

1. АРХІТЕКТУРА ТА КОМПОНЕНТИ МОБІЛЬНИХ ПЛАТФОРМ

1.1. Огляд операційного середовища Android

Платформа Android є розробкою групи Open Handset Alliance, яка поставила перед собою мету створити інноваційну модель телефону. Ця група на чолі з Google об'єднує операторів мобільних мереж, виробників телефонів і компонентів, розробників програмних рішень і постачальників послуг, а також маркетингові компанії. Таким чином, платформа Android є ядром розробки програмного забезпечення з відкритим кодом.

Першим телефоном, який просувався на ринок оператором T-Mobile і працював на платформі Android, був пристрій G1 від HTC. Цей телефон вийшов на ринок через рік після перших згадок про нього. Для розробки програмного забезпечення використовувався SDK, який постійно оновлювався. Напередодні випуску G1 команда Android анонсувала SDK v2.0, після чого почали з'являтися прикладні програми для нової платформи.

Щоб стимулювати інновації, Google спонсорувала два «Конкурси розробників для Android», переможці яких отримали мільйони доларів. Через кілька місяців після виходу G1 відкрився сайт Android Market, звідки користувачі могли завантажувати додатки прямо в свій телефон. Всього за півтора року нова мобільна платформа вийшла на арену.

1.2. Огляд структури Android-прикладної програми

За широтою можливостей платформа Android не поступається операційним системам настільних ПК. Це багаторівневе середовище на основі ядра Linux з великими функціональними можливостями. У підсистему інтерфейсу, призначеного для користувача, входять:

- вікна;
- подання;
- віджети для відображення загальних елементів, таких, як поля, що редагуються, списки та списки, що розгортаються.

В Android вбудовано браузер з движком WebKit з відкритим вихідним кодом, який лежить в основі браузера Safari мобільного телефону iPhone.

Android має широкий спектр можливостей для підключення, бездротового зв'язку з використанням Wi-Fi, Bluetooth та протоколів передачі даних через стільникову мережу (GPRS, EDGE, 3G і ін.). В Android активно використовуються сервіси, які надаються Google; наприклад, в своїх прикладних програмах можна посилатися на Google Maps для позиціонування пристрою. В стек про-

грамного забезпечення Android входить також підтримка сервісів, заснованих на визначенні місця розташування (наприклад, GPS), та акселерометрів, хоча не усі пристрої на цій платформі оснащені необхідним обладнанням. Присутня також підтримка відеокамери.

Через обмеження ресурсів мобільні пристрої завжди мали великі проблеми з обробкою графіки/мультимедіа та способами зберігання даних, на відміну від персональних комп'ютерів. Для вирішення даної проблеми платформа Android пропонує вбудовану підтримку 2-D і 3-D графіки, з використанням бібліотеки OpenGL. Для забезпечення зберігання даних платформа Android використовує популярну базу даних з відкритим вихідним кодом SQLite. На рис. 2.1 зображена спрощена схема рівнів програмного забезпечення Android.

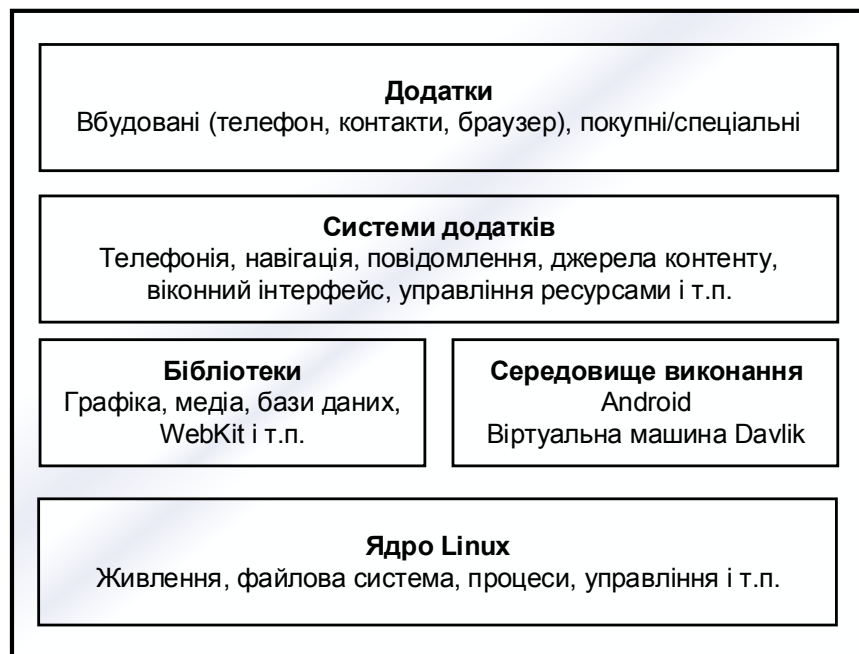


Рисунок 1.1 – Рівні програмного забезпечення Android

1.3. Огляд віртуальних машин Android

1.3.1. Віртуальна машина Dalvik

Операційна система Android працює поверх ядра Linux. Для створення Android-додатків спочатку використовувалася мова програмування Java, а потім виконувалися прикладні програми у віртуальній машині (VM). Необхідно звернути увагу на те, що віртуальна машина – це не віртуальна машина Java (JVM), а відкрита технологія Dalvik Virtual Machine. При запуску програми Android створюється і запускається окремий екземпляр Dalvik VM, який, в свою чергу, розташований в межах керованого ядром Linux процесу, як показано на рис. 1.2.



Рисунок 1.2 – Dalvik VM

Dalvik є віртуальною машиною в межах операційної системи Android від Google, яка виконує прикладні програми, що написані для Android. Dalvik є невід’ємною частиною стека програмного забезпечення Android в ОС Android версії 4.4 «KitKat» та більш ранніх версій, які зазвичай використовуються в мобільних пристроях, таких, як мобільні телефони та планшетні комп’ютери, а останнім часом на таких пристроях, як смарт-телевізори. Dalvik є програмним забезпеченням з відкритим вихідним кодом, написаним Dan Bornstein, який назвав його на честь рибальського селища Dalvik в Ісландії.

Програми для Android, як правило, написані на Java та скомпільовані в байт-код для віртуальної машини Java, яка потім транслюється на Dalvik байт-код і зберігається в .dex (Dalvik Executable) і .odex (оптимізований Dalvik Executable) файлах. Компактний формат Dalvik Executable призначений для систем, які обмежені з точки зору пам’яті та швидкості процесора.

Правонаступником Dalvik є Android Runtime (ART), який використовує той самий байт-код та .dex файли (але не .odex файли), з послідовністю, спрямованою на підвищення продуктивності для кінцевих користувачів.

1.3.2. Віртуальна машина ART

Android Runtime (ART) є середовищем виконання прикладних програм, яке використовується Android. ART виконує перетворення байт-коду програми в інструкції, які потім виконуються за допомогою середовища виконання пристрою.

В Android 2.2 «Froyo» з’явилися JIT-компіляція в Dalvik, оптимізація виконання прикладних програм за допомогою постійно профілюючих програм.

У той час як Dalvik інтерпретує частину в байт-кодi прикладної програми, виконання ART цих коротких відрізків байт-коду, яке називається «слідом», забезпечує значне підвищення продуктивності.

На відміну від Dalvik, ART вводить використання ahead-of-time (AOT) компіляцію шляхом компіляції цілих прикладних програм в машинний код при їх установці. Усуваючи інтерпретації Dalvik на основі JIT-компіляції, ART підвищує загальну ефективність виконання та знижує енергоспоживання, що приводить до підвищення автономії батареї на мобільних пристроях. Також ART забезпечує більш швидке виконання програм, поліпшений розподіл пам'яті та механізми збирання сміття (GC), нові можливості налагодження прикладних програм, а також більш точне профілювання прикладних програм високого рівня.

Для забезпечення зворотної сумісності ART використовує такий самий вхідний байт-код, що і Dalvik, який надається через стандартні .dex файли як частина APK файлів, в той час як .odex файли замінюються Executable and Linkable Format (ELF) файлами (рис. 1.3).

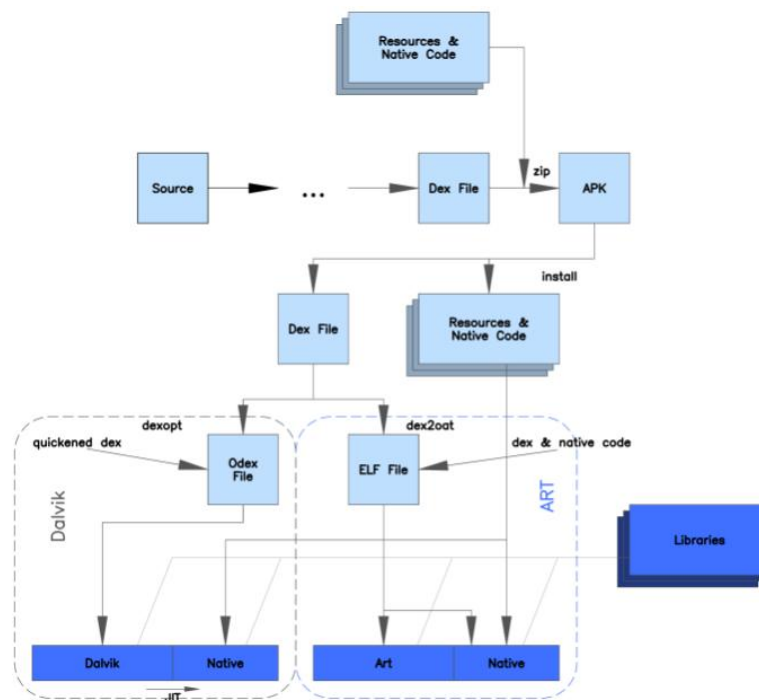


Рисунок 1.3 – Порівняння архітектури Dalvik та ART

Після того як прикладна програма буде зібрана та скомпільована на пристрої, утиліта dex2oat запускається з виконуваного ELF-файлу. У результаті ART усуває відмінності виконання прикладних програм, накладні витрати, пов'язані з інтерпретацією Dalvik, та трасування на основі JIT компіляції.

ART вимагає додаткового часу для компіляції при установці прикладної програми, а прикладні програми вимагають трохи більше місця для зберігання (як правило використовується флеш-пам'ять) скомпільованого коду.

В Android 4.4 «KitKat» з'явилася технологія попереднього огляду ART, в тому числі як альтернативного середовища виконання та збереження, замість Dalvik, як віртуальної машини поза вибором. У подальшому в Android 5.0 «Lollipop» Dalvik був повністю замінений на ART.

Android-прикладна програма містить елементи одного або декількох наступних типів.

Дії (Activities)

Прикладна програма з графічним інтерфейсом реалізується за допомогою дії. Коли користувач вибирає прикладну програму на головному екрані або екрані запуску програм, він викликає дію.

Сервіси (Services)

Сервіси використовуються для програм, які працюють протягом тривалого часу, таких, як мережевий монітор або перевірка оновлень.

Джерела даних (Content providers)

Джерело даних можна уявити собі як сервер баз даних. Його завдання – управління доступом до даних, що зберігаються, наприклад до баз даних SQLite. Якщо прикладна програма зовсім проста, джерело даних створювати не обов'язково. Якщо ви пишете більш складну прикладну програму або прикладну програму, в якій до даних звертається кілька дій або прикладних програм, джерело даних є засобом організації доступу до вашої інформації.

Приймачі (Broadcast receivers)

Android-прикладна програма може запускатися для обробки елемента даних або для реагування на події, наприклад на отримання текстового повідомлення.

Прикладна програма для Android розгортається на пристрої разом з файлом AndroidManifest.xml. Цей файл містить необхідну інформацію про конфігурацію, яка дозволяє правильно встановити прикладну програму на пристрої. Він містить також необхідні імена класів та типи подій, які може обробляти прикладна програма, та дозволи, необхідні для її роботи. Так, якщо з прикладною програмою потрібен доступ до мережі (наприклад, щоб завантажити файл), відповідний дозвіл має бути явно вказаний у файлі-маніфесті. Цей конкретний дозвіл можуть мати багато прикладних програм. Такий захист шляхом декларування допомагає зменшити ймовірність пошкодження пристрою через некоректно написану прикладну програму.

1.4. Керування ресурсами мобільних пристроїв

Ядро, що використовується в Android, є модернізованим ядром серії Linux для забезпечення виконання деяких особливих потреб мобільних платформ. Функції ядра в основному поширюються на драйвери, керування живленням, засоби керування та коригування обмежених можливостей Android. Функції керування живленням мають вирішальне значення для мобільних пристроїв.

Ядро Android знаходиться у вільному доступі. Всі зміни можна відстежувати через загальнодоступне сховище Android. Є кілька ядер, доступних в репозиторії. Деякі апаратні специфічні ядра для платформ, таких, як MSM7xxx, Open Multimedia Application Platform (OMAP) та Tegra, також доступні в сховищі.

Зміни, які вносяться до базової версії ядра, можна розділити на такі: виправлення помилок, засоби для підвищення простору для користувача (low memory killer, binder, ash mem, logger і т. д.), нова інфраструктура (особливо wake locks), підтримка нових SoCs (msm7k, msm8k і т. д.) та плат/пристроїв. В майбутньому специфічні ядра Android та базові ядра Linux повинні об'єднатися, але цей процес йде повільно і забере деякий час.

Android виділяє кожній прикладній програмі, яка призначена для користувача, окремий адресний простір. Програми, що працюють в Dalvik VM або ART, періодично переходять в сплячий режим або режим очікування. Це важливо, тому що поза вибором Android намагається перевести систему в режим сну або в режим очікування якнайшвидше.

При цьому екран залишається включеним, або CPU не спить, щоб швидше реагувати на пробудження. Інструменти Android, які використовуються для вирішення цього завдання, називаються блокіраторами засипання (wakelocks).

Функції wakelock можуть бути отримані компонентами ядра або адресним простором процесу користувача.

Інтерфейс користувача для створення блокіраторів використовує файл `/sys/power/wakelock`, в якому записується ім'я нового блокіратора. Для зняття блокування процес записує ім'я файлу в `/sys/power/wakeunlock`. Для блокування можна вказати час, після якого воно спрацює автоматично. Всі системи, які використовують на даний час сервіс блокування, вказуються у файлі `/proc/wakelocks`.

Інтерфейс ядра для блокіраторів пробудження дозволяє вказати, чи повинен wakelock запобігати низькому енергоспоживанню чи припиненню роботи системи. WakeLock створюється процесом `wakelockinit()` та вилучається `wakelockdestroy()`. Створений блокіратор може бути запущений процесом

`wakelock()` та розблокований `wakeunlock()`. Як і в просторі користувача, можна визначити тайм-аут для блокування. Концепція блокіраторів глибоко інтегрована в Android у вигляді драйверів; багато прикладних програм інтенсивно використовують їх.

Клас `PowerManager` – це службовий клас, який дозволяє віртуальним машинам Андроїд отримати доступ до можливостей `WakeLock` драйвера керування живленням. `PowerManager` забезпечує чотири різних види блокування:

- **PARTIAL WAKE LOCK** – CPU не спить, навіть якщо кнопка живлення пристрою натиснута.
- **SCREEN DIM WAKE LOCK** – блокування екрана залишається увімкненим, але екран залишається сірим.
- **SCREEN BRIGHT WAKE LOCK** – екран блокується з нормальною яскравістю.
- **FULL WAKE LOCK** – блокування клавіатури та екрана зі звичайним підсвічуванням.

Тільки **PARTIAL WAKE LOCK** гарантує, що процесор повністю увімкнений, решта три типи дозволяють процесору засипати після того, як кнопка живлення пристрою натиснута. Як і блокування адресного простору прикладних програм, блокування, що надаються `PowerManager`, можуть бути об'єднані з тайм-аутом. Крім того, можна розбудити пристрій при включенні екрана.

Керування пам'яттю пов'язане зі змінами у базовому ядрі для поліпшення використання пам'яті в системах з невеликим обсягом оперативної пам'яті. Обидва механізми керування пам'яттю – **Anonymous SHared MEMory (ASHMEM)** та **Physical MEMory (PMEM)** – додали новий спосіб виділення пам'яті для ядра. `Ashmem` може бути використаний для розподілу загальної пам'яті віртуальної пам'яті, а `pmem` дозволяє розподіляти фізичну пам'ять.

Anonymous SHared MEMory забезпечує іменовані блоки пам'яті, які можуть бути поділені між кількома процесами. На відміну від звичайної розподіленої пам'яті, `ashmem` може бути звільнений ядром. Для використання `ashmem`, процес відкриває файл `/DEV/ashmem` та виконує функцію `mmap()`.

Physical MEMory (PMEM), наприклад, дозволяє драйверам або бібліотекам виділяти блоки фізично безперервної пам'яті. Цей драйвер був написаний для компенсації апаратних обмежень конкретного SoC – the **MSM7201A**.

Оптимізатор пам'яті **Low memory killer** є стандартним оптимізатором ядра Linux **out of memory killer (oom killer)** та використовує евристичні аналізатори для визначення «шкідливості» процесу, щоб мати можливість припинити його роботу. Вибираються процеси з найбільшою кількістю комірок з малою кількістю пам'яті. Така поведінка може бути незручною для користувача, оскі-

льки `oom killer` може закрити поточну прикладну програму користувача, коли наразі існують інші процеси в системі, які не впливають на пам'ять.

Драйвер `lowmemory` починає свою роботу заздалегідь, до виникнення критичної ситуації нестачі пам'яті. Він виконує аналіз процесів та інформує їх про можливість закриття, для того, щоб процеси могли зберегти свій стан. Якщо ситуація з пам'яттю погіршується, `lowmemory` починає завершувати процеси.

1.5. API мобільних прикладних програм

У той час як більшість Android-прикладних програм написані мовою Java, є багато відмінностей між API Java та Android API. Основною відмінністю є те, що Android не використовує віртуальну машину Java, замість неї використовуються дві інші: Dalvik або Android Runtime (ART).

Android-платформа не містить віртуальної машини Java (Java VM), на якій виконується Java-байт-код. Замість цього класи Java компілюються у власний формат байт-коду, розроблений спеціально для віртуальної машини Dalvik. На відміну від віртуальних машин Java, які використовують стеки, Dalvik VM є архітектурою на базі регістрів.

Dalvik має деякі специфічні особливості, які відрізняють її від інших стандартних віртуальних машин: VM було розроблено, щоб використовувати менше пам'яті.

Пул регістрів був змінений, щоб використовувати тільки 32-бітові індекси для спрощення інтерпретатора.

Стандартний Java байт-код виконує 8-розрядні команди стека. Локальні змінні величини повинні бути скопійовані зі стека операндів з окремими інструкціями. Dalvik замість цього використовує свій власний 16-бітний набір команд, який працює безпосередньо на локальних змінних величинах. Локальна змінна величина зазвичай використовує поле в 4 біти «віртуального регістра».

Оскільки байт-код, який завантажується віртуальною машиною Dalvik відрізняється від байт-коду Java Virtual Machine, та в зв'язку з певним способом завантаження класів віртуальною машиною Dalvik, відсутня можливість завантажувати `jar`-файли. Інший порядок також повинен бути використаний для завантаження Android-бібліотеки. При цьому зміст базового DEX-файлу має бути скопійовано на карту пам'яті перед завантаженням.

Як і у випадку Java SE, Android клас `System` дозволяє витягувати властивості системи. Тим не менш, деякі обов'язкові властивості, визначені за допомогою віртуальної машини Java, не мають ніякого значення або мають інше значення в Android.

Наприклад:

- властивість «`Java.version`» повертає 0, тому що не використовується на Android;

- властивість «`Java.specification.version`» незмінно повертає 0,9 незалежно від версії Android;

- властивість «`Java.class.version`» незмінно повертає 50 незалежно від версії Android;

- властивість «`User.dir`» має інше значення на Android;

- властивостей «`User.home`» та «`User.name`» не існує в Android.

Бібліотека класів Dalvik не збігається ні з Java SE, ні з бібліотекою класів Java ME (наприклад, Java ME класи, AWT або Swing не підтримуються). Замість цього Dalvik використовує свою власну бібліотеку, побудовану на підмножині реалізації Java – Apache Harmony.

Пакет java.lang. Поза вимогою вихідні потоки `System.out` та `System.err` не виводять нічого; розробникам рекомендується використовувати клас `Log`, який записує рядки в `LogCat` (це змінилося, у версії HoneyComb, де рядки виводяться в лог-консолі).

Графіка та бібліотека віджетів. Android не використовує `Abstract Window Toolkit` та бібліотеку `Swing`. Інтерфейс користувача побудований з використанням подань об'єктів. Android використовує фреймворк, подібний до `Swing`, а не `JComponents`. Проте, Android-віджети не є `JavaBeans`.

Менеджер компонування. На відміну від `Swing`, де менеджери компонування можуть бути застосовані до будь-якого контейнера віджета, поведінка подання Android кодується безпосередньо в контейнерах.

Платформа Android підтримує відносно велику підмножину бібліотек `Java Standard Edition 5.0`. Деякі функції були видалені, тому що вони просто не мають сенсу (наприклад, `print`), а інші є специфічними для Android (наприклад, інтерфейси користувача).

Платформа Android підтримує такі стандартні пакети `Java 2 Platform Standard Edition 5.0 API`:

- `java.io` – файлове введення/виведення;
- `java.lang` (крім `java.lang.management`) – підтримка мови та виключень;
- `java.math` – великі числа, округлення, точність;
- `java.net` – мережа введення/виведення, URL, сокети;
- `java.nio` – файлове та каналне введення/виведення;
- `java.security` – авторизація, сертифікати, відкриті ключі;
- `java.sql` – інтерфейси баз даних;

- java.text – форматування, природна мова, параметри сортування;
- java.util (включаючи java.util.concurrent) – списки, карти, набори, масиви, колекції;
- javax.crypto – шифри, відкриті ключі;
- javax.net – сокет фабрики, SSL;
- javax.security (крім javax.security.auth.kerberos, javax.security.auth.spi та javax.security.sasl);
- javax.sound – музика та звукові ефекти;
- javax.sql (крім javax.sql.rowset) – додаткові інтерфейси баз даних;
- javax.xml.parsers – XML-синтаксичний аналіз;
- org.w3c.dom (але не субпакетом) – DOM-вузли та елементи;
- org.xml.sax – простий API для XML.

Пакети, які не підтримуються платформою Android:

- java.applet;
- java.awt;
- java.beans;
- java.lang.management;
- java.rmi;
- javax.accessibility;
- javax.activity;
- javax.imageio;
- javax.management;
- javax.naming;
- javax.print;
- javax.rmi;
- javax.security.auth.kerberos;
- javax.security.auth.spi;
- javax.security.sasl;
- javax.swing;
- javax.transaction;
- javax.xml (кроме javax.xml.parsers);
- org.ietf. *;
- org.omg. *;
- org.w3c.dom. * (суб-пакети).

Існує ряд сторонніх бібліотек для Android SDK:

- org.apache.commons.codec – утиліти для кодування та декодування;

- org.apache.commons.httpclient – HTTP-аутентифікація, cookies, методи та протоколи;
- org.bluez – підтримка Bluetooth;
- org.json – формат JavaScript Object Notation.

1.6. Огляд архітектурних моделей для розробки мобільних прикладних програм

З кожним роком збільшується частка розробок прикладних програм для мобільних платформ. Однак типові мобільні прикладні програми відрізняються своїм життєвим циклом від настільних прикладних програм.

Мобільні прикладні програми часто вимагають більшої взаємодії з користувачем в порівнянні з десктопними і веб-програмами, як правило, очікуючи дій з боку користувача (наприклад, натискання кнопки), перш ніж відповісти користувачеві, оновленням інтерфейсом із зазначенням інформації, яка запитується. У розділі розглядається модель програмного забезпечення, яка фокусується на інтерактивних аспектах мобільних прикладних програм. Також проводиться аналіз загальної концепції розробки мобільних прикладних програм.

Модель Model View Controller (MVC) є популярним підходом для розробки мобільних прикладних програм. Зауважимо, що основна робота більшості мобільних прикладних програм полягає в отриманні даних зі сховища даних та оновлення інтерфейсу користувача з новою запитуваною інформацією на основі запитів користувача. Таким чином, має сенс пов'язати компоненти інтерфейсу користувача з компонентами сховища даних. Однак через те, що компоненти інтерфейсу користувача оновлюються частіше, щоб задовольняти змінні вимоги користувачів та новіших технологій, ніж компоненти сховища даних, необхідно ввести додаткові компоненти. Метою такої схеми є поділ компонентів на компоненти інтерфейсу користувача (View), компоненти, які забезпечують основні функціональні можливості (Controller), та дані (Model).

Як було сказано раніше, основними компонентами Android-прикладних програм є:

1. Активність – являє собою єдиний клас інтерфейсу користувача.
2. Сервіс – пов'язаний із завданнями, які будуть виконуватися у фоновому режимі потоків (наприклад, мережових операцій), не зачіпаючи при цьому компоненти інтерфейсу користувача.
3. Постачальник контенту – дозволяє зберігати дані в прикладній програмі з використанням SQLite бази даних або SharedPreferences (дані, що зберігаються у файлі XML на пристрої).

4. Широкомовний приймач – відповідає на повідомлення системи (наприклад, попередження про низький рівень заряду) та забезпечує повідомлення користувача.

Зазвичай стандартний проект мобільної прикладної програми включає такий набір компонентів:

- компоненти інтерфейсу користувача;
- сервісні компоненти (місце розташування пристрою для відслідкування місця розташування користувача за допомогою GPS, фонові завдання для отримання даних з інших служб та ін.);
- компоненти даних (для зберігання та вилучення налаштувань);
- віджет-компоненти (кнопки, поля для редагування тексту, поля для перегляду тексту, обробки кліків користувачів і т. д.).

Компоненти для інтерфейсу користувача складаються з пакета Активностей, які за своєю суттю нагадують компонент **View** моделі MVC. Кожна Активність оновлює і модифікує призначений для користувача інтерфейс протягом всього життєвого циклу програми. Компоненти послуг та даних схожі на компонент **Model** в тому, що вони спостерігають за поведінкою даних.

Вони також обробляють запити від **View** і **Controller** для отримання інформації про їхній стан та інструкції щодо зміни їх стану. Компоненти віджета аналогічні компонентам **Controller** у тому, що вони очікують введення з боку користувача (наприклад, натискання кнопки), а потім інтерпретують ці введення та оповіщають **Model** або **View** для того, щоб внести зміни.

Таким чином, можна зробити висновок, що на рівні стандартної мобільної прикладної програми модель MVC добре вписується в Android системи, і кожен компонент програми має своє відображення в **Model**, **View** і **Controller**. Однак при найближчому розгляді деталей все не так просто. Прикладом може служити компонент **Activities**, який є складовою частиною програми. Кожна активність складається з Java-файлу і відповідного файлу макета XML. Розробник визначає розташування UI активності в файлі XML та реалізує функціональність і поведінку у файлі Java (поділ користувача інтерфейсу з логікою прикладної програми). Коли використовуються віджет-компоненти, необхідно спочатку визначити віджети як елементи XML-файлу макета для створення екземпляра віджета і його атрибутів. Обробка поведінки віджета у відповідь на дії користувача також реалізуються у файлі Java. Проте розробник має доступ до атрибутів віджета з макета XML при реалізації його функціональних можливостей, і це порушує принцип поділу **View** з **Model/Controller** шаблону MVC.

Можливе рішення цієї проблеми полягає у використанні шаблону

Observer (Спостерігач) в MVC (рис. 1.4). Можна реалізувати інтерфейс Observer в компоненті Активності (View), щоб отримувати оновлення та повідомлення про зміни в логіці програми (стан Model). Коли відбувається зміна, логіка прикладної програми повідомляє всі Активності (спостерігачів) про зміну стану. Це виключає зв'язок між моделлю та поданням.

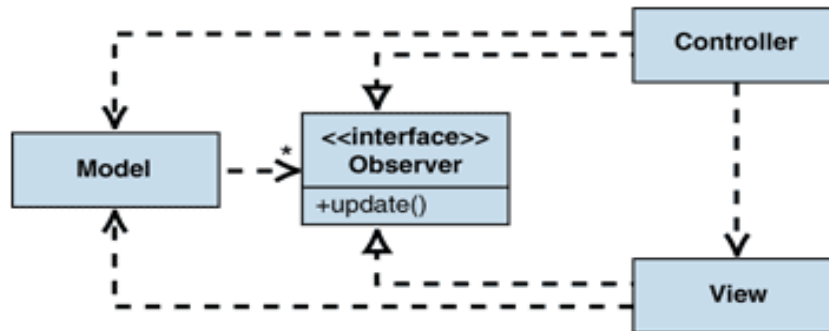


Рисунок 1.4 – Використання Observer у рамках MVC

Шаблон **Layered Abstraction** складається з ієрархії рівнів. На кожному рівні, ми маємо компоненти, які працюють разом в межах того ж рівня абстракції, і пов'язані з рівнем нижче для забезпечення функціональних можливостей вищого рівня.

Архітектура системи Android дотримується шаблону **Layered Abstraction**, як показано на рис. 1.5. Вона складається з чотирьох шарів абстракції з прямим зв'язком тільки між поточним шаром та шаром безпосередньо над або під ним:

1. Область застосування – цей шар складається з набору основних прикладних програм, а також розроблених прикладних програм.
2. Каркас прикладної програми – шар, який містить основні компоненти системи (Android діяльності, послуги, контент-провайдер, широкомовний приймач), а також інші компоненти системи.
3. Бібліотеки – складається з основних бібліотек системи Android, а також реляційної системи управління базами даних – SQLite.
4. Ядро Linux – містить драйвери пристроїв, які забезпечують зв'язок з апаратним забезпеченням пристрою.

З точки зору розробника прикладних програм з великою кількістю компонентів для інтерфейсу користувача було б доцільно використовувати шаблон MVC. Використання шаблону дає можливість зменшити кількість програмного коду, зберігаючи при цьому ядро бізнес-логіки прикладної програми, при зміні вимог користувача.

За допомогою шаблону Layered Abstraction, який є базовою моделлю для системи Android, можна будувати стандартні мобільні прикладні програми, які відсилають запити на роботу з сервісами та отримують повідомлення від них.



Рисунок 1.5 – Архітектура системи Android

1.7. Огляд засобів розробки мобільних прикладних програм

Комплект розробки програмного забезпечення Android (SDK) включає в себе повний набір інструментів розробника. [5] Він містить налагоджувач, бібліотеки, емулятор мобільного пристрою, оснований на QEMU, документацію, зразки коду та підручники (табл. 1.1).

Таблиця 1.1 – SDK Android

Розробник(и)	Google
Перший випуск	жовтень 2009 року
Стабільна версія	2.1.3, 15 серпня, 2016
Мова програмування	Java
Операційна система	Крос-платформне
Мова	Англійська
Тип	IDE, SDK
Сайт	developer.android.com/tools/sdk/eclipse-adt.html , developer.android.com/sdk/index.html

В даний час підтримуються платформи розробки, які працюють на операційній системі Linux (будь-який сучасний робочий стіл Linux), Mac OS X 10.5.8 або пізнішій версії та Windows XP або пізнішій версії. Станом на березень 2015 року SDK не доступна на Android, але розробка програмного забезпечення можлива за допомогою спеціалізованих програм для Android [6 – 8].

Приблизно до кінця 2014 року інтегрованим середовищем розробки (IDE), що офіційно підтримувалось, було Eclipse (рис. 1.6), яке пропонувало для розробки мобільних прикладних програм спеціальний плагін – інструменти розробки Android (ADT), хоча середовище IntelliJ IDEA (всі випуски) повністю підтримує Android-розробку «з ящика» [9] (рис. 1.7). NetBeans IDE також підтримує Android-розробку за допомогою плагіна [10] (рис. 1.8).

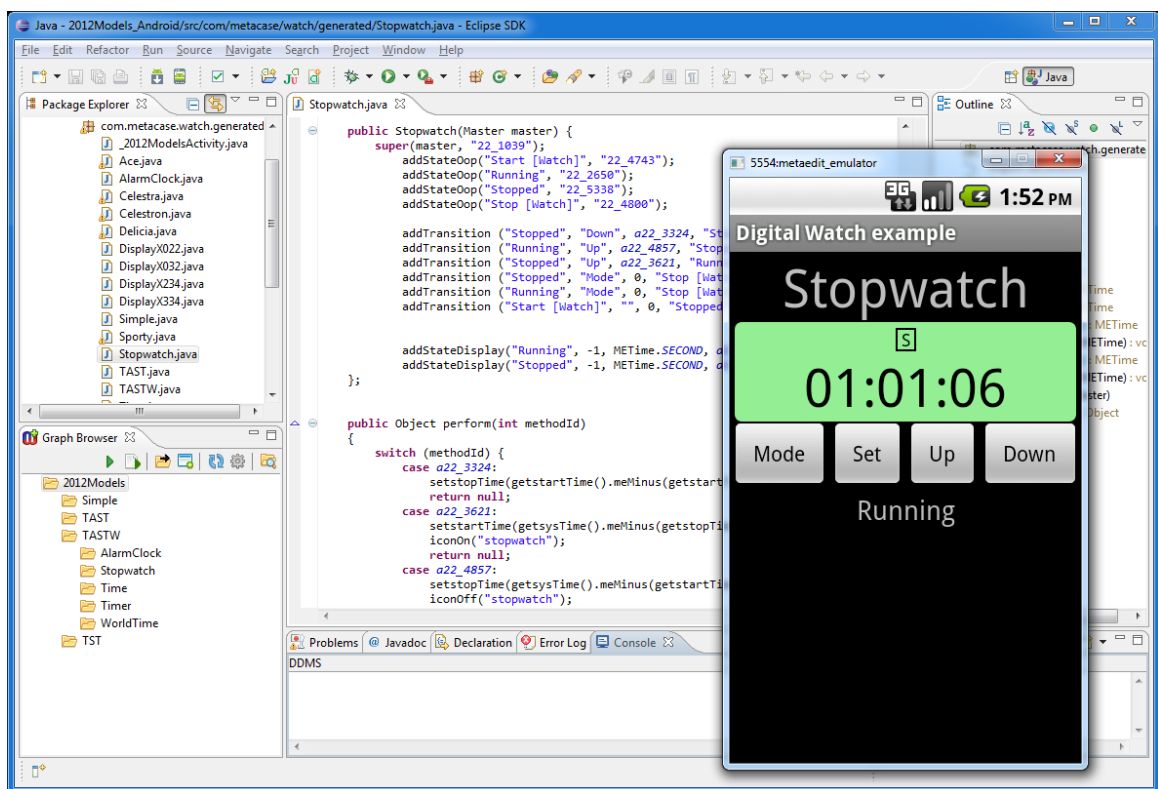


Рисунок 1.6 – IDE Eclipse

У 2015 році Google спільно з IntelliJ представили офіційну IDE від Google Android Studio [11]. Однак розробники можуть використовувати інші інтегровані середовища розробки. Крім того, розробники можуть використовувати будь-який текстовий редактор для редагування XML та Java-файлів, а потім використовувати інструменти командного рядка (комплект розробки Java і Apache Ant), щоб створити, скомпілювати та налагодити прикладні програми Android, а також керувати підключеними пристроями Android (наприклад, викликавши

перезавантаження, встановлення програмного забезпечення, видалення пакетів) [12].

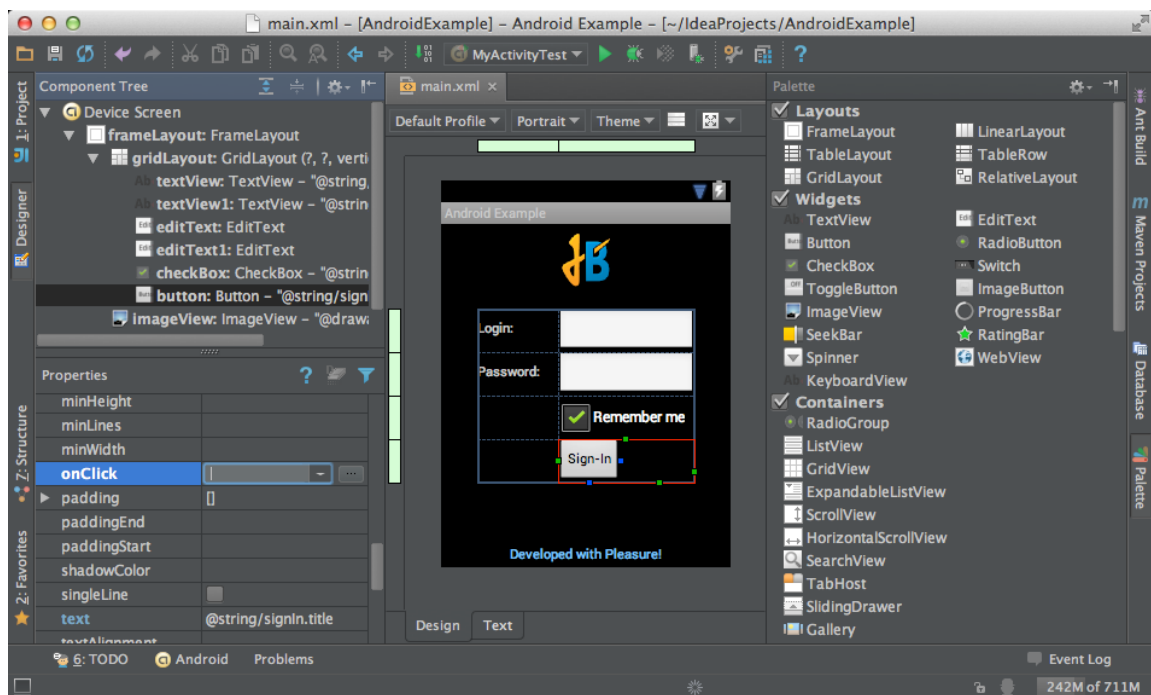


Рисунок 1.7 – IntelliJ IDEA

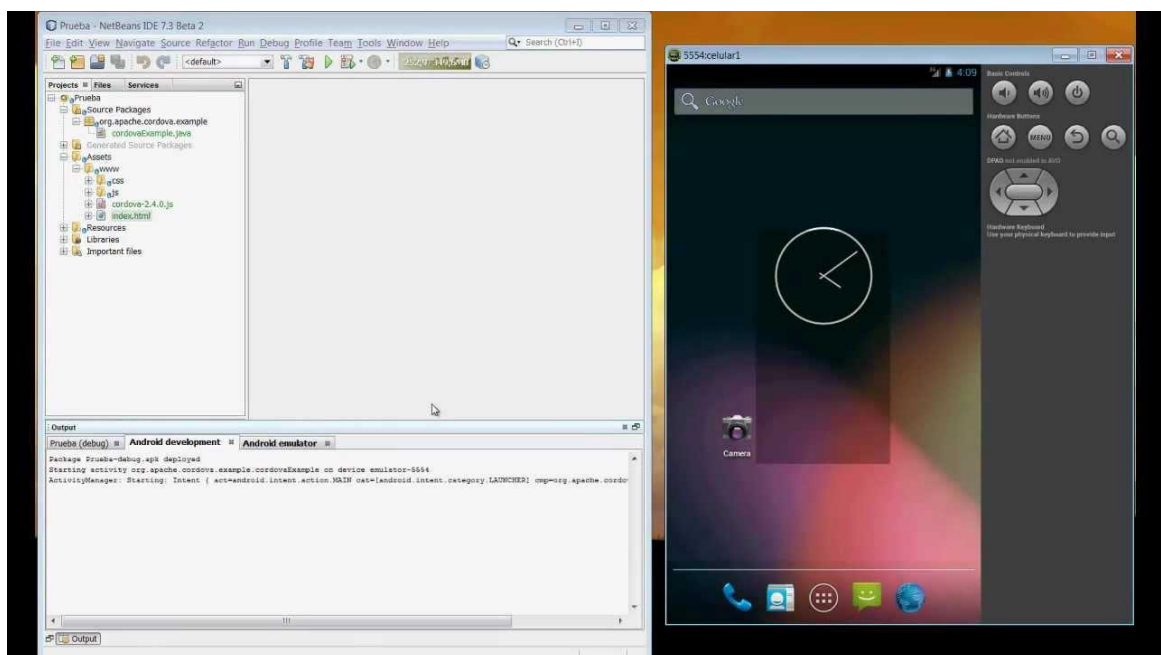


Рисунок 1.8 – NetBeans IDE

Покращення в SDK Android йдуть пліч-о-пліч з загальним розвитком Android-платформи. SDK також підтримує старі версії Android-платформи, якщо розробники хочуть запускати свої прикладні програми на старих пристроях.

Засоби розробки є компонентами, що завантажуються, таким чином, що після завантаження останньої версії та платформ, старі платформи та інструменти розробки також можуть бути завантажені для тестування сумісності [13].

Додатки Android упаковуються у файли у форматі **.apk** та зберігаються в папці **/data/app** на пристрої Android (папка доступна тільки для суперкористувача з міркувань безпеки). Пакети **apk** містять файли **.dex** [14] (байт-код, що виконується прикладною програмою для Dalvik VM), файли ресурсів та ін.

ADB (Android Debug Bridge). Це – клієнт-серверна прикладна програма, яка надає доступ до працюючого емулятора або пристрою (рис. 1.9). З її допомогою можна копіювати файли, встановлювати скомпільовані програмні пакети та запускати консольні команди. Використовуючи консоль, можна змінювати налаштування журналу та взаємодіяти з базами даних SQLite, які зберігаються на пристрої. У старих версіях SDK програма перебувала в папці **tools**, а тепер вона знаходиться в папці **platform-tools**.

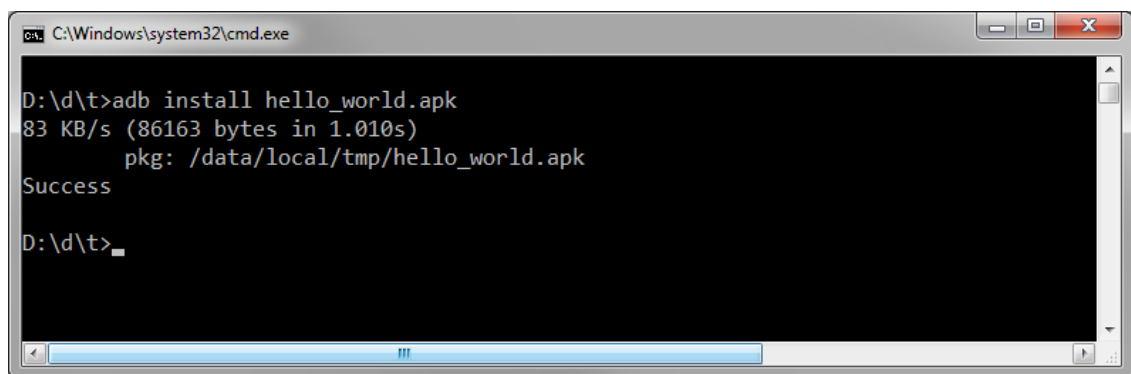


Рисунок 1.9 – Установлення пакета **apk** з використанням **adb**

ADB складається з трьох компонентів: фоновій служби (демона), що працює в емуляторі; сервісу, запущеного на комп'ютері розробника; клієнтської програми (на зразок DDMS), яка зв'язується зі службою через Сервіс.

Fastboot. Fastboot є діагностичним протоколом, який йде в комплекті з SDK та використовується в першу чергу для зміни флеш файлової системи через USB-з'єднання з комп'ютером. Fastboot вимагає, щоб пристрій запускався завантажувачем в режимі **Second Program Loader**, в якому виконуються тільки найосновніші ініціалізації обладнання. Після включення протоколу на самому пристрої, він буде приймати певний набір команд, які надіслані до нього через USB, використовуючи командний рядок. Деякі з найбільш часто використовуваних команд швидкого завантаження:

- **flash** – переписує розділ з двійковим зображенням, що зберігається на комп'ютері;

- `erase` – видаляє певний розділ;
- `reboot` – перезавантажує пристрій або в основну операційну систему, або назад в завантажувач;
- `devices` – відображає список всіх пристроїв (з серійним номером), підключених до комп'ютера;
- `format` – форматує певний розділ; файлова система розділу повинна розпізнаватися пристроєм.

Android NDK. Android Native Development Kit наведено в табл. 1.2. Бібліотеки, написані на C, C++ та інших мовах можуть бути скомпільовані в ARM, MIPS або в машинному коді x86 та встановлені на пристрій з використанням набору Android NDK. Рідні класи можна викликати з Java-коду, що виконується під Dalvik VM, використовуючи виклик `System.loadLibrary`, який є частиною стандартних класів Java в Android. [15, 16].

Таблиця 1.2 – Android Native Development Kit (NDK Android)

Розробник(и)	Google
Перший випуск	червень 2009 року
Стабільна версія	24.4.1, жовтень 2015
Мова програмування	C та C++
Операційна система	Крос-платформна
Доступна	Англійська
Тип	IDE, SDK
Сайт	developer.android.com/tools/sdk/ndk/index.html

Розроблені прикладні програми можуть бути скомпільовані та встановлені за допомогою традиційних інструментів розробки [17]. Тим не менш, відповідно до Android-документації, NDK не повинен використовуватися виключно для розробки прикладних програм тільки тому, що розробник вважає за краще програмувати на C/C++, тому що використання NDK збільшує складність прикладної програми, що не піде йому на користь.

ADB-налагоджувач дає права `root` в *Android Emulator*, що дозволяє завантажувати та виконувати програмний код, оптимізований під ARM, MIPS або x86-процесори. Машинний код може бути скомпільований з використанням GCC або Intel C++ Compiler на стандартному ПК. Запуск машинного коду на Android-платформі ускладнюється використанням нестандартної бібліотеки C (Libc, відомої як Bionic). Графічна бібліотека Android, яка використовується для арбітражу та контролю доступу до цього пристрою, називається Graphics Library Skia (SGL), вона випущена під відкритою ліцензією. Skia має движки для обох (Win32 та Unix) платформ, дозволяючи розвивати крос-платформні

прикладні програми. Skia також має графічний движок, що лежить в основі веб-браузера Google Chrome [18].

На відміну від додатків Java, основаних на використанні IDE, таких, як Eclipse, NDK оснований на командному рядку та вимагає введення команд вручну для компілювання, розгортання та налагодження прикладних програм. Деякі інструменти сторонніх розробників дозволяють інтегрувати NDK в Eclipse та Visual Studio.

Платформа ADK (Android Accessory Development Kit). ADK – це пристрій, що підтримує Android Open Accessory Protocol (рис. 1.10). ADK – це Arduino-сумісна платформа, що підключається до Android-пристрою через USB або Bluetooth та містить безліч датчиків, сенсорів та індикаторів.



Рисунок 1.10 – ADK 2012

Google пропонує два напрямки застосування ADK:

1) комерційний – аудіодок-станції, інтеграція в спортивні тренажери та ін.;

2) хобі – контролери роботизованої техніки.

ADK 2012 містить:

1. Arduino-сумісну плату на основі ARM 32-bit Cortex M3 мікропроцесора.

2. Два USB. Перший – для підключення до Android-пристрою, другий – для налагодження та програмування.

3. Датчики світла, кольору, наближення, температури, вологості, атмосферного тиску та акселерометр.

4. Слот для SD-карти.

5. Підтримка Bluetooth.

6. Шість семисегментних світлодіодних RGB-матриць, 12 «party mode» світлодіодів.

7. Ємнісний слайдер (гучність, яскравість і т. д.) та кнопки – по дві на кожную цифру і додатково ще вісім.

8. Підсилювач звуку та динамік.

9. NFC-мітка, доступна для запису.

Платформа дає можливість відтворювати аудіо з Android-пристрою по USB-з'єднанню. Вимоги: Android 4.1 (API Level 16 та вище).

Сам комплект ADK 2012 має вигляд закінченого пристрою у формі будильника з функцією аудіодока.

Вбудована підтримка Go. Починаючи з версії 2.4, для програмістів, які пишуть Android-програми мовою програмування Go, включена підтримка мови без будь-якого Java-коду, хоча і з обмеженим набором інтерфейсів Android.

Android APIMiner. Це платформа, яка є інструментом автоматичної генерації та вилучення документації Javadoc з реальних прикладних програм Android з відкритим вихідним кодом та з прикладами використання. Для поліпшення якості витягнутих прикладів APIMiner використовує внутрішньо-процесуальний статичний алгоритм витягування [19].

У табл. 1.3 подано опис основних компонентів Android APIMiner, а на рис. 1.11 відображено процес взаємодії між ними.

Таблиця 1.3 – Компоненти Android APIMiner

Компонент	Опис
Source Code Analyzer	Цей модуль аналізує вихідний код API та клієнтських систем для «витягування» структурних даних
Patterns Analyzer	Цей модуль витягує шаблони використання елементів API на основі їх використання
Examples Extractor	Цей модуль витягує приклади використання з вихідного коду клієнтських систем
Recommendation Engine	Цей модуль генерує рекомендації шаблонів та приклади використання на основі даних, витягнутих за допомогою попередніх модулів
JavaDoc Weaver	Цей модуль генерує документацію API, включаючи приклади використання

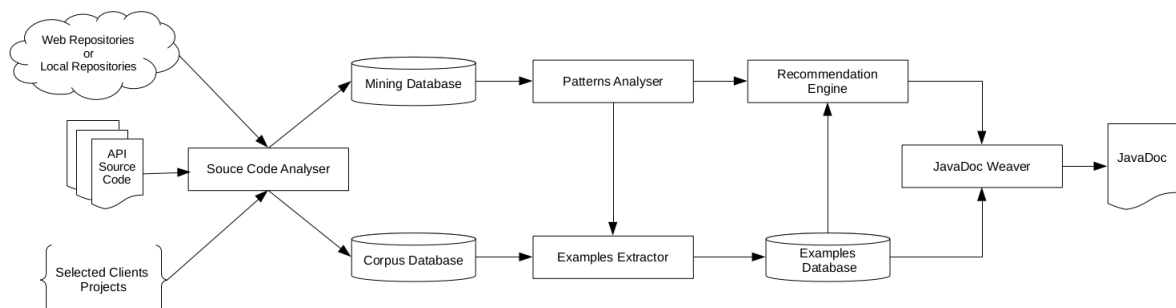


Рисунок 1.11 – Компоненти Android APIMiner

AndroWish. TCL (Tool Command Language) є дуже потужною, але легко досліджуваною динамічною мовою програмування, яка підходить для дуже широкого спектру застосування, в тому числі для мережових та настільних прикладних програм, мережового програмування, тестування та багато чого іншого. TCL має відкритий вихідний код і дійсно є крос-платформною мовою, яка легко розгортається та розширюється.

ТК-інструментарій для створення графічного інтерфейсу користувача піднімає розробку настільних прикладних програм на більш високий рівень, ніж звичайні засоби. ТК є стандартом інтерфейсу не тільки для TCL, але і для багатьох інших динамічних мов, та дозволяє створювати насичені програми, які працюють без змін під ОС Windows, Mac OS X, Linux.

AndroWish дозволяє запускати настільні TCL- і ТК-програми майже в незмінному вигляді на Android-платформі [20].

Деякі особливості AndroWish:

1. Рідний TCL/TK 8.6 порт для платформи Android (починаючи з версії 3.3.3 або вище) для процесорів ARM і x86.
2. Виконання існуючих TCL/TK-скриптів на Android-платформі без змін.
3. За основу взято ранній проект Тіма Бейкера SDLTk.
4. Емуляції X11 на основі AGG (Anti-Grain-Geometry) та SDL 3.0.
5. Забезпечує згладжування ліній, кіл, дуг.
6. Візуалізація шрифтів з використанням движка шрифтів Freetype.
7. Включає в себе віджет 3D-полотна, який використовує OpenGL для емуляції малювання OpenGL ES 2.2.
8. Включає в себе віджет `tkpath`, який розширює можливості канви SVG, такі, як рендеринг, альфа-канали, підтримка TrueType контурних шрифтів.
9. Деякі конкретні об'єкти Android доступні через SDL та викликаються командою `sdlTk`.
10. Використання AndroWish вимагає Android SDK та Android NDK.
11. Тестування та налагодження сценаріїв TCL на Android-пристроях може здійснюватися з системи розробки з використанням `tkconclient`. Файли можуть бути передані за допомогою підключення SSH/SFTP, як описано в `tkconclient`.
12. Експериментальна функція дозволяє будувати невеликі прикладні програми, які використовує встановлений AndroWish як основу.

App Inventor для Android. 12 липня 2010 року Google оголосила про доступність App Inventor для Android (рис. 1.12). App Inventor – це веб-орієнтоване візуальне середовище розробки для програмістів-початківців, основане на бібліотеці Open Blocks Java Массачусетського технологічного інституту (MIT).

Середовище забезпечує доступ до GPS, акселерометра, даних позиціонування пристрою, телефонних функцій, обміну текстовими повідомленнями, перетворення мови в текст, контактів, даних постійного зберігання та веб-служб.

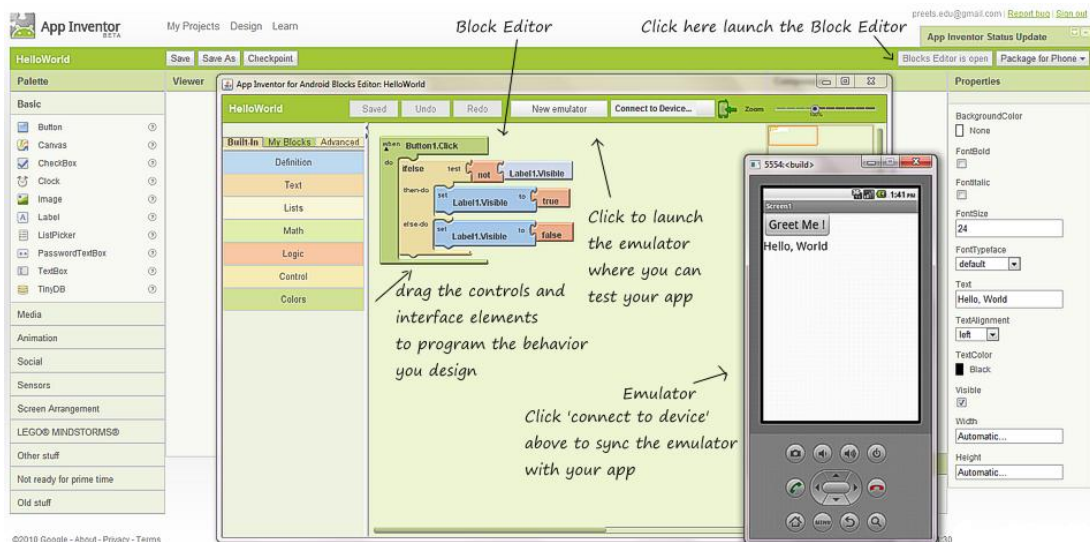


Рисунок 1.12 – App Inventor

Остання версія створена в результаті співробітництва Google та MIT, випущена у лютому 2012 року, в той час як перша версія, створена виключно MIT, була запущена у березні 2012 року та оновлена до App Inventor 2 у грудні 2013 року. З 2014 року App Inventor підтримується виключно MIT [21].

Basic4android – VisualBasic Native Android Language. Це простий і потужний інструмент розробки прикладних програм для пристроїв, що працюють під управлінням операційної системи Android. Мова Basic4Android (рис. 1.13) дуже схожа на популярну мову Visual Basic. При розробці прикладних програм використовується безліч різних додаткових бібліотек. Для виконання створених програм ніяких додаткових runtime-засобів не потрібно.

Corona SDK. Є комплект розробки програмного забезпечення, створеним Вальтером Лухом, засновником Corona Labs Inc. Він дозволяє програмістам створювати мобільні прикладні програми для iPhone, IPAD та Android-пристроїв (рис. 1.14).

Corona дозволяє розробникам створювати графічні прикладні програми, використовуючи інтегровану Lua-мову, яка нашаровується на C++/OpenGL. SDK поширюється на основі моделі продажу за передплатою, не вимагає яких-небудь відрахувань від продажу розроблених прикладних програм та не нав'язує ніяких вимог брендингу.

Середовище Delphi також може бути використане для створення прикладних програм для платформи Android із застосуванням мови Object Pascal.

Остання версія Delphi XE8 була розроблена Embarcadero Studio.

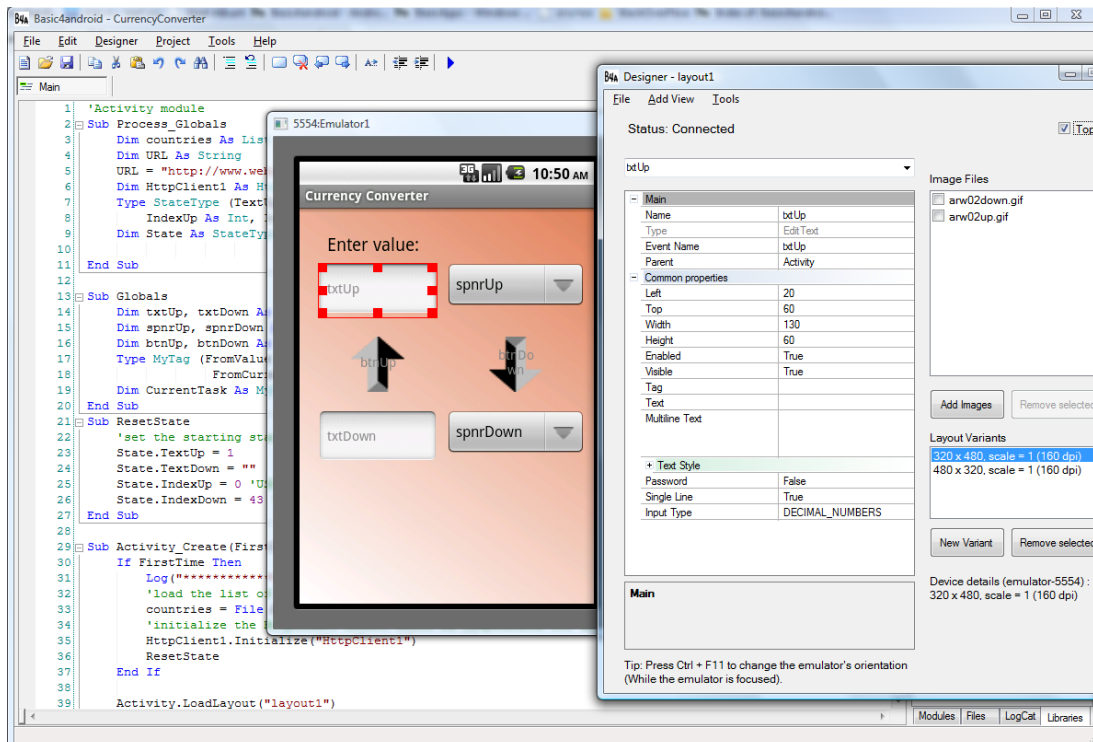


Рисунок 1.13 – Basic4android

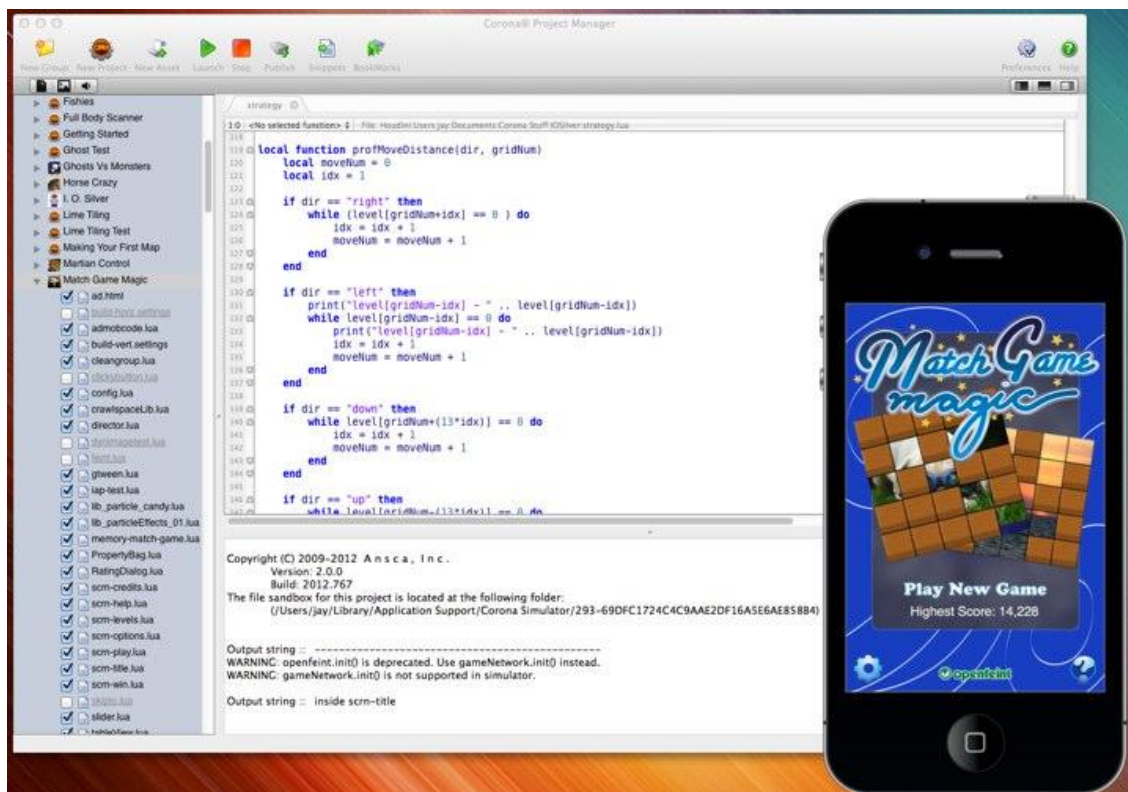


Рисунок 1.14 – Corona SDK

Embarcadero® RAD Studio XE8 (рис. 1.15) є закінченим рішенням для розробки програмного забезпечення для Windows, Mac, IOS, Android та IoT. RAD Studio дозволяє будувати готові рішення, які розробляються не тільки для клієнтських платформ, але також і для мобільних пристроїв, смарт-пристроїв, таких, як смарт-годинник та інші гаджети IoT [22].

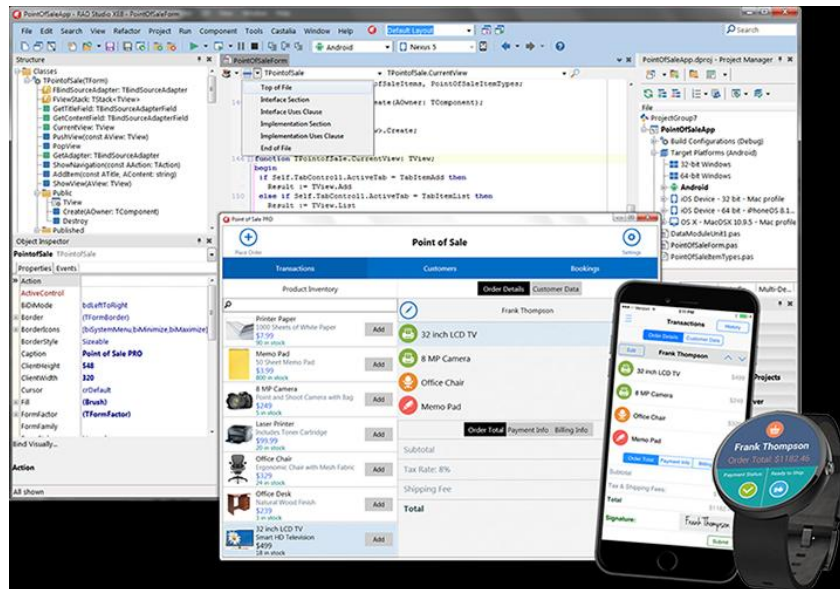


Рисунок 1.15 – RAD Studio XE8

HyperNext Android Creator (HAC) є системою розробки програмного забезпечення, орієнтованою на програмістів-початківців [23], яка допомагає їм створювати свої власні прикладні програми для Android, не знаючи Java та Android SDK (рис. 1.16). HAC основана на тому, що HyperCard обробляє програмне забезпечення у вигляді стопки карток, при цьому в будь-який момент часу видно тільки одну картку, що дуже добре підходить для прикладних програм мобільних телефонів, в яких відображається одноразово тільки одне вікно.

Kivy – це відкрита бібліотека мовою Python для розробки прикладного програмного забезпечення «мультитач» з природним інтерфейсом користувача (NUI) для широкого набору пристроїв (рис. 1.17). Kivy забезпечує можливість підтримки єдиної прикладної програми для численних операційних систем («кодуї один раз, запуская скрізь»). Kivy має виготовлений на замовлення інструмент розгортання мобільних прикладних програм під назвою *Buildozer*, доступний тільки для Linux. *Buildozer* в даний час існує в альфа-версії, але він набагато зручніший, ніж громіздкі старі методи розгортання Kivy. Прикладні програми, запрограмовані Kivy, можуть бути подані на будь-якій мобільній платформі Android-прикладних програм.

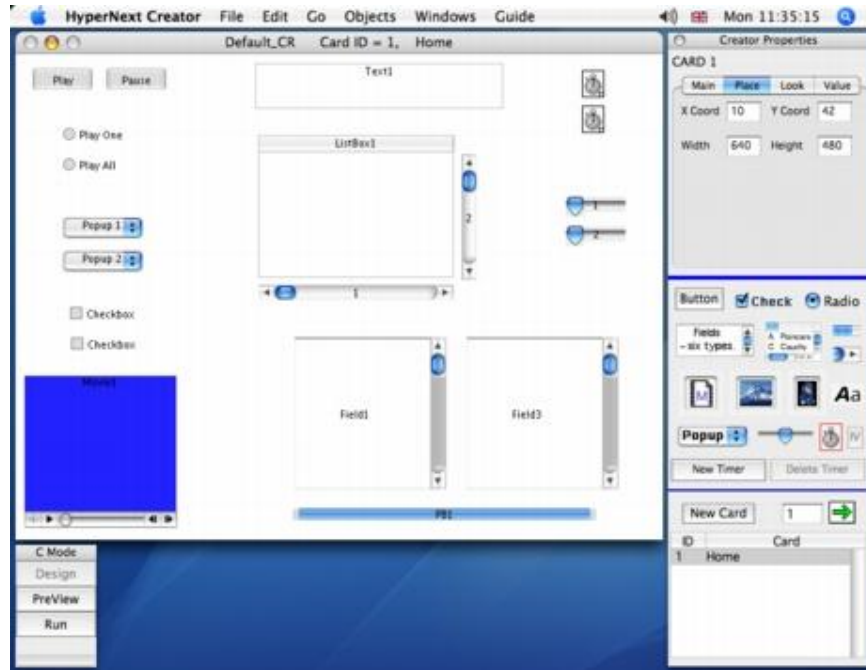


Рисунок 1.16 – HyperNext Android Creator

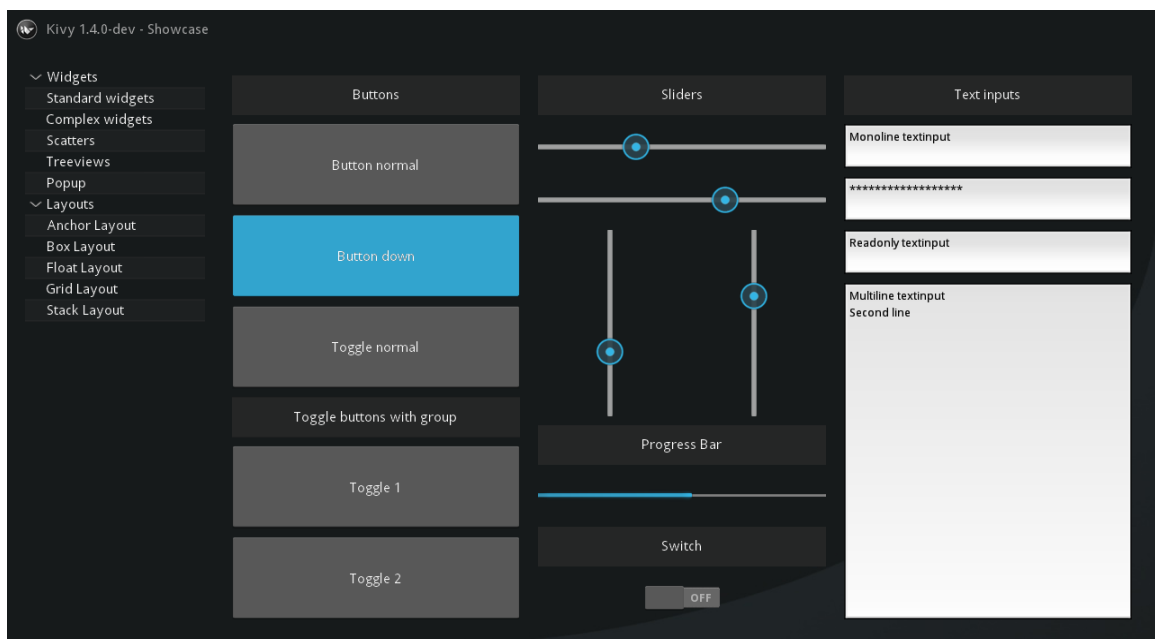


Рисунок 1.17 – Kivy 2.4 on Android

Lazarus можна використовувати для розробки прикладних Android-програм (рис. 1.18) на мові Pascal з компілятором Free Pascal, починаючи з версії 3.7.2.

Qt для Android, починаючи з версії Qt 5, створює прикладні програми для запуску на пристроях з Android v3.3.3 (рівень API 10) або пізнішої версії. Qt є основою для крос-платформних прикладних програм, які можуть запускатися

на цільових платформах, таких, як Android, Linux, IOS, Sailfish OS і Windows. Розробку Qt-прикладних програм виконують з використанням мови C++ та QML, вимагаючи при цьому встановлених Android NDK та SDK. Qt Creator є інтегрованим середовищем розробки і спільно з Qt Framework використовується для розробки мультиплатформних прикладних програм.

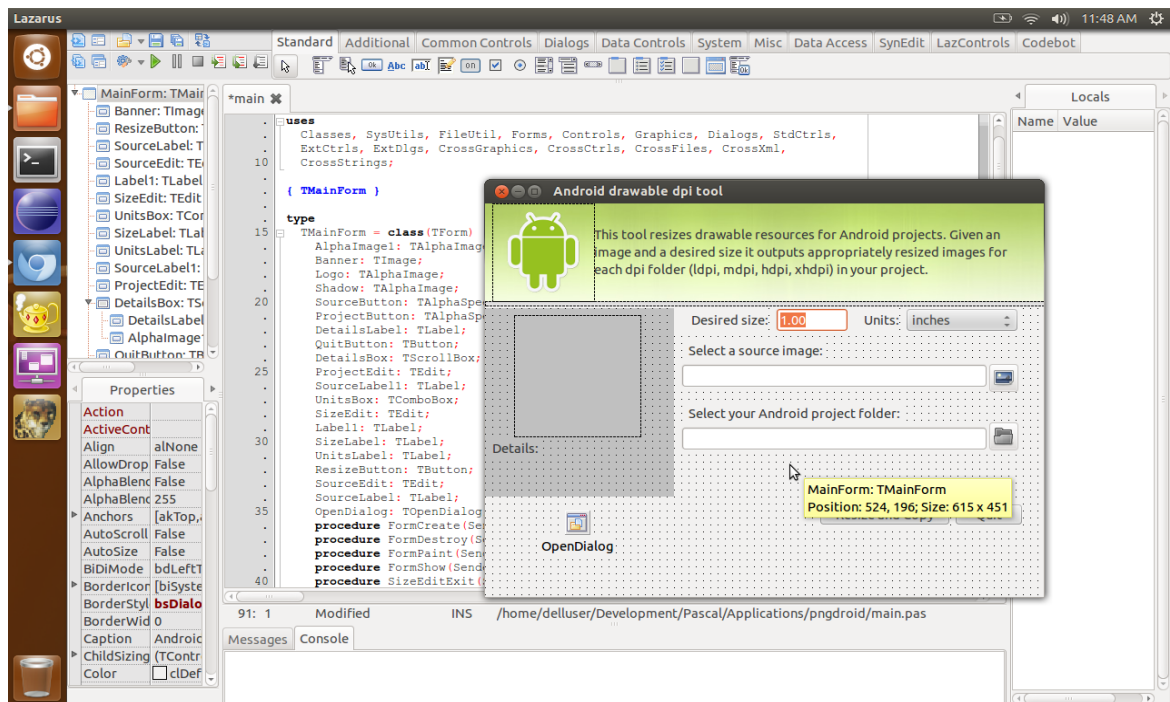


Рисунок 1.18 – Проект Lazarus

RFO BASIC – це діалект Dartmouth Basic, що є інтерпретатором з набором бібліотек для доступу до апаратного обладнання, датчиків, звуку, графіки, мультитачу, файлової системи, SQLite, мережі, HTML-інтерфейсу, шифрування, SMS, функцій телефону, електронної пошти, перетворення тексту в мову, розпізнавання голосу, GPS та інших функцій. Це програмне забезпечення з відкритим вихідним кодом може компілювати автономні APK-файли. RFO Basic активно розвивається з березня 2015 року.

RubyMotion є набором інструментів для створення мобільних прикладних програм на мові Ruby. Підтримка Android з'явилася у версії RubyMotion 3.0. Прикладні програми Android, створені з використанням RubyMotion, можна назвати в цілому набором Java API від Ruby, при цьому можливе використання сторонніх бібліотек Java.

Saphir є відгалуженням з відкритим вихідним кодом від проекту Rebol3 (R3). Вся функціональність R3, в тому числі GUI, графіка, доступ до мережі, доступ до файлів, парсинг та інші особливості портується на основні пор-

тативні ОС Android, Windows, Mac, Linux без будь-яких змін у вихідному коді. Saphir дозволяє використовувати шаблони діалектних моделей (DSL) для побудови графічних інтерфейсів користувача та виконання спільних обчислювальних операцій. Невеликий розмір компілятора (0,5–1,5 мегабайт) доповнюється простим утилітарним дизайном Saphir.

Бібліотека SDL пропонує, крім можливості розробки з використанням Java, можливість розробки з використанням C з наступним простим перенесенням існуючих SDL та власних прикладних програм C. Застосування Java-ін'єкцій та прокладок JNI дозволяє використовувати рідну бібліотеку SDL при портуванні на пристрої Android, наприклад, як у відео грі Jagged Alliance 3.

Метою The Simple project є забезпечення розробника простою в розумінні та використанні мовою для розробки прикладних програм для платформи Android. [24] The Simple project є основним діалектом для розробки прикладних програм Android. Він націлений на професійних і непрофесійних програмістів та дозволяє програмістам швидко створювати додатки для Android.

Подібно до Microsoft Visual Basic 6, The Simple project визначає форми (які містять компоненти) та код (який містить логіку програми). Взаємодія між компонентами та програмною логікою відбувається через події, викликані компонентами. Логіка програми складається з обробників подій, які містять код реагування на події.

The Simple project не надто активний, [25] останнє оновлення вихідний код зазнавав у серпні 2009 року.

Visual Studio 2015 (рис. 1.19) підтримує розробку крос-платформних прикладних програм, дозволяючи розробникам C++ створювати проекти з шаблонів для Android-прикладних програм або динамічні високопродуктивні колективні бібліотеки для включення їх в інші рішення. Функціонал середовища включає в себе інтелектуальний підказувач IntelliSense, точки зупину, розгортання пристроїв та емуляції. [26]

WinDev Mobile – власна IDE, яка створена PC SOFT (рис. 1.20), що використовується для створення графічного інтерфейсу користувача (GUI) прикладних програм для смартфонів та планшетів (включаючи Android-пристрої). Вона використовує мову програмування WLanguage та доступна англійською, французькою та китайською мовами.

Розробники на C# можуть використовувати *Xamarin* для створення прикладних програм для платформ IOS, Android, Windows. Станом на лютий 2014 року Xamarin використовують понад 505 тисяч розробників в більш ніж 120 країнах по всьому світу.

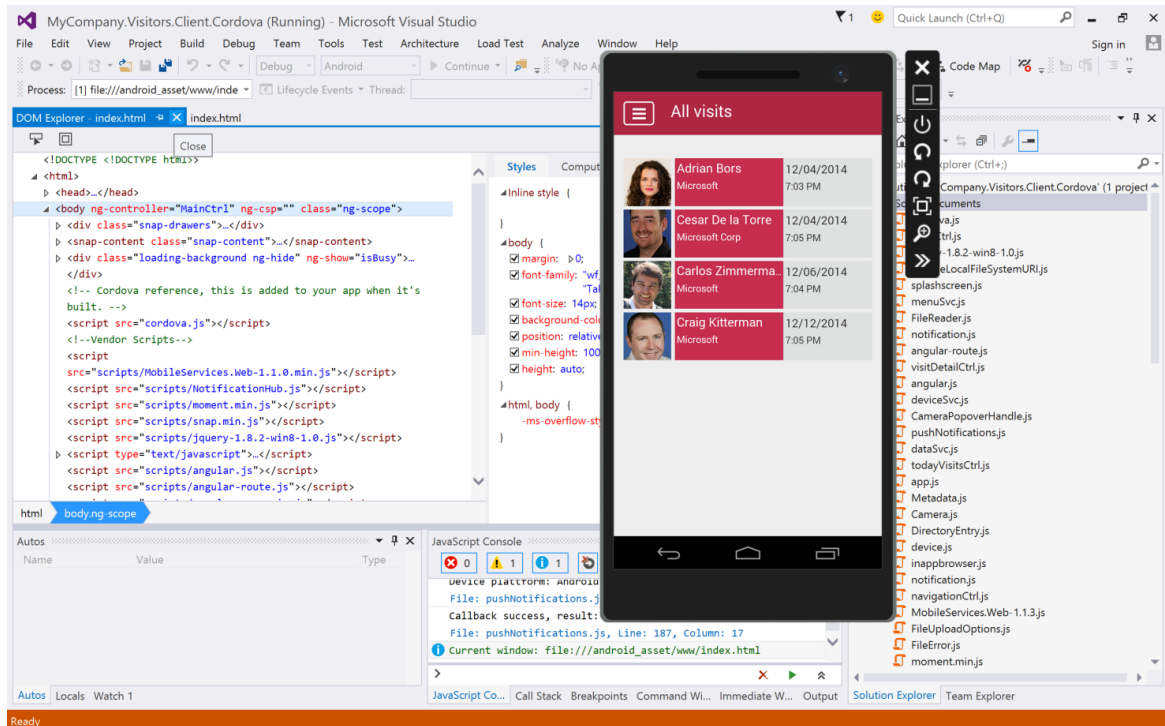


Рисунок 1.19 – Visual Studio 2015

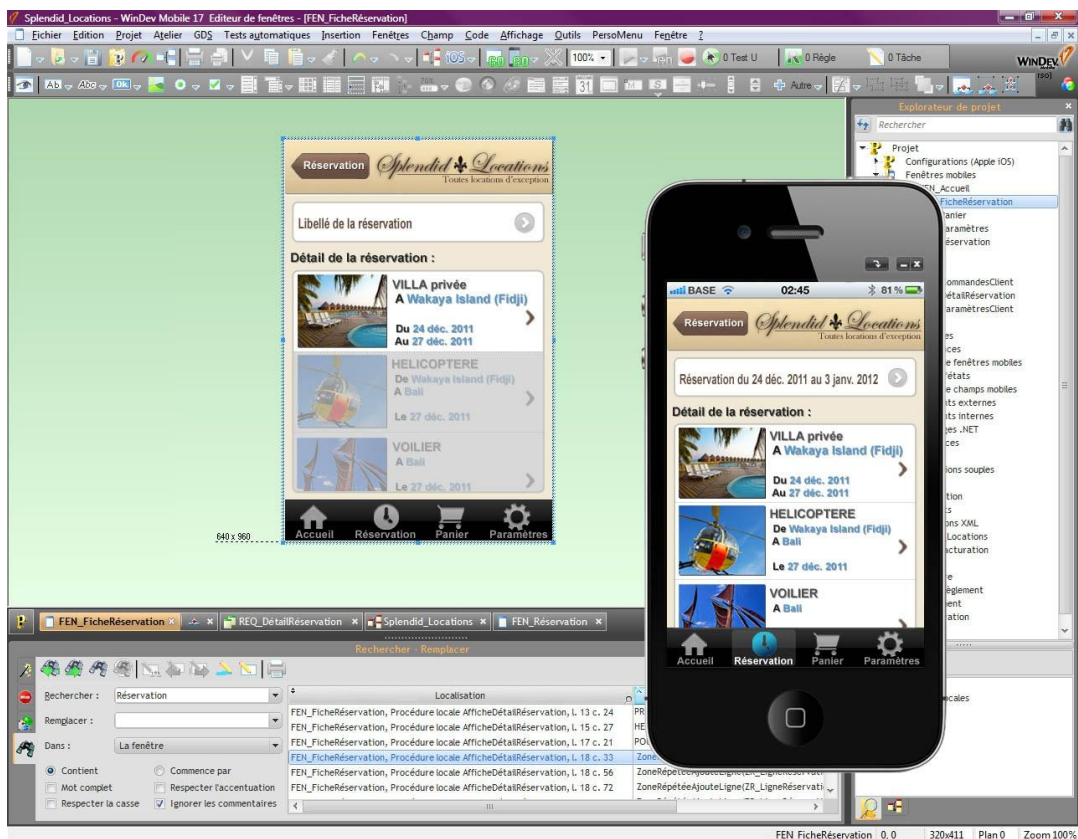


Рисунок 1.20 – WinDev Mobile

X11 Basic є діалектом мови програмування Basic з графічними можливостями, який об'єднує такі функції, як оболонки сценаріїв, програмування CGI та повна графічна візуалізація. Синтаксис в основному схожий на старий GFA Basic, який використовувався на комп'ютерах Atari ST.

1.8. Огляд інтегрованих середовищ розробки (IDE)

Інтегроване середовище розробки (IDE) є інструментом, який дозволяє розробнику прикладних програм виконати повний цикл розробки програмного забезпечення. Цей цикл включає проектування, кодування, компіляцію, тестування, налагодження та упаковку програмного забезпечення прикладної програми. Для створення Android-додатків Google в даний час підтримує дві IDE:

– Android Developer Tools (ADT) <http://developer.android.com/sdk/index.html>;

– Android Studio <http://developer.android.com/sdk/installing/studio.html>

Обидва середовища розробки потребують використання мови Java.

Середовища розробки від Google та мова програмування Java не є єдиними варіантами для розробки прикладних програм Android App. Деякі розробники не знають мови програмування Java або вважають за краще програмувати з використанням мови C/C++. Деякі розробники хотіли б використовувати єдиний базовий код для підтримки інших платформ: Apple (iOS), Windows, Blackberry і Web (HTML5). Така розробка відома як крос-платформна розробка. Існує багато альтернатив інструментам Google. Огляд альтернативних IDE наведено в табл. 1.4. У таких системах код може бути написаний різними мовами, такими, як BASIC, HTML5 або Lua. Багато з альтернативних IDE можуть використовуватися вільно, деякі мають відкритий вихідний код, деякі є обмеженими версіями. Деякі вимагають Android Software Development Kit (SDK), який поставляється разом з інструментами Google. Можна встановити кілька середовищ розробки на одному комп'ютері, щоб протестувати їх можливості.

Таблиця 1.4 – Огляд альтернативних середовищ розробки

Найменування	Мова програмування	Крос-платформність	URL
AIDE (Android IDE)	HTML5/C/C++	Так	http://www.android-ide.com/
Application Craft	HTML5	Так	http://www.applicationcraft.com/
Basic4Android	BASIC	Ні	http://www.basic4ppc.com/
Cordova	HTML5	Так	https://cordova.apache.org/
Corona	Lua	Так	http://coronalabs.com/

Продовження табл. 1.4

Найменування	Мова програмування	Крос-платформність	URL
Intel XDK	HTML5	Так	https://software.intel.com/en-us/html5/tools
IntelliJIDEA	Java	Ні	https://www.jetbrains.com/idea/features/android.html
Kivy	Python	Так	http://kivy.org/#home
MIT App Inventor	Blocks	Так	http://appinventor.mit.edu/explore/
Monkey X	BASIC	Так	http://www.monkeycoder.co.nz/
MonoGame	C#	Так	http://www.monogame.net/
MoSync	HTML5/C/C++	Так	http://www.mosync.com/
NS BASIC	BASIC	Так	https://www.nsbasic.com/
PhoneGap	HTML5	Так	http://phonegap.com/
RAD Studio XE	Object Pascal, C++	Так	http://www.embarcadero.com/
RFO Basic	BASIC	Ні	http://laughton.com/basic/
RhoMobile Suite	Ruby	Так	http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite
Telerik	HTML5	Так	http://www.telerik.com/platform#overview
Titanium	JavaScript	Так	http://www.appcelerator.com/titanium/titanium-sdk/
Xamarin	C#	Так	http://xamarin.com/

Запитання для самоконтролю

1. Назвіть рівні програмного забезпечення Android.
2. Назвіть засоби керування ресурсами мобільних пристроїв ОС Android.
3. Дайте характеристику віртуальній машині Dalvik.
4. Дайте характеристику віртуальній машині ART.
5. Назвіть обмеження на використання Java API у мобільному пристрої.
6. Наведіть основні засоби розробки мобільних додатків.
7. Наведіть основні середовища розробки IDE.

2. АРХІТЕКТУРА МОБІЛЬНИХ ДОДАТКІВ

2.1. Компоненти інтерфейсу

Макет визначає візуальну структуру користувальницького інтерфейсу, наприклад, призначеного для користувача інтерфейсу операції або віджета додатка. Існує два способи оголосити макет:

1. *Оголошення елементів призначеного для користувача інтерфейсу в XML.* В Android є зручний довідник XML-елементів для класів View і їх підкласів, наприклад таких, які використовуються для віджетів і макетів.

2. *Створення екземплярів елементів під час виконання.* Ваша програма може програмним чином створювати об'єкти View і ViewGroup (а також керувати їх властивостями).

Платформа Android забезпечує гнучкість при використанні будь-якого з цих способів для оголошення, призначеного для користувача інтерфейсу додатка і його управління. Наприклад, можна оголосити в XML макети за замовчуванням, включаючи елементи екрана, які будуть відображатися в макетах, і їх властивості. Потім можна додати в додаток код, який дозволяє змінювати стан об'єктів на екрані (включаючи оголошені в XML) під час виконання.

У модулі ADT для Eclipse передбачена функція попереднього перегляду, створеного файлу XML, для цього досить відкрити файл XML і вибрати вкладку Layout (Макет).

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

За допомогою інструменту `layoutopt` можна швидко проаналізувати макети і їх ієрархії на предмет низької ефективності чи інших проблем.

Перевага оголошення призначеного для користувача інтерфейсу у файлі XML полягає в тому, що таким чином можна більш ефективно відокремити подання свого додатка від коду, який управляє його поведінкою. Описи призначеного для користувача інтерфейсу знаходяться за межами коду вашої програми. Це означає, що можна змінювати або адаптувати інтерфейс без необхідності вносити правки у вихідний код і повторно компілювати його. Наприклад, можна створити різні файли XML-макета для екранів різних розмірів і різних орієнтацій екрана, а також для різних мов. Крім того, оголошення макета в XML спрощує візуалізацію структури призначеного для користувача інтерфейсу, завдяки чому налагодження проблем також стає простішим. У даному підрозділі ми навчимо вас оголошувати макет в XML. Якщо ви волієте за краще