Desarrollo Backend con NodeJS

Promesas

course[It]

Las promesas son objetos de Javascript, y como todo objeto, se pueden instanciar

```
const promise = new Promise(function(resolve, reject) {

 // hacemos algo

 if (/* todo sale bien  */) {
   resolve("esto es lo que recibimos en un try");
 }
 else {
   reject(Error("esto es lo que recibimos en un catch"));
 }
});
```

Las promesas son objetos de Javascript, y como todo objeto, se pueden instanciar

```
const promise = new Promise(function(resolve, reject) {

 // hacemos algo

 if (/* todo sale bien  */) {
   resolve("esto es lo que recibimos en un try");
 }
 else {
   reject(Error("esto es lo que recibimos en un catch"));
 }
});
```

?Me creen si les digo que ya usamos promesas?

Esto es una promesa:

```
axios.get(`https://api.github.com/users/${req.params.name}`)
```

Y esta es la manera ‘moderna’ de resolver y rechazar promesas

```
try {
// si la promesa se resuelve
   const user = await axios.get(
     `https://api.github.com/users/${req.params.name}`
   );
   console.log(user.data);
   res.status(200).json(user.data);
 } catch (e) {
// si la promesa se rechaza
   res.sendStatus(400);
 }
```

Pero a veces nos podemos encontrar con otra forma de resolver y rechazar promesas

```
   axios
     .get(`https://api.github.com/users/${req.params.name}`)
     .then(user => {
// si la promesa se resuelve
       console.log(user.data);
       res.sendStatus(200);
     })
     .catch(error => {
// si la promesa se rechaza
       res.status(400).send("Usuario no encontrado");
     });
```

Al ser objetos también tienen métodos que podemos utilizar para consumirlas

- `Promise.then()` -> recibe como parámetro una función que se ejecuta si la promesa se resuelve
- `Promise.catch()` -> recibe como parámetro una función que se ejecuta si la promesa se rechaza
- `Promise.all()` -> recibe como parámetro un array de promesas y espera a que se resuelvan todas para realizar una acción

Promise.all es muy útil cuando tenemos varias llamadas encadenadas a la misma api

```
// este código busca en la poke api un array de pokemones, cuando vuelve
// la data de todos ahí recién nos da el return

const pokeData = await Promise.all(
  pokemones.map(async pokemon => {
    const pokeData = await axios.get(
      `https://pokeapi.co/api/v2/pokemon/${pokemon}`
    );

    return pokeData.data;
  })
);
```