Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Єрмакова Олександра Андрійовича*
(ПІБ)

академічної групи *122-18ск-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка мобільного застосунку для системи інформаційного забезпечення вищого навчального закладу*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Реута О.В.*			
розділів:				
спеціальний	*доц. Реута О.В.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18ск2* (група) *Єрмакова Олександра Андрійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка мобільного застосунку для системи інформаційного забезпечення вищого навчального закладу*

затверджена наказом ректора НТУ «ДП» від *07.06.2021* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ (підпис) *доц. Реута О.В.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Єрмаков О.А* (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2020 р.*

РЕФЕРАТ

Пояснювальна записка: 71 с., 28 рис., 2 табл., 3 дод., 23 джерела.

Об'єкт розробки: інформаційний додаток на Android Studio.

Мета кваліфікаційної роботи: розробка інформативного додатку, який дозволить отримувати інформації про спеціальності набагато швидше та зручніше.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне призначення полягає в створенні зручного способу для ознайомлення з важливою інформацією, зменшуючи час на її пошук.

Актуальність програмного продукту визначається тим, що студенти або абітурієнти поступаючи та ту чи іншу спеціальність, зазвичай мало що про неї знають. І ця програма дає змогу швидко та легко дізнатися необхідну інформацію

Список ключових слів: ПРОГРАМА, ANDROID, ГЕНЕРАЦІЯ, ANDROID STUDIO, API, ІНФОРМАЦІЙНА СИСТЕМА, Gradle.

ABSTRACT

Explanatory note: 71 pages, 28 figures, 2 tables, 3 appendices, 23 sources.

Object of development: information application on Android Studio.

The purpose of the qualification work: development of an informative application that will allow to receive information about specialties much faster and more conveniently.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program, describes the program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical purpose is to create a convenient way to view important information, reducing the time to search for it.

The relevance of the software product is determined by the fact that students or entrants entering one or another specialty, usually know little about it. And this program allows you to quickly and easily find the information you need

Keywords: PROGRAM, ANDROID, GENERATION, ANDROID STUDIO, API, INFORMATION SYSTEM, Gradle.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – Прикладний програмний інтерфейс;

XML (eXtensible Markup Language) - розширювана мова розмітки;

SDK (software Development Kit) - набір із засобів розробки;

ЖЦ – життєвий цикл;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ПС – програмне середовище;

AVD – android virtual device;

SDK (software Development Kit) — набір із засобів розробки;

IDE (Integrated Drive) - Інтегроване середовище розробки;

APK (Android Package) - формат архівних файлів-додатків для «Android».

ЗМІСТ

ВСТУП

З розвитком технологій використання смартфонів у повсякденному житті стає дедалі поширенішим. В наш час майже у кожної дорослої людини є свій смартфон. Різні мобільні додатки дозволяють значно спростити наше життя та вирішувати певні проблеми. Вони мають застосування майже у всіх областях нашого життя. Навчання та освіта є невід'ємною складовою нашого життя. Тому поява додатків, які дозволяють отримувати нові знання безпосередньо зі смартфону було лише питанням часу.

Сучасні інструменти розроблення мобільних додатків дозволяють розробникам реалізувати велику кількість ідей для різних цілей. В тому числі і використання смартфона в ролі засобу для здобуття нових знань. Ці інструменти дозволяють відтворювати інформацію у різних формах: текст, відео, звук, тощо.

Мобільний додаток - це прикладне програмне забезпечення, призначене для роботи на телефонах, планшетних комп'ютерах та інших мобільних пристроях». Додаток має сенс, якщо метою є взаємодіяти з користувачами в інтерактивному режимі або надати програму, яка вимагає роботи більше схожої на комп'ютерну програму, ніж веб-сайт. Додатки доступні на платформах розповсюдження в конкретних магазинах програм. Є як безкоштовні, так і платні додатки. Є декілька додатків, які спочатку доступні безкоштовно, але пізніше для отримання преміальних переваг потрібна мінімальна плата.

Тому метою кваліфікаційної роботи є створеня програмного додатку, що дозволить підвищити обізнаність абітурієнтів, через зручне і швидке ознайомче інформування щодо напрямків навчання і основної інформації про предмети і кафедри на яких вони викладаються, що відповідає вимогам до кваліфікаціних робіт за спеціальністю «Комп'ютерні науки».

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Останнім часом розвиток смартфонів набуває все більших обертів. С кожним днем їх операційні системи стають досконалішими, мають все більше функцій та розширюють коло можливостей для реалізації додатків з використання найновіших інструментів розробника. Перші мобільні додатки почали з’являтися майже одночасно з появою операційних систем для мобільних пристроїв. Можливості та функціональність перших додатків були на досить низькому рівні. З кожною новою версією операційної системи кількість доступних інструментів для розробника зростала, Це дало змогу захоплювати все більше областей для застосування створених продуктів, а отже й інструментарій цих додатків ставав дедалі ширшим. Це сприяло створенню додатків для використання у різних областях нашого життя.

На сьогоднішній день існують безліч додатків, які об’єднуються в певні категорії, наприклад: відео плеєри, редактори, навігатори, онлайн магазини, додатки для соціальних мереж. Вони використовуються повсюди.

Говорячи про «розробку додатків», ми зазвичай говоримо про розробку для мобільних пристроїв - включаючи смартфони та планшети.

Більш конкретно, ми зупинимось на мобільних додатках для однієї з найбільших та найважливіших мобільних операційних систем Android.

Станом на 2020 рік Android контролює близько 74,5% ринку мобільних пристроїв у всьому світі. Кількість користувачів Android зросла з 1,8 мільярда пристроїв у вересні 2015 року до понад 2,8 мільярда сьогодні. Оскільки все більше і більше світового населення виходить в Інтернет протягом наступного десятиліття, Android буде продовжувати зростати.

Насправді Android - це операційна система з відкритим кодом, що означає, що створення додатків для Android більш легша ніж для інших систем.

Також на створення та розробку додатків вплинув розвиток обчислювальних технологій, за допомогою яких у двадцять першому столітті комп'ютери використовуються майже у всіх сферах життя суспільства. Загалом, обчислювальні технології застосовувалися майже у всіх ситуаціях, які підпадають під одну з двох категорій. Перша категорія охоплює програми, що вимагають організації, зберігання і пошуку великих обсягів інформації, наприклад бібліотечних каталогів або банківських записів. Друга категорія включає додатки, що вимагають координації складних процесів, таких як управління машинами, задіяними у виробництві автомобілів або друку книг і газет.

Комп'ютери не тільки змінили те, як робочі місця структурують свої завдання і співробітників, але і кардинально змінили саму роботу. Автоматизоване виробництво вперше було впроваджено в 1950-х роках на верстатах з числовим програмним управлінням. Ці та інші форми комп'ютерної автоматизації пов'язані з втратою робочих місць і певних навичок, а також з необхідністю оволодіння новими навичками. З середини двадцятого століття пристрої з комп'ютерним управлінням поступово усунули певні види робіт і необхідність для людей виконувати певні навички. Як наслідок, працівникам доводилося здобувати нові навички, щоб продовжувати працювати в середовищі, яке все більше залежить від комп'ютерів.

А поява електронної пошти, всесвітньої павутини і інших інтернет-технологій, можливо, найбільш вплинула на соціальну аспект суспільства. Тепер люди можуть спілкуватися з іншими людьми у віддалених місцях легко, доступно і часто анонімно. Вони можуть шукати, обмінюватися і передавати більше інформації і швидше, ніж будь-коли раніше. Люди, розподілені по віддаленим місцях, можуть організовуватися в «віртуальні спільноти» на основі спільних інтересів, незалежно від їх географічного положення.

Також у сучасному світі все більшого поширення набувають пристрої на базі мобільних платформ. Це зумовлено потребою сучасної людини бути завжди на зв'язку. Але якщо десять років тому мобільний телефон був всього

лише засобом зв'язку - пересувний версією стаціонарного апарату, то п'ять років тому він вже міг виконувати функції записної книжки, допомагав обмінюватися невеликими обсягами інформації і міг використовуватися як засіб для розваг. В наш час у поняття мобільний телефон вкладається надзвичайно широкий набір функцій. За допомогою сучасного мобільного телефону, який належить до середньої цінової категорії, можна робити високоякісні фотографії і відео, отримувати, зберігати, відтворювати і передавати значні обсяги даних, користуватися Інтернетом та грати в ігри, про якість та деталізації яких ще десять років тому не могли мріяти і власники настільних комп'ютерів.

І оскільки розвиваються технології, то разом з ними зявився ряд мобільних платформ, основні з яких наведено нижче:

_Windows Phone (Windows Mobile) - ця операційна система містить в собі безліч можливостей, серед яких слід вказати: роботу з повідомленнями (SMS, електронна пошта), медіаплеєр, Internet Explorer, календар, нотатки та адресна книга. Характерною рисою даної платформи є відкрита архітектура, що дозволяє встановлювати додаткові програми і виконувати надбудови для стандартних програм. Дизайн ОС схожий на Windows, але не має з нею нічого спільного [1];

_Simbian. На боці Symbian дружність до непідготовленого технічно користувача. Операційні системи Symbian можна розділити на дві основні підгрупи: Series 60 (як і додаткові варіанти - S80 і S90), UIQ з підтримкою сенсорного екрану і застосовується компанією SonyEricsson [3];

_iPhoneOS (iOS) - iPhone OS заснована на спеціальній версії Mac OS X, при цьому інтерфейс адаптований для роботи з пальцями. Особливістю апарата є великий сенсорний екран, який займає майже всю фронтальну панель і всього одна клавіша, все управління здійснюється дотиками і переміщенням пальців по екрану, все це робить смартфон як живим [4].

Створення нових інформаційних технологій має велике значення для розвитку суспільства. Вони активно перетворюють інші технології

матеріального і нематеріального виробництва, в кінцевому підсумку формуючи новий стиль роботи, спосіб життя в цілому. Персональний комп'ютер, мобільні пристрої та мережа Інтернет дали змогу створювати загальнодоступну, надзвичайно інформаційно містку, та, порівняно з іншими інформаційно-технологічними системами, дешеву й швидку інформаційну інфраструктуру, їх доступність та надійність сприяли входженню у всі сфери суспільства нових інформаційних технологій.

Раніше мобільні додатки використовувалися для швидкої перевірки електронної пошти, але їх високий попит призвів до розширення їхніх призначень і в інших областях, таких як ігри для мобільних телефонів, GPS, спілкування, перегляд відео та користування інтернетом.

_Android - за широтою можливостей платформа Android не поступається операційним системам навіть настільних ПК. Вона являє собою багаторівневе середовище на основі ядра Linux і володіє багатими функціональними можливостями. Основу користувальницького інтерфейсу складають:

- вікна;

- подання;

- віджети для відображення загальних елементів, таких як редаговані поля, списки і розгортаючі списки. Android володіє широким спектром можливостей підключення таким, як Wi-Fi, Bluetooth і протоколи передачі даних через мережу [2]. У стек програмного забезпечення Android входить і підтримка сервісів, заснованих на визначенні місця розташування (наприклад, GPS), і акселерометрів, проте слід зауважити, що не всі пристрої на цій платформі оснащені необхідним обладнанням;

Android — операційна система та платформа для мобільних телефонів і планшетних комп'ютерів, створена компанією Google на базі ядра Linux та підтримується альянсом Open Handset Alliance. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення та застосування спираються на цю реалізацію Java[5].

Перша версія Android була випущена 23 вересня 2008 року і носила назву

1.0 Astroboy, а наступна — 1.1 Bender. Від назв на честь відомих роботів згодом довелося відмовитися через розбіжності з правовласниками. З 2008 року Android пережив численні оновлення, які поступово покращували операційну систему, додаючи нові функції, та виправляючи помилки у попередніх випусках.

1.2. Призначення розробки та галузь застосування

Розробка мобільного додатку ведеться для системи інформаційного забезпечення вищого навчального закладу. Так як дуже часто студентам та абітурієнтам складно знайти потрібну їм інформацію і можуть статись якісь труднощі в процесі її впізнавання, що може відпугнути потенційних студентів, було вирішено створити засіб, який ознайомить студента або абітурієнта з основною інформацією, яку потрібно знати максимально швидко та легко.

Мета: створеня програми, що дозволить підвищити процент лояльних абітурієнтів, через зручне і швидке ознайомче інформування щодо напрямків навчання і основної інформації про кафедри.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;
- завдання на кваліфікаційну роботу на тему «Розробка мобільного застосунку для системи інформаційного забезпечення вищого навчального

закладу».

1.3.1 Засоби розробки мобільного додатку під ОС Android

Програми для ОС Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik.

Google пропонує для вільного завантаження інструментарій для розробки (Software Development Kit), який призначений для x86-машин під операційними системами Linux, Mac OS X (10.4.8 або вище), Windows XP, Windows Vista та Windows 7. Для розробки потрібен Java Development Kit 5 або новіший.

Розробку застосунків для Android можна вести мовою Java (не нижче Java 1.5). Офіційним середовищем розробки є Android Studio, представлене компанією Google в 2013. Крім цього існує плагін для Eclipse – «Android Development Tools» (ADT), призначений для Eclipse версій 3.3-3.7. Для IntelliJ IDEA також існує плагін, який полегшує розробку Androidзастосунків. Для середовища розробки NetBeans розроблено плагін, який починаючи з версії Netbeans 7.0 перестав бути експериментальним. Також існує Motodev Studio for Android, що являє собою комплексне середовище розробки, засноване на базі Eclipse і дозволяє працювати безпосередньо з Google SDK[6].

У 2009 році до ADT був опублікований Android Native Development Kit (NDK), пакет інструментаріїв і та бібліотек що дозволяє вести розробку застосунків мовою C/C++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості.

Доступнимиі є наступні бібліотеки:

- Bionic (Бібліотека стандартних функцій, несумісна з libc);

- libc (стандартна системна бібліотека мови Сі);

- мультимедійні бібліотеки (на базі PacketVideo OpenCORE; підтримують такі формати, як MPEG-4, H.264, MP3, AAC, AMR, JPEG та PNG);

- SGL (рушій двовимірної графіки);

- OpenGL ES 1.0 (рушій тривимірної графіки);

- Surface Manager (забезпечує для застосунків доступ до 2D/3D);

- WebKit (готовий рушій для Web-браузера; обробляє HTML, JavaScript);

- FreeType (рушій обробки шрифтів);

- SQLite (проста система керування базами даних, доступна для всіх застосувань);

- SSL (протокол, що забезпечує безпечну передачу даних по мережі).

В порівнянні із звичайними застосунками Linux, застосунки Android підкоряються додатковим правилам [7]:

Content Providers — обмін даними між застосунками;

Resource Manager — доступ до таких ресурсів, як файли XML, PNG, JPEG;

Notification Manager — доступ до рядка стану;

Activity Manager — управління активними застосунками.

Для Android був розроблений формат інсталяційних пакетів .apk.

1.4. Постановка завдання

Метою створення мобільного додатку є розповсюдження корисної інформації про кафедри, їх керівників, та спеціальності університету в зручному та привабливому вигляді. Додаток повинен відображати сторінку із потрібною інформацією для подальшого ознайомлення студентів та абітурієнтів з кафедрами, їх керівниками та спеціальностями на які вони можуть поступити або вже вчаться.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для спроможності скористатись додатком треба мати смартфон із версією Android не нижче 4-ої. Пам’ять: 16 ГБ, ОЗУ 2 ГБ. Процесор: 4 ядра.

1.5.2. Вимоги до інформаційної безпеки

Безпека даних включає їх цілісність і захист. Цілісність даних - стійкість даних, що зберігаються до руйнування і знищення, пов'язаних з несправностями технічних засобів, системними помилками і помилковими діями користувачів.

Вона передбачає:

- відсутність неточно введених даних або двох однакових записів про одне й те ж факт;
- захист від помилок при оновленні бази даних;
- неможливість видалення (або каскадне видалення) пов'язаних даних різних таблиць;
- збереження даних при збої техніки (відновлення даних).

Цілісність даних полягає у логічній і фізичній цілісності даних.

Логічна цілісність - виражається у вигляді обмежень або правил збереження даних, несуперечність яких не повинна порушуватися в базі. Можна вказати межі значень при описі структури бази даних або таблиць бази.

Фізична цілісність виражається в захисті даних від фізичного руйнування в результаті збоїв і відмов устаткування. Для цього використовують контрольні копії даних, тобто запам'ятовування станів бази даних у контрольних точках, а також ведення системного журналу, в якому відображаються усі маніпуляції над базою даних.

Оскільки інформація не є секретною і відкрита для розповсюдження, то ніякі заходи для забезпечення безпеки не потрібні.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для функціонування додатку потрібен телефон та комп’ютер

Нефункціональні вимоги компютера:

а) ОС сімейства Windows;

b) Щонайменш одноядерний процесор;

c) Мінімальна кількість ОЗУ 2ГБ;

d) Не менше 5 ГБ свободного місця на жорсткому диску;

e) Відсутність збоїв у роботі системи;

g) Стійкість до збоїв та можливість продовжити роботу з системою у випадку перезавантаження сторінки;

Нефункціональні вимоги телефону:

1) ОС сімейства Android;
2) Щонайменш одноядерний процесор;
3) Мінімальна кількість ОЗУ 2ГБ;
4) Не менше 5 ГБ свободного місця диску;
5) Мінімальна версія android 4.0.0

1.5.4. Вимоги до інформаційної та програмної сумісності

Розроблена програма повинна взаємодіяти з операційною системою Android і написаною на мові програмування Java.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Інформаційна система мобільного додатку призначена для інформування студента/абітурієнта. При цьому реалізована можливість вибору тієї інформації, яка цікавить користувача.

Вивід інформації реалізований у вигляді сторінки, які представлені користувачеві у вигляді структурованого документа в затвердженому форматі і доступні для перегляду.

2.2. Опис застосованих математичних методів

Ця інформаційна система не вимагає розробки математичної моделі і застосування математичних методів.

2.3. Опис використаних технологій та мов програмування

Для реалізації заданої роботи було обрано віртуальне середовище програмування Android Studio і мову програмування Java для ОС Android. Вибір був зроблений виходячи з наявних на підприємстві програмних засобів.

2.3.1 Обґрунтування функцій програмного продукту

2.3.1.1 Формування варіантів функцій

Головна функція F0 – розробка програмного продукту, який дає найменшу похибку при порівнянні реальних значень цільової змінної та результуючого значення побудованого багаточлену у відповідній точці. Виходячи з конкретної мети, можна виділити наступні основні функції

програмного продукту:

F1 – вибір мови програмування;

F2 – кількість одночасно підключених пристроїв;

F3 – представлення вихідних даних;

F4 – функціонал;

F5 – середовище розробки;

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування C#;

б) мова програмування Kotlin;

в) мова програмування Java.

Функція F2:

а) один

б) декілька

Функція F3:

а) вивід інформації у файл;

б) вивід інформації на екран.

Функція F4:

а) мінімальний функціонал, тільки розрахункова частина;

б) ілюстративний функціонал.

Функція F5:

а) Android Studio

б) Eclipse

2.3.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведено у морфологічній карті системи (рис. 2.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 2.1)

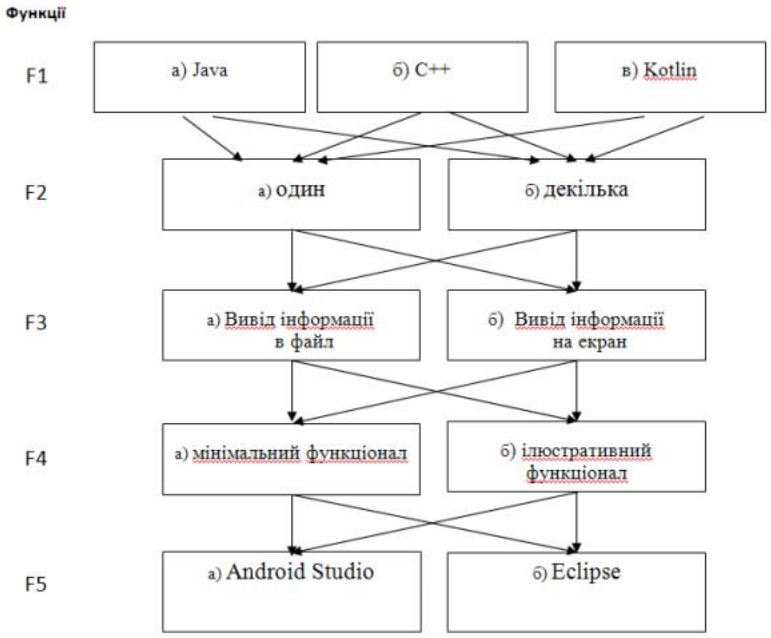

Рис. 2.1. Морфологічна карта

Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів програмного продукту.

Таблиця 2.1.

Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	а	Висока швидкість написання коду, кроссплатформлений	Посередний JVM
	б	Код швидко виконується, Працює прямо з ядром	Займає більше часу при написанні коду
	в	Займає менше часу при написанні коду	Не росплатформений, достатньо нова мова
F2	а	Займає менше часу при написанні коду	Конфлікти при підключенні більше 1го
	б	Можна підключи ти скільки завгодно пристроїв	Займає більше часу

Продовження таблиці 2.1

F3	а	Можливість послідуючого редагування та друку в будь-якому редакторі	Неспівпадання кодування українських символів в DOS та Windows
	б	Отримання результатіву реальному часі	Ніде не зберігається
F4	а	Простота реалізації	Складність у сприйнятті інформації
	б	Простота у сприйнятті інформації	Складність реалізації
F5	а	Професійне середовище, економія часу	Складність опанування
	б	Простота користування	Менше можливостей

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктам задачам.

Функція F1: оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час написання та виконання програмного коду є найважливішими факторами, отже варіант б) і в) потрібно відкинути.

Функція F2: оскільки програма повинна бути призначена для широкого взаємодії з декількома пристроями, то потрібно відкинути варіант а).

Функція F4: Оскільки ми робимо якісний продукт, то варіант а) потрідно відкинути

Функція F5: Оскільки ми добре розбираємость у середовищі Android Studio відкидуємо варіант б)

Таким чином будемо використовувати наступні варіанти реалізації програмного продукту:

1. F1а – F2б – F3а – F4б – F5а;

2. F1а – F2б – F3б – F4б – F5а;

2.3.2 Платформа Android

Платформа “Android” є продуктом групи “Open Handset Alliance” , яка ставить перед собою ціль створити найбільш досконалу мобільну систему. З точки зору розробки програмного забезпечення “Android” знаходиться в самому центрі розробки відкритого програмного забезпечення. Перши й пристрій під управлінням цією системою був випущений у 2008 році і єдиним інструментом розробки програм для нього були послідовні випуски пакета розробки “SDK”.

За шириною можливостей платформа “Android” не поступається місцем операційним системах персональних комп’ютерів. Це багаторівнева система на основі ядра “Linux” з широкими функціональними можливостями. В підсистему користувацького інтерфейсу входять вікна, представлення та віджети. “Android” володіє широким спектром можливостей підключення, таких як “Wi-Fi”, “Bluetooth” та протоколи передачі даних через стільникову мережу. В стек програмного забезпечення “Android” входить і підтримка сервісів, заснованих на визначенні місцеположення та акселерометрів , хоча не всі пристрої на цій платформі устатковані необхідним обладнанням. Також наявна підтримка відеокамери. Історично двома областями, це мобільні системи відставали від настільних були графіка та способи збереження даних. “Android” вирішує проблему графіки завдяки вбудованій підтримці графіки, включаючи бібліотеку “OpenGl”. Задача збереження даних спрощується завдяки наявності в платформі “Android” популярної бази даних з відкритим кодом “Sqlite”.

Операційна система “Android” працює поверх ядра “Linux”. Додатки пишуться на мові програмування “Java” і виконуються в віртуальній машині. Важливо уточнити, що віртуальна машина це не “JVM”, як можна було б очікувати, а відкрита технологія “Dalvik Virtual Machine”. Кожний додаток “Android” запускається всередині екземпляра “Dalvik VM” , який в свою чергу

укладений в рамках контрольованого ядром “Linux” процесу.

Компанія Google вклала багато ресурсів в скорочення платформи та зростання її ефективності. Оптимізація в першу чергу направлена на зменшення розміру, збільшення швидкості, економію заряду акамулятора та зниження витрат пам’яті. Вони провели роботу на декількох рівнях стеку. Віртуальна машина Dalvik була ретельно розроблена з метою задоволення цих вимог до продуктивності та має декілька незвичайних характеристик. На відміну від звичайного виконуваного середовища Java, Dalvik заснований на регістрі, а не на стеку і не підтримує JIT компіляцію. Кожен окремий додаток виконується в окремому екземплярі віртуальної машини Dalvik і пам’ять розподіляється між цими екземплярами для зменшення витрат. Для того щоб скоротити час запуску додатка, Dalvik має компонент під назвою Zygote, який попередньо ініціалізує екзмепляри віртуальних машин і розгалуджує їх, якщо в цьому виникає необхідність.

Радикально інший підхід Android до розробки мобільних додатків пропонує деякі унікальні переваги, але він також створює багато проблем для сторонніх розробників. Найбільшою перевагою в такому підході є те, що він забезпечує високий рівень однорідності. Переважна більшість додатків Android зможуть без проблем працювати практично на будь-якому пристрої на базі Android, не потребуючи при цьому подальших змін.

До складу “Android” входить комплект базових додатків: клієнти електронної пошти, календар, різноманітні карти, браузер, програма для керування контактами тощо.

“Android” дозволяє використовувати всю потужність “Android” , що використовуються в додатках ядра. Архітектура побудована таким чином, що будь який додаток може використовувати уже реалізовані можливості іншого додатка, при умові, що останнє відкриє доступ до використання своєї функціональності. Таким чином архітектура реалізує принцип багаторазового використання компонентів і додатків. Нижче, на (рис. 2.2) наведено архітектуру системи.

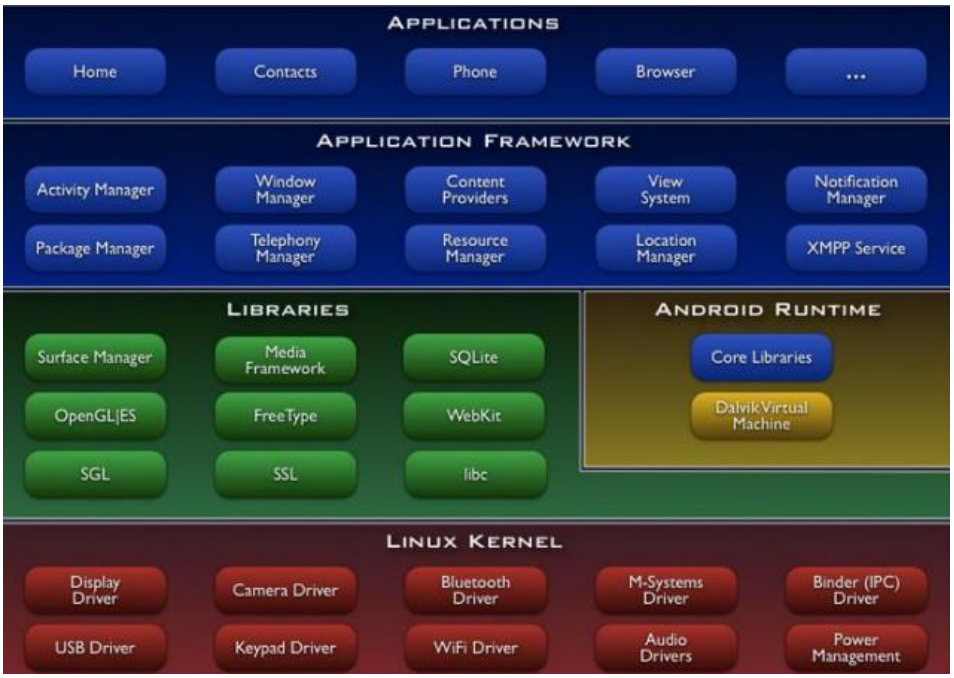

Рис. 2.2. Схема архітектури системи

Платформа Android надає повний та добре організований асортимент високорівневих API інтерфейсів для створення додатків та використовування основних функцій платформи. Інтерфейси API запезпечують надзвичайно високий рівень абстракції, що робить їх відносно інтуїтивно зрозумілими і простими у використанні в процесі розробки.

Сторонні додатки можуть реплікуватися або взаємодіяти практично з усіма основними компонентами платформи. Наприклад, Android надає методи для отримання інформації з списку контактів користувача та для розширення системи контактів новими полями даних. Також легко зробити новий інтрефейс дзвінка або реалізувати користувацьку поведінку для системних подій, таких як вхідні повідомлення. Зокрема, API дійсно дозволяють створювати додатки, які повністю інтегруються з рештою платформи. На (рис 2.3) зображено приклад використання API.

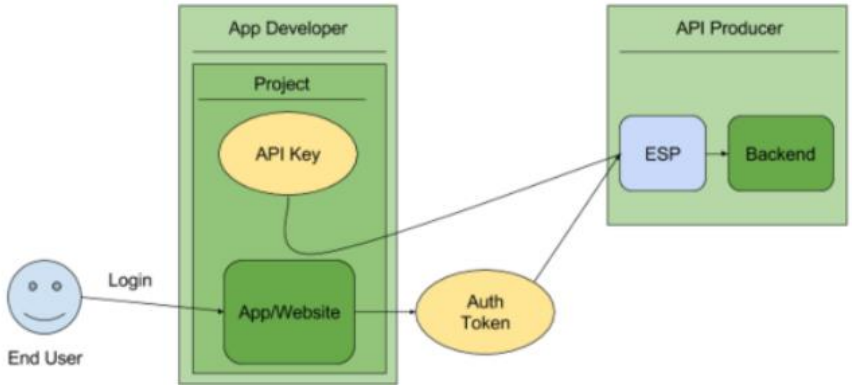

Рис. 2.3 .Приклад використання API

Набір інструментів для віджетів Android надає велику кількість дуже корисних компонентів відразу. Окремі віджети розроблені спеціально для зручної взаємодії з пальцями, з вже вбудованими функціями, такими як кінетична прокрутка. Розробники використовують мову опису користувацького інтерфейсу на основі XML для визначення макету та атрибутів віджетів, при цьому описи XML завантажуються в програму через систему ресурсів Android.

На окремі віджети, описані в в макеті XML, можна посилатися по ідентифікатору в програмі. Також є можливість створення віджетів програмно і маніпулювати ними користувацьким інтерфейсом в процесі виконання програми. Найбільш правильним способом створення макетів є написання описів XML вручну. Пакет Android SDK надає вбудований інструмент візуального макета, проте він не підтримує всі віджети і не завжди працює злагоджено.

Додатки Android складаються з провайдерів , служб, приймачів та активностей. Всі ці компоненти мають окрему задачу в стеку додатків Android. Спосіб, яким вони можуть взаємодіями один з одним це те, що наповнює Android його модульністю.

Важливою особливістю, яка відрізняє Android від інших платформ є те, що Android офіційно підтримує фонові процеси в сторонніх додатках. Вони реалізовані за допомогою сервісного компоненту Android. Сервіси – це операції без заголовку, які виконуються в фоновому режимі протягом тривалого часу.

Система повідомлень Android приблизно схожа з аналогічною системою

сигналів в Linux. Розробники вбудовують широкомовні приймачі, які можуть визначати, коли з’являються визначені системні повідомлення чи події, і виконувати певні дії у відповідь. Багато базових системних подій можна відслідковувати за допомогою системи мовлення, тому її можна використовувати для відстежування таких речей, як зміна стану акумуляторної батареї, отримання вхідних SMS повідомлень, натискання апаратних кнопок, зміна з’єднання, встановлення сховища та затемнення екрану.

Система Android Intents є ключем до розуміння того, як всі ці частини використовуються разом. Об’єкти Intents, які містять ідентифікатор дії та URI даних, використовуються для виклику дій, служб та отримувачів, Ідентифікатор дії вказує бажану поведінку, а необов’язковий URI даних надає місце розташування цих даних, над якими має виконуватися дія. На (рис 2.4) зображено загальну схему компонентів Android додатку.

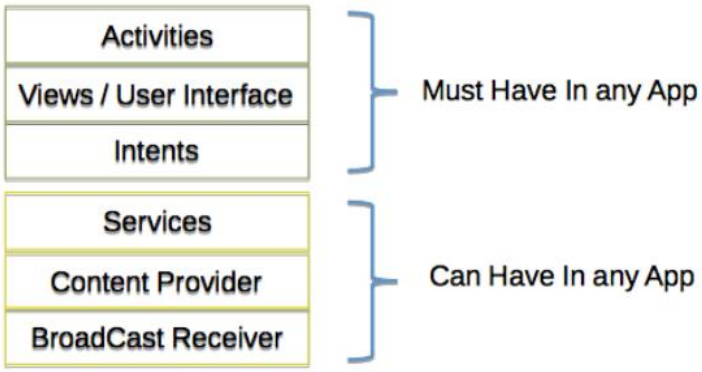

Рис. 2.4. Приклад використання API

Одна із головних переваг системи Android – її відкритість. Операційна система Android побудована на основі відкритого висхідного коду і розповсюджується на вільній основі. Це дозволяє розробникам отримати доступ до висхідного коду і зрозуміти яким чином реалізовані властивості і функції додатків. Кожен користувач може прийняти участь в удосконаленні операційної системи.

2.3.3 Середовище розробки Android Studio

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

• Гнучка Gradle-система збірки

• Побудувати варіанти і кілька АПК покоління файлу

• Шаблони коду, щоб допомогти вам побудувати загальні риси додатку

• Багатий редактор макетів з підтримкою перетягування і падіння редагування теми

• інструменти, щоб фіксувати продуктивність, зручність використання, сумісність версії, і інші проблеми

• Proguard

• Вбудована підтримка для Google Cloud Platform, що дозволяє легко інтегрувати Google Cloud повідомленнями.

Android SDK - включає в себе різноманітні бібліотеки, документацію та інструменти, які допомагають розробляти мобільні додатки для платформи Android. [8,9]

• API Android SDK-API бібліотеки Android, що надаються для розробки додатків.

• Документація SDK-включає велику довідкову інформацію, що деталізує, що включено в кожен пакет і клас і як це використовувати при розробці додатків.

• AVD (Android Virtual Device) -інтерактівний емулятор мобільного пристрою Android. Використовуючи емулятор, можна запускати і тестувати програми без використання реального Android пристрою.

• Development Tools - SDK включає кілька інструментальних засобів для розробки, які дозволяють компілювати і налагоджувати створювані додатки.

• Sample Code - Android SDK надає типові додатки, які демонструють деякі з можливостей Android, і прості програми, які показують, як

використовувати індивідуальні особливості API у вашому коді.

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

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

Оновлення API-структури розроблені так, щоб новий API залишався сумісним з більш ранніми версіями API. Таким чином, більшість змін в API є сукупною і вводить нові функціональні можливості або виправляє попередні. Оскільки частина API постійно оновлюється, застарілі API не рекомендуються до використання, але не видаляються з міркувань сумісності з наявними додатками.

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

У таблиці 2.2 наведено відповідність рівня API і версії платформи Android.

Таб. 2.2

Відповідність версії платформи та рівня API

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

Крім емулятора, SDK також включає безліч інших інструментальних засобів для налагодження та установки створюваних додатків [13,14,15] Якщо При розробці програми для Android за допомогою IDE Android Studio, багато інструментів командного рядка, що входять до складу SDK, вже використовуються при складанні і компіляції проекту. Однак крім них SDK містить ще ряд корисних інструментів для розробки і налагодження додатків:

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

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

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

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

• Draw 9-patch - графічний редактор, який дозволяє легко створювати NinePatch графіку для графічного інтерфейсу розроблюваних додатків;

• sqlite3 - інструмент для доступу до файлів даних SQLite, створених і використовуваних додатками для Android;

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

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

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

2.3.4 Розробка архітектури проекту

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

Розробка здійснюється у популярній IDE від Google – Android Studio.

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

Потім XML файли наповнюються потрібною інформацією динамічно за допомогою Java.

Java - це об'єктно-орієнтована мова програмування. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцезією GNU (General Public License[17]).

Java - дуже гнучка мова з відкритим вихідним кодом. Для розробки програми існують архітектурні підходи до проекту, які вирішують проблему

розробки стандартного програмного забезпечення. Шаблони - це абстрактні схеми, вони не є кодом, але вони допомагають розділити логіку програми на певні модулі. При використанні шаблону проекту, цей шаблон адаптується у відповідності зі своїми певними потребами (рис.2.5).

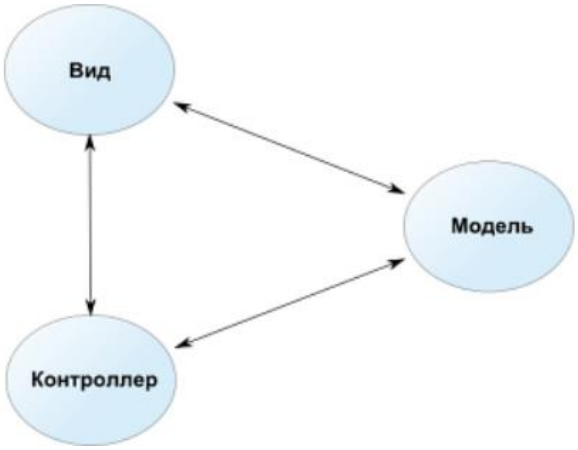

Рис. 2.5. Структура шаблону Модель - Вид – Контролер

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

Вид (View) являє собою компонент системи для відображення стану моделі в зрозумілому людині поданні. Це можуть бути діалоги, форми та інші візуальні (наприклад, синтезатор мови) засоби взаємодії людини з системою. Вид не змінює дані безпосередньо (режим тільки читання), дані змінюються за допомогою контролера.

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

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

Модель не залежить ні від виду, ні від контролера, що дозволяє одночасно будувати різні інтерфейси користувача для взаємодії з однією і тією ж моделлю даних. Наприклад, можна зробити як Java SWT або SWING десктопних програм, так і WEB додаток для взаємодії одними і тими ж даними[18].

2.3.5 Система автоматизованої збірки Gradle

Процес створення програмного продукту для розробників та тестувальників програмного забезпечення без засобів автоматизації є дуже повторюваним, нудним та значно підвищує ризик помилок. Кожен крок в процесі розробки програмного продукту, починаючи з джерела компілювання коду до завершальної збірки програмного забезпечення для релізу та перевірки на виробництві потрібно робити вручну. Автоматизація проекту допомагає знизити кількість ручного втручання в проект, робить процес розробки більш ефективним та мінімізує можливість виходу програмного забезпечення з ладу.

Автоматизація проекту має кілька очевидних переваг для процесу розробки програмного продукту. Перша із них це запобігання ручному втручанню в процес збірки. Необхідність вручну виконувати дії для створення та постачання програмного забезпечення вимагає багато часу і підвищує ризик виникнення помилок. Розробник та системний адміністратор мають багато ізних задач, які вони могли б вирішувати в той час коли займаються компіляцією вручну. Буль-який крок в процесі розробки який може бути автоматизований має бути автоматизований. Наступною перевагою є створення повторюваних збірок. Звичайна побудова програмного проекту відповідає попередньо визначеним і впорядкованим крокам. Наприклад, розробнику необхідно спочатку скомпілювати вихідний код, потім запустити тести і нарешті зібрати проект з отриманням певного результату. Доведеться виконувати ті самі кроки знову і знову кожного дня. Цей процес має бути таким же легким як натискання кнопки. Результат цього процесу має повторюватися

для всіх, хто запускає збірку проекту. Також до переваг можна віднести портативні збірки. Можливість запуску збірки з IDE дуже обмежена. По-перше, розробнику програмного продукту необхідно встановити на своєму пристрої певний продукт. По-друге, IDE може бути доступна лише для певної операційної системи. Автоматизована збірка не вимагає певного середовища виконання, незалежно від того, чи це операційна система або середовище розробки. Оптимально, автоматизовані завдання повинні виконуватися 48 з командного рядка, який дозволяє запускати процес збірки з будь-якого пристрою та в будь-який час.

Автоматизація проекту в свою чергу ділиться на декілька основних типів. Перший із них це збірка на вимогу. Типовий випадок використання автоматизації за вимогою, це випадок коли користувач запускає побудову на певному пристрої. Зазвичай система контролю версій VSC керує версифікацією збірок та файлами висхідного коду. У більшості випадків користувач виконує скрипт у командному рядку, який виконує завдання у попередньо визначеному порядку. Наприклад, компіляція вихідного коду, копіювання файлу з одного каталогу до іншого або збірка результату. Зазвичай цей тип автоматизації виконується по кілька разів на день. Наступним типом автоматизації є запущені збірки. Якщо розробник займається гнучкою розробкою програмного забезпечення, його цікавить швидке отримання зворотного зв’язку про стан проекту. Також важлива інформація про те, чи може вихідний код бути зібраним без будь-яких помилок та інформація про наявність в проекті потенційного програмного дефекту помічений в помилці блоку чи тесті інтеграції. Цей тип автоматизації зазвичай спрацьовує в результаті перевірки коду системою контролю версій. Ще одним типом автоматизації є заплановані збірки. Задумка полягає в плануванні автоматизації як планування завдань на основі часу. Вона виконується в певні проміжки часу або в певний конкретний час. Запланована автоматизація зазвичай виконується на виділеному сервері. Такий вид автоматизації особливо корисний для створення звітів або документації для проекту. Практика реалізації запланованих та запущених

збірок, зазвичай визначається як безперервна інтеграція CI.

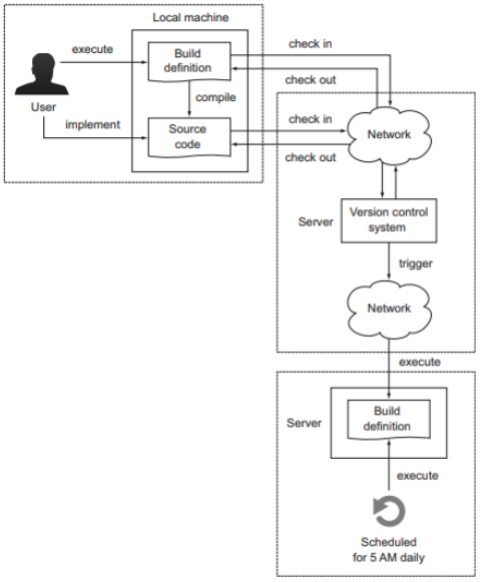

Рис .2.6. Приклад запланованої збірки

Для розробника автоматизація проекту є частиною повсякденної роботи. Сценарії побудови Gradle є декларативними, читабельними і чітко виражають свій намір. Написання коду в Groovy замість XML з елементами філософії Build-byConvention, дозволяє значно скоротити розмір сценарію збирання і є набагато більше читабельним , що зображено на (рис 2.7).

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Gradle

```
apply plugin: 'java'
group = 'com.mycompany.app'
archivesBaseName = 'my-app'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testCompile 'junit:junit:4.11'
}
```

Рис. 2.7. Порівняння скриптів Maven та Gradle

Кожна побудова Gradle починається з сценарію. Стандартне найменування сценарію для скрипту Gradle є build.gradle. При виконанні базової команди в оболонці система збірки шукає файл з саме таким іменем. Якщо його не буде знайдено буде відображено повідомлення про помилку. Після створення цього файлу необхідно визначити одне або декілька завдань.

Підводячи підсумок, можна визначити, що Gradle – це автоматизована система збірки, яка походить від декларативного і виразного Groovy DSL. Вона поєднує в собі гнучкість і простоту розширення з ідеєю про конфігурацію і підтримку традиційного управління залежностями. Gradle стає першочерговим вибором для побудови багатьох проектів з відкритим кодом.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Вибір схеми моделі життєвого циклу

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

Прикладами таких моделями є: «Каскадна модель», модель «Спіраль» і однією з примітних таких моделей для пр. оектування додатку є модель Уолта Діснея [7], вона складається з трьох етапів:

-концептуальне проектування;

-логічне проектування;

-фізичне проектування.

Етапи слідують послідовно один за іншим (рис. 2.8), але в деяких випадках можливий перехід до наступної стадії без закінчення попередньої. Це може відбуватися, наприклад, коли є декілька розробників, кожен з яких працює зі своєю частиною додатку. У будь-якому випадку, після закінчення етапу фізичного проектування слід повернутися до початку і внести відповідні корективи[8].

Рис. 2.8. Етапи проектування

Концептуальне проектування. Часом буває складно оцінити ефективність додатку. Необхідно знати і розуміти критерії оцінки для того, щоб визначити хорошим чи поганим є розроблений ресурс. Є універсальний критерій, який досить точно характеризує ефективність додатку – це досягнення розробниками додатку поставлених перед ними цілей. У цьому випадку додаток перетворюється на якісний інструмент, який виконує покладені на нього функції. Концептуальне проектування служить для вказівки цілей, завдань додатку та визначення аудиторії, на яку він розрахований.

На цьому етапі проектування слід описати наступне:

-основні і другорядні цілі;

-дії, які необхідно вжити для досягнення поставлених цілей;

-аудиторію додатку;

-інтереси груп користувачів;

-розділи додатку

-критерії досягнення мети.

Після визначення поставлених цілей й інтересів користувачів, можна скласти список сервісів й розділів, які будуть розташовуватися на додаток[9].

Логічне проектування. Визначені розділи додатку, на попередньому етапі, поки не впорядковані і не структуровані, тому їх потрібно привести до зручного та зрозумілого виду[10].

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

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

-тип структури додатку (лінійна, ієрархічна, контекстна, інша);

-назви розділів;

-що буде містити в собі кожен розділ;

-організація та зв'язок розділів між собою;

-що буде розміщено на певних розділах додатку.

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

Фізичне проектування. Даний етап пов'язаний з пошуком проблем, а не їх рішень, пов'язаних, здебільшого, з технічною реалізацією додатку. На цьому етапі слід описати наступне:

-технології, які будуть застосовуватися в додатку;

-програмне забезпечення, що використовується;

-можливі проблеми та способи їх усунення;

-як буде оновлюватися інформація.

Після завершення цього етапу слід повернутися до концептуального проектування і перевірити, чи не потрібно внести зміни, у зв'язку з переосмисленням проекту на інших стадіях[11].

При виборі схеми моделі життєвого циклу (ЖЦ) для конкретної предметної області, вирішуються питання включення важливих для створюваного продукту видів робіт або не включення несуттєвих робіт. На сьогодні основою формування нової моделі ЖЦ для конкретної прикладної системи є стандарт ISO/IEC12207, що задає повний набір процесів (понад 40),

що охоплює всі можливі види робіт і завдань, пов'язаних з побудовою програмного середовища (ПС), починаючи з аналізу предметної області і закінчуючи виготовленням відповідного продукту. Цей стандарт містить основні та допоміжні процеси (рис. 2.9 та рис. 2.10).

Рис. 2.9. Схема основних процесів ЖЦ ПС

На (рис. 2.9) представлені процеси, пов'язані безпосередньо з розробкою ПС. До категорії основних процесів відносяться також "первинні" процеси, що визначають порядок підготовки договору на розробку ПС, моніторинг діяльності постачальників ПС замовнику.

Стандарт ISO/IEC12207 надає структуру процесів ЖЦ, але не зобов'язує використовувати всі процеси в моделі ЖЦ ПЗ або в конкретній методології розробки програмного забезпечення (ПЗ)[11].

Рис. 2.10. Схема допоміжних процесів ЖЦ ПЗ

При створенні додатку було використано стандарт ISO/IEC12207 [12]. На основі цього розроблено методологію додатку для роботи з BLE (рис. 2.11).

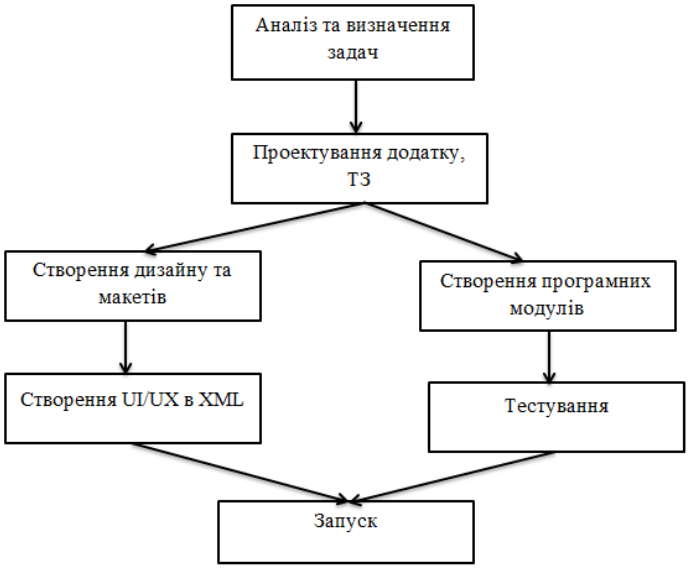

Рис. 2.11. Методологія створення додатку

Методологія розробки мобільного додатку включає наступні етапи:

-аналіз та визначення задач (планування) – визначення мети створення додатку, тематики і призначення майбутнього додатку, можливостей реалізації проекту;

-проектування та розробка технічного завдання – визначення структури додатку, матеріалів, вибір програмних засобів для його створення;

-веб-дизайн – створення графічних елементів макету додатку, стилів і елементів навігації;

-розробка програмного коду, модулів, бази даних і інших елементів додатку необхідних в проекті, вибір та інтегрування CMS;

-заповнення додатку матеріалами;

-тестування додатку;

-запуск [10].

2.4.2. Опис програмної реалізації

Система інформування студентів/абітурієнтів буде складатися з декількох вікон керування. Головним буде базове вікно вибору степеню акредитації. Потім три вікна з вибором галузі знань. Також буде вікно вибору спеціальності, яке наповнюється контентом в залежності від обраної спеціальності. Система матиме також декілька допоміжних вікон для взаємодії програмного продукту і користувача. Обрана структура буде простою та інтуїтивно зрозумілою для користувача за рахунок чіткої та зрозумілої організації модулів. На (рис. 2.12) наведена схема структури системи, на якій розташовані всі програмні модулі.

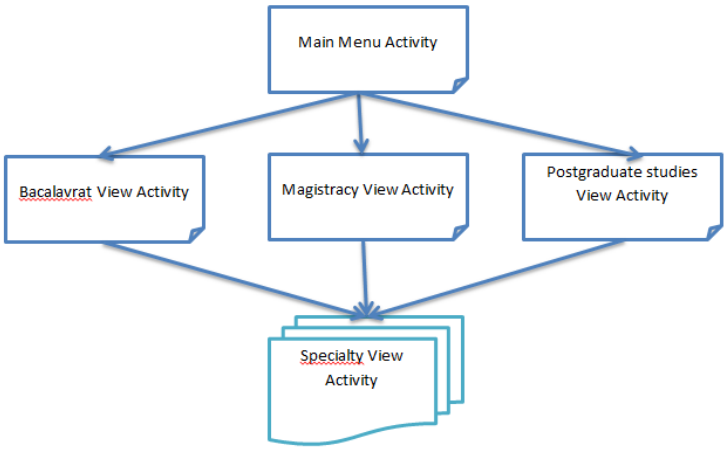

Рис. 2.12. Схема структури системи

2.4.3. Опис функціональності системи

Система дистанційного інформування містить у собі два актора, які взаємодіють із системою. Перший актор – це користувач який взаємодіє із системою за допомогою смартфона. Другий автор – це адміністратор додатку який наповнює систему контентом на надає необхідну технічну підтримку. На (рис 2.13) представлена діаграма прецедентів, яка описує функції та дії акторів у системі дистанційного навчання.

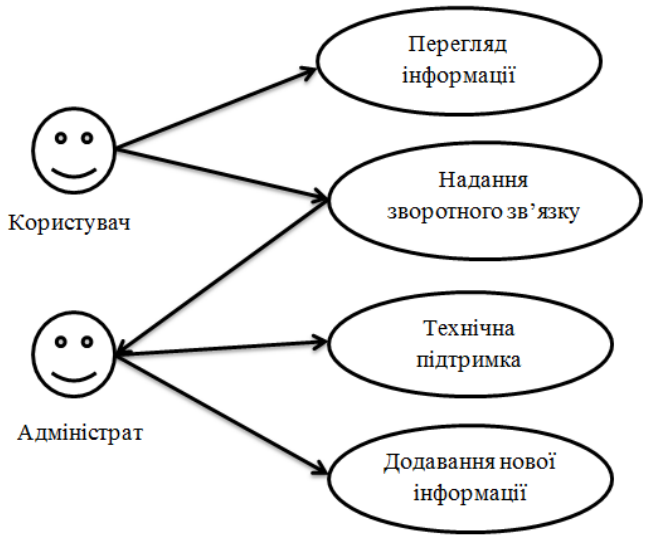

Рис. 2.13. Діаграма прецедентів системи

2.4.4. Розробка інтерфейсу користувача

Графічний інтерфейс користувача має відповідати наступним вимогам: ясність, відповідність, виразність, послідовність. Для забезпечення цих вимог був розроблений інтерфейс програмного продукту. Він включає в себе декілька модулів. Це головне меню, вікно з вибором галузі знань, вікно спеціальності. Коли користувач відриває додаток перед ним з'являється головне меню в якому він може обрати ступень акредитації. Після того як користувач обирає ступені акредитації, він переходить на вікно вибору галузі знань. Коли користувач обирає галузь знань, в залежності від того є в галузі знань декілька спеціальностей, або ж тільки одна. Він переходить на вікно вибору спеціальності (якщо їх декілька) і після обрання спеціальності, переходить на вікно спеціальності, в якому відображається вся інформація про спеціальність

2.4.4.1. Вікно головного меню

Дане вікно слугує для виведення на екран всіх степенем акредитації і надання користувачу можливості вибору одного із них. Воно складається з декількох елементів, а саме:

- Текстового екрану
- Кнопки Бакалаврат
- Кнопки Магістратура
- Кнопки Аспірантура

2.4.4.2. Вікно вибору спеціальності

Дане вікно слугує для виведення на екран всіх спеціальностей і надання користувачу можливості вибору однієї із них. Воно виводиться при умови, що в галузі знань декілька спеціальностей. І складається з декількох елементів, а саме:

- Текстового екрану
- Кнопок із спеціальностями
- Кнопки повернення на головне меню

2.4.4.3. Вікно спеціальності

Дане вікно призначене для виведення інформації про спеціальність. Воно складається з структури елементів курсу, а саме:

- Зображення
- Назви спеціальності
- Детальний опис спеціальності

2.4.5. Розробка додатку

Як завжди, першим кроком при створенні додатку для Android є налаштування середовища розробки. На щастя, це легко зробити за допомогою Android SDK. Нам просто потрібно було завантажити Android Studio (рис 2.14) і встановити її у своїй системі. Після завершення встановлення та ініціалізації у нас з’явились усі необхідні інструменти для запуску нашого проекту Android.

Наступним кроком є створення нового проекту Android за допомогою Android Studio. У діалоговому меню, що з’явилося, активували підтримку Java, оскільки саме цією мовою ми будемо користуватися. Ми використовували Kotlin та Java для нашого додатка, оскільки Kotlin вимагає набагато менше коду порівняно з Java, що призводить до швидшого написання коду, але не все можна реалізувати на ньому.

Що стосується рівня API, вам слід вибрати рівень API 26, оскільки наш додаток не вимагає розширених функцій, які існують в останньому SDK.

Остання частина процесу створення проекту є основною діяльністю програми. Отож, маючи це на увазі, ми назвали це HomeActivity та використаємо Empty Activity як шаблон для нашої початкової діяльності (рис 2.15).

Рис. 2.14. Установка Android Studio

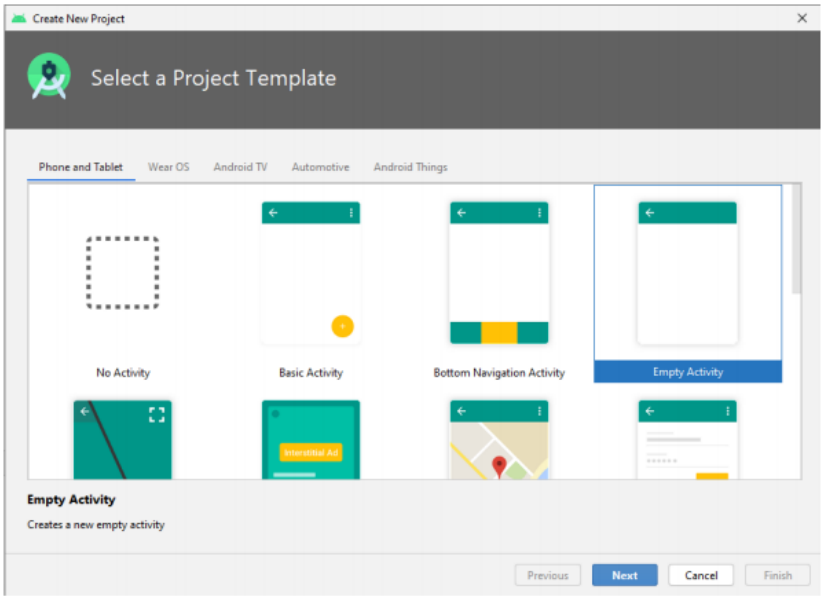

Рис 2.15. Вікно створення нового проекту у Android Studio.

Інтегроване середовище розробки Android Studio є віконним. Для того щоб максимально використати простір екрану і щоб не перевантажувати розробника Android Studio відображає лише невелику частину доступних вікон в будь-який час. Деякі вікна є контекстними і з'являються тільки в разі контекстного виклику з певних інструментів. Також Android Studio містить приховані вікни, які розрбник може активувати або деактивувати з меню налаштувань. Однією ж найважливіших функцій будь якого інтегрована середовища розробки є навігація. Проекти Android зазвичай складаються з багатьох тек, каталогів та файлів, що організовані між собою. Програмні додатки з багатьма вихідними файлами коду та ресурсами організовані в рамках структури проекту. Це зроблено для кращої класифікації файлів і визначення компіляції висхідного коду та генерації байт-коду. Вцілому ця файлова структура і називається проектом, який є організаційною одиницею, що являє собою повне програмне рішення. Проекти додатків зазвичай складаються з файлів висхідного коду Java або Kotlin, файлів конфігурації XML, зображення, стилі та інші ресурси в організаційній структурі. При створенні програми Android Studio допомогає у структуризації проекту. На

етапі збірки проекту створюються файли конфігурації та відбувається структуризація каталогів за ієрархією (рис 2.16).

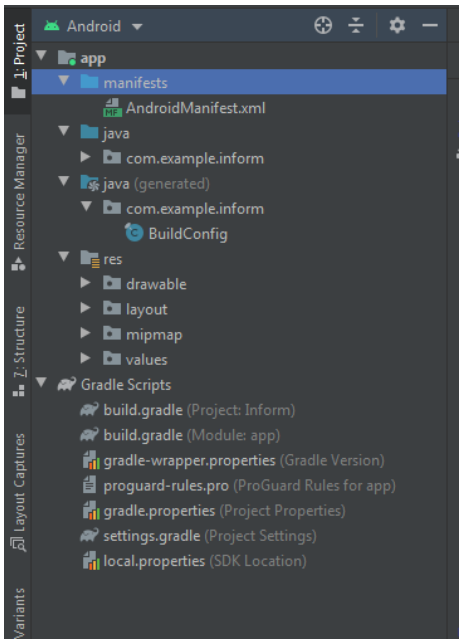

Рис. 2.16. Структура проекту в Android Studi

В процесі розробки додатків в інтегрованому середовищі розробки Android Studio необхідно буде компілювати та запускати додаток декілька разів. Розроблену програму Android можна перевірити, встановити та запускати на фізичному пристрої або на віртуальному пристрої Android (AVD – android virtual device), який називають емулятором. Перед тим як використовати віртуальний пристрій його необхідно завантажими та налаштувати. Емулятор надає практично всі можливості реального пристрою Android. Користувач може імітувати вхідні дзвінки так текстові повідомлення, вказувати місцезнаходження пристрою, імітувати обертання екрану та інші апаратні датчики, мати доступ до Google Play та інше. Тестування додатку на віртуальному пристрої (рис 2.17) в деякій мірі зручніше та швидше, ніж на фізичному пристрої. Наприклад, ви можете передавати дані в емулятор швидше, ніж на підключений фізичний пристрій.

Рис. 2.17. Віртуальний девайс в Android Studio

Після того як встановили Android Studio і запустили проект, потрібно з пустого макету (рис 2.18) зробити головний екран додатку який і буде зустрічати користувачів при вході в додаток

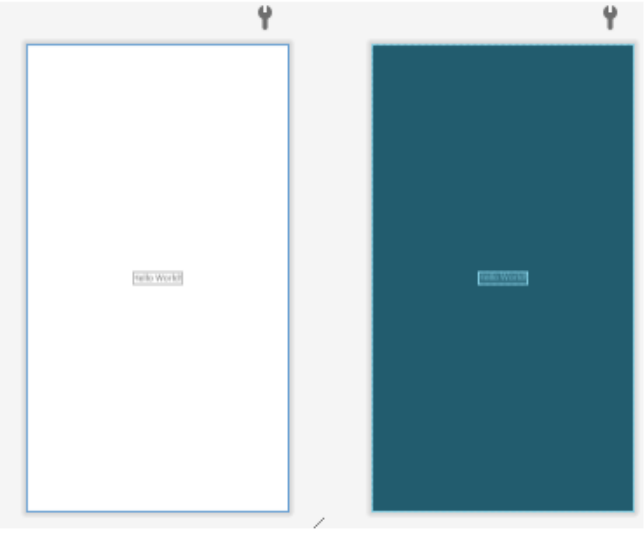

Рис 2.18. Пустий макет головного екрану

Щоб на ньому відображався список ступенів акредитації потрібно створити файл activity_main.xml. Тут розмістили елементи таким чином, щоб ними було комфортно користуватись і при цьому зберігся гарний візуальний

вигляд. Згодом вийшов такого вигляду шаблон (рис. 2.19). Тут розмішено текстове вікно, та три кнопки при натисканні на які з’являється новий шаблон.

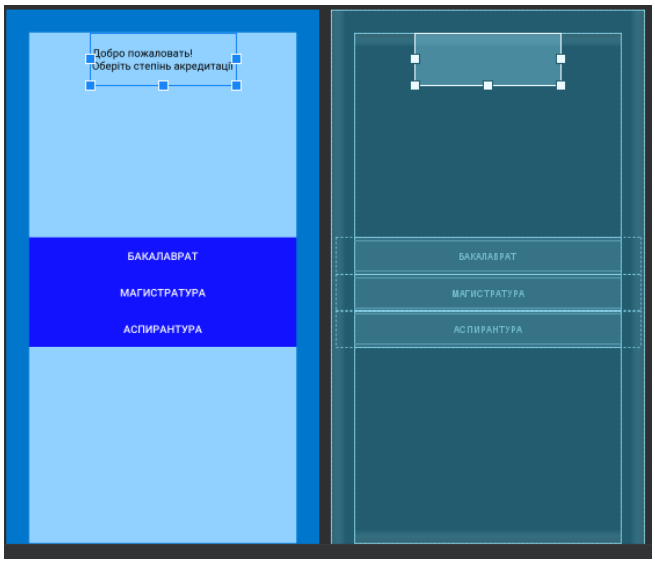

Рис. 2.19. Шаблон вигляду головного екрану.

Після створення шаблону необхідно було його налаштувати та зробити можливим перехід на інші вікна (рис 2.20).

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // найдем View-элементы
    tvOut = (TextView) findViewById(R.id.tvOut);
    button1 = (Button) findViewById(R.id.button1);
    button2 = (Button) findViewById(R.id.button2);
    button3 = (Button) findViewById(R.id.button3);

    View.OnClickListener onbutton1 = (view) -> {
            Intent intent = new Intent( packageContext: MainActivity.this, List1Activity.class);
            startActivity(intent) ;
    };
    button1.setOnClickListener(onbutton1);

    View.OnClickListener onbutton2 = (view) -> {
            Intent intent = new Intent( packageContext: MainActivity.this, List2Activity.class);
            startActivity(intent) ;
    };
    button2.setOnClickListener(onbutton2);

    View.OnClickListener onbutton3 = (view) -> {
            Intent intent = new Intent( packageContext: MainActivity.this, List3Activity.class);
            startActivity(intent) ;
    };
    button3.setOnClickListener(onbutton3);
```

Рис. 2.20. Код налаштування роботи головного екрану

Тепер після того як ми налаштували головний екран. Створюємо нові вікна для кожного елемента головного вікна. Щоб це зробити створюємо файли: activity_list1.xml, activity_list2.xml та activity_list3.xml та наповнюємо їх, за подобою шаблону головного екрану (рис. 2.21-2.23)

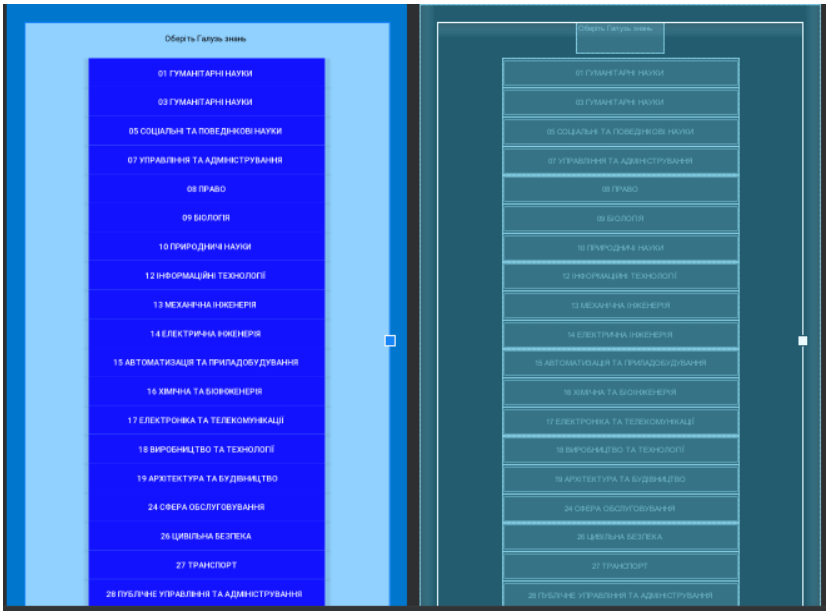

Рис 2.21. Шаблон вигляду екрану Бакалаврат

Рис. 2.22. Шаблон вигляду екрану магістратура

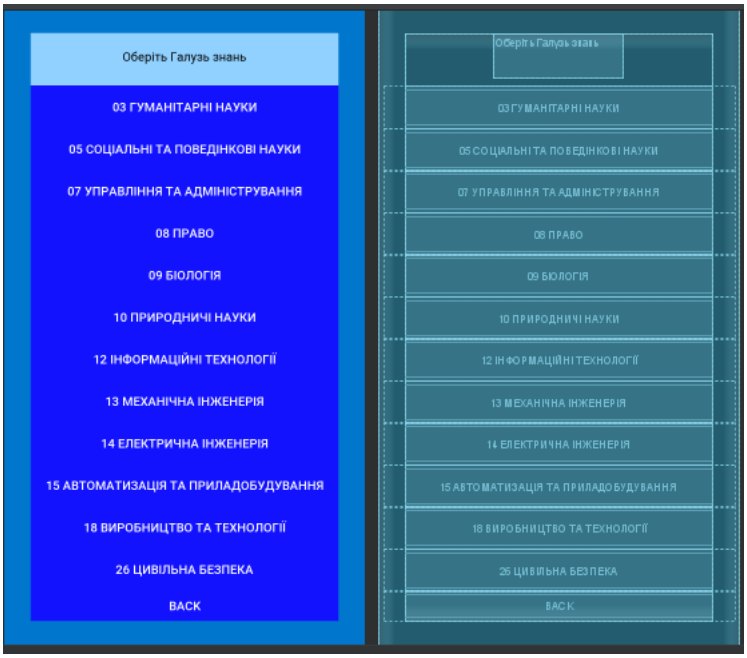

Рис. 2.23. Шаблон вигляду екрану Аспірантура

Після того як ми створили та налаштовували вікна з переходами, ми створюємо останнє вікно в якому буде відображатися зображення та інформація про спеціальність (рис. 2.24) та наповнюємо його необхідною інформацією (рис. 2.25)

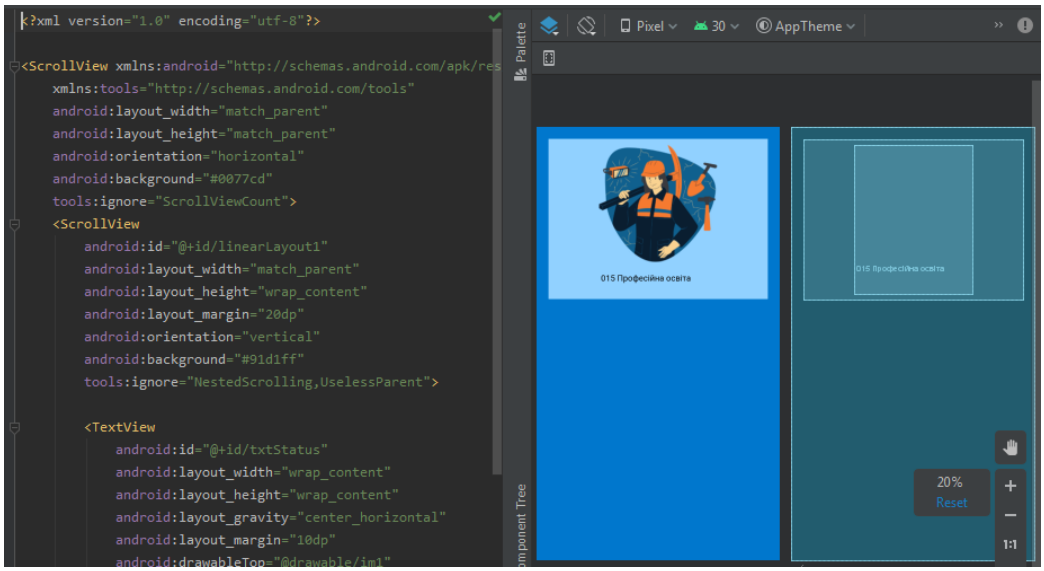

Рис. 2.24. Шаблон вигляду екрану зі спеціальністю

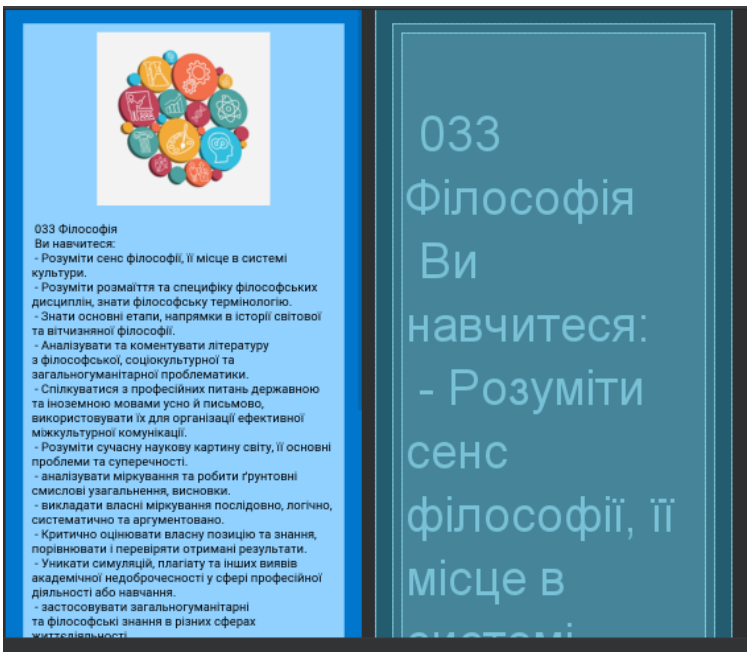

Рис. 2.25. Готовий шаблон із інформацією

2.5. Обґрунтування та організація вхідних та вихідних даних програми

В основному програмний засіб повинен працювати із форматами даних JSON. Всі вхідні дані будуть отримуватисть із віддалених серверів в цьому форматі. Даними сервер може заповняти тільки адміністратор (людина яка має відповідні права та доступ). З боку користувача програма ніякі дані не приймає, окрім формуванням відповідних команд, та вибору певних функцій.

Вихідними даними буде відображення інформації на екрані мобільного пристрою, відповідно до того що вибере користувач.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для технічних засобів обрана конфігурація, що забезпечує цілодобову роботу програмного комплексу. Для цих засобів були обрано сервер Dell PowerEdge 2600 Server, яких виконує функцію зберігання мобільного додатку і можливістю встановлення його на кишеньковий носій:

Як було зазначено вище, розроблена система буде працювати спільно з програмним комплексом. Для користувача ОС Android, для адміністратора ОС Windows. Тому, інформаційна система мобільного додаткупрацює під управлінням описаних вище програмних засобів. Технічні засоби були обрані, виходячи з необхідності забезпечити максимально відмовостійку систему для автоматизації процесу. Для реалізації необхідної відмовостійкості, а також для забезпечення необхідної швидкості обробки отриманої інформації, були обрано сервер Dell PowerEdge 2600 Server.

2.6.2. Використані програмні засоби

Для реалізації мобільного додатку, в якій буде зберігатися вся необхідна для роботи програми інформація, в даній роботі використана технологія автоматизованої збірки Gradle.

Для реалізації взаємозв'язку додатка з ОС Android використовується мова програмування Java.

Для реалізації клієнтської частини інформаційної системи використана середа візуального програмування Android Studio.

2.6.3. Виклик та завантаження програми

Після того як написаний весь програмний код необхідно створити apk файл, встановити його на телефон та розпочати тестування. Для збірки проекту в Android Studio вбудовано збірник проектів Gradle, де всі конфігурації прописуються на мові Groovy.

Тестування додатку – це перевірка додатку різними методами і способами на працездатність. Тестування необхідно як новому, так і вже працюючому додатку, для отримання гарантії його працездатності.

До основних видів тестування відносяться наступні:

• Тестування usability (перевірка зручності користування додатком). У

ході такого тестування визначається якість виконання і зручність інтерфейсу додатку, а так само проводяться роботи з виявлення можливих помилок в структурі. Результати дають можливість визначити, наскільки правильно використовує додаток «середньостатистичний» користувач, і як швидко він може знайти потрібні йому функції[19].

• Тестування на стійкість до великих навантажень. Цей тест імітує одночасне користування великою кількістю користувачів (сотень або навіть тисяч) для визначення працездатності ресурсу при великих навантаженнях або ж інтенсивна, довгочасна робота додатку в умовах не великих ресурсів. Таке тестування обов'язково для новинних додатоків, форумів і ресурсів з передбачуваною великою аудиторією. У ході тестування перевіряється не стільки сам ресурс, скільки комплексну роботу апаратної частини сервера, модулів, програмного ядра та інших компонентів додатку.

• Тестування XML коду. Перевіряється весь додаток на наявність помилок у програмному коді і відповідність стандартам.

• Тестування безпеки. Перевірка безпеки включає в себе тестування як самого додатку разом з, так і веб-сервера, операційної системи і всіх мережевих і локальних сервісів. Для збереження інформації та стабільної роботи додатку тестування безпеки необхідно проводити регулярно.

Оскільки додаток, як і будь-який інший програмний продукт не може бути без помилок, необхідно своєчасно виявляти і усувати ці помилки[10].

Після наповнення додатку контентом та повного його тестування створюється керівництво користувача, де детально пояснюється для чого створений додаток і як користуватись його специфічним функціоналом.

Запуск програми здійснюється через відкриття програми на талефоні.

2.6.4. Опис інтерфейсу користувача

Система дистанційного навчання розроблена для використання на мобільних пристроях (смартфони та планшети) під управлінням операційної системи “Android” з використанням підключення до мережі інтернет.

2.6.4.1 Інсталяція та системні вимоги

Додаток встановлюється на пристрій користувача шляхом встановлення “apk” файлу та його автоматичного налаштування. Оскільки додаток розроблений на платформі “Android”, пристрій користувача повинен мати цю операційну систему на своєму пристрої. Для коректної роботи додатку необхідно мати версію операційної системи не нижче 4.0.0. Також на пристрої користувача повинен бути доступ до мережі Інтернет.

2.6.4.2 Інструкція з використання програмного продукту

При вході в систему інформування студента/абітурієнта перед користувачем з’являється головне меню, в якому користувачу необхідно обрати один із запропонованих степенів акредитації. Вони розташовані у формі списку. Головне меню зображено на (рис. 2.26).

Рис. 2.26. Головне меню

Після того як користувач обрав один із запропонованих степенів акредитації, він натискає на нього. Перед ним з’являється вікно із вибором галузі знань. Вікна з галузями знань зображено на (рис. 2.7)

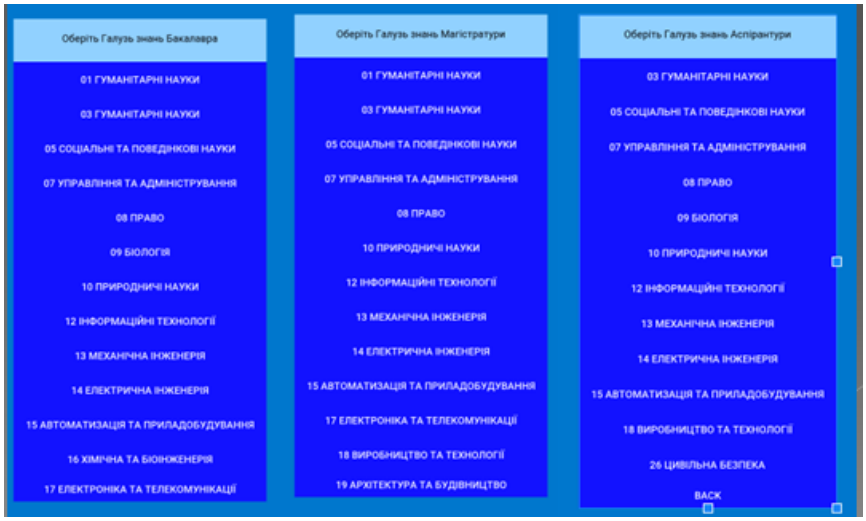

Рис. 2.27. Вікна з вибором галузі

Коли користувач обрав одну із запропонованих галузей, він натискає на цю галузь. Перед ним з’являється вікно з спеціальністю, в якому описується інформація про спеціальність яка знаходиться в вибраній галузі. Вікно з спеціальністю зображено на (рис 2.28).

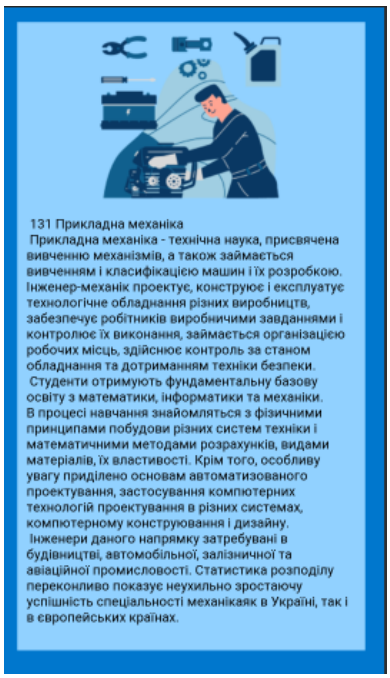

Рисунок 2.28. – Вікно спеціальності

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 1050;
- коефіцієнт складності програми – 1.8;
- коефіцієнт корекції програми в ході її розробки – 0,09;
- годинна заробітна плата програміста, грн / год – 50;
- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
- вартість машино-годин ЕОМ – 10 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = to + tu + ta + tn + t\text{отл} + t\text{д}\text{ , людино-годин,} \qquad (3.1)$$

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50),

tu - витрати праці на дослідження алгоритму рішення задачі,

ta - витрати праці на розробку блок-схеми алгоритму,

tn - витрати праці на програмування по готовій блок-схемі,

tотл - витрати праці на налагодження програми на ЕОМ,

tд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів,

C - коефіцієнт складності програми,

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1050 \cdot 1{,}8 \cdot (1 + 0{,}09) = 2060 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \qquad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 ... 1.5,

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2060 \cdot 1{,}3}{80 \cdot 1{,}1} = 30 \text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{2060}{22 \cdot 1{,}1} = 85 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{2060}{23 \cdot 1{,}1} = 81 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{2060}{5 \cdot 1{,}1} = 375 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.7}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.8}$$

$$t_{др} = \frac{2060}{18 \cdot 1{,}1} = 104 \text{ людино-годин,}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.9}$$

$$t_{до} = 104 * 0{,}75 = 78 \text{ людино-годин.}$$

$$t_д = 104 + 78 = 182 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 30 + 85 + 81 + 375 + 182 = 803 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.11)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 803 \cdot 50 = 40150, \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \quad (3.12)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год,

Смч - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 375 \times 10 = 3750 \text{ грн.}$$

$$Кпо = 3750 + 40150 = 43900, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.13)$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{803}{1 \cdot 176} = 4.56 \text{ міс.}$$

Висновки: визначено трудомісткість розробленого додатку (803 люд-год), проведений підрахунок вартості роботи по створенню програми (43900 грн.) та розраховано час на його створення (4,6 міс).

ВИСНОВКИ

Під час проходження розробки додатку було вивчено способи створення мобільних застосунків, їх взаємодію між собою а також з методами які можуть знаходитися у коді за стосунку та за допомогою яких він зможе працювати.

В процесі аналызу роботи був спроектований та розроблений програмний продукт для платформи Android, що повністю задовольняє вимоги. Було досягнуто наступне:

- Досліджено стан технологій створення мобільних додатків Обрана платформа Android, середовище розробки Android Studio, мова Java;

- Розроблено технічне завдання, структура, інтерфейс та дизайн додатку;

- Реалізовано найбільшн ефективний функціонал та дизайн мобільного додатку. Готовий додаток було протестовано.

Метою роботи додатку є спрощення процесу збору інформації студентом або абітурієнтом про спеціальності, кафедри та їх керівників.

Він надає користувачу доступ до інформації про спеціальності які викладаються та кафедри. Це дозволяє в повному обсязі отримати знання про певну спеціальність.

Розроблений інтерфейс зручний у користуванні, всі дії в додатку виконуються на інтуїтивному рівні та має приємне кольорове оформлення.

В економічному розділі визначено трудомісткість розробленого додатку 803 люд-год, проведений підрахунок вартості роботи по створенню програми 43900 грн. та розраховано час на його створення приблизно 5 міс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Огляд платформи Windows Mobile [Електронний ресурс]. - http://gamesmart.at.ua/publ/obzor_mobilnykh_os_platform/obzor_platformy_ windows_mobile/2-1-0-10

2. Введение в разработку для платформы Android [Електронний ресурс]. - http://android-club.com.ua/?articles=15

3. Огляд платформи Symbian - http://gamesmart.at.ua/publ/ obzor_mobilnykh_os_platform/obzor_platformy_symbian/2-1-0-8

4. Здзиарски Джонатан, iPhone Разработка приложений с открытым кодом - Санкт-Петербург: БХВ-Петербург, 2009, 386 стр.

5. Новая среда разработки Android Studio [Електронний ресурс]: cnews.ru – Режим доступу: http://www.cnews.ru/top/2013/05/20/novaya_sreda_ razrabotki_android_studio_sozdana_na_osnove_rossiyskogo_proekta529258

6. Android Studio – среда мобильной разработки на базе технологий JetBrains [Електронний ресурс]: soft.mail.ru – Режим доступу: http://soft.mail.ru/pressrl_page.php?id=51774

7. Операційна система Google Android [Електронний ресурс]: ALLS.IN.UA – Режим доступу: http://alls.in.ua/13729-operacijjna-sistema-googleandroid.html

8. Этапы разработки [Електронний ресурс]: itech-mobile.ru – Режим доступу: http://itech-mobile.ru/stages.html

9. UX – это не UI [Електронний ресурс]: cmsmagazine.ru – Режим доступу: http://www.cmsmagazine.ru/library/items/usability/ux-is-not-ui/

10. Родной язык Андроида [Електронний ресурс]: toster.ru – Режим доступу: https://toster.ru/q/8860

11. Языки программирования для Androidhttp [Електронний ресурс]: kakprosto.ru – Режим доступу://www.kakprosto.ru/kak-861348-yazykiprogrammirovaniya-dlya-android

12. Стандарт ISO/IEC 12207:2008 - https://ru.wikipedia.org/wiki/ISO/IEC_12207:2008

13. Android. Разработка приложений - https://infoshell.ru/blog/start-v-android-programmirovanii

14. Android SDK - https://ru.wikipedia.org/wiki/Android_SDK

15. Настройка Android SDK - https://metanit.com/java/android/1.7.php

16. Роджерс Р., Ломбардо Д. Android. Разработка приложений [Текст] / Роджерс Р., Ломбардо Д. – М.: ЭКОМ Паблишерз, 2010. — 400 с.

17. Java. Вікіпедія. Вільна енциклопедія [Електронний ресурс]. - Режим доступу: http://uk.wikipedia.org/wiki/Java

18. Genymotion — лучший эмулятор Андроид на ПК [Електронний ресурс]: 4idroid.com – Режим доступу: http://4idroid.com/genymotion-luchshijemulyator-android-na-pk/

19. Доля Android на рынке смартфонов [Електронний ресурс]: Ferra.ru – Режим доступу: http://www.ferra.ru/ru/techlife/news/ 2014/01/30/strategyanalytics-2013-smartphone/#.U6DXXyjiLt0

20. Panigrahy N. Xamarin Mobile Application Development for Android / Nilanchala Panigrahy. – Packt Publishing, 2015. – 296p.

21. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

22. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

23. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг головного вікна

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:background="#0077cd">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="30dp"
        android:background="#91d1ff"
        android:orientation="vertical"
        tools:ignore="UselessParent">
        <TextView
            android:id="@+id/tvOut"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginBottom="200dp"
            android:text="@string/text_00"
            android:textColor="@android:color/widget_edittext_dark" />
        <Button
            android:id="@+id/button1"
            android:layout_width="400dp"
            android:layout_height="wrap_content"
```

```
        android:layout_gravity="center_horizontal"
        android:background="#1212ff"
        android:text="@string/Bakalavrat"
        android:textColor="@android:color/white" />
    <Button
        android:id="@+id/button2"
        android:layout_width="400dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="#1212ff"
        android:text="@string/Magystratura"
        android:textColor="@android:color/white" />
    <Button
        android:id="@+id/button3"
        android:layout_width="400dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="#1212ff"
        android:text="@string/Aspyrant"
        android:textColor="@android:color/white" />
  </LinearLayout>
</LinearLayout>
```

Main.Activity.java

```
package com.example.inform;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
```

```
TextView tvOut;
Button button1;
Button button2;
Button button3;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   // найдем View-элементы
   tvOut = (TextView) findViewById(R.id.tvOut);
   button1 = (Button) findViewById(R.id.button1);
   button2 = (Button) findViewById(R.id.button2);
   button3 = (Button) findViewById(R.id.button3);

   View.OnClickListener onbutton1 = new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         Intent intent = new Intent(MainActivity.this, List1Activity.class);
         startActivity(intent) ;
      }
   };
   button1.setOnClickListener(onbutton1);

   View.OnClickListener onbutton2 = new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         Intent intent = new Intent(MainActivity.this, List2Activity.class);
         startActivity(intent) ;
      }
   };
   button2.setOnClickListener(onbutton2);
```

```
        View.OnClickListener onbutton3 = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this, List3Activity.class);
                startActivity(intent) ;
            }
        };
        button3.setOnClickListener(onbutton3);
    }

}
```

Лістинг AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.inform"
    android:installLocation="auto">
<uses-feature android:name="list01"/>

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="WrongManifestParent"
        android:allowBackup="false">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>

        <activity android:name=".List1Activity"
            android:label="@string/about_title">
        </activity>

        <activity android:name=".List2Activity"
            android:label="@string/about_title">
        </activity>

        <activity android:name=".List3Activity"
            android:label="@string/about_title">
        </activity>
<activity android:name=".ListActivity03"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity05"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity07"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity10"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity12"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity13"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity15"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity18"
```

```
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity19"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity27"
            android:label="@string/about_title">
        </activity>
        <activity android:name=".ListActivity29"
            android:label="@string/about_title">
        </activity>
    </application>
</manifest>
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему: «Розробка мобільного застосунку для системи інформаційного забезпечення вищого навчального закладу»

студента групи 122-18ск-2 Єрмакова Олександра Андрійовича

Керівник економічного розділу
Зав. каф. ПЕП та ПУ, д.е.н. **О. Г. Вагонова**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Єрмаков.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Єрмаков.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Єрмаков.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Єрмаков.ppt	Презентація кваліфікаційної роботи