Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Программирование мобильных приложений

ОТЧЁТ
по лабораторной работе №2
«ЛАБОРАТОРНАЯ РАБОТА 2»

Выполнил: студент гр. 253501
Щур М. А.

Проверил: старший преподаватель
кафедры информатики
Владымцев В.Д.

Минск 2025

1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является разработка мобильного приложения для отслеживания различных периодов тренировок, позволяет пользователю создавать, редактировать и управлять последовательностями. В процессе работы необходимо реализовать хранение данных (с использованием GSON или SQLite), редактирование, удалеине, создание новых последовательностей, управление настройками, а также таймером через фоновые службы.

2 ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было реализовано мобильное приложение-таймер для тренировок.

В качестве базы данных была использована sqlite совместно с библиотекой room, которая позволяет легко и эффективно взаимодействовать с базой данных.

Был реализован класс таймера, определяющий основную логику приложения: отсчет времени, переход на новый этап, завершение тренировки.

Интерфейс приложения разбит на 5 основных экранов: главный экран со списком всех тренировок, экран таймера и текущей тренировки, экран создания тренировки, экран редактирования тренировки и экран настроек.

Была использована mvvm архитектура приложения для удобного масштабирования и редактирования кода.

3 ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Data Storage относится к способам хранения данных внутри приложения. Для этого могут использоваться локальные базы данных, такие как SQLite, которые предназначены для работы со структурированными данными, или файловые системы для хранения неструктурированных данных, таких как изображения или текстовые файлы. Кроме того, популярным решением является использование библиотек для сериализации данных, например, GSON, позволяющих сохранять и загружать объекты в формате JSON. Например:

```
val sharedPreferences=getSharedPreferences("My», Context.MODE_PRIVATE)
val editor = sharedPreferences.edit()

// Сохранение данных
editor.putString("username", "JohnDoe")
editor.putInt("age", 30)
editor.apply()

// Чтение данных
val username = sharedPreferences.getString("username", null)
val age = sharedPreferences.getInt("age", 0)
```

App Architecture определяет организацию структуры приложения и взаимодействие между его компонентами. Использование продуманной архитектуры, например, MVVM (Model-View-ViewModel), способствует разделению логики приложения на отдельные слои, что упрощает поддержку, тестирование и масштабирование. Архитектура также учитывает управление жизненным циклом компонентов, таких как Activity и Fragment, что помогает избежать утечек памяти и обеспечивает стабильную работу приложения:

- Model: Содержит бизнес-логику и данные приложения.
- View: Отвечает за отображение данных и взаимодействие с пользователем.
- ViewModel: Посредник между моделью и представлением. ViewModel подготавливает данные для отображения и обрабатывает пользовательские действия.

Styling отвечает за визуальное оформление приложения, включая использование цветов, шрифтов, отступов и анимаций. В Android для стилизации применяются стили и темы, которые позволяют централизованно управлять внешним видом приложения. Это помогает добиться единообразного, эстетичного интерфейса, соответствующего бренду и ожиданиям пользователей.

Services — это компоненты Android, предназначенные для выполнения длительных операций в фоновом режиме, таких как воспроизведение музыки или отслеживание местоположения. В приложении-таймере службы могут использоваться для управления таймером и отправки уведомлений, даже если приложение свернуто. Ключевой особенностью является оптимизация работы

служб для снижения расхода батареи и соблюдение ограничений операционной системы, накладываемых на фоновые процессы. Например:

```
class MyService : Service() {

    override fun onBind(intent: Intent?): IBinder? {
        return null // В данном случае сервис не привязывается
    }

      override fun onStartCommand(intent: Intent?, flags: Int, startId:
Int): Int {
        // Выполнение фоновой работы
        return START_STICKY
    }

    override fun onDestroy() {
        super.onDestroy()
        // Очистка ресурсов
    }
}
```

Preferences используются для хранения пользовательских настроек, таких как выбор темы, языка или параметров таймера. Библиотека AndroidX Preference предоставляет удобные инструменты для создания и управления экранами настроек. Эти настройки позволяют персонализировать приложение в соответствии с предпочтениями каждого пользователя. Особенностью является возможность синхронизации настроек между устройствами, если приложение поддерживает такую функцию, что делает использование приложения более удобным. Например:

```
class MainActivity : AppCompatActivity() {

    private lateinit var sharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

                sharedPreferences = getSharedPreferences("MyPreferences",
Context.MODE_PRIVATE)

        // Сохранение данных
        val editor = sharedPreferences.edit()
        editor.putString("username", "JohnDoe")
        editor.apply()

        // Чтение данных
                val username = sharedPreferences.getString("username",
"default_user")
                        Toast.makeText(this, "Username: $username",
Toast.LENGTH_SHORT).show()
    }
}
```

ВЫВОДЫ

В ходе выполнения лабораторной работы было разработано мобильное приложение-таймер для тренировок, реализующее основной функционал для удобного управления тренировочным процессом.

Была использована база данных SQLite совместно с библиотекой Room обеспечило удобное и эффективное управление данными, что значительно упростило взаимодействие с хранилищем.

Основная логика приложения была реализована через класс таймера, который отвечает за ключевые функции: отсчет времени, автоматический переход между этапами тренировки и корректное завершение тренировочного процесса.

Интерфейс приложения был разделен на пять основных экранов, что обеспечивает удобную навигацию и интуитивное взаимодействие с функционалом приложения. Такая структура включает: главный экран со списком тренировок; экран таймера для отображения текущей тренировки; экран создания новых тренировок экран редактирования существующих тренировок; экран настроек.