Base de datos con SQLiteOpenHelper

Crearemos la aplicación **App_04_02**.

Vamos a crear una aplicación para gestionar una base de datos en SQLite de animales de Cantabria.

BASE DE DATOS

Se deberá utilizar una base de datos con la siguiente estructura:

La tabla **generos** contiene información sobre géneros de animales (cérvidos, felinos, bóvidos, etc.) y sus características (contiene una pequeña parte del supuesto texto). La columna id es la clave primaria y es autoincrementado.

La tabla **animales** tiene información sobre animales salvajes que podemos encontrar en Cantabria. La columna id es la clave primaria y es autoincrementada. En url_imagen se tiene la URL de acceso a un archivo de imagen con la foto del animal. En genero se tiene el identificador del género al que pertenece el animal.

VISTAS

En primer lugar vamos a crear un **menú** lateral para acceder a las distintas opciones de la aplicaciones.
Para ello, crea un fichero desde res → New → Android Resource File. Llámalo **menu_lateral** y elige la categoría Menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:id="@+id/bdanimales" android:checkableBehavior="single">
        <item android:id="@+id/menu_lista" android:title="@string/listado_animales"
```

```
        android:icon="@android:drawable/ic_menu_view"/>
    <item android:id="@+id/menu_add" android:title="@string/add_animal"
        android:icon="@android:drawable/ic_menu_add"/>
    <item android:id="@+id/menu_por_genero" android:title="@string/animales_genero"
        android:icon="@android:drawable/ic_menu_more"/>
    <item android:id="@+id/menu_delete" android:title="@string/eliminar_animales"
        android:icon="@android:drawable/ic_menu_delete"/>
  </group>

  <item android:id="@+id/informacion" android:title="@string/informacion"
    android:icon="@android:drawable/ic_menu_info_details"/>
  <item android:id="@+id/ayuda" android:title="@string/ayuda"
    android:icon="@android:drawable/ic_menu_help" />
  <item android:id="@+id/configurar" android:title="@string/configurar"
    android:icon="@android:drawable/ic_menu_manage"/>
</menu>
```

Crearemos un layout para la cabecera del menú lateral. Dentro de la carpeta layout crea un fichero llamado **cabecera_navegacion.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="200dp">
  <ImageView
    android:id="@+id/imageview_cabecera"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:src="@drawable/rebeco"
    android:scaleType="centerCrop" />

  <TextView
    android:id="@+id/texview_cabecera"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/animales_de_cantabria"
    android:textAppearance="@style/TextAppearance.AppCompat.Large.Inverse"
    android:textStyle="bold"
    android:layout_gravity="bottom"
    android:layout_marginBottom="10dp"
    android:layout_marginStart="10dp" />
</FrameLayout>
```

Crearemos un **gráfico de navegación** como en los ejercicios anteriores llamado **grafico_navegacion.xml**

En el **activity_main.xml** crearemos un elemento **DrawerLayout** con un **NavigationView**. Además, habrá un **FrameContainerView** que será el contenedor de Fragments de nuestra aplicación:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawerLayout"
    tools:context="vistas.MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.fragment.app.FragmentContainerView
            android:id="@+id/layoutFragmentHolder"
            android:name="androidx.navigation.fragment.NavHostFragment"
            app:defaultNavHost="true"
            app:navGraph="@navigation/grafico_navegacion"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navView"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        app:headerLayout="@layout/cabecera_navegacion"
        app:menu="@menu/menu_lateral"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"/>

</androidx.drawerlayout.widget.DrawerLayout>
```

Para que el menú funcione tenemos que configurarlo en clase **MainActivity**:

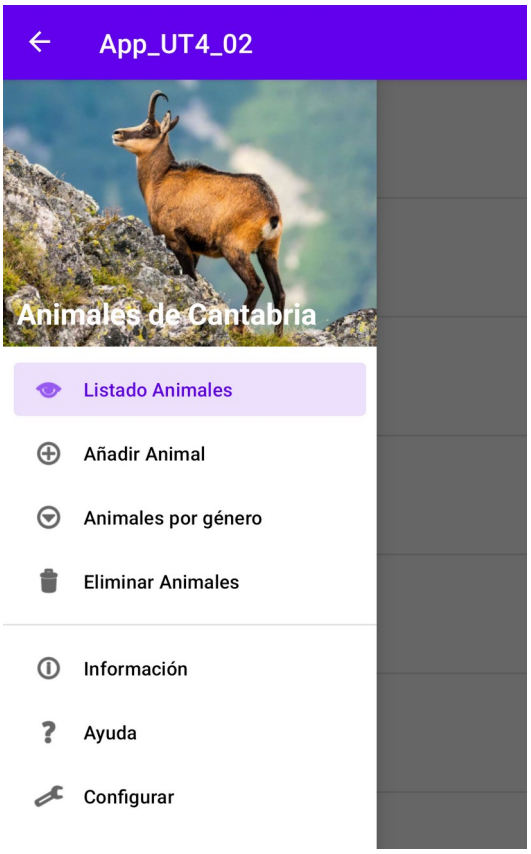
```
class MainActivity : AppCompatActivity()
{
    private lateinit var binding: ActivityMainBinding
    private lateinit var toggle: ActionBarDrawerToggle
```

```
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)


        binding.apply {
            toggle = ActionBarDrawerToggle(this@MainActivity, drawerLayout, R.string.open,
R.string.close)
            drawerLayout.addDrawerListener(toggle)
            toggle.syncState()

            supportActionBar?.setDisplayHomeAsUpEnabled(true)

            navView.setCheckedItem(R.id.menu_lista)
            navView.setNavigationItemSelectedListener {
                when (it.itemId) {
                  R.id.menu_lista -> {
                  }
                  R.id.menu_add -> {
                  }
                  R.id.menu_por_genero -> {
                  }
                  R.id.menu_delete -> {
                  }
                }
                drawerLayout.closeDrawer(GravityCompat.START)
                true
            }
        }
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (toggle.onOptionsItemSelected(item)){
            return true
        }
        return super.onOptionsItemSelected(item)
    }


    override fun onBackPressed() {
        if (binding.drawerLayout.isDrawerOpen(GravityCompat.START)) {
            binding.drawerLayout.closeDrawer(GravityCompat.START)
        } else {
            super.onBackPressed()
        }
    }

}
```

Crea ahora 4 fragmentos vacíos y añádelos al gráfico de navegación:

- FragmentNuevoAnimal
- FragmentListaAnimales
- FragmentAnimalesPorGenero
- FragmentEliminarAnimal

Si ahora ejecutamos la app deberíamos ver un menú similar al de la siguiente imagen:



En la clase MainActivity crear la propiedad navController:

```
private lateinit var navController: NavController
```

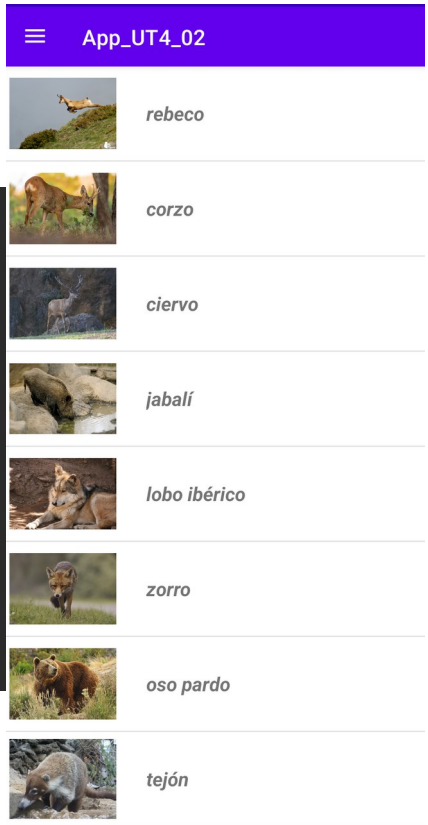
Inicializarla dentro del método onCreate y completa el when del evento setNavigationItemSelectedListener del navView para que al pulsar cada elemento del menú nos lleve al fragment adecuado.

Por ejemplo:

```
navView.setNavigationItemSelectedListener {
   when (it.itemId)
   {
      R.id.menu_add -> navController.navigate(R.id.fragmentNuevoAnimal)
```

FragmentListaAnimales

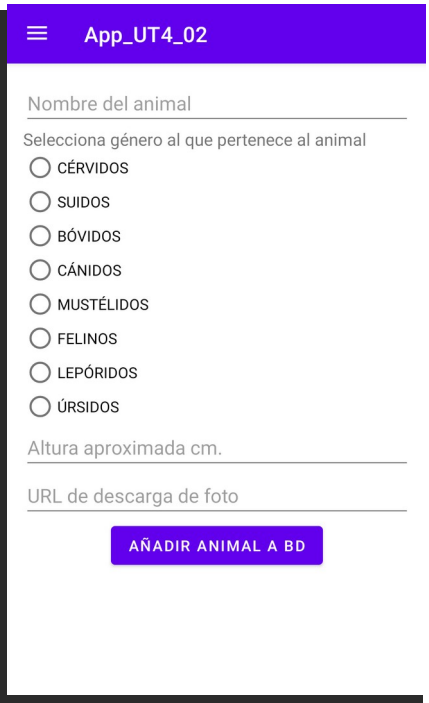
Dentro del mapa de navegación este fragmento será el de inicio. Únicamente tendrá un **RecyclerView** que ocupará todo el layout.




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".vistas.FragmentListaAnimales">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerciew_lista_fragment_lista"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

FragmentNuevoAnimal

Tendrá los campos necesarios para introducir los datos de un nuevo animal



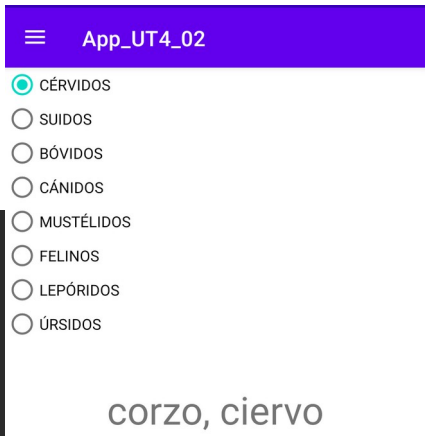

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="15dp"
    tools:context=".vistas.FragmentNuevoAnimal">

    <EditText
        android:id="@+id/edittext_nombre_fragment_nuevo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/nombre_del_animal"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/selecciona_genero"
        android:textSize="16sp"/>
```

```
    <RadioGroup
        android:id="@+id/radiogroup_fragment_nuevo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </RadioGroup>

    <EditText
        android:id="@+id/edittext_altura_fragment_nuevo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/altura_aproximada"
        android:inputType="number"/>

    <EditText
        android:id="@+id/edittext_url_imagen_fragment_nuevo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/url_de_descarga_de_foto"
        android:inputType="textUri"/>

    <Button
        android:id="@+id/button_add_fragment_nuevo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/anadir_animal_bd" />
</LinearLayout>
```

El RadioGroup estará vacío. Lo rellenaremos obteniendo posteriormente los datos de la base de datos.

FragmentAnimalesPorGenero

Tendrá un RadioGroup con los géneros y un texto en el que se mostrarán los nombres de los animales del género seleccionado

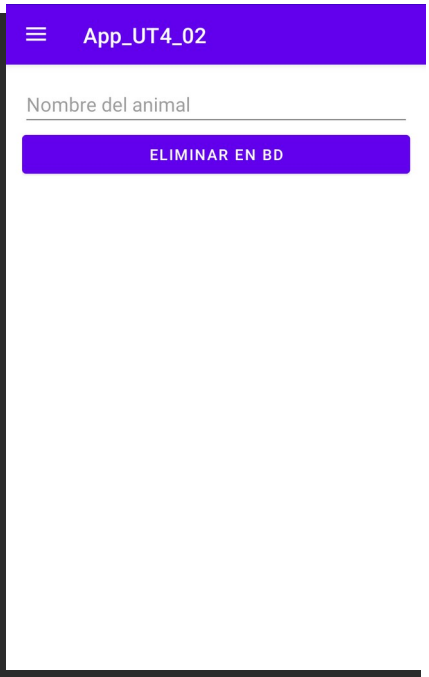



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="vertical"
    tools:context=".vistas.FragmentAnimalesPorGenero">

    <RadioGroup
        android:id="@+id/radiogroup_fragment_por_genero"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:orientation="vertical">
    </RadioGroup>

    <TextView
        android:id="@+id/textview_lista_fragment_por_genero"
        android:layout_marginTop="40dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:textSize="36sp" />

</LinearLayout>
```

FragmentEliminarAnimal

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="15dp"
    tools:context=".vistas.FragmentEliminarAnimal">

    <EditText
        android:id="@+id/editText_nombre_fragment_eliminar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/nombre_del_animal" />

    <Button
        android:id="@+id/button_eliminar_fragment_eliminar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/eliminar_en_bd" />
</LinearLayout>
```



MODELO

Crea las clases Genero y Animal (serán data class). Fijarse que el Animal tendrá un objeto de tipo Genero como atributo.

DATOS.BASEDATOS

Crearemos una clase llamada AnimalDbHelper que herede de SQLiteOpenHelper.

Las consultas de la base de datos serán las siguientes:

```
CREATE TABLE generos (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
nombre TEXT NOT NULL,
caracteristicas TEXT
);
INSERT INTO generos (id,nombre,caracteristicas) VALUES (1,'CÉRVIDOS','Tienen patas delgadas,
pezuñas partidas en dos y largos cuellos ...'),
(2,'SUIDOS','Se trata de ungulados no-rumiantes de alimentación omnívora y características
primitivas.'),
(3,'BÓVIDOS','Tienen una alimentación estrictamente herbívora ...'),
(4,'CÁNIDOS',' Familia de mamíferos carnívoros que engloba abundantes especies...'),
(5,'MUSTÉLIDOS','La mayoría son de dieta carnívora o son carroñeros...'),
(6,'FELINOS','Mamíferos placentarios del orden Carnivora. Poseen un cuerpo esbelto...'),
(7,'LEPÓRIDOS','Familia de mamíferos lagomorfos que engloba ...'),
(8,'ÚRSIDOS','Son animales de gran tamaño, generalmente omnívoros ...');

CREATE TABLE animales (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
nombre TEXT NOT NULL UNIQUE,
genero INTEGER,
altura NUMERIC,
url_imagen TEXT
);
INSERT INTO animales (id,nombre,genero,altura,url_imagen) VALUES
(1,'rebeco',3,0.75,'https://farm3.staticflickr.com/2855/33800710532_7c75617895_c.jpg'),
(2,'corzo',1,0.7,'https://live.staticflickr.com/65535/52450423043_485033fe04_b.jpg'),
(3,'ciervo',1,1.1,'https://live.staticflickr.com/6191/6050201058_ee48cde2ee_b.jpg'),
(4,'jabalí',2,0.78,'https://farm4.staticflickr.com/3559/3354559085_5ab8ef617a_b.jpg'),
(5,'lobo ibérico',4,0.75,'https://farm6.staticflickr.com/5263/5573299174_12e7906d0f_b.jpg'),
(6,'zorro',4,0.4,'https://farm4.staticflickr.com/3022/5836669541_b577b9ee0f_b.jpg'),
(7,'oso pardo',8,1,'https://live.staticflickr.com/2640/3717434902_3a954f4223_c.jpg'),
(8,'tejón',5,0.2,'https://live.staticflickr.com/179/485100494_27f2cb7483_c_d.jpg');
```

También tendremos una clase llamada **AnimalRepository** que recibirá un contexto. Esta clase tendrá los métodos insertarAnimal, getGeneros, getGenerosById, getAnimales, getAnimalesPorGenero y borrarAnimal:

```
class AnimalRepository(val context: Context)
{
    fun insertarAnimal(animal: Animal): Long
    {
        val basedatos: SQLiteDatabase = AnimalDbHelper.getInstance(context).writableDatabase
            // Falta implementar
    }


    fun getGeneros(): List<Genero>
    {
        val basedatos: SQLiteDatabase = AnimalDbHelper.getInstance(context).readableDatabase
        // Falta implementar...
    }

    private fun getGeneroById(idGenero: Int): Genero?
    {
    }

    fun getAnimales(): List<Animal>
    {
    }

    fun getAnimalesPorGenero(nombreGenero: String): List<Animal>
    {
    }

fun borrarAnimal(nombre: String): Int
{
    val basedatos: SQLiteDatabase = AnimalDbHelper.getInstance(context).writableDatabase
    return basedatos.delete("animales","nombre=?",arrayOf(nombre))
}
}
```

Habrá que implementar los métodos para que realicen las operaciones de inserción, borrado y consulta de la base de datos.

VIEWMODEL

Se tendrá una instancia del repositorio y dos liveData. El primero contendrá la lista de animales de la base de datos y el segundo los géneros:

```
class AnimalesViewModel(context: Context): ViewModel()
{
```

```
    private val repositorio: AnimalRepository = AnimalRepository(context)

    val animalesLiveData = MutableLiveData<List<Animal>>()
    val generosLiveData = MutableLiveData<List<Genero>>()

    init
    {
        generosLiveData.postValue(repositorio.getGeneros())
        animalesLiveData.postValue(repositorio.getAnimales())
    }


    fun getAnimalesPorGenero(nombreGenero: String): List<Animal>
    {
        return repositorio.getAnimalesPorGenero(nombreGenero)
    }

    fun insertarAnimal(animal: Animal): Long
    {
        val id = repositorio.insertarAnimal(animal)
        animalesLiveData.postValue(repositorio.getAnimales())
        return id
    }

    fun borrarAnimal(nombre: String): Int
    {
        val numero = repositorio.borrarAnimal(nombre)
        animalesLiveData.postValue(repositorio.getAnimales())
        return numero
    }


    // Define ViewModel factory in a companion object
    companion object {
        val Factory: ViewModelProvider.Factory = viewModelFactory {
            initializer {
                val contexto = (this[APPLICATION_KEY] as Context)
                AnimalesViewModel(
                    context = contexto
                )
            }
        }
    }

}
```

CODIFICACIÓN DE LAS VISTAS

En los fragments debemos crear los viewModel del siguiente modo:

```
private val animalesViewModel: AnimalesViewModel by activityViewModels
{AnimalesViewModel.Factory}
```

Luego, habrá que ir haciendo las acciones oportunas para realizar la funcionalidad de cada uno.

En el Fragment de NuevoAnimal y de AnimalesPorGenero hay que cargar varios radioButtons en el radioGroup dinámicamente. Se puede utilizar el siguiente método que recibe una lista de géneros:

```
private fun cargarRadioButtons(generos: List<Genero>)
{
   for (genero in generos)
   {
      val radioButton = RadioButton(activity)
      val parametros: LinearLayout.LayoutParams = RadioGroup.LayoutParams(
         RadioGroup.LayoutParams.WRAP_CONTENT,
         RadioGroup.LayoutParams.WRAP_CONTENT
      )
      radioButton.layoutParams = parametros
      radioButton.text = genero.nombre
      //añadir rb al radiogroup
      binding.radiogroupFragmentNuevo.addView(radioButton)
   }
}
```