Centro de Formación Profesional
//Afundación
Obra Social ABANCA

CUALIFICACIÓN

Nombre:	Nº:	Fecha:

Aplicación multiplataforma: Country Birds

Desarrollaremos una aplicación que permita a los usuarios consultar una lista de pájaros típicos en diferentes países.

La aplicación final consistirá en una única pantalla donde se lanzará una petición que mostrará los pájaros típicos de una región tal y como han sido almacenados en una base de datos.

Módulo: Acceso a datos
Criterio de cualificación: 3 tareas a efectuar en el repositorio del examen. Cada tarea tiene su puntuación en el enunciado. **Contará un 10% de la nota de evaluación.** **Es necesario un 5 para hacer la media ponderada y aprobar la evaluación.**

1. Creación de los modelos (2,5 puntos)

1.1. Clona en tu ordenador el repositorio “ExamenFebrero” dado de alta en tu *gitlab.afundacionfp.com*

1.2. Dentro del repositorio, inicializa un nuevo proyecto Django llamado `countrybirds` con el comando `django-admin startproject countrybirds`

1.3. Añade ese proyecto a un nuevo *commit* con el mensaje “Proyecto inicial backend”

1.4. Inicializa una app llamada `birds` con el comando `python manage.py startapp birds`

1.5. Crea los modelos necesarios en `models.py`:

1.5.1. Sólo necesitaremos un modelo llamado `Bird`. Contendrá los siguientes campos:

- `nombre`. Será un `CharField` con longitud máxima 100 caracteres
- `codigo_pais`. Será un `CharField` con longitud máxima 2 caracteres

1.6. Añade los cambios anteriores a un nuevo *commit* con el mensaje “Modelos creados”

Criterios de aceptación:

- Dos *commits* subidos al repositorio tal y como han sido planteados en la tarea.

2. Dar de alta datos en la base de datos (2,5 puntos)

2.1. Añade la aplicación `birds` a `INSTALLED_APPS` en `settings.py` del proyecto. Puedes consultar la sección *Activando los modelos* (https://docs.djangoproject.com/es/3.1/intro/tutorial02/) del tutorial *Polls* de Django.

2.2. Añade las siguientes líneas a `admin.py` para permitir la gestión del modelo `Birds` desde el sitio web de administración:

```
from .models import Bird
admin.site.register(Bird)
```

2.3. Efectúa las migraciones a base de datos con los comandos:

```
python manage.py makemigrations
python manage.py migrate
```

2.4. Crea un superusuario con el comando `python manage.py createsuperuser`. (Recuerda lanzarlo desde el terminal *cmd* de Windows, no desde *Mingw64*)

2.4.1. Usuario: admin

2.4.2. E-mail: Tu e-mail del dominio @fpcoruna.afundacion.org

2.4.3. Contraseña: admin

2.5. Lanza el servidor con `python manage.py runserver`

2.6. Accede al sitio web de administración en http://localhost:8000/admin. Loguéate y da de alta los siguientes datos:

nombre	codigo_pais
Garceta Común	ES
Abubilla Común	ES
Gaviota Picofina	ES
Kagu	NC

2.7. Añade todos los cambios en el proyecto a un *commit* con el mensaje “Creado superusuario admin y dados de alta datos de prueba”

Criterios de aceptación:

- Un *commit* subido al repositorio tal y como ha sido planteado en la descripción de la tarea.

3. Creación del endpoint `/backend/1/pajaro` (5 puntos)

3.1. Añade al archivo `urls.py` del proyecto un `include` apropiado que incluya el nuevo archivo `urls.py` que añadiremos en el siguiente paso, perteneciente a la *app* `birds`. Puedes consultar la sección *Escriba su primera vista* (https://docs.djangoproject.com/es/3.1/intro/tutorial01/) del tutorial Polls de Django.

3.2. Crea ese nuevo archivo `urls.py` en la carpeta de la *app* `birds`. Este archivo incluirá un `urlpatterns` con la línea necesaria para conectar el tráfico a `'backend/1/pajaro'` a una función llamada `get_pajaros` que crearemos en el siguiente paso. Dicha función puede residir en el archivo que tú decidas; `views.py`, `rest_facade.py`,... no es relevante.

3.3. Crea la función:

```
def get_pajaros(request):
  ...
```

...descrita anteriormente. Esta función asignará los pájaros de la base de datos a una variable `lista_de_pajaros` con:

```
lista_de_pajaros = Bird.objects.all()
```

Después, en un bucle `for`, añadirá cada pájaro en formato *json* a una lista empleando un código igual o similar a:

```
json = []
for pajaro in lista_de_pajaros:
  json.append({“nombre”: pajaro.nombre, “pais”: pajaro.codigo_pais })
```

Finalmente se devolverá un 200 OK con la lista de pajaros creada en `json`.

- Recuerda efectuar los `import` necesarios
- Recuerda que para devolver una lista es necesario usar el parámetro `safe=False` en el constructor de `JsonResponse`

3.4. Tras probar el código anterior lanzando el servidor y usando, por ejemplo, el navegador web o cURL, añade dicho código a un *commit* con el mensaje “Creado endpoint get_pajaros”.

Criterios de aceptación:

- Un *commit* subido al repositorio tal y como ha sido planteado en la descripción de la tarea.

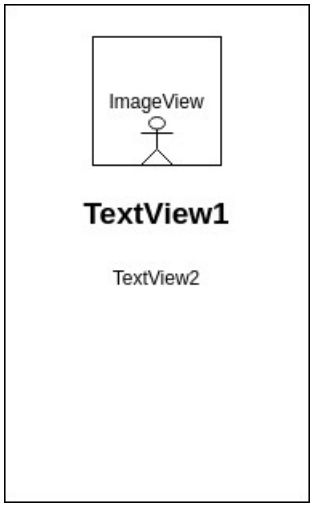
Módulo: Desarrollo de interfaces
Criterio de cualificación: 3 tareas a efectuar en el repositorio del examen. Cada tarea tiene su puntuación en el enunciado. **Contará un 10% de la nota de evaluación.** **Es necesario un 5 para hacer la media ponderada y aprobar la evaluación.**

1. Creación de una aplicación Android (2,5 puntos)

1.1. En el repositorio clonado anteriormente (http://...) crea una nueva carpeta llamada “android”

1.2. Abre Android Studio y crea un nuevo proyecto dentro de dicha carpeta. El proyecto se llamará “Country Birds” y tendrá una *Empty Activity* inicialmente.

1.3. Adapta la interfaz `.xml` de dicha activity para que muestre la siguiente información. Puedes usar `RelativeLayout` ó `ConstraintLayout`, pero se valorará si la interfaz es lo más adaptativa posible



1.4. *TextView1* mostrará el título “**Country Birds**”

1.5. *ImageView* mostrará la imagen que puedes encontrar en al siguiente URL (http://...)

1.6. Conecta *TextView2* a un atributo privado en la clase principal.

1.6.1. Para ello, crea una atributo en `MainActivity`:

```
private TextView miTextView;
```

1.6.2. En el cuerpo de `onCreate`, asigna esa variable a `findViewById(R.id.idDelTextView)`

1.7. Añade los cambios a un nuevo *commit* con el mensaje “Añadida interfaz principal Android”.

Criterios de aceptación:

- Un *commit* subido al repositorio tal y como ha sido planteado en la descripción de la tarea.

2. Obtener los datos del backend (versión sencilla) (2,5 puntos)

2.1. En el método `onCreate`, lanza una petición GET que reciba los datos de un *endpoint* como el creado anteriormente. No es necesario que ataques a `localhost`. En su lugar, lanza la petición a **https://una-ip-cualquiera-donde-tendré-el-backend/backend/1/pajaro**

2.1.1. Puedes consultar la documentación de Volley para hacerlo (https://developer.android.com/training/volley). Recuerda que si usas Volley, tendrás que añadir la dependencia a `build.gradle` del módulo de la *app*:

```
implementation 'com.android.volley:volley:1.1.1'
```

También se permite el uso de cualquier otra solución (e.g.: URLConnection) para lanzar el GET.

2.1.2. Recuerda añadir el permiso de acceso a la red en el manifest.xml

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

2.1.3. Usa `JsonArrayRequest`, no `JsonObjectRequest`.

2.2. En el `onResponse`, emplea el `JSONArray` obtenido y renderiza su contenido en el *TextView2* creado anteriormente.

2.2.1. No es necesario efectuar el parseo de ningún JSON. Únicamente se debe visualizar el contenido del JSON (tal y como lo devuelve el método `.toString()`) dentro del *TextView2*.

2.3. Una vez lo hayas probado, añade los cambios a un nuevo *commit* con el mensaje “Implementada petición REST para recuperar los datos de los pájaros”

Criterios de aceptación:

- Un *commit* subido al repositorio tal y como ha sido planteado en la descripción de la tarea.

3. Muestra la respuesta del servidor en un RecyclerView (5 puntos)

3.1. Sustituye el *TextView2* de la interfaz por un `RecyclerView` que se extienda hasta el inferior de la pantalla, con un margen opcional.

3.2. Crea una clase `MyViewHolder` y un archivo de interfaz .xml separado que represente una celda. El estilo es libre siempre que sirva para mostrar la respuesta del servidor

3.3. Crea una clase `MyRecyclerViewAdapter` que sirva para mostrar una lista con la información de los pájaros en el *RecyclerView*. Se permite que el *Adapter* guarde los datos en una lista de `strings` (`List<String>`) o en una lista de elementos POJO que almacenen por separado el nombre y el código de país.

3.4. Sustituye el código en `onResponse` para que se refresque el *RecyclerView* y muestre los datos de la respuesta del servidor

- No es necesario implementar control de errores más alla de `error.printStackTrace()`
- No es necesario bloquear la interfaz ni mostrar un diálogo de carga

3.5. Añade estos cambios a un *commit* con el mensaje “Implementado RecyclerView en la pantalla principal”

Criterios de aceptación:

- Un *commit* subido al repositorio tal y como ha sido planteado en la descripción de la tarea.