Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

СУЧАСНІ ТЕХНОЛОГІЇ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

Текстова частина до курсової роботи
за спеціальністю «Комп’ютерні науки»

Керівник курсової роботи
к.ф-м.н., доц. Тригуб О.С.
(прізвище та ініціали)

_____
(підпис)
“__” _____ 2021 р.
Виконав студент Винник А.І.
“__” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
к.ф-м.н., доц.,
_____ Тригуб О.С.
(підпис)
“___” _____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу
студенту Виннику А.І. факультету інформатики 3 курсу
ТЕМА: Сучасні технології розробки мобільних додатків

Зміст ТЧ до курсової роботи:

1. Індивідуальне завдання
2. Вступ
3. Огляд сфери мобільних технологій
4. Опис наявних технологій розробки мобільних додатків
5. Розробка програми
6. Висновки
7. Список літератури
8. Додатки

Дата видачі “___” _____ 2021 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: Сучасні технології розробки мобільних додатків

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	02.11.2010	
2.	Огляд технічної літератури за темою роботи.	15.11.2010	
3.	Виконати аналіз сучасних методів …	25.11.2010	
3.	Розробка алгоритму …	30.12.2010	
4.	Програмування розробленого алгоритму	15.01.2011	
5.	Застосування розробленого алгоритму до …	15.02.2011	
6.	Виконання порівняльного аналізу результатів прогнозування, отриманих за допомогою розробленого алгоритму на основі нейромережі, та результатів, отриманих за допомогою регресійних моделей.	30.03.2011	
7.	Написання пояснювальної роботи.	20.04.2011	
8.	Створення слайдів для доповіді та написання доповіді.	22.04.2011	
8.	Аналіз отриманих результатів з керівником, написання доповіді та попередній захист магістерської роботи.	25.04.2011	
10.	Корегування роботи за результатами попереднього захисту.	1.05.2011	
11.	Остаточне оформлення пояснювальної роботи та слайдів.	3.05.2011	
12.	Захист курсової роботи	17.05.2011	

Студент _____

Керівник _____

“_____”_____

ЗМІСТ

Перелік прийнятих скорочень

ОЗУ – Оперативна пам’ять

WAP – Протокол бездротової передачі даних

GPRS – Сервіс пакетної радіопередачі

EDGE – Технологія бездротової передачі даних

XML – Розширювана мова розмітки

LLVM – Універсальна система аналізу, трансформації і оптимізації програм

HTML – Мова розмітки документів

CSS – Каскадні таблиці стилів

JIT – Технологія трансляції байт-коду в машинний код

IDE – Інтегроване середовище розробки

SQL - Мова програмування для взаємодії з базами даних.

Вступ

Перші мобільні пристрої, їх використання не було можливе без підключення до електромережі, оскільки вони йшли у комплекті з проводом, а висока вартість такого пристрою сильно обмежувала його розповсюдження серед населення.

З тих пір мобільні технології зробили великий крок вперед, проте якісний ріст прийшовся на останнє десятиріччя. Сьогодні мобільні пристрої стали компактні, зручні у використанні, їх продуктивність значно виросла і тому їх роль у повсякденному житті значно виросла.

Популярними напрямами розробки мобільних застосунків на даний момент є сфера розважальних програм, а також соціальні мережі. Не менш корисними являються додатки для зберігання інформації, а також різноманітні програми-утиліти.

Сьогодні головною метою для розробника мобільного додатку є скорочення тривалості життєвого процесу розробки і зменшення різниці в часі між формуванням ідеї програми та її кінцевим запуском. На сьогоднішній день розробники цільових мобільних операційних систем (Android, iOS) створюють усі необхідні умови щоб програміст витрачав менше часу та зусиль для створення застосунку.

Від вибору операційної системи для вашого додатку також залежить вибір середовища розробки. Наприклад, можна використовувати програму, яку надає розробник вашої платформи або використовувати сторонні інструменти для створення додатку одразу на декілька операційних систем.

У розробника мобільних додатків є певні задачі під час створення програмного забезпечення, які мають відношення до кожної з існуючих операційних систем. До таких задач можна віднести:

- зменшення розміру файлу програми
- адаптація під різні розміри дисплею
- зменшення витрат живлення батареї мобільного пристрою

- оптимізація програми для зменшення навантаження на процесор
- зменшення використання ОЗУ
- забезпечення захисту даних користувача

Таким чином зі спрощенням процесу розробки, удосконаленням сучасних середовищ розробника та оптимізацією написання мобільних застосунків також зростають і вимоги до самих додатків. Кожного року програми стають більш оптимізованими, безпечними та зручнішими для кінцевого користувача.

Перший розділ присвячено аналізу ринку мобільних додатків та їх розвиток.

У другому розділі було описано сучасні технології розробки мобільних додатків на основі операційних систем Android та iOS. А також був створений застосунок на платформі Android.

Постановка задачі

1. Аналіз розвитку ринку мобільних додатків та аналіз сучасних методів та технологій розробки.

2. Навести приклад сучасних засобів розробки мобільних додатків на різних мобільних операційних системах.

3. Розробити мобільний застосунок використовуючи сучасні технології розробки.

1. РОЗВИТОК СФЕРИ РОЗРОБКИ МОБІЛЬНИХ ТЕХНОЛОГІЙ

1.1. Історія розвитку мобільних технологій

Мобільний застосунок – це комп'ютерна програма, яка призначена спеціально для роботи у мобільному пристрої, смартфоні чи планшеті, що була створена для виконання певної поставленої задачі. Найперші мобільні пристрої майже не мали додатків, а ті що були поставлялися з самим телефоном. Фактично вони створювались тільки з ціллю комунікації з людьми. Тобто перші мобільні пристрої могли тільки телефонувати один одному, а перша програма це була телефонна книга з контактами.

Перші мобільні застосунки, які могли бути встановлені поверх системних програм на перших телефонах були розроблені в першій половині 1990-их років. Це стало відбуватись коли стільниковий зв'язок почав з'являтися у повсякденному житті мільйонів людей. Разом з створенням стільникового зв'язку технології WAP, котра дозволяла отримати доступ до мережі інтернет з телефону почала зростати кількість мобільних застосунків[1]. Це стало дуже популярним та зручним, оскільки раніше для встановлення якогось стороннього додатку потрібно було мати спеціальний кабель.

На початку XXI століття почався сильний розвиток в сфері мобільних телефонів та мобільних додатків. Завдяки появі таких технологій стільникового зв'язку як GPRS та EDGE, які зробили інтернет трафік більш доступним, з'являється більше можливостей для розвитку мобільних технологій.
У цей період на ринку почали з'являтися смартфони та комунікатори, які стали відтісняти звичайні мобільні телефони. Смартфони та комунікатори мали ширший спектр використання оскільки мали більшу потужність аніж прості телефони. Вони мали розвинені операційні системи (Android, iOS, BlackBerry OS, Symbian OS), які були відкриті для розробки програмного забезпечення під них від сторонніх розробників.

Починаючи з 2000-го року розпочався дуже стрімкий зріст мобільних технологій. У 2001 був випущений перший телефон з підтримкою Java. Через

рік після цього була розроблена технологія Bluetooth, яка є у кожному сучасному телефоні.

Проте справжній переворот у сфері мобільних пристроїв стався у 2007 році. Саме того року, 9 квітня на виставці Macworld Conference & Expo, компанією Apple був випущений перший iPhone. У тому ж році був показаний перший смартфон на базі операційної системи Android. Проте це була тільки бета-версія пакету для розробників програмного забезпечення і мав Android – емулятор.

Телефони на операційній системі Android почали активно розвиватись тільки у 2009 році. Це сталося після виявлення та виправлення ряду помилок та додавання нового функціоналу у систему. Тоді виробники мобільних телефонів почали активно використовувати цю платформу для своїх смартфонів.

Розвиток мобільних операційних систем призвів до появи можливості розвитку в сфері мобільних застосунків. Мобільні платформи надавали розробникам широкий вибір можливих технологій для створення додатків на їх основі. Такий стрімкий розвиток відбувався і відбувається завдяки конкуренції двох операційних систем (Android та iOS) у мобільній сфері.

З 2010 року ринок мобільних телефонів обігнав ринок персональних комп’ютерів. Стало чітко зрозуміло, що з ринком мобільних пристроїв уже не можливо конкурувати. Смартфон зараз є у майже у кожної людини на планеті, він використовується у багатьох сферах життя.

За результатами крупної мобільної аналітики з 2015 року користувачі все менше стали користуватися веб-сайтами та все більше користуються мобільними додатками. З цього можна зробити висновок що сфера мобільних технологій має великий потенціал для удосконалення та розвитку.

1.2. Сучасні середовища розробки мобільних додатків

1.2.1. Android Studio

Android Studio – це інтегроване середовище розробки для операційної системи Android. Вперше дана програма була представлена у 2013 році. Тоді вийшла перша версія 0.1 даного продукту. Далі аж до версії 0.8 середовище було у режимі бета-тестування. Проте перша стабільна версія 1.0 вийшла у кінці 2014 року і стала повноцінною заміною розширенню для Eclipse під назвою Android Development Tools. Сьогодні це основне середовище для розробки застосунків на платформі Android[2]. Дана програма доступна для тих операційних систем як Windows, MacOS а також Linux.

Android Studio було розроблено на базі програмного забезпечення IntelliJ IDEA, яке було створено для розробки програм на мовах програмування Java, C++ та Kotlin. Таким чином Android Studio також підтримує ці мови. [2]

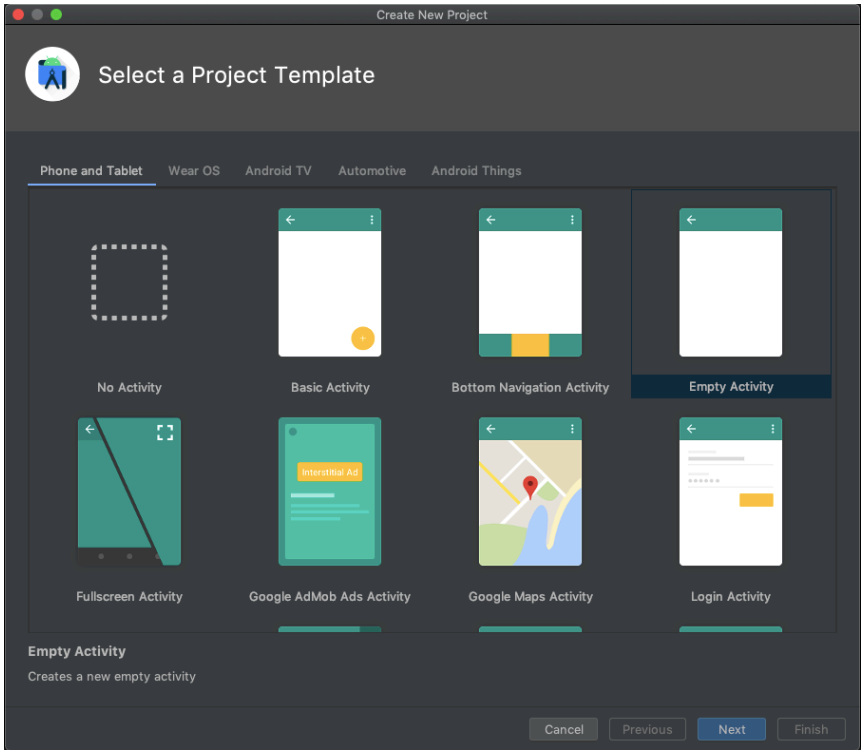


Рисунок 1.1 – Створення нового проекту Android Studio

Android Studio надає велику кількість початкових шаблонів під час створення проекту. У розробника є можливість обрати вже заготовлені макети для застосунку. На рисунку 1.1 можна побачити, що у середовищі Android Studio можна створити програмне забезпечення на такі пристрої як: смартфон, планшет, наручні годинники які мають операційну систему Wear OS, телевізори з підтримкою Android, автомобільні програвачі та платформи Android Things.

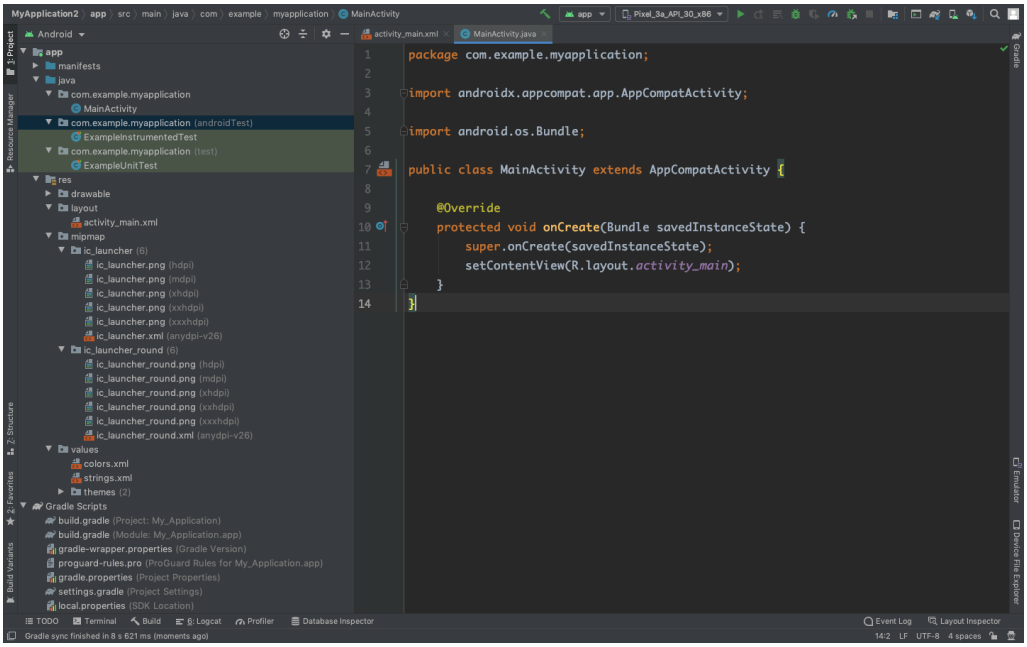


Рисунок 1.2 – інтерфейс Android Studio

Android Studio базується на системі автоматичного збирання Gradle. Ця система була розроблена для збірок великих проектів, які можуть розширятися і може визначати які компоненти проекту були змінені і потребують нової збірки, а які не потрібно перезапускати.

Android Studio має досить широкий редактор графічних компонентів з якими можна взаємодіяти перетягуючи їх у потрібне місце. Таким чином розробник може не прописувати кожен графічний елемент у XML редакторі.

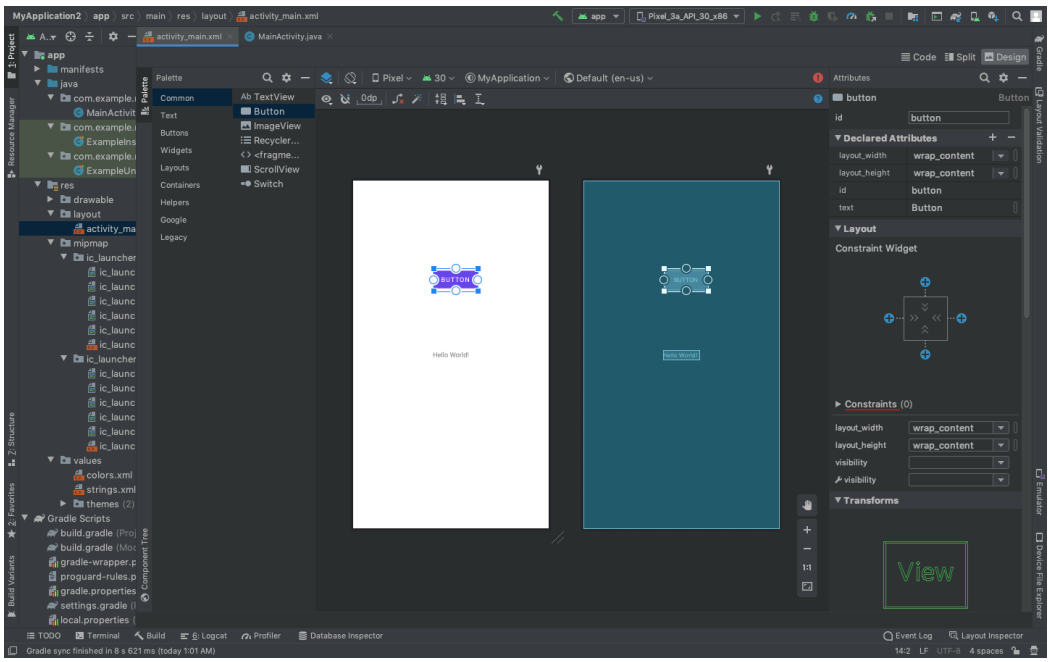


Рисунок 1.3 Редактор макету в Android Studio

За допомогою даного редактора розробник може розмістити потрібні йому графічні елементи на екрані смартфона, редагувати їх зовнішній вигляд, а також їх розташування відносно інших елементів. Редактор містить досить широкий арсенал різноманітних графічних елементів, які можуть знадобитися у процесі розробки мобільного додатка. Окрім цього є можливість тестувати розміщення елементів на різних розмірах екрану. Таким чином розробник може налаштувати зовнішній вигляд його додатку для більшості пристроїв.

З оновленням 2.2 у Android Studio з’явився новий спосіб розташування елементів на макеті яка має назву Constraint Layout[2]. Це новий спосіб розташування графічних елементів та зміни розмірів цих елементів досить гнучким способом.

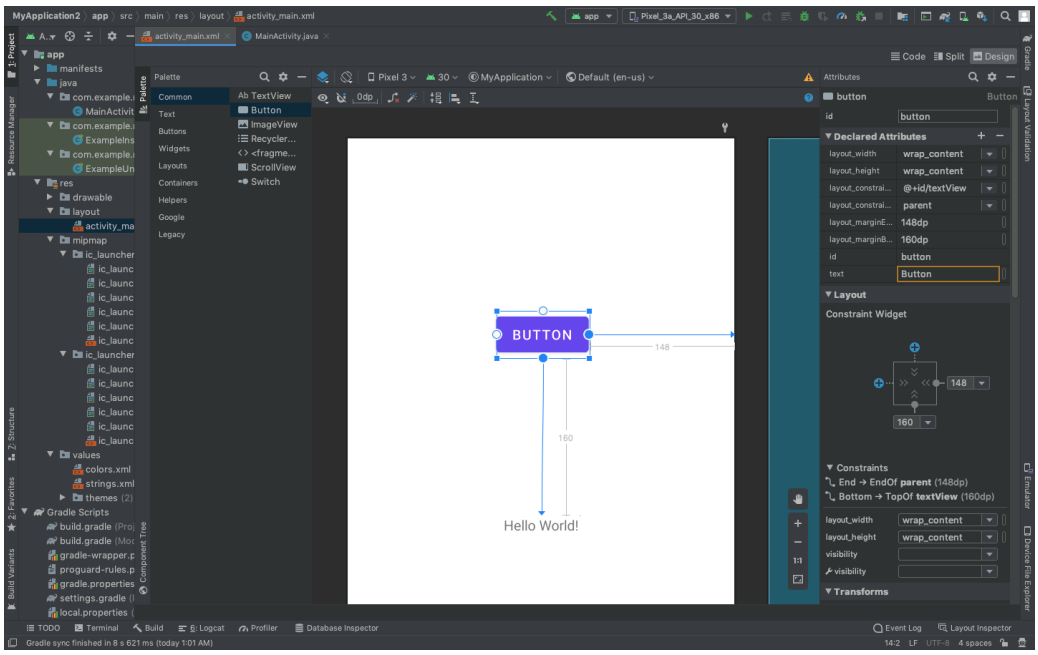


Рисунок 1.4 Використання Constraint Layout у Android Studio

На рисунку 1.4 можна побачити, що за його допомогою кнопка була розміщена відносно верхньої точки сусіднього текстового елемента вище на 160 пікселів та відносно правого краю батьківського елемента лівіше на 148 пікселів.

Проте якщо потрібні більш гнучкі налаштування інтерфейсу, Android Studio надає можливість змінювати макет за допомогою XML редактора.

Окрім можливостей, які були описані вище, Android Studio має ряд інших не менш важливих та зручних можливостей :

- розвинутий редактор коду, який має можливість автоматично заповнювати код, підсвічування помилок та їх швидке виправлення;
- вбудований емулятор для запуску додатку в реальному часі а також його налагодження;
- аналізатор коду (Lint), який знаходить проблеми у продуктивності програми, не відповідність версії та надає підказки щодо вирішення інших проблем;
- тестування програми за допомогою Unit тестів;
- вбудована можливість контролю версій за допомогою GitHub.

1.2.2. Xcode

Xcode – інтегроване середовище розробки для розробки системних продуктів таких операційних систем як: iOS, macOS, watchOS і tvOS. Перша версія даної програми була випущена у 2003 році. Це основне середовище розробки в екосистемі Apple.[3] Доступний для встановлення тільки на операційній системі macOS. Має підтримку для написання програм на мовах Java, C, C++, Objective-C, Swift, Python, AppleScript та Ruby.

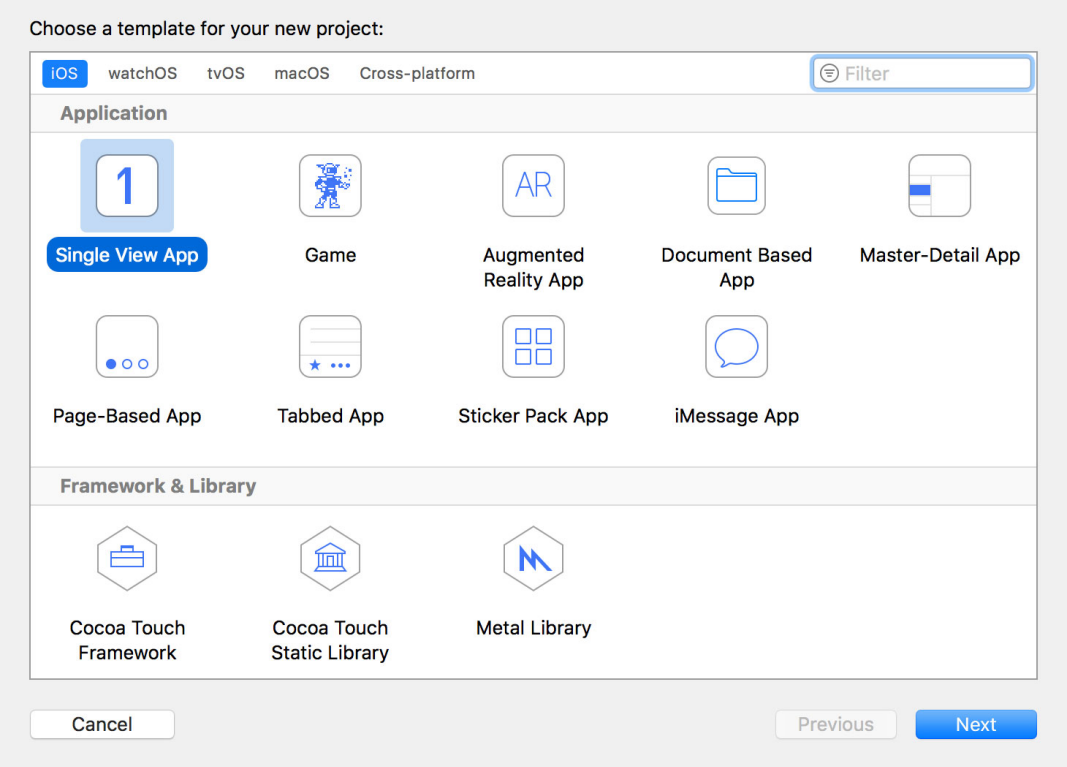


Рисунок 1.5 – створення нового проекту в Xcode.

При створенні проекту Xcode надає широкий вибір шаблонів для різних типів операційних систем. Окрім цього можна обрати різноманітні фреймворки та бібліотеки для розробки.

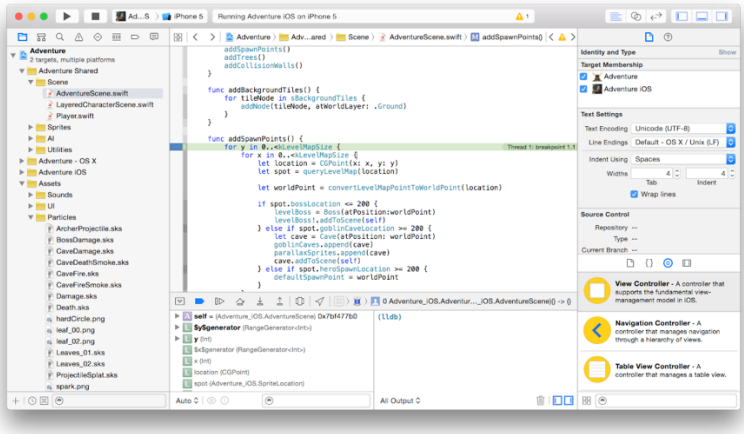


Рисунок 1.6 – інтерфейс Xcode[3]

Xcode має досить зручний інтерфейс та професійний редактор код з функцією автоматичного заповнювання коду, підсвічуванням синтаксису, підказками для оптимізації коду. Також Xcode негайно повідомить розробнику якщо він зробить помилку і надасть можливість швидко виправити цю помилку за допомогою декількох клавіш.

У даному середовищі розробки наявний досить потужний LLVM[3] компілятор з відкритим кодом. Він дозволяє швидко компілювати програми написані на мовах програмування C, C++, Objective-C. Таким чином код оптимізується для створення надзвичайно швидких додатків, спеціально налаштованих для процесорів iPhone, iPad, Mac.

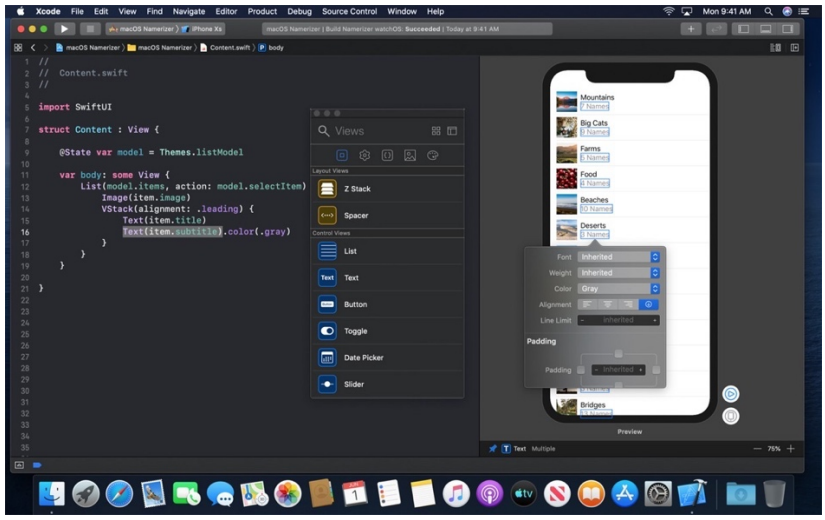


Рисунок 1.7 – редактор макету в Xcode[3]

Xcode має інтуїтивний редактор макету який сильно спрощує створювання інтерфейсу додатку. Під час роботи з дизайнерським полотном, все що редагується на ньому автоматично синхронізується з кодом у суміжному редакторі. Код миттєво видно у вигляді попереднього перегляду і тому будь які зміни зроблені у редакторі будуть одразу видні у коді. Xcode одразу перекомпілює зміни і вставляє їх у поточну версію програми. Використовуючи даний редактор можна розташовувати елементи просто перетягуючи потрібні графічні елементи на полотно. Цьому елементу можна одразу змінювати розмір, колір, розташування та ще багато інших опцій. Також у даному середовищі надається можливість адаптувати зовнішній вигляд застосунку під різні розміри пристроїв та залежно від їх орієнтації (горизонтальна чи вертикальна).

Компілятор Swift повністю вбудований в Xcode, тому під час розробки додаток постійно запускається та знаходиться у роботі. Таким чином програміст має можливість розроблювати програму та бачити результат у реальному часі.[3]

Також Xcode має низку різноманітних вбудованих можливостей для полегшення розробки додатків, в їх числі:

- містить вбудовану систему контролю версій інтегровану з такими веб-сервісами як GitHub, Bitbucket та Perforce;
- вбудована документація Apple з можливістю пошуку інформації по ній;

1.2.3. Flutter

Flutter – це набір інструментів розроблений компанією Google для побудови крос-платформних мобільних програм, веб сайтів а також програм для персональних комп’ютерів з одної сукупності коду. Даний продукт з’явився у травні 2017 року.[4] Підтримує розробку для таких операційних систем як: Android, iOS, Linux, Mac, Google Fuchsia а також для веб – застосунків.

Програми за допомогою Flutter можна писати у Android Studio, Xcode та Microsoft Visual Studio Code.

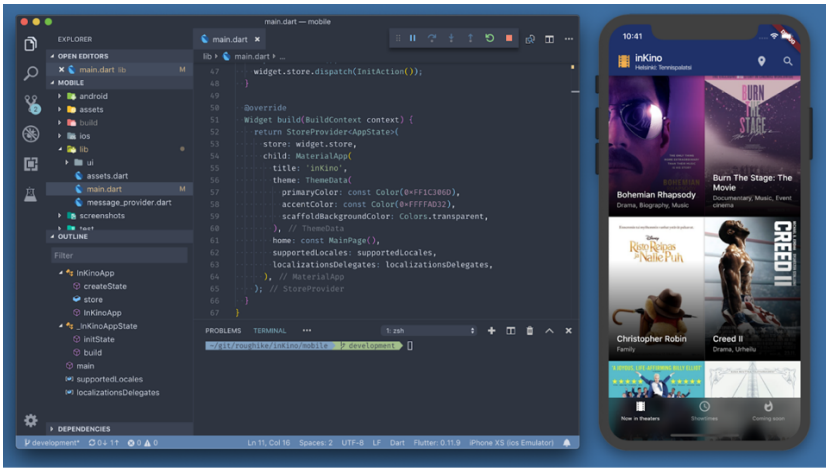


Рисунок 1.8 – приклад Flutter програми[4]

Програми на Flutter пишуться на мові програмування Dart, розробленої спеціально для даної платформи.[4] У цій мові використовуються переваги багатьох існуючих мов програмування. На операційних системах MacOS, Windows та Linux Flutter працює на віртуальній машині Dart, яка використовує динамічну компіляцію. Завдяки цьому дозволяється так зване «гаряче перезавантаження», за допомогою якої зміни будь-яких вихідних файлів можуть бути одразу застосовані у запущеній програмі без потреби у її перезапуску.

Механізм Flutter головним чином написаний на мові програмування C++, що забезпечує низькорівневу підтримку візуалізації за допомогою графічної бібліотеки Skia від Google. Також він взаємодіє з платформами SDK для Android та iOS. [4]

Flutter Engine – це портативне середовище для розміщення Flutter додатків. Вона реалізує основні бібліотеки Flutter, включаючи анімацію файлові та мережеві введення – виведення, підтримку доступності, архітектуру плагінів а також середовище виконання та компіляції Dart. Більшість розробників взаємодіють з Flutter за допомогою Flutter Framework, який забезпечує реактивну структуру а також набір базових віджетів та макетів.

На відміну від інших середовищ розробки які були описані вище, Flutter не має можливості взаємодіяти з графічними елементами напряму, перетягуючи їх на макет. У Flutter усе робиться безпосередньо у коді програми. Варто відмітити, що у даній платформі не використовуються нативні компоненти, а малює увесь інтерфейс самостійно за допомогою графічного двигуна Flutter.

Для побудови графічного дизайну Flutter використовує декларативний підхід, який оснований на компонентах (віджетах). Для збільшення швидкості роботи інтерфейсу компоненти перемальовуються тільки по необхідності тобто коли у них було щось змінено.

1.2.4. AppCode

AppCode – це інтегроване середовище розробки додатків на базі операційної системи iOS, яке було створено компанією JetBrains. Дана програма доступна тільки для платформи MacOS. Має підтримку написання на мовах програмування C++, Objective-C, Swift, XML, HTML, CSS, XPath, JavaScript. [5]

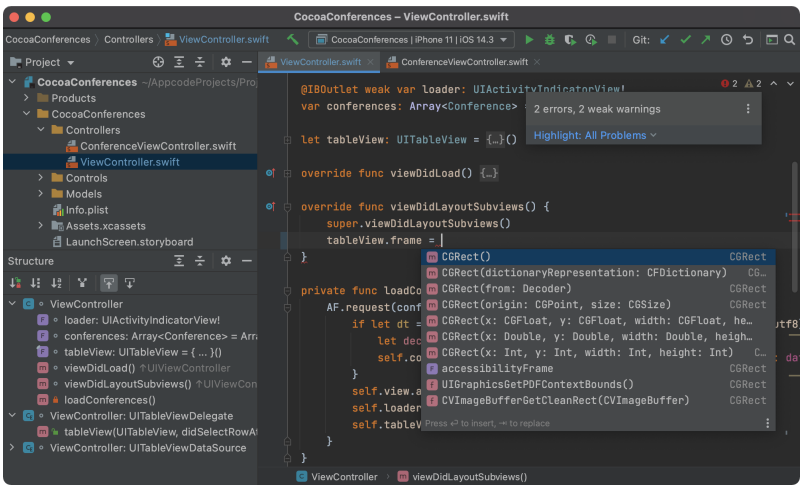


Рисунок 1.9 – інтерфейс AppCode[5]

AppCode має розумний редактор коду, який аналізує ваш проект і допомагає програмувати швидше. Він надає підказки щодо оптимізації коду,

виправленні помилок та має функції автоматичного форматування та заповнювання.

Рисунок 1.10 – приклад підказок у AppCode[5]

Також AppCode має у своєму арсеналі наступні функції :

- можливість запуску проекту на фізичному пристрої на базі iOS або на симуляторі;
- підтримка юніт тестів XCTest, Kiwi та Google Test;
- контроль версій Git, Github, Mercurian та інші;
- інтеграція з системами відстеження помилок Atlassian JIRA, JetBrains YouTrak та інші. [5]

1.3.Висновки до розділу 1

У цьому розділі були проаналізований розвиток сфери мобільних технологій. Була коротко розглянута історія розвитку даної сфери. Окрім цього Були описані сучасні середовища розробки мобільних додатків, їх інтерфейс, та основні можливості.

2. НОВІТНІ ТЕХНОЛОГІЇ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ

2.1.Сучасні мови програмування мобільних застосунків

2.1.1. Java

Java – мова програмування високого рівня, об’єктно-орієнтована, базується на класах. Була випущена компанією Sun Microsystems. Java призначена для того, щоб програми які на ній написані можна було запускати

де завгодно. Це означає, що скомпільований Java код може бути запущений у будь-якому середовищі, яке підтримує Java.[6] Синтаксис Java подібний до C, C++, але має менше засобів низького рівня. Згідно статистики Github, станом на 2020 рік Java є другою мою програмування за популярністю у світі.

Програми Java перетворюються у байт-код, який створюється за допомогою компілятору Javac та виконується за допомогою JVM. Віртуальна машина Java є головним компонентом Java. Завдяки використання єдиного байт-кода для багатьох платформ дозволяє описати Java як мову яку можна скомпілювати один раз та запускати на усіх підходящих платформах. Як правило JVM мають інтерпретатор байт-коду, проте на багатьох віртуальних машинах для підвищення швидкодії використовується JIT, таким чином фрагменти байт-коду які часто виконуються транслюються у машинний код.[6]

2.1.2. Java як основа платформи Android

Java було обрано як основу для розробки мобільних додатків на базі Android. Android SDK не використовує таких вбудованих у Java стандартів як :SE, ME та GUI. Android SDK працює на базі власної віртуальної машини Dalvik, яка була розроблена спеціально для пристроїв які мають невелику кількість ОЗУ – мобільні телефони, годинники, планшети та інші.

Для виконання на віртуальній машині Dalvik, файли класів компілюються у формат DEX. Після цього файли класів разом з іншими ресурсами з’єднуються у APK файли, задля подальшого встановлення на фізичних пристроях.

Можна виділити наступні переваги Java:

- Проста у навчанні
- Об’єктно – орієнтована
- Портативна
- Не залежить від платформи
- Захищена

- Сильно розповсюдженна
- Стійка
- Архітектурно нейтральна
- Велика потужність
- Підтримка багатьох потоків
- Динамічна

2.1.3. Kotlin

Kotlin – це крос-платформна, статично типізована, об’єктно орієнтована мова програмування. Була випущена у 2011 році компанією JetBrains та працює на платформі JVM. Kotlin є більш безпечною мовою програмування аніж Java та має стислий синтаксис.[7]

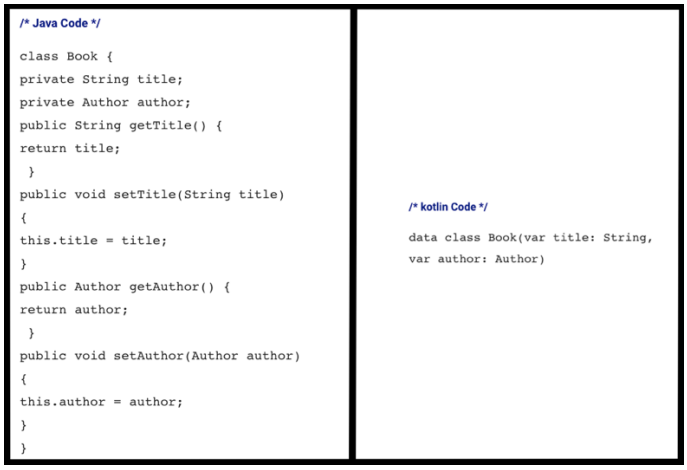
З 2019 року компанія Google повідомила, що Kotlin тепер стане більш привілейованою мовою для програмування на операційній системі Android, аніж Java. За замовчуванням Android Kotlin виробляє байт-код Java 6, який може працювати у будь-яких наступних JVM. Проте розробнику надається можливість обрати Java з 8 по 15 версію для оптимізації та підтримки більшої кількості спеціальних можливостей.

Станом на 2021 рік, Kotlin вважається кращою мовою для програмування ніж Java, але поки-що залишається повністю сумісною з Java, що дозволяє розробникам робити поступову міграцію з Java на Kotlin.[7]

Основні переваги Kotlin над Java:

- Програми на Kotlin швидше компілюються та менше займають пам’яті
- Код, написаний на Kotlin буде мати значно менший об’єм порівняно з Java, таким чином розробник буде робити менше помилок
- Kotlin захищений від NullPointerException

- Kotlin підтримує співпрограми, наприклад сумісність з JavaScript для веб-розробки.



```
/* Java Code */

class Book {
private String title;
private Author author;
public String getTitle() {
return title;
 }
public void setTitle(String title)
{
this.title = title;
}
public Author getAuthor() {
return author;
 }
public void setAuthor(Author author)
{
this.author = author;
}
}
```

```
/* kotlin Code */

data class Book(var title: String,
var author: Author)
```

Рисунок 2.1 – приклади написання коду на Java та Kotlin[]

2.1.4. Swift

Swift – це мова програмування з відкритим кодом, створена компанією Apple для розробки додатків на операційні системи iOS, MacOs, WatchOS та tvOS.[8] Дана мова була представлена у 2014 році, проте набирати популярність почала з виходом Swift 5.1 у 2019 році. Swift став альтернативою мові програмування Objective-C. Вона використовує сучасні концепції багатьох мов програмування та прагне надати більш простий синтаксис. Компанія Apple стверджує, що Swift є швидшим за Objective-C до 2,6 раза та до 8,4 раза швидша за Python 2.7.[8]

2.1.4.1. Порівняння Swift та Objective-C

Swift фактично був розроблений під впливом мови Objective-C. Сьогодні це дві основні мови для програмування мобільних додатків на платформі iOS, які мають декілька ключових відмінностей.

Для написання iOS додатку на мові Objective-C розробник має тільки одне середовище програмування Xcode. Swift ж має підтримку дещо більшої кількості середовищ розробки, такі як : Xcode, AppCode, Microsoft Visual Studio Code. Також для Swift було розроблено середовище Swift Playground, яке доступне для iOS та iPad. Це програма для вивчення Swift із підтримкою компіляції невеликих частин коду без необхідності створювати повноцінну проект.

Швидкість кодування відіграє велику роль у процесі розробки додатків, впливаючи на загальні витрати на створення застосунку а також на швидкість виходу програми на ринок. Незважаючи на те, що Swift та Objective-C є рідними мовами для iOS, вони не є схожими. Swift – набагато сучасніша мова, яка має простіший синтаксис та легка для опанування новачкам. При розробці на Swift об’єм коду буде до 2 разів менший у порівнянні з Objective-C. Це призводить до того, що розробник буде витрачати набагато менше часу на розробку та робити менше помилок.

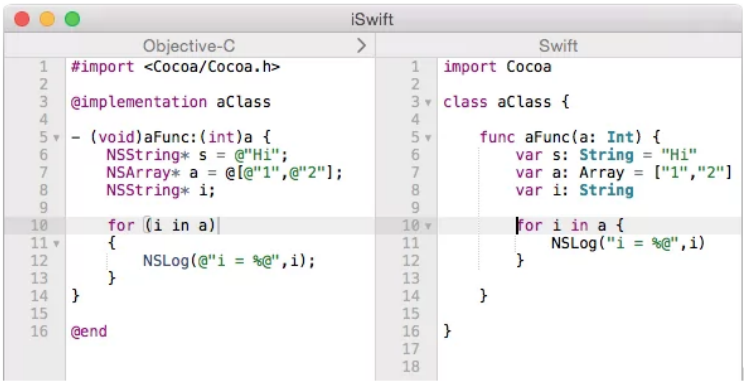


Рисунок 2.2 – приклад написання коду на Objective-C та Swift

Як було зазначено вище, Swift швидший за Objective-C. Це досягнуто завдяки тому, що Swift має простіший синтаксис та може виконувати перевірки типів під час компіляції. Окрім цього, Swift підтримує динамічні бібліотеки, які також збільшують швидкість додатків.

Завдяки тому, що Swift є мовою програмування з відкритим кодом, дана мова має набагато більший потенціал у її вивченні. Таким чином розробник має

змогу детально побачити як працює код і зрозуміти його більш ретельно. В Objective-C даної можливості немає, тому хоч ця мова програмування існує близько 30 років і має досить велику спільноту розробників, проте після появи Swift її популярність почала падати.

2.2. Обґрунтування використаних технології при розробці застосунку.

2.2.1. Мова програмування

Для розробки додатку була обрана мова програмування Java. Дана мова має досить простий, зрозумілий та звичний для мене синтаксис. Оскільки Java стала моєю першою мовою програмування, вона стала для мене так званим стандартом.

Також не менш важливою причиною вибору Java стало те, що це мова є об’єктно-орієнтованою. Всі об’єкти Java є нащадками класу Object, з якого вони наслідують властивості та поведінку.

Java вміє автоматично керувати пам’яттю під час життєвого циклу певного об’єкту. Таким чином мені не потрібно було самостійно звільнювати пам’ять після того як об’єкт перестав існувати. Все це робила віртуальна машина Java.

Ще одною перевагою як на мене є строга типізація тип даних. Таким чином перевірка типів буде відбуватися один раз на етапі компіляції програми. Не потрібно постійно думати чи я використовую не той тип даних. Завдяки цьому швидкість виконання програми буде швидшою ніж у програми написаній на нестрого типізованій мові програмування.

2.2.2. Середовище розробки

Серед усіх варіантів IDE, які підтримують мову програмування Java найкращим для себе я виділив Android Studio. Дане середовище розробки має зрозумілий і дружній інтерфейс (Рисунок 1.2).

У ньому вбудований редактор коду з підтримкою автоматичного заповнювання, автоматичного імпорту класів та бібліотек та можливістю додавання великої кількості плагінів. Також у даному середовищі вбудований інспектор баз даних, за допомогою якого можна легко взаємодіяти з вашими даними.

Варто відмітити, що Android Studio має один з найзручніших редакторів макету застосунку. Він дозволяє змінювати зовнішній вигляд програми не втручаючись у код власноруч.

2.2.3. База даних

Для роботи з базою даних я обрав SQLite. Це спрощена реляційна база даних, яка не використовує технологію клієнт-сервер. Для її використання досить імпортувати бібліотеку SQLiteDatabase, яка вбудована в Android Studio. Обрана мною база даних є швидшою аніж аналоги які працюють через сервер.

2.3.Висновки до розділу 2

У даному розділі були описані та проаналізовані сучасні мови програмування мобільних додатків. Також було обґрунтовано вибір технологій для розробки власного застосунку.

3. РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ЗА ДОПОМОГОЮ СУЧАСНИХ ТЕХНОЛОГІЙ

В якості практичної частини курсової роботи я реалізував застосунок на базі операційної системи Android за допомогою сучасних технологій. Мій застосунок написаний на мові програмування Java у середовищі Android Studio.

3.1. Опис функціоналу застосунку

Мною було створено застосунок під назвою Task Manager. Додаток було розроблено для керування власними повсякденними задачами.

Таким чином мій додаток дозволяє створювати задачі, встановлювати дату коли їх треба виконати. Є можливість позначення задачі як виконаної, редагування та видалення певної задачі.

3.2. Інтерфейс програми

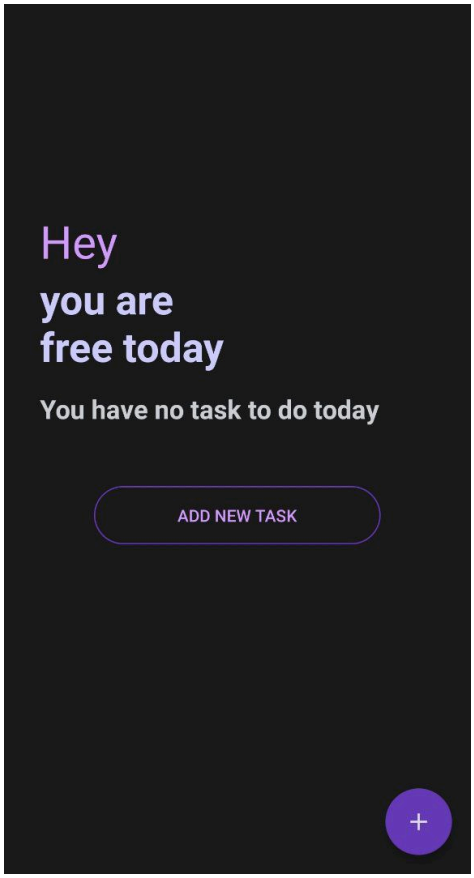


Рисунок 3.1 – Початковий екран застосунку

При запуску програми користувач може побачити привітливий дизайн. Користувачу пропонується створити нову задачу. Якщо він раніше створював задачі, йому буде показано їх перелік.

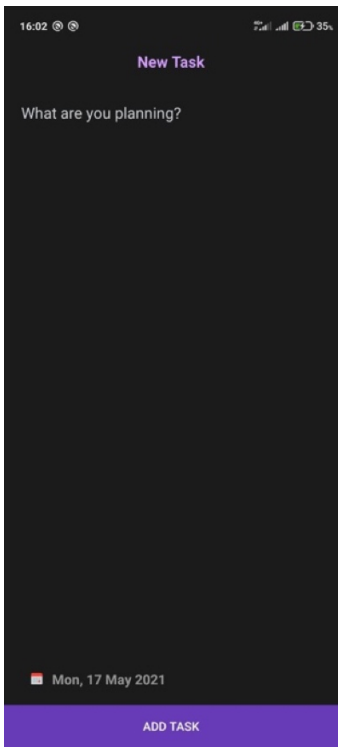


Рисунок 3.2 – створення нової задачі.

При натисненні на кнопку створити, користувачу відобразиться сторінка створення нової задачі. На ній є поле для вводу назви задачі, а також піктограма календаря, натиснувши на яку користувач зможе обрати дату планування його задачі.

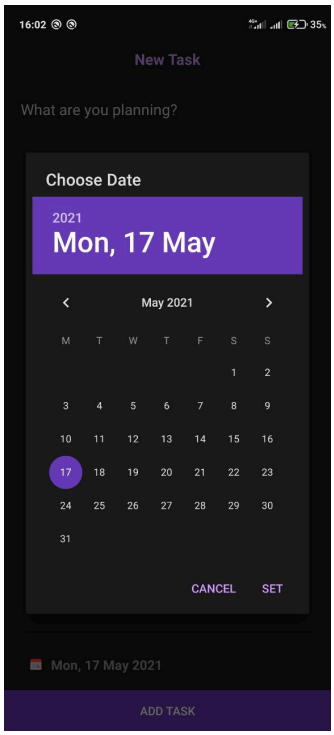


Рисунок 3.3 – вікно для вибору дати

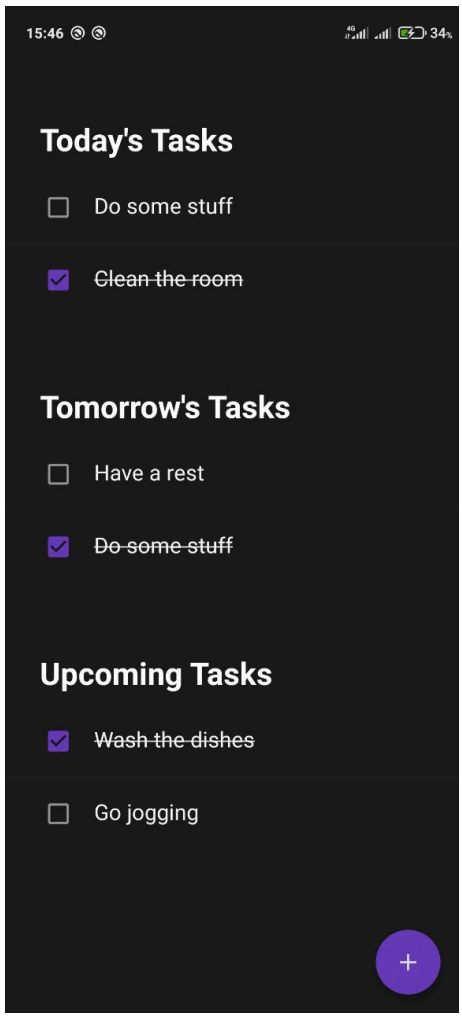


Рисунок 3.4 – Інтерфейс головної сторінки програми

Створивши усі необхідні задачі, користувач може побачити їх. Усі задачі поділені на три контейнери : сьогоднішні задачі, задачі на завтра та задачі які потрібно виконати незабаром.

На рисунку 3.5 можна побачити ще декілька можливостей застосунку. Це можливість редагувати текст задачі, змінити дату та можливість видалити задачу.

Рисунок 3.5 – Сторінка редагування та видалення задач

3.3.Висновок до розділу 3.

У цьому розділі був описаний функціонал розробленого застосунку до практичної частини курсової роботи, його можливості та надані ілюстрації роботи та зовнішнього вигляду програми.

Висновок

Розвиток сучасних технологій розробки мобільних додатків сьогодні є надзвичайно актуальною темою. Зі зростом попиту на смартфони зростає потреба у нових технологіях для розробки мобільних застосунків. У даній курсовій роботі були проаналізовані сучасні технології розробки мобільних технологій та тенденції їх розвитку. Було визначено що ключовими мобільними операційними системами на 2021 рік є Android та iOS і що в подальшому розвиток мобільних технологій буде стосуватися саме цих платформ.

Було наведено низку різних технологій для розробки. Розглянуто такі середовища розробки : Android Studio, Xcode, Flutter, AppCode. Для кожного з них було описано їх можливості, переваги та надано ілюстрації інтерфейсу.

Також було проаналізовано наступні сучасні мови програмування : Java, Kotlin, Swift. Було досліджено кожну з мов, описано їх можливості та основні переваги над іншими мовами програмування мобільних додатків.

В результаті виконання практичної частини курсової роботи, був розроблений додаток з використанням сучасних технологій розробки. Він дозволяє користувачеві створювати замітки для його повсякденних задач структурувати їх по даті, редагувати та видаляти. На мою думку даний застосунок є досить корисним та актуальним у сучасному житті, оскільки у людей є велика кількість задач та планів, які добре десь записувати та структурувати. Серед можливих варіантів розширення додатку на мій погляд є його розробка для iOS пристроїв.

Список використаних джерел

[1] Wireless data services : technologies, business models and global markets[електронний ресурс]/ Charma, Chetan, 2003 - Режим доступу: https://archive.org/details/wirelessdataserv0000shar_s7b4/page/n11/mode/2up

[2] Android Studio Documentation [електронний ресурс] - Режим доступу: https://developer.android.com/studio/intro

[3] Xcode Documentation [електронний ресурс] - Режим доступу : https://developer.apple.com/xcode/

[4] Flutter Documentation [електронний ресурс] - Режим доступу: https://flutter.dev/docs

[5] AppCode Documentation [електронний ресурс] - Режим доступу: https://www.jetbrains.com/help/objc/creating-and-viewing-doxygen-documentation.html

[6] Java Documentation [електронний ресурс] - Режим доступу: https://docs.oracle.com/en/java/

[7] Kotlin Documentation [електронний ресурс] - Режим доступу: https://developer.android.com/kotlin/add-kotlin

[8] Swift Documentation [електронний ресурс] - Режим доступу: https://swift.org/documentation/

ДОДАТКИ

```
public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "TaskManager.db";
    public static final String TABLE_NAME = "taks";

    public DBHelper(Context context) { super(context, DATABASE_NAME,  factory: null,  version: 2); }

    @Override
    public void onCreate(SQLiteDatabase db) {

        db.execSQL(
                "CREATE TABLE " + TABLE_NAME + "(id INTEGER PRIMARY KEY, task TEXT, task_at DATETIME, status INTEGER)"
        );
    }


    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }


    public boolean insertTask(String task, String task_at) {

        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("task", task);
```

Додаток А.1 – Клас для роботи з базою даних

```
        contentValues.put("task_at", task_at);

        contentValues.put("status", 0);
        db.insert(TABLE_NAME,  nullColumnHack: null, contentValues);
        return true;
    }

    public boolean updateTask(String id, String task, String task_at) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();

        contentValues.put("task", task);
        contentValues.put("task_at", task_at);

        db.update(TABLE_NAME, contentValues,  whereClause: "id = ? ", new String[]{id});
        return true;
    }

    public boolean deleteTask(String id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_NAME,  whereClause: "id = ? ", new String[]{id});
        return true;
    }

    public boolean updateTaskStatus(String id, Integer status) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();

        contentValues.put("status", status);

        db.update(TABLE_NAME, contentValues,  whereClause: "id = ? ", new String[]{id});
        return true;
    }
```

Додаток А.2 – Клас для роботи з базою даних

```
public Cursor getSingleTask(String id) {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.rawQuery( sql: "select * from " + TABLE_NAME + " WHERE id = '" + id + "' order by id desc", selectionArgs: null);

}

public Cursor getTodayTask() {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.rawQuery( sql: "select * from " + TABLE_NAME + " WHERE date(task_at) = date('now', 'localtime') order by id desc", selectionArgs: null);

}


public Cursor getTomorrowTask() {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.rawQuery( sql: "select * from " + TABLE_NAME + " WHERE date(task_at) = date('now', '+1 day', 'localtime')  order by id desc", selectionArgs: null);

}


public Cursor getUpcomingTask() {
    SQLiteDatabase db = this.getReadableDatabase();
    return db.rawQuery( sql: "select * from " + TABLE_NAME + " WHERE date(task_at) > date('now', '+1 day', 'localtime') order by id desc", selectionArgs: null);

}
```

Додаток А.3 – Клас для роботи з базою даних

```
public class ListTaskAdapter extends BaseAdapter {
    private Activity myActivity;
    private ArrayList<HashMap<String, String>> data;
    private DBHelper dbHelper;

    public ListTaskAdapter(Activity a, ArrayList<HashMap<String, String>> d, DBHelper mydb) {
        myActivity = a;
        data = d;
        dbHelper = mydb;
    }

    public int getCount() { return data.size(); }

    public Object getItem(int position) { return position; }

    public long getItemId(int position) { return position; }

    public View getView(int position, View convertView, ViewGroup parent) {
        ListTaskViewHolder viewHolder = null;
        if (convertView == null) {
            viewHolder = new ListTaskViewHolder();
            convertView = LayoutInflater.from(myActivity).inflate(R.layout.task_list_row, parent, attachToRoot: false);
            viewHolder.task_name = convertView.findViewById(R.id.task_name);
            viewHolder.checkBtn = convertView.findViewById(R.id.checkBtn);
            convertView.setTag(viewHolder);
        } else {
            viewHolder = (ListTaskViewHolder) convertView.getTag();
        }
```

Додаток А.4 – Клас адаптер для перетворювання даних з бази даних у View

```
        final HashMap<String, String> singleTask = data.get(position);
        final ListTaskViewHolder tmpHolder = viewHolder;

        viewHolder.task_name.setId(position);
        viewHolder.checkBtn.setId(position);

        try {


            viewHolder.checkBtn.setOnCheckedChangeListener(null);
            if (singleTask.get("status").contentEquals( cs: "1")) {
                viewHolder.task_name.setText(Html.fromHtml( source: "<strike>" + singleTask.get("task") + "</strike>"));
                viewHolder.checkBtn.setChecked(true);
            } else {
                viewHolder.task_name.setText(singleTask.get("task"));
                viewHolder.checkBtn.setChecked(false);
            }

            viewHolder.checkBtn.setOnCheckedChangeListener((buttonView, isChecked) -> {
                    if (isChecked) {
                        dbHelper.updateTaskStatus(singleTask.get("id"),  status: 1);
                        tmpHolder.task_name.setText(Html.fromHtml( source: "<strike>" + singleTask.get("task") + "</strike>"));
                    } else {
                        dbHelper.updateTaskStatus(singleTask.get("id"),  status: 0);
                        tmpHolder.task_name.setText(singleTask.get("task"));
                    }

            });


        } catch (Exception e) {
        }
        return convertView;
    }
}
```

Додаток А.5 – Клас адаптер для перетворювання даних з бази даних у View

```
public class AddModifyTask extends AppCompatActivity {

    Calendar task_calendar;
    Calendar today;
    DBHelper dbHelper;

    Boolean isModify = false;
    String task_id;
    TextView toolbar_title;
    EditText edit_text;
    TextView dateText;
    Button save_btn;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR);
        setContentView(R.layout.activity_add_modify_task);

        dbHelper = new DBHelper(getApplicationContext());
        task_calendar = new GregorianCalendar();
        today = new GregorianCalendar();
        toolbar_title = findViewById(R.id.toolbar_title);
        edit_text = findViewById(R.id.edit_text);
        dateText = findViewById(R.id.dateText);
        save_btn = findViewById(R.id.save_btn);


        dateText.setText(new SimpleDateFormat( pattern: "E, dd MMMM yyyy").format(task_calendar.getTime()));
```

Додаток А.6 – Клас який відповідає за додавання, редагування та видаляння задач

```
        Intent intent = getIntent();
        if (intent.hasExtra( name: "isModify")) {
            isModify = intent.getBooleanExtra( name: "isModify",  defaultValue: false);
            task_id = intent.getStringExtra( name: "id");
            init_modify();
        }


    }

    public void init_modify() {
        toolbar_title.setText("Modify Task");
        save_btn.setText("Update");
        LinearLayout deleteTask = findViewById(R.id.deleteTask);
        deleteTask.setVisibility(View.VISIBLE);
        Cursor task = dbHelper.getSingleTask(task_id);
        if (task != null) {
            task.moveToFirst();


            edit_text.setText(task.getString( i: 1));

             SimpleDateFormat iso8601Format = new SimpleDateFormat( pattern: "yyyy-MM-dd");
            try {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
                    task_calendar.setTime(Objects.requireNonNull(iso8601Format.parse(task.getString( i: 2))));
                }
            } catch (ParseException e) {
            }

            dateText.setText(new SimpleDateFormat( pattern: "E, dd MMM yyyy").format(task_calendar.getTime()));


        }
```

Додаток А.7 – Клас який відповідає за додавання, редагування та видаляння задач

```
    public void saveTask(View v) {

        //check if the task_date is today or over
        if (edit_text.getText().toString().trim().length() > 0) {
            if(today.get(Calendar.DAY_OF_MONTH) <= task_calendar.get(Calendar.DAY_OF_MONTH)) {
                //check whether task is modified or added
                if (isModify) {
                    dbHelper.updateTask(task_id, edit_text.getText().toString(), new SimpleDateFormat( pattern: "yyyy-MM-dd").format(task_calendar.getTime()));
                    Toast.makeText(getApplicationContext(),  text: "Task Updated.", Toast.LENGTH_SHORT).show();
                } else {
                    dbHelper.insertTask(edit_text.getText().toString(), new SimpleDateFormat( pattern: "yyyy-MM-dd").format(task_calendar.getTime()));
                    Toast.makeText(getApplicationContext(),  text: "Task Added.", Toast.LENGTH_SHORT).show();
                }
                finish();
            }
            else{
                Toast.makeText(getApplicationContext(),  text: "Please select date since today.", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(getApplicationContext(),  text: "Empty task can't be saved.", Toast.LENGTH_SHORT).show();
        }

    }

    public void deleteTask(View v) {
        dbHelper.deleteTask(task_id);
        Toast.makeText(getApplicationContext(),  text: "Task Removed", Toast.LENGTH_SHORT).show();
        finish();
    }
```

Додаток А.8 – Клас який відповідає за додавання, редагування та видаляння задач

```
public void chooseDate(View view) {
    final View dialogView = View.inflate( context: this, R.layout.date_picker,  root: null);
    final DatePicker datePicker = dialogView.findViewById(R.id.date_picker);
    datePicker.updateDate(task_calendar.get(Calendar.YEAR), task_calendar.get(Calendar.MONTH), task_calendar.get(Calendar.DAY_OF_MONTH));


    final AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setView(dialogView);
    builder.setTitle("Choose Date");
    builder.setNegativeButton( text: "Cancel",  listener: null);
    builder.setPositiveButton( text: "Set", (dialog, id) -> {

            task_calendar = new GregorianCalendar(datePicker.getYear(), datePicker.getMonth(), datePicker.getDayOfMonth());
            dateText.setText(new SimpleDateFormat( pattern: "E, dd MMMM yyyy").format(task_calendar.getTime()));

    });
    builder.show();
}
}
```

Додаток А.9 – Клас який відповідає за додавання, редагування та видаляння задач

```
// creates a list view with scrolling disabled
public class NoScrollListView extends ListView {

    public NoScrollListView(Context context) { super(context); }
    public NoScrollListView(Context context, AttributeSet attrs) { super(context, attrs); }
    public NoScrollListView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }
    @Override
    public void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int heightMeasureSpec_custom = MeasureSpec.makeMeasureSpec(
                size: Integer.MAX_VALUE >> 2, MeasureSpec.AT_MOST);
        super.onMeasure(widthMeasureSpec, heightMeasureSpec_custom);
        ViewGroup.LayoutParams params = getLayoutParams();
        params.height = getMeasuredHeight();
    }
}
```

Додаток А.10 – Клас який відповідає за відображення задач у певному контейнері

```java
public class MainActivity extends AppCompatActivity {

    DBHelper dbHelper;
    LinearLayout empty;
    NestedScrollView scrollView;
    LinearLayout todayContainer, tomorrowContainer, upcomingContainer;
    NoScrollListView taskListToday, taskListTomorrow, taskListUpcoming;
    ArrayList<HashMap<String, String>> todayList = new ArrayList<>();
    ArrayList<HashMap<String, String>> tomorrowList = new ArrayList<>();
    ArrayList<HashMap<String, String>> upcomingList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR);
        setContentView(R.layout.activity_main);

        dbHelper = new DBHelper( context: this);
        empty = findViewById(R.id.empty);
        scrollView = findViewById(R.id.scrollView);
        todayContainer = findViewById(R.id.todayContainer);
        tomorrowContainer = findViewById(R.id.tomorrowContainer);
        upcomingContainer = findViewById(R.id.otherContainer);
        taskListToday = findViewById(R.id.taskListToday);
        taskListTomorrow = findViewById(R.id.taskListTomorrow);
        taskListUpcoming = findViewById(R.id.taskListUpcoming);
    }

    public void openAddModifyTask(View view) {
        startActivity(new Intent( packageContext: this, AddModifyTask.class));
    }
```

Додаток А.11 – Клас який відповідає за відображення усього додатка

```java
    @Override
    public void onResume() {
        super.onResume();
        populateData();
    }


    public void populateData() {
        dbHelper = new DBHelper( context: this);

        runOnUiThread(() -> { fetchDataFromDB(); });
    }


    public void fetchDataFromDB() {
        todayList.clear();
        tomorrowList.clear();
        upcomingList.clear();

        Cursor today = dbHelper.getTodayTask();
        Cursor tomorrow = dbHelper.getTomorrowTask();
        Cursor upcoming = dbHelper.getUpcomingTask();

        loadDataList(today, todayList);
        loadDataList(tomorrow, tomorrowList);
        loadDataList(upcoming, upcomingList);


        if (todayList.isEmpty() && tomorrowList.isEmpty() && upcomingList.isEmpty()) {
            empty.setVisibility(View.VISIBLE);
            scrollView.setVisibility(View.GONE);
        } else {
            empty.setVisibility(View.GONE);
            scrollView.setVisibility(View.VISIBLE);
```

Додаток А.12 – Клас який відповідає за відображення усього додатка

```
        if (todayList.isEmpty()) {
            todayContainer.setVisibility(View.GONE);
        } else {
            todayContainer.setVisibility(View.VISIBLE);
            loadListView(taskListToday, todayList);
        }

        if (tomorrowList.isEmpty()) {
            tomorrowContainer.setVisibility(View.GONE);
        } else {
            tomorrowContainer.setVisibility(View.VISIBLE);
            loadListView(taskListTomorrow, tomorrowList);
        }

        if (upcomingList.isEmpty()) {
            upcomingContainer.setVisibility(View.GONE);
        } else {
            upcomingContainer.setVisibility(View.VISIBLE);
            loadListView(taskListUpcoming, upcomingList);
        }
    }
}
```

Додаток А.13 – Клас який відповідає за відображення усього додатка

```
public void loadDataList(Cursor cursor, ArrayList<HashMap<String, String>> dataList) {
    if (cursor != null) {
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {

            HashMap<String, String> mapToday = new HashMap<~>();
            mapToday.put("id", cursor.getString( i: 0));
            mapToday.put("task", cursor.getString( i: 1));
            mapToday.put("date", cursor.getString( i: 2));
            mapToday.put("status", cursor.getString( i: 3));
            dataList.add(mapToday);
            cursor.moveToNext();
        }
    }
}

public void loadListView(NoScrollListView listView, final ArrayList<HashMap<String, String>> dataList) {
    ListTaskAdapter adapter = new ListTaskAdapter( a: this, dataList, dbHelper);
    listView.setAdapter(adapter);
    listView.setOnItemClickListener((parent, view, position, id) -> {
            Intent intent = new Intent( packageContext: MainActivity.this, AddModifyTask.class);
            intent.putExtra( name: "isModify",  value: true);
            intent.putExtra( name: "id", dataList.get(+position).get("id"));
            startActivity(intent);
    });
}
}
```

Додаток А.14 – Клас який відповідає за відображення усього додатка

```
class ListTaskViewHolder {
    TextView task_name;
    CheckBox checkBtn;
}
```

Додаток А.14 - Допоміжний клас для відображення даних.