Отчет по уп 1.03

Практическая работа №1

Создание приложения ''Электронный конспект'' на Jetpack Compose.

**1.1. Подключение библиотек**

kotlin

plugins **{** id("com.android.application")  
 id("org.jetbrains.kotlin.android")  
 *kotlin*("kapt")  
**}***android* **{** namespace = "com.example.electronicnotes"  
 compileSdk = 34  
  
 defaultConfig **{** applicationId = "com.example.electronicnotes"  
 minSdk = 26  
 targetSdk = 34  
 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* **{** // Core Android dependencies  
 *implementation*("androidx.core:core-ktx:1.13.1")  
 *implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.8.6")  
   
 // Compose dependencies  
 *implementation*("androidx.activity:activity-compose:1.9.2")  
 *implementation*(platform("androidx.compose:compose-bom:2023.08.00"))  
 *implementation*("androidx.compose.ui:ui")  
 *implementation*("androidx.compose.ui:ui-graphics")  
 *implementation*("androidx.compose.ui:ui-tooling-preview")  
 *implementation*("androidx.compose.material3:material3")  
 *implementation*("androidx.compose.material:material-icons-extended:1.5.4")  
   
 // Room  
 val roomVersion = "2.6.0"  
 *implementation*("androidx.room:room-runtime:$roomVersion")  
 *implementation*("androidx.room:room-ktx:$roomVersion")  
 *kapt*("androidx.room:room-compiler:$roomVersion")  
  
 // Coroutines  
 *implementation*("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3")  
 *implementation*("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.7.3")  
  
 // Lifecycle components  
 *implementation*("androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.2")  
 *implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")  
 *implementation*("androidx.lifecycle:lifecycle-livedata-ktx:2.6.2")  
  
 // Jetpack Navigation for Compose  
 *implementation*("androidx.navigation:navigation-compose:2.5.0")  
   
 // Testing dependencies  
 *testImplementation*("junit:junit:4.13.2")  
 *androidTestImplementation*("androidx.test.ext:junit:1.2.1")  
 *androidTestImplementation*("androidx.test.espresso:espresso-core:3.6.1")  
 *androidTestImplementation*(platform("androidx.compose:compose-bom:2023.08.00"))  
 *androidTestImplementation*("androidx.compose.ui:ui-test-junit4")  
   
 // Debug dependencies  
 *debugImplementation*("androidx.compose.ui:ui-tooling")  
 *debugImplementation*("androidx.compose.ui:ui-test-manifest")  
**}**

**1.Плагины (Plugins)**

* com.android.application: Плагин для создания Android-приложения.
* org.jetbrains.kotlin.android: Плагин для поддержки Kotlin в Android-проекте.
* kotlin("kapt"): Плагин для Kotlin Annotation Processing Tool (kapt), который используется для обработки аннотаций в Kotlin.

kotlin

**2. Настройки Android (Android)**

namespace: Пространство имен для приложения.

* compileSdk: Версия SDK, которая будет использоваться для компиляции приложения.
* defaultConfig: Основные настройки приложения:
  + applicationId: Идентификатор приложения.
  + minSdk: Минимальная версия SDK, поддерживаемая приложением.
  + targetSdk: Целевая версия SDK.
  + versionCode: Номер версии приложения.
  + versionName: Имя версии приложения.
  + testInstrumentationRunner: Исполнитель тестов для инструментальных тестов.
  + vectorDrawables: Настройки для векторных изображений.
* buildTypes: Настройки для различных типов сборок (например, release).
  + isMinifyEnabled: Включение минификации кода.
  + proguardFiles: Файлы конфигурации ProGuard для оптимизации и обфускации кода.
* compileOptions: Настройки компиляции:
  + sourceCompatibility: Версия Java для исходного кода.
  + targetCompatibility: Версия Java для целевой платформы.
* kotlinOptions: Настройки для Kotlin:
  + jvmTarget: Версия JVM для Kotlin.
* buildFeatures: Включение функций сборки:
  + compose: Включение Jetpack Compose.
* composeOptions: Настройки для Jetpack Compose:
  + kotlinCompilerExtensionVersion: Версия расширения компилятора Kotlin для Compose.
* packaging: Настройки упаковки:
  + resources: Исключение определенных ресурсов из упаковки.

**3. Зависимости (Dependencies)**

* **Core Android dependencies**: Основные зависимости для работы с Android.
  + androidx.core:core-ktx: Основные расширения Kotlin для Android.
  + androidx.lifecycle:lifecycle-runtime-ktx: Зависимости для работы с жизненным циклом компонентов.
* **Compose dependencies**: Зависимости для Jetpack Compose.
  + androidx.activity:activity-compose: Поддержка Compose в Activity.
  + androidx.compose:compose-bom: Билд-платформа для Compose.
  + androidx.compose.ui:ui: Основные компоненты UI для Compose.
  + androidx.compose.ui:ui-graphics: Графические компоненты для Compose.
  + androidx.compose.ui:ui-tooling-preview: Инструменты для предварительного просмотра Compose.
  + androidx.compose.material3:material3: Material Design 3 для Compose.
  + androidx.compose.material:material-icons-extended: Расширенные иконки Material Design.
* **Room**: Зависимости для работы с базой данных Room.
  + androidx.room:room-runtime: Основная библиотека Room.
  + androidx.room:room-ktx: Расширения Kotlin для Room.
  + kapt("androidx.room:room-compiler"): Компилятор Room для обработки аннотаций.
* **Coroutines**: Зависимости для работы с корутинами.
  + org.jetbrains.kotlinx:kotlinx-coroutines-android: Корутины для Android.
  + org.jetbrains.kotlinx:kotlinx-coroutines-core: Основная библиотека корутин.
* **Lifecycle components**: Зависимости для работы с жизненным циклом компонентов.
  + androidx.lifecycle:lifecycle-viewmodel-ktx: ViewModel для Kotlin.
  + androidx.lifecycle:lifecycle-runtime-ktx: Зависимости для работы с жизненным циклом.
  + androidx.lifecycle:lifecycle-livedata-ktx: LiveData для Kotlin.
* **Jetpack Navigation for Compose**: Зависимости для навигации в Compose.
  + androidx.navigation:navigation-compose: Навигация для Compose.
* **Testing dependencies**: Зависимости для тестирования.
  + junit:junit: Библиотека для модульного тестирования.
  + androidx.test.ext:junit: Расширения JUnit для Android.
  + androidx.test.espresso:espresso-core: Основная библиотека Espresso для инструментального тестирования.
  + androidx.compose.ui:ui-test-junit4: Тестирование Compose с использованием JUnit4.
* **Debug dependencies**: Зависимости для отладки.
  + androidx.compose.ui:ui-tooling: Инструменты для отладки Compose.

androidx.compose.ui:ui-test-manifest: Манифест для тестирования Compose.

**1.2Создание сущности (Entity)**

package com.example.electronicnotes.data  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "notes")  
data class Note(  
 @PrimaryKey(autoGenerate = true) val id: Int = 0,  
 val title: String,  
 val content: String,  
 val type: NoteType = NoteType.*HERESY*)

**1. Пакет (Package)**

* **Пакет**: Определяет пространство имен, в котором находится класс Note. В данном случае, класс находится в пакете com.example.electronicnotes.data.

**2. Импорты (Imports)**

kotlin

ey

* **Импорты**: Импортируются аннотации из библиотеки Room, которые будут использоваться для определения сущности базы данных.
  + androidx.room.Entity: Аннотация, которая указывает, что класс является сущностью базы данных.
  + androidx.room.PrimaryKey: Аннотация, которая указывает, что поле является первичным ключом.

**3. Класс Note**

* **Аннотация @Entity**: Указывает, что класс Note является сущностью базы данных.
  + tableName = "notes": Определяет имя таблицы в базе данных, которая будет создана для хранения объектов этого класса. В данном случае, таблица будет называться notes.
* **Класс Note**: Это класс данных (data class), который содержит поля, представляющие столбцы в таблице notes.
  + @PrimaryKey(autoGenerate = true) val id: Int = 0: Определяет поле id как первичный ключ.
    - autoGenerate = true: Указывает, что значения для этого поля будут автоматически генерироваться базой данных.
    - val id: Int = 0: Поле id имеет тип Int и по умолчанию равно 0.
  + val title: String: Поле title имеет тип String и представляет заголовок заметки.
  + val content: String: Поле content имеет тип String и представляет содержимое заметки.
  + val type: NoteType = NoteType.HERESY: Поле type имеет тип NoteType (предположительно, это перечисление) и по умолчанию равно NoteType.HERESY.

**4. Перечисление NoteType**

Предположительно, NoteType — это перечисление (enum), которое определяет типы заметок.

package com.example.electronicnotes.data  
  
import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
import androidx.room.TypeConverters  
import com.example.electronicnotes.data.Note  
import com.example.electronicnotes.data.NoteDao  
  
// Определение базы данных с помощью аннотации @Database  
@Database(entities = [Note::class], version = 2, exportSchema = false)  
@TypeConverters(Converters::class)  
abstract class NoteDatabase : RoomDatabase() {  
 // Абстрактная функция для получения объекта NoteDao  
 abstract fun noteDao(): NoteDao  
  
 // Компаньон-объект для доступа к базе данных  
 companion object {  
 // Волатильная переменная для хранения единственного экземпляра базы данных  
 @Volatile  
 private var INSTANCE: NoteDatabase? = null  
  
 // Функция для получения экземпляра базы данных  
 fun getDatabase(context: Context): NoteDatabase {  
 // Если экземпляр базы данных уже существует, возвращаем его  
 return INSTANCE ?: *synchronized*(this) **{** // Если экземпляр базы данных еще не создан, создаем его  
 val instance = Room.databaseBuilder(  
 context.*applicationContext*,  
 NoteDatabase::class.*java*,  
 "note\_database"  
 )  
 .fallbackToDestructiveMigration() // При изменении схемы БД пересоздаст её  
 .build()  
 INSTANCE = instance  
 instance  
 **}** }  
 }  
}

**1. Пакет (Package)**

ta

* **Пакет**: Определяет пространство имен, в котором находится класс NoteDatabase. В данном случае, класс находится в пакете com.example.electronicnotes.data.

**2. Импорты (Imports)**

kotlin

Copy

* **Импорты**: Импортируются необходимые классы и аннотации из библиотеки Room, а также другие классы, которые будут использоваться в коде.
  + android.content.Context: Контекст приложения.
  + androidx.room.Database: Аннотация, которая указывает, что класс является базой данных.
  + androidx.room.Room: Класс, предоставляющий методы для создания и управления базой данных.
  + androidx.room.RoomDatabase: Абстрактный класс, который представляет базу данных Room.
  + androidx.room.TypeConverters: Аннотация, которая указывает, какие конвертеры типов использовать.
  + com.example.electronicnotes.data.Note: Класс, представляющий сущность базы данных.
  + com.example.electronicnotes.data.NoteDao: Интерфейс, предоставляющий методы для работы с базой данных.

**3. Класс NoteDatabase**

kotlin

* **Аннотация @Database**: Указывает, что класс NoteDatabase является базой данных.
  + entities = [Note::class]: Определяет сущности (таблицы), которые будут включены в базу данных. В данном случае, это класс Note.
  + version = 2: Указывает версию схемы базы данных. При изменении схемы, версия должна быть увеличена.
  + exportSchema = false: Отключает экспорт схемы базы данных в файл.
* **Аннотация @TypeConverters**: Указывает, какие конвертеры типов использовать для преобразования сложных типов данных в типы, поддерживаемые базой данных. В данном случае, используется класс Converters.
* **Класс NoteDatabase**: Это абстрактный класс, который наследуется от RoomDatabase.
  + abstract fun noteDao(): NoteDao: Абстрактная функция, которая возвращает объект NoteDao. Этот объект будет использоваться для выполнения операций с базой данных.
* **Компаньон-объект (Companion Object)**: Используется для создания и управления единственным экземпляром базы данных.
  + @Volatile private var INSTANCE: NoteDatabase? = null: Волатильная переменная, которая хранит единственный экземпляр базы данных. Волатильность гарантирует, что изменения в переменной будут видны всем потокам.
  + fun getDatabase(context: Context): NoteDatabase: Функция для получения экземпляра базы данных. Если экземпляр уже существует, возвращает его. Если нет, создает новый экземпляр.
    - synchronized(this): Используется для синхронизации доступа к блоку кода, чтобы предотвратить создание нескольких экземпляров базы данных в многопоточной среде.
    - Room.databaseBuilder: Создает билдер для базы данных.
      * context.applicationContext: Контекст приложения.
      * NoteDatabase::class.java: Класс базы данных.
      * "note\_database": Имя базы данных.
    - .fallbackToDestructiveMigration(): При изменении схемы базы данных пересоздает её, удаляя все данные.

.build(): Создает и возвращает экземпляр базы данных.