МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСУ "РОЗКЛАД ЗАНЯТЬ УНІВЕРСИТЕТУ"

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти "бакалавр"

Виконав: студент 4 курсу
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми
«Інженерія програмного забезпечення»
першого (бакалаврського) рівня освіти
Нерода Дмитро Олександрович
Керівники старший викладач
Черненко Ірина Євгенівна,
кандидат фізико-математичних наук,
доцент
Єрмолаєв Вадим Анатолійович
Рецензент доцент
Єрмакова-Черченко Наталія
Олександрівна

Херсон – 2020

ЗМІСТ

ВСТУП

Актуальність. Розклад занять є дуже важливою частиною навчального процесу. Він визначає, які предмети будуть вивчати студенти, коли і де вони це будуть робити. Це дозволяє систематизувати навчальний процес та дозволяє розпланувати виконання навчального плану як студентам, так і викладачам.

Тому досить багато уваги приділяють його створенню. Так, правильно складений розклад занять дуже допомагає у навчанні, коли погано складений – навпаки заважає. Як приклад погано складеного розкладу можна привести наявність “вікон” між заняттями, проведення занять, які потребують додаткового обладнання у звичайних класах, чередування часу навчання (одна половина днів – початок навчання з ранку, друга половина – з середини дня) тощо.

Але для користувачів розкладу (як студентів, так і викладачів) не менш важливими є форма його подачі, розповсюдження та оновлення. І справді, навіть ідеально складений розклад буде даремним, якщо користувач не зможе його завантажити, відкрити та подивитися. І дуже велика кількість навчальних закладів для цих дій не має зручного та кросплатформеного інструментарію. А навіть якщо і має, то майже завжди потрібно обов’язкове підключення до мережі “Internet”. Існує необхідність розробки сервісу, який зможе у зручному вигляді надати інформацію користувачу у будь-якому місці у будь-який час.

Метою даної роботи є створення сервісу для взаємодії із розкладом занять, який складається з WEB-сайту та Android-застосунку, здатних працювати як разом, так і окремо.

Тип сервісу – змішаний, бо поєднує в собі два застосунки, кожен на різних платформах: Android та Web. Такий поділ, по-перше, дозволить охопити велику кількість потенційних користувачів, по-друге – дозволить розділити між ними весь необхідний функціонал для максимальної простоти та зручності його використання.

Об’єкт дослідження – технології розроблення застосунків на платформі Android.

Предмет дослідження – розроблення Android-застосунку з можливістю під’єднання до бази даних WEB-сайту.

Для досягнення зазначеної мети необхідно реалізувати наступні **задачі**:

1. Аналіз актуальності сервісу.
2. Проєктування моделі та можливостей сервісу.
3. Ознайомлення та вивчення літератури з IDE PyCharm та Android Studio, фреймворку Django, мови програмування Python та Java, СУБД SQLite і мови розмітки XML для отримання знань щодо їх функціоналу.
4. Розроблення дизайну Android-застосунку.
5. Написання програмного коду для Android частини сервісу.
6. Дописування програмного коду до Web частини сервісу.
7. Тестування сервісу.

Структура й обсяг роботи. Курсова робота складається з вступу, двох розділів, висновків, списку використаних джерел та п’яти додатків. Повний обсяг курсової роботи складає 50 сторінок.

РОЗДІЛ 1
АКТУАЛЬНІСТЬ ТА МОДЕЛЬ СЕРВІСУ

1.1. Мета застосунку

Розклад занять в університеті є важливою складовою навчального процесу. Він дозволяє його систематизувати да структурувати, що підвищує ефективність навчання, причому це справедливо як для студентів, так і для викладачів. Проте замало просто створити гарний розклад, навіть якщо він враховує всі особливості навчання, учнів та викладачів. Потрібно і подати його у зручному вигляді, надати можливість переглядати у будь-який час та у зручній для користувачів формі. Також необхідно вчасно оновлювати розклад та надавати оновлення у автоматичному режимі.

На основі цього і буде розроблено проєкт і створено сервіс “Розклад занять університету”, призначенням якого є:

1. Покращення досвіду користування при переглядання розкладу користувачами.

2. Полегшення роботи адміністраторів із розкладом, його заповненням та редагуванням.

3. Надання можливості роботи із розкладом за допомогою найпоширеніших платформ, якими користуються або мають доступ майже всі.

1.2. Аналіз існуючих аналогів

Тепер необхідно проаналізувати всі існуючі на цей момент способи перегляду розкладу, окреслити їх переваги і недоліки. Нижче будуть використані як матеріали із попереднього дослідження, так і нові.

Розглядати будемо способи надання розкладу у Херсонському Державному Університеті, так як у всіх інших університетах використовуються або ті ж самі методи (чи подібні до них), або власні, розроблені саме для цього університету та яких немає у вільному доступі.

Їх усього два – роздрукована таблиця на листах ватману та електрона таблиця. Паперовий варіант є по суті роздрукованим варіантом .xls файлу. Але треба розглянути обидва, щоб зрозуміти, і покривають вони всі потреби студентів, викладачів та адміністраторів, які ними користуються.

1. *Електрона таблиця*

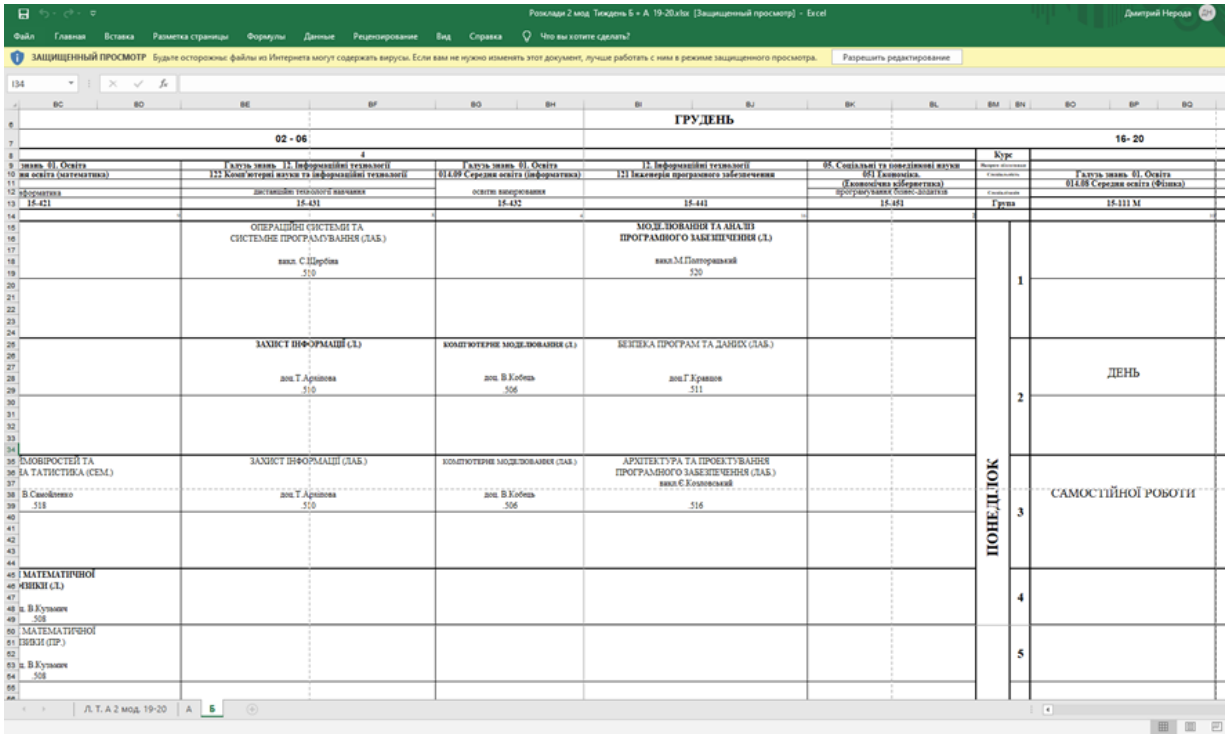

Рис 1.1 Приклад електронної таблиці у програмі Excel

Електрона таблиця для розкладу створюється зазвичай, у програмному забезпеченні Microsoft Excel.

Excel – це табличний процесор для створення та редагування електронних таблиць. [1]

Простота цього методу полягає у тому, що програма Microsoft Excel – це вже готовий продукт. Це дозволяє не розробляти своє програмне забезпечення чи WEB-застосунок. І хоча Microsoft Excel – платне програмне забезпечення, електроні таблиці можна створювати та редагувати у його безкоштовних аналогах. Вони мають не таку потужну підтримку користувачів та відсутні або реалізовані гірше деякі функції, але основний функціонал, який полягає у створенні та редагування таблиць в них ідентичний.

З іншого боку, створювати розклад у вигляді таблиці за допомогою програмного забезпечення, яке має більш загальні функції і не створювалось спеціально під це – задача досить складна. Усі рядки та стовбці треба підписувати, об’єднувати та переносити записи у них. За всім цим треба слідкувати, щоб не допустити орфографічних помилок та не зробити запис не у те поле, яке потрібно.

Це ускладняється ще тим, що розклад потрібно створити для усіх курсів та груп одночасно. Тобто увесь розклад повинен бути в одному файлі, що ще більше ускладнює його створення, редагування і роботу з ним в цілому.

Для студентів та викладачів, які є користувачами, переглядати розклад занять у такому вигляді теж дуже незручно. По-перше, треба, щоб на пристрої, який є у учня, було встановлене програмне забезпечення, яке дозволяє відкривати файли з розширенням .xls. По-друге, так як студенту зазвичай потрібен лише розклад своєї групи, більшість файлу буде для нього марним і лише заважатиме. І по-третє – для того щоб, щоб отримати оновлений розклад, треба ще раз скачати .xls файл. Це потребує зайвих дій та дуже незручно, так як треба регулярно переглядати сайт або групу університету, на якому викладають новий розклад.

І останній, досить суттєвий недолік – низька продуктивність застосунку при роботі з Excel-файлом розкладу. Через те, що розклад містить у собі списки занять всього факультету, файл з ним є досить важким, і навіть на сучасних смартфонах при роботі з ним можуть спостерігатись його довге відкриття та підлагування при навігації.

2. Лист ватману

Як вже зазначалося вище, цей спосіб не зовсім самостійний, так як є лише фізичною копією .xls файла.

Єдиною перевагою цього способу є те, що він не потребує жодного програмного забезпечення та будь-яких пристроїв.

А от недоліків набагато більше. По-перше, треба знаходитись в університеті, для того щоб побачити розклад. По-друге – усі складнощі складання розкладу за допомогою Excel та його оновлення тут також присутні.

Як бачимо, в обох способах надання розкладу недоліків набагато більше і вони є набагато критичнішими, аніж переваги. Так, ними можна користуватись, але було б доречним знайти інший спосіб, який буде більш зручний як тим, хто складає розклад, так і тим, хто цим розкладом користується.

На основі всього вищезазначеного можна зробити висновок, що методи представлення розкладу в Херсонському Державному Університеті не є оптимальними. Вони створюють незручності, які інколи заважають студентам нормально навчатись, а викладачам – працювати. Замість них буде доречніше використовувати інший спосіб, одним з яких буде результат цієї проєктної роботи. Він покликаний зберегти більшість переваг, примножити їх та усунути майже всі недоліки способів, які використовуються зараз.

1.3. Модель та можливості сервісу

Тепер необхідно окреслити модель та можливості сервісу “Розклад занять університету”, завдяки яким він буде зручним, зрозумілим, легким та кросплатформеним.

Як вже будо вище зазначено, сервіс буде складатись з двох компонентів: Web-застосунку та Android застосунку. У кожного з них буде своя роль.

1. Web-застосунок

Він складається з однієї головної Web-сторінки, на яку заходять користувачі і на якій відображається вся необхідна інформація, сторінки адміністрування Django, за допомогою яких адміністратори університету можуть додавати та редагувати розклад, і бази даних, у якій цей розклад зберігається.

Головна сторінка буде містити у собі блок із полями для вибору, за допомогою яких користувач буде обирати необхідний розклад, сам розклад у вигляді таблиці та блок із статистикою у вигляді діаграм.

Сторінка адміністрування матиме декілька сторінок, у яких будуть списки збережених у базі даних інформації, яку можна додавати, змінювати та видаляти.

База даних складається з 13 таблиць, кожна з котрих представлена класом. Нижче наведена діаграма класів, яка наочно демонструє ці класи та зв’язки між ними:

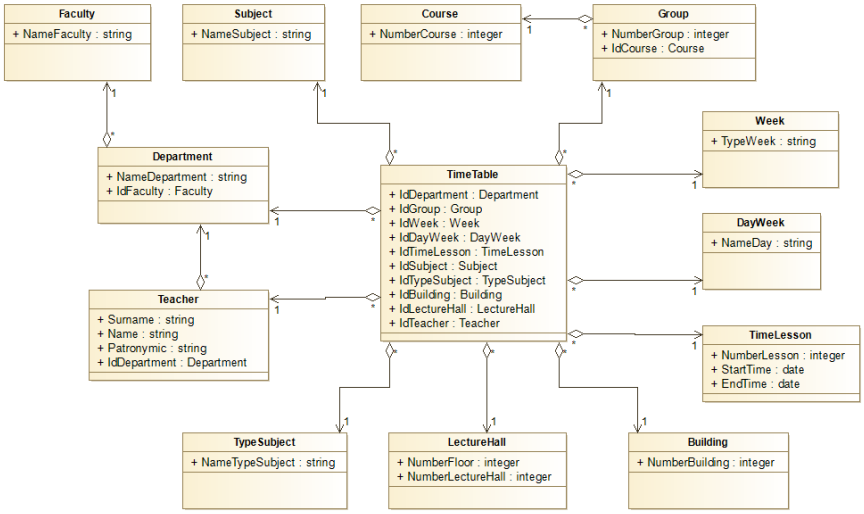

Рис. 1.2. Діаграма класів бази даних Web-застосунку

2. Android-застосунок

Застосунок буде складатися з головного меню, екрану із списками об’єктів, екрану із розкладом та допоміжними діалоговими вікнами.

За допомогою головного меню можна потрапити до кожної частини програми.

Екран зі списком компонентів дозволить переглядати, редагувати та видаляти об’єкти, які збережені у локальній базі даних застосунка.

Екран із розкладом буде містити у собі інформацію про розклад, яку можна редагувати.

Допоміжні діалогові вікна будуть використовуватися для налаштування після першого запуску, редагування бази даних та розкладу та завантаження розкладу з глобальної бази даних.

База даних є локальною, тобто зберігається у накопичувачу пристрою користувача. Вона складається з 7 таблиць, які представлені класами. Нижче наведена діаграма класів, яка демонструє таблиці, зв’язки між ними та поля таблиць:

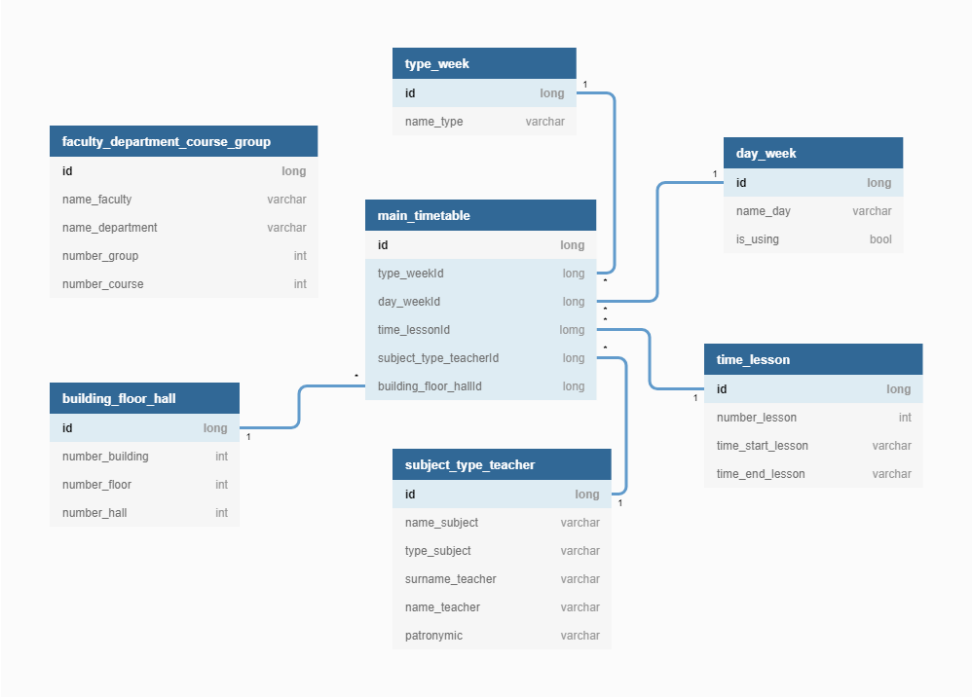

Рис. 1.3 Діаграма класів бази даних Android-застосунку

Також треба розглянути можливості сервісу. Для наочності буде приведена Use-case діаграма.

Вона демонструє, які можливості є у кожної ролі кожної частини сервісу. Виходячи з цієї діаграми, можна сказати, що отримати повну інформацію про свій розклад можна як за допомогою Web-частини сервісу, так і Android. Проте у кожному випадку є свої переваги та недоліки.

Наприклад, у Web-застосунку є можливість переглядати не тільки свій розклад, а й розклад будь-якого викладача чи групи в університеті. Водночас з цим, він потребує доступу до мережі Internet, якого може не бути з різних причин. А ось мобільний застосунок не потребує доступу до світової павутини для перегляданння розкладу. Проте, для створення розкладу мережа Internet може знадобитись, для зв’язку із сервером Web-сайту, або можна заповнити його власноруч.

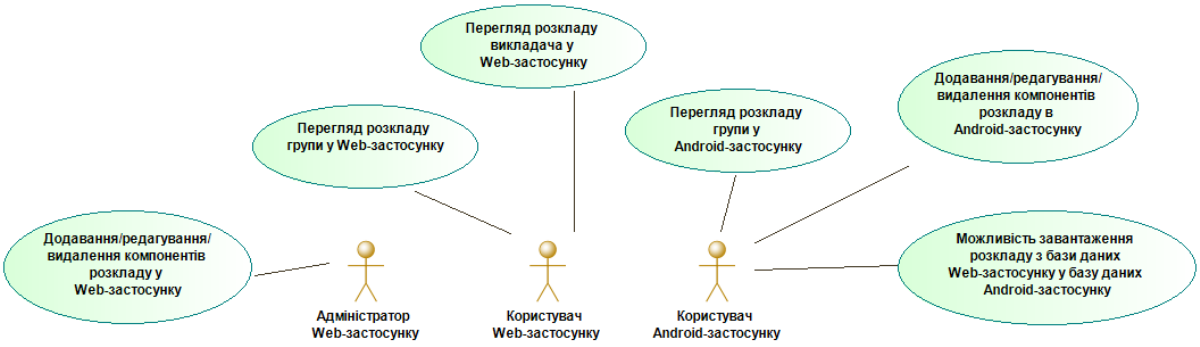

Рис. 1.4 Use-case діаграма сервісу “Розклад занять університету”

У висновку можна зазначити, що кожна частина сервісу спроєктована таким чином, щоб працювати самостійно, але за нагоди вони можуть взаємодіяти один з одним. Це робить сервіс дуже гнучким та зручним для використання.

Також треба зазначити, як саме буде здійснюватися зв’язок Android-застосунку та глобальної бази даних. Для цього було вирішено використовувати дані у форматі JSON, які будуть формуватися на боці серверу Web-сайту і прийматись та розпарсюватися у мобільному застосунку.

JSON – це текстовий формат даних із внутрішньою структурою у вигляді пар “ключ-значення” [2].

Цей спосіб є оптимальним, бо JSON і розроблявся для такого роду завдань, його дуже легко формувати та існує безліч бібліотек для його розпізнавання.

Тепер перейдемо безпосередньо до розроблення необхідних частин проєкту та технологій і засобів, за допомогою яких ми це будемо робити.

РОЗДІЛ 2
РОЗРОБЛЕННЯ СЕРВІСУ

Перед тим, як розпочинати безпосередньо створювати всі необхідні частини для нашого проекту, окреслимо більш детально інструменті, якими ми будемо користуватись під час цього. У цьому розділі буде розкрито, навіщо потрібна той чи інший засіб розроблення, окреслимо його переваги в цілому та конкретно для нашого проекту.

2.1. Переваги використання мови програмування Java

Мова програмування, на якій буде написаний основний код програми є одним з найважливіших інструментів розробки. Саме її ми будемо використовувати більшу частину часу, який буде використано для реалізації практичної частини нашого проєкту. Тому треба обрати таку мову програмування, яка буде комфортною у використанні, і при цьому містила б у собі весь необхідний функціонал.

Такою мовою була обрана Java. Перед тим, як описати її переваги, необхідно надати визначення та деяку загальну інформацію стосовно неї.

Java – це об’єктно орієнтована мова програмування, основою якої є класи. ЇЇ основна парадигма – максимально мультиплатформеність. Тобто, головним для цієї мови є можливість роботи на будь-якій машині з будь-якою операційною системою. Для цього використовується так звана “віртуальна Java-машина”, яка і виконує спеціальний байтовий код, який їй передає компілятор мови Java [3].

Права на цю мову належать компанії Oracle. Є як платні версії для комерційного використання з великим терміном підтримки, так і безкоштовні [4].

Все це робить Java однією з найпопулярніших мов у світі. Наприклад, за даними аналітичної фірми RedMonk, Java була другою за популярністю мовою програмування у світі, за даними фірми SlashData – Java займала 3 місце, а за індексом TIOBE – взагалі перше [5].

З усього вищезгаданого можна виділити перший вагомий плюс Java – велике ком'юніті. І дійсно, розробників на Java дуже багато, і тому, коли при програмуванні на цій мові виникають труднощі або питання, відповіді на них майже в завжди можна знайти на спеціальних сайтах, форумах тощо. Яскравий приклад – найвідоміший форум серед програмістів у форматі питання-відповідь – StackOverflow.

Друга перевага – надійність та безпека. У Java компілятор та інтерпретатор проводять декілька рівнів тестів, які перевіряють програму на помилки. Внаслідок цього, при виконанні програмного коду не відбуваються недопустимі операції (наприклад, з перетворення типів), на відміну від того ж C++.

Третій плюс, який ми вже зазначали – кросплатформеність. Разом з цим, саме Java – основна мова, на якій написано переважну кількість застосунків на платформі Android. Сама по собі ця операційна система є поєднанням ядра системи Linux та віртуальної машини Java, тому не дивно, що її використовують для розробки програм.

І четвертий вагомий плюс – поєднання функціональності та простоти. Java втілила в собі більшу частину багатого функціоналу C та C++, проте відкинула деякі їх складі елементи. Наприклад, у Java присутній автоматичний "збирач сміття", який вивільняє місце у пам'яті. Також в ній відсутні класичні покажчики.

Всі ці переваги і стали вирішальними у виборі саме мови програмування Java. Проте слід зазначити, що для програмування на платформі Android є і друга мова – Kotlin. Проте кількість матеріалів,

бібліотек та підтримки користувачів говорить проти неї, бо вона з'явилась відносно нещодавно (дата офіційного виходу Kotlin – 15 лютого 2016 рік, тоді як Java – 23 травня 1995 року) [6].

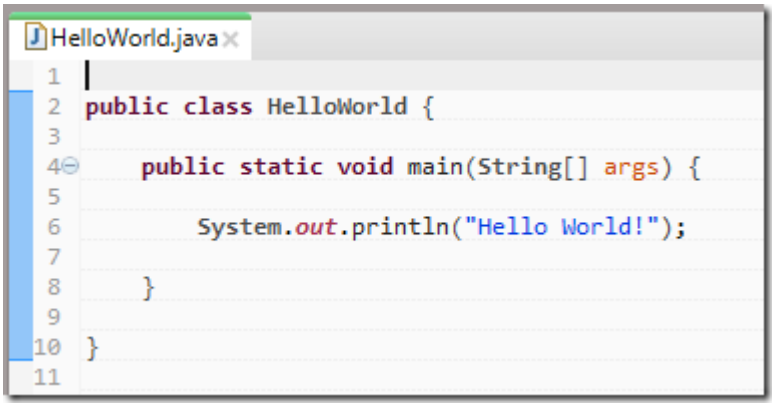

Рис. 2.1 Приклад коду на мові Java

2.2. Переваги використання СУБД SQLite

Структура та модель нашого застосунку передбачають наявність бази даних, у яких будуть зберігатись необхідна інформація, яка стосується розкладу, і яку потрібно зберігати навіть тоді, коли користувач закриває програму. Для нашого застосунку добре підійде SQL база даних, бо структуру даних, яка повинна зберігатись, можна легко представити у вигляді таблиць та зв'язків між ними. Але для роботи з SQL базою даних необхідна спеціальна система для її керування, і такою системою була обрана SQLite. Нижче буде наведено, що таке SQLite, та чому вибір зупинився саме на ній.

SQLite – це система управління баз даних (скорочено - СУБД). Її офіційний реліз відбувся у серпні 2000 року. Написана на мові C та розповсюджується як "суспільне надбання", тобто її можна використовувати будь-яка фізична чи юридична особа у приватних та комерційних цілях без обмежень[7].

Головною особливістю SQLite є те, що вона не потребує окремого процесу під сервер, тому що не використовує структуру за принципом “клієнт-сервер”, а безпосередньо вбудовується у програму та зберігає всі дані в одному файлі. Це робить її дуже легкою та зручною у використанні та підтримці.

Окрім цього, вона є кросплатформеною, тобто може працювати з усіма популярними мовами програмування та на великій кількості платформ.

Ще одна перевага – дуже малий розмір бібліотеки, яка реалізує СУБД. У середньому вона важить близько 600 кілобайт.

І останній суттєвий плюс – висока продуктивність при невеликій завантаженості та при роботі із простими операціями [8].

Всі ці особливості зробили SQLite однією з найпопулярніших баз даних. Її використовують у майже всіх популярних фреймворках та проектах у якості вбудованої СУБД. І вона також є за замовчуванням у IDE Android Studio, про яку ми поговоримо пізніше.

Таким чином, SQLite задовольняє все потреби нашого проекту – відсутність потреби у інтеграції, невеликий розмір, зручність при роботі. Також, за рахунок того, що у нашій локальній базі буде зберігатись відносно мала кількість записів (до 1000), то ця СУБД буде високопродуктивною, що забезпечить більшу зручність використання застосунком.

	id	NumberLesson	StartTime	EndTime
1	1	1	08:30:00	09:50:00
2	2	2	10:10:00	11:30:00
3	3	3	11:50:00	13:10:00
4	4	4	13:40:00	15:00:00
5	5	5	15:10:00	16:30:00

Рис. 2.2 Приклад таблиці з даними у SQLite

2.3. Огляд IDE Android Studio

На сучасному етапі розвитку програмування майже все програмне забезпечення не розробляється у так званому “текстовому файлі”. Замість цього, для керування проектом використовують інтегровану середовище розробки (скорочено на англійській – IDE). Це програма, у якій проходить створення застосунків, інших програм, Web-сайтів тощо. Вона дуже полегшує роботу с усім проектом, автоматизує рутинні дії та допомагає у написанні та рефакторінгу програмного коду. Ми будемо використовувати Android Studio у якості IDE. Нижче буде наведено інформацію про це інтегроване середовище розробки та пояснення, чому ми зупинились саме на ньому.

Android Studio – офіційне інтегроване середовище розробки для проектів на платформі Android. Дата виходу – грудень 2014 року. Її розробниками є компанія Google та, частково, JetBrains, тому що Android Studio побудована на основі іншої IDE, IntelliJ IDEA, яка є розробкою останньої [9].

Основною перевагою цієї IDE є її підтримка та рівень інтеграції із усіма необхідними компонентами розробки. На Android Studio регулярно виходять оновлення, які виправляють баги та надають новий функціонал або спрощують роботу з існуючим.

Окрім цього, є дуже зручна система Gradle, яка автоматизує збірку проєктів та за допомогою якої можна підключати додаткові бібліотеки чи модулі, які необхідні для розробки. Треба усього лише вписати у спеціальний блок dependencies строку зі словом implements та у лапках адрес, за яким у мережі Internet знаходиться потрібний модуль, та натиснути на повідомлення “Sync now”, яке з’явиться у верхній частині IDE. І якщо ви все зробили правильно, то після цього ви вже можете використовувати цей

модуль у своєму проєкті. У нашому випадку це більш ніж актуально, бо ми також будемо використовувати зовнішні бібліотеки.

Окрім цього, тут присутня зручна система для тестування проєкту – Android Virtual Device Manager, або, скорочено, AVD. Він дозволяє створювати, налаштовувати та використовувати емулятори Android, за допомогою яких можна тестувати застосунок у процесі його створення для визначення працездатності його функцій, їх відповідності початковим задумам та зовнішнього вигляду інтерфейсу. Та окрім стандартної установки поточної версії програми та її запуску “з нуля”, у Android Studio присутня така функція, як “Apply Changes”, яка дозволяє застосувати зміни у коді проєкту без перезавантаження його всього на емуляторі, а лише активного Activity. Це дуже зручно для тестування маленьких змін, або змін, для яких важливо, щоб був збережений поточний стан застосунку. До того ж, можна підключати і фізичні девайси, для більш надійного тестування, бо жодний емулятор не дає гарантії працездатності програми на реальних пристроях.

І наостанок, тут присутня дуже точна система рефакторингу коду, яка робить процес розробки більш зручним та допомагає у рутинних діях. Наприклад, якщо зробити свій клас в одній директорії, то можна ввести перший символ його назви в іншій директорії, то він буде присутній у списку можливих варіантів продовження. Окрім цього, якщо ви додасте його через цей список, то автоматично у тому файлі, у якому ви працюєте, у блоці імпорту автоматично з’явиться рядок з імпортом цього класу. Також, якщо ви декілька раз використовуєте одну і ту ж саму змінну, то IDE це запам’ятовує і у наступні рази першою пропонує саме цю змінну у якості продовження написаних початкових символів[10].

Усе це робить Android Studio найкращім вибором при розробці Android-застосунків. Він дозволить не перейматися за ручне налаштування більшої частини проєкту, надає зручний інструментарій для управління

бібліотеками та модулями, дозволяє швидко тестувати проєкт на будь-якому етапі його розробки та допомагає при написанні самого програмного коду.

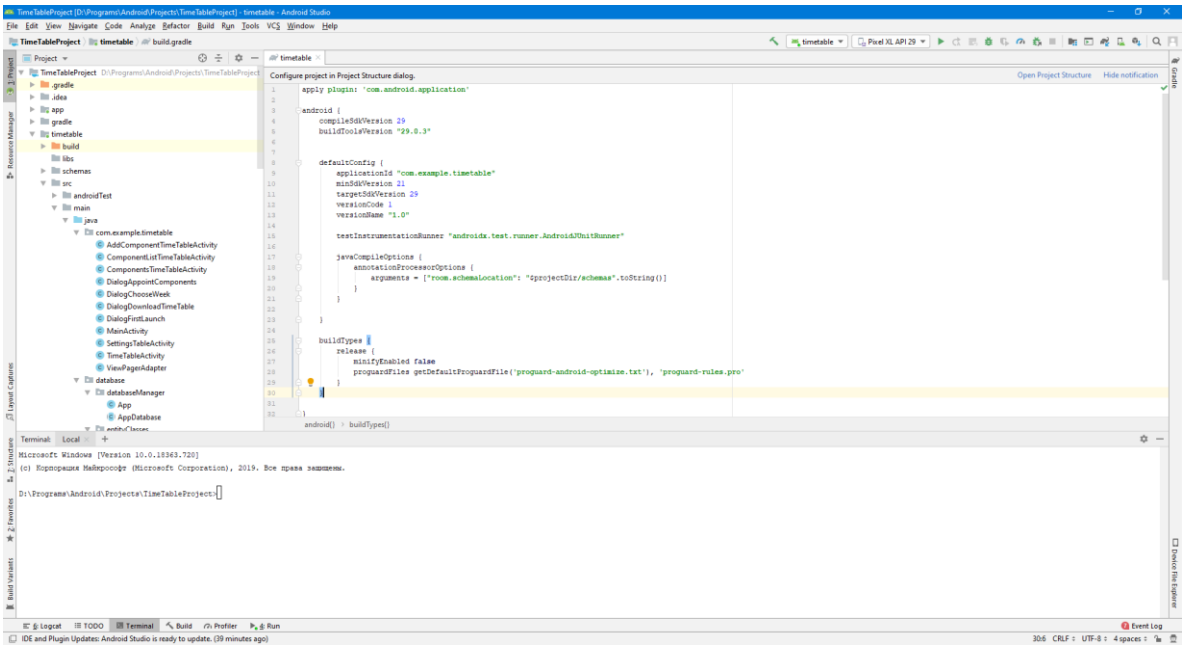

Рис. 2.3 Інтерфейс IDE Android Studio

2.4 Дизайн та інтерфейс сервісу

Тепер перейдемо до огляду зовнішнього вигляду інтерфейсу частин сервісу та його взаємодії із користувачем.

1. *Web-застосунок*

Як вже зазначалося, Web-застосунок складається з однієї головної сторінки та сторінок адміністратора. Нижче буде наведено скріншоти частин головної сторінки та опис її компонентів:

1.1 Верхня частина сторінки

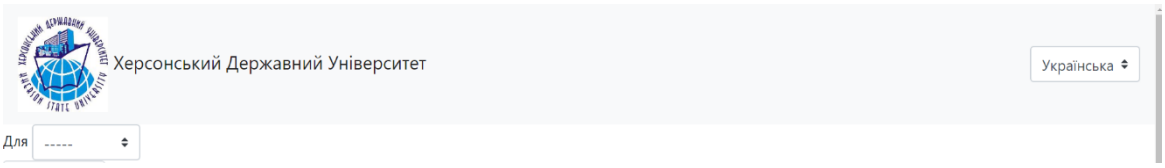

Рис. 2.4 “Шапка” головної сторінки

На верхній частині сторінки, яку ще називають “Шапка”, присутні логотип та назва Херсонського Державного Університету, правіше від них – селектор із випадаючим списком для вибору мови сторінки. Нижче присутній селектор для вибору суб’єкту розкладу – студента чи викладача. Коли у ньому буде обрано один з варіантів, блок розшириться і з’являться інші селектори для визначення, для якої групи чи вчителя потрібно продемонструвати розклад:

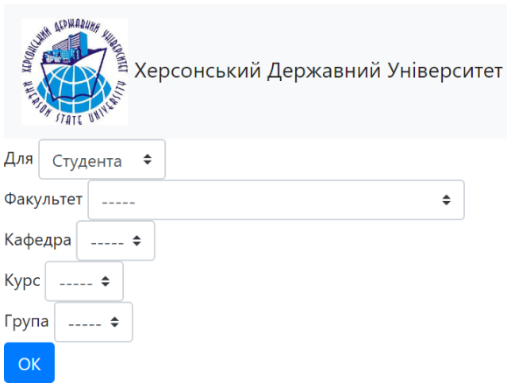

Рис. 2.5 Пункти для студента (групи)

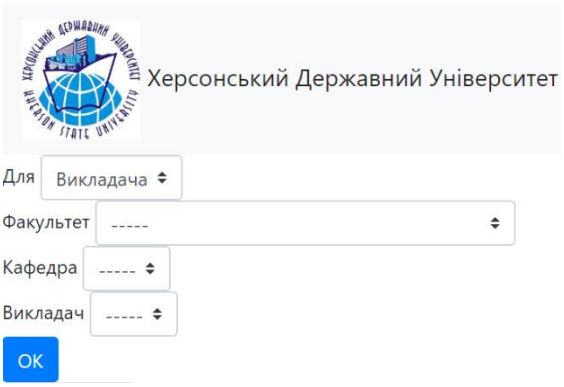

Рис. 2.6 Пункти для викладача

Після вибору усіх селекторів та натискання кнопки “ОК” сторінка перезавантажується, додаткові селектори зникають, але з’являється текст, з обраними варіантами для отримання розкладу та оновлюється основний блок.

1.2. Основний блок сторінки

У основному блоці присутній сам розклад у вигляді таблиці:

Зверху присутні вкладки для перемикання між тижнями, нижче йде сама таблиця, де по горизонталі йдуть дні тижня, а по вертикалі – номер заняття та діапазон часу його проведення. І на їх перетині йде інформація про саму пару, а саме: назва заняття, його тип та аудиторія, у якій воно проводиться. Якщо заняття у певний час немає, то у цьому блоці пишеться “Немає занять”.

Тиждень А Тиждень Б

А	Понеділок	Вівторок	Середа	Четвер	П'ятниця
№1 08:30 - 09:50	Якість програмного забеспечення та тестування Лекція 502	Нема занять	Нема занять	Нема занять	Бази даних розподілених інформаційних систем Лекція 502
№2 10:10 - 11:30	Бази даних розподілених інформаційних систем Лекція 502	Нема занять	Програмування - Інтернет Лабораторна 502	Нема занять	Емпіричні методі програмної інженерії Лекція 507
№3 11:50 - 13:10	Нема занять	Моделювання та аналіз програмного забеспечення Лекція 502	Основи комп`ютерної графіки Лекція 510	Нема занять	Аналіз даних Лабораторна 512
№4 13:40 - 15:00	Нема занять	Моделювання та аналіз програмного забеспечення Лабораторна 512	Основи комп`ютерної графіки Лабораторна 502	Бази даних розподілених інформаційних систем Лабораторна 510	Нема занять
№5 15:10 - 16:30	Нема занять	Нема занять	Нема занять	Архітектура та проектування програмного забеспечення Лабораторна 510	Нема занять

Рис. 2.7 Основний блок із розкладом

1.3. Нижня частина сторінки

І остання частина – так званий “футер” сторінки. У ньому будуть надаватись різні діаграми із цікавою статистикою обраного розкладу (наприклад, кількість та співвідношення кількості різних типів занять, таких як лабораторна, лекція чи практична):

Рис. 2.8 “Футер сторінки”

1.4 Панель адміністратора

Вона складається з декількох сторінок, необхідних для адміністраторів сайту, які будуть мати доступ до бази даних та змогу редагувати її. Функціонал та інтерфейс панелі адміністратора надає фреймворк Django, і так як вона є досить зрозумілою та зручною, було прийнято рішення використовувати його, а не створювати своє. Доступ до цієї панелі буде здійснюватися за допомогою логіну і паролю. Тому тут буде наведено

скріншоти головної сторінки, на якій присутній список усіх таблиць у базі даних, прикладу однієї з таблиць із записами та сторінка створення/редагування певного запису в таблиці:

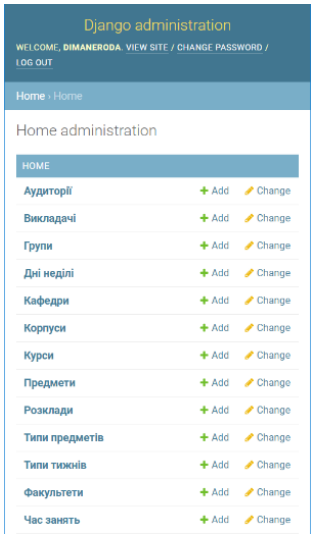

Рис. 2.9 Сторінка із таблицями

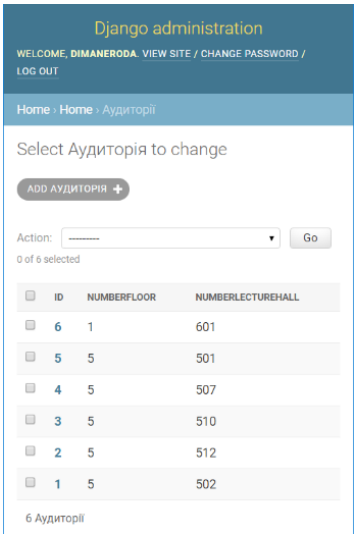

Рис. 2.10 Сторінка із записами таблиці

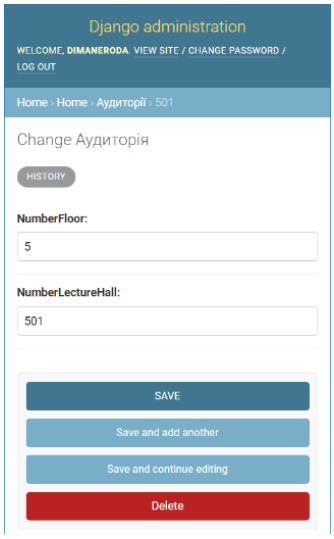

Рис. 2.11 Сторінка із редагуванням запису

2. *Android-застосунок*

У цьому розділі будуть надані скріншоти, які демонструють, як виглядає кожен екран чи діалогове вікно у застосунку.

2.1. Діалогове вікно початкового запуску

Коли користувач встановить застосунок, то при першому заході у нього йому відкриється діалогове вікно (рис. А, 1). У ньому зверху присутній текст із привітанням та інформацією, навіщо потрібно це вікно. Нижче йде чекбокс із кількістю типів тижнів та текст, який пояснює, що користувачу потрібно обрати. І у самому низу є 4 кнопки:

1. Кнопка “Вихід”. Її натискання завершить роботу застосунка. При цьому, якщо користувач ще раз його відкриє, то побачить те ж саме діалогове вікно.

2. Кнопка “Наступний крок”. За допомогою неї з’явиться наступний вибір, який необхідно заповнити/вибрати для отримання інформації щодо часу розкладу. Якщо при цьому не було обрано жодного варіанту, або у якесь поле було введено дані у невірному форматі, то з’явиться спливаюче повідомлення, яке позначить помилку, та перехід на наступний крок не здійсниться.

3. Кнопка “За замовчуванням”. Вона дозволяє пропустити усі кроки та заповнить базу даних стандартним розкладом, який найчастіше зустрічається (на цей час це 2 типи тижнів, усі 5 будніх днів, максимум шоста пара, початок занять о 8 годині та 30 хвилині, тривалість одного заняття – 80 хвилин, тривалість перерв – 10, 20, 10, 10, 10 хвилин).

4. Кнопка “Завантажити Розклад”. Натискання цієї кнопки закриє діалогове вікно першого запуску та відкриє інше діалогове вікно, за допомогою якого можна завантажити розклад із сервера.

Розглянемо наступні кроки у діалоговому вікні початкового запуску. При натисканні кнопки “Наступний крок” змінюється вся частина діалогового вікна, окрім кнопок (рис. А, 2). Тепер необхідно обрати один або декілька днів, коли у користувача є заняття в університеті. Зверху йде пояснювальний текст, а всередині – чекбокс із множинним вибором (тобто можна обрати одразу декілька, або навіть усі запропоновані варіанти). Також хочеться зазначити, що кнопка “Вихід” змінила свою назву на “Назад”, і тепер повертає на попередній крок.

Після вибору днів тижня та натискання кнопки “Наступний крок” з’являється наступна частина вибору часу занять, а точніше – поле для вводу максимального порядкового номеру пари, яка є у користувача та текст, пояснюючий, що треба зробити (рис. А, 3).

Після вводу номеру пари та натискання кнопки “Наступний крок” діалогове вікно змінюється та з’являється 3 поля для вводу та пояснюючі

замітки до них (рис. А, 4). У перше поле необхідно ввести час початку першої пари у форматі ГГ:ХХ. У друге поле – тривалість одного заняття у хвилинах. І в останнє поле – тривалість перерв. Останні треба вводити через кому, і їх кількість повинна дорівнювати номеру найпізнішої пари мінус один. Також кнопка “Наступний крок” змінює свою назву на “Зберегти”.

Після того, як всі дані будуть введені вірно та натиснута кнопка “Зберегти”, діалогове вікно закривається. Після цього у фокус попадає перше Activity – головне меню.

2.2 Головне меню, меню компонентів та діалогові вікна завантаження розкладу та вибору типу тижня

Після отримання інформації щодо часу розкладу, або його завантаження з глобальної бази даних, користувач бачить головне меню програми (рис. Б, 1), яке буде відкриватись тепер кожен раз після запуску застосунку. У ньому знаходяться 4 кнопки:

1. Кнопка “Розклад”. Після її натискання може буди два варіанти розвитку подій: якщо у введеній завантаженій інформації щодо розкладу типів тижнів більше одного – то відкриється діалогове вікно зі списком тижнів (рис. Б, 2), якщо у розкладі тільки один тип тижня – то відкриється одразу екран із розкладом.

2. Кнопка “Завантажити розклад”. Працює аналогічно кнопці “Завантажити” із діалогового вікна під час першого запуску.

3. Кнопка “Компоненти розкладу”. Відкриває меню компонентів (рис. Б, 3).

4. Кнопка “Вихід”. Вона закриває застосунок та переводить користувача до меню телефону.

Діалогове вікно для вибору тижня складається зі списку тижнів, які прописані у розкладі. Натискання на один з них переведе до екрану із розкладом цього тижня.

Діалогове вікно із завантаженням розкладу (рис. Б, 4) складається з 3 текстових полів, які описують, для чого воно та що треба зробити користувачу, двох полів для вводу, у перше з яких треба ввести кафедру, у друге – номер групи того розкладу, який потрібен на цей момент, та 2 кнопок: “Повернутися” та “Завантажити”. Перша закриває діалогове вікно без жодних додаткових дій та повертає користувача до головного меню (але треба зазначити, що при натисканні цієї кнопки, за умови, що це діалогове вікно будо відкрито не з головного меню, а з вікна для першого налаштування, то користувача поверне не до головного екрану, а до першого кроку із привітанням).

Екран із компонентами розкладу (рис. Б, 3) складається з 2 кнопок: “Список предметів та викладачів” та “Список місць проведення занять”. Кожна з них відкриває екран із відповідними елементами розкладу.

2.3. Екрани із списками компонентів розкладу та їх створенням та редагуванням

У застосунку присутні два екрана зі списками компонентів: один складається з предметів та вчителів, інший – з місць проведення пар. Але істотно вони нічим не відрізняються, принцип їх побудови один і той самий, тому тут буде продемонстровано їх вигляд і робота, а також екрани для створення чи редагування об’єктів списку тільки для місць проведення занять.

Після того, як користувач натисне кнопку “Список місць проведення занять”, він побачить екран, подібний до того, який продемонстрований на рис. В, 1. На ньому ми бачимо список усіх збережених у базі об’єктів цього типу та кнопку “Додати місце проведення пари”. Якщо у базі немає жодного збереженого об’єкту, то список буде відсутній. При утриманні на будь-якому елементі списку на екрані з’явиться контекстне меню (рис. В, 2) із варіантами редагувати цей елемент або видалити його. Видалення призведе

до зникнення елементу зі списку та з усіх елементів розкладу, до яких він був приєднаний. А варіант “редагувати” та кнопка для додавання по суті виконують одну і ту саму дію – відкривають екран, який показано на рис. В, 3. На ньому присутні поля для вводу необхідної інформації, тексту для опису цих полів та кнопки “Зберегти”. Якщо були заповнені не всі поля, то при натисканні на неї з’явиться спливаюче повідомлення про це. Якщо ж була введена всі необхідні дані, то знову відкривається екран зі списком, у якому тепер додатково відображається тільки що доданий елемент.

Це якщо користувач хотів додати елемент. Якщо ж він хотів його відредагувати, то на екрані для додавання/редагування поля для вводу вже заповнені інформацією обраного елемента. Якщо користувач нічого не змінює та натискає на кнопку для збереження, йому відкриється спливаюче повідомлення, що такий запис вже існує. Якщо змінить – то елемент списку оновиться згідно редагованої інформації.

2.4. Екран розкладу та діалогові вікна для його заповнення

Це основний екран у застосунку, який власне і відображає той розклад, який був налаштований власноруч користувачем або завантажений із глобальної бази даних. Він складається з вкладок, у кожному з яких є свій список (рис. Г, 1). Вкладки позначають дні тижня, а кожен елемент списку – номер пари і інформацію про неї. Переміщатися між вкладками можна натисканням на потрібну, тоді вона одразу відкриється, або свайпами ліворуч/праворуч. У останньому випадку кожен свайп буде переводити на сусідню із поточною вкладкою.

Самі елементи списку представляють собою текстові поля з інформацією про заняття, а саме: номер пари, діапазон часу проведення, назва предмету, його тип, прізвище, ім’я та по-батькові викладача та номер корпусу, поверху та аудиторії, де пара буде проводитись.

При утриманні на одному з елементів списку з’явиться контекстне меню з такими варіантами дій: “Призначити предмет та викладача”, “Призначити місце проведення”, “Призначити відсутність заняття” (рис. Г, 2).

Перший варіант відкриває діалогове вікно із списком предметів та викладачів (рис. Г, 3), які були відображені на екрані зі списками компонентів. Натискання на будь-який з них закриває діалогове вікно та дані з обраного елементу з’являться у списку із головним розкладом.

Аналогічно працює і другий варіант, тільки він працює із записами про місце проведення пар (рис. Г, 4).

І, нарешті, останній варіант видаляє поточні дані щодо обраного заняття, що у свою чергу повідомляє, що у цей час занять немає.

Підсумок

У результаті проєктування та розроблення дизайну та інтерфейсу ми отримали дуже простого, зручного та зрозумілого посередника між користувачем та виконавчим кодом кожної частини сервісу. Тепер можна перейти до написання програмного коду, огляд якого буде представлений нижче. Так як код більшої частини Web-застосунку був написаний під час попередньої курсової роботи, то у цій роботі буде описана логіка усього Android-застосунка та частини Web-сайту, яка буде відповідати за взаємодію між ними.

2.5. Логіка Android-застосунку

У цьому підрозділі буде пояснено, як саме працює Android частина сервісу зсередини. Будуть описані усі дії, які проходять у ньому при взаємодії із користувачем. За необхідністю також буде продемонстрований програмний код та описана функція, яку він виконує.

1. База даних

Як вже зазначалось, у якості СУБД ми будемо використовувати SQLite. Вона є базою даних за замовчуванням у IDE Android Studio і не потребує спеціальних налаштувань. Проте, для більшої зручності та надійності ми будемо працювати не безпосередньо з нею, а зі спеціальною бібліотекою Room, яка в свою чергу вже буде працювати з SQLite. Такий підхід значно спрощує створення та роботу з базою даних.

Для того, щоб створити таблицю, необхідно всього лише створити клас із необхідними полями, конструктор для створення об'єктів цього класу та позначити цей клас анотацією @Entity. Також необхідно, щоб було одне поле з анотацією @PrimaryKey, яка позначає первинний ключ, тобто унікальний ідентифікатор, за яким будуть розрізнятися майбутні записи у базі даних. Нижче наведено клас, який використовується бібліотекою Room для створення та роботи з таблицею, яка зберігає у собі записи про типи тижня. Інші класи будуть виглядати аналогічно, відповідно до своїх полів, які буди зазначені у діаграмі класів у першому розділі.

```
@Entity
public class TypeWeek {

    @PrimaryKey(autoGenerate = true)
    public long id;
    public String nameWeek;
    public TypeWeek(String nameWeek) {
        this.nameWeek = nameWeek;
    }

    @Ignore
    public TypeWeek(long id, String nameWeek) {
        this.id = id;
        this.nameWeek = nameWeek;
    }
}
```

Проте треба зазначити 2 деталі. Перша деталь – параметр “autogenerate” зі значенням “true” позначає, що у тому разі, коли до бази даних додається об’єкт із відсутнім значенням головного ключа, то автоматично підбирається його найближче вільне значення. Друга – анотація @Ignore позначає, що цей метод або поле класу не треба враховувати під час створення таблиці у базі даних.

Тепер перейдемо до того, як працювати із створеними таблицями у базі даних. Для цього у бібліотеці Room є анотація @Dao. Нею помічається інтерфейс, у якому прописані методи-запити до бази даних. Ось приклад такого інтерфейсу, який керує все ж тією таблицею типів тижнів:

```
@Dao
public interface TypeWeekDao {
    @Insert
    void insert(TypeWeek typeWeek);

    @Update
    void update(TypeWeek typeWeek);

    @Delete
    void delete(TypeWeek typeWeek);

    @Query("SELECT * FROM TypeWeek")
    List<TypeWeek> getAll();

    @Query("SELECT * FROM TypeWeek WHERE id = :id")
    TypeWeek getById(long id);

    @Query("DELETE FROM TypeWeek")
    void deleteAll();
}
```

Анотація @Insert позначає метод, який зберігає запис у базі. На вхід він приймає об’єкт типу TypeWeek, тобто об’єкт того самого класу, який використовувався для створення таблиці. Відповідно, анотації @Update та @Delete оновлюють та видаляють записи з бази, причому роблять вони це

по головному ключу, який повинен буди у переданому їм у якості параметру об’єкту.

Але ці анотації виконують прості запити. Для більш складних використовують @Query, у якій пишеться запит на мові SQL. У прикладі продемонстровані методи, які виконують вилучення усіх записів у таблиці, вилучення одного запису із обраним “id” та видалення усіх даних (“getAll()”, “getById()” та “deleteAll()” відповідно).

2. Головне меню

У цьому пункті ми розберемо програмний код у Activity головного меню та усіх пов’язаних із ним діалогових вікон. Розпочнемо із діалогового вікна першого запуску застосунку.

Воно з’являється після спеціальної перевірки на наявність записів у базі даних:

```
int sizeAll =  typeWeekDao.getAll().size() + dayWeekDao.getAll().size() +
timeLessonDao.getAll().size() + subject_type_teacherDao.getAll().size() +
building_floor_hallDao.getAll().size() + mainTimeTableDao.getAll().size();

        if (sizeAll == 0) {
            FirstLaunch = new DialogFirstLaunch();
            FirstLaunch.show(getSupportFragmentManager(), "FirstLaunch");
        }
```

Якщо їх немає, то викликається діалогове вікно “FirstLaunch”. Воно складається з класу “DialogFirstLaunch”, який унаслідується від “DialogFragment” та представлений у вигляді файлу DialogFirstLaunch.java та layout-файлу – dialog_first_launch.xml. У першому зберігається код, який описує логіку діалогового вікна, у другому – який описує зовнішній вигляд. Розглянемо їх більш детально.

Як вже зазначалося, при роботі з цим діалоговим вікном є декілька кроків, і на кожному з них необхідно надати якусь інформацію. Тому для того, щоб не створювати багато класів, було вирішено все зробити в одному

класі і програмно змінювати екран. Тому у layout-файлі присутні такі теги: 4 TextView, 4 EditText, 1 RadioGroup з двома RadioButton, 1 ListView та 4 Button. TextView необхідні для надання інформації користувачу, що треба робити або вводити, EditText позначають поля для вводу, RadioGroup потрібен для вибору одного з наданих варіантів, ListView – для списку, у якому можна вибрати декілька варіантів і Button – кнопки, які при натисканні виконують свою функцію.

На самому початку, коли треба обрати кількість типів тижня, користувачу видно лише 2 текстових поля, радіогрупу з двома кнопками та 4 звичайних кнопки. У них відповідно записані привітання та що необхідно зробити, підписано, яку відповідь позначає кожна радіокнопка та дії кожної звичайної кнопки. Всі інші елементи приховані за допомогою властивості visible зі значенням “gone”. Воно позначає, що цей тег не буде відображатись на екрані та займати на ньому місце. Інші значення – “invisible” та “visible”. Перше також не відображається на екрані, але займає свою площину, а друге відповідно займає місце та відображається користувачу. Ось приклад тегу EditText, який позначає поле для вводу найпізнішої пари:

```
<EditText

    android:id="@+id/edit_first_max_number_lesson"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:digits="12345678"
    android:maxLength="1"
    android:visibility="gone">
```

Коротко пройдемося по всім параметрам тегу.

id – це унікальний ідентифікатор, за яким цей блок можна знайти у програмному коді.

layout_width та layout_height позначають, скільки місця займає блок у ширину та висоту відповідно. Їх значення, “match_parent” та “wrap_content”

позначають увесь простір, який доступний батьківському елементу та мінімально необхідний простір для контенту цього блоку.

inputType, digits та maxLength позначають відповідно тип даних, який можна вводити у поле, конкретні допустимі символи та максимальна кількість цих символів.

Тепер розглянемо, що саме робить кожна кнопка. Якщо користувач натискає “Вихід”, то виконується такий код:

```
getActivity().finishAffinity();

System.exit(0);
```

Він закриває поточне Activity та завершує програму.

Коли натиснута кнопка “Наступний крок”, то спочатку йде перевірка, чи правильно ввів дані користувач (та чи ввів він їх узагалі). Якщо щось було зроблено неправильно, то викликається спливаюче повідомлення, наприклад таке:

```
Toast.makeText(getActivity(), "Ви не обрали жодного варіанту",
Toast.LENGTH_LONG).show();
```

Використовується клас “Toast” та його метод “makeText”, у який передається об’єкт класу “Context” (у цьому випадку в його ролі виступає Activity, у якому запущено діалогове вікно, бо клас “Activity” унаслідується від “Context”), за допомогою якого це повідомлення буде показано, сам текст повідомлення та тривалість його демонстрації на екрані. Сама демонстрація активується методом “show”.

Якщо ж все було введено правильно, то частина тегів приховується, а потрібні навпаки, стають видимі. Також, за необхідності, змінюється назва кнопок. Це реалізується за допомогою наступного коду:

```
radioGroupTW.setVisibility(View.GONE);
listFirstChooseDaysWeek.setVisibility(View.VISIBLE);
textQuestionTypeWeek.setText("\nОберіть дні тижня, у які у вас є хоча б 1
заняття:");
```

```
buttonFirstBack.setText("Назад");
textFirstWelcome.setVisibility(View.GONE);
```

Тут ми програмно змінюємо видимість елементів, повідомлення у текстових полях та назву кнопки.

Кнопка “Назад”, по суті виконує останній код навпаки, тобто робить всі елементи екрану такими, яким вони були на попередньому кроці.

Коли вся необхідна інформація була введена конкретно, то натискання на кнопку “Зберегти” створює “каркас” розкладу. Він складається з об’єктів класів “Building_Floor_Hall” та “Subject_Type_Teacher”, кожен по одному екземпляру:

```
Building_Floor_Hall building_floor_hallObject = new Building_Floor_Hall(1,
NumberBuildingI, NumberFloorI, LectureHallI);
building_floor_hallDao.insert(building_floor_hallObject);

Subject_Type_Teacher subject_type_teacherObject = new
Subject_Type_Teacher(1, SubjectS1, TypeSubjectS, TeacherFullNameM[0],
TeacherFullNameM[1], TeacherFullNameM[2]);
subject_type_teacherDao.insert(subject_type_teacherObject);
```

Вони заповнюються значеннями за замовчуванням і позначають відсутність занять. Потім йдуть 3 цикли “for”, у яких створюються та зберігаються у базі відповідно типи і дні тижня та часи занять. Ось приклад такого циклу:

```
for (int i = 0; i < sizeTypeWeekFunc; i++) {
    TypeWeek typeWeekTest = new TypeWeek(i + 1, arrayAllTypesWeekFunc[i]);
    typeWeekDao.insert(typeWeekTest);
}
```

Тут у змінній “sizetypeWeekFunc” зберігається кількість типів тижнів, і на кожній ітерації створюється та записується до бази даних об’єкт класу TypeWeek.

Після цього йде потрійний цикл, у якому створюються об’єкти класу “MainTimeTable”, у яких зберігаються id створених раніше об’єктів:

```
long iteratorMain = 1;
for (int i = 0; i < sizeTypeWeekFunc; i++) {
```

```
    for (int j = 0; j < usingDaysWeekFunc.size(); j++) {
        if (usingDaysWeekFunc.get(j) == 1) {
            for (int q = 0; q < maxNumberLessonFunc; q++) {
                MainTimeTable timeTableTest = new
MainTimeTable(iteratorMain, i + 1, j + 1, q + 1,
                        1, 1);
                mainTimeTableDao.insert(timeTableTest);
                iteratorMain += 1;
            }
        }
    }
}
```

Змінна “iteratorMain” потрібна для головних ключів записів таблиці MainTimeTable. Перший цикл – проходить по кількості типів тижнів, другий – дням тижня і третій по максимальної кількості занять в день. У якості зовнішніх ключів другого, третього та четвертого параметрів використовуються ітератори відповідних циклів плюс 1, тому що первинні ключі відповідних записів були збережені власноруч і ми можемо бути впевненими, що вони існують. Останні 2 параметра позначають id об’єктів класів “Building_Floor_Hall” та “Subject_Type_Teacher”.

Кнопка за замовчуванням робить всі ті ж операції, але замість інформації, яку вводить користувач використовується та, яка закладена у програму.

Діалогове вікно для завантаження потрібно, щоб заповнити інформацію про свою групу, яка потім використовується для фільтрації записів у глобальній базі даних. Після того, як прийшли дані у форматі JSON, їх необхідно розпарсити, тобто перевести у потрібний нам формат. Ось приклад парсингу інформації про типи тижнів:

```
ArrayList<Long> ListIdTW = new ArrayList<>();
JSONArray jsonArray2 = response.getJSONArray("TypeWeek");
for (int i = 0; i < jsonArray2.length(); i++) {
    JSONObject requestObject = jsonArray2.getJSONObject(i);
```

```
    long id_type_week = requestObject.getLong("id_type_week");
    String type_week = requestObject.getString("type_week");

    Long helpLong = id_type_week;
    ListIdTW.add(helpLong);

    TypeWeek testInsertObject = new TypeWeek(i + 1, type_week);
    long idAddTW = typeWeekDao.insertLong(testInsertObject);
}
```

Тут створюється допоміжний список, у якому будуть зберігатись id отриманих об’єктів. Після цього ми з об’єкту response класу JSONResponse, у якому і збережена отримана з серверу інформація, отримаємо список об’єктів за ключом “TypeWeek”. Проходимо по ньому у циклі “for”, зберігаємо отримані id у додатковому списку і сам об’єкт у локальній базі даних.

Таким чином зберігаються усі інші об’єкти.

Укінці, за таким же принципом, як у створенні розкладу власноруч створюється каркас розкладу, а потім він заповнюється заняттями, викладачами та місцями проведення у спеціальному циклі:

```
for (int i = 0; i < ListAllIdMT.size(); i++)
{
    Long[] itemList = ListAllIdMT.get(i);
    Long[] helpListForSTT = {itemList[3], itemList[4], itemList[7]};
    Long[] helpListForBFH = {itemList[5], itemList[6]};

    mainTimeTableDao.updateSTTandBFHByWeekDayTime(
            ListIdTW.indexOf(itemList[0]) + 1,
            ListIdDW.indexOf(itemList[1]) + 1,
            ListIdTL.indexOf(itemList[2]) + 1,
            CalculateIndex(ListIdSTT, helpListForSTT) + 2,
            CalculateIndex(ListIdBFH, helpListForBFH) + 2);
}
```

У ньому ми проходимо по всім об’єктам головної таблиці, яку ми отримали та по трьом параметрам (тип тижня, день та час проведення)

шукаємо необхідний вже збережений у локальній базі запис та присвоюємо його зовнішнім ключам id об’єктів розкладу.

Власна функція “CalculateIndex” повертає індекс масиву, якщо він є у списку. Вона необхідна, бо стандартний метод “indexOf” не працює, якщо шуканий елемент є масивом, а не рядком чи числом.

Тепер, коли розклад тим чи іншим методом створений, користувач може взаємодіяти із головним меню. Кнопка “Вихід” викликає аналогічний метод, що і однойменна кнопка у діалоговому вікні першого запуску. Аналогічно із кнопкою “Завантажити розклад”, окрім однієї деталі: якщо якийсь розклад все присутній у базі, він видаляється. Перейдемо тепер до створення елементів розкладу, таких як об’єкти класів Subject_Type_Teacher та Building_Floor_Hall.

3. Меню компонентів розкладу

Для того, щоб перейти до нього, треба натиснути кнопку “Компоненти розкладу” у головному меню. Перехід здійснюється за допомогою наступного коду:

```
intent = new Intent(this, ComponentsTimeTableActivity.class);
startActivity(intent);
```

Створюється об’єкт класу “Intent”, його параметрами є context (у цьому випадку використовуються сам об’єкт Activity) та клас Activity, який необхідно відкрити і він передається до методу “startActivity”, який і відкриває необхідний екран.

Після цього, для переходу до потрібного списку компонентів необхідно натиснути на відповідну кнопку. У залежності від останньої ми передаємо до Activity, що відкриється, інформацію, які саме об’єкти, збережені у локальній базі даних, ми хочемо бачити:

```
intent = new Intent(this, ComponentListTimeTableActivity.class);
intent.putExtra("Type component", "Subject_Type_Teacher");
startActivity(intent);
```

Вона передається у вигляді пари “ключ-значення”.

Розглянемо, яким чином формується список та як він редагується більш детально на прикладі списку із номерами корпусу, поверху та аудиторії (для списку із предметами та викладачами використовується аналогічний код, тільки інформація береться з іншої таблиці).

Ось код, який будує списки об’єктів:

```
Intent intentForResult = getIntent();

type_component = intentForResult.getStringExtra("Type component");
if (type_component.equals("Subject_Type_Teacher")) {
    create_or_change_list_of_SubjectTypeTeacher(listComponentTimetable);
    buttonAddComponent.setText("Додати предмет да викладача");
}
else if (type_component.equals("Building_Floor_Hall")) {
    create_or_change_list_of_BuildingFloorHall(listComponentTimetable);
    buttonAddComponent.setText("Додати місце проведення пари");
}
```

Ми приймаємо інформацію, отриману з попереднього Activity і в залежності від неї викликаємо свою функцію для формування списку та змінюємо текст кнопки для додавання об’єкту на відповідний.

Тепер розглянемо, що саме робить зазначена функція.

```
Building_Floor_HallDao building_floor_hallDao =
db.building_floor_hallDao();
List<Building_Floor_Hall> allBuilding_Floor_Hall =
building_floor_hallDao.getAll();

int sizeListBFH = allBuilding_Floor_Hall.size();

ArrayList<Map<String, Object>> data = new ArrayList<Map<String,
Object>>(sizeListBFH);
Map<String, Object> m;
```

Спочатку ми отримуємо дані з бази даних, їх кількість та створюємо допоміжний масив із словником. Потім у циклі ми заповнюємо словник парами “назва поля-його значення”, які складаються з порядкового номеру

та унікального id об’єкту, номеру будинку, номеру поверху та номеру аудиторії. І у тому ж циклі ми додаємо отриманий словник до списку. У кінці ми отримаємо список із словниками, у кожному з яких збережена інформація про свій запис з бази даних.

Потім ми створюємо об’єкт класу “SimpleAdapter” та застосовуємо його до нашого списку:

```
String[] from = { ATTRIBUTE_NAME_NUMBER, ATTRIBUTE_NAME_BUILDING,
ATTRIBUTE_NAME_FLOOR, ATTRIBUTE_NAME_HALL, ATTRIBUTE_NAME_ID };
int[] to = { R.id.number_item_component, R.id.first_item_component,
R.id.second_item_component, R.id.third_item_component,
R.id.id_item_component };
SimpleAdapter sAdapter = new SimpleAdapter(this, data,
R.layout.item_component_timetable, from, to);

listComponentTimetable.setAdapter(sAdapter);
```

У якості параметрів конструктора ми використовуємо поточне Activity у якості контексту, наш список із словниками, кастомне представлення елементу списку у вигляді layout-файлу та 2 масиви – перший із назвами полів, які ми застосовували ще у словниках і другий із відповідними їм унікальним ідентифікаторам тегів у тому ж layout-файлі.

Тепер перейдемо до контекстного меню. Для його створення також необхідний кастомний layout-файл, у якому ми прописуємо його елементи. Потім треба призначити меню до конкретного списку:

```
registerForContextMenu(listComponentTimetable);
```

Після цього треба перезаписати 2 методи: “onCreateContextMenu” та “onContextItemSelected”. У першому ми позначаємо, який файл ресурсів використовувати для контекстного меню, а у другому – що робити при натисканні на його елемент.

Ось який код виконується при натисканні на редагування обраного об’єкта:

```
View testView1 = info.targetView;

TextView testTextView1 = testView1.findViewById(R.id.id_item_component);

long selectedIdComponent1 =
Long.parseLong(testTextView1.getText().toString());

intent.putExtra("Type action", "Edit");
intent.putExtra("Id component", selectedIdComponent1);
startActivityForResult(intent, 1);
```

Ми отримаємо View обраного елементу, тобто його представлення у файлі ресурсів та знаходимо у ньому тег, у якому зберігається id об’єкту і передаємо його до спеціального Activity за допомогою методу “startActivityForResult”. Воно необхідно для того, щоб після введення у ньому даних та збережені об’єкта в базі даних воно автоматично закривалось, повертало до попереднього Activity та виконувало у ньому певний код у спеціальному методі “onActivityResult”:

```
if (type_component.equals("Subject_Type_Teacher")) {
    create_or_change_list_of_SubjectTypeTeacher(listComponentTimetable);
}
else if (type_component.equals("Building_Floor_Hall")) {
    create_or_change_list_of_BuildingFloorHall(listComponentTimetable);
}
```

У ньому ми визначаємо, об’єкт якого класу ми змінювали та перетворюємо адаптер у нашій функції, заповнюючи його новими даними. Це потрібно для оновлення інформації у списку. Видалення елементу проходить без додаткового Activity, тобто просто видаляється об’єкт з бази даних і так само оновлюється список.

4. Розклад

Екран із розкладом представлений окремим Activity, у якому певна кількість вкладок, у кожній з яких відображається свій список. Кількість та назва вкладок залежить від учбових днів тижня, які зазначені у розкладі, а

кількість елементів у кожному списку – від максимального номеру пари. Розберемо детальніше, як цей екран створюється та оновлюється.

Для початку ми отримаємо з бази даних усі необхідні об'єкти розкладу. Але нам вони треба у вигляді двовимірного списку, у якому кожний внутрішній список буде зберігати у собі інформацію про кожну пару, а сам позначати день обраного типу тижня. Для цього ми використовуємо окремий клас "ItemTimeTable", у якому ми за допомогою запиту до бази даних зберігаємо не зовнішні ключі, які у класі "MainTimeTable", а властивості тих об'єктів, які відповідають цим ключам, створюємо двовимірний список із пустими елементами та замінюємо їх отриманими об'єктами таким чином, щоб, наприклад, перший день тижня був на нульовому місці у зовнішньому списку, а перша пара відповідно – у внутрішньому списку:

```
List<ItemTimeTable> testWeekItemTimeTable =
mainTimeTableDao.getWeekItemTimeTable (selectedTypeWeek);

for (int i = 0; i < sizeUsingDayWeek; i++) {
    ArrayList<ItemTimeTable> listZeroObject = new ArrayList<ItemTimeTable>
(sizeTimeLesson);
    for (int q = 0; q < sizeTimeLesson; q++) {
        listZeroObject.add(ZeroObjectItemTimeTable);
    }
    sortedWeekItemTimeTable.add(listZeroObject);
}
for (int j = 0; j < testWeekItemTimeTable.size(); j++) {
    ItemTimeTable cycleObject = testWeekItemTimeTable.get(j);
    int indexDayWeekCycleObject = arrOfIdDayWeek.indexOf
(cycleObject.idDayWeek);
    int indexTimeLessonCycleObject = cycleObject.numberLesson - 1;
    sortedWeekItemTimeTable.get (indexDayWeekCycleObject).set
(indexTimeLessonCycleObject, cycleObject);
}
```

Потім ми знаходимо тег “ViewPager” у layout-файлі нашого Activity, створюємо об’єкт класу “ViewPagerAdapter”, який унаслідується від “PagerAdapter” та присвоюємо його тегу:

```
viewPager = findViewById(R.id.view_pager_main_timetable);
adapterViewPager = new ViewPagerAdapter(TimeTableActivity.this,
sortedWeekItemTimeTable, allUsingDayWeek);
viewPager.setAdapter(adapterViewPager);
```

І наприкінці визначаємо “таби”, тобто блок із назвами вкадок, який визначає, де саме зараз знаходиться користувач і який можна використовувати для навігації:

```
TabLayout tabLayout = findViewById(R.id.sliding_tabs_main_timetable);
tabLayout.setupWithViewPager(viewPager);
```

Тепер перейдемо безпосередньо до самого класу “ViewPagerAdapter”. Параметрами його конструктора є поточне Activity, яке потрібно у якості контексту, отриманий список із інформацією про розклад та список усіх днів тижня. У самому класі ми визначаємо кількість вкладок:

```
@Override
public int getCount() {
    return allUsingDayWeek.size();
}
```

Потім визначаємо назву “табів” в залежності від позиції вкладки:

```
@Override
public CharSequence getPageTitle(int position) {
    return allUsingDayWeek.get(position).nameDay;
}
```

І створюємо та вертаємо об’єкт класу “View”, який і буде виводити усю інформацію. Для цього ми спочатку знаходимо відповідний layout-файл, тег списку у ньому, додаємо до нього контекстне меню та об’єкт класу SimpleAdapter, створення якого аналогічне із списком компонентів, тільки список з даними та використані файли ресурсів інші.

У цих списках також присутнє контекстне меню, за допомогою якого можна призначити та перепризначити об’єкти розкладу, а також повернути

значення за замовчуванням, тобто позначити відсутність занять. Перших двох випадків викликається діалогове вікно із списком усіх об'єктів певного типу і натискання на один з них відповідно редагує інформацію ц базі даних та оновлює список із розкладом. Очистка просто повертає перші дефолтні об'єкти, у кому зазначена відсутність заняття і так само оновлює список. Але хочеться більш детально зупинитися на самому процесі оновлення. У попередньому пункті це проходило за допомогою спеціального Activity, яке дозволяло виконувати певні дії на попередньому екрані після свого закриття. Проти тут ми використовуємо діалогове вікно, і щоб виконати код для оновлення списку, необхідно створити спеціального "слухача", який буде відстежувати виклик певної функції у діалоговому вікні та виконувати її код у Activity. Ось код для визначення слухача:

```
public interface AppointComponentsListener {

    void syncAppointComponents();
}
public AppointComponentsListener listenerAppointComponents;
```

Потім нам необхідно переписати метод "onAttach", який використовується для зв'язку вікна із Activity:

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    try {
        listenerAppointComponents = (AppointComponentsListener) context;
    } catch (ClassCastException castException) {
    }
}
```

І після цього викликаємо функцію для оновлення після того, як діалогове вікно зачиниться:

```
listenerAppointComponents.syncAppointComponents();
```

А саме тіло функції ми описуємо вже у Activity, і воно аналогічне створенню розкладу.

2.6. Логіка Web-застосунку

Ми описали логіку Android-застосунку. Тепер перейдемо до тієї Web-частини сервісу, яка генерує та надає інформацію про розклад для нього.

1. Експорт даних у форматі JSON

Для початку нам треба створити адрес, за яким будуть доступні відповідні записи з бази даних. Пишемо у url.py таку строку:

```
path('load_data', views.GetRequest.as_view())
```

Тепер на треба у файлі view.py прописати клас GetRequest, який ми зазначили у якості обробника.

У цьому класі прописуємо метод get, який буде отримувати параметр request, у якому зберігається інформація про кафедру та групу. Після цього отримуємо усі об’єкти класу TimeTable, у якому зберігаються зовнішні ключи усіх інших компонентів за фільтром із отриманих даних у request:

```
data = request.GET
filter_timetable_list =
TimeTable.objects.filter(IdDepartment__id=data['department'],
                         IdGroup__id=data['group'])
```

Після цього ми створюємо словник, у якому будуть зберігатись пари “назва таблиці-список із її об’єктами”, а у самих списках будуть зберігатись словники із парами “назва поля-його значення”. І у циклі обходимо об’єкти головної таблиці, заповнюємо головний словник інформацією з них та повертаємо його за допомогою методу “JsonResponse”:

```
all_return["MainTimeTable"] = mainTimeTableReturn
all_return["TypeWeek"] = allTW
all_return["DayWeek"] = allDW
all_return["TimeLesson"] = allTL
all_return["SubjectTypeTeacher"] = allSTT
all_return["BuildingFloorHall"] = allBFH

return JsonResponse(all_return, safe=False)
```

Після цього можна зазначений URL використовувати у Android-застосунку.

2.7. Тестування сервісу

Як Web, так і Android-застосунки періодично тестувалися прямо під час розроблення. У випадку сайту – на локальному сервері, у випадку програми для смартфона – на емуляторі операційної системи Android. Це дозволяло перевіряти кожну зміну коду, дивитися, як це впливає на програму, та, якщо були присутні проблеми або баги, швидко їх локалізувати та виправляти.

Проте жодне локальне тестування не може дати повної гарантії працездатності на пристроях користувачів через різні апаратні та програмні конфігурації. Тому необхідно протестувати наш сервіс на реальних пристроях (у випадку із Android-застосунком) та у різних браузерах (у випадку із Web-сайтом).

Тестування сайту буде проводитись у наступних браузерах:

1. Google Chrome, версія 80.0.3987.149;
2. FireFox, версія 62.0.3;
3. Microsoft Edge, версія 80;
4. Opera, версія 67.0.3575.97;
5. Internet Explorer, версія 11.0.29;
6. Safari, версія 13.0;

Тестування мобільної програми буде проводитись на наступних пристроях:

1. Смартфон Meizu 15 Plus (Android 7.0)
2. Смартфон Google Pixel (Android 9.0)
3. Смартфон Samsung Galaxy Note 4 (Android 6.0)
4. Смартфон Xiaomi Mi Max 2 (Android 7.1.2)
5. Смартфон Xiaomi Redmi N (Android 5.1)

6. Смартфон Samsung Galaxy J3 (Android 5.1)

Для тестування сайту використано найпопулярніші браузери останніх версій, а для тестування Android-застосунку – пристрої із версіями операційних систем, які найчастіше зустрічаються серед користувачів.

Для тестування функцій сервісу ми будемо використовувати сценарії використання за діаграмою прецедентів, яка зазначена на рис.1. Також буде використане візуальне тестування для визначення правильності відображення дизайну та інтерфейсу та тестування за допомогою логів, яке дозволить виявити, у якій ділянці коду є проблема.

Проведене тестування не виявило жодних проблем як у Web-застосунку, так і у Android-застосунку. Були здійснені всі варіанти використання і кожний з них відпрацював так, як і було задумано. Усі взаємодії серверу, фронтенду та бази даних як усередині кожної частини сервісу, так і при їх спільній роботі були коректними. Розташування візуальних елементів на екрані повністю співпадало із задуманим макетом.

У підсумку можна зазначити, що тестування було пройдене успішно, жодних доробок чи виправлень проводити не потрібно.

ВИСНОВКИ

У ході проєктної роботи було спроєктовано та розроблено сервіс, який складається з Android та Web застосунків, призначений для перегляду та створення розкладу занять в університеті.

Для цього були виконані наступні завдання:

1. Проаналізувано актуальність сервісу.
2. Спроєктувано модель та окреслено можливості сервісу.
3. Виконано огляд необхідних джерел опису інструментів та технологій, які потрібні для розроблення застосунків.
4. Розроблено дизайн Android-частини сервісу.
5. Написано програмний код для мобільного застосунку.
6. Допрацьовано необхідні функції до бекенд частини Web-застосунку.
7. Успішно протестовано роботу всіх частин сервісу.

Все це дозволило нам створити зручний та легкий сервіс, який дозволяє охопити майже всіх потенційних користувачів, тобто студентів та викладачів, а також надає доступ до розкладу двома способами, які доповнюють один одного, і, що найголовніше, не заважають і не залежать один від одного.

Як показало тестування, його вже можна використовувати у межах Херсонського Державного Університету. Проте універсальність дозволяє також користуватись ним у інших вищих навчальних закладах, школах, гімназіях, коледжах тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft Excel [Електронний ресурс] // Википедия – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_Excel.
2. JSON [Електронний ресурс] // Википедия – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/JSON.
3. Java [Електронний ресурс] // Википедия – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Java.
4. Распространение Java [Електронний ресурс] // Java – Режим доступу до ресурсу: https://www.java.com/ru/download/faq/distribution.xml.
5. Widman J. The Most Popular Programming Languages of 2019 [Електронний ресурс] / Jake Widman // New Relic. – 2019. – Режим доступу до ресурсу: https://blog.newrelic.com/technology/most-popular-programming-languages-of-2019/.
6. Kotlin [Електронний ресурс] // Википедия – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Kotlin.
7. SQLite [Електронний ресурс] // Википедия – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/SQLite.
8. Smirnov D. SQLite/Преимущества [Електронний ресурс] / Denis Smirnov // FreeSource. – 2005. – Режим доступу до ресурсу: http://freesource.info/wiki/SQLite/Preimushhestva&.
9. Android Studio [Електронний ресурс] // Википедия – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Android_Studio.
10. Android Studio [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/studio/features.

11. Create an Android project [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/basics/firstapp/creating-project.

12. Run your app [Електронний ресурс] // Google Developers. – 2019. – Режим доступу до ресурсу: https://developer.android.com/training/basics/firstapp/running-app.

13. Build a simple user interface [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/basics/firstapp/building-ui.

14. Start another activity [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/basics/firstapp/starting-activity.

15. Request App Permissions [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/permissions/requesting.

16. Introduction to Activities [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/guide/components/activities/intro-activities.

17. Create a fragment [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/basics/fragments/creating.

18. Room Persistence Library [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture/room?hl=RU.

19. Save data in a local database using Room [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room?hl=RU.

20. Defining data using Room entities [Електронний ресурс] // Google Developers. – 2019. – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room/defining-data?hl=RU.

21. Define relationships between objects [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room/relationships?hl=RU.

22. Accessing data using Room DAOs [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room/accessing-data?hl=RU.

23. DialogFragment [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/reference/android/app/DialogFragment.

24. Макеты [Електронний ресурс] // Google Developers. – 2019. – Режим доступу до ресурсу: https://developer.android.com/guide/topics/ui/declaring-layout.

25. Linear Layout [Електронний ресурс] // Google Developers. – 2019. – Режим доступу до ресурсу: https://developer.android.com/guide/topics/ui/layout/linear.

26. Create a Java class or type [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/studio/write/create-java-class.

27. Add app resources [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/studio/write/add-resources.

28. Build a UI with Layout Editor [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/studio/write/layout-editor.

29. Create and manage virtual devices [Електронний ресурс] // Google Developers. – 2019. – Режим доступу до ресурсу: https://developer.android.com/studio/run/managing-avds.

30. Add build dependencies [Електронний ресурс] // Google Developers. – 2020. – Режим доступу до ресурсу: https://developer.android.com/studio/build/dependencies.

31. Еккель Б. Философия Java / Брюс Еккель., 2019.

32. Шилдт Г. Java 8. Руководство для начинающих / Герберт Шилдт., 2018.

33. Блох Д. Java. Эффективное программирование / Джошуа Блох., 2014.

34. Мартін Р. Чистый код. Создание, анализ и рефакторинг / Роберт Мартін., 2010.

35. Ed B. Hello, Android / Burnette Ed., 2017. – 302 с.

ДОДАТКИ

Додаток А

Рис. 1 Вибір тижня

Рис. 2 Вибір днів

Рис. 3 Номер найпізнішої пари

Рис. 4 Час початку та тривалість занять та перерв

Додаток Б

Рис. 1 Головне меню

Рис. 2 Діалогове вікно вибору типу тижня

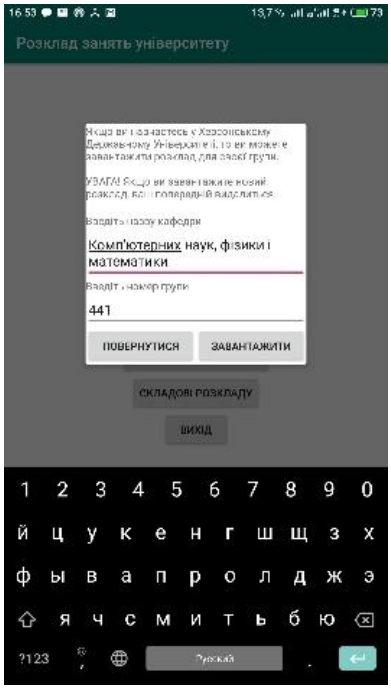

Рис. 3 Діалогове вікно завантаження розкладу

Рис. 4 Меню компонентів розкладу

Додаток В

Рис. 1 Список місць проведення занять

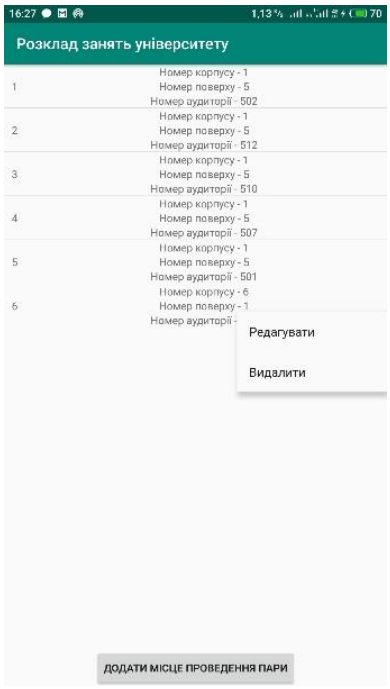

Рис. 2 Список місць проведення занять із активним контекстним меню

Рис. 3 Вікно для створення/редагування об’єкту списку

Додаток Г

Рис. 1 Екран із розкладом

Рис. 2 Контекстне меню для списку розкладу

Рис. 3 Діалогове вікно із предметами та викладачами

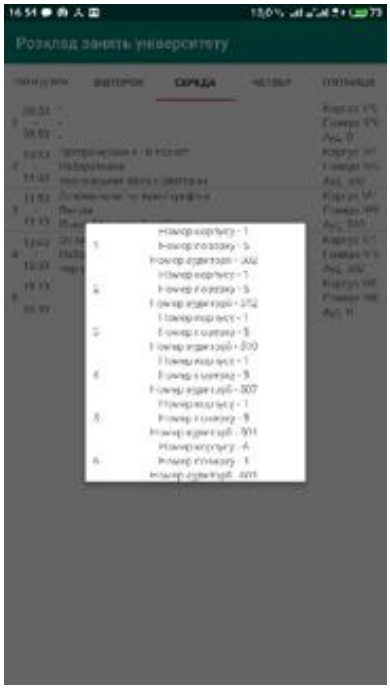

Рис. 4 Діалогове вікно із місцями проведення занять

Додаток Д

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО
ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Керода Дмитро Олександрович,
учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

09.04.2020 (дата) | (підпис) | Дмитро Керода (ім'я, прізвище)