МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук____
Кафедра інформаційних технологій

ДИПЛОМНА РОБОТА

Рівень вищої освіти бакалавр

на тему: Розробка мобільного додатку "MySchool на базі операційної системи Android"

Виконав студент 4 курсу групи К-45
Напрям підготовки 122 Комп'ютерні науки та інформаційні технології_____

Керівник асистент_____
Шуптар Наталія Йосипівна_____

Консультант к.геогр.н.,доцент___
Кузніченко Світлана Дмитрівна__

Рецензент геогр.н.,доцент_____
Лужбін Анатолій Михайлович___

Одеса 2018

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

OS (ОС) – операційна система

ПЗ – програмне забезпечення

API – Application Programming Interfaces

APK – Application Package Kit

GUI – Graphical User Interface

JSON – JavaScript Object Notation

SDK – Software Development Kit

XML – Extensible Markup Language – розширювана мова розмітки

ВСТУП

Людині XXI віку важко уявити своє життя без високотехнологічних пристроїв, як-то смартфони, планшетні ПК, бо саме ці науково-технічні новинки дозволяють не тільки підтримувати зв'язок з людьми із різних куточків світу, а й допомагають у формуванні та розвитку сучасного інформаційного простору.

Сфера IT прогресує кожного дня. Розробки у сфері мобільних додатків стають дедалі різноманітнішими, бо навіть побутова техніка(телевізори, холодильники і т.д.) зараз керується під Android OS. Сьогодні за допомогою смартфону можна не тільки робити високоякісні зображення, а й переглянути інформацію стосовно будь-кого або -чого(сторінки у соц.мер., блогах, інфо.порталах і т.д.).

Наявність сучасних технологій дає нам змогу не тільки полегшити життя, а ще й економить наш час. Інформація та дані все частіше розглядаються як життєво важливі ресурси, які повинні бути організовані таким чином, щоб ними можна було легко користуватися. Саме тому розробка нового інформаційного додатку для школярів є досить актуальною. Наявність збалансованого розширення для мобільних може стати у нагоді не тільки школярам, а й батькам, які зможуть контролювати та поліпшувати учбовий процес своїх дітей.

В наш час, найпоширенішою оперативною системою для мобільних пристроїв є ОС Android. Android підтримує велику кількість засобів розробки та девайсів від різних виробників. Головна причина поширення ОСAndroid-безкоштовні засоби розробки, в той час як розробка під систему IOS вимагає високих початкових витрат.

Розробка даного мобільного додатку має на меті створити за допомогою сучасної інтегрованої середи розробки (IDE) AndroidStudio, що містить усі необхідні інструменти та бібліотеки для роботи з XML-розміткою(дизайн, інтерфейс), Java-кодом(логіка взаємодії даних) та побудови проектів, зручне та інтуїтивно зрозуміле розширення, яке полегшить та зможе допомогти удосконалювати усі аспекти навчання у школі. Діти(а також батьки) матимуть змогу автоматизувати процес навчання, отримуючи з додатку усю необхідну для цього інформацію та набір інтерактивних інструментів.

Даний дипломний проект містить сторінок, рисунків, посилань.

1 ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика предметної області

Користувач додатку має отримати максимально зручне у повсякденному використанні мобільне розширення, яке було б водночас максимально інформативним та швидким у користуванні. Усі користувачі мають пройти етап реєстрації, які включатимуть персональні дані для подальшої авторизації. При реєстрації має бути враховані поля заповнення даних про користувача(наприклад мобільний номер та адреса ел.пошти.)

Для якісної та безвідмовної роботи треба залучити найостанніші розробки у сфері інформаційних технологій. Сервіс, що надаватиме інформацію стосовно успішності, буде працювати під операційною системою Android, яка лишається найпоширенішою ОС для смартфонів.

Мобільний додаток має надавати інформаційну базу, що відображатиметься у вигляді новин щодо організації навчальним закладом науково-дослідних, спортивних та розважальних заходів, інформацію про результати конкурсів та ін.

Батькам дане ПЗ надаватиме змогу вчасно отримувати результати навчання своїх дітей у школі, щоб за потреби мати змогу оперативно зреагувати на відставання по дисциплінах, поведінці, тощо.

Додаток має відображати для користувача наступну інформацію:

– розклад шкільних занять;

– вчительський склад;

– історія школи;

– інформація про гуртки, секції та ін.;

Додаток "My School" дозволяє по-новому поглянути на шкільну освіту. Оптимізувати багато елементів навчання, що дає додаткові можливості для вдосконалення педагогічного процесу і , як наслідок, зростання навчальних показників в якісному і кількісному сенсі, додаткове мотивування дітей. Ресурс об'єднує всі школи в єдину інформаційно-аналітичну систему, яка дозволяє системно і централізовано керувати навчальним процесом. Доступні опції контролю оцінок і відвідуваності учнів. Кожен день на телефон користувача приходить інформаційне зведення з результатами оцінок і відвідуваності. Інтерфейс і система меню логічні та інтуїтивно розташовані і не потребують поглибленого вивчення. Зручно представлено розклад уроків та список вчителів, якы проводять заняття.

1.2 Аналіз існуючих мобільних додатків

На сьогоднішній день існують декілька мобільних додатків для школярів (COMSOL MySchool, FlaringApp School, EduPage), але всі вони мають один й теж функціонал, не відповідають як вимогам усіх користувачів, так і критеріям сучасного мобільного додатку (рис. 1.1).

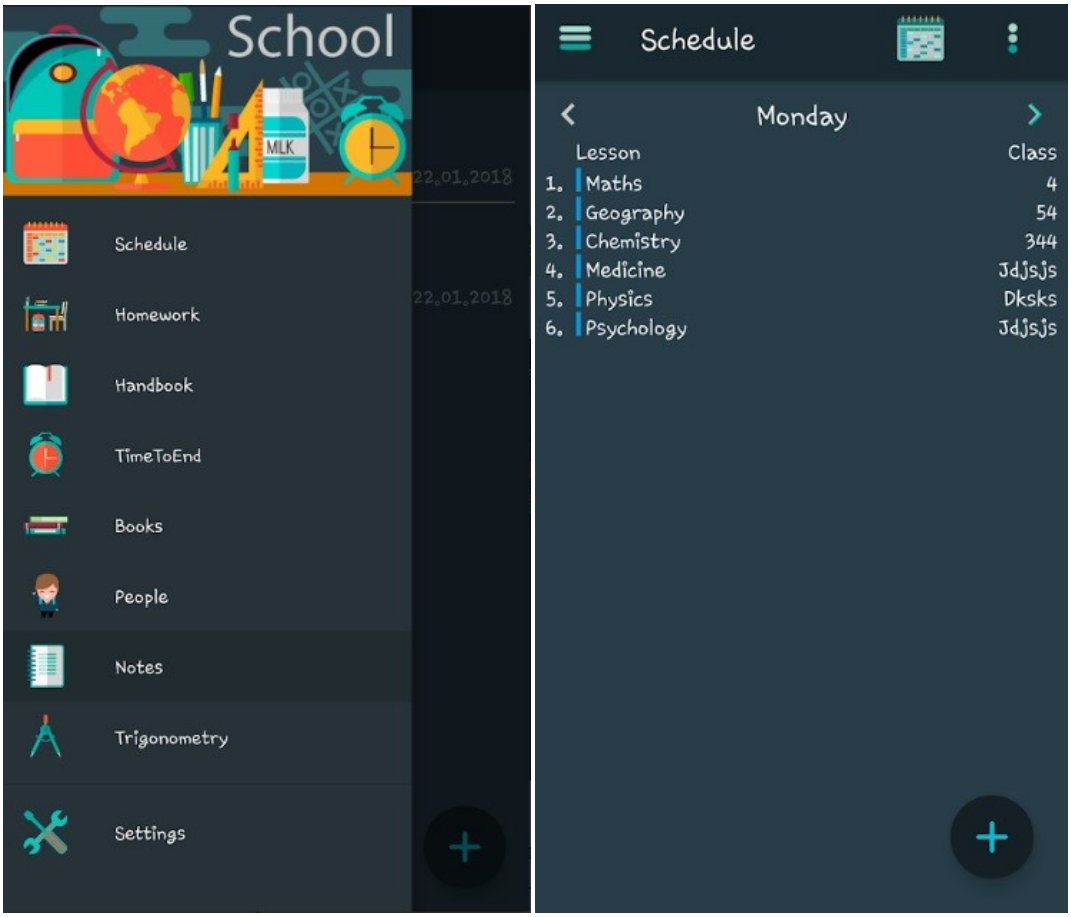

Рисунок 1.1 – Інтерфейс FlaringApp School

Як приклад, є додаток з Play market'y My School, розробник COMSOL. Цей програмний продукт дозволяє переглядати інформацію стосовно учбового процесу, але не надає жодної конкретної інформації стосовно особистих учбових закладів. Тому саме цей недолік спонукає до розробки більш різноманітної палітри функціоналу мобільного додатку (рис. 1.2).

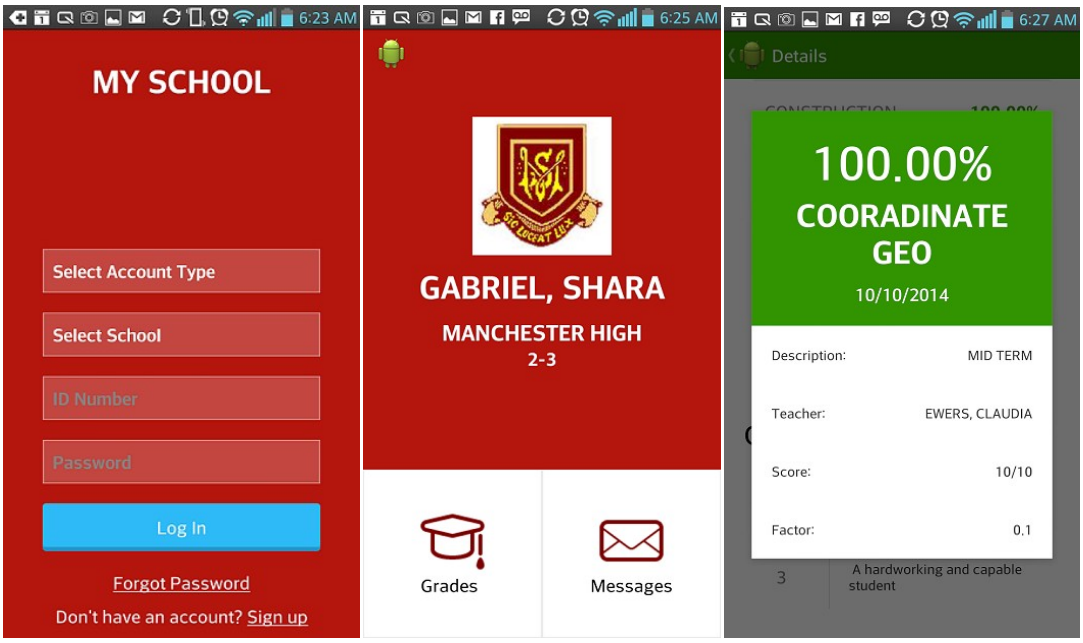

Рисунок 1.2 – Інтерфейс COMSOL My School

Проаналізувавши усі мобільні додатки та відгуки користувачів, були виведені основні вимоги до розробки (рис. 1.3).

Рисунок 1.3 – Інтерфейс мобільного додатку EduPage

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Інтегроване середовище розробки Android Studio

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

Android Studio прийшло на зміну плагіну ADT для платформи Eclipse. Середовище побудоване на базі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0.

Бінарні складання підготовлені для Linux (для тестування використаний Ubuntu), Mac OS X і Windows. Середовище надає засоби для розробки застосунків не тільки для смартфонів і планшетів, але і для носимих пристроїв на базі Android Wear, телевізорів (Android TV), окулярів Google Glass і автомобільних інформаційно-розважальних систем (Android Auto). Для застосунків, спочатку розроблених з використанням Eclipse і ADT Plugin, підготовлений інструмент для автоматичного імпорту існуючого проекту в Android Studio.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

2.1.1 Налаштування SDK

Комплект розробки програмного забезпечення (SDK або devkit), як правило, являє собою набір інструментів розробки програмного забезпечення, що дозволяє створювати програми для певного пакета програмного забезпечення, програмного забезпечення, апаратної платформи,

комп'ютерної системи, відеоігрової консолі, операційної системи або аналогічної платформи розробки. Щоб збагатити додатки за допомогою розширених функцій, реклами, натискання сповіщень тощо, більшість розробників програм впроваджують окремі комплекти розробки програмного забезпечення. Деякі SDK є надзвичайно важливими для розробки спеціальної програми. Наприклад, для розробки додатка для Android на платформі Java потрібен набір Java Development Kit, для iOS-додатків SDK для iOS, а для універсальної платформи Windows - платформи .NET Framework SDK (рис. 2.1).

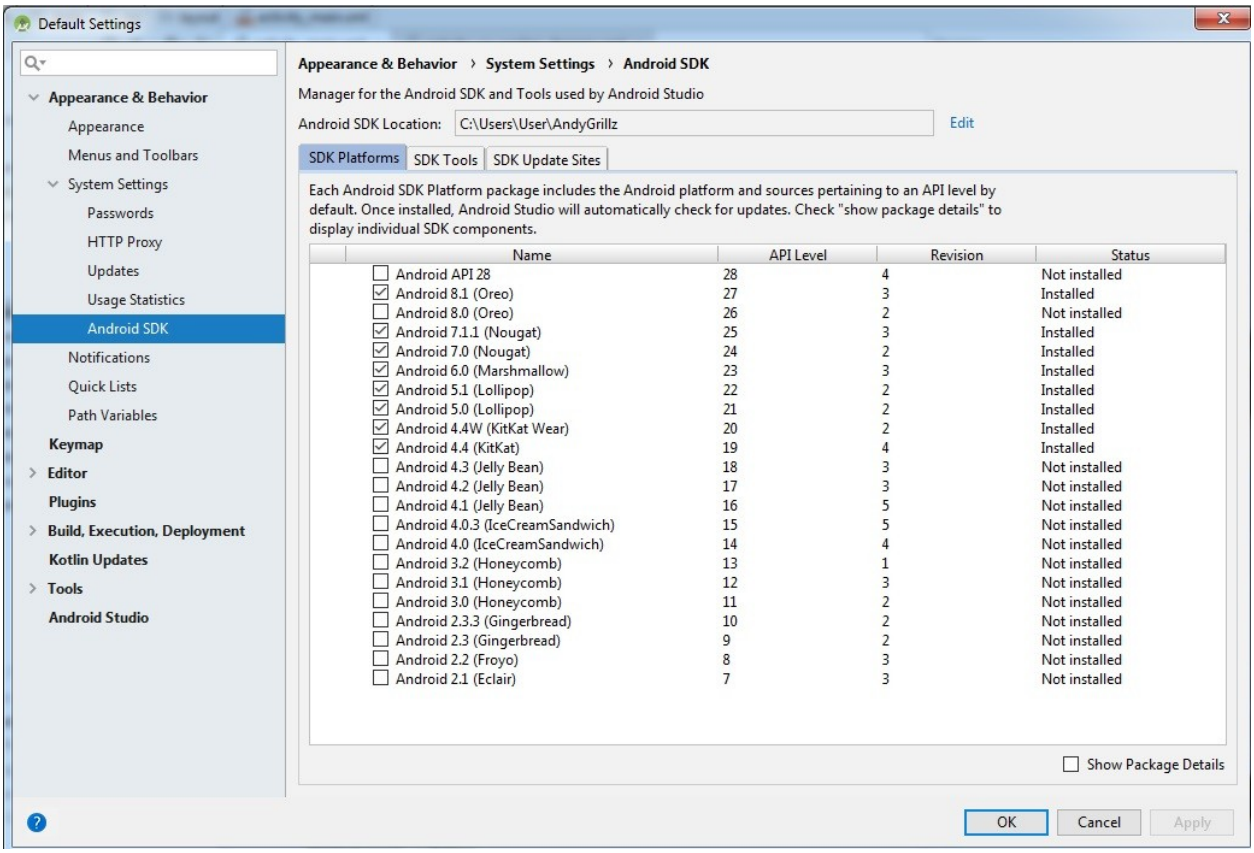

Рисунок 2.1 – Android SDK Manager

2.1.2 Інтерфейс та основні інструменти

Для прискорення розробки додатків представлена на рис. 2.2 колекція типових елементів інтерфейсу і візуальний редактор для їхнього

компонування, що надає зручний попередній перегляд різних станів інтерфейсу додатку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub.

До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.

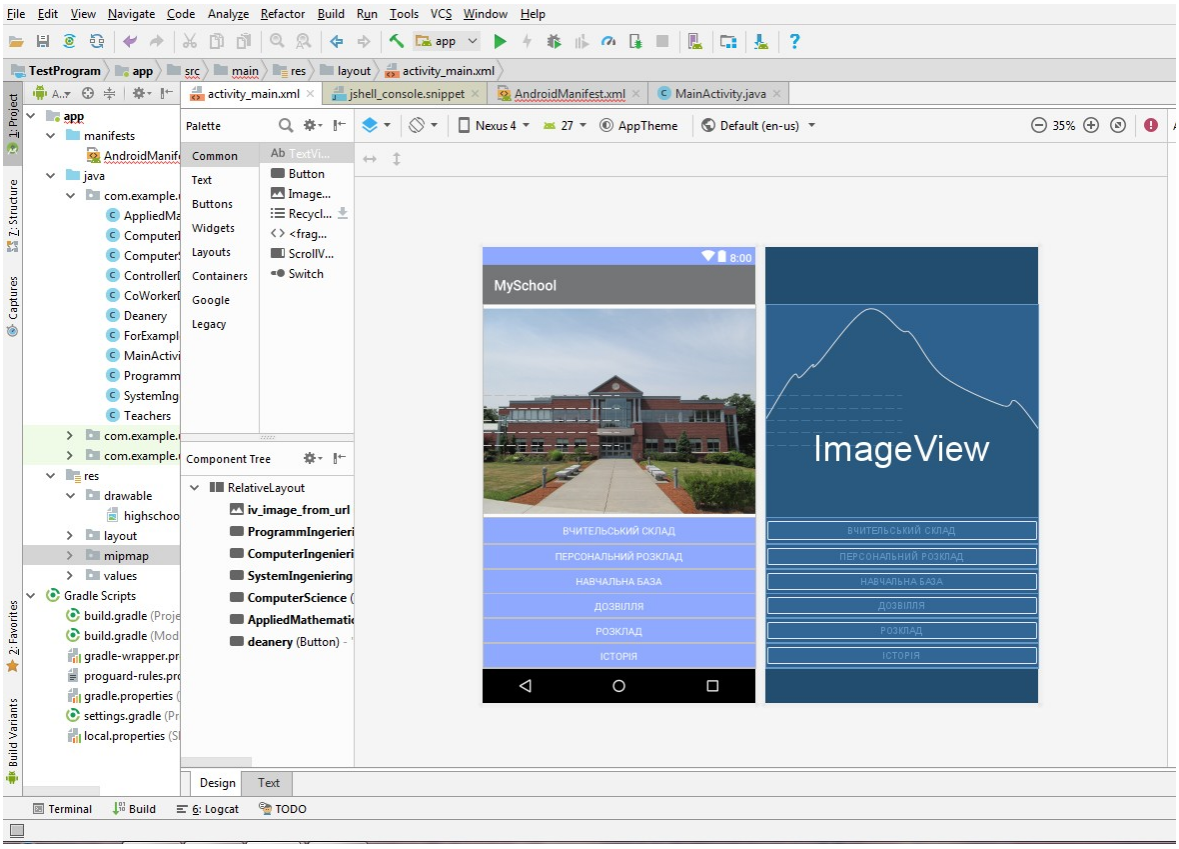

Рисунок 2.2 – Інтерфейс IDE Android Studio

Основні передбачені функції такі:

– Живі макети (layout): редагувальник WYSIWYG — живе кодування — подання (rendering) програми в реальному часі.

– Консоль розробника: підказки по оптимізації, допомога по перекладу, стеження за напрямком, агітації та акції — метрики Google аналітики.

– Резерви бета релізів та покрокові релізи.

– Базування на Gradle.

– Android-орієнтований рефакторинг та швидкі виправлення.

Lint утиліти для охоплення продуктивності, юзабіліті, сумісності версій та інших проблем.

Використання можливостей ProGuard та підписів до програм.

– Шаблони для створення поширених Android дизайнів та компонентів.

– Багатий редактор макетів (layouts) що дозволяє користувачам перетягнути і покласти (drag-and-drop) компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети (layouts) на різних конфігураціях екранів.

2.2 Загальні поняття Android-розробки

Activity (Діяльність). Android додаток включає в себе одну або кілька діяльностей (activities). Діяльність можна представити у вигляді контейнера, що містить користувальницький інтерфейс і код, який його запускає.

Наміри (intents) складають систему повідомлень на Android. Намір складається з дії (action), яку необхідно виконати (подивитися, редагувати і ін.) і даних.

Дія - це те, що повинно бути зроблено при отриманні наміру і даних, з якими необхідно оперувати. Наміри використовуються при запуску діяльності і при комунікації між різними частинами Android системи. Додаток може отримувати або відправляти наміри. При передачі наміри фактично відправляється повідомлення системі зробити що-небудь, наприклад, запустити нову діяльність в поточному додатку або відкрити іншу програму. Якщо просто відправити намір, це не означає, що щось станеться автоматично. Для цього необхідно зареєструвати приймач намірів (intent receiver), який отримує намір і вказує Android системі, що потрібно зробити: виконати завдання в новій діяльності або запустити іншу програму. Якщо ж є кілька приймачів для отримання intent, можна створити інструмент, що дозволяє користувачеві самому вибрати необхідну дію. Одним наміром можуть бути знайдені кілька приймачів, тому користувач особисто визначає

action, яке потрібно виконати. Наприклад, при довгому натисненні на зображення в галереї, з'являється інструмент вибору, який пропонує відправити картинку через e-mail або соціальні мережі, редагувати або видалити і ін. Якщо система не може знайти підходящий намір, а інструмент вибору не було створено розробником, то додаток потерпить крах і видасть помилку виконання. Тому важливо стежити за тим, щоб вибори для комерційних, чи не націлених на інші діяльності в даному додатку, були створені.

Віджети та видові вікна. Видова вікно (view) являє собою базовий елемент управління інтерфейсу у вигляді прямокутної області, де можна малювати і обробляти події. Прикладами видових вікон є: контекстне меню (ContextMenu), меню (Menu), вид (View), поверхня малювання (SurfaceView). Віджети (widgets) - це більш просунуті елементи призначеного для користувача інтерфейсу, наприклад, прапорці з перемикачами, де можна вибрати один з декількох можливих станів. Віджети є тими елементами управління, з якими взаємодіє користувач. До віджетів відносяться: кнопка (Button), вибір дати (DatePicker), галерея (Gallery), прапорець (CheckBox) та ін. Таким чином, видові вікна і віджети є елементами управління користувальницького інтерфейсу, проте перші здатні виконати не одну, а кілька функцій.

2.3 Kotlin — головний програмний інструмент.

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

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

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

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

Дві головні особливості Kotlin, якими розробники керуються при створенні цієї мови, є його простота і повна сумісність з Java. Kotlin створювався компанією, яка робить дуже багато продуктів на Java і яка добре розбирається в сучасних інструментах розробки (рис. 2.3). Запит на нову мову витає в повітрі давно, але зробити таку мову, яка би дозволяла взяти (велику) готову кодову базу Java, звичайних Java-розробників, дати їм новий інструмент і плавно (але більш ефективно) продовжувати розробку — такого інструменту до появи котлина не існувало. Творці нової мови, здається, дуже добре відчули потреби бізнесу та розробників: бізнесу дали можливість збільшити ефективність розробників, а розробникам дати сучасний інструмент для розробки. І коли кажеться про «сучасний інструмент», звичайно, мається на увазі не тільки компілятор, але і підтримка в IDE, без якої зазвичай діяльність розробника програмного заюезпечення бачиться зовсім немислимою.

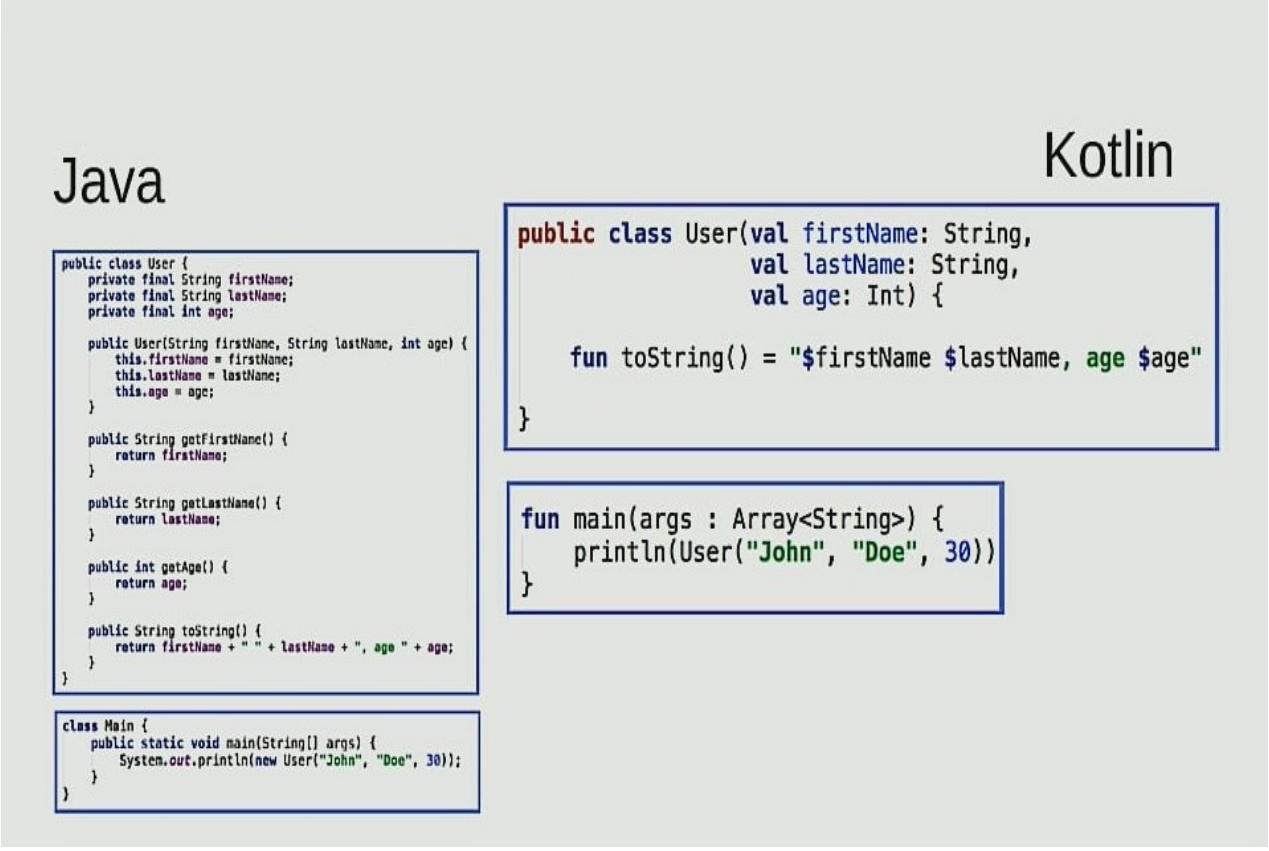

Рисунок 2.3 – Порівняння Java та Kotlin

Kotlin чудово підходить для розробки додатків Android, що приносить всі переваги сучасної мови для платформи Android без введення нових обмежень:

– Сумісність: Kotlin повністю сумісна з JDK 6, тож додатки Kotlin можуть працювати на старих пристроях Android без проблем. Інструмент Kotlin повністю підтримується в Android Studio і сумісний з системою Android build.

– Продуктивність: додаток на Kotlin працює так само швидко, як еквівалентний Java, завдяки дуже схожим структурам байт-кодів. За підтримки Kotlin для вбудованих функцій кодування за допомогою лямбда часто виконується навіть швидше, ніж той же код, написаний на Java.

– Оперативна сумісність: котлін є 100% сумісним з Java, що дозволяє використовувати всі існуючі бібліотеки Android у програмі Kotlin. Це включає в себе обробку анотацій, таким чином, також доступні прив'язка даних та Dagger.

– Час компіляції: котлін підтримує ефективну комбіновану поетапну компіляцію, так що при наявності додаткових накладних витрат на чисті збірки додаткові збірки зазвичай стають швидкими та швидшими, ніж з Java.

У підсумку: простота дозволяє використовувати мову майже будь-якому Java-розробнику, який готовий витратити півгодини на те, щоб подивитися туторіал(навчальний посібник) або специфікацію мови, зворотна сумісність дозволяє використовувати мову у вже існуючому проекті.

2.3 Розширювана мова розмітки XML

Стандарт XML (англ. Recommendation, перше видання від 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (англ. Schema), такими як XML Schema (XSD), DTDабо RELAX NG. На сьогоднішній день існує набір мов, які були

засновані за допомогою технології XML а саме: XSLT, XAML, XUL, RSS, MathML, XHTML, SVG, XMLSchema.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.user.testprogram">

    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:fontFamily="Arial"
        android:icon="@mipmap/ic_launcher"
        android:label="MySchool"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <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=".ComputerIngeniering"
            android:label="Розклад">
            <intent-filter>
                <action android:name="com.example.user.testprogram.ComputerIngeniering" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ProgrammIngeriering"
            android:label="Дозвілля">
            <intent-filter>
                <action android:name="com.example.user.testprogram.ProgrammIngeriering" />
```

Рисунок 2.4 – Приклад XML файлу

XML-документ має ієрархічну логічну структуру, і може представлятись у вигляді дерева. Вузлами цього дерева можуть бути:

1) елементи, фізична структура яких складається із:

– коректної пари відкриваючого та закриваючого тегів (<Назва-тега>) та (</Назва-тега>), або

– тега порожнього елемента (<Назва-тега />),

2) Атрибути, що мають вигляд пар ключ-значення (назва атрибута="значення атрибута") і знаходяться або у відкриваючому, або у порожньому тезі (подібно до метаданих),

3) Вказівки щодо обробки документа (англ. Processing Instruction) (<? Обробник параметр ?>)

4) Коментарі (<!-- Текст коментаря -->)

5) Текст, або у вигляді простого тексту, або фрагментів CDATA (<![CDATA[довільний текст]]>).

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

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

З точки зору активного аналізу (англ. *Pull parsing*) XML-документ розглядається як послідовність елементів, які зчитуються послідовно, використовуючи шаблон проектування ітератор. Такий підхід дозволяє створення рекурсивних аналізаторів, у яких структура коду відображає структуру аналізованих XML-документів, проміжні результати аналізу можуть бути використані і розміщені у вигляді локальних змінних в підпрограмах, що виконують аналіз, передані як параметри до підпрограм нижчого рівня або повернені до підпрограм вищого рівня. До прикладів активних аналізаторів можна віднести такі, як StAX у мові програмування Java, SimpleXML у PHP та System.Xml.XmlReader у .NET.

Активний аналізатор створює ітератор, що послідовно обходить різні елементи, атрибути та дані в XML-документі. Код, що використовує цей «ітератор», може перевіряти поточний елемент (аби дізнатись, наприклад, чи є цей елемент стартовим, кінцевим або текстовим), та дізнаватись про його атрибути (локальна назва, простір назв, значення XML-атрибутів, зміст тексту тощо) і може пересунути ітератор на наступний елемент. Таким чином, код аналізатора може зчитувати інформацію із документа під час обходу. Підхід рекурсивного спуску сприяє зберіганню даних у вигляді типізованих локальних змінних в коді аналізатора, в той час як SAX, наприклад, зазвичай вимагає, аби аналізатор явно зберігав проміжні дані в стеку елементів, що є вищими елементами від того, який зараз аналізується. Код активного аналізатора може бути більш прямолінійним та зрозумілим і простішим для підтримки, аніж код SAX аналізатора.

2.4 Автоматизація побудови із Gradle

Автоматизація побудови – етап написання скриптів або автоматизація широкого спектру завдань, вживаного розробниками в їхній повсякденній діяльності. Включає в себе такі дії, як:

- компіляція сирцевого коду в бінарний код
- складання бінарного коду
- виконання тестів
- розгортання програми на виробничій платформі
- написання супровідної документації або опис змін нової версії.

Історично так склалося, що розробники застосовували автоматизацію складання для виклику компіляторів і компонувальників зі скрипта складання, на відміну від виклику компілятора з командного рядка. Досить просто за допомогою командного рядка передати один сирцевий модуль компілятору, а потім і компонувальнику для створення кінцевого об'єкта. Однак, при спробі скомпілювати або скомпонувати велику кількість модулів з сирцевим кодом, причому в певному порядку, здійснення цього процесу вручну за допомогою командного рядка виглядає занадто незручним. Набагато привабливішою альтернативою є скриптова мова, підтримувана утилітою Make. Цей інструмент дозволяє писати скрипти складання, визначаючи порядок їхнього виклику, етапи компіляції і компонування для складання програми. GNU Make також надає такі додаткові можливості, як наприклад, «залежності» («makedepend»), які дозволяють вказати умови підключення сирцевого коду на кожному етапі складання. Це і стало початком автоматизації складання. Основною метою була автоматизація викликів компіляторів і компонувальника. У міру зростання і ускладнення процесу складання розробники почали додавати дії до і після викликів компіляторів, як наприклад, перевірку (check-out) версій копійованих об'єктів на тестову систему. Термін «автоматизація складання» вже включає в себе управління і дії до і після компіляції і компонування, так само як і дії при компіляції і компонуванні.

Gradle - це система автоматизації побудови з відкритим кодом, яка базується на концепціях Apache Ant і Apache Maven, а також вводить специфічну для домену мову (DSL) на основі Groovy замість форми XML, яку використовує Apache Maven для оголошення конфігурації проекту. Gradle використовує спрямований ациклічний графік ("DAG"), щоб визначити порядок виконання завдань (рис. 2.5).

Gradle був розроблений для створення багато-проектних збірок, які можуть бути досить великими. Він підтримує додаткові збірки, розумно визначаючи, які частини дерева збірки оновлюються; будь-яке завдання, залежне лише від цих частин, не потрібно повторно виконувати.

Переваги:

Поліпшення якості продукту

- Прискорення процесу компіляції і компонування
- Позбавлення від зайвих дій
- Мінімізація «поганих (некоректних) складань»
- Позбавлення від прив'язки до конкретної людини
- Ведення історії складання і релізів для розбору випусків
- Економія часу і грошей завдяки причинам, вказаним вище.

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

android {
    compileSdkVersion 25
    buildToolsVersion '27.0.3'
    defaultConfig {
        applicationId "com.example.user.testprogram"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    api fileTree(dir: 'libs', include: ['*.jar'])
    androidTestImplementation('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    //noinspection GradleDynamicVersion
    implementation 'com.android.support:appcompat-v7:25.3.1+'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.squareup.picasso:picasso:2.5.2'
    implementation 'com.android.support:support-v4:25.3.1'
    testImplementation 'junit:junit:4.12'
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
}
repositories {
    mavenCentral()
}
```

Рисунок 2.5 – Gradle-файл

Зіткнувшись з труднощами з доступом до інформації з інших модулів при складанні стало зрозуміло, що можливість написати перебір модулів збірки з використанням замикань в багатьох випадках просто «дар Божий».

Так само як і спосіб налаштувати стандартні параметри для всіх підпроектів в головному скрипті gradle – шедевр, що набагато перевершує концепцію наслідування в Maven.

Головною перевагою стала можливість створювати скрипти для тих частин збоку, які занадто складно описати в термінах «build by convention». У Gradle можна не тільки задавати залежності між модулями, а й неймовірно гнучко описувати залежності від завдань, модулів, каталогів і т.д. Наприклад, в Gradle вже можна сказати, що один модуль залежить від скомпільованих класів, а не від результату (jar) збірки іншого.

2.5 Файл маніфесту

У кореневій папці кожного додатка повинен знаходитися файл AndroidManifest.xml (який саме так і називається). Файл маніфесту містить важливу інформацію про програму, яка потрібна системі Android. Тільки отримавши цю інформацію, система може виконати будь-якої код програми. Серед іншого файл маніфесту виконує наступні дії:

1) Він задає ім'я пакета Java для програми. Це ім'я пакета є унікальним ідентифікатором додатки.

2) Він описує компоненти програми - операції, служби, приймачі широкомовних повідомлень і постачальників контенту, з яких складається програма. Він містить імена класів, які реалізують кожен компонент, і публікує їх можливості (вказує, наприклад, які повідомлення Intent вони можуть приймати). На підставі цих декларацій система Android може визначити, з яких компонентів складається додаток і за яких умов їх можна запускати.

3) Він визначає, в яких процесах будуть розміщуватися компоненти програми.

4) Він оголошує, які дозволи повинні бути видані з додатком, щоб воно могло отримати доступ до захищених частинах API-інтерфейсу і взаємодіяти з іншими додатками.

5) Він також оголошує дозволу, необхідні для взаємодії з компонентами цього додатка.

6) Він містить список класів Instrumentation, які при виконанні додатка надають інформацію у профілі та іншу інформацію. Ці оголошення присутні в файлі маніфесту тільки під час розробки та налагодження програми і видаляються перед його публікацією.

7) Він оголошує мінімальний рівень API-інтерфейсу Android, який потрібно синхронізувати з додатком.

8) Він містить список бібліотек, з якими має бути пов'язане додаток.

Наведена далі схема дозволяє ознайомитися із загальною структурою файлу маніфесту і всіма елементами, які можуть в ньому міститися. Кожен елемент разом з усіма своїми атрибутами, повністю описується в окремому файлі. Щоб переглянути детальну інформацію про будь-якому елементі, клацніть ім'я елемента на схемі, в алфавітному списку елементів, наведеному після схеми, або в будь-якому іншому місці, де цей елемент згадується.

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

  <uses-permission />
  <permission />
  <permission-tree />
  <permission-group />
  <instrumentation />
  <uses-sdk />
  <uses-configuration />
  <uses-feature />
  <supports-screens />
  <compatible-screens />
  <supports-gl-texture />

  <application>

    <activity>
      <intent-filter>
        <action />
        <category />
        <data />
      </intent-filter>
      <meta-data />
    </activity>

    <activity-alias>
      <intent-filter> . . . </intent-filter>
      <meta-data />
    </activity-alias>
```

```
    <service>
        <intent-filter> . . . </intent-filter>
        <meta-data/>
    </service>

    <receiver>
        <intent-filter> . . . </intent-filter>
        <meta-data />
    </receiver>

    <provider>
        <grant-uri-permission />
        <meta-data />
        <path-permission />
    </provider>

    <uses-library />

  </application>

</manifest>
```

2.6 Бібліотека зображень Picasso for Andoid

Android Picasso – це потужна бібліотека для завантаження та кешування та модифікації зображень, залучених до додатку (рис.2.6).

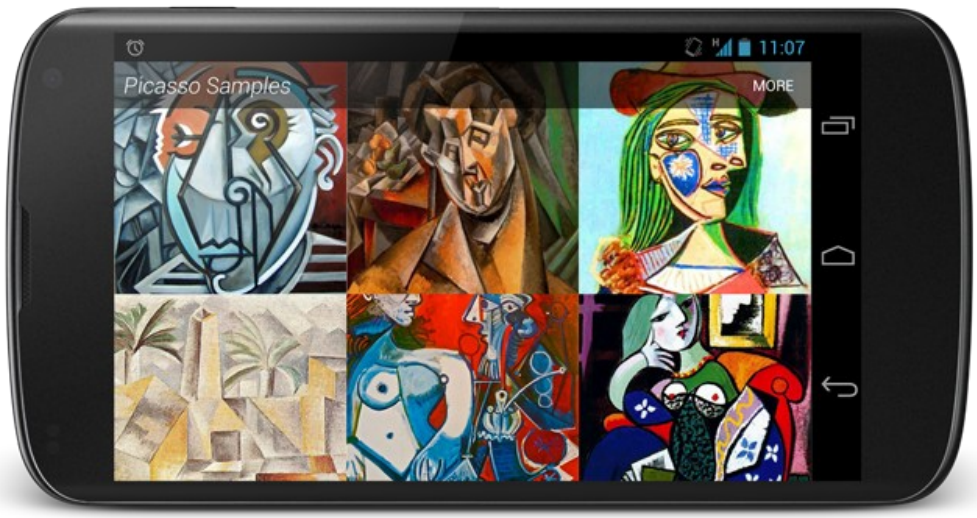

Рисунок 2.6 - програма збереження зображень Picasso

Android Picasso – це бібліотека для завантаження / обробки зображень, розроблена та підтримана компанією Square Inc. Вона надзвичайно популярна, оскільки часто вимагає всього лише одного рядка коду і має аналогічний стиль кодування для кожної з його функцій (ми скоро їх реалізуємо!) . Щоб використовувати бібліотеку Picasa в проекті Android Studio, додайте залежність у ваш файл build.gradle.

3 ПРОЕКТУВАННЯ

3.1 Розробка графічного інтерфейсу

Графі́чний інтерфе́йс кори́стувача́ (ГІК, англ. GUI, Graphical user interface) – тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

Користувач має бачити перед собою інтуїтивно зрозумілий при цьому ненав'язливий інтерфейс, який не викликатиме проблем або банального дратування у повсякденному використанні (рис. 3.1).

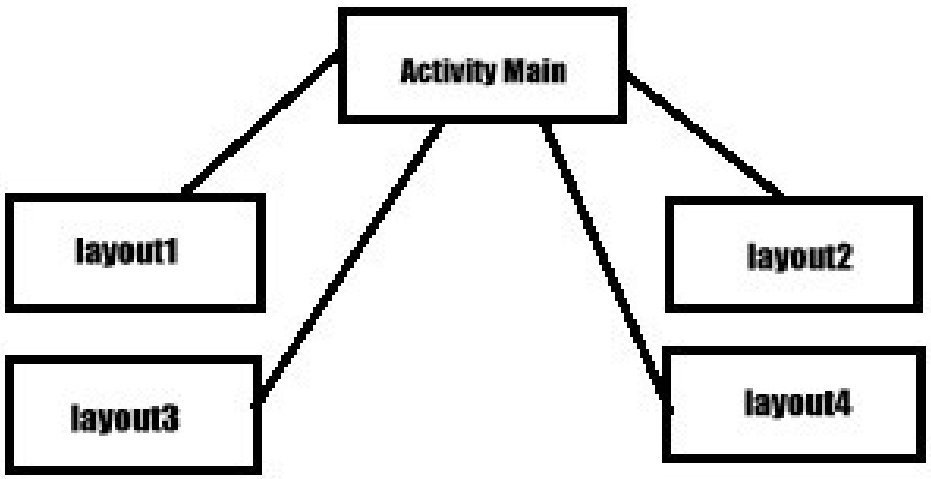

Рисунок 3.1 – Структура графічного інтерфейсу

Кожен мобільний додаток для успішного виконання ПЗ потрібен мати UI (інтерфейс користувача).

Інтерфе́йс користувача́ (англ. user interface, UI, дружній інтерфейс) – засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача; у графічних системах інтерфейс користувача,

втілюється багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштуваннями як самих вікон, так і окремих їх елементів (файли, теки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Завдяки зрозумілому і простому меню, що знаходиться на у головному макеті екрана, користувач відразу знайде цікавлячий його розділ та не витрачатиме час на пошук та налаштування.

В Android Studio для розробки власного дизайну використовуються макети.

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

- Оголошення елементів призначеного для користувача інтерфейсу в XML. В Android є зручний довідник XML-елементів для класів View і їх підкласів, наприклад таких, які використовуються для віджетів і макетів.
- Створення екземплярів елементів під час виконання. Ваша програма може програмним чином створювати об'єкти View і ViewGroup (а також керувати їх властивостями).

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

Перевага оголошення призначеного для користувача інтерфейсу в файлі XML полягає в тому, що таким чином ви можете більш ефективно відокремити уявлення свого додатку від коду, який управляє його поведінкою. Описи призначеного для користувача інтерфейсу знаходяться за межами коду вашої програми. Це означає, що ви можете змінювати або адаптувати інтерфейс без необхідності вносити правки в вихідний код і повторно компілювати його. Наприклад, можна створити різні файли XML макета для екранів різних розмірів і різних орієнтацій екрану, а також для різних мов. Крім того, оголошення макета в XML спрощує візуалізацію структури призначеного для користувача інтерфейсу, завдяки чому налагодження проблем також стає простіше. У даній статті ми навчимо вас

оголошувати макет в XML. Якщо ви віддаєте перевагу створювати екземпляри об'єктів View під час виконання, зверніться до довідкової документації для класів ViewGroup і View.

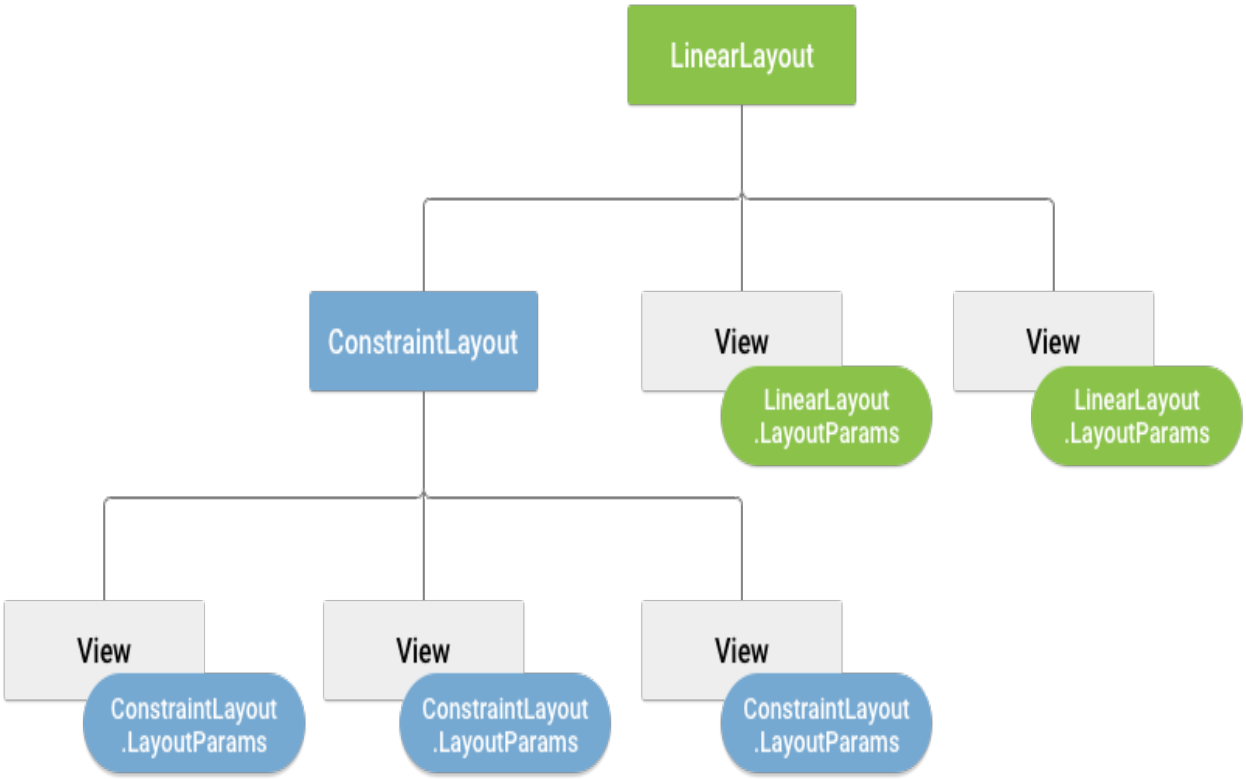

Рисунок 3.2 – Графічне уявлення ієрархії уявлення
з параметрами макета кожного уявлення

Зверніть увагу, що підклас LayoutParams має власний синтаксис для завдання значень. Кожен дочірній елемент повинен визначати LayoutParams, які підходять для його батьківського елемента, тоді як він сам може визначати інші LayoutParams для своїх дочірніх елементів.

Всі групи уявлень включають в себе параметри ширини і висоти (layout_width і layout_height), і кожне подання має визначати їх. Багато LayoutParams також включають додаткові параметри полів і кордонів.

Положення макету. Подання має прямокутну форму. Розташування подання визначається його координатами зліва і зверху, а його розміри - параметрами ширини і висоти. Розташування вимірюється в пікселах.

Розташування уявлення можна отримати шляхом виклику методів getLeft () і getTop (). Перший повертає координату зліва (по осі X) для прямокутника уявлення. Другий повертає верхню координату (по осі Y) для прямокутника уявлення. Обидва ці методи повертають розташування точки

зору щодо його батьківського елементу. Наприклад, коли метод getLeft () повертає 20, це означає, що подання знаходиться на відстані 20 пікселів від лівого краю його безпосереднього батьківського елемента.

Крім того, є кілька зручних методів (getRight () і getBottom ()), які дозволяють уникнути зайвих обчислень. Ці методи повертають координати правого і нижнього країв прямокутника уявлення. Наприклад, виклик методу getRight () аналогічне наступному обчисленню: getLeft () + getWidth ().

Розмір уявлення виражається його шириною і висотою. Фактично, уявлення володіє двома парами значень «ширина-висота».

Перша пара – це виміряна ширина і виміряна висота. Ці розміри визначають розмір уявлення в межах свого батьківського елемента. Виміряні розміри можна отримати, викликавши методи getMeasuredWidth () igetMeasuredHeight ().

Друга пара значень – це просто ширина і висота (іноді вони називаються креслярська ширина і креслярська висота). Ці розміри визначають фактичний розмір уявлення на екрані після розмітки під час їх відтворення. Ці значення можуть відрізнятися від виміряних ширини і висоти, хоча це і не обов'язково. Значення ширини і висоти можна отримати, викликавши методи getWidth () і getHeight ().

При вимірі своїх розмірів уявлення враховує заповнення. Відступ виражається в пікселах для лівої, верхній, правій і нижньої частин уявлення. Відступ можна використовувати для зсуву вмісту уявлення на певну кількість пікселів. Наприклад, значення відступу зліва, рівне 2, призведе до того, що вміст уявлення буде зміщено на 2 пікселя вправо від лівого краю уявлення. Для завдання відступів можна використовувати метод setPadding (int, int, int, int). Щоб запросити відступ, використовуються методи getPaddingLeft (), getPaddingTop (), getPaddingRight () і getPaddingBottom ().

Навіть якщо уявлення може визначити відступ, в ньому відсутня підтримка полів. Така можливість є у групи уявлень. Додаткові відомості представлені в довідці по об'єктах ViewGroup і ViewGroup.MarginLayoutParams.

Найпопулярніші види макетів:

- Лінійний макет - Макет, в якому дочірні елементи представлені в горизонтальних або вертикальних шпальтах. Якщо довжина вікна більше довжини екрану, в ньому створюється смуга прокрутки.

- Відносний макет - У цьому макеті можна задавати розташування дочірніх об'єктів відносно один одного (дочірній елемент А знаходиться зліва

від дочірнього елемента Б) або щодо батьківського об'єкта (вирівнювання щодо верхнього краю батьківського елементу).

- Уявлення веб-сторінки - відображення веб-сторінок;

3.2 Проектування програмних компонентів

Зваживши усі необхідні вимоги були виділені окремі компоненти:

– головна сторінка (відображає перелік усіх інших вкладок);

– розділ із розкладом занять;

– розділ із переліком викладачів;

– розділ новин;

– розділ з інформацією про дозвілля учнів;

– журнал відміток;

Для реалізації проекту використовувалися різні макети екранів у різних конфігураціях.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1 Створення макетів та компонентів екрану.

У самому початку роботи з Android Studio необхідно визначитися із шаблоном попередньо встановлених вікон додатку та елементів інтерфейсу.

Нам потрібно обрати EmptyActivity (рис. 4.1):

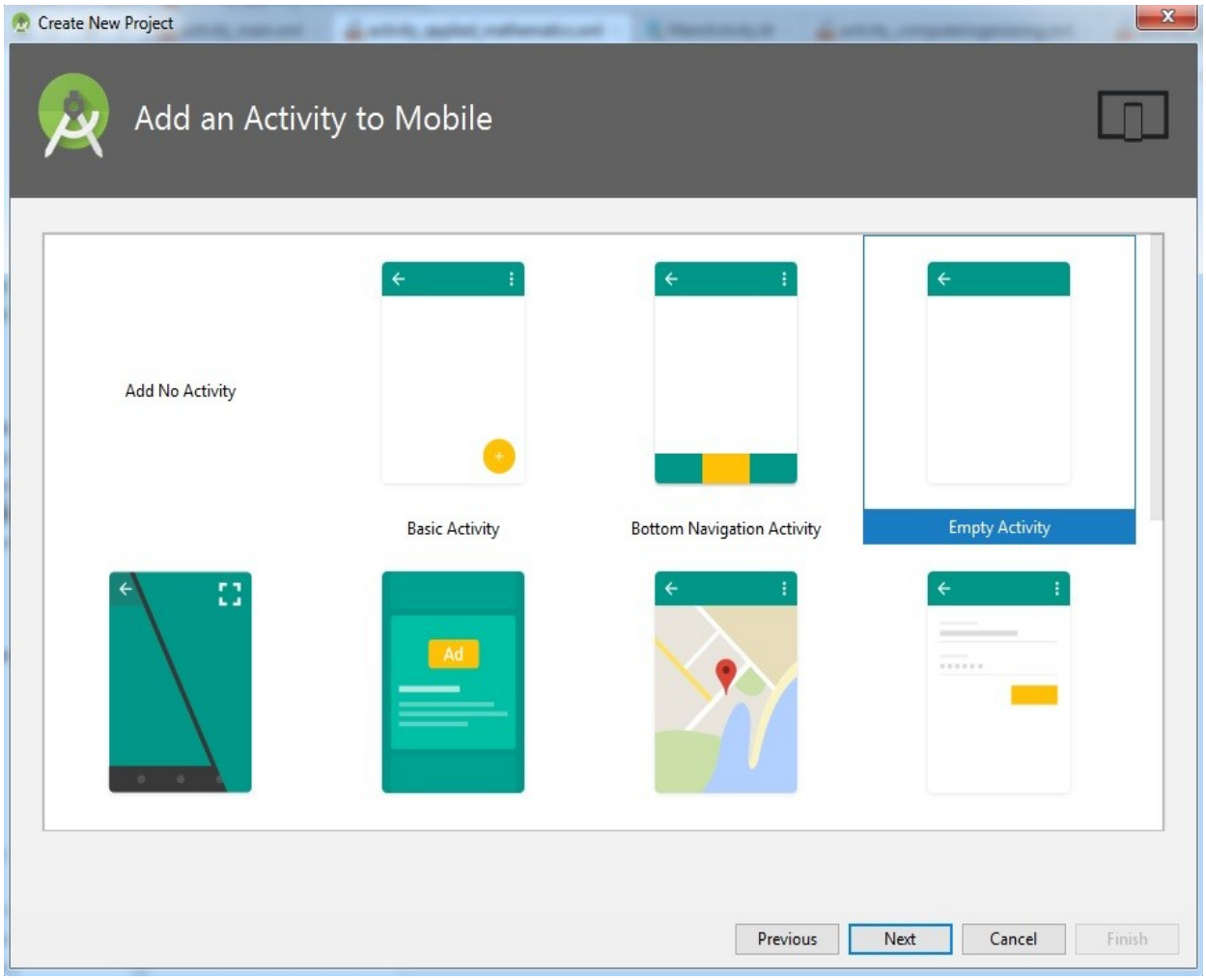

Рисунок 4.1 – Встановлення першого Activity

Першим завжди створюється головний екран - початкова сторінка майбутнього додатку.

Далі треба провести деякі налаштування компонентів програми, щоб можна було долучитися до повної розробки додатку.

Спочатку знайдемо файли типу .gradle - у цих файлах знаходиться уся необхідна інформація стосовно версій бібліотек та компонентів, які будуть використовуватися цим додатком (рис.4.2).

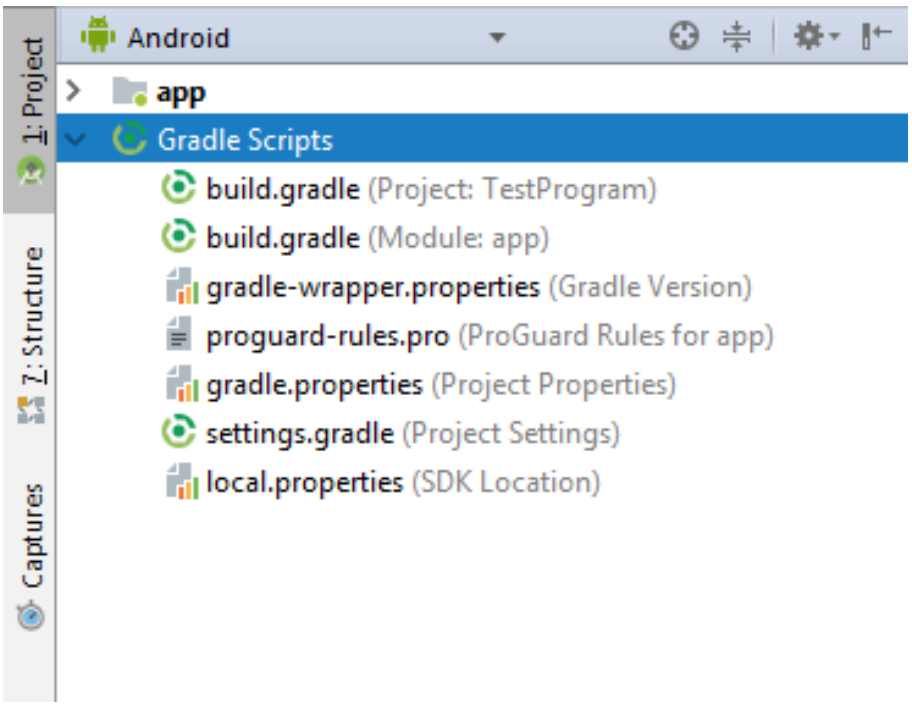

Рисунок 4.2 – Структура файлів Gradle в Android Studio

Файли .gradle являють собою конфігураційною конструкцію, тож усі зміни треба вносити обережно, бо пошкодження файлу призведе до неможливості побудови та запуску усього додатку. По-перше переглянемо файл build.gradle(Module app) – він містить у собі налаштування та залежності, що потрібні для коректної роботи системи.

Насамперед, додамо кілька необхідних для розробки компонентів.

Система збірки Gradle використовує плагіни для розширення своєї основної функціональності. Плагін – це розширення для Gradle, який зазвичай додає деякі попередньо задані завдання. Знищуйте кораблі з рядом плагінів, і ви можете створювати власні плагіни. Наприклад, додаток застосування 'com.android.application' робить плагін Android доступним для створення градації. Для мобільного додатку підключемо наступні плагіни :

– apply plugin: 'com.android.application'

– apply plugin: 'kotlin-android'

Вони надають змогу будувати проект, використовуючи технології Android та Kotlin (рис. 4.3).

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

android {
    compileSdkVersion 25
    buildToolsVersion '27.0.3'
    defaultConfig {
        applicationId "com.example.user.testprogram"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    api fileTree(dir: 'libs', include: ['*.jar'])
    androidTestImplementation('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    //noinspection GradleDynamicVersion
    implementation 'com.android.support:appcompat-v7:25.3.1+'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.squareup.picasso:picasso:2.5.2'
    implementation 'com.android.support:support-v4:25.3.1'
    testImplementation 'junit:junit:4.12'
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
}
repositories {
    mavenCentral()
}
```

Рисунок 4.3 – Структура файлу build.gradle

4.1.1 Головна сторінка.

На головній сторінці проекту міститься меню, усього головна сторінка містить вісім елементів макету екрана: ImageView та сім елементів Bottom (рис. 4.4).

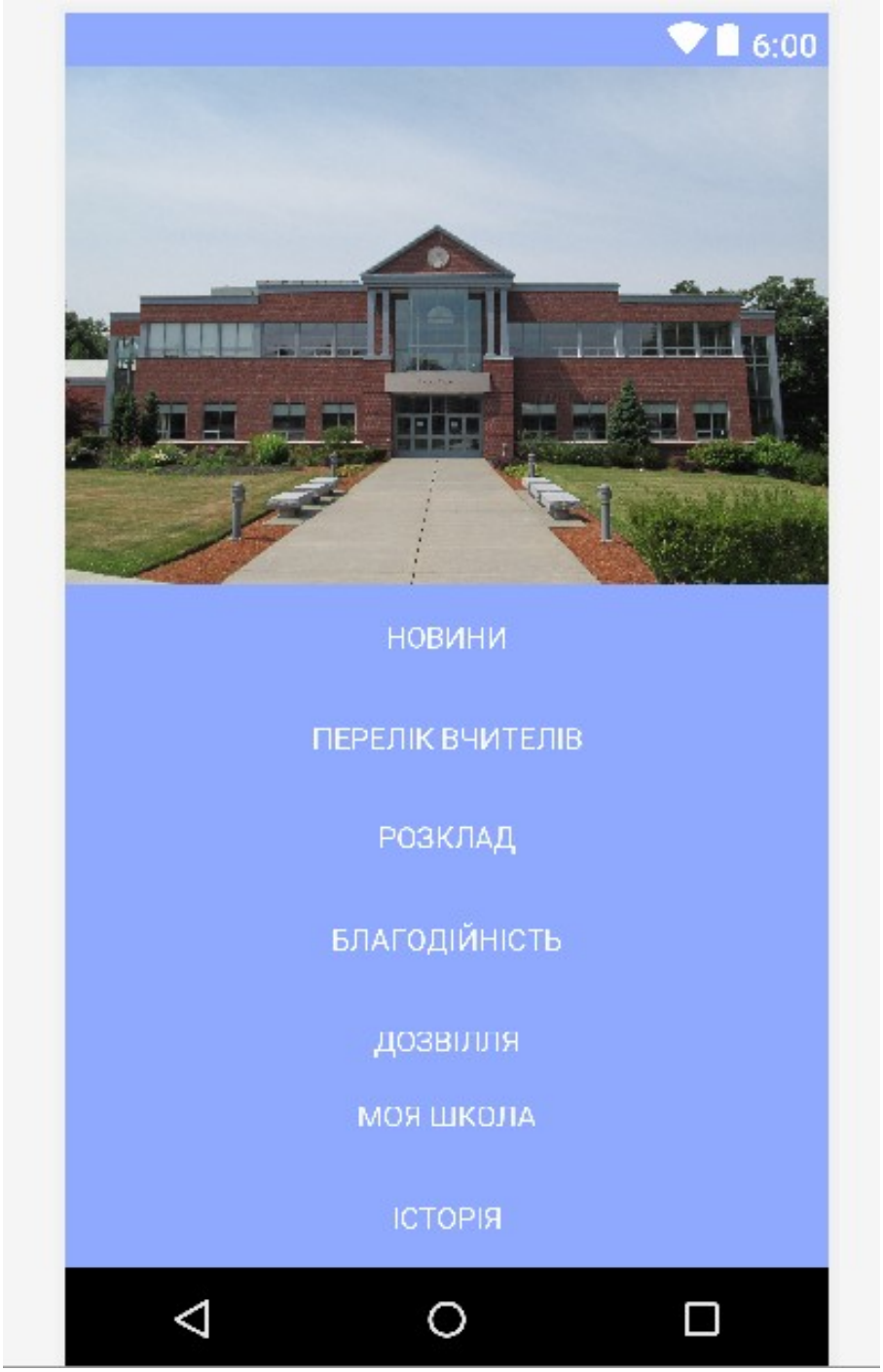

Рисунок 4.4 – Макет головної сторінки екрану

Як видно на рисунку, екран містить картинку та сім кнопок для взаємодії з іншими макетами екранів. Головна сторінка відразу відображає меню програми, тож ви починаєте роботу тільки-но запустивши додаток.

Меню містить сім пунктів:

1) Новини – містять інформацію стосовно останніх новин з життя школи(інформація про культурно-масові заходи, конференції та ін.) (рис. 4.5);

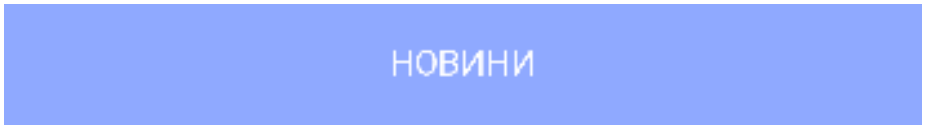

Рисунок 4.5 – Розділ Новини

2) Перелік вчителів – надає контактні дані усіх викладачів (рис. 4.6);

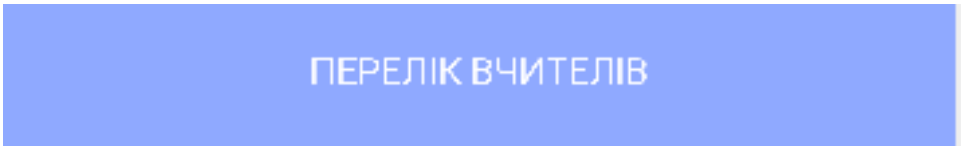

Рисунок 4.6 – Розділ Перелік вчителів

3) Розклад – в цьому розділі у вигляді таблиці відображено розклад занять (рис. 4.7);

Рисунок 4.7 – Розділ Розклад

4) Благодійність – сторінка, що містить посилання до благодійних фондів (рис. 4.8);

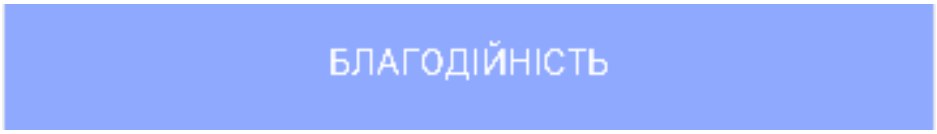

Рисунок 4.8 – Розділ Благодійність

5) Дозвілля – цей пункт включає у себе інформацію стосовно усіляких образотворчих гуртків та спортивних секцій;

6) Моя школа – персональна сторінка із примітками(правками до розкладу та ін);

7) Історія – відображає інформацію про історію школи;

Щоб додати на екран зображення, необхідно просто знайти вкладку Pallete у preview(передогляд) та додати потрібний елемент(ImageView) на екран - середа розробки згенерує необхідний xml-код (рис. 4.9).

```
<ImageView
    android:id="@+id/iv_image_from_url"
    android:layout_width="wrap_content"
    android:layout_height="245dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="0dp"
    android:scaleType="centerCrop"
    android:src="@drawable/highschoolmem" />

    <Button
        android:id="@+id/ComputerIngeniering"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:background="@color/colorPrimaryDark"
        android:backgroundTint="@color/colorPrimaryDark"
        android:fontFamily="Arial"
        android:text="ІСТОРІЯ"
        android:textColor="@android:color/background_light"
        android:textSize="14sp" />
```

Рисунок 4.9 – XML зображення елементів екрану

Основні параметри елементу ImageView:

– android: layout_width – задає ширину елемента;

– android: layout_heigh – задає висоту елемента;

– android: scaleType – встановлює подоження використаного зображення;

– android: src – задає шлях до ресурсу(зображення);

4.1.2 Розділ "Історія"

Історія на сьогоднішній день (принаймні в Україні) є однією з головних суспільнозначущих наук (рис. 4.10).

Рисунок 4.10 – Розділ "Історія"

Адже, “українське суспільство не живе сьогоденням, так само як і не живе майбутнім. Українське суспільство живе минулим”. Саме ця теза спонукає до написання окремої сторінки з історією школи.

Цей розділ містить інформацію про історію школи та загальні положення щодо поведінки та навчального процесу, яки склалися за роки існування самої школи.

4.1.3 Розділ "Вчителі"

Цей розділ включає у себе вичерпну інформацію стосовно викладачів, а також надає змогу при необхідності відразу зв'язатися із потрібним викладачем телефоном (рис. 4.11).

Рисунок 4.11 – Макет екрану "Вчителі"(ч.1)

У верхній частині екрану знаходиться ім'я вчителя, вказана дисципліна, яку він викладає, кабінет а також його фото.

Нижче приведений код лістингу даної вкладки:

```
        <?xml version="1.0" encoding="utf-8"?>
<ScrollView android:layout_height="4500dp"
   android:layout_width = "match_parent"
   xmlns:android="http://schemas.android.com/apk/res/android" >

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="6950dp"
   tools:context="com.example.user.testprogram.ControllerDeanery"
   android:orientation="vertical">

   <ImageView
      android:id="@+id/fICDeanery"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:src="@drawable/techer"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_above="@+id/textView10"
      android:layout_alignParentTop="true" />

   <TextView
      android:id="@+id/nameOfD"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Викладач з математики"
      android:layout_marginTop="234dp"
      android:layout_marginLeft="15dp"
      android:layout_marginStart="15dp"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />
```

Як видно з лістингу файлу, у блоці ImageView описуються параметри зображення, а у блоці TextView відповідно, описуються параметри теекстових полів (рис. 4.12).

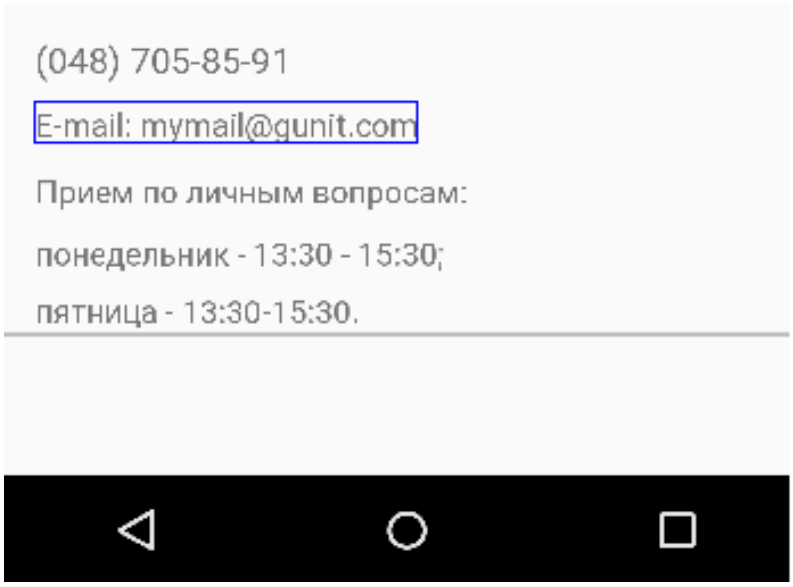

Рисунок 4.12 – Макет екрану "Вчителі"(ч.2)

У нижній частині екрану відображається уся контактна інформація: мобільний телефон вчителя, електронна пошта та часи прийому/консультацій. Нижче приведен лістинг коду, який підключає функцію дзвінків вчителю.

```
        callTeacher.setOnClickListener(View.OnClickListener {
    if (ActivityCompat.checkSelfPermission(this@Teachers,
Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                          int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return@OnClickListener
    }
    val callIntent = Intent(Intent.ACTION_CALL)
    callIntent.data = Uri.parse("tel:0487058379")
    startActivity(callIntent)
})
val privateMessage = findViewById(R.id.privateMessage) as Button
privateMessage.setOnClickListener { Toast.makeText(this@Teachers, "К сожалению
личные сообщения недоступны", Toast.LENGTH_SHORT).show() }
val takeSlider = findViewById(R.id.takeSlider) as Button
takeSlider.setOnClickListener {
```

```
    Toast.makeText(this@Teachers, "Пока что данная функция невозможна :(",
Toast.LENGTH_SHORT).show()
```

4.1.4 Розділ "Дозвілля"

Жоден з проаналізованих аналогів не надавав користувачам інформації, яка б допомогала планувати своє дозвілля із користю для здоров'я. Шкільні гуртки та секції мають не аби-який вплив на формування майбутньої особистості дитини (рис. 4.13).

Рисунок 4.13 – Сторінка "Дозвілля"

Даний розділ містить наступні пункти:

1) "Театральний гурток" – надає інформацію стосовно розкладу занять з акторської майстерності та репетування постановок (рис. 4.14);

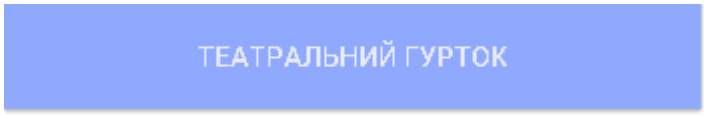

Рисунок 4.14 – Кнопка "Театральний гурток"

2) "Секція з баскетболу" – дозволяє обрати оптимальний розклад тренувань з баскетболу (рис. 4.15);

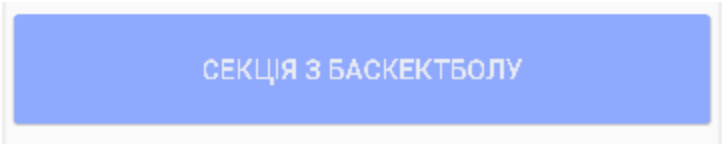

Рисунок 4.15 – Кнопка "Секція з баскетболу"

3) "Секція з футболу" – надає інформацію про правила тренувань а також час їх проведення (рис. 4.16) .

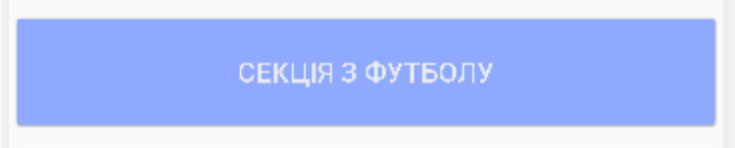

Рисунок 4.16 – Кнопка "Секція з футболу"

4) "Літній табір" – розділ, що допоможе сконструювати дитині її літній відпочинок за власним смаком та розсудом (рис. 4.17).

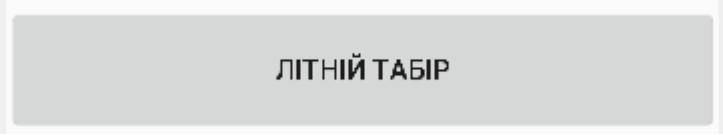

Рисунок 4.17 – Кнопка "Літній табір"

ВИСНОВКИ

У результаті виконання даної дипломної роботи був розроблений мобільний додаток для самоконтролю навчання з використанням технологій Gradle, Kotlin, XML.

При проектуванні мобільного додатку були застосовані всі можливості для того щоб зробити користування ним максимально зручним та в одночас простим та інтуїтивно зрозумілим.

Дизайн мобільного додатку створений у мінімалістичному стилі та з інтуїтивно зрозумілим інтерфейсом з урахуванням усіх можливостей та особливостей для швидкого користування.

Додаток був розроблений відповідно до вимог, які були висунуті, розширення є досить простим як в архітектурі, так і в користуванні.

Усі технології, що були використані у розробці, відповідають новітнім стандартам у розробці ПО.

Додаток має стати у нагоді перш за все дітям, щоб зробити їх навчання у школі більш інтерактивним.

ПЕРЕЛІК ПОСИЛАНЬ

1. Berglund, Tim; McCullough, Matthew (July 2011). Building and Testing with Gradle. Foreword by Hans Dockter (First ed.). O'Reilly Media. p. 116. ISBN 978-1-4493-0463-8.

2.Ikkink, Hubert (November 2012). Gradle Effective Implementation Guide (First ed.). Packt Publishing. p. 382. ISBN 978-1849518109.

3.Berglund, Tim; McCullough, Matthew (May 2013). Gradle DSLs (First ed.). O'Reilly Media. pp. 50 est. ISBN 978-1-4493-0467-6.

4.Muschko, Benjamin (Fall 2013). Gradle In Action (First ed.). Manning Publications. p. 390. ISBN 9781617291302.

ДОДАТОК А КОДИ ОСНОВНИХ ПРОГРАМНИХ МОДУЛІВ

```
package com.example.user.testprogram

import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.widget.Button
import android.widget.Toast

class AppliedMathematics : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_applied_mathematics)
        val firstkursAE = findViewById(R.id.aefirstkurs) as Button
        firstkursAE.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Прости но в 1 курсе АВ ничого нема", Toast.LENGTH_SHORT).show() }
        val secondkursAE = findViewById(R.id.aeseckurs) as Button
        secondkursAE.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Прости но во 2 курсе АВ таже история", Toast.LENGTH_SHORT).show() }
        val thirdkursAE = findViewById(R.id.aethirdkurs) as Button
        thirdkursAE.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Прости но на 3 курсе АВ таже история", Toast.LENGTH_SHORT).show() }
        val ae161 = findViewById(R.id.ae161) as Button
        ae161.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae162 = findViewById(R.id.ae162) as Button
        ae162.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae163 = findViewById(R.id.ae163) as Button
        ae163.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae153 = findViewById(R.id.ae153) as Button
        ae153.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae152 = findViewById(R.id.ae152) as Button
        ae152.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae151 = findViewById(R.id.ae151) as Button
        ae151.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae141 = findViewById(R.id.ae141) as Button
        ae141.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
```

```
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae142 = findViewById(R.id.ae142) as Button
        ae142.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae143 = findViewById(R.id.ae143) as Button
        ae143.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae131 = findViewById(R.id.ae131) as Button
        ae131.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae132 = findViewById(R.id.ae132) as Button
        ae132.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae133 = findViewById(R.id.ae133) as Button
        ae133.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae121 = findViewById(R.id.ae121) as Button
        ae121.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae122 = findViewById(R.id.ae122) as Button
        ae122.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val ae123 = findViewById(R.id.ae123) as Button
        ae123.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Находится на
стадии модернизации", Toast.LENGTH_SHORT).show() }
        val fivekurs = findViewById(R.id.fivekursae) as Button
        fivekurs.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Прости но на
5 курсе АВ ничого нема", Toast.LENGTH_SHORT).show() }

        val fourkurs = findViewById(R.id.fourkursae) as Button
        fourkurs.setOnClickListener { Toast.makeText(this@AppliedMathematics, "Прости но на
4 курсе АВ ничого нема", Toast.LENGTH_SHORT).show() }
        val lessonsAE = findViewById(R.id.lessoninAE) as Button
        lessonsAE.setOnClickListener { Toast.makeText(this@AppliedMathematics,
"Планируется в ближайшем будущем добавить", Toast.LENGTH_SHORT).show() }
        val teachersAE = findViewById(R.id.teachersinAE) as Button
        teachersAE.setOnClickListener { Toast.makeText(this@AppliedMathematics, "К
сожалению Преподавателей ещё не заполнил :(", Toast.LENGTH_SHORT).show() }
    }
}
```