[Práctica Módulo 9] Llamadas asíncronas

Enunciado

Para la práctica de este módulo, vamos a modificar el listado de usuarios del módulo anterior (*Trabajando con imports*) para mostrar en un documento HTML el listado de miembros de una organización de *Github*.

Partiremos del CodeSandbox bootcamp-lemoncode-mod8-practica-4 - CodeSandbox, haremos un *fork* para modificarlo.

Cambiar origen de datos

Como vimos en la práctica anterior, nuestro origen de datos era la función *getUsers* que devolvía datos falsos que construimos a partir de mocks. Vamos a cambiar el contenido de esta función para recuperar los datos reales de la API de *Github* desde la siguiente URL: "https://api.github.com/orgs/lemoncode/members"

Para hacer la petición, utilizaremos *fetch* o *axios*:

```
fetch("https://api.github.com/orgs/lemoncode/members").then(response =>
  response.json()
);
```

Esto devuelve una *Promesa* con los datos que sean devueltos por el servidor. Por tanto la función *getUsers* quedaría:

```
function getUsers() {
  return
fetch("https://api.github.com/orgs/lemoncode/members").then(response =>
    response.json()
  );
}
```

Hacemos incapié en lo que devuelve esta función ahora: una *Promesa* con los datos que se reciben del servidor, pero sigue siendo una *Promesa*.

Cómo utilizar ahora *getUsers*

Donde antes utilizábamos *getUsers* (en *index.js*) recibiendo de forma **síncrona** los datos mocks, ahora tendremos que actualizar el código para adaptarlo a recibir estos datos reales de forma **asíncrona**.

En *index.js* teníamos:

```
import * as UserBusiness from "./user-business";
import * as DataBusiness from "./data-business";
```

```
const users = DataBusiness.getUsers();
const nodes = [];

for (let user of users) {
  const node = UserBusiness.getUserNode(user);
  nodes.push(node);
}

for (let node of nodes) {
  document.getElementById("root").append(node);
}
```

Fijémonos en la línea en la que se obtienen los usuarios de forma síncrona:

```
const users = DataBusiness.getUsers();
```

Ahora, esta función *DataBusiness.getUsers()* no devuelve los datos inmediatamente, sino que devuelve una *Promesa* con los datos de forma *asíncrona*, por tanto:

```
DataBusiness.getUsers().then(datos => {
  const users = datos;
  const nodes = [];

  for (let user of users) {
    const node = UserBusiness.getUserNode(user);
    nodes.push(node);
  }

  for (let node of nodes) {
    document.getElementById("root").append(node);
  }
});
```

Como estamos asignando los datos a la constante *users*, podemos directamente nombrarlos así:

```
DataBusiness.getUsers().then(users => {
  const nodes = [];

  for (let user of users) {
    const node = UserBusiness.getUserNode(user);
    nodes.push(node);
  }

  for (let node of nodes) {
    document.getElementById("root").append(node);
  }
});
```

Cambiar el modelo de datos

Para dejar terminado el ejercicio con nuestra aplicación funcionando, deberemos cambiar aquellos puntos donde se utilizaba el modelo de datos *usuario* de la práctica anterior, por el nuevo modelo de datos que nos devuelve la API. Estos puntos son:

- getAvatar

```
function getAvatar(user) {
  const img = document.createElement("img");
  img.width = 150;
  img.src = user.avatar_url;

  return img;
}
```

- getFullName

```
function getFullName(user) {
  return user.login;
}
```

- getUserNode

```
function getUserNode(user) {
  const node = document.createElement("div");
  node.appendChild(AvatarBusiness.getAvatar(user));
  node.append(getFullName(user));

  return node;
}
```