GUIA PRACTICA 3

React y Django I

Desarrollo de aplicaciones y servicios

Introducción

En el desarrollo de esta práctica se va a implementar la funcionalidad de crear, editar y eliminar **subastas,** así como realizar **pujas**.

Para ello será necesario ampliar la práctica 2 de React, centrada en el desarrollo frontend con React. También construiremos un servidor usando el framework de Django visto en clase para atender las peticiones del cliente y proporcionar las funciones CRUD de subastas y pujas.

Organización del trabajo

Al igual que se realizó en la práctica anterior, se ha de crear en Git Project un nuevo ***milestone*** con nombre "**Sprint 3**" donde se realizará la distribución de tareas entre los integrantes del grupo de trabajo.

Se creará la documentación del sprint donde se debe de indicar de manera clara y resumida:

- Qué tareas se van a realizar.
- Organización de la carga.
- Dificultades encontradas en su realización.
- Desviaciones que se hayan podido producir.

El desarrollo de esta práctica se realizará en la rama "**feature/p3**", siguiendo la metodología de trabajo de gitFlow.

Se deberá crear un nuevo repositorio para albergar el código del servidor.

TTP_BACK

Tendremos dos repositorios independientes: uno para el frontend y otro para el backend.

Añadid a los profesores de la asignatura al nuevo repositorio:

- Álvaro Ruiz Calzada (Grupo A): ***arcalzada***
- Cristian Fernández del Pozo (Grupo B): ***cferndp***
- David Tortosa Escudero (Todos): ***esdtortosa***

Y aseguraros de que los profesores tienen visibilidad antes de la entrega.

La documentación de Scrum (milestones, etc) irá en el repositorio de Frontend (el que ya tenéis creado de la P1 y P2)

Estructura de URLs para peticiones Back-end

Cualquier aplicación web funciona mediante URLs y tipos de llamada (GET, PUT, POST, DELETE) mediante las cuales se pueden localizar los recursos a los que se accede en el servidor.

CRUD son las siglas de Create, Update, Read y Delete por lo que hay que crear la lógica completa para la gestión completa de la información.

Se van a desarrollar el CRUD de tres servicios:

- Categorías.
- Subastas.
- Pujas.

estos 3 CRUD

Para las subastas se crearán estas URLS, se entiende {URL} como la dirección base de despliegue:

que podamos recuperar las subastas y por texto de búsqueda
like

- **{URL}/subastas** -> **GET**, obtener subastas.
- **{URL}/subastas?texto** -> **GET**, obtener subastas que contenga en algún campo (título o descripción) la palabra texto.
- **{URL}/subastas?categoría**-> **GET**, obtener subastas de una determinada categoría.
- **{URL}/subastas?precioMin&precioMax**-> **GET**, obtener subastas de un rango de precio de salida.
- **{URL}/subastas**-> **POST**, crear subasta.
- **{URL}/subastas/:id_subasta** -> **GET**, obtener subasta.
- **{URL}/subastas/:id_subasta** -> **PUT**, actualizar subasta.
- **{URL}/subastas/:id_subasta**-> **DELETE**, eliminar subasta.

desde el front: si ya hay un id es editar o borrar si no creamos

En el caso del CRUD de categorías las URLs serán las siguientes:

- **{URL}/subastas/categorias** -> **GET**, obtener categorías.
- **{URL} /subastas/categorias** -> **POST**, crear categoría.
- **{URL}/subastas/categoría/:id_categoria** -> **GET**, obtener una categoría.

- **{URL}/subasta/categoría/:id_categoria** -> **PUT**, actualizar una categoría.
- **{URL}/subasta/categoría/:id_categoria** -> **DELETE**, eliminar una categoría.

QUIEN LLEVA LOGICA Y CONTROL ES BACK

En el caso de pujas las URLs serán las siguientes:

logica de la validacion del apuja front y revalidar en BACK

- **{URL}/subastas/:id_subasta/pujas/**-> **GET**, obtener las pujas de una determinada subasta.
- **{URL}/subastas/:id_subasta/pujas/**-> **POST**, crear una nueva puja en una determinada subasta.
- **{URL}/subastas/:id_subasta/pujas/:idPuja**-> **GET**, obtener una puja determinada.
- **{URL}/subastas/:id_subasta/pujas/:idPuja**-> **PUT**, actualizar una puja.
- **{URL}/subastas/:id_subasta/pujas/:idPuja**->**DELETE**, eliminar una puja.

BODY es dependiente del proyecto. LO ADAPTAMOS NOSOTROS

NOMBRES DE FRONT TIENEN Q COINCIDIR CON LOS DE BACK

Las peticiones de tipo POST y PUT tienen asociado un **body** (en formato json) el cual es totalmente dependiente de como llaméis y gestionéis los datos, por lo que tendréis que adaptarlo a vuestro proyecto.

No hay ningún problema en el caso de querer traducir el nombre de la URLs a inglés, pero respetad la estructura que se está siguiendo.

Obtención de las categorías

Las categorías se deberán mostrar un desplegable al crear o editar una subasta. En este desplegable se mostrarán todas las categorías de la base de datos. O, dicho de otra forma, React realizará una llamada al servidor para obtener el listado de categorías y se las presentará al usuario para que elija. En el caso de estar editando una subasta, el desplegable debe tener seleccionado la categoría actual.

De la misma manera que tenemos un listado de categorías al crear o editar, es igualmente necesario obtener ese listado al realizar las búsquedas de las subastas para que los ids sean coherentes al hacer el filtrado. Alternativamente, se puede implementar un filtro dinámico que recorra las categorías de las subastas obtenidas y genere el listado a partir de los datos obtenidos.

Pantalla de Subastas

Se ha de añadir la forma de crear una nueva subasta. Esta opción solo debe estar disponible para usuarios registrados. La forma de añadir esa subasta queda a elección del estudiante para que lo adaptado a la estética y funcionalidad que ya tiene desarrollada.

Los campos mínimos de creación de subastas deben de ser:

- Título.
- Descripción.
- Fecha límite para su cierre.
- Fecha de creación de la subasta.
- Campo de imagen.
- Precio de salida.
- Stock.
- Valoración.
- Categoría.
- Marca.

Adicionalmente, aunque no sea visible en el formulario, tened en cuenta que cada subasta tiene un identificador único que controlara el backend. Y que deberá haber un campo estado que indique si aún se puede pujar (abierta) o ya ha finalizado (cerrada).

VALIDACIONES MÍNIMAS, LAS TIENE QUE HACER

El backend deberá implementar validaciones que garanticen la integridad de los datos.

- Se deberá validar que el stock y el precio sean números naturales positivos.
- La valoración podrá tomar valores comprendidos entre 1 y 5, siendo la mejor valoración 5.
- Así mismo, la fecha de cierre de la subasta no podrá ser menor ni igual a la fecha de creación.
- De hecho, la fecha de cierre deberá ser como mínimo 15 días mayor que la de creación.

También es requisito que el backend gestione correctamente los errores 400 y 404.

Recuerda implementar en el servidor los tres servicios que permiten filtrar la lista de subastas por texto, categoría o rango de precio.

Pantalla de Detalle de Subasta

Dado que ahora ya se pueden empiezan a recuperar subastas con datos propios, se va a modificar la pantalla de detalle de subasta para poder obtener la información de esta desde el servicio de subastas de Django que vamos a desarrollar nosotros.

Es importante recalcar que la pantalla de React que se utilice para la creación de una subasta puede ser la misma que la de edición. La forma de hacer la distinción entre ambas será por el id de la subasta.

De esta forma, si el id de subasta viene informado, estaremos realizando la operación de editar. Sin embargo, si el id no viene, estaremos ante la creación de una nueva subasta. Esto se debe a que el id es un campo que genera la base de datos y que no se informa hasta que se crea.

El body para editar la subasta es el mismo que el de crear, pero solo se actualizarán los datos que se envíen.

Sección de pujas

Cuando se consulte una subasta deberá aparecer una sección dedicada a las pujas. Esta consistirá en una lista de las pujas recibidas ordenadas en orden descendente de precio. (Ejemplo: 1000 -> 800- > 500 -> ...)

De forma análoga a la creación de subasta, la opción de pujar solo debe estar disponible para usuarios registrados.

Los campos mínimos de una puja deben de ser:

- Precio de la puja.
- Fecha de creación de la puja.
- Pujador; es decir, nombre de usuario de la persona que hace la puja.
- Identificador de la subasta a la que pertenece la puja.

Además, cada puja tendrá su propio identificador gestionado por el backend.

El back deberá implementar validaciones que garanticen la integridad de los datos.

- El precio de la puja deber ser un número natural positivo.
- El precio de una nueva puja debe ser mayor que la anterior puja ganadora.

- Solo se podrá pujar si la subasta se encuentra en estado abierto; es decir, si la fecha de cierre de la subasta es posterior a la fecha actual.

Aparte de la creación de pujas, es necesario que tanto el frontal como el backend permita a un usuario editar y borrar una puja; siempre y cuando la puja esté en estado abierta.

También es requisito que el backend gestione correctamente los errores 400 y 404.

Despliegue e integración

Para la integración, seguimos el paradigma de gitFlow, una vez finalizado el desarrollo, se ha de realizar la PR a develop y posteriormente a main.

Una vez ya este el código integrado en main, se realiza dos despliegues:

- El frontend se desplegará en Vercel, como se ha estado realizando en las prácticas previas.
- El backend se desplegará en Render, siguiendo el anexo de despliegue proporcionado en el tema 3.

Entregable

No hay entregable individual de la práctica, ya que esta se divide en dos partes, siendo esta la primera. Se realizará una entrega conjunta de las dos prácticas en un único objeto entregable. A partir del **lunes 31 de marzo** los profesores podrán pedir revisar el desarrollo de este trabajo.