ESCUELA SUPERIOR DE INGENIERÍA

Programación en Internet

GII

FitnessZone App

Autores:

Carlos Antonio Cortés Lora

Roberto Rivero Díaz

Cádiz, 14 de mayo de 2024

Índice General

1. Introducción a la aplicación

La aplicación que hemos desarrollado en Android Studio mediante el lenguaje Kotlin, se denomina FitnessZone. Esta aplicación consiste en una plataforma para la reserva de diferentes salas fitness.

La aplicación presenta un menú accesible en todo momento, que nos permitirá navegar entre las diferentes pantallas de la misma.

1. Home: Pantalla principal de la aplicación, con noticias de FitnessZone

2. Salas: Pantalla para exponer todos los tipos de salas que se ofrecen

3. Reserva: Pantalla con dos funcionalidades:
    a. Podrás realizar una reserva de un tipo de sala, seleccionado fecha, hora y facilitando ciertos datos.
    b. Aparecerá un listado de reservas, donde si seleccionas una en concreto, nos dará la posibilidad de modificar o cancelarla

4. Localización: Pantalla que muestra la ubicación de FitnessZone con un mapa dinámico y las diferentes opciones para llegar

2. Menú

Para la realización del menú hemos optado por utilizar Navegation Drawler que viene por defecto cuando creas un proyecto en Android Studio.

Al crearlo, aparecen home, slideshow y gallery, que han sido refactor y renombrados por salas, reservas y localizacion, modificando cada uno de los archivos y sus asociados, además del mainActivity.

3. Splash

Para la implementación de la actividad de inicio de la App, vamos a crear un Activity nuevo, llamando SplashActivity, donde a su vez no creará su layout propio.

Podemos empezar modificando el layout, en nuestro caso hemos puesto nuestro logo, algunos textos y un ImagenView con id=”imageView ”que mostrará una animación.

Para la creación de esta animación, debemos crear un sprite.xml en drawable, donde definiremos una lista de animación, basada en varias fotos, almacenas también en drawable.

A continuación, debemos añadir al SplashActivity este código:

```
//Buscar el ImageView en el layout por su ID.
val imageView = findViewById<ImageView>(R.id.imageView)
// Intentar obtener el fondo del ImageView como un AnimationDrawable.
// Si el fondo es realmente un AnimationDrawable, iniciar la animación.
(imageView.background as? AnimationDrawable)?.start()
```

Después debemos de añadir una corutina para que se muestre antes del MainActivty:

```
// Coroutine para manejar el retardo antes de iniciar MainActivity
GlobalScope.launch {
   delay(5000) // Retraso de 5 segundos
```

```
  //Después del retraso de 5 segundos, se inicia la actividad MainActivity desde
  // la actividad actual (SplashActivity) utilizando un Intent.
  startActivity(Intent(this@SplashActivity, MainActivity::class.java))
  finish() // Cierra esta actividad para que el usuario no pueda volver a ella
}
```

Finalmente, debemos acceder al manifest, donde en la actividad Splash, debemos añadirle:

```
android:exported="true"
<intent-filter>
  <action android:name="android.intent.action.MAIN" />

  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Recuerda poner dentro del manifest en la actividad Main, poner exported a false.

Quedando MainActivity así:

```
<activity
   android:name=".MainActivity"
   android:exported="false"
   android:label="@string/app_name"
   android:theme="@style/Theme.FitnessZone.NoActionBar">
</activity>
```

Y la SplashAcitivty:

```
<activity
  android:name=".SplashActivity"
  android:exported="true"
  android:label="@string/app_name"
  android:theme="@style/Theme.FitnessZone.NoActionBar">
  <intent-filter>
     <action android:name="android.intent.action.MAIN" />

     <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

4. Widget

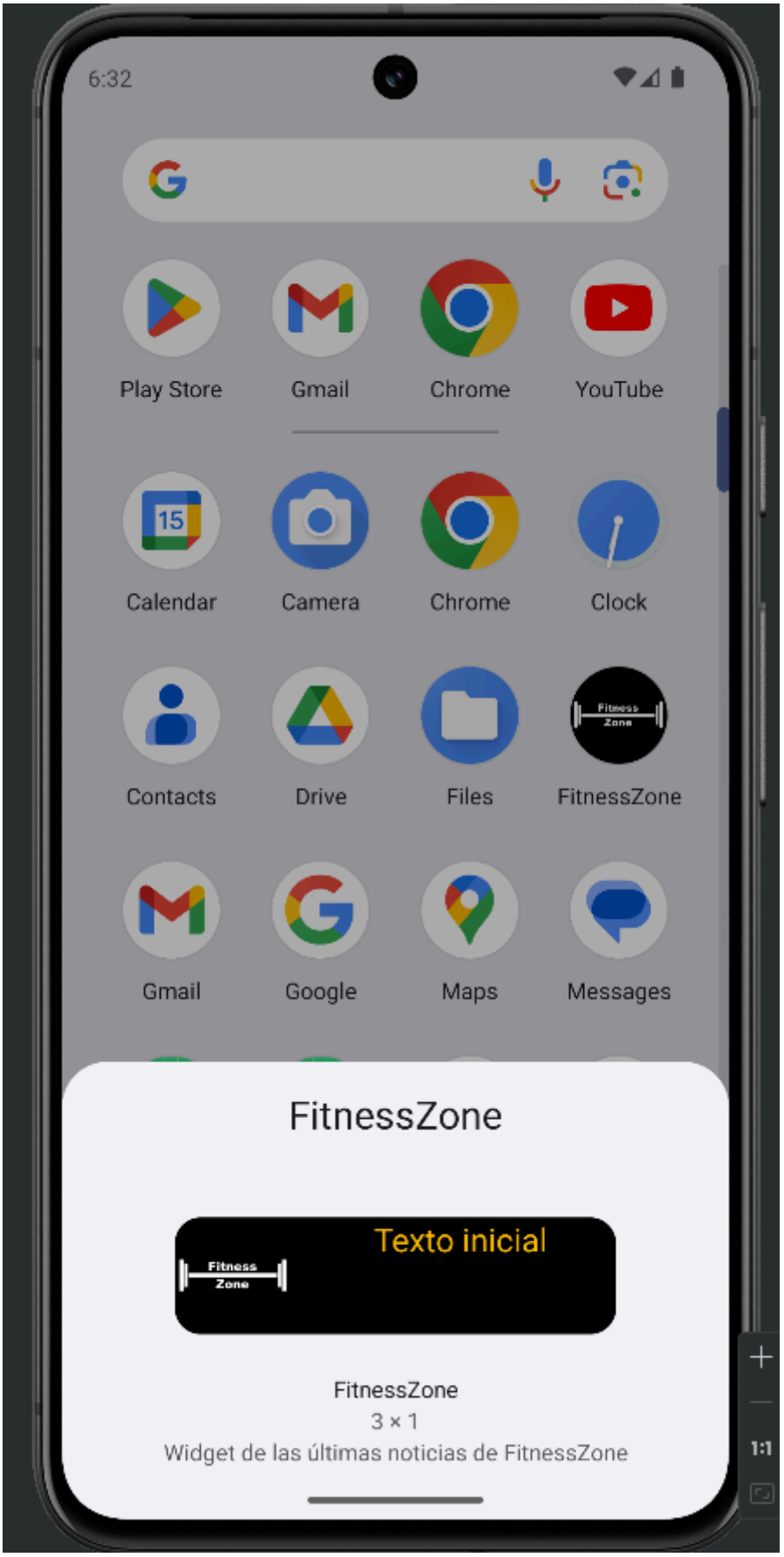
Vamos a crear un layout, llamada “mi_widget_dinamico”, donde vamos a modelar gráficamente nuestro Widget.

A destacar un TextView con id =”texto_widget”, que será el elemento que irá cambiando con el paso del tiempo.

Además en xml debemos crear “mi_widget_dinamico_info.xml”, para establecer los metadatos, donde podemos destacar la descripcion del widget que aparece en la foto de abajo o dos propiedades sobre el tamaño del widget por defecto al crearlo en el movil

```
android:targetCellWidth="3"
```

```
android:targetCellHeight="1"
```


6:32
Play Store
Gmail
Chrome
YouTube
Calendar
Camera
Chrome
Clock
Contacts
Drive
Files
FitnessZone
Gmail
Google
Maps
Messages
FitnessZone
Texto inicial
FitnessZone
3 × 1
Widget de las últimas noticias de FitnessZone

En cuanto a MiwidgetDinamico.kt, debemos destacar el hecho de que estamos creando un widget dinámico que muestra noticias de la app y estás se van cambiando cada 15 minutos de manera aleatoria.

Por tanto, debemos crear una función que genere el texto de manera aleatoria y donde se almace las noticias a mostrar en el widget.

```
fun generarTextoAleatorio(): String {

    val opciones = arrayOf("¡Semana de descuentos!", "¡Bienvenido!", "¡Muy pronto: nuevas salas!", "¡Lanzamiento exclusivo de TRX para principiantes!")

    return opciones[Random.nextInt(opciones.size)]

  }

}
```

Además en el MainActivity, debemos añadir este código para que se vaya actualizando cada 15 min:

```
//Actualiza el texto cada del widget cada minuto

val updateRequest = PeriodicWorkRequest.Builder(WorkerActualizarWidget::class.java, 15, TimeUnit.MINUTES)

  .setInitialDelay(15, TimeUnit.MINUTES)  // El intervalo de tiempo mínimo para WorkManager es de 15 minutos.

  .build()
```

Aparte debemos de crear la funcion que actualize el widget, que de manera adcional queremos que al hacer click en el widget nos lleve a las reserva, por tanto a través del intent, enviamos una variable fragmento.

```
fun actualizaWidget(contexto: Context, administradorAppWidget: AppWidgetManager, idAppWidget: Int){

  // Crea un intento para abrir MainActivity cuando se haga clic en el widget.

  val intento = Intent(contexto, MainActivity::class.java)
```

```
    intento.putExtra("fragmento", "ReservaFragment")

    // Crea un PendingIntent que envuelve nuestro intento, necesario para que el clic funcione.
    val intencionPendiente = PendingIntent.getActivity(contexto, 0, intento, PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE)


    // Crea un objeto RemoteViews para poder manipular la interfaz del widget.
    val vistas = RemoteViews(contexto.packageName, R.layout.mi_widget_dinamico)
    // Asocia el PendingIntent al widget de manera que se active cuando se haga clic en el elemento con ID texto_widget.
    vistas.setOnClickPendingIntent(R.id.texto_widget, intencionPendiente)


    // Genera un texto aleatorio para el widget.
    val textoAleatorio = Utilidades.generarTextoAleatorio()
    // Establece el texto en el widget.
    vistas.setTextViewText(R.id.texto_widget, textoAleatorio)
    // Actualiza el widget con la nueva configuración de vistas.
    administradorAppWidget.updateAppWidget(idAppWidget, vistas)
}
```

En el mainActivity, recogemos ese “fragemento” del SplashActivty y que nos servirá para que nos direccione a las reservas, para ellos debemos añadir el siguiente código:

```
// Verifica si se ha pasado un nombre de fragmento como extra en el intento
val fragmento = intent.getStringExtra("fragmento")
if (fragmento != null && fragmento == "ReservaFragment") {
```

```
    // Abre el fragmento de reserva
    navController.navigate(R.id.nav_reserva)
}
```

Finalmente en el manifest, debemos añadir los siguientes códigos:

```
<intent-filter>
    <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>
<meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/mi_widget_dinamico_info" />
```

Quedándonos de esta manera el Splash Activity del Manifest:

```
<receiver
    android:name=".MiWidgetDinamico"
    android:exported="false">
    <intent-filter>
        <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/mi_widget_dinamico_info" />
</receiver>
```

5. Mapa dinámico y localización

Para la creación del mapa dinámico debemos registrarnos en el google Cloud, crear un proyecto, habilitar el Api de Maps para SDK y conseguir una key que permite su utilización.

A continuación, debemos crear un Activity nuevo, para ello vamos a elegir uno predeterminado de Android Studio , de tipo Activity Map.

Donde le cambiaremos el nombre a “ActivityLocalizacion” a “fragment_localizacion”al que genera automáticamente.

Posteriormente, diseñaremos el layout, añadiendo dos Button que nos llevarán a la web de los medio de transporte, y además insertamos lo siguiente para el mapa:

```
<fragment
  android:id="@+id/map"
  android:name="com.google.android.gms.maps.SupportMapFragment"
  android:layout_width="match_parent"
  android:layout_height="300dp"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintTop_toBottomOf="@+id/text_slideshow"
  map:cameraZoom="13"
  map:mapType="hybrid"
  map:uiZoomControls="true" />
```

Donde nos interesa el id para vincularlo en el LocalizacionFragment, y las opciones propias de la api, marcas con el map:

De manera adicional incluimos esta líne al principio del documento, para establecer el namespace:

```
xmlns:map="http://schemas.android.com/apk/res-auto"
```

Respecto a LocalizacionFragment, nos interesa este trozo de código, donde insertamos la localización del marcador que queramos poner y una etiqueta:

```
private val callback = OnMapReadyCallback { googleMap ->

   val sydney = LatLng(36.537124, -6.201352)


googleMap.addMarker(MarkerOptions().position(sydney).title(
"FitnessZone"))


googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))

}
```

Además, añadiremos los listener de los botones y sus respectivas acciones:

```
buttonBus.setOnClickListener {

  abrirWeb("https://siu.cmtbc.es/es/horarios_lineas_tabla.php?linea=23")

}


buttonTren.setOnClickListener {

  abrirWeb("https://www.renfe.com/es/es/cercanias/cercanias-cadiz/horarios")

}
```

Finalmente, debemos añadir en el manifest las siguiente lineas con la key de google, facilitada anteriormente:

```
<meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="AIzaSyBvTQzxP_6bEdU9bq6H4BbEBkxbh8q7ZXk" />
```

Tras todo esto, nos debería de quedar así:


7:28
Localizacion
¿Dónde nos encontramos?
Escuela
FitnessZone
CA-32
105A
104B
CA-35
Google
Nos encontramos en Av. Universidad de Cádiz, 10, 11519 Puerto Real, Cádiz. Somos una empresa alojada en el campus de Puerto Real de la Universidad de Cádiz perteneciente a la misma población, cerca del Parque Natual Los Toruños y del Río San Pedro.
¿Cómo llegar?
1:1

6. Salas

El paquete de salas esta formado por:

- Salas
- SalasAdapter
- SalasFargment
- SalasViewModel

Para la implementación de la misma hemos usado un ReciclerView.

Este ReciclerView tomará de SalasViewModel, donde se encuentra la lista de salas. La clase Salas, presenta todos los datos y los tipos que forman una sala.

SalasAdapter, se encarga de toda la lógica que permite implementar correctamente el ReciclerView.

7. Reservas

Este paquete está formado por:

- Reserva
- ReservaFragment
- ReservaAdapter
- ReservaViewModel
- VerFragment
- ModificarFragment
- APIservicios
- AltaFragment

Para la implementación de la misma hemos usado un ReciclerView.

8. Api

La api se conecta con MongoDB, presenta la siguiente operaciones:

- get para obtener todas las reservas
- get para obtener una reserva
- post para añadir una reserva
- put para modificar una reserva