**Prácticas de Aplicaciones para Dispositivos Móviles**

**ANEXO – AUTOEVALUACIÓN**

Podéis ampliar las casillas todo lo que necesitéis.

|  |  |
| --- | --- |
| Tipo de Layouts Utilizados | Especifica cuáles y cómo:  **Layout\_Width**  **Layout\_Height**  **Layout\_marginStart**  **Layout\_marginTop**  **Layout\_maginEnd**  **Layout\_marginBottom**  **Layout\_constraintBottom\_toBottomOf**: Alinear la parte inferior de un elemento con la parte inferior de otro.  **Layout\_constraintEnd\_toEndOf :** Se utiliza para alinear el extremo (derecho) de un elemento con el extremo de otro elemento  **Layout\_constraintHorizontal\_bias:** Ajusta la posición horizontal entre dos restricciones  **Layout\_constraintStart\_toStartOf:** alinear el inicio (izquierdo) de un elemento con el inicio de otro elemento |
| Tipo de Controles Utilizados: | Especifica cuáles y cómo:  **val flip = findViewById<Button>(R.id.flip\_btn) as Button:**  findViewById<Button>(R.id.flip\_btn) busca un elemento en la UI con el ID flip\_btn y lo convierte en un objeto de tipo Button. El resultado se asigna a la variable flip  **val switch2 = findViewById<Switch>(R.id.switch2):**  findViewById<Switch>(R.id.switch2): Busca un elemento en la UI con el ID switch2 y lo convierte en un objeto de tipo Switch. El resultado se asigna a la variable switch2 |
| Actividades Utilizadas | Especifica cuáles y comunicación entre ellas:  **<activity**  **android:name=".PokemonAdapter"**  **android:exported="false" />**  **<activity**  **android:name=".detalle"**  **android:exported="false" />**  **<activity**  **android:name=".aboutUs"**  **android:exported="false" />**  **<activity**  **android:name=".crud"**  **android:exported="false" />**  **<activity**  **android:name=".userDetalle"**  **android:exported="false" />**  **<activity**  **android:name=".editFormUser"**  **android:exported="false" />**  **<activity**  **android:name=".profile"**  **android:exported="false" />**  **<activity**  **android:name=".pokemonEditDetalle"**  **android:exported="false" />**  **<activity**  **android:name=".editFormPokemon"**  **android:exported="false" />**  **<activity**  **android:name=".register"**  **android:exported="false" />**  **<activity**  **android:name=".homeApp"**  **android:label="@string/title\_activity\_home"**  **android:exported="true"**  **android:theme="@style/Theme.MainActivity"/>**  **<activity**  **android:name=".loginApp"**  **android:exported="true" >** |
| Intenciones (Implícitas) | Especifica cuáles y cómo:  val intent = Intent(context, loginApp::class.java).apply {  flags = Intent.FLAG\_ACTIVITY\_NEW\_TASK or Intent.FLAG\_ACTIVITY\_CLEAR\_TASK    val pendingIntent: PendingIntent = PendingIntent.getActivity(context, 0, intent,  PendingIntent.FLAG\_IMMUTABLE) |
| Intenciones (Explícitas) | Especifica cuáles y cómo:  **val intent = Intent(this, homeApp::class.java)**  **startActivity(intent)**  **val intent = Intent(this, profile::class.java)**  **startActivity(intent)**    **val intent = Intent(this, detalle::class.java)**  **startActivity(intent)**  **val intent = Intent(this, userDetalle::class.java)**  **startActivity(intent)**    **val intent = Intent(this, pokemonEditDetalle::class.java)**  **startActivity(intent)**    **val intent = Intent(this, register::class.java)**  **startActivity(intent)** |
| Fragments: | Especifica dónde y cómo:  **Primary\_details\_nav\_graph**  <fragment  android:id="@+id/fragment\_pokemon\_detail"  android:name="com.example.jorgesergiapp.pokemonDetailFragment"  android:label="@string/title\_pokemon\_detail"  tools:layout="@layout/fragment\_pokemon\_detail">  <argument  android:name="item\_id"  app:argType="string"  android:defaultValue="" />  </fragment>  <fragment  android:id="@+id/pokemon\_list\_fragment"  android:name="com.example.jorgesergiapp.pokemonListFragment"  android:label="pokemonListFragment" >  <action  android:id="@+id/show\_pokemon\_detail"  app:destination="@id/pokemon\_detail\_fragment" />  </fragment>  <fragment  android:id="@+id/pokemon\_detail\_fragment"  android:name="com.example.jorgesergiapp.pokemonDetailFragment"  android:label="pokemon\_detail"  tools:layout="@layout/fragment\_pokemon\_detail">  <argument  android:name="item\_id"  app:argType="string"  android:defaultValue="" />  </fragment>  **Nav\_graph.xml**  <fragment  android:id="@+id/FirstFragment"  android:name="com.example.jorgesergiapp.FirstFragment"  android:label="@string/first\_fragment\_label"  tools:layout="@layout/fragment\_first" >  <action  android:id="@+id/action\_FirstFragment\_to\_SecondFragment"  app:destination="@id/SecondFragment" />  </fragment>  <fragment  android:id="@+id/SecondFragment"  android:name="com.example.jorgesergiapp.SecondFragment"  android:label="@string/second\_fragment\_label"  tools:layout="@layout/fragment\_second" >  <action  android:id="@+id/action\_SecondFragment\_to\_FirstFragment"  app:destination="@id/FirstFragment" />  </fragment> |
| Shared Preferences: | Especifica cómo y para qué:  **LoginApp.kt**  // Guardar el nombre de usuario en SharedPreferences  if(storedUser!=null && id!=null){  prefs.setName(storedUser)  prefs.setidUsuario(id)  prefs.setisAdmin(isAdmin)  }  **Prefs.kt**  class Prefs(val context: Context) {  ……  val SHARED\_NAME = "Mydtb"  val storage = context.getSharedPreferences(SHARED\_NAME,0) |
| Ficheros | Internos, Externos, Raw:  **Google-services.json:** Fichero de configuración de la base de datos. |
| BBDD | Especifica la BBDD, campos, y el uso en tu app:  Para la base de datos hemos utilizado “Firebase” |
| Content Providers | Creados por ti: |
| Content Providers | Del sistema:  **Profile.kt** private val requestPermissionLauncher = registerForActivityResult(  ActivityResultContracts.RequestPermission()  ) { isGranted: Boolean ->  if (isGranted) {  // Permission is granted. Show the notification.  showNotification(this)  } else {  // Permission is denied. Handle accordingly.  }  }  val requestPermissionLauncher = registerForActivityResult(ActivityResultContracts.RequestPermission()) { isGranted: Boolean ->  if (isGranted) {  // Permission is granted. Continue the action or workflow in your app.  } else {  // Explain to the user that the feature is unavailable because the  // features requires a permission that the user has denied. At the  // same time, respect the user's decision. Don't link to system  // settings in an effort to convince the user to change their  // decision.  }  } |
| Uso de Pantalla: | Para qué, donde: |
| Sensores: | ¿Cuáles?, ¿Para qué? ¿Algún uso específico que comentar? |
| Toasts: | ¿Con gravity? ¿Personalizados?   * **Toast.makeText(this, "Correo o contraseña incorrectos", Toast.LENGTH\_SHORT):** este mensaje se muestra en la pantalla inicial cuando el usuario no se registra correctamente * **Toast.makeText(this,"Registroexitoso",Toast.LENGTH\_SHORT).show():** En la pagina register, cuando se realiza el registro correctamente. |
| Notificaciones en la barra de tareas: | Especifica donde: |
| Action Bar: | Donde:  **loginApp.kt** private lateinit var actionBarDrawerToggle: ActionBarDrawerToggle  // Inicializar ActionBarDrawerToggle  actionBarDrawerToggle = ActionBarDrawerToggle(this,  drawerLayout, R.string.nav\_open, R.string.nav\_close)  drawerLayout.addDrawerListener(actionBarDrawerToggle)  actionBarDrawerToggle.syncState()  // Mostrar el ícono de navegación en la barra de aplicaciones  supportActionBar?.setDisplayHomeAsUpEnabled(true)  // Listener para el ícono de navegación en la barra de aplicaciones  actionBarDrawerToggle.setToolbarNavigationClickListener {  Toast.makeText(this, "Icono de navegación clicado",  Toast.LENGTH\_SHORT).show()  } |
| Navigation Drawer: | Donde:  **Profile.kt:**  fotoUsuario.setBackgroundResource(R.drawable.profile\_button)  **loginApp.kt** drawerLayout = findViewById(R.id.myDrawer\_layout)  val navView: NavigationView = findViewById(R.id.nav\_view)  // Inicializar ActionBarDrawerToggle  actionBarDrawerToggle = ActionBarDrawerToggle(this, drawerLayout, R.string.nav\_open, R.string.nav\_close)  drawerLayout.addDrawerListener(actionBarDrawerToggle)  actionBarDrawerToggle.syncState()  // Mostrar el ícono de navegación en la barra de aplicaciones  supportActionBar?.setDisplayHomeAsUpEnabled(true)  // Listener para el ícono de navegación en la barra de aplicaciones  actionBarDrawerToggle.setToolbarNavigationClickListener {  Toast.makeText(this, "Icono de navegación clicado", Toast.LENGTH\_SHORT).show()  } |
| Iconos personalizados: | De la aplicación, del escritorio…  **Botton\_nav\_menu.xml**  <item android:id="@+id/home"  android:icon="@drawable/baseline\_home\_24"  android:title="home" />  <item android:id="@+id/listaTodosPokemons"  android:icon="@drawable/baseline\_format\_list\_bulleted\_24"  android:title="listPokemons" />  <item android:id="@+id/profile"  android:icon="@drawable/profile\_button"  android:title="Profile" />  **bottom\_nav\_menu\_admin:**  <item android:id="@+id/page\_1"  android:icon="@drawable/baseline\_home\_24"  android:title="home" />  <item android:id="@+id/page\_2"  android:icon="@drawable/baseline\_search\_24"  android:title="Search" />  <item android:id="@+id/page\_3"  android:icon="@drawable/baseline\_format\_list\_bulleted\_24"  android:title="Search" />  <item android:id="@+id/page\_4"  android:icon="@drawable/baseline\_supervised\_user\_circle\_24"  android:title="Search" /> |
| Audio: | Modo de empleo:  **homeApp.kt**  // Declaramos la variable que indicara si esta activado el sonido  val sound = findViewById<Button>(R.id.sound) as Button  sound.setOnClickListener {  if (isPlaying) {  // Detener la reproducción  mediaPlayer?.pause()  mediaPlayer?.seekTo(0)  // Opcional: para reiniciar desde el principio  isPlaying = false  sound.text = "Musica"  } else {  // Iniciar la reproducción  if (mediaPlayer == null) {  mediaPlayer = MediaPlayer.create(this, R.raw.musica)  // Ajusta aquí el nombre del archivo  mediaPlayer?.setOnCompletionListener {  // Restablecer cuando la música termina  isPlaying = false  sound.text = "Musica"  }  }  mediaPlayer?.start()  isPlaying = true  sound.text = "Parar"  }  } |
| Video: | Enlace directo, video en raw, enlace de YouTube… etc… |
| Animation Tween | Donde:  **homeApp.kt**  lateinit var front\_animation:AnimatorSet  lateinit var back\_animation: AnimatorSet  front\_animation = AnimatorInflater.loadAnimator(applicationContext,  R.animator.front\_animator) as AnimatorSet  back\_animation = AnimatorInflater.loadAnimator(applicationContext,  R.animator.back\_animator) as AnimatorSet  front\_animation.setTarget(front);  back\_animation.setTarget(back);  front\_animation.start()  back\_animation.start() |
| Sensible a tipos de pantalla | Especificar cuáles: |
| Sensible a rotaciones de pantallas | Especificar qué pantallas: |
| Sensible a cambios de Idiomas: | Especificar qué pantallas e idiomas: |
|  |  |
|  |  |
| **Algo** de búsqueda por Internet **que no sea del temario** |  |

**Robustez** de la aplicación, **calidad** en la documentación, **validación** de las entradas de datos…

SI/NO: ¿Dónde? ¿Cómo?

CALIFICACIÓN QUE ME OTORGO PREVIAMENTE A LA ENTREVISTA: