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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Інформаційна система «Чисте місто»»

Виконав:
студент IV курсу, групи ІО-64
Пащевський Руслан Дмитрович _____

Керівник:
Доцент кафедри ОТ, к.т.н.,
Русанова Ольга Веніаминівна _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Доцент кафедри СП СКС, к.т.н.
Орлова Марія Миколаївна _____

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

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____2020 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Пащевському Руслану Дмитровичу

1. Тема проєкту «Інформаційна система «Чисте місто»», керівник проєкту Русанова Ольга Веніаміновна, доцент, кандидат технічних наук, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проекту *3 червня 2020р.*

3. Вихідні дані до проекту *див. технічне завдання*

4. Зміст пояснювальної записки *Аналіз і характеристика об'єкта проектування, обґрунтування оптимального варіанта реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку. Висновки.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *приниципова схема, функціональна схема, структурна схема*

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	Сімоненко В. П., професор, д.т.н.		

7. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Затвердження теми роботи*	*01.09.2019*	
2.	*Вивчення та аналіз завдання*	*15.12.2019-15.03.2020*	
3.	*Розробка архітектури додатку*	*15.03.2020-25.03.2020*	
4.	*Написання програмної частини*	*25.03.2020-05.04.2020*	
5.	*Тестування та виправлення помилок*	*05.04.2020-15.04.2020*	
6.	*Оформлення пояснювальної записки*	*15.04.2020-20.05.2020*	
7.	*Захист програмного продукту*	*25.04.2020*	
8.	*Передзахист*	*26.05.2020*	
9.	*Захист*	*15.06.2020*	

Студент Руслан ПАЩЕВСЬКИЙ

Керівник Ольга РУСАНОВА

АНОТАЦІЯ

Дана бакалаврська дипломна робота присвячена розробці інформаційної системи у вигляді мобільного додатку на базі операційної системи Android. Метою цього дипломного проекту є розробка програми, яка зможе надати можливість людям брати участь у забезбеченні та підтриманні чистоти свого міста.

Даний сервіс надає можливість користувачу зафіксувати проблемну ділянку то додати її до бази даних системи з метою подальшого вирішення цієї проблеми.

Сервіс буде корисний для будь-якого міста чи населеного пункту, адже в наш час з проблемою сміття на вулицях кожна людина зіштовхується майже кожен день і даний додаток буде сприяти поліпшенню цієї ситуації та дотриманню чистоти довкілля.

ANNOTATION

This baccalaureate graduation work is devoted to development of an information system in the form of a mobile application on the basis of the Android operating system. The purpose of this diploma project is development of the program which will be able to give opportunity to people to participate in support and maintenance of purity of the city.

This service provides an opportunity for the user to record a problem section and to add it in a system database in order to further tackle the problem.

The service will be useful for any city or any inhabited locality, because nowadays every person faces the problem of garbage in the streets almost every day and this application will help to improve this situation and maintain a clean environment.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6419. 02.000 ТЗ	Технічне Завдання	2	
3	А4	ДП 6419. 03.000 ПЗ	Пояснювальна записка	56	
4	А4	ДП 6419. 04.000 Д1	Принципова схема	1	
5	А4	ДП 6419. 05.000 Д2	Функціональна схема	1	
6	А4	ДП 6419. 06.000 Д3	Структурна схема	1	
7	А4	ДП 6419. 07.000 Д4	Лістинг програми	11	

Зм.		№ документа	Підп.	Дата	ДП.6419.01.000 ВП		
Розробив		Пащевський Р.Д			Інформаційна система «Чисте місто» Відомість дипломного проєкту	Літ.	Аркуш 1 / 1
Перевір.		Русанова О.В..				Т	
Н.конт		Симоненко В.П.				НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64	
Затв.							

Технічне завдання
до дипломного проєкту

на тему «Інформаційна система «Чисте місто»»

ЗМІСТ

1.НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Інформаційна система "Чисте місто"».

Область застосування: програма може бути використана у вільному доступі будь-яким користувачем для участі у підтримці чистоти свого міста.

2.ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3.МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення інформаційної системи яка зможе долучати людей до підтримки чистоти на вулицях свого міста.

4.ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

5.ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Додаток, що розробляється повинен:

- Бути здатним до реєстрації нових користувачів та авторизації вже зареєстрованих людей.
- Надавати можливість додавання особистого запису до системи.
- Показувати повний список записів у системі.
- Показувати Google Карту з відображеними на ній проблемними ділянками.
- Надавати можливість перегляду власного профіля.

5.2. Вимоги до пристрою, де відбувається тренування моделей

- Мінімальна версія ОС - Android 6.0 та вище;
- Мінімальний обсяг оперативної пам'яті - 2 ГБ;
- Розмір сховища - 16 Г;
- Частота процесора - 1,4 ГГц;
- Камера (задня / фронтальна) - 12 Мпікс / 4 Мпікс
- Архітектура - 64-бітна.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення необхідної літератури	19.02.2020
Складання і узгодження технічного завдання	06.03.2020
Написання вступної частини та огляд рішень	19.03.2020
Розробка архітектури додатку	03.04.2020
Написання програмної частини	10.04.2020
Тестування та виправлення помилок	01.05.2020
Оформлення документації дипломного проекту	15.05.2020
Попередній захист та проходження нормативного контролю	29.05.2020
Захист дипломного проекту	15.06.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційна система «Чисте місто»»

Київ – 2020 року

ЗМІСТ

ВСТУП

Проблема забруднення навколишнього середовища є сьогодні актуальною як ніколи, особливо це стосується мегаполісів. Очищення вулиць від сміття - це болюче питання для всіх великих міст. Ця проблема є «видимою»: люди стикаються з нею щодня і саме тому вона їх турбує найбільше.

Розглянемо головні причини стрімкого збільшення кількості відходів[1]:

- Першим фактором є збільшення кількості населення. За останні кілька десятиліть науковий та медичний прогрес призвів до того, що середня тривалість життя людини зросла. І це зростаюче населення є однією з найбільших причин збільшення відходів у всьому світі.
- Наступним фактором є зміна способу життя людей. Харчові звички протягом певного часу змінюються у всьому світі. Значна частина населення, особливо в міських та напівміських районах, все більше віддає перевагу одразу готовим для вживання продуктам, перед традиційно приготовленою їжею. Люди їдять упаковану їжу і залишають упаковку позаду, що врешті-решт пробивається у смітники та смітники, чекаючи, що всюди задушиться земля та вода.
- Іншим фактором є промисловий зріст. За останні кілька десятиліть все більше і більше частин світу стає все більш індустріалізованими. Одним із побічних ефектів індустріалізації завжди було утворення відходів. Навіть сьогодні, незважаючи на підвищену обізнаність, глобальна промисловість продовжує виробляти багато метричних тонн необроблених відходів. Цей процес прискорюється, оскільки все більше сільських районів у всьому світі стає індустріалізованими, що призводить до створення ще більшої кількості біорозкладних відходів.

Тому взявши до уваги описану проблему з’являється потреба в інформаційній системі з допомогою якої люди зможуть долучатися до підтримки чистоти свого міста та довкілля. Користувачи зможуть самостійно додавати записи тих місць, які на їх думку потребують прибирання чи клінінгу.

РОЗДІЛ 1.
ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Загальні відомості про інформаційні системи

Інформаційною системою можна назвати середовище, яке складається з різних структурних елементів та засобів маніпулювання інформацією таких як мережі, бази даних, засоби зв’язку та ін. Основним завдання ІС є організація, обробка, зберігання, подання та передавання інформації.

Інформаційні системи здавна знаходять (в тому чи іншому вигляді) досить широке застосування в життєдіяльності людства. Це пов'язано з тим, що для існування цивілізації необхідний обмін інформацією.

Прикладом найдавніших і найпоширеніших ІС слід вважати бібліотеки. Здавна в бібліотеках збирають книжки, зберігають їх, дотримуючись певних правил, створюють каталоги різного призначення для полегшення доступу до книжкового фонду. Видаються спеціальні журнали та довідники, що інформують про нові надходження, ведеться облік видачі.

З розвитком інформаційних систем були обумовлені характери еволюції технічних засобів обробки інформації. Крім того, в теперішній час розвиток ІС на підприємствах настільки тісно пов'язаний з потребами корпоративного бізнесу, що процеси їх розвитку нерідко сприймаються як єдине ціле. Розвиток ІС[2] включає декілька поколінь.

Перший період (1960 - 1970 рр.) – від EDP до MIS. Після того, як в 1951 році розробили перший бізнес-комп’ютер, в організаціях почали формуватися спеціальні відділи ІС, вони сигналізували про початок нової ери, від EDP до MIS. У цей період було додано ще одну роль у використанні комп’ютерів, таку як переробка даних у корисні інформативні звіти. Інформаційні системи управління або MIS, таким чином, перетворилися на TPS.

Основна увага цієї нової ролі полягає у розробці бізнес-додатків, які надають кінцевим користувачам для керівників заздалегідь визначені звіти про управління, які дадуть менеджерам інформацію, необхідну для цілей прийняття рішень. Ця епоха також знаменує період розвитку, коли фокус організацій повільно переміщувався від простої автоматизації основних

бізнес-процесів у 50-х роках до консолідації контролю в межах функції обробки даних.

Інформаційні системи управління (MIS) - надають інформацію у вигляді заздалегідь визначених звітів та дисплеїв для підтримки прийняття бізнес-рішень. Прикладами випуску MIS є аналіз продажів, результативність виробництва та систем звітування про тенденції витрат.

Зазвичай MIS генерує три основні типи інформації, тобто детальну, резюме та виняток. Детальні звіти про інформацію зазвичай підтверджують діяльність по обробці транзакцій. Зведена інформація консолідує дані у форматі, який людина може швидко та легко переглядати, тоді як інформація про винятки фільтрує дані, щоб повідомляти інформацію, яка знаходиться поза нормальним станом. Прикладом звіту про виключення є звіт про інвентаризацію. Звіт може повідомити відділ закупівель про речі, які йому потрібно змінити.

Звіти про винятки допомагають менеджерам економити час, оскільки їм не доведеться шукати докладний звіт про винятки. Звіти про винятки, таким чином, допомагають менеджерам зосередитись на ситуаціях, які потребують негайних рішень чи дій.

Другий період (1970 - 1980) – PC and DSS. У цю другу епоху технологічний прогрес продовжував зростати. Найважливішим прогресом стала компанія впровадження персональних комп'ютерів (ПК). З впровадженням ПК, організації почали розподіляти свої обчислювальні / обробні потужності по всій організації. У міру того, як коло користувачів розширювалося, організації брали більш сильну орієнтацію управління на свій традиційно технічно-орієнтований підхід до операцій з ІС. Рух почав зосереджуватися на "інтерактивній комп'ютерній системі", щоб допомогти керівникам у вирішенні проблем.

Як результат цього, попередньо визначені звіти про управління вже не були достатніми для задоволення багатьох потреб управління у прийнятті рішень. Для задоволення таких потреб народилася концепція систем підтримки прийняття рішень (DSS). Нова роль інформаційних систем полягала у наданні управлінським кінцевим споживачам тимчасової та інтерактивної підтримки їх процесу прийняття рішень.

DSS використовує дані з внутрішніх та / або зовнішніх джерел. Внутрішні джерела даних можуть включати дані про продажі, виробництво, товарні запаси або фінансові дані з бази даних організації. Зовнішні джерела можуть включати процентні ставки, тенденції населення або ціноутворення матеріалів. Менеджери, які використовують DSS, можуть маніпулювати даними, які використовуються для DSS, щоб допомогти у прийнятті рішень. База даних - це сховище або сховище даних, яке організоване для ефективного доступу, пошуку, пошуку та оновлення. Фізично база даних зазвичай зберігається на жорстких дисках, магнітній стрічці, оптичних дисках, а також хмарі (хостинг сховища).

Системи підтримки прийняття рішень (DSS) - надають інтерактивну спеціальну підтримку процесу прийняття рішень керівникам та іншим професіоналам бізнесу. DSS обслуговує рівні управління, операції та планування організації, як правило, середнього та вищого керівництва для прийняття рішення. Прикладами DSS є прогнозовані показники доходу на основі припущень щодо продажу нових товарів, цін на продукцію та систем аналізу ризиків.

Третій період (1980 - 1990 рр.) - EIS та Інтернет. У цю епоху багато бізнес-підрозділів вдалися до придбання власного обладнання та програмного забезпечення відповідно до потреб відомства. Це була ера персональних обчислень, породжуючи відомчі обчислення. Ця тенденція спричинила нові проблеми несумісності, зв’язку та цілісності даних між функціональними підрозділами.

Кінцеві користувачі тепер можуть використовувати власні обчислювальні ресурси для підтримки своїх вимог до роботи, а не чекати опосередкованої підтримки централізованого відділу корпоративних інформаційних послуг. Стало очевидним, що більшість вищих керівників не використовували ані звітів MIS, ані можливостей аналітичного моделювання DSS, тому була розроблена концепція виконавчих інформаційних систем (EIS). Він також відомий як система підтримки виконавчої влади (ESS).

Виконавчі інформаційні системи (EIS) - це тип інформаційної системи управління, призначений сприяти та підтримувати інформаційні потреби та потреби у прийнятті рішень, забезпечуючи легкий доступ як до внутрішньої, так і до зовнішньої інформації, що має значення для досягнення стратегічних

цілей організації. Зазвичай це розглядається як спеціалізована форма DSS. Прикладами ICC є системи для легкого доступу до аналізу ефективності бізнесу, дій усіх конкурентів та економічних розробок для підтримки стратегічного планування.

У цей період також відбувається швидкий розвиток апаратного, програмного забезпечення та телекомунікацій для ПК, а також широке впровадження мережі TCP / IP або Інтернету. Це стає новим явищем в IT-індустрії.

Четвертий період (1990 - 2000) - Штучний інтелект (AI), Експертні системи (ES) та системи управління знаннями (KMS). Ця епоха знаменується значним зрушенням технології ІС та бізнес-середовища. Комерціалізація Інтернету дозволила отримати нові методи спілкування та способи їх ведення бізнес, який не був можливим у попередні епохи. Інтернет дозволяє поширювати знання в різних куточках світу, незалежно від часу та місця.

Крім того, відбулися прориви у розробці та застосуванні методів штучного інтелекту (AI) до бізнес-інформаційних систем. З меншою потребою в людському втручанні працівники знань можуть бути звільнені для вирішення складніших завдань.

Експертні системи (ES) та системи управління знаннями (KMS) взаємопов'язані між собою. ES використовує дані з KM для отримання бажаного виходу інформаційної системи, наприклад, системи затвердження заявки на позику.

Експертні системи (ES) - це комп’ютерна система, яка наслідує здатність людини прийняти рішення. Деякі ВН створені таким чином, щоб замість них діяти люди, а інші - для того, щоб допомогти їм. Наприклад, існують експертні системи, які дозволяють діагностувати хвороби людини, робити фінансові прогнози та планувати маршрути транспортних засобів. Система управління знаннями (KMS) - це система, заснована на знаннях, яка підтримує створення, організацію та розповсюдження ділових знань у межах підприємства. Прикладами KMS є системи доступу до внутрішньої мережі та служби довідки.

У цей період в середині-кінці 1990-х років відбулося революційне виникнення систем планування ресурсів підприємств (ERP). Ця специфічна для організації форма стратегічної інформаційної системи інтегрує всі аспекти

фірми, включаючи її планування, виробництво, продаж, управління ресурсами, людські ресурси та маркетинг - практично кожну функцію бізнесу.

П'ятий період (2000 - сьогодні) - електронний бізнес, електронна комерція, мобільні та хмарні обчислення. Швидке зростання Інтернету, інтрамереж, екстранетів та інших взаємопов'язаних глобальних мереж у 90-х роках різко змінило можливості ІС у бізнесі. Інтернет, підприємницькі та глобальні системи електронного бізнесу та торгівлі стають звичним явищем в роботі та управлінні сьогоднішніми бізнес-підприємствами. Сьогоднішні інформаційні системи роблять ті самі основні речі, як і 50 років тому: обробкуи транзакцій, ведення обліку, управління звітності та управління підтримкою до облікової системи, а також процеси організації. Що змінилося сьогодні – став ще більший зв’язок між подібними та різними компонентами системи, значно вищий рівень інтеграції системних функцій у додатках, чудова мережева інфраструктура та потужні машини з більшою ємністю зберігання.

Інтернет та пов'язані з ними технології та програми змінили спосіб роботи бізнесу та людей, а також те, як інформаційні системи підтримують бізнес-процес, прийняття рішень та конкурентну перевагу. Сьогодні багато підприємств в повній мірі використовують Інтернет-технології для інтеграції бізнес-процесів та створення інноваційних програм для електронного бізнесу. Електронний бізнес - це використання Інтернет-технологій для роботи та розширення можливостей бізнес-процесів, електронної комерції та співпраці підприємств у межах компанії та зі своїми клієнтами, постачальниками та іншими зацікавленими сторонами в бізнесі.

І нарешті, великі дані, мобільні та хмарні обчислення в останні епохи смартфонів, планшетів та соціальних медіа та швидке зростання бездротових мережевих технологій. Великі дані - це сукупність даних із традиційних та цифрових джерел всередині та поза організацією, яка є джерелом для постійного виявлення та аналізу. Хмарні обчислення дозволяють отримати зручний доступ до спільного пулу налаштованих обчислювальних пристроїв, таких як мережі, сервери, сховища даних, додатки та послуги, які можна швидко налаштувати та випустити з мінімальними зусиллями управління або взаємодії постачальника послуг.

1.2 Аналіз предметної області

В наш час проблема забруднення навколишнього середовища є дуже актуальною. Особливо проблемними в плані забруднення вважаються великі міста, або мегаполіси. Дуже болючим питання для таких міст є – очищення вулиць від сміття. Однією з найбільших причин збільшення відходів вважається зростання кількості населення. Тож іноді комунальні служби просто не мають змогу моніторити все місто та підтримувати його чистоту.

Тому з'являється потреба в інформаційній системі, яка зможе долучити небайдужих людей до підтримання чистоти та порядку в місті.

Інформаційна система «Чисте місто» надає можливість користувачу брати участь у збереженні довкілля та інформуванні відповідних комунальних служб. Громадянин має змогу висловити своє невдоволення про стан навколишнього довкілля міста, використавши цю систему у вигляді мобільного додатку.

Користувачі можуть повідомити про заповнений смітник, про проблеми на автодорозі загального користування, про несправне освітлення, про обмальовані будинки, занедбані двори та інше. Для того щоб це зробити людині потрібно лише завантажити додаток, авторизуватися за допомогою лише email та пароля. Після таких простих маніпуляцій користувачу відкриваються такі можливості як перегляд вже існуючих записів у системі, так і додавання власних.

Для того, щоб додати власний запис до системи користувачу потрібно підійти до проблемної ділянки, створити новий запис, прикріпити фото, опис та геолокацію і нажати кнопку додати. Запис одразу з'явиться у базі даних і його зможуть переглянути інші користувачі та відповідні органи, щоб в подальшому вирішити цю проблему.

1.3 Огляд систем аналогів

Внаслідок огляду ринку програмних продуктів, які вирішують проблеми контролю надмірно засмічених територій, було виявлено:

- Автоматизована інформаційна система «Народний контроль»;
- «Clean City» mobile application
- Мобільний додаток «Clean City»

Система «Народний контроль»

Призначена для об'єднання зусиль муніципальної влади і жителів міста з підтримки чистоти і порядку на території міста Череповця.

АІС «Народний контроль»[7] дає можливість жителям міста в оперативному режимі інформувати адміністрацію муніципалітету про наявні проблеми в утриманні території, порушення Правил благоустрою і відстежувати їх усунення.

Способи звернення громадян за фактами порушень Правил благоустрою та утримання:

- через програму «Народний контроль» на сайті;
- по телефону диспетчеру.

Додаток диспетчера забезпечує співробітникам органів місцевого самоврядування (ОМСУ) приймати заявки, в тому числі по телефону, направлятидля иконання в відповідальний орган, відстежувати їх виконання, формувати статистичні звіти за зверненнями громадян.

Рисунок 1.3 – Приклад роботи АІС «Народний контроль»

Особливості системи «Народний контроль»:

- Інформування адміністрації муніципалітету про можливі проблемні ділянки;
- Можливість додавати опис проблеми;
- Можливість додавання фото;
- Також в цій системі передбачено додавання геолокації для більш простого відслідковування проблеми;

Мінуси системи:

- Відсутність мобільного додатку. Повідомити про проблему можна тільки на офіційному сайті системи або ж подзвонивши диспетчеру, який займається моніторингом дзвінків в даній АІС;

Мобільний додаток «Clean City»

Надає можливість відправляти відгуки про стан заповненості сміттєвих баків у місті Львів. Користувач може вибрати найближчий до нього сміттєвий бак на Google карті, та залишити свій відгук про стан даного смітника чи потребує він вивозу, чи ні. З відправлених вами даних формується статистика, яку аналізують у міській раді. Це дозволить вносити зміни в роботу перевізників, ефективніше розміщувати нові смітники та загалом покращити ситуацію зі сміттям у місті.

Як це працює: оберіть стан заповненості сміттєвого баку і надішліть свій відгук.

Повний список таких ділянок з можливістю відкривати їх та ознайомлюватися з більш детальною інформацією.

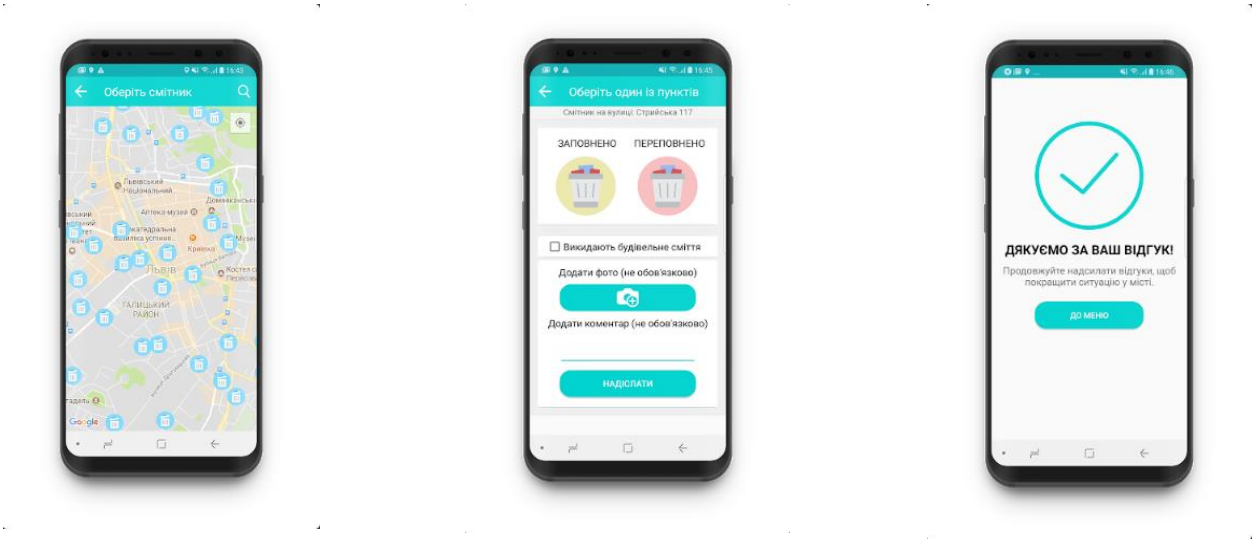


Рисунок 1.3 – Приклад роботи мобільного додатку «Clean City»

Особливості даного мобільного додатку:

- Можливість відслідковування всіх сміттєвих баків міста на Google карті;
- Користувач може залишити свій коментар опису смітника;

- Можливість додавання фото для більш детального висвітлення проблеми, якщо така наявна;
- Іконки Заповнено / Переповнено дуже прості та стильні;

Мінусом цього додатку є те, що він вузьконаправлений. Тобто ця система разрахована тільки на моніторинг сміттєвих баків і нічого іншого. Тут немає можливосі повідомити про якусь іншу проблемну ділянку, крім смітника.

«Clean City» mobile application

Це глобальна соціальна мережа, яка дозволяє користувачам приєднуватись до збереження чистоти свого міста, громадянин може повідомити про зламану каналізацію, про заповнений смітник, про сміття в недозволенному місті та інше через цей додаток. Досить сфотографувати відповідне місце, програма автоматично відновлює свою позицію та надсилає скаргу відповідальним органам, щоб пришвидшити процес вирішення.

«Clean City» також дозволяє громадянам підписатися на програму збору відходів, почати сортувати сміттєві баки в домашніх умовах та працювати з компанією з переробки сміття.

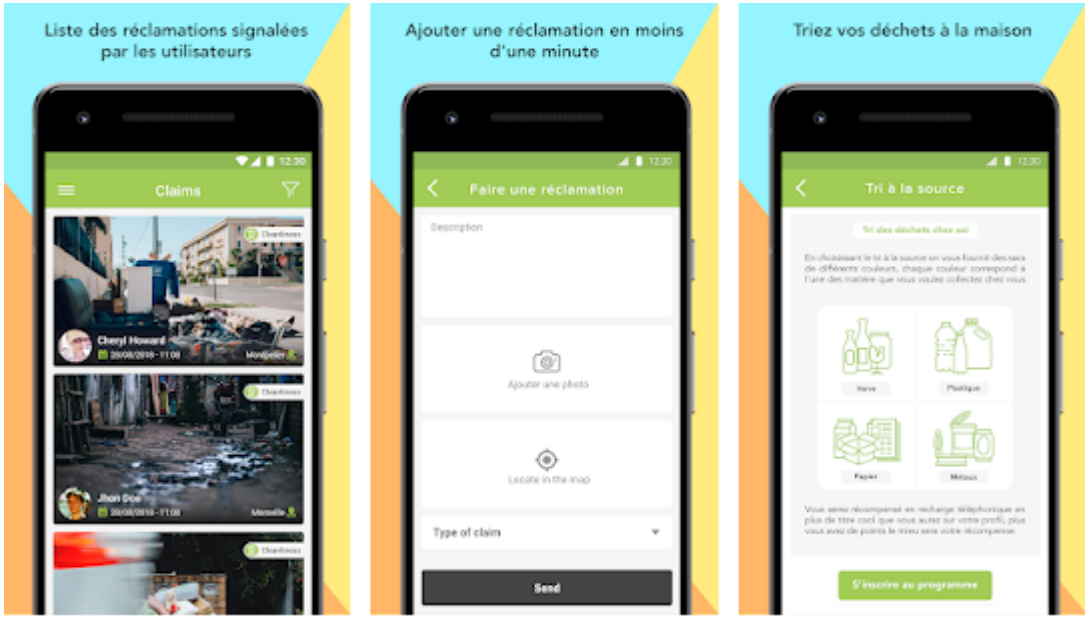


Рисунок 1.3 – приклад використання «Clean City» mobile application

Особливості «Clean City» mobile application:

- Можливість додавання запису будь-якого забрудненого місця, яке на думку користувача потребує клінінгу;

- Можливість додавання опису проблеми;
- Можливість додавання фото;
- Автоматичне прикріплення геолокації;
- Користувач може підписатися на програму збору відходів;

Мінусами даного мобільного додатку є те, що він повністю локалізований під французьку мову та країну Францію. Також цей додаток не безкоштовний, а тому, щоб мати можливість його використовувати користувачу потрібно заплатити майже 2$. На мою думку це є найбільшою проблемою цього додатку, адже якщо людина просто хоче внести якийсь свій внесок у підтримку чистоти свого міста та навколишнього середовища, їй не потрібно за це платити гроші.

Порівняння додатків

Таблиця 1.3 – Порівняння розлянутих додатків

Назва сервісу	**Мобільний додаток**	**Ціна**	**Локалізація**	**Додавання опису**	**Додавання фото**	**Додавання геолокації**
Інформаційна система «Чисте місто»	Так	Безкоштовно	Українська мова	Так	Так	Так
АІС «Народний контроль»	Ні	Безкоштовно	Російська мова	Так	Так	Так
Мобільний додаток «Clean City»	Так	Безкоштовно	Українська мова	Так	Ні	Так

Продовження таблиці 1.3 – Порівняння розлянутих додатків

Назва сервісу	**Мобільний додаток**	**Ціна**	**Локалізація**	**Додавання опису**	**Додавання фото**	**Додавання геолокації**
«Clean city» mobile application	Так	1.99$	Французька мова	Так	Так	Так

З усіх вище розглянутих додатків та таблиці їх порівняння можна побачити, що кожен сервіс має як свої плюси, так і мінуси. Усі системи орієнтовані на підтримання чистоти навколишнього середовища та довкілля. Але при порівнянні з моїм сервісом деякі з розглянутих додатків втрачають свою цінність або через вузьконаправлений функціонал, або через надто високу ціну для звичайного користувача, або ж через надто складний алгоритм повідомлення про проблему.

1.4 Постановка задачі

1.4.1 Призначення розробки

Інформаційна система «Чисте місто» призначена для того, щоб користувач міг брати участь у збереженні довкілля та чистоти свого міста, просто скориставшись своїм телефоном та зафіксувавши проблемне місце.

1.4.2 Цілі та задачі розробки

Цілями розробки є:

- дотримання чистоти вулиць;
- забезпечення чистоти будинків, історичних або культурних об’єктів;
- збереження довкілля свого міста;
- своєчасний вивіз сміття;
- нейтралізація наслідків порушень правил вигулу домашніх тварин;

Для досягнення поставлених цілей мають бути вирішені такі задачі:

- можливість додавання запису про нову забруднену ділянку;
- можливість ілюстрування проблеми фотографією;
- додавання геолокації;
- додавання опису;
- можливість перегляду списку усіх записів, які наявні в базі даних;
- наявність серверної бази даних

ВИСНОВКИ ДО РОЗДІЛУ 1

Було досліджено ринок товарів розглянутої предметної області. Це надало можливість звернути увагу на функціонал інших схожиг програмних продуктів. Даний аналіз допоможе не допустити схожих помилок при розробці даної інформаційної системи.

Отже, після дослідження ринку було виявлено, що сервісу, який би був доступний для будь-якої людини та мав би змогу надавати можливість долучатися до підтримки чистоти довкілля та вулиць міста на теперішній час немає. Кожен перелічений додаток має певні подібні частини функціоналу, але не дає того результату, який може надати дана інформаційна система. Інформаційна система «Чисте місто» дасть можливість користувачам безкоштовно завантажити цей додаток, та долучитися до підтримки чистоти довкілля.

РОЗДІЛ 2.
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ НА ОСНОВІ ОС ANDROID

2.1 Визначення основних функціональних можливостей системи

Основними функціями даної інформаійної системи будуть:

- Реєстрація – проводиться в 2 простих кроки, вказується адреса електронної пошти та пароль.
- Авторизація – користувач вказує у відповідних полях свою електронну адресу та пароль, якщо він вже проходив реєстрацію. Після чого йому надається доступ до основного функціоналу системи.
- Вивід списку усіх проблемних ділянок міста – можливість переглянути користувачу усі записи які знаходиться у базі даних.
- Перегляд окремого запису – можна відкрити будь-який запис для розгляду більш детальної інформації проблеми.
- Перегляд Google карти – можливість продивитись карту міста з відміченими на ній маркерами ділянок, які потребують клінінгу.
- Додавання особистої нотатки в систему – ця функція дозволяє користувачу зафіксувати будь-яку ділянку, яка на його думку потребую клінінгу.
- Видалення свого запису – можливість видалення запису.
- Вихід з особистого профілю – функція яка надає можливість вийти зі свого профілю.

2.2 Вибір основних інструментів розробки

2.2.1 Вибір операційної системи для розробки

Для роботи даної інформаційної системи буде використовуватися мобільний пристрій. Гаджет забезпече автономність та зручність у використанні. В теперішній час однією з найпоширеніших операційних систем мобільних пристріїв є система ОС Android. Android підтримує дуже велику кількість пристроїв від різних виробників. Так за минулий рік було продано 1,32 млрд смартфонів з Android або 85,9% від загальної кількості і близько 215

млн смартфонів з iOS або 14%. Та основна причина поширеня цієї операційної системи – це безкоштовні засоби розробки, в той час як рзробка під систему iOS потребує великих початкових витрат. Тож розглянувши всі ці аспекти мною була вибрана саме операційна система Android для розробки інформаційної системи.

Основними перевагами ОС Android є:

- Система Android може працювати на різних конфігураціях. Також вона є зовсім невибагливою у використанні. Виходячи з цих факторів, велика кількість світових компаній оснащують свої пристрої цією операційною системою.
- Через те, що система побудована на ядрі Linux, Android має велику гнучкість. Ця гнучкість пов’язана з тим, що ядро Linux має відкритий код та дає великі можливості для розробників. Android може запуститися навіть на пристроях, які мають невеликий об’єм оперативної пам’яті менше ніж 256 МБ. Але нові версії системи вимагають об’єм пам’яті не менш ніж 512 МБ. Android невибагливий до наявності у пристрої швидкого процесора і може працювати на ядрі з частотою 600 МГц.
- Операційна система використовує базу програм, яка є майже найбільшою в світі це дає можливість користувачам встановлювати додатки з офіційного Google репозиторію. Також кожен розробник може досить легко завантажити свій розроблений додаток до цього магазину. Така функція доступна лише завдяки відкритості системи. Додатки на мобільни пристрої з операційною системою Android можно встановлювати рызними способами будь-то завантаження .apk файлу через комп’ютер чи завантаження його безспосередньо з телефону або планшету.
- Відмінною особливістю Android є його інтегрованість з сервісами Google, такими як: Gmail, Hangouts, Voice Search і т.п. На Android офіційно реалізована підтримка Google Chrome, що дозволяє синхронізувати відкриваються в браузері вкладки на смартфоні з комп'ютерним браузером.
- Android має досить простий і інтуїтивно зрозумілий інтерфейс. Всі потрібні програми розміщуються одночасно на головному екрані і в

меню апарату, яке викликається натисканням на центральну сенсорну клавішу або відповідну кнопку на екрані.

- У Android можна запускати будь-які програми в фоні і вони будуть залишатися активними до тих пір, поки в пристрої не закінчиться вільна оперативна пам'ять або процес завантаження не закінчиться до кінця.
- Широкі можливості кастомізації зовнішнього вигляду, адже встановивши сторонній лаунчер і замінивши стандартні шрифти можна сильно змінити зовнішній вигляд системи.

Недоліками даної операційної системи можна вважати:

- Необхідність досить частої зарядки пристрою на базі операційної системи, що пов'язано з тим, що сама система не економить на наданих їй апаратних ресурсах. Частково недолік можна виправити налаштуванням енергозбереження, відключенням непотрібних функцій, але лише частково.
- Проблеми сумісності, адже нові версії операційної системи досить часто конфліктують зі знятими з продажу застарілими пристроями або пристроями, які випущені невідомими виробниками, тому можуть виникати проблеми після оновлення пристроїв, адже може трапитися так, що телефон після оновлення починає поводитися не так, як хотілося того - батарея швидко сідає, починаються зависання, перезавантаження і все в такому дусі. Доводиться примусово відкочуватися на стару версію і відключати автоматичне оновлення.
- Велика кількість налаштувань, що може бути досить складним для більшості користувачів.

2.2.2 Java

Однією з найбільших причин, чому Java настільки популярна, є незалежність платформи. Програми можуть працювати на декількох комп'ютерах різних типів; доки на комп'ютері встановлено середовище виконання Java (JRE), на ньому може працювати програма Java.

Більшість типів комп'ютерів будуть сумісні з JRE, включаючи ПК, що працюють на комп'ютерах Windows, Macintosh, Unix або Linux, а також на великих комп'ютерах з основним кадром, а також мобільних телефонах.

Оскільки це існує вже давно, деякі найбільші організації світу будуються з використанням мови. Багато банків, підприємств роздрібної торгівлі, страхових компаній, комунальних підприємств та виробників використовують Java.

Стів Зара, програміст більше 40 років, описує, як немає ознак того, що Java зменшується у використанні. Натомість це мова, що розвивається, яка майже унікально поєднує стабільність з інноваціями.

Підкреслюючи довговічність мови, Zara заявляє, що код, який ви, можливо, написали 15 років тому, буде працювати на найсучасніших JVM і отримає перевагу в швидкості останнього профілювання, перекладу нативного коду та управління пам'яттю.

Java принципово об'єктно-орієнтована. Код настільки надійний, тому що об’єкти Java не містять посилань на дані, зовнішні для себе.

Мова вважається дуже простою; однак, він постачається з бібліотекою класів, які пропонують широко використовувані утилітні функції, без яких не може працювати більшість програм Java.

Java API, бібліотека класів, є стільки ж частиною Java, скільки і самою мовою. Насправді справжнім завданням навчитися користуватися Java - це не вивчення мови, а вивчення API.

Мова складається з 50 ключових слів, але Java API має тисячі класів з десятками тисяч методів, які можна використовувати у своїх програмах.

Незважаючи на це, розробники не очікують вивчити всі Java API, і більшість з них лише досвідчені та володіють незначною частиною.

Java як і раніше є відповідною мовою програмування, яка не демонструє ознак зниження популярності, і тому варто вивчити її. Більшість розробників вважають його першою мовою програмування, оскільки її досить легко вивчити.

Оскільки мова має англо-подібний синтаксис із мінімальними спеціальними символами, Java може бути вивчена за короткий проміжок часу та використана для створення відповідних додатків.

Java є частиною сімейства мов, на які сильно впливає C ++ (як і C #), тому вивчення Java пропонує величезні переваги при вивченні цих двох інших мов.

Swarnim Srivastava, ентузіаст Java, погоджується, що мову варто вивчити. Він пояснює критичні області, в яких він використовується:

1. Створення програм для Android

Хоча існують інші способи створення додатків для Android, більшість програм написані на Java за допомогою Android-програмного забезпечення Android. Хоча Android використовує різницю JVM та способи упаковки, код все ще є переважно Java.

2. Веб-програми Java

Багато урядових, медичних, страхових, освітніх та оборонних відомств мають власні веб-програми, створені на Яві. Важливим прикладом цього є Gmail в Google.

3. Програмні засоби

Багато корисних програм та засобів розробки написані та розроблені на Java, напр. Eclipse, IntelliJ IDEA та NetBeans IDE.

4. Наукові програми

В даний час Java часто є типовим вибором для наукових застосувань, включаючи обробку природних мов. Основна причина цього полягає в тому, що вона безпечна, портативна, ремонтопридатна та оснащена кращими інструментами одночасності на високому рівні, ніж C ++ або будь-яка інша мова.

Залежно від ваших перспектив кар'єри, Java може перевести вас на різні кар’єрні шляхи. Незалежно від того, чи зацікавлені ви в створенні ігор, мобільних додатків, настільних додатків або веб-додатків, Java може працювати в цих умовах.

Незалежно від того, використовуєте ви чи ні Java у своїй кар'єрній чи службовій ролі, ви дізнаєтесь синтаксис, ідеї, шаблони, інструменти, стилі та ідіосинкразії, які також легко передаються на інші мови програмування.

Вивчення Java дасть вам можливість побачити, як одна мова робить щось у порівнянні з іншою. Крім того, знання того, що працює для Java та інших мов, зробить вас ще кращим всебічним розробником.

2.2.3 Kotlin

Це нова мова програмування для платформи Java. Kotlin - лаконічний, безпечний і прагматичний мову, сумісний з Java. Його можна використовувати практично скрізь, де застосовується Java: для розробки серверних додатків, додатків для Android і багато чого іншого. Kotlin прекрасно працює з усіма існуючими бібліотекам і фреймворками, написаними на Java, не уступаючи останньому в продуктивності.

Основними перевагами даної мови програмування є те, що:

- Kotlin - статично типізований мова, що підтримує автоматичний висновок типів, що дозволяє гарантувати коректність і продуктивність, зберігаючи при цьому вихідний код лаконічним.
- Kotlin підтримує як об'єктно-орієнтована, так і функціональний стиль програмування, дозволяючи створювати високорівневі абстракції за допомогою функцій, які є повноцінними об'єктами, і спрощуючи розробку і тестування багатопоточних додатків завдяки підтримці незмінних значень.
- Мова добре підходить для розробки серверних частин додатків, підтримує всі існуючі Java-фреймворки і надає нові інструменти для вирішення типових завдань, таких як створення розмітки HTML і операції з збереженими даними.
- Завдяки компактній середовищі виконання, спеціальної підтримки Android API в компіляторі і багатій бібліотеці Kotlin-функцій для вирішення основних завдань Kotlin відмінно підходить для розробки під Android.
- Це безкоштовна мова з відкритим кодом, підтримувана основними IDE і системами збірки.
- Kotlin - прагматичний, безпечний, лаконічний і сумісний мову, який приділяє велику увагу можливості використання перевірених рішень для популярних завдань, що запобігає поширені помилки (такі як виняток NullPointerException), що дозволяє писати компактний, легко читається код і забезпечує безшовну інтеграцію з Java.

2.3 Опис технологій використаних для розробки проекту

Для роботи даної інформаційної системи були використані всі необхідні технології, без яких реалізація клієнт-серверної архітектури буде неможлива. Розглянемо усі використані компоненти. Також для реалізації цієї системи буде використаний Firebase Service, з допомогою якого можна зимітувати роботу клієнт-серверної архітектури.

Gradle

Це система автоматичної збірки проекту, яка збудована на принципах Apache Ant та Apache Maven, надає DSL на мові Groovy замість традиційної XML форми представлення конфігурації проекта. Gradle[14] розробляли для побудови мультипроектів, які в подальшому можуть розростатися. Ця технологія підтримує інкрементальне збирання. Вона визначає, які частини програми були змінені, і виконує тільки ті задачі, які залежать від цих частин.

Основні переваги Gradle:

- Підтримує понад 600 плагінів, завдяки чому ми можемо використовувати його з безліччю технологій та інших мов програмуванняж;
- Підтримується всіма сучасними IDE;
- Підтримує Maven і Ivy репозиторії;
- Дозволяє інтегруватися з існуючими Ant і Maven проектами;
- Забезпечує нас вбудованими можливостями міграції з інших систем збірки;
- Дозволяє писати скрипти збірки на Java і Scala;
- Є можливість налаштування стандартних параметрів для всіх підпроектів в головному скрипті gradle;

Це всього лише частина переваг, про які, можна сказати. Головна концепція Gradle - позбавити розробника від більшості рутинних завдань і монстроідальних конфігурацій збірки, зробити збірку проекту простим і зрозумілим процесом.

Звісно у Gradle є і недоліки. Це і менш стабільна робота, властива продуктам, які швидко ростуть, і необхідність хоч трохи, але знати Groovy, і

більш складний DSL, і менша база знань, і більший час навчання. Але все ж кількість плюсів тут набагато перевищує кількість мінусів.

Firebase

Firebase [4] - це хмарний сервіс, який поєднує в собі дуже багато функцій, таких як: аутентифікацію, базу даних, зберігання файлів, повідомлення та інші. Якщо Firebase використовується, як база даних, то вона може оновлюватись в реальному часі та зберігати дані в JSON форматі. Будь-які зміни одразу ж синхронізуються між усіма девайсами, які використовують одну й ту ж саму базу даних. Іншими словами, оновлення інформації в сервісі Firebase відбувається миттєво.

Окрім сховища, Firebase також має функцію аутентифікації користувачів, тому всі дані передаються через захищене SSL з'єднання. Для аутентифікації є можливість вибрати будь-яку комбінацію email та пароля, будь то Facebook, Twitter, GitHub, Google, або інший сервіс.

Переваги цього сервісу:

- Перевагою цього сервісу є швидкість роботи. У пакеті Firebase зібрані прості та зрозумілі API, які дуже спрощують та прискорюють розробку мобільних додатків.
- Також заготовлена інфраструктура. Тобто не потрібно створювати складну інфраструктуру або працювати з декількома панелями управління. Замість цього ви зможете зосередитися на потребах користувачів.
- Використання статистики. В основі Firebase лежить безкоштовний аналітичний інструмент, розроблений спеціально для мобільних пристроїв. Google Analytics для Firebase збирає дані про дії користувачів та вживає відповідні заходи за допомогою додаткових функцій.
- Кросплатформеність. Firebase працює майже на будь-якій платформі завдяки пакетам розробника для Android, iOS, JavaScript і C ++.
- Масштабованість. Якщо мобільний додаток стане досить популярним і навантаження на нього буде зростати, вам не доведеться додавати зміни до коду сервера – Firebase зробить це за

вас. Багато функцій Firebase безкоштовні і залишаться такими незалежно від масштабу ваших проектів.

- Безкоштовна онлайн підтримка з використанням електронної пошти. Команда Firebase та фахівці з Google дадуть відповіді на будь-які ваші питання на таких ресурсах, як Stack Overflow та GitHub.

Firebase Authentication

Firebase надає простий у використанні SDK і готові бібліотеки призначені для для реалізації аутентифікації користувачів у вашому додатку. Він підтримує авторизацію як за допомогою email і пароля, так і за допомогою таких популярних сервісів, як Google, Facebook, Twitter чи GitHub.

Сервіс аутентифікації тісно інтегрується з іншими сервісами Firebase, використовує галузеві стандарти, такі як OAuth 2.0 і OpenID Connect, так що він може бути легко інтегрований з вашим бекендом. Firebase Authentication [5] SDK надає методи для створення і управління користувачами, які використовують адреси електронної пошти і паролі для входу в систему.

Основними перевагами Firebase Authentication є:

- Інтеграція з іншими функціями сервісу Firebase.
- Використання галузевих стандартів, таких як OAuth 2.0 і OpenID Connect спрощує інтеграцію з бекендом.
- Має два варіанти для розробки: FirebaseUI - повністю інтегроване універсальне рішення для виконання аутентифікації - або пакет Firebase Authentication SDK, який дозволяє вручну інтегрувати один або кілька методів входу в додаток.
- Безпечна та зручна авторизація забезпечуються за рахунок можливості виконувати вхід через акаунт Google та інші інтегровані системи.
- Зручна робоче середовище з миттєвим доступом до вашого сайту на будь-яких пристроях.
- Безпечний доступ до сервісів Google. Можливість зберігати файли на Google Диск, створювати заходи в Google Календарі і ділитися новинами зі своїми контактами прямо з програми.
- Оплата покупок в додатку.

Firebase Realtime Database

Мета сервісу Firebase Realtime Database[13] - дозволити обмін даними між кількома клієнтами, де "клієнт" може приймати форму додатків, що працюють на мобільних пристроях Android та iOS.

Основна мета системи - забезпечити безпечний, надійний і швидкий спосіб синхронізації даних з мінімальною кількістю зусиль кодування з боку розробника. Система баз даних також призначена для масштабування та підтримки мільйонів користувачів.

База даних називається "в режимі реального часу", оскільки швидкість, з якою синхронізуються дані між клієнтами, ймовірно, настільки близька до реального часу, наскільки це можливо (враховуючи фізичне обмеження передачі даних через Інтернет та бездротові з'єднання).

База даних у реальному часі також забезпечує збереження даних, зберігаючи дані локально, тим самим даючи змогу зберегти доступ до даних, навіть коли пристрій перебуває в автономному режимі. Коли відновлення буде відновлено, локальні дані автоматично синхронізуються та об'єднуються з віддаленими даними

Основними перевагами Realtime Database є:

- Синхронізація даних проводиться в реальному часі у форматі JSON. На базі даних Firebase Realtime представлено єдину обласну базу даних NoSQL, яка дозволяє зберігати та синхронізувати дані між вашими користувачами у режимі реального часу.
- Синхронізація в реальному часі дозволяє користувачеві отримати доступ до своїх даних з будь-якого пристрою.
- Створення безсерверних додатків. Realtime Database постачається з мобільним і веб-SDK, тож можна створювати мобільні додатки без необхідності наявного серверу.
- Оптимізований для автономного використання. Коли ваші користувачі переходять в автономний режим, SDK базує дані в режимі реального часу, використовуючи локальний доступ до пристрою для обслуговування та збереження вимірів. Коли пристрій підключається до мереж, локальні дані автоматично синхронізуються.

- Сильна користувальницька безпека. Базові дані в режимі реального часу інтегруються з аутентифікацією Firebase для забезпечення простої та інтуїтивної аутентифікації для розробників. Ви можете використовувати нашу декларативну модель безпеки, щоб вирішити доступну для основної ідентифікатора користувача або відповідність шаблонів вашим даним.

Розглянуті вище сервіси Firebase були вибрані через те, що вони досить гарно пристосовані до Android клієнта і це забебечує простоту та швидкість роботи з ними. Також вони моють дуже широкий спектр інструментарія, що викреслює необхідність реалізації додаткових компонентів.

Android Studio

Для розробки додатку було використано середовище розробки Android Studio.

Android Studio[12] є інтегрованим середовищем розробки для роботи з платформою Android. Головним конкурентом даної утиліти є інше середовище розробки під назвою Eclipse.

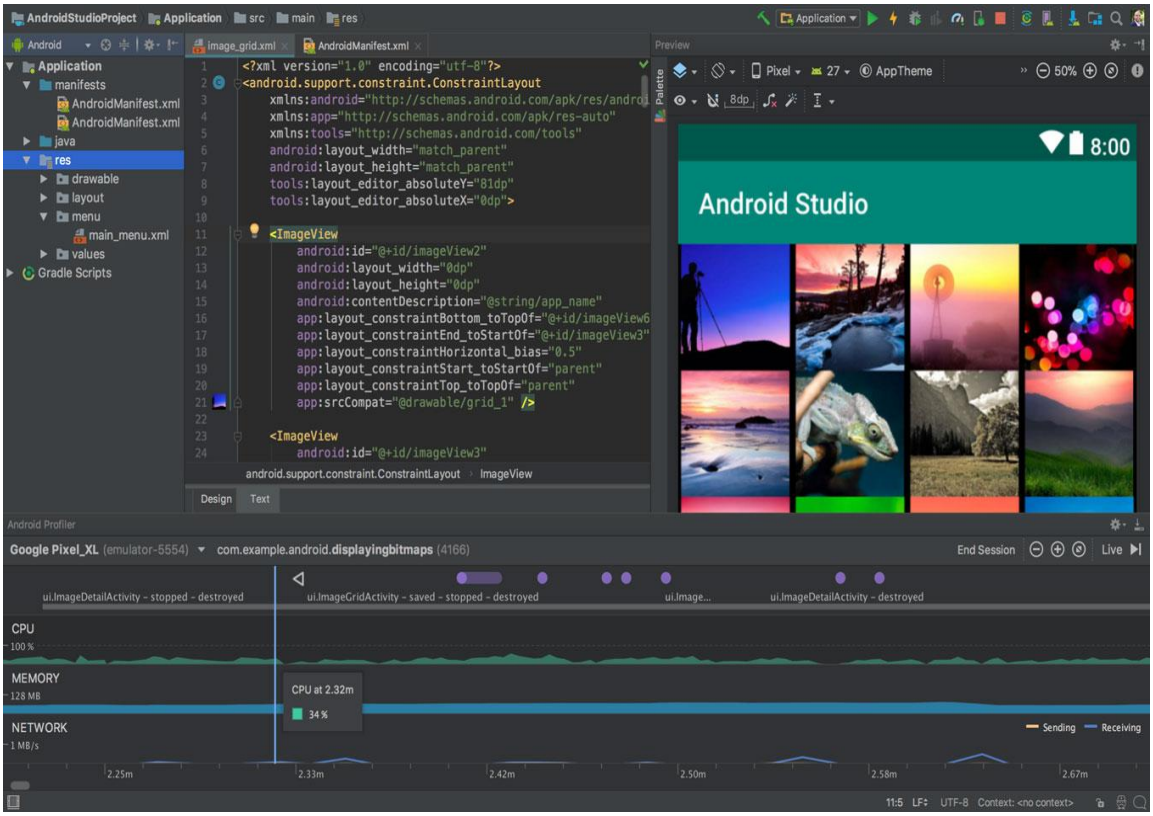


Рисунок 2.3 – Вигляд IDE Android Studio

У програму вбудований емулятор, що дозволяє перевірити коректну роботу програми на пристроях з різними екранами, з різними співвідношеннями сторін. Особливо актуальною ця функція стала після входу в тренди смартфонів, в яких встановлені екрани із співвідношенням сторін 18: 9.

Відмітна особливість емулятора - перегляд приблизних показників продуктивності при запуску програми на найпопулярніших пристроях.

Середовище розробки для додатків Android Studio останньої версії стала по справжньому зручною навіть для початківців розробників. У програмі реалізовані всі сучасні засоби для упаковки коду, його маркування. Затребувана багатьма творцями ПО функція Drag-n-Drop, що полегшує перенесення компонентів в середу розробки безпосередньо.

Локалізація додатків стає значно простіше з функцією SDK, яка також входить до переліку достоїнств Android Studio.

Переваги даної утиліти:

- Android Studio підтримує роботу з декількома мовами програмування, такими як: C / C ++, Java, Kotlin.
- Наявний зручний для робти редактор коду;
- В IDE можно розробляти не тільки програмне забезпечення для смартфонів / планшетів, а й для портативних ПК, приставок для телевізорів Android TV, пристроїв Android Wear;
- Дуже зручною функцією є рефакторинг вже готового коду;
- Розробка програм для Android N - самої останньої версії операційної системи;
- Можлива перевірка вже створеного додатку на предмет помилок;
- Дуже великий набір інструментарію для тестування кожного елемента програми;
- Для недосвідчених розробників спеціально створено посібник з використання Android Studio, розміщене на офіційному сайті утиліти.

2.4 Налаштування середовища розробки

Так як розробку системи виконується на мові програмування Java, то спочатку треба скачати та всановити Java SDK. Тобто, що нам потрібно:

1. Завантажити та встановити Java SDK.

2. Завантажити та встановити Android Studio.

Після встановлення необхідного програмного забезпечення, треба створити новий проект для подальшої можливості розробки і налаштувати його:

1. Спочатку вибираємо мову програмування, яка буде використовуватися для реалізації сервісу. В моєму випадку це буде Java. Далі вибираємо мінімальну версію Android API 23: Android 6.0 (Marshmallow).

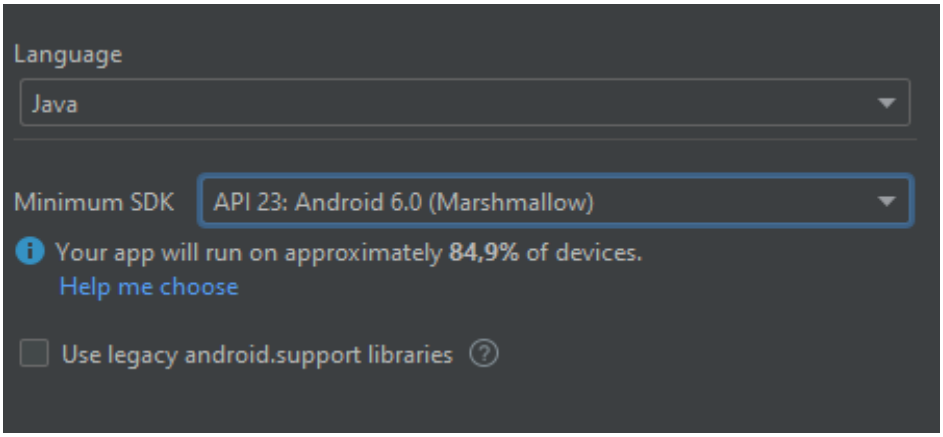


Рисунок 2.3 – Встановлення мови програмування та версії API

2. Далі треба налаштувати файл збірки проекту під назвою build.gradle. В нього потрібно додати всі необхідні бібліотеки, які будуть використовуватися для роботи системи.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.google.firebase:firebase-core:17.4.1'
    implementation 'com.google.firebase:firebase-auth:19.3.1'
    implementation 'com.google.firebase:firebase-storage:19.1.1'
    implementation 'com.google.firebase:firebase-database:19.3.0'
    implementation 'com.google.android.material:material:1.1.0'
    implementation 'com.squareup.picasso:picasso:2.71828'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.recyclerview:recyclerview:1.2.0-alpha03'
    implementation 'com.android.support:design:28.0.0'
    implementation 'androidx.navigation:navigation-fragment:2.2.2'
    implementation 'androidx.navigation:navigation-ui:2.2.2'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.annotation:annotation:1.1.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

Рисунок 2.3 – Вигляд імплементованих бібліотек у файлі build.gradle

3. В даному проекті замість серверу буде використовуватися сервіс Firebase. Для його використання потрібно створити новий проект на консолі розробника.

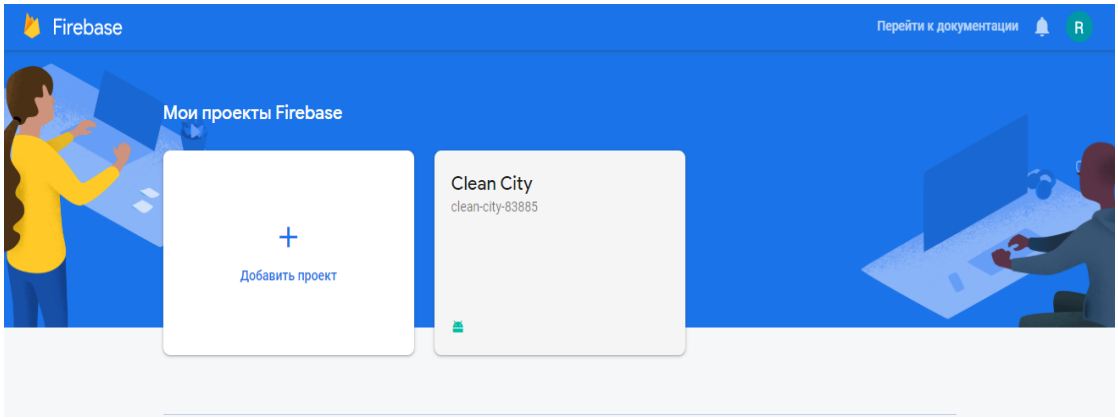


Рисунок 2.3 – Створення нового проекту сервісу Firebase

4. Після створення нового проекту, на треба налаштувати ті сервіси, які ми вирішили використовувати ри розробці, а саме: Firebase Authentication, Firebase Realtime Database.

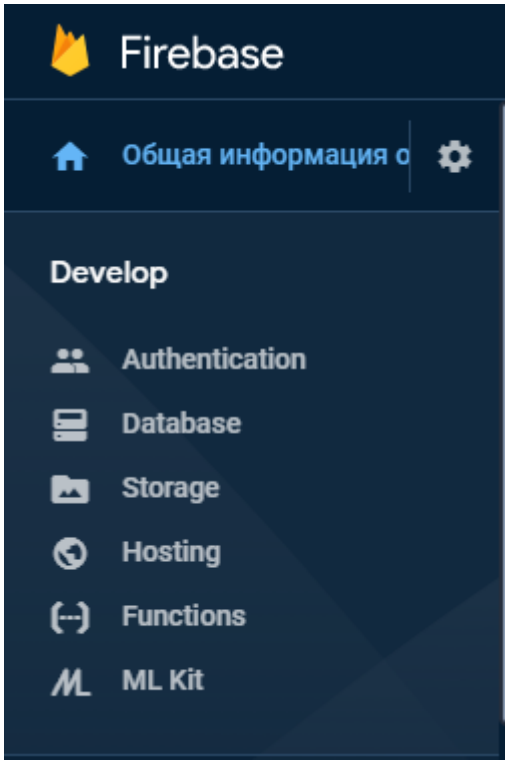


Рисунок 2.3 – Меню вибори сервісів для налаштування

5. Наступним кроком буде синхронізація самого додатку с сервісом Firebase. Щоб це зробити необхідно завантажити JSON файл з інструкції підключення та скопіювати його у папки самого проекту.

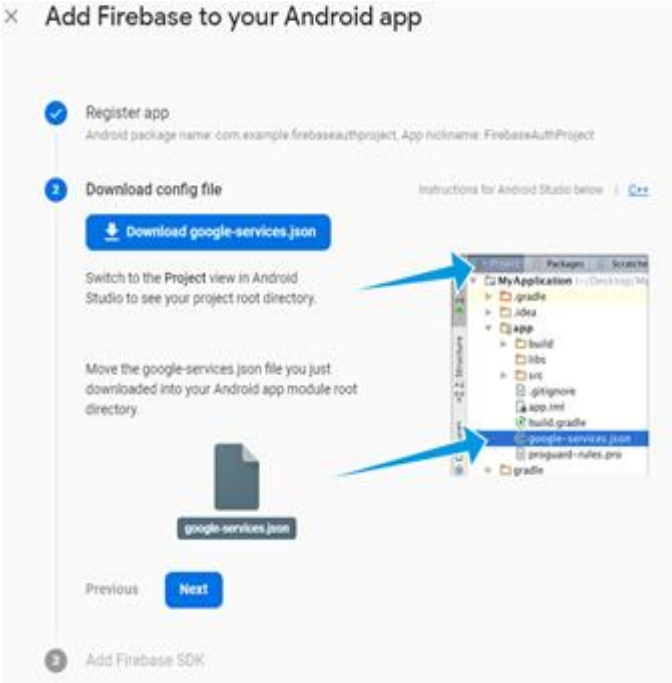


Рисунок 2.3 – Завантаження JSON файлу з інструкції

6. Запустити додаток і перевірити, чи немає помилки підключення між додатком та сервісом Firebase.

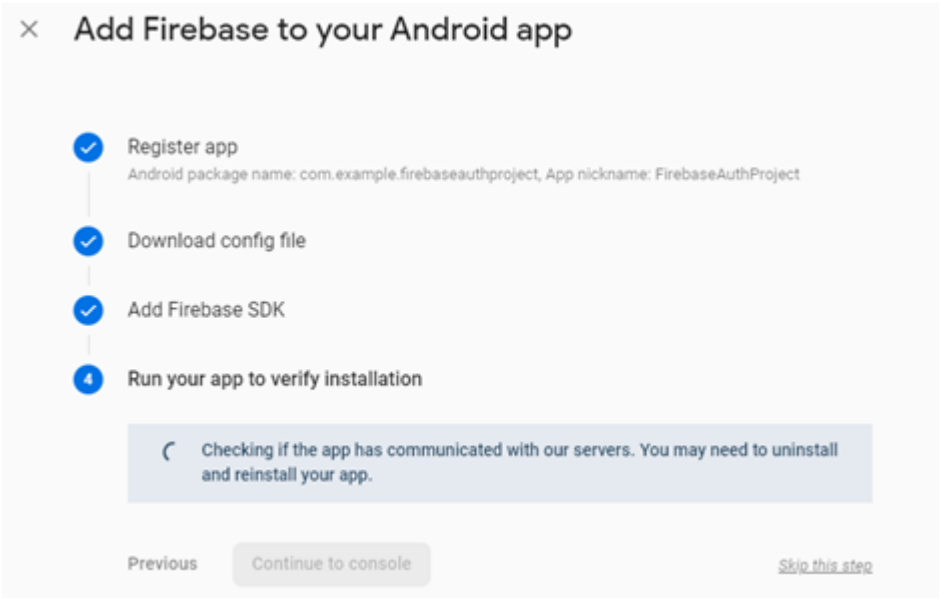


Рисунок 2.3 – Перевірка підключення додатку до сервісу

Після всіх цих пророблених маніпуляцій ми підключили наш Android проект до сервісу Firebase.

2.5 Огляд основних використанних бібліотек та віджетів

При розробці цієї інформаційної системи біли використанні наступні бібліотеки:

Бібліотека **ConstraintLayout**
implementation 'androidx.constraintlayout:constraintlayout:1.1.3'

ConstraintLayout[8] - це новий вид layout (шаблону екрану), який ви можете використовувати в ваших Android-додатках. Ця компоновка має зворотну сумісність аж до API Level 9. Мета бібліотеки ConstraintLayout - зменшити кількість ієрархій layout, поліпшити продуктивність layout, а також замінити роботу з RelativeLayouts. Вона сумісна з іншими компонуваннями, так що вони можуть прекрасно гармоніювати один з одним.

В Android Studio 2.3 і вище в шаблонах за замовчуванням тепер використовується ConstraintLayout.

Якщо в режимі дизайну обраний даний компонент, то на панелі інструментів вам доступні кілька кнопок.

View Options з пунктами Show Constraints - виводить задані обмеження в режимах перегляду і розкладки. В одних випадках цей режим перегляду корисний, в інших немає. При великій кількості обмежень ця кнопка виводить занадто багато інформації, Show Margins, Fade Unselected views.

Turn On Autoconnect - при включенні режиму Autoconnect обмеження будуть автоматично налаштовуватися при перетягуванні уявлень в область попереднього перегляду. Студія намагається вгадати, які обмеження повинен мати компонент, і створювати їх у міру необхідності

Default Margins - стандартне значення для відступів. Можете встановлювати окремо для кожного компонента. Вибрали компонент, встановили значення, потім знову вибрали інший компонент і встановили інше значення

Clear All Constraints - видаляє всі обмеження з макета

Infer Constraints - автоматично створює обмеження. Спрацьовує тільки при натисканні кнопки. Функціональність Autoconnect спрацьовує кожного разу, коли в файл макета додається новий компонент

GuideLines з двома опціями: Add Vertical GuideLine і Add Horizontal GuideLine.

Бібліотека **Picasso**

implementation 'com.squareup.picasso:picasso:2.71828'

Picasso[9], призначена для асинхронної завантаження зображень з мережі, ресурсів або файлової системи, їх кешування і відображення.

Ось як просто завантажити картинку з серверу:

```
Picasso.with(context)
  .load(url)
  .placeholder(R.drawable.user_placeholder)
  .error(R.drawable.user_placeholder_error)
  .into(imageView);
```

Ви вказуєте адресу картинки (url), заглушку (placeholder), заглушку для помилки після трьох невдалих спроб завантаження (error) і в методі into () вказуєте компонент ImageView, в який завантажуєте зображення.

При завантаженні картинка кешируєтся і при повторному запиті на скачування бібліотека може дістати картинку з кешу, а не завантажувати з інтернету, що прискорює роботу програми. Якщо кеш буде переповнений або видалено користувачем, то картина знову скочується з мережі. Дуже зручно.

Якщо ви зберігаєте великі картинки в ресурсах або на зовнішньому накопичувачі, то рекомендується використовувати окремий процес для завантаження. Бібліотека вже налаштована на роботу в асинхронному режимі, тому ви можете використовувати її і в цих випадках.

Бібліотека **AppCompat**

implementation 'androidx.appcompat:appcompat:1.1.0'

Бібліотека AppCompat[10] (aka ActionBarCompat) з'явилася як порт нового ActionBar API з Android 4.0 на пристрої з Gingerbread. Вона представила загальний шар API поверх бекпортірованной або стандартної реалізації. AppCompat v21 ж приносить API і набір можливостей з Android 5.0

У цій версії Android з'явився новий віджет Toolbar. Він являє собою узагальнення шаблону ActionBar і дає більше контролю та гнучкості. Toolbar - це елемент view в загальній ієрархії, що спрощує реалізацію його взаємодії з іншими віджетами, анімації і реакції на події прокрутки. Так само ви можете використовувати його як ActionBar вашої активності, а це значить, що стандартні елементи меню дій будуть показуватися в ньому.

Ви, ймовірно, вже користувалися якийсь час оновленою версією AppCompat. Ця бібліотека була включена в оновлення різних програм Google, що вийшли в останні тижні, в тому числі Play Маркет і Play Преса. Крім того, він був інтегрований в додаток Google I / O, зображене вище, яке має відкритий вихідний код.

Віджет **RecyclerView**

implementation 'androidx.recyclerview:recyclerview:1.2.0-alpha03'

RecyclerView[11] - це більш вдосконалена і гнучка версія ListView.

У моделі RecyclerView кілька різних компонентів працюють разом для відображення ваших даних. Загальний контейнер для вашого

користувальницького інтерфейсу - це об'єкт RecyclerView, який ви додаєте до свого макета. RecyclerView заповнює себе поданнями, наданими менеджером компонування, який ви надаєте. Ви можете використовувати один із наших стандартних менеджерів макетів (таких як LinearLayoutManager або GridLayoutManager) або ж реалізувати свій власний.

Перегляди у списку представлені об'єктами власників перегляду. Ці об'єкти є екземплярами класу, який ви визначаєте, розширюючи RecyclerView.ViewHolder. Кожен власник перегляду відповідає за показ одного елемента з видом. Наприклад, якщо ваш список містить музичну колекцію, кожен власник перегляду може представляти один альбом. RecyclerView створює лише стільки власників перегляду, скільки потрібно для відображення на екрані частини динамічного вмісту плюс кілька додаткових. Коли користувач прокручує список, RecyclerView приймає перегляди поза екраном і повертає їх до даних, що прокручуються на екран.

Об'єктами власників подання керує адаптер, який ви створюєте, розширюючи RecyclerView.Adapter. Адаптер створює необхідні тримачі для перегляду. Адаптер також прив’язує власників перегляду до своїх даних. Це робиться шляхом присвоєння власника перегляду в позицію та виклику методу onBindViewHolder () адаптера. Цей метод використовує позицію власника перегляду, щоб визначити, яким повинен бути вміст, виходячи з його списку.

Ця модель RecyclerView робить багато оптимізації, тому вам не доведеться стикатися з такими проблемами:

- Коли список вперше заповнюється, він створює та прив'язує деяких власників перегляду з обох боків списку. Наприклад, якщо представлення відображає позиції списку від 0 до 9, RecyclerView створює та зв'язує ці власники перегляду, а також може створювати та прив'язувати утримувач перегляду до позиції 10. Таким чином, якщо користувач прокручує список, наступним елементом є готовий до показу.
- Коли користувач прокручує список, RecyclerView створює нові власники перегляду, якщо це необхідно. Це також зберігає власників зображень, які прокручуються за межі екрана, тому їх можна повторно використовувати. Якщо користувач перемикає напрямок, в якому прокручується, власників перегляду, які були прокручені з

екрана, можна повернути назад. З іншого боку, якщо користувач продовжує прокручувати в тому ж напрямку, власники перегляду, які довше були поза екраном, можуть бути повторно пов'язані з новими даними. Власника перегляду не потрібно створювати або переглядати його; натомість додаток просто оновлює вміст перегляду, щоб відповідати новому елементу, до якого він був прив’язаний.

- Коли відображаються елементи змінюються, ви можете повідомити про це адаптер, зателефонувавши у відповідний метод RecyclerView.Adapter.notify… (). Вбудований код адаптера потім відновлює лише уражені елементи.

Віджет **CardView**

implementation "androidx.cardview:cardview:1.0.0"

Новий віджет під назвою CardView, який, по суті, можна розглядати як FrameLayout із закругленими кутами та тінню на основі його висоти. Зауважте, що CardView обгортає макет і часто є контейнером, який використовується в макеті для кожного елемента в ListView або RecyclerView.

2.6 Опис алгоритму функціональних модулів програми

Ключовим компонентом для створення візуального інтерфейсу в додатку Android є activity (активність). Нерідко activity асоціюється з окремим екраном або вікном додатка, а перемикання між вікнами буде відбуватися як перехід від однієї активності до іншої. Додаток може мати одну або кілька activity.

Всі додатки Android мають строго визначений системою життєвий цикл [3]. При запуску користувачем додатку система дає цьому додатку високий пріоритет. Кожна програма запускається у вигляді окремого процесу, що дозволяє системі давати одним процесам вищий пріоритет, на відміну від інших. Завдяки цьому, наприклад, при роботі з одним додатком не блокувати вхідні дзвінки. Після припинення роботи, система звільняє всі пов'язані з нею ресурси і закриває додаток.

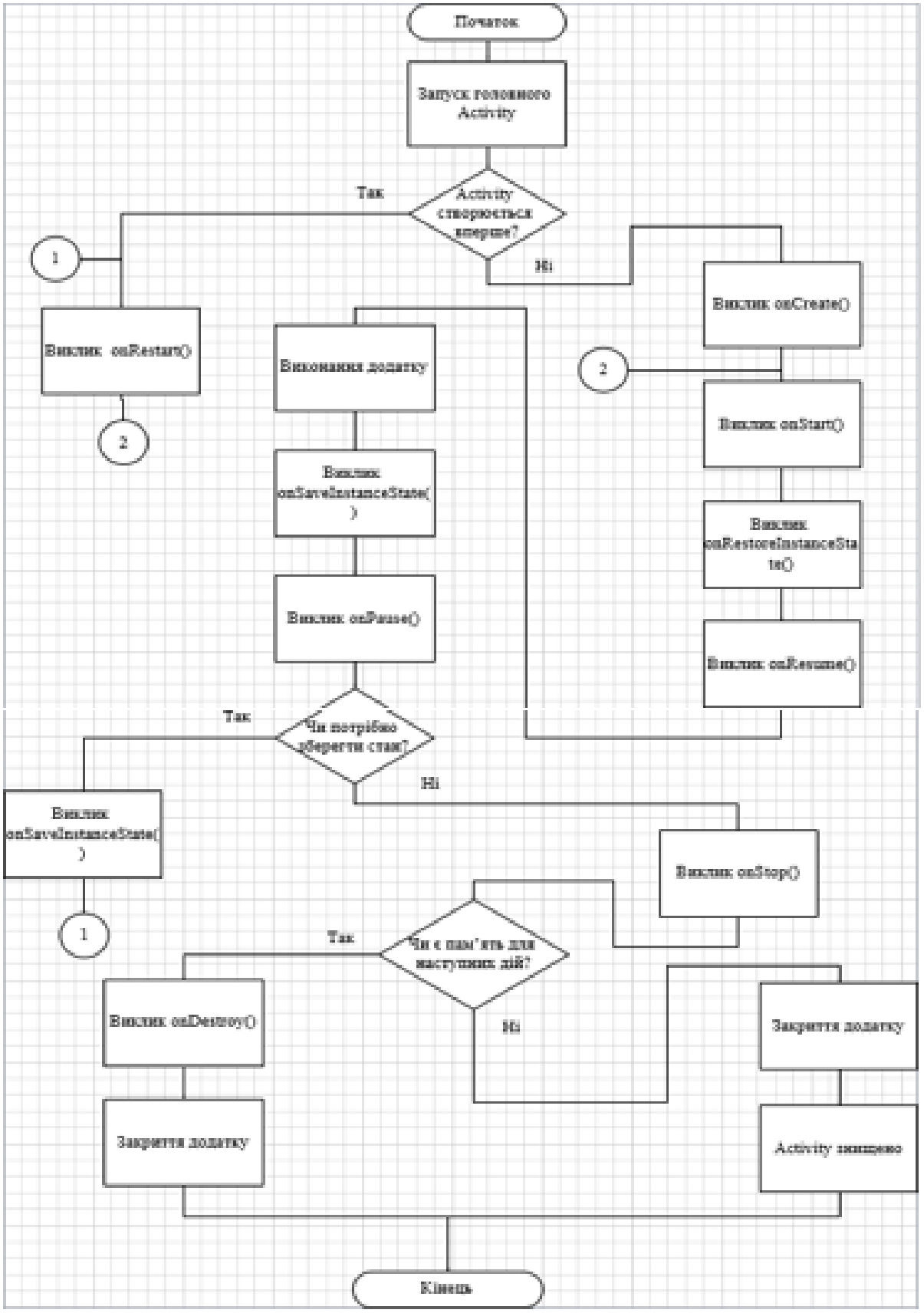


Рисунок 2.5 – Алгоритм роботи Android додатку

2.6.1 Опис роботи основних методів додатку

Метод onStart(). Після методу onCreate () завжди викликається метод о nStart (), але перед onStart () не обов'язково йде onCreate (), так як onStart() мо жна викликати для відновлення роботи припиненої. При виклику onStart () ві кно стає видимим для користувача, але він ще не може взаємодіяти з ним.

Метод onResume(). Метод викликається після onStart (). У цей момент користувач вже може взаємодіяти з даною активністю та вводити свої. Також в даному методі запускають відтворення анімації, аудіо та відео. Може викли катися після onPause ().

Викликається завжди, коли активність виходить на передній план та стає дост упною для користувача. Таким чином onResume () треба реалізовувати для іні ціалізації компонентів програми, відновленні зупинених процесів в методі on Pause().

Тут слід розміщувати швидкий та легкий код, для того щоб анімація ви ходу активності на передній план була більш швидкою.

Метод onPause(). Коли користувач вирішує перейти до роботи з новим вікном, система викликає цей метод. Відбувається призупинення та згортання активності, зберігаються незафіксовані дані. Зупиняється анімація, відтворен ня аудіо та відео для розвантаження процесора. Від методу onPause() можна п ерейти до двох інших методів, таких як: onResume (), або onStop ().

Перед тим як переходити до іншого методу спочатку слід зберегти дані, бо пі сля виконання даного методу активність може прервати свою роботу без попе редження.

Метод onStop(). Після виклику вікно стає невидимим для користувача. onStop() викликається при запуску іншої активності, яка перекрила вікно пот очного екрану, або ж для знищення даної активності. Цей метод завжди супро воджується викликом методу onRestart (), якщо активність перезапускається, щоб взаємодіяти з користувачем, або викликом методу onDestroy (), якщо акт ивність знищується.

Коли активність зупиняється, усі об'єкти зберігаються в пам'яті і віднов люються, коли активність відновлює свою роботу. Система також відстежує п оточний стан для кожного уявлення, тому, якщо наприклад ввести текст в текс тове поле, то його зміст збережеться.

Метод onRestart(). Викликається після onStop() для повернення вікна в пріорітетний режим. Тобто після того, як активність була зупинена і знову за пущена користувачем. Завжди супроводжується викликом методу onStart ().

Метод onDestroy(). Цей метод викликається за допомогою виклику методу finish() для завершення роботи активності. У іншому разі onDestroy() викликається автоматично коли системі потрібні ресурси. Це останній запит, який отримує активність від системи. Якщо певн а активність знаходиться на вершині стеку, але невидимо для користувачева і система вирішує завершити це вікно, викликається метод onDestroy().

2.7 Архітектура проекту

2.7.1 Діграма класів системи

При проектуванні даної системи та її графічного інтерфейсу, було виділено 4 основних екрани, кожен з яких матиме свою складну логіку. Це такі екрани як: CrimListFragment (екран виводу всього списку проблемних місць), CrimeFragment (екран додавання запису до бази даних), SignInFragment (екран реєстрації) та MapFragment (екран перегляду Google карти). Оскільки всі ці екрани мають складну логіку, я розробив таку архітектуру, яка зображена на діаграмі класів.

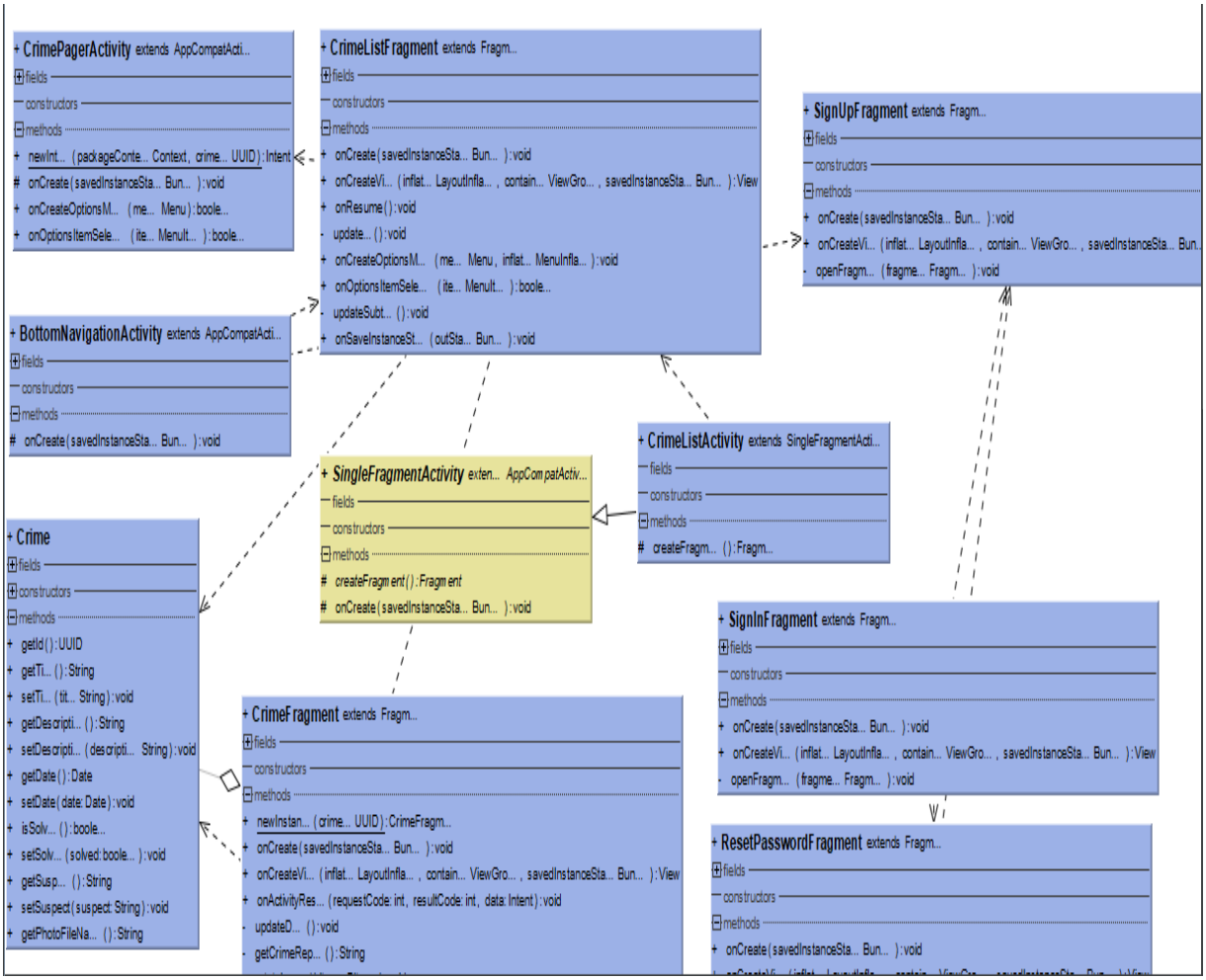


Рисунок 2.7 – Діаграма класів

Окрім класів які зображені вище існує ще один клас. Клас CrimeLab:

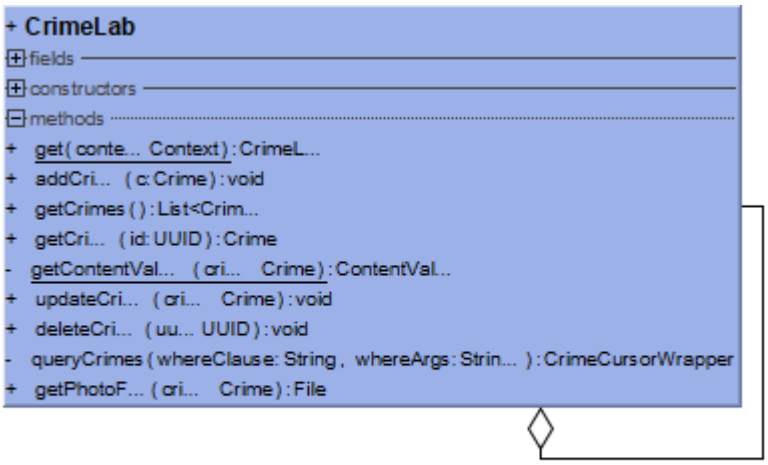


Рисунок 2.7 – Діаграма класів, клас CrimeLab

Клас CrimLab не взаємодіє з іншими класами, бо інші класи викликають його через неявні виклики. Він забезпечує оновлення інформації про записи в системі.

Клас BottomNavigationActivity забезпечує перехід між основними екранами додатку.

Клас Crime являє собою клас моделі системи. Він зберігає в собі всю логіку, яка пов’язана з фіксуванням проблемної ділянки.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі було визначено основні функціональні можливості системи та вибрано інструменти для реалізації даного дадатку. Я представив основні технолгії для розробки системи, які забезпечують спрощення створення цього додатку та збільшують надійність даної системи.

Також було пошагово показано налаштування середовища розробки додатку, оглянуто основні бібліотеки та віджети, які використані в розробці проекту.

Був представлений алгоритм роботи самого Android додатку, а також описаний кожен метод, який в ньому використовується. Це дозволить більш гарно уявити, як працює дана інформаційна система під капотом.

РОЗДІЛ 3.
ІНСТРУКЦІЯ З ВИКОРИСТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Авторизація користувача

При запуску програми користувача буде зустрічати вікно авторизації, де йому буде потрібно зареєструватися.

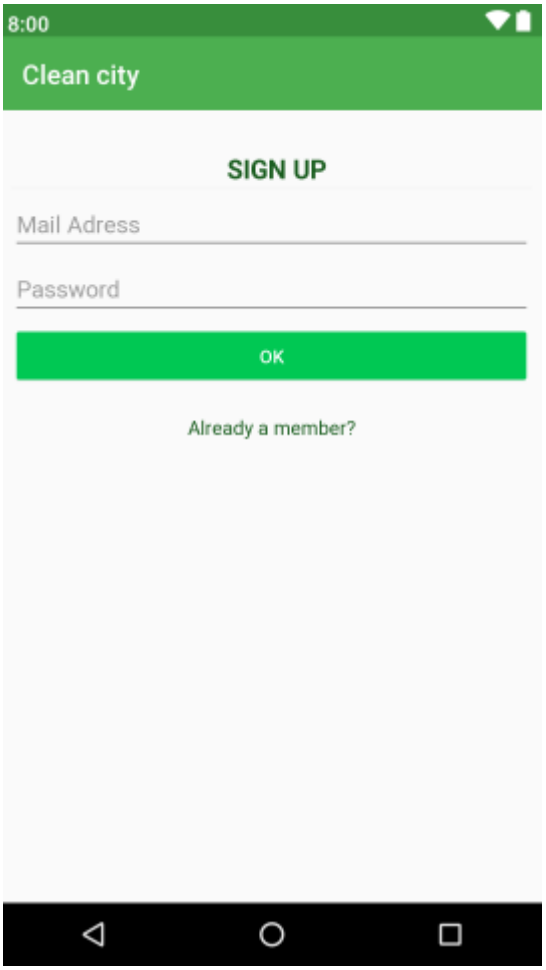


Рисунок 3.1 – Вікно реєстрації користувача

Адрес електронної пошти повинен бути валідним, а пароль - не меншим ніж на 8 символів. Після реєстрації користувача буде переправлено на вікно створення профіля. В якому йому буде потрібно ввести ім'я, прізвище та номер телефону. Також за бажанням можно додати фото профіля.

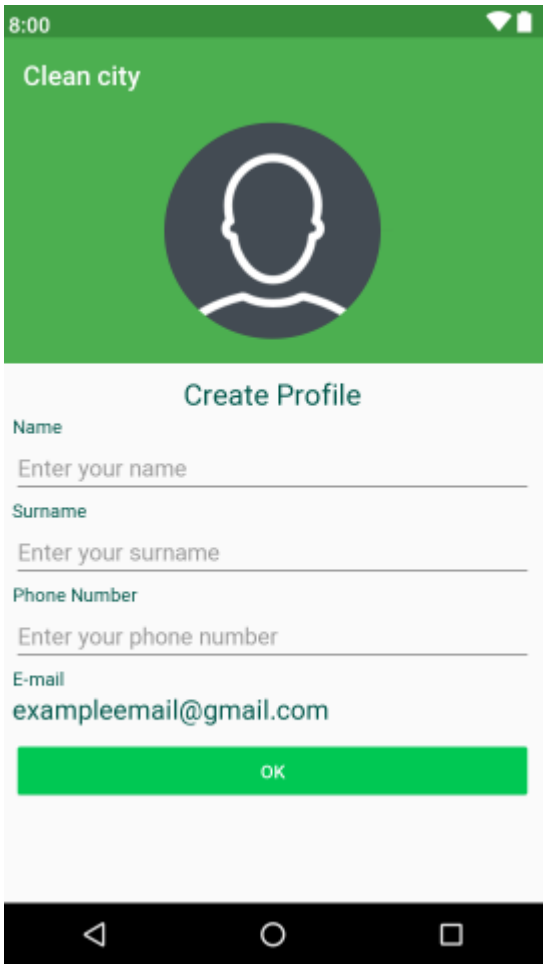


Рисунок 3.1 – Вікно створення профіля

Якщо ж користувач вже зареєстрований, то йому потрібно натиснути кнопку «Already a member?» та його перенаправить на вікно входу в систему. В якому потрібно заповнити два поля, та ввести свої дані, які були вказані при реєстрації.

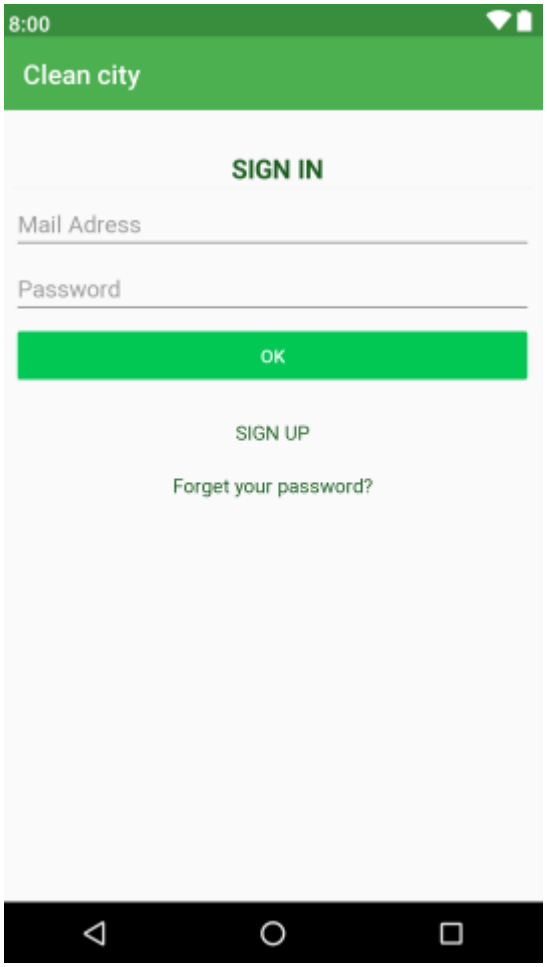


Рисунок 3.1 – Вікно входу до профіля користувача.

Трапляються такі ситуації, коли користувач може забути свій пароль для входу до особистого профілю. Для таких випадків в моєму додатку є кнопка «Forget your password?» після натискання на яку відкривається вікно для відновлення паролю.

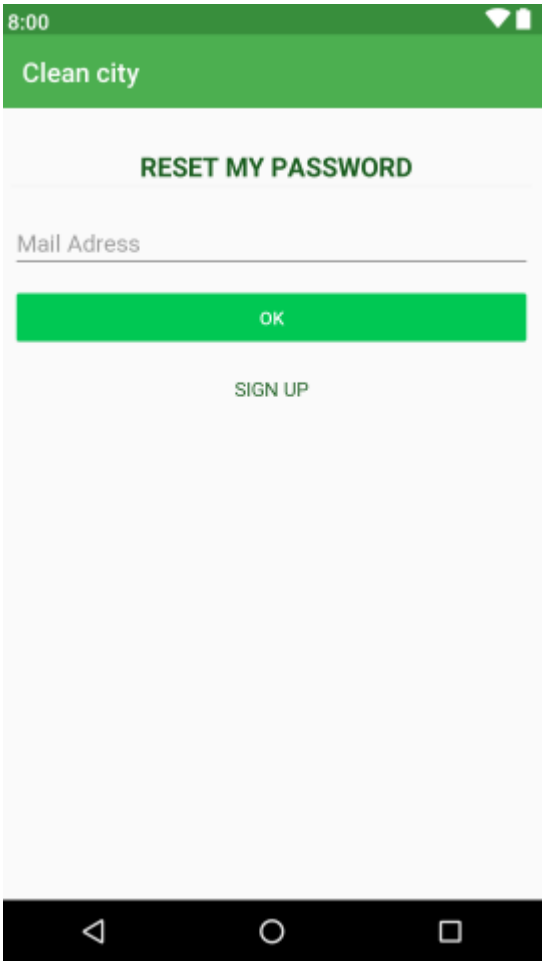


Рисунок 3.1 – Вікно відновлення паролю

У цьому вікні користувачу потрібно ввести свою електронну адресу на яку прийде лист з новим паролем, який людина зможе використати для входу у свій профіль.

3.2 Список забруднених місць та ділянок

Після того, як користувач увійшов у свій профіль, додаток перенаправляє його на вікно під назвою «List», воно показує список усіх забруднених місць та ділянок, які присутні в базі даних на цей момент часу. Але окрім списку, користувач має доступ ще до двох вікон, це: Map та Profile.

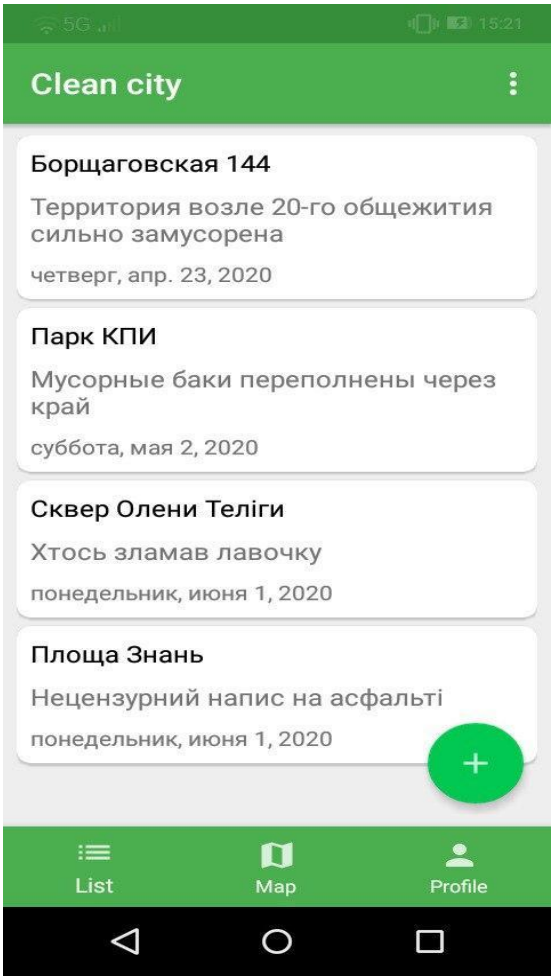


Рисунок 3.2 – Список забруднених місць та длянок

Головне вікно також має кнопку за допомогою якої можно створювати власний запис. Вона розташована в правій нижній частині екрану.

3.3 Вікно створення нового запису системи

Після того, як користувач натискає кнопку призначену для додавання нового запису, система переправляє його до вікна де він може це зробити. Вікно створення власного запису виглядає наступним чином:

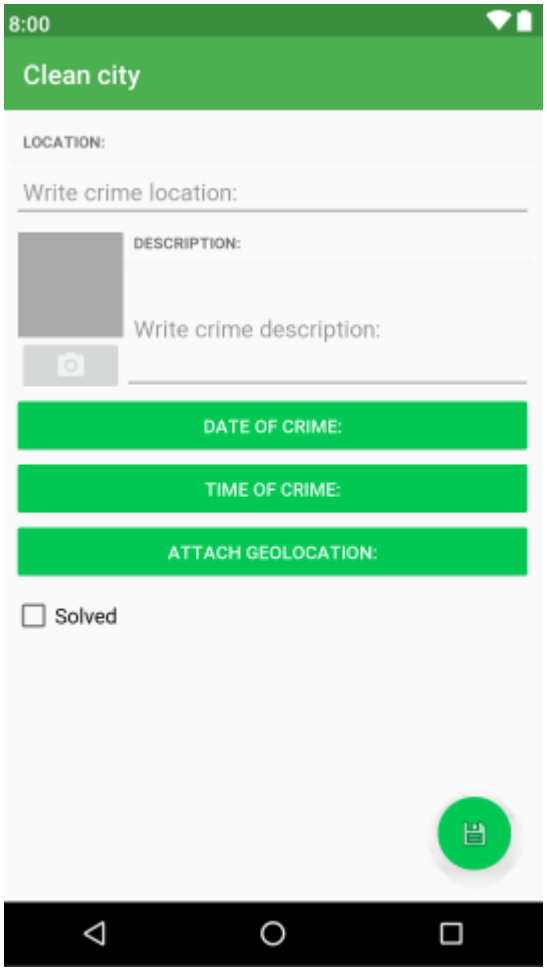


Рисунок 3.3 – Вікно створення власного запису

Розглянемо функціонал даного екрану. Для того щоб додати власний запис до системи користувачу спочатку потрібно заповнити два поля: «Location» та «Description». Інформація, яка буде вказана в них відображається у головному списку системи, для того щоб можно було дізнатись де знаходиться проблемна ділянка та її короткий опис.

Наступним кроком буде додовання дати та часу створення власного запису, які теж будуть відображатися в головному списку.

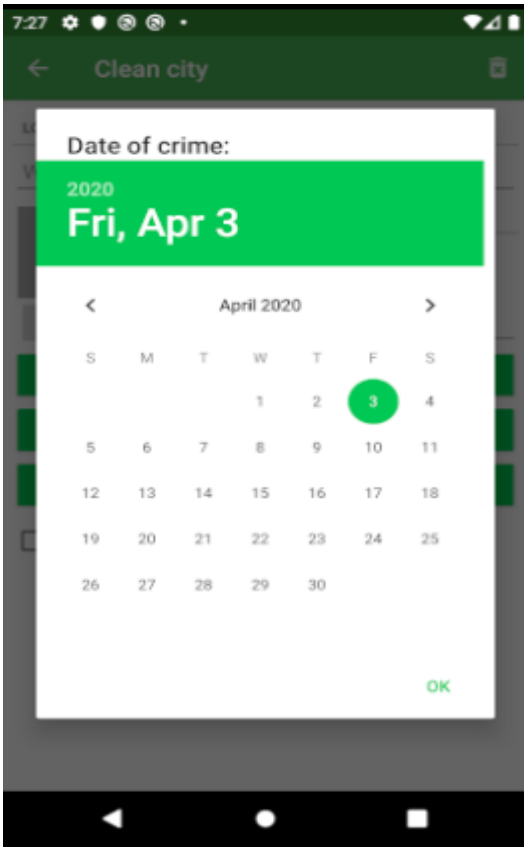


Рисунок 3.3 – Додавання дати

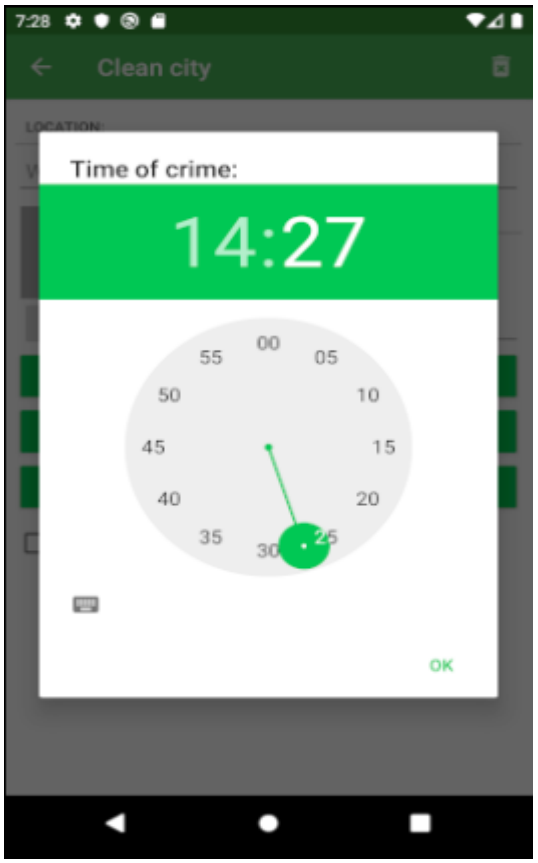


Рисунок 3.3 – Додавання часу

Наступним кроком буде фіксування проблемної ділянки за допомогою створення фотографії данного місця. Для того щоб це зробити, потрібно натиснути кнопку з фотокамерою в центрі, після чого буде визвано системну службу «Камера» для створення фото.

Рисунок 3.3 – Фіксування ділянки за допомогою камери телефону

Після всих цих нескладних маніпуляцій залишається тільки прикріпити геолокацію. Для того щоб можно було відстежити проблемну ділянку за допомогою маркера, який буде відображатися на Google Карті вікна «Map».

3.4 Вікно перегляду всіх забруднених місць на Google Карті

Для більш простої навігації по місту та відстеження проблемних ділянок, які потребують клінінгу, я створив спеціальне вікно з Google Картою, для відстеження усіх записів системи. Достатньо лише перейти до вікна «Map» і користувач зможе переглянути маркери усіх наявних записів системи.

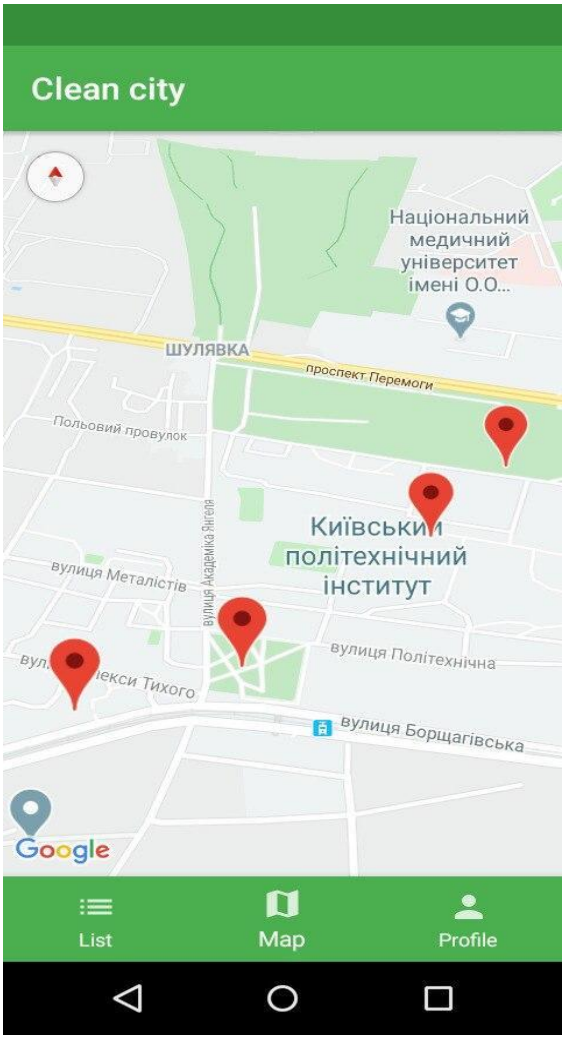


Рисунок 3.4 - Вікно перегляду всіх забруднених місць

Доступна функція відображення короткого опису та місцязнаходження, при натисканні на маркер. Ці поля задаються у вікні створення запису та відображаються у головному списку та на карті. На мою думку це буде спрощувати користування додатком та сприяти більш гарному уявлянню загальної картини чистоти міста.

ВИСНОВКИ ДО РОЗДІЛУ 3

В цьому розділі дипломної роботи я продемонстрував роботу інформаційної системи та коротко описав інструкцію користування додатком. Також було описано інтерфейс програми та основні дії, які потрібні для використання функціоналу даної системи. Розроблений додаток враховує поставлені задачі та надає можливість використовувати його найбільш ефективно.

Розлянувши даний розділ, можна сказати, що основним фукціоналом даної системи є:

- Регістрація та вхід
- Перегляд списку проблемних ділянок
- Додавання власного запису
- Додавання опису, фото, дати та геолокації до власного запису
- Перегляд інформації в обраному записі
- Перегляд Google карти з відображеними на ній проблемними місцями
- Перегляд особистого профілю
- Відновлення паролю
- Вихід з профілю

ВИСНОВКИ

В цьому дипломному проекті було розроблено інформаційну систему, з допомогою якої люди зможуть підтримувати чистоту свого міста та приміських зелених зон. Метою проекту було створення додатку на базі операційної системи Android, який буде простим та доступним для використання.

Було проведено аналіз предметної області, визначено цілі та задачі розробки, які повинен вирішувати даний додаток. Розглянувши всі існуючи аналоги систем, їх сильні та слабкі сторони, був розроблений функціонал системи, який дозволить виконувати поставлене завдання.

Були описані та обгрунтовані основні інструменти вибрані для розробки цієї системи під платформу Android. Дослідження та огляд технологій, використаних в процесі розробки, дозволили зробити надійне та постійне функціонування системи. Також було проаналізовано допоміжні бібліотеки та віджети, використані у розробці. А за допомогою сервісу Firebase, було виконано імітацію клієнт-серверної взаємодії та надано можливість для авторизації та регістрації користувачів.

Була розроблена блок-схема алгоритму роботи додатку, яка дає можливість зрозуміти, як ця система працює.

Систему було добре описано, вона має інтуїтивно простий інтерфейс, який не доставить труднощив у використанні та задовольняє всі вимоги поставленні перед початком виконання роботи.

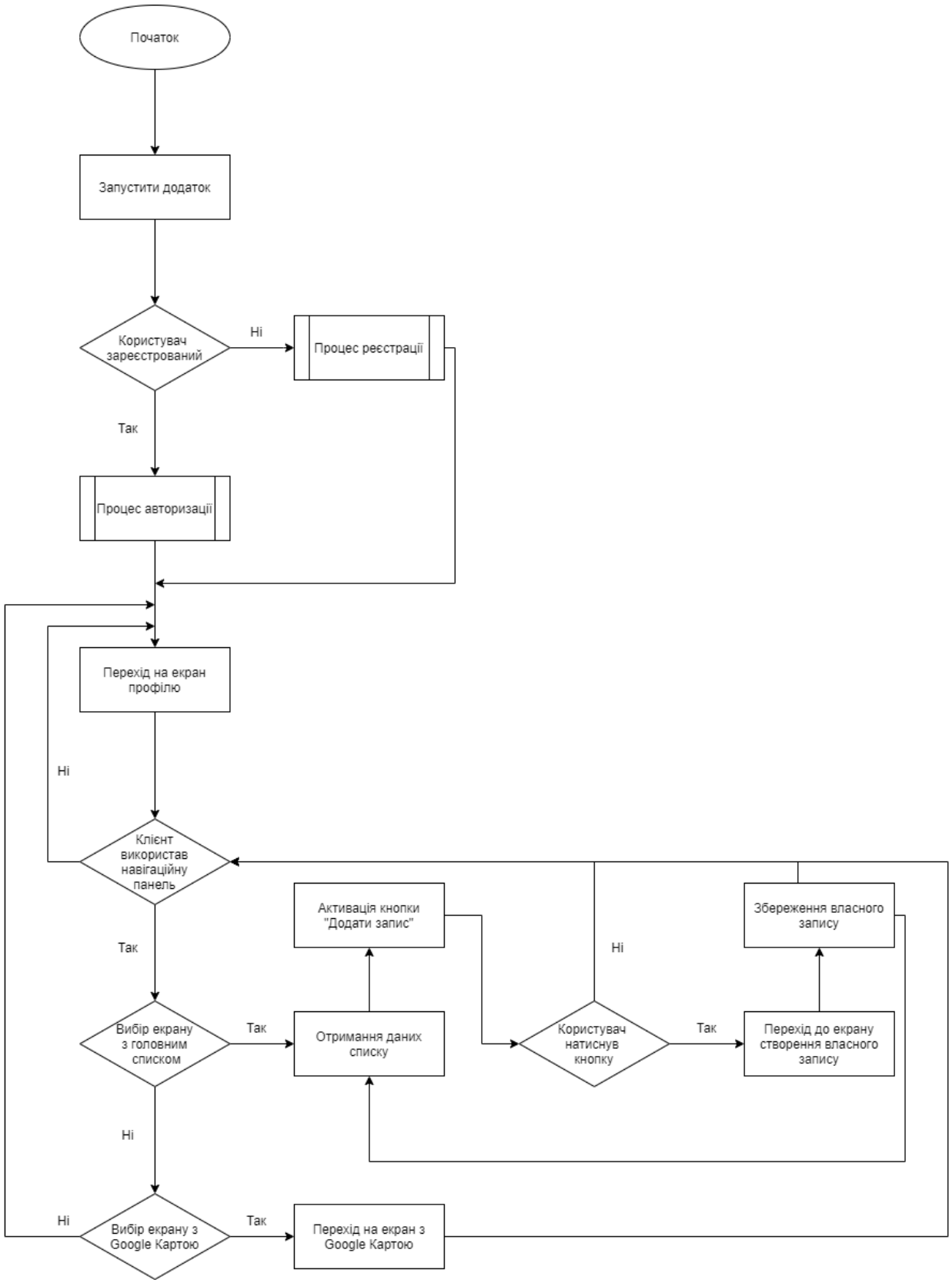
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What are the causes of increasing waste? [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.quora.com/What-are-the-causes-of-increasing-waste.
2. Information Systems Evolution (IS) [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://www.slideshare.net/SemputLa/information-systems-evolution-is.
3. Жизненный цикл приложения на Android [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://developer.alexanderklimov.ru/android/theory/lifecycle.php.
4. Firebase [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://ru.bmstu.wiki/Firebase#.D0.9E.D1.81.D0.BE.D0.B1.D0.B5.D0.BD.D0.BD.D0.BE.D1.81.D1.82.D0.B8.
5. Firebase Authentication [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://firebase.google.com/docs/auth.
6. Мусорное загрязнение – главная проблема мегаполисов [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://www.gaw.ru/html.cgi/txt/gl/mir2/musornoe-zagryaznenie-glavnaya-problema-megapolisov.htm.
7. АИС «Народный контроль» [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://www.cmirit.ru/ru/information-systems/ais-narodnyy-kontrol.html.
8. ConstraintLayout [Електронний ресурс]. – 2019 . – Режим доступу до ресурсу: http://developer.alexanderklimov.ru/android/layout/constraintlayout.php.
9. Picasso [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://developer.alexanderklimov.ru/android/library/picasso.php.

10. Библиотека appcompat [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://github.com/lanit-tercom-school/grouplock/wiki/Библиотека-appcompat.
11. RecyclerView и CardView. Новые виджеты в Android [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://habr.com/ru/post/237101/.
12. Android Studio: особенности, достоинства и недостатки [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://arduinoplus.ru/android-studio/.
13. Firebase Realtime Database [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.techotopia.com/index.php/Firebase_Realtime_Database/.
14. Android: Gradle [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://developer.alexanderklimov.ru/android/theory/gradle.php.

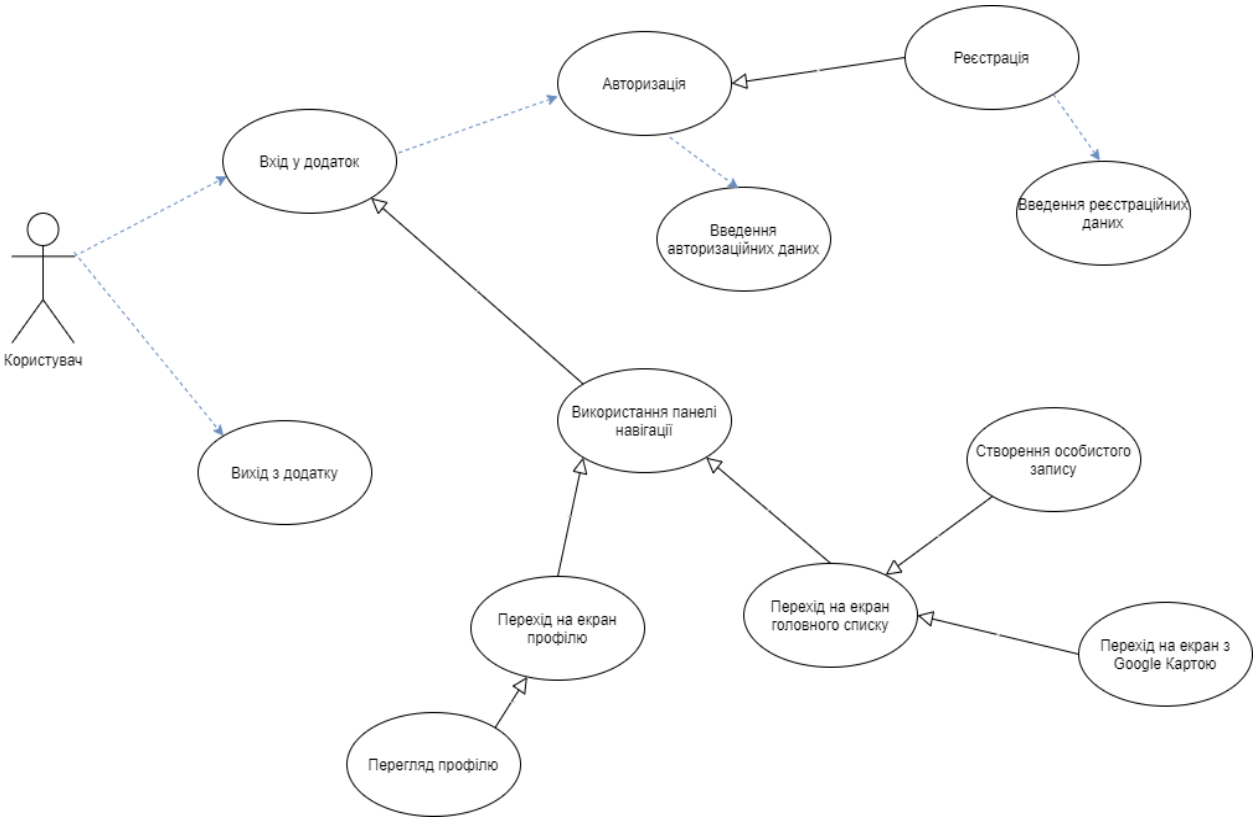
ДОДАТКИ

Схема Алгоритму



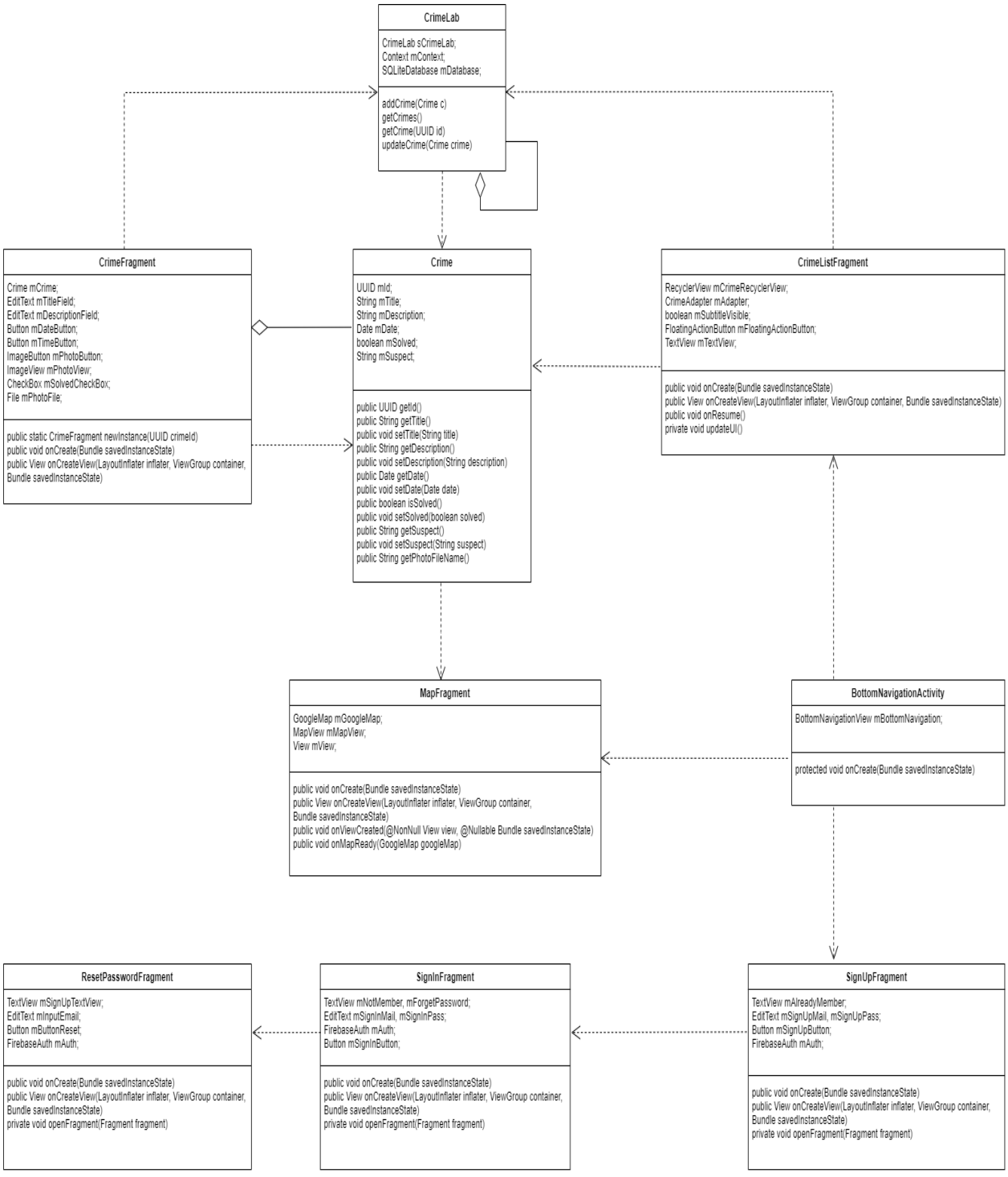
					ДП.6419.04.000 Д1		
Зм.		*№ документа*	*Підп.*	*Дата*			
					Інформаційна система «Чисте місто» Додаток А	*Літ.*	*Аркуш*
						Т	2
Н.конт		*Симоненко В.П.*					1
Затв.						*НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64*	

Функціональна Схема



					ДП.6419.05.000 Д2		
Зм.		*№ документа*	*Підп.*	*Дата*			
					Інформаційна система «Чисте місто» Додаток Б	*Літ.*	*Аркуш*
						Т	1
Н.конт		*Симоненко В.П.*					
Затв.						*НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64*	

Діграма Класів



					ДП.6419.06.000 Д3	
Зм.		*№ документа*	*Підп.*	*Дата*		
					Інформаційна система «Чисте місто» Додаток В	*Літ.* T	*Аркуш* 1 / 1
Н.конт		*Симоненко В.П.*				*НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64*
Затв.						

Лістінг Програми

```
public class Crime {

    private UUID mId;
    private String mTitle;
    private String mDescription;
    private Date mDate;
    private boolean mSolved;
    private String mSuspect;

    public Crime() {
        this(UUID.randomUUID());
    }

    public Crime(UUID id) {
        mId = id;
        mDate = new Date();
    }

    public UUID getId() {
        return mId;
    }

    public String getTitle() {
        return mTitle;
    }

    public void setTitle(String title) {
        mTitle = title;
    }

    public String getDescription() {
        return mDescription;
    }

    public void setDescription(String description) {
        mDescription = description;
    }

    public Date getDate() {
        return mDate;
    }

    public void setDate(Date date) {
        mDate = date;
    }

    public boolean isSolved() {
        return mSolved;
    }

    public void setSolved(boolean solved) {
        mSolved = solved;
    }
```

```
    public String getSuspect() {
        return mSuspect;
    }

    public void setSuspect(String suspect) {
        mSuspect = suspect;
    }

    public String getPhotoFileName() {
        return "IMG_" + getId().toString() + ".jpg";
    }
}
```

```
public class CrimeLab {
    private static CrimeLab sCrimeLab;

    private Context mContext;
    private SQLiteDatabase mDatabase;

    public static CrimeLab get(Context context) {
        if (sCrimeLab == null) {
            sCrimeLab = new CrimeLab(context);
        }
        return sCrimeLab;
    }

    private CrimeLab(Context context) {
        mContext = context.getApplicationContext();
        mDatabase = new CrimeBaseHelper(mContext).getWritableDatabase();
    }

    public void addCrime(Crime c) {
        ContentValues values = getContentValues(c);
        mDatabase.insert(CrimeTable.NAME, null, values);
    }

    public List<Crime> getCrimes() {
        List<Crime> crimes = new ArrayList<>();

        CrimeCursorWrapper cursor = queryCrimes(null, null);

        try {
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
                crimes.add(cursor.getCrime());
                cursor.moveToNext();
            }
        } finally {
            cursor.close();
        }
```

```
    return crimes;
  }

  public Crime getCrime(UUID id) {
    CrimeCursorWrapper cursor = queryCrimes(
        CrimeTable.Cols.UUID + " = ?",
        new String[]{id.toString()});
    try {
      if (cursor.getCount() == 0) {
        return null;
      }

      cursor.moveToFirst();
      return cursor.getCrime();
    } finally {
      cursor.close();
    }
  }

  private static ContentValues getContentValues(Crime crime) {
    ContentValues values = new ContentValues();
    values.put(CrimeTable.Cols.UUID, crime.getId().toString());
    values.put(CrimeTable.Cols.TITLE, crime.getTitle());
    values.put(CrimeTable.Cols.DESCRIPTION, crime.getDescription());
    values.put(CrimeTable.Cols.DATE, crime.getDate().getTime());
    values.put(CrimeTable.Cols.SOLVED, crime.isSolved() ? 1 : 0);
    values.put(CrimeTable.Cols.SUSPECT, crime.getSuspect());

    return values;
  }

  public void updateCrime(Crime crime) {
    String uuidString = crime.getId().toString();
    ContentValues values = getContentValues(crime);

    mDatabase.update(
        CrimeTable.NAME,
        values,
        CrimeTable.Cols.UUID + " = ?",
        new String[]{uuidString});
  }

  public void deleteCrime(UUID uuid) {
    mDatabase.delete(CrimeTable.NAME, CrimeTable.Cols.UUID + " = ?", new String[]{uuid.toString()});
  }

  private CrimeCursorWrapper queryCrimes(String whereClause, String[] whereArgs) {
    Cursor cursor = mDatabase.query(
        CrimeTable.NAME,
        null,
        whereClause,
        whereArgs,
        null,
        null,
        null);
```

```
        return new CrimeCursorWrapper(cursor);
    }

    public File getPhotoFile(Crime crime) {
        File filesDir = mContext.getFilesDir();
        return new File(filesDir, crime.getPhotoFileName());

    }
}


public class CrimeListFragment extends Fragment {

    private static final String SAVED_SUBTITLE_VISIBLE = "subtitle";

    private RecyclerView mCrimeRecyclerView;

    private CrimeAdapter mAdapter;

    private boolean mSubtitleVisible;

    private FloatingActionButton mFloatingActionButton;
    private TextView mTextView;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_crime_list, container, false);
        mCrimeRecyclerView = view.findViewById(R.id.crime_recycler_view);
        mCrimeRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

        if (savedInstanceState != null) {
            savedInstanceState.getBoolean(SAVED_SUBTITLE_VISIBLE);
        }

        mTextView = view.findViewById(R.id.empty_list_text_view);

        mFloatingActionButton = view.findViewById(R.id.floatingActionButton);
        mFloatingActionButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Crime crime = new Crime();
                CrimeLab.get(getActivity()).addCrime(crime);
                Intent intent = CrimePagerActivity.newIntent(getActivity(), crime.getId());
                startActivity(intent);
            }
        });

        updateUI();
```

```
    return view;
}

@Override
public void onResume() {
    super.onResume();
    updateUI();

    //updateSubtitle();
}

private void updateUI() {
    CrimeLab crimeLab = CrimeLab.get(getActivity());
    List<Crime> crimes = crimeLab.getCrimes();

    if (crimes.isEmpty()) {
        mTextView.setVisibility(View.VISIBLE);
    } else {
        mTextView.setVisibility(View.INVISIBLE);
    }

    if (mAdapter == null) {
        mAdapter = new CrimeAdapter(crimes);
        mCrimeRecyclerView.setAdapter(mAdapter);
    } else {
        mAdapter.setCrimes(crimes);
        mAdapter.notifyDataSetChanged();
    }
}

private class CrimeHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    private TextView mTitleTextView;
    private TextView mDescriptionTextView;
    private TextView mDateTextView;
    private ImageView mSolvedImageView;
    private Crime mCrime;

    public CrimeHolder(LayoutInflater inflater, ViewGroup parent) {
        super(inflater.inflate(R.layout.list_item_crime_card, parent, false));

        mTitleTextView = itemView.findViewById(R.id.crime_title);
        mDescriptionTextView = itemView.findViewById(R.id.crime_description);
        mDateTextView = itemView.findViewById(R.id.crime_date);
        mSolvedImageView = itemView.findViewById(R.id.crime_solved);

        itemView.setOnClickListener(this);
    }

    public void bind(Crime crime) {
        mCrime = crime;
        mTitleTextView.setText(mCrime.getTitle());
        mDescriptionTextView.setText(mCrime.getDescription());
        mDateTextView.setText(DateFormat.format("EEEE, MMM d, yyyy", mCrime.getDate()));
        mSolvedImageView.setVisibility(crime.isSolved() ? View.VISIBLE : View.GONE);
    }
```

```
    @Override
    public void onClick(View v) {
        Intent intent = CrimePagerActivity.newIntent(getActivity(), mCrime.getId());
        startActivity(intent);
    }
}

private class CrimeAdapter extends RecyclerView.Adapter<CrimeHolder> {
    private List<Crime> mCrimes;

    public CrimeAdapter(List<Crime> crimes) {
        mCrimes = crimes;
    }

    @NonNull
    @Override
    public CrimeHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(getActivity());
        return new CrimeHolder(inflater, parent);
    }

    @Override
    public void onBindViewHolder(@NonNull CrimeHolder holder, int position) {
        Crime crime = mCrimes.get(position);
        holder.bind(crime);
    }

    @Override
    public int getItemCount() {
        return mCrimes.size();
    }

    public void setCrimes(List<Crime> crimes) {
        mCrimes = crimes;
    }
}

@Override
public void onCreateOptionsMenu(@NonNull Menu menu, @NonNull MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater);
    inflater.inflate(R.menu.fragment_crime_list, menu);

    MenuItem subtitleItem = menu.findItem(R.id.show_subtitle);
    if (mSubtitleVisible) {
        subtitleItem.setTitle(R.string.hide_subtitle);
    } else {
        subtitleItem.setTitle(R.string.show_subtitle);
    }
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.new_crime:
            Crime crime = new Crime();
            CrimeLab.get(getActivity()).addCrime(crime);
```

```
            Intent intent = CrimePagerActivity.newIntent(getActivity(), crime.getId());
            startActivity(intent);
            return true;
          case R.id.show_subtitle:
            mSubtitleVisible = !mSubtitleVisible;
            getActivity().invalidateOptionsMenu();
            updateSubtitle();
            return true;
          default:
            return super.onOptionsItemSelected(item);
        }
    }

    private void updateSubtitle() {
        CrimeLab crimeLab = CrimeLab.get(getActivity());
        int crimeSize = crimeLab.getCrimes().size();
        String subtitle = getResources().getQuantityString(R.plurals.subtitle_plural, crimeSize, crimeSize);

        if (!mSubtitleVisible) {
            subtitle = null;
        }

        AppCompatActivity activity = (AppCompatActivity) getActivity();
        activity.getSupportActionBar().setSubtitle(subtitle);
    }

    @Override
    public void onSaveInstanceState(@NonNull Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putBoolean(SAVED_SUBTITLE_VISIBLE, mSubtitleVisible);
    }
}


public class CrimeFragment extends Fragment {

    private static final String ARG_CRIME_ID = "crime_id";
    private static final String DIALOG_DATE = "DialogDate";
    private static final String DIALOG_TIME = "DialogTime";
    private static final String DIALOG_PHOTO = "DialogPhoto";
    private static final int REQUEST_DATE = 0;
    private static final int REQUEST_TIME = 1;
    private static final int REQUEST_PHOTO = 2;

    private Crime mCrime;
    private EditText mTitleField;
    private EditText mDescriptionField;
    private Button mDateButton;
    private Button mTimeButton;
    private ImageButton mPhotoButton;
    private ImageView mPhotoView;
    private CheckBox mSolvedCheckBox;
    private File mPhotoFile;

    public static CrimeFragment newInstance(UUID crimeId) {
        Bundle args = new Bundle();
```

```
        args.putSerializable(ARG_CRIME_ID, crimeId);

        CrimeFragment fragment = new CrimeFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        UUID crimeId = (UUID) getArguments().getSerializable(ARG_CRIME_ID);
        mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);
        mPhotoFile = CrimeLab.get(getActivity()).getPhotoFile(mCrime);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.test_fragment_crime, container, false);

        mTitleField = v.findViewById(R.id.crime_title);
        mTitleField.setText(mCrime.getTitle());
        mTitleField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                mCrime.setTitle(s.toString());
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });

        mDescriptionField = v.findViewById(R.id.crime_description);
        mDescriptionField.setText(mCrime.getDescription());
        mDescriptionField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                mCrime.setDescription(s.toString());
            }

            @Override
            public void afterTextChanged(Editable s) {
```

```
        }
    });

    mDateButton = v.findViewById(R.id.crime_date);
    mDateButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            FragmentManager manager = getFragmentManager();
            DatePickerFragment dialog = DatePickerFragment.newInstance(mCrime.getDate());
            dialog.setTargetFragment(CrimeFragment.this, REQUEST_DATE);
            dialog.show(manager, DIALOG_DATE);
        }
    });

    mTimeButton = v.findViewById(R.id.crime_time);
    mTimeButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            FragmentManager manager = getFragmentManager();
            TimePickerFragment dialog = TimePickerFragment.newInstance(mCrime.getDate());
            dialog.setTargetFragment(CrimeFragment.this, REQUEST_TIME);
            dialog.show(manager, DIALOG_TIME);
        }
    });

    updateDate();

    mSolvedCheckBox = v.findViewById(R.id.crime_solved);
    mSolvedCheckBox.setChecked(mCrime.isSolved());
    mSolvedCheckBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            mCrime.setSolved(isChecked);
        }
    });

    PackageManager packageManager = getActivity().getPackageManager();

    mPhotoButton = v.findViewById(R.id.crime_camera);
    final Intent captureImage = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    boolean canTakePhoto = mPhotoFile != null && captureImage.resolveActivity(packageManager) != null;
    mPhotoButton.setEnabled(canTakePhoto);

    mPhotoButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Uri uri = FileProvider.getUriForFile(getActivity(),
                    "com.bignerdranch.android.criminalintent.fileprovider",
                    mPhotoFile);

            captureImage.putExtra(MediaStore.EXTRA_OUTPUT, uri);
            List<ResolveInfo> cameraActivities = getActivity().
                    getPackageManager().queryIntentActivities(captureImage,
PackageManager.MATCH_DEFAULT_ONLY);
```

```
            for (ResolveInfo activity : cameraActivities) {
                getActivity().grantUriPermission(activity.activityInfo.packageName,
                        uri, Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
            }
            startActivityForResult(captureImage, REQUEST_PHOTO);
        }
    });

    mPhotoView = v.findViewById(R.id.crime_photo);
    mPhotoView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            FragmentManager manager = getFragmentManager();
            PhotoFragment dialog = PhotoFragment.newInstance(mPhotoFile.getPath());
            dialog.show(manager, DIALOG_PHOTO);
        }
    });

    updatePhotoView();

    return v;
}

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if (resultCode != Activity.RESULT_OK) {
        return;
    }
    if (requestCode == REQUEST_DATE) {
        Date date = (Date) data.getSerializableExtra(DatePickerFragment.EXTRA_DATE);
        mCrime.setDate(date);
        updateDate();
    } else if (requestCode == REQUEST_TIME) {
        Date date = (Date) data.getSerializableExtra(TimePickerFragment.EXTRA_TIME);
        mCrime.setDate(date);
        updateDate();

    } else if (requestCode == REQUEST_PHOTO && data != null) {
        Uri uri = FileProvider.getUriForFile(getActivity(),
                "com.bignerdranch.android.criminalintent.fileprovider",
                mPhotoFile);

        getActivity().revokeUriPermission(uri, Intent.FLAG_GRANT_WRITE_URI_PERMISSION);

        updatePhotoView();
    }
}

private void updateDate() {
    Date d = mCrime.getDate();
    CharSequence day = DateFormat.format("EEEE, MMM d, yyyy", d);
    CharSequence time = DateFormat.format("HH:mm", d);

    mDateButton.setText(day);
    mTimeButton.setText(time);
}
```

```
    private String getCrimeReport() {
        String solvedString = null;

        if (mCrime.isSolved()) {
            solvedString = getString(R.string.crime_report_solved);
        } else {
            solvedString = getString(R.string.crime_report_unsolved);
        }

        String dateFormat = "EEE, MM, dd";
        String dateString = DateFormat.format(dateFormat, mCrime.getDate()).toString();

        String suspect = mCrime.getSuspect();
        if (suspect == null) {
            suspect = getString(R.string.crime_report_no_suspect);
        } else {
            suspect = getString(R.string.crime_report_no_suspect);
        }

        String report = getString(R.string.crime_report, mCrime.getTitle(), dateString, solvedString, suspect);

        return report;
    }

    private void rotateImage(Bitmap bitmap) {
        ExifInterface exifInterface = null;
        try {
            exifInterface = new ExifInterface(String.valueOf(CrimeLab.get(getActivity()).getPhotoFile(mCrime)));

        } catch (IOException e) {
            e.printStackTrace();
        }
        int orientation = exifInterface.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_UNDEFINED);
        Matrix matrix = new Matrix();
        switch (orientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                matrix.setRotate(90);
                break;
            case ExifInterface.ORIENTATION_ROTATE_180:
                matrix.setRotate(180);
                break;
            default:
        }
        Bitmap rotatedBitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(),
matrix, true);
        mPhotoView.setImageBitmap(rotatedBitmap);
    }

    private void updatePhotoView() {
        if (mPhotoFile == null || !mPhotoFile.exists()) {
            mPhotoView.setImageDrawable(null);
        } else {
            Bitmap bitmap = PictureUtils.getScaledBitmap(mPhotoFile.getPath(), getActivity());
            rotateImage(bitmap);
```

```
        }
    }

    @Override
    public void onPause() {
        super.onPause();

        CrimeLab.get(getActivity())
                .updateCrime(mCrime);
    }
}
```