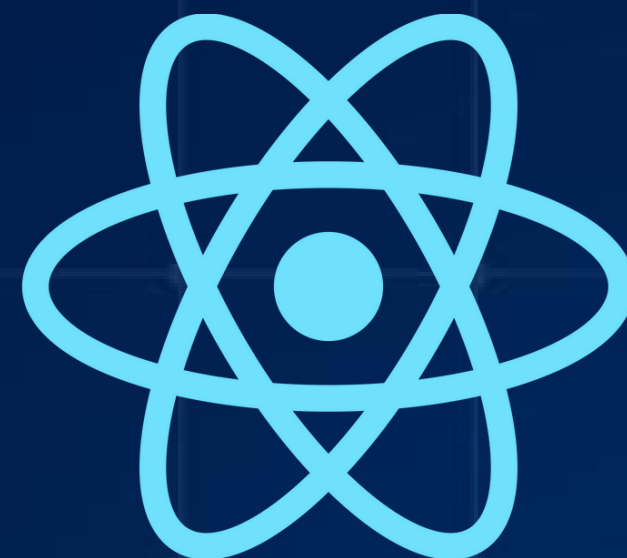


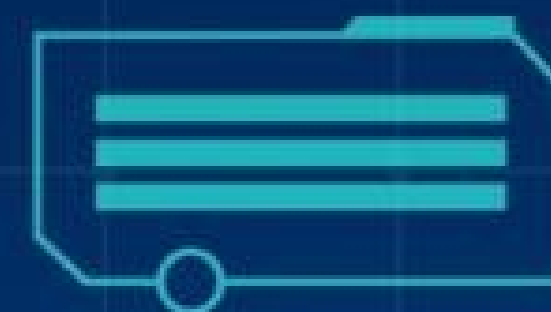
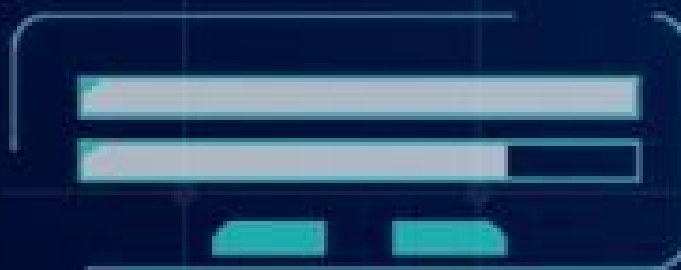


Escuela
Nacional de
Estudios
Superiores

IECAGto
Instituto Estatal de Capacitación



DESARROLLO WEB CON REACT



Temario

Fundamentos de desarrollo web

Javascript

Aplicaciones con React

Hooks y navegabilidad

Consumo de Web APIs

Herramientas en la nube

Sitios estáticos y SSR

Aplicaciones en tiempo real



Sitios estáticos y SSR

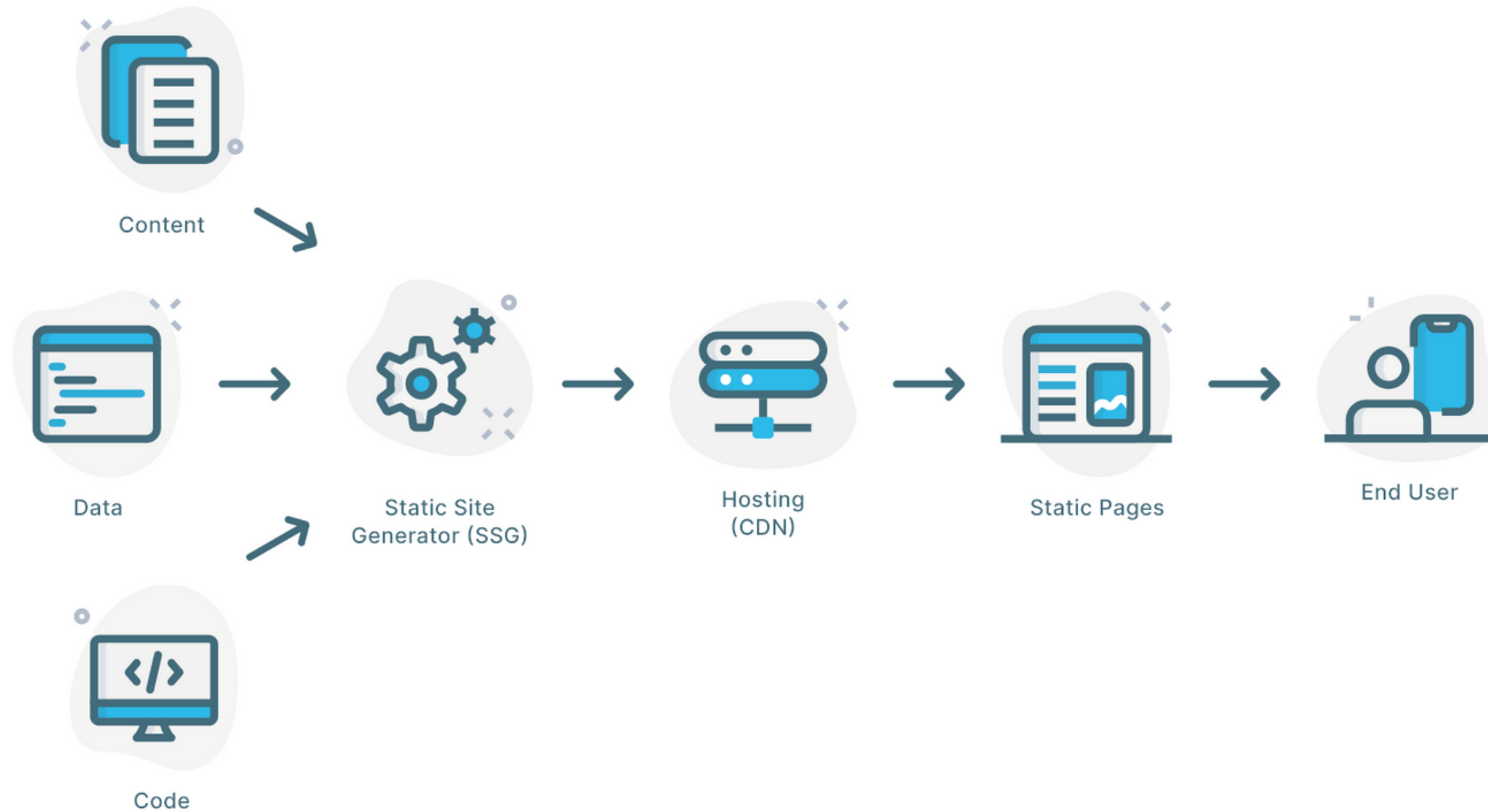
¿Qué es un sitio generado estáticamente (SSG)?

La generación de sitios estáticos (SSG, Static Site Generation) es el proceso de compilar y renderizar un sitio web o aplicación en tiempo de compilación. El resultado de la compilación es una colección de archivos estáticos, incluyendo el HTML, archivos CSS y JS, así como otros recursos como imágenes.

Es una técnica muy popular utilizada en la actualidad mediante diferentes frameworks en el mercado.



¿Cómo funciona un sitio generado estáticamente (SSG)?



Ventajas de un sitio generado estáticamente (SSG)

SEO:

Permite la creación de páginas fácilmente indexables para un mejor tráfico orgánico, el contenido generado puede posicionarse fácilmente en motores de búsqueda.

Mejoras de rendimiento:

Se reducen los tiempos de carga al generar parte de su contenido en tiempo de compilación.

Seguridad:

Son más seguros porque no ejecutan código del lado del servidor ni permiten interacciones complejas en el lado del cliente. Al evitar la ejecución de scripts y bases de datos, se reduce la superficie de ataque potencial para posibles vulnerabilidades.

Next.js

Next.js es un framework de React de código abierto creado por Vercel que facilita el desarrollo de aplicaciones web y sitios estáticos con React.

Combina el uso de React con herramientas adicionales para lograr creación de sitios con mejor rendimiento y características full-stack.

Next.js admite las técnicas de desarrollo Static Site Generation (**SSG**) y Server-Side Rendering (**SSR**).

NEXT.js



Creación de un sitio con Next.js

Al igual que **Create React App** y otros frameworks de desarrollo en React, la creación de un proyecto de Next se puede realizar utilizando un solo comando:

```
npx create-next-app@latest
```


Creación de un sitio con Next.js

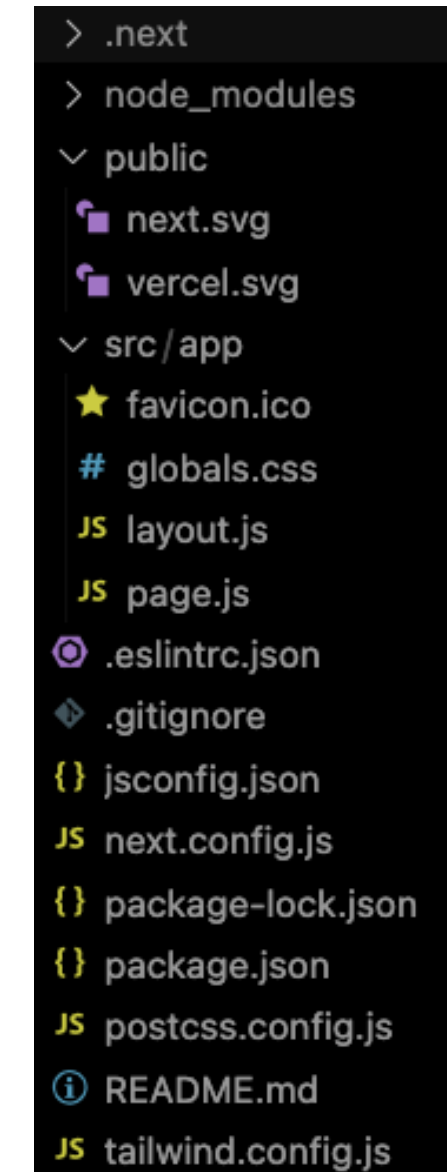
Al crear el proyecto, se solicitarán algunas configuraciones como el uso de TypeScript, la integración de TailwindCSS y el uso de App Router, entre otras. Estas configuraciones definen la estructura del proyecto para su modificación.

```
✓ What is your project named? ... github-tracker
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like to use `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to customize the default import alias? ... No / Yes
Creating a new Next.js app in /Users/uriel/Dev/Next/github-tracker.
```

Creación de un sitio con Next.js

Desplegado el proyecto se mostrará una estructura como la de la imagen con los siguientes elementos relevantes:

- **public:** Carpeta contenedora de los archivos públicos del proyecto (imágenes, fuentes, etc.).
- **src:** Carpeta que incluye todo el código fuente de React.
 - **app:** Contiene la estructura de rutas para la navegación en el sitio web y los componentes y layouts adecuados para desplegar en cada una.
- **next.config.json:** Archivo de configuración de Next.js



```
> .next
> node_modules
└─ public
   ├── next.svg
   └── vercel.svg
└─ src/app
   ├── ★ favicon.ico
   ├── # globals.css
   ├── JS layout.js
   ├── JS page.js
   ├── .eslintrc.json
   ├── .gitignore
   ├── {} jsconfig.json
   ├── JS next.config.js
   ├── {} package-lock.json
   ├── {} package.json
   ├── JS postcss.config.js
   ├── ⓘ README.md
   └── JS tailwind.config.js
```

Componentes de Servidor

Todos los componentes de React generados dentro de la carpeta **app** se consideran como **Componentes de servidor** debido a que pueden ejecutar código del lado del servidor para su renderizado.

```
1 import Image from "next/image";
2 import logo from "../../assets/logo.png";
3 import RepoList from "@components/Repolist";
4
5 export default async function ReposPage() {
6   const repos = await getRepos();
7
8   return (
9     <main className="flex min-h-screen items-center justify-center flex-col gap-4 p-24">
10       <Image src={logo} alt="GitHub" width={200} height={200} />
11       <h1>Mis repositorios</h1>
12       <RepoList repos={repos} />
13     </main>
14   );
15 }
16
17 async function getRepos() {
18   const repos = await fetch("https://api.github.com/users/ur13l/repos");
19   return await repos.json();
20 }
21
```

Elementos estructurales

Dentro de la carpeta app y sus subcarpetas pueden existir los siguientes elementos:

page.js: Contiene toda la lógica de la página a renderizar.

loading.js: Permite mostrar contenido en el primer renderizado previo a la carga del contenido.

layout.js: Funciona como elemento de estructura para compartir contenido visual entre pantallas.



Escuela
Nacional de
Estudios
Superiores

IECAGto[®]
Instituto Estatal de Capacitación

¡GRACIAS!

