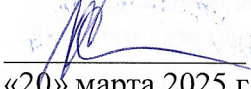


Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

УТВЕРЖДАЮ
Заведующий кафедрой ИСПИ

И.Е. Жигалов
«20» марта 2025 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ
МЕЖДИСЦИПЛИНАРНОГО КУРСА
«ОСНОВЫ МОБИЛЬНОЙ РАЗРАБОТКИ»
В РАМКАХ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ
«ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ РЕСУРСОВ»
09.02.09 Веб-разработка
Разработчик веб приложений

Владимир, 2025

Методические указания к лабораторным работам междисциплинарного курса «Основы мобильной разработки» разработал старший преподаватель кафедры ИСПИ Шамьшев А.А.

Методические указания к лабораторным работам рассмотрены и одобрены на заседании УМК специальности 09.02.09 Веб-разработка протокол № 1 от «10» марта 2025 г.

Председатель УМК специальности _____ И.Е. Жигалов

Методические указания к лабораторным работам рассмотрены и одобрены на заседании кафедры ИСПИ протокол № 7а от «12» марта 2025 г.

Рецензент от работодателя:
руководитель группы обеспечения
качества программного обеспечения
ООО «БСЦ МСК»



_____ С.С. Смирнова

Лабораторная работа №1. Знакомство с Android разработкой.

Цель работы: Познакомиться с основными инструментами и средой разработки Android-приложений, изучить структуру проекта Android и основные компоненты приложения.

Теоретический материал:

Android — это операционная система, разработанная компанией Google, которая используется на миллиардах устройств по всему миру. Она основана на ядре Linux и предоставляет платформу для создания мобильных приложений, которые могут взаимодействовать с аппаратными и программными ресурсами устройства.

Среда разработки Android Studio: Android Studio — это официальная интегрированная среда разработки (IDE) для создания Android-приложений. Она предоставляет инструменты для написания кода, отладки, тестирования и сборки приложений. Основные возможности Android Studio:

- Редактор кода с подсветкой синтаксиса и автодополнением.
- Визуальный редактор макетов (Layout Editor).
- Инструменты для анализа производительности приложения.
- Эмулятор Android для тестирования.

Структура проекта Android: Проект Android состоит из множества файлов и папок. Основные элементы структуры:

1. **Java/Kotlin-код** („src/main/java“): содержит исходный код приложения.
2. **Ресурсы** („src/main/res“): включает файлы интерфейса пользователя (макеты), изображения, строки и другие данные.
3. **AndroidManifest.xml**: файл, в котором описываются основные компоненты приложения и их взаимодействие.

Основные компоненты Android-приложения:

1. **Activity** — экран приложения, представляющий собой отдельный интерфейс.

2. **Fragment** — часть интерфейса, которая может быть встроена в Activity.
3. **Service** — компонент для выполнения фоновых задач.
4. **Broadcast Receiver** — компонент для получения системных уведомлений.
5. **Content Provider** — механизм для обмена данными между приложениями.

Процесс создания Android-приложения:

1. Создание нового проекта в Android Studio.
2. Настройка макета интерфейса пользователя.
3. Написание кода для обработки действий пользователя.
4. Компиляция и запуск приложения на эмуляторе или реальном устройстве.

Порядок выполнения работы:

1. **Установка Android Studio:**
 - Перейдите на официальный сайт Android Studio (<https://developer.android.com/studio>).
 - Скачайте установочный файл для вашей операционной системы.
 - Установите Android Studio, следуя инструкциям мастера установки.
2. **Создание нового проекта:**
 - Запустите Android Studio.
 - Выберите “New Project”.
 - Укажите название проекта, имя пакета (например, com.example.myfirstapp) и выберите минимальную версию SDK.

- Выберите “Empty Activity” в качестве шаблона и завершите настройку.
- 3. Ознакомление с интерфейсом Android Studio:**
- Изучите основные вкладки: “Project”, “Logcat”, “Run”.
 - Откройте файл `activity_main.xml` и изучите визуальный редактор макетов.
 - Изучите `MainActivity.java` (или `MainActivity.kt`), который содержит код логики приложения.
- 4. Запуск приложения на эмуляторе:**
- Откройте AVD Manager и создайте новое виртуальное устройство (например, Pixel 5).
 - Запустите эмулятор и выполните сборку проекта (“Run”).
 - Убедитесь, что приложение запускается без ошибок.
- 5. Изменение интерфейса приложения:**
- Откройте файл `activity_main.xml`.
 - Добавьте элемент `TextView` с текстом “Hello, Android!”.
 - Запустите приложение и проверьте изменения.

Варианты заданий (необходимо выбрать любые три варианта задания):

1. Создайте проект с минимальной версией SDK 21 и измените текст приветствия на “Welcome to Android!”.
2. Измените цвет фона макета на светло-голубой.
3. Добавьте кнопку (`Button`) и настройте её текст на “Click Me!”.
4. Добавьте изображение (`ImageView`) и настройте его на использование локального ресурса.
5. Измените шрифт текста в `TextView` на полужирный.
6. Добавьте второй `TextView` и настройте его текст на ваше имя.

7. Настройте макет так, чтобы элементы располагались по центру экрана.
8. Создайте новый макет и добавьте его в проект.
9. Настройте приложение для отображения даты и времени на главном экране.
10. Добавьте текстовое поле (EditText) для ввода данных.

Содержание отчета:

1. Скриншоты созданного проекта и работающего приложения (используйте ваше имя в названии проекта).
2. Описание структуры проекта и изменений, внесенных в макет.
3. Код, добавленный для выполнения задания.
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие основные компоненты входят в Android-приложение?
2. Что такое Activity и как оно используется?
3. Какова роль файла AndroidManifest.xml?
4. Какие возможности предоставляет Android Studio для разработчиков?
5. Как запустить приложение на эмуляторе и реальном устройстве?

Список рекомендованных источников:

1. Льюис, Ш. , Данн М. Нативная разработка мобильных приложений / Льюис Ш. , Данн М. , пер. с англ. А. Н. Киселева. - Москва : ДМК Пресс, 2020. - 376 с. - ISBN 978-5-97060-845-6. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - [URL:](https://www.studentlibrary.ru/book/ISBN9785970608456.html)

<https://www.studentlibrary.ru/book/ISBN9785970608456.html>

2. Заметти, Ф. Flutter на практике. Прокачиваем навыки мобильной разработки с помощью открытого фреймворка от Google / Заметти Ф. , пер. с англ. А. С. Тищенко. - Москва : ДМК Пресс, 2020. - 328 с. - ISBN 978-5-97060-

808-1. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : <https://www.studentlibrary.ru/book/ISBN9785970608081.html>

3. Пирская, Л. В. Разработка мобильных приложений в среде Android Studio : учебное пособие / Л. В. Пирская. - Ростов-на-Дону : ЮФУ, 2019. - 123 с. - ISBN 978-5-9275-3346-6. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : <https://www.studentlibrary.ru/book/YUFU-2021080534.html>

Лабораторная работа №2. Кроссплатформенная разработка.
Знакомство с React Native

Цель работы: Изучить основы кроссплатформенной разработки приложений с использованием React Native, ознакомиться с процессом создания и запуска простого приложения.

Теоретический материал:

React Native — это популярный фреймворк для разработки мобильных приложений, позволяющий создавать приложения для Android и iOS с использованием JavaScript и React. Главной особенностью React Native является возможность писать один код, который будет работать на обеих платформах, что значительно упрощает процесс разработки.

Основные особенности React Native:

1. **Кроссплатформенность:** Один код для Android и iOS.
2. **Компонентный подход:** Интерфейс строится из компонентов, которые можно повторно использовать.
3. **Горячая перезагрузка (Hot Reload):** Позволяет быстро видеть изменения в коде без необходимости полной перезагрузки приложения.
4. **Доступ к нативным API:** React Native предоставляет доступ к функциям устройства, таким как камера, геолокация и датчики.

Структура проекта React Native:

- **App.js** — главный файл приложения, содержащий основную логику.
- **node_modules** — директория с установленными зависимостями.
- **package.json** — файл с информацией о проекте и его зависимостях.
- **android** и **ios** — директории с нативным кодом для соответствующих платформ.

Пример простого приложения на React Native:

```
import React from 'react';
import { Text, View, StyleSheet } from 'react-native';

const App = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Hello, React Native!</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f0f0f0',
  },
  text: {
    fontSize: 20,
    color: '#333',
  },
});

export default App;
```


Для работы с React Native требуется установить Node.js, npm (или Yarn), Android Studio и React Native CLI.

Порядок выполнения работы:

1. Установка инструментов:

- Установите Node.js с официального сайта (<https://nodejs.org>).
- Установите Expo CLI, выполнив команду: `npm install -g expo-cli`.

2. Создание нового проекта:

- Откройте терминал и выполните команду: `expo init MyFirstApp`.
- Выберите шаблон “Blank” и дождитесь завершения установки.

3. Запуск проекта:

- Перейдите в папку проекта: `cd MyFirstApp`.
- Запустите приложение командой: `expo start`.
- Откройте приложение Expo Go на смартфоне и отсканируйте QR-код.

4. Изучение структуры проекта:

- Откройте файл `App.js` в текстовом редакторе.
- Найдите компонент `View` и изучите его содержимое.

5. Изменение интерфейса:

- Замените текст внутри компонента `Text` на “Hello, React Native!”.
- Добавьте новый компонент `Button` с текстом “Click Me”.

6. Запуск приложения:

- Сохраните изменения в файле `App.js`.
- Проверьте обновления в приложении на смартфоне.

Варианты заданий:

1. Измените текст приветствия на ваше имя.
2. Добавьте кнопку с текстом “Click Me!”.
3. Настройте фон приложения на градиентный цвет.

4. Добавьте изображение в приложение.
5. Создайте новый компонент и отобразите его в приложении.
6. Настройте текстовое поле для ввода данных.
7. Добавьте список (FlatList) с элементами.
8. Измените шрифт текста на курсивный.
9. Настройте приложение для отображения текущего времени.
10. Добавьте обработчик событий для кнопки..

Содержание отчета:

1. Скриншоты интерфейса React Native.
2. Скриншоты структуры проекта.
3. Скриншот работы приложения на эмуляторе.
4. Описание внесенных изменений в проект.
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое React Native и для чего он используется?
2. Опишите процесс создания нового проекта на React Native.
3. Как настроить стили в приложении React Native?
4. Какие компоненты используются для отображения текста и кнопок?
5. Как выполнить запуск приложения на эмуляторе Android?

Лабораторная работа №3. Кроссплатформенная разработка.
Знакомство с Flutter

Цель работы: Изучить основы кроссплатформенной разработки с использованием Flutter, создать первое приложение и освоить основные принципы работы с виджетами и темами.

Теоретический материал:

Flutter — это современный фреймворк от Google для разработки кроссплатформенных мобильных приложений. Он позволяет создавать приложения для Android, iOS, веба и настольных платформ с использованием одного кода. Основой Flutter является язык программирования Dart, который отличается простотой и высокой производительностью.

Основные особенности Flutter:

1. **Кроссплатформенность:** Один код работает на разных платформах.
2. **Горячая перезагрузка (Hot Reload):** Мгновенное обновление интерфейса при изменении кода.
3. **Полная кастомизация:** Широкие возможности для создания уникального дизайна.
4. **Высокая производительность:** Приложения работают быстро благодаря собственному движку рендеринга.

Структура проекта Flutter:

- **main.dart** — главный файл приложения.
- **lib/** — директория с исходным кодом приложения.
- **pubspec.yaml** — файл с настройками проекта и зависимостями.
- **android/** и **ios/** — директории с нативным кодом.

Пример простого приложения на Flutter:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
```

```
        appBar: AppBar(  
          title: Text('Welcome to Flutter'),  
        ),  
        body: Center(  
          child: Text('Hello, Flutter!'),  
        ),  
      ),  
    );  
  }  
}
```

Установка и настройка окружения:

Для работы с Flutter требуется установить SDK Flutter, Android Studio и Dart.

Порядок выполнения работы:

1. Установка Flutter SDK:

- Перейдите на официальный сайт Flutter (<https://flutter.dev>) и скачайте SDK для вашей операционной системы.
- Распакуйте архив и добавьте путь к Flutter в системные переменные.
- Проверьте установку, выполнив команду:

```
flutter doctor
```

2. Установка Android Studio:

- Скачайте и установите Android Studio с официального сайта (<https://developer.android.com/studio>).
- Убедитесь, что установлены Android SDK и эмулятор.

3. Создание нового проекта:

- Откройте терминал и выполните команду:

```
flutter create my_first_app
```

- Перейдите в директорию проекта:

```
cd my_first_app
```

4. Запуск проекта на эмуляторе Android:

- Убедитесь, что эмулятор запущен.
- Выполните команду:

```
flutter run
```

- Убедитесь, что приложение запускается без ошибок.

5. Изменение интерфейса приложения:

- Откройте файл `lib/main.dart` в текстовом редакторе или IDE.
- Измените текст внутри `Text` на “Welcome to Flutter Development!” и сохраните файл.
- Убедитесь, что приложение обновилось автоматически благодаря функции `Hot Reload`.

6. Добавление нового виджета:

- В теле метода `build` добавьте кнопку `ElevatedButton` с текстом “Click Me!” и обработчиком нажатия:

```
ElevatedButton(  
  onPressed: () {  
    print('Button clicked!');  
  },  
  child: Text('Click Me!'),  
)
```

- Проверьте работу кнопки в приложении.

7. Добавление кастомного виджета:

- Создайте новый файл `hello_widget.dart` в папке `lib` и добавьте следующий код:

```
import 'package:flutter/material.dart';  
  
class HelloWidget extends StatelessWidget {
```

```
@override
Widget build(BuildContext context) {
  return Center(
    child: Text('Hello from a custom
widget!'),
  );
}
```

- Импортируйте и используйте виджет в main.dart.

Варианты заданий (необходимо выполнить любые три варианта на выбор):

1. Измените текст приветствия на ваше имя.
2. Добавьте кнопку с текстом “Press Me!” и обработчиком события.
3. Настройте фон приложения на градиентный цвет.
4. Добавьте изображение из интернета в приложение.
5. Создайте кастомный виджет и отобразите его в приложении.
6. Настройте текстовое поле для ввода данных.
7. Добавьте список (ListView) с элементами.
8. Измените стиль текста на жирный и курсивный.
9. Настройте приложение для отображения текущей даты и времени.
10. Добавьте обработчик нажатия для виджета с изображением.

Содержание отчета:

1. Скриншоты интерфейса Flutter.
2. Скриншоты структуры проекта.
3. Скриншот работы приложения на эмуляторе.
4. Описание внесенных изменений в проект.
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое Flutter и для чего он используется?
2. Опишите процесс создания нового проекта на Flutter.
3. Как настроить стили и темы в приложении Flutter?
4. Какие виджеты используются для отображения текста и кнопок?
5. Как выполнить запуск приложения на эмуляторе Android?

Лабораторная работа №4. Использование датчиков окружающей среды в Android приложении

Цель работы: Изучить способы работы с датчиками окружающей среды в Android-приложении, получить навыки сбора и обработки данных с сенсоров устройства.

Теоретический материал:

Современные смартфоны оснащены различными датчиками, которые позволяют собирать информацию об окружающей среде. Среди них наиболее распространены:

1. **Акселерометр** — измеряет ускорение устройства по осям X, Y и Z.
2. **Гироскоп** — определяет угловую скорость вращения устройства.
3. **Магнитометр** — измеряет магнитное поле и используется для создания компаса.
4. **Барометр** — измеряет атмосферное давление.
5. **Датчик освещенности** — определяет уровень освещенности окружающей среды.
6. **Датчик температуры** — измеряет температуру устройства или окружающей среды.

Android предоставляет API для работы с датчиками через класс **SensorManager**, который позволяет:

- Получать список доступных датчиков.

- Регистрировать слушатели для получения данных с датчиков.
- Обработать данные в реальном времени.

Пример кода для работы с акселерометром:

```
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity
implements SensorEventListener {

    private SensorManager sensorManager;
    private Sensor accelerometer;
    private TextView textView;

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

        textView = findViewById(R.id.textView);
        sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
```



```

        accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETE
R);

        if (accelerometer != null) {
            sensorManager.registerListener(this,
accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
        } else {
            textView.setText("Акселерометр
недоступен");
        }
    }

    @Override
    public void onSensorChanged(SensorEvent event)
    {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];
        textView.setText("X: " + x + "\nY: " + y +
"\nZ: " + z);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor,
int accuracy) {
        // Обработка изменения точности сенсора
    }

    @Override

```

```
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}

@Override
protected void onResume() {
    super.onResume();
    sensorManager.registerListener(this,
accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
}
}
```

Порядок выполнения работы:

1. Подготовка проекта:

- Создайте новый проект в Android Studio с пустым Activity.
- Настройте интерфейс, добавив элемент TextView для отображения данных с датчика.

2. Получение доступа к датчику:

- В классе MainActivity создайте объект SensorManager для работы с сенсорами.
- Получите экземпляр нужного датчика, например, акселерометра, с помощью метода getDefaultSensor.

3. Регистрация слушателя:

- Зарегистрируйте слушатель событий сенсора, используя метод registerListener.
- Укажите частоту обновления данных (например, SensorManager.SENSOR_DELAY_NORMAL).

4. Обработка данных:

- Реализуйте метод `onSensorChanged` для получения данных с датчика.
- Отобразите данные в текстовом поле интерфейса.

5. Обработка ошибок:

- Проверьте, доступен ли датчик на устройстве. Если датчик отсутствует, выведите сообщение об ошибке.

6. Добавление функциональности:

- Реализуйте возможность выбора между несколькими датчиками (акселерометр, гироскоп, магнитометр).
- Добавьте кнопку для остановки сбора данных.

7. Тестирование:

- Запустите приложение на реальном устройстве или эмуляторе.
- Проверьте корректность отображения данных при движении устройства.

Варианты заданий:

1. Реализуйте приложение для работы с акселерометром.
2. Добавьте обработку данных гироскопа.
3. Реализуйте приложение для создания компаса с использованием магнитометра.
4. Создайте барометрическое приложение для измерения атмосферного давления.
5. Реализуйте измерение уровня освещенности с помощью датчика освещенности.
6. Добавьте возможность записи данных сенсоров в файл.
7. Реализуйте графическое отображение данных акселерометра (график движения).
8. Добавьте возможность переключения между датчиками через интерфейс.

9. Реализуйте приложение, которое определяет положение устройства (в горизонтальном или вертикальном положении).
10. Создайте приложение, которое реагирует на встряску устройства.

Содержание отчета:

1. Скриншоты интерфейса приложения.
2. Описание используемых датчиков и их назначения.
3. Код обработки данных с датчиков.
4. Описание внесенных изменений в приложение.
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие типы датчиков поддерживаются в Android?
2. Как получить доступ к датчику через SensorManager?
3. Что такое SensorEvent и как с ним работать?
4. Как реализовать обработку данных с нескольких датчиков одновременно?
5. Какие ограничения существуют при работе с сенсорами?

Лабораторная работа №5. Использование камеры в Android приложении

Цель работы: Изучить работу с камерой в Android-приложении, освоить методы захвата изображений и видео, а также сохранения данных на устройстве.

Теоретический материал:

Камера является одним из ключевых компонентов мобильных устройств, обеспечивая захват изображений и видео. Android предоставляет

разработчикам API для работы с камерой, позволяя интегрировать её функционал в приложения.

Основные классы и методы работы с камерой:

1. Intent для вызова камеры:

- Для запуска стандартного приложения камеры используется Intent. Это позволяет сделать процесс захвата изображения или видео максимально простым.

- Пример вызова камеры:

```
Intent intent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent,
REQUEST_IMAGE_CAPTURE);
```

2. Работа с CameraX:

- CameraX — это современная библиотека для работы с камерой, упрощающая интеграцию её функционала.

- Пример настройки CameraX:

```
val cameraProviderFuture =
ProcessCameraProvider.getInstance(context)
cameraProviderFuture.addListener({
    val cameraProvider =
cameraProviderFuture.get()
    val preview = Preview.Builder().build().also
{
it.setSurfaceProvider(viewFinder.surfaceProvider)
}
    val cameraSelector =
CameraSelector.DEFAULT_BACK_CAMERA
    cameraProvider.bindToLifecycle(this,
cameraSelector, preview)
```

```
} , ContextCompat.getMainExecutor(context))
```

3. Сохранение данных:

- Изображения и видео можно сохранять в хранилище устройства или в базу данных.
- Пример сохранения изображения:

```
File file = new  
File(getExternalFilesDir(Environment.DIRECTORY_P  
CTURES), "photo.jpg");
```

4. Запросы разрешений:

- Для работы с камерой требуется запрашивать разрешения у пользователя.
- Пример проверки разрешений:

Порядок выполнения работы:

1. Подготовка проекта:

- Создайте новый проект в Android Studio.
- Добавьте необходимые разрешения в файл AndroidManifest.xml:

```
<uses-permission  
android:name="android.permission.CAMERA" />  
<uses-permission  
android:name="android.permission.WRITE_EXTERNAL  
_STORAGE" />
```

2. Создание интерфейса:

- Добавьте в файл activity_main.xml кнопку для запуска камеры и ImageView для отображения изображения:

```
<Button  
android:id="@+id/btnCapture"  
android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="Capture Image" />
```

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

3. Запуск камеры:

- Настройте Intent для вызова камеры в MainActivity.java:

```
btnCapture.setOnClickListener(v -> {
    Intent intent = new
    Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(intent,
    REQUEST_IMAGE_CAPTURE);
});
```

4. Обработка результата:

- Реализуйте метод onActivityResult для обработки изображения:

```
@Override
protected void onActivityResult(int requestCode,
int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode,
data);
    if (requestCode == REQUEST_IMAGE_CAPTURE &&
resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        imageView.setImageBitmap(imageBitmap);
    }
}
```

5. Сохранение изображения:

- Реализуйте сохранение изображения на устройство:

```
String savedImageURL =  
MediaStore.Images.Media.insertImage(  
getContentResolver(), imageBitmap, "My Photo",  
"Photo captured by camera"  
);
```

Варианты заданий:

1. Захват изображения и сохранение его на устройство.
2. Реализация функции записи видео.
3. Добавление кнопки для переключения между фронтальной и задней камерами.
4. Реализация предварительного просмотра камеры в TextureView.
5. Добавление возможности изменять разрешение изображения.
6. Сохранение изображений в выбранную пользователем директорию.
7. Настройка таймера для захвата изображения.
8. Добавление фильтров к изображению после захвата.
9. Интеграция CameraX для работы с предварительным просмотром и захватом.
10. Реализация функции автофокуса и настройки яркости.

Содержание отчета:

1. Скриншоты интерфейса приложения.
2. Скриншоты работы приложения (захват изображения, видео).
3. Код основных методов.
4. Описание работы с разрешениями.
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какой Intent используется для вызова камеры?
2. Что такое CameraX, и какие его преимущества?
3. Как проверить и запросить разрешения для работы с камерой?
4. Как сохранить изображение на устройство?
5. Какие методы используются для обработки результата работы камеры?

Лабораторная работа №6. Работа с Canvas в Android приложении

Цель работы: Изучить возможности класса Canvas для рисования на экране Android-устройства, освоить базовые методы работы с графикой и реализовать интерактивное приложение с использованием Canvas.

Теоретический материал:

Canvas — это мощный инструмент для рисования графических элементов в Android. Он предоставляет методы для создания и отображения примитивов, таких как линии, круги, прямоугольники, а также для работы с изображениями и текстом.

Основные методы класса Canvas:

1. **drawLine(float startX, float startY, float stopX, float stopY, Paint paint):** рисует линию между двумя точками.
2. **drawRect(float left, float top, float right, float bottom, Paint paint):** рисует прямоугольник.
3. **drawCircle(float cx, float cy, float radius, Paint paint):** рисует круг.
4. **drawText(String text, float x, float y, Paint paint):** отображает текст.
5. **drawBitmap(Bitmap bitmap, float left, float top, Paint paint):** отображает изображение.

Класс Paint:

Для настройки стиля рисования используется объект класса Paint. Он позволяет задавать цвет, толщину линий, шрифт и другие параметры.

Пример создания Paint:

```
Paint paint = new Paint();
paint.setColor(Color.RED);
paint.setStrokeWidth(5);
paint.setStyle(Paint.Style.FILL);
```

Создание пользовательского представления (Custom View):

Для работы с Canvas обычно создаётся пользовательский класс, наследующий View, и переопределяется метод onDraw(Canvas canvas).

Пример:

```
public class CustomView extends View {
    public CustomView(Context context) {
        super(context);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        Paint paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setStrokeWidth(10);

        // Рисуем линию
        canvas.drawLine(50, 50, 200, 200, paint);

        // Рисуем круг
        paint.setColor(Color.GREEN);
        canvas.drawCircle(150, 150, 50, paint);

        // Рисуем текст
```

```

        paint.setColor(Color.BLACK);
        paint.setTextSize(40);
        canvas.drawText("Hello, Canvas!", 100, 300,
paint);
    }
}

```

Для отображения CustomView в Activity необходимо добавить его в макет или установить как основное представление:

```
setContentView(new CustomView(this));
```

Для создания интерактивных элементов можно переопределить метод onTouchEvent(MotionEvent event) и обрабатывать действия пользователя (например, рисование линий по касанию).

Пример:

```

@Override
public boolean onTouchEvent(MotionEvent event) {
    float x = event.getX();
    float y = event.getY();

    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            // Начало касания
            break;
        case MotionEvent.ACTION_MOVE:
            // Движение пальца
            break;
        case MotionEvent.ACTION_UP:
            // Конец касания

```

```
        break;
    }
    return true;
}
```

Порядок выполнения работы:

Порядок выполнения работы:

1. Создание нового проекта:

- Откройте Android Studio и создайте новый проект с пустой Activity.
- Убедитесь, что выбрана минимальная версия SDK, совместимая с вашим устройством.

2. Создание пользовательского класса для рисования:

- В папке java создайте новый класс CustomView, который наследует View.
- Переопределите метод onDraw(Canvas canvas) и добавьте код для рисования примитивов: линий, прямоугольников, кругов и текста.

3. Настройка Paint:

- Создайте несколько объектов Paint для разных элементов (например, линии, текста, круга).
- Задайте цвета и другие параметры стиля.

4. Добавление пользовательского представления в Activity:

- В методе onCreate установите CustomView как основное представление:

```
setContentView(new CustomView(this));
```

5. Добавление интерактивности:

- Переопределите метод onTouchEvent в CustomView.

- Реализуйте обработку касаний для рисования линий или других фигур в местах касания.

6. Тестирование приложения:

- Запустите приложение на эмуляторе или устройстве.
- Проверьте, отображаются ли нарисованные элементы и работает ли интерактивность.

7. Усложнение функционала (по желанию):

- Добавьте возможность изменения цвета или стиля рисования через кнопки.
- Реализуйте очистку экрана.

Варианты заданий:

1. Нарисуйте прямоугольник, заполняющий половину экрана.
2. Реализуйте рисование линии от точки касания.
3. Добавьте возможность рисования кругов по двойному нажатию.
4. Реализуйте отображение координат касания на экране.
5. Создайте функцию для изменения цвета линии через кнопки.
6. Нарисуйте сетку из линий на экране.
7. Добавьте возможность рисования прямоугольников с градиентной заливкой.
8. Реализуйте стирание нарисованных элементов по длинному нажатию.
9. Добавьте кнопку для очистки экрана.
10. Создайте приложение, которое отображает нарисованное пользователем изображение в виде растрового файла.

Содержание отчета:

1. Скриншоты интерфейса приложения с нарисованными элементами.
2. Код класса CustomView.
3. Описание реализации интерактивности.

4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Для чего используется класс Canvas в Android?
2. Какой метод отвечает за рисование на Canvas?
3. Какие параметры можно настроить с помощью класса Paint?
4. Как обрабатывать события касания в пользовательском представлении?
5. Как добавить пользовательское представление в Activity?

Лабораторная работа №7. Распознавание жестов в Android-приложении

Цель работы: Научиться работать с системой распознавания жестов в Android-приложениях, реализовать базовые жесты (такие как нажатие, свайп, масштабирование) и изучить возможности использования GestureDetector и ScaleGestureDetector.

Теоретический материал:

Распознавание жестов — это процесс интерпретации пользовательских взаимодействий с экраном устройства. Android предоставляет встроенные инструменты для работы с жестами, такие как классы GestureDetector и ScaleGestureDetector.

Основные жесты:

1. **Нажатие (Tap):** Однократное быстрое касание экрана.
2. **Длительное нажатие (Long Press):** Удержание пальца на экране в течение заданного времени.
3. **Свайп (Swipe):** Быстрое перемещение пальца по экрану.
4. **Масштабирование (Pinch/Zoom):** Сведение или разведение двух пальцев на экране.

Классы для работы с жестами:

- **GestureDetector:** Используется для распознавания простых жестов, таких как нажатие и свайп.
- **ScaleGestureDetector:** Предназначен для обработки жестов масштабирования.

Пример использования GestureDetector:

```
import android.view.GestureDetector;
import android.view.MotionEvent;

public class GestureListener extends
GestureDetector.SimpleOnGestureListener {
    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        // Обработка одиночного нажатия
        return true;
    }

    @Override
    public void onLongPress(MotionEvent e) {
        // Обработка длительного нажатия
    }

    @Override
    public boolean onFling(MotionEvent e1,
MotionEvent e2, float velocityX, float velocityY) {
        // Обработка свайпа
        return true;
    }
}
```

Пример использования ScaleGestureDetector:

```
import android.view.ScaleGestureDetector;

public class ScaleListener extends
ScaleGestureDetector.SimpleOnScaleGestureListener {
    @Override
    public boolean onScale(ScaleGestureDetector
detector) {
        float scaleFactor =
detector.getScaleFactor();
        // Обработка изменения масштаба
        return true;
    }
}
```

Регистрация слушателей жестов: Для работы с жестами необходимо подключить `GestureDetector` или `ScaleGestureDetector` к обработчику событий касания (`onTouchEvent`).

Порядок выполнения работы:

1. Создание проекта:

- Создайте новый проект в Android Studio с пустым активити.
- Убедитесь, что проект успешно компилируется и запускается на эмуляторе или устройстве.

2. Добавление `GestureDetector`:

- В `MainActivity` создайте экземпляр `GestureDetector` и подключите его к обработчику событий `onTouchEvent`.
- Реализуйте обработку одиночного нажатия и свайпа.

3. Добавление обработки масштабирования:

- Создайте экземпляр `ScaleGestureDetector` и добавьте его в `onTouchEvent`.

- Реализуйте изменение размера объекта (например, текста или изображения) при масштабировании.

4. Создание пользовательского View:

- Создайте новый класс, наследующий View, и переопределите метод onDraw для отображения графических элементов.
- Включите поддержку жестов в этом View.

5. Обработка нескольких жестов:

- Реализуйте одновременную поддержку нескольких жестов (например, масштабирование и свайп).
- Используйте флаги или состояния для переключения между типами жестов.

6. Тестирование приложения:

- Запустите приложение и протестируйте работу всех реализованных жестов.
- Убедитесь, что интерфейс корректно реагирует на пользовательские действия.

Варианты заданий:

1. Реализуйте обработку одиночного нажатия и отображение сообщения на экране.
2. Добавьте поддержку двойного нажатия для увеличения изображения.
3. Реализуйте обработку длительного нажатия для изменения цвета фона.
4. Создайте пользовательский View, который перемещается при свайпе.
5. Добавьте поддержку масштабирования текста.
6. Реализуйте жест для сброса масштаба (двойное касание двумя пальцами).
7. Настройте жест для поворота объекта (например, изображения).
8. Реализуйте жест для удаления объекта (свайп влево).

9. Добавьте поддержку комбинации жестов (например, свайп и масштабирование).
10. Реализуйте обработку касания несколькими пальцами одновременно.

Содержание отчета:

1. Скриншоты интерфейса приложения.
2. Описание реализованных жестов и их логики.
3. Фрагменты кода с комментариями.
4. Описание проблем, возникших в процессе выполнения работы, и их решений.
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие классы используются для распознавания жестов в Android?
2. Как подключить GestureDetector к обработчику событий onTouchEvent?
3. Опишите процесс работы с ScaleGestureDetector.
4. Какие жесты можно реализовать с помощью GestureDetector?
5. Как обработать одновременное выполнение нескольких жестов?

Лабораторная работа №8. Работа с локальной базой SQLite в Android приложении

Цель работы: Изучить основы работы с локальной базой данных SQLite в Android приложении. Научиться создавать, изменять и удалять записи в базе данных, а также отображать данные в пользовательском интерфейсе.

Теоретический материал:

SQLite — это встроенная реляционная база данных, которая является частью Android SDK. Она предоставляет простой способ хранения

структурированных данных внутри приложения. SQLite не требует настройки сервера и работает напрямую с файлами на устройстве.

Основные особенности SQLite:

1. Легковесность: база данных хранится в одном файле.
2. Поддержка стандартного SQL: можно использовать привычные SQL-запросы.
3. Встроенность: SQLite интегрирована в Android и не требует дополнительных библиотек.

Ключевые классы для работы с SQLite:

- **SQLiteOpenHelper**: абстрактный класс для управления созданием и обновлением базы данных.
- **SQLiteDatabase**: класс для выполнения операций с базой данных (запросы, добавление, удаление, обновление).

Пример создания базы данных:

```
public class DatabaseHelper extends
SQLiteOpenHelper {

    private static final String DATABASE_NAME =
"students.db";

    private static final int DATABASE_VERSION = 1;

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null,
DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String createTable = "CREATE TABLE students
(" +
```

```

        "id INTEGER PRIMARY
KEY AUTOINCREMENT, " +
        "name TEXT, " +
        "grade INTEGER);";
        db.execSQL(createTable);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS
students");
        onCreate(db);
    }
}

```

Пример добавления записи:

```

public void addStudent(String name, int grade) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", name);
    values.put("grade", grade);
    db.insert("students", null, values);
    db.close();
}

```

Пример чтения данных:

```

public List<String> getAllStudents() {
    List<String> students = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT * FROM
students", null);
}

```

```

        if (cursor.moveToFirst()) {
            do {
                String student =
cursor.getString(cursor.getColumnIndex("name")) +
                ", Grade: " +
cursor.getInt(cursor.getColumnIndex("grade"));
                students.add(student);
            } while (cursor.moveToNext());
        }
        cursor.close();
        db.close();
        return students;
    }
}

```

Порядок выполнения работы:

1. Создание проекта:

- Создайте новый проект в Android Studio с пустым Activity.

2. Настройка базы данных:

- Создайте класс, наследующий SQLiteOpenHelper, и реализуйте методы onCreate и onUpgrade.
- Определите таблицу для хранения данных (например, список студентов с их оценками).

3. Добавление данных:

- Реализуйте метод для добавления новых записей в базу данных.
- Добавьте кнопку в интерфейс приложения, при нажатии на которую данные будут добавляться в базу.

4. Отображение данных:

- Реализуйте метод для чтения данных из базы и отображения их в ListView или RecyclerView.

- Настройте адаптер для отображения списка данных.

5. Обновление данных:

- Добавьте возможность редактировать существующие записи (например, изменять оценки).
- Реализуйте метод для обновления данных в базе.

6. Удаление данных:

- Добавьте возможность удалять записи из базы данных.
- Реализуйте метод для удаления записи по её идентификатору.

7. Тестирование:

- Проверьте работу приложения: добавьте, измените, удалите и отобразите данные.

Варианты заданий:

1. Создать базу данных для учёта книг в библиотеке (название, автор, год издания).
2. Реализовать приложение для хранения списка покупок (название товара, количество, цена).
3. Создать базу данных для учёта посещаемости студентов (имя, дата, статус).
4. Разработать приложение для учёта тренировок (тип тренировки, дата, длительность).
5. Реализовать базу данных для учёта расходов (категория, сумма, дата).
6. Создать приложение для хранения списка фильмов (название, жанр, рейтинг).
7. Реализовать базу данных для учёта задач (название, описание, статус выполнения).
8. Разработать приложение для учёта контактов (имя, телефон, email).
9. Создать базу данных для хранения информации о путешествиях (страна, город, дата поездки).

10. Реализовать приложение для учёта медицинских записей (имя пациента, диагноз, дата приёма).

Содержание отчета:

1. Скриншоты интерфейса приложения.
2. Скриншоты структуры базы данных.
3. Примеры SQL-запросов, использованных в работе.
4. Описание реализованных функций.
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое SQLite и какие преимущества она предоставляет?
2. Как создаётся и управляется база данных в Android?
3. Какие методы предоставляет класс SQLiteOpenHelper?
4. Как выполнить добавление, изменение и удаление данных в SQLite?
5. Как организовать отображение данных из базы в пользовательском интерфейсе?

Лабораторная работа №9. Взаимодействие с серверной частью в Android приложении

Цель работы: Изучить основы взаимодействия Android приложения с серверной частью, научиться выполнять HTTP-запросы, обрабатывать ответы сервера и использовать библиотеки для работы с REST API.

Теоретический материал:

Современные Android приложения часто взаимодействуют с сервером для получения или отправки данных. Для этого используются протоколы

HTTP/HTTPS и REST API. REST (Representational State Transfer) — это архитектурный стиль взаимодействия, основанный на стандартах HTTP.

Основные понятия:

- HTTP-запросы:
- GET — для получения данных с сервера.
- POST — для отправки данных на сервер.
- PUT — для обновления данных на сервере.
- DELETE — для удаления данных.

JSON (JavaScript Object Notation):

- Формат обмена данными между клиентом и сервером.

- Пример JSON-объекта:

```
{  
  "id": 1,  
  "name": "Example",  
  "active": true  
}
```

Библиотеки для работы с сетью:

- Retrofit: Высокоуровневая библиотека для работы с REST API.

- OkHttp: Низкоуровневая библиотека для выполнения HTTP-запросов.
- Volley: Библиотека для работы с сетью от Google.

Пример работы с Retrofit:

1. Добавьте зависимость в build.gradle:

```
implementation
'com.squareup.retrofit2:retrofit:2.9.0'
implementation
'com.squareup.retrofit2:converter-gson:2.9.0'
```

2. Создайте интерфейс API:

```
import retrofit2.Call;
import retrofit2.http.GET;

public interface ApiService {
    @GET("posts")
    Call<List<Post>> getPosts();
}
```

3. Настройте Retrofit:

```
Retrofit retrofit = new Retrofit.Builder()

.baseUrl("https://jsonplaceholder.typicode.com/"
)
```

```
.addConverterFactory(GsonConverterFactory.create  
())
```

```
.build();
```

```
ApiService apiService =  
retrofit.create(ApiService.class);
```

4. Выполните запрос:

```
apiService.getPosts().enqueue(new  
Callback<List<Post>>() {  
    @Override  
    public void onResponse(Call<List<Post>>  
call, Response<List<Post>> response) {  
        if (response.isSuccessful()) {  
            List<Post> posts = response.body();  
            // Обработка данных  
        }  
    }  
  
    @Override  
    public void onFailure(Call<List<Post>> call,  
Throwable t) {  
        t.printStackTrace();  
    }  
});
```

Порядок выполнения работы:

1. Подготовка проекта:

- Создайте новый проект в Android Studio.

- Настройте подключение к интернету, добавив разрешение в файл AndroidManifest.xml:

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

2. Установка зависимостей:

- Откройте файл build.gradle и добавьте зависимости для Retrofit и Gson.

3. Создание интерфейса API:

- Определите методы API для выполнения запросов (например, получение списка пользователей или публикаций).

4. Настройка Retrofit:

- Укажите базовый URL сервера.
- Добавьте конвертер для работы с JSON.

5. Выполнение запросов:

- Создайте кнопку в интерфейсе приложения для запуска запроса.
- Обработайте ответ сервера и отобразите данные в приложении (например, в RecyclerView).

6. Обработка ошибок:

- Добавьте обработку ошибок, таких как отсутствие соединения с сервером или неверный ответ.

7. Тестирование:

- Проверьте работу приложения на эмуляторе или реальном устройстве.

Варианты заданий:

1. Получить список пользователей с сервера и отобразить их в списке.
2. Отправить данные формы (например, имя и email) на сервер.
3. Реализовать авторизацию через API (используя POST-запрос).

4. Получить изображение с сервера и отобразить его в приложении.
5. Создать функцию обновления данных пользователя на сервере.
6. Реализовать удаление записи через DELETE-запрос.
7. Добавить обработку ошибок (например, при отсутствии интернета).
8. Реализовать кеширование данных.
9. Использовать токен авторизации для выполнения запросов.
10. Подключить и использовать сторонний API (например, OpenWeatherMap).

Содержание отчета:

1. Скриншоты интерфейса приложения.
2. Скриншоты выполнения запросов и ответов сервера.
3. Описание структуры API и используемых методов.
4. Код основных компонентов (API-интерфейс, обработка запросов).
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое REST API и как оно используется в Android приложениях?
2. Какие HTTP-методы используются для взаимодействия с сервером?
3. Как настроить библиотеку Retrofit для работы с сервером?
4. Что такое JSON и как его использовать в Android?
5. Как обрабатывать ошибки при выполнении HTTP-запросов?

СПИСОК ЛИТЕРАТУРЫ

1. Заметти, Ф. Flutter на практике. Прокачиваем навыки мобильной разработки с помощью открытого фреймворка от Google / Заметти Ф. , пер. с англ. А. С. Тищенко. - М : ДМК Пресс. URL: <https://e.lanbook.com/book/355550> (дата обращения: 10.03.2025).
2. Льюис, Ш. , Данн М. Нативная разработка мобильных приложений / Льюис Ш. , Данн М. , пер. с англ. А. Н. Киселева. - М : ДМК Пресс. URL : <https://www.studentlibrary.ru/book/ISBN9785970608456.html> (дата обращения: 10.03.2025).
3. Пирская, Л. В. Разработка мобильных приложений в среде Android Studio : учебное пособие / Л. В. Пирская. – Ростов н/Д : ЮФУ. URL : <https://www.studentlibrary.ru/book/YUFU-2021080534.html> (дата обращения: 10.03.2025).