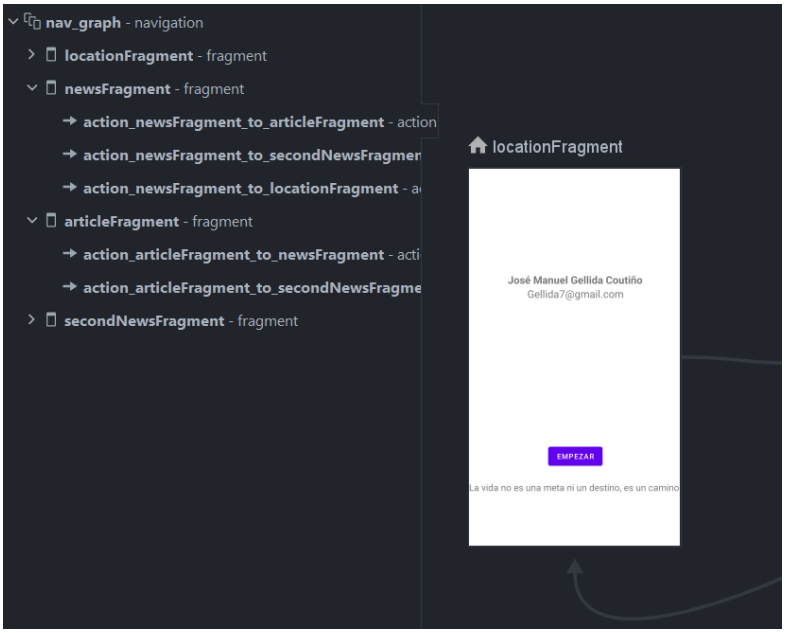
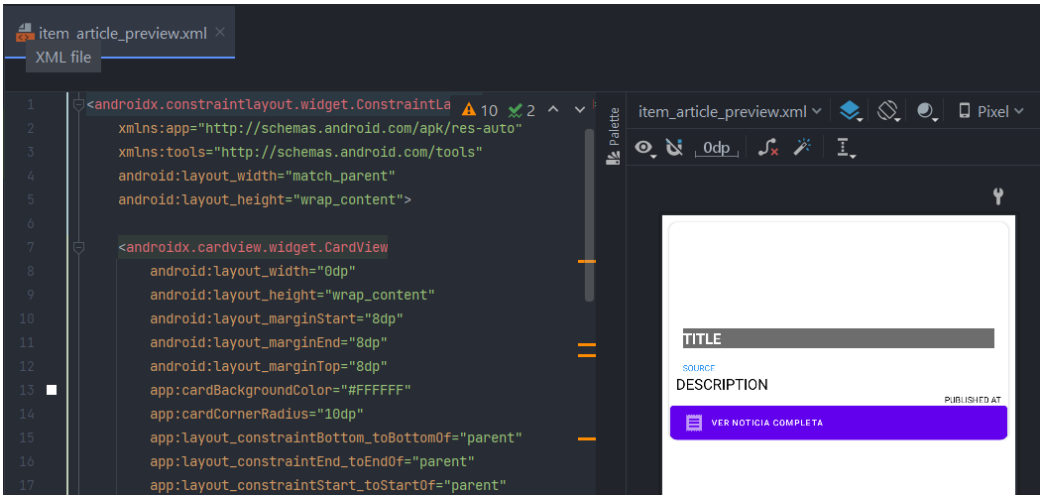
Documentación

El primer paso a implementar fue crear el resource de nav-graph junto la actividad que contendría todos los demás fragments.



El siguiente paso que tomé fue crear el item que contendría el cardview para cada uno de los artículos.



A continuación dispuse de crear los permisos necesarios para la localización y acceso a internet.

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

Empecé a realizar el primer pequeño reto de la prueba, lograr el permiso para acceder a la localización del usuario.

```
private fun isLocationPermissionGranted() = ContextCompat.checkSelfPermission(
    requireContext(),
    android.Manifest.permission.ACCESS_COARSE_LOCATION
) == PackageManager.PERMISSION_GRANTED

private fun requestLocationPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(
            requireActivity(),
            android.Manifest.permission.ACCESS_COARSE_LOCATION
        )
    ) {
        Toast.makeText(requireContext(), text: "Tendrás que aceptar los permisos", Toast.LENGTH_SHORT)
            .show()
    } else {
        ActivityCompat.requestPermissions(
            requireActivity(),
            arrayOf(android.Manifest.permission.ACCESS_COARSE_LOCATION),
            requestCode: 0
        )
    }
```

Primero, habría que usar retrofit para traer la información desde la API.

```
private val retrofit = Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

Después de realizar la primera tarea me dispuse a leer la documentación de la API de noticias dada, para después crear un HTTP Request con una API_KEY creada exclusivamente de la página oficial para las peticiones de la aplicación.

```
@GET("v2/top-headlines")
fun getNewsList(
    @Query("country")
    countryCode: String = "mx",
    @Query("page")
    pageNumber: Int = 1,
    @Query("apiKey")
    apiKey: String = API_KEY

): Call<NewsApiResponse>
```

Opté por no usar JsonToKotlinClass de manera automatizada, para entender mejor la transferencia de información entre la página web y mi app. Creo el response para cada la lista de las noticias.

```kotlin
data class NewsApiResponse(
    @Expose @SerializedName("articles") val articles: List<Article>,
    @Expose @SerializedName("status") val status: String,
    @Expose @SerializedName("total_results") val totalResults: Int,
)

data class Article(
    @Expose @SerializedName("author") val author: String,
    @Expose @SerializedName("content") val content: String,
    @Expose @SerializedName("description") val description: String,
    @Expose @SerializedName("publishedAt") val publishedAt: String,
    @Expose @SerializedName("source") val source: Source,
    @Expose @SerializedName("title") val title: String,
    @Expose @SerializedName("url") val url: String,
    @Expose @SerializedName("urlToImage") val urlToImage: String,
)

data class Source(
    @Expose @SerializedName("id") val id: Any,
    @Expose @SerializedName("name") val name: Any
)
```

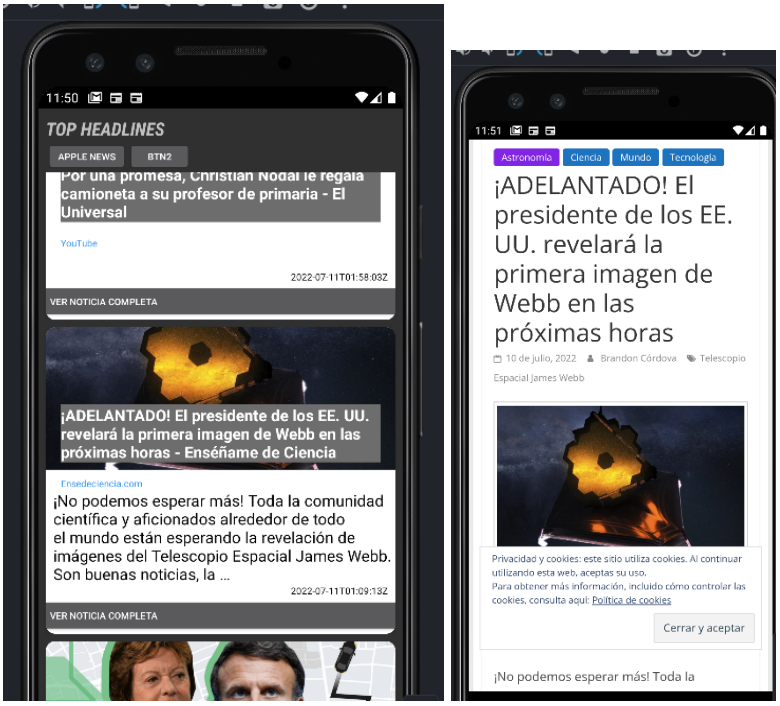
Creo el recyclerview para obtener el adaptador de las noticias.

Finalmente, creo el viemodel de la aplicación

En este momento tengo todo lo que necesito para desplegar la información de las noticias, así que continuo a crearlo en el fragmento de noticias (News Fragment). También en este punto utilizo binding para acelerar el trabajo. Se pasa a la siguiente prueba.

Por último queda crear una webview para obtener la url de la noticia y desplegarla dentro de la aplicación, todo esto dentro del siguiente fragment (ArticleFragment)

En este momento la información se despliega como debe de ser y se da por terminado el último reto.



Lo único que queda es repetir el proceso para crear más secciones de noticias similares.

Por último entra en vigor la etapa de arreglar errores pequeños, mejorar el funcionamiento de la aplicación y mejorar el diseño (opté por crear un diseño minimalista y sencillo) aunque siendo sincero me hubiera gustado detallar más los aspectos gráficos ya que tomé la prueba como si se tratase de solo para verificar la funcionalidad más que la de un diseño UX/UI bien estructurado. Me faltó implementar Dark/Light Mode de mejor manera.

Se utilizó MVVM dentro de todo el proyecto.