Summer School 2025

Frontend

Sergio Morales

0xDecode

Deusto Electronic Club Of
Developers & Engineers

Sergio Morales

Soy técnico superior en desarrollo de aplicaciones web con más de un año de experiencia en LKS Next.

Actualmente estoy cursando 2º del Grado en Ingeniería informática de la Universidad de Deusto.

Me gusta programar y los macarrones.

Y... prefiero windows a Linux (es una secta)

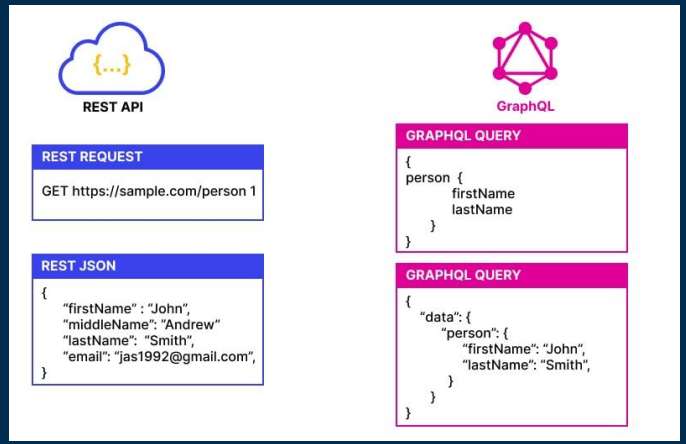
Puedes encontrarme en:
Web: https://sergiomorales.dev
Linkedin: https://es.linkedin.com/in/sergiomoralescobo
Github: https://github.com/sergitxin22

¿Qué es el frontend?

- Es la parte visual de una aplicación web (lo que el usuario ve y con lo que interactúa)
- Tecnologías principales: HTML, CSS, JavaScript
- Ejecuta en el navegador del usuario
- Se comunica con el backend para obtener/mandar datos
- Botones, formularios, diseño...

¿Cómo se comunica el frontend con una API?

- **Se usan peticiones HTTP: GET, POST, PUT, DELETE**
- **Frontend envía datos y recibe respuestas en formato JSON**
- **APIs REST y GraphQL son las más comunes**



¿Cómo hacer las peticiones?

- **fetch(): moderno, nativo en JS**

```
fetch('https://example.com/api/data')
  .then((response) => response.json())
  .then((data) => console.log(data));
```

- **axios: librería más completa y fácil de usar**

```
axios.get('http://localhost:4000/todos/')
    .then(res => {
        console.log("AXIOS RES", res)
        this.setState({ todos: res.data })
    })
    .catch(function (error) {
        console.log(error)
    })
```

- **XMLHttpRequest: método antiguo (en desuso)**

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.status === 200) {
       console.log(xhttp.responseText);
    }
};
xhttp.open("GET", "URL");
xhttp.send();
```

Configuración de peticiones HTTP

- **Parámetros (query params):** se agregan en la URL. Usados en GET para filtros, búsqueda, etc.

```
?clave=valor&otraClave=valor2
```

- **Cabeceras (headers):** se usan para autorización, tipo de contenido, etc.

```
const headers = {
  'Content-Type': 'application/json',
  'Authorization': 'Bearer <token>'
};
```

- **Body: se usa para enviar datos en POST/PUT (generalmente en JSON)**

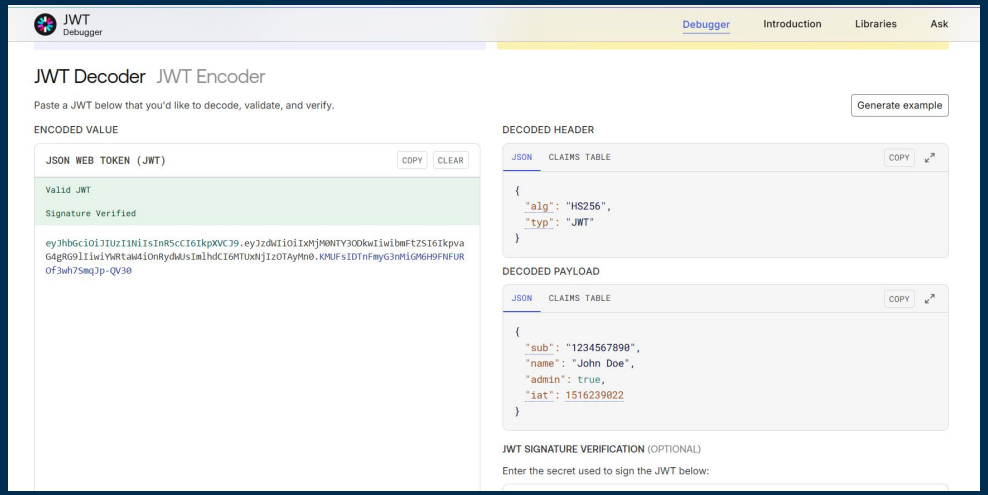
```
body: JSON.stringify({ nombre: 'Producto' })
```

fetch() básico en JS

```
fetch(url, {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ clave: 'valor' })
})
```

Autenticación y uso de tokens (JWT)

1. El backend genera un token tras login
2. El frontend guarda el token y lo envía en cada petición
3. Se manda en header: Authorization: Bearer <token>



Dónde guardar los tokens

- **localStorage:** persiste al cerrar navegador (menos seguro)
- **sessionStorage:** se borra al cerrar pestaña
- **Cookies (HttpOnly):** más seguras pero menos control desde JS

Riesgos al manejar tokens

Cuidado con exponer tokens

- No guardar tokens sensibles en lugares accesibles por JS
- Vulnerabilidad a XSS si el token se guarda en localStorage
- Usa HttpOnly si es posible

Seguridad básica en el frontend

- **XSS:** inyección de código malicioso (proteger inputs)
- **CSRF:** uso indebido de una sesión activa (proteger con tokens CSRF)
- **Validación de datos en frontend y backend**

Seguridad compartida

Seguridad: Frontend + Backend

- **Frontend:** UX, validación rápida, evita errores comunes
- **Backend:** control de acceso, autenticación, validación real
- **La seguridad nunca depende solo del frontend**

Frameworks de frontend

- **React:** componentes reutilizables, ecosistema enorme
- **Angular:** completo y robusto, ideal para apps grandes con TypeScript
- **Vue:** simple y progresivo, fácil de aprender
- **Svelte:** compila a JS puro, rendimiento excelente

Meta-frameworks

Algunos de los “Meta-frameworks” más conocidos son: Next.js (React), Nuxt (Vue), Remix, Astro

- **Se utilizan para:**
  - **Blogs, e-commerce, dashboards...**
  - **SSR (server-side rendering) o SSG (static site generation)**

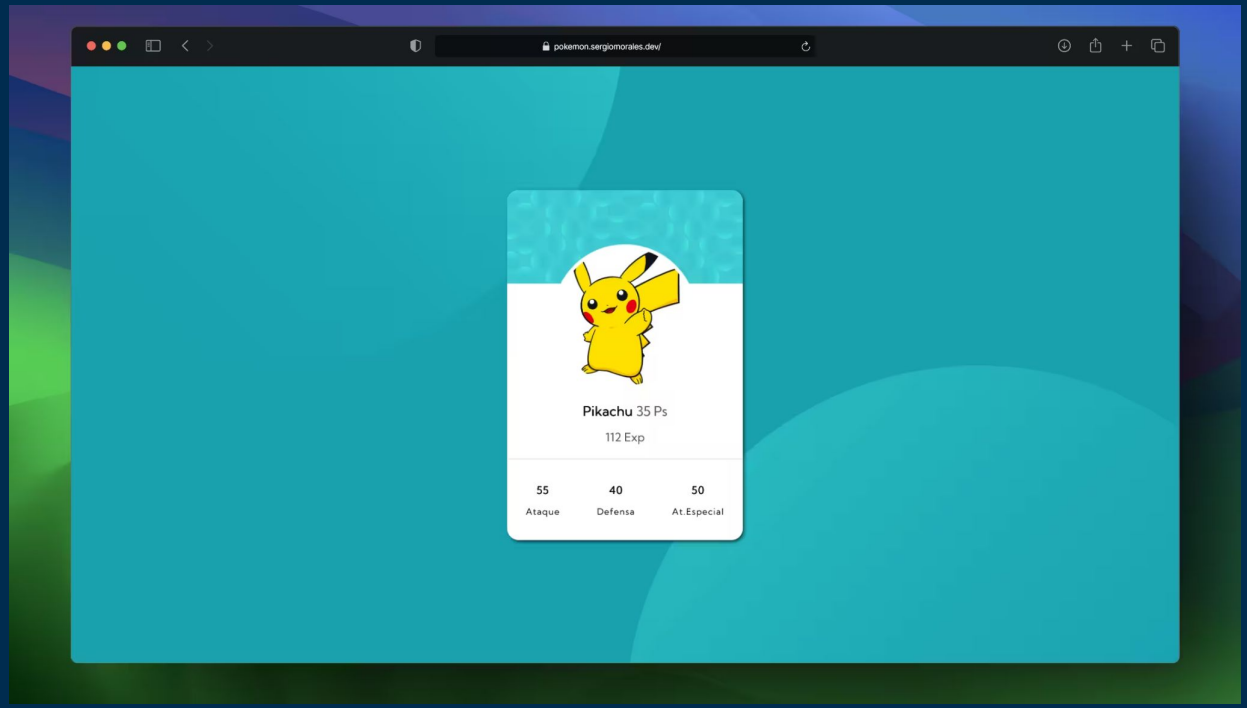
SPA vs MPA

- **SPA (Single Page Application):**
  - Una sola página HTML que se carga una vez.
  - El contenido cambia dinámicamente con JS sin tener que recargar la página.
  - Rápida, fluida. Ej: Gmail, Trello.
- **MPA (Multi Page Application):**
  - Cada click carga una nueva página del servidor.
  - Mejor para SEO (Search Engine Optimization). Ej: Amazon, periódicos.
- *SEO: Es el conjunto de técnicas para que una web aparezca lo más arriba posible en los resultados de Google, Bing, etc.*

Sitios estáticos y plantillas

- **Herramientas:** Hugo, Jekyll, Hexo
- **Ideales para sitios personales y blogs técnicos**
- **Muy rápidos, fáciles de desplegar con GitHub Pages**

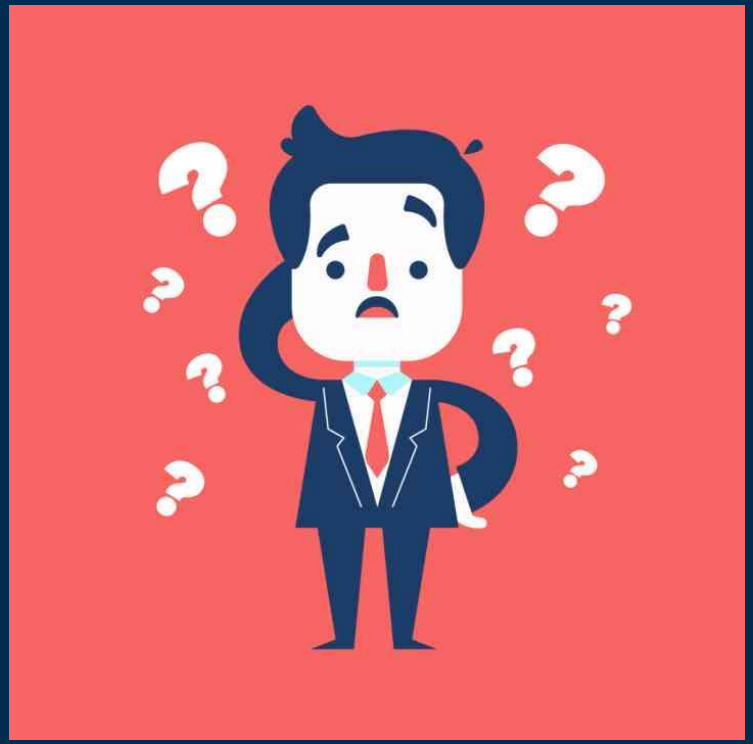
Nuestra primera Web (PokeAPI)



Optimizar nuestra web

- **Optimizar imágenes**
  - Optimizar imágenes “normales”: Recurso
  - Optimizar imágenes vectoriales (svg): Recurso
- **Fuentes locales**
  - Utilizar formatos woff2, woff, ttl: Recurso

Preguntas/aportaciones



¿Conectamos? (LinkedIn)

Evalúame

Sigue a 0xdecode

Discord

LinkedIn

Instagram

Summer School 2025

Muchas gracias

Descubre comunidad, descubre 0xDecode

0xDecode

Deusto Electronic Club Of
Developers & Engineers