Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

**Бакалаврська кваліфікаційна робота на тему:**

# «ІНФОРМАЦІЙНО-ДОВІДКОВА СИСТЕМА ГУРТКІВ МІСТА ВІННИЦІ»

Виконала: студентка 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____ Ганна ШПОРТ

Керівник: к.т.н., ас. каф. САІТ
_____ Ігор ШТЕЛЬМАХ
«06» 06 2025 р.

Рецензент: к.т.н., ас. каф. КН
_____ Ілона БОГАЧ
«12» 06 2025 р.

**Допущено до захисту**

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"28" 05 2025 року

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**
Шпорт Ганні Юріївні

1. Тема роботи: «Інформаційно-довідкова система гуртків міста Вінниці»
керівник роботи: Штельмах І. М., к.т.н., ас. каф. САІТ
затверджені наказом ВНТУ від «20» 05 2025 року № 97
2. Термін подання студентом роботи 31.05.25 р.
3. Вихідні дані до роботи:
1) Дані про гуртки міста Вінниця.
4. Зміст текстової частини:
1) Аналіз предметної області;
2) Вибір оптимальної ІТ-інфраструктури;
3) Розроблення інформаційно-довідкової системи гуртків міста Вінниці.
5. Перелік ілюстративного матеріалу:
1) Діаграма use case;
2) Діаграма взаємодії;
3) Діаграма варіантів використання;.
4) Діаграма класів;
5) Блок-схема алгоритму роботи мобільного додатка;
6) Діаграма взаємодії між пристроєм та хмарою.
7) Сторінка Clubs
8) Сторінка Plan
9) Сторінка Map

6. Консультанти розділів роботи:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 3 | Крижановський Є. М. | | |
| | | | |
| | | | |

7. Дата видачі завдання 28.05.25р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1 | Аналіз предметної області | 01.04 | 15.04 | вик |
| 2 | Вибір оптимальної IT-інфраструктури | 15.04 | 30.04 | вик |
| 3 | Розроблення інформаційно-довідкової системи гуртків міста Вінниці | 01.05 | 10.05 | вик |
| 4 | Тестування інформаційно-довідкової системи гуртків міста Вінниці | 10.05 | 20.05 | вик |
| 5 | Оформлення матеріалів до захисту БКР | 20.05 | 30.05 | вик |

Студент _____ Ганна ШПОРТ

Керівник роботи _____ Ігор ШТЕЛЬМАХ

# АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 85 сторінок формату А4, на яких є 34 рисунка, список використаних джерел містить 25 найменувань.

Метою роботи є розроблення інформаційно-довідкової системи гуртків міста Вінниці.

В роботі наведено етапи розроблення інформаційно-довідкової системи у вигляді мобільного додатку на платформі Android.

Проведено аналіз предметної області, що підкреслює актуальність розробки мобільного додатку для пошуку гуртків. Знайдено декілька аналогів для розуміння основних функцій, які можна впровадити в нашу систему. Зібрано необхідну кількість даних про гуртки в межах міста Вінниці.

Було вибрано оптимальну IT-інфраструктуру для якісного розроблення інформаційно-довідкової системи. Проаналізовано вибір мови Kotlin, мови розмітки XML, середовища розробки Android Studio, методу збереження даних за допомогою Room та картографічного сервісу Google maps SDK. Для функціонування системи визначено необхідне апаратне забезпечення.

Розроблено архітектуру інформаційної системи у вигляді UML діаграм та алгоритму роботи Android-застосунку. Здійснено програмну реалізацію мобільного додатку та тестування системи.

Ключові слова: інформаційно-довідкова система, мобільний додаток, Kotlin, Android, гуртки.

# ABSTRACT

The bachelor's qualification work consists of 85 pages of A4 format, on which there are 34 figures, the list of used sources contains 25 names.

The aim of the thesis is to develop an information and reference system for clubs in the city of Vinnytsia.

The work presents the stages of developing an information and reference system in the form of a mobile application on the Android platform.

An analysis of the subject area was conducted, which emphasizes the relevance of developing a mobile application for searching for clubs. Several analogues were found to understand the main functions that can be implemented in our system. The necessary amount of data about clubs within the city of Vinnytsia was collected.

The optimal IT infrastructure was selected for high-quality development of an information and reference system. The choice of the Kotlin language, XML markup language, Android Studio development environment, data storage method using Room and the Google maps SDK cartographic service were analyzed. The necessary hardware was determined for the system to function.

The architecture of the information system was developed in the form of UML diagrams and the algorithm of the Android application. The software implementation of the mobile application and system testing were carried out.



Keywords: information and reference system, mobile application, Kotlin, Android, clubs.

# ЗМІСТ

# ВСТУП

**Актуальність теми**. У сучасному світі мобільні технології глибоко проникли в усі сфери повсякдення, ставши не просто засобом зв'язку, а й потужним інструментом для вирішення різноманітних завдань. Наразі більшість людей не можуть уявити й дня без смартфону поруч, ця статистика відкриває безмежні можливості для розроблення інноваційних рішень, покликаних полегшити практично кожен аспект нашого життя. Одним з таких аспектів є організація дозвілля, зокрема, процес зручного пошуку гуртків та інших занять, як для дітей, так і для дорослих. На сьогоднішній день пошук відповідного гуртка часто стає складним процесом, що включає перегляд численних веб-сайтів, соціальних мереж або особисті візити до різних закладів.

**Мета і задачі дослідження.** Метою роботи є розроблення інформаційно-довідкової системи пошуку гуртків у межах міста Вінниці, шляхом реалізації довідкової системи у вигляді зручного мобільного додатку для платформи Android.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

- проаналізувати предметну область;
- визначити оптимальну IT-інфраструктуру;
- розробити інформаційно-довідкову систему, що забезпечує зручний пошук гуртків;
- оцінити працездатність та стабільність розробленої системи шляхом її тестування.

**Об’єктом дослідження** є процес створення інформаційно-довідкової системи для пошуку гуртків у місті Вінниці.

**Предметом дослідження** є методи і програмні засоби створення інформаційно-довідкової системи для пошуку гуртків у місті Вінниці.

**Публікації.** За результатами виконання даної роботи опубліковано тези доповідей на тему: «Інформаційно-довідкова система гуртків міста Вінниці» на LIV Всеукраїнській науково-технічній конференцій факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.) [1].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз предметної області

Згідно зі звітності Державної служби статистики України станом на перше січня 2024 року, зведеною по закладах системи МОН України та приватних закладах, в Україні налічується всього 51869 гуртків, груп та інших творчих об'єднань, в яких числяться 798854 вихованців, учнів, слухачів [2]. В обласних центрах скупчення таких позашкільних закладів найбільша відповідно пошук необхідного гуртка стає складнішим адже варіантів вибору дуже багато.

Часто користувачі, які бажають знайти гуртки та секції для себе або своїх дітей, стикаються з низкою проблем. А саме відчувають значні незручності, спричинені розпорошеністю інформації, адже відомості про різні гуртки часто розкидані по численних веб-сайтах, соціальних мережах, рекламних матеріалах або передаються неофіційними каналами, що ускладнює процес пошуку та об'єктивного порівняння доступних варіантів. Відсутність централізованого ресурсу, єдиного інформаційного простору, де була б зібрана повна та актуальна інформація про всі гуртки в межах міста, ще більше ускладнює завдання. Крім того, наявні методи пошуку часто не забезпечують ефективної фільтрації та сортування за важливими критеріями, такими як напрямок діяльності, розклад занять або вартість. Недоступність швидкого пошуку гуртка на карті може значно сповільнити спроби користувача знайти заняття в потрібному районі міста.

У зв'язку з цими проблемами існує чітка потреба у створенні зручного та ефективного інструменту, який би дозволив користувачам швидко та легко знаходити гуртки, що відповідають їхнім потребам та вподобанням.

Посилаючись на ту ж звітність, яка згадувалась раніше, створимо діаграму, на якій зображено кількість учнів різного віку, що відвідують позакласні заняття та гуртки (рисунок 1.1). Найбільшу активність спостерігаємо серед наймолодших дітей, яким до шести років. Також високі показники у віковій категорії до 10 років. З віком кількість учнів, які відвідують гуртки, поступово

зменшується. Помітне зниження починається після 13 років, і найменше число учасників у віковій групі «18 років і більше» — лише 9 866 осіб.

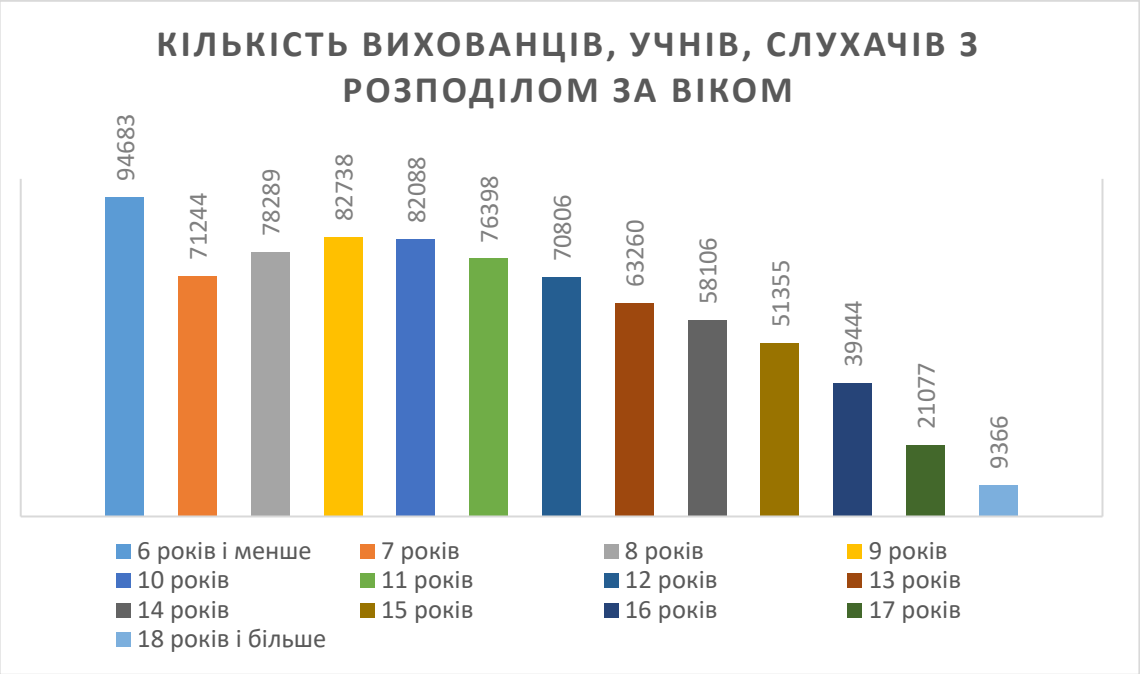



Рисунок 1.1 - Кількість вихованців, учнів, слухачів з розподілом за віком

Це свідчить про те, що найбільше зацікавлення гуртками припадає на молодший шкільний вік, а з віком, можливо через зміну пріоритетів або навантаження, кількість відвідувачів знижується. Аналізуючи дану статистику розуміємо, що одними з основних користувачів мобільного застосунку є батьки, які зацікавлені у пошуку та організації дозвілля для своїх дітей. Наступною чисельною групою користувачів є підлітки, які прагнуть знайти хобі, розширити свої знання та навички, або займатися спортом у місцевих гуртках та секціях. Крім того, потенційними користувачами можуть бути тимчасові мешканці міста або гості, які бажають ознайомитися з доступними можливостями для проведення вільного часу.

Отже, інформаційно-довідкова система у вигляді застосунку має бути орієнтована на широке коло користувачів, незалежно від їхнього рівня володіння технологіями. Основне завдання - розробити інтуїтивно зрозумілий інтерфейс та зручну навігацію, завдяки яким кожен вінничанин зможе легко знайти заняття, що відповідатимуть його вимогам.

## 1.2 Огляд аналогів

На ринку існують певні онлайн-платформи та веб-сайти, які пропонують інформацію про гуртки та секції. Однак вони часто мають обмежений функціонал та не завжди адаптовані для мобільних пристроїв або охоплюють лише певну категорію клубів. Саме мобільні додатки, що спеціалізуються на пошуку гуртків у місті є менш поширеними. Для кращого розуміння ринку та оцінки переваг розроблюваного застосунку було проаналізовано кілька подібних мобільних додатків.

Спочатку розглянемо застосунок Meetup, зображений на рисунку 1.2. Це платформа, яка дозволяє користувачам створювати та приєднуватися до груп, організованих за спільними інтересами, хобі, професіями тощо. Вона використовується для організації як онлайн, так і офлайн зустрічей та заходів. Організатори можуть створювати та рекламувати зустрічі, семінари, лекції, соціальні заходи тощо. MeetUp дещо перетинається з ідеєю клубного застосунку. Однак MeetUp є ширшою платформою для різних заходів та спільнот, а не спеціалізується на освітніх чи розвиваючих клубах, особливо для дітей [3].

Основними з функціональних можливостей Meetup є:

- створення власних груп за інтересами;
- створення анонсів майбутніх зустрічей:
- пошуку за ключовими словами, категоріями, місцем розташування та датою;
- реєстрація на події, можливість вказувати кількість учасників;
- нагадування про майбутні події для зареєстрованих учасників;
- можливість переписки з іншими користувачами;
- залишення відгуків про відвідані події та оцінювання організаторів.

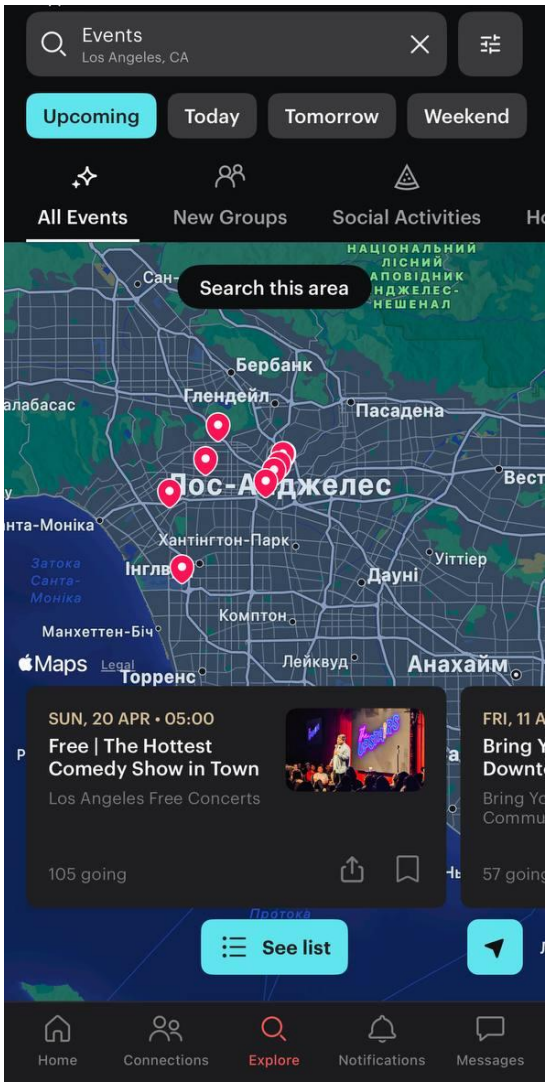



Рисунок 1.2 – Інтерфейс додатку Meetup

Наступний додаток - ClassPass, зображений на рисунку 1.3. Це платформа для пошуку та бронювання занять фітнесом, салонів краси та оздоровчих послуг за підпискою або одноразово. Користувачі можуть створювати групи, які відповідають їхнім інтересам у своєму місті або поблизу. Хоча ClassPass зосереджений на спорті, його модель пошуку за місцезнаходженням, категорією (наприклад, йога, танці), розкладом та відгуками є дуже актуальною для пошуку занять. Інтерфейс користувача ClassPass відзначається зручністю та візуальною привабливістю [4].

Основними функціональними можливостями ClassPass є:

– пошук класів за розташуванням, часом, типом активності, інструктором;
– бронювання та скасування занять;

– створення профілю, перегляд історії відвідувань;
– надсилання сповіщень про майбутні заняття;
– використання геолокації для пошуку студій поблизу;
– забезпечення способів оплати.

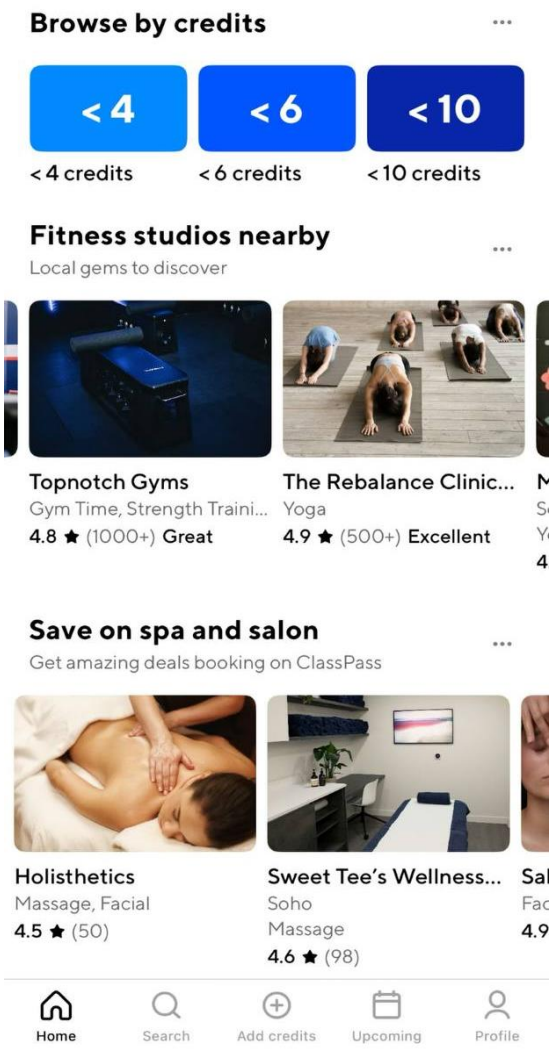



Рисунок 1.3 – Інтерфейс додатку ClassPass

Ще один застосунок Locals, показаний на рисунку 1.4, позиціонує себе як платформа, що дозволяє творцям контенту створювати власні спільноти на основі підписки. Однак, він також дозволяє знаходити групи за інтересами, з якими можна проводити час у реальному житті [5].

Основними функціональними можливостями Locals є:

– використання GPS для відображення подій поруч;

– підбір персоналізованих рекомендацій на основі попередньої активності;
– система сповіщень;
– зручний пошук подій, груп та інших користувачів;
– перегляд місця події на карті;
– можливість завантажувати медіа в профіль або групи;
– можливість переписки в особистих повідомленнях або групових чатах;
– створення власного профілю з інформацією про вподобання.

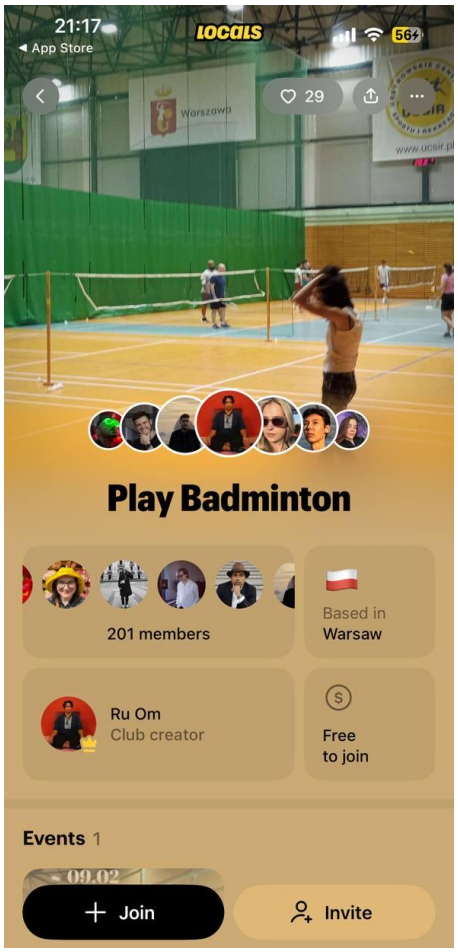



Рисунок 1.4 – Інтерфейс додатку Locals

Проаналізувавши ці додатки, отримали цінні ідеї щодо функціональності, дизайну та зручності використання, які можна реалізувати в інформаційно-довідковій системі гуртків міста Вінниці.

## 1.3 Формування технічного завдання

На основі аналізу предметної області та аналогів функціональності інших додатків була розроблена концепція створення власного функціонального та зрозумілого застосунку для пошуку у місті.

Перш за все метою цієї роботи є розроблення інформаційно-довідкової системи гуртків міста, яка забезпечить вінничан зручним інструментом для пошуку та вибору секцій різної спрямованості.

Основні функціональні вимоги розробленого мобільного застосунку полягають у наданні користувачам ефективних інструментів для пошуку, вибору та організації відвідувань гуртків у межах міста.

Перш за все, користувач повинен мати можливість шукати гуртки за різноманітними критеріями, включаючи їх тематичну спрямованість, вартість навчання, дні тижня, коли проводяться заняття, та години роботи. Це дозволить звузити пошук до найбільш релевантних варіантів, враховуючи особисті вподобання, фінансові можливості та розклад.

Для кожного гуртка в системі слід надати детальну інформацію. Вона повинна включати назву, контактну інформацію, адресу розташування та посилання на сайт або соціальні мережі.

Крім того, важливою вимогою є наявність зручного інструменту для візуального пошуку клубів на карті міста. Інтерактивна карта дозволить користувачам орієнтуватися в географічному розташуванні занять, оцінювати їх доступність та вибирати ті, що знаходяться в зручному для них районі.

Щоб користувач не забувся, які гуртки йому сподобались, необхідно розробити функцію додавання до збережених. Це дозволить швидко повернутись до цікавих варіантів без необхідності повторного пошуку.

Однією з корисних функцій додатку буде можливість запланувати відвідування гуртка на певну дату та час. Таким чином користувач зможе легко організувати свій час та скласти особистий розклад занять.

Щодо технічної реалізації, для розробки мобільного застосунку планується використання мови програмування Kotlin з застосуванням Android Studio. Для

зберігання та обробки даних про гуртки необхідно спроєктувати локальну базу даних, що забезпечить завантаження гуртків без необхідності виходу в інтернет. Для відображення карти найкращим варіантом буде використати Google Maps API.

Розроблений мобільний застосунок повинен демонструвати високу продуктивність, забезпечуючи швидкий час завантаження екранів та оперативне відображення інформації, особливо при пошуку та фільтрації даних. Інтерфейс застосунку повинен бути інтуїтивно зрозумілим для користувачів різного віку та рівня технічної підготовки. Навігація між розділами має бути простою та логічною. Застосунок повинен бути сумісним з більшістю сучасних Android-пристроїв.

## 1.4 Збір даних

Для пошуку достовірної інформації про гуртки міста Вінниці здебільшого користувались сайтом Вінницької міської ради [6] та Google Maps.

База даних міститиме 40 гуртків, для кожного буде відповідна назва, адреса, номер телефону, посилання на сайт, категорії за інтересами, днями тижня, робочими годинами, приблизною ціною, а також довгота та широта. В додатку основною інформацією про гурток будуть назва, адреса, посилання на сайт та номер. Категорії за інтересами, днями тижня, робочими годинами, ціною потрібні для функціоналу фільтрування гуртків. Оскільки ціни в межах гуртка можуть варіюватись, зробили узагальнюючий поділ на безкоштовні гуртки, гуртки, де година заняття коштує менше ста гривень й більше ста гривень. Довгота та широта необхідні для точного відображення гуртків та закладів на вбудованій мапі.

Таким чином створили таблицю в Ecxel (рис. 1.5), в якій зібрали усю необхідну інформацію про гуртки Вінниці.

| | name | address | phone | website | interests | days | hours | price | longitude | latitude |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Майстерня Novikova Paint&Craft | м. Вінниця, вул. Хлібна 14 | (073) 419 32 22 | https://www.paintcraftworkroom.com/ | Малювання | Сб-Нд | 9:00-17:00 | Більше 100 грн/год | 28.46022603367229 | 49.23368828782602 |
| 2 | Центр розвитку дитини "Бджілка" | м. Вінниця, вул. Ватутіна, 27 | (063) 286 82 82, (068) 644 00 44 | http://www.bdjilka.vn.ua/ | Саморозвиток | Пн-Нд | 9:00-18:00 | До 100 грн/год | 28.534603598090545 | 49.24547054738144 |
| 3 | Дитячий центр "BeFirst.Kids" | м. Вінниця, вул. Марії Литвиненко-Вольгемут, 26 | (067) 702-6045, (093) 093-9141 | www.befirst.in.ua | Театральні гуртки | Пн-Нд | 9:00-18:00 | До 100 грн/год | 28.45011871158231 | 49.226701299307344 |
| 4 | Курси акторської майстерності "Червоний Крокодил" | м. Вінниця, вул. Миколи Оводова, 22 | (067) 184 60 53 | https://www.facebook.com/schoolredcroco | Театральні гуртки | Пн-Пт | 9:00-19:00 | Більше 100 грн/год | 28.46878186925433 | 49.23406534814523 |
| 5 | Вінницька дитяча музична школа №1 | м. Вінниця, вул. Архітектора Артинова, 21 | (0432) 562-830 | https://msh1.edu.vn.ua/ | Музика, вокал | Пн-Пт | 9:00-17:00 | До 100 грн/год | 28.466537698090193 | 49.234427990744194 |
| 6 | Вінницька дитяча музична школа №2 | м. Вінниця, вул. Червоноармійська, 16 | (0432) 261-456, (0432) 261-637 | vdmsh2.vn.ua | Музика, вокал | Пн-Пт | 9:00-17:00 | До 100 грн/год | 28.486810098090743 | 49.24682822502492 |
| 7 | Artinov Studio, приватна музична школа | м. Вінниця, вул. Василя Стуса 26 | (097) 776 00 68 | https://studio.artinov.school/ | Музика, вокал | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.457395598090013 | 49.22861702404529 |
| 8 | Вінницька дитяча школа мистецтв | м. Вінниця, вул. Островського, 85 | (0432) 275-095 | schoolarts.vn.ua | Музика, вокал | Пн-Пт | 9:00-17:00 | До 100 грн/год | 28.491826226926126 | 49.23295157177888 |
| 9 | Школа програмування для дітей "ROBOCODE" | м. Вінниця, вулиця Академіка Ющенко, 6 | (093) 170-6445 | robocode.ua | Технічні гуртки | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.443050682746133 | 49.2204667925482 |
| 10 | Школа "Гарант" | м. Вінниця, вул. Київська, 134 | (067) 440-5546, (093) 884-7201 | https://garant-school.com/ | Технічні гуртки | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.47401231343488 | 49.26363910643602 |
| 11 | Комп'ютерна академія "ITSTEP" | м. Вінниця, пл. Гагаріна, 2 | (067) 522-1255, (073) 797-8831 | https://garant-school.com/ | Технічні гуртки | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.455978769254354 | 49.233235176881955 |
| 12 | Бізнес-школа для дітей "Rainbow" | м. Вінниця, вул. Пирогова 52а | (067) 631-4351 | https://rainbow-vinnytsia.webnode.com.ua/ | Саморозвиток | Сб-Нд | 9:00-18:00 | До 100 грн/год | 28.449325384597692 | 49.228176653770475 |
| 13 | Танцювальна студія "Альфа-Денс" | м. Вінниця, вул. 600-річчя, 66-Б | (096) 035-16-68 | https://alphadance.com.ua/ | Танці | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.423809453910312 | 49.22257249532725 |
| 14 | Спорткомплекс "Аква-Він" | вул. Академіка Янгеля, 48 | (097) 936-8427 | https://vezha.ua/abonement | Плавання | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.485370369254774 | 49.24452615795192 |
| 15 | SOUL DANCE CENTRE | м. Вінниця, вул. Соборна, 8 | (063) 695-0664 | http://souldancecentre.com/ | Танці | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.474554011582597 | 49.23366670997444 |
| 16 | СДЮСШ з легкої атлетики | м. Вінниця, вул. Генерала Арабея, 3 | (043) 235-4245 | https://ua.prostobaby.com.u | Спортивні секції | Пн-Пт | 9:00-18:00 | Безкоштовні | 28.439248153911045 | 49.239684075835456 |
| 17 | ВІННИЦЬКА ОБЛАСНА ФЕДЕРАЦІЯ ВЕСЛУВАННЯ НА БАЙДАРКАХ І КАНОЕ | м. Вінниця вул. Князів Коріатовичів, 123 | (097) 870-0600 | https://canoe-sprint.vn.ua/sh | Спортивні секції | Пн-Нд | 9:00-18:00 | Безкоштовні | 28.4629392557617 | 49.22581524238848 |
| 18 | СДЮСШ з баскетболу | м. Вінниця, вул. Пирогова, 4 | (067) 431-6546 | https://edusearch.com.ua/vn | Спортивні секції | Пн-Пт | 9:00-19:00 | Безкоштовні | 28.45615296740304 | 49.23221061472379 |
| 19 | СДЮСШ з акробатики | вул. Червонохрестівська, 11 | (043) 269-7968 | https://ua.prostobaby.com.u | Спортивні секції | Пн-Пт | 9:00-19:00 | Безкоштовні | 28.47453762692616 | 49.234700412148364 |
| 20 | Центр Іноземних Мов ASAP | м. Вінниця, вул. Ак. Ющенка, 6 | (096) 390-0090 | http://asap.vn.ua/ | Вивчення мов | Пн-Нд | 9:00-18:00 | Більше 100 грн/год | 28.44298631160597 | 49.22050185091505 |
| 21 | Школа "WizarD" | м. Вінниця, пр. Юності, 61 | (097) 077-4303 | http://wizardschool.com.ua/ | Вивчення мов | Пн-Нд | 9:00-18:00 | Більше 100 грн/год | 28.411884228776916 | 49.22299150157598 |
| 22 | Арт-студія Leonardo | м. Вінниця вул. Андрія Первозванного, 58а | (096) 465 67 51 | https://leonardo.intita.com/index#contacts | Малювання | Пн-Пт | 9:00-19:00 | Більше 100 грн/год | 28.403279260940064 | 49.224489312910706 |
| 23 | New IT School | м. Вінниця, вул. М.Оводова, 22 | (068) 485 39 65, (093) 041 38 18 | https://www.itschool.vn.ua | Технічні гуртки | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.468799668073178 | 49.233837475810645 |
| 24 | Міський палац мистецтв "Зоря" | м. Вінниця, вул. Стрілецька, 44 | (0432) 65-56-12 | http://zoria.vmr.gov.ua/article/38 | Саморозвиток | Пн-Нд | 9:00-19:00 | Безкоштовні | 28.491663135582264 | 49.2483955934113 |
| 25 | Вінницька дитяча художня школа | м. Вінниця, вул. Першотравнева, 66 | (0432)67-60-43 | https://artschool.vn.ua | Малювання | Пн-Нд | 9:00-19:00 | До 100 грн/год | 28.46700686796229 | 49.236894129870976 |
| 26 | Вінницька дитяча школа мистецтв "Вишенька" | м. Вінниця, вул. Порика, 28-Б | (0432) 55-71-50 | https://www.vyshenka.com.ua | Саморозвиток | Пн-Пт | 9:00-19:00 | До 100 грн/год | 28.403908427675393 | 49.232284525022294 |
| 27 | Центр розвитку дітей та юнацтва "Дивосвіт" | м. Вінниця, вул. Стельмаха, 43Б | (068) 984 06 23 | https://divosvit.vn.ua | Саморозвиток | Пн-Пт | 9:00-19:00 | Більше 100 грн/год | 28.39619327097363 | 49.23127672021063 |
| 28 | Вінницький міський палац дітей та юнацтва імені Лялі Ратушної | м. Вінниця, Хмельницьке шосе, 22 | (098) 309 98 39 | https://vmpdu.edu.vn.ua/palac-sogodni | Саморозвиток | Пн-Нд | 9:00-19:00 | До 100 грн/год | 28.440541423895837 | 49.234979286623705 |
| 29 | neshkola - студія розвитку інтелекту | м. Вінниця, вул. Ширшова, 3а | (063) 721-47-03 | https://www.neshkola.space/#sec1 | Саморозвиток | Пн-Нд | 9:00-21:00 | Більше 100 грн/год | 28.485325442430245 | 49.243471150111475 |
| 30 | Kudo club "Katana" | м. Вінниця, проспект Юності, 6А | (097) 404 95 33 | https://www.instagram.com/kudokatana.vn/ | Спортивні секції | Пн-Нд | 9:00-21:00 | Більше 100 грн/год | 28.413219965594934 | 49.231879137751925 |
| 31 | Спортивні Бальні танці "Feel Dance" | м. Вінниця, проспект Юності, 6а | (096) 583 56 47 | https://www.instagram.com/feel_dance_vn/ | Танці | Пн-Нд | 9:00-21:00 | Більше 100 грн/год | 28.412639329371608 | 49.23156880068109 |
| 32 | Дитяче плавання Аквапарк "Маяк" | м. Вінниця, вул. Василя Порика, 28 | (097) 616 26 69 | https://www.instagram.com/swimmingfor_kids/ | Плавання | Пн-Нд | 9:00-21:00 | Більше 100 грн/год | 28.404739915281358 | 49.23157801897967 |
| 33 | Танцювальна студія "IRENE DANCE STUDIO" | м. Вінниця, вул. Андрія Первозванного, 54 | (098) 871 41 54 | https://www.instagram.com/irene_dance_studio?igsh=MXZwcDBiOTJ5MnRxeA%3D%3D | Танці | Пн-Пт | 9:00-19:00 | Більше 100 грн/год | 28.403732989425315 | 49.2233718391226 |
| 34 | Студія іноземних мов та перекладу "De Lingo" | м. Вінниця, вул. Зодчих, 36 | (097) 621 28 58 | https://www.instagram.com/de_lingo_/ | Вивчення мов | Пн-Пт | 9:00-19:00 | Більше 100 грн/год | 28.438003322813046 | 49.21601250222144 |
| 35 | Бразильське джиу-джитсу "Southside" | м. Вінниця, вул. Костянтина Василенка, 16 | (099) 004 94 10 | https://bjjukraine.github.io/vinnytsia/ | Спортивні секції | Пн-Пт | 9:00-19:00 | Більше 100 грн/год | 28.430006392275004 | 49.22019172974815, |
| 36 | Вінницький обласний центр технічної творчості учнівської молоді | м. Вінниця, вул. Шолом-Алейхема, 9 | (043) 267 01 94 | http://octtum.vn.ua/ | Технічні гуртки | Пн-Нд | 9:00-18:00 | Безкоштовні | 28.47798755748648 | 49.23288379709328 |
| 37 | МДЮСШ №6 Занятя шашками | м.Вінниця, вул. Театральна, 24 | (043) 235 32 80 | https://draughtsplay.com | Спортивні секції | Пн-Пт | 9:00-18:00 | Безкоштовні | 28.46234021594008 5 | 49.232570768934444 |
| 38 | Вінницький молодіжний центр "KVADRAT" | м. Вінниця, вул. Театральна, 15 | (098) 398 75 15 | https://www.kvadrat.vn.ua | Саморозвиток | Пн-Нд | 9:00-19:00 | Безкоштовні | 28.46436546723869 | 49.2345563394466 |
| 39 | Бізнес-школа "UNITED" | м. Вінниця, вул. Миколи Оводова, 54 | (067) 631 43 51 | https://www.instagram.com/united.business.school/ | Саморозвиток | Пн-Нд | 9:00-19:00 | Більше 100 грн/год | 28.46885651340658 | 49.23111752876228, |
| 40 | Центр розвитку дитини "MIKLAND" | м. Вінниця, вул. Талаліхіна, 3 | (063) 614 68 01 | https://www.facebook.com/miklandclub/?locale=ru_RU | Саморозвиток | Пн-Пт | 9:00-19:00 | До 100 грн/год | 28.47949032857745 | 49.24609077440791 |

Рисунок 1.5 – Дані про гуртки

Після ці всі дані можна буде безпосередньо додати в базу даних, та використати їх при розробці застосунку.

## 1.5 Висновки

Аналіз предметної області та огляд наявних рішень засвідчили відсутність локальних мобільних застосунків, здатних ефективно забезпечити централізований пошук гуртків для різних вікових груп у межах одного міста. На основі проведеного аналізу сформовано технічне завдання на розробку зручного інформаційно-довідкового програмного забезпечення з функціями пошуку за найкращими критеріями, відображенням на карті, обраними варіантами та плануванням відвідувань. У результаті збору даних було сформовано структуровану таблицю, що містить повну та актуальну інформацію про 40 гуртків Вінниці. Отримані дані забезпечують основу для фільтрації за інтересами, часом, вартістю занять та геолокацією, що дозволяє реалізувати зручний та функціональний пошук гуртків у мобільному додатку.

# 2 ВИБІР ОПТИМАЛЬНОЇ ІТ-ІНФРАСТРУКТУРИ

## 2.1 Аналіз можливостей мови програмування Kotlin

Існує широкий вибір мов програмування для розробки високоякісних додатків. Деякі з них мають інтегровані середовища розробки, що значно спрощує процес розробки. Інші можуть потребувати додаткових інструментів, але вони все ще пропонують потужні можливості.

Kotlin було обрано основною мовою для розробки мобільного застосунку. Це сучасна мова програмування зі статичною типізацією, створена JetBrains та офіційно підтримувана Google для розробки під Android. Мова започаткована командою JetBrains у 2010 році та стала з відкритим вихідним кодом у 2012 році, а перший офіційний реліз відбувся у лютому 2016 року [7].

Мова програмування Kotlin спочатку була створена як удосконалення Java, метою було усунути деякі обмеження Java, зберігаючи при цьому повну сумісність з нею. Хоча Kotlin стала переважно мовою для розробки додатків для Android завдяки своєму лаконічному синтаксису, сучасним функціям та покращеній безпеці, його безшовна інтеграція з Java дозволила отримати широке впровадження в різноманітних програмних проектах. Сьогодні Kotlin використовується не лише для мобільних додатків, але й для розробки серверних компонентів, веб-додатків, програмного забезпечення для робочих столів і навіть серверного програмування, що робить його універсальним інструментом у наборі сучасних розробників[8].

Розглянемо деякі переваги Kotlin:

- Вона більш виразна: це одна з її найважливіших якостей. Ви можете написати більше, використовуючи набагато менше коду;
- Вона безпечніша: Kotlin розроблено, щоб допомогти уникнути типових помилок кодування, які можуть порушити ваш код або залишити в ньому вразливі місця. Мова забезпечує безпеку нуля та усуває помилки виключення нульового покажчика;

- Вона функціональна: Kotlin є об'єктно-орієнтованою мовою, однак, як і багато інших сучасних мов, вона використовує багато концепцій функціонального програмування, таких як лямбда-вирази, для вирішення деяких проблем набагато легшим способом. Ще однією приємною особливістю є те, як вона працює з колекціями;
- Вона використовує функції розширення: це означає, що ми можемо розширювати будь-який клас новими функціями, навіть якщо у нас немає доступу до вихідного коду;
- Вона має високу сумісність: можна продовжувати використовувати більшість бібліотек та коду, написаних на Java, оскільки сумісність між обома мовами чудова бо вони компілюються в той самий байт-код. Можливо навіть створювати змішані проекти, де файли Kotlin та Java співіснують;
- Класи даних : у Kotlin є класи даних, які призводять до автоматичної генерації шаблонів, таких як equals, hashCode, toString, геттери/сеттери та багато іншого.
- Відкритий код: мова програмування Kotlin, включаючи компілятор, бібліотеки та всі інструменти, є повністю безкоштовною та з відкритим кодом і доступна на github;
- Підтримка інструментів: Kotlin підтримує інструментарій від Android з інструментами, оптимізованими для розробки Android, включаючи Android Studio, Android KTX, Java IDE – IntelliJ IDEA, Eclipse і Android SDK [9].

Kotlin та Java — це статично типізовані мови програмування загального призначення. Статично типізована – це характеристика мови програмування, яка означає, що тип кожної змінної та виразу відомий під час компіляції, однак вона не вимагає явного вказування типу кожної оголошуваної змінної. Хоча Java була фундаментальною мовою в розробці додатків для Android та корпоративних систем протягом десятиліть, Kotlin стала більш сучасною альтернативою, розробленою для подолання багатьох обмежень Java. Незважаючи на відмінності в синтаксисі, вона повністю сумісна з Java. Це означає, що Kotlin може

безперешкодно працювати з існуючими кодовими базами та бібліотеками Java. Таким чином розробники можуть інтегрувати Kotlin в застарілі проекти без повного переписування. Крім того, ця мова має власний набір бібліотек, створених спеціально для розробки під Android, використовуючи API Android, пропонуючи більш лаконічний та виразний синтаксис, покращені функції безпеки та підвищену продуктивність розробників. Як результат, Kotlin часто розглядається не лише як доповнення до Java, але й як потенційний довгостроковий наступник.

У Java надмірність часто призводить до довгого та багатослівного коду. Kotlin, з іншого боку, пропонує більш лаконічний та зручний для початківців синтаксис. Ця мова робить акцент на спрощеному функціональному програмуванні та усуває повторюваний шаблонний код. Хоча крапка з комою є необов'язковою в Kotlin, її все ще можна використовувати без проблем. Її додаткові функції допомагають зменшити як складність, так і обсяг коду, необхідного для досягнення цілей, поставлених командою розробників [10].

Враховуючи ці переваги, вибір Kotlin є розумним рішенням, яке підтримує швидку, ефективну та високоякісну розробку мобільних додатків для Android. З допомогою сучасних функцій цієї мови зможемо створити надійний, високопродуктивний та орієнтований на користувача додаток, який задовольнить потреби користувачів.

## 2.2 Аналіз мови розмітки XML

У розробці для Android XML-файли відіграють вирішальну роль у визначенні різних аспектів майбутньої програми.

XML (Extensible Markup Language) – це гнучка та широко використовувана мова розмітки, призначена для зберігання та транспортування даних. Хоча за структурою вона нагадує HTML, XML принципово відрізняється тим, що не має попередньо визначених тегів. Натомість розробники можуть вільно визначати власні теги відповідно до потреб конкретного застосування, що забезпечує високий ступінь гнучкості у представленні даних.

XML походить від SGML (стандартної узагальненої мови розмітки), складнішого стандарту розмітки, і була створена, щоб бути простішою та доступнішою, зберігаючи при цьому здатність описувати структуровані дані. Головна мета XML – полегшити обмін даними між різними системами, особливо через Інтернет, у платформо-незалежному та машино-зчитуваному форматі.

Одна з ключових переваг XML полягає в його здатності зберігати та структурувати дані в чіткому, ієрархічному форматі. Теги в XML не лише визначають самі дані, але й надають контекст, що полегшує організацію, пошук та маніпулювання даними за потреби. Завдяки своїй легкості та масштабованості XML особливо добре підходить для застосувань, де чіткість та портативність даних є важливими.

У розробці для Android XML відіграє вирішальну роль у визначенні компонентів інтерфейсу користувача (UI), таких як макети, віджети та атрибути перегляду. Це дозволяє розробникам відокремити дизайн інтерфейсу користувача від логіки програми, покращуючи зрозумілість коду. Оскільки XML-файли ефективно аналізуються та не обтяжують рендеринг макета системи, вони сприяють швидшій роботі інтерфейсу користувача.

У реалізації XML просто передбачає оголошення необхідних тегів та атрибутів, до яких потім можна отримати доступ та маніпулювати ними програмно. Такий підхід забезпечує узгодженість, легкість обслуговування та спрощений процес розробки.

Отже, користувацький інтерфейс (UI) застосунку Android структурований як ієрархічна структура макетів та інтерактивних компонентів. В основі цієї структури лежать макети, які є екземплярами класу ViewGroup. ViewGroup - діє як контейнер, що визначає розташування, вирівнювання та організацію дочірніх елементів (views) на екрані. У цих макетах розміщені віджети, які є окремими елементами інтерфейсу, похідними від класу View. Також одна ViewGroup може мати іншу ViewGroup як дочірній елемент. До поширених віджетів належать кнопки, текстові поля, зображення та інші інтерактивні компоненти, що дозволяють користувачеві вводити дані та відображати дані застосунку. Разом

макети та віджети утворюють гнучку та модульну структуру, яка дозволяє розробникам створювати адаптивні та візуально привабливі інтерфейси Android.

На рисунку 2.1 наведено приклад ієрархії XML-файлу. ViewGroup (Linear Layout) містить одну ViewGroup (Relative Layout) та дві View (Button and TextView). Ще два View (EditText) вкладені всередині Relative Layout ViewGroup. Важливо зазначити, що один layout може бути вкладений в інший [11].

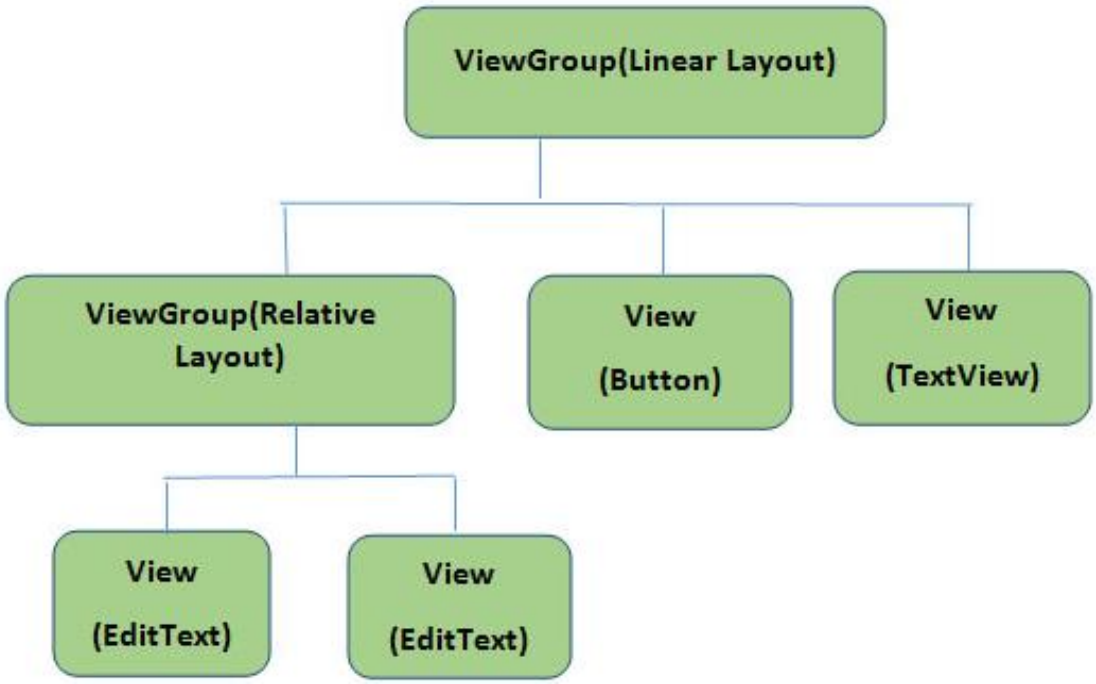



Рисунок 2.1 – Діаграма ієрархії XML-файлу

Взагалі в Android Studio різні XML-файли використовуються для різних, специфічних цілей:

- Layout.xml: XML-файли Layout визначають фактичний інтерфейс користувача програми. Вони містять усі елементи інтерфейсу користувача, такі як кнопки, текстові представлення, елементи редагування та інші, які організовані у ViewGroups;
- AndroidManifest.xml: цей файл містить ключові відомості про програму, такі як назви пакетів програми, що відповідають просторам імен коду, компоненти програми, такі як дії, служби, приймачі широкомовних розсилок та постачальники контенту. Він

також визначає дозволи, які програма повинна запитувати у користувача для доступу до певних функцій;

- Strings.xml: цей файл містить текстові значення для всіх віджетів TextView, що сприяє повторному використанню коду та полегшує локалізацію програми для кількох мов. Рядки, визначені тут, можуть замінити жорстко закодований текст у всій програмі;
- Themes.xml: цей файл визначає базову тему та користувацькі теми для програми. Він також використовується для встановлення стилів та зовнішнього вигляду інтерфейсу користувача (UI) програми;
- Drawable XML файли: ці XML-файли визначають графічні елементи, такі як фони кнопок, а також можуть бути використані для створення різних градієнтів. Вони можуть містити векторну графіку, наприклад іконки або інші зображення;
- Colors.xml: Даний файл використовується для зберігання всіх визначень кольорів, необхідних для програми. Усі кольори мають бути вказані за допомогою шістнадцяткових (hex) кодів;
- Dimens.xml: Як випливає з самої назви файлу, він відповідає за зберігання всіх розмірів представлень. Це може бути висота кнопки, відступи, поля для представлень тощо. Розміри повинні бути у форматі значень щільності пікселів (dp) [12].

Розуміння та ефективне використання цих XML-файлів є основоположним для створення добре структурованих та візуально привабливих Android додатків.

## 2.3 Аналіз середовища розробки Android Studio

Як основне інтегроване середовище розробки (IDE) для розробки мобільного застосунку було обрано Android Studio. Це офіційне IDE для розробки під платформу Android, побудоване на IntelliJ IDEA від JetBrains, тієї ж компанії, яка створила мову програмування Kotlin.

Android Studio надає повний набір інструментів для розробки Android-додатків. Він підтримує весь процес розробки — від написання коду до

тестування та розгортання — пропонуючи всі функції, необхідні розробникам для створення Android-додатків [13].

Для ясності порівняємо Android Studio з іншими інтегрованими середовищами розробки, з допомогою яких також можна створювати Android-застосунки.

До виходу Android Studio розробники Android використовували Eclipse, але тепер Google офіційно рекомендує Android Studio через кращі інструменти, швидшу збірку та тіснішу інтеграцію з платформою Android. Eclipse призначений не лише для Android. Він також є надійним та гнучким середовищем розробки (IDE) з такими функціями, як підтримка Gradle, підключення до програмного забезпечення для контролю версій та велика бібліотека плагінів для розширення можливостей. Але одними з основних мінусів є застарілий інтерфейс користувача та не всі розширені функції є доступними.

Як вже знаємо, Android Studio побудовано на IntelliJ IDEA, потужному середовищі розробки Java від JetBrains. Хоча IntelliJ IDEA підтримує розробку для Android за допомогою плагінів, Android Studio, як спеціалізована версія, пропонує більше функцій, специфічних для Android, та кращу інтеграцію інструментів та бібліотек. Одним з основних мінусів IntelliJ IDEA є те, що це платний сервіс, а безкоштовній версії бракує багатьох функцій.

Visual Studio від Microsoft орієнтована на розробку під Windows, хоча й має підтримку Android через Xamarin. Xamarin — це фреймворк для мобільних додатків з відкритим кодом, який можна завантажити та використовувати безкоштовно. Однак, Android Studio є більш спеціалізованим та оптимізованим середовищем, створеним виключно для розробки Android.

NetBeans - це інтегроване середовище розробки (IDE) для Java. Крім того, для полегшення розробки додатків для Android за допомогою плагінів, він також пропонує комплексну підтримку розробки на Java, включаючи необхідні інструменти, бібліотеки та функції інтеграції. NetBeans має базову підтримку для мови Kotlin, однак вона дуже обмежена та не є офіційною [14].

Отже, Android Studio є багатофункціональною платформу, спеціально розробленою для підвищення ефективності розробників, які працюють над проектами Android. Інтерфейс середовища детально продумано та містить кілька панелей, таких як панель «Проект» для навігації по файлах, панель «Редактор» для написання та редагування коду, а також панель «Logcat» для перегляду системних повідомлень у режимі реального часу та журналів налагодження. Ці компоненти працюють разом, щоб оптимізувати процес розробки.

В основі Android Studio лежить система збірки Gradle, яка автоматизує важливі завдання, такі як компіляція коду, виконання тестів, керування залежностями та пакування APK-файлів. Ця система особливо корисна для керування великими проектами.

Редактор коду підтримує різноманітні мови програмування, включаючи Java, Kotlin та C++. Пропонує розширені функції, такі як автозаповнення коду, підсвічування синтаксису в режимі реального часу, рефакторинг коду, виявлення помилок та готові до використання шаблони для пришвидшення розробки.

Головною родзинкою Android Studio є його візуальний редактор макетів. Він дозволяє розробникам створювати користувацькі інтерфейси за допомогою перетягування елементів, що економить час на написанні xml-файлів. Можна легко переглядати макети на екранах різних розмірів і роздільних здатностей.

Тестування здійснюється безперешкодно завдяки вбудованому емулятору. Він дозволяє імітувати широкий спектр пристроїв Android та версій ОС, гарантуючи, що додаток працюватиме добре в різних середовищах.

Для налагодження Android Studio містить потужний налагоджувач, який дозволяє перевіряти змінні, встановлювати точки зупинки та покроково проходити код для виявлення та вирішення проблем. Android Profiler надає інструменти моніторингу продуктивності, виводить інформацію про використання процесора, розподіл пам'яті та мережеву активність у режимі реального часу.

Співпраця та контроль версій також добре підтримуються, з безперешкодною інтеграцією таких інструментів, як Git. Це дозволяє командам відстежувати зміни, об'єднувати код та легко керувати гілками. Крім того,

Android Studio має високі можливості налаштування завдяки широкому спектру плагінів, що дозволяє налаштовувати функціональність IDE відповідно до конкретних проектних вимог [15].

Загалом, Android Studio є потужним, гнучким та незамінним інструментом для будь-якого Android-розробника, пропонуючи універсальне середовище, яке охоплює кожен етап розробки - від кодування та дизайну до тестування, налагодження та розгортання. Враховуючи ці переваги, вибір Android Studio є логічним та обґрунтованим рішенням для розробки мобільного додатку.

## 2.4 Аналіз методу збереження даних за допомогою Room

Більшість програм виробничої якості містять дані, які програмі потрібно зберігати. Наприклад, програма може зберігати список відтворення пісень, пункти зі списку справ, або історію особистих даних. У таких випадках використання для зберігання цих постійних даних використовується база даних.

Room — це потужна бібліотека для збереження даних, що входить до пакету Android Jetpack, яка спрощує роботу з базами даних у додатках для Android. Вона діє як рівень абстракції над SQLite, легкою реляційною системою баз даних, що використовує SQL (структуровану мову запитів) для керування та запитів даних. Хоча безпосереднє використання SQLite може бути складним та схильним до помилок, Room зменшує цю складність, автоматизуючи значну частину шаблонного коду, пов'язаного зі створенням, налаштуванням та доступом до бази даних.

Завдяки Room розробники отримують переваги більш інтуїтивного та структурованого способу обробки взаємодії з базою даних. Це відбувається за допомогою анотованого коду Java або Kotlin, що у багатьох випадках усуває необхідність писати SQL-запити. Крім того, однією з ключових особливостей Room є його здатність перевіряти SQL-запити під час компіляції, що допомагає виявляти помилки на ранніх етапах процесу розробки. Це підвищує надійність коду, зручність обслуговування та забезпечує кращу оптимізацію продуктивності. Усуваючи розрив між об'єктно-орієнтованим програмуванням

та реляційними базами даних, Room сприяє більш безшовній інтеграції збереження даних у додатки для Android.

Рівень абстракції — це структура, що складається з функцій, методів або класів, які приховують основні деталі реалізації та внутрішні складності системи. Таким чином, розробники можуть взаємодіяти з системою через спрощений та узгоджений інтерфейс. Наприклад, у контексті SQLite рівень абстракції дозволяє користувачам виконувати операції з базою даних, такі як запити, оновлення або керування даними, без необхідності розуміти внутрішніми механізмами, що забезпечують роботу SQLite. Таке розділення обов'язків підвищує модульність та покращує підтримку коду.

На рисунку 2.2 показано, як Room, як джерело даних, вписується в загальну архітектуру. Room є джерелом даних [16].

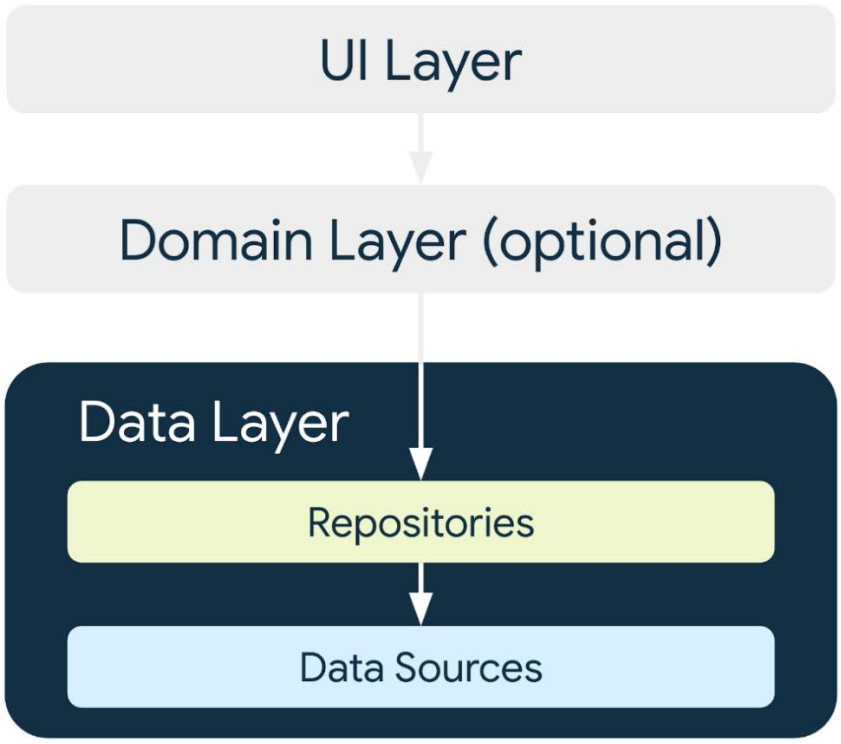



Рисунок 2.2 - Структура шару даних у багаторівневій архітектурі

Розглянемо чому варто обрати саме Room замість безпосереднього використання SQLite та які її переваги.

Перш за все, Room значно спрощує розробку, адже забезпечує абстракцію вищого рівня порівняно з необробленим SQLite, дозволяючи розробникам

взаємодіяти з базою даних за допомогою лаконічного, легкого для читання коду. Це усуває значну частину шаблонного коду та складності, зазвичай пов'язаної з написанням та керуванням SQL-запитами вручну.

Однією з головних переваг Room є його здатність перевіряти SQL-запити під час компіляції. Це допомагає виявляти помилки на ранніх етапах процесу розробки, зменшуючи кількість збоїв під час виконання та роблячи вашу кодову базу більш стійкою та надійною.

Також Room розроблено для безперебійної роботи з компонентами архітектури Android, такими як LiveData та ViewModel. Це забезпечує кращу обробку даних та управління життєвим циклом, особливо у сценаріях, що включають оновлення інтерфейсу користувача та зміни конфігурації.

Room підтримує сучасне асинхронне програмування. Інтегрується з сучасними інструментами паралельного програмування, такими як Kotlin Coroutines та RxJava, забезпечуючи ефективні операції з базою даних без блокування. Це дозволяє виконувати завдання бази даних поза основним потоком, покращуючи швидкість реагування додатків та взаємодію з користувачем.

Архітектура Room побудована навколо трьох основних компонентів, які працюють разом, щоб забезпечити надійний рівень абстракції над SQLite:

- Entity – це клас моделі даних, який визначає структуру таблиці бази даних. Кожна сутність позначається @Entity, а її поля відповідають стовпцям у таблиці. Сутності служать основою для даних, які ви хочете зберігати в базі даних;
- DAO (Об'єкт доступу до даних) – це інтерфейси або абстрактні класи, що містять методи для доступу до бази даних. Ці методи дозволяють виконувати такі операції, як вставка, оновлення, видалення та запит даних. DAO позначаються @Dao та використовують анотації, такі як @Insert, @Delete, @Update та @Query, для відображення операцій SQL;
- Database – це абстрактний клас, який розширює RoomDatabase. Він діє як основна точка доступу до збережених даних вашої програми.

Клас бази даних позначається @Database та пов'язує сутності та відповідні їм DAO. Він також надає однотонний екземпляр бази даних для забезпечення послідовного доступу по всій програмі.

Для того щоб забезпечити перевірку запитів під час компіляції Room перевіряє SQL-запити, аналізуючи запити в DAO. Якщо запит недійсний або не відповідає схемі таблиці, він генерує помилку під час компіляції, запобігаючи збоям під час виконання.

Також Room підтримує корутини для виконання операцій з базою даних (наприклад, вставки, оновлення) у фонових потоках, уникаючи блокування потоків інтерфейсу користувача. Це допомагає запобігти блокуванню потоків інтерфейсу користувача, що призводить до плавніших інтерфейсів користувача [17].

Підсумовуючи, використання Room як бібліотеки для зберігання даних на Android значно спрощує роботу з SQLite, забезпечуючи рівень абстракції, який зменшує обсяг шаблонного коду та підвищує надійність завдяки перевірці SQL-запитів під час компіляції. Саме тому це є найкращим варіантом для зберігання даних в нашому випадку.

## 2.5 Аналіз картографічного сервісу Google maps SDK

SDK Google Maps – це потужний інструмент, який дозволяє розробникам інтегрувати різноманітні функції картографування та геолокації у програми та веб-сайти. За допомогою Maps SDK можна безперешкодно інтегрувати Google Maps у програми Android, зокрема ті, що розроблені для пристроїв Wear OS.

Цей потужний SDK дозволяє відображати динамічні, інтерактивні карти, які реагують на широкий спектр жестів користувача, таких як панорамування, масштабування та обертання. Окрім базової візуалізації карти, можливо збільшити взаємодію з користувачем, додаючи власні маркери, полігони та накладання, щоб виділити певні області або точки інтересу. Ці елементи дозволяють передавати детальну інформацію на основі місцезнаходження [18].

SDK повністю сумісний з мовами програмування Kotlin та Java. Окрім основної функціональності, він включає повний набір додаткових бібліотек та потужних розширень, розроблених для використання розширених функцій, оптимізації робочих процесів розробки та підтримки сучасних методів програмування та найкращих практик [19].

Однією з ключових переваг SDK є його безшовна інтеграція з іншими сервісами Google, такими як Google Places, Directions API та Geocoding API. Це дозволяє створювати потужні функції на основі місцезнаходження в єдиній екосистемі. SDK підтримує різні типи карт (звичайні, гібридні, супутникові, рельєфні), динамічне розміщення маркерів, інструменти малювання для поліліній та полігонів, анімацію камери та налаштування стилів через JSON.

Google Maps API пропонує безліч функцій, які можна використовувати для створення привабливого та цікавих взаємодій з користувачем:

- Інтерактивні карти: можна вбудовувати інтерактивні карти, надаючи користувачам можливість досліджувати різні місця. Користувачі можуть збільшувати та зменшувати масштаб, перемикатися між типами карт і взаємодіяти з маркерами на карті;
- Геокодування: забезпечується функція геокодування, яка перетворює адреси на географічні координати. Ця функція особливо корисна для програм, яким потрібні дані про місцезнаходження;
- Вказівки та маршрутизація: розробники можуть надавати користувачам точні вказівки та інформацію про дорожній рух у режимі реального часу. Ця функція безцінна для програм, що передбачають планування маршруту або навігацію. Користувачі можуть ввести свої початкові та кінцеві адреси, а API розрахує найкращий маршрут, враховуючи дорожні умови та інші фактори;
- Перегляд вулиць: розробники можуть інтегрувати Перегляд вулиць, який забезпечує панорамний вид на 360 градусів. Цю функцію можна використовувати, щоб надати користувачам більш захопливий досвід та покращити їхнє розуміння певного місця.

Щодо продуктивності, SDK оптимізовано для швидкодії та плавної взаємодії з картою. Він також підтримує апаратне прискорення, що робить його придатним для програм з вимогами до інтерфейсу користувача. Оновлення місцезнаходження та обробка взаємодії з користувачем керуються подіями, і розробники можуть динамічно керувати елементами карти на основі введених користувачем даних або даних сервера.

Однак SDK має певні обмеження. Для цього потрібен ключ API від Google Cloud, а використання поза межами безкоштовного рівня передбачає певну плату залежно від обсягу запитів. Крім того, функціональність офлайн-карт не підтримується вбудовано, що може бути недоліком для програм, орієнтованих на регіони з обмеженим підключенням до Інтернету [20].

Незважаючи на ці міркування, Google Maps SDK залишається основним рішенням для багатьох розробників Android завдяки своїй надійності, обширній документації, регулярним оновленням та підтримці спільноти. Він особливо підходить для комерційних програм, які вимагають високої точності, широкого глобального покриття та інтеграції з іншими API Google.

## 2.6 Апаратне забезпечення для функціонування системи

Розроблений Android-додаток призначений для роботи на мобільних пристроях з операційною системою Android версії не нижче 8.0 (рівень API 26). Це зазначено у файлі конфігурації проекту (build.gradle), де встановлено параметр minSdkVersion = 26. Таким чином, додаток може бути встановлений та стабільно функціонувати на більшості сучасних смартфонів та планшетів, випущених після 2017 року.

Для коректної роботи програми рекомендується використовувати мобільний пристрій з такими мінімальними технічними характеристиками:

- Операційна система: Android 8.0 (Oreo) або вище;
- Процесор: ARM Cortex-A53 або еквівалент;
- Оперативна пам’ять: щонайменше 2 ГБ;
- Вільне місце в пам'яті пристрою: щонайменше 50 МБ;

- Дисплей: сенсорний, з роздільною здатністю 720×1280;
- Доступ до Інтернету (для повноцінного використання вбудованої карти).

Завдяки використанню сучасної версії Android SDK, програма має високу сумісність, безпеку та ефективність на більшості сучасних пристроїв.

## 2.7 Висновки

Аналіз існуючих мов програмування показав, що Kotlin найкраще відповідає вимогам сучасної Android-розробки. Переконались, що XML-файли є критично важливою складовою розробки Android-додатків, оскільки забезпечують чітке розмежування між логікою програми та її візуальним представленням. Android Studio обрано як оптимальне середовище для розробки, адже в порівнянні з іншими IDE, воно забезпечує кращу продуктивність, зручність розробки та адаптацію під потреби Android-проєктів. Визначили, що Room є оптимальним рішенням для зберігання даних в Android-застосунку, оскільки забезпечує зручний рівень абстракції над SQLite, перевірку запитів на етапі компіляції та підтримку асинхронної роботи. Серед різних картографічних сервісів обрали Google Maps SDK через його потужний функціонал інтерактивних карт, можливістю додавання кастомних маркерів та полігонів, а також сумісність із Kotlin. Пересвідчились, що розроблений Android-додаток сумісний з пристроями на базі Android 8.0 і вище.

# 3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ГУРТКІВ МІСТА ВІННИЦІ

## 3.1 Розроблення архітектури інформаційно-довідкової системи

Архітектура інформаційних систем (ISA) - це структурована основа для проектування, розробки та управління інформаційними системами. Вона дозволяє зрозуміти, як взаємодіють існуючі системи, виявити будь-які обмеження та визначити необхідні кроки для вирішення цих проблем.

Метою архітектури інформаційної системи є аналіз, визначення та структурування еволюції інформаційних систем відповідно до корпоративної стратегії, бізнес-процесів та технологічних інновацій [21].

Уніфікована мова моделювання (UML) була розроблена як потужний інструмент для візуального представлення складних систем з використанням об'єктно-орієнтованої парадигми. Її основна мета — оптимізувати та вдосконалити процес розробки програмного забезпечення. Це сприяє кращій комунікації між командами розробників та виявляє недоліки проектування на ранньому етапі. Як результат, вона не тільки прискорює терміни розробки, але й сприяє суттєвому покращенню загальної якості, зручності обслуговування та масштабованості кінцевого програмного продукту. Вона дозволяє будувати моделі для усіх фаз життєвого циклу ПЗ.

Моделювання за допомогою UML ґрунтується на трьох фундаментальних принципах, які керують створенням ефективних та змістовних системних представлень.

Перш за все в основі моделювання UML лежить принцип абстрагування. Це означає, що модель повинна включати лише ті елементи системи, які є важливими для розуміння та підтримки її основних функцій. Нерелевантні деталі навмисно опускаються, щоб зосередитися на компонентах, які безпосередньо впливають на поведінку та цілі системи. Таке вибіркове представлення спрощує складні системи та робить їх легшими для розуміння та управління.

Наступним принципом є багатомодельність, адже одна модель не може адекватно охопити всі аспекти складної системи. Тому UML заохочує використання кількох, взаємодоповнюючих моделей або видів. Кожна модель представляє певний аспект системи, такий як її структура, поведінка або взаємодія, і разом вони забезпечують більш повне та точне розуміння. Ці види взаємопов'язані та узгоджуються одне з одним, пропонуючи розуміння різних системних вимірів, таких як статична архітектура, динамічні процеси та сценарії розгортання.

UML підтримує моделювання на різних рівнях абстракції, що називається ієрархічністю. На найвищому рівні знаходиться концептуальний вигляд, який описує систему в загальних рисах, зазвичай зосереджуючись на бізнес-логіці або функціях високого рівня. У міру розвитку моделювання кожен наступний рівень заглиблюється в більш детальні технічні деталі. Ці проміжні представлення стосуються логічних структур, взаємодій та робочих процесів, що досягає кульмінації на фізичному рівні, який описує фактичну реалізацію системи. Це включає конкретні компоненти, механізми зберігання даних, конфігурації апаратного забезпечення та інтеграцію з програмними платформами.

На рисунку 3.1 зображено елементи, які ще називають класифікаторами, мови UML поділені на три групи:


Рисунок 3.1 - Класифікатори мови UML

Сутності в UML – це основні об’єктно-орієнтовані будівельні блоки, що використовуються для створення моделей. Вони поділяються на чотири типи:

- Структурні – статичні «іменники» моделі, такі як Клас, Інтерфейс, Співпраця, Варіант використання, Активний клас, Компонент та Вузол;
- Поведінкові – динамічні «дієслова», що показують, як модель поводиться з часом, включаючи Взаємодії та Стан;
- Групувальні – використовуються для впорядкування моделей, переважно представлені Пакетами;
- Примітки – надають пояснювальні примітки або коментарі, де Нотатки є основним типом.

Моделі UML визначають чотири ключові типи відносин між сутностями:

- Залежність: семантичне відношення, де зміни в одній (незалежній) сутності можуть вплинути на іншу (залежну);
- Асоціація: структурне відношення, що показує логічні зв'язки між об'єктами;
- Узагальнення: відношення «батько-дитина», де дочірній об'єкт успадковує та може замінити батьківський;
- Реалізація: семантичне відношення, де один класифікатор визначає контракт, а інший його виконує, що спостерігається у зв'язках «клас-інтерфейс» або «випадок використання-кооперація».

UML містить 13 діаграм, згрупованих у два типи:

- Структурні діаграми, які показують статичну структуру елементів системи (класи, інтерфейси, атрибути, зв'язки);
- Діаграми поведінки, які ілюструють поведінку системи, взаємодії та зміни стану [22].

Варіант використання (англ. Use Case) — це поширене поняття в розробці програмного забезпечення та проектуванні систем, яке ілюструє, як система реагує на зовнішні вхідні дані. Він уточнює як користувач чи інша система можуть взаємодіяти з системою та які дії можуть виконувати. Варіанти

використання відіграють ключову роль у визначенні поведінкових або функціональних вимог системи.

Компоненти сценарію використання включають наступні елементи:

- Актори: зовнішні сутності, які взаємодіють із системою. Це можуть бути окремі особи (наприклад, користувачі) або інші системи та програми;
- Сценарії: Вони стосуються послідовності кроків або подій, що відбуваються під час взаємодії між актором та системою. Сценарії можна класифікувати як первинні (базові), альтернативні або виняткові;
- Система: позначає програмну або апаратну інфраструктуру, яка реагує на дії, ініційовані акторами, та обробляє їх;
- Цілі: являють собою очікувані результати або завдання, яких актори прагнуть досягти через свою взаємодію із системою.

Отже, метою use case є упорядкування функціональних вимог, моделювання цілі взаємодії системи/акторів, запис сценаріїв від тригерних подій до цілей, опис одного основного потоку подій та різних альтернативних потоків, щоб один варіант використання використовував функціональність іншого [23].

Під час опису use case, його компоненти можна пов'язати за допомогою таких типів зв'язків:

- Асоціація (Association) – показує, що актор взаємодіє з конкретним варіантом використання;
- Узагальнення (Generalization) – вказує на те, що один варіант використання розширює або спеціалізує інший, поділяючи спільні риси;
- Включення (Include) – означає, що один варіант використання завжди включає функціональність з іншого, зазвичай повторно використовувану поведінку;
- Розширення (Extend) – додає до варіанту використання додаткову поведінку, яка спрацьовує за певних умов.

На рисунку 3.2 зображена use case діаграма, на якій показано функціональність інформаційно-довідкової системи гуртків Вінниці. Користувач може знаходити гуртки за назвою або за позначкою на карті. Також переглядати сторінку гуртків з можливістю фільтрації за категорією, ціною, днями тижня та часом. Користувач також може додавати гуртки до обраного, переглядати обране, планувати відвідування гуртків на сторінці плану, переглядати календар, вибирати дату та гурток для планування, додавати або видаляти гуртки з плану. Адміністратор має можливість керувати гуртками, додаючи, видаляючи та редагуючи їх.

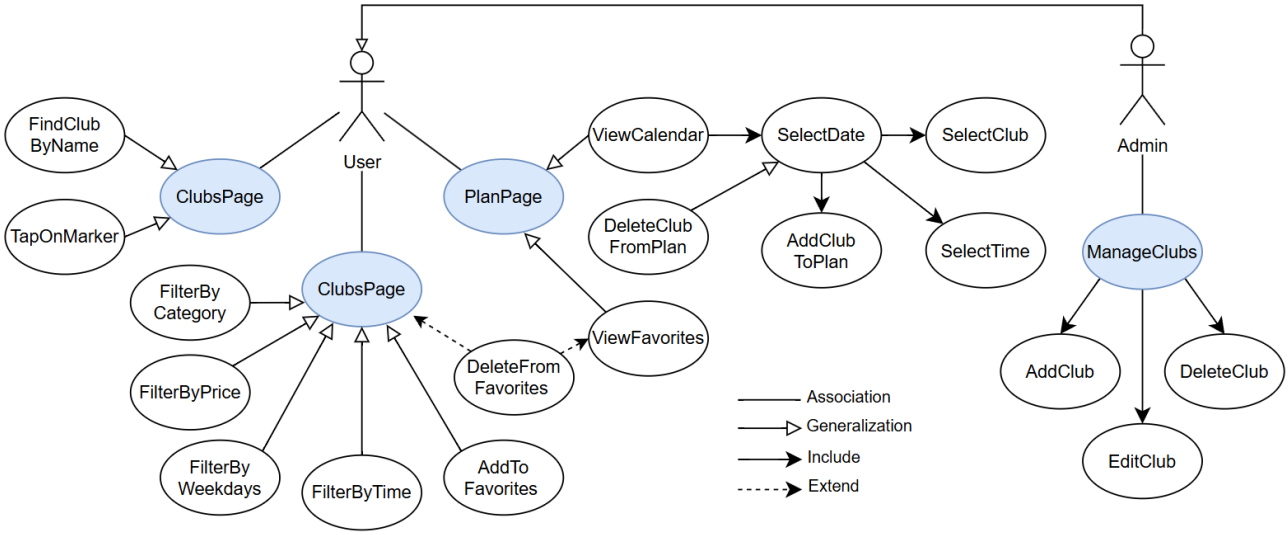



Рисунок 3.2 – Діаграма use case

Діаграма послідовності - це діаграма уніфікованої мови моделювання (UML), яка ілюструє послідовність повідомлень між об'єктами під час взаємодії. Діаграма послідовності складається з групи об'єктів , представлених лініями життя, та повідомлень, якими вони обмінюються з часом під час взаємодії.

Діаграма послідовності має містити наступні складові:

– Лінія життя – показує, як довго об’єкт існує в системі;
– Смуга активації – тонкий прямокутник на лінії життя, довжина якого вказує на тривалість перебування об’єктів в активному режимі;
– Повідомлення – стрілки між лініями життя, що показують порядок та напрямок зв’язку [24].

На діаграмі (рис. 3.3) показано процес планування заняття користувачем. Спочатку користувач вибирає гурток для планування через інтерфейс, ідентифікатор цього клубу передається контролеру планування. Контролер звертається до репозиторію гуртків, щоб отримати інформацію про вибраний гурток, яка потім повертається контролеру. Далі користувач вибирає дату та час через інтерфейс, і ця інформація передається контролеру. Контролер додає подію до календаря та зберігає зміни. Репозиторій підтверджує збереження, і контролер відображає підтвердження успішного планування користувачеві.

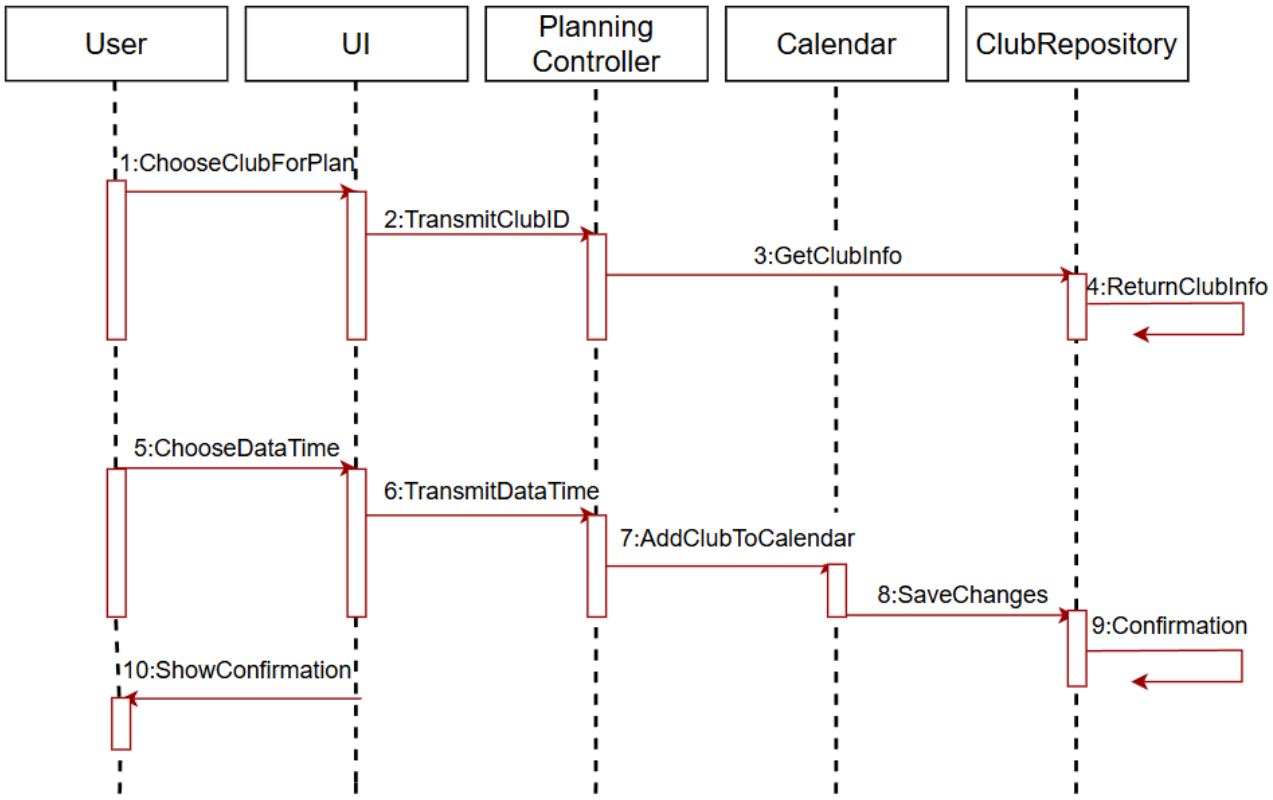



Рисунок 3.3 – Діаграма послідовності для планування занять

Діаграми діяльності показують кроки, що входять до складу системи, допомагаючи нам зрозуміти потік керування. Вони відображають порядок, у якому відбуваються дії, та чи відбуваються вони одна за одною (послідовно) чи одночасно (паралельно). Ці діаграми допомагають пояснити, що запускає певні дії або події в системі.

Усі діаграми діяльності містять:

– Початковий стан – аналогічний до діаграми станів та переходів;

– Кінцевий стан – аналогічний до діаграми станів та переходів;
– Стан прийняття рішення – стан, в якому здійснюється прийняття рішення про перенаправлення потоку управління до одного зі станів, пов’язаних із даним станом;
– Стан синхронізації – стан, в якому здійснюється розділення загального потоку управління на декілька гілок чи навпаки.

На діаграмі діяльності на рисунку 3.4 описано два можливі способи пошуку гуртків на карті, які починаються з переходу на сторінку Map. Перший спосіб передбачає введення назви гуртка в поле пошуку та натискання кнопки «Знайти на карті». Другий спосіб – самостійний пошук безпосередньо на карті. Обидва способи завершуються виділенням маркера знайденого гуртка на карті.

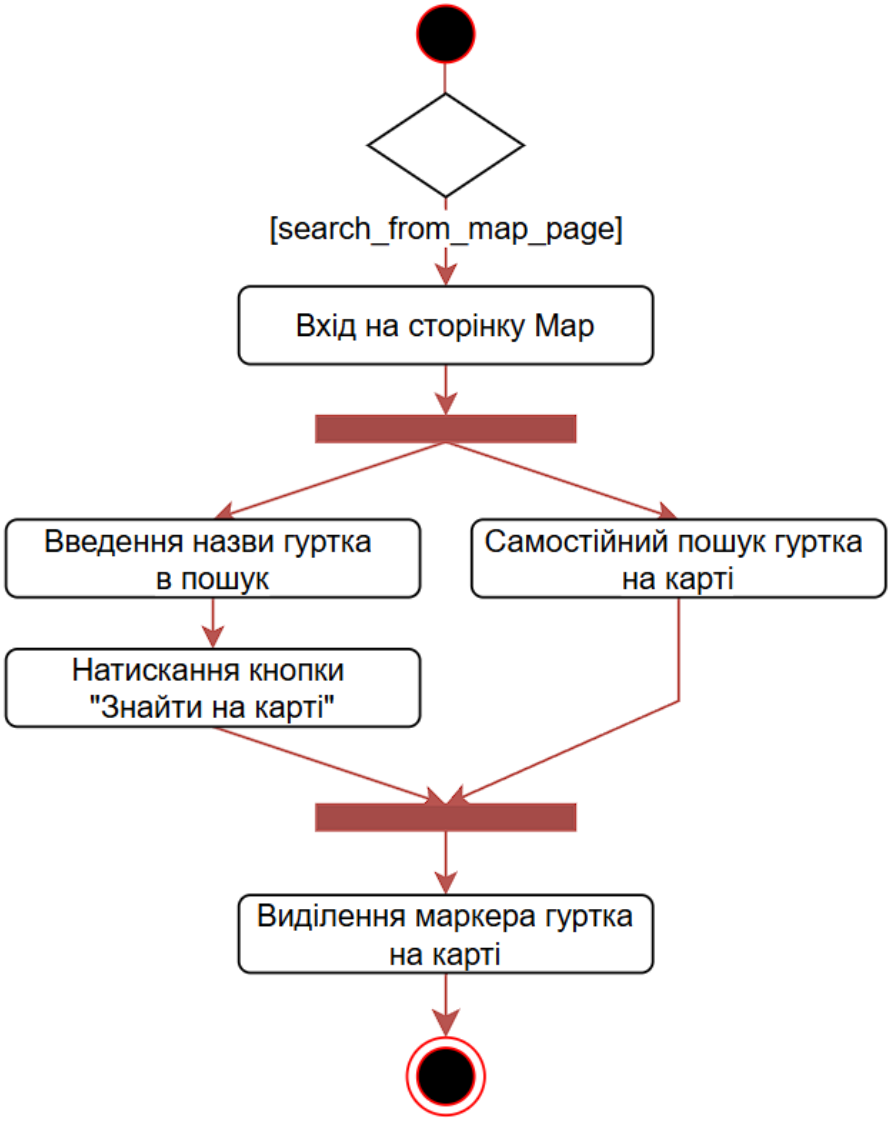



Рисунок 3.4 – Діаграма діяльності для пошуку гуртків на карті

У програмній інженерії діаграма класів у мові UML (Unified Modeling Language) є типом статичної структурної діаграми, яка описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та зв’язки між об’єктами [25].

На рисунку 3.5 показано структуру класів для роботи з даними гуртків в мобільному додатку. Вона поділена на пакети: repository, source (з підпакетами local та remote) та model. Репозиторій WorkshopRepositoryImpl виступає посередником, отримуючи дані з локальних (WorkshopDatabase, WorkshopDao) та віддалених (WorkshopRemoteDataSource, WorkshopMockDataSourceImpl) джерел, а також використовує WorkshopTimeConverter для перетворення часу. Інтерфейс WorkshopRepository визначає основні операції з гуртками. Локальний репозиторій використовує бібліотеку Room Persistence Library через класи WorkshopDatabase та WorkshopDao для взаємодії з базою даних, де дані представлені класом WorkshopEntity. Віддалене джерело даних представлено інтерфейсом WorkshopRemoteDataSource та його фіктивною реалізацією WorkshopMockDataSourceImpl, який повертає WorkshopModel. Клас WorkshopModel представляє модель даних гуртків, яку потім можна перетворити на WorkshopEntity для локального зберігання. Існує залежність WorkshopRepositoryImpl від WorkshopRepository та обох джерел даних, а також від WorkshopDao.

Таким чином відобразили як застосунок використовує багаторівневу архітектуру для роботи з даними гуртків, розділяючи відповідальності між репозиторієм, локальним та віддаленим джерелами даних, а також моделлю даних.

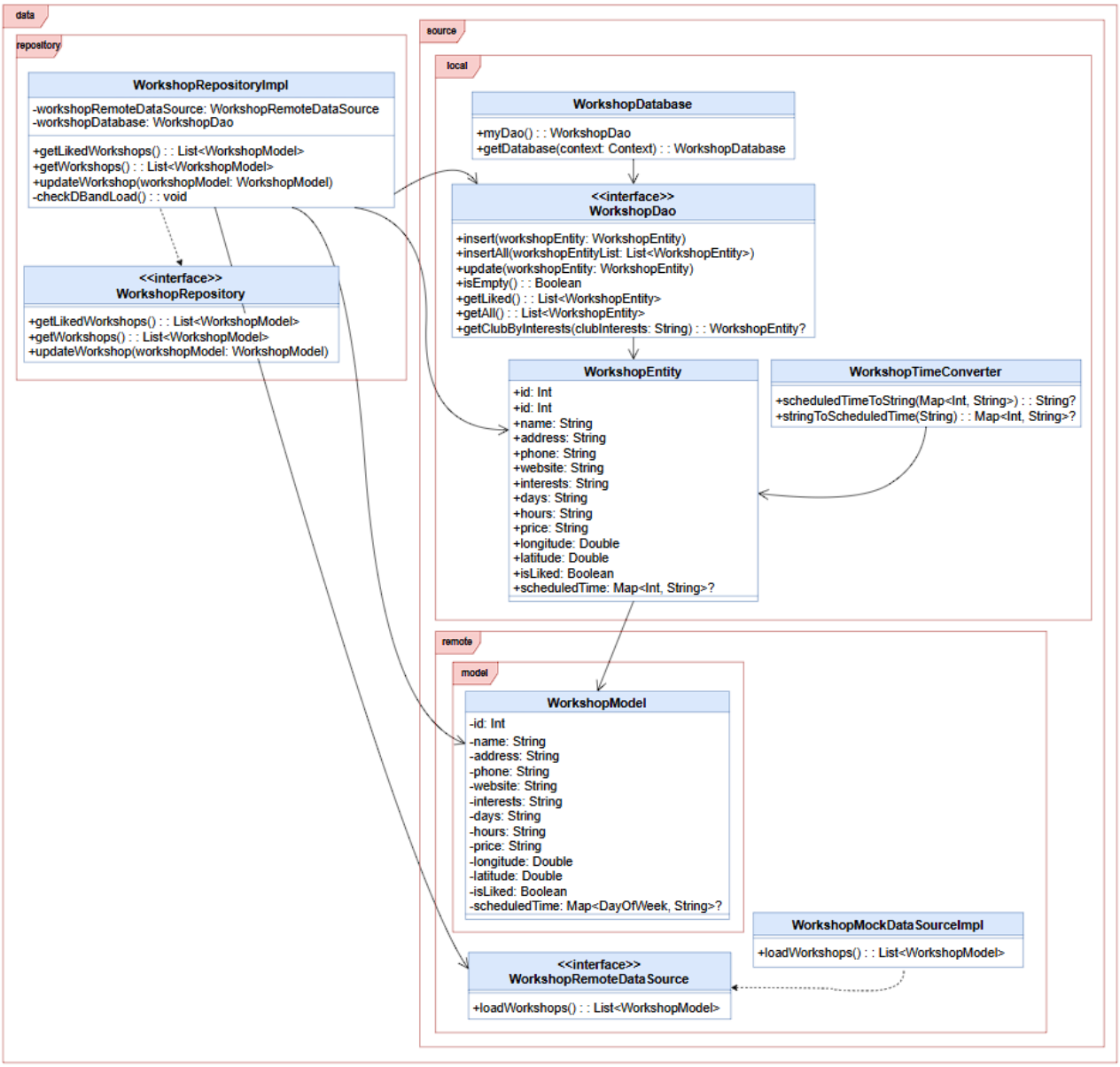



Рисунок 3.5 – Діаграма класів

Також відобразили алгоритм роботи програми у вигляді блок-схеми на рисунку 3.6.

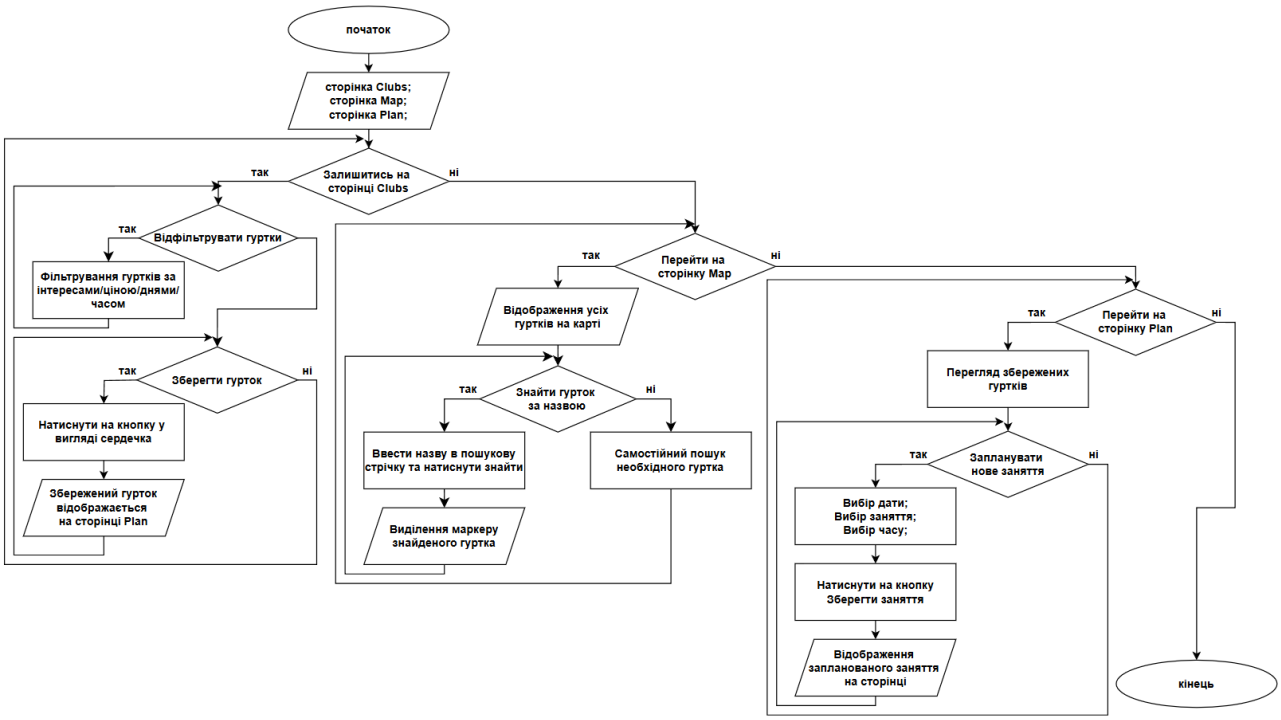



Рисунок 3.6 - Блок-схема алгоритму роботи програми

Відображена на діаграмі (рис. 3.7) архітектура ілюструє, як в майбутньому при завантаженні додатку користувачем Android-пристрій буде взаємодіяти з хмарною інфраструктурою.

Мобільний додаток "VinClubs" на Android-пристрої, використовуючи апаратне забезпечення та системні ресурси (процесор, пам'ять, сенсори, GPS, інтернет), надсилатиме HTTPS-запити до REST API сервера у хмарній інфраструктурі. У відповідь на ці запити, REST API виконуватиме SQL-запити до хмарної бази даних (Cloud DB), наприклад, для пошуку гуртків за інтересами. Отримані дані від бази даних будуть сформовані сервером у вигляді JSON-відповіді та повернені до додатку на Android-пристрої, що забезпечить актуальну інформацію для користувача.

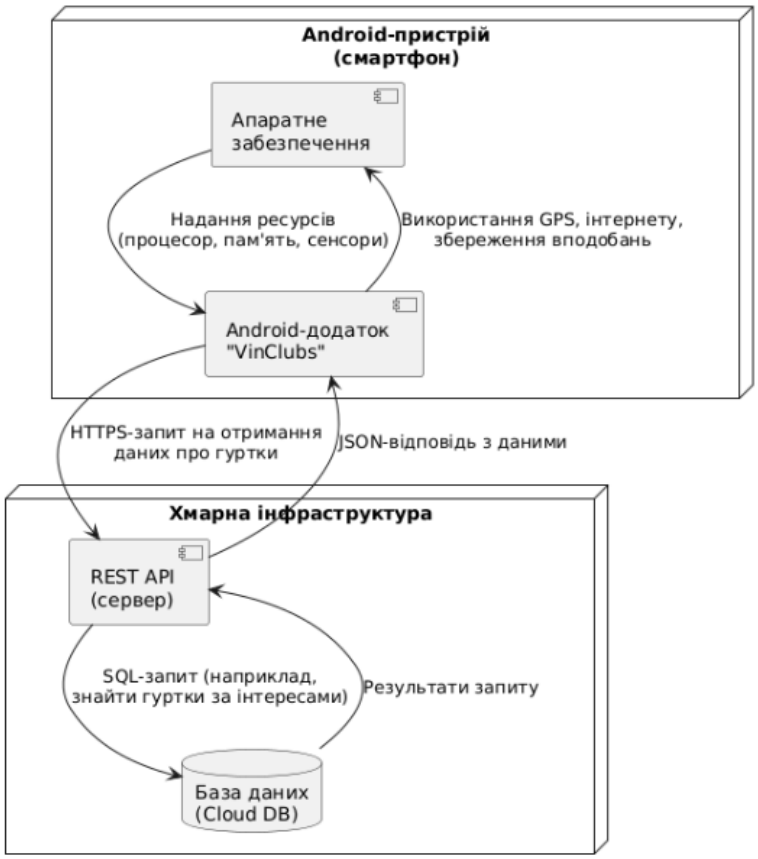



Рисунок 3.7 - Взаємодія між пристроєм та хмарою

## 3.2 Програмна реалізація інформаційно-довідкової системи гуртків

Розроблення мобільного додатку, призначеного для допомоги користувачам у пошуку гуртків, відбувається поетапно, що включає проектування інтерфейсу користувача, реалізацію основного функціоналу, інтеграцію з джерелами даних та проведення ретельного тестування.

Враховуючи обрані технології — Android Studio та Kotlin — проєкт структуровано навколо архітектурного шаблону MVVM (Model-View-ViewModel) для забезпечення чіткого розподілу завдань, підтримки ефективного тестування та управління життєвими циклами компонентів інтерфейсу користувача. Модель обробляє дані та бізнес-логіку, Представлення відповідає за представлення інформації через дії, фрагменти та файли макета, тоді як ViewModel діє як посередник, обробляючи введені користувачем дані та готуючи дані для відображення.

Додаток широко використовує бібліотеки Android Jetpack, такі як LiveData, ViewModel, Room, Navigation Component та Data Binding, для оптимізації розробки та забезпечення дотримання найкращих практик Android. Крім того, кодова база організована в модулі для підвищення зрозумілості, спрощення обслуговування та сприяння повторному використанню окремих компонентів.

### 3.2.1 Розроблення фрагментів

Перш за все необхідно створити окремі екрани інтерфейсу користувача, кожен із яких виконуватиме специфічну функцію в межах мобільного застосунку.

ClubsFragment реалізує головний екран для пошуку та перегляду доступних гуртків. За допомогою PageClubsViewModel витягується та фільтрується список гуртків. Обробляється кнопка «Назад» (onBackPressedCallback), яка закриває фільтри або завершує роботу програми. Метод subscribeToWorkshops реалізує спостереження за списком гуртків у LiveData та оновлює адаптер для їх відображення. Натискання кнопки фільтра btnFilter відкриває DrawerLayout — бічну панель з фільтрами, яка ініціалізується в initDrawer. Подальше спостереження filtersLiveData передбачено в методі subscribeToFilters, де динамічно створюється інтерфейс для фільтрації категорій за інтересами, цінами, днями тижня та годинами. Кожна категорія має заголовок, список фільтрів з кнопками додавання/видалення. Спостереження за застосованими фільтрами (filteredByLiveData) у subscribeToFilteredBy дозволяє відображати активні фільтри як теги, які можна видалити. Функція Flow?.removeAllViews() використовується для очищення контейнера фільтрів.

MapFragment відповідає за відображення гуртків на карті. Для отримання координат використовується MapViewModel. Для того щоб інтегрувати Google maps до нашого проекту необхідно спершу підключити залезності implementation 'com.google.android.gms:play-services-maps:18.1.0' в build.gradle. Потім отримати API-ключ для Google Maps на сайті консоль розробника Google. Згенерований ключ додати у файл AndroidManifest.xml як показано на рисунку 3.8.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyADnqR-3pAlbUQ1G_8VTOMWX_BoC5FS_Pc" />
```

Рисунок 3.8 – Додавання API-ключа

Далі у fragment_map створюємо поле на якому буде відображатись карта (рис. 3.9)

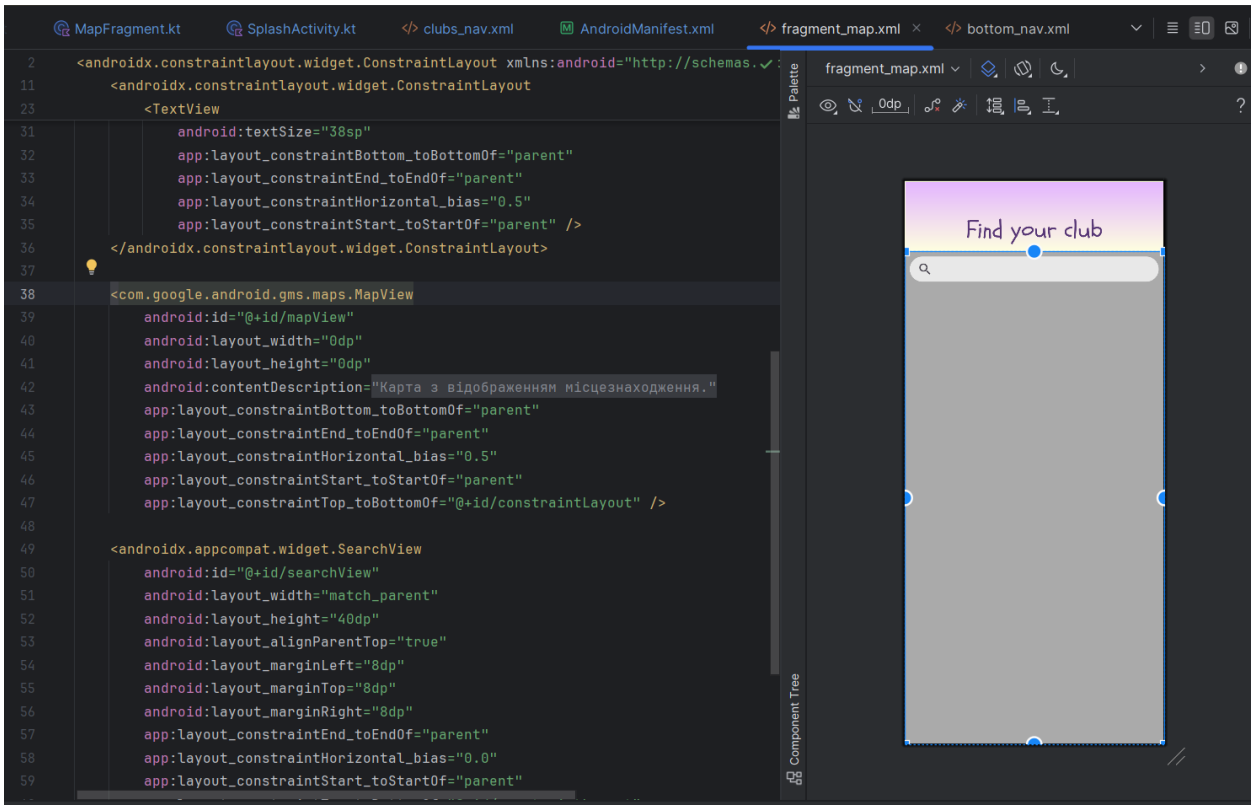



Рисунок 3.9 – Створення макету з картою

Для того щоб карта працювала в емуляторі необхідно додати дозвіл на доступ до інтернету AndroidManifest.xml (рис. 3.10).

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.CALL_PHONE" />
```

Рисунок 3.10 – Дозвіл на доступ до інтернету

Як показано на рисунку 3.8, основними елементами інтерфейсу є MapView та SearchView. Метод onMapReady активується після готовності карти та встановлює початкову позицію в місті Вінниця. Функція subscribeToWorkshopsAndAddMarkers підписується на LiveData з гуртками та додає маркери на карту — з урахуванням координат, назви, адреси та окремих значків, які формуються методом getBitmapDescriptor. setupSearchView забезпечує пошук гуртків за назвою, а метод searchMarkers відповідає за переміщення камери до знайденого маркера або відображення повідомлення, якщо результату немає.

PlanFragment реалізує функціональність перегляду збережених гуртків та редагування розкладу. За допомогою PlanViewModel управляється список уподобаних гуртків і їхній розклад. Інтерфейс містить CalendarView, який дозволяє обрати день. При цьому відкривається нижня шторка (BottomSheetBehavior), що показує відповідний розклад. Кнопка btnAdd запускає фрагмент AddScheduleFragment для додавання нових елементів до розкладу. Метод initBackPress відповідає за коректну обробку натискання кнопки “назад”: якщо є фрагменти у стеку — відбувається повернення, якщо відкрита нижня шторка — вона ховається, інакше — активність закривається. Через subscribeToLikedWorkshops реалізовано оновлення списку вподобаних гуртків у адаптері LikedAdapter. Метод initBottomSheetBehavior налаштовує початкову висоту та поведінку шторки. initSchedule виконує фільтрацію уподобаних гуртків за вибраним днем і передає їх у ScheduleAdapter для виведення на екран.

AddScheduleFragment реалізує функціональність додавання гуртка до розкладу на вибраний день. Для взаємодії з даними використовується ViewModel — PlanViewModel, який містить список вподобаних гуртків. У методах onCreateView та onViewCreated відбувається ініціалізація binding для зручного доступу до елементів макету, а також отримується значення dayOfWeek з аргументів. За натисканням кнопки ivClose фрагмент закривається, а натискання на btnChoose відкриває Spinner для вибору гуртка. ViewModel спостерігає за змінами у likedWorkshopsLiveData, після чого викликається функція initSpinner, яка заповнює список назв доступних для планування гуртків. Після вибору

гуртка відбувається ініціалізація вибору часу за допомогою initHoursSpinner, який обробляє рядок годин (наприклад, “9:00–18:00”) через метод parseHoursRange та дозволяє користувачеві обрати конкретний час. Після завершення вибору час і гурток зберігаються через vm.updateWorkshop, а фрагмент закривається.

### 3.2.2 Розроблення адаптерів

Основне призначення адаптерів полягає у відображенні списків об'єктів типу WorkshopModel, а також у забезпеченні взаємодії користувача з елементами списку (наприклад, збереження, видалення, розподобання тощо).

Перший адаптер – ClubsAdapter – використовується для відображення загального списку гуртків на екрані пошуку. Його конструктор приймає список об'єктів WorkshopModel та функцію-колбек onLikedClickListener, яка викликається у відповідь на натискання кнопки "зберегти/вподобати". Внутрішній клас ClubViewHolder ініціалізує TextView для назви, адреси, телефону, вебсайту та ImageView для кнопки "зберегти". Метод bind(workshop: WorkshopModel) заповнює ці поля відповідними даними з моделі. За допомогою Linkify.addLinks(siteTV, Linkify.ALL) забезпечується клікабельність вебсайту. Подія натискання на кнопку "зберегти" обробляється через save.setOnClickListener, у якому викликається onLikedClickListener з оновленим станом isLiked. Метод updateLikeImage(isLiked: Boolean) візуально оновлює зображення кнопки залежно від того, чи гурток є вподобаним, чи ні.

Наступний адаптер – LikedAdapter – подібний до ClubsAdapter, однак він використовується для відображення лише тих гуртків, які були вподобані користувачем. Конструктор LikedAdapter приймає змінюваний список вподобаних об'єктів WorkshopModel і колбек onUnlikedClickListener, який викликається при "розподобанні" гуртка. Внутрішній клас ViewHolder ініціалізує елементи інтерфейсу, аналогічні попередньому адаптеру. Метод bind(workshop: WorkshopModel) перевіряє, чи гурток є вподобаним, і відповідно

оновлює кнопку "зберегти". У разі натискання викликається функція onUnlikedClickListener, яка дозволяє прибрати гурток із переліку вподобаних.

Останній адаптер – ScheduleAdapter – відповідає за візуалізацію розкладу гуртків на певний день тижня. Конструктор приймає список гуртків, день тижня (dayOfWeek) та колбек deleteClickListener для обробки видалення гуртка з розкладу. Внутрішній ViewHolder містить TextView для часу і назви гуртка та ImageView для кнопки видалення. Метод bind(workshop: WorkshopModel, dayOfWeek: DayOfWeek) наповнює елемент актуальними даними про час (scheduledTime) і назву гуртка. При натисканні на кнопку "видалити" викликається deleteClickListener, що дозволяє оновити модель WorkshopModel, видаливши інформацію про запланований час для обраного дня, а також оновлює сам список в адаптері, вилучаючи відповідний елемент.

### 3.2.3 Розроблення ViewModel

У межах реалізації Android-додатку було створено кілька ViewModel-класів, кожен із яких відповідає за керування даними та бізнес-логікою для окремих екранних компонентів, забезпечуючи розділення відповідальностей відповідно до архітектурного підходу MVVM.

ViewModel для екрану з картою (MapViewModel) відповідає за завантаження та зберігання списку гуртків, які відображаються на карті у вигляді маркерів. Він містить приватне поле _workshopsLiveData типу MutableLiveData<List<WorkshopModel>>, що дозволяє змінювати дані всередині ViewModel, та публічне поле workshopsLiveData типу LiveData, яке спостерігається фрагментами, зокрема MapFragment, для отримання актуального списку. Після ініціалізації MapViewModel автоматично запускає корутину у viewModelScope, яка звертається до workshopRepository і встановлює результат у LiveData. Завдяки цьому, користувацький інтерфейс автоматично оновлюється при надходженні нових даних.

PageClubsViewModel забезпечує логіку для списку гуртків на окремому екрані, включаючи підтримку фільтрації. Він зберігає повний список гуртків у змінній allWorkshops, а також працює з кількома LiveData-змінними:

- _workshopsLiveData — для поточного відфільтрованого списку;
- _filtersLiveData — для доступних фільтрів;
- _filteredByLiveData — для зберігання вибраних фільтрів користувачем.

У методі init завантажуються всі дані з репозиторію, формуються початкові фільтри, і встановлюються у відповідні LiveData. Функції addFilter та removeFilter керують логікою вибору фільтрів, а performFiltering — виконує фільтрацію у фоновому потоці, використовуючи корутину з Dispatcher.IO. Важливо, що попередні задачі фільтрації можуть скасовуватись через змінну filterJob, що дозволяє уникати надмірного навантаження при швидкій зміні фільтрів. Фільтрація здійснюється на основі збереженого повного списку, що гарантує цілісність і узгодженість результатів. Відфільтровані результати передаються через postValue у _workshopsLiveData, після чого UI оновлюється відповідно.

PlanViewModel реалізує логіку для екрану з розкладом, де користувач взаємодіє з улюбленими гуртками. Основним елементом є _likedWorkshopsLiveData, яке містить список вподобаних користувачем гуртків, і яке підписане PlanFragment та AddScheduleFragment. Після створення ViewModel викликається метод reload(), що запускає корутину для отримання актуального списку з репозиторію. Для уникнення конфліктів паралельних запитів використовується змінна loadJob, що дозволяє скасовувати попередні завдання. Крім того, метод updateWorkshop дає змогу змінити стан гуртка у репозиторії (наприклад, додати або видалити його з розкладу). Після таких змін reload() викликається повторно, щоб UI відображав найактуальнішу інформацію.

### 3.2.4 Розроблення кастомного календаря

Також реалізовано кастомний компонент календаря, який забезпечує інтуїтивний інтерфейс для вибору місяця, року та конкретного дня. Основу компонента складають дві ключові частини: клас адаптера CalendarAdapter і кастомний View — MyCalendarView.

CalendarAdapter відповідає за відображення днів поточного місяця у вигляді сітки за допомогою RecyclerView. У його конструкторі передаються список днів місяця (як рядки), індекс вибраного елемента, а також колбек-функція, яка викликається при виборі дати користувачем. Якщо день не порожній, адаптер оновлює вибрану позицію, перерисовує відповідні елементи та повідомляє зовнішній компонент про вибір. Крім того, адаптер забезпечує візуальне виділення вибраного дня завдяки зміні фону та працює з макетом item_day.xml, що містить TextView для відображення чисел.

MyCalendarView — це повноцінний компонент, що інкапсулює весь функціонал календаря. Його інтерфейс включає два спінери для вибору місяця та року і RecyclerView для відображення днів. Під час ініціалізації компонент "роздуває" власний макет view_custom_calendar.xml та налаштовує необхідні елементи. За вибір місяця та року відповідає метод setupSpinners(), який генерує відповідні списки й обробляє події зміни значень. Метод setupCalendar() формує сітку днів відповідного місяця з урахуванням дня тижня, на який припадає перше число, додає порожні клітинки для правильного вирівнювання сітки та передає ці дані в адаптер.

При виборі дати користувачем формується об’єкт LocalDate, який потім передається через публічний метод setOnDateSelectedListener зовнішньому обробнику. Це дозволяє використовувати компонент у різних частинах додатку, де необхідний контроль над вибраною датою. Завдяки своїй архітектурі MyCalendarView є зручним у використанні та легко інтегрується в інші екрани застосунку.

### 3.2.5 Розроблення навігації між сторінками

MainActivity виступає як центральна точка в мобільному додатку на платформі Android. Вона виконує ключову роль у початкових налаштуваннях інтерфейсу користувача та забезпечує керування навігацією між основними екранними додатками за допомогою нижньої панелі навігації (BottomNavigationView). Клас MainActivity успадковується від базового класу AppCompatActivity, що забезпечує сумісність з іншими версіями Android і надає базову функціональність для активності.

Однією з важливих частин реалізації є обробка елементів вибору нижньої навігаційної панелі. Для цього використовується метод setOnItemSelectedListener(), який через оператор обробляє вибір користувача. У зв’язку з вибраним пунктом меню потрібно створити метод replaceFragment() із відповідним фрагментом: ClubsFragment для перегляду списку гуртків, MapFragment для перегляду карти або PlanFragment для відображення розкладу. Після цієї події обробки завершується повернення істинного значення. Для забезпечення стартового вигляду програма встановлює початковий вибраний елемент у BottomNavigationView — пункт з ID R.id.clubs.

Для забезпечення коректного відображення контенту з урахуванням системних відступів використовується метод ViewCompat.setOnApplyWindowInsetsListener(). Він також до основного контейнера макету (ідентифікатор R.id.main) і дозволяє зменшити відступи (padding) відповідно до системних панелей (наприклад, статусного рядка або навігаційної панелі), запобігаючи їх перекриттю вмісту додатком.

Метод replaceFragment(fragment: Fragment), реалізований у цій же активності, відповідає для динамічної заміни фрагментів у межах контейнера з ID R.id.frameLayout. Він використовує supportFragmentManager для запуску транзакцій, заміни активного фрагмента та коміту змін. Такий підхід дозволяє організувати зручну і гнучку навігацію між екранами без потреби створювати окремі активності для кожного з них.

Таким чином, MainActivity виконує роль додатка навігаційної хаби, об’єднуючи в свою логіку ініціалізацію інтерфейсу, обробляючи взаємодію користувача з нижньою панеллю навігації та керуючи відображенням відповідних фрагментів.

### 3.2.6 Роль класу MyApplication

Клас MyApplication відіграє ключову роль у структурі Android-додатку, оскільки він є точкою входу, яка ініціалізується першою під час запуску застосунку. Він успадковує базовий клас Application, що дозволяє вносити налаштування, що будуть дійсними протягом усього життєвого циклу застосунку.

Його головним завданням є ініціалізація системи впровадження залежностей за допомогою бібліотеки Koin. У перевизначеному методі onCreate() викликається функція initKoin(), яка запускає механізм налаштування залежностей. В рамках цієї ініціалізації вмикається ведення журналу через androidLogger(), контекст застосунку надається через androidContext(this@MyApplication), а також завантажуються всі модулі, визначені у функції getKoinModule().

Модуль Koin описує, як створюються та надаються об'єкти. Наприклад, конструкція single оголошує один екземпляр DAO для доступу до локальної бази даних — WorkshopDatabase.getDatabase(this@MyApplication).myDao(). Також визначено фабрику для WorkshopRemoteDataSource, яка створює новий об'єкт WorkshopMockDataSourceImpl для кожного запиту, що імітує роботу з віддаленим джерелом даних. Крім того, конструкція factory використовується для створення WorkshopRepository, який отримує залежності від DAO та RemoteDataSource, поєднуючи доступ до локальних та віддалених даних відповідно до шаблону "репозиторій".

Для забезпечення архітектури MVVM, модуль також описує, як створювати ViewModels: PageClubsViewModel, PlanViewModel та MapViewModel, які автоматично отримують екземпляр WorkshopRepository

через механізм Koin. Таким чином, клас MyApplication забезпечує централізовану, гнучку та масштабовану ініціалізацію всієї системи залежностей у застосунку.

### 3.2.7 Розроблення локальної бази даних

У процесі розробки бази даних для мобільного застосунку було реалізовано архітектурне рішення, що включає багаторівневий доступ до даних через репозиторій. Репозиторій виступає єдиною точкою взаємодії з даними та абстрагує бізнес-логіку застосунку від того, звідки надходить інформація – з віддаленого джерела або з локальної бази даних.

Базовий інтерфейс WorkshopRepository описує набір функцій для доступу до даних про гуртки: отримання всіх гуртків, які сподобалися користувачеві, та оновлення конкретного гуртка. Ці методи є корутинами, які дозволяють працювати з ними асинхронно, не блокуючи потік інтерфейсу користувача.

Реалізація цього інтерфейсу відбувається в класі WorkshopRepositoryImpl, який поєднує роботу з локальною базою даних (через WorkshopDao) та віддаленим джерелом даних (WorkshopRemoteDataSource). Якщо база даних порожня, гуртки завантажуються з віддаленого джерела, після чого зберігаються локально для офлайн-доступу. Всі запити на читання та оновлення гуртків виконуються у фоновому режимі (через Dispatchers.IO), що забезпечує стабільну роботу інтерфейсу користувача.

Особливу роль у системі відіграють функції перетворення між моделлю для інтерфейсу користувача (WorkshopModel) та сутністю бази даних (WorkshopEntity). Оскільки Room не підтримує зберігання складних типів шляхів, таких як Map<DayOfWeek, String>, перетворення в Map<Int, String> виконується за допомогою TypeConverter.

Сховище даних реалізовано за допомогою Room, сучасної бібліотеки для роботи з базами даних в Android. Клас WorkshopEntity описує структуру таблиці «my_table» (рис. 3.11), що містить атрибути гуртків, включаючи розклад занять, який зберігається у вигляді рядка JSON.

```
@Entity(tableName = "my_table")
data class WorkshopEntity(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val name: String,
    val address: String,
    val phone: String,
    val website: String,
    val interests: String,
    val days: String,
    val hours: String,
    val price: String,
    val longitude: Double, // Додаємо поле довготи
    val latitude: Double,
    val isLiked: Boolean = false,
    val scheduledTime: Map<Int, String>? = null,
)
```

Рисунок 3.11 - Структура таблиці

WorkshopDao (Data Access Objects) описує SQL-операції над таблицею: вставка, оновлення, вибір вподобаних або всіх клубів та перевірка доступності даних (рис. 3.12).

```
@Dao
interface WorkshopDao {

    @Insert
    suspend fun insert(workshopEntity: WorkshopEntity)

    @Insert
    suspend fun insertAll(workshopEntityList: List<WorkshopEntity>)

    @Update(onConflict = OnConflictStrategy.REPLACE)
    suspend fun update(workshopEntity: WorkshopEntity)

    @Query("SELECT (SELECT COUNT(*) FROM my_table) == 0")
    fun isEmpty(): Boolean

    @Query("SELECT * FROM my_table WHERE isLiked = 1")
    suspend fun getLiked(): List<WorkshopEntity>

    @Query("SELECT * FROM my_table")
    suspend fun getAll(): List<WorkshopEntity>


    @Query("SELECT * FROM my_table WHERE interests = :clubInterests")
    suspend fun getClubByInterests(clubInterests: String): WorkshopEntity?
}
```

Рисунок 3.12 - SQL-запити

Клас WorkshopDatabase забезпечує єдиний доступ до екземпляра бази даних та визначає відповідні сутності та конвертери. companion object з методом getDatabase() реалізує патерн Singleton, гарантуючи, що існує лише один екземпляр бази даних у додатку.

Для зберігання складних структур, таких як Map<Int, String>, у полі scheduledTime використовується клас WorkshopTimeConverter, який перетворює цю карту у формат JSON для запису в базу даних та навпаки для читання.

Віддалене джерело даних реалізовано через інтерфейс WorkshopRemoteDataSource та його фіктивну реалізацію WorkshopMockDataSourceImpl, зображену на рисунку 3.13, яка повертає заздалегідь підготовлений список гуртків.

```
class WorkshopMockDataSourceImpl: WorkshopRemoteDataSource {

    override suspend fun loadWorkshops(): List<WorkshopModel> {
        return listOf(
            WorkshopModel(
                name = "Майстерня Novikova Paint&Craft",
                address = "м. Вінниця, вул. Хлібна 14",
                phone = "(073) 419 32 22",
                website = "https://www.paintcraftworkroom.com/",
                interests = "Малювання",
                days = "Сб-Нд",
                hours = "9:00-17:00",
                price = "Більше 100 грн/год",
                longitude = 28.46022603367229,
                latitude = 49.23368828782602

            ),
            WorkshopModel(
                name = "Центр розвитку дитини «Бджілка»",
                address = "м. Вінниця, вул. Ватутіна, 27",
                phone = "(063) 286 82 82, (068) 644 00 44",
                website = "http://www.bdjilka.vn.ua/",
                interests = "Всебічний розвиток",
                days = "Пн-Нд",
```

Рисунок 3.13 – Джерело даних

Клас WorkshopModel – це центральна структура даних, з якою взаємодіють UI та ViewModel. Він містить корисні методи для обробки даних, такі як розділення текстового представлення днів тижня на список DayOfWeek (рис. 3.14).

```
data class WorkshopModel(
    val id: Int = 0,
    val name: String,
    val address: String,
    val phone: String,
    val website: String,
    val interests: String,
    val days: String,
    val hours: String,
    val price: String,
    val longitude: Double,
    val latitude: Double,
    val isLiked: Boolean = false,
    val scheduledTime: Map<DayOfWeek, String>? = null,
) {
    fun getDaysOfWeek(): List<DayOfWeek> {
        return when(days) {
            "Пн-Нд" -> DayOfWeek.values().asList()
            "Пн-Пт" -> DayOfWeek.values().asList().subList(0, 5)
            "Сб-Нд" -> DayOfWeek.values().asList().subList(5, 7)
            else -> emptyList()
        }
    }
}
```

Рисунок 3.14 - Клас WorkshopModel

Загальна взаємодія така: запити інтерфейсу користувача надсилаються до ViewModel, яка звертається до репозиторію; репозиторій вирішує, чи використовувати локальні чи віддалені дані, та забезпечує їх перетворення для подальшої обробки або відображення. Такий підхід забезпечує масштабованість, гнучкість та легкість обслуговування програми, дозволяючи змінювати джерела даних за потреби, не впливаючи на інші рівні архітектури.

## 3.3 Тестування роботи інформаційно-довідкової системи

Розпочнемо тестування мобільного застосунку, запустивши його на емуляторі в Android Studio. Натиснувши на ярлик під назвою VinClubs (рис. 3.15), застосунок готується до початку роботи, в цей час бачимо екран завантаження зображений на рисунку 3.16


Рисунок 3.15 – Ярлик мобільного додатку


Рисунок 3.16 – Splash Screen

Після цього першою користувач бачить сторінку Clubs (рис 3.17), на якій відображені всі гуртки. Знизу знаходиться навігаційне меню по сторінкам додатку.

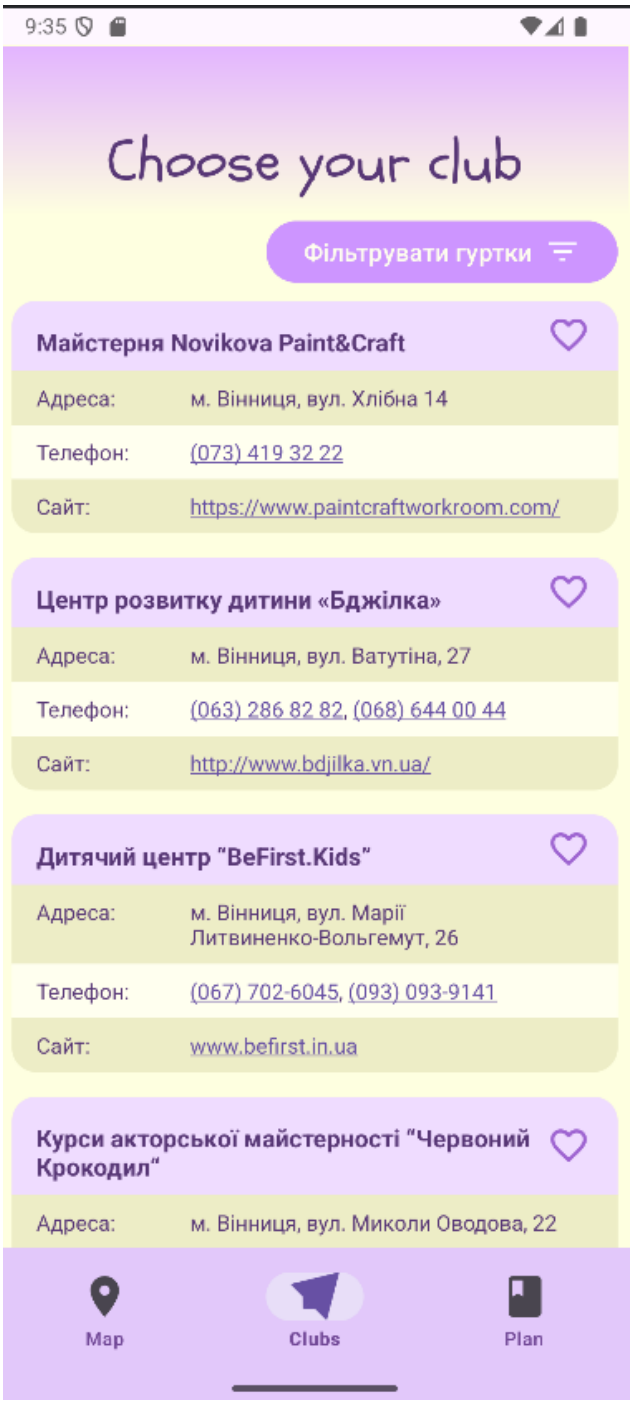



Рисунок 3.17 – Сторінка Clubs

На рисунку 3.18 відображено фільтрування. Натиснувши на кнопку Фільтрувати гуртки, праворуч випливає вікно на якому бачимо фільтри за інтересами, цінами, робочими днями та годинами. Натискаючи на плюсик біля підкатегорії, гуртки фільтруються по даному значенню.

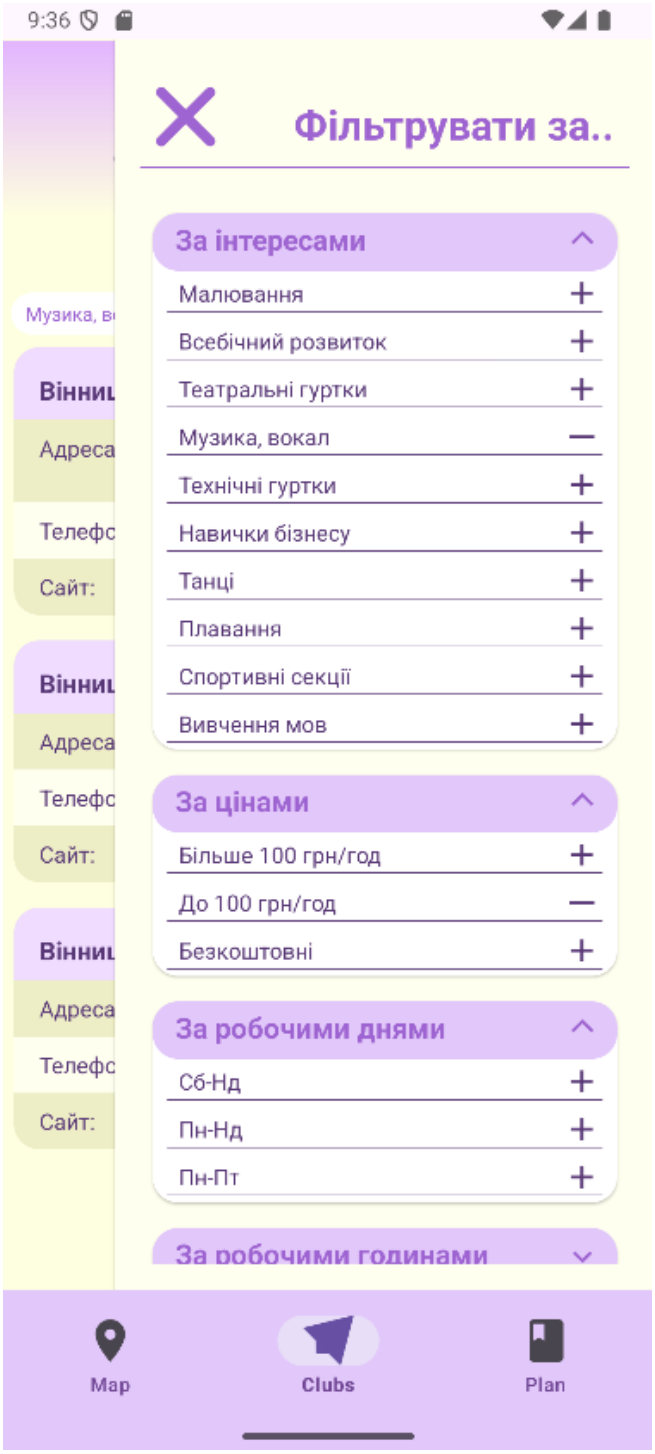



Рисунок 3.18 – Фільтрування гуртків

Після можемо побачити як гуртки відфільтрувались по заданим категоріям, а саме музичні та до ста гривень за годину. Також можемо додати гурток в збережені з допомогою кнопки у вигляді серця, як це показано на рисунку 3.19.

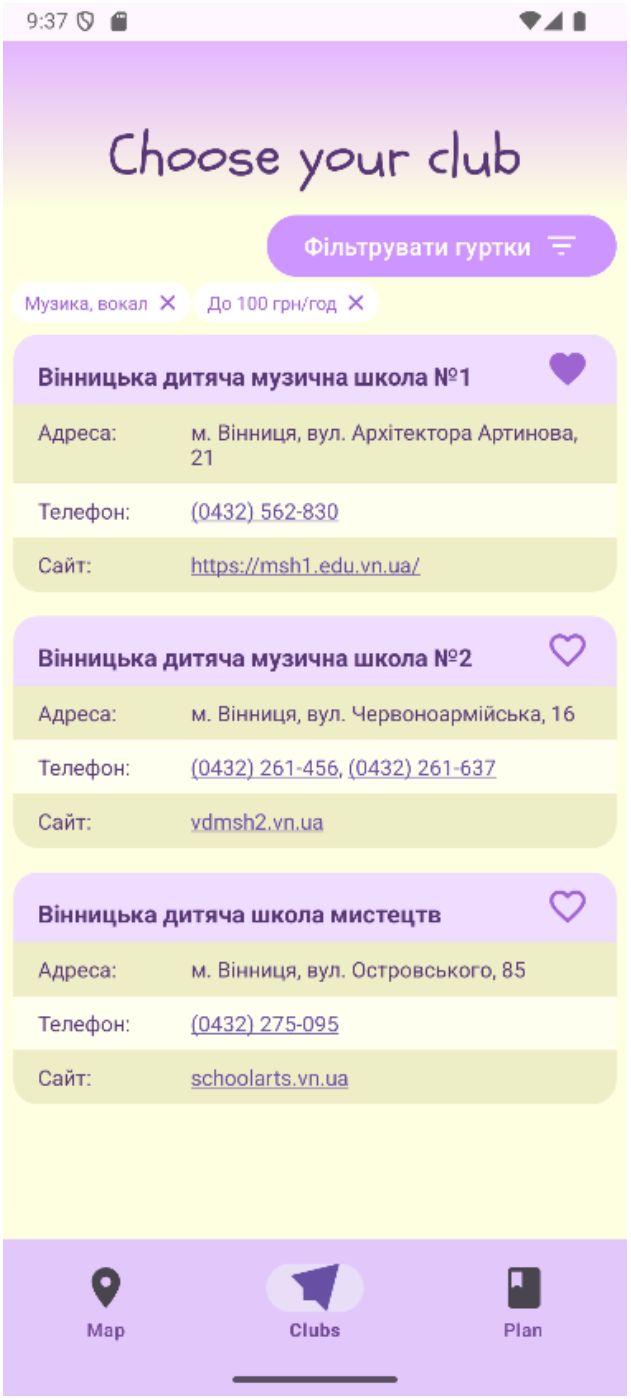



Рисунок 3.19 – Додавання гуртка в обрані

На рисунку 3.20 бачимо наступну сторінку Plan, на якій відображений кадендар та збережені гуртки. Натиснувши на кнопку у вигляді сердечка біля назви гуртка, можемо видалити його з збережених.

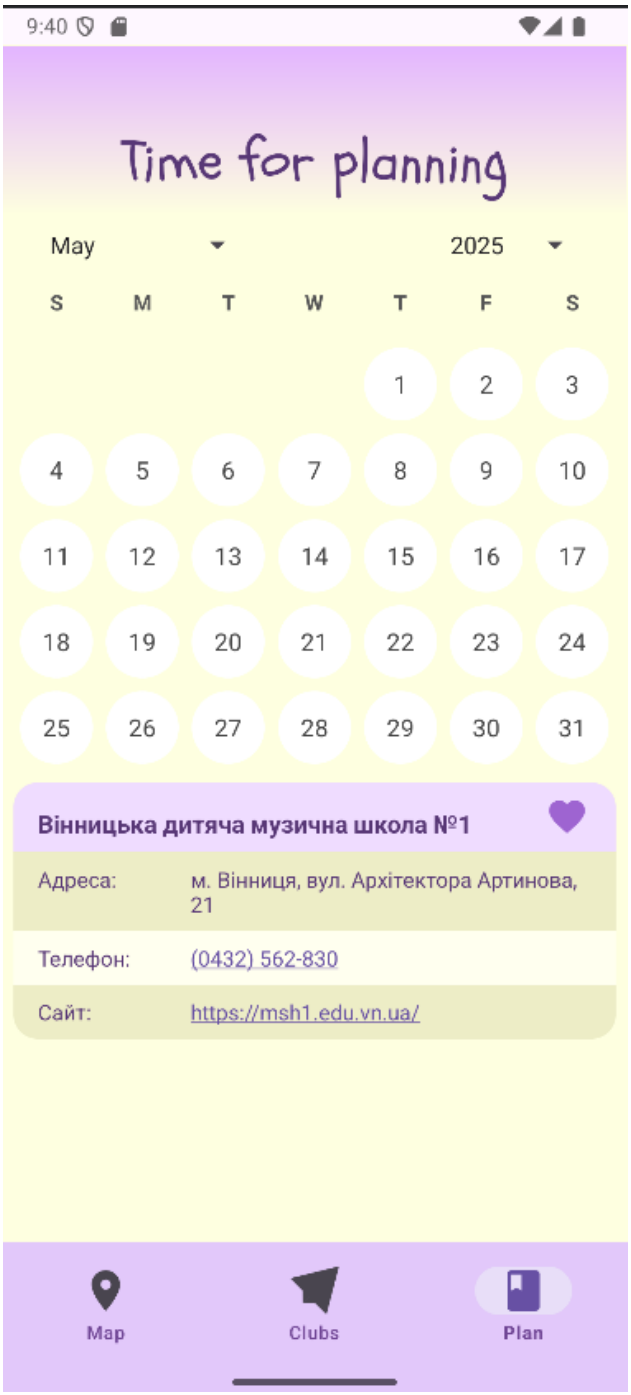



Рисунок 3.20 – Сторінка Plan

Натиснувши на дату на календарі, знизу випливає вікно в якому мають відображатись заплановані заняття (рисунок 3.21). Для того, щоб запланувати заняття натиснемо на великий плюс в правому нижньому кутку.

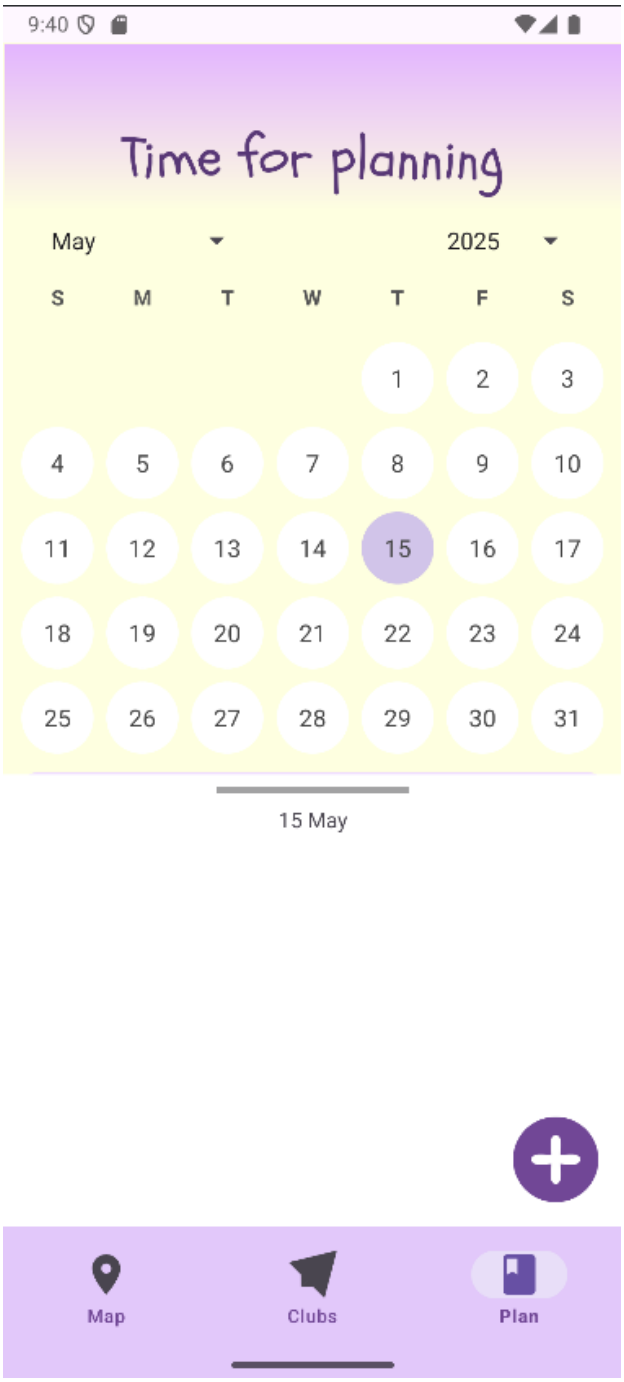



Рисунок 3.21 – Функціонал календаря

Після відкривається вікно (рис. 3.22), де потрібно обрати гурток (лише той, який користувач зберіг попередньо) та час, на який хочемо запланувати відвідування гуртка.

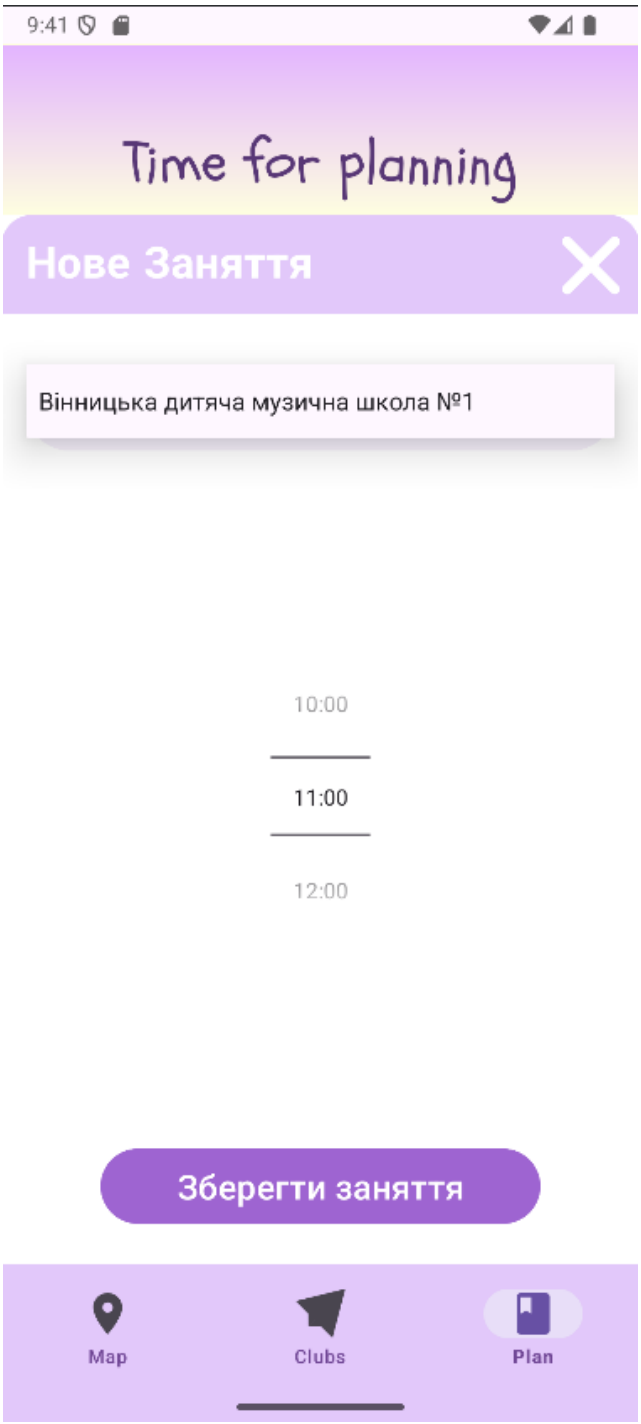



Рисунок 3.22 – Вибір заняття та часу

Після користувач натискає на кнопку Зберегти заняття і гурток з’являється в списку запланованих занять на обрану дату як показано на рисунку 3.23. Можемо видалити гурток з запланованих, натиснувши на кнопку видалення в кінці назви.

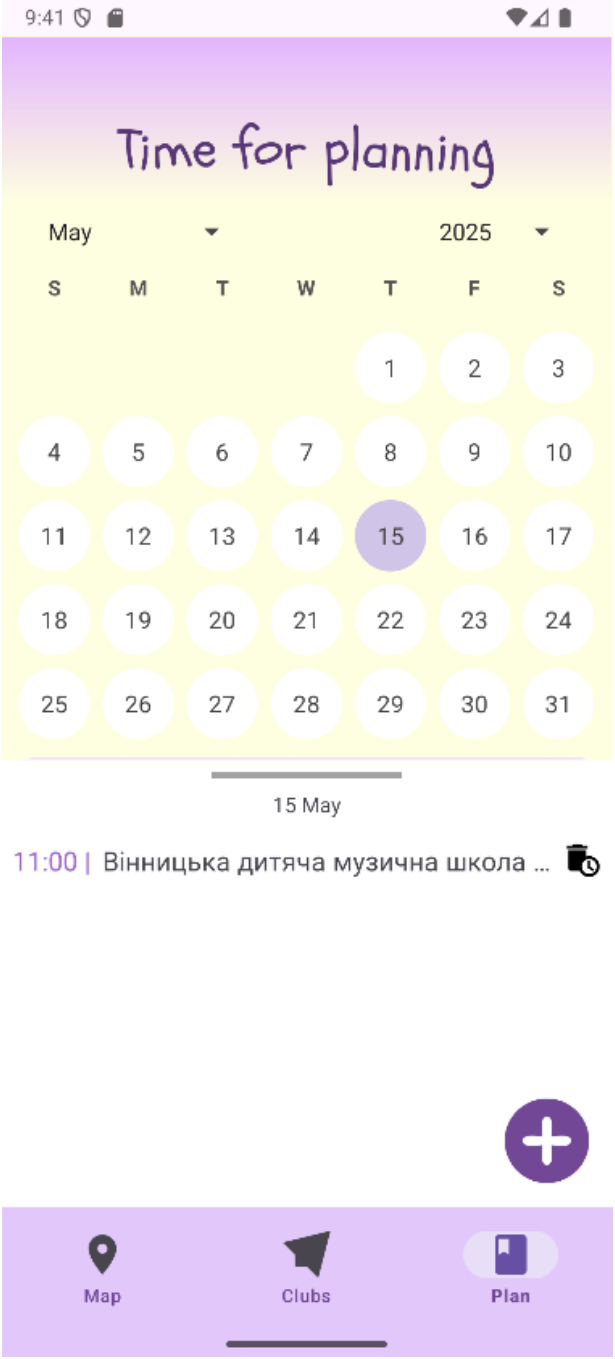



Рисунок 3.23 – Заплановане заняття

Перейшовши на сторінку Map (рис. 3.24), бачимо вбудовану карту Google. На ній усі гуртки позначені кастомними маркерами у вигляді булавки.

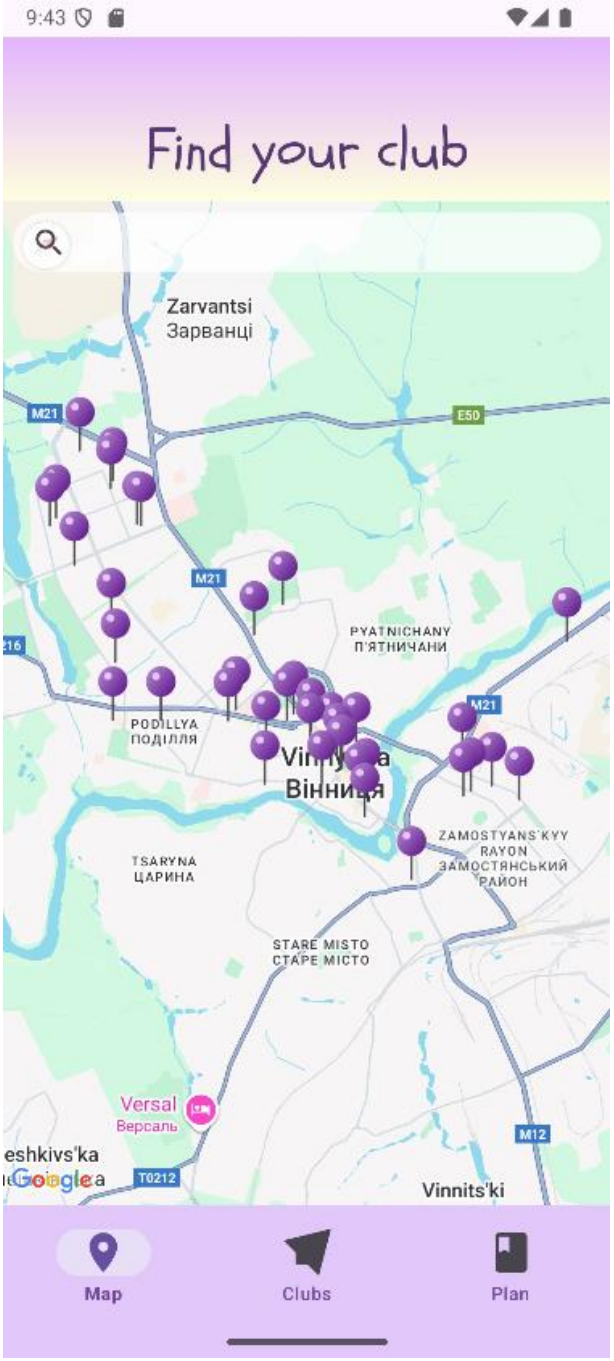



Рисунок 3.24 - Сторінка Map

З допомогою пошукової стрічки можна знайти гурток за назвою (рис. 3.25). Також при натисканні на маркер відображається назва гуртка та його адреса.

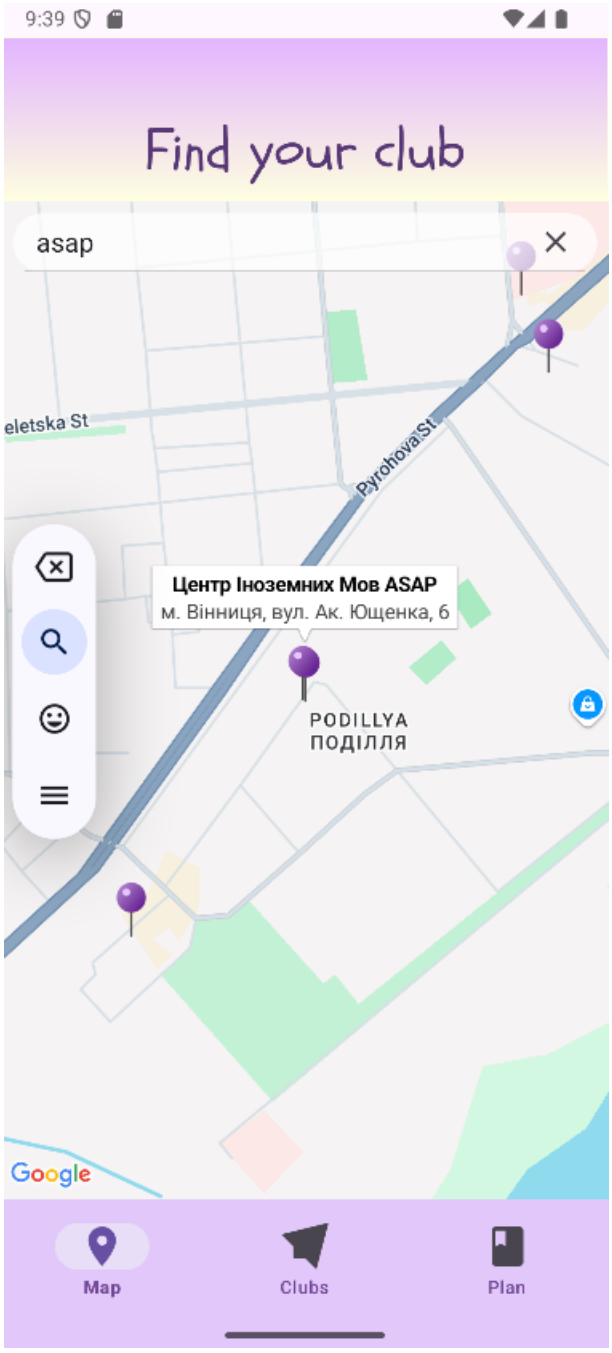
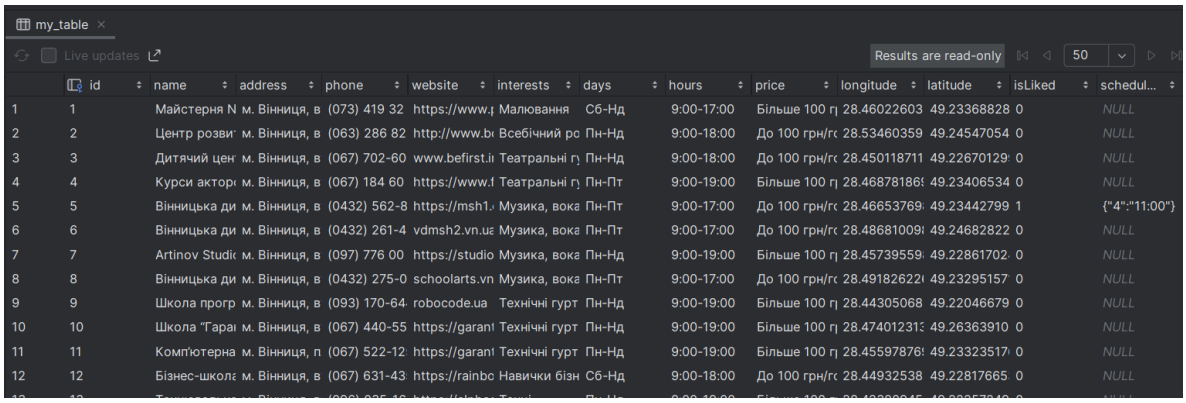



Рисунок 3.25 – Пошук за назвою

Тестування розробленого мобільного додатку показало, що всі основні та допоміжні функції функціонують вірно.

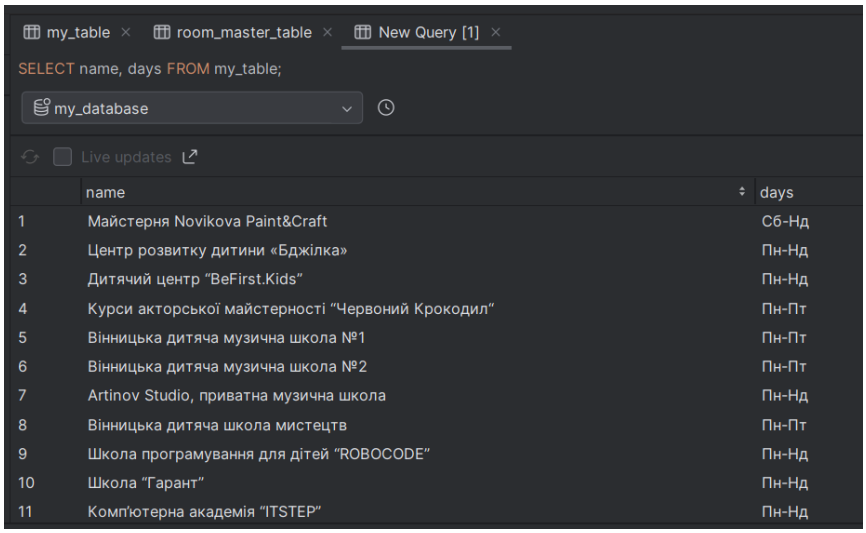
Також під час запущеного додатку в емуляторі в App Inspection можемо оглянути як працює локальна база даних SQLite (рис. 3.26). Бачимо, що біля гуртка з id 5 змінилась характеристика isLiked на одиницю. Це означає, що гурток успішно додався у вибрані. Також змінилось scheduleTime на {«4»:«11:00»}, що означає даний гурток запланований на четвер на одинадцяту годину.




| | id | name | address | phone | website | interests | days | hours | price | longitude | latitude | isLiked | schedul... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Майстерня N | м. Вінниця, в | (073) 419 32 | https://www.г | Малювання | Сб-Нд | 9:00-17:00 | Більше 100 гр | 28.46022603 | 49.23368828 | 0 | NULL |
| 2 | 2 | Центр розви | м. Вінниця, в | (063) 286 82 | http://www.bc | Всебічний ро | Пн-Нд | 9:00-18:00 | До 100 грн/гс | 28.53460359 | 49.24547054 | 0 | NULL |
| 3 | 3 | Дитячий цен | м. Вінниця, в | (067) 702-60 | www.befirst.i | Театральні гу | Пн-Нд | 9:00-18:00 | До 100 грн/гс | 28.450118711 | 49.22670129 | 0 | NULL |
| 4 | 4 | Курси актор | м. Вінниця, в | (067) 184 60 | https://www.f | Театральні гу | Пн-Пт | 9:00-19:00 | Більше 100 гр | 28.46878186 | 49.23406534 | 0 | NULL |
| 5 | 5 | Вінницька ди | м. Вінниця, в | (0432) 562-8 | https://msh1.i | Музика, вока | Пн-Пт | 9:00-17:00 | До 100 грн/гс | 28.46653769 | 49.23442799 | 1 | {"4":"11:00"} |
| 6 | 6 | Вінницька ди | м. Вінниця, в | (0432) 261-4 | vdmsh2.vn.ua | Музика, вока | Пн-Пт | 9:00-17:00 | До 100 грн/гс | 28.48681009 | 49.24682822 | 0 | NULL |
| 7 | 7 | Artinov Studi | м. Вінниця, в | (097) 776 00 | https://studio | Музика, вока | Пн-Нд | 9:00-19:00 | Більше 100 гр | 28.45739559 | 49.22861702 | 0 | NULL |
| 8 | 8 | Вінницька ди | м. Вінниця, в | (0432) 275-0 | schoolarts.vn | Музика, вока | Пн-Пт | 9:00-17:00 | До 100 грн/гс | 28.49182622 | 49.23295157 | 0 | NULL |
| 9 | 9 | Школа прогр | м. Вінниця, в | (093) 170-64 | robocode.ua | Технічні гурт | Пн-Нд | 9:00-19:00 | Більше 100 гр | 28.44305068 | 49.22046679 | 0 | NULL |
| 10 | 10 | Школа "Гара | м. Вінниця, в | (067) 440-55 | https://garant | Технічні гурт | Пн-Нд | 9:00-19:00 | Більше 100 гр | 28.474012313 | 49.26363910 | 0 | NULL |
| 11 | 11 | Комп'ютерна | м. Вінниця, п | (067) 522-12 | https://garant | Технічні гурт | Пн-Нд | 9:00-19:00 | Більше 100 гр | 28.45597876 | 49.23323517 | 0 | NULL |
| 12 | 12 | Бізнес-школа | м. Вінниця, в | (067) 631-43 | https://rainbo | Навички бізн | Сб-Нд | 9:00-18:00 | До 100 грн/гс | 28.44932538 | 49.22817665 | 0 | NULL |

Рисунок 3.26 - Локальна база даних SQLite

Також можемо виконувати прості sql-запити, наприклад виведемо лише назви та робочі дні гуртків, написавши запит SELECT name, days FROM my_table (рис. 3.27).




| | name | days |
|---|---|---|
| 1 | Майстерня Novikova Paint&Craft | Сб-Нд |
| 2 | Центр розвитку дитини «Бджілка» | Пн-Нд |
| 3 | Дитячий центр "BeFirst.Kids" | Пн-Нд |
| 4 | Курси акторської майстерності "Червоний Крокодил" | Пн-Пт |
| 5 | Вінницька дитяча музична школа №1 | Пн-Пт |
| 6 | Вінницька дитяча музична школа №2 | Пн-Пт |
| 7 | Artinov Studio, приватна музична школа | Пн-Нд |
| 8 | Вінницька дитяча школа мистецтв | Пн-Пт |
| 9 | Школа програмування для дітей "ROBOCODE" | Пн-Нд |
| 10 | Школа "Гарант" | Пн-Нд |
| 11 | Комп'ютерна академія "ITSTEP" | Пн-Нд |

Рисунок 3.27 – Sql-запит

## 3.4 Висновки

Архітектура інформаційних систем визначає структуру та розвиток IT-рішень. UML допомагає візуалізувати функціональність системи через діаграми, що спрощує аналіз, проєктування та реалізацію. Таким чином use case діаграма, допомогла зрозуміти функціональність системи гуртків Вінниці. Діаграма послідовності проілюструвала послідовність повідомлень між об'єктами під час планування занять. З допомогою діаграми діяльності описали два можливі

способи пошуку гуртків на карті. На діагарамі класів структурували класи для роботи з даними мобільному додатку, таким чином відобразили як застосунок використовує багаторівневу архітектуру для роботи з даними гуртків. Блок-схема алгоритму роботи програми допомогла зрозуміти логіку виконання програми та послідовність дій. Діаграма хмарної інфраструктури, проілюструвала як у майбутньому система зможе забезпечувати динамічний обмін даними та актуальну інформацію для користувача. При розробленні мобільного додатку для пошуку гуртків дотримувались системного і структурованого підходу до реалізації програмного забезпечення згідно з сучасними принципами Android-розробки. Проведене тестування інформаційно-довідкової системи гуртків на емуляторі Android Studio показало коректну роботу всіх основних та допоміжних функцій. Отже реалізовані рішення відповідають вимогам і створюють основу для подальшого розвитку додатку.

# ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи досягнуто поставлену мету дослідження - розроблено інформаційну систему пошуку гуртків у місті Вінниці у вигляді мобільного Android-додатку. Сформульовані у вступі задачі дослідження були успішно виконані, що дозволило реалізувати повноцінну, функціональну та зручну в користуванні інформаційно-довідкову систему.

Результати аналізу предметної області та вивчення існуючих рішень показали, що наразі відсутні локальні мобільні додатки, які б ефективно вирішували проблему централізованого пошуку гуртків для різних вікових категорій у конкретному місті. Запропонована інформаційна система дозволяє значно спростити цей процес, знижуючи часові витрати користувача та підвищуючи ефективність взаємодії з інформаційними ресурсами.

Вибір IT-інфраструктури та технологій для реалізації інформаційної системи здійснено на основі порівняльного аналізу доступних інструментів за критеріями продуктивності, стабільності, зручності інтеграції, наявності документації, активності спільноти підтримки та відповідності специфіці Android-розробки. У результаті було обґрунтовано використання мови програмування Kotlin, архітектурного підходу MVVM, бібліотеки Room для локального збереження даних, XML для розмітки інтерфейсу та середовища розробки Android Studio. Таке поєднання технологій забезпечило стабільну роботу, гнучкість і масштабованість системи. Крім того, інтеграція з Google Maps SDK надала змогу реалізувати зручну просторову візуалізацію місць розташування гуртків, що підвищує зручність навігації для користувачів.

Порівняно з існуючими веб-рішеннями або неструктурованою інформацією в соціальних мережах, створена система має низку переваг: локальну автономну роботу без потреби постійного доступу до Інтернету, інтерфейс українською мовою, адаптований під потреби місцевого користувача, а також чітку категоризацію гуртків.

Ефективність інформаційної системи оцінювалась за такими критеріями, як зручність пошуку, швидкодія інтерфейсу, стабільність при навантаженні, точність відображення даних. За результатами тестування підтверджено відповідність системи функціональним вимогам та очікуванням користувачів.

Разом з тим, поточна версія має певні обмеження, зокрема відсутність онлайн-синхронізації, що унеможливлює обмін даними між пристроями. У подальшому розвиток системи може включати розширення функціоналу за рахунок інтеграції з базами даних закладів освіти, впровадження системи реєстрації/бронювання місць, чатів між користувачами та адмінами гуртків, а також додавання системи рекомендацій на основі уподобань користувача.

Таким чином, розроблена інформаційна система для пошуку гуртків у місті Вінниці демонструє реальну практичну цінність, забезпечуючи користувачам простий, швидкий і зручний доступ до структурованої інформації про позашкільну зайнятість, що особливо важливо в умовах динамічного ритму життя сучасного суспільства.

За результатами виконання даної роботи опубліковано тези доповідей на тему: «Інформаційно-довідкова система гуртків міста Вінниці» на LIV Всеукраїнській науково-технічній конференцій факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.).

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Крижановський Є.М., Шпорт Г.Ю., Штельмах І.М., Інформаційно-довідкова система гуртків міста Вінниці. *LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.).* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23686/19657 (дата звернення: 06.05.2025)

2. Зведений звіт позашкільних навчальних закладів. URL: https://iea.gov.ua/wp-content/uploads/2024/04/1-1-pz-derzh-kom-pryvat-2023-2024.pdf (дата звернення: 06.05.2025)

3. Meetup: Social Events & Groups. URL: https://play.google.com/store/apps/details/Meetup_Social_Events_Groups?id=com.meetup&hl=uk (дата звернення: 07.05.2025)

4. ClassPass: Fitness, Spa, Salon. URL: https://play.google.com/store/apps/details?id=com.classpass.classpass&hl=uk&pli=1 (дата звернення: 07.05.2025)

5. Locals: Clubs, Events, People. URL: https://play.google.com/store/apps/details?id=org.locals.android&hl=uk (дата звернення: 07.05.2025)

6. Вінницька міська рада. URL: https://2021.vmr.gov.ua/HobbyGroups/Lists/Groups/AllGroups.aspx (дата звернення: 07.05.2025)

7. Peter Späth. Learn Kotlin for Android Development : підручник. Видавництво Apress, 2019. 508 с.

8. Talib M. S., Al-Najar A. A.-M., Hassan A. H., Talib Z.S., Kotlin Programming Language: A Comprehensive Overview of Definition, Applications, Advantages, and Limitations. *International Journal of Computer Applications Technology and Research.* 2024. Vol. 13, Issue 03. P. 07–09. DOI: 10.7753/IJCATR1303.1002. (дата звернення: 08.05.2025)

9. Antonio Leiva. Kotlin for Android Developers: підручник. Видавництво Leanpub, 2017. 180 с.

10. What is Kotlin? URL: https://www.techtarget.com/whatis/definition/Kotlin (дата звернення: 08.05.2025)

11. XML in Android: Basics And Different XML Files Used In Android. URL: https://abhiandroid.com/ui/xml#gsc.tab=0 (дата звернення: 09.05.2025)

12. Гребенюк Д.О., Константинова Л.В., Огляд та порівняння інструментів для створення інтерфейсів мобільних додатків на Android. *VIII Міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" (Кропивницький, 2025 р.).* URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/5851f27f-be1b-4069-ae81-7f7fe7fac87c/content#page=62 (дата звернення: 09.05.2025)

13. Neil Smyth. Android Studio Arctic Fox Essentials - Java Edition: Developing Android Apps Using Android Studio 2020.31 and Java : підручник. Видавництво eBookFrenzy, 2021. 778 с.

14. 7 Top IDEs for Android Development in 2024. URL: https://lanars.com/blog/ide-for-android-development (дата звернення: 09.05.2025)

15. Kyle Mew. Mastering Android Studio 3 : підручник. Видавництво Packt Publishing Ltd, 2017. 220 с.

16. Persist data with Room. URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-persisting-data-room#0 (дата звернення: 09.05.2025)

17. Bartoszek, A., Comparative analysis of database types in mobile applications running on the Android operating system. *Journal of Computer Sciences Institute*. 2024. №31. С. 82–88. doi: 10.35784/jcsi.5915. (дата звернення: 09.05.2025)

18. Svennerberg G. Beginning Google Maps API 3 / G. Svennerberg. – 2-е вид. – New York : Apress, 2010. 384 с.

19. Maps SDK for Android overview. URL: https://developers.google.com/maps/documentation/android-sdk/overview (дата звернення: 09.05.2025)

20. Exploring the Benefits of the Google Maps API. URL: https://www.theneo.io/blog/exploring-the-benefits-of-the-google-maps-api (дата звернення: 09.05.2025)

21. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. Черкаси : Черкаський національний університет ім. Б. Хмельницького, 2017. 434 с.

22. Технології програмування та створення програмних продуктів: конспект лекцій / укладач О. В. Алексенко. Суми : Сумський державний університет, 2013. 133 с.

23. use case. URL: https://www.techtarget.com/searchsoftwarequality/definition/use-case (дата звернення: 12.05.2025)

24. Sequence diagrams. URL: https://www.ibm.com/docs/es/radfws/9.6.1?topic=diagrams-sequence (дата звернення: 12.05.2025)

25. Marand E. A., Sheikhahmadi A., Challenger M., Moradi P., Khalilipour A., Recommender Systems for Unified Modeling Language and Vice Versa-A Systematic Literature Review. *IEEE Access*. 2025. vol. 13. pp. 23426-23460, doi: 10.1109/ACCESS.2025.3535527. (дата звернення: 12.05.2025).

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

# ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

ІНФОРМАЦІЙНО-ДОВІДКОВА СИСТЕМА ГУРТКІВ МІСТА ВІННИЦІ

08-34.БКР.026.00.000 ТЗ

Керівник: к.т.н., ас.
_____ Ігор ШТЕЛЬМАХ
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Ганна ШПОРТ
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) Джемеров Д., Ісакова С. Kotlin in Action : підручник. Видавництво Manning Publications, 2017. 336 с.;

2) Kyle Mew. Mastering Android Studio 3 : підручник. Видавництво Packt Publishing Ltd, 2017. 220 с.;

3) Persist data with Room. URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-persisting-data-room#0.

3. Мета і призначення роботи.

Метою дослідження є розроблення інформаційно-довідкової системи гуртків міста Вінниці.

4. Вихідні дані для проведення робіт:

Дані про гуртки міста Вінниці.

5. Методи дослідження:

Аналіз вимог, аналіз потреб користувачів, аналіз та порівняння існуючих мобільних додатків, вибір оптимальної ІТ-інфраструктури, моделювання архітектури інформаційної системи, розроблення прототипу у середовищі Android Studio, тестування функціоналу системи.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Вибір оптимальної ІТ-інфраструктури _____ – _____
c) Розроблення інформаційно-довідкової системи гуртків міста Вінниці _____ – _____
d) Тестування інформаційно-довідкової системи гуртків міста Вінниці _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Розроблена інформаційно-довідкова система гуртків міста Вінниці.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.
Граничні терміни виконання БКР «___» _____ 2025 р.

Розробив студент групи 2ІСТ-21б _____ Ганна ШПОРТ

Додаток Б

(обов’язковий)

**ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

Назва роботи: «Інформаційно-довідкова система гуртків міста Вінниці»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,94 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

□ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

□ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ігор ШТЕЛЬМАХ, к.т.н., ас. каф. САІТ
(підпис)

Здобувач _____ Ганна ШПОРТ
(підпис)

# Додаток В

## (довідниковий)

## Фрагмент лістингу програми

```
package com.example.myyyyapplication.presentation
import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.fragment.app.Fragment
import com.example.myyyyapplication.R
import com.example.myyyyapplication.databinding.ActivityMainBinding
import com.example.myyyyapplication.presentation.fragments.ClubsFragment
import com.example.myyyyapplication.presentation.fragments.MapFragment
import com.example.myyyyapplication.presentation.fragments.PlanFragment
class MainActivity : AppCompatActivity() {
    private lateinit var binding : ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        binding.bottomNavigationView.setOnItemSelectedListener{
            when(it.itemId){
                R.id.clubs -> replaceFragment(ClubsFragment())
                R.id.map -> replaceFragment(MapFragment())
                R.id.plan -> replaceFragment(PlanFragment())
                else -> {
                }
            }
            true
        }
        binding.bottomNavigationView.selectedItemId = R.id.clubs
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
```

```
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, 0)
            insets
        }
    }
    private fun replaceFragment(fragment: Fragment){
        val fragmentManager = supportFragmentManager
        val fragmentTransaction = fragmentManager.beginTransaction()
        fragmentTransaction.replace(R.id.frameLayout, fragment)
        fragmentTransaction.commit()
    }
}
```

```
package com.example.myyyyapplication.presentation.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.TextView
import androidx.activity.OnBackPressedCallback
import androidx.constraintlayout.helper.widget.Flow
import androidx.core.view.GravityCompat
import androidx.core.view.isGone
import androidx.drawerlayout.widget.DrawerLayout.DrawerListener
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.myyyyapplication.R
import com.example.myyyyapplication.databinding.FragmentClubsBinding
import com.example.myyyyapplication.presentation.adapters.ClubsAdapter
import com.example.myyyyapplication.presentation.viewmodel.PageClubsViewModel
import org.koin.androidx.viewmodel.ext.android.viewModel
class ClubsFragment : Fragment() {
    private var _binding: FragmentClubsBinding? = null
    private val binding get() = _binding!!
    private val vm: PageClubsViewModel by viewModel()
```

```
private val onBackPressedCallback: OnBackPressedCallback = object : OnBackPressedCallback(true) {
    override fun handleOnBackPressed() {
        if (binding.drawerLayout.isDrawerOpen(GravityCompat.END)) {
            binding.drawerLayout.closeDrawer(GravityCompat.END)
        } else {
            activity?.finish()
        }
    }
}
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    _binding = FragmentClubsBinding.inflate(inflater, container, false)
    return _binding!!.root
}
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    activity?.onBackPressedDispatcher?.addCallback(onBackPressedCallback)
    subscribeToWorkshops()
    subscribeToFilters()
    subscribeToFilteredBy()
    binding.btnFilter.setOnClickListener {
        binding.drawerLayout.visibility = View.VISIBLE
        binding.drawerLayout.openDrawer(GravityCompat.END)
    }
    initDrawer()
}
private fun initDrawer() {
    binding.drawerLayout.addDrawerListener(object : DrawerListener {
        override fun onDrawerSlide(drawerView: View, slideOffset: Float) {
        }
        override fun onDrawerOpened(drawerView: View) {
            binding.drawerLayout.visibility = View.VISIBLE
        }
        override fun onDrawerClosed(drawerView: View) {
            binding.drawerLayout.visibility = View.GONE
        }
```

```
        override fun onDrawerStateChanged(newState: Int) {
        }
    })
    binding.ivClose.setOnClickListener {
        binding.drawerLayout.closeDrawer(GravityCompat.END)
    }
}
private fun subscribeToWorkshops() {
  vm.workshopsLiveData.observe(viewLifecycleOwner) { workshops ->
      binding.rvClubs.layoutManager = LinearLayoutManager(context)
      binding.rvClubs.adapter = ClubsAdapter(workshops) { workshop ->
          vm.updateWorkshop(workshop)
      }
  }
}
private fun subscribeToFilters() {
    vm.filtersLiveData.observe(viewLifecycleOwner) {
        it.forEach { (filterHeader, filters) ->
            val root = layoutInflater.inflate(R.layout.filters_container, binding.filtersContainer, false)
            val tvFiltersHeared = root.findViewById<TextView>(R.id.tvFiltersHeader)
            tvFiltersHeared.text = filterHeader
            val container = root.findViewById<LinearLayout>(R.id.filtersContainer)
            val expand = root.findViewById<ImageView>(R.id.ivExpand)
            expand.setOnClickListener {
                if (container.isGone) {
                    container.visibility = View.VISIBLE
                    expand.setImageResource(R.drawable.arrow_up_24)
                } else {
                    container.visibility = View.GONE
                    expand.setImageResource(R.drawable.arrow_down_24)
                }
            }
            filters.forEach {
                val filterRoot = layoutInflater.inflate(R.layout.filter_item, root as? ViewGroup, false)
                val tvFilter = filterRoot.findViewById<TextView>(R.id.tvFilter)
                tvFilter.text = it
                filterRoot.tag = it
```

```
                val ivAdd = filterRoot.findViewById<ImageView>(R.id.ivAdd)
                ivAdd.setOnClickListener {
                    if (ivAdd.tag == "add"){
                        ivAdd.setImageResource(R.drawable.baseline_remove_24) //todo
                        ivAdd.tag = ""
                        vm.addFilter(tvFilter.text as String)
                    } else {
                        ivAdd.tag = "add"
                        ivAdd.setImageResource(R.drawable.baseline_add_24)
                        vm.removeFilter(tvFilter.text as String)
                    }
                }
                container.addView(filterRoot)
            }
            binding.filtersContainer.addView(root)
        }
    }
}
    private fun subscribeToFilteredBy() {
        vm.filteredByLiveData.observe(viewLifecycleOwner) { filtersList ->
            binding.filtersFlow.post {
                binding.filtersFlow.removeAllViews()
                if (filtersList.isNotEmpty()) {
                    filtersList.forEach {
                        val filter = layoutInflater.inflate(R.layout.grid_filter_item, binding.rootConstraintLayout, false)
                        val tv = filter.findViewById<TextView>(R.id.tvFilter)
                        tv.text = it
                        filter.setOnClickListener {
                            val ivAdd = binding.filtersContainer.findViewWithTag<View>(tv.text.toString()).findViewById<ImageView>(R.id.ivAdd)
                            ivAdd.tag = "add"
                            ivAdd.setImageResource(R.drawable.baseline_add_24)
                            vm.removeFilter(tv.text.toString())
                        }
                        filter.id = View.generateViewId()
                        binding.rootConstraintLayout.addView(filter)
                        binding.filtersFlow.addView(filter)
```

```
                }
            }
        }
    }
}
private fun Flow?.removeAllViews() {
    this?.referencedIds?.forEach { viewId ->
        binding.rootConstraintLayout.findViewById<View>(viewId)?.let {
            binding.rootConstraintLayout.removeView(it)
        }
    }
    this?.referencedIds = intArrayOf()
}
override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
}
}
```

Додаток Г

(обов’язковий)

**ІЛЮСТРАТИВНА ЧАСТИНА**

ІНФОРМАЦІЙНО-ДОВІДКОВА СИСТЕМА ГУРТКІВ МІСТА ВІННИЦІ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

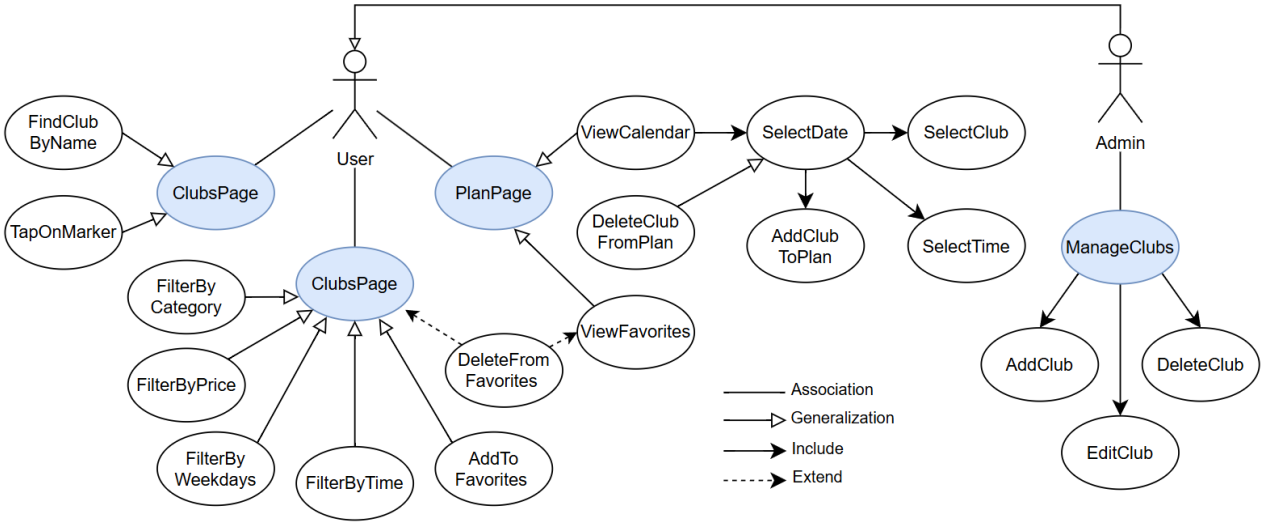



Рисунок Г.1 - Діаграма use case

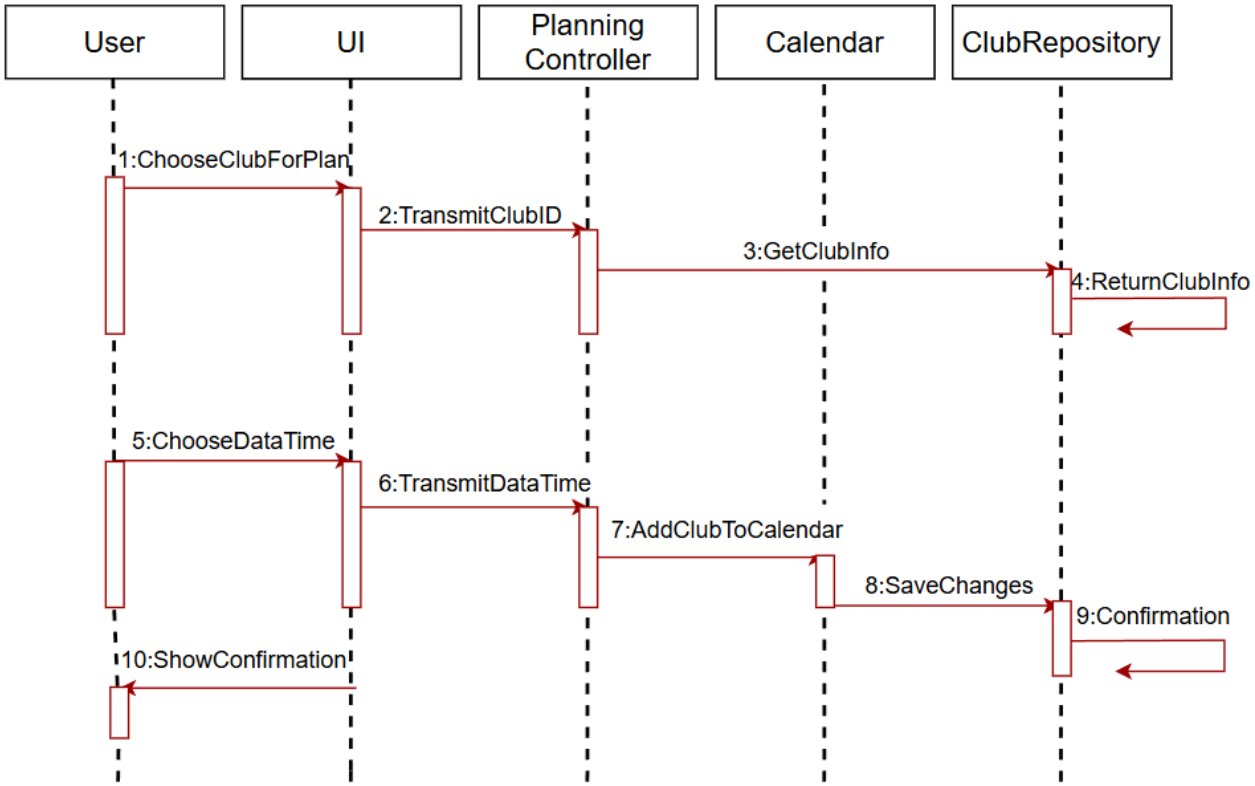



Рисунок Г.2 – Діаграма послідовності для планування занять

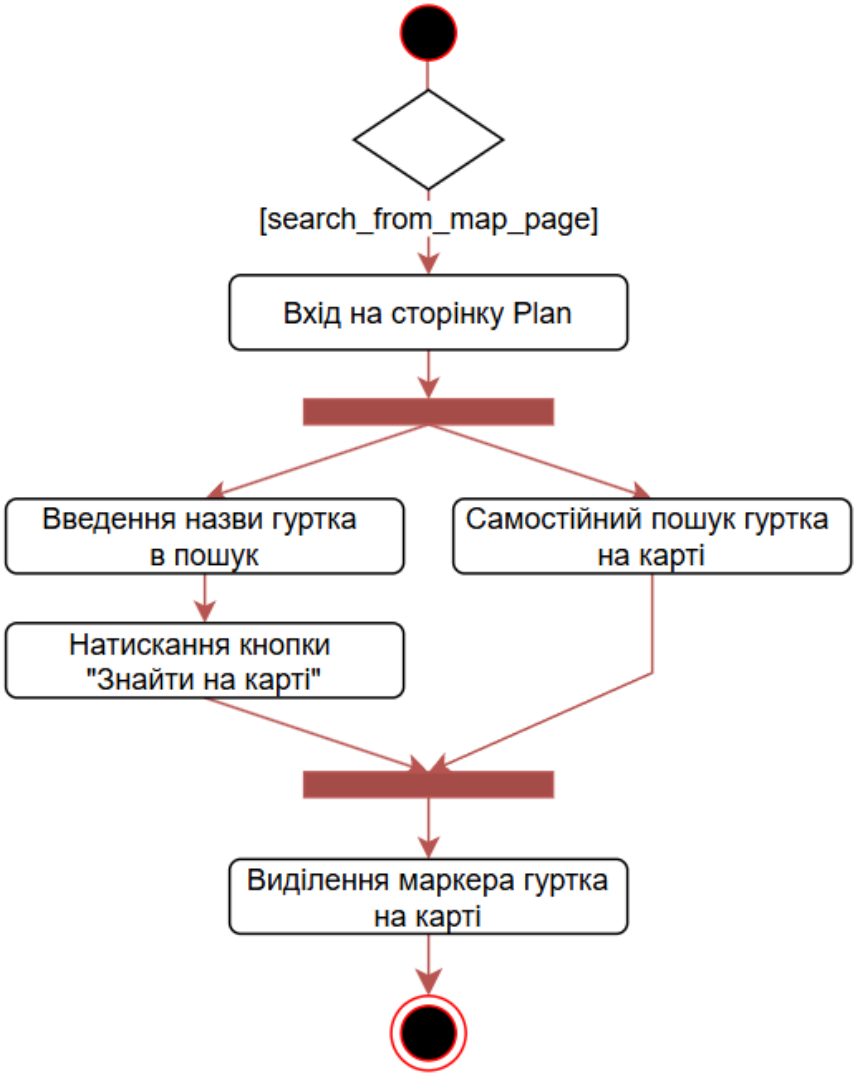



Рисунок Г.3 – Діаграма діяльності для пошуку гуртків на карті

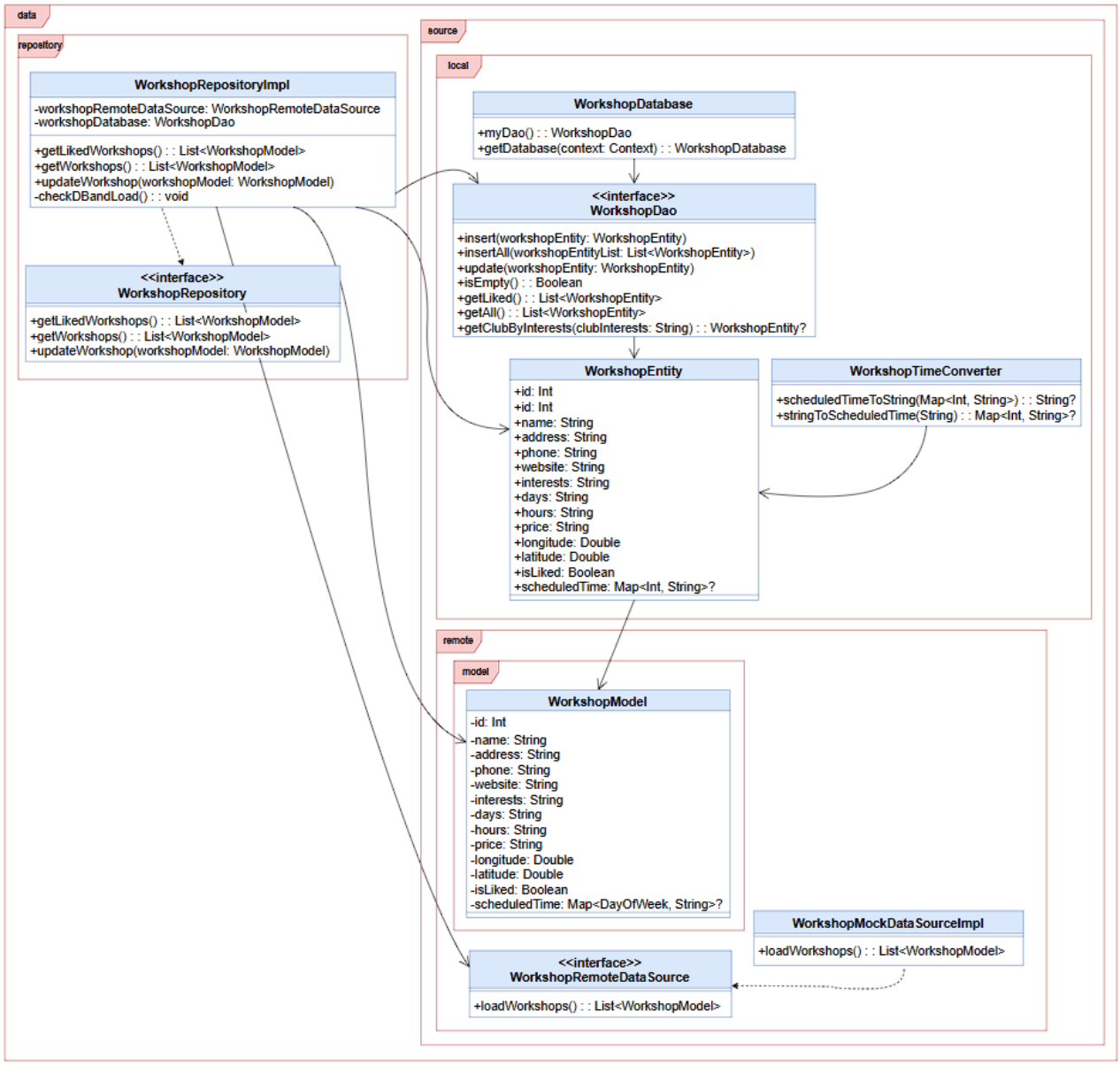



Рисунок Г.4 – Діаграма класів

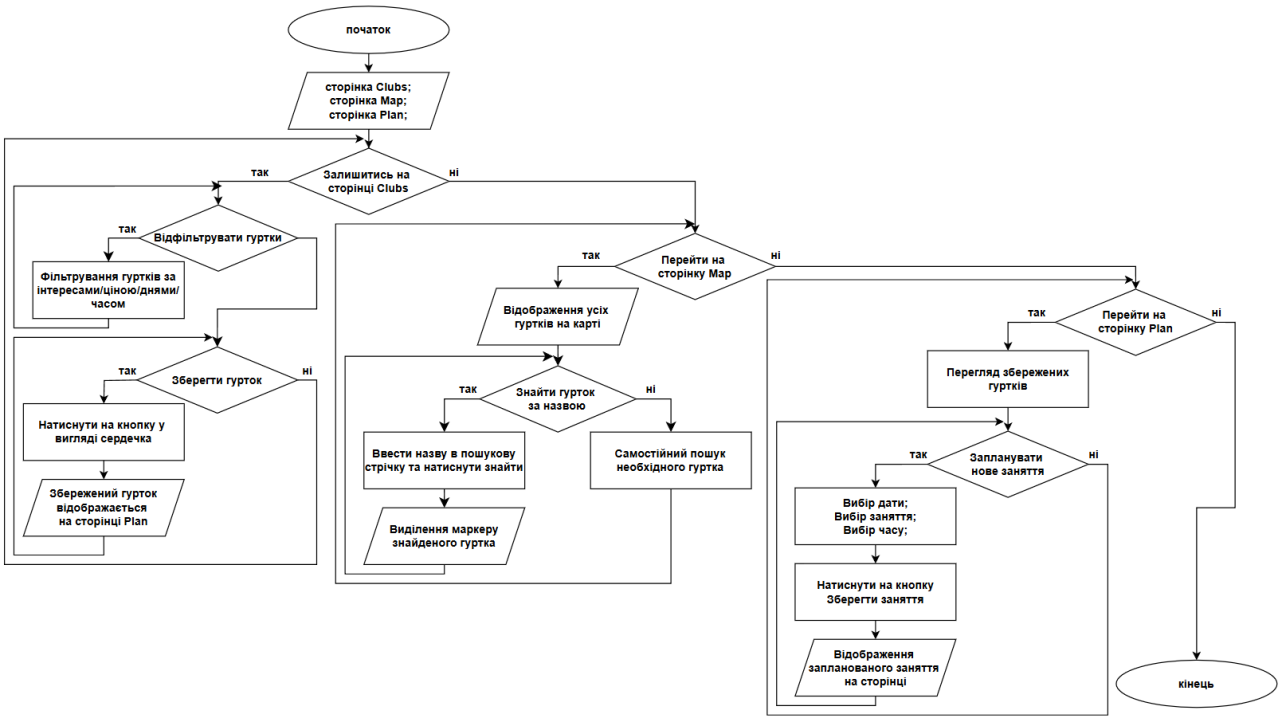



Рисунок Г.5 - Блок-схема алгоритму роботи програми

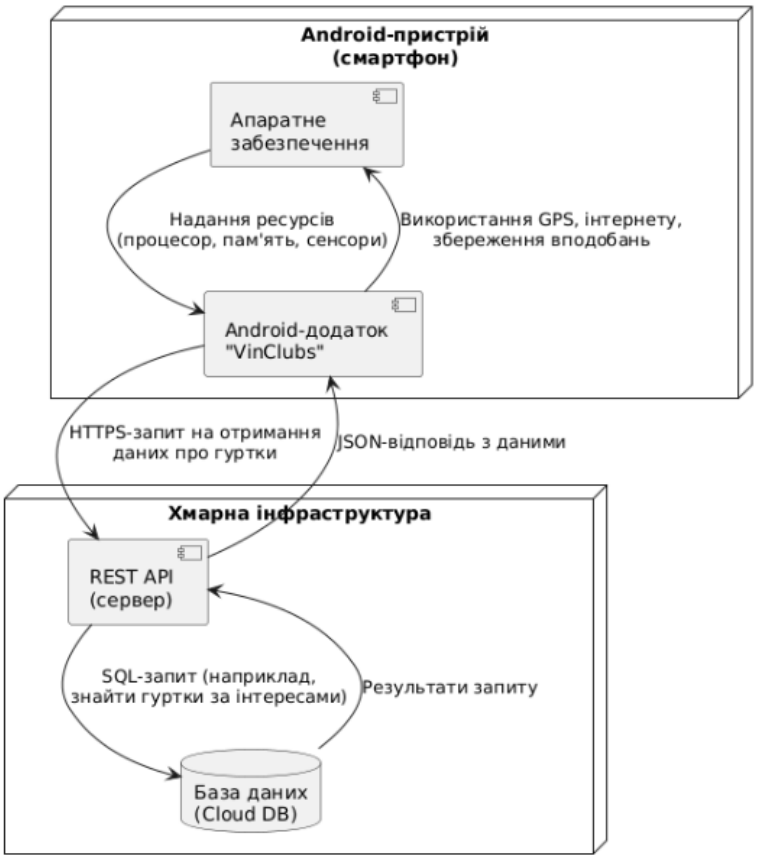



Рисунок Г.6 - Взаємодія між пристроєм та хмарою

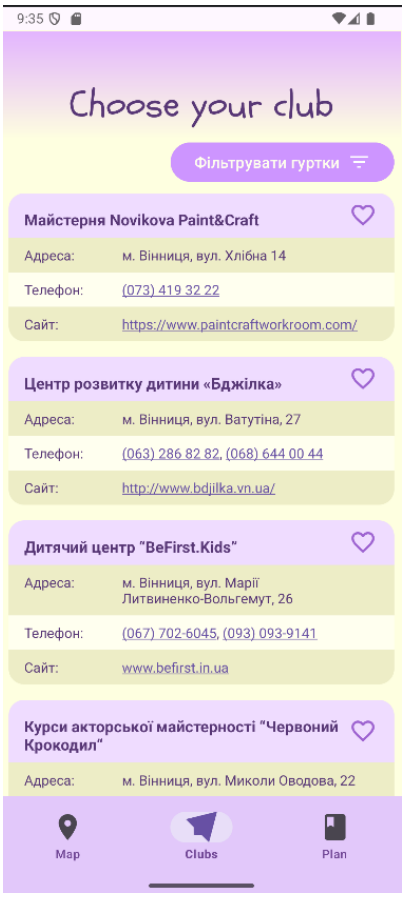



Рисунок Г.7 – Сторінка Clubs

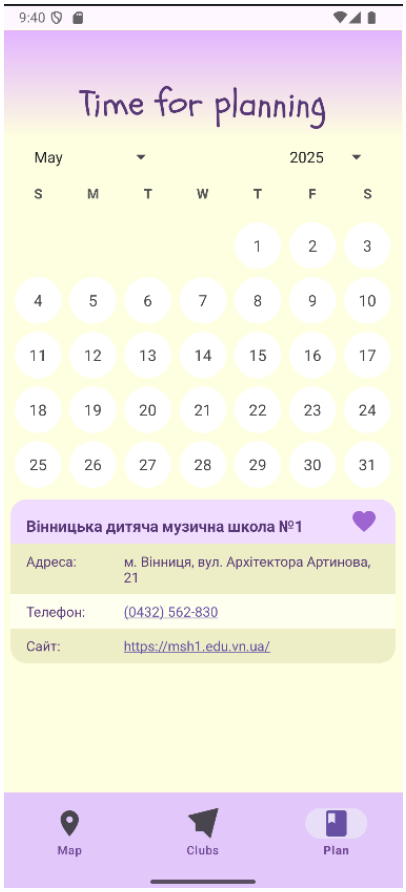



Рисунок Г.8 – Сторінка Plan

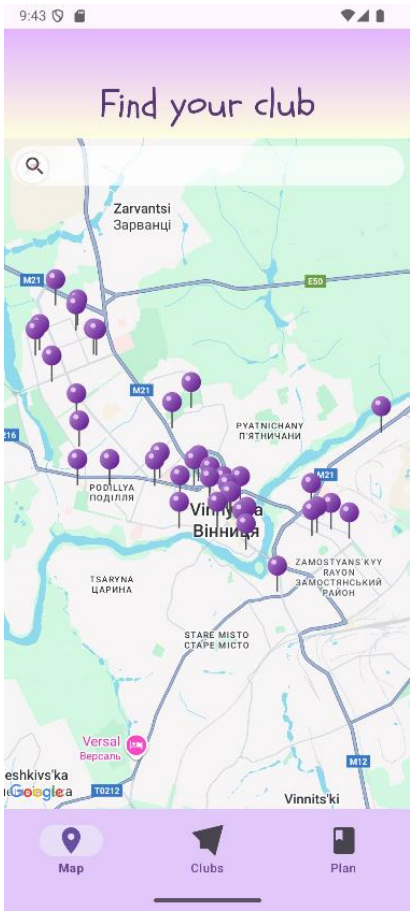



Рисунок Г.9 - Сторінка Map