

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук

Кафедра інформаційних технологій

ДИПЛОМНИЙ ПРОЕКТ

Рівень вищої освіти бакалавр

на тему: «Розробка програми “Аптечний пункт”»

Виконав студент 4 курсу групи К-42

Напрямок 6.050101 Комп'ютерні науки

Самедов Турал Нариман огли

Керівник: д.ф.-м.н., проф.

Ковальчук Володимир Володимирович

Рецензент: д.ф.-м.н., проф.

Тюрин Олександр Валентинович

ЗМІСТ

ЗМІСТ.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	6
1 СУЧАСНИЙ СТАН МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ПОШУКУ ЛІКРСЬКИХ ПРЕПАРАТІВ.....	8
1.1 Огляд програмних продуктів схожих за призначенням та функціями.....	8
1.1.1 Опис програми “Geo Apteka”.....	8
1.1.2 Опис програми “Аптеки України”.....	9
1.1.3 Опис програми “Аптечний пункт”.....	10
1.2 Аналіз вимог до програмного продукту.....	11
2 ПРОГРАМНІ ТЕХНОЛОГІЇ, ЯКІ ВИКОРИСТОВУВАЛИСЯ ПРИ РОЗРОБЦІ ПРОЕКТУ.....	13
2.1 Коротке описання можливих інструментів для розробки.....	13
2.2 Мова Java.....	13
2.3 Опис середовища Android Studio.....	14
2.4 Карти та їх застосування.....	15
2.5 Графічний інтерфейс мовою XML.....	16
2.6 Опис серверної взаємодії.....	16
2.7 Система контролю версій.....	18
3 ТЕХНІЧНЕ ЗАВДАННЯ ТА ПРОЕКТУВАННЯ.....	20
3.1 Назва та призначення.....	20
3.2 Вимоги до проекту.....	20
3.2.1 Вимоги до вхідних даних.....	20
3.2.2 Вимоги до функціональних характеристик.....	20
3.2.3 Вимоги до забезпечення надійного функціонування ПП.....	21
3.2.4 Час відновлення після відмови.....	21
3.2.5 Вимоги до засобів реалізації.....	21
3.2.6 Вимоги до зберігання даних.....	21
3.2.7 Вимоги до інформаційної та програмної сумісності.....	21
3.2.8 Вимоги до програмної документації.....	22
3.3 Умови до експлуатації та програмного забезпечення.....	22
3.3.1 Кліматичні умови експлуатації.....	22
3.3.2 Вимоги до кваліфікації та чисельності персоналу.....	22
3.3.3 Економічні переваги розробки.....	22
3.4 Стадії та етапи розробки.....	22
3.4.1 Стадії розробки.....	22
3.4.2 Етапи розробки.....	23
3.5 Проектування.....	23
3.5.1 Діаграма функціонального моделювання (SADT).....	24
3.5.2 Діаграма декомпозиції IDEF0.....	25
3.5.3 Діаграма дерева вузлів.....	28
3.5.4 Схема бази даних.....	28

3.5.5 Модель життєвого циклу.....	28
3.5.6 Побудова ієрархічної структури та розрахунків нев'язки.....	29
4 ОГЛЯД ГОТОВОГО ПРОГРАМНОГО ПРОДУКТУ.....	31
4.1 Встановлення програми.....	31
4.2 Опис програмного продукту.....	31
ВИСНОВКИ.....	40
ПЕРЕЛІК ПОСИЛАНЬ.....	41
Д О Д А Т К И.....	43
ДОДАТОК А ФРАГМЕНТ ПРОГРАМНОГО КОДУ.....	44

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

- БД – База даних
- ЗІЗ – Засоби індивідуального захисту
- ART – Android Runtime (віртуальна машина Android)
- EDGE – Підвищена швидкість передачі даних для розвитку GSM
- GPS – Global Positioning System (Супутникова система навігації)
- GSM – Global System for Mobile Communications (Глобальна система мобільного зв'язку)
- HTML – HyperText Markup Language
- JVM – java virtual machine (віртуальна машина Java)
- NDK – native development kit (комплект розробки програмного забезпечення базується на C++)
- OS – Operating System (Операційна система)
- SADT – Structured Analysis and Design Technique
- SDK – Software development kit (комплект розробки програмного забезпечення)
- UML – Unified Modeling Language
- XML – Extensible Markup Language (розширювана мова розмітки)

ВСТУП

В наш час все частіше інформаційні технології приходять на допомогу пересічному споживачеві, для полегшення використання великого обсягу інформації, орієнтації в ньому і правильної й швидкої обробки цього потоку інформації. Це відноситься до різних сфер життєдіяльності людини і, в тому числі і до сфери охорони здоров'я. Сьогодні налічується більше 100 тисяч мобільних додатків mHealth, розроблених на двох великих платформах: iOS і Android. Обидві є абсолютними лідерами серед операційних систем для мобільних телефонів. Однак варто зазначити, що протягом останніх 2 років спостерігається значне зростання кількості додатків в Google Play, розроблених на базі Android, які відносяться до сфери здорового способу життя та медичної галузі [1]. Додатки для смартфонів в сфері охорони здоров'я мають різні напрямки, одне з яких, це можливість постійного доступу до інформації про наявність лікарських препаратів в найближчих аптеках. Дуже зручно мати в своєму мобільному телефоні програму, яка могла б підказати, найближчу аптеку, в якій є необхідні лікарські препарати, їх ціни, а також можливість пересування до обраної аптеки.

Метою цієї роботи є розробка мобільної пошукової системи на базі Android для пошуку медичних препаратів та найближчих аптек, яка б враховувала переваги існуючих програм і була позбавлена їх недоліків.

В якості завдання було поставлено такі задачі:

- провести аналіз існуючих на ринку додатків зі схожим функціоналом;
- сформулювати вимоги до розроблюваного програмного забезпечення;
- розробити технічне завдання;
- вибрати засоби реалізації програмного продукту;
- реалізувати та впровадити програмний продукт.

Результатом даної роботи буде пошукової системи на базі Android для пошуку медичних препаратів та найближчих аптек, яка б враховувала переваги існуючих програм і була позбавлена їх недоліків.

1 СУЧАСНИЙ СТАН МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ПОШУКУ ЛІКАРСЬКИХ ПРЕПАРАТІВ

Для аналізу було обрано 21 мобільний безкоштовний додаток для пошуку лікарських препаратів в аптеках [1]. З усіх представлених програм 11 пропонують пошук лікарських препаратів в інших країнах (Росія – 9, Білорусія – 2, Індія – 1, Кіпр – 1, Буркіна-Фасо – 1). Частина мобільних додатків є спеціальними медичними додатками (пропонують інструкції по застосуванню лікарських препаратів, аналоги ліків, з різною медичною інформацією, яка не належить до пошуку ліків в аптеках – 4). Таким чином, на даний час існує для жителів України всього три безкоштовних мобільних додатка.

1.1 Огляд програмних продуктів схожих за призначенням та функціями

Провівши дослідження на наявність існуючих систем зі схожим функціоналом для розгляду було обрано 3 додатки, враховуючи максимальну схожість призначення систем та функціональності із програмним забезпеченням, що розробляється. В огляді наведено основні можливості ПЗ та приклади із зображенням їхнього інтерфейсу.

1.1.1 Опис програми “Geo Apteka”

У додатку представлена інформація про наявність і ціни на ліки в більш ніж 1500 аптек в Києві, Львові, Дніпропетровську, Харкові, Одесі, Миколаєві та ін. Містах України. Сервіс GeoApteka пропонує пошук препаратів в найближчих до вас аптеках, підбір ціни і кількості. Також доступні телефони і адреси аптек, маршрути руху.

Основний функціонал додатку “Geo Apteka” включає:

- пошук препаратів у найбільших містах України;
- перегляд на карті трьох найближчих аптек;
- перегляд адреси аптеки та можливість зателефонувати у відділення;
- прокладання маршруту до вибраної аптеки;
- перегляд інструкції до вибраного препарату;
- налаштування критеріїв пошуку по дозуванню, формі випуску.

На рис. 1.1 наведено приклад графічного інтерфейсу користувача. Щоб почати пошук аптек у верхній частині екрану потрібно ввести назву. Після цього буде відкрита карта з найближчими аптеками. Можливо переглянути знайдені результати в списку де показано назву та адресу аптеки, ціну та кількість препарату [2].

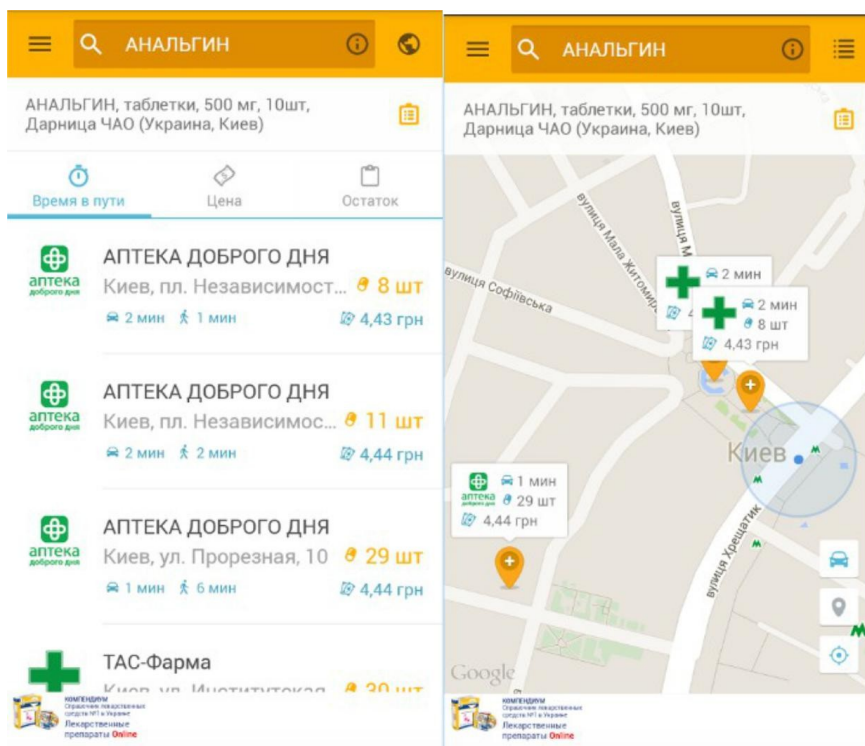


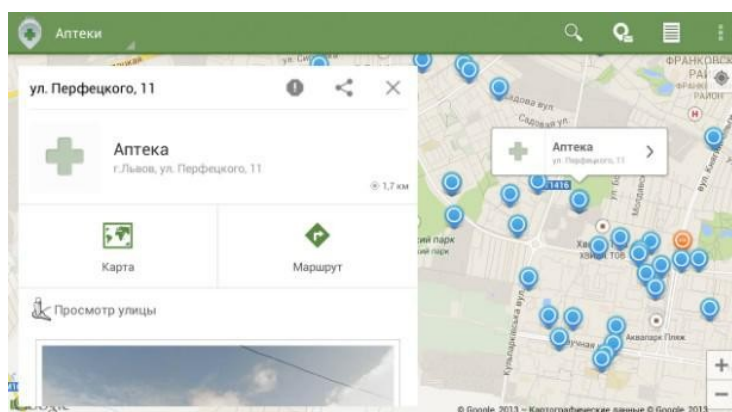
Рисунок 1.1 – Приклад інтерфейсу додатку Geo Apteka

1.1.2 Опис програми “Аптеки України”

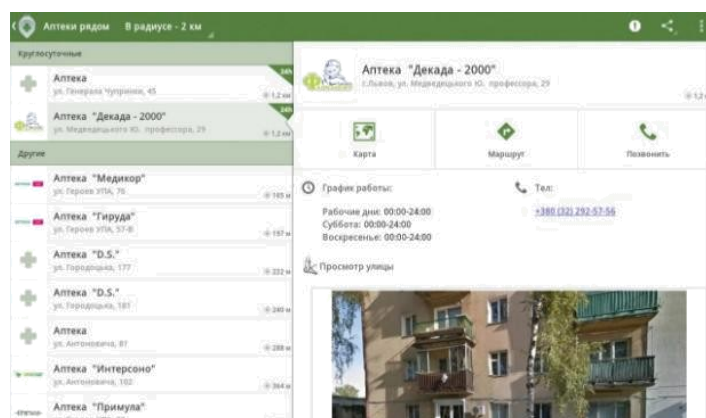
В додатку представлено інформаційний опис (форма випуску) ліків, і дані про їх наявність та середню ціну в аптеках мережі "Біокон" (Київ), "DS", "Аптека нашого міста" (Львів), "Конекс" (Вінниця), "Медпрепарати" (Миколаїв, Херсон). Програма дозволяє:

- переглянути список аптек, розташованих біля вашого місцезнаходження (в діапазоні від 1 до 10 кілометрів);
- прокласти зручний маршрут до обраної аптеці (використовуючи карти);
- переглянути місцезнаходження аптеки на карті;
- зателефонувати в аптеку за вказаним номером;
- дізнатися про наявність лікарських засобів та їх середньої вартості (в аптеках міст Києва, Львова, Вінниці, Миколаєва, Херсона та Севастополя).

На рис. 1.2 (а,б) наведено приклад графічного інтерфейсу користувача. У верхній частині екрану вводиться назва препарату та задається радіус пошуку аптек. В головному вікні програми можна переглянути результати пошуку на карті та у вигляді списку з детальною інформацією про аптеку. До вибраної аптеки можна прокласти маршрут, зателефонувати, переглянути графік роботи та адресу [3].



а



б

Рисунок 1.2 – Приклад інтерфейсу додатку “Аптеки України”
а) пошук на карті б) результат пошуку

1.1.3 Опис програми “Аптечний пункт”

Програма призначена для пошуку лікарських препаратів в аптечних пунктах. Як і в інших додатках можна переглянути результати пошуку на карті та в списку. Є можливість пошуку аналогів якщо бажаних ліків не знайдено. Також в програмі є функціонал прокладання маршруту до вибраної аптеки транспортом або пішки. Зручною є можливість зберігати інструкції препаратів для подальшого їх перегляду при необхідності (рис. 1.3).

Програма “Аптечний пункт” дозволяє:

- переглянути актуальні ціни на ліки в аптеках;
- порівняння ціни на ліки;
- знайти мінімальні ціни в найближчих аптеках;
- довідник ліків;
- переглянути інструкції до ліків (спосіб застосування, дозування);
- провести пошук аналогів до вибраного препарату;
- можливість зберігати інструкції.

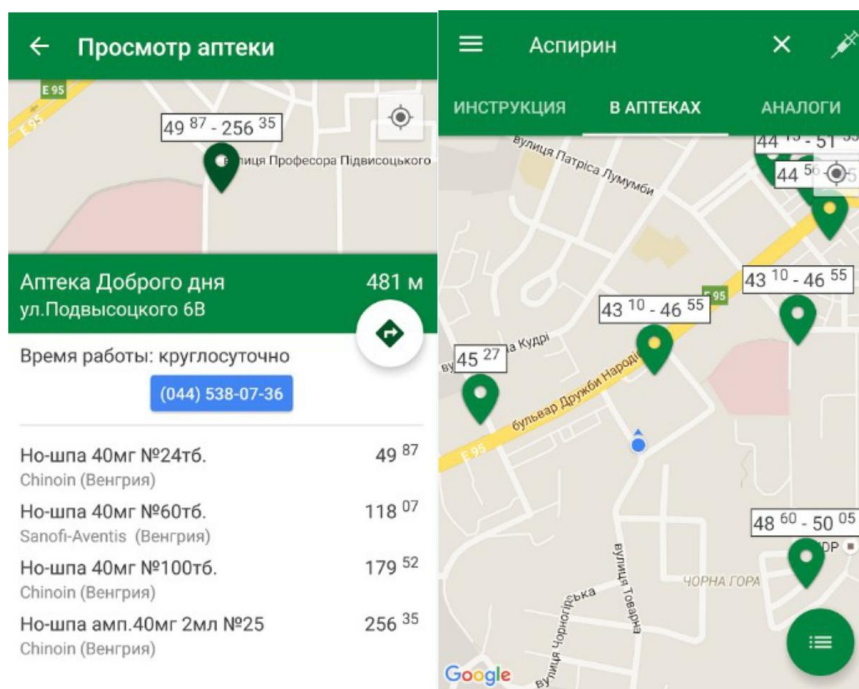


Рисунок 1.3 – Приклад інтерфейсу додатку “Tabletki”

При пошуку препаратів відкривається екран перегляду карти. При кліку на певну аптеку можна переглянути інформацію по препаратах, зателефонувати в аптеку, побудувати маршрут за адресою. На головному екрані доступні меню аналогів, де можна знайти препарати тієї ж фармакологічної групи або відкрити інструкцію препарату [4].

1.2 Аналіз вимог до програмного продукту

Так як розробка нашого додатку відбувається на замовлення, основні вимоги до функціоналу висуваються компанією-замовником. До основних функцій мобільної системи пошуку медичних препаратів та аптекних мереж повинні входити:

- пошук препаратів по назві, можливість задання критеріїв пошуку по дозуванню, виробнику;
- перегляд знайдених аптек на карті з можливістю швидкого перегляду короткої інформації;
- перегляд усіх результатів пошуку у вигляді списку;
- переглянути детальну інформацію про аптеку: графік роботи, адресу, відстань, телефон;
- переглянути детальну інформацію про препарат (інструкція).

Проте, проаналізувавши існуючі на ринку додатки зі схожими функціями, можна виділити ряд недоліків котрі також повинні бути враховані при розробці нашого додатку (табл. 1.1):

- відсутність пошуку аналогів. (важлива складова так як часто шуканого препарату немає в наявності, або ціна на нього завищена, в такому випадку пошук аналогічних препаратів може допомогти у виборі ліків);
- відсутність пошуку по штрих-коду товару (при наявності упаковки зі штрих-кодом шуканого товару можна його просканувати та швидко знайти препарат);
- відсутність можливості скласти список покупок. (необхідна функція яка заміняє записну книжку або інший додаток в який потрібно записати складений список покупок);
- обмежена кількість аптек та несвоєчасне оновлення бази даних (більшість додатків націлені на конкретні мережі та надають пошук по обмеженій базі даних або дані оновлюються несвоєчасно)

Таблиця 1.1 – Порівняння функціоналу існуючих додатків

Додатки/ Функціонал	ОптімаФарм (розроблюва не ПП)	GeoApteka	Аптеки України	Tabletki
Пошук аналогів	+	-	-	+
Пошук препарату по штрих коду	+	-	-	-
Можливість скласти список покупок	+	-	-	-
Своєчасне оновлення бази даних, актуальні дані	+	+	-	+
Побудувати маршрут до вибраної аптеки	+	-	+	-
Можливість вибрати виробника та дозування	+	+	-	-

2 ПРОГРАМНІ ТЕХНОЛОГІЇ, ЯКІ ВИКОРИСТОВУВАЛИСЯ ПРИ РОЗРОБЦІ ПРОЕКТУ

Програми для Android є програмами в нестандартному байт-кодi для віртуальної машини [Dalvik](#), а починаючи з 5 версії – Android Runtime (ART). Google пропонує для вільного завантаження інструментарій для розробки, який призначений для x86-машин під операційними системами [Linux](#), Mac OS X, Windows XP, Windows Vista та Windows 7. Для розробки потрібен [Java Development Kit 5](#) або новіший [5].

2.1 Коротке описання можливих інструментів для розробки

Розробку [застосунків](#) для Android можна вести мовою [Java](#) (не нижче Java 1.5). Офіційним середовищем розробки є [Android Studio](#), представлено компанією Google в 2013. Крім цього існує плагін для [Eclipse](#) – «Android Development Tools» (ADT), призначений для Eclipse версій 3.3-3.7. Для [IntelliJ IDEA](#) також існує плагін, який полегшує розробку Android-[застосунків](#).

Крім того в 2009 році на застосунок до ADT був опублікований Android Native Development Kit (NDK), пакет інструментаріїв і бібліотек дозволяє вести розробку застосунків мовою C/C++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості [6, 7].

2.2 Мова Java

Java – об'єктно-орієнтована мова програмування, що розробляється компанією Oracle (Sun Microsystems) з 1991 року і офіційно випущений 23 травня 1995 року. Програми на Java можуть бути трансльовані в байт-код, що

виконуються на віртуальній java-машині (JVM) – програмою, що обробляє байт-код і передає інструкції обладнанню, як інтерпретатор, але з тією відмінністю, що байт-код, на відміну від тексту, обробляється значно швидше [8].

Мова Java зародився як частина проекту створення передового програмного забезпечення для різних побутових приладів. Реалізацію проекту було розпочато на мові C++, але незабаром виникло ряд проблем, найкращим засобом боротьби з якими була зміна самого інструменту - мови програмування. Стало очевидним, що необхідна платформи незалежна мова програмування, що дозволяє створювати програми, які не доводилося б компілювати окремо для кожної архітектури і можна було б використовувати на різних процесорах під різними операційними системами.

Мова Java потрібна була для створення інтерактивних продуктів для мережі Internet. Фактично, більшість архітектурних рішень, прийнятих при створенні Java, було продиктовано бажанням надати синтаксис, схожий з C і C++. У Java використовуються практично ідентичні угоди для оголошення змінних, передачі параметрів, операторів і для управління потоком виконання коду [9].

Проте Java свідомо уникає арифметики з покажчиками й іншими ненадійними елементами, якими буває C++, тому, розробляючи на ній додатки, ми позбавляємося багатьох проблем, звичайних при створенні програмного забезпечення. Три ключові елементи об'єдналися в технології мови Java:

- Java надає для широкого використання свої аплети (applets) – невеликі, надійні, динамічні, які не залежать від платформи активні мережеві додатки, що вбудовуються в сторінки Web. Аплети Java можуть налаштовуватися і поширюватися споживачам з такою ж легкістю, як будь-які документи HTML;
- Java показує силу об'єктно-орієнтованої розробки додатків, поєднуючи простий;
- синтаксис з надійним і зручним в роботі середовищем розробки. Це дозволяє широкому колу програмістів швидко створювати нові програми;
- Java надає програмісту багатий набір класів об'єктів для ясного абстрагування багатьох системних функцій, використовуваних при роботі з вікнами, мережею і для введення-виведення. Ключова риса цих класів полягає в тому, що вони забезпечують створення незалежних від використовуваної платформи абстракцій для широкого спектра системних інтерфейсів [10].

2.3 Опис середовища Android Studio

Середовище розробки Android Studio адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, [ноутбуки](#), годинники, окуляри тощо). Крім можливостей, присутніх в [IntelliJ IDEA](#), в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів [безперервної інтеграції](#).

Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає

зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів.

До складу також включені пристосовані під особливості платформи Android розширені інструменти [рефакторингу](#), перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови [11, 12].

2.4 Карти та їх застосування

При розробці додатку було вирішено використовувати Google Maps для відображення знайдених аптек на карті. Сервіс є найзручнішим та найшвидшим способом підключення картографічних функції до будь якого Android додатку. Google Maps може працювати з або без приймача GPS. Функція працює за допомогою GPS/A-GPS, якщо такі є, доповнені шляхом визначення найближчих бездротових мереж і стільникових станцій. Програмне забезпечення відшукує місце розташування ділянки осередка, використовуючи базу даних відомих бездротових мереж. За допомогою тріангуляції різних рівнів сигналів від різних передавачів і потім, використовуючи їх властивості розташування (отримане з онлайн бази даних стільникових станцій), визначається поточне місце розташування користувача. Бездротовий мережевий метод розташування розраховується шляхом виявлення прилеглих WiFi точок доступу і використання їх властивостей позиціонування (отримане з онлайн бази даних Wi-Fi, таким же чином, як база даних стільникових станцій) для подальшого виявлення місця розташування. Порядок, в якому вони мають пріоритет:

- на основі GPS;
- на основі WLAN-, WiFi-сервісів;
- на основі стільникового зв'язку [13].

2.5 Графічний інтерфейс мовою XML

В Android є зручний довідник XML-елементів для класів View і їх підкласів. Платформа надає вам гнучкість при використанні оголошення

призначеного для користувача інтерфейсу програми та його управління. Наприклад, ви можете оголосити в XML макети за замовчуванням, включаючи елементи екрану, які будуть відображатися в макетах, і їх властивості. Потім ви можете додати в додаток код, який дозволяє змінювати стан об'єктів на екрані (включаючи оголошені в XML) під час виконання.

В Android Studio передбачена функція попереднього перегляду створеного вами файлу XML – досить відкрити файл XML і вибрати вкладку Layout (Макет).

Для налагодження макетів можна скористатися інструментом Hierarchy Viewer – з його допомогою можна переглянути значення властивостей, рамки з індикаторами заповнення або полів, а також повністю промальовані макети прямо під час налагодження програми в емуляторі або на пристрої.

За допомогою інструменту layoutopt можна швидко проаналізувати макети і їх ієрархії на предмет низької ефективності чи інших проблем. Перевага оголошення призначеного для користувача інтерфейсу в файлі XML полягає в тому, що таким чином ви можете більш ефективно відокремити інтерфейс свого додатку від коду, який управляє його поведінкою. Описи призначеного для користувача інтерфейсу знаходяться за межами коду вашої програми. Це означає, що ви можете змінювати або адаптувати інтерфейс без необхідності вносити правки в вихідний код і повторно компілювати його. Наприклад, можна створити різні файли XML макета для екранів різних розмірів і різних орієнтацій екрану, а також для різних мов. Крім того, оголошення макета в XML спрощує візуалізацію структури призначеного для користувача інтерфейсу [14].

2.6 Опис серверної взаємодії

Так як розроблювана система буде використовувати дані які дуже часто оновлюватися то ці дані не можна зберігати локально на пристрої, а завантажувати їх із серверу, тобто використовувати клієнт-серверну архітектуру. Архітектура клієнт-сервер є одним із [архітектурних шаблонів програмного забезпечення](#) та є домінуючою концепцією у створенні розподілених мережних [застосунків](#) і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір [серверів](#), які надають інформацію або інші послуги програмам, які звертаються до них;
- набір [клієнтів](#), які використовують сервіси, що надаються серверами;
- [мережа](#), яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до

серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

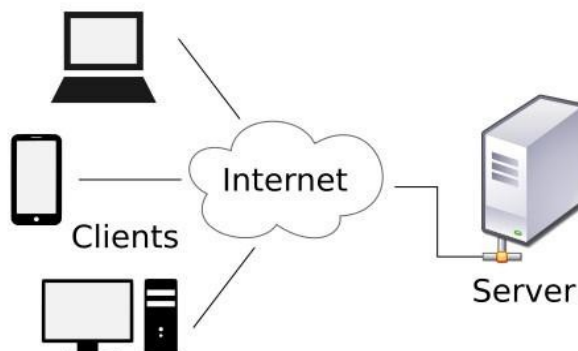


Рисунок 2.1 – Клієнт-серверна архітектура

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна виокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку [застосунку](#) і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних [модулів](#) – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка [застосунку](#) та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта.

Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, та як у нашому випадку – мобільні телефони [15].

В якості формату обміну даними було вибрано JSON як найбільш економний, зручний та використовуваний формат.

JSON це текстовий [формат](#) обміну [даними](#) між [комп'ютерами](#). JSON базується на тексті, і може бути з легкістю прочитаним людиною. Формат дозволяє описувати [об'єкти](#) та [структури даних](#). Цей формат головним чином використовується для передачі структурованої інформації через [мережу](#) (завдяки процесу, що називають [серіалізацією](#)) [16].

2.7 Система контролю версій

Для зручної розробки додатку було використано систему контролю версій Git. Система контролю дозволяє зберігати попередні версії файлів та завантажувати їх за потребою. Вона зберігає повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Місце зберігання даних файлів називають репозиторієм. В середині кожного з репозиторіїв можуть бути створені паралельні лінії розробки – гілки.

Гілки зазвичай використовують для зберігання експериментальних, незавершених(alpha, beta) та повністю робочих версій проекту(final). Більшість систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

Використання системи контролю версії є необхідним для роботи над великими проектами. Системи контролю версії надають ряд додаткових можливостей:

- можливість створення різних варіантів одного документу;
- документування всіх змін (коли ким було змінено/додано, хто який рядок змінив);
- реалізує функцію контролю доступу користувачів до файлів. Є можливість його обмеження;
- дозволяє створювати документацію проекту з поетапним записом змін в залежності від версії;
- дозволяє давати пояснення до змін та документувати їх.

Існують два основні типи систем керування версіями: з централізованим сховищем та розподіленим.

Централізована система контролю версії (клієнт-серверна) – система, дані в якій зберігаються в єдиному «серверному» сховищі. Весь обмін файлами відбувається з використанням центрального сервера. Є можливість створення та роботи з локальними репозиторіями (робочими копіями).

Переваги:

- загальна нумерація версій;

- дані знаходяться на одному сервері;
- можлива реалізація функції блокування файлів;
- можливість керування доступом до файлів.

Недоліки:

- потреба в мережевому з'єднанні для оновлення робочої копії чи збереження змін.

До таких систем відносять Subversion, Team Foundation Server.

Розподілена система контролю версії (англ. Distributed Version Control System, DVCS) – система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів [17]. Переваги:

- кожний з розробників працює зі своїм власним репозиторієм;
- рішення щодо злиття гілок приймається керівником проекту;
- немає потреби в мережевому з'єднанні.

Недоліки:

- немає можливості контролю доступу до файлів;
- відсутня загальна нумерація версії файла;
- значно більша кількість необхідного дискового простору;
- немає можливості блокування файлів;

До таких систем відносять Git, Mercurial, SVK, Monotone, Codeville, Bit-Keeper.

3 ТЕХНІЧНЕ ЗАВДАННЯ ТА ПРОЕКТУВАННЯ

3.1 Назва та призначення.

Назва програмного продукту: «Розробка програми «Аптека»».

ПП дає можливість зручно, за допомогою мобільного телефону, знайти найближчу аптеку в якій наявний потрібний медичний препарат або просто переглянути всі найближчі аптеки.

По препарату доступна така інформація як назва, інструкція, виробник, діапазон цін.

З інформації про аптеку доступні назва, адреса, контактні дані, режим роботи.

За допомогою функціоналу програми також можна налаштувати критерії пошуку препаратів за виробником, назвою, дозуванням, а також, за наявності штрих-коду, знайти аналогічний товар. Крім цього є можливість скласти список покупок та переглянути аналоги препаратів.

Користувачами програми можуть бути всі хто час від часу має необхідність у пошуку медичних препаратів.

3.2 Вимоги до проекту

Всі вимоги до програмного продукту було сформовано перед початком реалізації. А саме, необхідно провести аналіз предметної області, проаналізувати існуючі система, визначити основні вимоги до програмного продукту та його функції і змодельовати їх, створити програмний продукт.

3.2.1 Вимоги до вхідних даних

Коректне введення даних по препарату для повноцінного звернення до серверу: назви препарату, виробника, дозування, дійсного штрих-коду при пошуку за цим критерієм.

3.2.2 Вимоги до функціональних характеристик

ПП повинен забезпечувати можливість виконання перерахованих нижче функцій:

- 1) пошуку медикаментів;
- 2) пошуку аптечних мереж;
- 3) перегляд результатів на карті та у вигляді списку;
- 4) пошуку препаратів по назві, виробнику, штрих-коду;
- 5) перегляду найближчих аптечних пунктів з потрібним товаром;
- 6) перегляду детальної інформації по товару;
- 7) перегляд інформації про аптеку;
- 8) можливість зателефонувати в аптеку;

- 9) формування списку покупок;
- 10) пошук аналогів.

3.2.3 Вимоги до забезпечення надійного функціонування ПП

Надійне (стійке) функціонування ПП має бути забезпечене виконанням користувачем сукупності організаційно-технічних заходів, перелік яких наведено нижче:

- організацією безперебійного живлення технічних засобів;
- використанням ліцензійного програмного забезпечення;
- відсутністю сторонніх або шкідливих програм, що можуть привести до непрацездатності даної програми.

3.2.4 Час відновлення після відмови

Час відновлення після відмови, викликаного збоєм електроживлення технічних засобів (іншими зовнішніми чинниками), не фатальним збоєм (не крахом) операційної системи, не повинно перевищувати 10-ти хвилин за умови дотримання умов експлуатації технічних і програмних засобів.

Час відновлення після відмови, викликаного несправністю технічних засобів, фатальним збоєм (крахом) операційної системи, не повинно перевищувати часу, необхідного для усунення несправностей технічних засобів і переустановлення програмних засобів.

3.2.5 Вимоги до засобів реалізації

Програмний продукт розробляється в середовищі Android Studio не нижче версії 1.5. Усі елементи графічного інтерфейсу розроблені у редакторі макетів за допомогою мови XML.

Програмний код написаний на мові програмування Java з системою автоматичного збирання Gradle.

Для відображення результатів на карті використовується сервіс Google Maps.

Взаємодія з сервером бази даних відбувається через http запити, дані передаються у форматі JSON.

3.2.6 Вимоги до зберігання даних

Програмне забезпечення розробляється як stateless мобільний клієнт для обміну інформацією з сервером. Це означає, що фактично дані в додатку не зберігаються для уникнення оперування застарілими даними. Збереження інформації про препарати відбувається тільки для організації функціоналу списку покупок.

3.2.7 Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційних структур і методів розв'язання, програмних засобів та захисту інформації:

– системні програмні засоби, що використовуються програмою, повинні бути представлені ліцензійною локалізованою версією ОС Android не нижче версії 4.1.

3.2.8 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- 1) документацію етапу системного аналізу та аналізу вимог;
- 2) документацію етапу проектування;
- 3) документацію етапу кодування;
- 4) документацію етапу тестування;
- 5) посібник користувача;
- 6) висновки по роботі.

3.3 Умови до експлуатації та програмного забезпечення

Сукупність тверджень щодо атрибутів, властивостей або якостей програмної системи, що підлягає реалізації. Створюються в процесі розробки вимог до програмного забезпечення, у результаті аналізу вимог.

Вимоги можуть виражатися у вигляді текстових тверджень і графічних моделей.

У класичному технічному підході сукупність вимог використовується на стадії проектування ПЗ.

Етапу розробки вимог, можливо, передувало техніко-економічне обґрунтування, або концептуальна фаза аналізу проекту.

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги до кваліфікації та чисельності персоналу

Для експлуатації мобільного додатку від адміністратора не повинно вимагатися спеціальних технічних навичок, знання технологій або програмних продуктів, за винятком загальних навичок роботи з мобільним телефоном та програм, що використовуються в повсякденному житті.

3.3.3 Економічні переваги розробки

Економічні переваги до уваги не беруться.

3.4 Стадії та етапи розробки

3.4.1 Стадії розробки

Розробка повинна бути проведена в наступні стадії:

- 1) системний аналіз та аналіз вимог;

- 2) проектування;
- 3) програмування;
- 4) тестування і налагодження роботи ПП;
- 5) розробка документації;

3.4.2 Етапи розробки

На стадії системного аналізу та аналізу вимог повинно бути сформоване технічне завдання, діаграма Ганта та мережевий графік.

На стадії проектування повинні бути виконані перераховані нижче етапи робіт:

- розробка Ієрархічної структури та розрахунок нев'язки;
- розробка ERD, SADT, DFD діаграм;
- розробка блок-схем основної частини ПП та усіх її допоміжних функцій;
- розробка USE-CASE діаграми за допомогою мови UML.

На стадії програмування повинні бути виконані перераховані нижче етапи робіт:

- реалізація інтерфейсу ПП в Android Studio;
- реалізація модулів ПП в середовищі Android Studio.

На стадії тестування і налагодження роботи ПП проводиться випробування системи на наявність помилок, достовірність роботи всіх підсистем і системи в цілому, оцінка простоти та інформативності інтерфейсу користувача.

На стадії розробки документації повинні бути виконані перераховані нижче етапи робіт:

- оформлення документації, яка перерахована на етапі системного аналізу та аналізу вимог;
- оформлення усіх результатів діяльності (діаграм та схем) на етапі проектування;
- оформлення результатів етапів кодування та тестування;
- оформлення посібника користувача.

3.5 Проектування

Проектування - це комплекс робіт який складається з пошуку, досліджень, розрахунків та розрахування з метою отримання опису достатнього для створення нового об'єкту або виробу, його реконструкції, модернізації, що відповідає заданим вимогам.

У будь-якому випадку проектування починається при наявності завдання на проектування, що відбиває потреби суспільства в одержанні деякого технічного виробу. Завдання представляється у виді тих чи інших документів і є

вихідним (первинним) описом об'єкта. Результатом проектування, як правило, служить повний комплект документації, що містить достатні дані для виготовлення об'єкта в заданих умовах. Ця документація являє собою остаточний опис об'єкта.

3.5.1 Діаграма функціонального моделювання (SADT)

Методологія SADT являє собою сукупність методів, правил і процедур, призначених для побудови функціональної моделі об'єкта будь-якої предметної області. Функціональна модель SADT відображає функціональну структуру об'єкта, тобто вироблені ним дії і зв'язки між цими діями. Основні елементи цієї методології ґрунтуються на наступних концепціях:

- 1) графічне представлення блокового моделювання. Графіка блоків і дуг SADT-діаграми відображає функцію у вигляді блоку, а інтерфейси входу/виходу представляються дугами, відповідно входять у блок і виходять з нього. Взаємодія блоків один з одним описуються за допомогою інтерфейсних дуг, що виражають "обмеження", які в свою чергу визначають, коли і яким чином функції виконуються й управляються;
- 2) строгість і точність. Виконання правил SADT вимагає достатньої строгості і точності, без внесення в той же час надмірних обмежень на дії аналітика. Правила SADT включають:
 - обмеження кількості блоків на кожному рівні декомпозиції (як правило 3-6 блоків);
 - зв'язність діаграм (номери блоків);
 - унікальність міток і найменувань (відсутність повторюваних імен);
 - синтаксичні правила для графіки (блоків і дуг);
 - поділ входів та управлінь (правило визначення ролі даних);
 - відділення організації від функції, тобто виключення впливу організаційної структури на функціональну модель.

Методологія SADT може використовуватися для моделювання широкого кола систем і визначення вимог і функцій, а потім для розробки системи, яка задовольняє цим вимогам і реалізує ці функції. Для вже існуючих систем SADT може бути використана для аналізу функцій, виконуваних системою [19].

На рис. 3.1 представлено вхідні дані (вимоги та дизайн), ресурси (сторонні бібліотеки, база даних, серверний програмний інтерфейс, відомості про аналогічні системи), фактори впливу (життєвий цикл програмного забезпечення, стандарти ГОСТ та ISO/IEC, шаблони проектування) а також результат розробки – мобільна пошукова система.

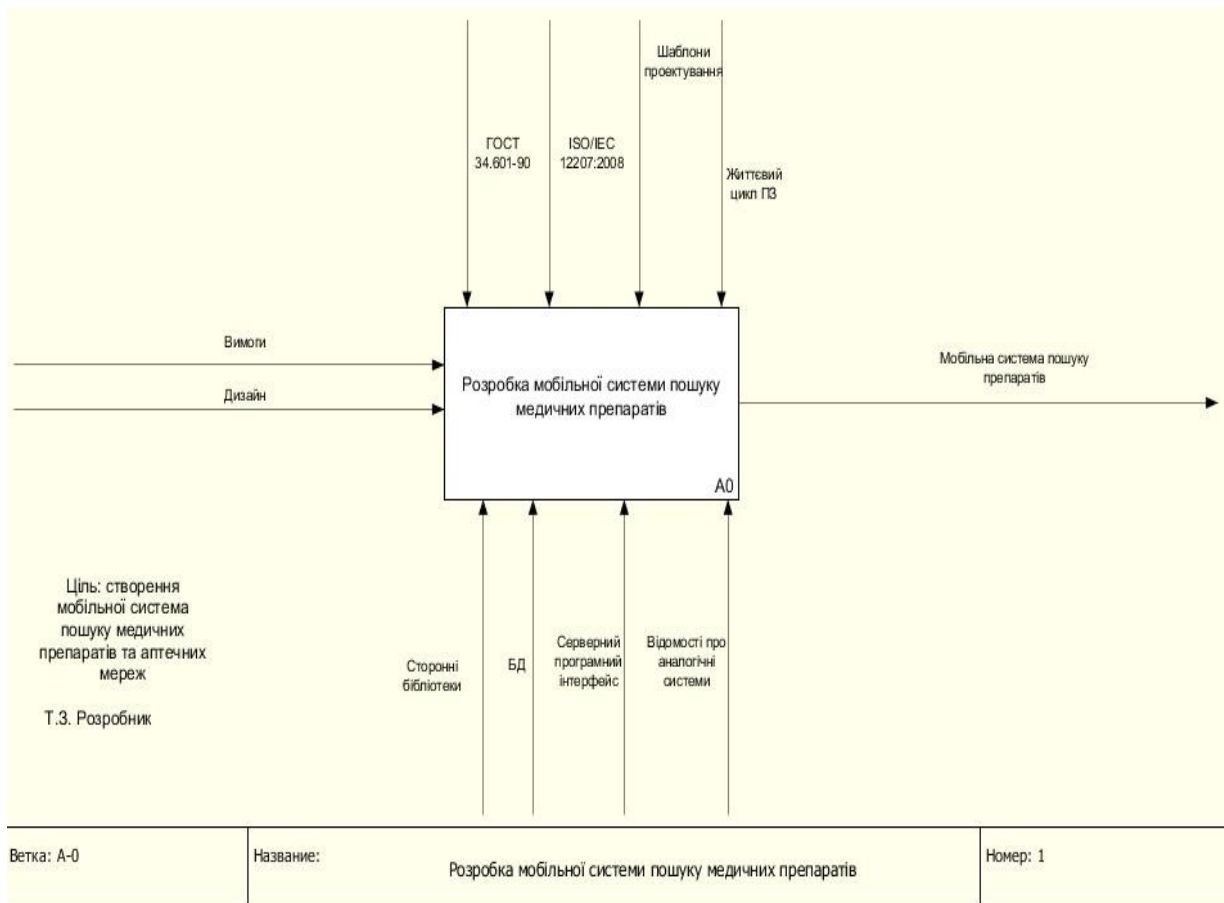


Рисунок 3.1 – Діаграма функціонального моделювання

3.5.2 Діаграма декомпозиції IDEF0

Діаграма першого рівня декомпозиції A0 (рис. 3.2), а також всі наступні діаграми декомпозиції, надають інтерфейс і обмеження (контекст) для дочірніх діаграм. Було вирішено розбити створення програмного продукту на наступні складові: аналіз (рис. 3.3), проектування (рис. 3.4), реалізація та тестування, що зображено на діаграмах.

Діаграма декомпозиції першого рівня показує, що сторонні ресурси та фактори впливу знаходять застосування на всіх стадіях реалізації проекту від аналізу до тестування. Вимоги до реалізації та дизайн найбільш важливі на ранніх етапах, тобто, при аналізі та проектуванні.

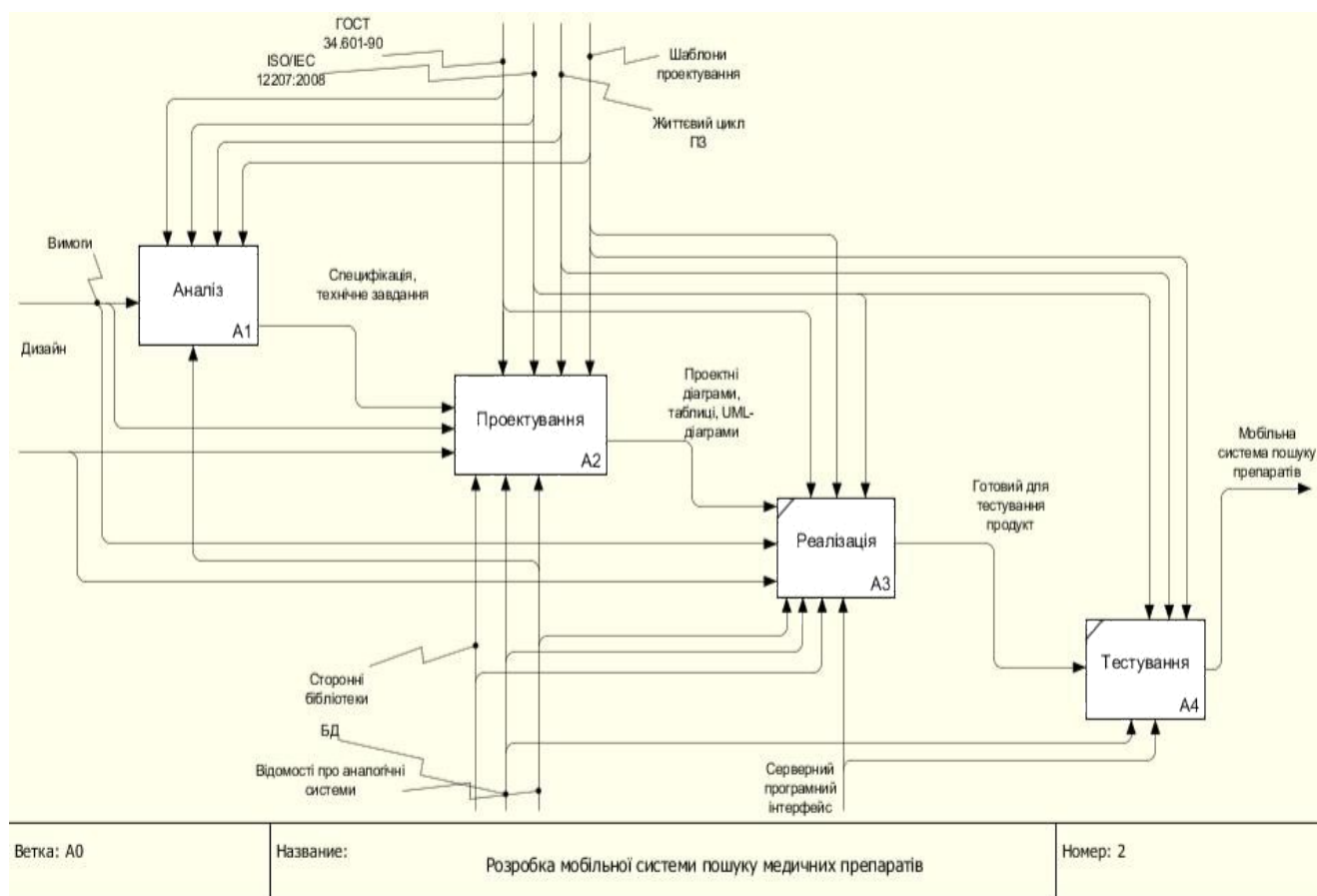


Рисунок 3.2 – Діаграма декомпозиції IDEF0

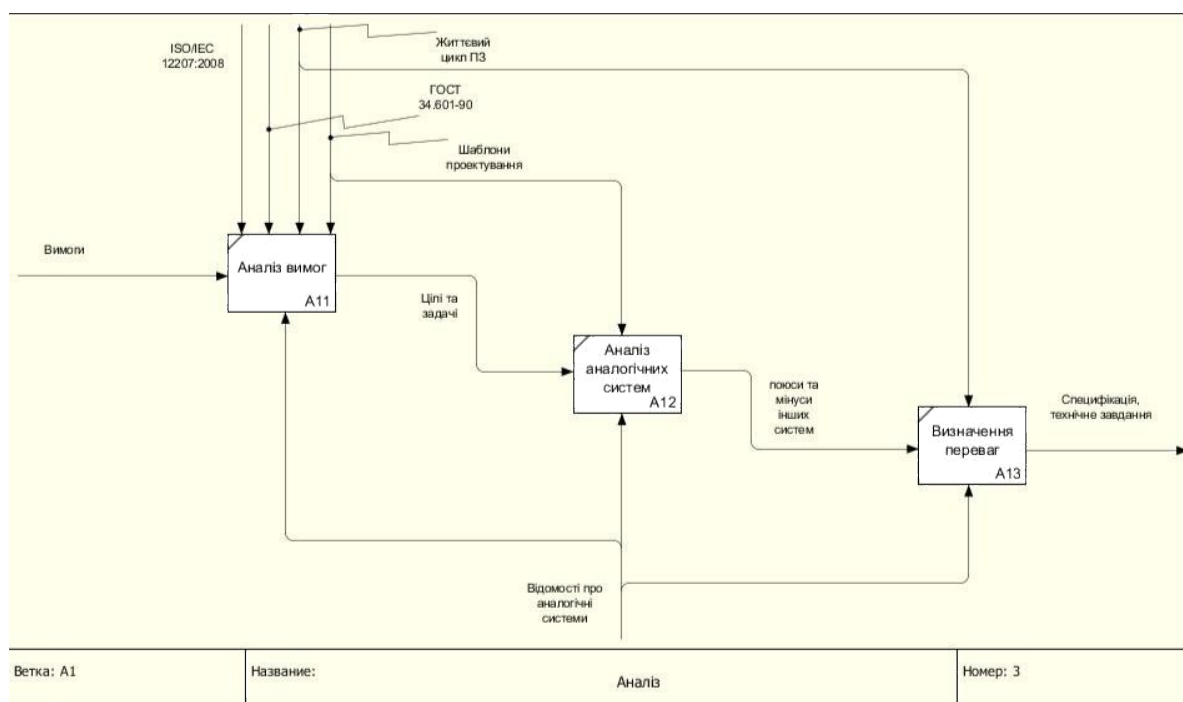


Рисунок 3.3 – Діаграма декомпозиції IDEF0 (аналіз)

Також з діаграми видно які проміжні ресурси ми отримаємо з кожного етапу та передаємо їх на вхід наступного. Таким чином після етапу аналізу ми отримаємо специфікацію та технічне завдання, що буде вхідними даними для проектування. Після проектування будемо мати проектні діаграми, таблиці, UML-діаграми. Після отримання цих складових можна приступати до реалізації програмної частини.

- Після цього готовий продукт проходить етап тестування на налагодження. Розглядаючи детально етап аналізу, його можна розділити на три складові:
- аналіз вимог – результатом є цілі та задачі;
 - аналіз схожих існуючих систем – це дасть їх зрозуміти переваги та недоліки;
 - врахування результатів попереднього етапу та визначення конкурентних переваг.

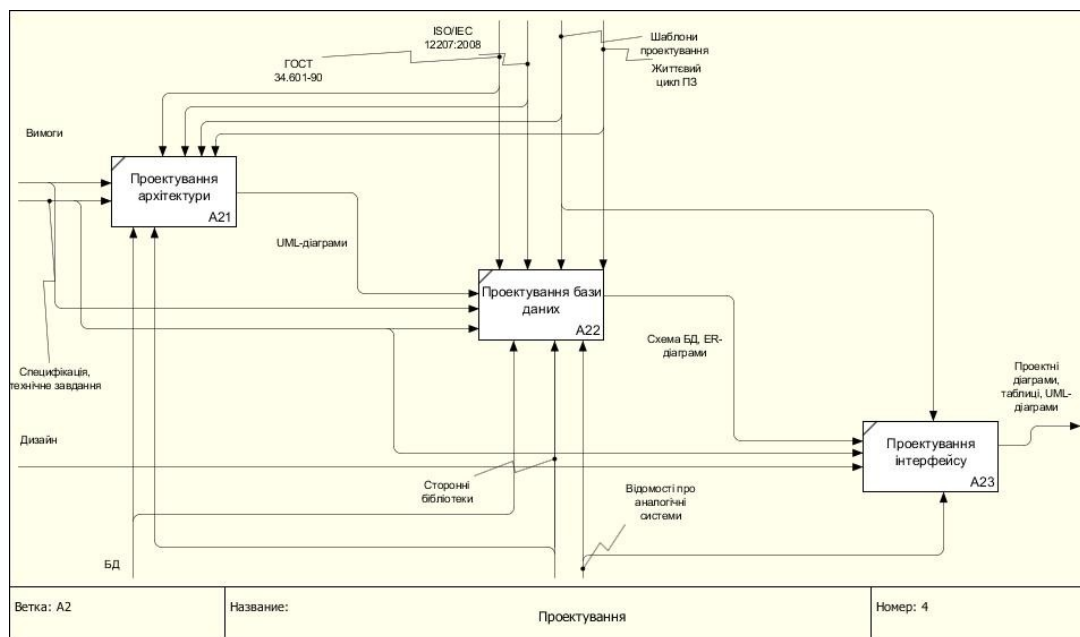


Рисунок 3.4 – Діаграма декомпозиції IDEF0 (проектування)

Розглядаючи детально етап проектування його можна поділити також на три частини: проектування архітектури, в результаті чого отримаємо UML-діаграми; проектування бази даних, що дасть схеми бази даних та ER-діаграми; проектування графічного вигляду програми – дозволить візуально представити функціонал розроблюваного додатку.

3.5.3 Діаграма дерева вузлів

Діаграма дерева вузлів (рис. 3.5) показує ієрархічну залежність робіт, що детально описані в попередніх розділах, але не взаємозв'язки між роботами. Це дає змогу схематично зобразити всі етапи робіт в одному місці, щоб оцінити об'єм виконуваних робіт.

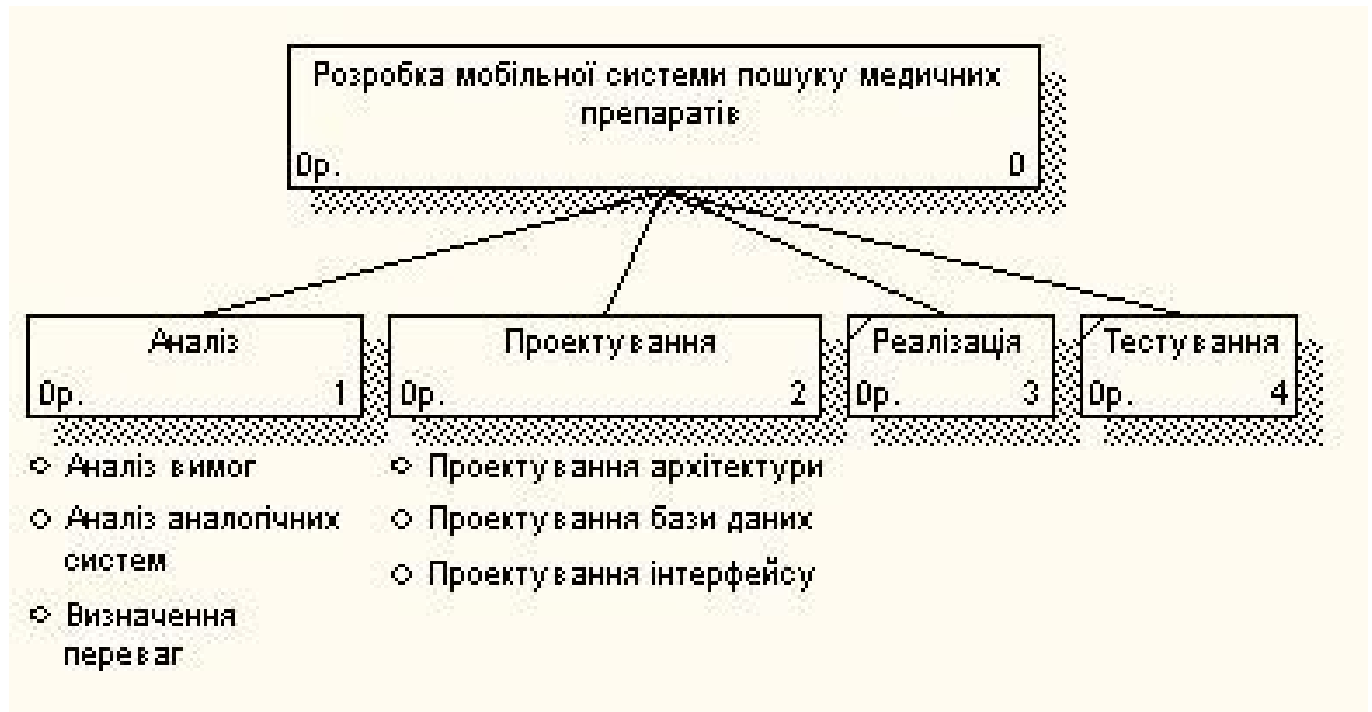


Рисунок 3.5 – Діаграма дерева вузлів

3.5.4 Схема бази даних

Як було сказано раніше, збереження даних у додатку необхідне для формування та збереження списку покупок. У базі даних будуть зберігатися медичні препарати. Схема бази даних в якій показано поля таблиці описана на рис. 3.6. Первинним ключем виступає унікальний ідентифікатор id [20].

3.5.5 Модель життєвого циклу

Модель життєвого циклу – структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання.

Препарат	
РК	<u>Id</u>
	назва
	виробник
	дозування
	максимальна ціна
	мінімальна ціна
	кількість в упаковці
	посилання на зображення

Рисунок 3.6 – Схема бази даних

Каскадна (або послідовна) модель. Передбачає строго послідовне в часі і однократне виконання всіх фаз проекту з детальним попереднім плануванням і визначеними вимогами.

Основною особливістю цієї моделі є розбиття всієї розробки на етапи. Перехід від одного етапу до іншого відбувається лише при умові повного завершення робіт на попередньому етапі. Кожен етап завершується випуском документації, достатньої для того, щоб розробка могла бути продовжена іншою командою розробників [21].

Поміж існуючих моделей ЖЦ (каскадна, гнучка, спіральна та еволюційна) було обрано каскадну модель. Оскільки це поетапне виконання визначених дій і може виконуватися одним розробником. Також в цій моделі значна увага приділяється інженерії вимог та власне проектуванню, що застраховує від вагомих помилок.

3.5.6 Побудова ієрархічної структури та розрахунок нев'язки

Побудовану ієрархічну структуру зображено на рис. 3.8. Розрахуємо нев'язку за допомогою MATLAB (рис. 3.7).

Як можна побачити, ми отримали дуже малу нев'язку, яка прагне до нуля. Це свідчить про добре сплановану систему, яку доцільно розробляти.

```

>> n = 8; %количество вершин полученного графа
E = 8; %количество ребер полученного графа
Et = n-1; %количество ребер дерева
Ec = n*(n-1)/2; %количество ребер полного графа
Nev = (E-Et)/(Ec-Et); %расчет невязки
Nev %вывод значений невязки

Nev =

    0.0476

```

Рисунок 3.7 – Розрахунок невідповідності



Рисунок 3.8 – Ієрархічна структура ПП

4 ОГЛЯД ГОТОВОГО ПРОГРАМНОГО ПРОДУКТУ

4.1 Встановлення програми

Установлення найпростіших програм може полягати в копіюванні їхніх файлів, але переважна більшість програм установлюються з використанням допоміжної програми-інсталятора, яка реєструє програму в середовищі ОС, запитує у користувача, які модулі програми встановлювати, створює ярлики.

Програма розроблена та може бути встановлена на мобільні телефони, що працюють на операційній системі Android версії не нижче 4.1. Встановлення програмного продукту може відбуватися у два способи:

- з магазину додатків Google Play;
- при наявності файлу програми.

Google Play – [магазин додатків](#) від компанії [Google](#), що дозволяє власникам пристроїв з мобільною операційною системою [Android](#) завантажувати і купувати різні додатки, книги, фільми і музику. При встановленні з Google Play потрібно ввести назву в графу пошуку, знайти потрібний додаток та встановити його [22].

При наявності файлу програми з розширенням apk можна також встановити програму. Для цього необхідно завантажити її на телефон, відкрити за допомогою будь якого файлового менеджера та відкрити apk файл, встановлення розпочнеться автоматично.

Програмний продукт є безкоштовним, не потребує реєстрації. Мобільний додаток має захист від несанкціонованого доступу до коду програми, за допомогою інструменту Proguard.

4.2 Опис програмного продукту

При відкритті програми на екрані можна побачити дві основні секції: “Карта” та “Аптеки” (рис. 4.1). Пункт “Карта” з’являється за замовчуванням і після визначення місця знаходження відображає найближчі аптеки в окрузі. Якщо щільність розміщення аптек велика то на карті вони об’єднуються в групи. При натисканні на будь яку аптеку буде показано невеликий баннер з назвою та режимом роботи.

При переході в пункт меню “Аптеки” будуть показані ті ж аптеки, що знаходяться поблизу, але у вигляді списку та можливістю детального перегляду. У списку для кожної знайденої аптеки можна переглянути: назву, адресу, відстань від вашого місця розташування, графік роботи, а також можливість зателефонувати до відділення.

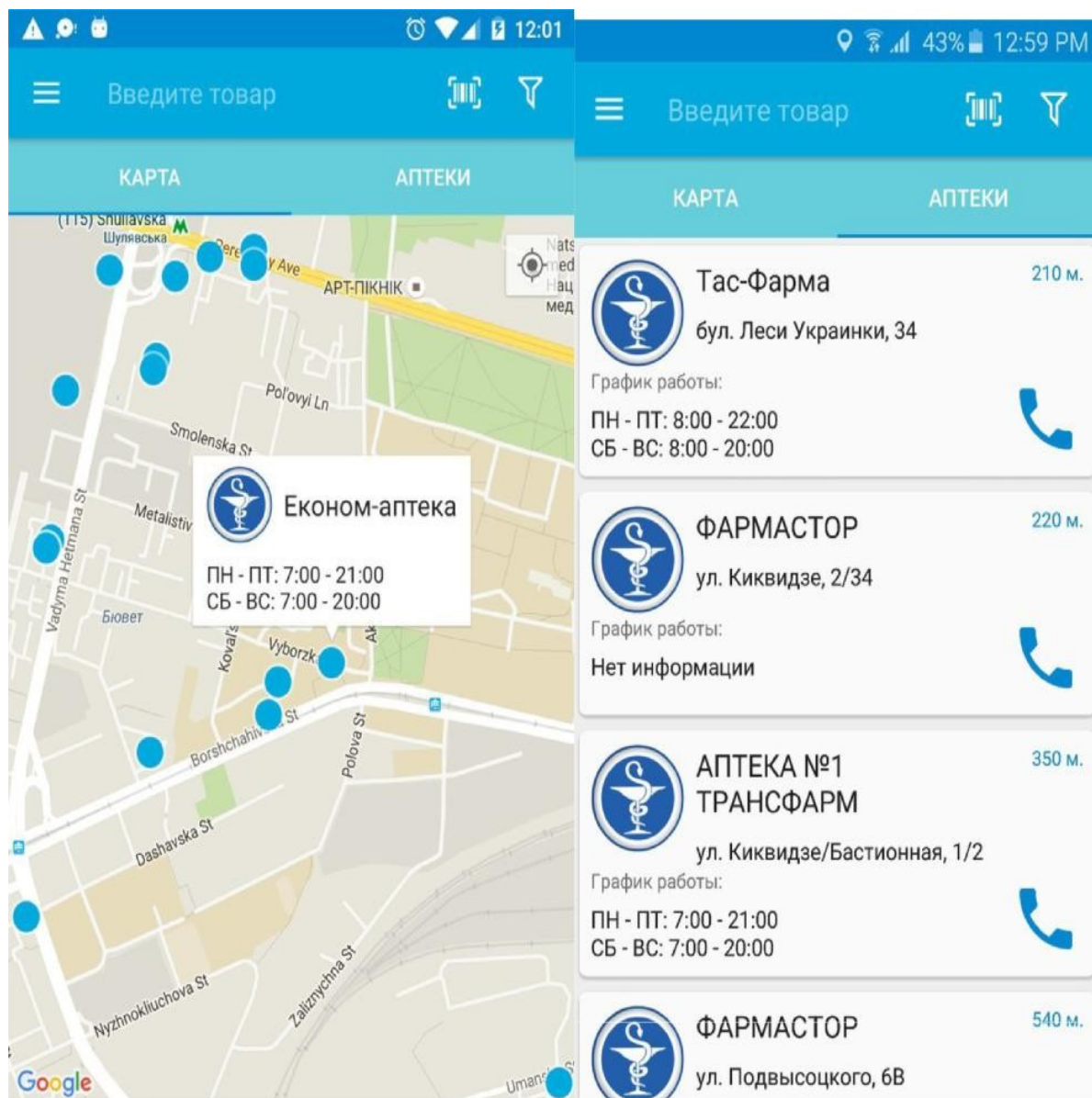


Рисунок 4.1 – Вигляд головного екрану

При натисканні в списку на аптеку відкриється екран з більш розгорнутою інформацією про неї (рис. 4.2). Крім вище перерахованої інформації можна дізнатися контактні дані власника чи завідуючого аптекою, а також час за який можна дістатися до адреси її розташування.

Для швидкого пошуку медичного препарату достатньо ввести його назву у верхній частині головного екрану. Після коректного введення декількох літер, програма запропонує декілька препаратів (рис. 4.3).

Після введення назви препарату буде проведено пошук та показано результати. Результатом буде список аптек, в яких наявний шуканий препарат, який можна переглянути на карті та у списку (рис. 4.1).

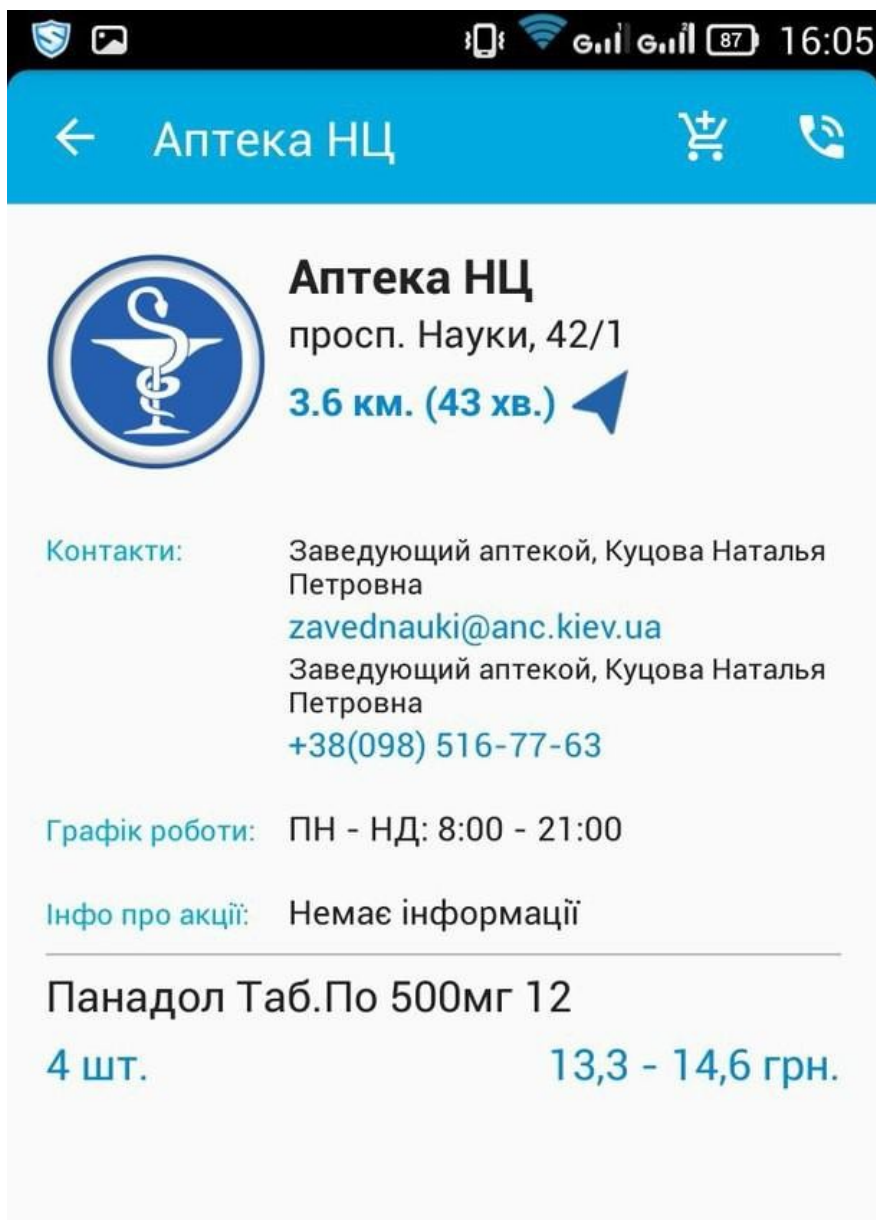


Рисунок 4.2 – Розгорнута інформацію про аптеку

Крім пошуку по назві можна здійснити пошук по штрих-коду товару. Для цього потрібно у верхній частині головного екрану натиснути на відповідну іконку. Після цього відкриється вікно сканування штрих-коду за допомогою камери телефону (рис. 4.4). Після успішного сканування потрібно підтвердити пошук натиснувши “ОК”. Результатом сканування, як і в інших випадках буде список аптек з шуканим препаратом.

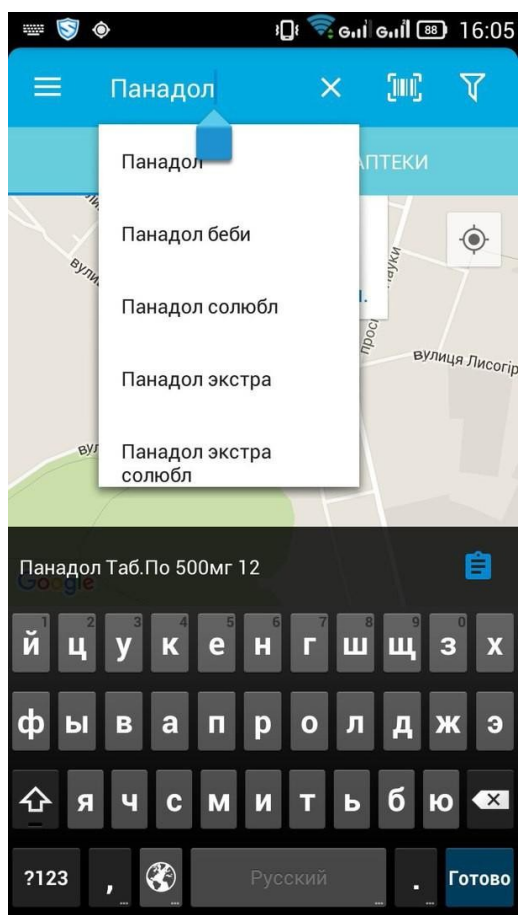


Рисунок 4.3 – Пошук препаратів на головному екрані

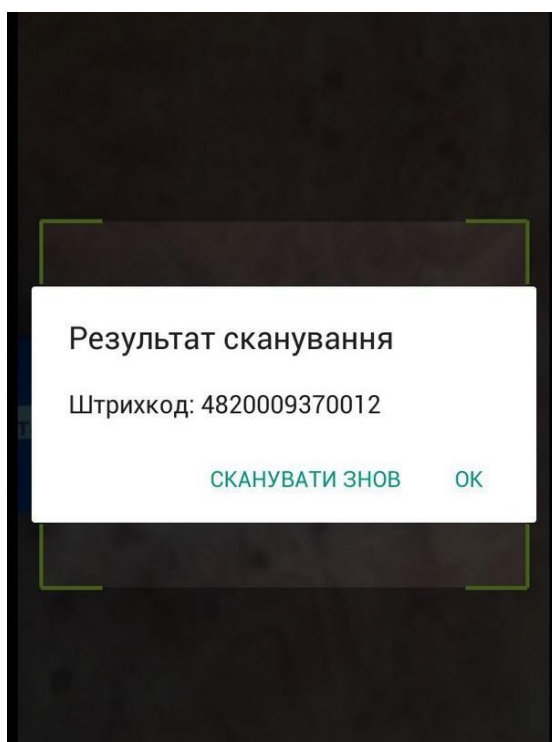


Рисунок 4.4 – Екран сканування штрих-коду

Бокове меню програми складається з вкладок “Аптеки”, “Товари”, “Список покупок”, “Додатково”, “Настройки”(рис. 4.5). Якщо натиснути у боковому меню програми на пункт “Товари”, відкривається екран, на якому можна провести пошук препаратів за назвою із зазначенням виробника, дозування (рис. 4.6).

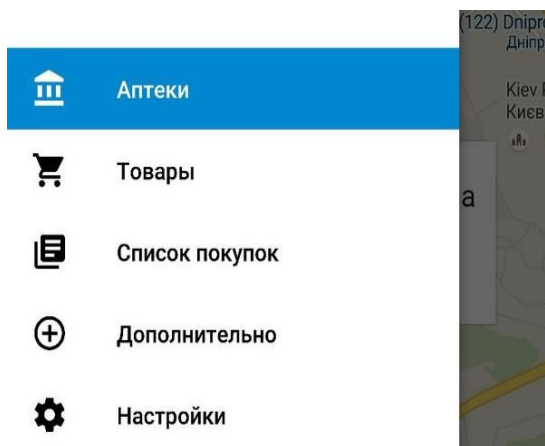


Рисунок 4.5 – Бокове меню програми

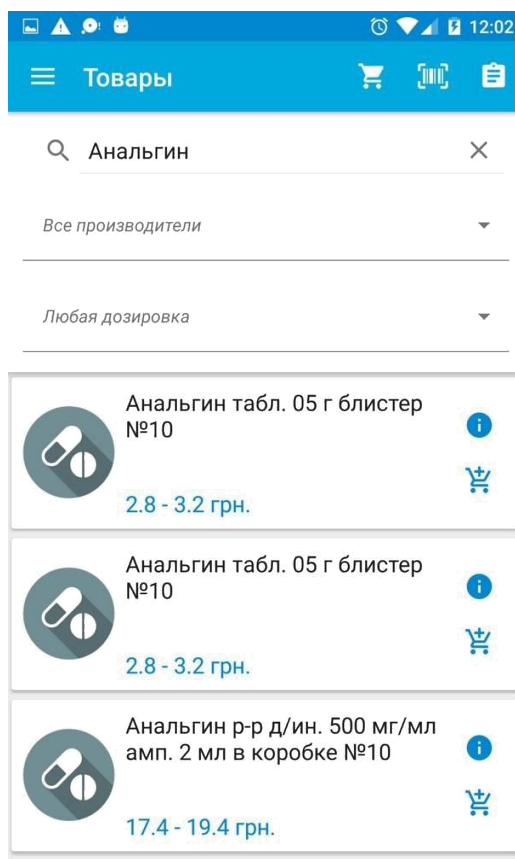


Рисунок 4.6 – Розгорнутий пошук препаратів

Програма для обліку та реалізації медикаментів та супутніх товарів в аптеці.

Це програмне забезпечення є спеціалізованим та розробленим виключно для аптек. В комплекті з

програмою встановлюється довідник найменувань зареєстрованих препаратів в Україні (86 тис. позицій), база «Аналогів» та Інструкцій (Анотацій) до препаратів.

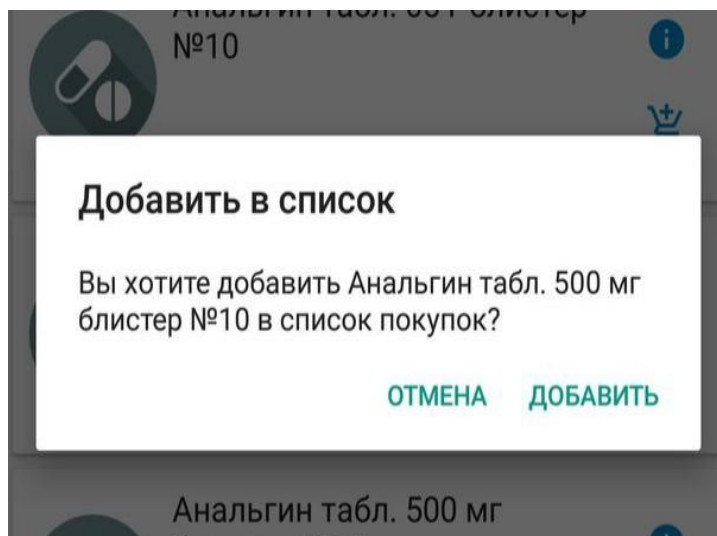


Рисунок 4.8 – Додавання препарату в список покупок

На екрані детальної інформації про препарат є три кнопки (рис. 4.9):



Рисунок 4.9 – Меню на екрані препарату

- додати до списку покупок – препарат буде доданий до списку покупок для зручного доступу в майбутньому та відображений на відповідному екрані;
- знайти аналоги препаратів – буде проведено пошук на відображено список препаратів, що можуть замінити даний. Це можуть бути інші дозування, виробники чи форми випуску цього препарату, або ж інші препарати такої ж фармакологічної групи;
 - пошук акцій з даним препаратом – пошук на наявність акцій з препаратом.

Добавивши медичні препарати до списку покупок, їх можна переглянути на відповідному екрані (рис. 4.10). Тут можна переглянути загальну суму всього списку, додати більше товарів, провести пошук по цьому списку якщо позицій забагато, очистити список.

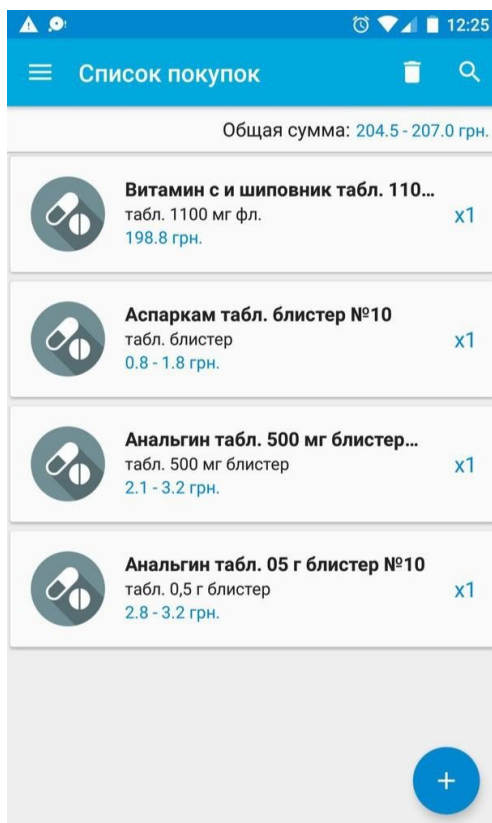


Рисунок 4.10 – Экран списка покупок

В нижній частині екрану знаходиться синя кнопка за допомогою якої можна додати інші препарати до списку покупок. При натисканні на елемент з цього списку буде показано детальну інформацію про препарат як показано на рис. 4.7.

Крім основних функцій у меню є пункт “Налаштування” в якому можна змінити стандартний вигляд головного екрану. А також екран з додатковою інформацією, який дає змогу переглянути інформацію про додаток, зателефонувати чи написати в технічну підтримку, перейти на сайт програми .



Рисунок 4.11 – Налаштування стандартного вигляду головного екрану

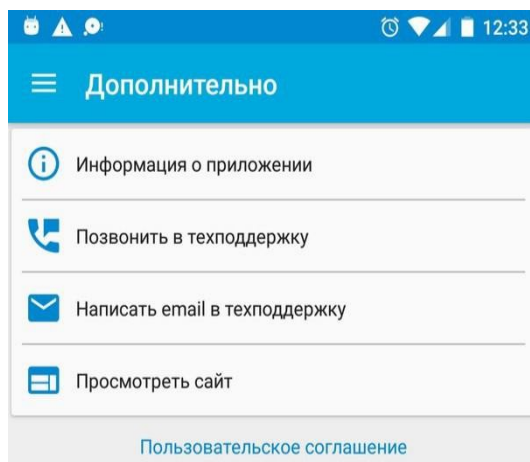


Рисунок 4.12 – Экран з додатковою інформацією

ВИСНОВКИ

Додатки для смартфонів в різних сферах медицини з кожним днем стають все більш популярними та використовуються великою кількістю людей. Одним з напрямків є додатки для пошуку лікарських препаратів в аптеках, що знаходяться в безпосередній близькості до людини.

Для розробки ПП було проведено аналіз існуючих мобільних додатків, які мають схожий функціонал до нашого. Детально розглянуто основні функції, виявлено недоліки у роботі та на основі цього сформовано основні вимоги, яким має відповідати розроблювана програма.

Розроблене технічне завдання охоплює загальне направлення для роботи, детальне описання вимог до проекту та технічних характеристик програмних засобів та обладнання. Сформовано вимоги до функціональних характеристик, засобів реалізації та надійності програмного забезпечення.

Було обрано інструменти, для реалізації програмного забезпечення. В якості основної мови програмування було вибрано Java, та XML для розмітки графічних елементів інтерфейсу. Розробка велася в середовищі Android Studio. Протягом всього періоду розробки використовувалася система контролю версій Git.

В результаті проведеної роботи було розроблено програмний продукт, який дає змогу зручно, за допомогою мобільного телефону, здійснити пошук медичних препаратів та аптечних мереж. Було реалізовано можливість формування списку покупок, його збереження та використання. Є можливість пошуку товару по штрих-коду та відображення аптек на карті.

ПЕРЕЛІК ПОСИЛАНЬ

1. Информационные технологии в здравоохранении: будущее или настоящее? [Електронний ресурс]. – 1603. – Режим доступу до ресурсу: <http://www.apteka.ua/article/326752?print=1>.
2. Додаток GeoApteka [Електронний ресурс] – Режим доступу до ресурсу: <https://play.google.com/store/apps/details?id=com.trinetix.geoapteka>.
3. Додаток Аптеки України [Електронний ресурс] – Режим доступу до ресурсу: <https://play.google.com/store/apps/details?id=com.ls.ukrainiandrugstores>.
4. Додаток Tabletki [Електронний ресурс] – Режим доступу до ресурсу: <https://play.google.com/store/apps/details?id=com.tabletkiua.tabletki>.
5. Операционная система Android [Електронний ресурс] – Режим доступу до ресурсу: <https://ru.wikipedia.org/wiki/Android>.
6. Голощапов А. Google Android: программирование для мобильных устройств / Голощапов. – СПб: БХВ-Петербург, 2010. – 448 с.
7. Коматинэни С. Google Android: программирование для мобильных устройств / С. Коматинэни, Д. Маклин, С. Хэшими. – СПб: Питер, 2011. – 736 с. – (1).
8. Хорстманн К. Java SE 8. Вводный курс. / К. Хорстманн.. – 864 с.
9. Лонг Ф. Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищенных программ / Фред Лонг. – СПб: Вильямс, 2014.
10. Ткаченко О. М. Об'єктно-орієнтоване програмування мовою Java. Навчальний посібник / О. М. Ткаченко, В. А. Каплун. – Вінниця: Вінниця, 2006. – 107 с.
11. Середовище розробки Android Studio [Електронний ресурс] – Режим доступу до ресурсу: 1. https://uk.wikipedia.org/wiki/Android_Studio. 12. Офіційний сайт Android Studio [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/studio/index.html>.
13. Сервіс Google maps [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Maps.
14. Сергеев Александр Петрович. [HTML і XML. Професійна робота](#) = HTML и XML. Профессиональная работа. – М. : «Діалектика», 2004. – С. 880.
15. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура.
16. Комп'ютерні мережі: [навчальний посібник] / А. Г. Микитишин, М. М. Митник, [П. Д. Стухляк](#), В. В. Пасічник. – Львів: «Магнолія 2006», 2013ю – 256 с.

17. Chacon S. Pro Git / S. Chacon, B. Straub. – Los Angeles: Apress, 2014. – 605 с. – (2).

18. ДСТУ 3321:2003 Система конструкторської документації. Терміни та визначення основних понять.

19. Методология функционального моделирования SADT [Электронный ресурс] – Режим доступа до ресурсу: http://www.infosystem.ru/designing/methodology/sadt/theory_sadt.html.

20. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений/Под ред. проф. А.Д. Хомоненко. – СПб.: КОРОНА принт, 2002. – 672с.

21. Братищенко В.В. Проектирование информационных систем. – Иркутск: Изд-во БГУЭП, 2004. – 84 с.

22. Магазин Google Play [Электронный ресурс] – Режим доступа до ресурсу: https://uk.wikipedia.org/wiki/Google_Play.

Д О Д А Т К И

ДОДАТОК А

ФРАГМЕНТ ПРОГРАМНОГО КОДУ

```

public class MapPharmaciesFragment extends BaseFragment implements On-
MapReadyCallback {
public static final int FOCUS_DELAY_MILLIS = 500; public static final float
ZOOM_ASPECT = 15f;
private SupportMapFragment mapFragment; private GoogleMap googleMap;
private DefaultClusterRenderer<MarkerClusterItem> defaultClusterRenderer;
private MarkerClusterItem nearestClusterItem; private Handler delayHandler =
new Handler(); private List<Pharmacy> currentPharmacies; private LastEventCall-
back lastEventCallback; private boolean isAllPharmaciesSearch = false; private
boolean isSearchForSetOfDrugs = false;
private Map<Integer, Integer> searchedDrugsIdsAndQuantities = new
HashMap<>();
@Override
protected int getContentView() {
return R.layout.map_pharmacies_fragment;
}
@Override
public void onViewCreated(View view, Bundle savedInstanceState) { super.on-
ViewCreated(view, savedInstanceState);
getBaseActivity().setTitle(R.string.empty_string);
if (!realm.allObjects(ShoppingListDrug.class).isEmpty()) {
for (ShoppingListDrug shoppingListdrug : realm.allObjects(ShoppingListDrug.class))
{ searchedDrugsIdsAndQuantities.put(shoppingListdrug.getId(),
shoppingListdrug.getQuantity());
}
}
mapFragment = (SupportMapFragment) getChildFragmentManager().findFragment-
ById(R.id.map); mapFragment.getMapAsync(this);}
@Override
public void onAttach(Activity activity) { super.onAttach(activity);
Fragment parentFragment = getParentFragment(); if (parentFragment instanceof
LastEventCallback) {
lastEventCallback = (LastEventCallback) parentFragment;
}
}
@Override

```

```

public void onMapReady(GoogleMap googleMap) { this.googleMap = oogleMap;
if (!((MainActivity) getBaseActivity()).isSearchInProgress()) { lastEventCallback-
.getLastEvent();
}}
@Override
public void onDestroyView() { super.onDestroyView();
if (mapFragment != null)
getChildFragmentManager().beginTransaction().remove(mapFragment).commitAl-
lowingStateLoss();
}
private void setUpMapUI() {
if (!googleMap.isMyLocationEnabled()) googleMap.setMyLocationEnabled(true);
if (googleMap.getUiSettings().isMapToolbarEnabled())
googleMap.getUiSettings().setMapToolbarEnabled(false);
}
public void onEventMainThread(LocationEvent locationEvent) { googleMap.ani-
mateCamera(CameraUpdateFactory.newLatLngZoom(locationEvent.location,
ZOOM_ASPECT));
}
public void onEventMainThread(PharmaciesEvent pharmaciesEvent) {
if (!((MainActivity) getBaseActivity()).isSearchInProgress() && !pharmaciesEvent.re-
set) { currentPharmacies = pharmaciesEvent.pharmacies;
isAllPharmaciesSearch = pharmaciesEvent.isAllPharmaciesEvent; isSearch-
ForSetOfDrugs = pharmaciesEvent.isSearchForSetOfDrugs;
if (!isAllPharmaciesSearch) {
if (!((MainActivity) getBaseActivity()).checkSearchBarVisibility()) { ((MainActivity)
getBaseActivity()).setBarWithDrugToSearch(pharmaciesEvent.allSearchedDrugs,
pharmaciesEvent.selectedDrugs, isSearchForSetOfDrugs);
}
} else if (!((MainActivity) getBaseActivity()).getSearchViewQuery().isEmpty())
{ ((MainActivity)
getBaseActivity()).setSearchViewQuery(getString(R.string.empty_string));
}
onEventMainThread(new FilteringEvent(preferences.getCurrentPharmacy())); se-
tUpMapUI();
}}
public void onEventMainThread(FilteringEvent filteringEvent) { List<Pharmacy> fil-
teredValues = new ArrayList<>();

```

```

Stream<Pharmacy> filteredStream = Stream.of(currentPharmacies).filter(pharmacy -
> pharmacy.getTitle().toLowerCase().contains(filteringEvent.criteri-
a.toLowerCase()));
if (preferences.isOnlyWorking()) {
    filteredStream = filteredStream.filter(pharmacy -> pharmacy.getSchedule().getI-
sOpen());
}
if (preferences.isBonusesFiltering()) { filteredStream = filteredStream.filter(pharmacy
->
pharmacy.getBonusesInfo() != null && !pharmacy.getBonusesInfo().isEmpty());
}
filteredValues.addAll(filteredStream.collect(Collectors.toList()));
performClustering(filteredValues);
}

private void performClustering(List<Pharmacy> pharmacies) {
    ClusterManager<MarkerClusterItem> clusterManager = new ClusterManager<>(get-
BaseActivity(),
googleMap);
    clusterManager.clearItems(); googleMap.clear();
    googleMap.setInfoWindowAdapter(new PharmacyInfoWindowAdapter(pharmacies));
    googleMap.setOnCameraChangeListener(clusterManager); googleMap.setOnMark-
erClickListener(clusterManager); googleMap.setOnInfoWindowClickListener(cluster-
Manager);
    if (pharmacies.size() != 0) {
        defaultClusterRenderer = new CustomMapRenderer(getBaseActivity(), googleMap,
clusterManager);
        clusterManager.setRenderer(defaultClusterRenderer); clusterManager.setOnClus-
terItemClickListener(markerClusterItem -> {
            Marker marker = defaultClusterRenderer.getMarker(markerClusterItem); mark-
er.setSnippet(markerClusterItem.getIndex()); marker.showInfoWindow();
            return false; });
        clusterManager.setOnClusterItemInfoWindowClickListener(markerClusterItem ->
        { Pharmacy pharmacy = markerClusterItem.getPharmacy();
        if (pharmacy != null) { DrugstoreInfoActivity.startActivityForResult(getBaseActiv-
ity(), pharmacy,
isSearchForSetOfDrugs);
        }
    });
}

```

```

for (int i = 0; i < pharmacies.size(); i++) { Pharmacy pharmacy = pharmacies.get(i);
MarkerClusterItem markerClusterItem = new MarkerClusterItem(pharmacy, String.-
valueOf(i)); clusterManager.addItem(markerClusterItem);
if (i == 0)
nearestClusterItem = markerClusterItem;
}
} else { googleMap.clear();
}
clusterManager.cluster();
delayHandler.postDelayed(() -> focusOnNearest(clusterManager), FOCUS_DE-
LAY_MILLIS);
}
private void focusOnNearest(ClusterManager<MarkerClusterItem> clusterManager) {
if (googleMap != null && nearestClusterItem != null) {
googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(nearestClus-
terItem.getPosition(), ZOOM_ASPECT), new GoogleMap.CancelableCallback() {
@Override
public void onFinish() { delayHandler.postDelayed(() -> {
Marker marker = defaultClusterRenderer.getMarker(nearestClusterItem); nearest-
ClusterItem = null;
if (marker != null) { clusterManager.onMarkerClick(marker);
}
}, FOCUS_DELAY_MILLIS);
}
@Override
public void onCancel() { nearestClusterItem = null;
});
}}
}

```