1. Normas

- El examen es individual.
- El examen debe realizarse en los ordenadores del laboratorio, por lo que no está permitido su elaboración en los equipos de los estudiantes.
- Se podrá hacer uso del código de las prácticas realizadas durante el curso.
- Se puntuará la corrección de los resultados, pero también la calidad del código (legibilidad, comentarios, seguimiento del patrón de diseño proporcionado por Django, tolerancia a fallos, etc.).
- Se entregará en Moodle, como resultado del examen, el fichero ZIP producido por el comando `"zip -r ../examen1.zip ."` o similar, asegurándote de incluir todos los ficheros del proyecto.
- Se incluirá un fichero de texto (con el nombre `memoria.txt`) describiendo cómo se ha realizado cada uno de los apartados descritos a continuación.
- La duración del examen es de 105 minutos, por lo que el trabajo debe entregarse por Moodle antes del tiempo establecido.

2. Enunciado

Una conocida plataforma de streaming decide llevar el registro de los canales y usuarios existentes en su sistema, así como de las suscripciones que se llevan a cabo. Para ello, la base de datos que deberá dar soporte a este proyecto sigue el esquema relacional mostrado a continuación:

canal (id, nombreCanal)
usuario (id , nombreUsuario)
suscripcion (id, *canal* (id)⇑, *usuario* (id)⇑, fechaDeSuscripcion)

donde ⇑ indica clave extranjera y el valor por defecto de fechaDeSuscripcion será el momento en que un usuario realiza una suscripción a un canal.

Recuerda que *Django* añade automáticamente el atributo *id* que funciona como clave primaria.

3. Actividades a realizar

1. Crea un proyecto llamado `project` en *Django*. Dentro del proyecto crea una aplicación llamada *application*.

2. Crea una base de datos en PostgreSQL para la persistencia de datos. La base de datos debe llamarse *examen* y debe ser accesible usando la cadena `postgres://alumnodb:alumnodb@localhost:5432/examen`.

3. Crea el modelo de datos y configura el proyecto para que los modelos puedan ser accedidos utilizando el sistema de administración de *Django* (`http://localhost:8001/admin/`).

4. Escribe un script llamado *populate* (que se ejecute utilizando el comando "python manage.py *populate* ") que, al ejecutarse como comando *Django*, inserte los siguientes datos en la base de datos utilizando las facilidades ORM ofrecidas por *Django*.

   usuario (1001,'antonio2985')
   usuario (1002,'respetaCamiones124')
   usuario (1003,'asierUR')

   canal (1001,'elxokasTV')
   canal (1002,'ibai')
   canal (1003,'playz')

   suscripcion (1001,1001,1001,'05-04-2023')
   suscripcion (1002,1001,1002, '11-04-2023')
   suscripcion (1003,1003,1003, '12-04-2023')

5. Puebla la base de datos a través del comando *populate*, así mismo activa la interfaz de administración utilizando como nombre de usuario y password `alumnodb`.

6. Utilizando el template `canal.html` incluido en el Anexo A, crea una página web accesible en la dirección $PROJECT_URL/*application*/*canal*/<int:PK>, de forma que devuelva una lista con todos las suscripciones de un canal con identificador

PK. Nota: *canal*`.html` no debe modificarse. Si la base de datos no contiene ningún canal con el identificador proporcionado, se debe informar al usuario a través de la variable `error`.

7. Creación de un API REST utilizando *Django*. Crea una aplicación `api` en el proyecto *Django* creado en los apartados anteriores y desarrolla en ella un API que acepte peticiones de lectura (GET), creación (POST), actualización (PUT) y borrado de datos (DELETE). El API debe devolver datos en formato JSON y además ser accesible desde el navegador a través de la dirección $PROJECT_URL/api/v1.

8. Crea un proyecto `project_vue` en *Vue.js* que acceda al API creada en el apartado anterior y muestre en su página principal un resumen de los usuarios, canales y suscripciones existentes. Para facilitar la visualización puedes utilizar tres tablas distintas, una para cada categoría (usuarios, canales y suscripciones).

9. Crea un test (a ejecutarse con la orden `python manage.py test` *application*`.tests`) que:

   - borre todos los canales, usuarios y suscripciones
   - cree el *usuario* (1001, 'jordi')
   - cree el *usuario* (1002, 'nacho')
   - cree el *canal* (1001, 'wildproject')
   - cree la *suscripcion* (1001,1001,1001,'08-05-2023')
   - cree la *suscripcion* (1002,1001,1002,'07-05-2023')
   - acceda a la vista relacionada con la URL $PROJECT_URL/*application*/*canal*/<int:> con el identificador de canal 1001
   - compruebe que las suscripciones proporcionadas son las correctas

4. Normas de calificación

Para obtener un máximo de 5 puntos (aprobar) es necesario que tras ejecutar los comandos:

```
dropdb -U alumnodb -h localhost examen
createdb -U alumnodb -h localhost examen
python3 manage.py makemigrations
python3 manage.py migrate
python3 manage.py createsuperuser
python3 manage.py populate
```

Se pueda acceder a la interfaz de administración http://localhost:8001/admin/ utilizando el username/password alumnodb, donde se podrá acceder a todos los modelos y

obtener un listado con los datos enumerados en el punto (4) del apartado anterior. Igualmente es necesario que los datos se persistan en una base de datos creada en PostgreSQL. Además, la página $PROJECT_URL/*application*/*canal*/<int:PK> debe funcionar correctamente. Se deben suministrar todas las variables usadas en el template `canal.html`. Finalmente, se debe incluir un fichero (`memoria.txt`), donde debe incluirse un resumen de las operaciones realizadas, así como los comandos necesarios para llevarlas a cabo.

Para obtener un máximo de 7 puntos se deben satisfacer todos los puntos del apartado anterior, y desarrollar de manera satisfactoria un API que implemente la funcionalidad CRUD utilizando un servidor REST en *Django*.

Para obtener un máximo de 9 puntos se deben satisfacer todos los puntos del apartado anterior, y además desarrollar de manera satisfactoria la vista en *Vue.js*, de forma que acceda correctamente al API REST de *Django*.

Para obtener un máximo de 10 puntos se deben satisfacer todos los puntos del apartado anterior, y además el test debe satisfacer todos los requerimientos solicitados.

En todo caso se penalizará con 1 punto cada minuto de retraso en la entrega del examen.

A. Anexo (template `canal.html` a utilizar)

```
<html>
 <head>
 </head>
<body>

{% if error %}
   {{ error }}
{% else %}
Suscripciones del canal {{ nombreCanal }}
<table>
{% for subs in suscripciones %}
        <tr>
            <th>{{ subs.usuario.id }}</th>
            <td>{{ subs.usuario.nombreUsuario }}</td>
            <td>{{ subs.fechaDeSuscripcion }}</td>
        </tr>
{% endfor %}
</table>
{% endif %}
```

</body>
</html>