МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних
технологій

Кваліфікаційна робота бакалавра

на тему: «Розробка системного застосунку "Cleaner" для мобільних пристроїв на платформі Android»

Виконала студентка групи К-19і
спеціальності 122 Комп'ютерні науки
Касьян Марина Олександрівна

Керівник к.т.н., доцент
Фразе-Фразенко Олексій Олексійович

Рецензент регіональний
координатор Програми EGAP
Копиченко Іван Юрійович

Одеса 2021

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗП – постійний запам'ятовуючий пристрій.

ОЗП – оперативний запам'ятовуючий пристрій.

ІС – інформаційна система.

SADT – Structured Analysis and Design Technique – методологія структурного аналізу і проектування.

VPN – Virtual Private Network - віртуальна приватна мережа.

GUI – Graphical User Interface – графічний користувацький інтерфейс.

JVM - Java Virtual Machine – віртуальна машина Java.

ВСТУП

В даний час обчислювальна техніка розвивається бурхливими темпами: всім відомий закон Мура, згідно з яким обчислювальна потужність комп'ютерів подвоюється кожні півтора року. Дослідники вважають, що цифрові технології, в тому числі мобільні застосунки, будуть стрімко розвиватися, виходячи з минулих тенденцій.

Якщо провести аналогію між додатками, які ми використовуємо сьогодні і які ми використовували кілька років тому, різниця буде дивовижною.

Люди шукають застосунок, який був би цілісним і безмежним.

Мобільний застосунок – це спеціальне програмне забезпечення, яке розробляється для смартфонів, планшетів та інших мобільних пристроїв.

Ми використовуємо по кілька різних мобільних застосунків кожен день, і кожне з них значно полегшує життя.

Мобільні застосунки стають все більш різноманітними, дозволяючи користувачам знаходити оптимальний варіант для себе.

Так само, як це відбувається з комп'ютером, мобільні пристрої з часом накопичують безліч непотрібних файлів. Це відбувається тому, що смартфони стають все більш потужними, щоб задовольнити потреби покупців, і зберігають різні непотрібні дані, наприклад, файли cookie та залишки застосунків. Тому, в ідеалі, час від часу потрібно чистити гаджет, щоб прибрати ці файли з пристрою. При регулярному чищенні телефон буде мати більше пам'яті для швидкого запуску застосунків, перегляду інтернету і навіть для запуску досить великовагових ігор.

Можна почистити пристрій вручну, видаляючи файли, кеш кожної програми, переглядаючи зображення і завантаження, але зручніше – за допомогою спеціальних програм, які роблять це ефективніше і дуже швидко.

Cleaner – це інтелектуальний інструмент для очищення і аналізу застосунків, який дозволяє звільнити місце в пам'яті, підвищити продуктивність і рідше заряджати пристрій.

Спеціальні застосунки чистильники працюють на більш глибоких рівнях, і тому ефективніше, ніж ручна очистка. Система Android має багато різних прихованих процесів, які завжди працюють у фоновому режимі, але на відміну від комп'ютера або ноутбука доступ користувачів до цих процесів не завжди можливий.

Перед тим, як очистити андроїд від сміття, потрібно знати і розуміти, що таке «сміття» для телефону. Поспішне видалення всіх незнайомих файлів може привести до втрати важливої системної інформації

До сміття на телефоні можна віднести наступні файли.

Кеш. Це тимчасові файли, які застосунки створюють для своїх потреб. Кеш-файли дозволяють застосункам працювати швидше, підтягуючи потрібну інформацію. Після їх застосування, такі файли повинні автоматично видалятися, але дуже часто це не відбувається. Таким чином, кеш з усіх застосунків накопичується і забирає вільну пам'ять.

Файли віддалених програм. Дуже часто видалення програми не призводить до того, що вона повністю зникає з гаджета. Часто на телефоні залишаються втрачені або пошкоджені файли застосунків, в яких вже немає необхідності.

Такі непотрібні файли значно ускладнюють роботу гаджета. Застосунки з очищення знаходять ці приховані, фонові процеси і вбивають незайняті, які поглинають простір пам'яті, вивільняючи ресурси для продуктивної роботи. В наш час основними платформами для мобільних пристоїв є Android, Windows Phone, iOS и Symbian. Для даного проекту нас цікавить саме платформа Android.

Смартфони та інші мобільні пристрої стали частиною нашого повсякденного життя. За допомогою мобільних телефонів можна не тільки спілкуватися один з одним, але і замовляти товари з магазинів, купувати

квитки, бронювати житло, викликати таксі, використовувати телефони як навігатори, фото- і відеокамери, онлайн книги, онлайн банки і тд. Актуальність розробки мобільних застосунків зростає не те що з кожним роком, а й з кожним місяцем.

Темою мого дипломного проекту є розробка системного застосунку «Cleaner» для мобільних пристроїв на платформі «Android».

Розроблюване програмне забезпечення повинно забезпечити прискорення роботи мобільного пристрою, очищення оперативної пам'яті, збільшення терміну служби батареї, охолодження перегрітого телефону, очищення кеша і «сміттєвих» файлів та мати зручний інтерфейс.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Платформа Android для мобільних пристроїв

ОС Android це відкрита платформа для смартфонів і планшетних комп'ютерів, не прив'язана до конкретного пристрою. Апарати на базі цієї ОС виробляють такі великі фірми, як HTC, Motorola, Samsung, Sony Ericsson, LG та ін. На основі Android випускається широкий спектр пристроїв, від економ-класу до преміум-класу.

Дана ОС заснована на ядрі операційної системи Linux, але в ній використовуються не всі її можливості. Перша версія Android була представлена у вересні 2008 р Реліз Android 2.2 Frovo включав підтримку веб-технологій HTML5 і Flash 10.1. Версія ОС Android 2.3 Gingerbread вийшла в 2011 р, в ній були виправлені багато колишні недоробки і додані нові можливості. Вони включали оновлений користувальницький інтерфейс, підтримку стандарту NFC (Ncar Field Communication -технологія бездротового високочастотного зв'язку малого радіусу дії), кількох камер і дисплеїв високої роздільної здатності. А у версії Android 3.0 Honeycomb, розробленої спеціально для планшетних комп'ютерів, застосовані зовсім інший користувальницький інтерфейс, тривимірні ефекти, зручний браузер і безліч інших поліпшень.

Застосунки для пристроїв на основі цієї ОС пропонуються в магазині Play Market (Android Market). Синхронізувати програми можна через Google-аккаунт, або доведеться завести Google-пошту (Gmail) і через неї встановити контакти в мобільний пристрій.

До достоїнств пристроїв на основі цієї ОС слід віднести можливість використовувати їх в якості зовнішнього диска комп'ютера, щоб копіювати з нього файли. Починаючи з версії 2.2 операційна система Android дозволяє використовувати смартфон як USB-модема або точки доступу, забезпечуючи цим вихід в Інтернет іншим пристроям.

Сьогодні Android, мабуть, найпопулярніша в світі платформа для мобільних пристроїв. Android навколо нас – вдома, в офісі, в камерах і годинниках, в автомобілях і телевізорах. Застосунки Android мають доступ до провідного відкритого торгового майданчика для поширення: Google Play. У магазині Google Play більше 1 мільйона застосунків доступних для завантаження на ваш пристрій Android. Вирішити, яку з них завантажити і встановити на телефон або планшет Android, може виявитися складним завданням. Частиною з них ви вже користуєтеся, є і більш нові або просто не знайомі застосунки, які варто спробувати.

1.2 Типи мобільних застосунків

Мобільні застосунки розрізняються залежно від операційної системи, в якій вони працюють – основними з яких є Android і iOS.

1.2.1 Нативні (рідні) застосунки

Нативні застосунки працюють з ОС пристрою так, що вони мають можливість працювати швидше та гнучкіше, ніж альтернативні типи програм. Якщо застосунок поширюється на користувачів різних типів пристроїв, розробники створюють окрему версію програми для кожного з них.

Нативна розробка має на увазі створення програми для мобільного пристрою на конкретній мові під конкретну платформу. Нативні застосунки досить продуктивні і не мають обмежень в розробці (Java і Kotlin – для Android, а Swift – для iOS).

Переваги нативних застосунків включають[1]:

– широкі функціональні можливості завдяки використанню можливостей базового пристрою;

– швидке та чутливе виконання програмного забезпечення;
– Push-повідомлення;
– інтерфейс користувача, який краще відповідає його досвіду ОС;
– забезпечення якості, завдяки рейтингам в магазинах застосунків.

1.2.2 Гібридні застосунки

На противагу нативним, гібридні застосунки універсальні, вони створюються одразу для багатьох платформ. Можна сказати, що це веб-сайт у форматі додатку.

Переваги гібридних застосунків:

1. Економія. Гібридний застосунок дає можливість розробити простий і ефективний продукт недорого і швидко, причому, одразу для кількох платформ.
2. Швидкий вихід на ринок. Менший термін розробки дозволяє швидше вийти на ринок і швидше отримати перший прибуток.
3. Універсальність. Гібридні застосунки створюються під різні платформи одразу, після чого доволі просто адаптуються під кожну з них, вони доступні більш широкому колу користувачів, тому мають більше шансів бути поміченими користувачами.

Як і нативні, гібридні застосунки мають ряд недоліків:

1. Швидкість. Це одна із жертв в ім’я універсальності. На жаль, гібридні застосунки часто програють у швидкості. Наприклад, при скролінгу картинок, анімації.
2. Дизайн. iOS та Android застосунки мають власні стандарти дизайну, тому дизайн для різних ОС потрібно розробляти окремо.

1.2.3 Веб-застосунки (PWA)

Значної популярності сьогодні набирають PWA (Progressive Web App). Це веб-застосунок, який є веб-сторінкою, але поводиться і виглядає як мобільний застосунок. Тобто, користувач отримує доступ до інформації та можливостей без скачування додатку. Це можливо завдяки Service Worker (це скрипт, який браузер запускає «на фоні», окремо від веб-сторінки). Це основа кожного PWA. Service Worker здійснюють надійне та розумне кешування, оновлення фонового вмісту, push сповіщення та дають доступ до офлайн функціоналу попередньо відвіданих сайтів. Тобто, після першого відвідуванння веб-сайту, сайт і застосунок будуть працювати надійно навіть при нестабільному покритті.

1.2.4 Порівняльний аналіз існуючих аналогів

Перед тим, як почати розробку будь-якого проекту, необхідно провести аналіз існуючих систем, виявити «сильні» та «слабкі» сторони і не зробити свій проект ще кращим та ефективнішим. Тож почнемо з розгляду одного з найвідоміших застосунків – CCleaner.

Застосунок CCleaner дозволить:

– поліпшити продуктивність – смартфон буде вільний від непотрібного сміття та швидше працювати;
– отримати контроль над смартфоном – можна побачити, які програми «з'їдають» його пам'ять і зможете вирішити що з ними робити;
– видалити файли – позбутися від старих WhatsApp і текстових повідомлень, кеша, дзвінків та інших файлів які вам не потрібні. Просто оберіть які файли ви хочете видалити: все або певні;
– отримати більш детальну інформацію про застосунки і вирішити, що з ними робити: видалити або вимкнути ті, якими не користуєтеся;

– звільнити місце – для застосунків, музики або фото, які дійсно потрібні;
– зберегти файли, які потрібні – застосунок видалить тільки те що вирішите видалити.

Розчарування в CCleaner почалися відразу після першого запуску програми. Відкривши додатком вперше можна дізнатися, що CCleaner оптимізує пристрій, зробить його чистим, безпечним і швидким. А ось далі одразу ж запропонує оновити програму до рівня PRO-версії, вказавши на такі переваги:

– відсутність реклами;
– планове очищення (Запланувати автоматичне очищення – можна обрати, коли на вашому телефоні будуть видалятися файли і CCleaner зробить це автоматично);
– аналізатор сховища (Дозволить моментально видалити непотрібні файли зі сховища – знайти «ворогів» і відразу ж видалити);
– пріоритет підтримки (На випадок, якщо знадобиться додаткова допомога);
– майбутні функції (Більше можливостей очікується в майбутньому і всі вони будуть доступні в CCleaner Pro).

На головному екрані утиліти відображається кількість ПЗП і ОЗП у вигляді діаграми і цифр, поруч є клавіші для аналізу і очищення системи. Відразу недолік, вся пам'ять в даному додатку об'єднана в загальний масив, в моєму випадку він становить близько 40 Гбайт. Але ж один носій – це вбудована пам'ять, а інший – знімна. Навіщо сканувати MicroSD-карту, витрачати на це зайвий час і ресурси, якщо, припустимо, на флеш-карті зберігаються тільки фільми і музика? Можливості відключити знімний носій або розділити пристрої зберігання в додатку немає.

При чищенні можна вибирати, що саме потрібно видаляти. У разі, якщо це якась важлива папка, наприклад, «Завантаження» розробники повідомлять нас про це.

Висновок: на жаль, CCleaner для android-пристроїв – це аж ніяк не той «СіСіКлінер» до якого всі так звикли. Це абсолютно інша програму, що нагадує про приналежність до хіту лише кольоровою гамою інтерфейсу, назвою та іменем розробника.

Не варто й казати, що колишньої функціональності не отримати, зате в навантаження йде реклама і серйозні обмеження функцій. Правда, в очищенні системи «безкоштовним» користувачам не відмовлять, та й на якості це ніяк не відіб'ється. Видаляє програма дуже детально і навіть дає корисні поради по ручному видаленню даних.

В цілому, застосунок показує непоганий результат, просто від такого іменитого додатку очікувалося більше.

Наступним розглянемо не менш відомий помічник – SD Maid.

SD Maid – це той застосунок, що, за словами розробників, допоможе нам тримати пристрій в чистоті і порядку.

Цей застосунок дасть вам такі можливості:

- переглядати все, що є на пристрої та керувати файлами через повноцінний провідник;
- видаляти непотрібні файли з системи;
- можна керувати встановленими користувацькими і системними додатками;
- знаходити файли, що залишилися від віддалених застосунків;
- шукати файли по імені, за змістом або датою;
- отримувати детальний аналіз пам'яті мобільного пристрою;
- оптимізувати бази даних;
- виконувати поточну очистку та вилучення зайвих файлів, процес, який замінює те, що у інших може позначатися як «очищення кешу»;
- виявляти дублікати фотографій, музики або документів, незалежно від імені або розташування;
- автоматично запускати застосунок за розкладом або за допомогою віджетів.

Базове сканування здійснюється в чотири ступені: сміття, система, застосунки і бази даних. Відповідно, всі кроки відбуваються послідовно, загальний час сканування пристрою складає приблизно 20-25 секунд. Природно, цей параметр варіюється від обсягу файлового сховища, його швидкості, потужності девайса і кількості встановлених застосунків.

За підсумком тестування біля кожного розділу відображається кількість пам'яті, доступної для вивільнення. І якщо порівняти ці результати з тим же CCleaner, то можна помітити явну відсутність пари сотень мегабайт. Та й ще й обмеження по очищенню кеша застосунків в безкоштовній версії SD Maid не потішать.

SD Maid вміє шукати дублікати файлів, причому, це стосується абсолютно всіх типів файлів. Ще утиліта може аналізувати файли в пристроях зберігання і розподіляти їх розміру. Це допоможе виділити об'ємні папки і файли для більш детальної перевірки. Окремої уваги вартий розділ, іменований як «Винятки». У ньому можна задати заборона на очистку певних розділів пам'яті.

З специфічних функції SD Maid відзначимо розділ «Пошук» і «Провідник». Остання опція буде корисна тим, хто придбав на своєму пристрої права Суперкористувача. При наявності ROOT-прав вона стане ROOT-провідником і зможе видаляти, копіювати, переміщати і перейменовувати файли в системному розділі ОС Android.

Власники пристроїв з безкоштовною версією SD Maid не зможуть виконувати очищення виробу за розкладом, та й доступу до статистики у них теж не буде. Ще можна згадати про заборону видалення кеша застосунків. Виходить, розробники «стиснули» застосунок як могли.

Висновок: У теорії програма сканує пристрій, встановлені застосунки та інші файли, але як справа доходить до результатів, то вона може видалити лише пару Мбайт файлів, решта або не сканується, або це можливо тільки в розширеній версії додатка.

Далі подивимось як працює Clean Master, його недоліки та переваги.

Clean Master – це саме та утиліта, яку відразу уявляєш після таких слів, як Android, оптимізація, очищення пам'яті.

Основні функції Clean Master:

1. Очищення сміття. Clean Master допомагає звільняти пам'ять, видаляючи непотрібні і залишкові файли, файли кешу, які уповільнюють роботу телефону.
2. За допомогою цього професійного “прибиральника” також зможнна звільнити ще більше пам'яті, видаляючи дані застосунків соціальних мереж, таких як Facebook, Messenger, Whatsapp, Instagram, і при цьому не хвилюватися про те, що випадково видаліть не ті файли.
3. Безкоштовний антивірус. Безкоштовний антивірусний “движок” Clean Master-a, №1 в рейтингу Av-Test, сканує всі застосунки (не важливо, встановлені чи ні) на наявність вірусу, блокує і видаляє вірус, щоб убезпечити ваш телефон, захищає вашу особисту інформацію.
4. «Приватні фото» – приховати фото. Функція «приватні фото» захищає ваші фотографії шляхом їх блокування. Використовуючи “приватні фото” ви можете убезпечити свої зображення і захистити свою конфіденційність.
5. Безпека Wi-Fi. Для виявлення фальшивого Wi-Fi і несанкціонованих підключень в Clean Master додана функція “Безпека Wi-Fi”. Збережіть безпеку свого телефону при користуванні ненадійним публічним Wi-Fi.
6. Прискорювач телефону. One tap boost допомагає прискорити телефон шляхом очищення ОЗП. Після виконання прискорення можна запустити тест швидкості і переконатися, наскільки швидше почав працювати телефон.
7. Економія заряду батареї. Перекладом застосунків в режим гібернації Clean Master допомагає економити енергію батареї і продовжити термін її служби.

Спочатку програма сканує пристрій на предмет різних файлів, після чого пропонує нам їх очистити. Якомога докладніше показується і розписується, що програма пропонує видалити, скільки пам'яті звільниться і так далі. Загалом, дуже зрозуміло і наочно, за це дякуємо розробникам, особливо від починаючих користувачів.

Також цікавий розділ «Розширена очищення» дозволяє в більш ручному режимі знайти і видалити великі файли, залишкові файли, фотографії, невикористовувані програми та навіть SMS-повідомлення.

Такий підхід найбільш цікавий тим, що Clean Master підійде як новачкам, які будуть раді автоматичному очищенню, так і більш просунутим користувачам, так як для них програма приготувала додаткові можливості по очищенню пам'яті і оптимізації системи.

Ще в android-додатку Clean Master є дуже багато різних інструментів:

– майстер ігор;
– блокування застосунків;
– приватна галерея (захист фотографій);
– безпека підключення до Wi-Fi (VPN-сервіс);
– безпечний інтернет (браузер без історії);
– антивірус;
– охолоджувач;
– економія енергії;
– майстер зарядки.

Висновок: Clean Master – це непогана утиліта для роботи з системою, її оптимізацією, обслуговуванням і очищенням пам'яті. І на перший погляд все добре: вона прекрасно очищає пам'ять пристрою і включає додаткові інструменти, в тому числі вбудований антивірус, безпечне підключення до інтернету за допомогою VPN і багато іншого.

Зате в недоліки програми можна записати величезну кількість реклами, постійні повідомлення і часті пригальмовування навіть на потужних пристроях.

1.2 Постановка завдання

Темою мого дипломного проекту є розробка системного застосунку “Cleaner” для мобільних пристоїв на платфомі Android.

Необхідно створити дизайн-проект для програмного додатку. Згідно з останніми тенденціями та дизайн-проектом, створити адаптивний інтерфейс користувача.

Розроблюване програмне забезпечення повинно виконувати наступні функції:

- очистка кеш-пам'яті смартфону;
- збір та аналіз даних про використання ОЗП;
- оптимізація роботи мобільного пристрою;
- зменшення температури процесору та продовження роботи батареї за рахунок енергоекономії.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Обґрунтування вибору мови програмування

Для порівняння розглянемо декілька найпопулярніших мов програмування.

Мова програмування С є однією з найпопулярніших і поширених мов. Вона являє собою скомпільовані загального призначення мови програмування з статичними типізація, розроблений в 1969-1973 Bell Labs програміст Денніс Річі (Денніс Річі). «»

Мова С часто згадується як “середнього рівня” або навіть “низькорівнева” мова програмування, так як вона поєднує в собі мови високого рівня з їх функціональністю і продуктивністю, і працює поруч з устаткуванням комп'ютера. В результаті, ми можемо маніпулювати даними на низькому рівні і використовувати високорівневу структуру для контролю роботи програми.

Спочатку мова С була призначена для написання операційної системи UNIX. Згодом С стала однією з найпопулярніших мов, а її основна сфера застосування – системне програмування, зокрема, створення операційних систем, драйверів, різних утиліт, антивірусів тощо. до речі, Linux в основному написаний на С. Однак, тільки системне програмування не обмежує використання цієї мови. Ця мова може бути використана в програмах будь-якого рівня, де важливі швидкість і продуктивність. Таким чином, ми можемо написати з використанням С і застосунки, і навіть веб-сайти (з використанням технології CGI-загальний шлюз інтерфейс). Але, звичайно, для створення GUI і веб-застосунків, є відповідні інструменти і технології, але тим не менше, діапазон використання С досить широкий.

Незважаючи на великі можливості мови С досить проста в той же час. Вона не містить багато конструкцій, бібліотек, її легко освоїти і вивчати. Тому її часто вибирають як мову для вивчення в загальному програмуванні.

С – це скомпільована мова, що означає, що компілятор переводить початковий код до виконуваного файлу, який містить набір інструкцій на

комп'ютері. Але різні платформи мають свої власні риси, тому скомпільовані програми не можуть переходити з однієї платформи на іншу. Однак, на рівні вихідного коду, С програми мають портативність, а також наявність компіляторів, бібліотек та інструментів розробки для майже всіх загальних платформ дозволяє компілювати той же вихідний код С в додатках на цих платформах .

Розвиток С має великий вплив на розвиток мов програмування. Зокрема, його синтаксис став основою для таких мов, як C++, C#, Java, PHP, JavaScript. Особливої згадки повинно бути зроблено з відносин з С. C++ безпосередньо походить від С. Але згодом їх розвиток був окремо одна від одної, і навіть не було несумісності між ними. Стандарт C++ додав ряд конфліктуючих функцій C++ на мову С. В результаті, зараз обидві мови насправді самодостатні і розвиваються самостійно.

Основні особливості мови «С»:

– універсальність – один і той же код можна зібрати практично на кожній платформі (якщо є компілятор для нього);
– висока швидкість виконання;
– компактний розмір, малий обсяг скомпільованих файлів.

Мова програмування C++ – високорівнева скомпільована мова програмування загального призначення з статичною типізацією, яка підходить для створення найрізноманітніших програм. Сьогодні C++ є однією з найпопулярніших і найпоширеніших мов.

Вона корениться в мові С, яка була розроблена в 1969-1973 від компанії Bell Labs програмістом Деннісом Річі. На початку 1980-х років данський програміст Бйорн Страуструп, який у той час працював на Bell Labs, розробив C++ як розширення мови С. Насправді, на початку, C++ просто доповнювало мову С з деякими об'єктно-орієнтованими можливостями програмування. Ось чому Страуструп сам назвав її “С з класами” в першу чергу [3].

Згодом нова мова почала набирати популярність. Вона додала нові функції, які зробили це не просто доповненням до С, с абсолютно нову мову програмування. В результаті, “С з класами” було перейменовано на С++. І від тих пір, обидві мови почали розвиватися незалежно одна від одної.

С++ – це потужна мова, успадкована від ресурсів С із багатою пам'яттю. Таким чином, часто С++ знаходить своє застосування в системному програмуванні, зокрема, при створенні операційних систем, драйверів, різних утиліт, антивірусів тощо до речі, Windows в основному написана на С++. Але тільки системне програмування не обмежує використання цієї мови. С++ можна використовувати в програмах будь-якого рівня, де важливі швидкість та продуктивність. Часто використовується для створення графічних застосунків, різних прикладних програм. Також особливо часто використовується для створення ігор з насиченою візуалізацію. Крім того, останнім часом мобільний напрям набирає обертів, де С++ також знайшла застосування. І навіть у веб-розробці, ви також можете використовувати С++ для створення веб-застосунків або деяких допоміжних послуг, які обслуговують веб-застосунки. У загальному С++ є мовою широкого використання, де можна створити практично будь-які програми.

С++ – це скомпільована мова, яка означає, що компілятор переводить початковий код на С++ у виконуваний файл, який містить набір інструкцій на комп'ютері. Але різні платформи мають свої власні риси, тому скомпільовані програми можуть не тільки переходити з однієї платформи на іншу і там вже запущено. Однак, на рівні вихідного коду, програми С++ є більш портативними, якщо ви не використовуєте ніяких специфічних функцій для поточної операційної системи. І наявність компілятора, бібліотек та інструментів розробки практично для всіх типових платформ дозволяє компілювати однаковий початковий код у застосунках С++ на цих платформах.

На відміну від цього, C++ дозволяє писати програми в об'єктно-орієнтованому стилі, представивши програму як колекцію взаємодіючих класів і об'єктів. Що спрощує створення великих застосувань.

У 1979-80, Бйорн Страуструп розробив розширення мови С-класу. У 1983 мову було перейменовано на C++.

У 1985 була випущена перша комерційна версія мови C++, а також перше видання книги, “мова програмування C++”, яка представила перший опис мови за відсутності офіційного стандарту.

У 1989 була випущена нова версія мови C++ 2,0, яка включала ряд нових можливостей. Після цього мова розвивається відносно повільно до 2011 року. Але в 1998 була зроблена перша спроба стандартизувати мову організацією ISO (Міжнародна організація стандартизації). Перший стандарт був названий ISO/IEC 14882:1998 або скорочено C++98. Пізніше в 2003 вийшла нова версія стандарту C++03.

У 2011 було опубліковано новий стандарт C++11, в якому міститься багато доповнень і велика кількість нових функцій. Невелике доповнення до стандарту, також відоме як C++14, було випущено в 2014.

Для розробки C++ програм потрібен компілятор – він переводить початковий код з C++ у виконуваний файл, який потім може бути запущено. Але на даний момент є багато різних компіляторів. Вони можуть відрізнятися в різних аспектах, зокрема щодо реалізації стандартів.. Рекомендується для розвитку, щоб вибрати тих компіляторів, які розробляють і реалізують всі останні стандарти.

Мова програмування Java одна з найвідоміших для розробки мобільних застосунків для операційної системи Android.

Мова Java розроблена компанією Sun Microsystems, творцем якої був Джеймс Гослінг, і випущений в 1995 році в якості основних компонентів компанії Sun Microsystems – Java платформ (Java 1.0 [J2SE]).

Станом на 2018 рік останньою версією Java Standard Edition є 8 (J2SE). З розвитком Java, і її широкою популярністю, кілька конфігурацій були

побудовані для різних типів платформ. Наприклад: J2EE – застосунки для підприємств, J2ME – для мобільних застосунків [2].

Sun Microsystems перейменувала колишню версію J2 і ввела нові: Java SE, Java EE і Java ME. Введення в програмування Java різних версій підтверджувало знаменитий слоган компанії «Напиши один раз, запускай скрізь».

Історія створення мови Java починається в червні 1991 року, коли Джеймс Гослінг створив проект для використання в одному зі своїх численних сет-топ проектів. Мова, яка росла поза офісом Гослінга, як дуб, Oak – первинна назва Java до 1995 року, після надалі історія Java тривала під ім'ям Green, а пізніше була перейменована як Java.

Але офіційною датою створення мови Java вважається 23 травня 1995 року, після випуску компанією Sun першої реалізації Java 1.0. Вона гарантувала «Напиши один раз, запускай скрізь», забезпечуючи недорогу вартість на популярних платформах.

13 листопада 2006 року, Sun випустила більшу частину як вільне і відкрите програмне забезпечення відповідно до умов GNU General Public License (GPL).

Після 8 травня 2007 року доля Java склалася інакше. Компанія завершила процес, роблячи все щоб вихідний код був безкоштовним і відкритим, крім невеликої частини коду, на яку компанія не мала авторських прав.

Переваги Java як мови програмування:

1. Об'єктно-орієнтована: в Java все є об'єктом. Доповнення може бути легко розширено, тому що вона заснована на об'єктній моделі.
2. Платформонезалежність: на відміну від багатьох інших мов, включаючи C і C ++, Java, коли була створена, вона не компілювалась в платформі конкретної машини, а в незалежній від платформи байт-коді. Цей байт код поширюється через інтернет і інтерпретується в Java Virtual Machine (JVM), на якій він в даний час працює.

3. Проста: процеси вивчення та введення в мову програмування Java залишаються простими. Якщо Ви розумієте основні концепції об'єктно-орієнтованого програмування, то вона буде проста в освоєнні.
4. Безпечна: методи перевірки автентичності засновані на шифруванні з відкритим ключем.
5. Архітектурно-нейтральна: компілятор генерує архітектурно-нейтральні об'єкти формату файлу, що робить скомпільований код виконуваним на багатьох процесорах, з наявністю системи Java Runtime.
6. Портативна: архітектурно-нейтральна і не має залежності від реалізації аспектів специфікацій – все це робить Java портативною. Компілятор в Java написаний на ANSI C з чистою переносимістю, який є підмножиною POSIX.
7. Міцна: виконує зусилля, щоб усунути помилки в різних ситуаціях, спираючись в основному на час компіляції, перевірку помилок і перевірку під час виконання.
8. Багатопотокова: функції багатопоточності, можна писати програми, які можуть виконувати безліч завдань одночасно. Введення в мову Java цієї конструктивної особливості дозволяє розробникам створювати налагоджені інтерактивні застосунки.
9. Інтерпретована: Java байт-код переводиться на льоту в машинні інструкції та ніде не зберігається. Роблячи процес більш швидким і аналітичним, оскільки зв'язування відбувається як додаткове з невеликою вагою процесу.
10. Високопродуктивна: введення Just-In-Time компілятора, дозволило отримати високу продуктивність.
11. Поширена: призначена для розподіленого середовища інтернету.
12. Динамічна: програмування на Java вважається більш динамічним, ніж на C або C ++, так як вона призначена для адаптації до мінливих

умов. Програми можуть виконувати велику кількість під час обробки інформації, яка може бути використана для перевірки і дозволу доступу до об'єктів на час виконання.

Але для свого проекту я обрала саме мову Kotlin, через ряд переваг які детальніше розглянемо.

Kotlin – статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту.

Автори ставили перед собою мету створити лаконічнішу та безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE.

Мова розробляється з 2010 року, публічно представлена в липні 2011. Сирцевий код було відкрито в лютому 2012. В лютому було випущено milestone 1, який містив плагін для IDEA. У червні – milestone 2 з підтримкою Android. У грудні 2012 року вийшов milestone 4 та забезпечив підтримку Java 7. Станом на листопад 2015 року основні можливості мови стабілізовані, готується реліз версії 1.0. В грудні 2015 року з'явився реліз-кандидат версії 1.0, а 15 лютого 2016 року відбувся реліз версії 1.0.

З 17 травня 2017 року входить в список офіційно підтримуваних мов для розробки застосунків для платформи Android.

З 7 травня 2019 року є рекомендованою мовою програмування для розробки Android застосунків.

Kotlin надає набір інструментів для розробки під Android, окрім саме стандартних можливостей мови:

1. Kotlin Android Extensions є розширеннями компілятора, що дозволяє вам відмовитись від викликів findViewById() в вашому коді, та замінити їх на згенеровані компілятором властивості.

2. Anko є бібліотекою, що провадить дружні до Kotlin обертки навколо Android API, так само як DSL, що дозволяє вам замінити ваш файл розташування .xml на код Kotlin.
3. Kotlin чудово пасує до Android застосувань, привносячи всі переваги сучасної мови на Android платформу, без введення нових обмежень.
4. Сумісність: Kotlin повністю сумісний з JDK 6, гарантуючи, що Kotlin застосування можуть робити на старіших пристроях Android без жодних проблем. Інструментарій Kotlin повністю підтримується в Android Studio, та сумісний з системою побудови Android.
5. Продуктивність: Застосування Kotlin роблять так само швидко, як еквівалентні Java застосування, дякуючи дуже подібному байткоду. З підтримкою в Kotlin інлайн функцій, код з використанням лямбда часто виконується навіть швидше, ніж той самий код на Java.
6. Взаємодія: Kotlin є на 100% взаємодіючим з Java, що дозволяє використовувати всі існуючі бібліотеки Android в застосуванні Kotlin. Це включає процес анотацій, так що прикріплення даних та Dagger також працюють.
7. Місце: Kotlin має дуже компактну бібліотеку часу виконання, що може бути ще зменшеним завдяки використанню ProGuard. В реальному застосуванні, рантайм Kotlin додає тільки декілька сотен методів, та менше ніж 100Кб до розміру .apk файлу.
8. Час компіляції: Kotlin підтримує ефективну інкрементальну компіляцію, так що хоча для чистих побудов існує деяке додаткове навантаження, інкрементальні побудови звичайно так сами швидкі, або ще кращі, ніж Java.
9. Крива навчання: для розробника Java почати роботу з Kotlin дуже просто. Автоматичний конвертор з Java до Kotlin, включене в плагін Kotlin допомагає зробити перші кроки. Kotlin Koans надає проходження по ключовим можливостям нової мови через серію інтерактивних навчань [5], [6].

2.2 Мова розмітки XML

XML – це мова розмітки, створена Консорціумом World Wide Web Consortium (W3C) для визначення синтаксису для кодування документів, які можуть читати і люди, і машини. Це робиться за допомогою тегів, які визначають структуру документа, а також спосіб зберігання та транспортування документа.

XML не має попередньо визначеної мови розмітки, як, наприклад, HTML. Замість цього, XML дозволяє користувачам створювати свої власні символи розмітки для опису контенту, роблячи безлімітний і самовизначувальний набір символів.

По суті, HTML є мовою, яка фокусується на презентації контенту, а XML – це мова опису даних, яка використовується для зберігання даних.

XML часто використовується як основа для інших форматів документів – їх дуже багато. Ось декілька, які можна визнати[8]:

- Обидва RSS і ATOM описують, як програми читача обробляють веб-канали.
- Microsoft .NET використовує XML для своїх конфігураційних файлів.
- Microsoft Office 2007 і пізніше використовують XML як основу для структури документів. Ось що означає, наприклад, “X” у форматі документа .DOCX Word, а також використовується в файлах Excel (XLSX) і PowerPoint (файли PPTX).

Коректний документ (англ. well-formed document) відповідає всім синтаксичним правилам XML. Документ, що не є коректним, не може називатись XML-документом. Сумісний синтаксичний аналізатор (англ. Conforming parser) не повинен обробляти такі документи.

Зокрема, коректний XML-документ має:

- Лише один елемент у корені.

– Непорожні елементи розмічено початковим та кінцевим тегами (наприклад, <пункт>Пункт1</пункт>). Порожні елементи можуть позначатися «закритим» тегом, наприклад <IAmEmpty/>. Така пара еквівалентна <IAmEmpty></IAmEmpty>.
– Один елемент не може мати декілька атрибутів з однаковою назвою. Значення атрибутів перебувають або в одинарних ('), або у подвійних (“) лапках.
– Теги можуть бути вкладені, але не можуть перекриватись. Кожен некореневий елемент мусить повністю перебувати в іншому елементі.
– Документ має складатися тільки з правильно закодованих дозволених символів Юнікоду. Єдиними кодуваннями, які обов'язково має розуміти XML-процесор, є UTF-16 та UTF-8. Фактичне та задекларованe кодування (англ. character encoding) документа мають збігатись. Кодування може бути задекларовано ззовні, як у заголовку «Content-Type» при передачі за протоколом HTTP, або в самому документі використанням явної розмітки на самому початку документа. У разі відсутності інформації про кодування документ має бути в кодуванні UTF-8 (або його підмножині ASCII).

Фізична структура

1. Сутності (англ. Entity). Головною сутністю є зміст документа. Інші можливі сутності вказуються за допомогою
2. Посилання на сутності (&назва; в самому документі, та, наприклад %назва; у визначені його типу) можуть слугувати в ролі позначень спеціальних символів, посилань на спеціальні символи (вказуючи коди символів &#десяткове) або окремих документів чи фрагментів тексту.
3. XML-декларація, в ній вказується версія XML, кодування та інша допоміжна інформація.

4. Декларація типу документа може застосовуватись для того, аби додавати нові типи сутностей та визначати логічну структуру документа.
5. Логічна структура
6. XML-документ має ієрархічну логічну структуру, і може представлятись у вигляді дерева. Вузлами цього дерева можуть бути:
7. елементи, фізична структура яких складається із:
   a) коректної пари відкриваючого та закриваючого тегів (<Назва тегу>) та (</Назва тегу>), або тегу порожнього елемента (<Назва тегу/>);
   b) атрибути, що мають вигляд пар ключ-значення (назва атрибута=“значення атрибута”) і знаходяться або у відкриваючому, або у порожньому тегу (подібно до метаданих);
   c) вказівки щодо обробки документа (англ. Processing Instruction) (<?Обробник параметр ?>;
   d) коментарі (<!—Текст коментаря>);
   e) текст, або у вигляді простого тексту, або фрагментів CDATA (<![CDATA[довільний текст]]>).

XML-документ повинен мати лише один кореневий елемент. Решта елементів є піделементами цього кореневого елемента.

Деякі веб-браузери здатні безпосередньо відображати XML-документи. Це може досягатись шляхом застосування таблиці стилів (англ. Stylesheet). Вказані у таблиці стилів операції можуть призводити до перетворення XML-документа в інший, відмінний від XML формат.

2.3 Вибір середовища розробки

При розробці і реалізації програмного продукту було використано середовище Android Studio

Android Studio – інтегроване середовище розробки (IDE) для платформи Android.

Ця програма пропонує повний інструмент для розробки та налагодження програм для мобільної операційної системи Google.

За допомогою нього можна виконувати редагування коду, налагодження, використовувати інструменти продуктивності, він має гнучку систему компіляції та миттєве створення та розгортання, що дозволяє зосередитись на створенні застосунків [9].

Android-студіо включає шаблони проектів та коду, які спрощують додавання усталених шаблонів такі як бічна панель навігації та перегляд сторінки.

Серед основних його характеристик можна виділити:

- функції інтеграції та підписання програм ProGuard;
- візуалізація в режимі реального часу;
- консоль розробника: поради щодо оптимізації, довідка щодо перекладу, статистика використання;
- підтримка збірки на основі Gradle;
- спеціальний рефакторинг для Android та швидкі виправлення;
- розширений редактор макетів, що дозволяє користувачам перетягувати компоненти інтерфейсу користувача;
- інструменти Lint для виявлення продуктивності, зручності використання, сумісності версій та інших проблем;
- шаблони для створення загальних макетів Android та інших компонентів;
- підтримка програм для програмування для Android Wear;
- вбудована підтримка Google Cloud Platform, яка забезпечує інтеграцію з Google Cloud Messaging та App Engine;
- віртуальний пристрій Android, що використовується для запуску та тестування застосунків[11].

2.4 Дизайн-принципи та підходи

Дизайн-проект створювався у редакторі Figma.

Figma – це онлайн сервіс розробки інтерфейсів. Працює у двох форматах: у вікні браузера, та як клієнтський застосунок на пристрої користувача. Цей редактор підходить як для розробки простих прототипів, так і для створення складних проектів, наприклад мобільні застосунки та портали.

Користувачі – головний об’єкт уваги. Потрібно визначити що їм потрібно, використовувати це в дизайні, а потім сформувати продукт так, щоб користувачі могли виконати дію якнайшвидше. Flat інтерфейс заснований на правилах типографіки та сітки і направлений на молодшу аудиторію, а більш дружелюбні скевоморфічні елементи підійдуть віковим користувачам, які не настільки просунуті.

Оточення – це не тільки платформа для якої робиться дизайн, але також і умови використання, та місце в якому він буде працювати. Наприклад мобільний телефон суттєво відрізняється від телевізора, – на нього дивляться здалеку, він завжди використовується в приміщенні і керується на віддаленні. Це передбачає інші підходи до розміру тексту, кольору та контрастності інтерфейсу.

Можливості – здатність об’єкта висловлювати свої функції через органи чуття: кнопка своєю опуклістю натякає на її клікабельність, дверна ручка, завдяки формі та розташуванню натякає на те, що її можна потягнути. Все це може бути використано в цифровому дизайні, щоб спонукати користувачів взаємодіяти з об’єктами. Випуклі кнопки, текст який виходить за межі сторінки – це демонстрація можливостей.

Текст. Хороший текст не тільки робить застосунок зрозумілішим, але й створює зв’язок з користувачем, завдяки використовуваному тону. Коли ми говоримо з людьми рідною мовою, а не як з машиною, створюється емоційний зв’язок, який покращує досвід взаємодії з продуктом.

Типографіка. В цифровому дизайні типографіку часто незаслуговано обділяють увагою. Дуже важливо використовувати хорошу типографіку, адже більшість інформації передається у вигляді тексту. Існують базові фактори на які потрібно звертати увагу.

Анімації. З постійним збільшенням потужності комп'ютерів, ми можемо робити не тільки гарний дизайн, але й додавати в нього елегантну динаміку. Легка анімація може задати характер інтерфейсу, поліпшити його інтуїтивність, виділити функції та можливості. Наприклад, якщо ви хочете привернути увагу до кнопки – примусьте її пульсувати.

Тестування. Ніщо не порівняти з тестом дизайну на призначеному пристрої. Роздільна здатність, технологія екрану, кути перегляду та методи введення можуть істотно відрізнятись від вашого комп'ютера. У наш час існує безліч засобів для попереднього перегляду, які перенесуть дизайн з комп'ютера прямо у ваш девайс, оновлюючись у реальному часі.

Прототипи. Тестування дуже тісно пов'язане з прототипуванням. Прототипи дозволяють нам швидко випробовувати ідеї, не витрачаючи часу на фінальне «полірування». Починаючи з ескізів на папері, закінчуючи створенням простих застосунків в рідному середовищі розробки. Все залежить від того що саме потрібно: ескізи можна зробити і на папері, а от плавну анімацію доведеться показати на відео[12].

При розробці додатку використовувались такі компоненти:

– ConstraintLayout для створення адаптивного користувацького інтерфейсу з мінімальною кількістю вкладених «layout»;
– BottomNavigation – для створення меню на головній сторінці мобільного додатку;
– ObjectAnimator – для створення анімації в макеті.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Проектування інформаційної системи за допомогою методології функціонального моделювання SADT

Методологія SADT – це сукупність методів, процедур, правил, які призначені для побудови функціональної моделі об'єкту якої-небудь предметної області. Функціональна модель SADT відображає функціональну структуру об'єкту, тобто дії, які над ним виконуються та зв'язки між цими діями. Основні елементи цієї методології ґрунтуються на таких концепціях:

1. Графічне представлення блокового моделювання. Графіка блоків і дуг SADT-діаграми відображає функцію у вигляді блоку, а інтерфейси входу/виходу представляються дугами, що, відповідно, входять у блок і виходять з нього. Взаємодія блоків між собою описується за допомогою інтерфейсних дуг, що виражають “обмеження”, котрі у свою чергу визначають, коли і яким чином функції виконуються й керуються;
2. Строгість і точність. Виконання правил SADT вимагає достатньої строгості й точності, не накладаючи в той же час надмірних обмежень на дії аналітика.

Правила SADT включають:

– обмеження кількості блоків на кожному рівні декомпозиції (як правило, 3-6 блоків);
– унікальність міток і найменувань (відсутність імен, що повторювалися б);
– синтаксичні правила для графіки (блоків і дуг);
– поділ входів і керувань (правило визначення ролі даних).
– відділення організації від функції, тобто виключення впливу організаційної структури на функціональну модель.

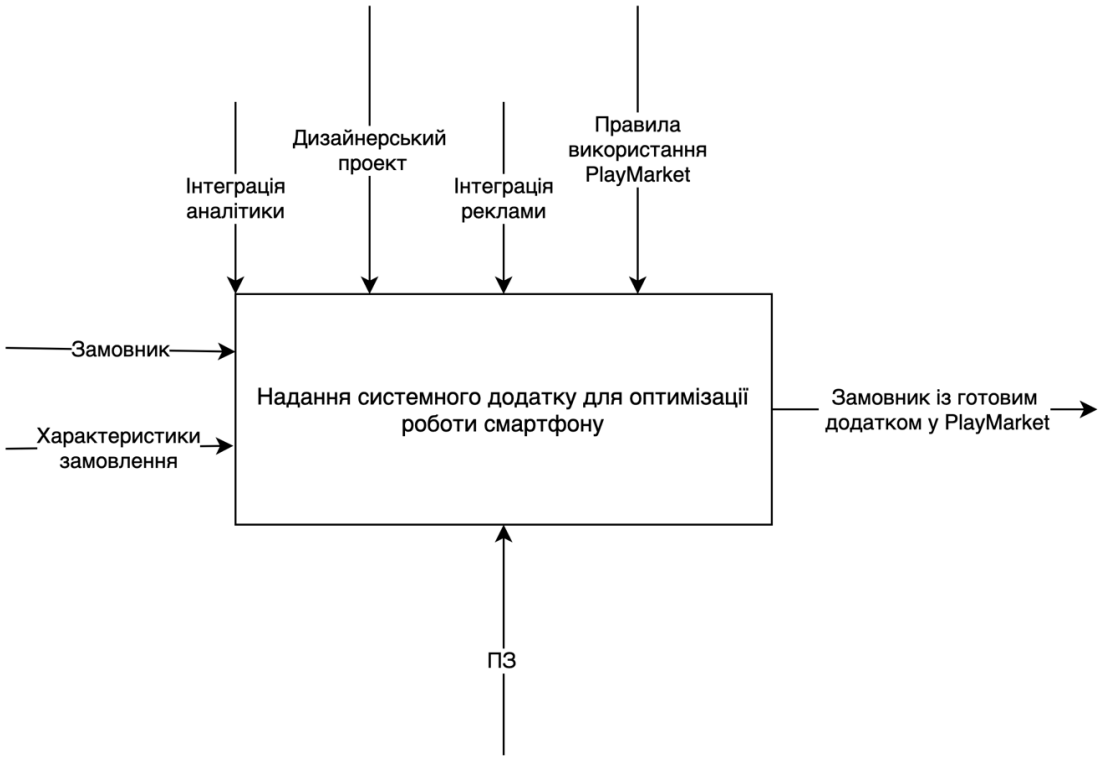

Рисунок 3.1 – Контекстна діаграма

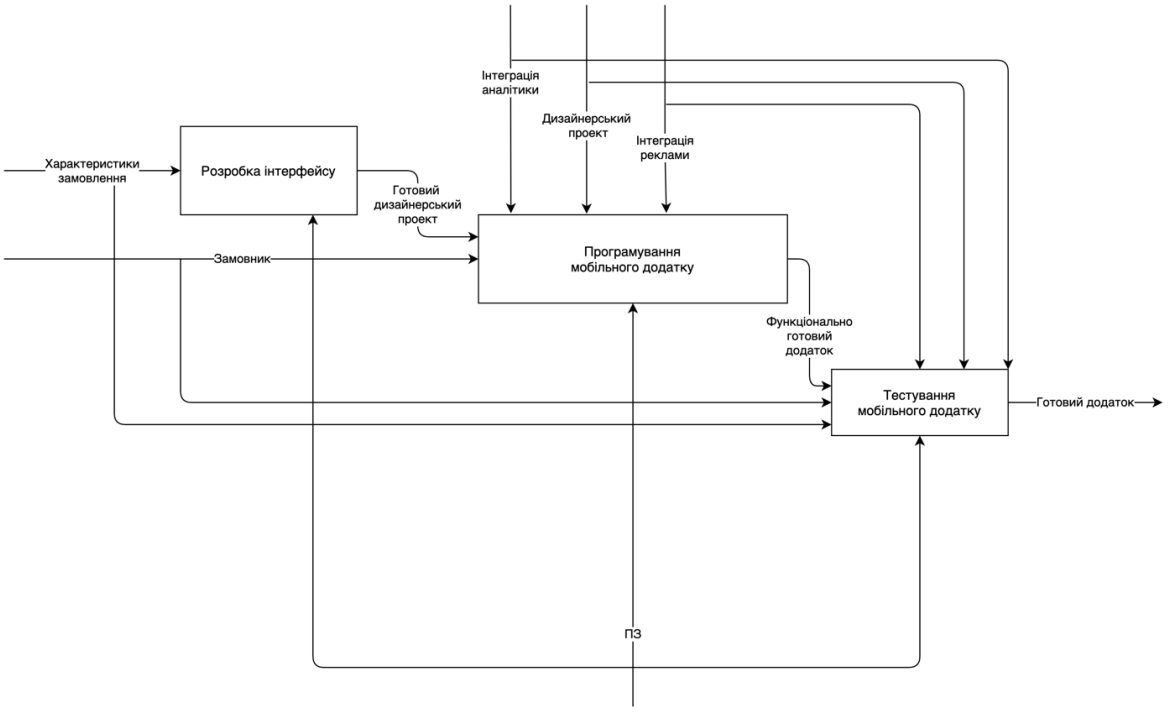

Рисунок 3.2 – Деталізована діаграма 1 рівня.

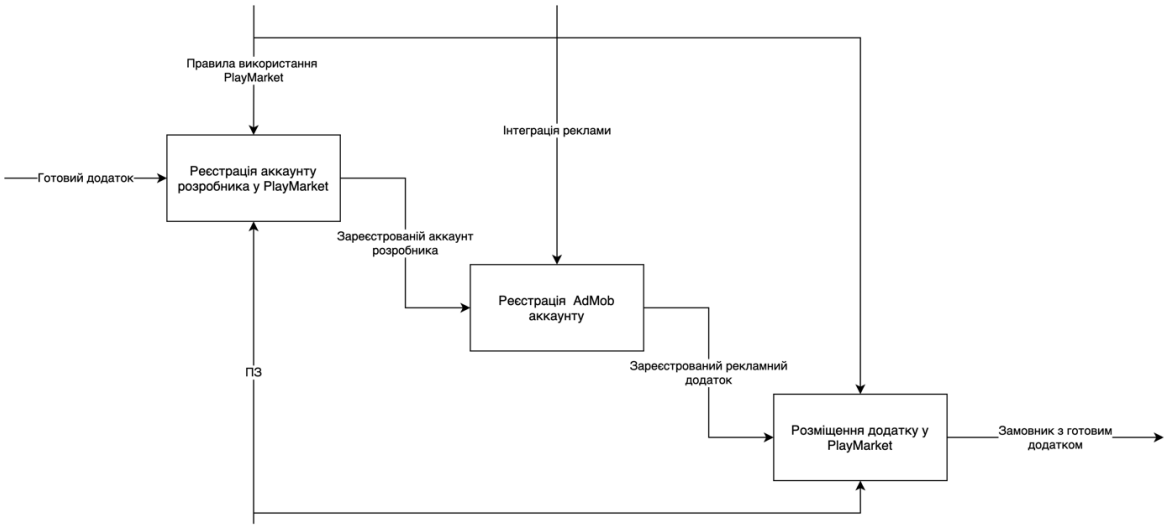

Рисунок 3.3 – Деталізована діаграма 2 рівня.

3.2 Проектування інформаційної системи за допомогою методології Workflow Diagramming

Діаграми методології IDEF3 (Workflow Diagramming) – методологія моделювання і стандарт документування процесів, що відбуваються в системі. Метод документування технологічних процесів є механізм документування та збору інформації про процеси. IDEF3 показує причинно-наслідкові зв'язки між ситуаціями і подіями в зрозумілій експерту формі, використовуючи структурний метод вираження знань про те, як функціонує система, процес або підприємство. Система описується як упорядкована послідовність подій з одночасним описом об'єктів, що мають відношення до моделюючого процесу.

IDEF3 складається з двох методів: Process Flow Description (PFD) - опис технологічних процесів, із зазначенням того, що відбувається на кожному етапі технологічного процесу. Object State Transition Description (OSTD) - опис переходів станів об'єктів, із зазначенням того, які існують проміжні стани у об'єктів в моделюється системі.

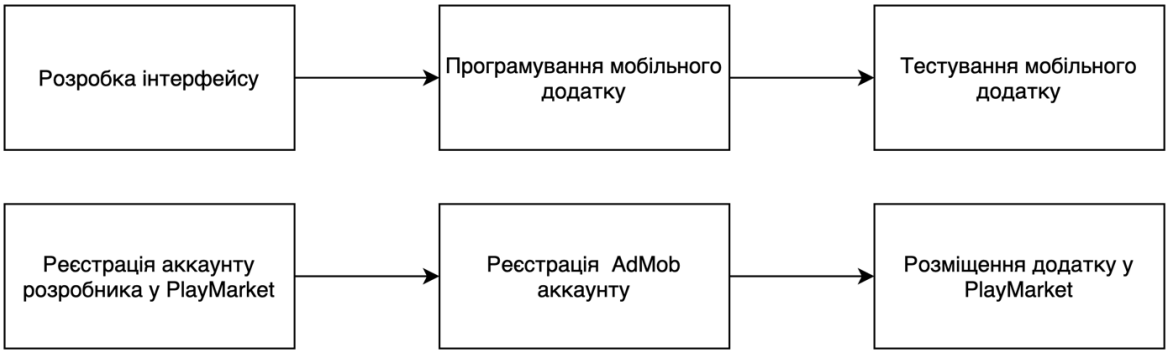

Рисунок 3.4 – Контекстна діаграма

3.3 Створення дизайн-проекту системи

Дизайн-проект створювався у редакторі Figma.

Figma – це хмарний багатоплатформовий сервіс для дизайнерів інтерфейсів і web-розробників, з яким можна працювати безпосередньо в браузері. І це лише одне з важливих переваг платформи.

Було обрано саме цей редактор, через ряд переваг, наприклад, хмарне зберігання файлів, можливість спільного доступу до проекту та роботи над ним.

За допомогою Figma можна розробляти:

- прототипи майбутніх продуктів;
- дизайн сайтів для ПК, моб пристроїв, планшетів;
- логотипи, іконки;
- векторні зображення.

З точки зору функціоналу це зручний графічний редактор, в якому можна створювати:

- прототипи web-сайтів і застосунків;
- окремі елементи інтерфейсу: іконки, кнопки, форми і багато іншого;
- векторні зображення та ілюстрації, інше.

Створення сторінки «Charge Booster»:

– створення дизайну сторінки;
– створення системи збору та аналізу даних про використання ОЗП пам’яті;
– створення системи очищення ОЗП пам’яті.

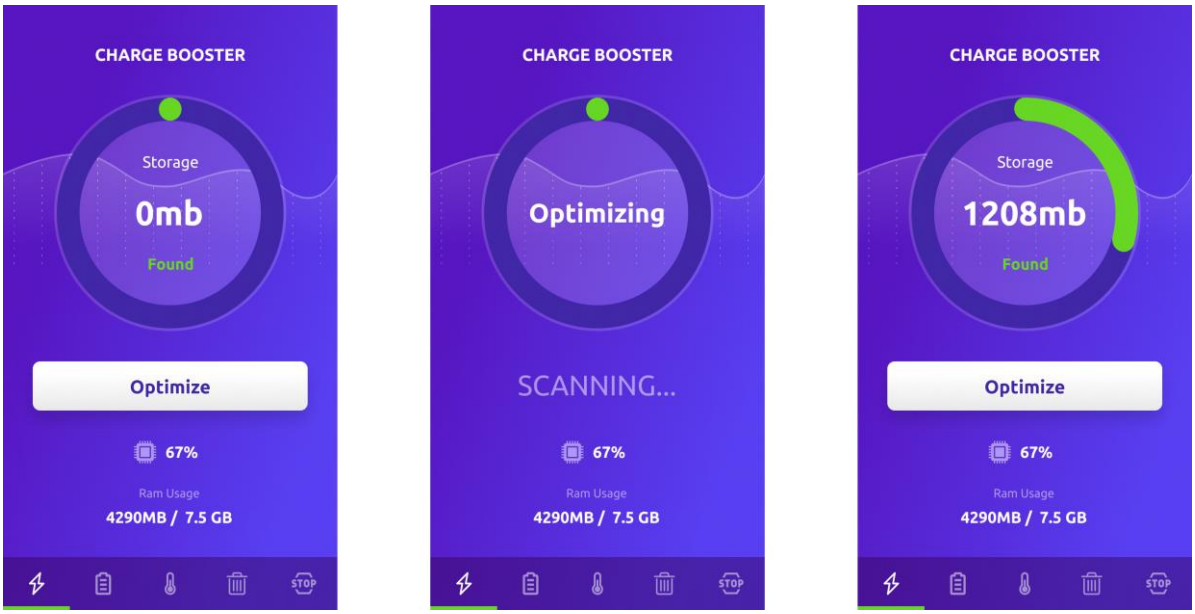

Рисунок 3.5 – Дизайн сторінки «Charge Booster»

Створення сторінки «Battery Saver»:

– створення дизайну сторінки;
– створення сторінки з можливістю вибору та застосування режимів енергозбереження “Normal mode”, “Ultra power saving”, “Extreme power saving”;
– створення системи тимчасового блокування застосунків на період застосування режиму енергозбереження.

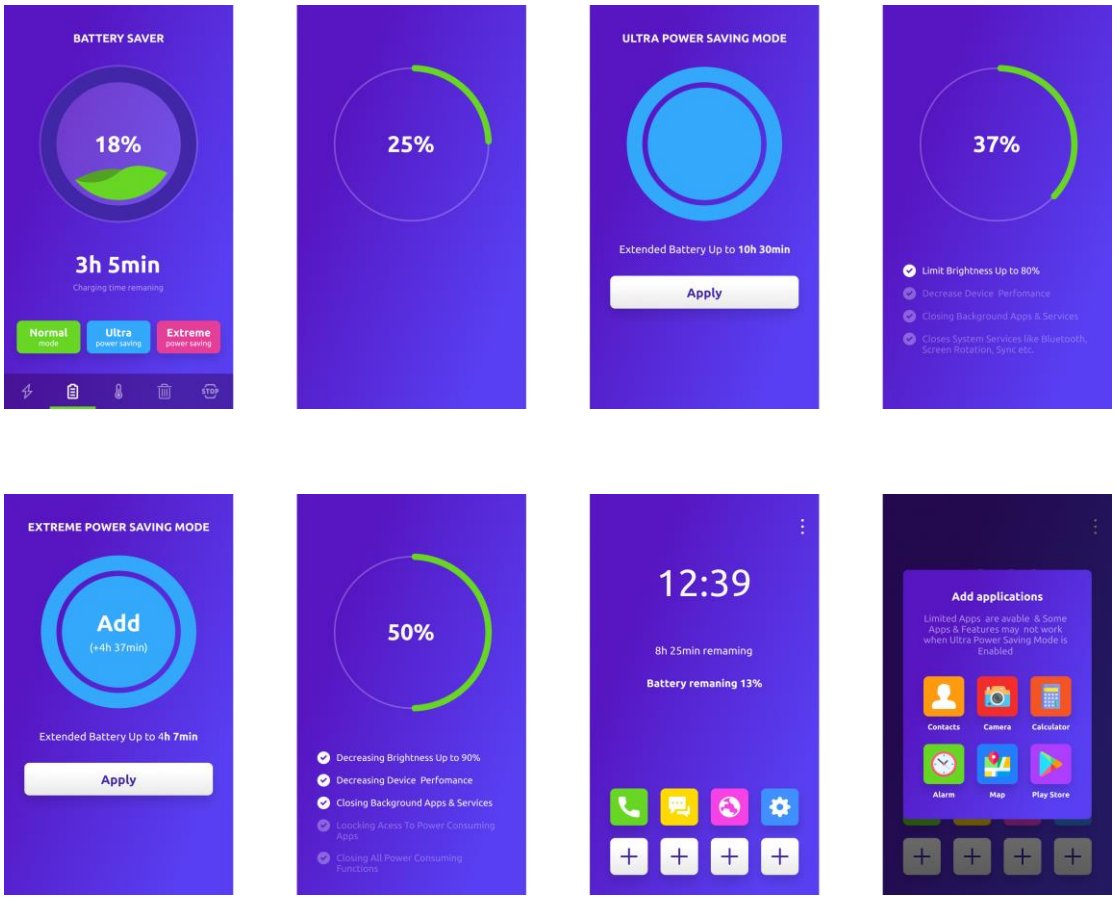

Рисунок 3.6 – Дизайн сторінки «Battery Saver»

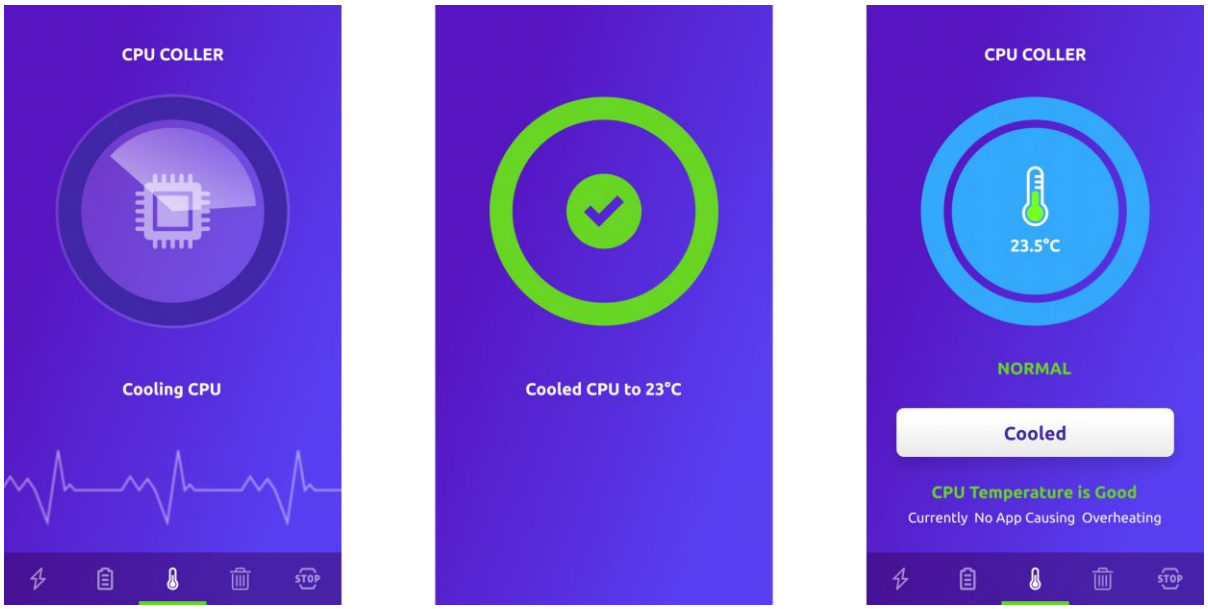

Рисунок 3.7 – Дизайн сторінки «CPU Coller»

Створення сторінки «CPU Coller»:

– створення дизайну сторінки;
– створення системи виявлення даної та оптимальної температури процесору та охолодження процесору.

Створення сторінки «Junk Cleaner»:

– створення дизайну сторінки;
– створення системи для знаходження та очищення кеш-пам’яті, видалення “сміттєвих”, небажаних та залишкових файлів.

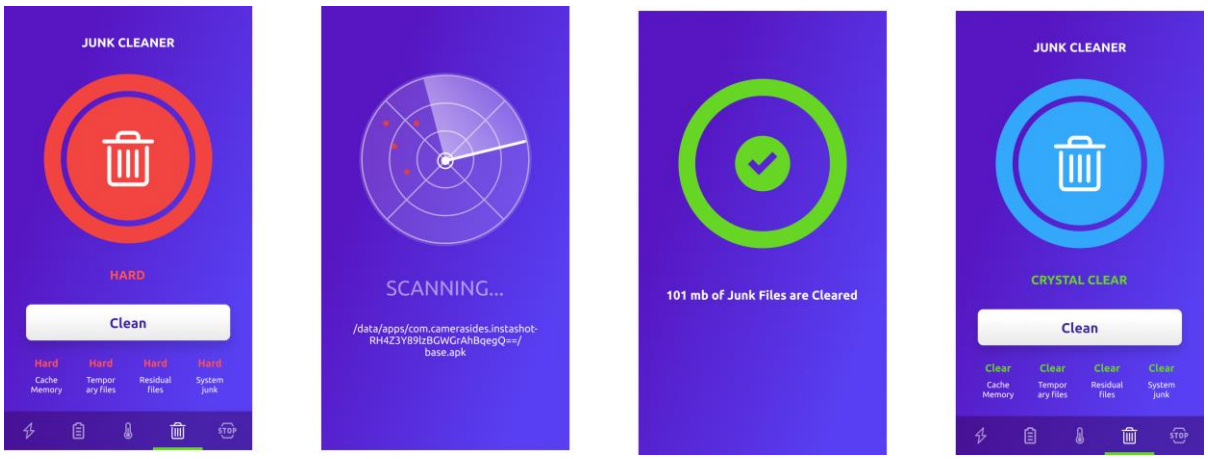

Рисунок 3.8 – Дизайн сторінки «Junk Cleaner»

3.4 Реалізація функціональних можливостей

Через сміття, яке накопичується по ходу роботи телефону, особливо якщо користувач веде активне мережеве життя. У смартфоні зберігаються файли, що залишилися від віддалених програм, а також різні версії софта, некоректні поновлення і, звичайно, все це займає пам’ять.

Для очищення смартфона звичайному користувачеві потрібно, щоб програма видалила сміття, а от як це виглядає з точки зору розробника розглянемо далі.

На рис. 3.9 описаний процес пошуку застосунків, які займають найбільщу кількість місця на смартфоні і проводять очистку.

```
if (packageName != "pro.cleaner.app.boost") {

    //              String size = packages.getInstance(k).metaData.size()+"";
    //              Log.e("Size-->", "" + packageName);
    var ico: Drawable? = null
    try {
        val pName = pm.getApplicationLabel(pm.getApplicationInfo(packageName, PackageManager.GET_META_DATA)) as String

        //                  app.setSize("" + pName);

        val file = File(pm.getApplicationInfo(packageName, PackageManager.GET_META_DATA).publicSourceDir)
        val size = file.length()

        Log.e( tag: "SIZE",  msg: (size / 1000000).toString() + "")

        val a = pm.getApplicationInfo(packageName, PackageManager.GET_META_DATA)
        ico = activity!!.packageManager.getApplicationIcon(packages[k].packageName)
        val app = ApplicationsClass( size: (size / 1000000 + 20).toString() + "MB", ico)
        activity!!.packageManager
        Log.e( tag: "ico-->",  msg: "" + ico!!)

        if (a.flags and ApplicationInfo.FLAG_SYSTEM == 0) {
            //                      System.out.println(">>>>>>packages is system package"+pi.packageName);

            if (check <= 5) {
                check++
                apps.add(app)
            } else {
                activity!!.unregisterReceiver(batteryReceiver)
                //                          batterytemp.setText("25.3" + "°C");
                break
            }

        }
        mAdapter.notifyDataSetChanged()


    } catch (e: PackageManager.NameNotFoundException) {
```

Рисунок 3.9 – Пошук застосунків

Заповнена пам'ять може гальмувати роботу пристрою. Андроїд попереджає, коли місце в смартфоні закінчується, і краще не ігнорувати це повідомлення. Проведіть ревізію в вашому пристрої - видаліть непотрібні фото, відео, музику, файли великого розміру. Якщо вона вам потрібні - краще перенести їх на знімні карти пам'яті або в хмарне сховище.

Якщо ви довго не використовуєте програми, система вам про це нагадає. Можливо, додаток вам вже не потрібно, але місце воно займає. Такі програми теж потрібно видаляти.

```
val totalRAM: String
    get() {

        var reader: RandomAccessFile? = null
        var load: String? = null
        val twoDecimalForm = DecimalFormat( pattern: "#.##")
        var totRam = 0.0
        var lastValue = ""
        try {
            try {
                reader = RandomAccessFile( name: "/proc/meminfo",  mode: "r")
                load = reader.readLine()
            } catch (e: Exception) {

            }

            val p = Pattern.compile( regex: "(\\d+)")
            val m = p.matcher(load)
            var value = ""
            while (m.find()) {
                value = m.group( group: 1)
            }
            try {
                reader!!.close()
            } catch (e: Exception) {

            }

            totRam = java.lang.Double.parseDouble(value)

            val mb = totRam / 1024.0
            val gb = totRam / 1048576.0
            val tb = totRam / 1073741824.0

            if (tb > 1) {
                lastValue = twoDecimalForm.format(tb) + " TB"
            } else if (gb > 1) {
                lastValue = twoDecimalForm.format(gb) + " GB"
            } else if (mb > 1) {
```

Рисунок 3.10 – Розрахунок загальної кількості ОЗП

На рис. 3.10 та 3.11 показаний фрагмент коду на якому проводиться розрахунок загальної кількості оперативної пам’яті.

Усі смартфони мають певний обсяг оперативної пам'яті. Чим її обсяг менше, тим швидше вона заповнюється і повільніше працює.

```
            var value = ""
            while (m.find()) {
                value = m.group( group: 1)
            }
            try {
                reader!!.close()
            } catch (e: Exception) {

            }

            totRam = java.lang.Double.parseDouble(value)

            val mb = totRam / 1024.0
            val gb = totRam / 1048576.0
            val tb = totRam / 1073741824.0

            if (tb > 1) {
                lastValue = twoDecimalForm.format(tb) + " TB"
            } else if (gb > 1) {
                lastValue = twoDecimalForm.format(gb) + " GB"
            } else if (mb > 1) {
                lastValue = twoDecimalForm.format(mb) + " MB"
            } else {
                lastValue = twoDecimalForm.format(totRam) + " KB"
            }
        } catch (e: Exception) {
            e.printStackTrace()
        } finally {
        }

        return lastValue
    }
```

Рисунок 3.11 - Розрахунок загальної кількості ОЗП

На рис 3.12 показано яка частина пам’яті зайнята шляхом віднімання вільної пам’яті від загальної кількості.

```
val usedMemorySize: Long
    get() {

        try {
            val mi = ActivityManager.MemoryInfo()
            val activityManager = activity!!.getSystemService(ACTIVITY_SERVICE) as ActivityManager
            activityManager.getMemoryInfo(mi)

            return (mi.totalMem - mi.availMem) / 0x100000L
        } catch (e: Exception) {
            return 200
        }

    }
```

Рисунок 3.12 – Зайнята пам’ять

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

В цьому розділі розглянемо як саме працює вже готовий застосунок та протестуємо його.

4.1 Опис роботи програми

Після запуску застосунку на екрані з’являється вікно (рис. 4.1), у якому пропонується оптимізувати роботу телефона, а також можна побачити яка частина пам’яті використовується.

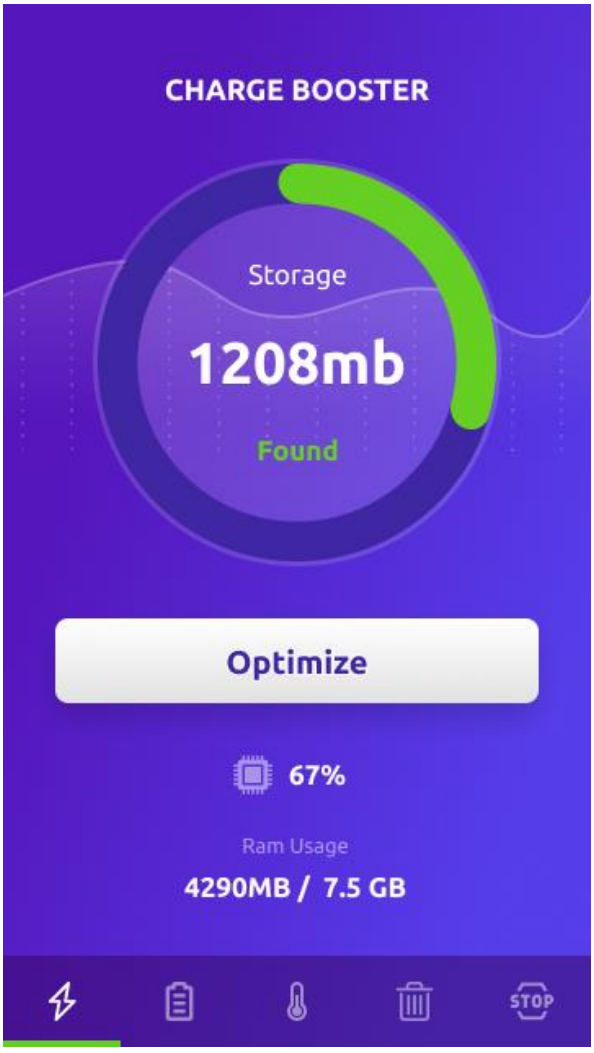

Рисунок 4.1 – Вікно оптимізації

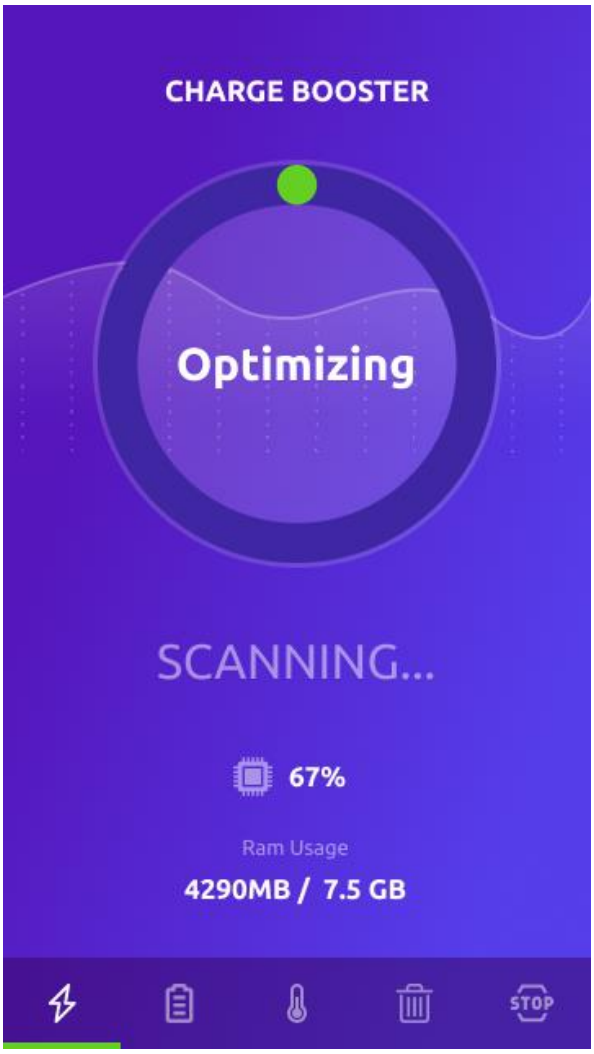

Рисунок 4.2 – Процес сканування та оптимізації

Натиснувши «Optimize» можна оптимізувати роботу смартфону (рис. 4.2). Меню знаходиться в нижній частині вікна.

Неважливо, скільки ядер містить процесор мобільного пристрою, і який обсяг оперативної пам'яті є в гаджеті, завжди потрібно бути готовим до того моменту, коли його швидкодія перестане бути оптимальною. Багато користувачів скаржаться на те, що смартфони та планшети, випущені кілька років тому, припиняють відповідати вимогам сучасних ОС і додатків.

Рисунок 4.3 – Вікно з можливістю вибору економії заряду батареї

При натисканні на другий пункт у меню «Battery saver» (рис. 4.3) надається інформація про заряд батареї, можливий час використання смартфону та можливість вибору режиму енергоекономії.

Скільки часу можна виграти за рахунок режиму економії енергії? Складно сказати. Все залежить, по-перше, від того, як користуватися пристроєм (наприклад, постійно, не вимикаючи екран, або ж рідко, включаючи екран раз на годину на 5 хвилин), а по-друге, від обмежень, які використовуються.

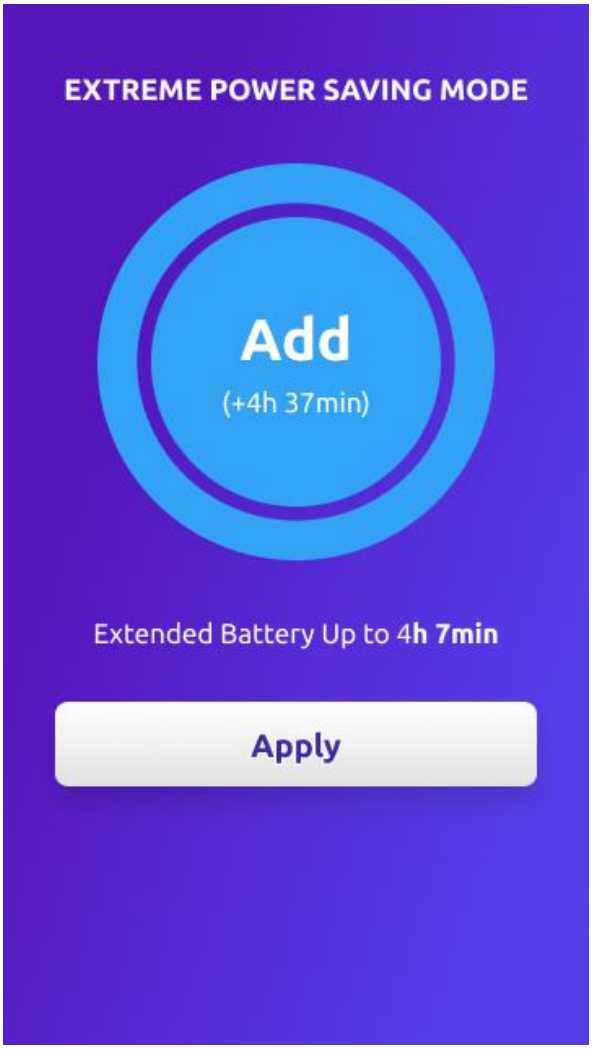

Рисунок 4.4 – Вікно застосування режиму енергозбереження

При натисканні «Extreme power saving» відкривається вікно (рис. 4.4), в якому можна застосувати режим енергозбереження за рахунок пониження яскравості екрану, зниження продуктивності пристрою та закриття всіх запущених застосунків.

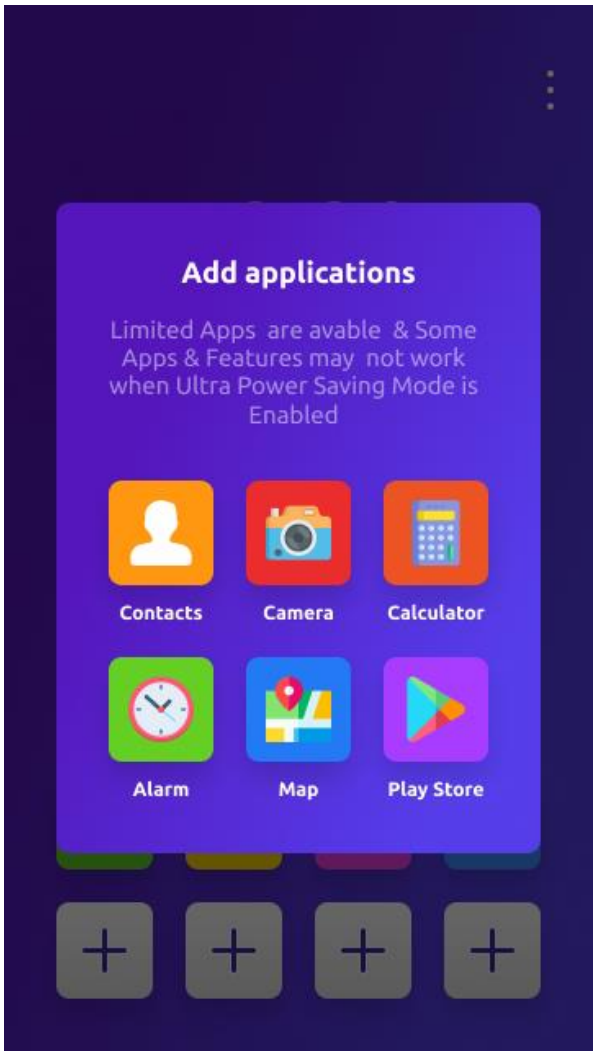

Рисунок 4.5 – Спливаюче вікно з вибором застосунків, які тимчасово будуть недоступні

На рисунку 4.5 показано, що можна обрати застосунки та деякі функції застосунків, які не будуть працювати при увімкненому режимі «Extreme power saving mode».

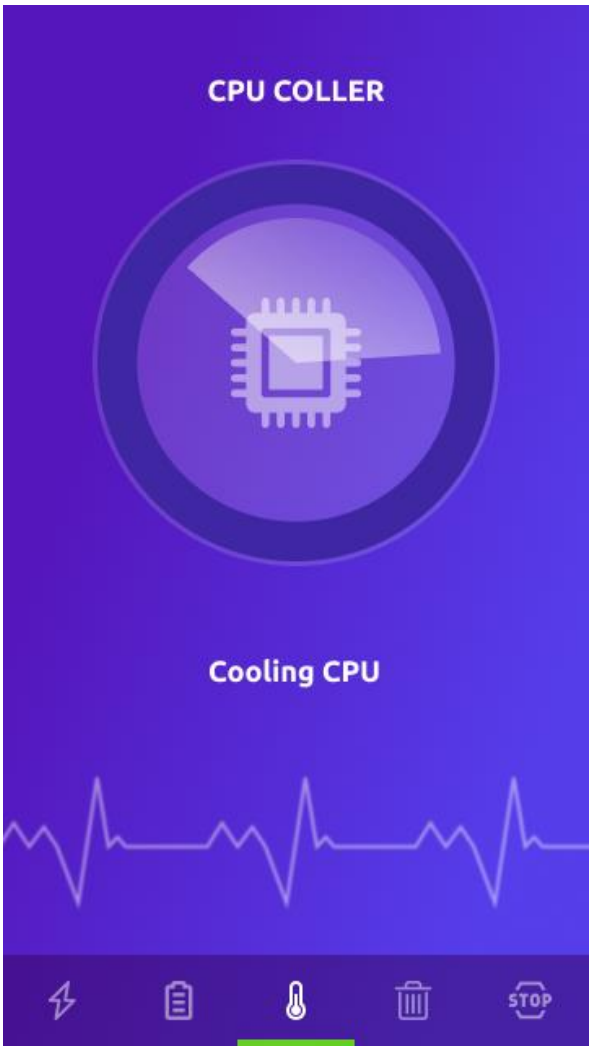

Рисунок 4.6 – Охолодження процесору

Натиснувши «CPU Coller» у третьому пункті меню (рис. 4.6) можна знизити температуру процесору смартфону, а також виводяться застосунки, які підвищують температуру та викликають перегрівання.

Багатозадачність девайса дозволяє виконувати кілька завдань одночасно. Використання інтернету, ігор, відтворення відео навантажують процесор, який нагрівається під час інтенсивної роботи.

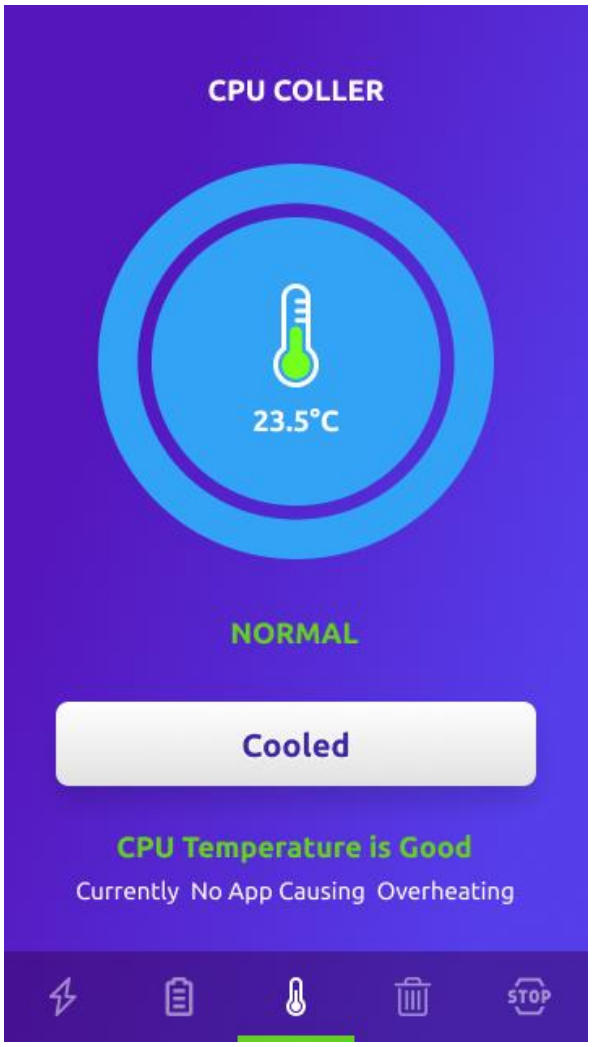

Рисунок 4.7 – Температуру знижено

Рисунок 4.7 показує, що температуру процесору смартфону знижено на 9%, а повідомлення внизу екрану дозволяє дізнатися, що наразі застосунків, які перегрівають процесор, немає.

Причини перегріву процесору: порушення в роботі додатків, системні помилки, неякісний зарядний пристрій або зарядний пристрій з великою силою струму.

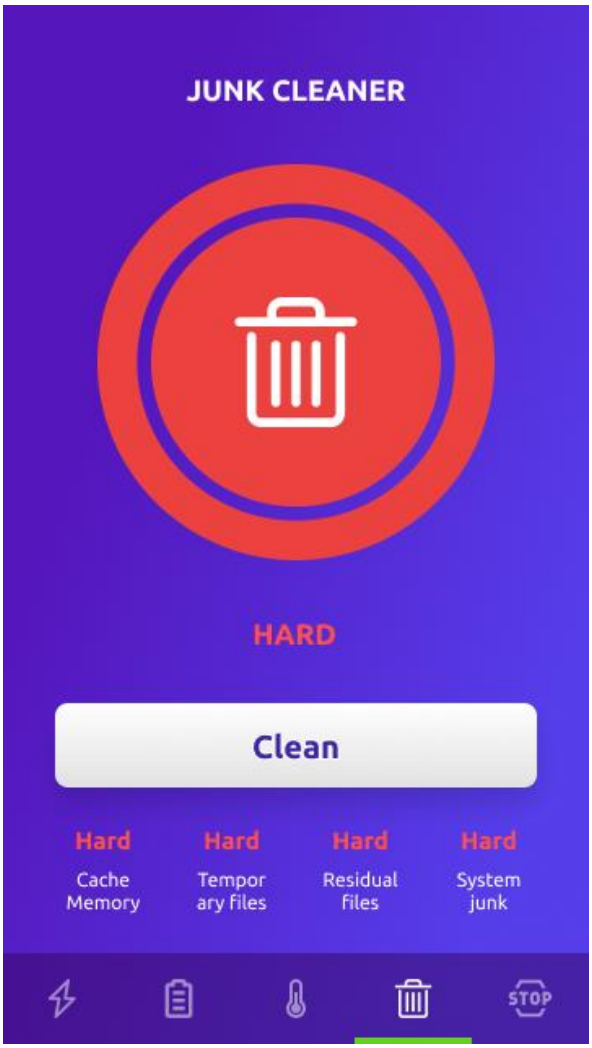

Рисунок 4.8 – Вікно очистки кеш-пам’яті

Четвертий пункт у меню «Junk Cleaner» пропонує очистити телефон від непотрібного сміття. Натиснувши «Clean» (рис. 4.8), відбувається пошук та очистка кеш-пам’яті (рис. 4.9), небажаних та тимчасових файлів, залишків файлів та системного сміття.

Очищення кеша і даних на Android – це дві абсолютно різних дії. Наприклад, музичні сервіси часто зберігають у кеш інформацію, що відноситься до виконавців, яких нещодавно прослуховували, але які не входять до бібліотеки. Коли кеш застосунку очищається, всі згадані дані стираються.

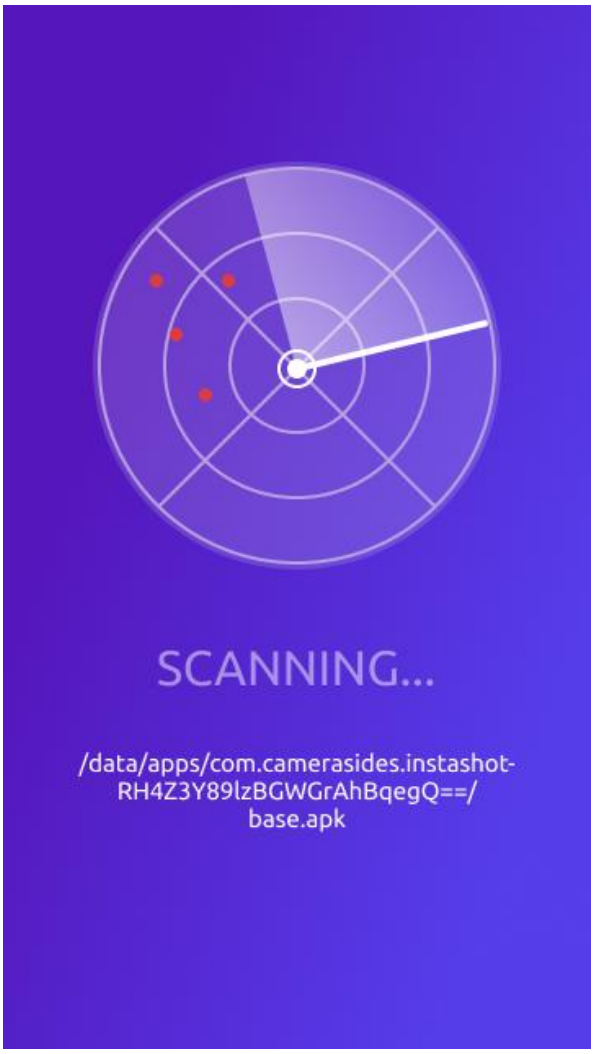

Рисунок 4.9 – Сканування та очистка пам’яті

Можна очистити вміст кеш-пам'яті, щоб звільнити додаткові місця на Вашому пристрої, що допомагає усунути проблеми, пов'язані з запуском або роботою застосунків з пошкодженими файлами кеша.

На рис. 4.10 можна побачити результати роботи «Junk Cleaner». Як видно з рисунку, очищені: кеш-пам’ять, тимчасові файли, залишкові файли та системне сміття.

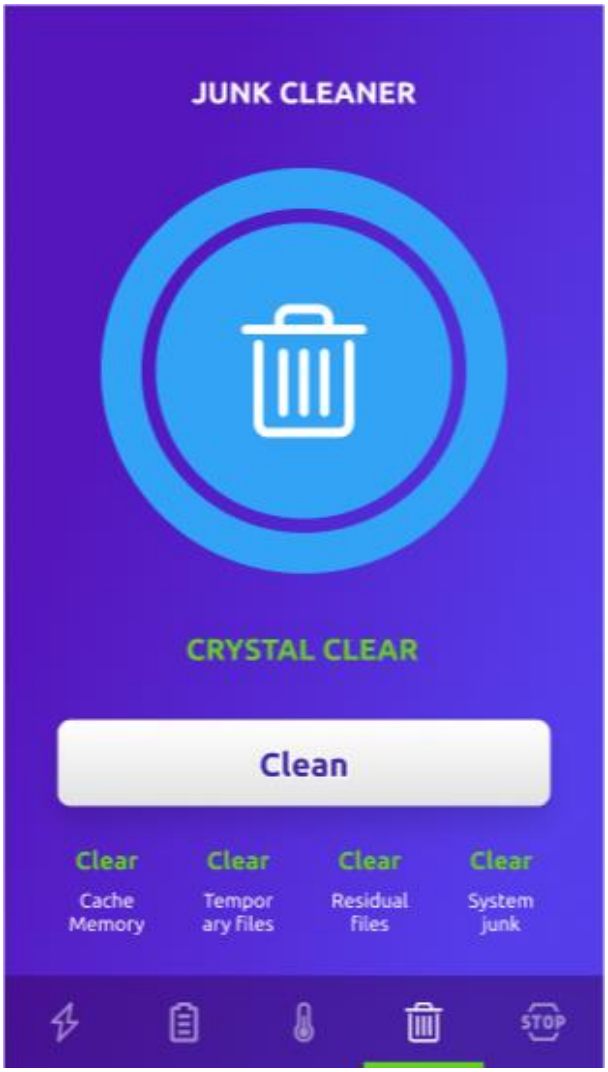

Рисунок 4.10 – Результат роботи очистки

4.2 Тестування програми

Не обов’язково використовувати стандарт документ, тест план повинен як мінімум описувати наступне:

1. Що треба тестувати?

   Опис об'єкта тестування: системи, застосунки, обладнання

2. Що буде протестовано?

   Список функцій і опис тестованої системи і її компонент окремо

3. Як буде проводиться тестування?

Стратегія тестування, а саме: види тестування (функціональне, нефункціональне) і їх застосування по відношенню до об'єкта тестування

4. Коли буде проводитися тестування?

Послідовність проведення робіт.

Зміст

1 Мета тестування

Мета тестування: виявлення функціональних помилок, невідповідностей технічному завданню і очікуванням Замовника шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

2 Ідентифікація об'єктів тестування

Контроль якості повинен бути підданий програмно-апаратний комплекс в цілому, а також його окремі частини.

Так зокрема, має бути проведено тестування:

- Додатка в цілому, розгорнутого в промисловому середовищі.
- Окремі компоненти програми.

3 Стратегія тестування

Поточний підхід до контролю якості має на увазі наступні віхи проекту:

- Підсистема готова до демонстрації замовнику.

- Підсистема готова до промислової експлуатації.

4 Види тестів

4.1 Функціональне тестування

Використовується для контролю якості “Функціональних можливостей” в частині “Придатності”, “правильності” і “Здібності до взаємодії”.

Функціональне тестування є основним видом тестування. Проводиться вручну через інтерфейс користувача.

Рекомендується використовувати тестування методом чорного та білого ящиків.

4.2 Тестування продуктивності

Використовується для контролю якості “Ефективності”.

Для аналізу поведінки призначеного для користувача інтерфейсу на реальних обсягах даних використовується ручне тестування.

4.3 Стрес тестування

Використовується для контролю якості “Надійності” в частині “Стабільності” і “Стабільності до помилки”.

4.4 Тестування користувальницького інтерфейсу

Використовується для контролю якості “Практичності” в частині “Зрозуміло”, “здатність до навчання”, “Простоти використання”.

Класифікація функцій, які будуть протестовані:

1) Функціональне тестування.

2) Конфігураційне тестування.

3) Тестування продуктивності

4) Стрес тестування

5) Тестування користувальницького інтерфейсу

Календарний план тестування наведено у таблиці 1.

Таблиця 1 – Календарний план робіт.

Номер тесту	Дата початку	Дата кінця	Дні				
			25.05	26.05	27.05	28.05	29.05
1	25.05.2021	29.05.2021	+	+			
2				+	+		
3					+		
4					+		
5					+	+	+

ВИСНОВКИ

В ході виконання дипломної роботи був спроектований та розроблений системний застосунок для мобільних пристроїв на платформі Android, який забезпечує прискорення роботи мобільного пристрою та збільшує термін використання батареї за рахунок закриття фонових застосунків та очищення ОЗП.

Незважаючи на зрослу популярність мобільних пристроїв, програми для очищення і оптимізації роботи смартфону все ще затребувані, а спеціальні застосунки чистильники працюють на більш глибоких рівнях, і тому ефективніше, ніж ручна очистка. Система Android має багато різних прихованих процесів, які завжди працюють у фоновому режимі, але на відміну від комп'ютера або ноутбука доступ користувачів до цих процесів не завжди можливий. Смартфони та інші мобільні пристрої стали частиною нашого повсякденного життя. За допомогою мобільних телефонів можна не тільки спілкуватися один з одним, але і замовляти товари з магазинів, купувати квитки, бронювати житло, викликати таксі, використовувати телефони як навігатори, фото- і відеокамери, онлайн книги, онлайн банки і тд. Актуальність розробки мобільних застосунків зростає не те що з кожним роком, а й з кожним місяцем.

В ході виконання дипломної роботи було отримано повнофункціональне програмне забезпечення «Cleaner для платформи Android», повністю готовий до застосування. Розроблене програмне забезпечення має можливість проводити очистку кеш-пам'яті смартфону, збирати та аналізувати дані про використання ОЗП пам’яті, проводити оптимізацію роботи мобільного пристрою, зменшувати температури процесору та продовжувати роботу батареї за рахунок енергоекономії. Інтерфейс створеної програми зручний, простий, наочно відображає її можливості.

Після розробки, проект можна удосконалювати увесь час, робити його більш функціональним та більш зручним для користування.

Розроблене програмне забезпечення задовольняє всім вимогам, поставленим на етапі постановки завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Типи мобільних застосунків [Електронний ресурс] – Режим доступу: https://avivi.academy/blogs/tipi-mobilnikh-dodatkiv;

2. Шилдт Г., Java 8. Полное руководство; 9-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 1376 с.;

3. Страуструп Б., Программирование: принципы и практика использования C++, испр. изд.: Пер. сангл. – М. Вильямс, 2011. –1248 с.;

4. Герберт Ш., C 4.0: повне керівництво.: Пер. з англ. – М.: ООО”І.Д. Вільямс”, 2011– 1056 с.;

5. Справочник по Kotlin [Електронний ресурс] – Режим доступу: https://medium.com/android-news/learn-kotlin-while-developing-an-android-app-introduction-567e21ff9664;

6. Гриффитс Д., Гриффитс Д., «Head First. Kotlin»; 2-е изд.: Пер. з англ. – М. Вільямс, 2018. – 464 с.;

7. Мінухін С. В., Беседовський О. М., Знахур С. В, «МЕТОДИ І МОДЕЛІ ПРОЕКТУВАННЯ НА ОСНОВІ СУЧАСНИХ CASE-ЗАСОБІВ», 2008 – 278 с.;

8. Шапошніков І., «Справочник WEB-мастера XML», 2013 – 257 с.;

9. Вивчаємо Android. Перший курс. [Електронний ресурс] – Режим доступу: http://developer.alexanderklimov.ru/android/;

10. Дэвид Гриффитс, Дон Гриффитс «Head First. Программирование для Android», 2018 – 912 с., O’Reilly, 2-е изд., ISBN: 978-5-4461-0708-7;

11. Neil Smyth, «Android Studio Development Essentials», 2014 – 620 с., ISBN-13: 978-1500613860;

12. Підручник з дизайну «PIXEL PERFECT PRECISION V.3»[Електронний ресурс] – Режим доступу: https://shron1.chtyvo.org.ua/Gyppsy/Pixel_Perfect_Precision_Ver3.pdf?PHPSESSID=n86s979c7tbe0vvj18phr4fj41