

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА ЗАСТОСУНКУ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ
РОБОТИ СПІВРОБІТНИКІВ
APPLICATION DEVELOPMENT FOR MONITORING AND ANALYZING THE
EMPLOYEE PERFORMANCE

Виконав: студентка 4 курсу, групи КНТ-127
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Гринчишина М.С.
(прізвище та ініціали)

Керівник Степаненко О.О.
(прізвище та ініціали)

Рецензент Голуб Т.В
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гринчишиної Марини Сергіївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка застосунку для моніторингу та аналізу роботи співробітників

Application Development for Monitoring and Analyzing the Employee Performance

керівник проєкту (роботи) Степаненко Олександр Олексійович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 29 квітня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Дослідження предметної області.

2. Методи та моделі. 3. Проєктування та розроблення застосунку. 4. Керівництво програміста. 5. Керівництво користувача

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	Степаненко О.О., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “15” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2–3 тижні	Розділ 1
3	Провести аналіз чинників та критеріїв оцінки персоналу; провести аналіз основних принципів і методів оцінки роботи персоналу	4–5 тижні	Розділ 2
4	Провести аналіз середовища розробки	6 тиждень	Розділ 3
5	Провести системне та функціональне проєктування	7–8 тижні	Розділ 3
6	Розробити керівництво програміста	9 тиждень	Розділ 4
7	Розробити керівництво користувача	10 тиждень	Розділ 5
8	Оформлення висновків	11 тиждень	Висновки
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

_____ Гринчишина М.С.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Степаненко О.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
96 с., 1 табл., 18 рис., 2 дод., 12 джерел.

МОНІТОРИНГ, ОС ANDROID, INTENT, JAVA, URI, JAVA JDK

Об'єкт дослідження – система методів, принципів і порядок оцінки співробітників.

Предмет роботи – можливості вдосконалення роботи з персоналом для підвищення якості його роботи.

Мета роботи – розробити програмну реалізацію моніторингу та аналізу роботи співробітників відділу маркетингу.

Матеріали, методи та технічні засоби: мова програмування java, операційна система Android, середовище розробки Android Studio, персональний комп'ютер з процесором Intel Core 5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмну реалізацію моніторингу співробітників.

Висновки. Розроблено програмну реалізацію застосунку для моніторингу та аналізу роботи співробітників з використанням ОС Android та мови програмування java.

Галузь використання – маркетинг.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 96 pages, 1 tables, 18 figures, 2 appendixes, 12 sources.

MONITORING, ANDROID OS, INTENT, JAVA, URI, JAVA JDK

Object of study is a system of methods, principles and procedures for evaluating employees.

Subject of study is the possibility of improving work with staff to improve the quality of their work.

The purpose of the work is to develop a software implementation of monitoring and analysis of the staff of the marketing department.

Materials, methods, and tools: the java programming language, Android operating system, personal computer with Intel Core 5 processor running Microsoft Windows 10 operating system.

Results. A software implementation of employee monitoring has been created.

Conclusions. Developed a software implementation of the application for monitoring and analyzing the work of employees using Android and java programming language.

The field of use is marketing.

ЗМІСТ

С.

Перелік скорочень та умовних познач	8
Вступ	9
1 Дослідження предметної області та постановка задач дипломної роботи	11
1.1 Поняття, чинники та критерії оцінки персоналу	11
1.2 Розгляд аналогів	15
1.2.1 Програма Repsly	15
1.2.2 Сервіс amoCRM	16
1.2.3 Сервіс EasyMerch	17
1.3 Висновки до розділу	18
2 Методи та моделі	19
2.1 Сучасні методи, принципи і порядок оцінки роботи персоналу	19
2.2 Вибір середовища розробки	21
2.3 Висновки до розділу	29
3 Проєктування та розроблення застосунку	30
3.1 Модулі проєкту	30
3.2 Функціональне проєктування	38
3.2.1 Опис взаємодії з сервером	38
3.2.2 Опис класів системних компонентів	40
3.2.3 Опис класів взаємодії з даними	55
3.2.4 Опис класів пакета dialogs	58
3.2.5 Огляд функціональних класів з пакета helpers	60
3.2.6 Опис класів пакета networking	63
3.3 Висновки до розділу	65
4 Керівництво програміста	66
5 Керівництво користувача	73
Висновки	79
Перелік джерел посилання	80

Додаток А Текст програми	81
Додаток Б Слайди презентації.....	89

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE	– Integrated development environment;
JVM	– Java Virtual Machine;
БД	– база – даних;
ПЗ	– програмні засоби;
ПК	– персональний комп'ютер;
ОС	– операційна система.

ВСТУП

Успіх роботи підприємства забезпечують працівники, зайняті на ньому і тому, сучасна концепція управління підприємством передбачає виділення з великого числа функціональних сфер управлінської діяльності тієї, яка пов'язана з управлінням кадрової складової виробництва — персоналом підприємства. Першочергові завдання підвищення ефективності управління персоналом та оцінки праці його ключових категорій висувають нові вимоги до існуючих систем оцінки роботи персоналу з урахуванням підвищення якості робочої сили, зростання його професіоналізму, мотивації на високопродуктивну роботу. При цьому розробка системи моніторингу роботи персоналу стає однією з найважливіших складових загальної системи управління. Вивчення теорії і практики оцінки якості та ефективності роботи персоналу є актуальним завданням.

Оцінка роботи персоналу — це "процес збору, аналізу та оцінки інформації про те, як працівники виконують доручену роботу, і з'ясування того, якою мірою їх робоче поведінка, робочі показники і індивідуальні характеристики відповідають вимогам організації і керівництва" [1]. Оцінка роботи персоналу є складовою частиною процесу управлінського контролю, здійснюваного щодо людських ресурсів організації.

Дана робота є частиною системи, яка використовується маркетинговим відділом компанії для моніторингу та аналізу роботи співробітників, а також її можна використовувати для стеження за динамікою продажів відповідно до виконаної роботи.

Мета дипломної роботи – розробити програмну реалізацію моніторингу та аналізу роботи співробітників відділу маркетингу.

Для реалізації поставленої мети поставлені наступні завдання:

- провести аналіз чинників та критеріїв оцінки персоналу;

- провести аналіз основних принципів і методів оцінки роботи персоналу;
- провести аналіз середовища розробки;
- провести системне та функціональне проєктування;
- розробити програму для моніторингу та аналізу роботи співробітників.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДИПЛОМНОЇ РОБОТИ

1.1 Поняття, чинники та критерії оцінки персоналу

Центральне місце в теорії управління персоналом займає персонал організації.

Персонал організації — це сукупність об'єднаних людей, на яких проводиться цілеспрямована дія для досягнення поставлених організацією цілей і завдань.

Персонал організації може розглядатися в різних аспектах, що дозволяє найбільш повно охарактеризувати його з точки зору його багатовимірності. Неоднозначність трактування категорії «персонал» обумовлена складністю людини як органічного так і психологічного феномена. Вироблення і застосування різних методологій впливу на персонал обумовлені багатоаспектністю, ситуативністю та відносністю основоположних якостей персоналу взагалі та конкретної особистості зокрема [1, 2].

Під оцінкою персоналу розуміється діяльність уповноважених на її здійснення осіб щодо визначення ступеня придатності тієї чи іншої людини до виконання покладених на неї обов'язків, досягнутих успіхів, ступеня вираженості необхідних для роботи якостей.

Правильна оцінка якостей людини можлива в тому випадку, коли чітко сформовані вимоги до них. Тому такі вимоги розробляються висококваліфікованими експертами.

Якості людей, що підлягають оцінці, називаються її факторами, які утворюють єдину систему, тому що об'єктивність оцінки можлива тільки на основі комплексного підходу. У разі відсутності чіткості і конкретності в підходах, одні і ті ж якості можуть трактуватися з різних позицій [2].

Ступінь вираженості факторів оцінки характеризується її показниками, а також критеріями — граничними значеннями, за якими стан чинників буде

задовольняти або не задовольняти певним вимогам (критерії є орієнтирами для оцінки). Показники оцінки можуть бути абсолютними і відносними, істотними і неістотними, прийнятними і неприйнятними, жорсткими і м'якими. Жорсткі показники, на відміну від м'яких, повинні виражатися кількісно, бути легкими для розрахунку і не залежати від суб'єктивної думки оцінювача. До показників пред'являються такі вимоги, як повнота і достовірність характеристики об'єкта, забезпечення сумісності з іншими особами, відповідність мети організації та ін.

а) До основних факторів оцінки відносяться [2]:

1) діяльність і її результати:

- складність, якість праці;
- досягнення мети;
- якісні та кількісні характеристики індивідуального результату;
- внесок в загальний підсумок підрозділу і організації в цілому;
- знання роботи, розуміння її проблем і ситуацій, що виникають;
- оперативність дій;
- комплексність, масштабність, технологічна складність виконуваних завдань;
- творчий підхід до справи;

2) ділові якості працівників:

- дисциплінованість, акуратність, виконання в термін завдань;
- готовність до додаткової роботи;
- вміння долати труднощі, доводити справу до кінця;
- ставлення до навчання і самонавчання;
- ініціативність;
- здатність приймати рішення;
- вміння оптимізувати діяльність, раціонально використовувати свій час;
- кількість помилок;

- особливості поведінки в конфліктній ситуації (орієнтація на співробітництво, компроміс, суперництво, усунення конфлікту будь-якими способами);
- організаторські здібності;
- потенціал професійного і службового зростання;
- товариськість, комунікабельність;
- стиль керівництва, підпорядкування, взаємодії та ін;

3) професіоналізм:

- рівень інтелектуального розвитку (виявляється за допомогою тестів на інтелект, спеціальних тестів обдарувань);
- здатність до аналізу і узагальнень;
- володіння теоретичними та спеціальними знаннями, навичками, вміннями;
- логіка, чіткість мислення;
- прагнення до пошуку;

4) моральні якості:

- працьовитість;
- принциповість;
- чесність;
- відповідальність, сумлінність;
- обов'язковість;
- самокритичність;
- мотиви спонукання до праці;

5) потенціал, здібності та особистісні якості, необхідні для виконання службових обов'язків [3]:

- концентрація і переключення уваги;
- агресивність;
- емоційна і нервово-психічна стійкість;
- самостійність, рішучість;
- самоконтроль, самовладання;

- тривожність;
- рухливість або врівноваженість нервових процесів;
- швидкість реакції та ін;

б) до додаткових факторів оцінки відносяться:

- здатність планувати діяльність і ресурси (ставити і коригувати цілі, розподіляти і контролювати ресурси);
- здатність організовувати роботу підлеглих;
- здатність керувати в критичних ситуаціях (виявляти такі, вибирати курс дій, виявляти відхилення в роботі підлеглих, своєчасно приймати рішення);
- лідирувати;
- працювати з документами (розробляти, погоджувати проекти рішень контролювати їх виконання, підтримувати документообіг);
- делегувати повноваження (забезпечувати підлеглих чіткими вказівками, раціонально розподіляти обов'язки, визначати і контролювати терміни виконання, надавати необхідну допомогу);
- мотивувати (використовувати матеріальні і нематеріальні стимули, враховувати і нейтралізувати фактори, що негативно впливають на мотивацію, надихати своїм прикладом);
- розвивати підлеглих (допомагати в адаптації, освоєнні нової роботи, організовувати навчання і підвищення кваліфікації);
- надавати увагу і повагу співробітникам (підтримувати хороші відносини в колективі, демонструвати увагу до особистих проблем підлеглих);
- здійснювати комунікації (брати участь в обміні діловою інформацією, забезпечувати працівників і вищі інстанції необхідними відомостями, підтримувати контакти з клієнтами і громадськістю);

в) до об'єктивних і суб'єктивних критеріїв оцінки показників відносяться:

- об'єктивні критерії - стандарти, нормативи якості та продуктивність, які встановлюються для будь-якого виду діяльності;
- суб'єктивні критерії це такі показники і характеристики, які оцінюються на підставі думок і оцінок експертів;

г) критерії, що використовуються при оцінці персоналу повинні відповідати трьом основним вимогам [2, 3]:

- валідність — найважливіша вимога до критеріїв оцінки роботи персоналу. Критерій оцінки є дійсним, якщо з його допомогою можна максимально точно оцінити ефективність роботи конкретної категорій персоналу;
- надійність — критерій оцінки роботи, що забезпечує сталість результатів вимірювання;
- показник, який дозволяє виявляти відмінності в рівні професійної успішності співробітників. Найважливішими вимогами до системи оцінки є її дієвість і практичність. Система оцінки вважається дієвою, коли в результаті її застосування забезпечується чіткий зв'язок результатів праці та її оплати, високий рівень мотивації і максимальна віддача від працівників. Вимога практичності до системи оцінки — це зручність використання системи оцінки на практиці людям, що проводять оцінку і тим, хто оцінюється [3].

1.2 Розгляд аналогів

1.2.1 Програма Repsly

Програма Repsly розроблена в якості єдиної системи постійного відстеження, контролю за допомогою GPS-модуля в смартфоні, а також управління на віддаленій основі торговими представниками та командою співробітників, які працюють віддалено. Підійде для компанії будь-якого розміру і напрямку діяльності [4].

Repsly – це системне рішення серед конкурентів в областях:

- контроль повсякденної роботи. Відстежується початок роботи і закінчення візиту працівника на точку, по GPS можна відстежити місце розташування вашого персоналу, враховується навіть пройдена за робочий день відстань;

- спостереження за співробітниками. Вони роблять нотатки про зроблених візитах, які можна переглянути в режимі онлайн. Заповнення спеціальних опитувальних форм і викладання фотографій зроблять контроль з вашого боку простим;

- аналітичні звіти для поліпшення якості роботи. Відстеження поточного стану роздрібних мереж, таблиці з оновленими даними і графіками для наочності;

- розширені можливості Repsly для організації вашої діяльності. Зручний і функціональний щоденник з нагадуваннями і повідомленнями, актуальна база клієнтів і список ваших співробітників з контактами, а також коментарі про проведені ними візити в торгові точки.

Програма є відмінним варіантом для того, щоб якісно організувати діяльність будь-якої кількості співробітників, що працюють на позиціях торгових представників і виїзних фахівців.

1.2.2 Сервіс amoCRM

AmoCRM — це хмарний сервіс для ведення клієнтської бази, збільшення продажів і виконання фінансових операцій. Платформа використовується в організаціях торгівлі та сфери послуг. Комп'ютерна версія системи може використовуватися в будь-якому браузері. Для входу в профіль з мобільного пристрою потрібна установка програми. Програми передбачені для смартфонів з операційною системою iOS і Android [5].

Платформа складається з програмних модулів для керівника і співробітників компанії.

Блок керівника володіє наступним функціоналом:

- аналіз роботи менеджерів;
- постановка завдань для співробітників;
- контроль виконаних та відкладених угод;
- планування заходів щодо підвищення ефективності бізнесу;
- формування звітів про ефективність виконаної роботи.

Персонал компанії може виконувати наступні завдання:

- здійснювати операції;
- взаємодіяти з клієнтами;
- створювати і відправляти документи в електронному вигляді;
- відстежувати завдання, призначені керівником, планувати їх

виконання.

В особистому кабінеті системи можна налаштовувати відображення віджетів в залежності від особливостей діяльності компанії.

Платформа має просте оформлення і зручне управління. В середньому менеджерам потрібно 1-2 дні для повного освоєння її функцій. При виникненні труднощів в роботі з сервісом користувач може отримати допомогу від професійного консультанта. Однак, дана програма має тільки пробний період [4].

1.2.3 Сервіс EasyMerch

EasyMerch — сервіс автоматизації мерчендайзингу. В системі реалізовані наступні можливості для мерчендайзерів і торгових агентів [6]:

- планування маршруту на день;
- інформація по кожній торговельній точці;
- правила викладки для різних товарних груп;
- актуальні промоакції;
- складання списку завдань;
- замовлення товарів для магазинів;
- контроль дебіторської заборгованості;

- перегляд історії замовлень;
- виконання плану продажів і промоакцій.

За результатами роботи фахівець може сформувати в програмі звіти за наявністю і кількістю товарів в магазинах, відповідно до запланованої викладки.

Також в системі доступні особисті кабінети супервайзера, керівника, регіонального представника. Для цих фахівців запропонований розширений функціонал відповідно до займаної посади.

Офісні співробітники можуть використовувати вебверсію платформи. Для мерчендайзерів і торгових представників розроблено спеціальний додаток EasyMerch. Воно встановлюється на планшет або смартфон співробітника, дає йому можливість користуватися сервісом в будь-якому місці.

Пропонуються базовий і розширений тарифні плани. Вартість кожного з них розраховується за кожного підключеного до системи співробітника. При підключенні менше 50 співробітників щомісяця додатково оплачується абонентська плата [5].

1.3 Висновки до розділу

Отже провівши аналіз чиників та критеріїв оцінки ефективності персоналу виникає необхідність розробити застосунок, яке буде допоміжним інструментом для аналізу. Перш за все програма орієнтована на збір та аналіз інформації, переважно на роботу з фотографіями.

Аналіз роботи співробітників дозволяє полегшити співробітникам ведення звітності про виконану роботу та надає керівникам можливість контролю за роботою підлеглих. Було проведено аналіз відомих аналогів програми, вони мають багато переваг, але всі розглянуті програми є комерційними рішеннями та жодна з них немає змоги працювати з графічним матеріалом.

2 МЕТОДИ ТА МОДЕЛІ

2.1 Сучасні методи, принципи і порядок оцінки роботи персоналу

Оцінка роботи персоналу дозволяє забезпечувати нормальне функціонування всієї організації і успішно вирішувати виробничі завдання за допомогою таких методів оцінки роботи персоналу [7]:

- встановлення стандартів і нормативів: оцінка роботи полягає у встановленні стандартів або нормативів з подальшим порівнянням робочих показників кожного працівника з даними стандартами. Даний метод найчастіше застосовується в умовах виробництва;

- оцінювання на підставі письмових характеристик, зазвичай проводиться при підготовці до атестації, при вирішенні питань про підготовку резерву на заміщення керівних посад і при призначеннях на посаду і переміщеннях в інші підрозділи. Письмова характеристика на працівника зазвичай готується його безпосереднім керівником. Необхідність стандартизації характеристик, що надаються атестаційній комісії, вимагає, щоб керівники готували їх відповідно до встановленої форми, що визначає ті питання, які повинні бути освітлені (наприклад, робочі досягнення, рівень професійної підготовки, оцінка професійного потенціалу працівника, потреба в навчанні і підвищенні кваліфікації, можливість переведення на більш складну і відповідальну роботу та ін.);

- оціночні шкали дають можливість керівнику оцінити ступінь розвитку працівників ділових якостей, схильність до певних видів робочого поведінки або готовність до досягнення певних робочих результатів. Шкали оцінки припускають використання спеціальних оціночних форм. Оціночна форма (бланк) складається з ряду шкал, що оцінюють різні аспекти роботи, такі, як професійні знання, якість і кількість роботи, здатність до самостійної роботи, рівень розвитку професійних навичок і ін;

– методи ранжирування — методи оцінки роботи підлеглих, які дають можливість здійснювати порівняння працівників один з одним, а не тільки з встановленим стандартом або нормативом. Ці методи характеризуються тим, що вони прості у використанні, дозволяють легко розділити хороших і поганих працівників, дають інформацію, необхідну для прийняття адміністративних рішень (підвищення зарплати, звільнення, підвищення або пониження в посаді та ін.). Існує кілька різновидів ранжирування: пряме ранжування та парне порівняння;

– заданий розподіл — форма порівняльної оцінки, при якій керівник відносить підлеглих до певної категорії відповідно до заздалегідь заданих правил. При цьому виходять з того, що прояв оцінюваних робочих характеристик підпорядковується закону нормального розподілу;

– оцінка робочої поведінки використовується в тому випадку, коли оціночні форми, що містять твердження або формулювання описують ефективні і неефективні види робочої поведінки. При цьому оцінка здійснюється по ряду параметрів, пов'язаних з виконанням роботи або з досягненням намічених цілей;

– управління по цілям як метод оцінки ефективності, заснований на постановці перед виконавцями цілей, які повинні бути досягнуті за певний період часу (місяць, квартал, рік). Найчастіше цей метод застосовується для оцінки роботи фахівців і керівників різного рангу. Управління по цілям як метод оцінки працівників передбачає широке використання об'єктивних показників, таких, як обсяг продажів, прибуток в рублях, кількість рекламцій і ін. Основними елементами управління по цілям є: постановка цілей, планування роботи, поточний контроль, оцінка досягнутих результатів і підбиття підсумків.

Для розробки системи оцінки роботи персоналу, необхідно обрати найбільш ефективну систему оцінки.

Порівняємо різні методи оцінки з точки зору прийняття адміністративних рішень (А), розвитку працівників (Р) [1-3], надання

зворотного зв'язку (ЗЗ), контролю якості управлінської діяльності (КУ), вдосконалення процесу управління персоналом в цілому (УП) у вигляді табл. 2.1.

Таблиця 2.1 – Порівняння методів

Метод оцінки	Характеристики	Завдання				
		А	Р	ЗЗ	КУ	УП
Оцінка робочих результатів	Робочі результати	+		+		
Письмові характеристики	Вільна форма	+	+	+	+	+
Шкала оцінювання	Індивідуальні якості	+	+	+		
Пряме ранжування	Загальна оцінка	+	+	+		
Змінне ранжування	Загальна оцінка	+				
Парне порівняння	Загальна оцінка	+				
Задане розподілення	Загальна оцінка	+				
Управління за метою	Робочі результати	+	+	+	+	+

Після проведення аналізу та порівняння методів оцінки роботи персоналу, в якості основного підходу був обраний метод управління за метою.

2.2 Вибір середовища розробки

В основі програмування для операційної системи (ОС) Android лежить програмування на Java. Незважаючи на незручність в роботі, Java є однією з найбільш популярних мов програмування. Мова має велику історію: перший реліз Java відбувся ще в 1995 році, тому не дивно, що існує величезна кількість підручників і спеціалізованих сайтів, присвячених Java-розробці. Одним з прикладів можна навести найбільш поширені [8, 9].- об'єктно-орієнтована мова

програмування, розроблена компанією Sun Microsystems. Програми Java зазвичай транслюється в спеціальний байт-код, тому вони можуть працювати на будь-якій віртуальній Java-машині незалежно від комп'ютерної архітектури. Віртуальна машина Java (Java Virtual Machine, далі JVM) – основна частина виконучої системи Java, так званої Java Runtime Environment (JRE). Віртуальна машина Java інтерпретує байт-код, попередньо створений з вихідного тексту програми компілятором.

На даний момент Java відчутно програє в зручності використання більшості сучасних мов програмування, проте не поспішає здавати позиції в рейтингу популярності. Це обумовлено політикою компанії Oracle, через яку темпи розвитку Java істотно відстають від інших поширених мов програмування. Java-співтовариство як і раніше широко, а екосистема мови містить гігантську кодову базу з тисячами сторонніх бібліотек, істотно розширюють функціональність [11].

Однією з головних властивостей Java, яка забезпечила йому величезну популярність, є переносимість. Java проголошує принцип «Write once – run everywhere», який вказує на те, що, в теорії, Java-код може бути виконаний на будь-якій JVM, що працює на будь-якому пристрої. У реальному житті, звичайно, даний принцип працює з великими застереженнями, проте це не заважає успішної працездатності JVM більш ніж на трьох мільярдах пристроїв. Швидше за все, завдяки привабливій перспективі переносимості, саме Java була обрана в якості мови програмування для операційної системи (ОС) Android.

ОС Android є гарним прикладом неточності принципу «Write once – run everywhere»: через обмеженість ресурсів використання стандартної JVM було утруднено, і єдиним можливим вибором було використання Java ME JVM, розробленої для мобільних пристроїв. Однак, компанія Google вирішила розробити власну віртуальну машину Dalvik VM, яка відрізняється від інших JVM наступним [3]:

- використовується спеціальний формат DEX для зберігання двійкових кодів з метою зменшення їх розміру;

- dalvik VM оптимізована для виконання декількох процесів одночасно;
- використовується архітектура, заснована на регістрах проти стекової архітектури в інших JVM;
- використовується власний набір інструкцій, а не стандартний байткод JVM;
- можливий запуск декількох незалежних Android-застосунків в одному процесі;
- реалізовано спеціальний механізм серіалізації об'єктів, заснований на класах Parcel і Parcelable;
- є особливий спосіб для здійснення зв'язку між процесами, заснований на Android Interface Definition Language (AIDL).

Крім нової JVM, також були розглянуті стандартні пакети Java JDK API. Деякі з них були видалені, наприклад, все, що стосувалося Swing (бібліотеки для створення графічного інтерфейсу користувача), але в тому числі додалося кілька власних android-пакетів [9]. Таким чином, далеко не кожна програма або бібліотека, написана на Java, зможе без проблем бути компільована та запущена на Android-пристрої.

У зв'язку з великою популярністю мобільних розробок, існує велика кількість джерел інформації по Android-розробці. Основним джерелом, в першу чергу, є офіційна документація [10]. Крім технічної документації комплексу засобів розробки (Software Development Kit, SDK), офіційний сайт містить безліч прикладів, керівництва по реалізації стандартних компонентів, покрокові рекомендації з розробки додатків з нуля, а також вимоги до дизайну.

Однією з найбільш популярних книг є [11]. Дана книга являє собою практичний курс з написання програмного забезпечення на базі другої версії Android SDK. Безсумнівно, книга сильно застаріла на даний момент, але все ще є хорошим структурованим керівництвом з вивчення платформи. Оскільки безліч застосунків створюється з умовою сумісності з попередніми версіями ОС Android, багато прикладів з даної книги не втратили актуальності.

Будь-який Android-застосунок, містить в собі хоча б один з так званих App Components - компонентів програми. Існують чотири види основних компонентів програми: Activity, Service, Content Provider і Broadcast Receiver. У перекладеній літературі зазвичай використовується калька з англійської, наприклад, Activity - «Активність». Оскільки термін «Активність» не зовсім відповідає тому, для чого використовується клас Activity, в більшості інтернет-джерел використовується транскрипція термінів, наприклад, Activity - «Активіті».

Activity призначена для інтерфейсу користувача. Всі призначені для користувача Activity є класами, успадкованими від стандартного класу Activity. Можна провести деяку паралель з технологією Windows Forms, проте, оскільки Java не підтримує часткові (partial) класи, в Android-застосунках використовується інший спосіб зв'язування елементів інтерфейсу і призначеного для користувача класу- Activity. Кожна Activity володіє певним життєвим циклом (рис. 2.1), який є точною демонстрацією специфіки мобільних застосунків. Оскільки ресурси мобільних пристроїв обмежені, достовірно можна сказати лише те, що в пам'яті існують лише ті об'єкти, які так чи інакше відносяться до відображеного на екрані вмісту. Другою особливістю Activity є те, що кожен раз при зміні орієнтації пристрою з альбомної на портретну або навпаки, Activity знищується і створюється заново. Складно сказати, чим продиктоване таке рішення, швидше за все це зроблено для того, щоб забезпечити підтримку різних варіантів користувацького інтерфейсу для різної орієнтації пристрою. Дана особливість породжує другу важливу проблему - проблему збереження стану при зміні конфігурації (наприклад, при повороті екрану) [10].

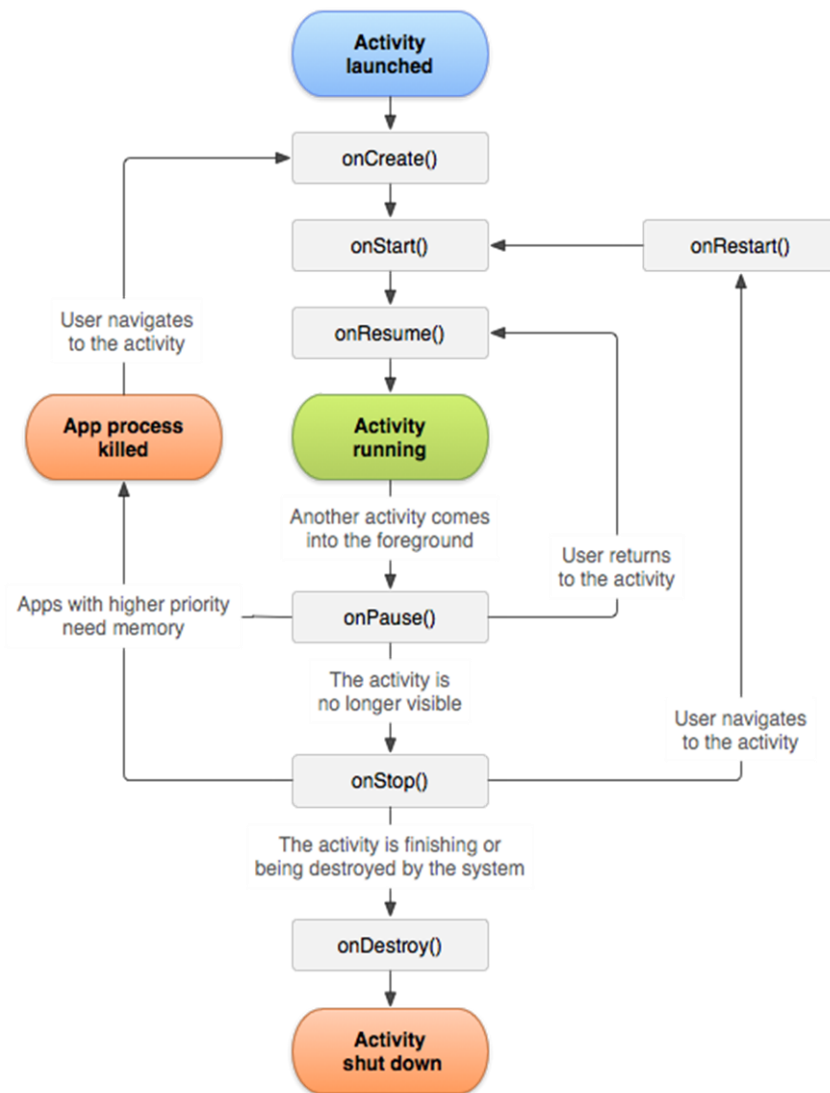


Рисунок 2.1 - Життєвий цикл Activity [8]

Сервіс – це компонент, який працює в фоні і використовується для тривалих і ресурсоемних операцій. Відмінною особливістю сервісу є можливість запуску в привілейованому режимі (foreground mode), завдяки якому ОС не завершує додаток в тлі максимально довгий час. Привілейований режим зазвичай використовується для запису або відтворення потокового вмісту, а також для скачування або завантаження файлів [9].

Ще одним часто використовуваним компонентом є Broadcast Receiver. Broadcast receiver – це компонент, службовець для обробки широкомовних повідомлень. ОС Android використовує спеціальні об'єкти для сполучення між компонентами - INTENT (intent). Іноді в літературі зустрічається прямий

переклад - «наміри». Intent – це об'єкт класу Intent представляє собою деяке повідомлення. З кожним intent може асоціюватися якийсь код помилки, дія, URI (Uniform Resource Identifier), а також довільний набір об'єктів, що пересилаються. Об'єкти, так звані extras, зберігаються в асоціативному контейнері з унікальними рядковимим ключами. Підтримуються всі примітивні типи, рядки, а також об'єкти, що реалізують інтерфейси Serializable або Parcelable. INTENT можуть бути вузьконаправленими, наприклад, для запуску сервісу або Activity, або ж широкомовними, які зазвичай використовуються в якості нотифікації про будь-яку подію. INTENT широко використовуються в ОС Android: Завдяки широкомовній INTENT, за допомогою широкомовних приймачів можна підписуватися на різні системні події, від спрацювання будильника і до переходу за посиланням в браузері [11].

Останнім нерозглянутим компонентом є постачальник контенту – content provider. Це, мабуть, найменш часто використовуваний компонент, службовець для надання доступу до деякого сховища даних. Існують системні постачальники контенту для доступу до телефонних контактів, отримання інформації про встановлені додатки. Доступ здійснюється за допомогою URI за аналогією з доступом до веб-сторінок. Низька популярність даного компонента пов'язана з його специфікою: далеко не кожен додаток має необхідність в наданні доступу до своїх даних [12].

Елементом, що об'єднує розрізнені компоненти в єдиний Android-застосунок, є маніфест. Маніфест — це спеціальний обов'язковий XML-файл, який містить перелік всіх компонентів програми, мінімальну і використовувану для збірки версії ОС, а також необхідний для роботи дозвіл і необхідні можливості пристрою (наприклад, наявність фронтальної камери або певної версії OpenGL ES).

Крім вивчення роботи з основними компонентами, кожен розробник повинен ознайомитися з основними шаблонами проєктування ОС Android. Найбільш часто використовуваними стандартними шаблонами проєктування є спостерігач (observer) і слухач (listener). Також широко використовуються такі

принципи як обмін повідомлень та пізніше зв'язування. Крім цього, Android диктує свої правила при реалізації деяких стандартних процесів, тим самим змушуючи розробникам слідувати певним специфічним шаблонами проектування.

Яскравим прикладом цього можуть бути завантажувачі (loaders). Завантажники є нащадками класу `Loader` і необхідні для асинхронного завантаження даних. Головною перевагою завантажників є те, що їх життєвий цикл не залежить від інших компонентів [8]. Для доступу до завантажувача використовується спеціальний компонент `Loader Manager`, який може працювати з завантажувачами будь-яких класів, які реалізують інтерфейс `Loader Manager Loader Callbacks`. Незважаючи на деякі вбудовані у фреймворк реалізації, розробникам досить часто доводиться реалізовувати свої специфічні завантажувачі, наприклад, для роботи з локальною базою даних або отримання масивів інформації по мережі [9].

Існує чотири основні інтегровані середовища розробки (Integrated development environment, далі IDE) з підтримкою Android-проектів:

- Eclipse + ADT Plugin;
- IntelliJ IDEA;
- NetBeans;
- Android Studio.

Довгий час найчастіше використовувалися перші дві IDE. Eclipse є скоріше платформою для створення універсальних середовищ розробки, заснованих на інтегруванні спеціальних модулів розширення - плагінів. Відповідний плагін для Android-розробки носить ім'я ADT Plugin і володіє дуже великим функціоналом. Спочатку саме його широка функціональність принесла популярність інструментам Eclipse + ADT Plugin, оскільки вони дозволяють редагувати користувальницький інтерфейс в WYSIWYG стилі. Незважаючи на це, останнім часом Eclipse сильно поступилася IntelliJ IDEA в зв'язку з бурхливим і динамічним розвитком останньої. IntelliJ IDEA також підтримує

розширення у вигляді плагінів і дозволяє створювати Android-проєкти, редагувати та проводити рефакторинг вихідного коду в порівнянні з Eclipse.

Найменш поширеною IDE є NetBeans, оскільки вона використовується в основному користувачами Linux, які раніше застосовували NetBeans для своїх проєктів.

На даний момент досить перспективно виглядає IDE від Google – Android Studio. Вона позиціонується як спеціалізований засіб для Android-розробки від компанії розробника самої ОС. Android Studio ґрунтується на IntelliJ IDEA і володіє її перевагами. Також Android Studio містить спеціалізовані засоби розробки і аналізу коду, специфічні для Android-розробки. При розробці даної роботи використовувалася саме ця IDE [10].

Відмінною особливістю Android Studio є використання системи збирання Gradle. Її опис можна знайти на офіційному сайті [11] або ж на ресурсі [12], стосовно конкретно до Android Studio. До особливостей Gradle, варто віднести «build by convention», що використовуватиметься для всіх популярних аналогів, наприклад, Maven, з використанням скриптів на мові Groovy. Подібний підхід дозволяє легко створювати багатомодульні збірки, що вельми актуально для Android-проєктів, що використовують для бібліотеки. Ще однією можливістю Gradle, що активно використовується в Android Studio є можливість створювати засновані на Gradle проєкти. Подібна лаконічна структура досить ефектно виглядає при використанні системи контролю версій: фактично всі файли проєкту є функціонально необхідними і не захищаються репозиторієм.

Варто відзначити, що всі вище перераховані середовища розробки або є повністю безкоштовними, або мають безкоштовні версії.

У роботі планується використання взаємодії з базою даних (БД). На ОС Android в якості БД використовує SQLite. SQLite є простою реляційною базою даних, використовуваною на всіх популярних мобільних платформах (iOS, Windows Phone). SQLite володіє певними особливостями, ознайомлення з якими необхідно навіть при наявності досвіду роботи з іншими реляційними

БД. SQLite містить всього лише чотири типи даних і не підтримує збережені процедури [9].

2.3 Висновки до розділу

В якості середовища розробки було обрано Android Studio. Однією з ключових особливостей дипломної роботи, є робота з візуальною інформацією, тобто використання камери. Незважаючи на вичерпний опис процесу роботи з камерою як в офіційній документації, так і в літературі, камера залишається вельми специфічним пристроєм. В основному це пов'язано з великою різноманітністю використовуваних фотомодулів, так чи інакше впливає на стандартний процес взаємодії. На жаль, не існує єдиного джерела, що описує всі, або хоча основні, підводні камені в роботі з камерою, проте в якості корисного ресурсу, багато в чому допомагає розробці, в тому числі і в питаннях взаємодії з камерою.

3 ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ ЗАСТОСУНКУ

3.1 Модулі проєкту

В даній роботі можна виділити чотири основні компоненти, три з яких асоційовані з основними Activity застосунками, а четвертий відповідає за фонову обробку даних. Структурна схема програми представлена на рис. 3.1

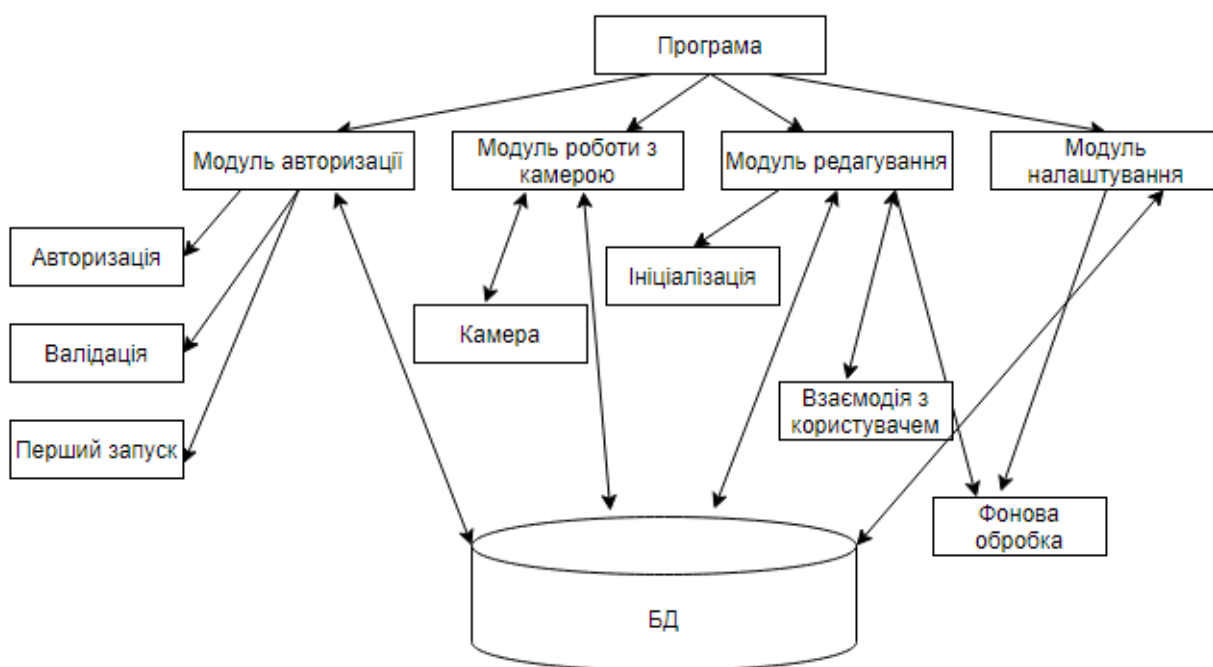


Рисунок 3.1 – Структурна схема програми

На схемі зображені основні модулі програми та їх функції. Поряд з основними модулями, які логічно виділяються виходячи з вхідних вимог до додатка, нескладно визначити набір додаткових модулів. До них можна віднести модуль налаштувань, присутній в більшості Android-застосунків, а також інтерфейс для взаємодії з користувачем.

Застосунок також використовує БД для кешування частини даних з сервера, тому логічно буде створити окремий модуль для роботи з БД. Застосунок має невелику кількість фонових операцій, тому, на одному з етапів, немає необхідності у виділенні окремих сервісів для роботи з БД і веб-

сервером. Android дозволяє використовувати окремі компоненти – завантажувачі для асинхронного отримання масивів даних. Дані завантажувачі мають автономний життєвий цикл та є зовнішніми компонентами по відношенню до модуля взаємодії з БД, тому їх виділимо в окремий модуль. Тому, щоб підкреслити функції фонового сервісу, як контексту виконання операцій, веб-клієнт також слід винести в окремий модуль.

Модуль авторизації, асоційований з Activity, складається з двох частин, однією з яких є уявлення, а іншою клас-контролер, успадкований від класу Activity. Дані файли завантажуються динамічно класом-контролером, між ними все ще існує неявний, односторонній зв'язок. У той час як представлення не має інформації про клас-контролер та про структуру класів. Клас-контролер оперує певними елементами представлення. Спроба отримати елемент, відсутній в розмітці, призведе до виключення під час виконання. Крім цього, слід враховувати, що для отримання об'єктів, асоційованих з елементами управління використовується метод `find View ById`, що приймає на вхід унікальний числовий ідентифікатор, визначений у розмітці, і повертає базовий тип `View`. Таким чином, кожен об'єкт, отриманий за допомогою даного методу, необхідно приводити до конкретного типу.

Activity авторизації є першим, що постає перед користувачем програми. Крім своїх основних функцій (надання можливості для введення логіна і пароля, а також їх валідації) даний модуль також виконує функцію визначення першого запуску програми.

Архітектурою Android-застосунку не передбачений метод, який викликається при першому початку роботи з програмою, або ж хоча б просто при старті програми. Внаслідок цього, часто обробка подібних ситуацій проводиться в початковому Activity, тобто, в даному випадку, в модулі авторизації. Крім того, в Activity авторизації використовуються різні розмітки для альбомної і портретної орієнтації екрану.

Модуль роботи з камерою зроблений на базі основної, другої за рахунком Activity, яка постає перед користувач. В даному випадку, модуль

роботи з камерою – це щось більше, ніж просте представлення. Android надає певні можливості для написання призначених для користувача застосунків, що працюють безпосередньо з камерою. Рішення передбачає використання певного елемента управління та реалізації кількох інтерфейсів зворотного виклику для отримання подій від камери і елементів уявлення. Також, важливим є те, що поворот екрану призводить до перестворення компонента представлення, що відповідає за виведення відео потоку з камери. Певну складність представляє собою власна обробка визначення положення екрану та обробка його повороту [7].

Ключовим модулем програми є модуль редагування метаданих. В роботі модуля можна виділити три основні стадії, в ході яких він взаємодіє з різними частинами програми. На етапі ініціалізації модуль отримує вихідні дані у вигляді знімка від модуля роботи з камерою, а також встановлює зв'язок з модулями завантаження даних для асинхронного отримання кешованої інформації. У наступній фазі своєї роботи, модуль забезпечує взаємодію з користувачем: надає можливості для заповнення форми метаданих, а також забезпечує збереження введених даних до моменту відправки. У завершальній фазі модуль стає слухачем подій від фонових сервісів для того, щоб показувати користувачеві прогрес виконання завдання та надати можливість зворотного зв'язку в разі виникнення помилок.

Модуль налаштувань є здебільшого системною компонентою, яка надає доступ до захищеного файлу налаштувань програми, доступному в контексті програми. Залежно від складності застосунку, модуль, який відповідає за взаємодію з користувачем може мати вигляд не тільки тонкого клієнта, а й досить складної системи, яка включає в себе набір власних елементів представлення для редагування налаштувань, а також набору компонент, відповідальних за швидке реагування на зміну налаштувань. В даному проєкті модуль призначеного для користувача інтерфейсу не складний та надає можливості для редагування досить загальних властивостей, важливих в контексті всієї програми, а не конкретної сесії. З цієї причини програма надає

лише одну точку входу в інтерфейс налаштувань. Модуль налаштувань досить широко використовується основними компонентами системи.

Досить відокремленим виглядає модуль роботи з БД. Це наслідок шаблонів проєктування ОС Android. Для взаємодія серверу та БД описана наступною моделлю, рис. 3.2.

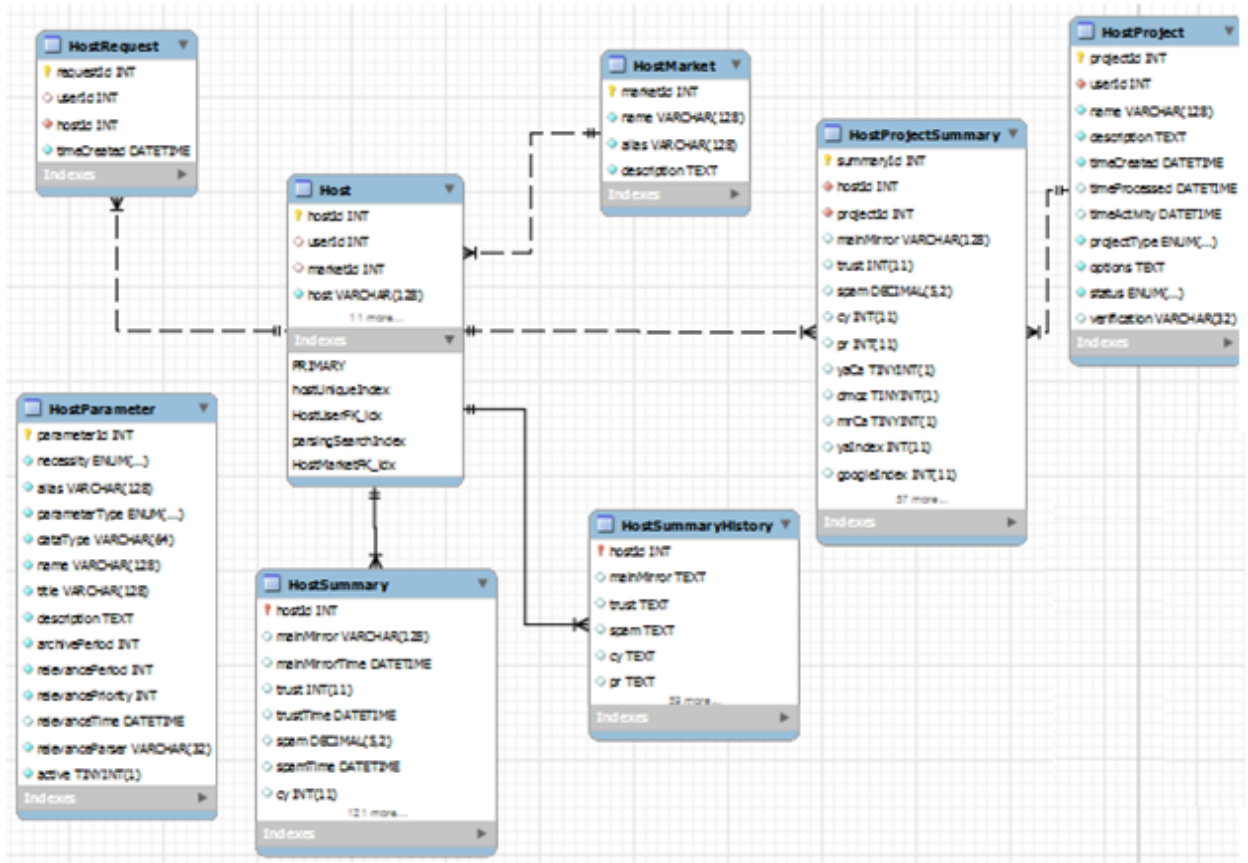


Рисунок 3.2 – Взаємодія з сервером

Платформа надає можливості для автономної ініціалізації БД при першому запуску програми, а також дозволяє вести версію БД. Також, є можливість реалізації постачальника вмісту (Content Provider), що надає зовнішній інтерфейс для взаємодії з БД. Постачальник вмісту можна трактувати як аналог сервісу, що представляє публічний API для будь-яких зовнішніх застосувань. Існує можливість обмеження доступу до постачальника. Можна дати права на підключення лише деяким застосункам або вимагати запиту дозволів від зовнішніх застосувань, які планують контактувати з

постачальником, або ж обмежити видимість постачальника. В останньому випадку, фактично, необхідність в постачальнику вмісту відпадає, і програміст може сам вирішувати, як реалізовувати інтерфейс для взаємодії з БД. В даному проєкті БД використовується для кешування даних сервера та немає необхідності робити доступними ці дані для зовнішніх компонент. Модуль роботи з БД надає набір відкритих методів для асинхронного оновлення вмісту БД, а також надає можливість для підпису класу-оглядача на події від БД. Процес підключення до БД для читання даних зображень на рис. 3.2 та виглядає наступним чином:

- модуль-ініціатор читання (модуль редагування метаданих) запитує у менеджера завантаження створення нового об'єкта-завантажувача, вказуючи події, до яких буде підключений клас оглядач, а також клас-слухач, що викликається по готовності даних;
- створюється новий автономний завантажувач, який, через проксі-об'єкт, підписується на події поновлення певних даних в БД. Якщо менеджер завантаження бачить, що необхідний завантажувач вже був створений, він віддає існуючий замість створення нового;
- по готовності завантажувача, а також при подальших подіях від БД викликається метод зворотного виклику в класі-ініціатора читання, в який передаються отримані дані.

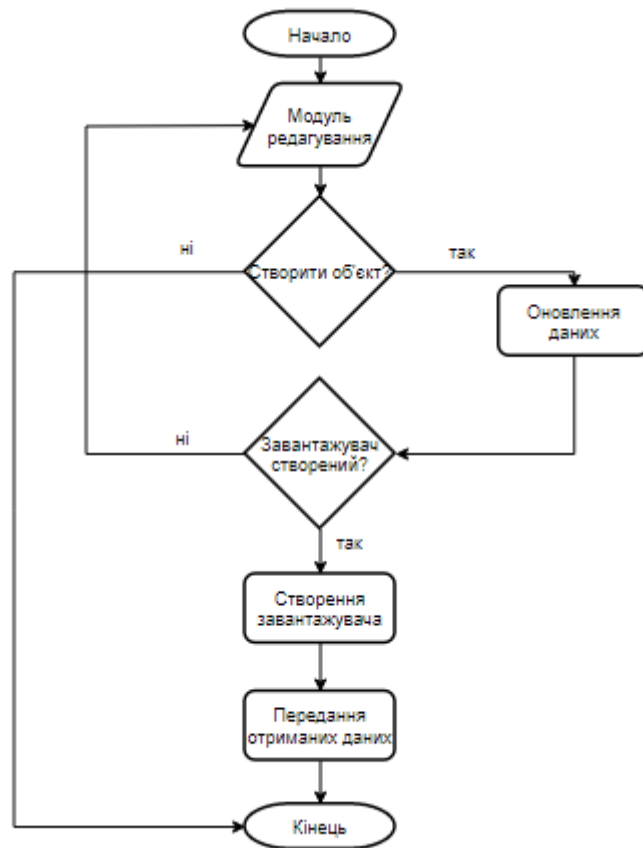


Рисунок 3.3 – Блок-схема взаємодії з БД

Менеджер завантажувача – це системний об'єкт класу Loader Manager, який необхідний для взаємодії з конкретними завантажувачами (спадкоємцями класу Loader), а також забезпечує підтримку їх автономного життєвого циклу. У той же час, конкретні завантажувачі є класами, що реалізують необхідні методи для обробки даних, а також надають можливість підписки на довільні події ззовні.

Ще одним модулем є веб-клієнт. В даному випадку, цей клас реалізує взаємодію з веб API на стороні клієнта. Основними його функціями є встановлення з'єднання з сервером, виклик віддалених веб-методів, а також перетворення структур даних від використовуваних застосунком абстракцій до встановленого формату веб API. Основною причиною винесення веб-клієнта в окремий модуль є універсальність модуля обробки фонових операцій. Таким чином, виділення окремих компонентів: веб-клієнта і модуля роботи з БД,

підкреслює їх спеціалізацію, а також вказує на роль модуля фоновієї обробки даних, як контексту виконання фонових операцій.

Модуль фоновієї обробки операцій побудований на базі Android-сервісу та призначений для здійснення витратних за часом операцій поза основного потоку виконання. Крім основного свого призначення, модуль визначає слабо зв'язаний інтерфейс для обміну з іншими великими модулями засунку через Intent-аналіз. Типовий хід роботи програми не визначає паралельних операцій, тому сервіс використовується як черга для обробки ресурсномістких операцій в окремому потоці. Для цього використовується клас Handler. За допомогою успадкування від цього класу, можна отримати готову чергу подій з призначенням для користувача оброблювачем повідомлень – об'єктів класу Message, здатних передавати в обробник всі необхідні дані. Екземпляр підкласу Handler може бути асоційований з будь-яким потоком, що дозволяє організовувати межпотоківу комунікацію для забезпечення зворотного зв'язку з основним потоком виконання.

В даній роботі використовується інший спосіб комунікації, заснований на Intent-аналізі. Подібний підхід дозволяє більш просто організувати доступ до сервісу з декількох модулів програми. Але ОС Android надає часткову реалізацію подібного шаблону обробки повідомлень - клас Intent Service. Об'єкт даного класу є Android-сервіс, який при старті запускає окремий фоновий потік виконання та асоціює з ним об'єкт Handler, залишаючи користувачу право самому визначити, як будуть оброблятися отримані дані. Якщо один або кілька Intent приходять під час обробки попередніх, то вони автоматично додаються в кінець поточної черги обробки операцій. IntentService самостійно поміщає Intent в об'єкти-повідомлення для постановки в чергу обслуговування і витягує їх безпосередньо передаючи в метод-обробник. Особливістю даної реалізації є те, що Intent Service підтримує лише одну чергу обробки повідомлень, а також автоматично завершується після виконання всіх операцій з черги для економії ресурсів телефону.

Таким чином, можна виділити дві роздільні послідовності даних, перша з яких використовується для поновлення стану кеша, а друга – для читання даних з БД рис. 3.4 та рис. 3.5.

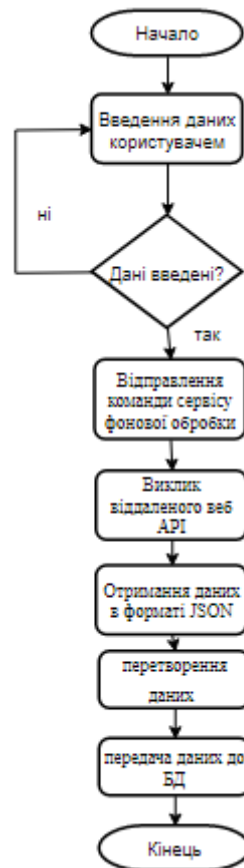


Рисунок 3.4 – Алгоритм для поновлення стану кеша

Послідовності незалежні одна від одної та ініціюються різними модулями. Крім того, одночасно запускається відразу кілька послідовностей другого типу, при цьому вони можуть почати своє виконання не чекаючи закінчення першої послідовності. Дана схема є працездатною за рахунок того, що операції оновлення БД ініціюються модулем фоновій обробки операцій, а операції читання даних з БД виконуються модулями завантаження даних після того, як модуль роботи з БД видасть повідомлення про успішне оновлення даних.



Рисунок 3.5 – Алгоритм читання з БД

3. 2 Функціональне проєктування

3.2.1 Опис взаємодії з сервером

Однією з головних функцій мобільного клієнта є обмін даними з сервером. Для цього був розроблений інтерфейс, який описує набір веб-методів для комунікації між компонентами – інтерфейс програмування застосунків. API базується на трьох основних принципах:

- формат переданих даних – JSON;
- інформація про користувача передається з кожним запитом за допомогою базової аутентифікації HTTP (Basic Authentication);
- для відправлення даних використовуються HTTP POST запити.

На даний момент сервер надає п'ять веб-методів доступних клієнтові, опис яких представлено нижче. Методи перераховані в порядку їх виклику застосунком. Після натиснення кнопки «Увійти» на екрані авторизації, викликається метод login для перевірки прав доступу. Якщо доступ отримано, застосунок по черзі викликає методи stores і categories для поновлення локального кешу, в той час користувач може робити знімок. Після того як

користувач зробить фото, він потрапляє на сторінку метаданих, заповнивши які, відправляє фото на сервер, здійснюється виклик методів `upload _ photo` і `save _ photo`.

Даний метод (`../api/login`) використовується для початкової авторизації користувача і отримання його ролі, яка може бути використана в подальшому. На поточний момент існує три ролі: «user», «merchandizer» та «admin». Будь-яка з цих ролей дозволяє користуватися застосунком, хоча в подальшому права можуть бути змінені, зокрема, деякі функції можуть стати недоступними для деяких ролей.

Метод `/api/stores` повертає список магазинів, оновити після запитаної версії (поле «version» з вхідних даних). При кожній зміні даних про магазин на сервері повинен збільшуватися номер версії. Номер версії є глобальним для клієнта і сервера лічильником. Клієнт при запиті відправляє максимальну версію з наявних у нього. Важливою особливістю системи є те, що дані про магазини ніколи не видаляються, проте їм можна присвоювати неактивний статус (поле «active» з відповіді сервера). Неактивні магазини не відображаються при виборі точки продажу на клієнті.

Метод `api / categories` повертає список категорій, оновлений після запитаної версії. (Поле «version» з вхідних даних). Метод аналогічний попередньому, але замість списку магазинів отримує список категорій товарів. Номер версії також є глобальним для клієнта і сервера, а категорії, аналогічно магазинам, не видаляються, а переводяться до неактивного режиму.

Метод завантаження фотографії (`api / upload_photo`) використовується для завантаження фотографій на сервер. Після завантаження сервер генерує унікальний ідентифікатор для файлу (GUID) та зберігає його під даним ім'ям в сховище, після чого повертає ідентифікатор як відповідь на запит. Клієнт після успішного завантаження файлу використовує ідентифікатор для передачі додаткової інформації, пов'язаної з файлом при виклику наступного методу.

Метод `/ api / save_photo` використовується для збереження офлайн попередніх фотографій з ідентифікатором, переданим в якості параметра «file»

запиту. В даному випадку в тілі POST-запит передаються метадані знімка, які включають в себе наступні елементи:

- ідентифікатор магазину («store»);
- ідентифікатор категорії («category»);
- довільний коментар («comment»);
- спеціальне поле «was_taken», що дозволяє визначити знімок як зроблений до, або після роботи; можливі значення: «unknown» (по замовчуванню), «before» або «after»;
- ідентифікатор файлу («file»);
- координати точки, в якій була зроблена фотографія (структура «coordinates»), що включають в себе широту («latitude»), довготу («longitude») і точність визначення («accuracy»).

У разі неправильного логіна або пароля, будь-який з вище перерахованих методів повертає помилку у вигляді HTTP відповіді з кодом 401 - «Unauthorized». Відповідь сервера зроблена не пустою на випадок, якщо знадобиться додавання розширених повідомлень про помилки.

3.2.2 Опис класів системних компонентів

Клас системних компонентів, пов'язаних з даним проєктом, є клас Worker Service, успадкований від класу Intent Service, а також чотири класи, успадковані від класу Activity: Login Activity, Camera Activity Settings Activity, та Data Activity. Особливістю даних класів є те, що вони описують той чи інший великий компонент застосунку, важливий, в першу чергу, для ОС.

Для чотирьох Activity-класів можна виділити кілька загальних методів, визначених в базовому класі і часто переобумовлених в класах спадкоємців. Ці методи викликаються в різні моменти життєвого циклу Activity та служать для коректної обробки зміни стадій життєвого циклу:

- void onCreate (Bundle savedInstanceState) – метод, що викликається при створенні Activity. У ньому здійснюється створення і розміщення (initializing)

візуальних компонентів, а також початкова ініціалізація компонентів. Даний метод можна розглядати як аналог конструктора, з тією різницею, що на момент виклику Activity не тільки створена, але і відображатися на екрані телефону. Об'єкт типу Bundle є контейнером, що містить пари ключ-значення та служить для відновлення параметрів при перестворенні Activity після повороту екрану або інших змін конфігурації;

- `boolean on Create Options Menu (Menu menu)` – метод, який викликається при створенні контекстного меню Activity, пункти якого додавалися до об'єкта menu. На даний момент шаблон проєктування з використанням контекстного меню застарів і в цьому методі проводиться ініціалізація так званої «рядки дій» (Action Bar), що являє актуальним на даний момент дизайнерським шаблоном для надання контекстно залежних опцій. На відміну від контекстного меню, рядок дій, зазвичай, ініціалізується не в коді, а в файлі розмітки та повертається методом значення служить індикатором для створення меню: якщо метод поверне false, то контекстне меню створено не буде, навіть якщо воно буде ініціалізувати значеннями. Така поведінка справедливо також і для рядка дій;

- `boolean on Options Item Selected (MenuItem item)` – метод викликається при виборі пункту меню або дії з рядка дій (в дійсності, це одне й те саме: різниця лише в термінології і способі відображення). Об'єкт типу Menu Item асоційований з візуальним елементом, що обробляють натискання, а повертається значення є індикатором закінчення обробки натискання: якщо метод повертає false, для певного пункту меню, значить натискання на нього буде оброблено в іншому місці;

- `void on Resume ( )` – метод, який викликається безпосередньо перед відображенням Activity на екрані. В даному методі зазвичай проводиться відновлення всіх тимчасово перерваних процесів;

- `void on Pause ( )` – метод, що викликається при переході Activity в неактивний режим. Іншими словами, кожен раз, коли Activity пропадає з екрану. В даному методі зазвичай припиняються всі пов'язані з призначенням

для користувача інтерфейсом процеси, а також проводиться збереження стану важливих змінних;

– void on Activity Result (int request Code, int result Code, Intent data) – даний метод представляє собою обробник результатів, отриманих від дій ініційованих за допомогою Intent, наприклад, таких, як виклик Activity, що містить детальне уявлення або отримання інформації від датчиків телефону. В даному випадку request Code – унікальний ідентифікатор Intent, result Code – код результату, зазвичай показує успіх або помилку, data – безпосередньо сам Intent з результатами виконання запиту;

– void on Click (View view) – даний метод не є частиною класу Activity, проте дуже часто використовується в його спадкоємців. Даний метод є єдиним компонентом інтерфейсу View. On Click Listener, який повинні реалізувати об'єкти-слухачі подій-кліків по візуальним елементам управління. Існує два основні підходи до обробки кліків, кожен з яких має свої переваги і недоліки: або для кожного елемента управління створюється свій клас-слухач (зазвичай анонімний), або Activity реалізує інтерфейс View. On Click Listener і стає слухачем для всіх своїх елементів управління. В даному проєкті використовується другий спосіб.

Крім цих методів, можна відзначити наявність в класах Activity цілочислених констант виду «ACTION_CODE_ XXX » і строкових констант виду «YYY_DIALOG_TAG».

Константи першого типу являють собою унікальні ключі, які використовуються при створенні Intent. Вони необхідні для ідентифікації результатів, отриманих в якості відповіді на певні Intent. Ідентифікатор, заданий при створенні Intent буде надалі переданий в метод on Activity Result як параметр requestCode. Значення цілочисельних констант формуються з трьох цифр на зразок кодів HTTP відповідей. Перша цифра показує компонент (клас), якому належить константа, друга – область застосування константи (ідентифікатор INTENT, ідентифікатор завантажувача і т.д.), третя –

порядковий номер константи в рамках заданого компонента і області застосування.

Константи другого типу є ідентифікатори-теги діалогових вікон. Згідно з новими шаблонами проєктування, діалогові вікна представляють собою фрагменти, успадковані від базового класу `Dialog Fragment`. Це зроблено для того, щоб не зберігати стан діалогових вікон кожен раз, коли користувач повертає екран або згортає програму натисканням кнопки додому. Діалогові вікна, засновані на фрагментах, мають свій життєвий цикл, що не залежить від життєвого циклу `Activity`, тому вони можуть бути створені одним екземпляром класу, а використовуватися іншим. Для того, щоб не було плутанини, кожен діалоговий фрагмент відзначається тегом, за яким його завжди можна знайти.

Клас `Login Activity` асоційований з `Activity` авторизації – першим, що бачить користувач програми. Основними його функціями є авторизація користувача і обробка самого першого запуску програми.

Константи:

- `ACTION_CODE_LOGIN` – ідентифікатор `INTENT`, асоційованого з дією авторизації;
- `PROGRESS_DIALOG_TAG` – тег діалогового фрагмента, що використовується для відображення прогресу виконання завдання;
- `REPORT_DIALOG_TAG` – тег діалогового фрагмента, що використовується для відображення інформації про помилки;
- `REDIRECT_DIALOG_TAG` – тег діалогового фрагмента, що використовується для переадресації на сторінку побудови при першому запуску програми.
- `SAVE_CREDENTIALS_PROPERTY` – константа-ключ для доступу до опції запам'ятовування логіна і пароля.

Поля:

- `Save Credetials` – поле типу `boolean`, що містить актуальне значення опції запам'ятовування логіна і пароля;

- Login Edit Text – поле типу Edit Text, що містить посилання на текстове поле для введення імені користувача (логіну);
- Password Edit Text – поле типу Edit Text, що містить посилання на текстове поле для введення пароля;
- Save Credentials Check Box – поле типу Check Box, що містить посилання на прапорець опції запам'ятовування логіна і пароля;
- Signin Button – поле типу Button, що містить посилання на кнопку «Увійти»;
- Progress Dialog – поле типу Progress Dialog Fragment, що містить посилання на діалоговий фрагмент, який використовується для відображення прогресу виконання завдання;
- Preferences – поле типу Shared Preferences, що містить посилання на модуль налаштувань.

Методи:

- void try Begin Authorize User () – метод, який здійснює валідацію призначеного для користувача введення перед операцією авторизації, а також відображає результати валідації;
- void begin Authorize User (String login, String password) – метод, що запускає процес асинхронної авторизації. В якості вхідних параметрів виступають введені користувачем логін і пароль;
- void end Authorize User (String role) – метод, що викликається при закінченні авторизації. Вхідним параметром є роль користувача;
- void navigate To Camera Activity () – метод, який здійснює перехід до Activity управління камерою;
- void before Text Changed (Char Sequence char Sequence, int i, int i2, int i3) – метод інтерфейсу Text Watcher, не використовується;
- void on Text Chan ged (Char Sequence char Sequence, int i, int i2, int i3) – метод інтерфейсу Text Watcher, невикористовується;

– void after Text Changed (Editable editable) – метод інтерфейсу Text Watcher.

Викликається після введення кожного символу. Розблокує кнопку signin Button при наявності логіна і пароля;

– boolean on Editor Action (Text View text View, int i, Key Event key Event)

– метод інтерфейсу Text Watcher. Даний метод обробляє натискання спеціальних кнопок клавіатури, наприклад, кнопки «Далі»;

– void report Error (int message Id) – метод, який здійснює відображення діалогу з інформаційним повідомленням. Повідомлення зберігається в ресурсах застосунку, в той час як метод приймає ідентифікатор ресурсу – message Id;

– void show Redirect Dialog () – метод, який здійснює відображення діалогу для перенаправлення на сторінку налаштувань;

– void restore Properties () – метод, який здійснює відновлення раніше введених даних;

– void save Preferences () – метод, який здійснює збереження поточного стану Activity;

– void on Click (Dialog Interface dialog Interface, int i) – метод інтерфейсу Dialog Interface. On Click Listener, що викликається при натисканні на кнопку в діалоговому вікні.

Клас SettingsActivity асоційований з Activity налаштувань і служить для відображення та роботи з опціями програми. Клас містить єдиний метод void on Create (Bundle savedInstanceState). В даному методі створюється фрагмент Settings Fragment, який заповнює всю корисну площу Activity.

Фрагмент Settings Fragment, в свою чергу, є спадкоємцем спеціального класу Preference Fragment, що надає зручні засоби для роботи з налаштуваннями програми. Клас Settings Fragment також містить єдиний метод void on Create (Bundle savedInstanceState). В даному методі фрагмент створює і розміщує елементи управління на основі спеціального XML -файлу, що описує розмітку. Крім цього, тут же не започатковано обробники подій для візуальних елементів, що представляють ту чи іншу опцію.

Клас Camera Activity асоційований з Activity управління камерою. Основними його функціями є відображення на екрані відеопотоку з камери, робота з настройками камери, отримання знімків, а також обробка поворотів екрану телефону.

Константи:

- PROPERTY_SIZE_INDEX – константа-ключ для доступу до опції розміру зображення;

- PROPERTY_FLASH – константа-ключ для доступу до опції спалаху.

Поля:

- camera – поле типу Camera, що містить посилання на об'єкт-камеру;

- cameraId – поле типу Camera, що містить ідентифікатор камери;

- Bottom Panel – поле типу Linear Layout, що містить посилання на панель з кнопками вибору знімка;

- Bottom Panel Hidden use Auto Focus – прапор типу boolean, що показує стан панелі bottom Panel;

- Camera Button Layout – поле типу Relative Layout, що містить посилання на панель – контейнер для кнопки знімка;

- Camera Button – поле типу Button, що містить посилання на кнопку знімка;

- Camera IconImage – поле типу Image View, що містить посилання на іконку кнопки знімка;

- Orientation Observer – поле типу Orientation Event Listener, що містить посилання на оглядач зміни орієнтації дисплея;

- Icon Orientation – поле типу int, що містить поточну орієнтацію дисплея;

- Picture Sizes – поле, що містить масив елементів типу Camera. Size, відповідних трьом (або менше) мінімальним дозволами камери телефону;

- Picture Size Index – поле типу int, що містить позицію поточного дозволу знімка в списку pictureSizes;

- preferences – поле типу Shared Preferences, що містить посилання на модуль налаштувань;
- Use Auto Focus – прапор типу boolean, вказує на можливість використання автофокусу при зйомці;
- Use Flash – прапор типу boolean, вказує на необхідність використання спалаху під час зйомки;
- Has Flash – прапор типу boolean, вказує на наявність спалаху біля телефону;
- Action Bar – поле типу ActionBar, що містить посилання на панель меню;
- Settings Button – поле типу Button, що містить посилання на кнопку настройки роздільної здатності знімка;
- Settings Image View – поле типу Image View, що містить посилання на іконку кнопки настройки роздільної здатності знімка;
- Log Out Button – поле типу Button, що містить посилання на кнопку виходу з облікового запису;
- Log OutImage View – поле типу Image View, що містить посилання на іконку кнопки виходу з облікового запису;
- Flash Button – поле типу Button, що містить посилання на кнопку налаштування спалаху;
- Flash Image View – поле типу Image View, що містить посилання на іконку кнопки налаштування спалаху;
- Popur Window – поле типу Popur Window, що містить посилання на спливаюче вікно зі списком доступних дозволів знімка;
- Popur Window Center – поле типу int, що містить горизонтальну координату центру спливаючого вікна popurWindow;
- Surface Holder – поле типу Surface Holder, що містить посилання на спеціальний об'єкт для взаємодії з візуальним компонентом для відображення потокового відео.

Методи:

- `int find Back Facing Camera ()` – метод, який здійснює пошук ідентифікатора основної камери телефону;
- `Camera try Get Back Camera ()` – метод, який повертає об'єкт-камеру, за умови її наявності;
- `void close Camera ()` – метод, що вивільняє об'єкт-камеру;
- `boolean get Flash Availability ()` – метод, що обчислює кут повороту камери;
- `Camera. Size [ ] get Picture` Перейти до оголошення `Sizes ( Camera camera )` – метод, який повертає список з трьох або менше мінімальних розподілень камери;
- `void set Picture` Перейти до оголошення `Size ( int index )` – метод, який встановлює один з розподілень зі списку `picture Sizes`;
- `void set Flash Mode ( boolean use Flash )` – метод, що включає і вимикає спалах;
- `void set Up Camera Parameters ()` – метод, який здійснює установку необхідних для зйомки параметрів камери;
- `void reset Camera Preview ( Surface Holder holder )` – метод, який здійснює переініціалізацію візуального компонента для виведення потоку з камери;
- `void onItemClick ( AdapterView <?> AdapterView, View view, int i, long l )` – метод інтерфейсу `List View. On Item Click Listener`, який здійснює обробку кліків по елементах списку `List View`;
- `void navigate To Login Activity ()` – метод, який здійснює перехід до `Activity` авторизації;
- `void navigate To Data Activity ()` – метод, який здійснює перехід до `Activity` метаданих;
- `void surface Created (Surface Holder surface Holder)` – метод інтерфейсу `Surface Holder. Callback`, що викликається при зміні розмірів елемента управління `Surface View`;

- void surface Changed (Surface Holder surface Holder, int i, int i2, int i3) – метод інтерфейсу Surface Holder. Callback, що викликається при знищенні елемента управління Surface View;
- void surface Destroyed (Surface Holder surface Holder) – метод інтерфейсу Surface Holder. Callback, що викликається при створенні елемента управління Surface View;
- void on Picture Taken (byte [] bytes, Camera camera) – метод інтерфейсу Camera. Picture Callback, що викликається при отриманні знімка;
- void on Auto Focus (boolean b, Camera camera) – метод інтерфейсу Camera. Auto Focus Callback, що викликається при спрацьовуванні автофокусу;
- byte [] correctImage (byte [] data) – метод, що перетворює отриманий з камери знімок до формату JPEG;
- Bitmap rotate Image (Bitmap bitmap, int degree) – метод, який здійснює поворот фотографії відповідно до орієнтації дисплея;
- void swap Elements Visibility () – метод, який здійснює зміну елементів управління між режимами зйомки та збереження знімка;
- void rotate Popup (int to Orientation) – метод, який здійснює поворот спливаючого вікна popupWindow;
- void try Start Rotation (View view, int toOrientation) – метод, який здійснює поворот однієї іконки;
- void rotateImages (int to Orientation) – метод, який здійснює поворот всіх іконок;
- Rotate Animation get Animation (int to Orientation) – метод, який повертає об'єкт – анімацію для повороту іконок;
- void update Flash Icon () – метод, який здійснює відображення коректної іконки, що відображає режим спалаху;
- void set Up Orientation Event Listener () – метод, який повертає об'єкт - слухач orientation Observer;
- Popup Window set Up Popup Window () – метод, який створює дочірні елементи управління для спливаючого вікна popup Window;

- `Array List < String > get Human Readable Headers ( )` – метод, що перетворює набір дозволів камери до зрозумілого для людини виду;
- `void set Popup List Content ( List View list, final Array List < String > human Readable Headers )` – метод, яким заповнюють значеннями спливаюче вікно `popup Window`;
- `Popup Window get Popup Window (View view, int width, int height)` – метод, що створює спливаюче вікно `popup Window`.

Клас `Data Activity` асоційований з `Activity` метаданих і служить для відображення і роботи з метаданими фотографії, а також ініціює збереження знімка на сервері і повідомляє користувача про прогрес виконання.

Константи:

- `ACTION_CODE_UPLOAD` – ідентифікатор `Intent`, асоційованого з дією завантаження фото;
- `ACTION_CODE_SAVE` – ідентифікатор `Intent`, асоційованого з дією відправки метаданих та збереження фото;
- `ACTION_CODE_LOCATION` – ідентифікатор `Intent`, асоційованого з дією отримання координат;
- `CITIES_LOADER` – ідентифікатор завантажувача списку міст;
- `STREETS_LOADER` – ідентифікатор завантажувача списку вулиць;
- `STORES_LOADER` – ідентифікатор завантажувача списку магазинів;
- `CATEGORIES_LOADER` – ідентифікатор завантажувача списку категорій;
- `REPORT_DIALOG_TAG` – тег діалогового фрагмента, що використовується для відображення інформації про помилки;
- `PROGRESS_DIALOG_TAG` – тег діалогового фрагмента, що використовується для надання шансу повторного збереження при помилці;
- `TRY_AGAIN_DIALOG_TAG` – тег діалогового фрагмента, що використовується для відображення інформації про помилки;

– SKIP_LOCATION_DIALOG_TAG – тег діалогового фрагмента, що використовується для надання можливості пропустити етап отримання координат.

Поля:

– Loader Manager – поле типу Loader Manager, що містить посилання на менеджер завантажчика;

– View Mapping – поле, що містить масив ідентифікаторів типу int, які задають об'єкти для відображення даних у списку;

– sendItem – поле типу Menu Item, що містить посилання на пункт меню для відправки знімка;

– Send Button – поле типу Button, що містить посилання на кнопку відправки знімка;

– Progress Dialog – поле типу Progress Dialog Fragment, що містить посилання на діалоговий фрагмент, який використовується для відображення прогресу виконання завдання;

– Report Dialog - поле типу Report Dialog Fragment, що містить посилання на діалоговий фрагмент, який використовується для відображення інформації про помилки;

– Try Again Dialog – поле типу Two Option Dialog Fragment, що містить посилання на діалоговий фрагмент, який використовується для надання можливості пропустити етап отримання координат;

– Skip Location Dialog – поле типу Two Option Dialog Fragment, що містить посилання на діалоговий фрагмент, який використовується для надання можливості пропустити етап отримання координат;

– Has Problem Check Box – поле типу Check Box, що містить посилання на фрагмент, що показує наявність проблем;

– Comments Edit Text – поле типу Edit Text, що містить посилання на тестове поле для введення коментаря;

– Last FileId – поле типу String, що містить ідентифікатор останнього завантаженого знімка;

- Successfully Saved – прапор типу boolean, вказує на успішне збереження знімка;
- preferences – поле типу Shared Preferences, що містить посилання на модуль налаштувань.

Методи:

- void try Begin Upload () – метод, який здійснює валідацію призначеного для користувача введення перед операцією авторизації, а також відображає результати валідації;
- void begin Upload () – метод, що запускає процес асинхронного завантаження фото. Як завершальний метод використовується on Activity Result;
- void begin Get Location () – метод, що запускає процес асинхронного отримання координат. Як завершальний метод використовується on Activity Result;
- void begin Save (Location location) – метод, що запускає процес асинхронного збереження фото;
- void end Save () – метод, що викликається при закінченні збереження;
- void setup Spinners () – метод, ініціалізацію випадаючих списків;
- String get Spinner Value (String valueName) – метод, що дозволяє отримати поточне значення зі списку по імені колонки з таблиці БД, яка асоційована з певним списком;
- int getSpinner ValueId (Spinner spinner) – метод, що дозволяє отримати первинний ключ запису БД, значення якої вибрано в випадаючому списку;
- void update Store Adapter () – метод, який здійснює ініціалізацію списку магазинів;
- void update Street Adapter () – метод, який здійснює ініціалізацію списку вулиць;
- void update City Adapter () – метод, який здійснює ініціалізацію списку міст;

- void update Categories Adapter () – метод, який здійснює ініціалізацію списку категорій;
- void restore Spinner (Spinner spinner, Cursor cursor) – метод, розблоковуючий випадок для введення;
- void restore Selection (String field Name) – метод, який відновлює останнє вибране значення для списку;
- Loader <Cursor> on Create Loader (int i, Bundle bundle) – метод інтерфейсу Loader Manager. Loader Callbacks < Cursor >, який здійснює ініціалізацію завантажчика;
- void on Load Finished (Loader < Cursor > cursor Loader, Cursor cursor) – метод інтерфейсу Loader Manager. Loader Callbacks < Cursor >, що викликається при готовності завантажувача віддати даних;
- void on Loader Reset (Loader <Cursor> cursor Loader) – метод інтерфейсу Loader Manager. Loader Callbacks < Cursor >, що викликається при скиданні завантажувача;
- Progress Dialog Fragment get Progress Dialog () – метод, який повертає діалоговий фрагмент, який використовується для відображення прогресу виконання завдання;
- Report Dialog Fragment set Report Dialog (int title Id) – метод, який повертає діалоговий фрагмент, який використовується для відображення інформації про помилки із заданим заголовком;
- Report Dialog Fragment set Report Dialog (int titleId, int messageId) – метод, який повертає діалоговий фрагмент, який використовується для відображення інформації про помилки із заданим заголовком і тілом повідомлення.;
- Two Option Dialog Fragment get Try Again Dialog () – метод, який повертає діалоговий фрагмент, який використовується для надання можливості пропустити етап отримання координат;

– Two Option Dialog Fragment get Skip Location Dialog () – метод, який повертає діалоговий фрагмент, який використовується для надання можливості пропустити етап отримання координат.

Клас WorkerService є спадкоємцем класу Inent Service і представляє собою сервіс фонові обробки операцій з вбудованою чергою обробки повідомлень. Клас містить набір строкових констант виду «ACTION _ XXX », використовуваних зі схожою метою, що і аналогічні цілочисельні константи в класах-Activity. Однак, якщо цілочисельні константи використовуються для ідентифікації необхідного результату, то строкові константи використовуються для ідентифікації викликається методу.

Константи:

– ACTION_LOGIN – ідентифікатор INTENT, що запускає операцію авторизації;

– ACTION_UPLOAD_PICTURE – ідентифікатор INTENT, що запускає операцію завантаження фото;

– ACTION _ SAVE _ PICTURE – ідентифікатор INTENT, що запускає операцію збереження фото;

– PENDING_INTENT_FIELD – ключ за яким доступний очікує Intent для зворотного зв'язку з Activity–ініціатором;

– SERVICE_ERROR – код помилки, що означає помилку сервісу. Може бути переданий в зухвалу Activity в метод on Activity Result як параметр result Code;

– UNKNOWN_HOST – код помилки, що означає неправильну адресу сервера;

– HOST_NOT_SPECIFIED – код помилки, що означає відсутність зв'язку;

– IO_ERROR – код помилки, що означає помилку вводу / виводу;

– SERVICE_UPLOADING – ідентифікатор нотифікації в рядку стану при переході сервісу в привілейований режим.

Поля:

- Pending Intent – поле типу IWeb Client, що містить посилання на яке чекає Intent для зворотного зв'язку з Activity-ініціатором.

- Web Client – поле типу Pending Intent, що містить посилання на веб-клієнт;

- Host Name – поле типу String, що містить ім'я сервера.

Методи:

- void onCreate () – метод викликається після створення сервісу;

- void onHandleIntent (Intent intent) – основний метод сервісу. Витягує черговий Intent з черги повідомлень і передає його на обробку сервісу;

- void login (Intent intent) – метод, який здійснює операцію авторизації, а також, в разі успіху, що запускає оновлення кешу;

- void uploadPicture Перейти до оголошення () – метод, який здійснює завантаження фото;

- void savePicture (Intent intent) – метод, який здійснює збереження фото;

- void sendPending (Intent intent) – метод, який здійснює відправку;

- void reportError (String message) – метод, який здійснює відправку з заданим повідомленням про помилку;

- Notification getUploadNotification () – метод, який створює нотифікацію необхідну для переходу сервісу в привілейований режим.

3.2.3 Опис класів взаємодії з даними

Однією з вимог до реалізованого клієнту була мінімізація використовуваного трафіку. Для виконання цієї вимоги було вирішено реалізувати кешування даних на стороні клієнта. Android SDK надає можливість спрощеного створення і супроводу бази даних, тому досить зручно використовувати прості реляційні структури для збереження інформації на стороні клієнта. У реалізованому дипломному проєкті за кешування даних відповідають класи з пакета data.

Клас Db Helper успадкований від абстрактного класу SQ Lite Open Helper, який надає простий і зручний інтерфейс для створення баз даних в ОС Android.

Константи:

- DB_NAME – константа, яка містить ім'я файлу бази даних;
- STORES_TABLE – константа, яка містить ім'я таблиці магазинів;
- CATEGORIES_TABLE – константа, яка містить ім'я таблиці категорій;
- DB_ID_FIELD – константа, яка містить ім'я службового стовпця з первинними ключами.

Статичні поля:

- instance – поле типу, Db Helper містить Сінглтон-об'єкт даного класу.

Методи:

- Db Helper getInstance (Context context) – статичний метод доступу;
- void onCreate (SQ Lite Database sq Lite Database) – метод базового класу SQLiteOpenHelper, що викликається при створенні бази даних. Об'єкт sq Lite Database є абстракцією БД, і дозволяє виконувати ініціалізацію SQL-запитів;
- void recreateTables () – метод, що дозволяє видалити поточні таблиці і створити нові. Використовується для очищення кеша.

Клас Data Provider є локальною альтернативою повноцінного постачальника даних. Клас надає доступ до записів з таблиць БД у вигляді спеціальних об'єктів типу Cursor, що дозволяють здійснювати ітерацію по деякому набору даних, що є результатом асоційованого з курсором SQL - запиту.

Константи:

- STORES_URI – константа типу Uri, що містить URI таблиці магазинів;
- CATEGORIES_URI – константа типу Uri, що містить URI таблиці категорій.

Поля:

- instance – поле типу, Data Provider містить об'єкт даного класу;

– context – поле типу, Context містить посилання на контекст додатки, необхідні для операцій з БД.

Методи:

– Cursor get Categories () – метод, який повертає об'єкт типу Cursor для ітерації за категоріями;

– Cursor get Streets (String city) – метод, який повертає об'єкт типу Cursor для ітерації по всіх вулицях обраного в параметрі city міста;

– Cursor getStores (String city, String street) – метод, який повертає об'єкт типу Cursor для ітерації по всіх магазинах, обраних в параметрі city і street міста і вулиці;

– void update Stores (List <StoreEntity> new Stores) – метод, який здійснює оновлення інформації в таблиці магазинів даними зі списку newStores;

– void update Categories (List <DbEntity> new Categories) – метод, який здійснює оновлення інформації в таблиці категорій даними зі списку new Categories;

– int get Last Version (String table Name) – метод, що дозволяє отримати останню версію зазначеної в параметрі table Name таблиці;

– int get Last Stores Version () – метод, що дозволяє отримати останню версію таблиці магазинів;

– int get Last Categories Version () – метод, що дозволяє отримати останню версію таблиці категорій.

Вкладений пакет loaders містить в собі реалізації класів-завантажувачів. Найбільш складним елементом пакета є абстрактний клас Simple Cursor Loader, від якого успадковані всі використовувані в проєкті завантажувачі. Він являє собою типову реалізацію класу-завантажувач, призначеного для роботи з локальними об'єктами доступу до даних, на зразок класу Data Provider. Simple Cursor Loader по своїй суті є злегка модифікованою версією класу Cursor Loader, що є частиною Android SDK і орієнтованого на роботу з постачальниками даних. Оскільки завдання реалізації завантажувача, орієнтованого на роботу з локальної заміною постачальників даних, є досить

поширеною, використовувану в даному проєкті реалізацію Simple Cursor Loader можна вважати типовою. Її опис можна знайти, наприклад, в літературному джерелі [8].

Решта класів-завантажувачів є спадкоємцями класу Simple Cursor Loader і скасовують метод Cursor get Cursor (), який повертає курсор з актуальних вмістом, який, в свою чергу, повертає один з методів класу доступу до даних – Data Provider. Крім цього, деякі класи-завантажувачі містять набір методів-установників для завдання додаткових даних асоційованих з курсором SQL-запитів, наприклад, назви міст і / або вулиць.

3.2.4 Опис класів пакета dialogs

Даний пакет містить визначення класів, так чи інакше успадкованих від класу Dialog Fragment і службовців для створення діалогових вікон. Як і у випадку з пакетом loaders, даний пакет містить базовий абстрактний клас Base Dialog Fragment, безпосередньо успадкований від класу Dialog Fragment, в той час як конкретні реалізації діалогових фрагментів успадковані від класу Base Dialog Fragment.

Клас Base Dialog Fragment є базовим для всіх інших класів діалогових фрагментів. Він успадкований від класу Dialog Fragment, залишаючи метод on Create Dialog, безпосередньо створює діалогове вікно, для перевизначення в класах нащадках.

Поля:

– titleId – підлогу типу int, що містить ідентифікатор строкового ресурсу, використовуваного в якості заголовку вікна.

Методи:

– int getTitleId () – метод повертає поточне значення поля titleId;
– void setTitleId (int titleId) – метод дозволяє встановити нове значення для поля titleId;

– void on Destroy View () – метод, визначений у базовому класі DialogFragment. Тут переопределяється для обходу помилки в Android SDK;

– Dialog on Create Dialog (Bundle savedInstanceState) – метод, визначений у базовому класі Dialog Fragment і є в даному класі абстрактним, проте перевизначених у всіх класах-нащадках. Використовується для створення діалогового вікна, представленого спадкоємцем класу Dialog.

Клас Progress Dialog Fragment є діалоговий фрагмент, службовець для відображення нескінченного прогресу з заданим заголовком і текстом повідомлення.

Поля:

– messageId – підлогу типу int, що містить ідентифікатор строкового ресурсу, використовуваного в якості тексту повідомлення.

Методи:

– int get MessageId () – метод повертає поточне значення поля messageId;

– void set MessageId (int messageId) – метод дозволяє встановити нове значення для поля messageId.

Клас Redirect To Settings Dialog Fragment найбільш простий клас діалогового фрагмента. Схожий з класом Report Dialog Fragment, проте використовує строго певні тексти заголовка і тіла повідомлення, а також не може бути закритий за допомогою кнопки «Назад».

Клас Report Dialog Fragment є діалогове вікно повідомлення про помилку з довільним текстом заголовка і тіла повідомлення, а також єдиною кнопкою «ОК». Важливим моментом є те, що Activity, що використовує даний фрагмент повинна реалізовувати інтерфейс Dialog Interface. On Click Listener для обробки натискань по єдиною кнопці діалогу.

Поля:

– messageId – підлогу типу int, що містить ідентифікатор строкового ресурсу, використовуваного в якості тексту повідомлення.

Методи:

– int get MessageId () – метод повертає поточне значення поля messageId;

– void set M essageId (int messageId) – метод дозволяє встановити нове значення для поля messageId;

– void on Attach (Activity activity) – метод, певний в базовому класі Dialog Fragment. Викликається в момент асоціації фрагмента з певною Activity, але до створення візуальних елементів. Використовується для перевірки реалізації Activity інтерфейсу Dialog Interface. On Click Listener.

Класу Report Dialog Fragment є діалогове вікно повідомлення про помилку з незмінним текстом заголовка і тіла повідомлення, а також двома кнопками, що містять довільний текст. Всі параметри задаються в конструкторі при створенні фрагмента. Activity, що використовує даний фрагмент, також, як і в випадку з класом Report Dialog Fragment, повинна реалізовувати інтерфейс Dialog Interface. On Click Listener для обробки натискань діалогу.

Поля:

– message Id – підлогу типу int, що містить ідентифікатор строкового ресурсу, використовуваного в якості тексту повідомлення;

– Positive Text Id – підлогу типу int, що містить ідентифікатор строкового ресурсу, використовуваного в якості тексту на позитивній кнопці діалогу;

– Negative Text Id – підлогу типу int, що містить ідентифікатор строкового ресурсу, використовуваного в якості тексту на негативній кнопці діалогу;

Методи : void on Attach (Activity activity) - метод, певний в базовому класі Dialog Fragment. Описано в класі Report Dialog Fragment і службовець для аналогічної мети.

3.2.5 Огляд функціональних класів з пакета helpers

Даний пакет містить допоміжний класи-утиліти, що є наборами статичних функцій і використовуваних в різних частинах програми.

Клас `Accessibility Helper` надає набір методів для перевірки доступності мережі Інтернет і статусу GPS на телефоні, а також надає методи для створення діалогових вікон, що перенаправляє на відповідні сторінки системних налаштувань для включення необхідних функцій.

Методи:

- `boolean is Networking Enabled ( Context context )` – метод, який показує наявність інтернет-з'єднання;
- `boolean is Gps Enabled ( Context context )` – метод, який показує доступність провайдерів GPS –координат;
- `Alert Dialog get Dialog ( final Context context, int message Id, final String action )` – допоміжний метод для створення діалогу з довільним сполученням, перенаправляє на певну сторінку системних налаштувань;
- `Alert Dialog get Gps Dialog ( final Context context )` – метод, який повертає діалог, що перенаправляє на сторінку налаштувань інтернет-з'єднання;
- `Alert Dialog get Network Dialog ( final Context context )` – метод, який повертає діалог, що перенаправляє на сторінку налаштувань GPS.

Клас `Authorization Helper` надає функціонал для авторизації на клієнтській стороні, тобто забезпечення доступу згідно ролі користувача.

Константи:

- `USER _ ROLE` – строкова константа, яка містить значення ролі «користувач»;
- `MERCHANDISER _ ROLE` – строкова константа, яка містить значення ролі «мерчендайзер»;
- `AUTHORIZATION _ ERROR` – строкова константа-ідентифікатор, що вказує на помилку авторизації.

Методи:

- `boolean isValidRole ( String role )` – метод, який показує наявність доступу до додатком користувачеві з вказаною в параметрі `role` роллю.

Клас `Image Store` використовують для передачі зображення в бінарному вигляді між різними `Activity`.

Статичні поля:

- `image` – фото в вигляді масиву байт.

Статичні методи:

- `byte [ ] get Image ()` – повертає зображення, збережене в поле `image`;
- `void store Image ( byte [ ] image )` – зберігає зображення, в поле `image`;
- `void clean ()` – очищає поле `image`.

Клас `Json Helper` призначений для спрощеного створення JSON об'єктів і використовується класами взаємодії з мережею. На відміну від інших класів-помічників, даний клас має набір не тільки статичних методів. Клас має набір методів `add` з різною сигнатурою, які є повними аналогами методів `put` класу `JSON Object`, проте маскують неактуальне в даному проєкті винятки `JSON Exception`.

Поля:

- `json` – поле типу `JSONObject`, що представляє собою редагований JSON-об'єкт.

Методи:

- `Json Reader get Reader For Response ( Http Response response )` – статичний метод, що дозволяє отримати потоковий JSON парсер типу `Json Reader`, асоційований з мережевим потоком з HTTP-відповіді, переданого в якості параметра `response`;
- `String Entity to String Entity ( )` – метод, що перетворює отримати редагований JSON-об'єкт, що зберігається в поле `json` до об'єкту типу `String Entity`, який можна використовувати в якості тіла HTTP-запиту.

Клас `Location Helper` використовується для отримання даних про поточне місцезнаходження, а також, при тестуванні, для емуляції пов'язаних з цим помилок.

Константи:

- `LATITUDE _ FIELD` – константа-ключ, за яким доступна широта в `INTENT` з геоданих;

- `LONGITUDE _ FIELD` – константа-ключ, за яким доступна довгота в `INTENT` з геоданих;
- `ACCURACY _ FIELD` – константа-ключ, за яким доступна точність в `INTENT` з геоданих;
- `TIMEOUT` – таймаут в мілісекундах, після якого видається помилка про неможливість отримання координат.

Поля:

- `Error Reported` – прапор типу `boolean`, що запобігає повторну відправку `Intent` з повідомленням про помилку.

Методи:

- `void report Error ( Context context, Pending Intent pending Intent )` – метод здійснює відправку `INTENT` з повідомленням про помилку;
- `void get Location ( final Context context, final Pending Intent pending Intent )` – метод, що дозволяє компоненту `context` асинхронно отримати час за допомогою відкладеного `INTENT pending Intent`.

3.2.6 Опис класів пакета `networking`

Даний пакет містить елементи, необхідні для взаємодії з мережею. Всі класи пакета можна розділити на дві групи: класи, що представляють собою структури даних і веб-клієнти.

Класи, що представляють собою структури даних реалізують ряд властивостей у вигляді набору `get-` та `set-` методів для отримання полів класу і установки їх нових значень. Важливою особливістю всіх таких класів, є реалізація інтерфейсу `Parcelable`, необхідного для пересилання примірників даних класів разом з `Intent`.

Класи веб-клієнти реалізують інтерфейс `IWeb Client`, що описують програмні еквіваленти веб-методів серверного API. При цьому існує два класи веб-клієнта: робочий `Web Client` і тестовий `Fake Web Client`, який використовується для емуляції роботи сервера.

Клас `Web Client` – безпосередньо клієнт, який реалізує методи веб-інтерфейсу, що надається сервером і здійснює обмін даних по мережі Інтернет. Клас оголошує набір констант, названих за шаблоном `API_ *`, які представляють собою назви веб-методів.

Поля:

- `client` – поле типу `Default http Client`, що представляє собою об'єкт, який займається комунікацією по протоколу HTTP;
- `Auth Header` – строкове поле з заголовком авторизації;
- `Base Url` – строкове поле з адресою сервера.

Методи:

- `Http Response post ( String method, Http Entity entity )` – метод, що відправляє HTTP POST запит за адресою веб-методу з параметра `method` з тілом, заданим параметром `entity`;
- `String login ( String login, String password )` – метод, який здійснює авторизацію з зазначеними логіном і паролем і повертає роль користувача;
- `List < Store Entity > update Stores ( int version )` – метод, що дозволяє отримати оновлення таблиці магазинів вище версії переданої в параметрі `version` у вигляді списку структур даних `Store Entity`;
- `List < Db Entity > update Categories ( int version )` – метод, що дозволяє отримати оновлення таблиці категорій вище версії переданої в параметрі `version` у вигляді списку структур даних `DbEntity`;
- `List < Store Entity > parse Stores List ( Json Reader reader )` – метод, що дозволяє вважати потоковим парсером `reader` масив елементів `Store Entity`;
- `List < DbEntity > parse Categories List ( Json Reader reader )` – метод, що дозволяє вважати потоковим парсером `reader` масив елементів `Db Entity`;
- `String parse Common Parameter ( Db Entity entity, Json Reader reader )` – метод, що дозволяє отримати з потоку з даними в форматі JSON, за допомогою парсера `reader`, одне з полів об'єкта `entity`;

– `String upload Picture` Перейти до оголошення (`byte [ ] picture`) – метод завантажує фото в бінарному вигляді на сервер, повертаючи певний сервером GUID;

– `boolean send Metadata ( Metadata data )` – метод, що дозволяє зберегти фото з зазначеними метаданими. Що повертається методом значення, показує успішність виконання операції.

3.3 Висновки до розділу

В даному розділі було побудовано функціональну схему програми, яка складається з чотирьох основних модулів: модуля авторизації, налаштування, роботи з камерою та редагування. В якості способу комунікації використовується підхід, заснований на Intent-аналізі. Подібний підхід дозволяє більш просто організувати доступ до сервісу з декількох модулів програми.

Також були визначені основні методи для організації взаємозв'язку з сервером, класи системних компонент, класи взаємодії з даними, функціональні класи.

4. КЕРІВНИЦТВО ПРОГРАМІСТА

Однією з найважливіших частин програми є клас Web Client, який відповідає за комунікацію з сервером. Даний клас реалізує інтерфейс IWeb Client, що описує серверне API у вигляді набору Java-методів. Особливістю цих методів є взаємодія з мережею, тому вони все вони можуть викинути обов'язкове для обробки виключення IO Exception, а також не можуть бути викликані в основному потоці, оскільки це викличе виключення Network On Main Thread Exception. Виходячи з цього, методи, певні в IWeb Client, повинні викликатися в контексті, що забезпечує виконання в відмінному від основного потоці і обробному виключення введення / виведення. Таким контекстом є метод on Handle Intent класу Worker Service:

```
@Override
protected void on Handle Intent ( Intent intent ) {
    try {( hostname! = null &&! hostname.isEmpty () ) {action =
intent.getAction (); (intent.hasExtra (PENDING_INTENT_FIELD)) {=
intent.getParcelableExtra (_INTENT_FIELD);
    } ( ACTION_LOGIN.equals (action)) {(intent);
    } Else if (ACTION_UPLOAD_ PICTURE. ( Action)) {();
    } Else if (ACTION_SAVE_ PICTURE. ( Action)) {(intent);}
    } Else {(HOST_NOT_SPECIFIED); }
    } Catch (UnknownHostException ex) {(UNKNOWN_HOST);
    } Catch (IOException ex) {.printStackTrace (); (IO_ERROR); }}
```

Метод on Handle Intent визначений в базовому класі Intent Service і працює наступним чином: коли який-небудь компонент посилає сервісу запит за допомогою INTENT, переданого в метод start Service, даний Intent обертається в об'єкт Message і додається в чергу обробки повідомлень, асоційовану з екземпляром класу Handler. Кожна чергу також асоційована з певним потоком, в якому буде проводитися обробка повідомлень. Якщо об'єкт

Handler визначає, що потік обробки простоює, то він витягує чергове повідомлення (об'єкт класу Message) з черги і передає його в метод handle Message. Цей метод класу Handler не виробляє ніякої обробки повідомлення, тому для створення працездатної черзі обробки використовуються спадкоємці класу Handler. Клас Intent Service містить посилання на нащадок класу Handler, який в методі handle Message витягує Intent з об'єкта Message і передає в метод on Handle Intent. Сам Intent Service, аналогічно класу Handler, не забезпечує ніякої обробки переданих в on Handle Intent Intent, для чого також використовуються класи-спадкоємці, яким і є клас Worker Service. Таким чином, метод on Handle Intent гарантовано викликається в відмінному від головного потоці, а блок catch з типом IO Exception забезпечує обробку помилок введення / виведення.

У методі on Handle Intent немає прямих викликів методів інтерфейсу IWeb Client, проте вони є всередині методів log In, upload Picture і save Picture. Лістинг цих методів наведено нижче разом з лістингом методу report Error, що забезпечує інформування про помилки, а також аналогічного йому методу send Pending, інформує про успішне закінчення операції.

```
private void reportError ( String message) {{.send (this, SERVICE_ERROR,
Intent (). (CommonConstants.ERROR_FIELD, message));
} Catch (PendingIntent.CanceledException e) {.printStackTrace ();
} void sendPending (Intent intent) {{.send (this, Activity.RESULT_OK,
intent);
} Catch (PendingIntent.CanceledException e) {.printStackTrace (); }
```

Неважко помітити, що метод send Pending відрізняється від report Error тим, що використовує код Activity. RESULT _ OK і не передає ніякої додаткової інформації про помилки.

```
private void logIn ( Intent intent) throws IOException {login =
intent.getStringExtra (.LOGIN_FIELD); password = intent.getStringExtra
(.PASSWORD_FIELD); role = webClient.logIn (login, password); (new Intent ().
```

```
(CommonConstants. ROLE_FIELD, role)); (AuthorizationHelper.isValidRole (role))
{provider =
    DataProvider.getInstance (this ) ;. updateStores (webClient.updateStores
(.getLastStoresVersion ())) ;. updateCategories (webClient. ( provider.
getLastCategoriesVersion ( ))) ; }
```

Даний метод інкапсулює витяг з `INTENT` необхідних параметрів, а також здійснює оновлення кешу при успішній авторизації. Тривалі і ресурсомісткі операції з базою даних не будуть займати основний потік виконання. По-друге, застосунок не повинен допускати користувача до редагування метаданих до тих пір, поки кеш знаходиться в неактуальному стані. Таким чином, операції завантаження і збереження фотографії гарантовано не будуть витягнуті з черги обробки повідомлень до тих пір, поки не закінчиться оновлення БД.

Метод `upload Picture` Перейти до оголошення, що здійснює завантаження фото на сервер, показує відразу два характерних для Android - додатків рішення. По-перше, оскільки операція завантаження знімка займає дуже багато часу і при цьому не повинна перериватися, для її виконання сервіс входить в привілейований режим за допомогою виклику `start Foreground`. Це означає, що користувач може спокійно згорнути програму чекаючи закінчення завантаження, не побоюючись того, що ОС завершить додаток з метою економії ресурсів. По завершенню завантаження, сервіс виходить з привілейованого режиму використовуючи метод `stop Foreground`. В якості одного з параметрів метод `start Foreground` приймає спеціальний об'єкт-нотифікацію, що представляє собою повідомлення в рядку стану телефону, яке вказує на те, що в даний момент відбувається деяка тривала високопріоритетних операція. Метод `stop Foreground`, в свою чергу, приймає як параметр прапор, який вказує на те, чи слід автоматично прибрати нотифікацію, створену при вході в привілейований режим або ж користувач повинен буде зробити це сам.

По-друге, в метод `upload Picture` можна бачити, що зображення для завантаження передається не в якості параметра `INTENT`, як це зроблено з логіном і паролем в методі `login`, а через статичне поле в класі `Image Store`. Це зроблено для того, щоб обійти обмеження, що накладаються `INTENT` на розмір своїх параметрів. Зображення часто перевищують певний для параметрів ліміт байт, що призводить до виникнення виключень під час виконання програми.

Метод `save Picture`, аналогічно методу `login`, інкапсулює вилучення даних з `INTENT`, однак якщо логін і пароль передавалися як прості строкові параметри, то тут використовується складна структура `Metadata`. Для забезпечення пересилання складних структур даних можна скористатися наданою Java бінарної серіалізацією об'єктів, проте це не є ефективним способом Конверсія даних. Замість цього, Android SDK надає куди більш гнучкий і швидкий спосіб заснований на інтерфейсі `Parcelable`. Кожен `Parcelable`-об'єкт повинен реалізовувати метод `void write To Parcel (Parcel parcel, int i)`, який викликається кожен раз при перетворенні даних для пересилання. Об'єкт `parcel` є чергу, здатну зберігати рядки, примітивні типи і їх масиви. Крім цього, реалізують `Parcelable` класи повинні містити статичне поле `CREATOR` з об'єктом класу спадкоємця типу `Parcelable`. `Creator <T>`, де `T` - тип, який містить поле `CREATOR`. Інтерфейс `Parcelable`. `Creator <T>` визначає метод `T create From Parcel (Parcel in)`, який викликається на приймаючій стороні. Цей метод витягує з черги `in` дані і створює з них копію пересилається `Parcelable`-об'єкта. Приклад реалізації інтерфейсу `Parcelable` представлений нижче за допомогою фрагментів класу `Metadata`:

```
public class Metadata implements Parcelable {
    <Оголошення властивостей і констант>

    @Override void writeToParcel ( Parcel parcel, int i) { . writeInt ( storeId )

    <Запис інших властивостей в чергу parcel >

    parcel.writeString (wasTaken);
```

```

    } static final Parcelable.Creator <Metadata> CREATOR = Parcelable.Creator
    <Metadata> () {Metadata createFromParcel (Parcel in) {metadata = new Metadata ()
    ;. storeId = in.readInt ();

```

<Витяг інших властивостей з черги in>

```
metadata.wasTaken = in.readString (); metadata;
```

```
} Metadata [] newArray (int size) { new Metadata [ size ]; }
```

В даному випадку, об'єкт CREATOR представлений анонімні класом, що спрощує ініціалізацію створюваного ним об'єкта класу Metadata.

Метод log In добре показує особливості реалізації веб-клієнта в даному проєкті. У ньому можна виділити кілька основних елементів, також присутніх у всіх інших методах веб-клієнта:

- виклик методу post. Даний метод формує HTTP POST запит з тілом, переданим в другому параметрі, і відправляє його за адресою веб-методу, зазначеного першим параметром. Лістинг цього методу буде приведений далі по тексту;

- використання об'єкта Json Helper для формування тіла запиту. Решта методів спочатку використовую кілька викликів методу Json Helper.add для ініціалізації об'єкта, а потім приводять його до необхідного типу викликом методу to String Entity, також наведеного нижче. Винятком є метод upload Picture, відразу використовує об'єкт Byte Array Entity, оскільки тілом запиту для цього методу не є JSON-об'єкт;

- отримання з HTTP-відповіді потокового JSON-парсера, за допомогою статичного методу Json Helper. Get Reader For Response;

- послідовне вилучення даних з HTTP-потoku за допомогою раніше отриманого JSON-парсера.

У методі здійснюється об'єднання базового адреси веб API з ім'ям конкретного веб-методу, створення нового екземпляра HTTP POST запиту, установка його тіла, а також HTTP -заголовков, що містять тип вмісту, тип відповіді і токен авторизації. Як повертається виступає об'єкт типу Http

Response, що представляє собою HTTP-відповідь на створений HTTP POST запит.

Логіка методів `toString` і `getReaderForResponse` вельми прямолінійна, проте добре ілюструє типові способи конвертації даних при взаємодії з REST API в Android-застосунках.

Система тестування Android надає архітектуру і потужні інструменти для тестування всіх частин програми на всіх рівнях розробки (від окремої частини програми до додатка в цілому). Система тестування включає в себе такі особливості:

- набори тестів Android основаних на JUnit. Для тестування класу, який не використовує Android API, можна використовувати звичайні тести з JUnit, або використовувати розширення Android JUnit, для тестування Android компонентів;
- Android розширення JUnit надають компоненти-орієнтовані класи тестів. Ці класи надають допоміжні методи для створення mock-об'єктів і методів, які допомагають керувати життєвим циклом компонентів;
- набори тестів містяться в тестових пакетах, які схожі на основні пакети програм;
- інструменти SDK для створення і тестування доступні у вигляді плагінів до IDE, а також у вигляді консольних додатків. Ці інструменти отримують інформацію з проєкту тестованої програми і використовують її для автоматичного створення build-файлів, файлу маніфесту і структури каталогів для тестового пакета;
- SDK також надає `monkeyrunner` – API тестування пристроїв з програмами на Python, і `UI / Application Exerciser Monkey` (консольний інструмент для стрес-тестування для користувача інтерфейсів шляхом відправки псевдовипадкових подій на пристрій).

У процесі роботи над проєктом компанія-замовник вирішила відмовитися від повноцінного тестування на догоду прискоренню процесу розробки, тому тестування даного проєкту є виключно функціональним.

В основному тестування проводилося на смартфоні Samsung Galaxy A30 з операційною системою Android і розрішенням екрану в 800x480 пікселів. Смартфон має всі особливостями, необхідними для повноцінної роботи всіх можливостей програми, тому додатково проводилися тести на інших пристроях. У деяких тестах використовувався смартфон Huawei U 8230 зі встановленою на нього модифікованою версією ОС Android - CyanogenMod 9, проте належної варіативності це не принесло, хоча і дозволило перевірити відмінне від 800x480 розрішення на реальному пристрої.

Для того, щоб перевірити роботу програми в різних умовах, використовувався вбудований в SDK емулятор ОС Android. Емулятор дозволяє створювати віртуальні пристрої з практично будь-якими конфігураціями обладнання, однак не може емулювати деякі функції камери, наприклад, такі як спалах і автофокус. Виходячи з цього, емулятор використовувався в першу чергу для тестування користувацького інтерфейсу на різних розрішеннях екрану. Для цього був створений один екземпляр встановленою віртуальної машини «Nexus 4», що володіє характеристиками однойменного смартфона, для якої, надалі, змінювалося дозволу екрану, не зачіпаючи інших технічних параметрів. Оскільки емулятор, що входить в Android SDK має дуже поганим швидкодією, побічним позитивним ефектом такого тестування з'явилася перевірка на працездатність програми на «слабких» пристроях, що не володіють достатньо потужними апаратними характеристиками.

5. КЕРІВНИЦТВО КОРИСТУВАЧА

Застосунок не вимогливий до ресурсів і здатний працювати на всіх сучасних моделях телефонів. Єдиною обов'язковою вимогою є наявність задньої камери. Крім цього, для того, щоб скористатися всіма функціями програми, потрібна наявність GPS, а також камери з автофокусом і спалахом. Мінімальна версія ОС Android - 4.0, « Ice Cream Sandwich ».

При першому запуску програма попросить вказати адресу сервера для зберігання фотографій (рис 5.1).

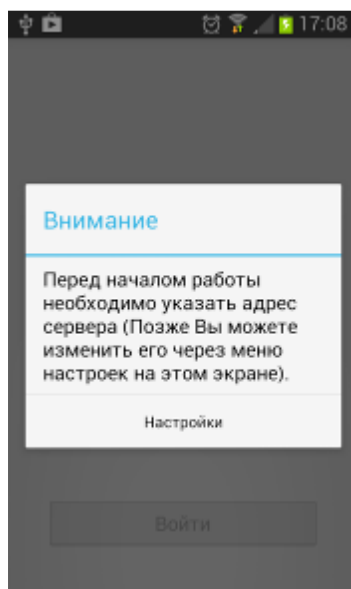


Рисунок 5.1 – Перший запуск програми - спливаюче вікно

Крім адреси сервера, сторінка налаштувань дозволяє скасувати отримання координат при збереженні фотографії для прискорення завантаження.

Після початкового запуску, сторінка налаштувань доступна через контекстне меню на сторінці авторизації.

Після ініціалізації налаштувань, а також при всіх наступних запусках, користувач потрапляє на сторінку авторизації (рис. 5.3), яка містить поля для

введення логіна і пароля. Крім цього, сторінка містить прапорець, що дозволяє запам'ятати дані, щоб не вводити їх в наступний раз.

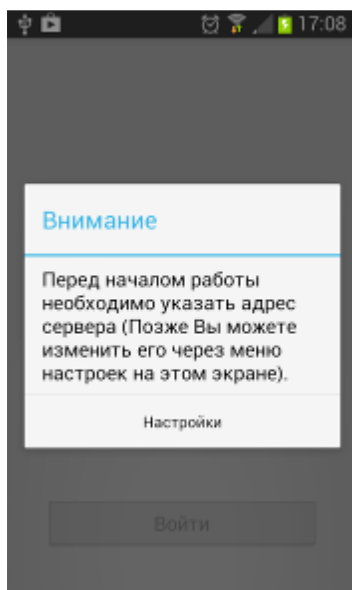


Рисунок 5.2 – Сторінка налаштувань

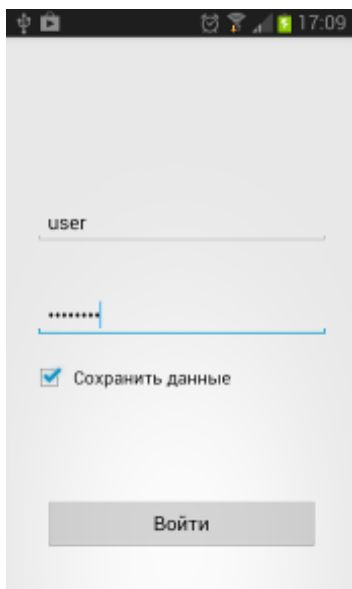


Рисунок 5.3 – Сторінка авторизації

При успішній авторизації, користувач потрапляє на сторінку взаємодії з камерою (рис. 5.4). За допомогою кнопок у верхньому правому куті, користувач

може змінити здатність знімка, а також встановити режим спалаху. Кнопка в верхньому лівому кутку дозволяє вийти з облікового запису, а кнопка в нижній частині екрана - зробити знімок. Після отримання фото, в нижній частині з'являється панель з кнопками «Скасування» і «Далі», яка дозволяє прийняти знімок або зробити повторний. Після отримання задовільного зображення, слід натиснути на кнопку «Далі», яка відкриє сторінку збереження фото.

У верхній частині екрану розташовані кілька випадаючих списків для фільтрації списку магазинів. Перший список необхідний для вибору міста, після чого наступного за ним список вулиць заповнюється відповідними значеннями. Після вибору вулиці значеннями заповнюється третій за рахунком, що випадає з магазинами. Далі слід вибрати категорію товару з четвертого зверху списку, а потім заповнити опціональні поля: встановити час знімка (до роботи, після роботи або прочерк), прапорець наявності проблеми і дати довірливий коментар.



Рисунок 5.4 – Загальний вигляд камери

Після заповнення всіх полів необхідно натиснути кнопку «Відправити» внизу сторінки або ж скористатися її аналогом в правому верхньому куті екрану. Подальший процес відправки фото представлений на рис. 5.7 - 5.10.

При успішному завершенні завантаження, з'явиться спливаюче вікно з повідомлення про закінчення і кнопкою «ОК», натискання на яку повертає на сторінку камери.

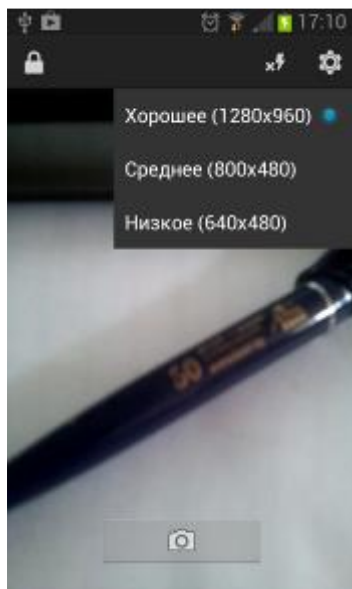


Рисунок 5.5 – Меню вибору розрішення



Рисунок 5.6 – Підтвердження збереження знімка

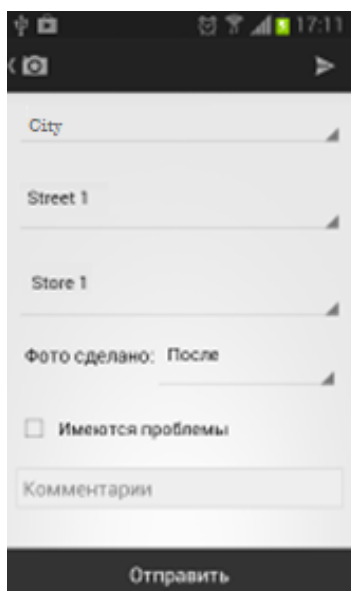


Рисунок 5.7 – Сторінка метаданих

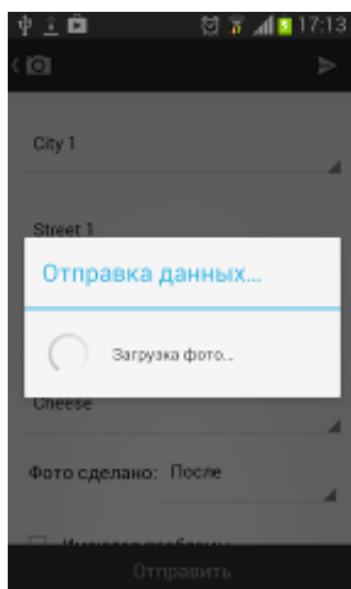


Рисунок 5.8 – Завантаження фото

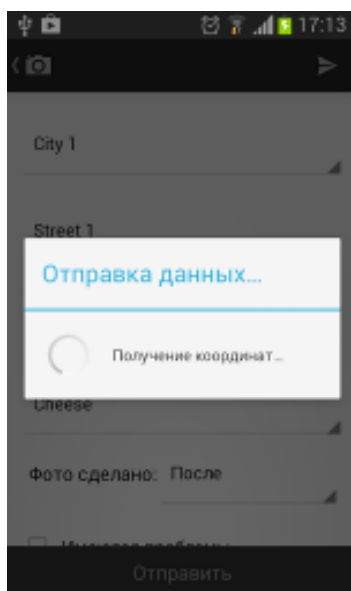


Рисунок 5.9 – Отримання координат

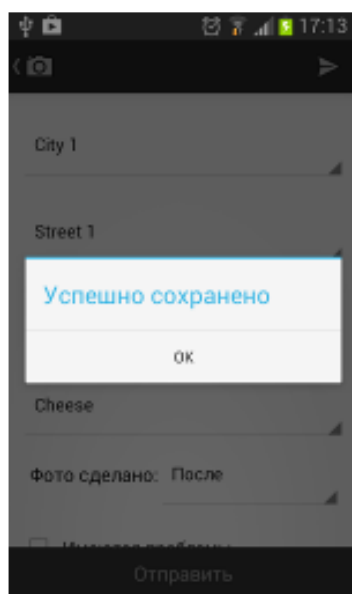


Рисунок 5.10 – Завершення збереження

ВИСНОВКИ

В останні десятиліття маркетинг став однією з найважливіших областей в індустрії торгівлі. При приблизно рівній якості товарів, великі компанії прагнуть конкурувати з допомогою реклами та яскравого, примітного виду і розташування товарів. На відміну від якісних показників, ефективність тих чи інших маркетингових дій не може бути заміряна в лабораторії, проте цілком можна зіставити інтерес споживачів і, як наслідок, рівень прибутку з певною маркетинговою активністю.

Великі компанії зацікавлені в рішеннях, що дозволяють знизити трудомісткість операцій по контролю за проведенням маркетингових акцій, в чому їм може допомогти розроблений застосунок. Очевидно, що перехід на швидку, автоматизовану звітність, поєднану з основними обов'язками мерчендайзерів, знизить час і витрати, що відводяться на контроль за виконанням роботи співробітниками компанії.

Розроблений проєкт має лише базовим функціоналом, але при цьому є цілком достатнім засобом для ведення звітності.

В результаті виконання роботи виконано наступні завдання:

- проведено аналіз чинників та критеріїв оцінки персоналу;
- проведено аналіз основних принципів і методів оцінки роботи персоналу;
- проведено ровести аналіз середовища розробки;
- проведено системне та функціональне проєктування;
- розроблено програму для моніторингу та аналізу роботи співробітників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Что такое мониторинг результативности и зачем он нужен [Электрон. ресурс]. – 2013 – Режим доступа: https://urbaneconomics.ru/sites/default/files/4013_import.pdf
2. Управление персоналом на предприятии в Украине [Электрон. ресурс]. – 2020 – Режим доступа: https://www.prostobiz.ua/biznes/upravlenie_biznesom/stati/upravlenie_personalom
3. Критерии оценки персонала [Электрон. ресурс]. – 2019 – Режим доступа: <https://bbooster.online/stati/kriterii-otsenki-personala.html>
4. Repsly [Электрон. ресурс]. – 2021 – Режим доступа: <https://gde-saas.ru/applications/repsly/>
5. CRM-система для ведения клиентской базы [Электрон. ресурс]. – 2021 – Режим доступа: <https://gde-saas.ru/applications/amocrm/>
6. Сервис автоматизации мерчендайзинга. [Электрон. ресурс]. – 2021 – Режим доступа: <https://gde-saas.ru/applications/easymerch/>
7. Методы оценки персонала [Электрон. ресурс]. – 2021 – Режим доступа: <https://hrhelpline.ru/metody-i-metodiki-ocenki-personala/>
8. Колисниченко Д. «Программирование для Android» / Д. Колисниченко // СПб. БХВ-Петербург – 2012- С272.
9. Федотенко М. А. Разработка мобильных приложений. Первые шаги / М. А. Федотенко // Лаборатория знаний, - 2019 - С.372
10. Android Programming: The Big Nerd Ranch Guide — Bill Phillips, Chris Stewart & Kristin Marsicano [Электрон. ресурс]. – 2018 – Режим доступа: <https://otus.ru/nest/post/942/>
11. The Busy Coder's Guide to Android Development — Mark L. Murphy [Электрон. ресурс]. – 2019 – Режим доступа: <https://otus.ru/nest/post/547/>
12. Herbert Schildt Java: A Beginner's Guide [Электрон. ресурс]. – 2019 – Режим доступа: https://skillbox.ru/media/code/razrabotka_na_androd_i_ios/

ДОДАТОК А
Текст програми

Вихідний текст класу LoginActivity

```

public class LoginActivityActivityView.OnClickListener,
TextWatcher,.OnEditorActionListener,.OnClickListener {static final int ACTION_CODE_LOGIN = 100;static final
String SAVE_CREDENTIALS_PROPERTY = "skip_login";static final String PROGRESS_DIALOG_TAG =
"progress";static final String REPORT_DIALOG_TAG = "report";static final String REDIRECT_DIALOG_TAG =
"redirect";EditText loginEditText = null;EditText passwordEditText = null;CheckBox saveCredentialsCheckBox =
null;Button signinButton = null;boolean saveCredetials = false;ProgressDialogFragment
progressDialog;SharedPreferences preferences;

@Overridevoid onCreate(Bundle savedInstanceState)
{.onCreate(savedInstanceState);(R.layout.activity_login);=
(ProgressDialogFragment)getFragmentManager().(PROGRESS_DIALOG_TAG);=
(EditText)findViewById(R.id.loginEditText);= (EditText)findViewById(R.id.passwordEditText);=
(CheckBox)findViewById(R.id.saveCredentialsCheckBox);=
(Button)findViewById(R.id.signinButton);.addTextChangedListener(this);.addTextChangedListener(this);.setOnEditor
ActionListener(this);.setOnClickListener(this);= PreferenceManager.getDefaultSharedPreferences(this);();
}

@Overrideboolean onCreateOptionsMenu(Menu menu) {(1, 1, 1,
R.string.action_settings).(R.drawable.ic_action_settings).(new Intent(this,
SettingsActivity.class));super.onCreateOptionsMenu(menu);
}

@Overridevoid onClick(View view) {(view.getId()) {R.id.signinButton :();
}void tryBeginAuthorizeUser() {(AccessibilityHelper.isNetworkingEnabled(this))
{(.getText().toString(),.getText().toString());
} else {.getNetworkDialog(this).show();
}
}

@Overridevoid onPause() {();.onPause();
}

@Overridevoid onActivityResult(int requestCode, int resultCode, Intent data) {(resultCode == RESULT_OK)
{(requestCode) {ACTION_CODE_LOGIN :(data.getStringExtra(CommonConstants.ROLE_FIELD));
}
} else {.dismiss();error =
data.getStringExtra(CommonConstants.ERROR_FIELD);(WorkerService.HOST_NOT_SPECIFIED.equals(error)){();.
setEnabled(true);
} else {(WorkerService.UNKNOWN_HOST.equals(error) ?.string.err_unknown_host : R.string.err_unknown);
}void navigateToCameraActivity() {intent = new Intent(this,
CameraActivity.class);.setFlags(FLAG_ACTIVITY_CLEAR_TOP|.FLAG_ACTIVITY_CLEAR_TASK|.FLAG_ACT
IVITY_NEW_TASK);.dismiss();(intent);
}

//region TextWatcher implementation
@Overridevoid beforeTextChanged(CharSequence charSequence, int i, int i2, int i3) {
}

```

```

@Override void onTextChanged(CharSequence charSequence, int i, int i2, int i3) {
}

@Override void afterTextChanged(Editable editable) {setEnabled(
!TextUtils.isEmpty(loginEditText.getText()) &&
!TextUtils.isEmpty(passwordEditText.getText()));
}

@Override boolean onEditorAction(TextView textView, int i, KeyEvent keyEvent) {(i ==
EditorInfo.IME_ACTION_DONE && isEnabled()) {inputManager =
(InputMethodManager)this.getSystemService(INPUT_METHOD_SERVICE);hideSoftInputFromWindow(.ge
tWindowToken(),HIDE_NOT_ALWAYS);();
}true;
}

//endregion void beginAuthorizeUser(String login, String password) {setEnabled(false);= new
ProgressDialogFragment(.string.login_progress_title, R.string.login_progress_text);.show(getFragmentManager(),
PROGRESS_DIALOG_TAG);pending = createPendingResult(_CODE_LOGIN,Intent(),FLAG_ONE_SHOT);intent =
new Intent(this, WorkerService.class).(CommonConstants.LOGIN_FIELD,
login).(CommonConstants.PASSWORD_FIELD, password).(WorkerService.PENDING_INTENT_FIELD,
pending);.setAction(WorkerService.ACTION_LOGIN);(intent);
}void endAuthorizeUser(String role) {(AuthorizationHelper.isValidRole(role)) {();
} else {(AuthorizationHelper.AUTHORIZATION_ERROR.equals(role) ?.string.err_invalid_credentials :
R.string.err_unknown);
}
}void reportError(int messageId) {reportDialog = new ReportDialogFragment(R.string.error,
messageId);.show(getFragmentManager(), REPORT_DIALOG_TAG);.dismiss();setEnabled(true);
}void showRedirectDialog() {RedirectToSettingsDialogFragment()
.show(getFragmentManager(), REDIRECT_DIALOG_TAG);
}
@Override void onClick(DialogInterface dialogInterface, int i) {.dismiss();
}

//region Work with preferences void restoreProperties() {=
preferences.getBoolean(SAVE_CREDENTIALS_PROPERTY, false);.setChecked(saveCredetials);(saveCredetials)
{login = preferences.getString(CommonConstants.LOGIN_FIELD, "");password =
preferences.getString(CommonConstants.PASSWORD_FIELD, "");.setText(login);.setText(password);
}hostName = PreferenceManager.(this).(CommonConstants.HOST_NAME_PROPERTY, null);(hostName ==
null || hostName.isEmpty()) {();
}void savePreferences() {= saveCredentialsCheckBox.isEnabled();Editor editor =
preferences.edit();.putBoolean(SAVE_CREDENTIALS_PROPERTY, saveCredetials);(saveCredetials)
{.putString(CommonConstants.LOGIN_FIELD,
loginEditText.getText().toString());.putString(CommonConstants.PASSWORD_FIELD,
passwordEditText.getText().toString());
}
editor.commit();
}

```

```
}
```

Вихідний текст класу CameraActivityclass

```
CameraActivityActivitySurfaceHolder.Callback,.PictureCallback,.OnClickListener,.AutoFocusCallback,.OnItemClickListener{static String PROPERTY_SIZE_INDEX = "PictureSizeIndex";static String PROPERTY_FLASH = "Flash";Camera camera;int cameraId;LinearLayout bottomPanel;RelativeLayout cameraButtonLayout;Button cameraButton;OrientationEventListener orientationObserver;int iconOrientation = Surface.ROTATION_0;Camera.Size[] pictureSizes;int pictureSizeIndex = 0;SharedPreferences preferences;boolean useAutoFocus = false;boolean useFlash = false;boolean hasFlash = false;ActionBar actionBar = null;Button settingsButton;ImageView settingsImageView;Button logOutButton;ImageView logOutImageView;Button flashButton;ImageView flashImageView;ImageView cameraIconImage = null;PopupWindow popupWindow = null;int popupWindowCenter = 0;SurfaceHolder surfaceHolder = null;

//region Activity lifecycle

@Overridevoid onCreate(Bundle savedInstanceState)
{.onCreate(savedInstanceState);(R.layout.activity_camera);.clean();= tryGetBackCamera();= getPictureSizes(camera);= PreferenceManager.getDefaultSharedPreferences(this);= preferences.getInt(PROPERTY_SIZE_INDEX, 0);= getFlashAvailability();= hasFlash && preferences.getBoolean(PROPERTY_FLASH, false);= ((SurfaceView)findViewById(R.id.cameraSurface)).getHolder();surfaceHolder != null;.addCallback(this);= getActionBar();actionBar != null;.setDisplayShowTitleEnabled(false);.setDisplayUseLogoEnabled(false);.setDisplayShowHomeEnabled(false);.setDisplayShowCustomEnabled(true);(R.id.cameraButton).setOnClickListener(this);(R.id.cancelButton).setOnClickListener(this);(R.id.continueButton).setOnClickListener(this);= (LinearLayout)findViewById(R.id.bottomPanel);= (Button)findViewById(R.id.cameraButton);= (ImageView)findViewById(R.id.cameraIconImage);= (RelativeLayout)findViewById(R.id.cameraButtonLayout);

}void onPause() {.disable();().edit().(PROPERTY_SIZE_INDEX, pictureSizeIndex).(PROPERTY_FLASH, useFlash).().(!bottomPanelHidden) {();

}
.onPause();
}

@Overridevoid onResume() {();().(settingsButton != null) {.enable();

}(camera == null) {.tryGetBackCamera();

}().(surfaceHolder);.startPreview();.onResume();

}

//endregion

//region Methods for cameraCamera tryGetBackCamera()
{(getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)) {= findBackFacingCamera();(cameraId >= 0) {Camera.open(cameraId);

}null;

}int findBackFacingCamera() {cameraId = -1;numberOfCameras = Camera.getNumberOfCameras();(int i = 0; i < numberOfCameras; i++) {.CameraInfo info = new Camera.CameraInfo();.getCameraInfo(i, info);(info.facing == Camera.CameraInfo.CAMERA_FACING_BACK) {i;;

}cameraId;

}void closeCamera() {(camera != null) {.release();= null;
```

```

    }int getRotationValue() {rotation = this.getWindowManager().getDefaultDisplay().getRotation();degrees =
0;(rotation) {Surface.ROTATION_0: degrees = 0; break;Surface.ROTATION_90: degrees = 90;
break;Surface.ROTATION_180: degrees = 180; break;Surface.ROTATION_270: degrees = 270; break;
    }.CameraInfo info = new Camera.CameraInfo();getCameraInfo(cameraId, info);(info.orientation - degrees +
360) % 360;
    }boolean getFlashAvailability() {
        //noinspection ConstantConditions(camera != null &&
getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA_FLASH)) {<String> modes =
camera.getParameters().getSupportedFlashModes();modes != null && // workaround for some tablets an chinese
devices
        !(modes.size() == 1 &&.Parameters.FLASH_MODE_OFF.equals(modes.get(0)));
        } false;
        } Camera.Size[] getPictureSizes(Camera camera) {<Camera.Size> sizes =
camera.getParameters().getSupportedPictureSizes();sizes != null;count = sizes.size();(count > 3) {= sizes.subList(count
- 3, count);
        } sizes.toArray(new Camera.Size[sizes.size()]);
        } void setPictureSize(int index) {.Size newSize = pictureSizes[index];= index;.Parameters parameters =
camera.getParameters();.setPictureSize(newSize.width, newSize.height);.setParameters(parameters);
        } void setFlashMode(boolean useFlash) {.useFlash = useFlash;.Parameters parameters =
camera.getParameters();.setFlashMode(useFlash ?.Parameters.FLASH_MODE_ON :
Camera.Parameters.FLASH_MODE_OFF);.setParameters(parameters);
        } void setUpCameraParameters() {.Size newSize = pictureSizes[pictureSizeIndex];.Parameters parameters =
camera.getParameters();<String> focusModes = parameters.getSupportedFocusModes();focusModes !=
null;(focusModes.contains(Camera.Parameters.FOCUS_MODE_AUTO))
        {.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);= true;
        } else if (focusModes.contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE))
        {.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE);
        }.setSceneMode(Camera.Parameters.SCENE_MODE_AUTO);.setWhiteBalance(Camera.Parameters.WHITE_
BALANCE_AUTO);.setExposureCompensation(0);.setPictureSize(newSize.width, newSize.height);(hasFlash)
        {.setFlashMode(useFlash ?.Parameters.FLASH_MODE_ON : Camera.Parameters.FLASH_MODE_OFF);
        }.setParameters(parameters);
        } void resetCameraPreview(SurfaceHolder surfaceHolder) {{.setDisplayOrientation(getRotationValue());
//setCameraRotation(getRotationValue());.setPreviewDisplay(surfaceHolder);
        } catch (IOException exception) {};
        }
    }
    //endregion
    //region Options menu
    @Overrideboolean onCreateOptionsMenu(Menu menu) {inflater = (LayoutInflater)
this.getSystemService(LAYOUT_INFLATER_SERVICE);view = inflater.inflate(R.layout.menu_camera,
null);.setCustomView(view);= (Button)view.findViewById(R.id.logoutButton);=
(ImageView)view.findViewById(R.id.logoutImageView);= (Button)view.findViewById(R.id.flashButton);=
(ImageView)view.findViewById(R.id.flashImageView);= (Button)view.findViewById(R.id.settingsButton);=

```

```

(ImageView)view.findViewById(R.id.settingsImageView);setOnClickListener(this);setOnClickListener(this);(hasFlash) {setOnClickListener(this);
    } else {setEnabled(false);setAlpha(0.7f);
    }();enable();true;
}
//endregion
//region Click listener callbacks
@Overridevoid onClick(View view) {(view.getId()) {R.id.cameraButton :(camera != null)
{.setEnabled(false);(useAutoFocus) {.autoFocus(this);
    } else {.takePicture(null, null, this);
    }
};R.id.cancelButton:.clean();();(camera != null) {.startPreview();
};R.id.continueButton:();R.id.logoutButton:();R.id.flashButton:(!useFlash);();R.id.settingsButton:(popupWindow.isShowing()) {.dismiss();
    } else {.showAsDropDown(settingsButton);
    };
}
@Overridevoid onItemClick(AdapterView<?> adapterView, View view, int i, long l) {(i);
}
//endregion
//region Navigationvoid navigateToLoginActivity() {intent = new Intent(this,
LoginActivity.class);.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);(intent);
    }void navigateToDataActivity() {intent = new Intent(this, DataActivity.class);();(intent);
    }
//endregion
//region SurfaceHolder.Callback implementation
@Overridevoid surfaceCreated(SurfaceHolder surfaceHolder) {(surfaceHolder);
}
@Overridevoid surfaceChanged(SurfaceHolder surfaceHolder, int i, int i2, int i3) {.startPreview();
}
@Overridevoid surfaceDestroyed(SurfaceHolder surfaceHolder) {();
}
//endregion
//region Camera callbacks
@Overridevoid onPictureTaken(byte[] bytes, Camera camera) {.storeImage(correctImage(bytes));();
}void onAutoFocus(boolean b, Camera camera) {.takePicture(null, null, this);
}
//endregionbyte[] correctImage(byte[] data) {{realImage = BitmapFactory.decodeByteArray(data, 0,
data.length);bitmap = rotateImage(realImage, Math.abs(iconOrientation - 90));stream = new
ByteArrayOutputStream();compress(Bitmap.CompressFormat.JPEG, 100, stream);[] newData =
stream.toByteArray();close();d("Camera", String.format("Image size: %d", newData.length));newData;
    } catch (IOException e) {.printStackTrace();
}

```

```

    }null;

    }Bitmap rotateImage(Bitmap bitmap, int degree) {width = bitmap.getWidth();height =
bitmap.getHeight();matrix = new Matrix();postRotate(degree);Bitmap.createBitmap(bitmap, 0, 0, width, height, matrix,
true);

    }

    //region Visual state managementboolean bottomPanelHidden = true;void swapElementsVisibility() {=
!bottomPanelHidden;.setEnabled(bottomPanelHidden);setEnabled(bottomPanelHidden);setEnabled(bottomPanelHidd
en);(bottomPanelHidden) {.setVisibility(View.GONE);setVisibility(View.VISIBLE);

    } else {.setVisibility(View.VISIBLE);setVisibility(View.GONE);

    }void rotatePopup(int toOrientation) {(popupWindow != null) {needsToShow =
popupWindow.isShowing();dismiss();view =
popupWindow.getContentView();setPivotY(popupWindowCenter);setPivotX(popupWindowCenter);setRotation(toOr
ientation);(needsToShow) {.showAsDropDown(settingsButton);

    }void tryStartRotation(View view, int toOrientation) {(view != null) {animation =
getAnimation(toOrientation); // New animation object is needed to restore animation's center
point.startAnimation(animation);

    }void rotateImages(int toOrientation) {(logoutImageView, toOrientation);(flashImageView,
toOrientation);(settingsImageView, toOrientation);(cameraIconImage, toOrientation);

    }RotateAnimation getAnimation(int toOrientation) {animation = new RotateAnimation(,
toOrientation,RELATIVE_TO_SELF, 0.5f, RotateAnimation.RELATIVE_TO_SELF,
0.5f);setFillAfter(true);setDuration(600);animation;

    }void updateFlashIcon() {(useFlash) {.setImageResource(R.drawable.ic_action_flash_on);makeText(this,
R.string.toast_flash_on, Toast.LENGTH_SHORT).show();

    } else {.setImageResource(R.drawable.ic_action_flash_off);makeText(this, R.string.toast_flash_off,
Toast.LENGTH_SHORT).show();

    }

    //endregion

    //region Setup methodsvoid setUpOrientationEventListener() {(orientationObserver == null) {= new
OrientationEventListener(this, SensorManager.SENSOR_DELAY_NORMAL) {

        @Overridevoid onOrientationChanged(int i) {newOrientation = -100;(i >= 315 || i < 45) {= 0;

        } else if (i >= 45 && i < 135) {= -90;

        } else if (i >= 225 && i < 315) {= 90;

        }(newOrientation > -100 && iconOrientation != newOrientation) {(newOrientation);(newOrientation);=
newOrientation;

        }PopupWindow setUpPopupWindow() {(popupWindow == null) {layoutInflater =
(LayoutInflater)getContext()

        .getSystemService(LAYOUT_INFLATER_SERVICE);popupView =
layoutInflater.inflate(R.layout.menu_camera_popup, null);popupView != null;list =
(ListView)popupView.findViewById(R.id.qualitiesListView);dummy =
(TextView)popupView.findViewById(R.id.dummyTextView);<String> humanReadableHeaders =
getHumanReadableHeaders();(list,
humanReadableHeaders);setOnItemClickListener(this);setItemChecked(pictureSizeIndex, true);

```

```

// Measure is need to set PopupWindow width similar to WRAP_CONTENT behaviour
// Unfortunately this property of popupView is ignored and without this fix
// PopupWindow's width is equal to screen width..measure(View.MeasureSpec.UNSPECIFIED,
View.MeasureSpec.UNSPECIFIED);side = popupView.getMeasuredWidth();= side / 2;=
getPopupWindow(popupView, side, side);setOnClickListener(new View.OnClickListener() {
    @Overridevoid onClick(View view) {.dismiss();
    }popupWindow;
    }ArrayList<String> getHumanReadableHeaders() {[] headers =
this.getResources().getStringArray(R.array.quality_headers);<String> humanReadableHeaders = new
ArrayList<String>();sizeCount = pictureSizes.length;(int i = 0; i < 3; i++) {(i < sizeCount) {.Size size =
pictureSizes[i];.add(.format("%s (%dx%d)", headers[i], size.width, size.height));
    } else {;
    }humanReadableHeaders;
    }void setPopupListContent(ListView list, final ArrayList<String> humanReadableHeaders) {array =
getTheme().obtainStyledAttributes(int[] {android.R.attr.listPreferredItemHeightSmall});array != null;int height =
array.getDimensionPixelSize(0, -1);.recycle();
    // Extended adapter is needed to set equal width to all items in pop up list.
    // It's supposed that the first row is the widest one (and it's true).
    // It's obvious that such solution can cause bugs with list item's size.
    // Unfortunately there is no other way to simulate action bar drop down
behaviour..setAdapter(ArrayAdapter<String>(this,.layout.quality_list_item,.id.headerTextView,) {int maxWidth = 0;
    @OverrideView getView(int position, View convertView, ViewGroup parent) {view =
super.getView(position, convertView, parent);view != null;(position == 0)
{.measure(View.MeasureSpec.UNSPECIFIED, View.MeasureSpec.UNSPECIFIED);= view.getMeasuredWidth();
    } else {.setLayoutParams(new ListView.LayoutParams(maxWidth, height));
    }view;
    }PopupWindow getPopupWindow(View view, int width, int height) {popupWindow = new
PopupWindow(view, width, height, true);
    //noinspection deprecation.setBackgroundDrawable(new
BitmapDrawable());.setTouchable(true);.setOutsideTouchable(true); popupWindow;
    }
    //endregion
}

```

ДОДАТОК Б
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ РОБОТИ СПІВРОБІТНИКІВ

Виконав: студентка 4 курсу, групи КНТ-127
Гринчишина Марина Сергіївна
Керівник: Степаненко Олександр Олексійович,
доцент

2021 рік

Рисунок Б.1 – Слайд №1

Успіх роботи підприємства забезпечують працівники, зайняті на ньому і тому, сучасна концепція управління підприємством передбачає виділення з великого числа функціональних сфер управлінської діяльності тієї, яка пов'язана з управлінням кадрової складової виробництва - персоналом підприємства.

- Об'єктом дослідження є система методів, принципів і порядок оцінки співробітників.
- Предметом розгляду є можливості вдосконалення роботи з персоналом для підвищення якості його роботи.
- Мета дипломної роботи – розробити програмну реалізацію моніторингу та аналізу роботи співробітників відділу маркетингу.

Слайд 2

Рисунок Б.2 – Слайд №2

Слайд 3

Для реалізації поставленої мети поставлені наступні завдання:

- ▶ провести аналіз чинників та критеріїв оцінки персоналу;
- ▶ провести аналіз основних принципів і методів оцінки роботи персоналу;
- ▶ провести аналіз середовища розробки;
- ▶ провести системне та функціональне проєктування;
- ▶ розробити програму для моніторингу та аналізу роботи співробітників.

Рисунок Б.3 – Слайд №3

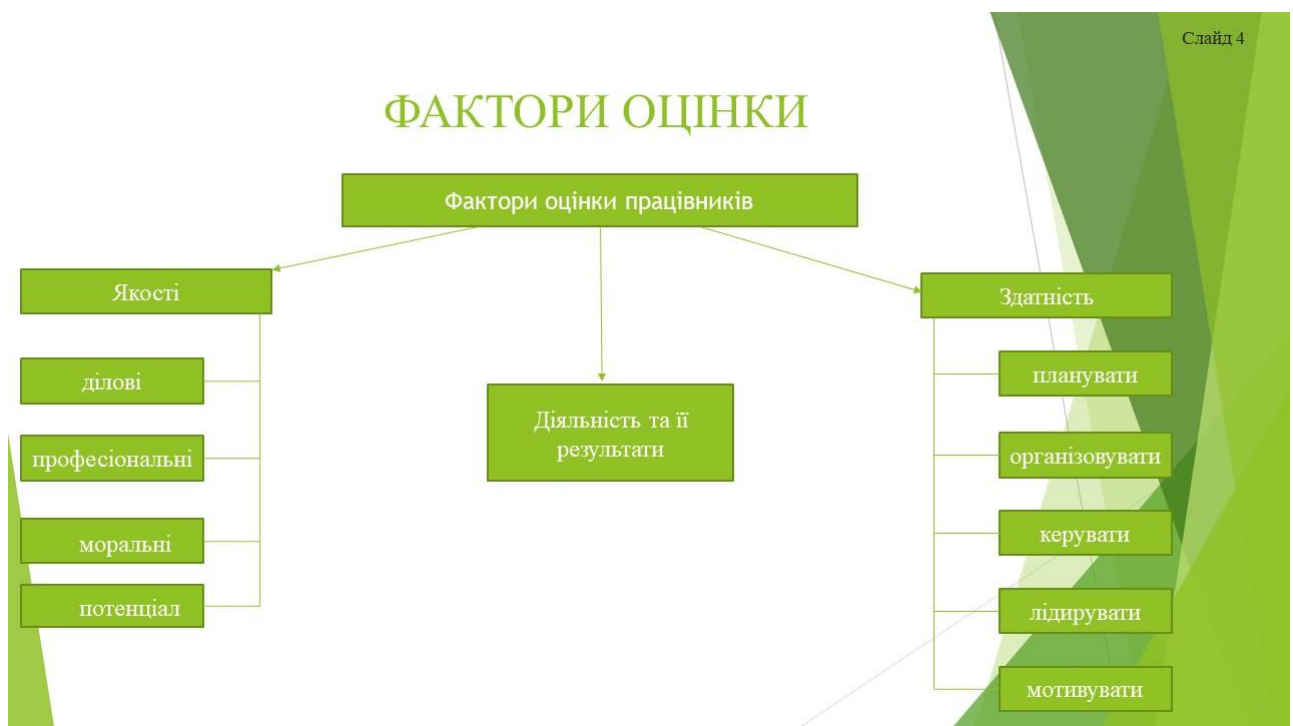


Рисунок Б.4 – Слайд №4

ОЦІНКА РОБОТИ ПЕРСОНАЛУ



Рисунок Б.5 – Слайд №5

СЕРЕДОВИЩЕ РОЗРОБКИ

Існує чотири основні інтегровані середовища розробки з підтримкою Android-проектів:

- Eclipse + ADT Plugin;
- IntelliJ IDEA;
- NetBeans;
- Android Studio.

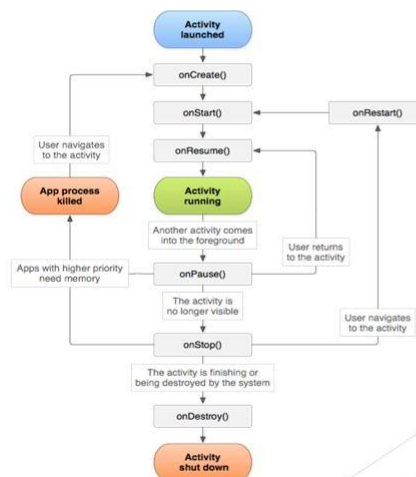


Рисунок 1 - Життєвий цикл Activity

Рисунок Б.6 – Слайд №6

СТРУКТУРНА СХЕМА ПРОГРАМИ

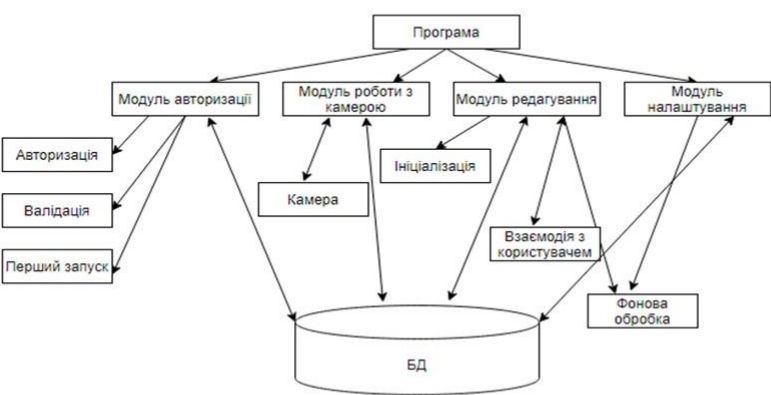


Рисунок 2 – Структурна схема програми

Рисунок Б.7 – Слайд №7

ПРОЄКТУВАННЯ

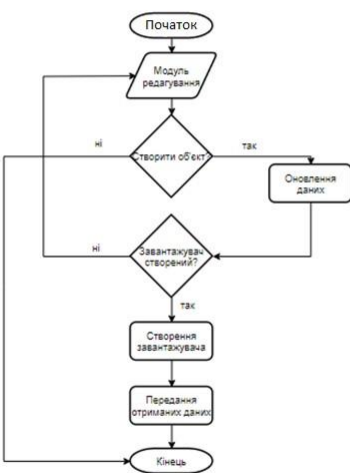


Рисунок 3 – Блок-схема взаємодії з БД

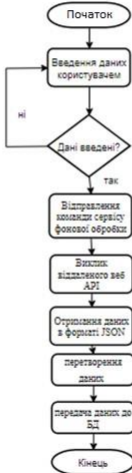


Рисунок 4 – Алгоритм для поновлення стану кеша

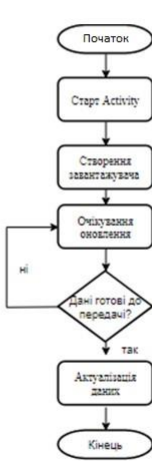


Рисунок 5 – Алгоритм читання з БД

Рисунок Б.8 – Слайд №8



Рисунок Б.9 – Слайд №9

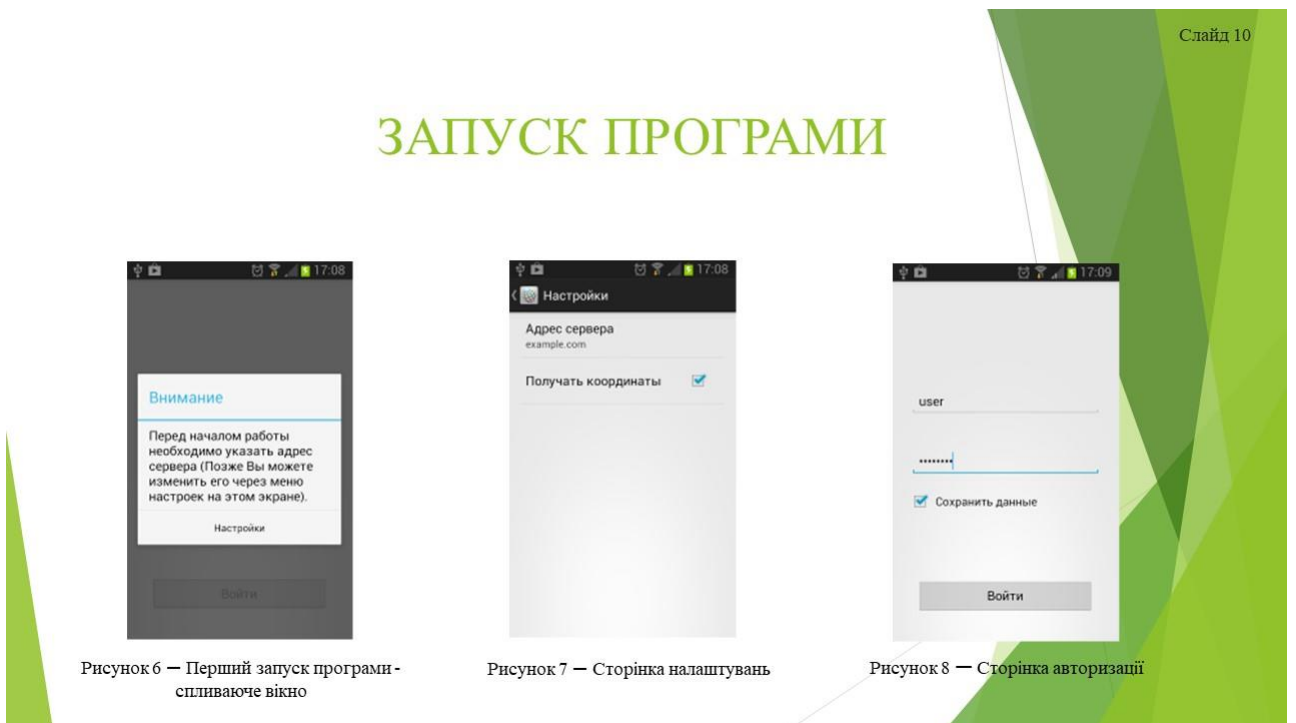


Рисунок Б.10 – Слайд №10

Слайд 11

СТОРІНКА ВЗАЄМОДІЇ З КАМЕРОЮ



Рисунок 9 —
Загальний вигляд камери

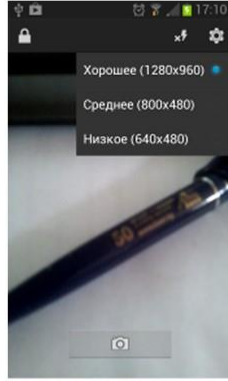


Рисунок 10 —
Меню вибору розрішення



Рисунок 11 —
Підтвердження збереження знімка

Рисунок Б.11 – Слайд №11

Слайд 12

ПРОЦЕС ВІДПРАВКИ ФОТО

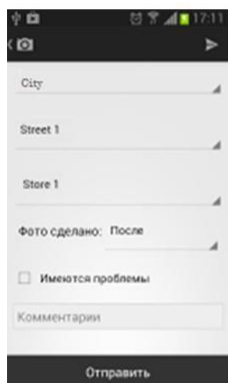


Рисунок 5.7 —
Сторінка метаданих

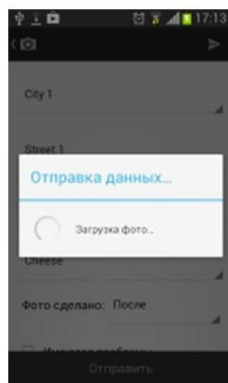


Рисунок 5.8 —
Завантаження фото

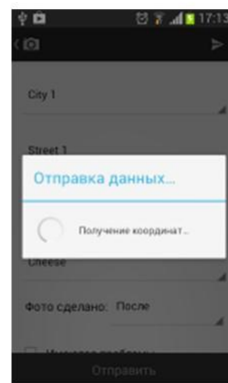


Рисунок 5.9 —
Отримання координат

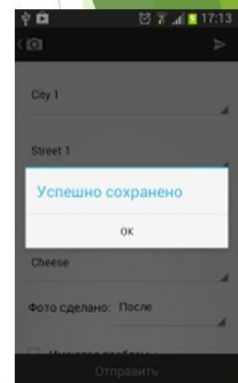


Рисунок 5.10 —
Завершення збереження

Рисунок Б.12 – Слайд №12

ВИСНОВОК

В результаті виконання роботи виконано наступні завдання:

- ▶ проведено аналіз чинників та критеріїв оцінки персоналу;
- ▶ проведено аналіз основних принципів і методів оцінки роботи персоналу;
- ▶ проведено аналіз середовища розробки;
- ▶ проведено системне та функціональне проєктування;
- ▶ розроблено програму для моніторингу та аналізу роботи співробітників.

Розроблений проєкт є лише базовим функціоналом, але при цьому є цілком достатнім засобом для ведення звітності.

Дякую за увагу!

Рисунок Б.13 – Слайд №13