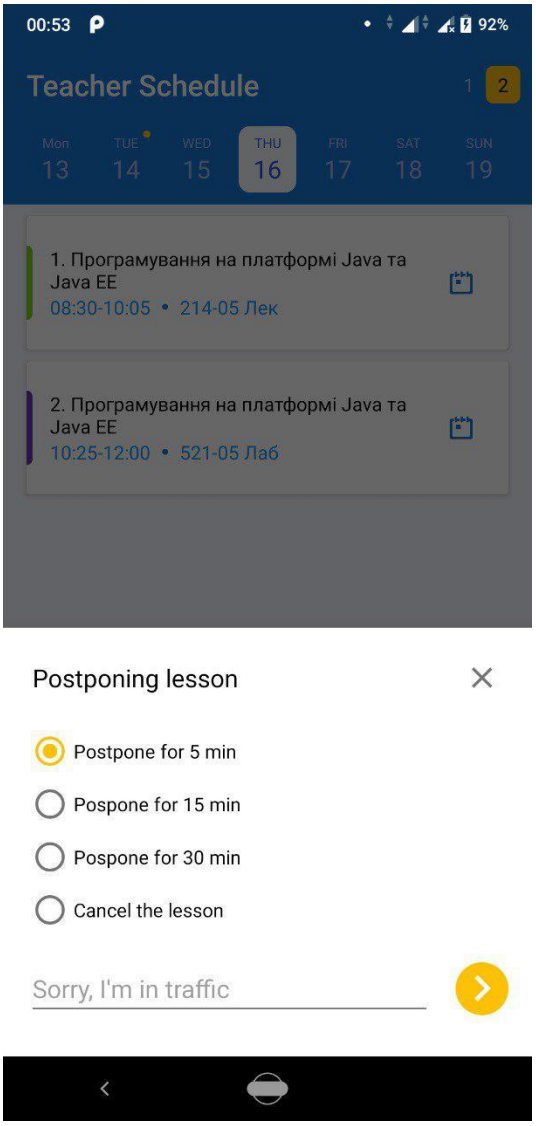

Рисунок 5.3 — Відкладення заняття

В такій системі важливим моментом є забезпечення безпеки, тому для того щоб будь-хто не зміг зайти в додаток і змінити розклад, налагоджена система аутенфікації користувача через номер телефону (Рисунок 5.4). В шапці виводиться імя користувача а знизу пунктом його номер телефону в якого мое бути 2 стани: підтверджено або не

підтверджено, на Рисунку 5.4 зображено стан коли користувачеві потрібно підтвердити свій номер телефону

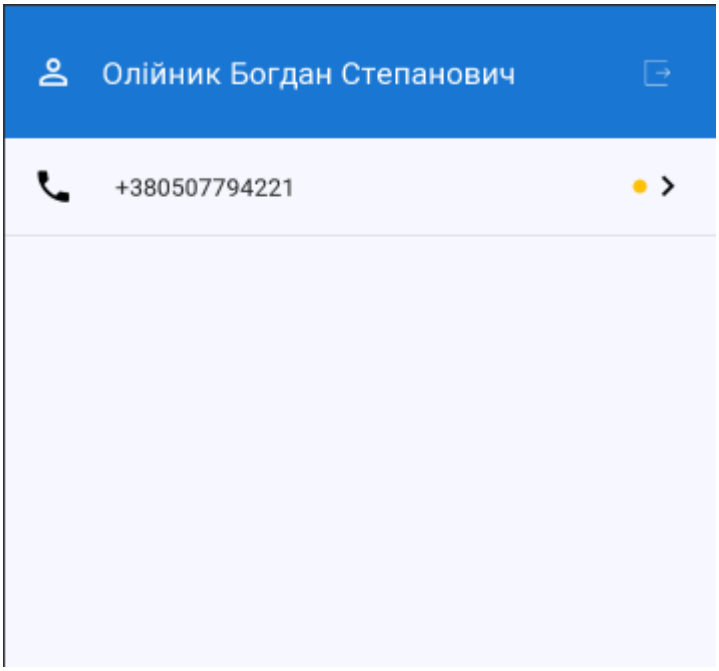

Рисунок 5.4 — Меню користувача

При нажатті на поле з номер телефону користувач пересилається на вікно підтвердження номеру, яке складається з наступних екранів: вікно вводу номеру телефону, яке є передзаповненим (Рисунок 5.5) та екран вводу підтверджуючого смс (Рисунок 5.6). Після такої аунтефікації користувач підтверджує що він є адміністратором цієї події і зможе змінювати розклад, вносити правки.

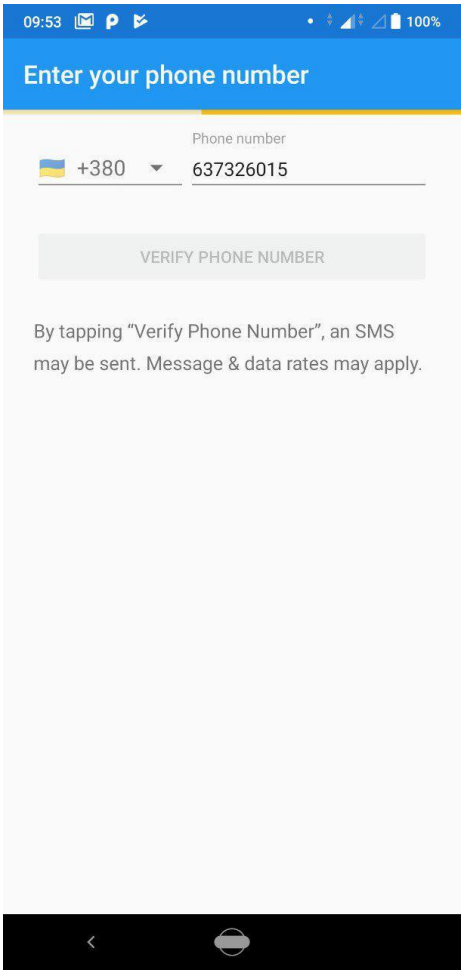

Рисунок 5.5 — Екран для вводу номеру телефону

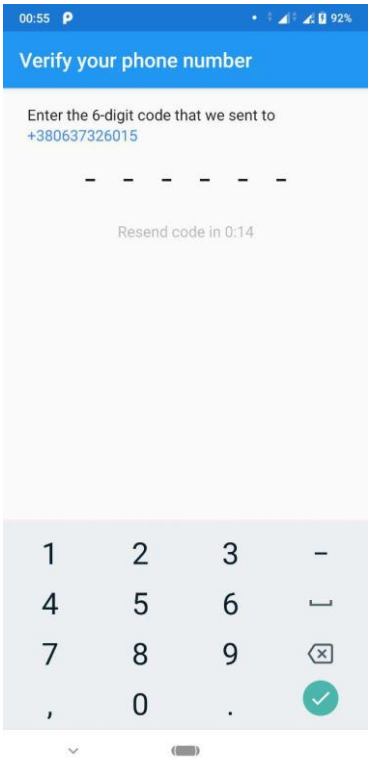

Рисунок 5.6 — Екран для вводу номеру телефону

На наступному Рисунку 5.7 ми можемо бачити як виглядає середній пункт меню в клієнтському додатку для адміністратора. Тут виводиться список сповіщень, їх заголовок, текст і також окремо виділено поле яка позначає скільки відвідувачів заходу прочитало дане сповіщення.

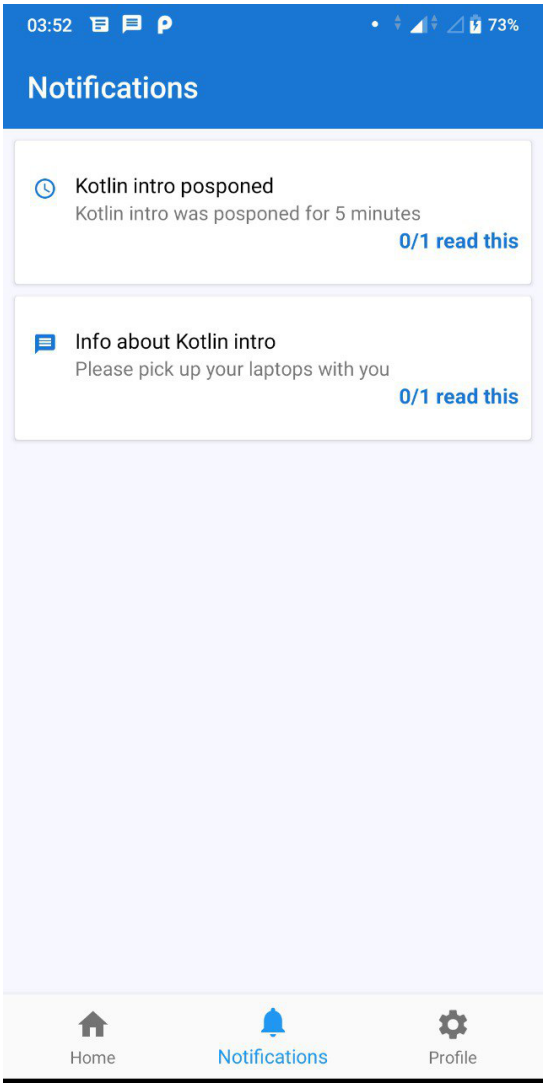

Рисунок 5.7 – Вікно сповіщень

5.3. Висновки до розділу

В процесі розробки системи системи для гнучкого управління розкладом заходів було розроблено та впроваджено зручний користувацький інтерфейс для застосунків для адміністратора і відвідувача заходу.

Дизайн розроблених застосунків відповідає всім правилам яких повинні дотримуватись розробники мобільних застосунків на платформі Android. А саме Material Guidelines. Його стилю відповідають фрагменти екранів а також модальні вікна програм

ВИСНОВКИ

В даному дипломному проектів було досліджено та проаналізовано доцільність створення системи для гнучкого управління розкладом заходів.

Для зручного представлення даних користувачу було створено мобільні додатоки для пристроїв, що керуються операційною системою Android. Розроблений програмний продукт дозволяє дозволяє головному адміністратору створювати події та призначати адміністраторів для кожної з них, а останнім надає можливість гнучкого управління розкладом забезпечую при цьому надійний рівень безпеки.
Також було представлено повну інструкцію користувача для системи, яка зручно та доступно описує послідовність дій при роботі з системою

Проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір створення програмної системи, орієнтованої на використання за допомогою мобільних пристроїв. Це системі ряд таких переваг як гнучкість, зручність у використанні, вирішує проблему з поширенням ПО.

Користувачами системи є адміністратори подій, головний адміністратор а також відвідувачі заходу. Захід розуміється як тренінг або будь-який інший захід підвищення кваліфікації на якому присутній графік лекцій, що може змінюватись.

Загалом в дипломному проекті повністю вирішена поставлена задача, розглянуто всі супутні завдання та вирішено проблеми, які виникали в ході виконання завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Леонард Річардсон, Майкл Амундсен // RESTful Web APIs: Services for a Changing World. – 2013.
2. Android Framework [електронний ресурс]. – 2017. – Режим доступу: https://www.quora.com/What-is-meant-by-Android-framework
3. Мартин Фаулер // GUI Architectures. - 2006
4. Kotlin [електронний ресурс]. – 2018. – Режим доступу: https://kotlinlang.org/
5. Android Studio [електронний ресурс]. – 2019 – Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
6. REST [електронний ресурс]. – 2019 – Режим доступу: https://uk.wikipedia.org/wiki/REST
7. Sublime Text 3 [електронний ресурс]. – 2019 – Режим доступу: https://uk.wikipedia.org/wiki/Sublime_Text
8. Дубова Н. SOA: подходы к реализации // Открытые системы. — 2004. — № 6. — С. 37–41.
9. The MIT License (MIT) [Електронний ресурс] – Режим доступу до ресурсу: https://opensource.org/licenses/MIT
10. Причины возникновения проблем совместимости программного обеспечения [Електронний ресурс] – Режим доступу до ресурсу: http://helpiks.org/7- 46217.html
11. Рик Роджерс — Android. Разработка приложений / Роджерс Рик. — М.: Эком, 2010. — 130 с.
12. Цехнер Марио — Программирование под Android / Марио Цехнер. — М.: Питер, 2012. — 688 с.
13. Martin Fowler — GUI Architectures. Часть 1 [Електронний ресурс]. — 2009. — Режим доступу: http://www.rusdoc.ru/articles/18358/.
14. Martin Fowler — GUI Architectures. Часть 2 [Електронний ресурс]. — 2009. — Режим доступу: https://habrahabr.ru/post/53536/.

15. Майер Р. — Android 2. Программирование приложений для планшетных компьютеров и смартфонов [Електронний ресурс]. — 2011. — Режим доступу: https://www.ozon.ru/context/detail/id/6752687/.
16. П. Дейтел — Android для программистов. Создаем приложения / П.Дейтел — М.: Питер, 2012. — 560 с.

ДОДАТОК А

Система гнучкого управління розкладом заходів

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51150_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51150_19Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51150_19Б 12-1	App-client.apk	Клієнт для відвідувача
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51150_19Б 12-2	App-admin.apk	Клієн для адмін.

ДОДАТОК Б

Система гнучкого управління розкладом заходів

Лістинг програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51150_19Б

Аркушів 6

Київ 2019

CreateEditLessonActivity.kt

```
package androks.kpi.schedule.rozklad.ui.pages.createEditLesson

import android.app.Activity
import android.app.DatePickerDialog
import android.app.TimePickerDialog
import android.os.Build
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.Observer
import androks.kpi.schedule.rozklad.R
import androks.kpi.schedule.rozklad.core.remote.firebase.model.Lesson
import androks.kpi.schedule.rozklad.utils.TextWatcher
import androks.kpi.schedule.rozklad.utils.toast
import kotlinx.android.synthetic.main.activity_create_edit_lesson.*
import org.koin.android.ext.android.inject
import java.text.SimpleDateFormat
import java.util.Calendar
import java.util.Date
import java.util.Locale

class CreateEditLessonActivity : AppCompatActivity() {

    companion object {
        const val LESSON_KEY = "LESSON_KEY"
    }

    private val createEditLessonViewModel: CreateEditLessonViewModel by inject()

    private val lesson: Lesson? by lazy {
        intent?.extras?.getParcelable<Lesson>(LESSON_KEY)
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_create_edit_lesson)
        initToolbar()
        subscribeOnValidation()
        createEditLessonViewModel.init(lesson)
        lesson?.let { filledInOldValues(it) }
        implementOnClickListeners()
        createEditLessonViewModel.lessonUpdatedLiveData.observe(this, Observer {
            if (it) {
```

```
            setResult(Activity.RESULT_OK)
        }
        finish()
    })
}

private fun initToolbar() {
    setSupportActionBar(toolbar)
    toolbar.title = if(lesson != null) "Editing lesson" else "Creating lesson"
}

private fun subscribeOnValidation() {
    createEditLessonViewModel.lessonFieldsFilledInCorrectlyLiveData.observe(this,
Observer {
        saveButton.isEnabled = it
    })
}

private fun filledInOldValues(lesson: Lesson) {
    lessonNameEditText.setText(lesson.lessonName)
    lessonLocationEditText.setText(lesson.locationInText)
    val cal = Calendar.getInstance()
    cal.time = Date()
    showDate(lesson.timeStart)
    showStartTime(lesson.timeStart)
    showEndTime(lesson.timeEnd)
}

private fun showDate(calendar: Date) {
    lessonDateTextView.text = SimpleDateFormat("dd.MM.yyyy",
Locale.getDefault()).format(calendar)
}

private fun showDate(year: Int, month: Int, dayOfMonth: Int) {
    showDate(Calendar.getInstance().apply { set(year, month, dayOfMonth) }.time)
}

private fun getDateForHourAndMinute(hour: Int, minute: Int): Date {
    return Calendar.getInstance().apply { set(0, 0, 0, hour, minute) }.time
}

private fun showStartTime(time: Date) {
    lessonTimeFromTextView.text = SimpleDateFormat("HH:mm",
Locale.getDefault()).format(time)
```

```
    }

    private fun showEndTime(time: Date) {
        lessonTimeToTextView.text = SimpleDateFormat("HH:mm",
Locale.getDefault()).format(time)
    }

    private fun implementOnClickListeners() {
        pickLessonDateLinearLayout.setOnClickListener {
            val timeStartOrNow = Calendar.getInstance().apply {
                if (lesson != null) {
                    time = lesson!!.timeStart
                }
                createEditLessonViewModel.newLessonDate?.let {
                    set(it.get(Calendar.YEAR), it.get(Calendar.MONTH),
it.get(Calendar.DAY_OF_MONTH))
                    return@apply
                }
            }
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
                val datePickDialog = DatePickerDialog(
                    this,
                    DatePickerDialog.OnDateSetListener { _, year, month, dayOfMonth ->
                        createEditLessonViewModel.onDatePicked(year, month, dayOfMonth)
                        showDate(year, month, dayOfMonth)
                    }, timeStartOrNow.get(Calendar.YEAR),
timeStartOrNow.get(Calendar.MONTH),
timeStartOrNow.get(Calendar.DAY_OF_MONTH)
                )
                datePickDialog.datePicker.minDate = Date().time
                datePickDialog.datePicker.maxDate = Calendar.getInstance().apply {
                    add(Calendar.MONTH, 1)
                }.timeInMillis
                datePickDialog.show()
            } else {
                toast("Go buy new phone")
            }
        }
        lessonTimeFromTextView.setOnClickListener {
            val timeStartOrNow = Calendar.getInstance().apply {
                if (lesson != null) {
                    time = lesson!!.timeStart
                }
                createEditLessonViewModel.newLessonStartTime?.let {
```

```
                set(Calendar.HOUR_OF_DAY, it.get(Calendar.HOUR_OF_DAY))
                set(Calendar.MINUTE, it.get(Calendar.MINUTE))
                return@apply
            }
        }
        TimePickerDialog(this, TimePickerDialog.OnTimeSetListener { view, hourOfDay,
minute ->
            createEditLessonViewModel.onTimeFromPicked(hourOfDay, minute)
            showStartTime(getDateForHourAndMinute(hourOfDay, minute))
        }, timeStartOrNow.get(Calendar.HOUR_OF_DAY),
timeStartOrNow.get(Calendar.MINUTE), true).show()
    }
    lessonTimeToTextView.setOnClickListener {
        val timeEndOrNow = Calendar.getInstance().apply {
            if (lesson != null) {
                time = lesson!!.timeEnd
            }
            createEditLessonViewModel.newLessonEndTime?.let {
                set(Calendar.HOUR_OF_DAY, it.get(Calendar.HOUR_OF_DAY))
                set(Calendar.MINUTE, it.get(Calendar.MINUTE))
                return@apply
            }
        }
        TimePickerDialog(this, TimePickerDialog.OnTimeSetListener { view, hourOfDay,
minute ->
            createEditLessonViewModel.onTimeToPicked(hourOfDay, minute)
            showEndTime(getDateForHourAndMinute(hourOfDay, minute))
        }, timeEndOrNow.get(Calendar.HOUR_OF_DAY),
timeEndOrNow.get(Calendar.MINUTE), true).show()
    }
    lessonNameEditText.addTextChangedListener(TextWatcher {
createEditLessonViewModel.onLessonNameChanged(it) })
    lessonLocationEditText.addTextChangedListener(TextWatcher {
createEditLessonViewModel.onLocationChanged(it) })
    saveButton.setOnClickListener {
        createEditLessonViewModel.saveLesson()
    }
  }
}
```

Трігер для сповіщень index.js

```
const functions = require('firebase-functions');
```

```
const admin = require('firebase-admin');
const app = admin.initializeApp();
const firestore = app.firestore();
const messaging = app.messaging();

exports.myFunctionName = functions.firestore
   .document('NOTIFICATIONS/{notificationId}')
   .onCreate((snap, context) => {
        const newNotification = snap.data();
        delete newNotification.date
        delete newNotification.readList
        const eventId = newNotification.eventId;
        console.log("Notification: ", newNotification)
        return getTokens(eventId)
                .then(tokens => {
                        return messaging.sendToDevice(tokens,
{data:toString(newNotification)});
                })
});

function getTokens(eventId) {
        return firestore.collection("LISTENING_TOKENS").where('eventId', '==',
eventId).get()
                .then(querySnapshot => {
                        var listOfTokens = [];
                        querySnapshot.forEach(documentSnapshot => {
                                listOfTokens.push(documentSnapshot.get('token'))
                        });
                        console.log("Tokens to send:", listOfTokens)
                        return listOfTokens
                });
}

function toString(o) {
  Object.keys(o).forEach(k => {
    if (typeof o[k] === 'object') {
      return toString(o[k]);
    }

    o[k] = '' + o[k];
  });

  return o;
}
```

ДОДАТОК В

Система гнучкого управління розкладом заходів

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51150_19Б

Аркушів 9

Київ 2018

АНОТАЦІЯ

Додаток містить опис основні компоненти системи гнучкого управління розкладом, а саме:

- Клієнстський додаток для відвідувача заходу;
- Клієнстський додаток для адмінстратора заходу.

Взаємодія між компонентами здійсьнюється за домогою архітектурної платформи Firebase;

Додаток розроблений за допомогою мов програмування Kotlin та JavaScript у середовищі Android Studio з використанням Android SDK та інших третіх бібліотек.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основних компонентів системи гнучкого управління розкладом заходів, що виконує деякі із завдань, поставлених в розділі 1. У додатку Б міститься програмний код компоненту.

Система гнучкого управління розкладом заходів працює на базі архітектурної платформи Firebase та за допомогою двох клієнтських додатків.

Система розроблена за допомогою мов програмування Kotlin, JavaScript в середовищі розробки Android Studio з використанням Android SDK.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Додаток надає функціональність для роботи з графіком лекції певного заходу.

— створення, редагування та видалення заходів, їх адміністраторів;

— надійна авторизація адміністратора заходу;

— створення, редагування та видалення лекцій в певному заході адміністратором;

— перегляд розкладу;

— отримання сповіщень про зміни в розкладі.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, додаток складається з трьох частин:

— серверна частина з використанням Firebase;

— клієнтський додаток для відвідувача заходу;

— клієнтський додаток для адміністратора заходу.

У системі є такі обєкти як захід, до якого прикріплений адміністратор з вказаним номером телефону. За цю частину відповідає головний адмінстратор системи.

Захід містить в собі курс лекції, практичних занять тощо. Адмінстратор заходу після успішної авторизації має змогу редагувати розклад занять, змінювати місце проведення занять, завантажувати мапу заходу. При будь-якій зміні у базу даних будут записувати такі обєкти як сповіщення.

Відвідувач заходу матиме змогу переглянути свій розклад після того як обере потрібний захід та відсканує квиток. Як тільки він увійде в систему він буде отримувати сповіщення про зроблені зміни в розкладі заходу, який він обрав.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання додатку користувач повинен мати обліковий запис Google мобільний пристрій заснований на операційній системі Android версії 4.x або вище, а також підключення до мережі Інтернет.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для того щоб встановити дане програмне забезпечення необхідно скопіювати потрібний файл .apk (відвідувач або адміністратор заходу) на свій мобільний пристрій та запустити встановлення програми.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними для такої системи є обрання потрібного заходу і також потрібна авторизація через телефон чи квиток, а вихідними данними буде розклад заходу, нотифікації про зміни в розкладі та мапа заходу.