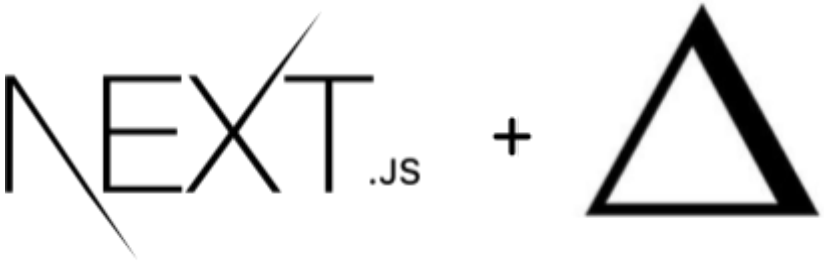
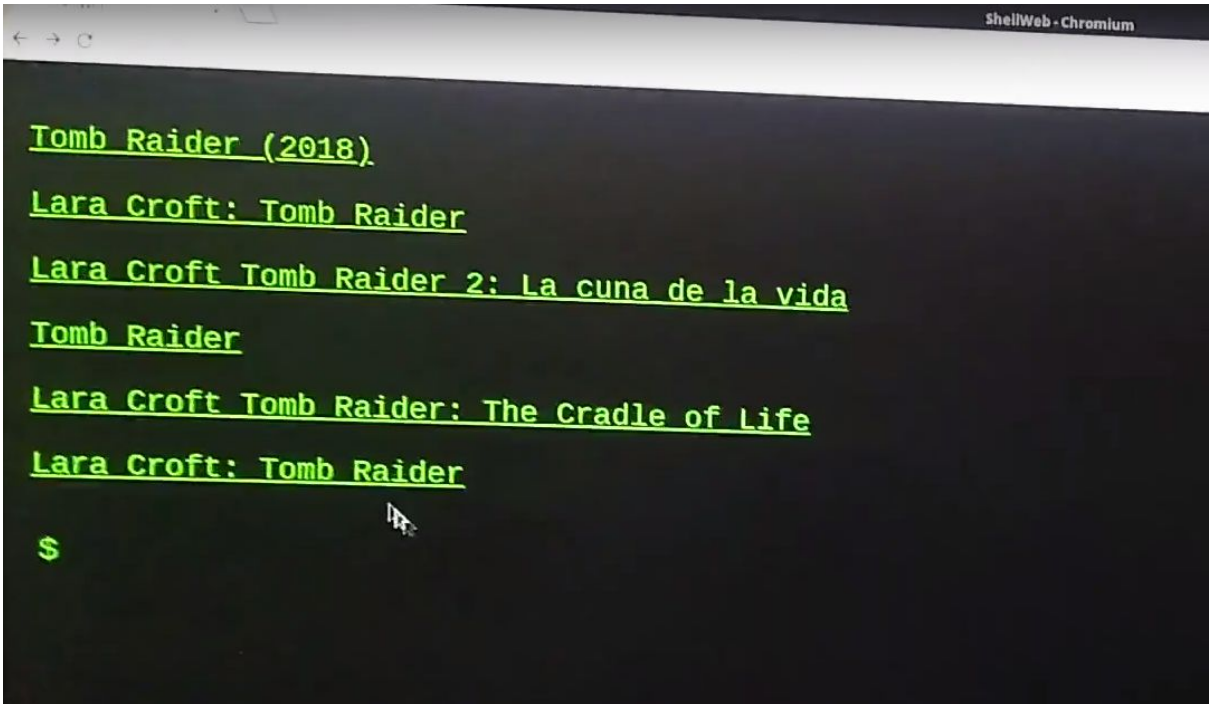

NEXT.JS
+

# ÍNDICE

# Introducción

La realización de este proyecto ha sido posible gracias al conocimiento aportado en el curso de desarrollo de aplicaciones web, cada línea de código ha tenido borradores, pruebas y demasiadas horas de investigación previas al resultado obtenido. El proceso ha sido emocionante y muy gratificante.

El proyecto fue evolucionando según fui aprendiendo a lo largo de mi formación en las diferentes disciplinas del desarrollo web. La idea se fue construyendo poco a poco, al principio no tenía ni interfaz, era una línea de comando simulada en el navegador, allá por julio del 2018.




Este prototipo conseguía hacer scraping con un navegador automatizado, utilizando la librería puppeteer en pocos segundos, eso ya era un avance.

Durante todo el verano fui aprendiendo sobre el concepto de scrapear o extraer información a través de navegadores automatizados o por peticiones http, con librerías como request o cheerio. Scrapear la web ha sido algo muy emocionante para mí, que me ha llamado mucho la atención en mi etapa de aprendizaje, quizás por el sentimiento de que tenía internet a mis pies y un camino largo por recorrer en mi formación.

Al llegar a segundo, aplique todos los conocimientos adquiridos sobre scraping en proyectos desarrollados en PHP con librerías especializadas en la extracción de datos por selectores y el proyecto de las camisetas el cual hacía scraping a google para conseguir imágenes según el término por el que buscaba el usuario.

Es entonces cuando fui forjando los pilares y retomando aquellos prototipos que tenía para retomarlo como proyecto personal.

Cuando aterrice en las prácticas, la plataforma que estábamos desarrollando se apoyaba en **Next** para solucionar aspectos como SEO, enrutamiento y otros aspectos que escapaban a mi entender, pero poco a poco fui viendo otra vertiente de este framework, que encajaba perfectamente con la filosofía serverless, *no te preocupes por el despliegue, te lo damos todo hecho para que tú solo desarrolles...*

Hasta tal punto que con un comando puedes, levantar un servidor en local para desarrollar, exportar los estáticos, pero la misma gente que desarrolla Next te ofrece un servicio cloud para que no te preocupes del mantenimiento o configuración del servidor y una vez pusheado a tu repo o ejecutando el comando para subir y desplegar ya te está generando los bundles automáticamente y puedes ver su pipeline mostrándote si falló o salió bien la build.

## Explicación

## ¿Cómo funciona?

En sí, la funcionalidad de esta aplicación consiste en la búsqueda y extracción de enlaces a servidores de streaming, que usuarios llamados uploaders comparten con infinidad de comunidades para que sus usuarios puedan visualizar el contenido audiovisual, en varias calidades, idiomas y/o subtítulos.

El funcionamiento es simple; el usuario introduce un término en el input de búsqueda y en cuanto comienza a introducir caracteres hace una petición a TheMovieDB una API externa que administra una base de datos de películas y series inmensa, de ahí obtiene respuesta con el resultado de todas las coincidencias obtenidas. Los resultados obviamente pueden ser o películas o series y dependiendo de una otra

irá mostrando información diferente al momento de seleccionar un resultado, como por ejemplo la sinopsis, temporadas, episodios y por último la lista de enlaces a servidores de streaming y un reproductor para ver el contenido directamente.

Una vez llevado a producción, la aplicación funciona en serverless, apoyándose en las function lambdas de now, sus diferentes funciones son apis montadas con express que recepciona mediante query strings los parámetros necesarios, al hacerse la petición se origina la invocación de la función y esta responde retornando el resultado.

**Functions lambdas:**

estas son las diferentes funciones que demanda el cliente:

- /search , /search-seasons y /search-episode

```
./search/index.js ./search-seasons/index.js ./search-episodes/index.js
```

Estas funciones se nutren de la API TheMovieDB, las cual hace consulta mediante el término marcado y retornando la información de los resultados.

- /adede

```
./adede/index.js
```

La función que hace scraping a otra página y comprueba su disponibilidad, mediante la librería *request*, pide el cuerpo de la página inyectando las cookies en la solicitud, permitiendo el acceso y capturando los enlaces. Una vez obtenidos son comprobados con *cheerio* en un bucle, si esta disponible o ha caído.

- /msg4admin

```
./msg4admin/index.js
```

Con esta función, la aplicación se sirve de puente para comunicar mediante mensajes de chat a un bot de Telegram, utilizando su API; es utilizada cuando un usuario busca algo, se envia el ID al bot ó cuando se utiliza el textarea del apartado de desarrollo about.js para enviar un mensaje al bot.

# ¿Qué es Next?

Podría decirse que es una extensión de React, la cual aporta características que dan facilidad a los desarrolladores web a la hora de llevar a producción un proyecto, además de aportar mecanismos que ayudan al SEO y tiene una perfecta integración en un entorno serverless u corriendo en un servidor y sirviendo por SSR (server side rendering)

Next proclama su zero configuración y no le falta razón, pues en muy poco tiempo puedes tener una página web o un proyecto más amplio desplegada y en producción. Pues te lo da ya casi todo y si falta algo seguro que puedes extenderlo, como por ejemplo integrar Sass o typescript o comprimir imágenes, pues Next lleva webpack como dependencia y puede ser configurado. Su estructura de archivos tiene un sentido organizativo y funcional, a continuación se explica:

---

- **pages:** En esta carpeta irán todas las rutas que demandan páginas, las subcarpetas indican subrutas, el nombre que tengan será el nombre de la ruta, Next permite cambiar el nombre desde el código.
- **static o public:** ambas carpetas tienen el mismo cometido, servir archivos estáticos, como imágenes, vídeos, estilos, fuentes, etc.
- **.next:** Cuando se ejecuta next, creara esta carpeta donde esta todos los bundles, chunks tanto de css como de javascript.

---

Continuando con una estructura que llevará un orden, decidí mantener estas carpetas:

---

- **components:** En ella guardo todos los componentes de React, organizados por el nombre de la página y en la raíz de esta, el layout y el header.
- **styles:** Aqui guardo los archivos Scss que serán transpilados a css por webpack.
- **utils:** todas la funciones que son útiles para la parte del frontend.
- **faas:** en esta carpeta he almacenado todas las funciones lambda que serán desplegadas para la parte del backend.

---

# ¿Y eso de FaaS?

Es parte de lo que comúnmente se llama serverless, es otro servicio cloud que intenta resolver un problema como por ejemplo Firebase intenta solucionar el tener una base de datos sin tener que mantener y despreocupandose de mantenimiento, configuraciones u otros como servicios de notificaciones push. Pues FaaS pretende satisfacer que el código que estaría corriendo en un servidor para backend tenerlo disponible en cloud, con ventajas muy interesantes, aun así tiene desventajas.

Funciona de una manera muy particular, cuando el cliente hace la petición a una ruta, este esta haciendo una invocación de una función previamente desplegada lista para montarse en un contenedor tipo docker, con un entorno de desarrollo montado, como podría ser node.js o PHP, e incluso bash.

El contenedor funciona de una forma fugaz, se lanzará, ejecutará la función que tenga cargada retornando el resultado y finalmente acabará su proceso.

A simple vista se podrían ver ventajas en cuanto al aprovechamiento de recursos de una máquina, pues el procesador solo es utilizado cuando se necesita y no de una manera lineal en el tiempo. Por lo tanto solo pagarás por el uso que le das. Entonces si el servidor cae cada vez que la función termina quiere decir que no tiene estado, no recordará nada de lo que otra función haya realizado. Pues si, esto es cierto, pero existen maneras para apoyarte en otro servicio de persistencia externo en el cual salvar el estado de cada función.

Otra de las desventajas visibles, son sus tiempos de ejecución que están limitados e incluso a veces penalizados por algunos proveedores, además existe una pequeña latencia la primera vez que se ejecuta una función o si lleva mucho sin ejecutarse.

# Conclusión

La razón principal por la que decidí usar este stack es que está preparado y perfectamente integrado para lanzarlo y utilizarlo en serverless con ayuda de now y el servicio que ofrece **Zeit.** Otro de los motivos por los que decidí usar FaaS fue al tratarse de un proyecto personal que iba a utilizar de vez en cuando y quizás sea este uno de los motivos por lo que es ideal una tecnología así. Si eres desarrollador y te interesa desarrollar prototipos de una aplicación o tienes un proyecto personal que quieres ir llevando a cabo, es muy sencillo comenzar y ya te preocuparás de migrarlo a un servidor convencional. Esa es la filosofía que reza: *solo desarrolla.*

Lo mismo pasa con Next, parece evidente que sean los mismos que fundaron Zeit. estos chicos se preocupan y dan muchas facilidades a los desarrolladores que quieren sin más empezar con un proyecto sin tener que sufrir mucho y enfrentarse a despliegues, mantenimiento, dolores de cabeza en cuanto a decisiones que no tienen porque ser tomadas en ese momento.

# Instalación

Para la instalación y despliegue son necesarias algunos tokens a APIs y el cli de now

## Requerimientos

### Instalando Now de zeit.co

sigue esta guía para la instalación de now: https://zeit.co/download

o simplemente, ejecuta:

```
curl -sfLS https://zeit.co/download.sh | sh
```

## APIs necesarias:

- TMDB
- Telegram

## Dependencias utilizadas:

- antd
- request
- cheerio
- express
- next
- node-sass
- @zeit/next-sass
- react
- react-dom
- cors

## Proceso de instalación

Descarga el repo:

```
git clone https://github.com/pangeasi/scraperDede.git
```

Crea un archivo `config.js` y editalo con tus claves:

```
module.exports = {

    cookie: [

        {name: 'PHPSESSID',

        value: '',

        domain: '',

        path: '/',

        expires: 1605376742.173697,

        size: 35,

        httpOnly: true,

        secure: false,

        session: false }

        ],

        TELEGRAM_TOKEN: 'YOUR_KEY',

        MASTODON_TOKEN: 'YOUR_KEY',

        THEMOVIEDB_API_TOKEN: 'YOUR_KEY',

        LOGIN_VERYSTREAM_API: 'YOUR_KEY',

        KEY_VERYSTREAM_API: 'YOUR_KEY'

}
```

Instala dependencias:

```
npm install
```

o

```
yarn install
```

Ejecuta estos dos scripts:

para levantar next en modo desarrollo en el puerto 3000

```
yarn now-dev 3000
```

y para el arrancar el servidor en local imitando FaaS

```
yarn start
```

## Despliegue

Para desplegar en now, puedes hacerlo directamente a producción o con tu rama a fase de pruebas

```
now
```

para el despliegue a producción, recuerda cambiar el alias en now.json

```
now --target production
```

## APIs

- TMDB
- Telegram

## Presentación

Presentación aportada del proyecto

https://docs.google.com/presentation/d/1EXmQpOzSnqw4zFiHwJApAxHuK7HjXiqyv6ko8wEfdi8/edit?usp=sharing

## Enlaces de interés

Aplicación del proyecto: https://scraperdede.now.sh
Repositorio del proyecto: https://github.com/pangeasi/scraperDede

## NextJS

- Web de Next: https://nextjs.org/
- github: https://github.com/zeit/next.js
- ejemplos prácticos: https://github.com/zeit/next.js/tree/canary/examples
- Documentación: https://nextjs.org/docs

## FaaS - Now

- Now: https://zeit.co/now
- Documentación de now: https://zeit.co/docs/v2/getting-started/introduction-to-now

## Proveedores de FaaS

- AWS Lambda: https://aws.amazon.com/lambda/
- Google cloud functions: https://cloud.google.com/functions/
- Azure functions: https://azure.microsoft.com/en-us/services/functions/
- Zeit: https://zeit.co/
- IBM Bluemix: https://console.ng.bluemix.net/openwhisk/

## Para montar tu propio FaaS

- Iron functions: http://open.iron.io/
- Openstack Picasso: https://wiki.openstack.org/wiki/Picasso
- Fission: http://fission.io/
- Apache Openwhisk: https://openwhisk.incubator.apache.org/
- OpenFaaS: https://www.openfaas.com/