PRÀCTICA API REST : Consum de dades

Una API REST és una interfície web que ofereix serveis des del costat del backend.
Aquest tipus d'interfície canalitza les interaccions amb la base de dades, això ho fa definint un seguit de URLs en funció de l'acció a realitzar. Ofereix informació consumible mitjançant protocol HTTP.

HTTP Method	CRUD Action
POST	Create
GET	Read
PUT	Update
DELETE	Delete

Aquesta metodologia d'ordenació de l'accés a les dades permet deslligar el desenvolupament del backend del frontend, és a dir, de les dades de l'aplicació i del client que usarà les dades en última instància.

Aquí tens un exemple de consum d'API REST. Concretament es sol·liciten els recursos de PokéAPI: https://pokeapi.co/

https://github.com/xbaubes/DesenvolupamentWeb/blob/main/Frontend/API/exemple/

Exemples usant .then() i async/await amb Node.js:
https://github.com/xbaubes/DesenvolupamentWeb/tree/main/Frontend/API/exemple/pokeapiNode

S'hi crida l'API REST amb la funció asíncrona *fetch* i per controlar-ne el flux de dades s'utilitzen *promises* i *async/wait* que fa el codi més llegible i similar a l'estructura síncrona.
No s'hi mostren *callbacks*, que és un sistema més antic de gestió de flux.

En aquesta pràctica usarem una fake API que retorna la informació en format JSON:
{JSON} Placeholder
És un fake API perquè les dades ofertes no provenen d'una base de dades sinó d'un script, les dades són estàtiques i no són modificables, tot i que es simula per propòsits educatius.

Rutes bàsiques que ofereix la API	
/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

Com formatejar en JSON les dades rebudes

Podem utilitzar una extensió de navegador per donar a les dades rebudes un aspecte que els fa més fàcils de llegir, per Google Chrome pots provar JSON Formatter.

Funcions útils per treballar amb grans volums de dades

Hi ha funcions d'arrays especialment dissenyades per treballar amb arrays d'objectes, per exemple: *find()*, *filter()* o *map()*.
Per cercar dins arrays d'objectes és molt útil la funció find(), ja que permet personalitzar la condició de cerca; find() retorna el primer element que compleix la condició, si no el troba retorna «undefined».

Exemple d'ús:

```
const animals = [
  { id: 1, nom: 'Rex', especie: 'gos', edat: 5 },
  { id: 2, nom: 'Mia', especie: 'gat', edat: 3 },
  { id: 3, nom: 'Nemo', especie: 'peix', edat: 1 }
];

function buscarAnimalPerNom(nom) {
  // Es compara el nom de cada animal amb el nom proporcionat
  const animal = animals.find(a => a.nom === nom);
  return animal;
}

const resultat = buscarAnimalPerNom('Mia');
console.log(resultat); // { id: 2, nom: 'Mia', especie: 'gat', edat: 3 }
```

ACTIVITAT

Coneixedors de la brillant fornada de programadors web a l'Institut del Pla de l'Estany, uns inversors de la zona sol·liciten els nostres serveis per realitzar una xarxa social per la comarca del Pla de l'Estany: L'Estany Social.

Ja han creat el logo i la seva paleta de colors, l'hauràs d'incloure a l'aplicació:
https://github.com/xbaubes/DesenvolupamentWeb/tree/main/Frontend/API/exercici-1-consum-json-logo
De totes maneres, estan disposats a realitzar qualsevol canvi en pro de millorar el resultat final.

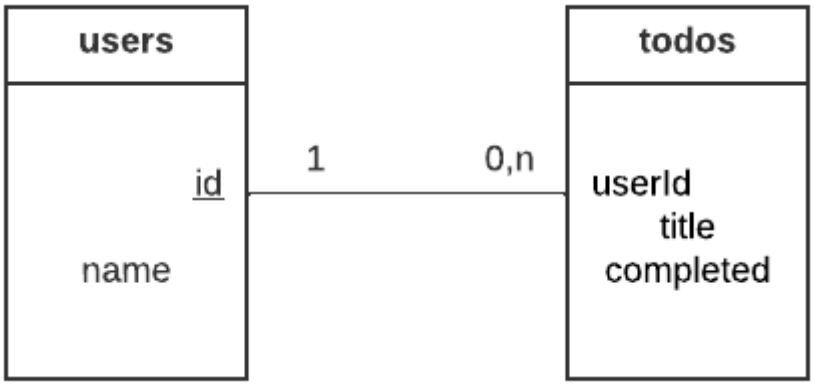
Prèviament a la implementació del disseny, realitzarem una representació visual bàsica que haurà de ser aprovada pel client:
https://github.com/xbaubes/AplicacionsInformatiques/blob/main/Wireframe/Wireframe.pdf

Pel que fa a l'obtenció de les dades, com a API provisional s'usarà JSONPlaceholder. Pots utilitzar tots els camps que creguis interessants encara que no apareguin a les especificacions, l'API final els inclourà.
Fes el filtratge amb la pròpia petició a l'API. {JSON} Placeholder permet paràmetres de consulta, per exemple amb aquesta URL obtindríem la photo amb id=3000:
https://jsonplaceholder.typicode.com/photos?id=3000

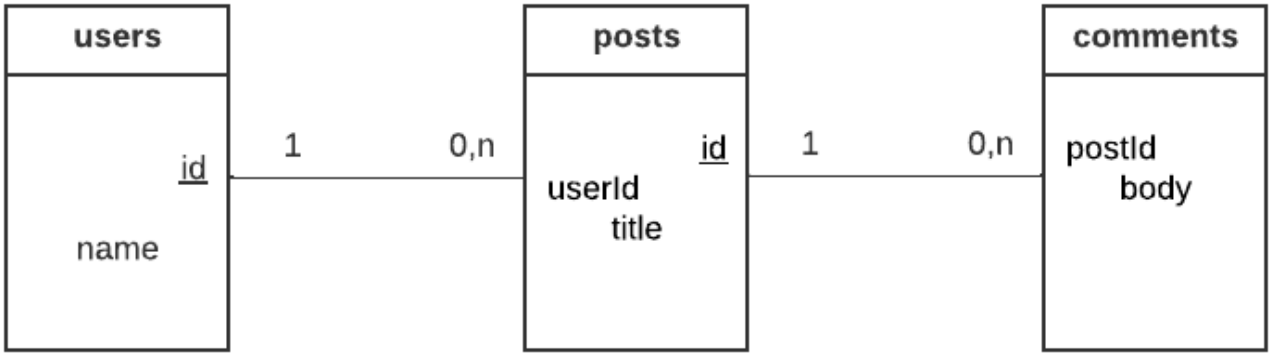
Mostra el resultat de la petició a la pàgina principal o a una de secundària.
Es valorarà la qualitat de la interfície elaborada. Pots realitzar totes les millores que creguis oportunes, es valorarà positivament.

A continuació es llisten els requisits mínims que ha de complir l'aplicació, integra'ls de manera que quedi una aplicació el més user-friendly possible.

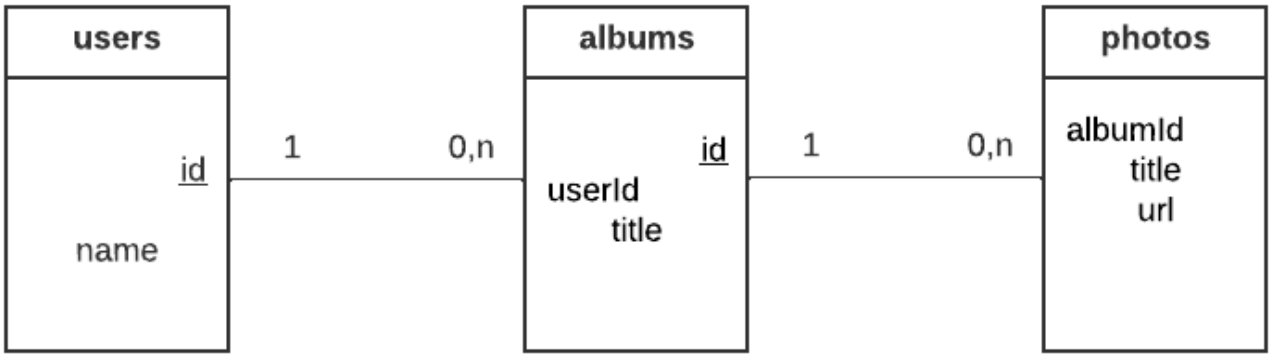
1. Obté el títol de totes les tasques d'un usuari, indica si està finalitzada o no.
   Per exemple, de la usuària Glenna Reichert.



2. Mostra els missatges (post) d'un usuari i els comentaris que ha rebut per cadascun d'ells, mostra-ho com una llista.
   Per exemple, de la usuària Glenna Reichert.



3. Mostra les fotos amb el seu títol i agrupades per àlbums d'un usuari.
   Per exemple, de la usuària Glenna Reichert.



4. Afegeix tots els usuaris en un select, un cop se n'escolleixi un, que s'habiliti la possibilitat de consultar cadascun dels resultats dels exercicis previs. Indica sempre el nom de l'usuari i què s'està mostrant.

5. **OPCIONAL** Crea comandes per afegir, modificar i esborrar registres de la base de dades.

BIBLIOGRAFIA I WEBGRAFIA

«DIRe. Manual de API de Consumo para integradores».
https://administracionelectronica.gob.es/ctt/resources/Soluciones/2412/Descargas/Manual%20de%20uso%20del%20API%20Consumo-v1.pdf?idIniciativa=2412&idElemento=12469
«LOGO». https://app.logo.com/
«MDN Web Docs. Fetching data from the server».
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data
«Microsoft. Consumo de servicios web de RESTful».
https://docs.microsoft.com/es-es/xamarin/xamarin-forms/data-cloud/web-services/rest
«Profile. Cómo consumir una API desde JavaScript nativo». https://profile.es/blog/consumir-api-javascript/