2023/24

PROYECTO INTEGRADO

Pablo Rey Ramos

DESARROLLO DE APLICACIONES WEB

Introducción

Este proyecto consiste en la elaboración de ÁGORA, una aplicación para gestionar actividades lúdicas en general. Los proveedores (ofertantes) podrán ponerse en contacto con los consumidores (clientes) y viceversa.

Los ofertantes podrán registrarse como tales, consultar las solicitudes de los clientes y publicar actividades. Los clientes podrán registrarse como tales, consultar las actividades publicadas por los ofertantes, y publicar sus propias solicitudes.

La aplicación se desplegará en la web, de forma que los usuarios sólo necesitarán un navegador y conexión a internet para poder acceder.

Objetivo

La razón de elaboración del proyecto viene dada por la necesidad del ser humano de salir de su espacio de confort e innovar en su tiempo libre. Esta aplicación facilitará el contacto entre proveedores de actividades y demandantes de formas de salir de la rutina en su tiempo ocioso.

Tecnologías de desarrollo

La arquitectura de nuestra aplicación será descentralizada y modular, pudiendo distinguir diferentes secciones:

- En un contenedor Docker, se instalará una imagen de MariaDB que alojará la base de datos, empleada para guardar la información personal de los usuarios, actividades registradas, etc.

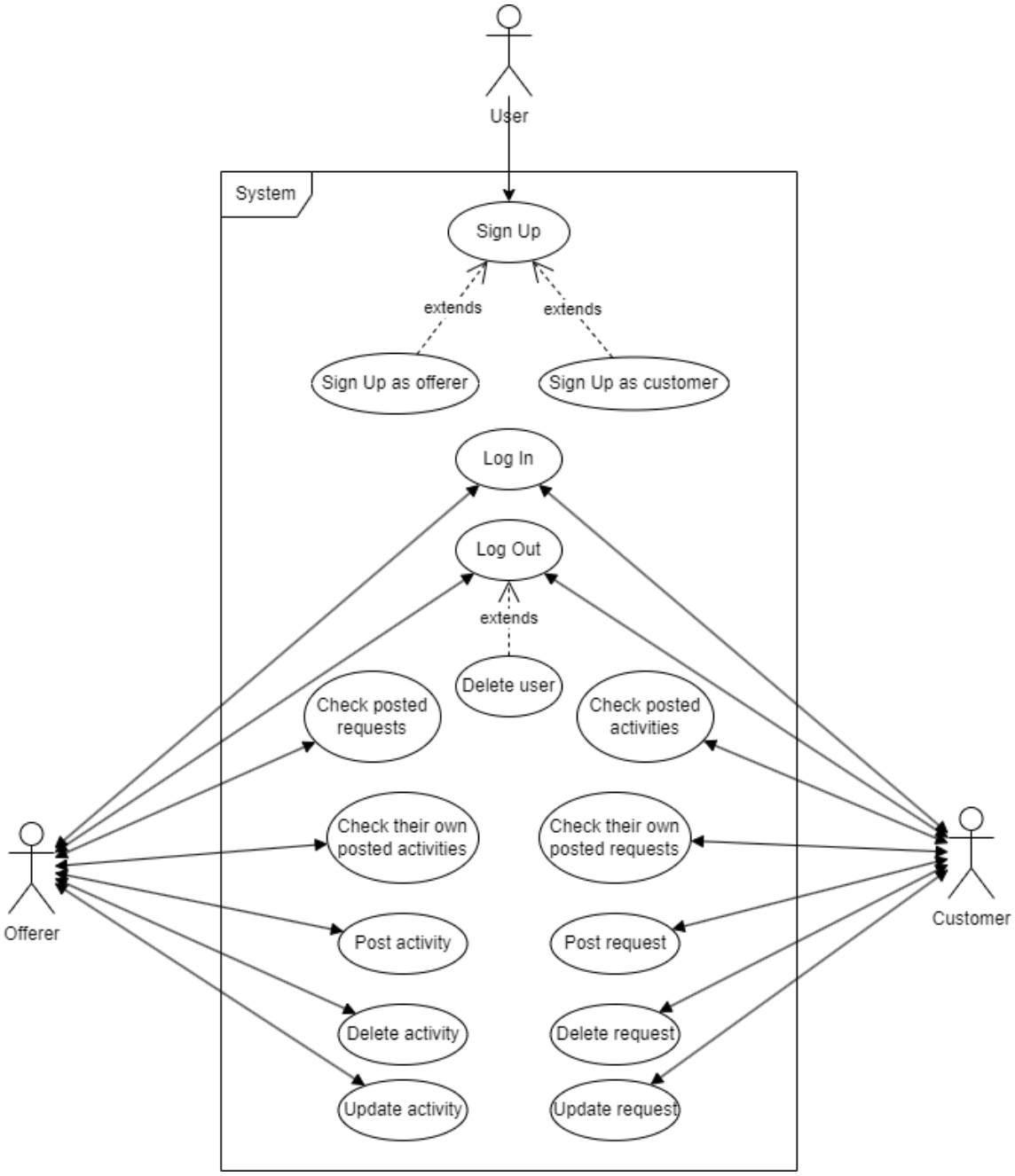
- La base de datos será utilizada mayormente por una API REST realizada con Spring Boot, un *framework* de desarrollo en Java. Este será nuestro *backend*, y nos proporcionará los métodos que serán llamados desde la web en sí y devolverán la información guardada en la base de datos.

- El *frontend* de la aplicación consistirá en un proyecto de Angular, un *framework* de desarrollo en JavaScript óptimo para plantear un diseño modular, y emplearemos algunas librerías adicionales, como Bootstrap, para dotar de un aspecto más atractivo a las vistas del usuario.

Todo el software utilizado es libre y puede instalarse de manera sencilla mediante comandos en el terminal o con sus respectivos comprimidos.

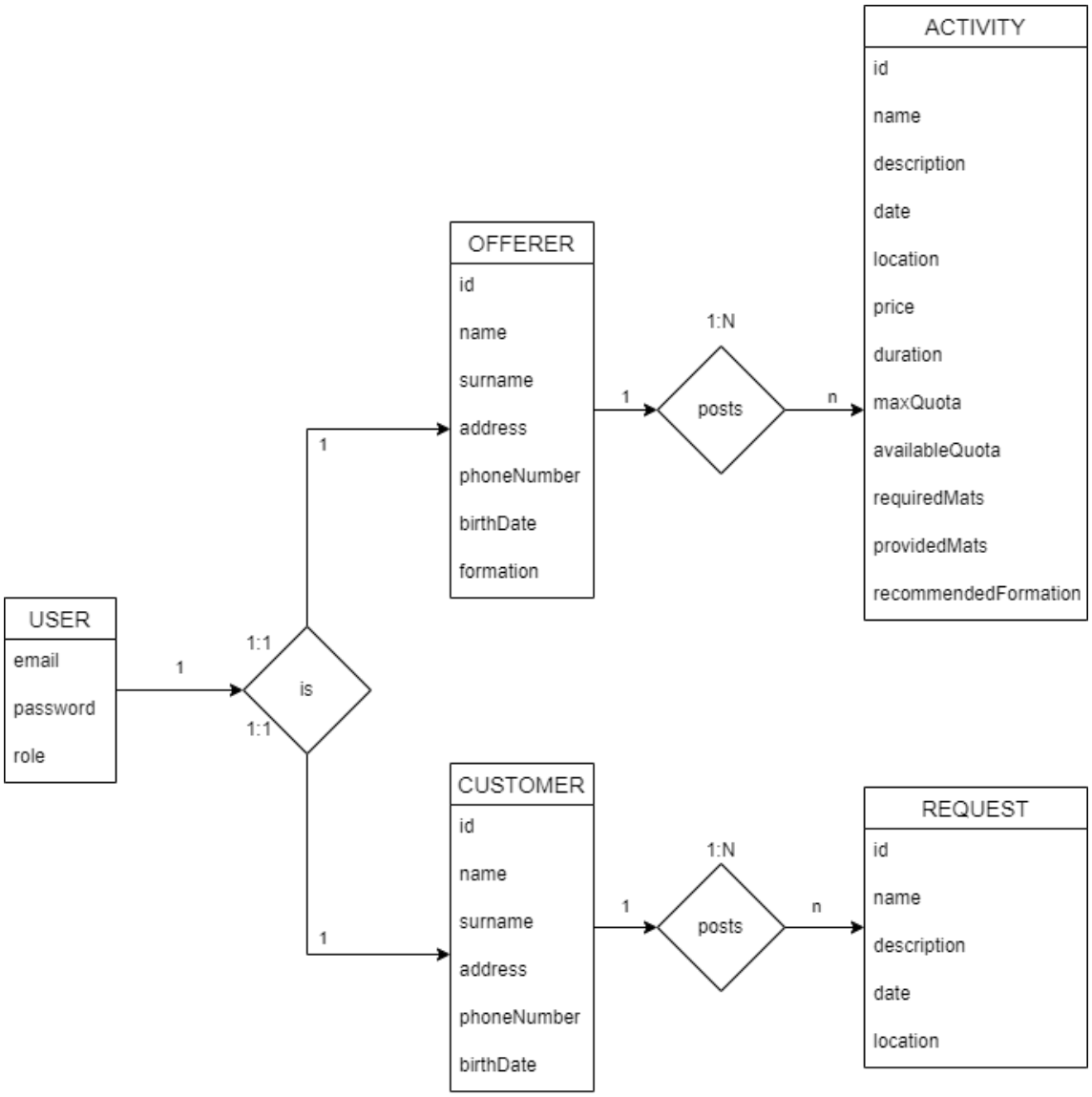
Diagrama de casos de uso

Se contemplan dos roles principales: ofertantes, que gestionarán actividades y consultarán solicitudes; y consumidores, que gestionarán solicitudes y consultarán actividades.



Modelo Entidad – Relación

Este es el modelo que sigue la base de datos de la aplicación. Representa las relaciones entre usuarios de diferente tipo y sus publicaciones correspondientes.



Base de datos (SQL)

Los scripts de creación de tablas y de inserción de datos de ejemplo se detallan a continuación y se adjuntan en el repositorio de GitHub.

```
create schema if not exists proyectoIntegrado;

use proyectoIntegrado;


create table if not exists user
(
    email    varchar(255)                 not null,
    password varchar(255)                 null,
    role     enum ('ROLE_OFF', 'ROLE_CUS') null,
    primary key (email)
);


create table if not exists customer
(
    birth_date   datetime(6)  not null,
    id           bigint auto_increment
        primary key,
    address      varchar(255) not null,
    name         varchar(255) not null,
    phone_number varchar(255) not null,
    surname      varchar(255) not null,
    user_email   varchar(255) null,
    constraint UK_hho535192qm8r8xchi17sqlfn
        unique (user_email),
    constraint FK33ty8iwpkni4gv1mpq8vdpo5u
        foreign key (user_email) references user (email)
);


create table if not exists offerer
(
    birth_date   datetime(6)  not null,
    id           bigint auto_increment
        primary key,
    address      varchar(255) not null,
    formation    varchar(255) null,
    name         varchar(255) not null,
    phone_number varchar(255) not null,
```

```
    surname      varchar(255) not null,
    user_email   varchar(255) null,
    constraint UK_b8xa5c9foprmy9yfl4riaj7g5
        unique (user_email),
    constraint FKr6f7sly9xsi2x71fv73ysmspq
        foreign key (user_email) references user (email)
);

create table if not exists activity
(
    available_quota       int          not null,
    max_quota             int          not null,
    price                 float        not null,
    date                  datetime(6)  not null,
    id                    bigint auto_increment
        primary key,
    offerer_id            bigint       null,
    description           varchar(255) not null,
    duration              varchar(255) not null,
    location              varchar(255) not null,
    name                  varchar(255) not null,
    provided_mats         varchar(255) null,
    recommended_formation varchar(255) null,
    required_mats         varchar(255) null,
    constraint FK55co3xganwfqopw29urr7tq8i
        foreign key (offerer_id) references offerer (id)
);

create table if not exists request
(
    customer_id bigint       null,
    date        datetime(6)  not null,
    id          bigint auto_increment
        primary key,
    description varchar(255) not null,
    location    varchar(255) not null,
    name        varchar(255) not null,
    constraint FK6wuyy6femh1tl1jxmw1ilrs6b
        foreign key (customer_id) references customer (id)
);
```

```
insert into proyectoIntegrado.user (email, password, role)

values    ('cliente@gmail.com',  '$2a$10$AhdPnZ3xlCidVX/CAuruVeRzZXcNLqKRCgL2ZEAyQAMbtUNhPzJ5e',
'ROLE_CUS'), # Contraseña: cus

       ('ofertante@gmail.com',   '$2a$10$4Q6eFtqN3YGXPi0XWUGeb.PxYZQeNugI04DgltjO8SKP51KaeVGpG',
'ROLE_OFF'); #Contraseña: off


insert into proyectoIntegrado.offerer (birth_date, id, address, formation, name, phone_number,
surname, user_email)

values  ('2003-06-12 02:00:00.000000', 13, 'c/Calle 1, 1 A', null, 'John', '123456789', 'Doe
Second', 'ofertante@gmail.com');


insert into proyectoIntegrado.customer (birth_date, id, address, name, phone_number, surname,
user_email)

values    ('1992-07-30  02:00:00.000000',  7,  'c/Calle  1,  1B',  'John',  '987654321',  'Doe',
'cliente@gmail.com');


insert into proyectoIntegrado.activity (available_quota, max_quota, price, date, id, offerer_id,
description, duration, location, name, provided_mats, recommended_formation, required_mats)

values  (5, 5, 15, '2024-07-28 00:00:00.000000', 13, 13, 'Quedada para asistir juntos al festival
de música folclórica de Hawáii.', '3 horas', 'La Cartuja', 'Festival de música hawaiana', null,
null, null);


insert into proyectoIntegrado.request (customer_id, date, id, description, location, name)

values  (7, '2024-07-01 01:30:00.000000', 10, 'Me gustaría encontrar un grupo de gente para salir
a hacer ejercicio: correr, calistenia, jugar a cualquier deporte de equipo, etc.', 'Parque
Miraflores', 'Ejercicio en el parque');
```

Manual de usuario

Para poner en marcha el proyecto, primero necesitaremos instalar Docker en nuestra máquina y ejecutar el comando:

```
docker run -d -it --name proyectoIntegrado -p 3336:3306 -e
MYSQL_ROOT_PASSWORD=XXXXX mariadb
```

Eso nos creará un contenedor sobre una imagen de MariaDB y lo lanzará.

Después, debemos instalar Node.JS y NPM, y sobre el directorio del *frontend* ejecutar los comandos:

```
npm i
ng build
```

Esto nos generará los archivos de producción en el directorio /dist/proyecto_integrado_front/browser/. Estos archivos debemos copiarlos sobre el directorio del *backend* /proyectoIntegradoAPI/src/main/resources/static/. Si existe el

directorio /target, debemos eliminarlo. Luego, ejecutamos el comando:

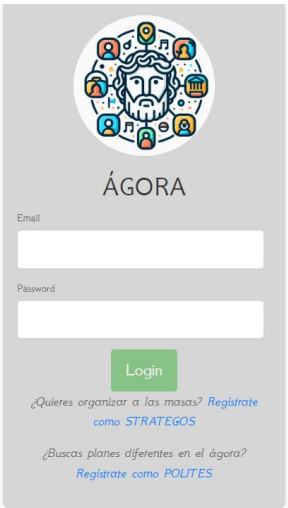
```
mvn clean package
```

Si el empaquetado es correcto, generará un archivo JAR con un nombre similar a *proyectoIntegradoAPI-0.0.1-SNAPSHOT.jar* en dicho directorio /target. Este tipo de empaquetado no necesita de contenedores externos para poder ser ejecutado. Un archivo WAR requiere de un contenedor de aplicaciones (como Tomcat) para ser ejecutado, y dio problemas en la aplicación cuando se intentó su despliegue. Con el JAR, se emplea el propio Tomcat embebido en Spring.

Para ejecutar el archivo JAR, se abrirá una consola en el directorio /target y ejecutamos el comando:

```
java -jar {nombre del archivo JAR}
```

La conocida consola de Spring se mostrará, y sabremos que la aplicación se ha iniciado correctamente cuando veamos el mensaje de que el Tomcat de Spring se ha iniciado en el puerto 8080. Así, si abrimos un navegador y accedemos a http://localhost:8080, podremos entrar en la pantalla inicial de la aplicación.

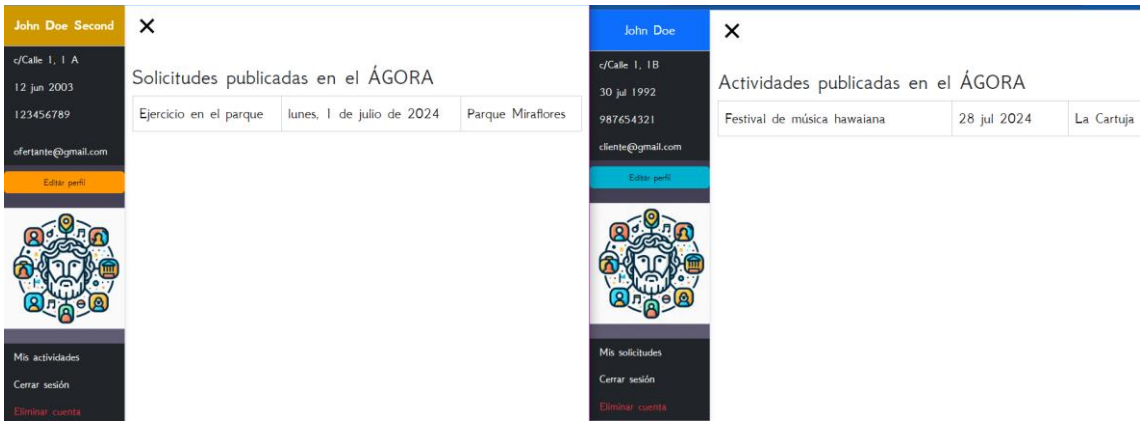


ÁGORA es una aplicación fundamentada en dos roles bien diferenciados respecto a las actividades: los que las proponen y los que las demandan. Se ha tematizado en el marco de la Grecia clásica, donde el ágora de la *polis* era el centro de la vida urbana y congregaba la política, la cultura, la economía, la religión y toda clase de actividades.

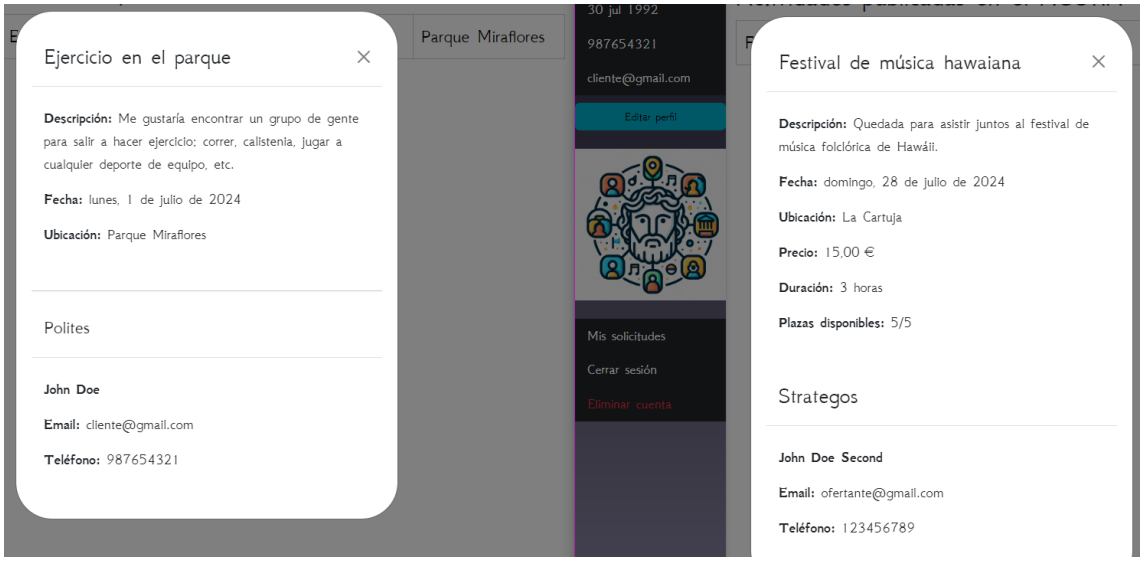
Los *STRATEGOS* (στρατηγός) eran los líderes militares de la época, similares a policías o generales, lo que los identifica con los organizadores de la vida pública. Del mismo modo, se ha interpretado que son los que tomarán la iniciativa, es decir, los ofertantes.

Los *POLITES* (πολῖτες) eran los ciudadanos de las *polis* griegas, con plenos derechos políticos, jurídicos y religiosos, pero también con obligaciones fiscales y militares. Así como los *polites* acuden al ágora como centro de reunión cultural, los consumidores accederán al sistema buscando actividades y solicitándolas.

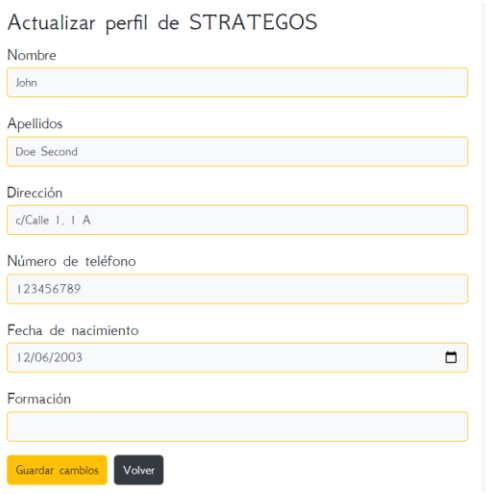
Al acceder con cualquiera de los dos roles, veremos perfiles análogos, pero fundamentalmente opuestos. Los *strategos* verán las solicitudes publicadas, por si quieren crear actividades acordes. Los *polites* verán las actividades publicadas, para decidir en cuál quieren inscribirse.



Si hacemos click en las filas de la tabla, podremos acceder a los datos de la actividad/solicitud, y con ello a los datos de contacto de su publicador.



Al hacer clic en “Editar perfil” podremos modificar casi todos los datos personales.



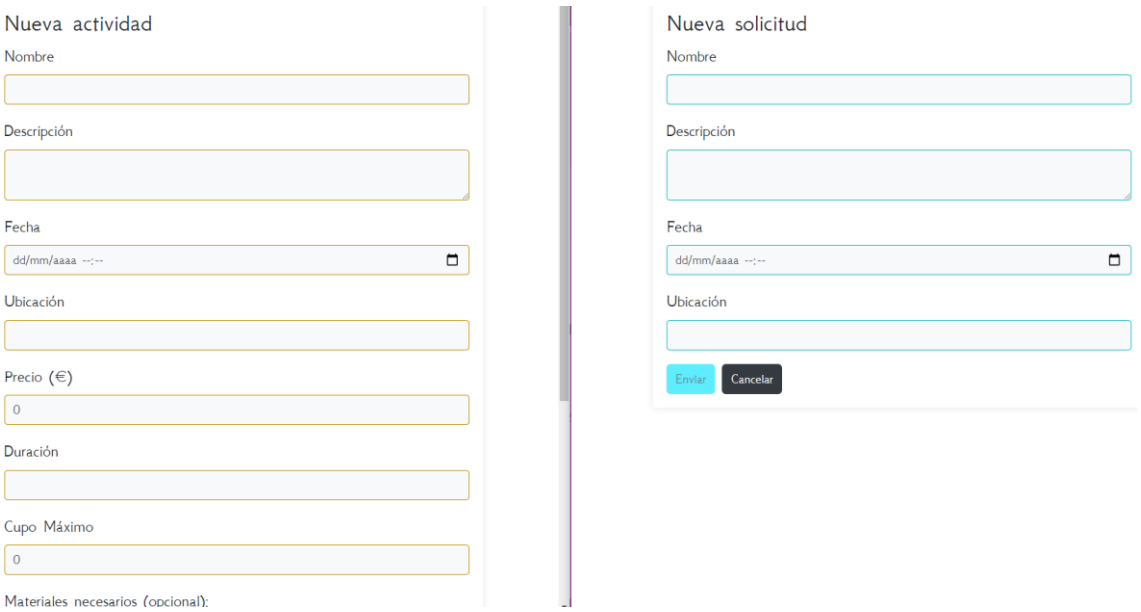
El botón de "Cerrar sesión" nos lleva de vuelta a la pantalla de inicio, y el de "Eliminar cuenta" borra TODA la información relacionada con el usuario del sistema: esto es, credenciales, datos personales y publicaciones asociadas.

Al hacer clic en "Mis actividades/solicitudes", accedemos a nuestra sección personal.

El botón "Borrar" elimina la publicación del sistema.

El botón "Editar" abre un formulario para modificar los detalles de la publicación.

Con el botón de “Nueva actividad/solicitud”, podemos detallar las características de una publicación para insertarla en el sistema.



Con esto, conseguimos una aplicación sencilla con un *backend* robusto y un *frontend* intuitivo que satisface los requerimientos y objetivos de este proyecto integrado. Además, la aplicación es escalable, pudiéndose añadir numerosas funcionalidades y optimizaciones en el futuro.