Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра комп’ютерної інженерії та інноваційних технологій

Педяш В.В.

ПРОГРАМУВАННЯ ДЛЯ МОБІЛЬНИХ ПЛАТФОРМ

Методичні рекомендації до практичних робіт
для здобувачів вищої освіти,
які навчаються за спеціальностями
галузі «Інформаційні технології»

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

Педяш В.В. Програмування для мобільних платформ. Методичні рекомендації до практичних робіт для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» [Електронне видання]. Кафедра комп’ютерної інженерії та інноваційних технологій Міжнародного гуманітарного університету. Одеса, 2025. 30 с.

Навчальна дисципліна «Програмування для мобільних платформ» присвячена вивченню основних підходів і технологій розроблення програмного забезпечення для сучасних мобільних операційних систем, насамперед Android та iOS. У межах дисципліни розглядаються принципи побудови мобільних застосунків, особливості архітектури мобільних операційних систем, моделі взаємодії компонентів застосунку, організація користувацького інтерфейсу, робота з локальними даними та мережевими сервісами. Особлива увага приділяється використанню сучасних інструментів розроблення та мов програмування, що застосовуються у створенні мобільних застосунків. У процесі навчання студенти опановують основні технології розроблення мобільного програмного забезпечення, знайомляться з середовищами розробки, засобами налагодження та тестування мобільних застосунків, а також набувають практичних навичок створення програмних рішень для мобільних платформ.

ЗМІСТ

Практична робота 1. Аналіз мобільних платформ Android та iOS

Мета роботи – ознайомитися з особливостями сучасних мобільних платформ, дослідити архітектуру, інструменти розробки та підходи до створення мобільних застосунків на платформах Android та iOS, а також виконати їх порівняльний аналіз.

Ключові положення

Мобільні платформи є програмно-апаратними середовищами, що забезпечують виконання мобільних застосунків, взаємодію з користувачем та доступ до ресурсів пристрою. Найбільш поширеними платформами є Android та iOS, які мають різні підходи до архітектури, розробки та розповсюдження програмного забезпечення.

Android є відкритою платформою, що базується на ядрі Linux і дозволяє використовувати широкий спектр пристроїв різних виробників. Вона характеризується високою гнучкістю, можливістю налаштування системи та підтримкою різних мов програмування, зокрема Java і Kotlin. Розробка застосунків здійснюється переважно у середовищі Android Studio, яке надає інструменти для створення, тестування та налагодження програм.

iOS є закритою платформою, що використовується виключно на пристроях компанії Apple. Вона відзначається високим рівнем оптимізації, стабільністю роботи та строгими вимогами до якості застосунків. Основними мовами розробки є Swift і Objective-C, а розробка здійснюється у середовищі Xcode.

Архітектура мобільних платформ визначає спосіб організації компонентів системи. У Android використовується багаторівнева архітектура, що містить рівень ядра, бібліотек, середовища виконання та прикладного рівня. У iOS архітектура також є багаторівневою, але має більш жорстку інтеграцію компонентів і контроль з боку операційної системи.

Важливою складовою мобільних платформ є модель життєвого циклу застосунку. У Android застосунки складаються з компонентів (Activity, Service, Broadcast Receiver, Content Provider), кожен з яких має власний життєвий цикл. У iOS використовується модель, що базується на життєвому циклі застосунку та його станах (active, inactive, background).

Розповсюдження мобільних застосунків здійснюється через офіційні магазини. Для Android це Google Play, а для iOS – App Store. Кожна платформа має власні вимоги до публікації застосунків, перевірки безпеки та якості.

Таким чином, Android та iOS мають суттєві відмінності у підходах до розробки, архітектури та розповсюдження програмного забезпечення, що необхідно враховувати під час створення мобільних застосунків.

Практичне завдання

1. Ознайомитися з основними характеристиками платформ Android та iOS.
2. Дослідити архітектуру кожної платформи.
3. Визначити основні інструменти розробки для Android та iOS.
4. Проаналізувати підтримувані мови програмування.
5. Дослідити модель життєвого циклу мобільного застосунку на кожній платформі.
6. Проаналізувати вимоги до публікації застосунків у Google Play та App Store.

7. Виконати порівняльний аналіз платформ за заданими критеріями (архітектура, інструменти, продуктивність, безпека).

8. Оформити результати у вигляді таблиці порівняння.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з базовими поняттями мобільних платформ і принципами функціонування операційних систем для мобільних пристроїв. Особливу увагу слід приділити архітектурі систем та моделі виконання застосунків.

На першому етапі необхідно зібрати інформацію про платформу Android. Слід дослідити її архітектуру, основні компоненти, інструменти розробки та підтримувані мови програмування. Результати аналізу доцільно оформити у вигляді короткого опису.

На наступному етапі необхідно виконати аналогічний аналіз платформи iOS. Слід звернути увагу на відмінності у підходах до розробки, рівень інтеграції системи та вимоги до застосунків.

Далі необхідно виконати порівняльний аналіз платформ. Для цього слід визначити критерії порівняння, такі як відкритість системи, інструменти розробки, продуктивність, безпека, зручність використання та вимоги до публікації застосунків.

Результати порівняння доцільно представити у вигляді таблиці, що дозволить наочно відобразити відмінності між платформами. У таблиці слід зазначити основні характеристики кожної платформи за обраними критеріями.

У процесі виконання роботи необхідно зробити висновки щодо доцільності використання кожної платформи залежно від умов розробки, вимог до застосунку та цільової аудиторії.

Контрольні питання

1. Що таке мобільна платформа?
2. Які основні особливості платформи Android?
3. Які основні особливості платформи iOS?
4. Які інструменти використовуються для розробки застосунків під Android?
5. Які інструменти використовуються для розробки застосунків під iOS?
6. Які мови програмування використовуються на кожній платформі?
7. У чому полягають особливості архітектури Android?
8. У чому полягають особливості архітектури iOS?
9. Як відбувається розповсюдження мобільних застосунків?
10. Які основні відмінності між Android та iOS?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис платформи Android;
– опис платформи iOS;
– таблиця порівняльного аналізу;

– результати дослідження;
– висновки.

Практична робота 2. Аналіз життєвого циклу мобільного застосунку

Мета роботи – ознайомитися з етапами життєвого циклу мобільного застосунку, дослідити особливості керування станами застосунку на платформах Android та iOS, а також навчитися аналізувати зміну станів і реакцію програми на події системи.

Ключові положення

Життєвий цикл мобільного застосунку визначає послідовність станів, у яких перебуває програма під час свого виконання, а також події, що викликають переходи між цими станами. Розуміння життєвого циклу є необхідною умовою для забезпечення стабільної роботи застосунку, ефективного використання ресурсів пристрою та коректної обробки подій користувача.

У платформі Android життєвий цикл реалізується через компоненти застосунку, зокрема Activity. Кожен компонент має визначені методи життєвого циклу, такі як onCreate, onStart, onResume, onPause, onStop та onDestroy. Кожен із цих методів викликається системою у відповідь на зміну стану застосунку. Наприклад, метод onCreate викликається під час створення компонента, а onPause – коли застосунок втрачає фокус.

У платформі iOS життєвий цикл базується на станах застосунку, серед яких активний стан, неактивний стан, фоновий режим та завершення роботи. Керування життєвим циклом здійснюється через делегати застосунку та сцени, які реагують на події системи, наприклад запуск, перехід у фоновий режим або повернення до активного стану.

Важливим аспектом життєвого циклу є управління ресурсами. Під час переходу застосунку у фоновий режим необхідно звільняти ресурси, зберігати стан даних і припиняти непотрібні обчислення. Це дозволяє зменшити навантаження на систему та уникнути втрати даних.

Події життєвого циклу можуть бути викликані різними факторами, такими як дії користувача, системні повідомлення або обмеження ресурсів. Наприклад, отримання телефонного дзвінка або перемикання між застосунками призводить до зміни стану програми.

Таким чином, життєвий цикл мобільного застосунку визначає логіку його роботи у різних станах та забезпечує коректну взаємодію із системою та користувачем.

Практичне завдання

1. Ознайомитися з поняттям життєвого циклу мобільного застосунку.
2. Дослідити методи життєвого циклу Activity у Android.
3. Проаналізувати стани застосунку у iOS.
4. Визначити події, що викликають зміну станів застосунку.
5. Реалізувати простий мобільний застосунок або демонстраційний проєкт, що відслідковує зміни життєвого циклу (виведення повідомлень у лог).
6. Зафіксувати послідовність викликів методів життєвого циклу під час запуску, згортання та закриття застосунку.

7. Порівняти життєвий цикл Android та iOS.
8. Проаналізувати вплив життєвого циклу на управління ресурсами.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з документацією щодо життєвого циклу мобільних застосунків на обох платформах. Особливу увагу слід приділити послідовності викликів методів та їх призначенню.

На першому етапі необхідно дослідити життєвий цикл у Android. Слід розглянути кожен метод життєвого циклу Activity та визначити, у яких ситуаціях він викликається. Для кращого розуміння доцільно створити простий застосунок, у якому кожен метод виводить повідомлення у журнал (Logcat).

На наступному етапі необхідно дослідити життєвий цикл у iOS. Слід проаналізувати основні стани застосунку та відповідні методи, що викликаються під час переходів між ними. Особливу увагу слід звернути на обробку переходу у фоновий режим та повернення до активного стану.

Далі необхідно виконати серію експериментів із застосунком. Слід запустити застосунок, згорнути його, повернути у активний стан, а також завершити роботу. Для кожного випадку необхідно зафіксувати послідовність викликів методів життєвого циклу.

Після цього необхідно виконати порівняльний аналіз життєвого циклу Android та iOS. Слід визначити спільні риси та відмінності, а також оцінити складність реалізації обробки подій життєвого циклу.

У завершальній частині роботи необхідно проаналізувати, як життєвий цикл впливає на використання ресурсів пристрою, збереження стану застосунку та забезпечення його стабільності.

Контрольні питання

1. Що таке життєвий цикл мобільного застосунку?
2. Які основні методи життєвого циклу Activity у Android?
3. Які стани має застосунок у iOS?
4. Які події викликають зміну станів застосунку?
5. Яке призначення методу onCreate?
6. Яке призначення методу onPause?
7. Як відбувається перехід застосунку у фоновий режим?
8. Чому важливо зберігати стан застосунку?
9. Як життєвий цикл впливає на використання ресурсів?
10. Які основні відмінності життєвого циклу Android та iOS?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис життєвого циклу Android;
– опис життєвого циклу iOS;
– результати експериментів;
– аналіз послідовності викликів методів;
– висновки.

Практична робота 3. Аналіз методів розповсюдження мобільних застосунків

Мета роботи – ознайомитися з основними способами розповсюдження мобільних застосунків, дослідити вимоги до публікації та оновлення застосунків на платформах Android та iOS, а також виконати аналіз каналів доставки програмного забезпечення користувачам.

Ключові положення

Розповсюдження мобільних застосунків є завершальним етапом розробки програмного забезпечення, що забезпечує доступ користувачів до встановлення та використання застосунку. Вибір методу розповсюдження впливає на доступність застосунку, його безпеку, швидкість оновлення та контроль якості.

Найпоширенішим способом розповсюдження є використання офіційних магазинів застосунків. Для платформи Android основним каналом є Google Play, а для iOS – App Store. Ці сервіси забезпечують централізоване розміщення застосунків, автоматичне оновлення та механізми перевірки безпеки.

Публікація застосунків у магазинах передбачає дотримання встановлених вимог. Для Android процес публікації є відносно відкритим і передбачає завантаження APK або AAB-файлів, заповнення метаданих та проходження автоматичної перевірки. Для iOS процес є більш регламентованим і включає обов'язкову модерацію застосунку з боку компанії Apple.

Альтернативними способами розповсюдження є встановлення застосунків поза офіційними магазинами. У Android це може бути встановлення APK-файлів безпосередньо з вебсайтів або внутрішніх корпоративних ресурсів. Такий підхід використовується для тестування або розповсюдження внутрішніх застосунків. У iOS альтернативні методи обмежені та реалізуються через механізми тестування або корпоративного розповсюдження.

Важливим аспектом є організація процесу оновлення застосунків. Офіційні магазини забезпечують автоматичне оновлення, що дозволяє швидко виправляти помилки та додавати нову функціональність. У разі альтернативного розповсюдження необхідно самостійно реалізовувати механізми оновлення.

Безпека розповсюдження є критично важливою. Офіційні магазини здійснюють перевірку застосунків на наявність шкідливого коду, тоді як встановлення з неофіційних джерел може становити ризик для користувача.

Таким чином, існують різні методи розповсюдження мобільних застосунків, кожен з яких має свої переваги та обмеження, що необхідно враховувати під час розробки програмного забезпечення.

Практичне завдання

1. Ознайомитися з основними способами розповсюдження мобільних застосунків.
2. Дослідити процес публікації застосунку у Google Play.
3. Проаналізувати процес публікації застосунку в App Store.
4. Визначити вимоги до застосунків для кожної платформи.
5. Дослідити альтернативні способи розповсюдження застосунків.
6. Проаналізувати механізми оновлення застосунків.
7. Оцінити рівень безпеки різних способів розповсюдження.
8. Виконати порівняльний аналіз методів розповсюдження.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з офіційною документацією платформ щодо розповсюдження застосунків. Особливу увагу слід приділити вимогам до публікації та процедурі перевірки застосунків.

На першому етапі необхідно дослідити процес розповсюдження застосунків у Google Play. Слід проаналізувати порядок створення облікового запису розробника, підготовку застосунку до публікації, формування метаданих та проходження перевірки.

На наступному етапі необхідно дослідити процес розповсюдження застосунків в App Store. Слід звернути увагу на особливості підготовки застосунку, процедуру модерації та вимоги до якості програмного забезпечення.

Далі необхідно проаналізувати альтернативні способи розповсюдження. Слід розглянути встановлення застосунків поза офіційними магазинами, а також їх використання у тестуванні та корпоративних системах.

Після цього необхідно виконати порівняльний аналіз методів розповсюдження. Для цього слід визначити критерії оцінювання, такі як доступність, безпека, зручність оновлення та контроль якості.

У завершальній частині роботи необхідно зробити висновки щодо доцільності використання різних методів розповсюдження залежно від типу застосунку та умов його використання.

Контрольні питання

1. Що таке розповсюдження мобільних застосунків?
2. Які основні способи розповсюдження застосунків існують?
3. Яке призначення Google Play?
4. Яке призначення App Store?
5. Які вимоги до публікації застосунків у Android?
6. Які вимоги до публікації застосунків у iOS?
7. Що таке альтернативні способи розповсюдження?
8. Як організовується оновлення застосунків?
9. Які ризики пов’язані з неофіційним розповсюдженням?
10. Які основні відмінності між методами розповсюдження Android та iOS?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис методів розповсюдження;
– аналіз процесу публікації у Google Play;
– аналіз процесу публікації в App Store;
– порівняльний аналіз;
– висновки.

Практична робота 4. Ознайомлення зі структурою Android-застосунку

Мета роботи – ознайомитися зі структурою проєкту мобільного застосунку на платформі Android, дослідити основні компоненти та файли проєкту, а також навчитися орієнтуватися у середовищі розробки Android Studio.

Ключові положення

Android-застосунок має чітко визначену структуру проєкту, яка забезпечує організацію програмного коду, ресурсів та конфігураційних файлів. Розуміння цієї структури є необхідною умовою для ефективної розробки мобільних застосунків.

Основою проєкту є каталог app, що містить вихідний код, ресурси та конфігураційні файли застосунку. Вихідний код зазвичай розташовується у каталозі java або kotlin, де організовано пакети та класи програми. Основним компонентом є Activity, яка відповідає за взаємодію з користувачем і відображення інтерфейсу.

Важливим елементом є файл AndroidManifest.xml, який містить опис застосунку, його компонентів, дозволів та параметрів конфігурації. У цьому файлі визначається точка входу в застосунок, а також перелік необхідних дозволів для доступу до ресурсів пристрою.

Ресурси застосунку зберігаються у каталозі res. До них належать макети інтерфейсу (layout), графічні ресурси (drawable), рядкові ресурси (values) та інші допоміжні файли. Макети інтерфейсу описуються за допомогою XML-файлів і визначають розташування елементів на екрані.

Система збірки застосунку базується на інструменті Gradle, який відповідає за компіляцію коду, керування залежностями та формування інсталяційного пакета. Файли build.gradle містять налаштування проєкту та параметри збірки.

Також важливу роль відіграє структура пакетів, яка забезпечує логічну організацію коду та полегшує його підтримку. Використання модульного підходу дозволяє розділити застосунок на окремі компоненти та підвищити його масштабованість.

Таким чином, структура Android-застосунку є сукупністю взаємопов’язаних компонентів, що забезпечують його функціонування, організацію коду та взаємодію з системою.

Практичне завдання

1. Встановити та запустити середовище розробки Android Studio.
2. Створити новий проєкт Android-застосунку.
3. Ознайомитися зі структурою створеного проєкту.
4. Дослідити вміст каталогу app.
5. Відкрити та проаналізувати файл AndroidManifest.xml.
6. Ознайомитися зі структурою каталогу res та його підкаталогами.
7. Відкрити XML-файл макета інтерфейсу та проаналізувати його структуру.
8. Ознайомитися з файлами build.gradle та їх призначенням.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно встановити середовище розробки Android Studio та переконатися у його коректній роботі. Після запуску середовища слід створити новий проєкт, обравши базовий шаблон застосунку.

На першому етапі необхідно ознайомитися зі структурою проєкту. Слід звернути увагу на основні каталоги та файли, які створюються автоматично. Особливу увагу слід приділити каталогу app, оскільки він містить основні компоненти застосунку.

Далі необхідно відкрити файл AndroidManifest.xml та проаналізувати його структуру. Слід визначити, які компоненти описані у цьому файлі та які дозволи використовуються застосунком.

На наступному етапі необхідно дослідити каталог ресурсів res. Слід відкрити файли макетів інтерфейсу та проаналізувати, як описується розташування елементів на екрані за допомогою XML.

Після цього необхідно ознайомитися з файлами конфігурації Gradle. Слід визначити, які параметри відповідають за збірку застосунку та керування залежностями.

У процесі виконання роботи доцільно змінити деякі елементи інтерфейсу або текстові ресурси, щоб перевірити, як зміни відображаються у застосунку після збірки та запуску.

Контрольні питання

1. Яка структура Android-застосунку?
2. Яке призначення каталогу app?
3. Що містить файл AndroidManifest.xml?
4. Яке призначення каталогу res?
5. Як описується інтерфейс користувача у Android?
6. Яке призначення файлів build.gradle?
7. Що таке Activity?
8. Як організовується вихідний код у проєкті?
9. Яке призначення ресурсів застосунку?
10. Яке значення має структура проєкту для розробки застосунку?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис структури проєкту;
– аналіз основних файлів;
– результати дослідження;
– висновки.

Практична робота 5. Використання інтентів для взаємодії компонентів

Мета роботи – ознайомитися з механізмом інтентів у платформі Android, дослідити їх роль у взаємодії компонентів застосунку та навчитися реалізовувати передачу даних між Activity.

Ключові положення

Інтент є основним механізмом взаємодії між компонентами Android-застосунку. Він використовується для запуску Activity, Service або надсилання повідомлень іншим компонентам системи. Інтент дозволяє визначити дію, яку необхідно виконати, а також передати додаткові дані.

Існують два основні типи інтентів: явні та неявні. Явні інтенті використовуються для запуску конкретного компонента застосунку, коли його ім’я відоме заздалегідь. Неявні інтенті описують дію, яку потрібно виконати, без зазначення конкретного компонента, і система самостійно визначає відповідний об’єкт для обробки.

Передача даних між компонентами здійснюється за допомогою додаткових параметрів (extras), які додаються до інтенту у вигляді пар «ключ–значення». Це дозволяє передавати як прості типи даних, так і складні структури.

Інтенти також використовуються для взаємодії із системними застосунками. Наприклад, за допомогою неявного інтенту можна відкрити браузер, здійснити дзвінок або надіслати повідомлення. Такий підхід забезпечує повторне використання функціональності системи без необхідності реалізації її у власному застосунку.

Обробка інтентів виконується у методах життєвого циклу компонентів. Під час запуску Activity отримані дані можуть бути оброблені та використані для налаштування інтерфейсу або логіки роботи програми.

Таким чином, інтенти є універсальним механізмом взаємодії компонентів Android-застосунку, що забезпечує гнучкість, модульність та можливість інтеграції з іншими застосунками.

Практичне завдання

1. Створити новий Android-застосунок або використати існуючий проєкт.
2. Додати другу Activity до застосунку.
3. Реалізувати запуск другої Activity за допомогою явного інтенту.
4. Передати дані з першої Activity до другої через extras.
5. Реалізувати обробку отриманих даних у другій Activity.
6. Додати можливість повернення результату до першої Activity.
7. Реалізувати використання неявного інтенту для відкриття вебсторінки або іншого системного застосунку.
8. Проаналізувати роботу інтентів під час виконання програми.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з поняттям інтентів та їх роллю у системі Android. Особливу увагу слід приділити відмінностям між явними та неявними інтентами.

На першому етапі необхідно створити проєкт у середовищі Android Studio та додати другу Activity. Слід переконатися, що новий компонент правильно зареєстрований у файлі конфігурації застосунку.

Далі необхідно реалізувати запуск другої Activity за допомогою явного інтенту. Для цього слід створити об’єкт інтенту, вказати цільовий компонент та викликати метод запуску Activity.

На наступному етапі необхідно реалізувати передачу даних між Activity. Слід додати до інтенту параметри у вигляді пар «ключ–значення» та отримати їх у другій Activity. Отримані дані доцільно відобразити в інтерфейсі застосунку.

Після цього необхідно реалізувати повернення результату з другої Activity. Слід сформувати відповідь, передати її через інтент та обробити у першій Activity.

Далі необхідно реалізувати приклад використання неявного інтенту. Наприклад, можна відкрити вебсторінку у браузері або викликати інший системний застосунок.

У процесі виконання роботи необхідно протестувати застосунок, перевірити коректність передачі даних та послідовність взаємодії компонентів.

Контрольні питання

1. Що таке інтент у Android?
2. Які типи інтентів існують?
3. У чому різниця між явним та неявним інтентом?
4. Як здійснюється передача даних між Activity?
5. Що таке extras?
6. Як обробляються інтенти у компоненті?
7. Як реалізувати запуск іншої Activity?
8. Як отримати результат з іншої Activity?
9. Для чого використовуються неявні інтенти?
10. Яке значення інтентів у взаємодії компонентів застосунку?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис механізму інтентів;
– приклади реалізації;
– результати тестування;
– висновки.

Практична робота 6. Запуск і налагодження застосунку в емуляторі

Мета роботи – ознайомитися з процесом запуску та налагодження мобільного застосунку в емуляторі платформи Android, навчитися використовувати інструменти середовища Android Studio для тестування, виявлення та усунення помилок.

Ключові положення

Емулятор є програмним засобом, що дозволяє відтворювати роботу мобільного пристрою на персональному комп'ютері. Він забезпечує можливість тестування застосунків без використання фізичного пристрою, що є важливим на початкових етапах розробки.

У середовищі Android Studio використовується Android Virtual Device (AVD), який дозволяє створювати віртуальні пристрої з різними параметрами, такими як версія операційної системи, розмір екрана, обсяг пам'яті та інші характеристики. Це дає змогу тестувати застосунок у різних умовах.

Запуск застосунку в емуляторі включає процес компіляції, збірки та встановлення програми у віртуальне середовище. Після запуску застосунок виконується так само, як і на реальному пристрої, що дозволяє перевірити його функціональність.

Налагодження застосунку є процесом виявлення та усунення помилок у програмному коді. Для цього використовуються інструменти відлагодження, зокрема точки зупину, покрокове виконання програми та перегляд значень змінних.

Важливим інструментом є Logcat, який дозволяє переглядати системні повідомлення, журнали виконання програми та повідомлення про помилки. Аналіз журналів дає змогу визначити причини некоректної роботи застосунку.

Емулятор також дозволяє імітувати різні умови роботи пристрою, такі як зміна орієнтації екрана, рівень заряду батареї, стан мережі або надходження викликів. Це дає змогу перевірити поведінку застосунку у різних ситуаціях.

Таким чином, використання емулятора та інструментів налагодження є важливою складовою процесу розробки мобільних застосунків, що забезпечує підвищення їх якості та надійності.

Практичне завдання

1. Запустити середовище Android Studio.
2. Створити або відкрити існуючий проєкт Android-застосунку.
3. Створити віртуальний пристрій (AVD) з заданими параметрами.
4. Запустити емулятор та встановити застосунок.
5. Перевірити роботу застосунку у емуляторі.
6. Додати точки зупину у програмному коді.
7. Виконати покрокове налагодження програми.
8. Проаналізувати повідомлення у Logcat.
9. Виявити та усунути помилки у коді.
10. Перевірити роботу застосунку після внесення змін.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що середовище Android Studio встановлене та налаштоване коректно. Слід перевірити наявність необхідних компонентів SDK та інструментів емуляції.

На першому етапі необхідно створити віртуальний пристрій за допомогою менеджера AVD. Слід обрати модель пристрою, версію операційної системи та інші параметри, що відповідають умовам тестування.

Далі необхідно запустити емулятор та виконати збірку застосунку. Після встановлення застосунку слід перевірити його основні функції та коректність роботи інтерфейсу.

На наступному етапі необхідно виконати налагодження програми. Слід встановити точки зупину у коді та виконати програму у режимі налагодження. Це дозволить проаналізувати виконання програми та значення змінних.

Особливу увагу слід приділити аналізу повідомлень у Logcat. Слід виявити повідомлення про помилки та попередження, визначити їх причини та внести необхідні зміни у програмний код.

Після внесення змін необхідно повторно запустити застосунок та перевірити його роботу. У процесі тестування доцільно змінювати параметри емулятора для перевірки роботи застосунку у різних умовах.

Контрольні питання

1. Що таке емулятор мобільного пристрою?
2. Яке призначення Android Virtual Device?
3. Як створити віртуальний пристрій у Android Studio?
4. Як виконується запуск застосунку в емуляторі?
5. Що таке налагодження програми?
6. Для чого використовуються точки зупину?
7. Яке призначення Logcat?
8. Як аналізувати повідомлення про помилки?
9. Які переваги використання емулятора?
10. Як перевірити роботу застосунку після виправлення помилок?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис процесу запуску застосунку;
– опис процесу налагодження;
– результати тестування;
– виявлені помилки та їх усунення;
– висновки.

Практична робота 7. Створення та налаштування проєкту в Android Studio

Мета роботи – ознайомитися з процесом створення нового мобільного застосунку у середовищі Android Studio, вивчити основні параметри конфігурації проєкту та навчитися виконувати базове налаштування застосунку для подальшої розробки на платформі Android.

Ключові положення

Створення проєкту є початковим етапом розробки мобільного застосунку, під час якого визначаються основні параметри майбутньої програми. До них належать ім’я застосунку, унікальний ідентифікатор (applicationId), мінімальна версія операційної системи та вибір шаблону інтерфейсу.

Середовище Android Studio надає набір шаблонів проєктів, які дозволяють швидко створити структуру застосунку. Найпростішим є шаблон Empty Activity, який містить базову Activity та мінімальний набір ресурсів.

Важливим елементом є система збірки Gradle, яка відповідає за конфігурацію проєкту, керування залежностями та процес компіляції. Файли build.gradle визначають параметри збірки, такі як версія SDK, типи збірки та підключені бібліотеки.

Під час налаштування проєкту необхідно враховувати сумісність із різними версіями Android. Визначення мінімальної версії системи впливає на доступність функціональних можливостей та кількість підтримуваних пристроїв.

Також важливим є налаштування структури пакетів і організації коду. Використання зрозумілої ієрархії пакетів дозволяє забезпечити зручність підтримки та масштабування застосунку.

Додатково можуть бути налаштовані залежності, що підключають сторонні бібліотеки для розширення функціональності застосунку. Це можуть бути бібліотеки для роботи з мережею, базами даних або інтерфейсом користувача.

Таким чином, створення та налаштування проєкту є базовим етапом розробки, який визначає структуру застосунку та забезпечує умови для подальшої реалізації його функціональності.

Практичне завдання

1. Запустити середовище Android Studio.
2. Створити новий проєкт із використанням шаблону Empty Activity.
3. Вказати основні параметри проєкту (ім’я, package name, місце збереження).
4. Обрати мінімальну версію Android для застосунку.
5. Ознайомитися зі створеною структурою проєкту.
6. Відкрити та проаналізувати файли build.gradle.
7. Внести зміни до текстових ресурсів застосунку.
8. Додати простий елемент інтерфейсу та перевірити його відображення.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно запустити середовище Android Studio та обрати опцію створення нового проєкту. У процесі створення слід обрати шаблон Empty Activity та задати основні параметри застосунку.

На першому етапі необхідно уважно заповнити всі параметри проєкту. Особливу увагу слід приділити ідентифікатору застосунку, оскільки він має бути унікальним і використовується під час публікації.

Далі необхідно ознайомитися зі структурою створеного проєкту. Слід звернути увагу на основні каталоги, файли конфігурації та вихідний код.

На наступному етапі необхідно проаналізувати файли build.gradle. Слід визначити параметри, що відповідають за версію SDK, залежності та налаштування збірки.

Після цього необхідно внести прості зміни до застосунку, наприклад змінити текст, що відображається на екрані, або додати новий елемент інтерфейсу. Це дозволить перевірити правильність налаштування проєкту.

У завершальній частині роботи необхідно виконати запуск застосунку та переконатися у його коректній роботі.

Контрольні питання

1. Яке призначення Android Studio?
2. Які параметри задаються під час створення проєкту?
3. Що таке applicationId?
4. Яке значення має мінімальна версія Android?
5. Що таке Gradle?
6. Яке призначення файлів build.gradle?
7. Як організовується структура проєкту?
8. Як додати залежності до проєкту?
9. Як змінити ресурси застосунку?
10. Яке значення має правильне налаштування проєкту?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис процесу створення проєкту;
– аналіз структури проєкту;
– внесені зміни;
– результати виконання;
– висновки.

Практична робота 8. Реалізація роботи з локальною базою даних SQLite

Мета роботи – ознайомитися з принципами роботи локальних баз даних у мобільних застосунках, навчитися створювати, налаштовувати та використовувати базу даних SQLite у середовищі Android Studio для платформи Android.

Ключові положення

SQLite є вбудованою реляційною системою керування базами даних, що використовується у мобільних застосунках для збереження структурованої інформації. Вона не потребує окремого сервера і працює безпосередньо у межах застосунку, що забезпечує високу продуктивність та автономність роботи.

База даних SQLite організована у вигляді таблиць, які складаються зі стовпців і рядків. Кожен запис у таблиці представляє окремий об’єкт, а кожен стовпець визначає атрибут цього об’єкта. Для роботи з базою даних використовується мова SQL, яка дозволяє виконувати операції створення, читання, оновлення та видалення даних.

У Android для роботи з SQLite використовуються спеціальні класи, зокрема SQLiteOpenHelper, який забезпечує створення та оновлення бази даних. Через цей клас реалізується ініціалізація таблиць та керування версіями бази даних.

Взаємодія з базою даних здійснюється через об’єкт SQLiteDatabase, який дозволяє виконувати SQL-запити та отримувати результати у вигляді курсора. Cursor забезпечує послідовний доступ до результатів запиту та використовується для обробки отриманих даних.

Важливим аспектом є забезпечення цілісності даних та коректної обробки операцій. Необхідно враховувати можливі помилки, виконувати перевірку введених даних та використовувати параметризовані запити для уникнення помилок і підвищення безпеки.

Також важливо організувати правильну структуру бази даних, визначити ключі таблиць та типи даних. Це дозволяє забезпечити ефективний доступ до інформації та спростити подальшу обробку даних у застосунку.

Таким чином, використання SQLite є ефективним рішенням для збереження локальних даних у мобільних застосунках, що дозволяє реалізувати функціональність зберігання та обробки інформації без використання зовнішніх сервісів.

Практичне завдання

1. Створити новий або використати існуючий Android-проєкт.
2. Реалізувати клас для роботи з базою даних на основі SQLiteOpenHelper.
3. Створити таблицю у базі даних із декількома полями.
4. Реалізувати операції додавання даних у таблицю.
5. Реалізувати отримання даних із бази даних.
6. Реалізувати оновлення записів.
7. Реалізувати видалення записів.
8. Відобразити дані у інтерфейсі застосунку.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з основами роботи реляційних баз даних та мовою SQL. Особливу увагу слід приділити операціям створення таблиць та виконання запитів.

На першому етапі необхідно створити клас, що наслідується від SQLiteOpenHelper. У цьому класі слід реалізувати методи створення бази даних та її оновлення.

Далі необхідно визначити структуру таблиці. Слід обрати назву таблиці, визначити поля та їх типи, а також встановити первинний ключ.

На наступному етапі необхідно реалізувати операції додавання даних. Слід створити метод, який виконує вставлення запису у таблицю.

Після цього необхідно реалізувати отримання даних із бази даних. Для цього слід виконати SQL-запит та обробити результати за допомогою курсора.

Далі необхідно реалізувати оновлення та видалення записів. Слід забезпечити можливість зміни значень полів та видалення записів за певними умовами.

У завершальній частині роботи необхідно інтегрувати роботу з базою даних у інтерфейс застосунку. Слід відобразити отримані дані у вигляді списку або таблиці.

Контрольні питання

1. Що таке SQLite?
2. Які переваги використання SQLite у мобільних застосунках?
3. Яке призначення класу SQLiteOpenHelper?
4. Що таке таблиця у базі даних?
5. Які основні операції виконуються з даними?
6. Що таке SQL-запит?
7. Яке призначення Cursor?
8. Як реалізується вставлення даних у таблицю?
9. Як виконати оновлення записів?
10. Як забезпечити цілісність даних у базі даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис структури бази даних;
– реалізація основних операцій;
– результати роботи програми;
– висновки.

Практична робота 9. Обробка JSON-даних та асинхронні запити

Мета роботи – ознайомитися з принципами обміну даними у форматі JSON у мобільних застосунках, навчитися виконувати асинхронні мережеві запити та реалізовувати обробку отриманих даних у середовищі Android Studio для платформи Android.

Ключові положення

Сучасні мобільні застосунки часто взаємодіють із віддаленими сервісами для отримання, передавання та оновлення даних. Така взаємодія здійснюється через мережеві запити, у відповідь на які сервер зазвичай повертає дані у форматі JSON. JSON є текстовим форматом подання структурованих даних, що має простий синтаксис, компактну форму та широко використовується у вебсервісах і мобільних застосунках.

Структура JSON може містити об’єкти, масиви, рядки, числа, логічні значення та порожні значення. JSON-об’єкт складається з пар ключ–значення, а JSON-масив містить набір елементів. Такий спосіб подання даних є зручним для передавання інформації про користувачів, товари, повідомлення, налаштування та інші сутності, що використовуються у програмі.

У мобільному застосунку обробка JSON передбачає отримання текстової відповіді від сервера, її розбір та перетворення у внутрішні структури даних програми. Для цього можуть використовуватися вбудовані засоби роботи з JSON або спеціальні бібліотеки. У процесі розбору необхідно правильно визначати типи полів, обробляти вкладені об’єкти та масиви, а також враховувати можливу відсутність окремих значень.

Мережеві запити не можна виконувати у головному потоці застосунку, оскільки це може призвести до блокування інтерфейсу та погіршення взаємодії з користувачем. Саме тому для роботи з мережею використовуються асинхронні запити. Асинхронне виконання дозволяє надсилати запит, очікувати відповідь та обробляти результат без зупинки роботи інтерфейсу.

Під час виконання асинхронних запитів важливо правильно організовувати обробку результату. Після отримання відповіді необхідно перевірити код стану, проаналізувати вміст відповіді, виконати розбір JSON та відобразити результат у застосунку. Крім того, потрібно передбачити обробку помилок, пов’язаних із відсутністю з’єднання, неправильним форматом даних, перевищенням часу очікування або помилками сервера.

Важливим є також розділення логіки мережевої взаємодії та логіки інтерфейсу. Це спрощує підтримку коду, покращує структуру застосунку та дозволяє повторно використовувати модулі, що виконують запити і обробляють відповіді сервера.

Таким чином, обробка JSON-даних та виконання асинхронних запитів є важливою складовою розробки мобільних застосунків, що забезпечує взаємодію з віддаленими джерелами даних і дозволяє створювати динамічні та функціональні програмні системи.

Практичне завдання

1. Створити новий або використати існуючий Android-проєкт.
2. Додати дозвіл на доступ до мережі у конфігурацію застосунку.
3. Реалізувати асинхронний запит до вебсервісу, що повертає дані у форматі JSON.
4. Отримати та вивести текст відповіді сервера.

5. Реалізувати розбір JSON-об’єкта або JSON-масиву.

6. Виділити з отриманих даних окремі поля та зберегти їх у змінних або об’єктах програми.

7. Відобразити отримані дані в інтерфейсі застосунку.

8. Реалізувати обробку помилок під час виконання мережевого запиту та розбору JSON.

9. Перевірити роботу програми для коректної відповіді сервера та для ситуації з помилкою з’єднання.

10. Проаналізувати результати виконання програми.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з основами мережевої взаємодії у мобільних застосунках та структурою формату JSON. Особливу увагу слід приділити тому, як подаються об’єкти, масиви та вкладені елементи.

На першому етапі необхідно підготувати Android-проєкт до роботи з мережею. Для цього слід додати у файл AndroidManifest.xml дозвіл на доступ до Інтернету. Після цього необхідно визначити вебресурс або тестовий сервіс, з якого будуть отримуватися дані у форматі JSON.

Далі необхідно реалізувати асинхронне виконання мережевого запиту. Запит має виконуватися поза головним потоком, щоб не блокувати інтерфейс застосунку. Після отримання відповіді необхідно перевірити, чи успішно завершився запит, і зберегти вміст відповіді для подальшої обробки.

На наступному етапі необхідно реалізувати розбір JSON-даних. Якщо відповідь має вигляд об’єкта, слід виділити потрібні поля за їх ключами. Якщо відповідь містить масив, необхідно організувати послідовну обробку його елементів. У разі наявності вкладених структур потрібно реалізувати доступ до внутрішніх елементів.

Після розбору даних необхідно відобразити отриману інформацію в інтерфейсі застосунку. Це може бути текстове поле, список або інший елемент інтерфейсу. Слід забезпечити зрозуміле подання результатів для користувача.

Особливу увагу необхідно приділити обробці помилок. Слід передбачити ситуації, коли сервер недоступний, відповідь має неправильний формат або окремі поля відсутні. У таких випадках застосунок повинен коректно реагувати на помилку та повідомляти користувача про проблему.

Після завершення реалізації необхідно виконати тестування програми. Слід перевірити правильність отримання та обробки JSON-даних, а також переконатися, що інтерфейс не блокується під час виконання мережевого запиту.

Контрольні питання

1. Що таке JSON і для чого він використовується?
2. Які основні типи даних можуть міститися у JSON?
3. Чим JSON-об’єкт відрізняється від JSON-масиву?
4. Чому мережеві запити не можна виконувати у головному потоці?
5. Що таке асинхронний запит?
6. Які етапи містить обробка відповіді сервера?

7. Для чого потрібен розбір JSON-даних?
8. Які помилки можуть виникати під час мережевої взаємодії?
9. Як відобразити отримані дані в інтерфейсі мобільного застосунку?
10. Яке значення має асинхронна обробка даних у мобільних застосунках?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис реалізації мережевого запиту;
– опис структури отриманих JSON-даних;
– результати розбору та відображення даних;
– результати тестування;
– висновки.

Практична робота 10. Створення інтерфейсу у файлах XML

Мета роботи – ознайомитися з принципами створення інтерфейсу користувача в Android-застосунках за допомогою XML-файлів, навчитися описувати структуру екрана, розміщувати елементи керування та налаштовувати їх властивості у середовищі Android Studio.

Ключові положення

У Android інтерфейс користувача може створюватися програмним шляхом або описуватися у спеціальних XML-файлах. Найпоширенішим підходом є використання XML, оскільки він дає змогу відокремити логіку програми від опису зовнішнього вигляду екрана. Це спрощує проєктування інтерфейсу, його редагування та подальшу підтримку.

XML-файли інтерфейсу зберігаються у каталозі res/layout. Кожен такий файл описує структуру окремого екрана або його частини. У файлі визначаються контейнери компонування та елементи керування, які будуть відображатися на екрані. До таких елементів належать текстові поля, кнопки, поля введення, перемикачі, прапорці та інші компоненти інтерфейсу.

Основою побудови інтерфейсу є використання контейнерів компонування. Вони визначають, як саме розміщуються вкладені елементи на екрані. У Android найчастіше використовуються LinearLayout, RelativeLayout, FrameLayout та ConstraintLayout. Вибір контейнера залежить від структури екрана, складності інтерфейсу та вимог до адаптації під різні пристрої.

Кожен елемент інтерфейсу має набір властивостей, що задаються у XML. До основних властивостей належать ширина, висота, відступи, текст, ідентифікатор, фон, вирівнювання та інші параметри. Частина властивостей визначає зовнішній вигляд елемента, а частина використовується для його подальшого опрацювання у програмному коді.

Важливим є використання ресурсів замість безпосереднього задання значень у файлі макета. Наприклад, текстові значення доцільно зберігати у файлі рядкових ресурсів, а кольори – у відповідних ресурсних файлах. Такий підхід спрощує локалізацію застосунку, повторне використання значень та зміну оформлення.

Під час створення інтерфейсу необхідно враховувати адаптацію до різних розмірів екрана та орієнтацій пристрою. Для цього використовуються гнучкі схеми компонування, обмеження, відступи та ресурсні каталоги для різних конфігурацій.

Таким чином, XML-файли є основним засобом опису інтерфейсу Android-застосунку, що забезпечує структуроване подання елементів екрана, зручність розробки та можливість подальшого масштабування застосунку.

Практичне завдання

1. Створити новий або використати існуючий Android-проєкт.
2. Відкрити файл макета головного екрана у каталозі res/layout.
3. Додати до інтерфейсу текстовий напис, кнопку та поле введення.
4. Налаштувати основні властивості елементів: розміри, текст, відступи та ідентифікатори.
5. Використати один із контейнерів компонування для впорядкування елементів на екрані.
6. Винести текстові значення до файлу рядкових ресурсів.
7. Запустити застосунок та перевірити коректність відображення інтерфейсу.
8. Проаналізувати структуру XML-файлу та призначення його основних елементів.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися зі структурою ресурсів Android-застосунку та призначенням каталогу layout. Особливу увагу слід приділити ролі XML-файлів у побудові інтерфейсу користувача.

На першому етапі необхідно відкрити наявний файл макета або створити новий XML-файл інтерфейсу. Слід визначити, який контейнер компонування буде використано для побудови екрана. Для простих інтерфейсів доцільно використовувати LinearLayout, а для складніших – ConstraintLayout.

Далі необхідно додати до макета основні елементи інтерфейсу. Для кожного елемента слід задати ідентифікатор, ширину, висоту та інші властивості. Особливу увагу слід приділити правильному використанню значень match_parent, wrap_content та відступів.

На наступному етапі необхідно винести текстові значення до файлу strings.xml. Це дозволить уникнути безпосереднього задання тексту у макеті та забезпечить більш правильну організацію ресурсів застосунку.

Після цього необхідно перевірити структуру XML-файлу та переконатися, що всі елементи вкладені коректно, а контейнер компонування описаний правильно. У разі використання ConstraintLayout слід також перевірити наявність необхідних обмежень для кожного елемента.

У завершальній частині роботи необхідно виконати запуск застосунку в емуляторі або на реальному пристрої та перевірити, як створений інтерфейс відображається на екрані. За потреби слід скоригувати розміри, відступи або розташування елементів.

Контрольні питання

1. Для чого в Android використовуються XML-файли інтерфейсу?
2. У якому каталозі зберігаються файли макетів?
3. Яке призначення контейнерів компонування?
4. Які контейнери компонування найчастіше використовуються в Android?
5. Для чого елементам інтерфейсу задають ідентифікатори?
6. Чим відрізняються значення match_parent і wrap_content?
7. Чому текстові значення доцільно зберігати у ресурсах?
8. Які властивості визначають зовнішній вигляд елемента інтерфейсу?
9. Чому важливо враховувати адаптацію інтерфейсу до різних екранів?
10. Яке значення має XML-опис інтерфейсу у процесі розробки Android-застосунку?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис структури XML-файлу;
– характеристика використаних елементів інтерфейсу;
– результати виконання та перевірки;
– висновки.

Практична робота 11. Робота з елементами управління (TextView, Button тощо)

Мета роботи – ознайомитися з основними елементами управління інтерфейсу в застосунках на платформі Android, навчитися використовувати компоненти TextView, Button, EditText, CheckBox та інші, а також реалізовувати обробку подій користувача у середовищі Android Studio.

Ключові положення

Елементи управління є основними складовими інтерфейсу користувача, що забезпечують взаємодію користувача із застосунком. Вони дозволяють вводити дані, виконувати дії, отримувати інформацію та керувати поведінкою програми.

TextView використовується для відображення текстової інформації. Він може містити статичний або динамічний текст, який змінюється під час виконання програми. Button є елементом, що дозволяє ініціювати певну дію, наприклад запуск функції або перехід до іншого екрана.

EditText призначений для введення текстових даних користувачем. Він підтримує різні режими введення, такі як числовий або текстовий. CheckBox використовується для вибору одного або декількох параметрів, а RadioButton – для вибору одного варіанта з групи.

Важливим аспектом є обробка подій, що виникають під час взаємодії користувача з елементами інтерфейсу. Найпоширенішою подією є натискання на кнопку, яке обробляється за допомогою слухачів подій (listeners).

Зв’язок між XML-описом інтерфейсу та програмним кодом здійснюється через ідентифікатори елементів. Після ініціалізації компонентів у коді можна змінювати їх властивості, отримувати введені дані та реагувати на дії користувача.

Також важливим є забезпечення зручності використання інтерфейсу. Елементи повинні бути логічно розташовані, мати зрозумілі підписи та забезпечувати коректну обробку введених даних.

Таким чином, елементи управління є ключовими компонентами інтерфейсу мобільного застосунку, що забезпечують ефективну взаємодію користувача з програмою.

Практичне завдання

1. Створити новий або використати існуючий Android-проєкт.
2. Додати до інтерфейсу елементи TextView, EditText та Button.
3. Реалізувати введення тексту користувачем у поле EditText.
4. Реалізувати обробку натискання кнопки Button.
5. Відобразити введений текст у TextView після натискання кнопки.
6. Додати елементи CheckBox або RadioButton та реалізувати їх обробку.
7. Реалізувати перевірку введених даних.
8. Проаналізувати взаємодію користувача з інтерфейсом.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з основними елементами інтерфейсу Android та їх призначенням. Особливу увагу слід приділити властивостям елементів і способам їх налаштування.

На першому етапі необхідно створити або відкрити XML-файл інтерфейсу та додати основні елементи управління. Для кожного елемента слід задати ідентифікатор та необхідні властивості.

Далі необхідно реалізувати зв’язок між інтерфейсом та програмним кодом. Слід отримати посилання на елементи за їх ідентифікаторами та виконати їх ініціалізацію.

На наступному етапі необхідно реалізувати обробку подій. Слід визначити дію, яка буде виконуватися при натисканні кнопки, та реалізувати відповідний обробник події.

Після цього необхідно реалізувати логіку обробки введених даних. Слід отримати текст із поля введення, виконати його перевірку та відобразити результат у TextView.

Далі необхідно додати елементи CheckBox або RadioButton і реалізувати обробку їх стану. Слід визначити, які параметри обрані користувачем, та використати ці дані у програмі.

У завершальній частині роботи необхідно протестувати застосунок, перевірити коректність обробки подій та зручність використання інтерфейсу.

Контрольні питання

1. Яке призначення елемента TextView?
2. Для чого використовується Button?
3. Яке призначення EditText?
4. Як отримати введені користувачем дані?
5. Що таке обробка подій?

6. Як реалізується обробка натискання кнопки?
7. Яке призначення CheckBox і RadioButton?
8. Як зв’язати XML-інтерфейс із програмним кодом?
9. Як виконати перевірку введених даних?
10. Яке значення мають елементи управління у мобільному застосунку?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис використаних елементів управління;
– реалізація обробки подій;
– результати виконання;
– висновки.

Практична робота 12. Розробка адаптивного користувацького інтерфейсу

Мета роботи – ознайомитися з принципами створення адаптивного інтерфейсу мобільного застосунку на платформі Android, навчитися забезпечувати коректне відображення елементів на різних пристроях та екранних конфігураціях у середовищі Android Studio.

Ключові положення

Адаптивний користувацький інтерфейс є підходом до проєктування, що забезпечує коректне відображення застосунку на пристроях із різними розмірами екранів, роздільною здатністю та орієнтацією. У мобільних застосунках це особливо важливо через різноманітність пристроїв, на яких може працювати програма.

Основою адаптивного інтерфейсу є використання гнучких контейнерів компонування. Найбільш універсальним є ConstraintLayout, який дозволяє задавати взаємозв’язки між елементами та будувати складні інтерфейси без вкладених структур. Такий підхід забезпечує ефективне використання простору екрана та покращує продуктивність.

Для забезпечення адаптивності використовуються відносні розміри елементів, відступи та обмеження. Замість фіксованих значень доцільно застосовувати параметри, що дозволяють елементам змінювати свої розміри залежно від доступного простору.

Важливим інструментом є використання ресурсних каталогів для різних конфігурацій. Наприклад, можна створювати окремі макети для портретної та альбомної орієнтації, а також для пристроїв із великим екраном. Це дозволяє оптимізувати інтерфейс під конкретні умови використання.

Також необхідно враховувати щільність пікселів екрана. Для цього використовуються одиниці вимірювання dp та sp, які забезпечують однаковий вигляд елементів на різних пристроях.

Адаптація інтерфейсу передбачає також коректну роботу з текстом, зображеннями та іншими елементами. Важливо забезпечити читабельність тексту, зручність взаємодії та відсутність перекриття елементів.

Таким чином, адаптивний інтерфейс є необхідною умовою створення якісного мобільного застосунку, що забезпечує зручність використання та коректну роботу на різних пристроях.

Практичне завдання

1. Створити новий або використати існуючий Android-проєкт.
2. Розробити інтерфейс із використанням ConstraintLayout.
3. Розмістити декілька елементів інтерфейсу (TextView, Button, ImageView).
4. Налаштувати обмеження для правильного розташування елементів.
5. Використати відносні розміри та відступи замість фіксованих значень.
6. Створити альтернативний макет для іншої орієнтації екрана.
7. Перевірити відображення інтерфейсу на різних віртуальних пристроях.
8. Проаналізувати адаптивність створеного інтерфейсу.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з принципами адаптивного дизайну та можливостями контейнерів компонування в Android. Особливу увагу слід приділити ConstraintLayout та його механізму обмежень.

На першому етапі необхідно створити XML-файл інтерфейсу та обрати ConstraintLayout як основний контейнер. Слід розмістити у ньому декілька елементів інтерфейсу та задати для них обмеження відносно батьківського контейнера або інших елементів.

Далі необхідно налаштувати розміри елементів. Слід використовувати значення dp для розмірів і відступів та sp для тексту. Необхідно уникати жорстко заданих розмірів, які можуть призвести до некоректного відображення на різних пристроях.

На наступному етапі необхідно створити альтернативний макет для іншої орієнтації екрана або іншого типу пристрою. Для цього слід використати відповідні каталоги ресурсів.

Після цього необхідно виконати тестування інтерфейсу. Слід перевірити відображення застосунку на різних емуляторах із різними розмірами екрана та орієнтаціями.

У процесі тестування необхідно звернути увагу на розташування елементів, їх доступність та зручність взаємодії. За потреби слід скоригувати макет і повторно перевірити результат.

Контрольні питання

1. Що таке адаптивний інтерфейс?
2. Чому важливо враховувати різні розміри екранів?
3. Яке призначення ConstraintLayout?
4. Що таке обмеження (constraints)?
5. Які одиниці вимірювання використовуються в Android?
6. Для чого використовуються ресурсні каталоги?
7. Як створити альтернативний макет інтерфейсу?
8. Як перевірити адаптивність застосунку?
9. Які помилки можуть виникати при створенні інтерфейсу?
10. Яке значення має адаптивність у мобільних застосунках?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис реалізації інтерфейсу;
– результати тестування на різних пристроях;
– аналіз адаптивності;
– висновки.

Рекомендована література

Основна

1. Дворецький М. Л., Нездолій Ю. О., Дворецька С. В., Кандиба І. О. Розробка мобільних застосунків для OS Android : навч. посіб. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2021. 140 с.

2. Радченко К. О. Розроблення мобільних застосунків : конспект лекцій : навч. посіб. – Київ : КПІ ім. Ігоря Сікорського, 2023. 546 с.

3. Власій О. О., Винничук М. Д. Розробка мобільних додатків засобами блочного програмування : навч.-метод. посіб. Івано-Франківськ : Прикарпатський нац. ун-т ім. Василя Стефаника, 2021. 130 с.

4. Ткаченко О. А., Ткаченко К. О., Чайковська О. А. Розробка мобільних додатків під Android : навч. посіб. Київ : Київський нац. ун-т культури і мистецтв, 2017. 274 с.

5. Fazio M. Kotlin and Android Development Featuring Jetpack: Build Better, Safer Android Apps. Pragmatic Bookshelf, 2021. 446 p.

Допоміжна

6. Давидов М. В., Демчук А. Б., Лозинська О. В. Програмне забезпечення мобільних пристроїв. – Київ : Новий світ 2000, 2021. 218 с.

7. Lim G. Beginning Android Development with Kotlin. 2022. 177 p.

8. Wambua M. S. Modern Android 13 Development Cookbook: Over 70 Recipes to Solve Android Development Issues and Create Better Apps with Kotlin and Jetpack Compose. – Birmingham : Packt Publishing, 2023. 322 p.

9. Smyth N. Android Studio 4.0 Development Essentials – Java Edition: Developing Android Apps Using Android Studio, Java and Android Jetpack. Payload Media, 2020. 794 p.

Інформаційні ресурси

10. Національна бібліотека України ім. В. І. Вернадського : веб-сайт. URL: http://www.nbuv.gov.ua

11. Android Developers – офіційна документація та інструменти розробки Android. URL: https://developer.android.com

12. Android Basics with Compose – безкоштовний курс з розробки Android-застосунків від Google. URL: https://developer.android.com/courses/android-basics-compose/course

Навчальне видання

Педяш Володимир Віталійович

ПРОГРАМУВАННЯ ДЛЯ МОБІЛЬНИХ ПЛАТФОРМ

Методичні вказівки до практичних робіт для здобувачів вищої освіти,
які навчаються за спеціальностями галузі «Інформаційні технології»

Українською мовою