PRÁCTICA 1:

DISEÑO E IMPLEMENTACIÓN DE API REST

ADI 2019/2020

EDUARDO CORREAL BOTERO

CASOS DE USOS IMPLEMENTADOS:

- Un usuario debe poder hacer login en la aplicación
- Un usuario sin estar autentificado debe poder ver los datos más importantes de la lista de publicaciones
- Un usuario sin estar autentificado debe poder ver todos los datos de una publicación
- Un usuario sin estar autentificado debe poder ver los datos más importantes de los usuarios.
- Un usuario sin estar autentificado debe poder ver los comentarios de una publicación.
- Un usuario autentificado debe poder comentar en una publicación
- Un usuario autentificado debe poder actualizar su información
- El usuario que ha creado una publicación, si esta autentificado debe poder eliminar esa publicación.
- El usuario que ha creado una publicación, si esta autentificado debe poder eliminar todos los comentarios de su publicación
- Un usuario autentificado debe poder crear un grupo
- Un usuario sin estar autentificado debe poder ver los datos más importantes de todos los grupos.

OTROS CASOS DE USOS:

- Un usuario autentificado debe poder seguir una publicación.
- El usuario que ha creado una publicación, si está autentificado debe poder enviar actualizaciones sobre el estado de este.
- Un usuario debe poder unirse a un grupo
- Un usuario autentificado debe poder seguir a otro usuario
- El usuario que ha creado una publicación, si esta autentificado debe poder eliminar un comentario de su publicación.
- El usuario que ha creado un comentario en una publicación, si esta autentificado debe poder eliminar dicho comentario.
- El usuario que ha creado un comentario en una publicación, si esta autentificado debe poder actualizar dicho comentario.
- Un usuario autentificado debe poder borrar los grupos que ha creado.
- Un usuario sin autentificar debe poder ver todos los datos de un grupo

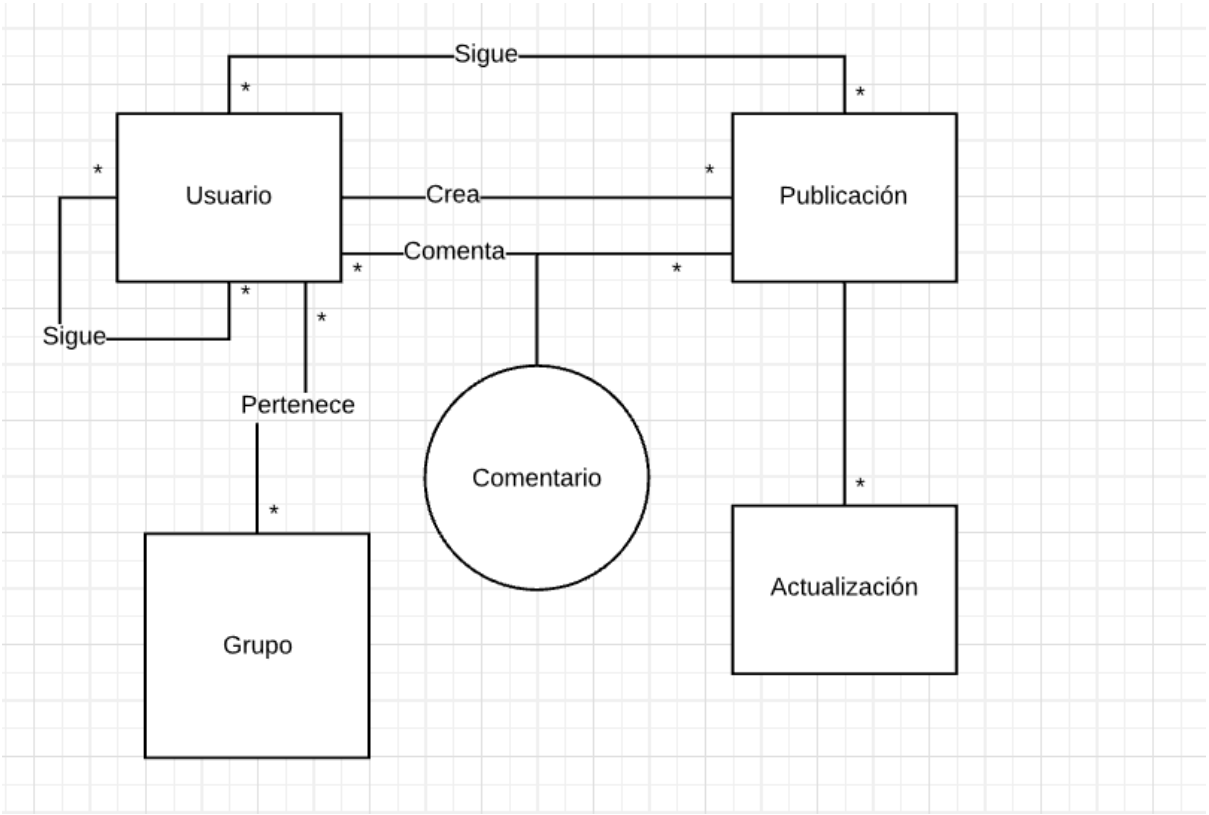
RELACIÓN ENTRE LLAMADAS A LA API Y CASOS DE USOS:

La url base de la api es localhost:3000/api

TIPO DE PETICIÓN	API	CASO DE USO
POST	/login	Un usuario debe poder hacer login en la aplicación
GET	/publicaciones	Un usuario sin estar autentificado debe poder ver los datos más importantes de la lista de publicaciones
GET	/publicaciones/:idPublicacion	Un usuario sin estar autentificado debe poder ver todos los datos de una publicación
GET	/usuarios	Un usuario sin estar autentificado debe poder ver los datos más importantes de los usuarios
GET	/publicaciones/:idPublicacion/comentarios	Un usuario sin estar autentificado debe poder ver los comentarios de un proyecto.
POST	/publicaciones/:idPublicacion/comentarios	Un usuario autentificado debe poder comentar en una publicación
PUT	/usuarios/:nombre	Un usuario autentificado debe poder actualizar su información
DELETE	/publicaciones/:idPublicacion	El usuario que ha creado una publicación, si esta autentificado debe

		poder eliminar esa publicación.
DELETE	/publicaciones/:idPublicacion/comentarios	El usuario que ha creado una publicación, si esta autentificado debe poder eliminar todos los comentarios de su publicación
POST	/grupos	Un usuario autentificado debe poder crear un grupo
GET	/grupos	Un usuario sin estar autentificado debe poder ver los datos más importantes de todos los grupos.

DIAGRAMA RELACIONES ENTRE RECURSOS:



BASE DE DATOS USADA:

Para la práctica he usado como base de datos postgresql, usando la librería pg para la conexión y consultas, y node-pg-migrate para la migración de la base de datos.

Pasos para migrar la base de datos:

-Crear una base de datos en postgres

-En config/default.json poner la información para la conexión a la base de datos.

-Para hacer la migración usar: npm run migrate up

-Para poblar la base de datos en db/database.js en la línea 8 poner los datos de la conexión a la base de datos

-En config.env cuando se quiera poblar la base de datos, se debe poner POBLAR = true

-Al poner en marcha el servidor se hará la población de la base de datos.

-Si se quieren borrar todas las tablas, se puede usar el comando npm run migrate down

REQUISITOS ADICIONALES:

Los requisitos adicionales que he hecho son:

-Base de datos, en db/database.js y la carpeta migrations

-Llamadas a api externa:

- Api para calcular la edad de una persona según su nombre: https://agify.io/, se encuentra en db/database.js poblarUsuario (línea 158)
- Api para obtener una imagen de perro aleatoria https://dog.ceo/, se encuentra en db/database.js poblarPublicacion (línea 95)

-Documentación del api, con swagger, está en el archivo swagger.yml y se puede ver abriendo redoc.html en un navegador

HERRAMIENTA USADA PARA LOS TESTS:

Postman, los tests están en:

https://documenter.getpostman.com/view/8741913/SVzxagP5?version=latest

Desde ahí la mejor opción para ejecutar los tests es clickear en run en postman, donde te abrirá la aplicación de escritorio de postman importando la colección de las peticiones, cada una con sus tests y también el entorno. Si el token ya no es válido por alguna razón, solo se tendría que lanzar la petición de login y el token dado ponerlo en la variable de entorno tokenAuth.

La otra opción es con el archivo exportado de postman:

ADI practica 1.postman_collection.json

El problema de este método es que al parecer no coge el entorno, por lo que tocaría crear uno el entorno para tener las variables baseUrl y tokenAuth, o reemplazarlas a mano.