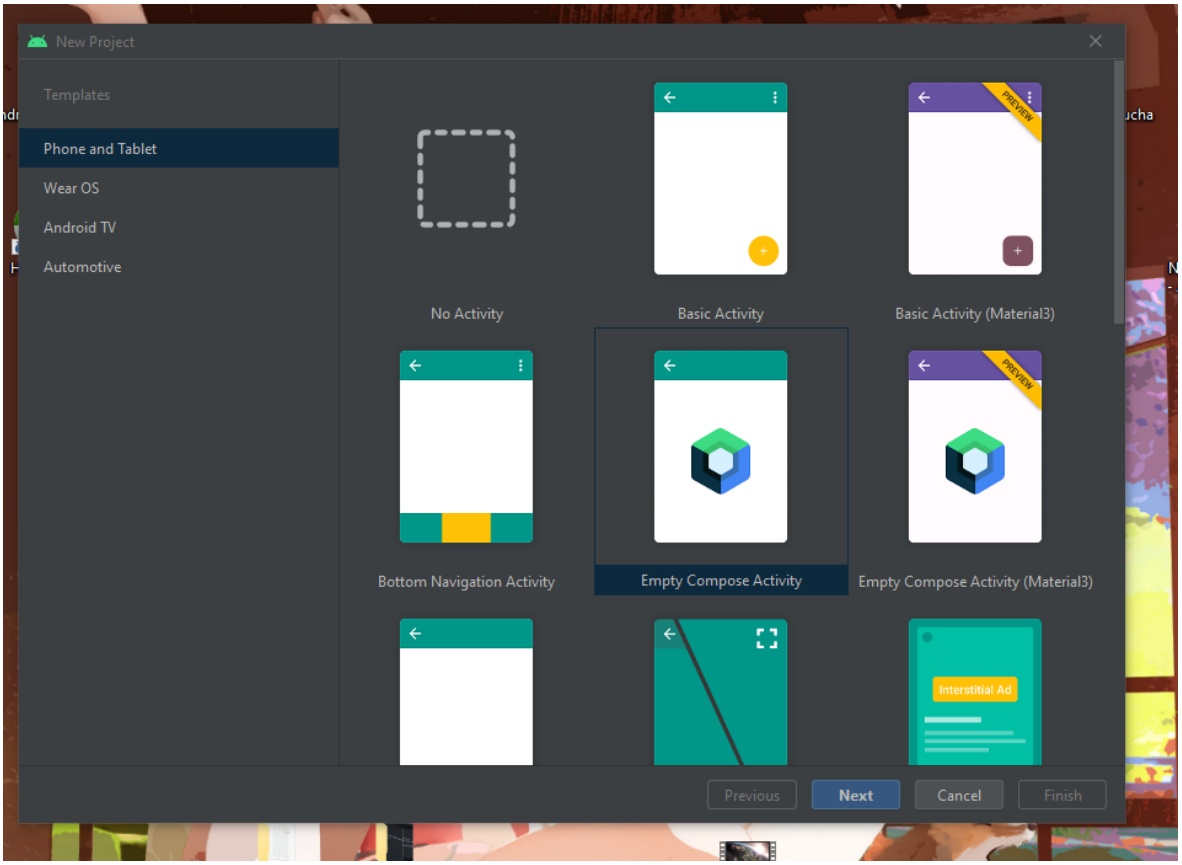
SS_HLC
HORAS DE LIBRE CONFIGURACIÓN

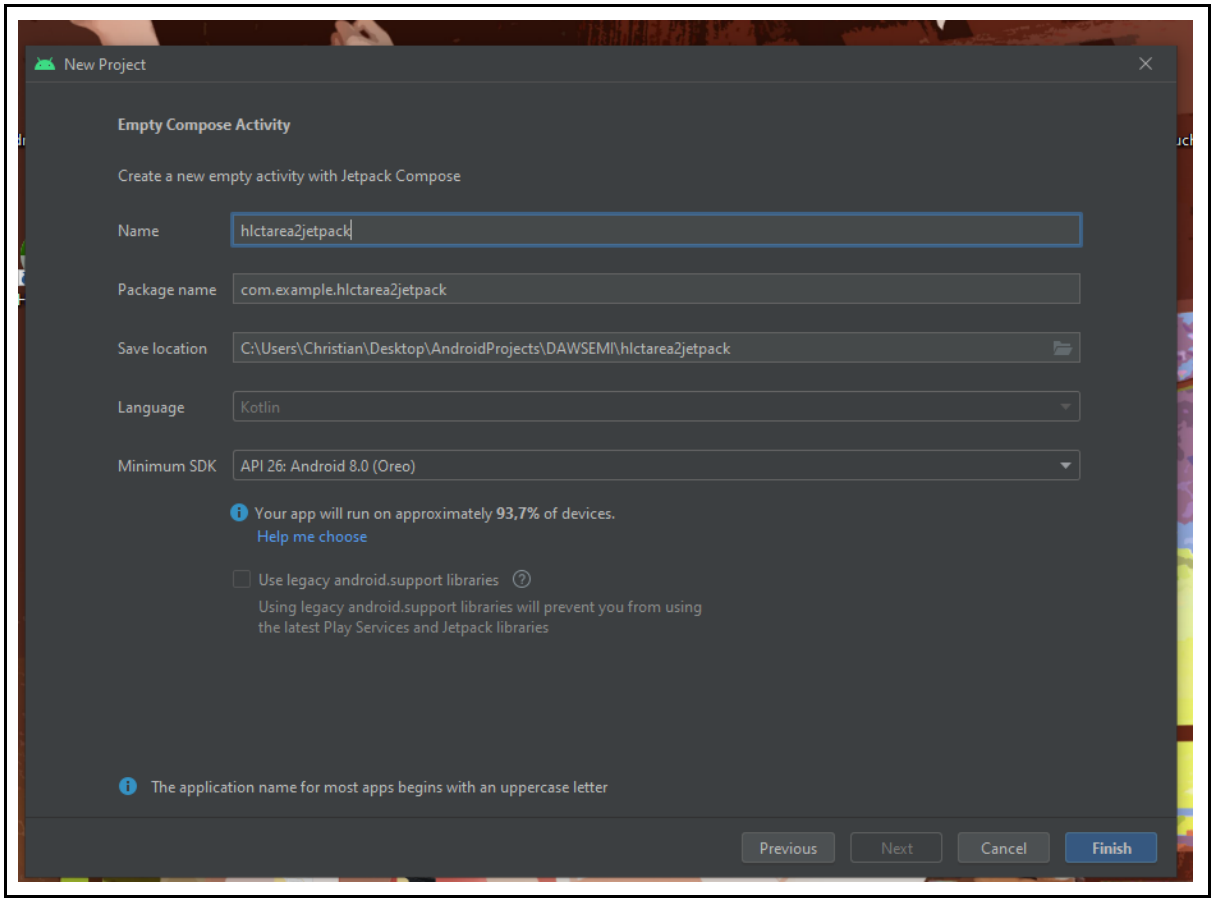
Tema 2

Tarea Online

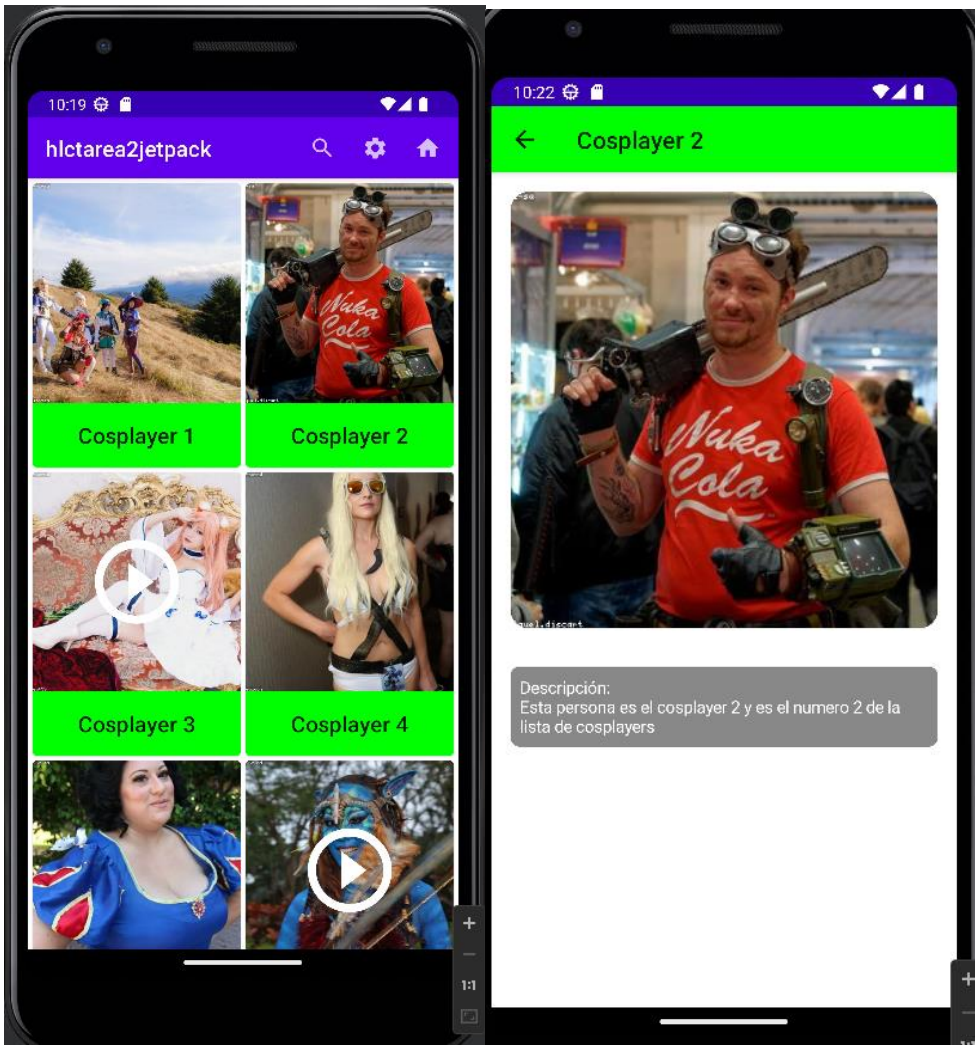
by: Christian Martín Infantes

Enlace a github: https://github.com/NeoChrisMIN/hlctarea2jetpack.git

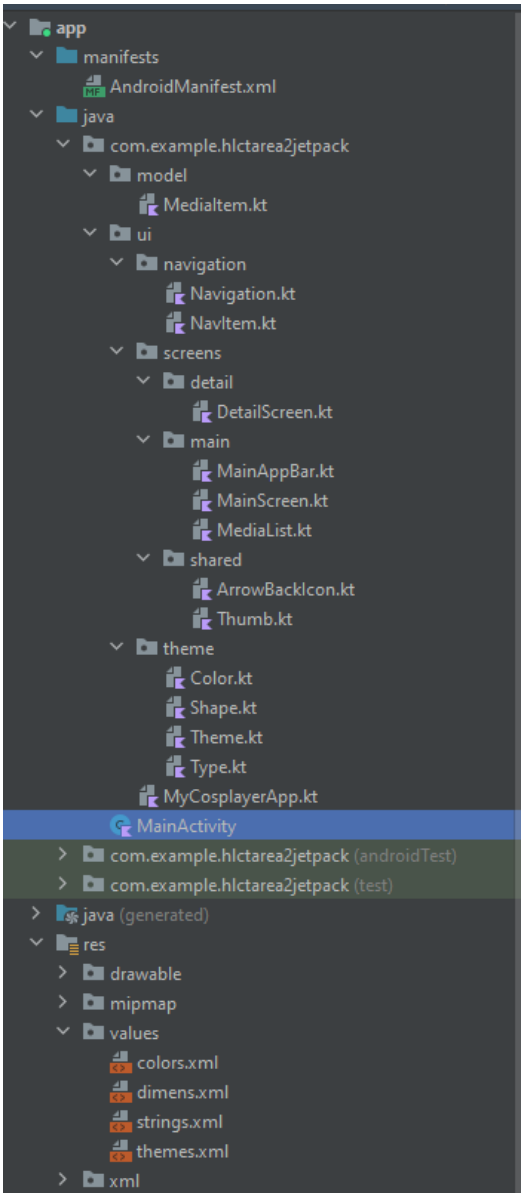
Creamos proyecto



New Project
Empty Compose Activity
Create a new empty activity with Jetpack Compose
Name
hlctarea2jetpack
Package name
com.example.hlctarea2jetpack
Save location
C:\Users\Christian\Desktop\AndroidProjects\DAWSEMI\hlctarea2jetpack
Language
Kotlin
Minimum SDK
API 26: Android 8.0 (Oreo)
Your app will run on approximately 93,7% of devices.
Help me choose
Use legacy android.support libraries
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries
The application name for most apps begins with an uppercase letter
Previous
Next
Cancel
Finish

Pantallas: Lista y Detalle



Estructura de archivos



Archivo: DetailScreen.kt

```
@Composable
fun DetailScreen(mediaId: Int, onUpClick: () -> Unit) {
    val mediaItem = remember { getMedia().first { it.id == mediaId } }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = mediaItem.title) },
                navigationIcon = { ArrowBackIcon(onUpClick) },
                backgroundColor = Color.Green
            )
        }
    ) {
        //Thumb(mediaItem = mediaItem)
        Column(
            modifier = Modifier.fillMaxSize(),
            content = {
                Thumb(mediaItem = mediaItem,
                    modifier = Modifier
                        .size(400.dp)
                        .padding(16.dp)
                        .clip(RoundedCornerShape(16.dp))
                )
                Text(
                    text = "Descripción: ${mediaItem.description}",
                    modifier = Modifier
                        .padding(16.dp)
                        .background(
                            color = Color.Gray,
                            shape = RoundedCornerShape(8.dp)
                        )
                        .padding(8.dp)
                        ,
                    color = Color.White
                )
            }
        )
    }
}
```

Aquí está definido el cómo se verá la pantalla detalle

Archivo: MainAppBar.kt

```
@Composable
fun MainAppBar() {
    TopAppBar(
        title = { Text(stringResource(id = R.string.app_name)) },
        actions = {
            AppBarAction(
                imageVector = Icons.Default.Search,
                onClick = { /* TODO */ }
            )
            AppBarAction(
                imageVector = Icons.Default.Settings,
                onClick = { /* TODO */ }
            )
            AppBarAction(
                imageVector = Icons.Default.Home,
                onClick = { /* TODO */ }
            )
        }
    )
}

@Composable
private fun AppBarAction(
    imageVector: ImageVector,
    onClick: () -> Unit
) {
    IconButton(onClick = onClick) {
        Icon(
            imageVector = imageVector,
            contentDescription = null
        )
    }
}

@Preview
@Composable
fun AppBarActionPreview() {
    MyCosplayApp {
        MainAppBar()
    }
}
```

Aqui esta la barra del menú superior (la top bar) además se declara 3 appbarAction que son 3 logos con función de botón

Archivo: MediaList.kt

```
@ExperimentalFoundationApi
@Composable
fun MediaList(
    onClick: (MediaItem) -> Unit,
    modifier: Modifier = Modifier
) {
    LazyVerticalGrid(
        cells = GridCells.Adaptive(dimensionResource(R.dimen.cell_min_width)),
        contentPadding = PaddingValues(dimensionResource(R.dimen.padding_xsmall)),
        modifier = modifier
    ) {
        items(getMedia()) {
            MediaListItem(
                mediaItem = it,
                onClick = { onClick(it) },
                modifier = Modifier.padding(dimensionResource(R.dimen.padding_xsmall))
            )
        }
    }
}

@Composable
fun MediaListItem(
    mediaItem: MediaItem,
    onClick: () -> Unit,
    modifier: Modifier = Modifier
) {
    Card(
        modifier = modifier.clickable { onClick() }
    ) {
        Column {
            Thumb(mediaItem)
            Title(mediaItem)
        }
    }
}

@Composable
private fun Title(mediaItem: MediaItem) {
    Box(
        contentAlignment = Alignment.Center,
        modifier = Modifier
            .fillMaxWidth()
            //.background(MaterialTheme.colors.secondary)
            .background(Color.Green)
            .padding(dimensionResource(R.dimen.padding_medium))
    ) {
        Text(
            text = mediaItem.title,
            style = MaterialTheme.typography.h6
        )
```

```
    }
}

@Preview
@Composable
fun MediaListItemPreview() {
    MyCosplayApp {
        val mediaItem = MediaItem(1, "Item 1", "", MediaItem.Type.VIDEO, "")
        MediaListItem(mediaItem = mediaItem, onClick = {})
    }
}
```

Aquí se encuentra la itemlist y el diseño de la misma

Archivo: MediaItem.kt

```
data class MediaItem(
    val id: Int,
    val title: String,
    val thumb: String,
    val type: Type,
    val description: String
) {
    enum class Type { PHOTO, VIDEO }
}

fun getMedia() = (1..10).map {
    MediaItem(
        id = it,
        title = "Cosplayer $it",
        thumb = "https://loremflickr.com/400/400/cosplay?lock=$it",
        type = if (it % 3 == 0) Type.VIDEO else Type.PHOTO,
        description = "\nEsta persona es el cosplayer $it y es el numero $it de la lista de
cosplayers"
    )
}
```

Esta es la clase de cada ítem, en ella se guarda cada atributo, además con la función getMedia cargamos las imágenes usando la biblioteca “io.coil-kt:coil-compose”, ademas de darle valor al resto de atributos.