Geekbrains

Создание социальной сети с помощью языка программирования Kotlin

Программа: Android разработчик

Специализация: android разработчик

Васина Анна Вадимовна

Г. Петрозаводск

2025

Оглавление

Введение

В современном мире коммуникация на дальних расстояниях уже является частью нашей привычной жизни, люди заводят знакомства в социальных сетях, ведут бизнес и просто делятся моментами из своей жизни, и роль приложений в нашей жизни, которые способны выполнять эти задачи, увеличивается с каждым днем.

Целью данного проекта является создание социальной сети, позволяющей пользователям совершать основные действия, для которых они были созданы: делиться мыслями, переписываться, подписываться на других людей, чтобы видеть их публикации.

Разработка приложения будет выполнена с помощью языка kotlin, он позиционирует себя как наиболее предпочтительный для мобильной разработки и стремительно набирает популярность из-за преимуществ, добавленных при создании этого языка.

Для выполнения этого проекта были поставлены следующие задачи:
1. Изучить язык kotlin
2. Разработать мобильное приложение социальной сети с минимальными функциями
3. Добавить больше функций для взаимодействия между пользователями

Теоретическая часть

История языка программирования Kotlin

Kotlin (Ко́тлин) — кроссплатформенный, статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains.

Язык назван в честь российского острова Котлин в Финском заливе, на котором расположен город Кронштадт. Андрей Бреслав, бывший ведущий дизайнер Kotlin, упомянул, что команда решила назвать его в честь острова, так же как язык программирования Java был назван в честь индонезийского острова Ява (есть мнение, что название языка было навеяно «java» — американским сленговым термином для кофе, который сам по себе происходит от названия острова).

Язык разрабатывается с 2010 года под руководством Андрея Бреслава, представлен общественности в июле 2011. В феврале 2012 года JetBrains открыла исходный код проекта под лицензией Apache 2. Тогда же в феврале был выпущен milestone 1, включающий плагин для IDEA, в июне — milestone 2 с поддержкой Android, в декабре — milestone 4, включающий, в частности, поддержку Java 7. Компания JetBrains надеялась, что новый язык будет способствовать продажам IntelliJ IDEA.

Kotlin 1.0 был выпущен 15 февраля 2016 года. Он считается первым официально стабильным релизом и начиная с этой версии, JetBrains взяла на себя обязательство по долгосрочной обратной совместимости.

Преимущества и недостатки языка Kotlin

Преимущества:

1. *Совместимость с Java*. Kotlin можно легко интегрировать с кодом на Java и использовать библиотеки и фреймворки Java. Он также поддерживает вызов кода на Java из Kotlin и наоборот.
2. *Выразительность и лаконичность*. Kotlin позволяет писать код кратко и понятно, избегая лишних символов и повторений. Он поддерживает современные возможности языка: лямбда-выражения, расширения функций, деструктуризацию, операторы in, is, when.
3. *Безопасность от NullPointerExceptions*. Система типов данного языка программирования различает nullable и non-null типы и не допускает присваивания null к non-null переменным. Это помогает избежать одной из самых частых ошибок в Java — NullPointerExceptions. Kotlin также предоставляет операторы для работы с nullable значениями, такие как ?., ?:, !! и ?.let.
4. *Поддержка сопрограмм.* Язык поддерживает структурированную конкурентность с помощью сопрограмм — легковесных потоков исполнения, которые позволяют писать асинхронный код в императивном стиле. Сопрограммы упрощают работу с задачами, которые требуют ожидания результата, такими как сетевые запросы или обновления базы данных. Сопрограммы также поддерживают потоки данных с помощью Flow — асинхронных последовательностей элементов.

Недостатки:

1. *Сложность обучения*. Синтаксические особенности и возможности Kotlin непривычны для начинающих программистов или тех, кто переходит с других языков. Например: поддерживает функциональное программирование и содержит много ключевых слов и операторов, которые нужно запомнить и правильно использовать.
2. *Производительность*. Kotlin медленнее Java при работе с коллекциями или с лямбда-выражениями. Это связано с тем, что он генерирует больше байт-кода и использует много памяти для своих функций. Также этот язык иногда вызывает проблемы с производительностью при взаимодействии с Java-кодом из-за разных систем типов и конвенций именования.
3. *Совместимость с библиотеками*. Хотя он совместим с Java, некоторые библиотеки и фреймворки не поддерживают Kotlin или работают с ним неоптимально. Например, некоторые аннотации или рефлексия не всегда корректно работают с Kotlin-кодом и требуют дополнительных настроек.
4. *Не подходит для низкоуровневого программирования*. Kotlin — это язык высокого уровня, и он не подходит для низкоуровневого программирования, такого как написание драйверов устройств или операционных систем.

Практическая часть

Архитектура проекта

Android приложение, созданное в Android Studio, состоит из нескольких блоков:

app - это наше приложение

manifests - в папке хранится манифест приложения с описанием приложения, где указываются названия приложения, разрешения используемые приложением, activity приложения.

java - тут хранятся все файлы классов приложения, классы Activity, Fragment, классы объектов, адаптеров.

res - папка служит для хранения файлов изображений в папках drawable и mipmap, файлов макетов в папке layout, файлов с пунктами меню в папке menu и разных других значений: цвета, строковые значения, стили в папке values с названиями colors.xml, strings.xml, styles.xml

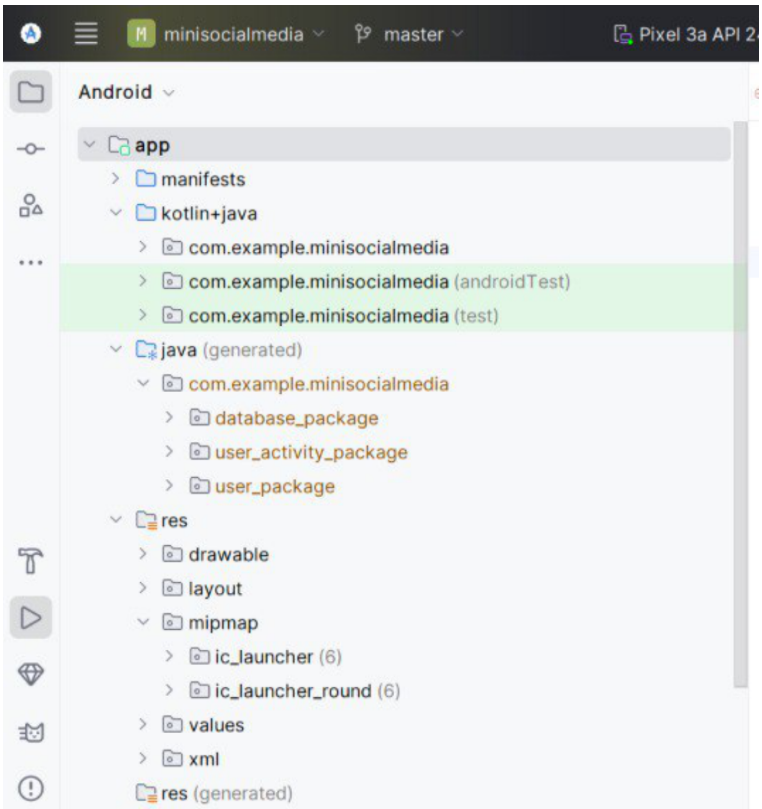

Рисунок 1. Архитектура Android приложения

Каталог Gradle Scripts содержит конфигурационные файлы сборки приложения в среде Gradle. Это программный инструмент автоматической упаковки приложения в готовый для установки пакет. Основной файл здесь — build.gradle.kts (Module: app). В нём содержатся параметры сборки проекта. Главные из них:

android{…} — тут указаны минимальная версия Android, с которой будет работать

приложение, идентификатор и версия программы.
Dependencies{…} — список зависимостей и библиотек, подключённых к проекту.

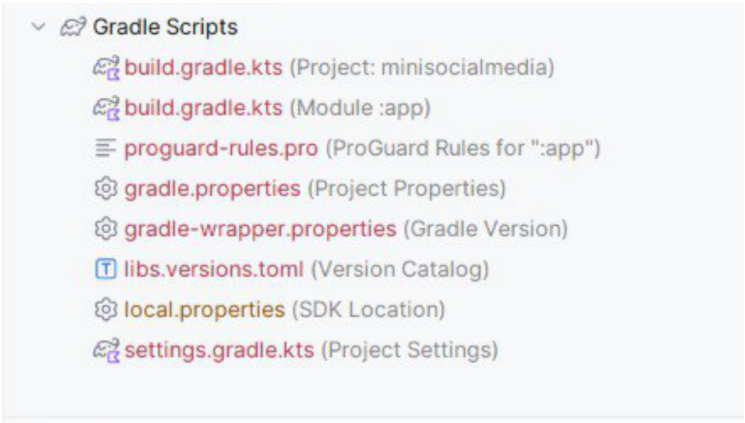

Рисунок 2. Архитектура Gradle

Работа над проектом

В ходе работы над проектом были изучены материалы по языку kotlin о создании приложения, связи xml файлов (файлов, ответственных за стили) и файлов с классами (файлов, ответственных за функциональность программы).

Для создания социальной сети с минимальными функциями был выбран следующий план действий:

1. Создание программы авторизации: добавлена система регистрации, входа и выхода из аккаунта
2. Оформление профиля: создание никнейма - уникального имени пользователя, добавление аватара, статус в виде поля “о себе”
3. Возможность создавать публикации, возможность комментировать и ставить лайки (отметку “нравится пост”)

Позднее была добавлена функция загрузки музыкальных треков.

В ходе работы была использована база данных Room для возможности работы с фактической базой данных SQLite, чтобы получить доступ к экземплярам DAO, связанных с базой данных.

Все созданные файлы были разделены в папки для лучшей навигации:

- Папка для файлов, связанных с базой данный,
- Папка для файлов, отвечающих за регистрацию и вход
- Папка для взаимодействия пользователя с приложением, куда входят файлы, отвечающие за
    - Отправку сообщений,
    - Музыку,
    - Комментарии к постам
    - Управление подписками
    - Создание постов
- Папка для файлов, отвечающих за профиль пользователя

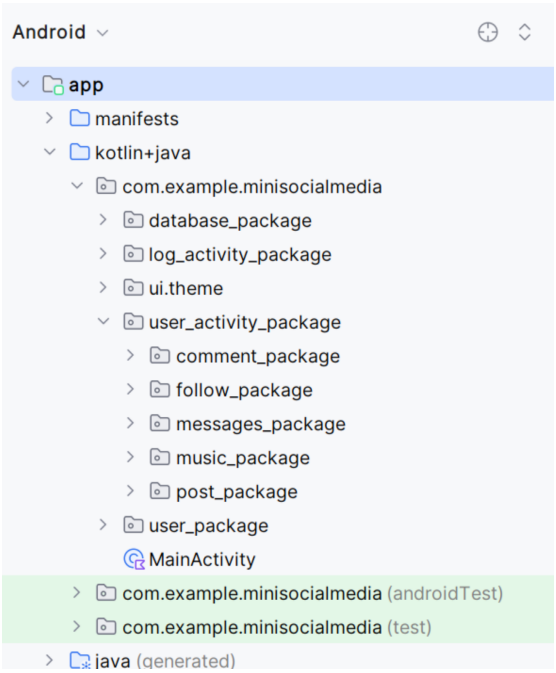

Рисунок 3. Структура проекта

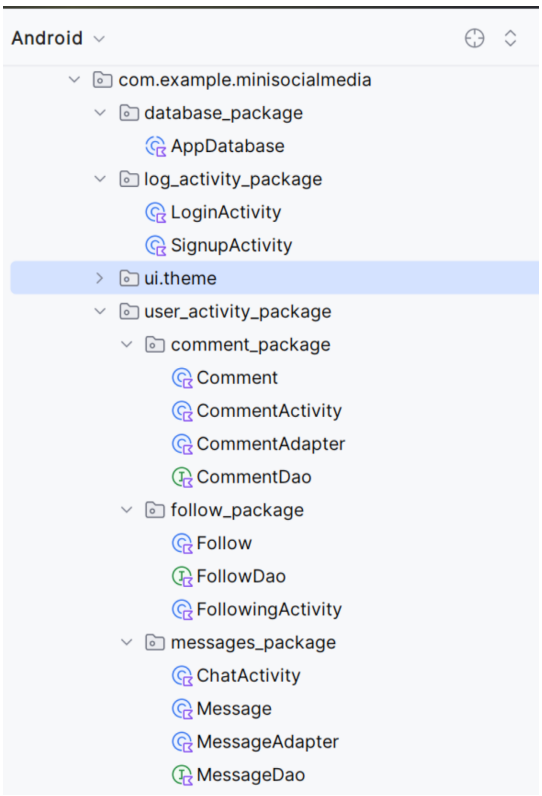

Рисунок 4. Структура папки “действия пользователя”, часть 1

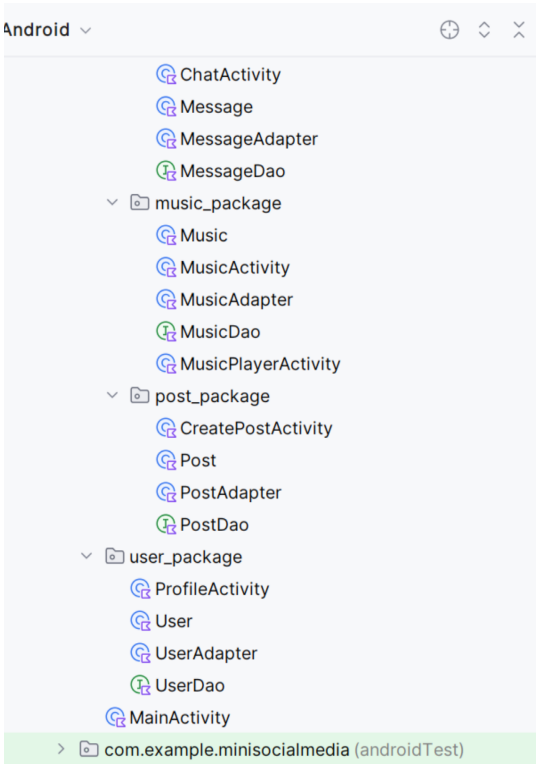

Рисунок 4. Структура папки “действия пользователя”, часть 2

Также были добавлены три цветовые гаммы для работы с профилем (лавандовый), отправкой сообщений (мятный) и музыкой (темно-синий)

Заключение

В ходе реализации проекта была достигнута основная цель - создание социальной сети, в возможности которой входят основные функции, присущие каждой социальной сети: создание и публикация постов, возможность оставлять комментарии, и переписываться с другими пользователями; также была реализована функция загрузки музыки с устройства.

Работа над проектом позволила углубить знания по разработке на языке kotlin, работе с базой данных. В процессе работы были решены различные задачи, связанные с проектированием и реализацией логики приложения. В дальнейшем можно расширить функционал получившегося приложения, например, добавив возможность пользователям подписываться на группы по интересам, создавать альбомы с фотографиями, и так далее.

Список литературы

java // merriam webster URL: https://www.merriam-webster.com/dictionary/java

Андрей Бреслав — о разработке Kotlin, профессии программиста и о том, как все успевать // itmo news URL: https://news.itmo.ru/ru/startups_and_business/business_success/news/8040/

Kotlin 1.0 Released: Pragmatic Language for the JVM and Android // jetbrains blog URL: https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/

Приложения

Приложение 1 AppDatabase.kt

```
package com.example.minisocialmedia.database_package

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import com.example.minisocialmedia.user_activity_package.comment_package.Comment
import com.example.minisocialmedia.user_activity_package.comment_package.CommentDao
import com.example.minisocialmedia.user_activity_package.follow_package.Follow
import com.example.minisocialmedia.user_activity_package.follow_package.FollowDao
import com.example.minisocialmedia.user_activity_package.messages_package.Message
import com.example.minisocialmedia.user_activity_package.messages_package.MessageDao
import com.example.minisocialmedia.user_activity_package.music_package.Music
import com.example.minisocialmedia.user_activity_package.music_package.MusicDao
import com.example.minisocialmedia.user_activity_package.post_package.Post
import com.example.minisocialmedia.user_activity_package.post_package.PostDao
import com.example.minisocialmedia.user_package.UserDao
import com.example.minisocialmedia.user_package.User

/**
 * Основной класс базы данных приложения, использующий Room.
 *
 * Определяет сущности, используемые в базе данных, и предоставляет методы для доступа к DAO (Data Access Objects).
 *
 * @property userDao DAO для доступа к данным пользователей.
 * @property postDao DAO для доступа к данным постов.
 * @property commentDao DAO для доступа к данным комментариев.
 * @property followDao DAO для доступа к данным подписок.
 * @property messageDao DAO для доступа к данным сообщений.
 * @property musicDao DAO для доступа к данным музыки.
 */
@Database(entities = [User::class, Post::class, Comment::class, Follow::class, Message::class, Music::class], version = 8)
abstract class AppDatabase : RoomDatabase() {

    /**
     * Получает DAO для доступа к данным пользователей.
     *
     * @return UserDao для работы с таблицей пользователей.
     */
    abstract fun userDao(): UserDao
    /**
     * Получает DAO для доступа к данным постов.
     *
     * @return PostDao для работы с таблицей постов.
     */
    abstract fun postDao(): PostDao
    /**
```

```
     * Получает DAO для доступа к данным комментариев.
     *
     * @return CommentDao для работы с таблицей комментариев.
     */
    abstract fun commentDao(): CommentDao
    /**
     * Получает DAO для доступа к данным подписок.
     *
     * @return FollowDao для работы с таблицей подписок.
     */
    abstract fun followDao(): FollowDao
    /**
     * Получает DAO для доступа к данным сообщений.
     *
     * @return MessageDao для работы с таблицей сообщений.
     */
    abstract fun messageDao(): MessageDao
    /**
     * Получает DAO для доступа к данным музыки.
     *
     * @return MusicDao для работы с таблицей музыки.
     */
    abstract fun musicDao(): MusicDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null
        /**
         * Получает экземпляр базы данных приложения.
         *
         * Если экземпляр базы данных не существует, создает его.
         *
         * @param context Контекст приложения.
         * @return Экземпляр AppDatabase.
         */
        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "app_database"
                ).allowMainThreadQueries().build() // Внимание: allowMainThreadQueries() только
для разработки.
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Приложение 2 LoginActivity.kt

```
package com.example.minisocialmedia.log_activity_package

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.MainActivity
import com.example.minisocialmedia.R
import com.example.minisocialmedia.user_package.UserDao

/**
 * Activity для экрана входа в приложение.
 *
 * Позволяет пользователям вводить имя пользователя и пароль для входа в приложение.
 * Проверяет учетные данные в базе данных и перенаправляет на MainActivity при успешном входе.
 */
class LoginActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * DAO для доступа к данным пользователей.
     */
    private lateinit var userDao: UserDao

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует базу данных, получает DAO для работы с пользователями,
     * устанавливает обработчик нажатия на кнопку входа.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        db = AppDatabase.getDatabase(this)
        userDao = db.userDao()

        val etNickname = findViewById<EditText>(R.id.etLoginNickname)
        val etPassword = findViewById<EditText>(R.id.etLoginPassword)
```

```
        val btnLogin = findViewById<Button>(R.id.btnLogin)

        btnLogin.setOnClickListener {
            val nickname = etNickname.text.toString()
            val password = etPassword.text.toString()

            val user = userDao.getUser(nickname, password)
            if (user != null) {
                Toast.makeText(this, "Login Successful!", Toast.LENGTH_SHORT).show()
                startActivity(Intent(this, MainActivity::class.java))
                finish()
            } else {
                Toast.makeText(this, "Invalid Credentials", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Приложение 3 SignupActivity.kt

```
package com.example.minisocialmedia.log_activity_package

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.example.minisocialmedia.R

/**
 * Activity для экрана регистрации пользователя.
 *
 * Устанавливает макет для регистрации и обрабатывает вставки системных окон.
 */
class SignupActivity : AppCompatActivity() {

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует макет регистрации и устанавливает обработчик для обработки
     * вставок системных окон, чтобы контент не перекрывался системными элементами.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge() // Включает отображение контента до краев экрана
        setContentView(R.layout.activity_signup)

        // Устанавливает слушатель для обработки вставок системных окон (например, статус-бар, навигационная панель)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            // Устанавливает отступы для корневого вида, чтобы контент не перекрывался системными элементами
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets // Возвращает исходные вставки
        }
    }
}
```

Приложение 4 Color.kt

```
package com.example.minisocialmedia.ui.theme

import androidx.compose.ui.graphics.Color

val Purple80 = Color(0xFFD0BCFF)
val PurpleGrey80 = Color(0xFFCCC2DC)
val Pink80 = Color(0xFFEFB8C8)

val Purple40 = Color(0xFF6650a4)
val PurpleGrey40 = Color(0xFF625b71)
val Pink40 = Color(0xFF7D5260)
```

Приложение 5 Theme.kt

```
package com.example.minisocialmedia.ui.theme

import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40

    /* Other default colors to override
    background = Color(0xFFFFFBFE),
    surface = Color(0xFFFFFBFE),
    onPrimary = Color.White,
    onSecondary = Color.White,
    onTertiary = Color.White,
    onBackground = Color(0xFF1C1B1F),
    onSurface = Color(0xFF1C1B1F),
    */
)

@Composable
fun MiniSocialMedia(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme
        else -> LightColorScheme
```

```
    }

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

Приложение 6 Type.kt

```
package com.example.minisocialmedia.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    )
    /* Other default text styles to override
    titleLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 22.sp,
        lineHeight = 28.sp,
        letterSpacing = 0.sp
    ),
    labelSmall = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Medium,
        fontSize = 11.sp,
        lineHeight = 16.sp,
        letterSpacing = 0.5.sp
    )
    */
)
```

Приложение 7 Comment.kt

```
package com.example.minisocialmedia.user_activity_package.comment_package

import androidx.room.Entity
import androidx.room.PrimaryKey

/**
 * Класс данных, представляющий комментарий к посту.
 *
 * @property id Уникальный идентификатор комментария.
 * @property postId Идентификатор поста, к которому относится комментарий.
 * @property author Имя автора комментария.
 * @property content Содержание комментария.
 * @property timestamp Время создания комментария в миллисекундах с начала эпохи Unix.
 */
@Entity(tableName = "comments")
data class Comment(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val postId: Int,
    val author: String,
    val content: String,
    val timestamp: Long = System.currentTimeMillis()
)
```

Приложение 8 CommentActivity.kt

```
package com.example.minisocialmedia.user_activity_package.comment_package

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.R

/**
 * Activity для отображения и добавления комментариев к посту.
 *
 * Позволяет пользователям просматривать комментарии к посту и добавлять новые.
 */
class CommentActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * DAO для работы с комментариями.
     */
    private lateinit var commentDao: CommentDao

    /**
     * Поле ввода для добавления нового комментария.
     */
    private lateinit var etComment: EditText

    /**
     * Кнопка для отправки нового комментария.
     */
    private lateinit var btnSubmitComment: Button

    /**
     * RecyclerView для отображения списка комментариев.
     */
    private lateinit var rvComments: RecyclerView

    /**
     * Идентификатор поста, к которому относятся комментарии.
     */
    private var postId: Int = -1

    /**
     * Вызывается при создании Activity.
```

```
     *
     * Инициализирует базу данных, получает DAO для работы с комментариями,
     * устанавливает обработчики событий и загружает комментарии.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_comment)

        db = AppDatabase.getDatabase(this)
        commentDao = db.commentDao()

        etComment = findViewById(R.id.etComment)
        btnSubmitComment = findViewById(R.id.btnSubmitComment)
        rvComments = findViewById(R.id.rvComments)

        postId = intent.getIntExtra("postId", -1)

        btnSubmitComment.setOnClickListener {
            val commentText = etComment.text.toString()
            if (commentText.isNotEmpty()) {
                val comment = Comment(postId = postId, author = "User", content = commentText)
                commentDao.insertComment(comment)
                etComment.text.clear()
                loadComments()
            }
        }

        loadComments()
    }

    /**
     * Загружает комментарии для заданного поста и отображает их в RecyclerView.
     */
    private fun loadComments() {
        val comments = commentDao.getCommentsForPost(postId)
        rvComments.layoutManager = LinearLayoutManager(this)
        rvComments.adapter = CommentAdapter(comments)
    }
}
```

Приложение 9 CommentAdapter.kt

```
package com.example.minisocialmedia.user_activity_package.comment_package

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.R

/**
 * Адаптер для отображения списка комментариев в RecyclerView.
 *
 * @param comments Список комментариев для отображения.
 */
class CommentAdapter(private val comments: List<Comment>) :
RecyclerView.Adapter<CommentAdapter.CommentViewHolder>() {

    /**
     * ViewHolder для элементов списка комментариев.
     *
     * @param view Представление элемента списка комментариев.
     */
    class CommentViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvAuthor: TextView = view.findViewById(R.id.tvAuthor)
        val tvComment: TextView = view.findViewById(R.id.tvComment)
    }

    /**
     * Создает новый ViewHolder для элемента списка комментариев.
     *
     * @param parent ViewGroup, в который будет добавлен ViewHolder.
     * @param viewType Тип представления нового представления.
     * @return Новый ViewHolder.
     */
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CommentViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_comment, parent, false)
        return CommentViewHolder(view)
    }

    /**
     * Привязывает данные комментария к ViewHolder.
     *
     * @param holder ViewHolder, к которому нужно привязать данные.
     * @param position Позиция элемента в списке.
     */
    override fun onBindViewHolder(holder: CommentViewHolder, position: Int) {
        val comment = comments[position]
        holder.tvAuthor.text = comment.author
        holder.tvComment.text = comment.content
    }
```

```
    /**
     * Возвращает количество элементов в списке комментариев.
     *
     * @return Количество элементов в списке.
     */
    override fun getItemCount() = comments.size
}
```

Приложение 10 CommentDao.kt

```
package com.example.minisocialmedia.user_activity_package.comment_package

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

/**
 * DAO для работы с комментариями.
 */
@Dao
interface CommentDao {

    /**
     * Вставляет новый комментарий в базу данных.
     *
     * @param comment Комментарий для вставки.
     */
    @Insert
    fun insertComment(comment: Comment)

    /**
     * Получает список комментариев для заданного поста, отсортированных по времени создания.
     *
     * @param postId Идентификатор поста, для которого нужно получить комментарии.
     * @return Список комментариев для поста.
     */
    @Query("SELECT * FROM comments WHERE postId = :postId ORDER BY timestamp ASC")
    fun getCommentsForPost(postId: Int): List<Comment>
}
```

Приложение 11 Follow.kt

```
package com.example.minisocialmedia.user_activity_package.follow_package

import androidx.room.Entity
import androidx.room.PrimaryKey

/**
 * Класс данных, представляющий связь подписки между пользователями.
 *
 * @property id Уникальный идентификатор подписки.
 * @property userId Идентификатор пользователя, который подписался.
 * @property followedUserId Идентификатор пользователя, на которого подписались.
 */
@Entity(tableName = "follows")
data class Follow(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val userId: Int,
    val followedUserId: Int
)
```

Приложение 12 FollowDao.kt

```
package com.example.minisocialmedia.user_activity_package.follow_package

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query

/**
 * DAO для работы с подписками.
 */
@Dao
interface FollowDao {

    /**
     * Добавляет новую подписку.
     *
     * @param follow Объект подписки для добавления.
     */
    @Insert
    fun followUser(follow: Follow)

    /**
     * Удаляет подписку.
     *
     * @param follow Объект подписки для удаления.
     */
    @Delete
    fun unfollowUser(follow: Follow)

    /**
     * Получает список подписок пользователя.
     *
     * @param userId Идентификатор пользователя, чьи подписки нужно получить.
     * @return Список подписок пользователя.
     */
    @Query("SELECT * FROM follows WHERE userId = :userId")
    fun getFollowing(userId: Int): List<Follow>
}
```

Приложение 13 FollowingActivity.kt

```
package com.example.minisocialmedia.user_activity_package.follow_package

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.R
import com.example.minisocialmedia.user_package.UserAdapter

/**
 * Activity для отображения списка пользователей, на которых подписан текущий пользователь.
 */
class FollowingActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * DAO для работы с подписками.
     */
    private lateinit var followDao: FollowDao

    /**
     * RecyclerView для отображения списка пользователей, на которых подписан текущий пользователь.
     */
    private lateinit var rvFollowing: RecyclerView

    /**
     * Идентификатор текущего пользователя.
     */
    private var currentUserId = 1

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует базу данных, получает DAO для работы с подписками,
     * устанавливает RecyclerView и загружает список подписок.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_following)
```

```
        db = AppDatabase.getDatabase(this)
        followDao = db.followDao()

        rvFollowing = findViewById(R.id.rvFollowing)

        loadFollowing()
    }

    /**
     * Загружает список пользователей, на которых подписан текущий пользователь, и
отображает их в RecyclerView.
     */
    private fun loadFollowing() {
        val follows = followDao.getFollowing(currentUserId)
        val followedUserIds = follows.map { it.followedUserId }

        val db = AppDatabase.getDatabase(this)
        val userDao = db.userDao()

        val users = userDao.getUsersByIds(followedUserIds)

        rvFollowing.layoutManager = LinearLayoutManager(this)
        rvFollowing.adapter = UserAdapter(users)
    }
}
```

Приложение 14 ChatActivity.kt

```
package com.example.minisocialmedia.user_activity_package.messages_package

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.R
import com.example.minisocialmedia.database_package.AppDatabase

/**
 * Activity для отображения чата между двумя пользователями.
 */
class ChatActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * DAO для работы с сообщениями.
     */
    private lateinit var messageDao: MessageDao

    /**
     * RecyclerView для отображения сообщений.
     */
    private lateinit var rvMessages: RecyclerView

    /**
     * Поле ввода для отправки нового сообщения.
     */
    private lateinit var etMessage: EditText

    /**
     * Кнопка для отправки сообщения.
     */
    private lateinit var btnSend: Button

    /**
     * Адаптер для отображения сообщений.
     */
    private lateinit var adapter: MessageAdapter

    /**
     * Идентификатор отправителя сообщений.
     */
    private var senderId: Int = 0
```

```
    /**
     * Идентификатор получателя сообщений.
     */
    private var receiverId: Int = 0

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует базу данных, получает DAO для работы с сообщениями,
     * устанавливает RecyclerView, EditText, Button и загружает сообщения.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat)

        db = AppDatabase.getDatabase(this)
        messageDao = db.messageDao()

        rvMessages = findViewById(R.id.rvMessages)
        etMessage = findViewById(R.id.etMessage)
        btnSend = findViewById(R.id.btnSend)

        senderId = intent.getIntExtra("SENDER_ID", 0)
        receiverId = intent.getIntExtra("RECEIVER_ID", 0)

        loadMessages()

        btnSend.setOnClickListener {
            val text = etMessage.text.toString().trim()
            if (text.isNotEmpty()) {
                val message = Message(0, senderId, receiverId, text, System.currentTimeMillis())
                messageDao.insertMessage(message)
                etMessage.text.clear()
                loadMessages()
            }
        }
    }
    /**
     * Загружает сообщения между отправителем и получателем и отображает их в
RecyclerView.
     */
    private fun loadMessages() {
        val messages = messageDao.getChatMessages(senderId, receiverId)
        adapter = MessageAdapter(messages)
        rvMessages.layoutManager = LinearLayoutManager(this)
        rvMessages.adapter = adapter
    }
}
```

Приложение 15 Message.kt

```
package com.example.minisocialmedia.user_activity_package.messages_package

import androidx.room.Entity
import androidx.room.PrimaryKey

/**
 * Класс данных, представляющий сообщение между пользователями.
 *
 * @property id Уникальный идентификатор сообщения.
 * @property senderId Идентификатор отправителя сообщения.
 * @property receiverId Идентификатор получателя сообщения.
 * @property content Содержание сообщения.
 * @property timestamp Время отправки сообщения в миллисекундах с начала эпохи Unix.
 */
@Entity(tableName = "messages")
data class Message(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val senderId: Int,
    val receiverId: Int,
    val content: String,
    val timestamp: Long
)
```

Приложение 16 MessageAdapter.kt

```
package com.example.minisocialmedia.user_activity_package.messages_package

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.R

/**
 * Адаптер для отображения списка сообщений в RecyclerView.
 *
 * @param messages Список сообщений для отображения.
 */
class MessageAdapter(private val messages: List<Message>) :
    RecyclerView.Adapter<MessageAdapter.MessageViewHolder>() {

    /**
     * ViewHolder для элементов списка сообщений.
     *
     * @param view Представление элемента списка сообщений.
     */
    class MessageViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvMessage: TextView = view.findViewById(R.id.tvMessage)
    }

    /**
     * Создает новый ViewHolder для элемента списка сообщений.
     *
     * @param parent ViewGroup, в который будет добавлен ViewHolder.
     * @param viewType Тип представления нового представления.
     * @return Новый ViewHolder.
     */
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MessageViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_message, parent, false)
        return MessageViewHolder(view)
    }

    /**
     * Привязывает данные сообщения к ViewHolder.
     *
     * @param holder ViewHolder, к которому нужно привязать данные.
     * @param position Позиция элемента в списке.
     */
    override fun onBindViewHolder(holder: MessageViewHolder, position: Int) {
        val message = messages[position]
        holder.tvMessage.text = message.content
    }

    /**
```

```
     * Возвращает количество элементов в списке сообщений.
     *
     * @return Количество элементов в списке.
     */
    override fun getItemCount() = messages.size
}
```

Приложение 17 MessageDao.kt

```
package com.example.minisocialmedia.user_activity_package.messages_package

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

/**
 * DAO для работы с сообщениями.
 */
@Dao
interface MessageDao {

    /**
     * Вставляет новое сообщение в базу данных.
     *
     * @param message Сообщение для вставки.
     */
    @Insert
    fun insertMessage(message: Message)

    /**
     * Получает список сообщений между двумя пользователями, отсортированных по времени отправки.
     *
     * @param user1 Идентификатор первого пользователя.
     * @param user2 Идентификатор второго пользователя.
     * @return Список сообщений между пользователями.
     */
    @Query("SELECT * FROM messages WHERE (senderId = :user1 AND receiverId = :user2) OR (senderId = :user2 AND receiverId = :user1) ORDER BY timestamp ASC")
    fun getChatMessages(user1: Int, user2: Int): List<Message>
}
```

Приложение 18 Music.kt

```
package com.example.minisocialmedia.user_activity_package.music_package

import androidx.room.Entity
import androidx.room.PrimaryKey

/**
 * Класс данных, представляющий музыкальный трек.
 *
 * @property id Уникальный идентификатор трека.
 * @property title Название трека.
 * @property filePath Путь к файлу трека.
 * @property duration Длительность трека в миллисекундах.
 */
@Entity(tableName = "music")
data class Music(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val title: String,
    val filePath: String,
    val duration: Long
)
```

Приложение 19 MusicActivity.kt

```
package com.example.minisocialmedia.user_activity_package.music_package

import android.app.Activity
import android.content.Intent
import android.media.MediaMetadataRetriever
import android.net.Uri
import android.os.Bundle
import android.provider.OpenableColumns
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.minisocialmedia.R
import com.example.minisocialmedia.database_package.AppDatabase
import java.io.File
import java.io.FileOutputStream
import java.io.InputStream

/**
 * Activity для выбора и сохранения музыкальных файлов.
 */
class MusicActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * Кнопка для выбора музыкального файла.
     */
    private lateinit var btnSelectMusic: Button

    /**
     * Код запроса для выбора аудиофайла.
     */
    private val PICK_AUDIO_REQUEST = 1

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует базу данных, устанавливает обработчик нажатия на кнопку выбора
музыки.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_music)
```

```
        db = AppDatabase.getDatabase(this)

        btnSelectMusic = findViewById(R.id.btnSelectMusic)
        btnSelectMusic.setOnClickListener {
            selectAudioFile()
        }
    }

    /**
     * Открывает Intent для выбора аудиофайла.
     */
    private fun selectAudioFile() {
        val intent = Intent(Intent.ACTION_GET_CONTENT)
        intent.type = "audio/*"
        startActivityForResult(intent, PICK_AUDIO_REQUEST)
    }

    /**
     * Обрабатывает результат выбора аудиофайла.
     *
     * @param requestCode Код запроса.
     * @param resultCode Результат операции.
     * @param data Intent с данными.
     */
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == PICK_AUDIO_REQUEST && resultCode == Activity.RESULT_OK) {
            data?.data?.let { uri ->
                saveMusicFile(uri)
            }
        }
    }

    /**
     * Сохраняет выбранный музыкальный файл в локальное хранилище и добавляет
информацию о нем в базу данных.
     *
     * @param uri URI выбранного аудиофайла.
     */
    private fun saveMusicFile(uri: Uri) {
        val inputStream: InputStream? = contentResolver.openInputStream(uri)
        if (inputStream != null) {
            val fileName = getFileName(uri)
            val file = File(filesDir, fileName)
            val outputStream = FileOutputStream(file)

            inputStream.copyTo(outputStream)
            inputStream.close()
            outputStream.close()

            val duration = getAudioDuration(file.absolutePath)
```

```
            val music = Music(title = fileName, filePath = file.absolutePath, duration = duration)
            db.musicDao().insertMusic(music)

            Toast.makeText(this, "Music saved!", Toast.LENGTH_SHORT).show()
        }
    }

    /**
     * Получает имя файла из URI.
     *
     * @param uri URI файла.
     * @return Имя файла.
     */
    private fun getFileName(uri: Uri): String {
        var name = "audio_file.mp3"
        val cursor = contentResolver.query(uri, null, null, null, null)
        cursor?.use {
            if (it.moveToFirst()) {
                name = it.getString(it.getColumnIndex(OpenableColumns.DISPLAY_NAME))
            }
        }
        return name
    }

    /**
     * Получает длительность аудиофайла.
     *
     * @param filePath Путь к аудиофайлу.
     * @return Длительность аудиофайла в миллисекундах.
     */
    private fun getAudioDuration(filePath: String): Long {
        val retriever = MediaMetadataRetriever()
        retriever.setDataSource(filePath)
        val durationStr =
retriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DURATION)
        retriever.release()
        return durationStr?.toLong() ?: 0
    }
}
```

Приложение 20 MusicAdapter.kt

```
package com.example.minisocialmedia.user_activity_package.music_package

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import android.widget.Toast
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.R

/**
 * Адаптер для отображения списка музыкальных треков в RecyclerView.
 *
 * @param context Контекст приложения.
 * @param musicList Список музыкальных треков для отображения.
 * @param onItemClick Функция, вызываемая при нажатии на элемент списка.
 */
class MusicAdapter(
    private val context: Context,
    private val musicList: List<Music>,
    private val onItemClick: (Music) -> Unit
) : RecyclerView.Adapter<MusicAdapter.MusicViewHolder>() {

    /**
     * ViewHolder для элементов списка музыкальных треков.
     *
     * @param view Представление элемента списка музыкальных треков.
     */
    class MusicViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvTitle: TextView = view.findViewById(R.id.tvMusicTitle)
        val tvDuration: TextView = view.findViewById(R.id.tvMusicDuration)
    }

    /**
     * Создает новый ViewHolder для элемента списка музыкальных треков.
     *
     * @param parent ViewGroup, в который будет добавлен ViewHolder.
     * @param viewType Тип представления нового представления.
     * @return Новый ViewHolder.
     */
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MusicViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_music, parent, false)
        return MusicViewHolder(view)
    }

    /**
     * Привязывает данные музыкального трека к ViewHolder.
     *
```

```
     * @param holder ViewHolder, к которому нужно привязать данные.
     * @param position Позиция элемента в списке.
     */
    override fun onBindViewHolder(holder: MusicViewHolder, position: Int) {
        val music = musicList[position]
        holder.tvTitle.text = music.title
        holder.tvDuration.text = formatDuration(music.duration)

        holder.itemView.setOnClickListener {
            Toast.makeText(context, "Playing: ${music.title}", Toast.LENGTH_SHORT).show()
            onItemClick(music)
        }
    }

    /**
     * Возвращает количество элементов в списке музыкальных треков.
     *
     * @return Количество элементов в списке.
     */
    override fun getItemCount() = musicList.size

    /**
     * Форматирует длительность музыкального трека в строку "MM:SS".
     *
     * @param duration Длительность музыкального трека в миллисекундах.
     * @return Форматированная строка длительности.
     */
    private fun formatDuration(duration: Long): String {
        val minutes = duration / 60000
        val seconds = (duration % 60000) / 1000
        return String.format("%02d:%02d", minutes, seconds)
    }
}
```

Приложение 21 MusicDao.kt

```
package com.example.minisocialmedia.user_activity_package.music_package

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

/**
 * DAO для работы с музыкальными треками.
 */
@Dao
interface MusicDao {

    /**
     * Вставляет новый музыкальный трек в базу данных.
     *
     * @param music Музыкальный трек для вставки.
     */
    @Insert
    fun insertMusic(music: Music)

    /**
     * Получает список всех музыкальных треков, отсортированных по названию.
     *
     * @return Список всех музыкальных треков.
     */
    @Query("SELECT * FROM music ORDER BY title ASC")
    fun getAllMusic(): List<Music>
}
```

Приложение 22 MusicPlayerActivity.kt

```
package com.example.minisocialmedia.user_activity_package.music_package

import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.media.MediaPlayer
import android.net.Uri
import android.os.Bundle
import android.os.Environment
import android.provider.MediaStore
import android.provider.OpenableColumns
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.R

/**
 * Activity для воспроизведения музыкальных треков.
 */
class MusicPlayerActivity : AppCompatActivity() {

    companion object {
        private const val REQUEST_PERMISSION_CODE = 1000
        private const val REQUEST_CODE_PICK_AUDIO = 1001
    }

    /**
     * RecyclerView для отображения списка музыкальных треков.
     */
    private lateinit var rvMusicList: RecyclerView

    /**
     * Кнопка для остановки воспроизведения.
     */
    private lateinit var btnStopMusic: Button

    /**
     * Кнопка для выбора музыкального файла.
     */
    private lateinit var btnChooseMusic: Button

    /**
     * Объект MediaPlayer для воспроизведения музыки.
     */
    private var mediaPlayer: MediaPlayer? = null
```

```
    /**
     * Адаптер для списка музыкальных треков.
     */
    private lateinit var musicAdapter: MusicAdapter

    /**
     * Список музыкальных треков.
     */
    private var musicList = mutableListOf<Music>()

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует UI, проверяет разрешения, загружает список музыки и устанавливает
обработчики событий.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_music_player)

        rvMusicList = findViewById(R.id.musicListView)
        btnStopMusic = findViewById(R.id.btnStopMusic)
        btnChooseMusic = findViewById(R.id.btnChooseMusic)

        checkPermissions()
        musicList = loadMusicFiles().toMutableList()

        musicAdapter = MusicAdapter(this, musicList) { music ->
            playMusic(music)
        }
        rvMusicList.layoutManager = LinearLayoutManager(this)
        rvMusicList.adapter = musicAdapter

        btnStopMusic.setOnClickListener { stopMusic() }
        btnChooseMusic.setOnClickListener { openFilePicker() }
    }

    /**
     * Проверяет и запрашивает разрешения на чтение внешнего хранилища.
     */
    private fun checkPermissions() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE),
REQUEST_PERMISSION_CODE)
        }
    }
```

```
/**
 * Воспроизводит музыкальный трек.
 *
 * @param music Музыкальный трек для воспроизведения.
 */
private fun playMusic(music: Music) {
    stopMusic()
    mediaPlayer = MediaPlayer().apply {
        setDataSource(music.filePath)
        prepare()
        start()
    }
}

/**
 * Останавливает воспроизведение музыкального трека.
 */
private fun stopMusic() {
    mediaPlayer?.apply {
        if (isPlaying) {
            stop()
            release()
        }
    }
    mediaPlayer = null
}

/**
 * Открывает файловый менеджер для выбора музыкального файла.
 */
private fun openFilePicker() {
    val intent = Intent(Intent.ACTION_OPEN_DOCUMENT).apply {
        addCategory(Intent.CATEGORY_OPENABLE)
        type = "audio/*"
    }
    startActivityForResult(intent, REQUEST_CODE_PICK_AUDIO)
}

/**
 * Обрабатывает результат выбора музыкального файла.
 *
 * @param requestCode Код запроса.
 * @param resultCode Результат операции.
 * @param data Intent с данными.
 */
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == REQUEST_CODE_PICK_AUDIO && resultCode == RESULT_OK) {
        data?.data?.let { uri ->
            val filePath = getFilePathFromUri(uri)
            val fileName = getFileName(uri)
```

```
            if (filePath != null && fileName != null) {
                saveMusicToDatabase(fileName, filePath)
                musicList.add(Music(title = fileName, filePath = filePath, duration = 0L))
                musicAdapter.notifyDataSetChanged()
            }
        }
    }
}

/**
 * Получает путь к файлу из URI.
 *
 * @param uri URI файла.
 * @return Путь к файлу или null, если не удалось получить.
 */
private fun getFilePathFromUri(uri: Uri): String? {
    val cursor = contentResolver.query(uri, arrayOf(MediaStore.Audio.Media.DATA), null, null,
null)
    cursor?.use {
        if (it.moveToFirst()) {
            val columnIndex = it.getColumnIndex(MediaStore.Audio.Media.DATA)
            return it.getString(columnIndex)
        }
    }
    return null
}

/**
 * Получает имя файла из URI.
 *
 * @param uri URI файла.
 * @return Имя файла или null, если не удалось получить.
 */
private fun getFileName(uri: Uri): String? {
    val cursor = contentResolver.query(uri, null, null, null, null)
    cursor?.use {
        if (it.moveToFirst()) {
            val columnIndex = it.getColumnIndex(OpenableColumns.DISPLAY_NAME)
            return it.getString(columnIndex)
        }
    }
    return null
}

/**
 * Сохраняет информацию о музыкальном треке в базу данных.
 *
 * @param title Название музыкального трека.
 * @param filePath Путь к файлу музыкального трека.
 */
private fun saveMusicToDatabase(title: String, filePath: String) {
```

```
        val db = AppDatabase.getDatabase(this)
        val musicDao = db.musicDao()

        val newMusic = Music(title = title, filePath = filePath, duration = 0L)
        musicDao.insertMusic(newMusic)
    }

    /**
     * Загружает список музыкальных треков из внешнего хранилища.
     *
     * @return Список музыкальных треков.
     */
    private fun loadMusicFiles(): List<Music> {
        val musicList = mutableListOf<Music>()
        val musicDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC)

        if (musicDir.exists() && musicDir.isDirectory) {
            val musicFiles = musicDir.listFiles { file -> file.extension == "mp3" }

            musicFiles?.forEach { file ->
                val music = Music(
                    title = file.nameWithoutExtension,
                    filePath = file.absolutePath,
                    duration = getMusicDuration(file.absolutePath)
                )
                musicList.add(music)
            }
        }
        return musicList
    }

    /**
     * Получает длительность музыкального трека.
     *
     * @param filePath Путь к файлу музыкального трека.
     * @return Длительность музыкального трека в миллисекундах.
     */
    private fun getMusicDuration(filePath: String): Long {
        val mediaPlayer = MediaPlayer()
        mediaPlayer.setDataSource(filePath)
        mediaPlayer.prepare()
        return mediaPlayer.duration.toLong()
    }
    /**
     * Останавливает воспроизведение музыки при уничтожении Activity.
     */
    override fun onDestroy() {
        super.onDestroy()
        stopMusic()
    }
}
```

Пложение 23 CreatePostActivity.kt

```
package com.example.minisocialmedia.user_activity_package.post_package

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.net.Uri
import android.os.Bundle
import android.provider.MediaStore
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.R

/**
 * Activity для создания нового поста.
 */
class CreatePostActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * DAO для работы с постами.
     */
    private lateinit var postDao: PostDao

    /**
     * SharedPreferences для хранения данных пользователя.
     */
    private lateinit var sharedPreferences: SharedPreferences

    /**
     * Поле ввода для содержания поста.
     */
    private lateinit var etPostContent: EditText

    /**
     * ImageView для отображения выбранного изображения поста.
     */
    private lateinit var ivPostImage: ImageView

    /**
```

```
     * Кнопка для выбора изображения поста.
     */
    private lateinit var btnSelectImage: Button

    /**
     * Кнопка для публикации поста.
     */
    private lateinit var btnPost: Button

    /**
     * URI выбранного изображения поста.
     */
    private var imageUri: Uri? = null

    /**
     * Код запроса для выбора изображения.
     */
    private val PICK_IMAGE_REQUEST = 1

    /**
     * Вызывается при создании Activity.
     *
     * Инициализирует UI, получает DAO и SharedPreferences, устанавливает обработчики
событий.
     *
     * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
     * этот Bundle содержит данные, которые он ранее сохранил.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_create_post)

        db = AppDatabase.getDatabase(this)
        postDao = db.postDao()
        sharedPreferences = getSharedPreferences("UserPrefs", Context.MODE_PRIVATE)

        etPostContent = findViewById(R.id.etPostContent)
        ivPostImage = findViewById(R.id.ivPostImage)
        btnSelectImage = findViewById(R.id.btnSelectImage)
        btnPost = findViewById(R.id.btnPost)

        btnSelectImage.setOnClickListener {
            val intent = Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
            startActivityForResult(intent, PICK_IMAGE_REQUEST)
        }

        btnPost.setOnClickListener {
            val content = etPostContent.text.toString()
            val author = sharedPreferences.getString("loggedInUser", "") ?: ""

            if (content.isNotEmpty() && author.isNotEmpty()) {
```

```
            val post = Post(author = author, content = content, imageUri = imageUri?.toString() ?: "")
            postDao.insertPost(post)
            Toast.makeText(this, "Posted!", Toast.LENGTH_SHORT).show()
            finish()
        } else {
            Toast.makeText(this, "Enter some text", Toast.LENGTH_SHORT).show()
        }
    }
}

/**
 * Обрабатывает результат выбора изображения.
 *
 * @param requestCode Код запроса.
 * @param resultCode Результат операции.
 * @param data Intent с данными.
 */
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == PICK_IMAGE_REQUEST && resultCode == Activity.RESULT_OK) {
        data?.data?.let { uri ->
            imageUri = uri
            ivPostImage.visibility = View.VISIBLE
            ivPostImage.setImageURI(uri)
        }
    }
}
}
```

Приложение 24 Post.kt

```
package com.example.minisocialmedia.user_activity_package.post_package

import androidx.room.Entity
import androidx.room.PrimaryKey

/**
 * Класс данных, представляющий пост пользователя.
 *
 * @property id Уникальный идентификатор поста.
 * @property author Имя автора поста.
 * @property content Содержание поста.
 * @property imageUri URI изображения, прикрепленного к посту (если есть).
 * @property timestamp Время создания поста в миллисекундах с начала эпохи Unix.
 * @property likes Количество лайков поста.
 */
@Entity(tableName = "posts")
data class Post(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val author: String,
    val content: String,
    val imageUri: String = "",
    val timestamp: Long = System.currentTimeMillis(),
    var likes: Int = 0
)
```

Приложение 25 PostAdapter.kt

```
package com.example.minisocialmedia.user_activity_package.post_package

import android.content.Intent
import android.net.Uri
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.R
import com.example.minisocialmedia.user_activity_package.comment_package.CommentActivity

/**
 * Адаптер для отображения списка постов в RecyclerView.
 *
 * @param posts Список постов для отображения.
 * @param postDao DAO для работы с постами.
 */
class PostAdapter(
    private val posts: List<Post>,
    private val postDao: PostDao
) : RecyclerView.Adapter<PostAdapter.PostViewHolder>() {

    /**
     * ViewHolder для элементов списка постов.
     *
     * @param view Представление элемента списка постов.
     */
    class PostViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvAuthor: TextView = view.findViewById(R.id.tvAuthor)
        val tvContent: TextView = view.findViewById(R.id.tvContent)
        val ivPostImage: ImageView = view.findViewById(R.id.ivPostImage)
        val tvLikes: TextView = view.findViewById(R.id.tvLikes)
        val btnLike: Button = view.findViewById(R.id.btnLike)
        val btnComment: Button = view.findViewById(R.id.btnComment)
    }

    /**
     * Создает новый ViewHolder для элемента списка постов.
     *
     * @param parent ViewGroup, в который будет добавлен ViewHolder.
     * @param viewType Тип представления нового представления.
     * @return Новый ViewHolder.
     */
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PostViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_post, parent, false)
        return PostViewHolder(view)
    }
```

```
    /**
     * Привязывает данные поста к ViewHolder.
     *
     * @param holder ViewHolder, к которому нужно привязать данные.
     * @param position Позиция элемента в списке.
     */
    override fun onBindViewHolder(holder: PostViewHolder, position: Int) {
        val post = posts[position]

        holder.tvAuthor.text = post.author
        holder.tvContent.text = post.content
        holder.tvLikes.text = "${post.likes} Likes"

        if (post.imageUri.isNotEmpty()) {
            holder.ivPostImage.visibility = View.VISIBLE
            holder.ivPostImage.setImageURI(Uri.parse(post.imageUri))
        } else {
            holder.ivPostImage.visibility = View.GONE
        }

        holder.btnLike.setOnClickListener {
            post.likes += 1
            postDao.insertPost(post)
            holder.tvLikes.text = "${post.likes} Likes"
        }

        holder.btnComment.setOnClickListener {
            val intent = Intent(holder.itemView.context, CommentActivity::class.java)
            intent.putExtra("postId", post.id)
            holder.itemView.context.startActivity(intent)
        }
    }

    /**
     * Возвращает количество элементов в списке постов.
     *
     * @return Количество элементов в списке.
     */
    override fun getItemCount() = posts.size
}
```

Приложение 26 PostDao.kt

```
package com.example.minisocialmedia.user_activity_package.post_package

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

/**
 * DAO для работы с постами.
 */
@Dao
interface PostDao {

    /**
     * Вставляет новый пост в базу данных.
     *
     * @param post Пост для вставки.
     */
    @Insert
    fun insertPost(post: Post)

    /**
     * Получает список всех постов, отсортированных по времени создания в обратном порядке.
     *
     * @return Список всех постов.
     */
    @Query("SELECT * FROM posts ORDER BY timestamp DESC")
    fun getAllPosts(): List<Post>
}
```

Приложение 27 ProfileActivity.kt

```
package com.example.minisocialmedia.user_package

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.net.Uri
import android.os.Bundle
import android.provider.MediaStore
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.R
import com.example.minisocialmedia.user_activity_package.follow_package.Follow
import com.example.minisocialmedia.user_activity_package.follow_package.FollowDao

/**
 * Activity для отображения и редактирования профиля пользователя.
 */
class ProfileActivity : AppCompatActivity() {

    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase

    /**
     * DAO для работы с пользователями.
     */
    private lateinit var userDao: UserDao

    /**
     * SharedPreferences для хранения данных пользователя.
     */
    private lateinit var sharedPreferences: SharedPreferences

    /**
     * ImageView для отображения аватара пользователя.
     */
    private lateinit var ivAvatar: ImageView

    /**
     * EditText для ввода био пользователя.
     */
    private lateinit var etBio: EditText

    /**
```

```
 * Кнопка для изменения аватара.
 */
private lateinit var btnChangeAvatar: Button

/**
 * Кнопка для сохранения изменений профиля.
 */
private lateinit var btnSaveProfile: Button

/**
 * URI выбранного аватара.
 */
private var avatarUri: Uri? = null

/**
 * Код запроса для выбора изображения.
 */
private val PICK_IMAGE_REQUEST = 1

/**
 * DAO для работы с подписками.
 */
private lateinit var followDao: FollowDao

/**
 * Кнопка для подписки/отписки.
 */
private lateinit var btnFollowUnfollow: Button

/**
 * Идентификатор текущего пользователя.
 */
private var currentUserId = 1

/**
 * Идентификатор просматриваемого пользователя.
 */
private var viewedUserId = 2

/**
 * Вызывается при создании Activity.
 *
 * Инициализирует UI, получает DAO и SharedPreferences, загружает данные пользователя,
 * устанавливает обработчики событий.
 *
 * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
 * этот Bundle содержит данные, которые он ранее сохранил.
 */
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_profile)
```

```
        db = AppDatabase.getDatabase(this)
        userDao = db.userDao()
        sharedPreferences = getSharedPreferences("UserPrefs", Context.MODE_PRIVATE)

        ivAvatar = findViewById(R.id.ivAvatar)
        etBio = findViewById(R.id.etBio)
        btnChangeAvatar = findViewById(R.id.btnChangeAvatar)
        btnSaveProfile = findViewById(R.id.btnSaveProfile)

        val nickname = sharedPreferences.getString("loggedInUser", "") ?: ""

        if (nickname.isNotEmpty()) {
            val user = userDao.getUserByNickname(nickname)
            user?.let {
                etBio.setText(it.bio)
                if (it.avatarUri.isNotEmpty()) {
                    ivAvatar.setImageURI(Uri.parse(it.avatarUri))
                }
            }
        }

        btnChangeAvatar.setOnClickListener {
            val intent = Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
            startActivityForResult(intent, PICK_IMAGE_REQUEST)
        }

        btnSaveProfile.setOnClickListener {
            val bio = etBio.text.toString()
            val avatarUriString = avatarUri?.toString() ?: ""

            if (nickname.isNotEmpty()) {
                userDao.updateProfile(nickname, bio, avatarUriString)
                Toast.makeText(this, "Profile updated!", Toast.LENGTH_SHORT).show()
            }
        }

        db = AppDatabase.getDatabase(this)
        followDao = db.followDao()

        btnFollowUnfollow = findViewById(R.id.btnFollowUnfollow)

        checkFollowStatus()

        btnFollowUnfollow.setOnClickListener {
            toggleFollowStatus()
        }
    }

    /**
     * Обрабатывает результат выбора изображения.
     *
```

```
     * @param requestCode Код запроса.
     * @param resultCode Результат операции.
     * @param data Intent с данными.
     */
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == PICK_IMAGE_REQUEST && resultCode == Activity.RESULT_OK) {
            data?.data?.let { uri ->
                avatarUri = uri
                ivAvatar.setImageURI(uri)
            }
        }
    }

    /**
     * Проверяет статус подписки и устанавливает текст кнопки.
     */
    private fun checkFollowStatus() {
        val follows = followDao.getFollowing(currentUserId)
        val isFollowing = follows.any { it.followedUserId == viewedUserId }

        if (isFollowing) {
            btnFollowUnfollow.text = "Unfollow"
        } else {
            btnFollowUnfollow.text = "Follow"
        }
    }

    /**
     * Переключает статус подписки и обновляет текст кнопки.
     */
    private fun toggleFollowStatus() {
        val follows = followDao.getFollowing(currentUserId)
        val isFollowing = follows.any { it.followedUserId == viewedUserId }

        if (isFollowing) {
            val follow = follows.find { it.followedUserId == viewedUserId }
            follow?.let { followDao.unfollowUser(it) }
            btnFollowUnfollow.text = "Follow"
        } else {
            val follow = Follow(userId = currentUserId, followedUserId = viewedUserId)
            followDao.followUser(follow)
            btnFollowUnfollow.text = "Unfollow"
        }
    }
}
```

Приложение 28 User.kt

```
package com.example.minisocialmedia.user_package

import androidx.room.Entity
import androidx.room.PrimaryKey
import androidx.room.Index

/**
 * Класс данных, представляющий пользователя приложения.
 *
 * @property id Уникальный идентификатор пользователя.
 * @property nickname Имя пользователя (никнейм).
 * @property password Пароль пользователя.
 * @property bio Биография пользователя.
 * @property avatarUri URI аватара пользователя (если есть).
 */
@Entity(
    tableName = "users",
    indices = [Index(value = ["nickname"], unique = true)]
)
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val nickname: String,
    val password: String,
    val bio: String = "",
    val avatarUri: String = ""
)
```

Приложение 29 UserAdapter.kt

```
package com.example.minisocialmedia.user_package
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.R
/**
 * Адаптер для отображения списка пользователей в RecyclerView.
 *
 * @param users Список пользователей для отображения.
 */
class UserAdapter(private val users: List<User>) :
RecyclerView.Adapter<UserAdapter.UserViewHolder>() {
    /**
     * ViewHolder для элементов списка пользователей.
     *
     * @param view Представление элемента списка пользователей.
     */
    class UserViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvUserName: TextView = view.findViewById(R.id.tvUserName)
    }
    /**
     * Создает новый ViewHolder для элемента списка пользователей.
     *
     * @param parent ViewGroup, в который будет добавлен ViewHolder.
     * @param viewType Тип представления нового представления.
     * @return Новый ViewHolder.
     */
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): UserViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_user, parent, false)
        return UserViewHolder(view)
    }
    /**
     * Привязывает данные пользователя к ViewHolder.
     *
     * @param holder ViewHolder, к которому нужно привязать данные.
     * @param position Позиция элемента в списке.
     */
    override fun onBindViewHolder(holder: UserViewHolder, position: Int) {
        val user = users[position]
        holder.tvUserName.text = user.nickname
    }
    /**
     * Возвращает количество элементов в списке пользователей.
     *
     * @return Количество элементов в списке.
     */
    override fun getItemCount() = users.size
}
```

Приложение 30 UserDao.kt

```
package com.example.minisocialmedia.user_package

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query

/**
 * DAO для работы с пользователями.
 */
@Dao
interface UserDao {

    /**
     * Вставляет нового пользователя в базу данных или заменяет существующего, если никнейм совпадает.
     *
     * @param user Пользователь для вставки или замены.
     */
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertUser(user: User)

    /**
     * Получает пользователя по никнейму и паролю.
     *
     * @param nickname Никнейм пользователя.
     * @param password Пароль пользователя.
     * @return Пользователь, если найден, иначе null.
     */
    @Query("SELECT * FROM users WHERE nickname = :nickname AND password = :password")
    fun getUser(nickname: String, password: String): User?

    /**
     * Получает пользователя по никнейму.
     *
     * @param nickname Никнейм пользователя.
     * @return Пользователь, если найден, иначе null.
     */
    @Query("SELECT * FROM users WHERE nickname = :nickname")
    fun getUserByNickname(nickname: String): User?

    /**
     * Обновляет профиль пользователя.
     *
     * @param nickname Никнейм пользователя.
     * @param bio Биография пользователя.
     * @param avatarUri URI аватара пользователя.
     */
```

```
    @Query("UPDATE users SET bio = :bio, avatarUri = :avatarUri WHERE nickname =
:nickname")
    fun updateProfile(nickname: String, bio: String, avatarUri: String)

    /**
     * Получает список пользователей по их идентификаторам.
     *
     * @param userIds Список идентификаторов пользователей.
     * @return Список пользователей.
     */
    @Query("SELECT * FROM users WHERE id IN (:userIds)")
    fun getUsersByIds(userIds: List<Int>): List<User>
}
```

Приложение 31 MainActivity.kt

```
package com.example.minisocialmedia

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.minisocialmedia.database_package.AppDatabase
import com.example.minisocialmedia.log_activity_package.LoginActivity
import com.example.minisocialmedia.user_activity_package.post_package.CreatePostActivity
import com.example.minisocialmedia.user_activity_package.post_package.PostAdapter
import com.example.minisocialmedia.user_activity_package.post_package.PostDao
import com.example.minisocialmedia.user_package.ProfileActivity

/**
 * Activity для отображения основной ленты постов.
 */
class MainActivity : AppCompatActivity() {
    /**
     * Экземпляр базы данных приложения.
     */
    private lateinit var db: AppDatabase
    /**
     * DAO для работы с постами.
     */
    private lateinit var postDao: PostDao
    /**
     * RecyclerView для отображения списка постов.
     */
    private lateinit var rvPosts: RecyclerView
    /**
     * Кнопка для создания нового поста.
     */
    private lateinit var btnCreatePost: Button

    /**
     * Кнопка для выхода из аккаунта.
     */
    private lateinit var btnLogout: Button

    /**
     * Кнопка для перехода в профиль.
     */
    private lateinit var btnProfile: Button

    /**
     * Вызывается при создании Activity.
     *
```

```
    * Инициализирует UI, получает DAO, устанавливает обработчики событий и загружает
посты.
    *
    * @param savedInstanceState Если Activity пересоздается после предыдущего завершения,
    * этот Bundle содержит данные, которые он ранее сохранил.
    */
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)

      db = AppDatabase.getDatabase(this)
      postDao = db.postDao()

      rvPosts = findViewById(R.id.rvPosts)
      btnCreatePost = findViewById(R.id.btnCreatePost)
      btnLogout = findViewById(R.id.btnLogout)
      btnProfile = findViewById(R.id.btnProfile)

      btnCreatePost.setOnClickListener {
         startActivity(Intent(this, CreatePostActivity::class.java))
      }

      btnLogout.setOnClickListener {
         startActivity(Intent(this, LoginActivity::class.java))
         finish()
      }

      btnProfile.setOnClickListener {
         startActivity(Intent(this, ProfileActivity::class.java))
      }

      loadPosts()
   }

   /**
    * Вызывается при возобновлении Activity.
    *
    * Обновляет список постов.
    */
   override fun onResume() {
      super.onResume()
      loadPosts()
   }
   /**
    * Загружает и отображает список постов.
    */
   private fun loadPosts() {
      val posts = postDao.getAllPosts()
      rvPosts.layoutManager = LinearLayoutManager(this)
      rvPosts.adapter = PostAdapter(posts, postDao)
   }
}
```

Приложение 32 AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="minisocialmedia"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MiniSocialMedia"
        tools:targetApi="31">
        <activity
            android:name=".log_activity_package.SignupActivity"
            android:exported="false" />
        <activity
            android:name=".log_activity_package.LoginActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="minisocialmedia"
            android:theme="@style/Theme.MiniSocialMedia">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
</manifest>
```

Приложение 33 activity_chat.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/light_lavender"
    android:padding="8dp">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvMessages"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/lavender"
        android:padding="4dp"
        android:layout_margin="4dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@color/light_lavender"
        android:padding="8dp">

        <EditText
            android:id="@+id/etMessage"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Type a message"
            android:background="@color/lavender"
            android:padding="8dp"
            android:layout_marginEnd="4dp"/>

        <Button
            android:id="@+id/btnSend"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Send"
            android:background="@color/violet"
            android:textColor="@android:color/white"/>
    </LinearLayout>
</LinearLayout>
```

Приложение 34 activity_comment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/light_lavender"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@color/light_lavender"
        android:padding="8dp">

        <EditText
            android:id="@+id/etComment"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Write a comment..."
            android:background="@color/lavender"
            android:padding="10dp"
            android:layout_marginBottom="8dp"/>

        <Button
            android:id="@+id/btnSubmitComment"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Post Comment"
            android:background="@color/violet"
            android:textColor="@android:color/white"
            android:layout_gravity="end"
            android:padding="8dp"/>

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rvComments"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:background="@color/lavender"
            android:padding="4dp"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 35 activity_create_post.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/light_lavender">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:gravity="center"
        android:background="@color/lavender"
        tools:ignore="MissingConstraints">

        <EditText
            android:id="@+id/etPostContent"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="What's on your mind?"
            android:minHeight="100dp"
            android:background="@color/light_lavender"
            android:padding="8dp"
            android:layout_marginBottom="10dp"/>

        <ImageView
            android:id="@+id/ivPostImage"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:visibility="gone"
            android:scaleType="centerCrop"
            android:layout_marginTop="10dp"
            android:background="@color/light_lavender"/>

        <Button
            android:id="@+id/btnSelectImage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Add Image"
            android:background="@color/violet"
            android:textColor="@android:color/white"
            android:layout_marginTop="10dp"/>

        <Button
            android:id="@+id/btnPost"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Post"
```

```
        android:background="@color/violet"
        android:textColor="@android:color/white"
        android:layout_marginTop="20dp"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 36 activity_following.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/soft_blue">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvFollowing"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="16dp"
        android:background="@color/mint_green"
        android:padding="8dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 37 activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/soft_blue"
    tools:context=".log_activity_package.LoginActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:background="@color/mint_green"
        tools:ignore="MissingConstraints">

        <EditText
            android:id="@+id/etLoginNickname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Nickname"
            android:background="@color/soft_blue"
            android:padding="8dp"
            android:layout_marginBottom="10dp"/>

        <EditText
            android:id="@+id/etLoginPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Password"
            android:background="@color/soft_blue"
            android:padding="8dp"
            android:layout_marginBottom="10dp"/>

        <Button
            android:id="@+id/btnLogin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Log In"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 38 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/soft_blue">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center"
        android:padding="16dp"
        android:background="@color/mint_green">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Welcome to the Social Media App!"
            android:textSize="18sp"
            android:textStyle="bold"
            android:textColor="@color/dark_blue"/>

        <Button
            android:id="@+id/btnLogout"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Logout"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"
            android:layout_marginTop="20dp"/>

        <Button
            android:id="@+id/btnProfile"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Go to Profile"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"
            android:layout_marginTop="20dp"/>

        <Button
            android:id="@+id/btnCreatePost"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Create Post"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"
            android:layout_marginBottom="10dp"/>
```

```
        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rvPosts"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/soft_blue"
            android:padding="8dp"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 39 activity_music.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/bright_blue">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:background="@color/vibrant_green"
        tools:ignore="MissingConstraints">

        <Button
            android:id="@+id/btnSelectMusic"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Select Music"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 40 activity_music_player.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:padding="16dp"
    android:background="@color/vibrant_green">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/musicListView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/btnStopMusic"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:background="@color/bright_blue"/>

    <Button
        android:id="@+id/btnStopMusic"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Stop Music"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:background="@color/dark_blue"
        android:textColor="@android:color/white"/>

    <Button
        android:id="@+id/btnChooseMusic"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Choose Music"
        android:background="@color/dark_blue"
        android:textColor="@android:color/white"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 41 activity_profile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/soft_blue">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="16dp"
        android:gravity="center"
        android:background="@color/mint_green">

        <ImageView
            android:id="@+id/ivAvatar"
            android:layout_width="120dp"
            android:layout_height="120dp"
            android:scaleType="centerCrop"
            android:src="@drawable/ic_profile_placeholder"
            android:background="@color/soft_blue"
            android:padding="4dp"/>

        <Button
            android:id="@+id/btnChangeAvatar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Change Avatar"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"
            android:layout_marginTop="10dp"/>

        <EditText
            android:id="@+id/etBio"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Enter your bio"
            android:background="@color/soft_blue"
            android:padding="8dp"
            android:layout_marginTop="20dp"/>

        <Button
            android:id="@+id/btnSaveProfile"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Save"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"
            android:layout_marginTop="20dp"/>
```

```
        <Button
            android:id="@+id/btnFollowUnfollow"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Follow"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"
            android:layout_gravity="center_horizontal"
            android:padding="8dp"
            android:layout_marginTop="16dp"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 42 activity_signup.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/soft_blue"
    tools:context=".log_activity_package.SignupActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:background="@color/mint_green"
        tools:ignore="MissingConstraints">

        <EditText
            android:id="@+id/etNickname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Enter Nickname"
            android:background="@color/soft_blue"
            android:padding="8dp"
            android:layout_marginBottom="10dp"/>

        <EditText
            android:id="@+id/etPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Enter Password"
            android:background="@color/soft_blue"
            android:padding="8dp"
            android:layout_marginBottom="10dp"/>

        <Button
            android:id="@+id/btnSignup"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Sign Up"
            android:background="@color/dark_blue"
            android:textColor="@android:color/white"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 43 item_comment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="8dp"
        android:background="@android:color/white"
        android:layout_marginBottom="4dp"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/tvAuthor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:text="Username"/>

        <TextView
            android:id="@+id/tvComment"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="This is a comment."
            android:paddingTop="4dp"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 44 item_message.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/light_lavender">

    <TextView
        android:id="@+id/tvMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="8dp"
        android:textSize="16sp"
        android:textColor="@color/violet"
        tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 45 item_music.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="10dp"
    android:background="@color/bright_blue">

    <TextView
        android:id="@+id/tvMusicTitle"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="wrap_content"
        android:text="Music Title"
        android:textSize="16sp"
        android:textStyle="bold"
        android:textColor="@color/dark_blue"/>

    <TextView
        android:id="@+id/tvMusicDuration"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="00:00"
        android:textSize="14sp"
        android:textColor="@color/dark_blue"/>
</LinearLayout>
```

Приложение 46 item_post.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/light_lavender">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dp"
        android:background="@color/lavender"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/tvAuthor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textColor="@color/violet"/>

        <TextView
            android:id="@+id/tvContent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/violet"/>

        <ImageView
            android:id="@+id/ivPostImage"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:visibility="gone"
            android:background="@color/light_lavender"/>

        <TextView
            android:id="@+id/tvLikes"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="0 Likes"
            android:textColor="@color/violet"/>

        <Button
            android:id="@+id/btnLike"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Like"
            android:background="@color/violet"
            android:textColor="@android:color/white"/>
```

```
        <Button
            android:id="@+id/btnComment"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Comment"
            android:background="@color/violet"
            android:textColor="@android:color/white"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Приложение 47 item_user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="8dp"
    android:background="@color/soft_blue">

    <TextView
        android:id="@+id/tvUserName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="User Name"
        android:textSize="16sp"
        android:textStyle="bold"
        android:textColor="@color/dark_blue"
        android:layout_marginStart="8dp"/>
</LinearLayout>
```

Приложение 48 colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>

    <color name="light_lavender">#FFF0F5</color>
    <color name="violet">#EE82EE</color>
    <color name="lavender">#E6E6FA</color>

    <color name="soft_blue">#ADD8E6</color>
    <color name="mint_green">#98FB98</color>
    <color name="dark_blue">#00008B</color>

    <color name="bright_blue">#4682B4</color>
    <color name="vibrant_green">#32CD32</color>
</resources>
```

Приложение 49 strings.xml

```
<resources>
    <string name="app_name">minisocialmedia</string>
</resources>
```

Приложение 50 themes.xml

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

    <style name="Theme.MiniSocialMedia" parent="android:Theme.Material.Light.NoActionBar"
/>
</resources>
```

Приложение 51 build.gradle.kts Project level

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
plugins {
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.jetbrains.kotlin.android) apply false
}
```

Приложение 52 build.gradle.kts Module level

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    alias(libs.plugins.ksp)
}

android {
    namespace = "com.example.minisocialmedia"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.minisocialmedia"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary = true
        }
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
    buildFeatures {
        compose = true
    }
    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.1"
    }
    packaging {
        resources {
            excludes += "/META-INF/{AL2.0,LGPL2.1}"
        }
    }
```

```
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)

    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")

    implementation("androidx.room:room-runtime:2.6.1")

    implementation(libs.room.runtime)
    ksp(libs.room.compiler)

}
```

Приложение 53 libs.versions.toml

```
[versions]
agp = "8.5.0-rc01"
kotlin = "1.9.0"
coreKtx = "1.15.0"
junit = "4.13.2"
junitVersion = "1.2.1"
espressoCore = "3.6.1"
lifecycleRuntimeKtx = "2.8.7"
activityCompose = "1.10.1"
composeBom = "2023.08.00"
ksp = "1.9.0-1.0.13"
appcompat = "1.7.0"
material = "1.12.0"
activity = "1.10.1"
constraintlayout = "2.2.1"

[libraries]
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-core", version.ref =
"espressoCore" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-ktx",
version.ref = "lifecycleRuntimeKtx" }
androidx-activity-compose = { group = "androidx.activity", name = "activity-compose", version.ref
= "activityCompose" }
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref =
"composeBom" }
androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-tooling-preview" }
androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-manifest" }
androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-junit4" }
androidx-material3 = { group = "androidx.compose.material3", name = "material3" }
room-runtime = { group = "androidx.room", name = "room-runtime", version = "2.6.1" }
room-compiler = { group = "androidx.room", name = "room-compiler", version = "2.6.1" }
androidx-appcompat = { group = "androidx.appcompat", name = "appcompat", version.ref =
"appcompat" }
material = { group = "com.google.android.material", name = "material", version.ref = "material" }
androidx-activity = { group = "androidx.activity", name = "activity", version.ref = "activity" }
androidx-constraintlayout = { group = "androidx.constraintlayout", name = "constraintlayout",
version.ref = "constraintlayout" }

[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
jetbrains-kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
ksp = { id = "com.google.devtools.ksp", version.ref = "ksp" }
```

Приложение 54 settings.gradle.kts Project Settings

```
pluginManagement {
    repositories {
        google {
            content {
                includeGroupByRegex("com\\.android.*")
                includeGroupByRegex("com\\.google.*")
                includeGroupByRegex("androidx.*")
            }
        }
        mavenCentral()
        gradlePluginPortal()
    }
}
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.name = "minisocialmedia"
include(":app")
```