React intermedio

Comunicación con el servidor

Creación de servicios dedicados en React

OpenWebinars

ÍNDICE

- ¿A qué llamamos servicio en el front-end?
- Modelos de entidades
- Definiendo nuestros servicios

¿A qué llamamos servicio en el front-end?

Servicio Back-end → Servicio front-end → Componente

- Proporciona acceso seguro a la información del servidor.
- La información viaja en un formato estructurado (JSON, XML...)

- Punto de acceso único con los servicios backend
- Procesa respuesta y transforma parámetros para realizar la petición
- Devuelve modelos o listas de modelos

- Gestiona las interacciones y llama a los servicios front
- Almacena el estado o lo consume
- Renderiza a partir de los modelos de respuesta

Modelos de entidades

- Llamamos modelo a una clase que representa una entidad concreta de nuestra aplicación (Artículos, productos, usuarios…)
- Los modelos de Front-end **NO** tienen porqué coincidir de manera exacta con los modelos devueltos o almacenados en el Back-end
- El modelo es una forma rápida de procesar una serie de datos y permite añadir funcionalidades extras con rapidez
- Un modelo de Front-end debe de almacenar todas las propiedades relativas a la entidad que sean necesarias para su renderización o procesamiento habitual.

Modelos de entidades

```
export default class User {
  constructor(userDTO) {
    this.id = userDTO.id;
    this.mail = userDTO.user_mail || null;
    this.verified = userDTO.is_verified === 1 ? true : false;
    this.name = userDTO.user_name || '';
    this.image = userDTO.profile_image ? `http://myurl/${userDTO.profile_image}` :
                 `http://myurl/placeholder.png`;
    this.permission = userDTO.user_permission || null;
  }
  toDTO() {
    return {
      id: this.id,
      user_mail: this.mail,
      is_verified: this.verified ? 1 : 0,
      user_name: this.name,
      profile_image: this.image == `http://myurl/placeholder.png` ? null :
                     this.image.replace('http://myurl/'),
      user_permission: this.permission
    }
  }
}
```

Definiendo nuestros servicios

- Podemos crear en primer lugar servicios genéricos, ajustados a nuestras necesidades, que nos ayuden a simplificar el uso de Fetch.

```
import { API_URL } from '../constants';

export function get(endpoint, params, options={}) {
  // Formamos la url de la petición con parámetros introducidos en un diccionario
  const url = new URL(API_URL + endpoint);
  Object.keys(params).forEach(key => url.searchParams.append(key, params[key]))
  fetch(url, {
    method: 'GET',
    headers: { 'content-type': 'application/json' },
    ...options
  }).then(response => response.json());
}
```

Definiendo nuestros servicios

- Y en el caso de una petición **POST** también podemos simplificar el proceso

```
import { API_URL } from '../constants';

export function post(endpoint, body, options={}) {
  // De esta forma podemos recibir siempre el body como un diccionario
  const url = new URL(API_URL + endpoint);
  fetch(url, {
    method: 'POST',
    headers: { 'content-type': 'application/json' },
    body: JSON.stringify(body),
    ...options
  }).then(response => response.json());
}
```

ES IMPORTANTE INVERTIR TIEMPO EN NUESTROS SERVICIOS BASE

Ellos serán el punto de comunicación con el servidor y parte fundamental en el correcto funcionamiento de nuestra aplicación.

PARA RESUMIR

- Un modelo de front es una **representación abstracta de la entidad de backend** con una orientación a la renderización del contenido
- Un servicio front-end **consume información** de un backend y simplifica el proceso de consumo del servidor