КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

РОЗРОБКА ДОДАТКУ «СПИСОК СПРАВ» З ВИКОРИСТАННЯМ АРХІТЕКТУРНИХ КОМПОНЕНТІВ ВІД GOOGLE НА ANDROID

Виконав студент 4-го курсу
Станіслав КАЦУБО _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Оксана ШКІЛЬНЯК _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«____» _____ 2021 р.,
протокол No ___
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 59 сторінок, 15 ілюстрацій, 2 додатка, 25 джерел посилань.

АРХІТЕКТУРНИ КОМПОНЕНТИ, ГНУЧКЕ УПРАВЛІННЯ, ІНТЕРФЕЙС, МОБІЛЬНИЙ ДОДАТОК, КОРИСТУВАЧ, МОВА JAVA, ПЛАТФОРМА ANDROID, СПИСОК СПРАВ.

Об’єктом роботи є процес вибору засобів і технологій реалізації та розробка мобільного додатку.

Метою роботи є розробка додатку з використанням архітектурних компонентів від Google на Android з функціями розпорядку дня, заміток, нагадувань.

Інструменти розроблення: середовищем для програмування було обрано Android Studio. Для тестування мобільного додатку було обрано мінімальну версію SDK: API 21: Android 5.0. Додаток було розроблено на платформі Firebase, мова програмування Java.

Розроблено програмний продукт «СПИСОК СПРАВ», який дозволяє створювати, редагувати, видаляти завдання, налаштувати розклад виконання запланованих завдань; в ньому є функція створення і видалення папок, відображення списку активних процесів, відображення справ у вигляді відсортованого списку; є також можливість вибирати дату планування та планувати на тиждень та відзначити справу, як виконану.

Як результат вдалося створити працюючий мобільний додаток який має великий потенціал на розширення функціоналу, та може зайняти свою нішу серед схожих додатків.

ЗМІСТ

ВСТУП

У сучасному світі одне з головних багатств людини це час, а його планування це дуже важлива частина життя. Один з основних принципів - не зберігати справи в голові. Спроби хапатися за все одночасно тільки позбавляють нас контролю над ситуацією. Треба вчитися правильно розставляти пріоритети.

Додаток Effective Time має допомогти контролювати розклад всього дня. Додаток дозволяє впоратися зі списками справ, залишити час і сили для довгострокових цілей і знизити стрес.

Актуальність програми вельми висока, тому що питання контролю часу актуально сьогодні як ніколи. Протягом дня, кожна людина виконує якусь кількість завдань. Згодом, з’являються нові завдання, а до деяких старих доводиться повертатися. Разом з кількістю завдань зростає і обсяг інформації, яка стає складною для запам’ятовування: мимоволі ми починаємо випускати з виду деякі важливі завдання, можливо, менш важливі, ніж інші, або починаємо робити їх несвоєчасно. Для вирішення перерахованих проблем можна дотримуватися кількох правил:

- щовечора планувати час відповідно до завдань, які необхідно виконати завтра;

- сортувати завдання на категорії;

- дотримуватися принципу «70/30», тобто виділяти 70% вільного часу на заплановані справи;

- виконувати велику частини справ до обіду;

- розбивати комплексні і складні завдання на більш дрібні;

- знаходить час для відпочинку.

Можливо, комусь достатньо дотримуватися даних рекомендацій, але досить ефективним буде використання спеціалізованого програмного забезпечення.

Мета роботи розробити додаток з використанням архітектурних компонентів від Google на Android з функціями розпорядку дня, заміток і нагадувань.

Для досягнення мети нам потрібно вирішити такі завдання:

- обґрунтувати доцільність розробки мобільного додатку щодо автоматизованої системи списку справ з використанням архітектурних компонентів від Google.
- розробити модель і структуру мобільного додатку;
- визначити засоби реалізації системи гнучкого управління списком справ у мобільному додатку;
- розробити програмне забезпечення мобільного додатку на Android;
- провести тестування програмного забезпечення.

Середовищем для програмування було обрано Android Studio. Для тестування мобільного додатку було обрано версію SDK: API 21: Android 5.0. Додаток було розроблено на платформі Firebase. Основним критерієм при виборі платформи був орієнтир на використання додатку на мобільних пристроях. Мова програмування Java. Ця мова офіційною вважається мовою Android та широко застосовується для веб-сайтів та мобільних додатків.

Розроблений програмний продукт «СПИСОК СПРАВ» дозволяє створювати, редагувати, видаляти завдання, налаштувати розклад виконання запланованих завдань; в ньому є функція створення і видалення папок, відображення списку активних процесів, відображення справ у вигляді відсортованого списку; є також можливість вибирати дату планування та планувати на тиждень та відзначити справу, як виконану.

Як результат вдалося створити працюючий мобільний додаток який має великий потенціал на розширення функціоналу, та може зайняти свою нішу серед схожих додатків.

РОЗДІЛ 1. ТЕХНІЧНЕ ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ ЩОДО АВТОМАТИЗОВАНОЇ СИСТЕМИ СПИСКУ СПРАВ З ВИКОРИСТАННЯМ АРХІТЕКТУРНИХ КОМПОНЕНТІВ ВІД GOOGLE

1.1. Суть технічної проблеми яка виникла на сучасному розвитку науки, техніки та технологій

Експерти з продуктивності стверджують, що для досягнення успіху потрібно навчитися керувати своїм часом. Якщо маєш багато планів на 2021 рік, то правильний додаток для ефективного тайм-менеджменту зможе допомогти тобі перейти від виживання до процвітання.

Планування та облік свого часу полягає в контролі щоденних справ з метою досягнення довгострокових цілей.

Мобільний додаток призначений для ведення особистих справ і вирішує проблеми користувачів таким чином:

- додаток має хмарне зберігання для даних особистих справ; усі користувачі мають обліковий запис для доступу до своїх даних;
- користувач має доступ до хмари на мобільному додатку.

З допомогою мобільного додатку користувач впорається з ефективним обліком своїх справ протягом дня.

1.2. Розробка мобільного додатку на базі платформи Android

Проаналізуємо особливості створення мобільних додатків. Мобільний додаток – це програма, розроблена спеціально для мобільних пристроїв на певній платформі (Android, WindowsPhone, iOS) з метою оптимізувати вирішення якоїсь важливої задачі в житті користувача.

Мобільний додаток є можливість створити за допомогою наступних мов програмування: C++, C#, Java, Python, Kotlin.

Java – це об’єктно-орієнтована мова програмування, яка була розроблена компанією «Sun Microsystems» і запущена в 1995 році як основний компонент платформи Java.

Середовищем для програмування вирішено було обрати Android Studio, як зручне для написання коду, та для створення дизайну мобільного додатку.

Для початку потрібно було вибрати мінімальну версію SDK - набір засобів розробки, за допомогою яких можна створити програми, програмне забезпечення базису розробки комп’ютерної системи, ігрової консолі та ін. Для тестування було обрано версію SDK: API 21: Android 5.0.

Наступний крок - це створення необхідних файлів згідно архітектури MVC. Усі об’єкти в програмі діляться на 3 категорії:

- Model (містять дані та «бізнес-логіку», їх використовують для моделювання змісту);
- View (відображаються на екрані і реагують на дії користувача, їх заповнюють на основі розмітки XML в файлі макету);
- Controller (які пов’язують об’єкти view і model - містять «логіку додатку»).

Перевагами такої архітектури є те, що програмний код розділяється на класи – таким чином можна підвищити його функціональність і спростити розуміння коду взагалі.

В Java файлах описується логіка програми - то, що ви хочете, щоб ваше додаток виконувало.

У XML файлах розробляються макети - зовнішній вигляд.

Як тільки мобільний додаток буде написано, потрібно використовувати технічний інструмент збірки для того, щоб скомпілювати всі файли і упакувати їх разом в .apk файл, який можна запускати на пристроях Android і / або опублікувати в Google Play.

Всі утиліти і файли, які дозволяють створювати додаток під Android, об’єднані в інтегроване середовище розробки (IDE). IDE - це програма, яка

використовується для редагування ваших файлів коду, а також компіляції і запуску.

Примітно: Раніше стандартної IDE для розробки була Eclipse. Зараз затребуваною IDE є більш функціональна Android Studio. Це середовище для розробки додатків від Google, основою для якої послужив продукт компанії JetBrains - IntelliJ IDEA (альтернатива Eclipse).

Розробка для ОС Android має кілька особливостей. Операційне середовище Android за весь час пережила ряд випусків. Багато навіть зараз все ще використовують пристрої з версією 5.0, яка була випущена у відносно далекому 2014 році, хоча зараз вже ведеться розробка для випуску 10 Q.

Гаджети Android характеризуються широким вибором дозволів дисплеїв і фізичних розмірів. Всі виробники намагаються здивувати своїх користувачів новими функціями, якістю камери, потужністю батареї. Тому в продажу є пристрої з розміром дисплея від 4 до 10 дюймів, серед смартфонів можна знайти 6-дюймові пристрої. Це ще одна особливість гаджетів Android.

База користувачів Android дуже велика, і, як результат, продукт, представлений в додатку, буде мати широку аудиторію.

1.3. Варіантний аналіз та обґрунтування вибору засобів реалізації автоматизованої системи списку справ з використанням архітектурних компонентів від Google

1.3.1. Аналіз технологій розробки мобільних додатків

На сьогоднішній день смартфони стали незамінними гаджетами для кожної людини. Зараз набагато частіше зустрічаються люди без персонального комп’ютера, але з декількома мобільними пристроями.

Використання додатків істотно позбавляє від довгих очікувань завантаження графіки, зображення, звуку та інших компонентів, а також економить час, оскільки програма вже встановлена на телефон.

Розробники додатків можуть створювати мобільні додатки, які доступні для кожної операційної системи телефону або планшета в окремо. Сьогодні мобільний додаток має особливе значення для компаній, які в своїй діяльності приділяє велику увагу використання Інтернету. Поліпшення програми та зміцнення авторитету і репутації компанії взаємопов'язані.

PhoneGap – це безкоштовна програма з відкритим вихідним кодом, яка може працювати з апаратними функціями пристрою, включаючи акселерометр, GPS місце розташування, камеру, звук і багато інших подібних елементів. Середовище цього інструменту інтегроване з великою кількістю бібліотек, які допомагають поліпшити функціональність програми і тим самим допомагають заощадити багато часу [7, с. 28].

Цей універсальний засіб є унікальним завдяки властивої йому гнучкості, тому він допомагає розробникам створювати різні типи мобільних додатків при значно менших витрачених на них зусиль. Крім того, розробник, який працює з цим інструментом, може розширювати функціональність і зручність використання мобільного додатку за допомогою різних архітектур, що підключаються в міру необхідності.

Інструмент посилений потужною «бекенд-системою», а це означає, що мобільні додатки, розроблені за допомогою PhoneGap, мають високу швидкість виконання. У даного інструменту є певні переваги, які роблять його привабливим для розробників, але найбільш корисна особливість полягає в тому, що при розробці ви відразу отримаєте додаток, який працює на всіх мобільних пристроях. За допомогою PhoneGap Build можна робити збірки для iOS, Android і Windows Phone одночасно, уникаючи необхідності встановлювати будь-які інструменти. Але головне те, що за допомогою цього сервісу можна робити збірки для iOS в хмарі без наявності Mac. Також, розробники можуть створювати додатки для пристроїв Symbian, Palm, BlackBerry, iTouch і iPad.

Фреймворк PhoneGap став популярним завдяки тому, що дозволяє створювати мобільні додатки, використовуючи JavaScript, HTML і CSS3.

Rhodes – фреймворк, заснований на мові програмування Ruby, формує основу для RhoMobile. Rhodes дозволяє розробнику створювати Крос платформні нативні додатки, які будуть сумісні з величезним діапазоном операційних систем і смартфонів.

За своєю анатомією або архітектурою RhoMobile дуже схожий на PhoneGap. Головною його особливістю є використання Ruby і MVC архітектури додатків. Створений вами код на мові Ruby інтерпретується в нативний код пристрою [14, с. 53].

У даного фрейм ворка є готовий платний «бекенд». Це нереляційна база даних «ключ-значення», для якої є впроваджені в фреймворк засоби взаємозв'язку. Є платний сервер, що надає легку інтеграцію вашого додатку з існуючими. Є безкоштовний засіб для створення додатків в хмарі, для вирішування невеликих завдань. Тому немає необхідності встановлювати і налаштовувати на комп'ютері – все зберігається і компілюється в хмарі.

Appcelerator є відкритим ресурсом фреймворків мобільних додатків. Дана платформа значно прискорює процес розробки, оскільки дозволяє розробникам створювати додатки з найменшою кількістю рядків коду. Appcelerator є платформою розробки Titanium, яка допомагає створювати власні додатки для мобільних телефонів, планшетів і навіть персональних комп'ютерів. Інструмент сумісний з такими мовами, як JavaScript, HTML, PHP, Ruby і Python.

Додатки, створені за допомогою даного інструменту, можуть бути повністю апаратними, пропонуючи можливість зберігати всі дані на пристрої або в Cloud ресурсі. Оскільки для власника додатка процес розробки не закінчується одним лише написанням коду, творці Appcelerator передбачили можливість отримання доступу до деяких відомостей про використання самого додатку. Ця інформація може бути корисною для відстеження найважливіших характеристик програми та її подальшого удосконалення. Аналітична платформа дозволяє здійснювати моніторинг даних про програму та її продуктивність в режимі реального часу.

Xamarin – ще один популярний інструмент серед розробників додатків на мові C #. Платформа для цього інструменту унікальна тим, що дозволяє розробникам працювати з власними IDE (інтегрованими середовищами розробки), API (інтерфейсами прикладного програмування) і мовами. Саме ці характеристики роблять Xamarin кращим вибором, коли справа доходить до власних додатків.

Xamarin повторно використовує рівні бізнес-логіки і доступ до даних на різних платформах, що особливо корисно для розробників, коли необхідно реалізувати певні функції, такі як великий обсяг локальних даних, автономний режим і розпізнавання зображень. Як ми вже згадували раніше, Xamarin побудован на мові програмування C #, що означає, що він працює на загальній мовній інфраструктурі .NET.

Xamarin як інструмент досить популярний для створення додатків на всіх трьох великих платформах – Android, Windows і iOS. Він підтримує моніторинг якості і функціональності тестування в широкому діапазоні пристроїв, який допомагає розробникам придумувати масштабовані і надійні мобільні додатки. Це моно-фреймворк, який дозволяє встановлювати і підтримувати зв’язок з API на мобільних пристроях.

Особливість, яка робить Xamarin хорошим інструментом, в тому, що додатки, створені з його допомогою, як правило, мають меншу кількість помилок в порівнянні з іншими і тим самим будуть забезпечувати швидший вихід на ринок. Інструмент має менший за обсягом код, що означає меншу кількість помилок. Таким чином, за допомогою всього одного тесту код обох платформ може бути перевірений з великою ефективністю.

Фреймворк Ionic заснований на мові SASS CSS і є крос платформним. Він має можливість працювати з багатьма операційними системами. Це один з найбільш простих у використанні інструментів, і навіть може бути інтегрований з AngularJS, якщо ви плануєте розробляти трохи складніші програми.

Ionic переважно використовується для створення гібридних мобільних додатків. Оскільки це повний пакет SDK з відкритим вихідним кодом (Software Development Kit), Ionic ідеально підходить для створення гібридних мобільних додатків з використанням веб-технологій, включаючи (але не обмежуючись) CSS, HTML5 і SASS.

Це відмінний інструмент з унікальними функціями і сервісами для створення додатків. У Ionic є ціла бібліотека оптимізованих для мобільних пристроїв компонентів, інструментів і жестів HTML, CSS і JS CSS.

Appy Pie – являє собою платформу для самостійного створення мобільних додатків для тих, хто не знає програмування або не хоче в нього заглиблюватися. Такі програми можна публікувати в Google Play, Apple App Store, Windows App Store або в будь-якому іншому магазині, в якому побажаєте.

Одним з головних переваг даної платформи є те, що вам не доведеться писати жодного рядка коду, а просто клацнути мишею і додати необхідні функції. На виході ви зможете отримати унікальний додаток, на створення якого було потрібно мінімум часу і ресурсів. Відмінною особливістю платформи є те, що вона надає можливість використання сотні різних функцій, які ви можете додати в свій мобільний додаток, що створюється самостійно без зайвих витрат і допомоги агентств по розробці.

Крім того, у даній платформи є велика бібліотека в розділі поширених запитань, а також досить насичений канал на YouTube, який буде тримати вас в курсі останніх подій і новин в світі тенденцій, давати корисні поради про те, що може зробити ваш додаток успішним.

Appy Pie володіє досить всеосяжною інформаційної панеллю, яка дозволить вам отримувати важливі дані про ваш додаток. Вона містить інформацію про те, наскільки це популярно для цієї програми, а також чому ваша аудиторія віддає перевагу саме вам. Вам нададуть дані про те, які функції, кнопки або елементи потрібно налаштувати, якщо ви хочете залучити й утримати своїх користувачів.

NativeScript також є крос платформним інструментом розробки з відкритим вихідним кодом, але, на відміну від багатьох інших конкурентів, його можна використовувати безкоштовно.

Додатки NativeScript розробляються на таких мовах, як JavaScript або TypeScript. У NativeScript реалізована повна підтримка фреймворку AngularJS. Мобільні додатки, побудовані з NativeScript, мають повний доступ до API платформ.

За допомогою NativeScript розробники можуть використовувати Angular, JavaScript або TypeScript і створювати мобільні додатки для Android і iOS. NativeScript також має інтеграцію з Vue.JS, на додаток до його здатності підтримувати сотні плагінів для розширеної функціональності. Це, однак, вимагає хорошого знання командного рядка, а це значить, що розробник також повинен буде надати свій власний текстовий редактор.

Firebase - це платформа розробки мобільних додатків з величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих крос платформних додатків. Головне достоїнство платформи в тому, що вона дозволяє розробнику не відволікатися на створення бекенд, тобто прихованої від користувача програмної частини проекту, наприклад, серверного коду. І це спрощує і прискорює створення мобільних додатків, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді. Саме зв’язка Firebase з фрейм ворком Flutter дозволяє програмістам компанії AVADA MEDIA створювати швидкі програми для Android і iOS, що дозволяють вирішувати найрізноманітніші завдання.

Firebase - це одне з BaaS-рішень (Backend as a Service), яке дає розробнику масу можливостей.

Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі.

Додаток підключається до бази даних через WebSocket, який відповідає за синхронізацію даних протягом усього сеансу (рисунок 1.1).

Рисунок 1.1 – Синхронізація даних на мобільних платформах

Також Firebase виступає в якості сховища файлів. Firebase Storage забезпечує надійне завантаження і вивантаження файлів для додатка. Хмарне зберігання файлів відео, аудіо або будь-якого іншого типу підтримується Google Cloud Storage. Вміст хмарного сховища надійно захищене власною системою безпеки.

При створенні нового мобільного додатку в AVADA MEDIA взагалі багато уваги приділяється питанням безпеки. Створювати систему автентифікації кожен раз з нуля досить затратно, причому витрати ці найчастіше не виправдані. Впоратися з більшістю викликів дозволяє система автентифікації Firebase Auth, в якій можлива автентифікація користувача додатку по паролю і електронній пошті. Firebase Auth також підтримує відкритий протокол авторизації OAuth 2.0, який використовується Google, Twitter, Facebook. Система аутентифікації Firebase інтегрується безпосередньо в базу даних.

Статичні файли програми розміщуються на хостингу Firebase. Підтримується хостинг файлів JavaScript, HTML, CSS та інших. Через Cloud Functions реалізована динамічна підтримка Node.js. Передача файлів здійснюється через мережу доставки контенту з використанням захищених протоколів SSL і HTTPS.

Розробники AVADA MEDIA створюють за допомогою Firebase легкі і надійні додатки для iOS і Android. Використання платформи дозволяє скоротити час, необхідний для розробки - а також скоротити час завантаження програми і підвищити охоплення. Переваги платформи Firebase в вашому додатку:

- висока швидкість роботи програми;
- надійна інфраструктура - відсутність збоїв в роботі;
- зручна статистика, що дозволяє отримувати дані про дії користувачів і підтримувати зворотний зв’язок;
- крос платформність - програма створена один раз і налаштована для роботи з усіма операційними системами;
- проста масштабованість - зростання кількості користувачів не потребують змін у серверному коді.

В результаті аналізу були обрані і описані засоби розробки мобільних додатків. Як середовище, за допомогою якої було створено програму, була обрано платформа Firebase.

Підсумовуючи, скажемо, що використання платформи Firebase і її окремих функцій дозволяють нам зосередитися виключно на те, як наш додаток буде виглядати і наскільки зручним буде його використання.

1.3.2. Аналіз мов програмування та інструментальних засобів

В наш час для розробки мобільних додатків існують різноманітні мови програмування. Для розробки додатків для платформи Android частіше

використовуються мови Java, C# та C++. Кожна з них має свої особливості і створена для вирішення певних задач.

Мова Java спрощує життя розробників, встановлюючи жорсткі обмеження на способи передачі даних. Це помітно при порівнянні з C++, де данні можна передавати за значенням та за посиланням, а при наявності покажчиків код програми може стати дуже заплутаним.

Java виграє у C++ по багатьох критеріях і спрощує написання коду за рахунок відсутності заголовних файлів, покажчиків та множинного наслідування.

Мова C# ураховує досвід більш ранніх мов програмування (C++ та Java) [10, с.59]. Мови Java і C# є подібними. Їх можна розглядати як спробу вдосконалення C++. Властивості C# і Java:

- текст програми компілюється в проміжний код, і це не залежить від мови та платформи; надалі цей код виконується в спеціальному середовищі;

- Garbage Collection - автоматичний збір сміття;

- заборона на використання покажчиків. (C# допускає обмежене використання покажчиків, позначених як «ненадійні»;

- створення об’єктів за допомогою ключового слова new;

- виділення пам’яті здійснюється з купи (heap), яка знаходиться в середовищі виконання;

- багато потоковість, яка підтримується за допомогою блокування об’єктів;

- внутрішні класи;

- відсутність констант та глобальних функцій;

- рядки і масиви з перевіркою кордонів та вбудованою довжиною;

- оператори «->«,«::» не застосовують, во всіх випадках використовують оператор «.»;

- ключови слова - null і boolean/bool;

- для управління операторами if не використовують цілі числа (integers)

При досить великий схожості мов C# і Java, їх не можна ототожнювати. Мова C# сьогодні підтримується на багато меншій кількості платформ.
Перевагами мови Java є її незалежність від платформи, на якій виконується програма: один і той же код можна запускати на операційних системах Windows, Linux, Solaris, Machintosh та ін. Це необхідно при завантаженні програм через Інтернет для подальшого використання під управлінням різних операційних систем. Синтаксис мови Java є схожим на синтаксис мови C++, і програмісти, що знають мови C і C++, вивчають її без труднощів.

Крім того, Java є повністю об'єктно-орієнтованою мовою, навіть більш ніж C++. Всі значення в Java є об'єктами, за винятком небагатьох основних типів (primitive types), таких як числа.

Розробники забезпечили мову Java високою надійністю за допомогою методів, що дозволяють виключити навіть можливість створювати програми, з прихованими помилками. Для цього в мові Java виключена можливість явного звільнення пам'яті. Пам'ять звільняється тільки автоматично за допомогою механізму збірки сміття. Також у мові заборонена арифметика покажчиків та введені істинні масиви. Внаслідок неправильного використання покажчиків програмісти не можуть стерти дані з пам'яті. В Java виключена можливість переплутати оператор присвоювання з оператором порівняння. Наприклад, скомпілювати вираз if (entries = 3) тут не можна. Також поняття множинного наслідування тут замінене новим поняттям - інтерфейсом. Інтерфейс дає програмісту майже все, що той міг отримувати від множинного наслідування, при цьому уникаючи складнощів при управлінні ієрархіями класів.

Враховуючи всі вище вказані переваги мови Java, було вирішено здійснювати розробку мобільного додатку саме з її застосуванням.

1.3.3. Порівняльний аналіз середовищ програмування

Найпопулярнішими середовищами для розробки додатків на мові програмування Java є Android Studio, RAD Studio та Eclipse.

Android Studio - це інтегроване середовище розробки (IDE) для роботи з платформою Android, випущена компанією Google.

Архітектура системи Android складається з наступних рівнів:

- ядро операційної системи Linux.
- бібліотеки та система виконання.
- рівень каркаса додатків наділяє розробника доступом до інтерфейсу прикладного програмування API (application programming interface).
- рівень додатків - комплекс стандартних додатків.

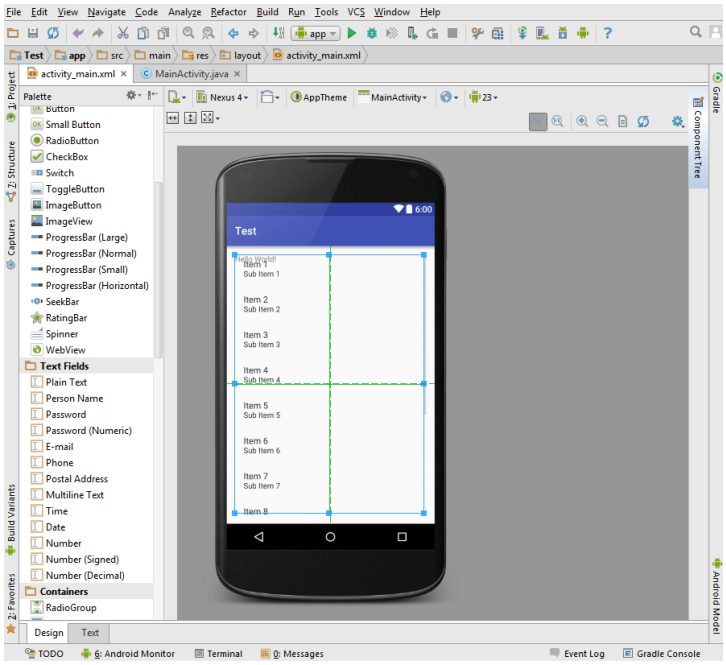


Рис. 1.2. – Каркас додатку

Як правило, розробник додатку працює з рівнями і каркасами додатків (рисунок 1.2). Від нього приховані ядро, система виконання та бібліотеки операційної системи Linux.

Найважливішим інструментом для розробки Android-додатків є Android SDK - комплекс засобів, що містить інструменти, необхідні для створення, збірки та компіляції мобільного додатку [19, с.8].

Створення програмного забезпечення найчастіше здійснюється із застосуванням IDE - середовища інтегрованої розробки. В IDE процес компіляції, збирання і запуску додатка є автоматизованим, що полегшує роботу і дозволяє початківцю програмісту без особливих зусиль створити свій перший додаток.

Два дуже популярних середовища розробки мобільних додатків під операційну систему Android, рекомендовані Google – це Android IDE та Android Studio.

Android IDE – це середовище розробки, засноване на інтегрованому середовищі розробки додатків Eclipse. В ньому є вбудовані інструменти для створення, компіляції та налагодження мобільних додатків.

Android Studio – це середовище, засноване на інтегрованому середовищі IntelliJ IDEA [25]. В Android Studio реалізовані:

- підтримка системи Gradle (система автоматичного складання);

- система рефакторінга коду, яка є унікальною;

- інструменти для пошуку різних проблем та їх усунення;

- попередній перегляд запущеного застосування відразу на декількох пристроях.

Як і Android IDE, Android Studio містить вбудовані інструменти для створення і налагодження мобільних додатків.

Підтримка хмарної платформи Google Cloud Platform.

В даний момент компанія Google припиняє підтримку інструментів для розробки в операційній системі Android для середовища Android IDE.

RAD Studio. Нова версія RAD Studio під назвою Berlin - це повноцінний інструмент розробки крос платформних додатків, в тому числі і мобільних програм, на мовах Object Pascal і C ++. Його головна перевага перед іншими аналогічними програмними середовищами полягає в тому, що він дозволяє

дуже швидко вести розробку за рахунок використання хмарних сервісів. Нові напрацювання цього середовища дозволяють в режимі реального часу бачити результат виконання програми і всі процеси, що відбуваються в додатку, що дозволяє говорити про точність розробки (рисунок 1.3).

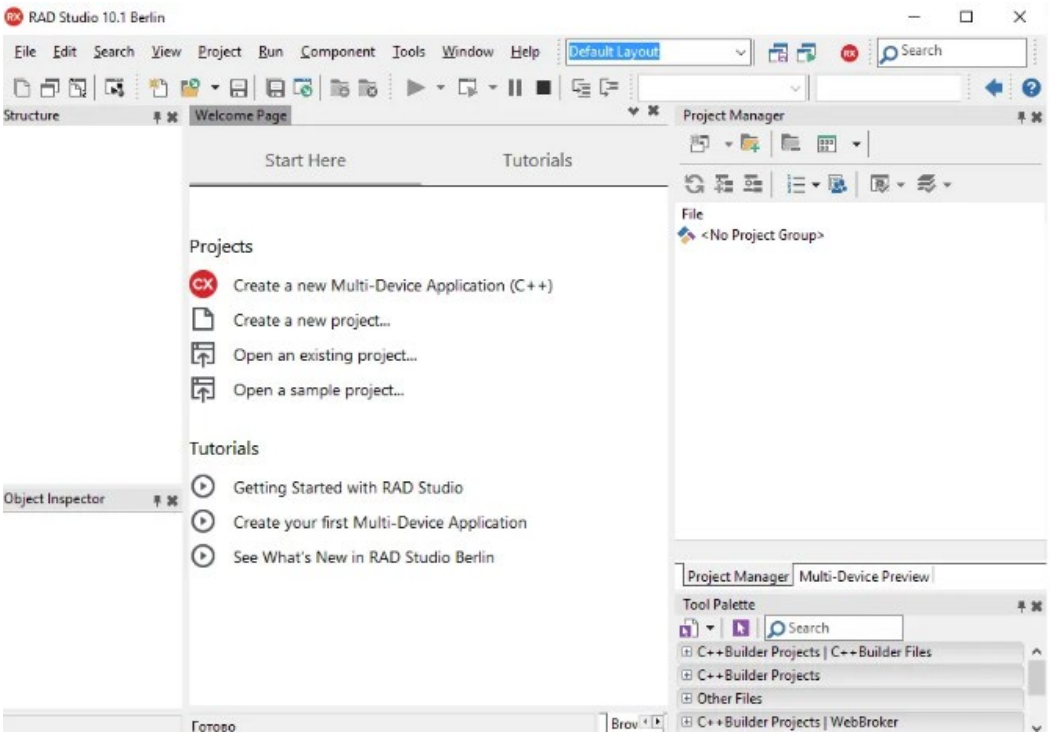


Рис. 1.3 – RAD Studio

Тут можна гнучко перемикатися з однієї платформи на іншу або на серверні сховища. Мінус RAD Studio Berlin – це платна ліцензія. Але при реєстрації можна отримати безкоштовну тріал-версію продукту на 30 днів. Інтерфейс - англійською.

Eclipse – одна з найпопулярніших програмних платформ з відкритим кодом для створення програмного забезпечення, в тому числі і мобільних. Серед головних переваг Eclipse - величезний набір API, для створення програмних модулів і використання RCP-підходу, що дозволяє написати практично будь-який додаток (рисунок 1.4).

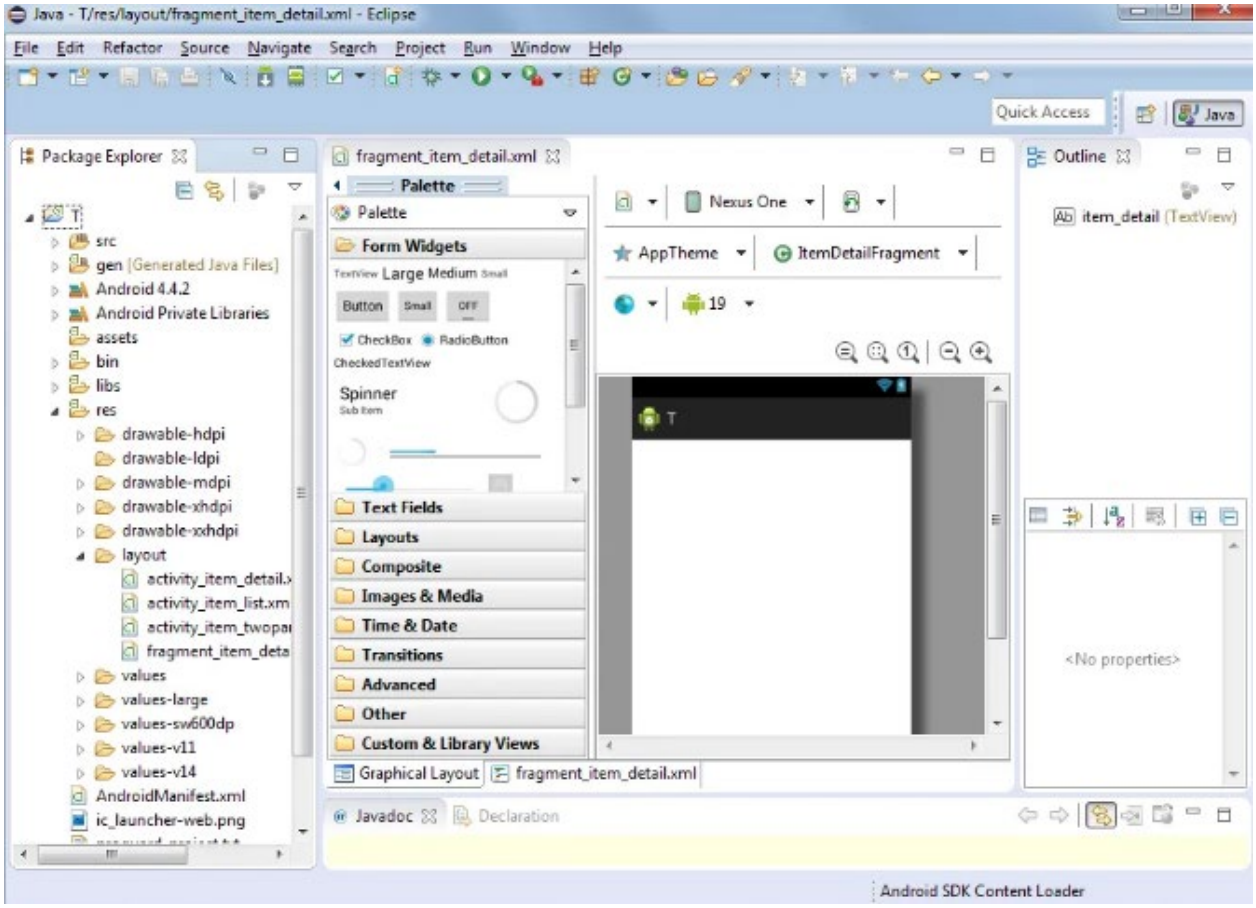


Рис. 1.4 – Eclipse

Ця платформа надає користувачам такі елементи комерційних IDE, як зручний редактор з підсвічуванням синтаксису, відладчик, що працює в потоковому режимі, класовий навігатор, менеджери файлів і проектів, системи контролю версій, рефакторинг коду. Особливо радує можливість додаткової установки необхідних для написання програми SDK. Але для використання Eclipse також доведеться вивчити англійську мову.

1.4. Постановка задач дослідження

В ході дослідження були поставлені наступні задачі:

- зробити аналіз особливостей розробки бази даних для мобільних додатків;

- виконати розробку структури бази даних;

- створити модель мобільного додатку;
- провести розробку структури мобільного додатку;
- розробити алгоритм функціонування мобільного додатку;
- розробити інтерфейс автоматизованої системи;
- реалізувати програму мобільного додатку;
- провести тестування розробленого програмного продукту.

1.5. Висновок до розділу 1

Розділ технічне обґрунтування доцільності розробки нового мобільного додатку є одним із головних розділів даної роботи. Цей розділ приводиться з метою з’ясування актуальності та необхідності розробки даного програмного продукту.

В розділі наведено особливості класифікації мобільних додатків та визначено, до якого типу належить додаток, що розроблюється. Проведено аналіз засобів та обґрунтовано вибір реалізації автоматизованої системи списку справ.

Проведено технічний аналіз основних аспектів розробки мобільних додатків, що дозволяє зробити висновок, що розробка цього програмного продукту є доцільною і своєчасною.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ І СТРУКТУР МОБІЛЬНОГО ДОДАТКУ

2.1. Аналіз основних принципів розробки мобільних додатків

В Android для опису дій використовуються спеціальні механізми засновані на Intent. При необхідності виконати якусь дію викликається Intent. Щоб провести обмін даними між додатками використовують Content Providers.

Ефективними методами програмування під Android є:

- розмітка RelativeLayout і її властивість «fill_parent»;
- порожні елементи розмітки TextField з нульовою висотою та шириною з параметром «centerInParent» для вирівнювання елементів по центру екрану;
- елегантні засоби доступу до даних, таки як values [SensorManager.DATA_X];
- методи onPause()/onResume() для збереження або закриття;
- кольорова розмітка об’єктів, що дозволяє виділити помилки і швидше їх знаходити;
- IDE, яка відповідає особистим вимогам розробника;
- команди Source - Format для форматування XML-файлів;
- плагін XML Tools для Notepad++ - для редагування XML-файлів і розуміння структури коду;
- пулі потоків (thread pools), вбудовани класи AsyncTask;
- шаблони (заздалегідь заготовлені шматочки коду), якими можна швидко скористатися для вставки в проект.

Принципи розробки продуктивних додатків під Android: протоколювати і аналізувати хід виконання додатку, економити апаратні ресурси, ефективно працювати з виділенням пам’яті, уникати зайвих об’єктів і створювати методи статичними.

2.2. Аналіз особливостей розробки бази даних

Сучасні ІТ будуються на принципах взаємодії з базами даних. Бази даних поширені всюди - від невеликих підприємств, які мають свої власні бази товарів, працівників, постачальників до баз даних глобального масштабу - держави чи кількох держав, як приклад - реєстраційні дані про людину чи машину. Бази даних дозволяють внести та упорядкувати дані щоб отримати ефективний пошук та взаємодію. Тому вони займають важливе місце на ринку програмних продуктів.

В розробленому нами додатку необхідно ефективно зберігати інформацію про справи на день та на тиждень. Щоб вирішити цю задачу необхідно використати концепцію баз даних.

База даних – сукупність даних, що зберігаються у відповідності зі схемою та відображають стан об’єктів та взаємозв’язків.

Бази даних є дуже зручним способом зберігання і користування інформацією. Їх важливість важко переоцінити. Бази даних необхідні у великій кількості закладів, установ, організацій та підприємств.

В Android використовується база даних SQLite. Це досить легка і швидка реляційна база даних. SQLite зберігає всю базу даних в єдиному стандартному файлі на тому пристрої, де виконується додаток.

Завдяки архітектурі движка можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Особливості SQLite:

- встановлення без конфігурації - не потребує ані установки, ані адміністрування;
- зберігається в одному крос платформному файлі;
- малий розмір коду;
- швидкість, простота у використанні API;

- прокоментований сирцевий код зі 100% тестовим покриттям гілок;
- автономність;
- крос платформність: підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE);
- поставляється з автономним інтерфейсом командного рядка, який може бути використаний для управління базами даних SQLite.

Розмірковуючи про побудову різних таблиць, потрібних в нашій базі даних, нам треба розглянути, як організувати ці таблиці та зв'язати одну з одною.

Головним чином, це буде зроблено за допомогою ключів.

В таблиці є 2 типи ключів - первинний ключ (унікальне значення для кожного запису, та зовнішній ключ (первинний ключ однієї таблиці, що знаходиться в іншій таблиці).

Переваги використання SQLite для розробника та користувача:

- встановлювати додаткове програмне забезпечення не потрібно;
- локальний файл в якому зберігаються дані, створює прозорість їх контролю (їх можна переглядати і редагувати незалежно від додатку);
- в коді використовують звичні SQL запити.

2.3. Розробка моделей мобільного додатку

Для роботи мобільного додатку необхідна локальна база даних, в якої буде зберігатися інформація про користувачів. Вона синхронізується з серверною базою в хмарному середовищі. Ця синхронізація здійснюється через web-сервіс за допомогою HTTP-запитів (при наявності підключення до інтернету). За допомогою мобільного додатку відображується інформація користувача, завдяки чому він може маніпулювати своїми даними.

2.3.1. Розробка структури бази даних

На етапі технічного проектування розробляються алгоритми і приймаються рішення по функціям, структурі, інтерфейсів і захисту програмного забезпечення. У процесі технічного проектування програми, призначеної для роботи з базою даних, вирішуються такі завдання.
проводиться класифікація і розподіл автоматизуються функцій по пунктах меню головного діалогового вікна користувача. В один клас слід включати функції, що об'єднуються загальною логікою застосування або загальними використовуваними об'єктами.

- виділяються найбільш уживані функції, які доцільно представити кнопками в загальних для багатьох екранних форм панелях інструментів.
- вибираються функції, для ефективної реалізації яких доцільно використовувати збережені процедури. Для складних функцій розробляються алгоритми, в необхідних випадках подаються блок-схемами і текстовими описами.
- вирішуються питання динаміки (зміни обсягу) БД. Визначаються умови і режим (автоматично або користувачем) архівування неактуальних даних. Створюється перелік функцій, необхідних для архівування бази і роботи з архівом.
- вводяться додаткові функції, що забезпечують захист даних від руйнування при програмних і апаратних відмовах, шляхом автоматичного або викликається користувачем копіювання інформації на резервний носій, а також відновлення БД з копії.
- у системах для багатьох користувачів визначаються повноваження (права) користувачів по роботі з функціями і даними. Встановлюється необхідний рівень ізоляції транзакцій, що виконують функції і збережені процедури. В необхідних випадках передбачаються індивідуальні блокування даних в операторах транзакцій.

- вирішується питання захисту програми від стороннього доступу і вибирається комплекс заходів по організації захисту програми і окремих функцій.

- визначається тип діалогу у вигляді одно- або багато документального інтерфейсу для функцій користувача.

- розробляються екранні форми для організації діалогу при виконанні функцій і структура створюваних програмою документів (звітів, довідок, листів і т.д.). Створення макетів екранних форм і звітів слід виконувати засобами візуального програмування, якщо вони є в обраному інструменті розробки, поєднуючи цим етапи технічного і робочого програмування.

- розробляються описи екранних форм і контекстних підказок, підключаються в якості оперативної допомоги (Help) користувачеві.

- продумується спосіб установки програмної системи на комп'ютер користувача і її переустановлення в разі відмови комп'ютера.

На етапі робочого проектування створюється структура програмної системи. Структура програми повинна будуватися за модульним принципом так, щоб самостійні функції були реалізовані окремими модулями або методами об'єктно-орієнтованого програмування і допускали автономні зміни. Функції розподіляються по програмним модулям.

Визначається спосіб виклику (запуску) головний програми користувачем. При цьому необхідно, щоб додаток, що змінює середу операційної системи і СУБД, відновлювало її після закінчення роботи.

Програмування починається зі створення призначеного для користувача меню і триває послідовної розробкою, підключенням і налагодженням окремих функцій і процедур, що зберігаються, починаючи з функцій введення і редагування даних.

Схема реляційної бази даних переходить у першу, другу, третю і т.д. нормальні форми. Якщо відношення відповідає критеріям нормальної форми,

та всім попереднім нормальним формам, тоді вважається, що це відношення знаходиться у нормальній формі.

Перша нормальна форма:

- кожна таблиця має основний ключ - мінімальний набір колонок, які ідентифікують запис;

- уникнете повторення груп (категорії даних, що можуть зустрічатись різну кількість разів у різних записах);

- кожен атрибут атомарен (повинен мати лише одне значення, а не множину значень);

Друга нормальна форма:

- схема бази даних відповідає вимогам першої нормальної форми;

- повторні дані, що з’являються в декількох рядках, виносяться в окремі таблиці;

Третя нормальна форма:

- схема бази даних відповідає всім вимогам другої нормальної форми;

- будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля виноситься в окрему таблицю.

У роботі базу даних приведено до третьої нормальної форми, тобто, тоді вона відповідає вимогам першої та другої нормальних форм.

2.3.2. Розробка ієрархічної моделі мобільного додатку

При обробці взаємодії між інтерфейсом користувача і його логікою Android слідує шаблону «Model-View-ViewModel» (MVVM).

Model-View-ViewModel – це архітектурний шаблон проектування додатків для розділення коду інтерфейсу користувача та іншого коду. За допомогою MVVM визначається інтерфейс користувача і використовується розмітка прив’язки даних, щоб пов’язати його з іншими рівнями, які містять дані та команди користувача.

Переваги шаблону MVVM:

- спрощене тестування модулів;

- можливе використання ітеративного, довільного стилю написання коду;

- використання інструментів проектування є більш ефективним.

Логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel. Функціональні зв’язки між інтерфейсом користувача і ViewModel реалізуються через Біндинги (bindings). Біндинги можуть бути написані в коді або визначені декларативним шляхом.

View – це базовий клас для всіх віджетів інтерфейсу користувача.

Клас Activity і його підкласи містять логіку, яка реалізує інтерфейс користувача. Цей клас відповідає ViewModel в архітектурному шаблоні ModelView-ViewModel (MVVM). Зазвичай кожен підклас Activity має тільки один пов’язаний з ним користувацький інтерфейс, і навпаки.

2.3.3. Розробка алгоритму роботи мобільного додатку

Алгоритм – це набір інструкцій, які описують порядок дій для досягнення результату за визначене число дій.

Для створення алгоритму необхідно знати:

- мету створення алгоритму (кінцевий стан об’єкта);

- повний набір вихідних даних (початковий стан об’єкта);

- систему команд виконавця (тобто набір команд, які він розуміє і може виконати).

Відомі два методи розробки складних алгоритмів:

- метод послідовної деталізації завдання («зверху-вниз»). Вихідна складна задача розбивається на під задачі. Кожна з підзадач розглядається і вирішується окремо. Якщо які-небудь з підзадач складні, вони також розбиваються на підзадачі. Процес продовжується до тих пір, поки підзадачі не зведуться до елементарних. Потім вирішення окремих підзадач збираються в

єдиний алгоритм розв'язання. Цей метод широко використовується, тому що він дозволяє вести розробку загального алгоритму одночасно кільком програмістам, які вирішують локальні підзадачі, що є необхідною умовою швидкої розробки програмних продуктів.

- складальний метод («знизу-вверх»). Він полягає у створенні безлічі програмних модулів, що реалізують рішення типових задач. При вирішенні складного завдання в якості допоміжних алгоритмів (процедур) програміст може використовувати розроблені модулі. Кожний алгоритм повинен відповідати ряду загальних вимог, таких як дискретність (алгоритм повинен представляти процес вирішення завдання як послідовне виконання деяких простих кроків), детермінованість (у кожен момент часу наступний крок роботи визначається станом системи і алгоритм видає один і той же результат (відповідь) для одних і тих же початкових даних), зрозумілість (алгоритм повинен включати тільки ті команди, які доступні виконавцю і входять в його систему команд), завершеність (алгоритм повинен завершувати роботу і видавати результат за кінцеве число кроків при коректно заданих початкових даних), результативність (завершення алгоритму певними результатами).

В залежності від мети, шляхів її вирішення, початкових умов завдання, та визначення дій виконавця алгоритми поділяються наступним чином:

- механічні алгоритми - задають певні дії, позначаючи їх в єдиній і достовірної послідовності. Тим самим забезпечується однозначний необхідний або шуканий результат, якщо виконуються ті умови завдання, для яких алгоритм був розроблений;

- гнучкі алгоритми (наприклад стохастичні, тобто імовірнісні та евристичні);

- імовірнісний алгоритм – який дає програму рішення задачі кількома шляхами або способами, що призводять до ймовірного досягнення результату;

- евристичний алгоритм (від грецького слова «еврика») - алгоритм, що використовує різні розумні міркування без якихсь строгих обґрунтувань;

- лінійний алгоритм - набір команд (вказівок), виконуємих послідовно в часі, один за одним;

- розгалужений алгоритм - алгоритм, який містить хоча б одну умову, в результаті перевірки якої може здійснюватися поділ на кілька гілок алгоритму;

- циклічний алгоритм – такий що передбачає багаторазове повторення одних і тих же операцій над новими вихідними даними. Цикл програми – це послідовність команд, яка може виконуватися багато разів (для нових початкових даних) до задоволення якоїсь умови;

Алгоритм розробленої нами автоматизованої системи - розгалужений, так як він містить декілька умов, в результаті перевірки яких здійснюється поділ на кілька паралельних гілок алгоритму.

2.4. Висновок до розділу 2

У цьому розділі було проаналізовано принципи розробки мобільних додатків та особливості розробки бази даних на платформі Android. Крім цього розроблено модель мобільного додатку та зроблено аналіз його архітектури з використанням шаблону Model-View-ViewModel, головною перевагою якого є розділення коду інтерфейсу користувача з іншим кодом. Виконано розробку структури бази даних та також наведено блок-схему алгоритму роботи блоку реєстрації та авторизації користувача, який належить до розгалуженого типу.

РОЗДІЛ 3. ЗАСОБИ РЕАЛІЗАЦІЇ СИСТЕМИ ГНУЧКОГО УПРАВЛІННЯ СПИСКОМ СПРАВ У МОБІЛЬНОМУ ДОДАТКУ

3.1. Програмне середовище клієнтських додатків

Для написання додатків системи управління розкладом було обрано програмне середовище Android Studio.

Android Studio – це редактор вихідного коду від компанії Google. Він надає можливості для підтримки налагодження, вбудованого керування Git, підсвічування синтаксису, інтелектуального завершення коду, фрагментування та рефакторингу коду. Також можна налаштувати, щоб користувач міг змінити редактор та поєднання клавіш. Це середовище – безкоштовне, хоча офіційне завантаження знаходиться під фірмовою ліцензією (рисунок 3.1.).

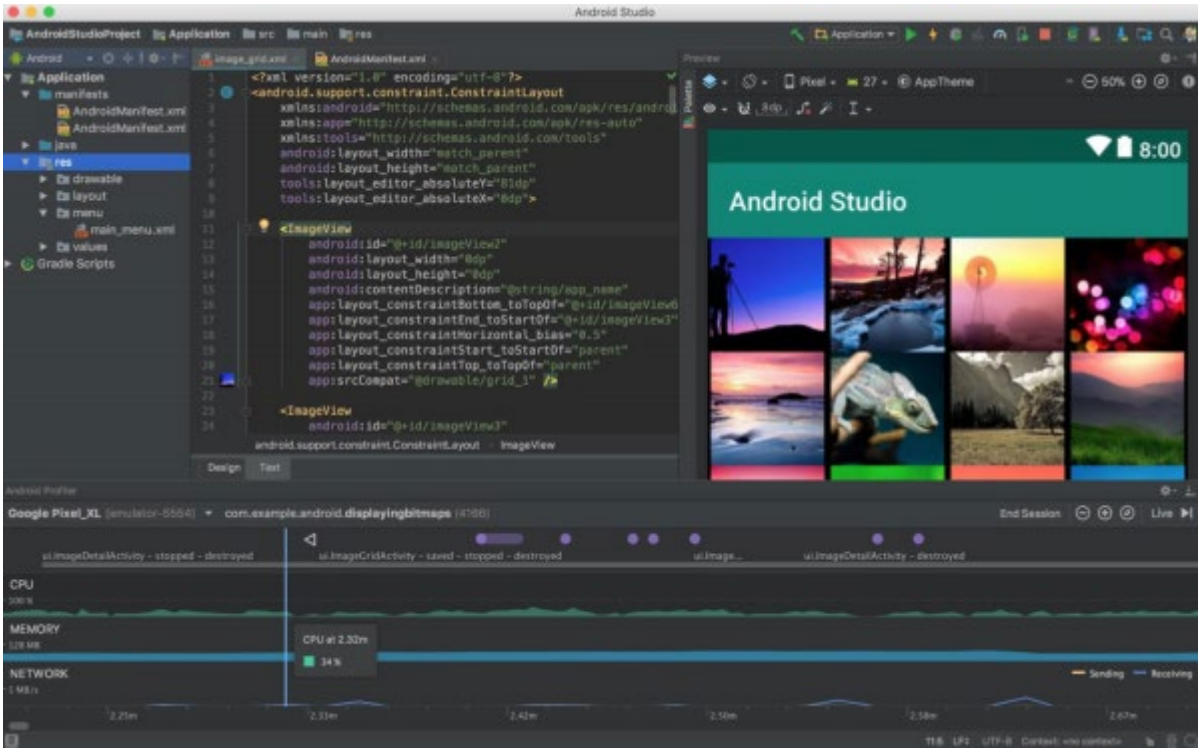


Рисунок 3.1 – Інтерфейс Android Studio

Додаткові переваги обраного програмного середовища:

- живі макети і подання програми в реальному часі;

- консоль з підказками щодо оптимізації, допомога з перекладом;
- орієнтований під Android рефакторинг коду та швидкі правки;
- утиліти для продуктивності, сумісності версій та інших проблем;
- шаблони для створення Android компонентів.

3.2. Вибір архітектури клієнтських додатків

Взявшись за написання програми, ми переконалися, наскільки важливо те, щоб програма не тільки добре працювала, але і була добре організована. Продумана архітектура потрібна не тільки великим проектам. Складність, як правило, зростає набагато швидше розмірів програми. І якщо не подбати про це заздалегідь, то досить швидко настає момент, коли ти перестаєш її контролювати. Правильна архітектура економить дуже багато сил, часу і грошей. А нерідко взагалі визначає те, чи виживе ваш проект чи ні.

Програму с гарною архітектурою легше розширювати і змінювати, а також тестувати, налагоджувати і розуміти. Можна сформулювати такий список цілком розумних і універсальних критеріїв:

- ефективність системи. В першу чергу програма повинна вирішувати поставлені завдання і добре виконувати свої функції, причому в різних умовах. Сюди можна віднести такі характеристики, як надійність, безпеку, продуктивність, здатність справлятися зі збільшенням навантаження (масштабованість) і т.п.- Гнучкість системи. Будь-який додаток доводиться міняти з часом - змінюються вимоги, додаються нові.

- можливість розширення системи. Можливість додавати в систему нові сутності та функції, не порушуючи її основної структури. На початковому етапі в систему має сенс закладати тільки основний і самий необхідний функціонал. Але при цьому архітектура повинна дозволяти легко нарощувати додатковий функціонал в міру необхідності.

Вимога, щоб архітектура системи володіла гнучкістю і розширюваністю навіть сформульована у вигляді окремого принципу -

«принципу відкритості / закритості» (Open-Closed Principle - другий з п’яти принципів SOLID): Програмні суті (класи, модулі, функції і т .п.) повинні бути відкритими для розширення, але закриті для модифікації. Іншими словами: Повинна бути можливість розширити (змінити) поведінку системи без зміни (листування) вже існуючих частин системи.- Масштабованість процесу розробки. Можливість скоротити термін розробки за рахунок додавання до проекту нових людей. Архітектура повинна дозволяти, щоб безліч людей могли працювати над програмою одночасно.

Над програмою, як правило, працює безліч людей. Після написання супроводжувати програму доводиться людям, які не брали участь в її розробці. Тому хороша архітектура повинна давати можливість відносно легко і швидко розібратися в системі новим людям. Все повинне бути добре структурованим, не містити дублювання, мати добре оформлений код і документацію.

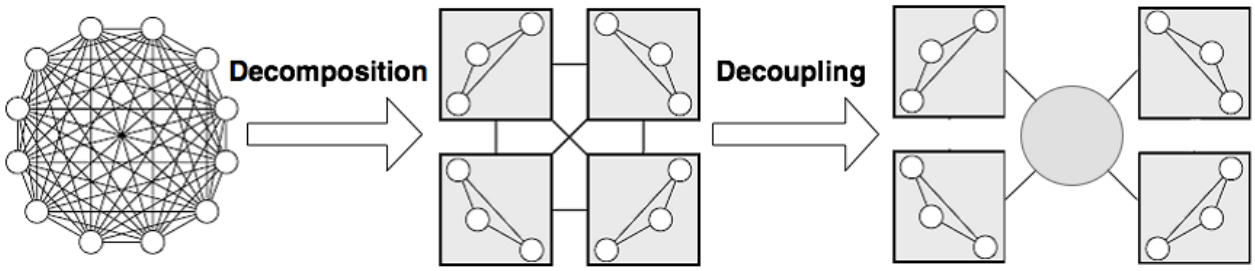


Рисунок 3.3 – Ієрархична декомпозиція

Коли мова йде про створення структури програми в більшості випадків маємо на увазі декомпозицію програми на підсистеми, які в свою чергу представляють: сервіси, підпрограми, функціональні модулі і організація їх взаємодії один з одним і зовнішнім світом (рисунок 3.3.).

3.3. Опис інструментів розробки

Для нашого мобільного додатку була обрана унікальна платформа Firebase. Користувачі Firebase оцінюють його високу продуктивність і масштабованість. Платформа пропонує безкоштовний план, який допомагає познайомитися з серверної платформою.

Основні особливості Firebase:

- база даних в реальному часі
- масштабований хостинг
- аналітика
- звіт про збої
- місце зберігання

На рис. 3.4 зображена план-схема завантаження картинки на сервер Firebase.

Рисунок 3.4 – Завантаження картинки на сервер Firebase

Для клієнтського рівня було використано такий набір технологій: мова програмування Java, фреймворк для побудови android-додатків Android SDK, а також додаткові бібліотеки - Room, Android Jetpack, Koin, Android Arch. В якості IDE використовувалась Android Studio.

3.4. Обґрунтування вибору програмної реалізації

При проектуванні системи було вивчено та проаналізовано предметну область та вимоги замовника.

Технології, які ми вирішили використовувати на сервері, були нами обрані за принципом зручності у їх використанні, а також повній відкритості вихідних кодів. Платформа Firebase дає змогу створювати програми на будь-якій операційній системі.

При створенні локальної бази даних було використано ORM Room, бібліотеку яку сам Google радить використовувати як БД, а також через зручність та універсальність цього фреймворку. За його допомогою легко абстрагуватися від конкретної реалізації бази даних і створити відповідні моделі. При цьому вся увага зосереджується на створенні якісної і протестованої бізнес-логіки.

Перелічені технології разом дозволяють збудувати якісний та надійний продукт, захищений від патентних оспорювань з боку розробників, тому що ці технології мають ліцензії, які виключають таку можливість і надають доступ до кодів даних проектів.

3.5. Висновок до розділу 3

В даному розділі було розглянуто Android Framework і його функціонал. Нами було розглянуто середовище розробки клієнтських додатків Android Studio та його допоміжні інструменти.

Було наведено переваги щодо використання мови програмування Java як основної мови для розробки клієнтських додатків. У цьому розділі було обрано архітектурну платформу для побудови розроблюваної системи.

Було вирішено будувати систему на базі платформи Firebase. Обґрунтовано саме її використання через деталі реалізації клієнтських додатків і загальну орієнтованість на мобільні платформи. Розглянуто переваги бази даних NoSQL та обрані певні інструменти.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ДОДАТКУ НА ANDROID

4.1 Розробка бази даних автоматизованої системи

В Android є вбудована підтримка однієї з поширених систем управління базами даних - SQLite. Для цього в пакеті android.database.sqlite визначено набір класів, які дозволяють працювати з базами даних SQLite. І кожен додаток може створити свою базу даних. Щоб використовувати SQLite в Android, треба створити базу даних за допомогою вираження на мові SQL. Після цього база даних буде зберігатися в каталозі додатки шляхом:

```
1 DATA/data/[Назва_додатку]/databases/[Назва_файлу_бази_даних]
```

ОС Android за замовчуванням вже містить ряд вбудованих бад SQLite, які використовуються стандартними програмами - для списку контактів, для зберігання фотографій з камери, музичних альбомів і т.д.

Основну функціональність по роботі з базами даних надає пакет android.database. Функціональність безпосередньо для роботи з SQLite знаходиться в пакеті android.database.sqlite. База даних в SQLite представлена класом android.database.sqlite.SQLiteDatabase. Він дозволяє виконувати запити до бд, виконувати з нею різні маніпуляції. Клас android.database.sqlite.SQLiteCursor надає запит і дозволяє повертати набір рядків, які відповідають цьому запиту.

Клас android.database.sqlite.SQLiteQueryBuilder дозволяє створювати SQL-запити. Самі sql-вирази представлені класом android.database.sqlite.SQLiteStatement, які дозволяють за допомогою плейсхолдеров вставляти в вираження динамічні дані.

Клас android.database.sqlite.SQLiteOpenHelper дозволяє створити базу даних з усіма таблицями, якщо їх ще не існує.

У SQLite застосовується наступна система типів даних:

INTEGER: представляє ціле число, аналог типу int в java

REAL: представляє число з плаваючою точкою, аналог float і double в java

TEXT: представляє набір символів, аналог String і char в java

BLOB: представляє масив бінарних даних, наприклад, зображення, аналог типу int в java

Збережені дані повинні подавати відповідні типи в java.

4.2 Розробка мобільного додатку

4.2.1 Реалізація логіки програмного продукту

Для розробки Android додатку головним класом є клас Activity.

Для того щоб створити або відкрити нову БД з коду Activity в Android у нас є можливість викликати відомий метод openOrCreateDatabase ().

Працюючи з класом Activity, спочатку об’єкт створюється, потім запускається, відпрацьовується та знищується.

Протягом своего життєвого циклу клас Activity може перебувати в одному з 3-х станів:

- активне. Activity знаходиться на передньому плані і користувач взаємодіє з нею.

- призупинене. Activity закрита частково.

- завершене. В цьому випадку Activity прихована користувач її не бачe.

Процес роботи класу Activity наведено на рис. 4.1.

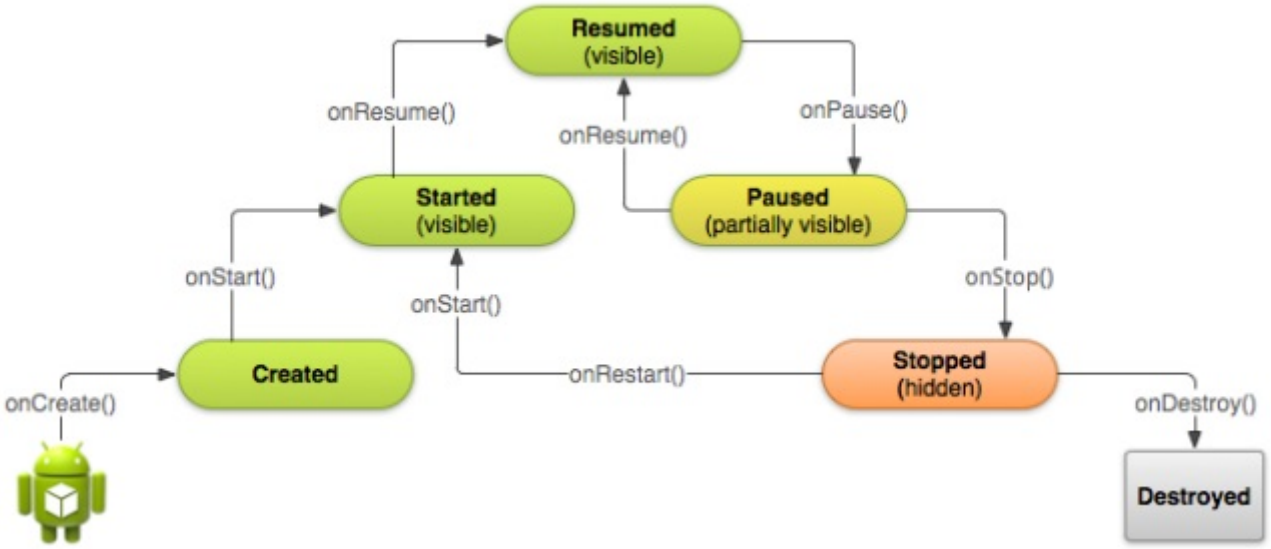


Рисунок 4.1 – Процес роботи класу Activity

Викликаємо onCreate (), потім викликається onStart (), як результат переходить до onResume ().

Після того як запустився додаток, користувач бачить LoginActivity, в нашому випадку це вікно авторизації користувача.

Пер за все проводиться перевірка полів для вводу даних за допомогою if-else. Після того як поля заповнені, викликається функція для перевірки логіна та пароля. Використовуючи клас StringRequest, посилається запит в базу на перевірку коректності введених даних, яко дані є коректні, то відкривається кабінет користувача.

4.2.2 Реалізація розмітки Android-додатку

Основна ідея - паралельно дереву об’єктів розмітки (View), яке генерує Android при завантаженні Layout, програма створює дзеркальне дерево власних об’єктів (однойменні класи з префіксом Z - Zview, ZButton, ...). Кожен з них має посилання на об’єкт - оригінал. Класи мають спадкування, аналогічне View, і підтримують весь функціонал, пов’язаний з обходом дерева, наприклад, при генерації xml-файлів розмітки, щоб створити список параметрів View. Крім того, дзеркальне дерево є основною структурою даних

для опису відкритого Layout: кожен об’єкт містить два важливі параметри - вектор параметрів розмітки для View і дзеркальний об’єкт-описувач для LayoutParams (рисунок 4.2).

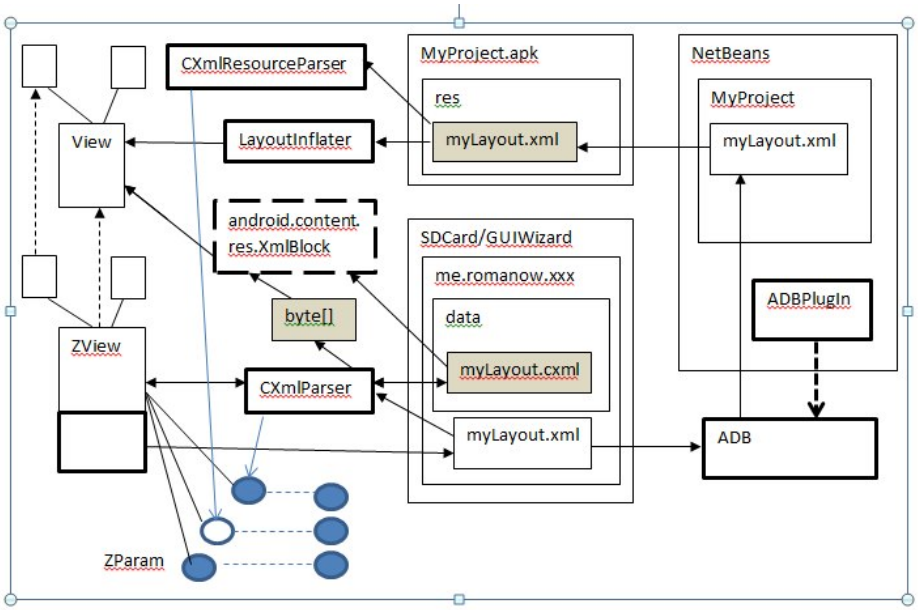


Рисунок 4.2 – Зеркальне дерево об’єктів розмітки

Початкове різноманіття параметрів View підтримується групою класів - спадкоємців ZParam, кожен з них реалізує всю специфіку зовнішнього і внутрішнього представлення параметрів даного типу, конвертації, виклику діалогу його редагування, визначення відповідності типам значень в скомпільованому файлі. На даний момент реалізовані типи - int, hex (16-ковий int), boolean, float, string, charSequence, dimen (розмірності), id (ідентифікатор ресурсу), image, color, enumInt (список іменованих цілих констант), enumString, enumClass (список іменованих цілих класів). В останньому випадку для кожного enum-класу створюється свій похідний клас.

Вектор параметрів для кожного View десеріалізуется з відповідного xml-файлу, розміщеного в assets. Щоб спростити процедуру створення таких файлів, в додатку є компонента, яка для всіх View шукає геттери / сеттери і по ним створює опис передбачуваних параметрів. Потім вже ці файли редагуються вручну по документації.

Для кожного View читається LayoutParams (його тип залежить від типу предка), потім за допомогою рефлексії для нього генерується однойменний клас описувателя - ZLP і його похідні, кожен клас створює вектор параметрів і записує в нього значення, лічені з відповідного LayoutParams.

Скомпільований xml-файл Layout-a. У файлі Android-додатки (apk) xml-файли опису Layout містяться в скомпільованому вигляді.

Внутрішній парсер Android - клас android.content. res.XmlBlock здатний створювати View з байтного масиву, що містить дані в аналогічному форматі.

4.3 Розробка інтерфейсу програмного продукту

Розробка дизайну і інтерфейсу мобільного додатку є найважливішим етапом створення програмного продукту. Можна створити класний додаток з широким функціоналом і можливостями. Але він не матиме успіху, якщо в ньому неправильно підібрані кольори, незручно розміщені кнопки, іконка ніяк не виділяється. Тому розробляється інтерфейс мобільного додатка, що дозволяє зробити візуальну частину зручною, простою, привабливою і логічною.

Інтерфейс це інструмент взаємодії між користувачем і програмою. Важливо зробити так, щоб користувач легко розібрався в функціоналі, візуал був приємний і не відштовхував, а на будь-якому пристрої програма працювала коректно.

Інтерфейс мобільного додатка відповідає за те, щоб всі вищевказані умови виконувалися. Це комплекс, що дозволяє чітко зрозуміти, як має виглядати додаток, де розташовуються всі елементи, яка логіка програми. Орієнтованість на простоту, інтерактивність, залучення, легкість освоєння – це основні критерії.

Щоб спроектувати інтерфейс мобільного додатка, необхідно чітко слідувати правилам і законам побудови логіки додатків. Проектування інтерфейсу мобільного додатка складається з декількох ключових частин.

- розуміння користувачів. Спочатку необхідно зрозуміти логіку користувача і його потреби. Для цього створюється «персонаж», у якого повинно бути ім’я, вік, статус, його звички, потреби, інтереси. На основі взаємодії з таким «персонажем» створюється призначений для користувача сценарій, який передбачає поведінку клієнта.

- орієнтація на поведінкові шаблони, звички і неписані стандарти. Є ряд шаблонів поведінки, які допоможуть розташовувати елементи правильно. Також слід враховувати особливості жестів, анімації, анатомічні фактори. Як користувач тримає смартфон, куди дивиться, як натискає на екран.

- застосування інтерактивного підходу. Він дозволяє зробити проектування інтерфейсу мобільного додатка швидким і актуальним. Суть інтерактивного підходу полягає в тому, щоб спочатку необхідно продумати мінімальний функціонал з найголовнішими інструментами, а після його розширювати.

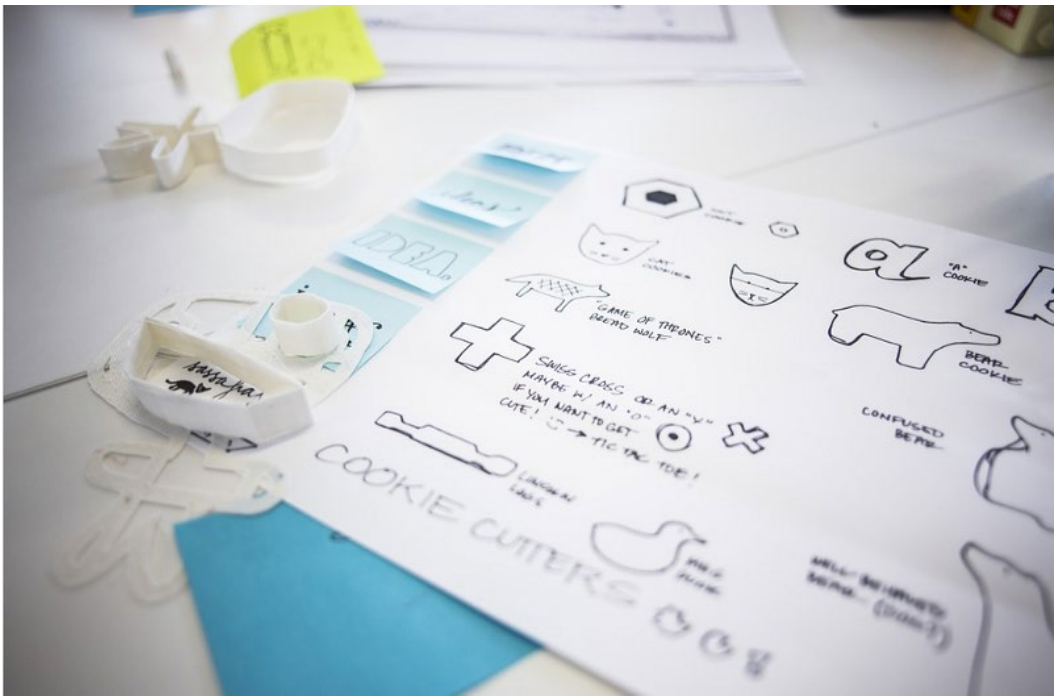


Рис. 4.3 – Створення ескізу інтерфейсу

Принципи розробки інтерфейсу мобільного веб додатку:

- додаток має бути унікальним і виділятися серед конкурентів;

- енергозбереження, легкість в управлінні, збереження дій - все це показує турботу про користувача;
- облік користувацького досвіду, розташування елементів на зручному рівні, наприклад, кнопки «видалити» і «редагувати» повинні бути на такій відстані, щоб не зачепити випадково одну з них;
- опрацювання відгуку, користувач повинен розуміти, що його запит виконується, відбуваються якісь дії;
- легке введення тексту з урахуванням особливостей мобільного клавіатури;
- помітна і приваблива іконка;
- упор на роботу з сенсорним екраном, все елементи повинні бути зрозумілі і легко взаємодіяти;
- мінімум спливаючих вікон.

Основні вимоги інтерфейсу мобільного додатка:

- Використання звичних UI елементів - вертикальна стрічка новин, прямокутні кнопки, розташування меню;
- високий рівень візуалізації, що дозволяє гармонійно перебувати в додатку;
- зниження рівня «шуму» інтерфейсу, важливі елементи мають бути показані на самому початку і бути великими і видимими;
- наявність призову до дії, де він необхідний;
- зручний висновок даних, наприклад, округлені ціни;
- поступове запрошування прав, наприклад, поки користувач не захоче відкрити камеру в додатку, не потрібно заздалегідь запитувати доступ до неї;
- явний показ можливостей, щоб користувач відразу зрозумів, що він отримає;
- можливість індивідуальної настройки під потреби користувача.

Виконуючи ці вимоги, додаток вийде ефективним, зрозумілим і корисним. Залишається тільки зробити акцент на візуальному супроводі, щоб в програмі було приємно перебувати і хотілося її відкривати.

Під час розроблення мобільного додатку було дотримано всіх вище наведених правил та спроектовано оптимальний на вигляд і для застосування інтуїтивно зрозумілий інтерфейс.

4.4. Висновки

У розділі розроблено базу даних мобільного додатку, реалізовано логіку автоматизованої системи та розроблено розмітку екрану Android-додатку. Розроблено інтуїтивно-зрозумілий інтерфейс користувача та визначено, що він належить до діалогу типу меню.

РОЗДІЛ 5. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мета тестування програмного забезпечення є випуск програмного забезпечення (ПЗ) високої якості. Тестування ПЗ виконується на вимогу замовників та сприяє виявленню інформації про якість продукту.

Процес тестування включає планування, проектування і, власне, виконання тестів.

Основні типи тестування:

- функціональне тестування. Направлене на контроль відповідностей функціональних вимог програмного забезпечення з реальними характеристиками.

- нефункціональне тестування. Використовується для визначення характеристик системи.

Залежно від того, чи використовуються додаткові програмні засоби для тестування додатків або програм, тестування буває: ручне, мануальне, та автоматизоване.

Ці типи тестування нерідко проводяться паралельно. Адже працюючи над певною функціональністю простіше оцінити її поведінку і в стандартних, і в нестандартних умовах.

В процесі тестування інженер може працювати з ПО, не звертаючись до його коду, а може визначити правильність роботи, глянувши на код. За доступом до коду програмного продукту тестування ділиться на:

- тестування «білого ящика» - з доступом до коду.
- тестування «чорного ящика» - без доступу до коду продукту.
- тестування «сірого ящика» - на основі обмеженого знання внутрішньої структури ПО. Часто кажуть, що це суміш тестування «білого ящика» і «чорного ящика», але це в корені невірно. В даному випадку тестувальник не працює з кодом програмного

продукту, але він знайомий з внутрішньою структурою програми і взаємодією між компонентами.

Тестування програмного коду – це процес виконання програмного коду, спрямований на виявлення в ньому дефектів. Під дефектом тут розуміється ділянка програмного коду, виконання якого призводить до несподіваної поведінки системи, яка може давати збої і відмови у її роботі. В цьому випадку говорять про суттєві дефекти програмного коду. Деякі дефекти викликають незначні проблеми, що не порушують процес функціонування системи, але ускладнюють роботу з нею. В цьому випадку говорять про середні або малозначні дефекти.

Якість програмного продукту визначається за кількома критеріями. Якісний програмний продукт повинен відповідати функціональним і нефункціональним вимогам, відповідно до яких він створювався, мати цінність для бізнесу, відповідати очікуванням і потребам користувачів.

Важливим аспектом створення якісного ПЗ є забезпечення таких вимог як зручність в експлуатації, надійність, продуктивність, захищеність, зручність супроводу.

Тестування програмного продукту дозволяє протягом усього його життєвого циклу гарантувати, що програмні проекти відповідають заданим параметрам якості.

5.1. Аналіз методів тестування

5.1.1. Тестування Чорної Скриньки

BLACK BOX TESTING (BBT) – методика тестування, при якій функціональність програми тестується без урахування внутрішньої структури коду. BBT засноване повністю на вимогах та специфікаціях ПЗ (рисунок 5.1).

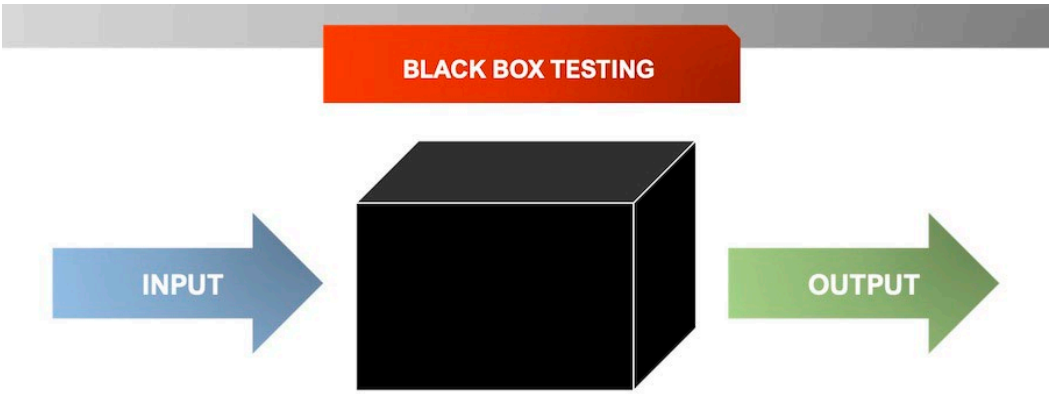


Рис. 5.1 – Тестування Чорної Скриньки

Ось алгоритм проведення тестування ВВТ:

- розглядаємо вимоги і специфікації системи.
- обираємо позитивний сценарій тестування
- визначаємо результати для всіх цих входів.
- створюємо тестові набори з вибраними входами.
- виконуємо тестові випадки.
- порівнюємо фактичні результати з результатами які ми очікували.
- якщо знаходимо дефекти, то вони виправляються і тестуються ще раз.

5.1.2. Тестування Білої Скриньки

WHITE BOX TESTING - це тестування внутрішньої структури, дизайну і кодування програмного рішення. У цьому типі тестування код видно тестувальнику. Основна увага приділяється перевірці потоку вхідних і вихідних даних через додаток, поліпшенню дизайну та зручності використання, посилення безпеки. Тестування білого ящика також відомо як тестування Clear Box, тестування Open Box, структурне тестування, тестування прозорого боксу, тестування на основі коду і тестування Glass Box. Це зазвичай виконується розробниками (рисунок 5.2).

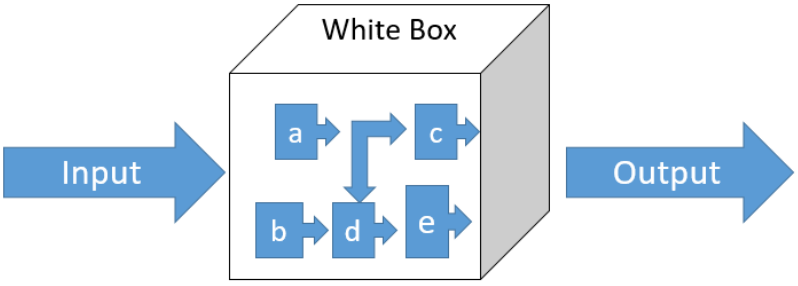


Рис. 5.2 – Тестування Білої Скриньки

При тестуванні перевіряються:

- внутрішні діри в безпеці
- зламані або погано структуровані шляхи в процесах кодування
- потік конкретних входів через код
- очікуваний результат
- функціональність умовних циклів
- тестування кожного оператора, об’єкта і функції на індивідуальній основі

Тестування може проводитися на системному, інтеграційному і модульному рівнях розробки програмного забезпечення. Однією з основних цілей тестування whitebox є перевірка робочого процесу для додатку.

5.2. Тестування з використанням емулятора

При тестуванні мобільних додатків можна зіткнутися з проблемою, коли в наявності немає потрібного пристрою, на який прийшов запит від клієнта. Можна вийти з цієї ситуації і виконати необхідну задачу використовуючи емулятори мобільних ОС.

Емулятори мобільних телефонів допомагають перевіряти роботу функціоналу додатка на різних мобільних платформах, емулюють програмне забезпечення і дозволяють створити необхідні перевірки.

У разі iOS-пристроїв все складно. Повноцінних емуляторів даної платформи не існує через закритість її архітектури.

Для тестування пристроїв компанії Apple (iPhone, iPad) є кілька варіантів. Перший серед них - офіційний Apple iOS Simulator, що входить в поставку Xcode. Дозволяє тестувати різні комбінації ПО та пристроїв, але тільки на Mac. В цьому і полягає головний мінус. Якщо немає доступу до MacBook - не буде можливості використовувати даний симулятор. Для запуску симулятора необхідно встановити і запустити Xcode.

У випадку з Android-пристроями все набагато простіше. Різних варіантів емуляторів даної платформи існує безліч. Розглянемо деякі приклади.

- Google Android Emulator. Цей емулятор запускається на Windows як окремий додаток без необхідності повністю завантажувати та встановлювати Android SDK.
- офіційний Android SDK Emulator. Він включає в себе емулятор мобільного пристрою, який реалізує всі апаратні і програмні особливості типового пристрою.
- MobiOne. Цей емулятор допомагає розробнику програмувати, тестувати, налагоджувати упаковувати і впроваджувати мобільні веб-додатки на різних пристроях, таких як iPhone, BlackBerry, та на Android і Palm Pre.
- TestiPhone. Заснований на веб-браузері, для швидкого тестування веб-додатків для iPhone. Працює з використанням Internet Explorer 7, Firefox 2 і Safari 3.
- BlackBerry Simulator. Існує безліч офіційних емуляторів BlackBerry. З будь-яким з них можливо перевірити, як ПО, екран, клавіатура будуть працювати з додатком.

Для платформи Android найпопулярнішими емуляторами є: Bluestacks, Android SDK Emulator, Andy, Youwave, Oracle VM VirtualBox, GenyMotion. Нижче наведено їх переваги та недоліки.

Android SDK Emulator має такі переваги:

- різні налаштування, необхідні для тестування,

- можливість протестувати додаток на різних версіях ОС Android, на пристроях з різними типами дисплея;

- емуляція SD-карти.

З недоліків можна позначити такі:

- емулятор працює дуже повільно;

- є великий проміжок часу між натисканням кнопки «Run» і запуском програми на емуляторі.

Після аналізу переваг та недоліків різних емуляторів, було вирішено використовувати Android SDK Emulator, так як він найкраще підходить для розробки додатків на ОС Windows x64.

5.3. Тестування автоматизованої системи

Для тестування нашій автоматизованої системи було застосовано функціональний вид тестування на Android SDK Emulator. При даному підході програма розглядається як «чорна скринька», текст програми не доступний.

Мета тестування автоматизованої системи - це визначення того, наскільки правильно вона виконує свої функції. У нашому випадку додаток повинен надавати інформацію про зареєстрованого користувача. Якщо ж користувач не зареєстрований – у нього повинна бути можливість зареєструватися.

Після запуску мобільного додатку з’являється вікно авторизації, в якому користувачу пропонується ввести свій логін та пароль. Якщо користувач не є зареєстрованим, йому пропонується перейти на вікно реєстрації. Якщо реєстрація пройшла успішно, на екрані повинно з’явитися про це повідомлення.

При реєстрації з’являється модальне вікно з повідомленням про успішну реєстрацію.

Додаток видає правильний результат, отже, тестування функції реєстрації користувача пройшло успішно.

5.4. Висновок до розділу 5

У розділі проаналізовано існуючі методи тестування програмного забезпечення, та проведено порівняльний аналіз популярних емуляторів для тестування. У якості тестуючого пристрою обрано Android SDK Emulator. Тестування системи проведено функціональним методом, під час якого код програми не доступний, і вона розглядається як «чорна скринька».

Тестування підтвердило ефективність та правильність функціонування розробленої автоматизованої системи. Розроблений нами програмний продукт має привабливий та інтуїтивно зрозумілий інтерфейс та є легким у використанні.

ВИСНОВКИ

Проаналізувавши все вищевикладене можемо зазначити, що на сучасному етапі розвитку комп'ютерних технологій існує велика кількість мов програмування для розробки мобільних додатків. Для розробки додатків під платформу Android найчастіше використовують мову Java, C# та C++. Кожна з цих мов програмування має свої особливості і створена для вирішення задач певного типу.

Для розробки додатків на мові програмування Java найпопулярніші є Android Studio, RAD Studio, Eclipse.

Android Studio - це інтегроване середовище розробки (IDE) для роботи з платформою Android, випущена компанією Google.

Android дозволяє фонове виконання якої-небудь дії, підтримує двовимірну і тривимірну графіку, доступ до файлової системи і бази даних, забезпечує велику бібліотеку елементів призначеного для користувача інтерфейсу.

Одним з найважливіших інструментів для розробки Android-додатків є універсальний засіб розробки мобільних додатків для операційної системи Android (Android SDK) – комплекс засобів програмування, що містить інструменти, які необхідні для створення, компіляції і збірки мобільного додатка.

В розробленій автоматизованій системі необхідно ефективно зберігати інформації про справи на день. Для вирішення цієї задачі у мобільному додатку необхідно використати концепцію баз даних.

Однією з обов'язкових умов при розробці баз даних є її нормалізація. Нормалізація схеми бази даних – покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Для створення системи гнучкого управління розпорядком дня було розглянуто Android Framework та функціонал який він надає. Було детально розглянуто середовище розробки клієнтських додатків Android Studio та його допоміжні інструменти.

Вирішено будувати систему на базі платформи Firebase. Обґрунтовано використання саме цієї платформи через деталі реалізації клієнтських додатків і загальну орієнтованість системи на мобільні платформи. Розглянуто переваги бази даних NoSQL. Окрім того було розглянуто середовище для розробки тригерів на базі архітектурної платформи Firebase.

При розробці додатку під Android головним класом є клас Activity. Він представляє візуальну активність додатку і визначає дії, які може виконувати користувач. У процесі роботи класу Activity, спочатку створюється об’єкт класу Activity, потім він запускається, відпрацьовує та знищується, а користувач смартфона переходить до нового об’єкта.

Під час розробки Android-додатку, візуальна частина вікон реалізується окремо від логіки програми. Код розмітки екрану знаходиться у файлах з розширенням .xmly папці проекту, яка має назву «layout».

Розмітка для Android програми будується з використанням ієрархії елементів View і ViewGroup. View елементи це віджети для інтерфейсу користувача, такі як кнопки або текстові поля і ViewGroup - це невидимий вид контейнерів, які визначають розташування дочірніх представлень, як наприклад, в сітці або вертикальному списку.

Розробка дизайну і інтерфейсу мобільного додатку – найважливіший етап створення програмного продукту. Від даної роботи залежить те, як користувач сприйме додаток, чи сподобається воно йому, чи буде додаток зручним в експлуатації, і чи стане він популярним.

Тестування мобільних додатків відбувається в середовищі розробки з використанням емулятора. Після цього додаток тестується на пристрої.

Для платформи Android найпопулярнішими емуляторами є: Bluestacks, Android SDK Emulator, Andy, Youwave, Oracle VM VirtualBox, GenyMotion.

Розроблений нами додаток з використанням архітектурних компонентів від Google на Android дозволяє ефективно здійснювати планування власних справ. Нам вдалося створити працюючий мобільний додаток який має великий потенціал на розширення функціоналу, та може зайняти свою нішу серед схожих додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Балдин К. В. Информационные технологии в менеджменте / К. В. Балдин – М.: Академия, 2012. – 288 с.
2. Белов В. В. Проектирование информационных систем / В. В. Белов – М.: Академия, 2013. – 352 с.
3. Бурнет Эд. Разработка мобильных приложений / Эд. Бурнет – СПБ.: Питер, 2012. – 256 с.
4. Гагарина Л. Г. Технология разработки программного обеспечения / Л. Г. Гагарина – М.: ИНФРА-М, 2009. – 400 с.
5. Гарсиа-Молина Г. Системы баз данных. Полный курс = Database Systems: The Complete Book / Г. Гарсиа-Молина – Вильямс, 2003. – 1088 с.
6. Голощапов А. Google android. Системные компоненты и сетевые коммуникации / А. Голощапов – СПб.: БХВ-Петербург, 2012. – 384 с.
7. Голощапов А. Google android. Создание приложений для смартфонов и планшетных ПК / А. Голощапов – СПб.: БХВ-Петербург, 2013. – 832 с.
8. Горбань А. Г. Програмування в Java / А. Г. Горбань 2008. – 310 с.
9. Дейтел П. Android для разработчиков / П. Дейтел – СПб.:Питер, 2016. – 512 с.
10. Дэвид Гриффитс Head first. Программирование для android / Дэвид Гриффитс, Дон Гриффитс. – СПб.: Питер, 2016. – 704 с.
11. Затонский А. В. Информационные технологии / А. В. Затонский – Академия, 2001. – 217с.
12. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика = Database Systems: A Practical Approach to Design, Implementation, and Management / Т. Коннолли М.: Вильямс, 2003. – 1436 с.

13. Михеева Е. В. Информационные технологии в профессиональной деятельности / Е. В. Михеева – М.: Проспект, 2010. – 448 с

14. Орлов С. А. Теория и практика языков программирования / С. А. Орлов СПб.: Питер, 2017. – 688 с.

15. Фелкер Д. Android. Разработка приложений для чайников / Д. Фелкер. М.: Диалектика, 2012. – 336 с.

16. Філліпс Б. Android. Програмування для професіоналів / Б. Філліпс СПб.: Питер, 2017. – 688 с.

17. Харди Б. Android програмирование для профессионалов / Б. Харди – СПБ.: Питер, 2016. – 636 с.

18. Хашими С. Разработка приложений для Android / С. Хашими – СПб.: Питер, 2011. – 736 с.

19. Цехнер Марио – Программирование под Android / Марио Цехнер М.: Питер, 2012. – 688 с.

20. Черников Б. В. Управление качеством программного обеспечения / Б. В. Черников. В. М.: Форум; Инфра-М, 2012. – 240 с.

21. Шелухин О. И. Моделирование информационных систем / О. И. Шелухин – М.: Горячая линия - Телеком, 2011. – 536 с.

22. Шматко О. В. Аналіз методів і технологій розробки мобільних додатків для платформи Android / О. В. Шматко. – Харків: НТУ «ХПІ», 2018. – 284 с.

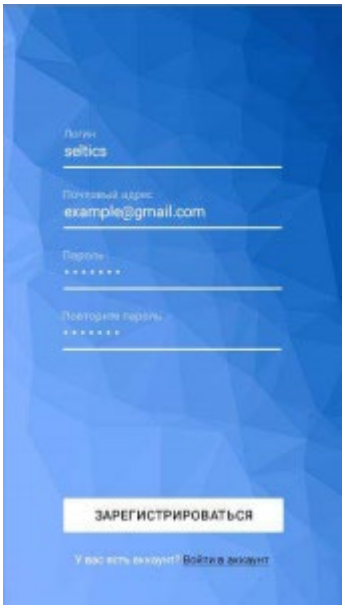
23. Яценков В. С. Java за неделю / В. С. Яценков. – Видавнича система: Ridero, 2018. – 312 с.

24. Языки программирования для веб-разработки // Блог вебпрограммиста [Электронный ресурс]. - Режим доступа: http://juice-health.ru/programming/web-development/505-programming-languages

25. Android Studio [Электронный ресурс] – Режим доступа: Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio

Додаток А

Макет екранів авторизації



Додаток Б

Макет екранів відображення розпорядку дня (зліва) і нотаток (справа)