МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ __Гамаюн В.П.
(підпис) (ПІБ)

"_____"_____2021р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: Android-додаток шифрування даних

Виконавець: _____ Бут Валентин Анатолійович
(підпис) **(ПІБ)**

Керівник: _____ Ходаков Данііл Вікторович
(підпис) **(ПІБ)**

Нормоконтролер: _____ Боровик Володимир Миколайович
(підпис) **(ПІБ)**

Київ 2021

ВСТУП

У сучасному світі мобільні електронні пристрої поширюються дуже швидко. Потужності збільшуються в геометричній прогресії. З'являються нові технології, які дозволяють розмістити ще більші потужності на меншій площі. Більшість з них використовує Android. Android володіє найбільшою користувальницької базою серед всіх настільних і мобільних операційних систем. При цьому «Зелений робот» пробрався не тільки в смартфони та планшети: в наші дні цілком буденними стали телевізори, розумні годинник і навіть автомобілі з Android. Компанії зі сфери розробки програмного забезпечення пропонують безліч засобів для розв'язання тих чи інших проблем. Проте часто виникає потреба розв'язати нетривіальну задачу або просто створити новий продукт. У такому разі єдине, що залишається - це розробка унікального продукту. Взагалі розробка будь-якого програмного продукту потребує великих затрат: як часу, так і ресурсів (людських та матеріальних). Тому під час розробки чогось нового, потрібно врахувати усі ризики.

Вибрано тему «Androi-додаток шифрування даних», так як люди зберігають на смартфонах безліч даних: контакти, фотографії, важливі документи, аудіозаписи та відеоролики, історії дзвінків і СМС. Люди спілкуються в месенджерах і соціальних мережах, читають пошту, роблять замовлення в інтернет-магазинах, проводять банківські платежі, прокладають маршрути по містах і за їх межами. Сучасний мобільний телефон – багатоводне джерело корисної інформації для зловмисника. Власникам смартфонів безумовно є що захищати.

Мета даного проекту – створити програму для безпечного обміну даними з віддаденим сервером. У программі повинна бути реєстрація користувача та можливість доступу до інформації, яка завантажується з серверу шляхом подвійної авторизації.

РОЗДІЛ 1
ТЕХНОЛОГІЇ РОЗРОБКИ ДОДАТКІВ ПІД ОПЕРАЦІЙНУ СИСТЕМУ ANDROID

1.1. Історія і розвиток мобільних додатків та операційної системи Android

Мобільний додаток – це програмне забезпечення, спеціально розроблене під конкретну мобільну платформу (iOS, Android, Windows Phone і т. д.). Призначено для використання на смартфонах, планшетах, розумних годинах і інших мобільних пристроях.

Зазвичай мобільні пристрої продаються вже з деякими встановленими додатками. Решта за бажанням користувача можна завантажити (як платно, так і безкоштовно) на спеціалізованих сервісах: Apple AppStore, Google Play, Windows Phone Store та інших. Перші магазини додатків, такі як Apple AppStore і Android Market, який став згодом Google Play, з'явилися в 2008 році. Через два роки Американське діалектичне суспільство назвало термін «додаток» словом року.

Android — операційна система ¡ та платформа для мобільних телефонів і планшетних комп'ютерів, створена компанією Google на базі ядра Linux та підтримується альянсом Open Handset Alliance. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення та застосування спираються на цю реалізацію Java[9].

Перша версія Android була випущена 23 вересня 2008 року і носила назву 1.0 Astroboy, а наступна — 1.1 Bender. Від назв на честь відомих роботів згодом довелося відмовитися через розбіжності з правовласниками.

З 2008 року Android пережив численні оновлення, які поступово покращували операційну систему, додаючи нові функції, та виправляючи помилки у попередніх випусках.

1.2. Засоби розробки програмного забезпечення для Android

Для розробки програмного забезпечення для Android використовується Android Software Development Kit (Android SDK). Код пишеться на Java. Для тестування програм можна використовувати як пристрій, підключений до комп'ютера, так і емулятор Android, який іде у пакеті з Android SDK. Також можна використовувати Android Native Development Kit (Android NDK). NDK дозволяє писати програмне забезпечення для Android на C, C++ та інших поширених мовах програмування. Проте у цьому разі при розробці не буде доступне відлагодження. Програми для Android можна розповсюджувати через Android Market (Google Play).

ПЗ (програмне забезпечення) для Android зазвичай розробляється у програмі Eclipse. У пакеті Android SDK є версія даної програми зі всіма потрібними для розробки бібліотеками.

Щоб створити порожній проект, достатньо вибрати відповідний пункт меню. Після цього будуть створені базові папки та файли, необхідні для роботи з проектом. Створюються папки src, gen, assets, bin, libs, res. У папці src (source) знаходяться вихідні коди програми у файлах з розширенням.java. У папці gen (generated) знаходяться автоматично згенеровані файли. Вони присвоюють кожному логічному елементу програми унікальний ідентифікатор, зберігають дані про ресурси та інше. У папці asssets знаходяться додаткові ресурси, якщо вони потрібні. У папці bin розташовані файли компільованого проекту. Також там знаходиться.apk файл, який використовується для встановлення програми на пристрої з Android. У папці libs (libraries) розміщені додаткові бібліотеки, якщо такі використовуються. Щоб додати додаткову бібліотеку, достатньо її завантажити та додати, використовуючи відповідний пункт меню. У останній згенерованій папці res (resources) знаходяться ресурси, потрібні для роботи програми. У цій папці розташовані дочірні папки drawable, layout, menu, values та інші. У папці

drawable розміщують графічні ресурси, у папці layout - файли-макети програми, у папці menu - макети меню, у папці values - ресурси, такі як стрічки (Strings), кольори (Colors), розміри (Dimens), стилі макетів (Styles) та інші. Якщо для різних розширень потрібно використати різні ресурси, їх розміщують у відповідних папках з суфіксами. У папках з суфіксом - hdpi - ресурси та макети для великих екранів, - ldpi - не великих екранів, - mdpi - середніх за щільністю екранів. Щільність екрана вимірюється у DPI (dots per inch) - крапках на дюйм. Відповідні розміри: ~120 dpi - невеликий екран (-ldpi), ~160 dpi - середній екран (-mdpi), ~240 dpi - великий екран (-hdpi). З появою екранів з великим розширенням (Retina) були введені ще два розміри екранів: ~260 dpi - дуже великий екран (-xhdpi) та >260 dpi - надвеликий екран (-xxhdpi). Також використовують суфікси, які залежать не від щільності, а від розміру екрану:

- small, - medium, - large, - xlarge.

Якщо програма використовує різні макети для різних положень екранів, файли відповідних макетів розміщують у різних папках. За вертикальне положення відповідає суфікс - port, за горизонтальне - - land. При повороті екрану програма буде використовувати ресурси з відповідних директорій. Для локалізації програми при розробці достатньо продублювати папку values з відповідними суфіксами. Суфікс папки values повинен відповідати міжнародному позначенню країни (ресурси для української версії розміщують у папці values-ua, англійської - values-en). Якщо для країни не вказана відповідна папка, програма буде використовувати ресурси з папки values (без суфікса).

Отже, структура програм для Android добре продумана. Можна легко додати локалізації чи адаптувати програму для різних розмірів екранів. Це надзвичайно важливо, так як існують сотні різних пристроїв на Android з різною конфігурацією та різними розмірами екранів.

1.3. Середовище розробки Android Studio

Android Studio – є офіційним *IDE* для розробки додатків *Android*, на основі *IntelliJ IDEA*. На вершині можливостей, які ви очікуєте від *IntelliJ*, *Android* Студія пропонує.

- Гнучка *Gradle*-система збірки.
- Побудувані варіанти і кілька АПК покоління файлу.
- Шаблони коду, щоб допомогти вам побудувати загальні риси додатку
- Багатий редактор макетів з підтримкою перетягування і падіння редагування теми.
- Інструменти, щоб фіксувати продуктивність, зручність використання, сумісність версії, і інші проблеми *Proguard*.
- Вбудована підтримка для *Google Cloud Platform*, що дозволяє легко інтегрувати *Google Cloud* повідомленнями.

Android SDK - включає в себе різноманітні бібліотеки, документацію та інструменти, які допомагають розробляти мобільні додатки для платформи *Android*. [13, 14]

- *API Android SDK-API* бібліотеки *Android*, що надаються для розробки додатків.
- Документація *SDK*-включає велику довідкову інформацію, що деталізує, що включено в кожен пакет і клас і як це використовувати при розробці додатків.
- *AVD* (*Android Virtual Device*) -інтерактівний емулятор мобільного пристрою *Android*. Використовуючи емулятор, можна запускати і тестувати програми без використання реального *Android* пристрою.
- *Development Tools - SDK* включає кілька інструментальних засобів для розробки, які дозволяють компілювати і налагоджувати створювані додатки.
- *Sample Code - Android SDK* надає типові додатки, які демонструють деякі з можливостей *Android*, і прості програми, які показують, як використовувати індивідуальні особливості *API* у вашому коді.

Перед початком розробки додатків для *Android* корисно зрозуміти загальний підхід платформи до управління зміною *API*. Також важливо зрозуміти *Android API Level* (ідентифікатор рівня *API*) і його роль у забезпеченні сумісності вашого застосування з пристроями, на яких воно буде встановлюватися.

Рівень *API* - цілочисельне значення, яке однозначно визначає версію *API* платформи *Android*. Платформа забезпечує структури *API*, які додатки можуть використовувати для взаємодії з системою *Android*. Кожна наступна версія платформи *Android* може містити оновлення *API*.

Оновлення *API*-структури розроблені так, щоб новий *API* залишався сумісним з більш ранніми версіями *API*. Таким чином, більшість змін в *API* є сукупною і вводить нові функціональні можливості або виправляє попередні.

Оскільки частина *API* постійно оновлюється, застарілі *API* не рекомендуються до використання, але не видаляються з міркувань сумісності з наявними додатками.

Рівень *API*, який використовує додаток для *Android*, визначається цілочисловим ідентифікатором, який вказується у файлі конфігурації кожного *Android*-додатку.

У таблиці 1.1 наведено відповідність рівня *API* і версії платформи *Android*.

Крім емулятора, *SDK* також включає безліч інших інструментальних засобів для налагодження та установки створюваних додатків [15, 16].

Якщо При розробці програми для *Android* за допомогою *IDE Android Studio*, багато інструментів командного рядка, що входять до складу *SDK*, вже використовуються при складанні і компіляції проекту.

Таблиця 1.1

Відповідність версії платформи та рівня API

Версія платформи	Рівень *API*
Android 6.0	23
Android 5.0	21
Android 4.3	19
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

Однак крім них *SDK* містить ще ряд корисних інструментів для розробки і налагодження додатків:

• *Android* - важливий інструмент розробки, що запускається з командного рядка, який дозволяє створювати, видаляти і конфігурувати віртуальні пристрої, створювати і оновлювати *Android* проекти (при роботі поза середовища *Eclipse*) і оновлювати *Android SDK* новими платформами, доповненнями і документацією;

• *Dalvik Debug Monitor Service* (*DDMS*) - інтегрований з *Dalvik Virtual Machine*, стандартної віртуальні машини платформи *Android*, цей інструмент дозволяє управляти процесами на емуляторі, а також допомагає в налагодженні додатків. Можна використовувати цей сервіс для завершення процесів, вибору певного процесу для налагодження, генерування трасувань даних, перегляду "купи" або інформації про потоки, робити скріншоти емулятора та багато іншого[16];

• *Hierarchy Viewer*- візуальний інструмент, який дозволяє налагоджувати і оптимізувати користувальницький інтерфейс розробляється. Він показує візуальне дерево ієрархії уявлень, аналізує швидкодію перемальовування графічних зображень на екрані і може виконувати ще багато інших функцій для аналізу графічного інтерфейсу додатків;

• *Layoutopt*- інструмент командного рядка, який допомагає оптимізувати схеми розмітки та ієрархії розміток в створюваному додатку. Необхідний для вирішення проблем при створенні складних графічних інтерфейсів, які можуть зачіпати продуктивність програми;

• *Draw* 9-*patch* - графічний редактор, який дозволяє легко створювати *Nin*ePa*tch* графіку для графічного інтерфейсу розроблюваних додатків;

• *sqlite*3 - інструмент для доступу до файлів даних *SQLite*, створених і використовуваних додатками для *Android*;

• *Traceview* - цей інструмент видає графічний аналіз трасувань логів, які можна генерувати з додатків;

• *mksdcard* - інструмент для створення образу диска, який ви можете використовувати в емуляторі для симуляції наявності зовнішньої карти пам'яті (наприклад, карти *SD*).

Найбільш важливий з них цих інструментів - є емулятор мобільного пристрою, однак до складу *SDK* входять і інші інструменти для налагодження, упаковки та інсталяції ваших додатків на емулятор.

1.4. Розробка інтерфейсу користувача

Інтерфейс користувача створюється за допомогою виглядів (View), груп виглядів (ViewGroup), та макетів (Layouts) (рисунок 1.1). Об’єкти вигляду зазвичай є елементами інтерфейсу користувача, такими як кнопки, поля для введення інформації, текстові поля та інші. Групи виглядів - невидимі об’єкти. Вони є контейнерами, які визначають вигляд дочірніх виглядів, наприклад, сіткою (grid) чи вертикальним списком (vertical list). Весь

інтерфейс базується на ієрархії виглядів. Самі вигляди зберігаються у файлах формату.xml.

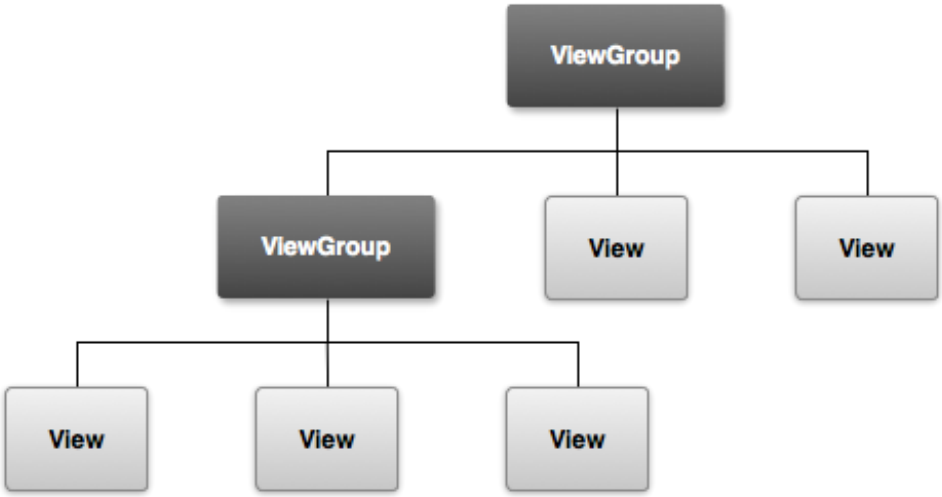

Рис. 1.1. Ієрархія виглядів деякого макету

Вигляди та макети можуть бути задані як у файлах з розширенням.xml, так і програмним шляхом. Але розробники Android пропонують визначати елементи інтерфейсу у файлах макетів, оскільки це забезпечує більш гнучке пристосування до різних екранів: достатньо створити альтернативний макет для іншого розширення та розмістити його у відповідній папці, після чого програму можна буде використовувати на іншому екрані. Визначення інтерфейсу програмним шляхом вимагає дублювання частини коду. Також у програмі Eclipse є візуальний редактор, який дозволяє візуально редагувати файли макетів. Якщо їх визначати програмним шляхом, візуальний редактор буде недоступний.

1.5. Основні типи використаних елементів інтерфейсу

Linear Layout - лінійне розташування. У ньому об’єкти інтерфейсу розміщуються горизонтально або вертикально у тій послідовності, у якій вони задані. Розробник сам визначає, як відображати об’єкти у блоці LinearLayout за допомогою атрибуту android:orientation, який може приймати усього два значення: horizontal(горизонтально) або vertical (вертикально).

При використанні LinearLayout доступний ще один важливий атрибут - android:layout_weight. Він дозволяє задати розміри дочірніх елементів пропорційно батьківському.

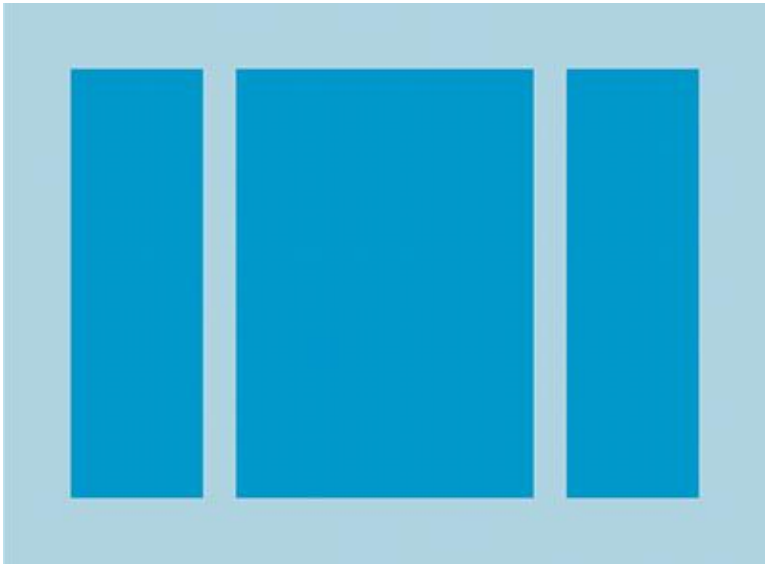

Рис. 1.2. Приклад лінійного розташування

Relative Layout - відносне розташування. У ньому об’єкти інтерфейсу розташовуються відносно один одного (справа, зліва, під та інші). Для визначення розміщення використовуються такі атрибути: android: layout_alignParentTop (приймає значення true або false, якщо true, то верхня межа елементу рівняється з верхньою межею батьківського елемента), android:layout_centerVertical (якщо true, то дочірній елемент розміщується вертикально по середині), android:layout_below (аргументом виступає ідентифікатор елементу, під яким буде розташований даний елемент) та багато інших. Всі атрибути такого типу приймають значення boolean (true або false) або ідентифікатор елементу, відносно якого вони мають розташовуватись.

Рис. 1.3. Відносне розташування

ListView - список, група виглядів (View Group), у якій всі елементи розташовуються у списку з можливістю прокрутки по вертикалі. Для додання елементів у список використовується клас адаптерів (Adapter), який отримує інформацію з деякого джерела (масив, список і т. д.), та конвертує її у елементи списку, які додаються до елементу ListView.

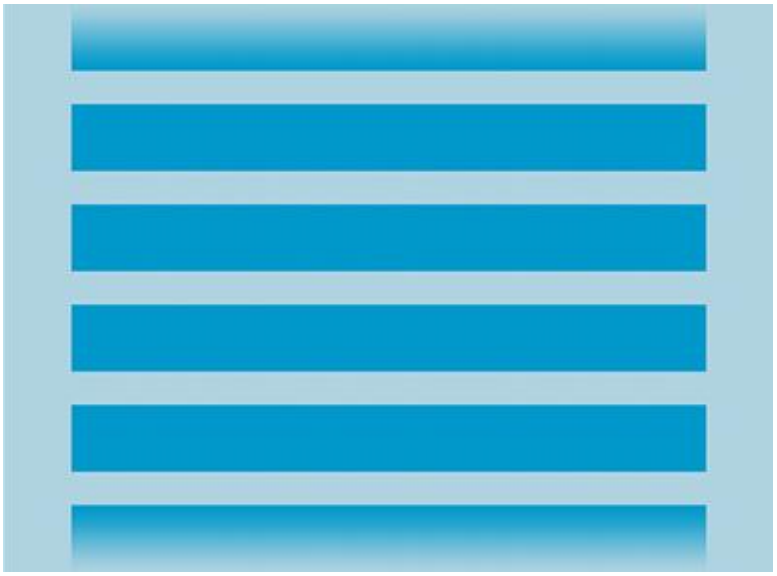

Рис. 1.4. Приклад списку елементів

Grid View - сітка, група виглядів, яка відображає елементи у вигляді двовимірної сітки, елементи якої можуть прокручуватися вертикально. Для

додання елементів до елементу GridView також використовують клас адаптерів.

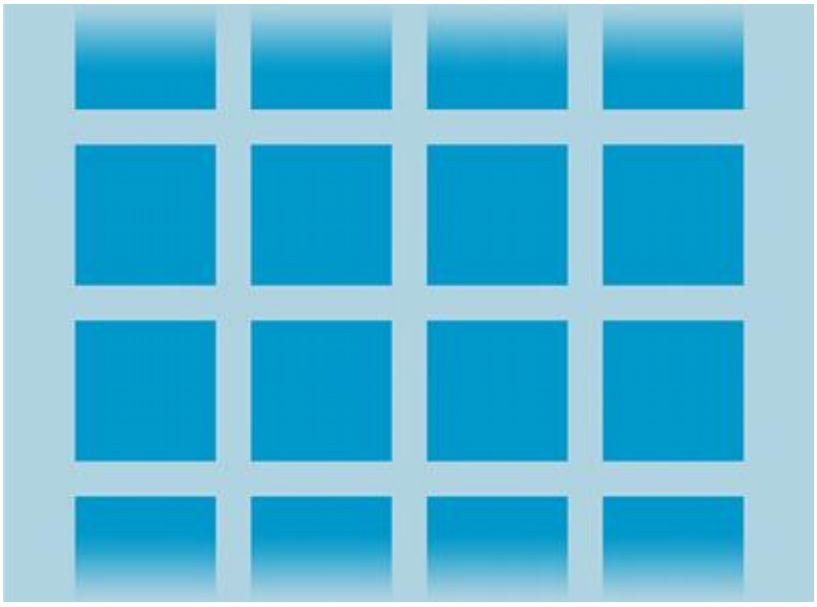

Рис. 1.5. Приклад сітки

1.6. Процеси і потоки в Android

Коли хоча б один з компонентів програми (або весь додаток) буде затребуваний, система Android запускає процес, який містить єдиний основний потік для виконання. За замовчуванням всі компоненти програми працюють в цьому процесі і потоці.

Однак можна вжити заходів, щоб компоненти працювали в інших процесах і порождали додаткові потоки для будь-якого процесу.

Всі компоненти ініціалізуються в основному потоці процесу. Окремі потоки для кожного екземпляра зазвичай НЕ створюються. Отже, все методи зворотного виклику, Визначені в компоненті та що викликаються системою, завжди працюють в основному потоці процесу. Це означає, що компоненти не повинні виконувати в методах зворотного виклику тривалі операції (наприклад, завантаження файлів з мережі або цикли обчислення) або блокувати системний виклик, так як це блокує будь-які інші компоненти в цьому процесі. Для таких операцій породжують окремі потоки.

Під час користування програмою, об’єкти типу Activity («діяльність») змінюють свої стани, зупиняються, продовжуються та завершуються. Під час написання програми, можна вказати, як повинна поводитись діяльність впродовж своєї роботи. Наприклад, якщо ви створюєте онлайн відео програвач, можна переривати з’єднання з Інтернетом та призупиняти мовлення коли користувач згортає програму та переключається на іншу. Це забезпечить економію інтернет-трафіку користувача. Коли користувач знову переключиться на вашу програму, діяльність може відновити підключення та продовжити відео-мовлення з моменту, на якому воно було зупинене. Принцип роботи Activity продемонстровано на (рис 1.6).

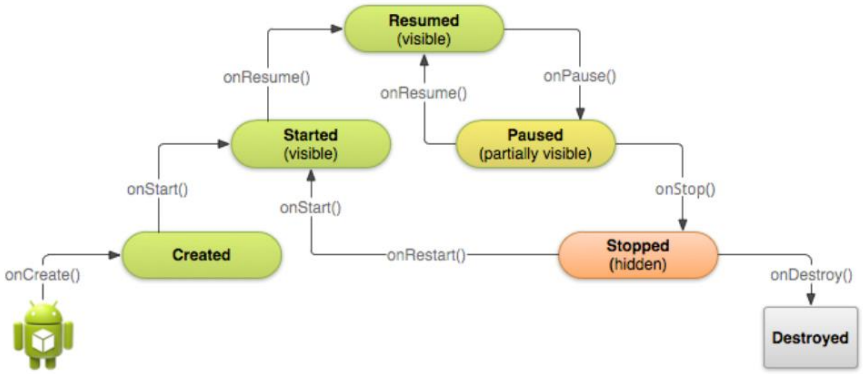

Рис. 1.6. Принцип роботи Activity

Кожна активність починається з методу void onCreate (Bundle instance). Для того, щоб створити свою активність, потрібно створити клас, який буде розширювати клас Activity, та перевизначити метод onCreate(). У ньому потрібно викликати батьківський конструктор, та передати йому об’єкт типу Bundle, який буде передано в перевизначений метод onCreate() автоматично при запуску діяльності. Далі, якщо дана діяльність використовує макет, потрібно його вказати, передавши методу setContentView() ідентифікатор макету. Тільки після цього з ним можна буде працювати. Наведено простий приклад діяльності.

```
public class Start extends Activity {
@Overridevoid onCreate
(Bundle savedInstanceState)
{.onCreate(savedInstanceState);(R.layout.activity_start);
}
}
```

Важливо, що діяльність перезапускається при повороті екрану. Тому якщо це може вплинути на коректну роботу програми, потрібно обробити такі ситуації. Якщо потрібно, можна заборонити перезапуск діяльності. Для цього досить вказати у файлі маніфесту атрибут для даної діяльності.

Файл маніфесту (AndroidManifest.xml) обов'язково повинен бути у кожній програмі. Він містить важливу інформацію для системи Android про програму, її вимоги, які діяльності вона включає, назву програми, які можливості пристрою вона буде використовувати, як вона буде взаємодіяти з іншими програмами, а також мінімальну та максимельну версію API, потрібну для коректної роботи системи.

Висновки до розділу

Під час дослідження було освоєно основи проектування і розробки додатків під *Android*, вивчено способи і можливості виходу на перспективний ринок *Android Market* . Надалі планується не зупинятися на досягнутому, а далі продовжувати займатися розробкою додатків для мобільних пристроїв, що використовують операційну систему *Android*, адже зараз з розвитком інформаційних технологій, дана сфера є дуже перспективною. Заслуги за це належать *Google*, тому слід зазначити, що розробники *Android* пішли дуже грамотним шляхом, давши будь-якому охочому можливість розробляти програми для даної платформи. Відкритий вихідний код дуже сильно спрощує життя стороннім розробникам.

Дана платформа в найближчі кілька років буде все активніше розвиватися і виштовхувати з ринку конкурентів. І це досить логічно. Простота розробки, якісна зворотній зв'язок, величезне число розробників по всьому світу, відкритий вихідний код, доступність виходу на ринок - ось шлях успіху в сучасному світі комп'ютерних мобільних технологій.

РОЗДІЛ 2
ОПИС ТА ВИБІР МЕТОДОЛОГІЇ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

2.1. Вибір схеми моделі життєвого циклу

Розробка під мобільні платформи має важливі відмінні характеристики, що відносяться до більшості фаз життєвого циклу, такі як:

- необхідність короткого часу доставки продукту на динамічний ринок, і подальших постійних оновлень;
- велика кількість користувачів з усього світу;
- складність у виявленні вимог до додатка, в тому числі з причини труднощів в ідентифікації стейкхолдерів;
- висока ймовірність змін потреб і очікувань користувачів, і відповідно необхідності вносити зміни по ходу розробки;
- високий темп технічної еволюції - нові пристрої, релізи ОС, МП, технології мобільного зв'язку, тощо.

Сьогодні існує чимало стандартних моделей проектування, які дозволяють поетапно, крок за кроком, реалізувати будь-який проект від ідеї до її втілення. Їх вибір залежить тільки від розробників і тих цілей, які вони переслідують.

Прикладами таких моделями є: «Каскадна модель», модель «Спіраль» і

однією з примітних таких моделей для проектування додатку є модель Уолта Діснея [5], вона складається з трьох етапів:

- концептуальне проектування;
- логічне проектування;
- фізичне проектування.

Етапи слідують послідовно один за іншим (рис. 2.1), але в деяких випадках можливий перехід до наступної стадії без закінчення попередньої.

Це може відбуватися, наприклад, коли є декілька розробників, кожен з яких працює зі своєю частиною додатку. У будь-якому випадку, після закінчення етапу фізичного проектування слід повернутися до початку і внести відповідні корективи[13].

Рис. 2.1. Етапи проектування

Концептуальне проектування. На початковій стадії проектування приймаються рішення, визначають подальший вигляд, і проводиться дослідження і узгодження параметрів створених технічних рішень з можливою їх організацією. На цьому етапі складно оцінити ефективність майбутнього додатку. Необхідно знати і розуміти критерії оцінки для того, щоб визначити хорошим чи поганим є розроблений ресурс. Є універсальний критерій, який досить точно характеризує ефективність додатку – це досягнення розробниками додатку поставлених перед ними цілей. У цьому випадку додаток перетворюється на якісний інструмент, який виконує покладені на нього функції. Концептуальне проектування служить для

вказівки цілей, завдань додатку та визначення аудиторії, на яку він розрахований.

На цьому етапі проектування слід описати наступне:

- основні і другорядні цілі;
- дії, які необхідно вжити для досягнення поставлених цілей;
- аудиторію додатку;
- інтереси груп користувачів;
- розділи додатку
- критерії досягнення мети.

Після визначення поставлених цілей й інтересів користувачів, можна скласти список сервісів й розділів, які будуть розташовуватися на додатокі[14].

Логічне проектування. Логічна модель мобільного додатка - версія концептуальної моделі, яка може бути реалізована конкретним додатком.

Логічна модель відображає логічні зв'язки між атрибутами об'єктів незалежно від їх змісту і середовища зберігання. Як правило, при розробці логічних структур використовуються в основному евристичних методи проектування з формальної оцінкою якості прийнятих рішень.

Визначені розділи додатку, на попередньому етапі, поки не впорядковані і не структуровані, тому їх потрібно привести до зручного та зрозумілого виду[7].

Логічне проектування включає організацію інформації в додатку, побудови її структури і навігації по розділах. На даному етапі слід задатися питанням, яким чином буде впорядкована інформація. Варіанти можуть бути самими різними і залежати від типу даних і переваг творців додатку: за часом, розділами, в алфавітному порядку, певним групам або іншими критеріями.

Одночасне використання різних способів охоплює більшу аудиторію і дозволяє швидше знайти потрібну інформацію на додатокі. На цьому етапі слід описати наступне:

- тип структури додатку (лінійна, ієрархічна, контекстна, інша);
- назви розділів;
- що буде містити в собі кожен розділ;
- організація та зв'язок розділів між собою;
- що буде розміщено на певних розділах додатку.

Кінцевим результатом логічного проектування є блок схема або структурна діаграма, що показують взаємозв'язок різних частин додатку.

Фізичне проектування. Фізичне проектування є третім і останнім етапом створення проекту програми, при виконанні якого проектувальник приймає рішення про способи реалізації розробляється. Під час попереднього етапу проектування була визначена логічна структура мобільного застосування (яка описує відносини і обмеження в даній прикладній області). Хоча ця структура не залежить від конкретного цільового програми, воно створюється з урахуванням обраної моделі. Однак, приступаючи до фізичного проектування додатка, перш за все необхідно вибрати конкретний цільовий додаток. Тому фізичне проектування нерозривно пов'язане з конкретним мобільним додатком. Між логічним і фізичним проектуванням існує постійний зворотний зв'язок, так як рішення, що приймаються на етапі фізичного проектування з метою підвищення продуктивності системи, здатні вплинути на структуру логічної моделі даних.

Як правило, основною метою фізичного проектування мобільного додатка є опис способу фізичної реалізації логічного проекту мобільного додатка.

У нашому випадку даний етап пов'язаний з пошуком проблем, а не їх рішень, пов'язаних, здебільшого, з технічною реалізацією додатку. На цьому етапі слід описати наступне:

- технології, які будуть застосовуватися в додатку;
- програмне забезпечення, що використовується;
- можливі проблеми та способи їх усунення;
- як буде оновлюватися інформація.

Після завершення цього етапу слід повернутися до концептуального проектування і перевірити, чи не потрібно внести зміни, у зв'язку з переосмисленням проекту на інших стадіях[6].

При виборі схеми моделі життєвого циклу (ЖЦ) для конкретної предметної області, вирішуються питання включення важливих для створюваного продукту видів робіт або не включення несуттєвих робіт. На сьогодні основою формування нової моделі ЖЦ для конкретної прикладної системи є стандарт *ISO*/*IEC*12207, що задає повний набір процесів (понад 40), що охоплює всі можливі види робіт і завдань, пов'язаних з побудовою програмного середовища (ПС), починаючи з аналізу предметної області і закінчуючи виготовленням відповідного продукту. Цей стандарт містить основні та допоміжні процеси (рис. 2.2 та рис. 2.3).

Рис. 2.2. Схема основних процесів ЖЦ ПС

На рис. 2.3 представлені процеси, пов'язані безпосередньо з розробкою ПС. До категорії основних процесів відносяться також "первинні" процеси, що визначають порядок підготовки договору на розробку ПС, моніторинг діяльності постачальників ПС замовнику.

Стандарт *ISO*/*IEC*12207 надає структуру процесів ЖЦ, але не зобов'язує використовувати всі процеси в моделі ЖЦ ПЗ або в конкретній методології розробки програмного забезпечення (ПЗ)[6].

Рис. 2.3. Схема допоміжних процесів ЖЦ ПЗ

При створенні додатку використовуємо стандарт *ISO/IEC* 12207 [17]. На основі цього розроблено методологію додатку для безпечної передачі файлів (Рис.2.4.).

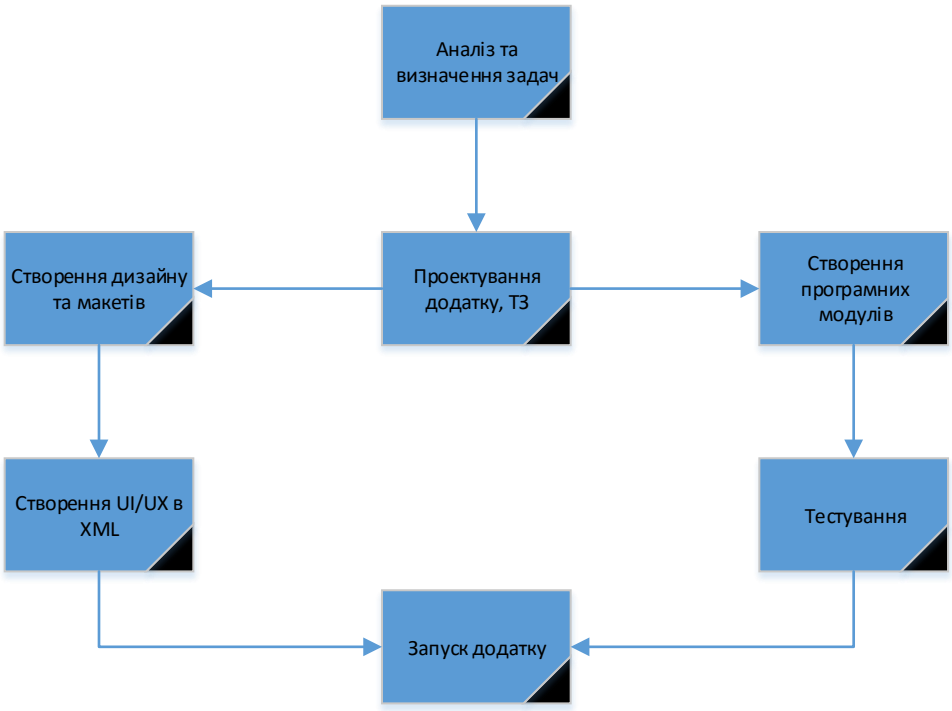

Рис. 2.4. Методологія створення додатку

Методологія розробки мобільного додатку включає наступні етапи:

- аналіз та визначення задач (планування), визначення мети створення додатку, тематики і призначення майбутнього додатку, можливостей реалізації проекту;
- проектування та розробка технічного завдання, визначення структури додатку, матеріалів, вибір програмних засобів для його створення;
- веб-дизайн – створення графічних елементів макету додатку, стилів і елементів навігації;
- розробка програмного коду, модулів, бази даних і інших елементів додатку необхідних в проекті, вибір та інтегрування *CMS*;
- заповнення додатку матеріалами;
- тестування і розміщення додатку на хостингу в мережі інтернет;
- запуск та навчання [7].

2.2. Аналіз та визначення задач мобільного додатку

На етапі планування, перш за все, слід визначити призначення майбутнього додатку. Тут доцільно визначити, буде додаток вузьконаправленним чи різні його розділи будуть з різнім функціоналом і яким саме. На етапі планування можна виділити наступні задачі:

- фіксація цілей і завдання додатку;
- максимально детальний опис майбутньої аудиторії;
- визначення того, що ми чекаємо від користувачів додатку;
- аналіз додатків – конкурентів;
- планування того, чим наш майбутній додаток буде відрізнятися від конкурентів;
- вибір сервісів та функцій;
- аналіз структури і функціоналу додатку;
- створення візуального плану ключових розділів;

- оцінка зручності використання додатку користувачем;
- планування способу отримання зворотного зв'язку;
- оцінка перспективи розвитку додатку [8].

2.3. Проектування та розробка технічного завдання

Наступним етапом є виконання представлених нижче етапів складання технічного завдання на створення мобільного додатку.

Формулювання завдання на розробку додатку – необхідно оформити своє уявлення про додаток документально – це технічне завдання (ТЗ), яке регламентує всі відносини замовника і виконавця[8].

У ТЗ зазвичай містяться вимоги по дизайну, навігації, способам представлення інформації, термінам і об’ємам робіт кожного етапу розробки додатку. Чітко сформульоване ТЗ допоможе уникнути конфліктів «нерозуміння» між виконавцем і замовником.

Технічне завдання найчастіше представлено таким чином: мета створення додатку, структура додатку, функціонал, побажання по дизайну додатку, взаємодія з користувачами.

Специфікація вимог програмного забезпечення (Software Requirements Specification, SRS), на практиці цей документ може називатися "технічне завдання" (ТЗ), Terms of Reference (TOR), Functional Requirements.

У роботі над специфікацією необхідно враховувати велику кількість даних:

- загальний блок інформації про проект: цілі, завдання, термінологія, аудиторія користувачів;
- загальний опис продукту: функціональність, деталізація користувачів (персони), операційне середовище, в якому буде експлуатуватися продукт, рамки, обмеження, правила і стандарти;
- більшу частину специфікації займають описи алгоритмів і процесів, flow-діаграми, функціональні вимоги, вимоги до інтерфейсів (UX, API, обладнання, якщо є);

– нефункціональні вимоги являють собою вимоги до гарантування безпеки даних, безпеки, швидкості, інтелектуальної власності і ліцензійної політики.

Наведемо діаграму варіантів використання для акторів системи (рис 2.5).

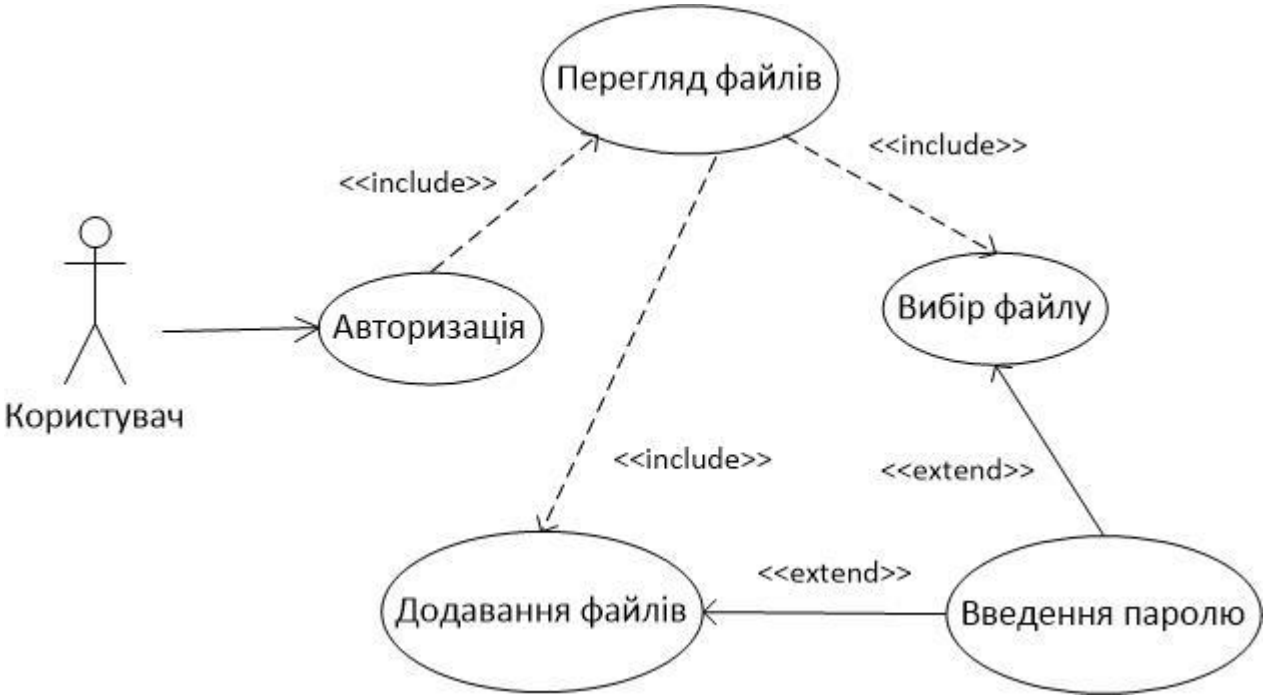

Рис. 2.5. Діаграма варіантів використання системи шифрування

2.4. UI/UX-дизайн створення графічних елементів макету додатку, стилів і елементів навігації та інтерфейсу

Наступним етапом розробки додатку є розробка інтерфейсу та створення дизайну основних розділів.

UI/*UX*-дизайн (від англ. *User Experience Design*) – в перекладі означає «досвід взаємодії» і включає в себе різні *UX*-компоненти: інформаційну архітектуру, проектування взаємодії, графічний дизайн і контент (рис.2.6).

Рис. 2.6. UX дизайн

UX дизайн має на увазі комплексний підхід до взаємодії користувача з інтерфейсом, будь-то веб-сайт, мобільний додаток або будь-яка інша програма.

При розробці інтерфейсу повинен по можливості максимально врахувати всі дрібниці, починаючи від середовища користувача і типу електронного пристрою і закінчуючи способами введення і відображення інформації.

Основні питання, які вирішуються *UX* дизайном.

– Постановка цілей і завдань - чого в підсумку необхідно досягти?

– Підбір відповідних *UX* інструментів для реалізації цілей.

– Розробка продукту, максимально зручного і легкого в сприйнятті цільовою аудиторією.

– Аналіз кінцевого результату - відповідає продукт очікуванням замовника і наскільки високий рівень задоволеності користувачів.

User Interface Design або користувальницький інтерфейс - це більш широке поняття, яке включає в себе певний набір графічно оформлених технічних елементів (кнопки, чекбокси, селектори та інші поля). Його завдання - допомогти користувачеві організувати взаємодію з програмою. На сьогоднішній момент існують деякі правила *UI* дизайну.

- Організованість елементів інтерфейсу. Це означає, що вони повинні бути логічно структуровані і взаємопов'язані.
- Угруповання елементів інтерфейсу. *О*б'єднання в групи логічно пов'язаних елементів (меню, форми).
- Вирівнювання елементів інтерфейсу. Складно уявити, що погано вирівняний інтерфейс може бути для когось зручним!
- Єдиний стиль елементів інтерфейсу. Стильове оформлення грає не останню роль, адже саме воно зберігається в пам'яті користувача.
- Наявність вільного простору. Це дозволяє розмежовувати інформаційні блоки, зосереджуючи увагу на чомусь одному.

Розроблений по всім правилам користувальницький інтерфейс значно підвищує ефективність ресурсу і дає йому конкурентні переваги.

При створенні дизайну додатку необхідно керуватися одним принципом: "Створювати дизайн необхідно тільки для ключових екранів". Ключові екрани – це ті екрани, які суттєво відрізняються від інших.

Зазвичай, створюється один або декілька варіантів дизайну, відповідно до технічного завдання. При цьому окремо створюється дизайн головної сторінки, і дизайни типових сторінок. Власне «дизайн сторінки» представляє собою графічний файл, листковий малюнок, що складається з найбільш дрібних картинок елементів загального малюнка.

Унікальний дизайн коштує дорожче, але й передбачає розробку з нуля, повністю унікальну розробку під конкретне замовлення. Залежно від професіоналізму та/або політики замовника веб-дизайнер або розробляє ідею і концепцію дизайну повністю самостійно, або одержує ряд вимог (колір,

стиль і т.п.), очікувань та ідей від замовника і намагається триматися цього напрямку при розробці макета[23].

Іноді пропонуються дизайн-рішення на основі шаблонів, що прискорює роботу. Кінцевим продуктом роботи є дизайн-макет: картинка, що представляє передбачуваний майбутній зовнішній вигляд сторінок додатку, розміром приблизно 960x640 *px* (пікселів) – розмір, відповідний середньому стандарту, пов'язаний з необхідністю подальшої прив'язки до різних дозволами екрану монітора. Картинка ця є багатошаровою, де майже кожна деталь – окремий шар, прикладений до інших верств-картинок, за рахунок чого може легко виконуватися доробка, заміна, перекомпонування і інші завдання. Залежно від ідеї і цілей макет може включати фотографії, складні колажі, ілюстрації, текстові шари, унікальні іконки. Для головної сторінки і внутрішніх іноді малюються окремі макети з доповненнями або змінами відповідно до тематики сторінки.

Зображення спочатку може бути векторним або растровим, виконаним в *Adobe Illustrator*, *Adobe Photoshop, GIMP* або іншому візуальному редакторі (наприклад, *Scribus* або *Inkscape*), але для верстальника зображення, як правило, переводиться в растровий формат [22].

2.5. Огляд структури проекту в середовищі Android Studio

За замовчуванням, Android Studio відображає файли проекту в Android project view. Ця точка зору показує сплюснуту версію структури вашого проекту, що забезпечує швидкий доступ до ключових вихідних файлів проектів Android і допомагає вам працювати з Gradle основі системи збірки. Android-вид проекту.

- Показує найбільш важливі каталоги джерело у верхньому рівні ієрархії модулів.
- Групи файлів збірки для всіх модулів в загальну папку.

- Групи всі файли маніфесту для кожного модуля в загальну папку.
- Показує файли ресурсів з усіх джерел наборів Gradle.
- Групи файли ресурсів для різних місцях, орієнтації і типів екранів в одній групі для кожного типу ресурсів

Android-view проекту показує всі файли збірки на вищому рівні ієрархії проекту під Gradle Сценарії (рис. 2.7). Кожен модуль проекту з'являється в папці на верхньому рівні ієрархії проекту і містить три елементи на верхньому рівні:

- java / - Вихідні файли для модуля;
- manifests / - файли маніфесту для модуля;
- res / - файли ресурсів для модуля;

Gradle Scripts / - Gradle збірка і файли властивостей (рис. 2.8).

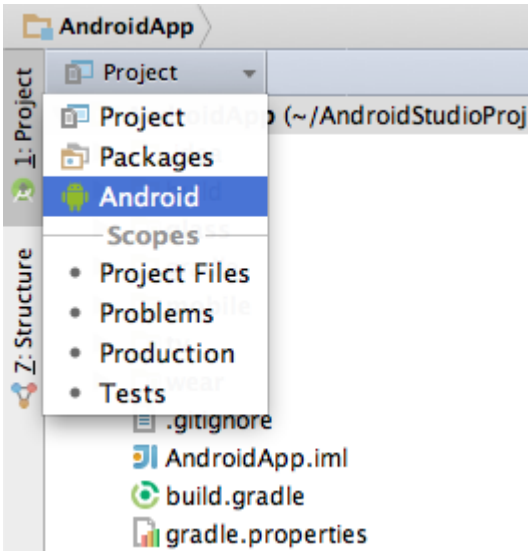

Рис. 2.7. Android project view

Наприклад, представлення проекту Android розглядає всі екземпляри ресурсу ic_launcher.png для різних щільностей екрану одного елемента.

Примітка: Структура проекту на диску відрізняється від цього подання. Щоб переключитися на вид з точки зору проекту, виберіть проект із списку Project.

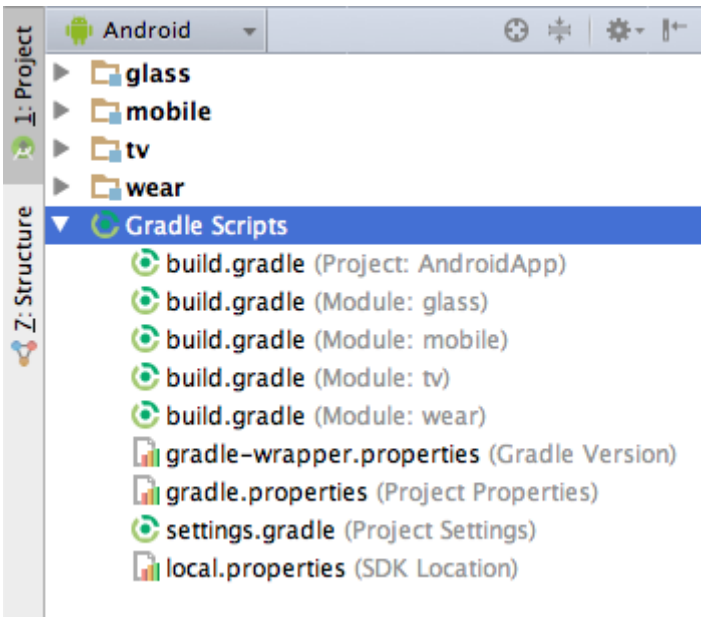

Рис. 2.8. Вигляд файлів збірки проекту

При використанні вид проекту в Android Studio (рис. 2.9), можна помітити, що структура проекту виглядає інакше, ніж в Eclipse. Кожен екземпляр Android Studio містить проект з одним або декількома модулями програми. Кожна папка модуль додатка містить комплекти джерела для цього модуля, у тому числі src/main і src/androidTest каталогів, ресурсів, створення файлу і Android маніфест. Здебільшого, ви повинні будете змінювати файли в src/main каталог кожного модуля оновлень вихідного коду, в файл gradle.build для складання специфікації і файлів в каталозі src/androidTest для створення тесту.

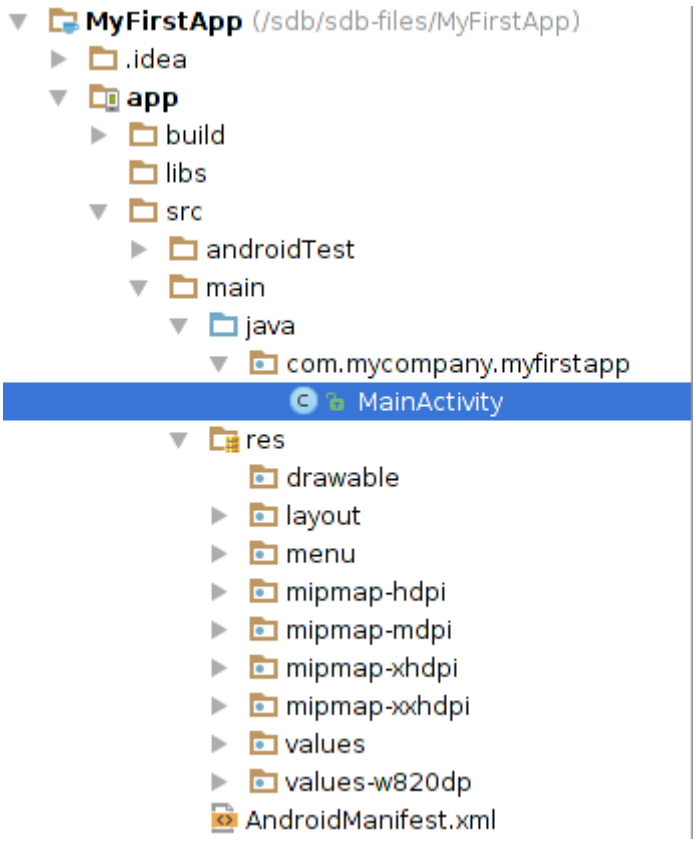

Рис. 2.9. Структура Android-проекту

Система збірки проектів в Android Studio. Система Android-збірки інструментарій, який використовується для створення, тестування, запуску і упакування додатків. Ця система збирання замінює Ant систему, використовувану з Eclipse, ADT. Вона може працювати в якості інтегрованого інструменту з меню Android Studio і самостійно з командного рядка. Стає можливим використовувати особливості побудови системи на:

– налаштування і розширення процесу складання;

– створення декількох APK для розроблюваного застосування з різними функціями, використовуючи той же проект і модуль;

– повторне використання коду і ресурсів різних наборів змінних.

Гнучкість системи збирання Android дозволяє досягти всього цього, не змінюючи вихідні файли ядра розроблюваного застосування.

З Android системи збирання, ApplicationID атрибут використовується для унікальної ідентифікації пакетів додатків для публікації. Ідентифікатор програми знаходиться в розділі андроїд файлу build.gradle.

```
  apply plugin: 'com.android.application'

android {
    compileSdkVersion 19
    buildToolsVersion "19.1"

defaultConfig {
    applicationId "com.example.my.app"
    minSdkVersion 15
    targetSdkVersion 19
    versionCode 1
    versionName "1.0"
}
...
```

При використанні побудування система збирання дозволяє однозначно ідентифікувати різні пакети для кожного продукту. Application ID додаються як суфікс до flavor проекту.

```
  productFlavors {
    pro {
        applicationId = "com.example.my.pkg.pro"
    }
    free {
        applicationId = "com.example.my.pkg.free"
    }
}

buildTypes {
    debug {
        applicationIdSuffix ".debug"
    }
}
....
```

Ім'я пакету має ще бути вказане у файлі маніфесту. Він використовується у вихідному коді, щоб звернутися до вашого R класу і для дозволу реєстрації activity/service.

Налагодження і продуктивність

Android Virtual Device(AVD) менеджер – оновив екрани з посиланнями, щоб допомогти вибрати найбільш популярні конфігурації пристрою, розміри екрану і щільності для ваших оглядів програм.

Натисніть Android Virtual Device Manager, в панелі інструментів, щоб відкрити його і створювати нові віртуальні пристрої для роботи вашого застосування в емуляторі.

Менеджер AVD поставляється з емуляторів для Nexus 6 і 9 Nexus пристроїв, а також підтримує створення скінів користувальницького Android пристрою на основі специфічних властивостей емулятора і призначення скінам профілів обладнання. Android-студія встановлює Intel® x86 Апаратне прискорення виконання Manager (HAXM) емулятор прискорювач і створює емулятора за замовчуванням для швидкого прототипування додатків.

Використовуючи вбудоване налагодження для поліпшення коду в цілях відладчика з інлайн перевірки посилань, виразів і значень змінних. Вбудована налагоджувальна інформація включає в себе:

- значення змінних;
- об'єкти, які посилаються на вибраний об'єкт;
- повернення значення методу;
- лямбда-вирази і оператор;
- значення-підказки.

Щоб включити вбудоване налагодження, у вікні Debug натискаємо Налаштування та встановіть прапорець для відображення значень в редакторі.

Пам'ять та CPU Monitor

Android-студія надає вид на пам'ять і процесор монітора, так що можливо легко відстежувати продуктивність і використання пам'яті

розроблюваного додатку, щоб відстежувати використання процесора, знайти звільненні об'єкти, виявити витоки пам'яті, і відслідковувати обсяг пам'яті підключених пристроїв. Додаток працює на пристрої або емуляторі, виберіть вкладку Android в лівому нижньому кутку вікна виконання, щоб відкрити вікно Android виконання. Перейдіть на вкладку пам'ять і процесор (рис. 2.10).

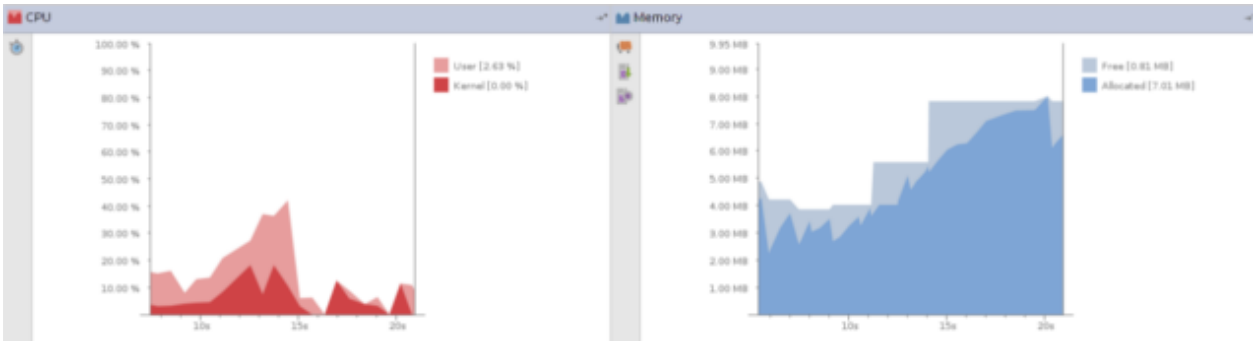

Рис. 2.10. Моніторинг пам'яті та використання CPU

Android-студія підтримує анотації для змінних, параметрів і значень, для «відлову» помилок. Наприклад, виключення нульових покажчиків і конфліктів типу ресурсів. Android SDK менеджер запаковує бібліотеку підтримки-анотації в Android репозиторії для використання з Android Studio. Android-студія перевіряє налаштовані анотації під час огляду коду.

Щоб додати анотації до коду в Android Studio, необхідно додати залежність для бібліотеки Support-Annotations.

1. Виберіть Файл> Структура проекту.
2. У діалоговому вікні «Структура проекту» виберіть потрібний модуль, натисніть вкладку «Залежності».
3. Натисніть на значок «Додати», щоб включити бібліотеки.
4. У діалоговому вікні Choose library dependency виберіть support-annotations і натисніть кнопку ОК.

Файл build.gradle оновлюється із залежністю support-annotations.

Також можливо вручну додати цю залежність у build.gradle файлу, як показано в наступному прикладі.

```
    dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.0.0'
    compile 'com.android.support:support-annotations:22.0.0'
}
```

Android Studio дозволяє працювати з макетами в двох видах конструктора. Інтерфейс додатку в режимі Design View (рис. 2.11). Інтерфейс додатку в режимі Text View (рис. 2.12).

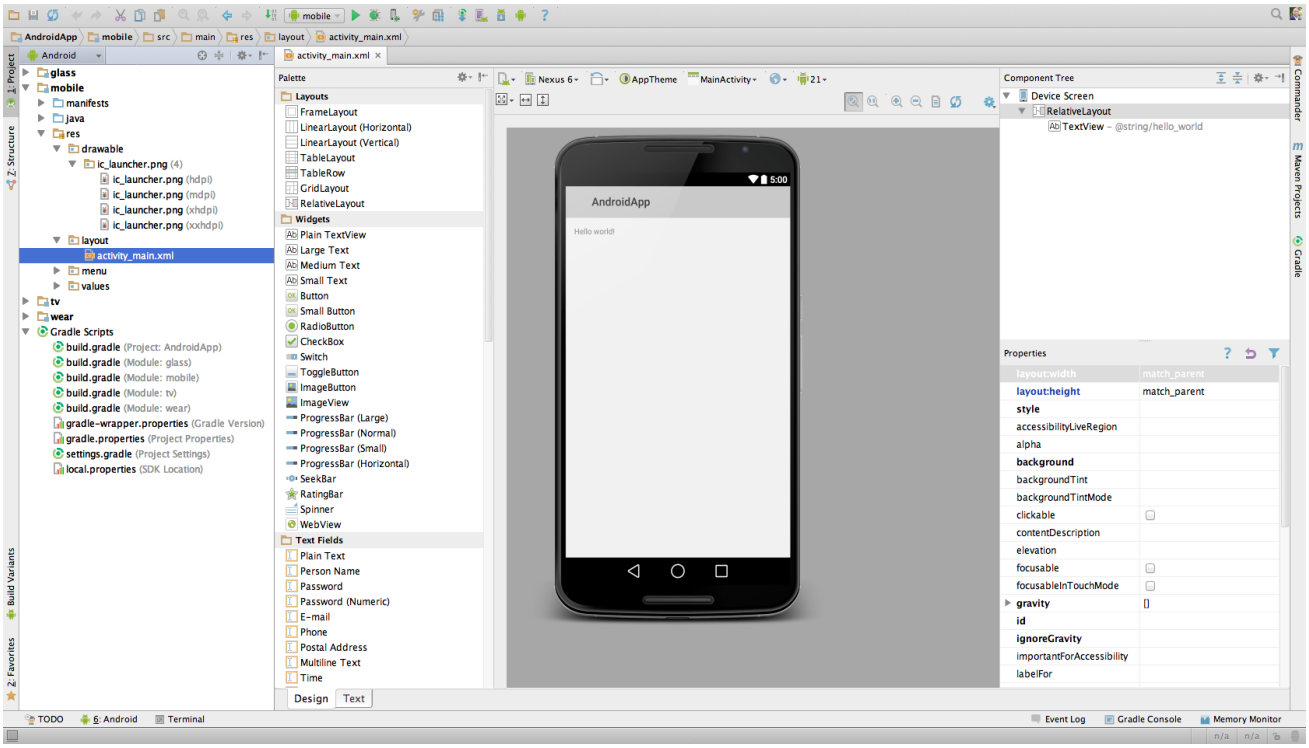

Рис. 2.11. Інтерфейс додатку в режимі Design View

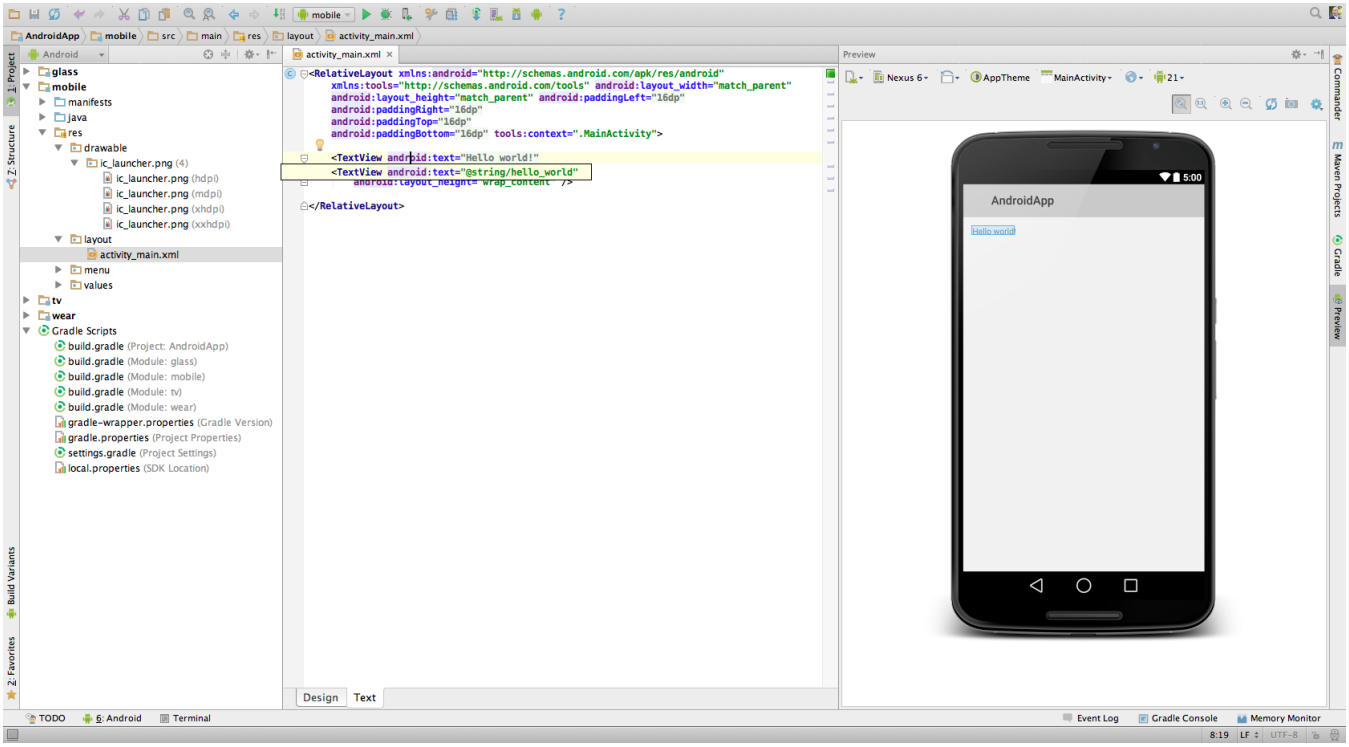

Рис. 2.12. Інтерфейс додатку в режимі Text View

Легко вибрати і переглянути зміни макета для різних пристроїв, зображень щільності відображення, режимів користувача інтерфейсу, місцях, і версій Android (мульти-API версія послуг) (рис. 2.13).

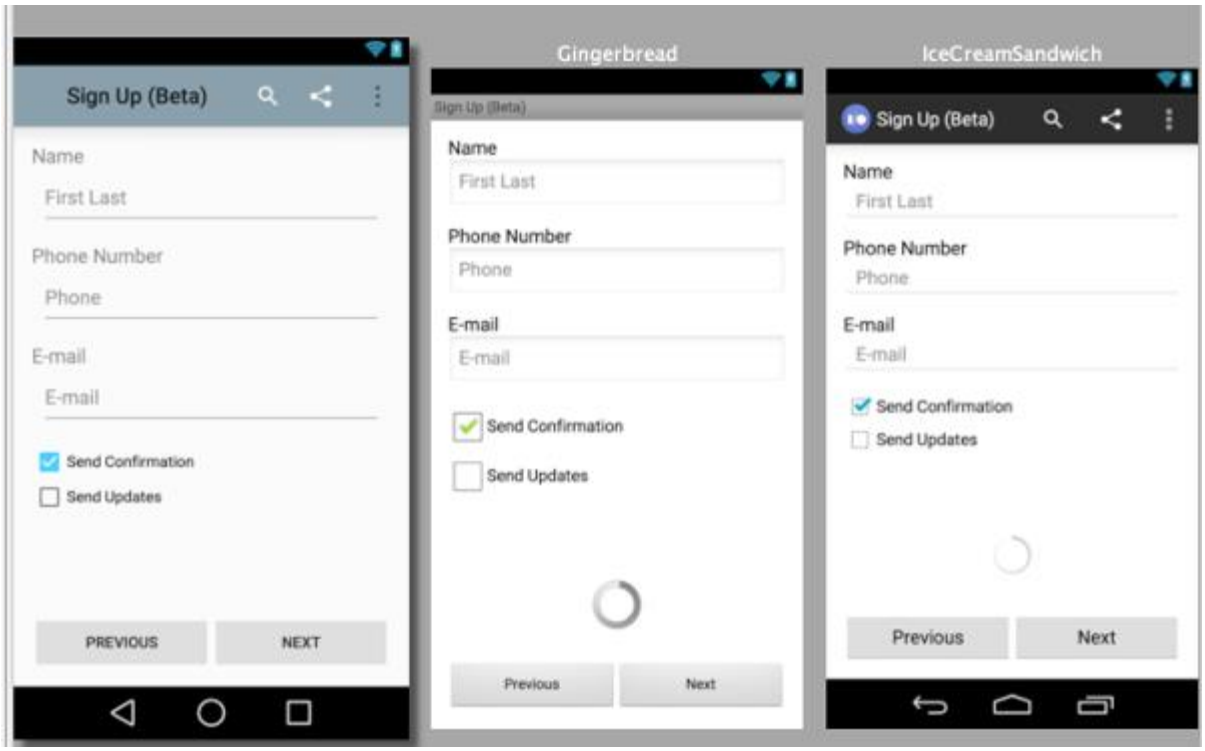

Рис. 2.13. Multi-API рендерінг

2.6. Розробка архітектури проекту

Мобільна розробка поділяється на два етапи: верстка статичних екранів в *XML* та програмування бізнес логіки на *Java*.

Розробка здійснюється у популярній *IDE* від *Google – Android Studio*.

Затверджений дизайн «нарізається» на окремі малюнки, з яких згодом складається *xml*-розмітка для додатку. У результаті створюється код, який можна переглядати за допомогою мобільного телефону. А типові сторінки згодом будуть використовуватися як шаблони.

Потім *XML* файли наповнюються потрібною інформацією динамічно за допомогою *Java*[15].

Java - об'єктно-орієнтована мова програмування. Програми *Java* зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь якій віртуальній *Java*-машині незалежно від комп'ютерної архітектури
(рис.2).

Java - дуже гнучка мова з відкритим вихідним кодом. Для розробки програми існують архітектурні підходи до проекту, які вирішують проблему розробки стандартного програмного забезпечення. Шаблони - це абстрактні схеми, вони не є кодом, але вони допомагають розділити логіку програми на певні модулі. При використанні шаблону проекту, цей шаблон адаптується у відповідності зі своїми певними потребами (рис.2.14).

Модель являє собою дані, з якими оперує додаток. Це можуть бути як дані бази даних, так і будь-яка інша структура даних, що описує деякі об'єкти системи і їх стан.

Вид (*View*) являє собою компонент системи для відображення стану моделі в зрозумілому людині поданні. Це можуть бути діалоги, форми та інші візуальні (наприклад, синтезатор мови) засоби взаємодії людини з системою.

Вид не змінює дані безпосередньо (режим тільки читання), дані змінюються за допомогою контролера.

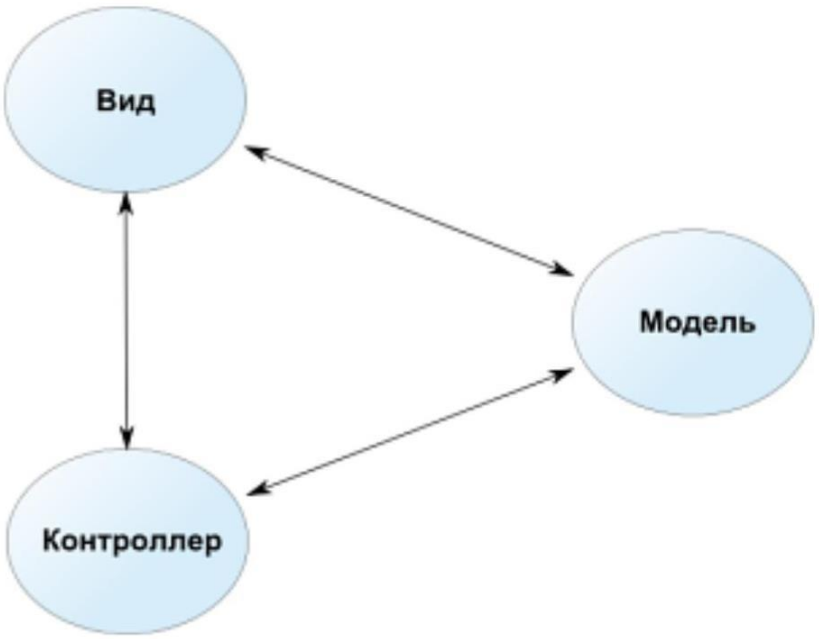

Рис. 2.14. Структура шаблону Модель - Вид - Контролер Модель (*Model*)

Контролер (*Controller*) є засобом, за допомогою якого користувачі взаємодіють з системою. Це може бути клавіатура, маніпулятор миша і т. Д. А також є керуючим елементом для обміну даними та повідомленнями між видом і моделлю.

Фактично, зв'язка виду і контролера є інтерфейсом користувача. Причому, якщо компоненти виду зазвичай можна повторно використовувати в інших компонентах системи, то контролер часто є специфічним для даного конкретного випадку

Модель не залежить ні від виду, ні від контролера, що дозволяє одночасно будувати різні інтерфейси користувача для взаємодії з однією і тією ж моделлю даних. Наприклад, можна зробити як *Java SWT* або *SWING* десктопних програм, так і *WEB* додаток для взаємодії одними і тими ж даними[11].

Розрізняють як пасивний, так і активний режими роботи з моделлю даних.

При пасивному режимі дані на клієнтській стороні перечитуються при виконанні деяких дій користувача, наприклад, запиту на виведення інформації про об'єкти.

При активному режимі вводяться додаткові обробники і слухачі повідомлень, які посилають компонентам рівня подання повідомлення про те, що дані моделі були змінені і потрібне оновлення даних на стороні клієнта "*View*".

Практичне застосування шаблону *MVC* має як позитивні, так і негативні сторони. Шаблон *MVC* може призвести до збільшення складності по забезпеченню синхронізації даних та подання до простих додатках.

Висновки до розділу

Обрано оптимальну модель життєвого циклу та описано її основні етапи. Розглянуто методологію розробки додатку для взаємодії з *BLE* пристроями, виділено основні задачі, що підлягають подальшому аналізу та виконанню.

Також описані середовище розробки, мова та основні етапи розробки додатку для ОС Android.

РОЗДІЛ 3

АНАЛІЗ ВИМОГ ТА СТВОРЕННЯ ДОДАТКУ

3.1. Аналіз та визначення задач створення додатку

Завдання дипломного проекту розробити додаток для роботи з віддаленим сервером з функцією шифрування даних. Мета цього додатку спростити передачу даних між користувачами певної організації, що працюють у групах, та спростити їх доступ до спільних ресурсів,

забезпечуючи цим досить серйозний захист. Реалізувати захист планується шляхом подвійної авторизації та шифрування файлів. Програма взаємодіє не на пряму з сайтом, а з його базою даних. Сайт використовує базу даних *mySql*. При компіляції програми, створюється файл з розширенням *.apk*, який потім може бути відкритий за допомогою архіватору, після чого можна отримати доступ до ресурсів програми. Також можна декомпілювати і сам файл *.dex*, у якому знаходиться байт-код програми. Через це клієнт не повинен мати доступу до бази даних на пряму. Доступ реалізовано через *POST*-запити. Це забезпечує достатній рівень безпеки. Скрипти на сервері оброблюють всі запити та відповідно на них відповідають. Обмін інформації відбувається у формати *JSON* (*JavaScript Object Notation*) - це текстовий обмін даними, який походить від *JavaScript*. Проте він використовується у багатьох мовах програмування через свою лаконічність, зручність та швидкість обробки.

3.2 Структура програми

Нижче приведений список файлів та папок з написаним кодом, створених для нього макетів та ресурсів.

Source :

- *.grandle*
- *.idea*
- *.app*
- *.build*
- *gradle*
- *build.gradle*
- *gradle.properties*
- *HideMyFile-master.iml*
- *LICENSE*
- *local.properties*
- *README.md*
- *settings.gradle*

Основне призначення додатку - повне шифрування довільних файлів за допомогою стійких алгоритмів шифрування даних. Шифрування відбувається за допомогою ключа користувача, а доступ до списку файлів та усіх можливостей додатку захищено 4-значним пін-кодом. Вираз “повне шифрування” означає побітове шифрування файла повністю, а не лише його заголовку - шифрування файлів великого розміру може зайняти деякий час, але, використовуючи саме такий метод, файл гарантовано буде захищений від дешифрування.

Основні переваги. Стійкий алгоритм шифрування. Для шифрування файлів використовується симетричний алгоритм *AES* з довжиною ключа у 128 біт.

Advanced Encryption Standard (*AES*), також відомий під назвою *Rijndael* — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу *AES* і прийнятий як американський стандарт шифрування урядом США. Вибір припав на *AES* з розрахуванням на широке використання і активний аналіз алгоритму, як це було із його попередником, *DES*. Державний інститут стандартів і технологій (англ. *National Institute of Standards and Technology*, *NIST*) США опублікував попередню специфікацію *AES* 26 жовтня 2001 року, після п'ятилітньої підготовки. 26 травня 2002 року *AES* оголошено стандартом шифрування. Станом на 2009 рік *AES* є одним із найпоширеніших алгоритмів симетричного шифрування.

Опис алгоритму

AES має фіксовану довжину у 128 біт, а розмір ключа може приймати значення 128, 192 або 256 біт. Через фіксований розмір блоку *AES* оперує із масивом 4×4 байт, що називається станом (версії алгоритму із більшим розміром блоку мають додаткові колонки).
Для ключа 128 біт алгоритм має 10 раундів у яких послідовно виконуються операції:

– *subBytes*();

– *shiftRows*();

– *mixcolumns*() (у 10-му раунді пропускається);

– *xorRoundKey*().

У процедурі *SubBytes*, кожен байт в *state* замінюється відповідним елементом у фіксованій 8-бітній таблиці пошуку, *S*; *b ij* = *S* (*a ij*) (рис.3.1).

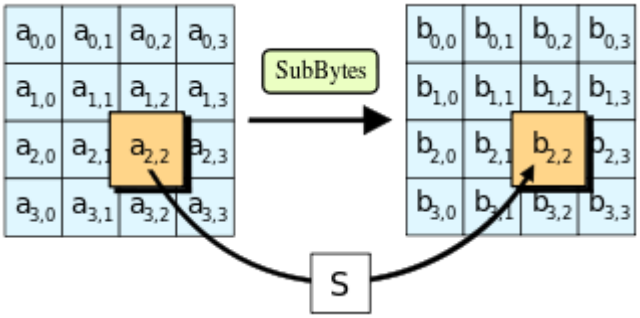

Рис.3.1. Вигляд процедури *SubBytes*()

У процедурі *ShiftRows* , байти в кожному рядку *state* циклічно зсуваються вліво. Розмір зміщення байтів кожного рядка залежить від її номера (рис.3.2).

У процедурі *MixColumns*, кожна колонка стану перемножується з фіксованим многочленом *c* (*x*) (рис.3.3)

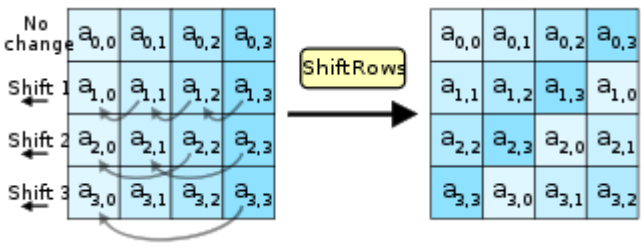

Рис. 3.2. Вигляд процедури *ShiftRows*()

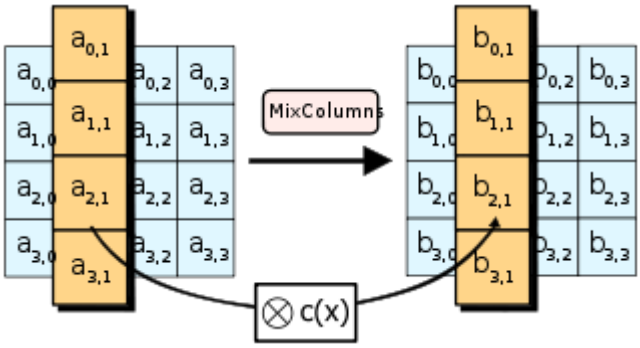

Рис. 3.3. Вигляд процедури *MixColumns*

У процедурі *AddRoundKey*, кожен байт стану об'єднується з *RoundKey* використовуючи операцію *XOR* (рис.3.4).

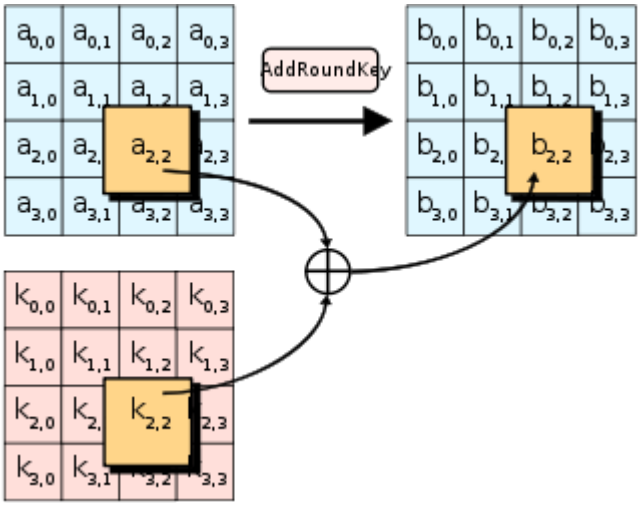

Рис.3.4. Вигляд процедури *AddRoundKey*

Довільні паролі для різних файлів.

Користувач має змогу встановити різні паролі для кожного файлу (або однакові). Це дає змогу захистити дані, навіть після компрометації одного з паролів. Також перевагою додатку є той факт, що паролі для шифрування файлів не зберігаються у будь-якому сховищі даних, а вираховуються (створюються) кожного разу у процесі розшифрування. Алгоритм створення ключа розшифрування на основі паролю користувача представлений ліворуч (рис. 3.5).

Рис. 3.5. Алгоритм створення ключа шифрування

Захист додатку 4-значним пін-кодом.

Користувач має змогу захистити доступ до процесу шифрування/розшифрування, перегляду зашифрованих файлів та інших функцій додатку за допомогою 4-значного цифрового паролю. За змовчуванням пароль додатку: 0000. Користувач може у будь-який момент змінити стандартний пароль у розділі “Налаштування”.

Чотиризначний пароль зберігається у вигляді одного входження у сховищі даних *Android Shared Preferences*. Дані зберігаються у папці “/*data*/*data*/…”, що дає змогу обмежити їх перегляд неавторизованому користувачу, проте при наявності *ROOT* прав, користувач все ж таки має змогу переглядати усі файли додатку.

Для вирішення цієї проблеми додаток має ще одну особливість: 4-значні паролі ніколи не зберігаються у відкритому вигляді, лише у вигляді гешу, взятого з первісного паролю. Алгоритм авторизації наведений на (рис.3.6.).

Рис.3.6. Алгоритм автоматизації

3.3. Презентація готового додатку

Користувач має змогу зашифрувати абсолютно будь-який файл будь-якого розміру (фото, картинку, аудіо, відео, текст, виконуючий файл і т.д.). Це дозволяє використовувати один додаток для усіх потреб (рис. 3.7).

Щоб отримати доступ до додатку, в якому вже знаходяться зашифровані файли потрібно пройти авторизацію. Процес авторизації зображено на (рис.3.8).

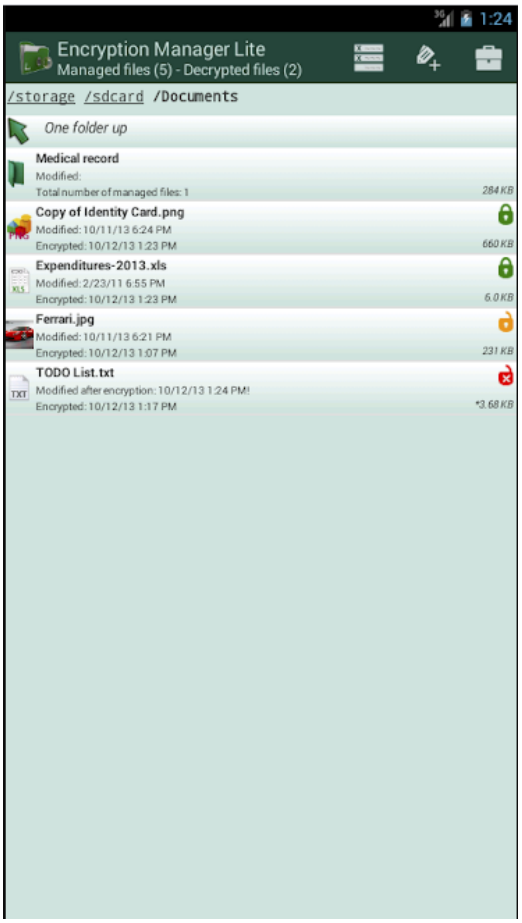

Рис. 3.7. Головний екран - список зашифрованих файлів

Але ввівши чотиризначний пароль, потрібно ввести ще і пароль до зашифрованих файлів, відкриваючи їх окремо. Ця функція дає можливість збільшити безпеку збереження даних майже в декілька разів.

Тому, якщо хтось і може підібрати чотиризначний пароль для авторизації в додатку, то прочитати якийсь файл, чи відкрити фото він точно не зможе.

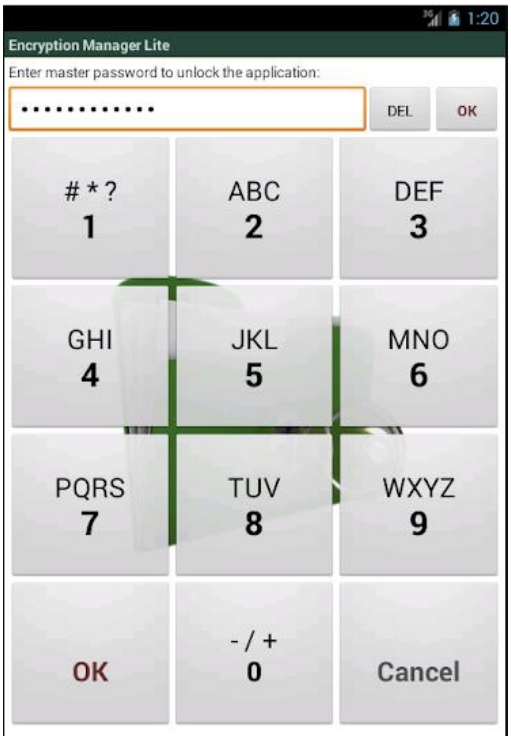

Рис. 3.8. Екран авторизації до додатку

Також до стандартного ділового вікна операційної системи додається можливість зашифрувати любий файл. Діалогове вікно шифрування зображено на (рис.3.9).

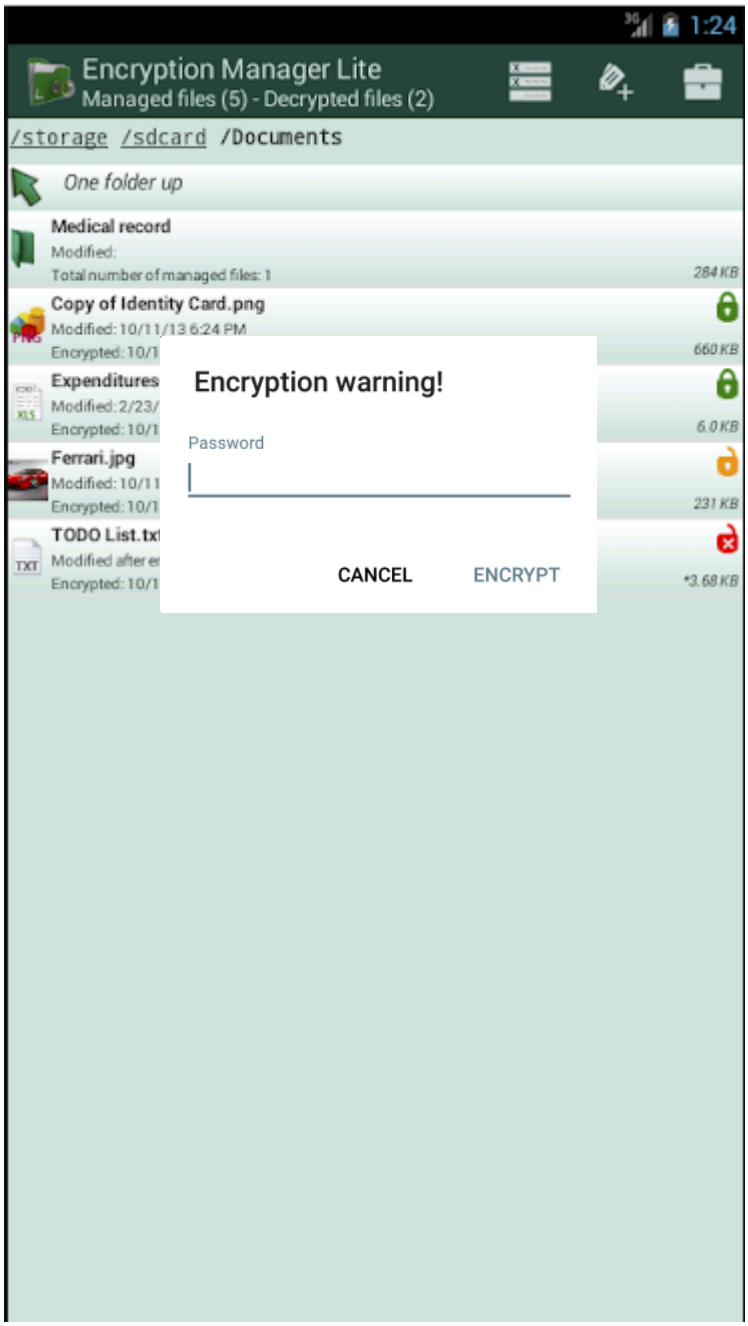

Рис. 3.9. Діалогове вікно шифрування

Звісно, кожна програма, в даному випадку це є додаток для шифрування даних із подвійною авторизацією, повинен мати і деякі налаштування. Під час реалізації додатку було вирішено додати основні параметри налаштувань додатку (рис.3.10).

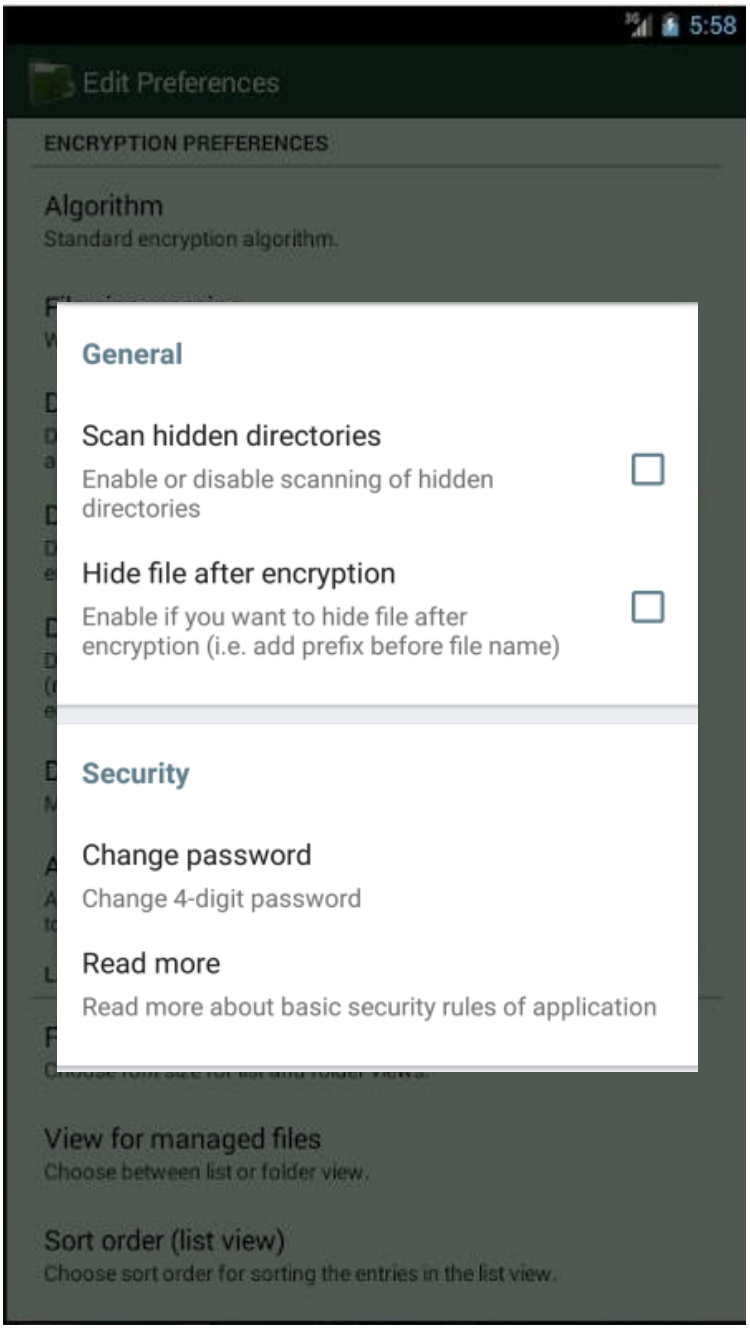

Рис. 3.10. Основні параметри налаштувань додатку

Висновки до розділу

В даному розділі був проведений аналіз завдання та потреб для створення додатку. Була створена структура файлів додатку, яка буде забезпечувати необхідне його функціонування. Щоб забезпечити реалізацію подвійної авторизації потрібно було прочитати немало літератури по шифруванню, а також вибрати самі необхідні функції для подвійної авторизації. Тестування додатку пройшло успішно. Результат роботи готового додатку представлений на рисунках.

ВИСНОВКИ

У рамках виконання дипломного проекту розроблений мобільний додаток для платформи Android для взаємодії з віддаленим сервером. Досліджено стан технологій створення мобільних додатків. Обрана платформа Android, середовище розробки Android Studio, мова Java. Розроблено технічне завдання, структура, інтерфейс та дизайн додатку. Реалізовано найбільшн ефективний функціонал та дизайн мобільного додатку. Готовий додаток було протестовано.

Тож, були досягнуті всі поставлені задачі у повному обсязі.

Додаток може працювати з віддаленим сервером, та обмінюватися з ним інформацією у POST-запитах. Всі запити відбуваються у паралельному потоці. У разі виникнення помилок програма повідомляє про це користувача за допомогою діалогових вікон. Програма протестована на планшетному комп'ютері Ainol Novo 9 Spark (Android 4.4), смартфоні Sony Ericsson Xperia Arc (Android 5.1) та ще одному планшетному комп'ютері Ainol Novo 7 (Android 5.0). На всіх пристроях програма коректно встановлювалась та працювала. За рахунок використаних алгоритмів збереження мережного трафіку, програма швидко отримувала потрібну інформацію від серверу навіть при повільній швидкості підключення.

У перспективі можливо покращити інтерфейс користувача - можна додати більше функції у випадаюче меню. Також можна максимально ефективно використовувати екран якщо використовується пристій з великим екраном. Для цього потрібно додати декілька макетів та задати їм відповідну поведінку в залежності від розміру екрану. Також замість стандартних макетів можна використовувати фрагментацію інтерфейсу. Ця технологія дозволяє більш гнучко керувати розміщенням елементів на екрані користувача.

Для збільшення швидкості роботи у програму можна додати кешування файлів, які програма отримує від серверу. Кешування у Android відбувається за допомогою запису потрібної інформації до бази даних SQLite. SQLite -

база даних, яка для зберігання інформації використовує текстові файли. При запуску програмі потрібно буде лише зрівняти останній ідентифікатор листа з останнім ідентифікатором листа на сервері, та, якщо вони різні, підвантажити потрібні заголовки. При першому запуску програми будуть завантажені лише 30 останніх заголовків. Якщо користувачу потрібно отримати доступ до старіших листів, йому достатньо прокрутити блок заголовків униз. Під час прокрутки старіші файли будуть кешовані та надалі доступні користувачу без потреби взаємодії з мережею. Кешування всіх файлів може вимагати забагато пам’яті. Тому лише після того, як користувач відкриє окремий файл, програма його завантажить з серверу, збереже та покаже користувачу. Тому якщо у користувача виникне потреба переглянути уже прочитаний файл, це можна буде зробити без підключення до мережі.