

10

Coroutines and Flow

Activity 10.1 – creating a TV guide app

Solution

Here is one way you can develop the TV guide app:

1. Create a new project in Android Studio named TV Guide with a package name of `com.example.tvguide`.
2. Add the `INTERNET` permission in the `AndroidManifest.xml` file, inside the `manifest` tag but outside the `application` tag:

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

3. Add Ktor, Coroutines, `kotlinx.serialization`, and other libraries to your project by adding the following to your `app/build.gradle.kts` file:

```
...  
implementation(libs.coil.compose)  
implementation(libs.ktor.client.android)  
implementation(libs.ktor.client.content.negotiation)  
implementation(libs.ktor.client.serialization)  
implementation(libs.kotlinx.serialization.json)  
implementation(libs.ktor.serialization.kotlinx.json)  
implementation(libs.kotlinx.coroutines.android)  
implementation(libs.kotlinx.coroutines.core)
```

4. Open the `strings.xml` file and add the following string values:

```
<resources>
  <string name="app_name">TV Guide</string>

  <string name="back">Back</string>

  <string
    name="tv_show_overview">
    Overview: %s
  </string>
  <string name="tv_show_poster">Poster</string>
  <string
    name="tv_show_release">First Air Date: %s
  </string>
  <string name="tv_show_title">Title: %s</string>
</resources>
```

5. Create a new file called `TVShow` in a new package, `com.example.tvguide.model`, with a model class named `TVShow`:

```
@Serializable
data class TVShow(
  @SerializedName("backdrop_path")
  val backdropPath: String? = "",
  @SerializedName("first_air_date")
  val firstAirDate: String? = "",
  val id: Int = 0,
  val name: String = "",
  @SerializedName("original_language")
  val originalLanguage: String = "",
  @SerializedName("original_name")
  val originalName: String = "",
  val overview: String = "",
  val popularity: Float = 0f,
  @SerializedName("poster_path")
  val posterPath: String? = "",
  @SerializedName("vote_average")
  val voteAverage: Float = 0f,
```

```
@SerializedName("vote_count")
val voteCount: Int = 0
)
```

This will be the model class representing a TV show object from the API.

6. Create another class, named `TVShowResponse`, in the same file:

```
@Serializable
data class TVShowResponse(
    val page: Int,
    val results: List<TVShow>
)
```

This will be the model class for the response you get from the API endpoint, which contains the list of TV shows on air.

7. Create a new file called `DetailsScreen` and add a `DetailsScreen` composable function:

```
@Composable
fun DetailsScreen(
    title: String,
    release: String,
    overview: String,
    image: String,
    modifier: Modifier = Modifier,
    onBackButtonClick: () -> Unit
) {
    ...
}
```

This will be for the details screen of the app.

8. Inside the `DetailsScreen` composable, add the following to display the details of the TV show:

```
Scaffold(
    modifier = modifier.fillMaxSize(),
    topBar = {
        TopAppBar(
            title = {
```

```
        Text(
            text = stringResource(
                id = R.string.app_name
            ),
            color = MaterialTheme.colorScheme
                .onPrimary
        )
    },
    colors = topAppBarColors(
        MaterialTheme.colorScheme.primary
    ),
    navigationIcon = {
        IconButton(
            onClick = {
                onBackButtonClick()
            }
        ) {
            Icon(
                imageVector = Icons
                    .AutoMirrored
                    .Filled.ArrowBack,
                contentDescription =
                    stringResource(
                        id = R.string.back
                    ),
                tint = MaterialTheme
                    .colorScheme
                    .onPrimary
            )
        }
    }
) {
    innerPadding ->
        //TODO
}
```

9. Inside the content block, replace the TODO comment with the following code:

```
Column(  
    verticalArrangement = Arrangement.spacedBy(16.dp),  
    horizontalAlignment = Alignment  
        .CenterHorizontally,  
    modifier = Modifier  
        .padding(innerPadding)  
        .fillMaxWidth()  
        .padding(16.dp)  
) {  
    AsyncImage(  
        model = image,  
        contentDescription = stringResource(  
            id = R.string.tv_show_poster  
        ),  
        contentScale = ContentScale.Fit,  
        placeholder = painterResource(  
            id = R.drawable.ic_launcher_foreground  
        ),  
    )  
    Text(  
        text = stringResource(  
            id = R.string.tv_show_title, title  
        ),  
        overflow = TextOverflow.Ellipsis,  
        modifier = Modifier.fillMaxWidth()  
    )  
    Text(  
        text = stringResource(  
            id = R.string.tv_show_release, release  
        ),  
        overflow = TextOverflow.Ellipsis,  
        modifier = Modifier.fillMaxWidth()  
    )  
    Text(  
        text = stringResource(  
            id = R.string.tv_show_overview, overview
```

```

    ),
    overflow = TextOverflow.Ellipsis,
    modifier = Modifier.fillMaxWidth()
)
}

```

This will display the name, release date, and overview of the TV show on the details screen.

10. Create a new activity named `DetailsActivity` and add the following companion object:

```

class DetailsActivity : ComponentActivity() {

    companion object {
        const val EXTRA_TITLE = "title"
        const val EXTRA_RELEASE = "release"
        const val EXTRA_OVERVIEW = "overview"
        const val IMAGE_URL =
            "https://image.tmdb.org/t/p/w500"
        const val EXTRA_POSTER = "poster"
    }
}

```

These constants will be used to pass the details from the main screen to the details screen.

11. In the `onCreate` function, replace the `setContent` function with the following:

```

setContent {
    TVGuideTheme {
        val extras = intent.extras
        val posterPath = extras
            ?.getString(EXTRA_POSTER)
            .orEmpty()

        DetailsScreen(
            title = extras
                ?.getString(EXTRA_TITLE)
                .orEmpty(),
            release = extras
                ?.getString(EXTRA_RELEASE)
                .orEmpty().take(4),

```

```

        overview = extras
            ?.getString(EXTRA_OVERVIEW)
            .orEmpty(),
        image = "$IMAGE_URL$posterPath",
        onBackButtonClick = { finish() }
    )
}
}

```

This will display the details screen with the poster, name, release, and overview of the TV show selected.

12. Create TVShowService in the `com.example.tvguide.network` package:

```

class TVShowService(private val apiKey: String) {
    private val baseUrl =
        "https://api.themoviedb.org/3/"

    private val httpClient = HttpClient(Android) {
        install(ContentNegotiation) {
            json(
                Json {
                    ignoreUnknownKeys = true
                }
            )
        }
    }

    suspend fun getTVShows(): List<TVShow> {
        val response =
            getTVResponse("${baseUrl}tv/on_the_air")
        return response.results
    }

    private suspend fun getTVResponse(
        url: String
    ): TVShowResponse {
        return httpClient.get(url) {
            url {

```

```
        parameters.append("api_key", apiKey)
    }
    }.body<TVShowResponse>()
}
}
```

This will define the endpoint you will use to retrieve the TV shows that are on the air.

13. Create a `TVShowRepository` class with a constructor for `tvShowService`:

```
class TVShowRepository(private val tvShowService:
TVShowService) {

    suspend fun getTVShows(): Flow<List<TVShow>> {
        return flow {
            emit(tvShowService.getTVShows())
        }.flowOn(Dispatchers.IO)
    }
}
```

14. Create a `TVShowViewModel` class with a constructor for `tvShowRepository` and dispatcher:

```
class TVShowViewModel(
    private val tvShowRepository: TVShowRepository,
    private val dispatcher: CoroutineDispatcher =
        Dispatchers.IO
) : ViewModel() {

    fun getTVShows() {
        viewModelScope.launch(dispatcher) {
            tvShowRepository.getTVShows()
                .catch {
                    _error.value =
                        "An error occurred:
                        ${it.message}"
                }
                .collect {
                    _tvShows.value = it
                }
        }
    }
}
```



```

    }

    companion object {
        val Factory: ViewModelProvider.Factory =
            viewModelFactory
    }
    initializer {
        val application =
            (this[APPLICATION_KEY]
             as TVShowApplication)
        TVShowViewModel(tvShowRepository =
            application.tvShowRepository
        )
    }
}
}
}

```

15. Add a `tvShows` state flow for the list of TV shows and an error state flow for the error message:

```

private val _tvShows =
    MutableStateFlow(emptyList<TVShow>())
val tvShows = _tvShows.asStateFlow()

private val _error = MutableStateFlow("")
val error = _error.asStateFlow()

```

16. Add a `getTVShows` function with a coroutine using `viewModelScope` to collect the TV shows from `tvShowRepository`:

```

fun getTVShows() {
    viewModelScope.launch(dispatcher) {
        tvShowRepository.getTVShows()
            .catch {
                _error.value =
                    "An error occurred: ${it.message}"
            }
        .collect {

```

```
        _tvShows.value = it
    }
}
}
```

17. Create an application class named `TVShowApplication` with a property for `tvShowRepository` and add the API key you got from the `The Movie Database API`:

```
class TVShowApplication : Application() {
    private val apiKey = "your_api_key_here"

    private val tvShowService: TVShowService by lazy {
        TVShowService(apiKey = apiKey)
    }

    lateinit var tvShowRepository: TVShowRepository

    override fun onCreate() {
        super.onCreate()

        tvShowRepository = TVShowRepository(
            tvShowService = tvShowService
        )
    }
}
```

This will be the application class for the app. It will hold a reference to `tvShowRepository`.

18. Set `TVShowApplication` as the value for the `android:name` attribute of the application in the `AndroidManifest.xml` file:

```
<application
    ...
    android:name=".TVShowApplication"
    ... >
    ...
</application>
```

19. Create a new file called `MainScreen` and add a `MainScreen` composable function:

```
@Composable
fun MainScreen(
    viewModel: TVShowViewModel = viewModel(
        factory = TVShowViewModel.Factory
    ),
    onSelectTVShow: (TVShow) -> Unit
) {
    ...
}
```

This will be the main screen for the app.

20. Inside the `MainScreen` composable function, add the following:

```
LaunchedEffect(key1 = Unit) {
    viewModel.getTVShows()
}
Scaffold(
    modifier = Modifier.fillMaxSize(),
    topBar = {
        TopAppBar(
            title = {
                Text(
                    text = stringResource(
                        id = R.string.app_name
                    ),
                    color = MaterialTheme
                        .colorScheme
                        .onPrimary
                )
            },
            colors = topAppBarColors(
                MaterialTheme.colorScheme.primary
            ),
        )
    }
}
```

```
    ) { innerPadding ->
        //TODO
    }
```

21. Inside the content block, replace the TODO comment with the following code:

```
Box(
    modifier = Modifier
        .padding(innerPadding)
        .fillMaxSize()
) {
    val tvShows = viewModel.tvShows.collectAsState()
    LazyVerticalGrid(
        columns = GridCells.Adaptive(160.dp)
    ) {
        items(tvShows.value) { tvShow ->
            TVShowItemView(tvShow = tvShow) {
                onSelectTVShow(tvShow)
            }
        }
    }

    val error = viewModel.error.collectAsState()
    if (error.value.isNotEmpty()) {
        Text(
            text = error.value,
            textAlign = TextAlign.Center,
            modifier = Modifier.align(
                Alignment.Center
            )
        )
    }
}
```

This will display a grid of TV shows or error message text on the screen.

22. Open `MainActivity` and add the `openTVShowDetails` function to open the details screen after selecting a TV show from the list:

```
private fun openTVShowDetails(tvShow: TVShow) {
    val intent = Intent(
        this,
        DetailsActivity::class.java
    ).apply {
        putExtra(
            DetailsActivity.EXTRA_TITLE,
            tvShow.name
        )
        putExtra(
            DetailsActivity.EXTRA_RELEASE,
            tvShow.firstAirDate
        )
        putExtra(
            DetailsActivity.EXTRA_OVERVIEW,
            tvShow.overview
        )
        putExtra(
            DetailsActivity.EXTRA_POSTER,
            tvShow.posterPath
        )
    }
    startActivity(intent)
}
```

23. In the `onCreate` function, replace the `setContent` code with the following:

```
setContent {
    TVGuideTheme {
        MainScreen { tvShow ->
            openTVShowDetails(tvShow)
        }
    }
}
```

This will set the `MainScreen` composable function as the content of `MainActivity`.

24. Run your application. The app will display a list of TV shows. Click on a TV show. You will see its details, such as the release year and an overview of the TV show:

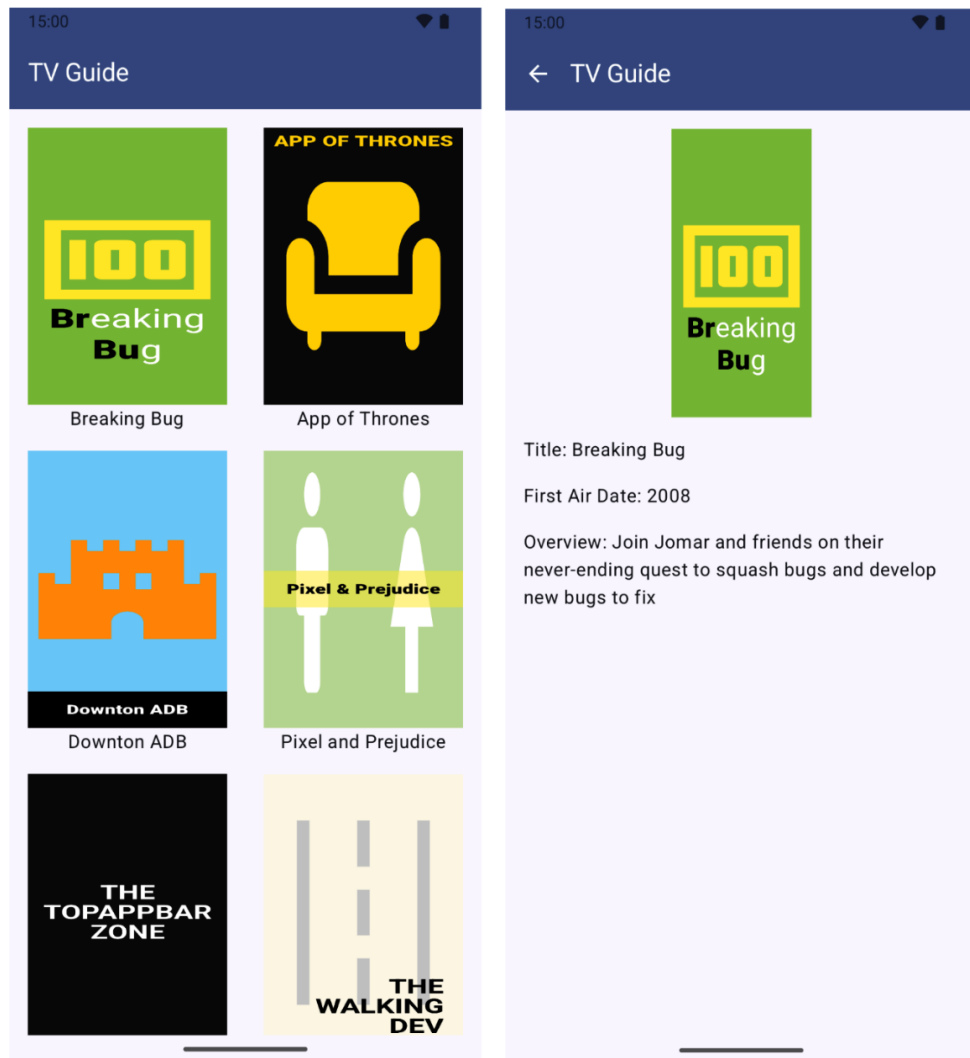


Figure 10.1 – The TV guide app's main screen and details screen