

14

Architecture Patterns

Activity 14.1: Revisiting the TV Guide App

Solution:

You can use the *TV Guide* app you developed in the *Chapter 10, Coroutines and Flow* or make a copy of it. Here is one way you can improve the app using the **Model-View-ViewModel (MVVM)** architectural pattern with the Repository pattern using Room:

1. Open the *TV Guide* in Android Studio.
2. Open the project `build.gradle.kts` file and add the plugin for KSP:

```
plugins {  
    ...  
    alias(libs.plugins.ksp) apply false  
}
```

3. Go to the `app/build.gradle.kts` file and add the KSP plugin at the end of the `plugins` block:

```
plugins {  
    ...  
    id("com.google.devtools.ksp")  
}
```

4. Add the dependencies for the Room library:

```
implementation(libs.androidx.room.ktx)  
ksp(libs.androidx.room.compiler)
```

5. Open the TVShow class and add an Entity annotation for it and add a PrimaryKey annotation to the id property:

```
@Entity(tableName = "tvshows")
data class TVShow(
    ...
    @PrimaryKey val id: Int = 0,
    ...
) @PrimaryKey val id: Int = 0,
...
)
```

This creates a tvShows table for the list of TV shows with id as the primary key.

6. Create a TVShowDao data access object for accessing the TV shows table in a new package called com.example.tvguide.database:

```
@Dao
interface TVShowDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun addTVShows(tvShows: List<TVShow>)

    @Query("SELECT * FROM tvshows")
    fun getTVShows(): Flow<List<TVShow>>
}
```

This class has a function for getting the list of TV shows from the database and another function for adding a list to the database.

7. Create a TVShowDatabase class in the com.example.tvguide.database package:

```
@Database(entities = [TVShow::class], version = 1)
abstract class TVShowDatabase : RoomDatabase() {

    abstract fun tvShowDao(): TVShowDao

    companion object {
        @Volatile
        private var instance: TVShowDatabase? = null
        fun getInstance(context: Context):
            TVShowDatabase
    }
}
```

```

        {
            return instance ?: synchronized(this) {
                instance ?: buildDatabase(context).also {
                    instance = it
                }
            }
        }
    }

    private fun buildDatabase(context: Context):
        TVShowDatabase
    {
        return Room.databaseBuilder(
            context,
            TVShowDatabase::class.java,
            "tvshows-db"
        ).build()
    }
}

```

This database has a version of 1, a single entity for TVShow, and a data access object for the TV shows.

8. Update the TVShowRepository class with a constructor for tvShowDatabase:

```

class TVShowRepository(
    private val tvShowService: TVShowService,
    private val tvShowDatabase: TVShowDatabase
) {
    ...
}

```

9. Update the getTVShows function to get the TV shows from the database and retrieve the list from the endpoint then save it:

```

suspend fun getTVShows(): Flow<List<TVShow>> {
    val tvShowDao: TVShowDao =
        tvShowDatabase.tvShowDao()

    CoroutineScope(SupervisorJob() +

```

```
        Dispatchers.IO).launch
    {
        try {
            val tvShows = tvShowService.getTVShows()
            tvShowDao.addTVShows(tvShows)
        } catch (exception: Exception) {
            Log.e("TVShowRepository",
                exception.toString())
        }
    }
    return tvShowDao.getTVShows()
}
```

10. Open the TVShowApplication file and on the onCreate function, replace the initialization of the tvShowRepository with the following:

```
val tvShowDatabase =
    TVShowDatabase.getInstance(applicationContext)

tvShowRepository = TVShowRepository(
    tvShowService = tvShowService,
    tvShowDatabase = tvShowDatabase
)
```

11. Run your application. It will display a list of TV shows. If you turn off mobile data or disconnect from the wireless network, you will still see the list because it is now cached in the database:

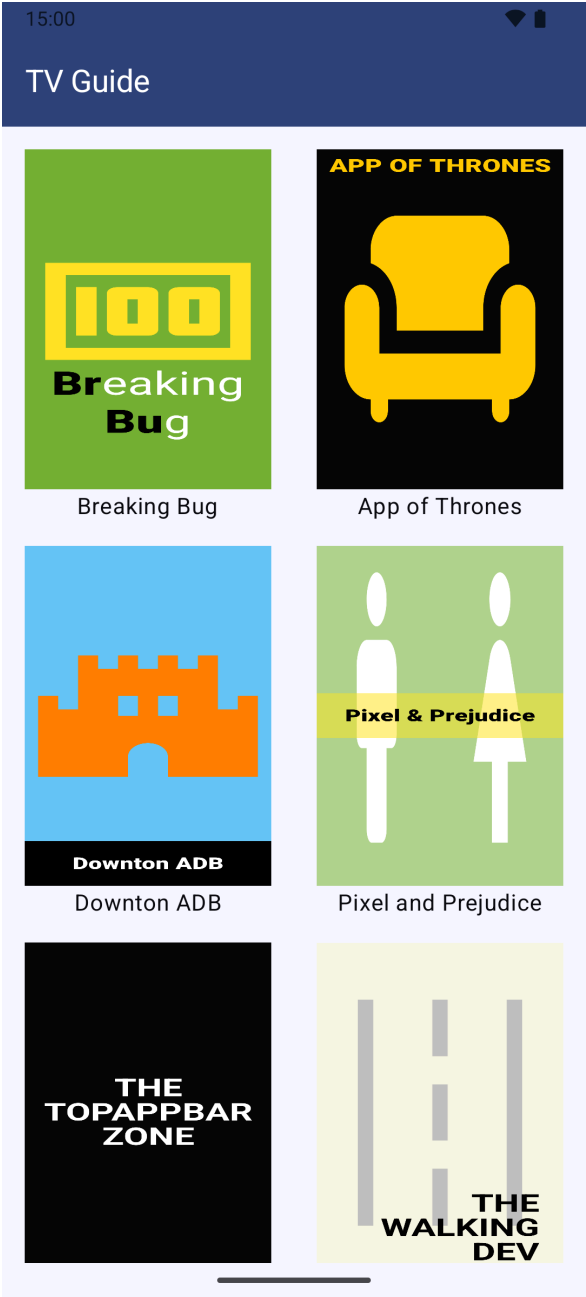


Figure 14.5 – The main screen of the TV Guide app with the list of TV shows

12. When you click on a TV show, the details screen will be displayed:

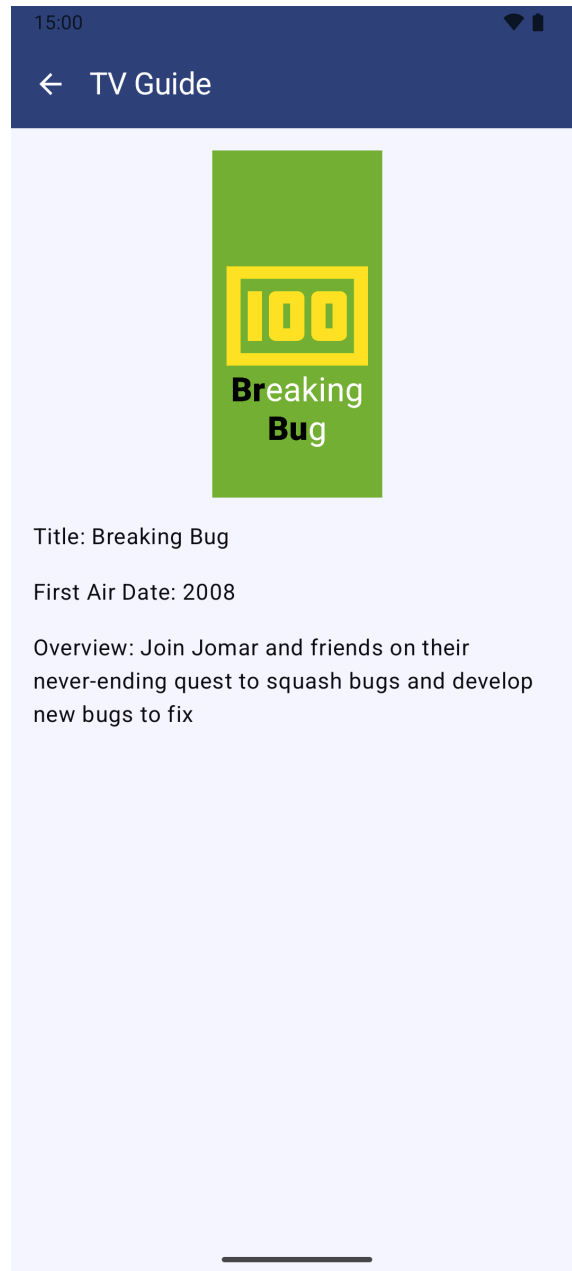


Figure 14.6 – The details screen showing more information about the chosen TV show

You have cached the list of TV shows in the local database.