WatchThis

**Índice**

[**Manual de Despliegue** 1](#_Toc63199862)

[ **MongoDB** 1](#_Toc63199863)

[ **GitHub** 5](#_Toc63199864)

[ **Python** 6](#_Toc63199865)

[ **Flask** 7](#_Toc63199866)

[ **NodeJS** 9](#_Toc63199867)

[ **Yarn** 10](#_Toc63199868)

[ **Despliegue de aplicaciones** 11](#_Toc63199869)

# **Manual de Despliegue**

Con el fin de poder desarrollar y ejecutar la aplicación durante sus diferentes fases y pruebas ha sido necesario establecer un entorno de ejecución lo suficientemente rápido y cómodo.

Para ello, ha sido necesario establecer un entorno común de desarrollo con tres tecnologías diferentes, las cuales se ejecutan y despliegan de forma independiente, pero trabajan de forma conjunta:

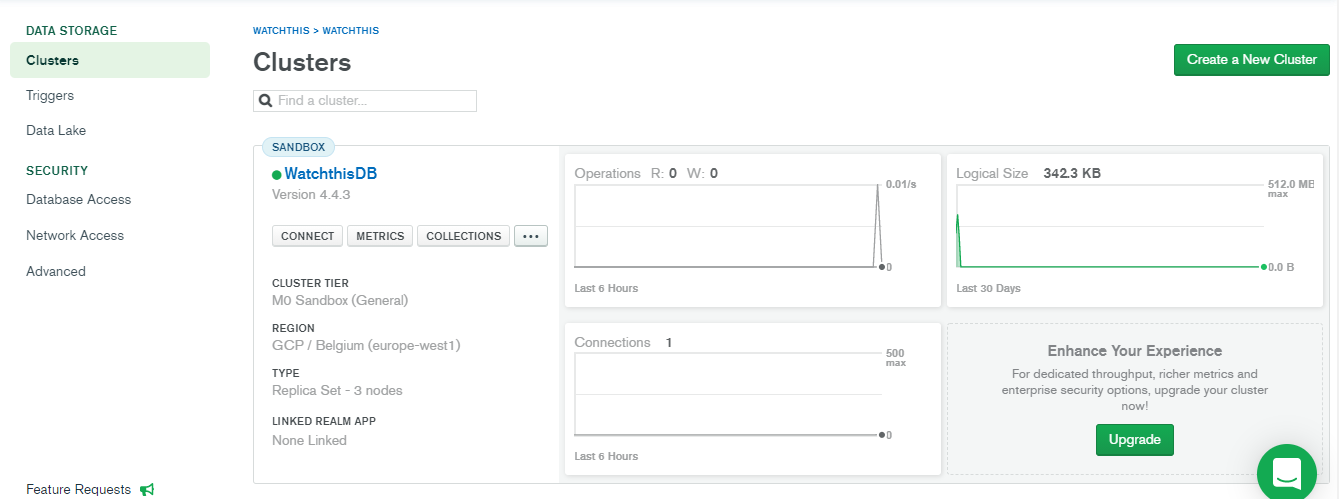
## **MongoDB**

MongoDB es un sistema de base de datos no relacional, es decir, NoSQL, orientado a documentos y de código abierto. Esto proporciona una flexibilidad a la hora de guardar datos, ya que no se almacenan en tablas tal y como se hace en tablas relacionales, si no que los guarda con un esquema dinámico haciendo que su integración sea mucho más fácil y rápida.

Para poder acceder al sistema de base de datos de MongoDB es necesario acceder a este [**enlace**](https://account.mongodb.com/account/login?signedOut=true)con las siguientes credenciales**:**

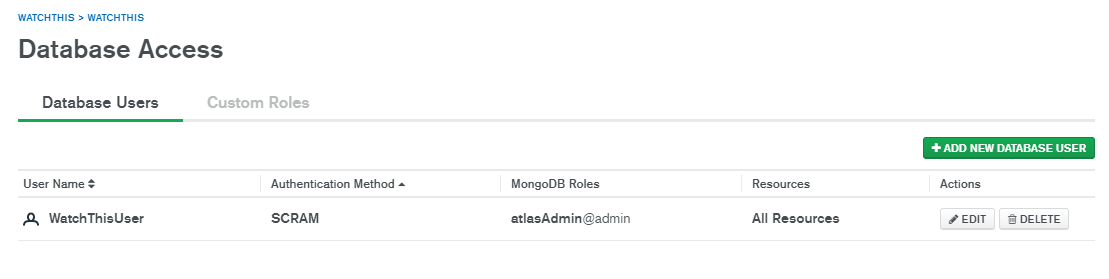
* **Email**: [watchthises@gmail.com](mailto:watchthises@gmail.com)
* **Contraseña:** patrones123

Una vez dentro, podremos ver el cluster activo que tenemos integrado dentro de nuestras aplicaciones llamado “WatchthisDB”, al cual podemos conectarnos, editar las métricas y sus colecciones, así como gestionar sus accesos y triggers:



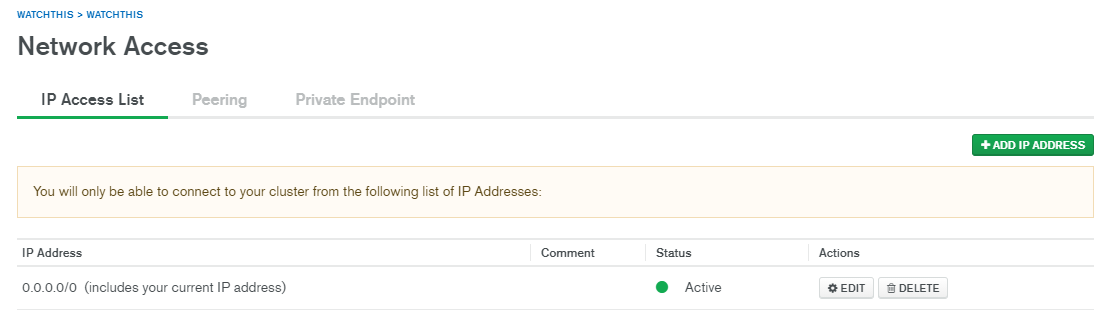
*Cuadro de mandos de clusters MongoDB*

Para gestionar los accesos a la base de datos se ha creado un usuario con el rol de administrador, el cual puede gestionar el acceso y el control de todos los recursos:



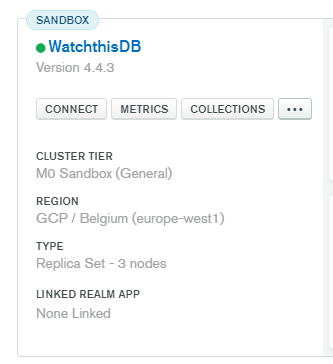
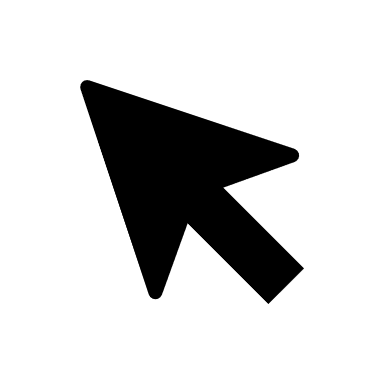
*Control de accesos clusters MongoDB*

Con el fin de poder suministrar un acceso global independientemente del usuario que se quiera conectar, se ha establecido una IP genérica global, donde se incluyen todas las IPs (0.0.0.0/0):



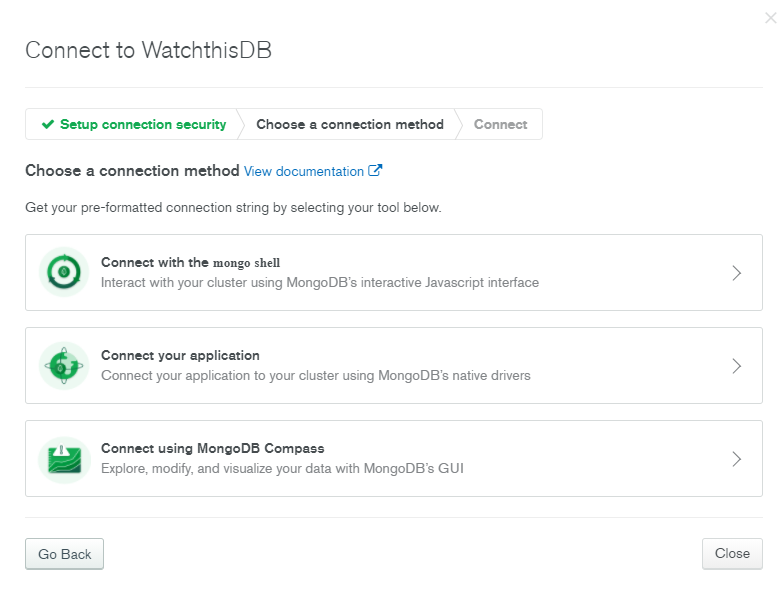
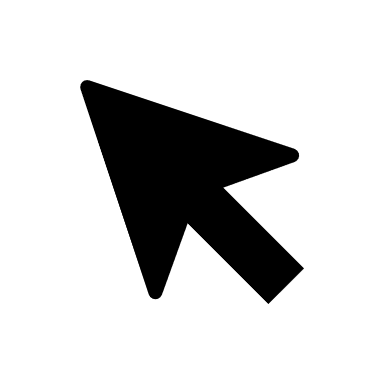
*Gestión de conexiones MongoDB*

Una vez configurado nuestro cluster, podremos establecer conexión a través de nuestra app creada en “Flask”, para ello accederemos al apartado de **CONNECT:**



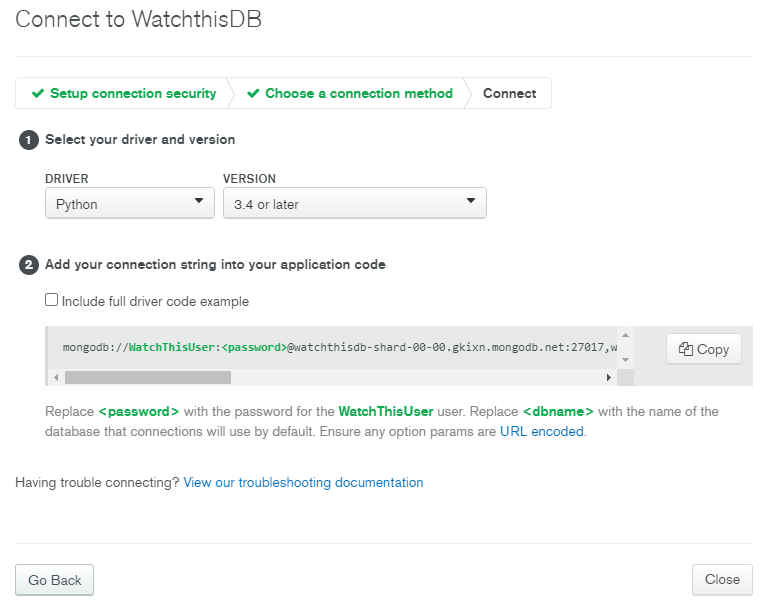
*Conexión API con MongoDB (Connect)*

A continuación, se nos proporcionará tres formas diferentes de conectarnos, en nuestro caso elegiremos la segunda ya que nos vamos a conectar a través de nuestra aplicación:



*Conexión API con MongoDB (Driver)*

Por último, tendremos que seleccionar el driver y la versión correspondientes de nuestra aplicación, en nuestro caso estamos utilizando Python con la versión 3.9:



*Conexión API con MongoDB (API)*

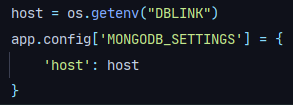
Una vez seleccionados, MongoDB nos proporcionará un enlace el cual tendremos que incluir en nuestra aplicación como parámetro de conexión, modificando los campos de **<password>** y **<dbname>** por los correspondientes:

* **dbname:** ‘WatchthisDB’
* **password:** ‘patrones123’

Teniendo el siguiente enlace como resultado:

***'mongodb://WatchThisUser:patrones123@watchthisdb-shard-00-00.gkixn.mongodb.net:27017,watchthisdb-shard-00-01.gkixn.mongodb.net:27017,watchthisdb-shard-00-02.gkixn.mongodb.net:27017/WatchthisDB?ssl=true&replicaSet=atlas-s035bc-shard-0&authSource=admin&retryWrites=true&w=majority'***

Por último, tenemos que incluir la configuración de nuestro cluster dentro de nuestra aplicación, al estar utilizando “Flask” este nos proporciona una librería específica para MongoDB (“flask\_mongoengine”) donde utilizaremos e implementaremos MongoEngine, incluyendo el enlace como parámetro de host:



*Conexión API con MongoDB (Configuración)*

Una vez configurado podremos empezar a customizar nuestros modelos y operaciones para poder gestionar nuestra base de datos.

## **GitHub**

Github es una plataforma la cual permite alojar proyectos utilizando un sistema de control de versiones, favoreciendo el desarrollo y el control de proyectos entre diferentes contribuyentes pudiendo trabajar con múltiples versiones tanto locales como remotas.

Para el desarrollo de nuestro proyecto hemos creado un repositorio llamado **WatchThis** el cual contiene todo el código de nuestras aplicaciones:

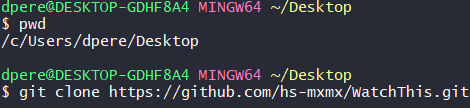
* [**hs-mxmx/WatchThis (github.com)**](https://github.com/hs-mxmx/WatchThis)

Para poder descargarnos el proyecto y ejecutarlo / modificarlo necesitaremos bajarnos la consola de Git:

* [**Git - Downloads (git-scm.com)**](https://git-scm.com/downloads)

Una vez descargada e instalada, podremos clonar y copiar el proyecto en nuestra máquina local, para ello accederemos desde la consola de Git y ejecutaremos el siguiente comando desde la ruta en la que queramos almacenar el proyecto:

En este caso, se instalará en la ruta ‘/c/Users/dpere/Desktop’:



*Descarga del repositorio Git*

Para verificar que se nos ha descargado y clonado sin ningún problema tendremos que acceder a la carpeta del proyecto, donde nos saldrá el nombre de ‘**main’** en azul clarito al lado de la ruta de nuestro archivo:

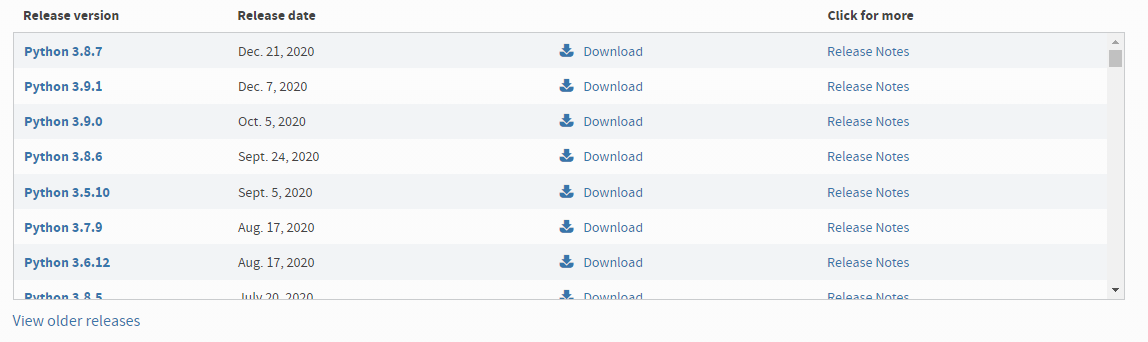


*Repositorio WatchThis Git*

## **Python**

Python es un lenguaje de programación interpretado multiparadigma el cual soporta parcialmente la orientación a objetos, programación imperativa, y en menor medida la programación funcional.

Para el desarrollo de nuestro proyecto hemos implementado una aplicación Flask en Python, siendo necesaria la descarga del driver correspondiente en el siguiente [**enlace**](https://www.python.org/downloads/)**:**



*Versiones Python*

Una vez descargado e instalado el driver seleccionado, podemos probar si se ha instalado todo correctamente ejecutando el siguiente comando en la consola:



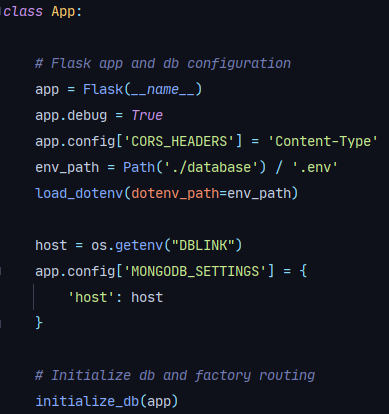
*Comprobación de versión Python*

Durante la instalación es importante marcar la casilla de instalar e incorporar los comandos Python dentro del PATH de Windows, con el fin de poder ejecutar los comandos desde cualquier parte de nuestro sistema.

## **Flask**

“Flask” es un framework de Python el cual permite crear aplicaciones web de forma muy rápida y sencilla, el cual incorpora unos motores de templates de Jinja2 junto con una licencia BSD.

Para el despliegue y la configuración de nuestra aplicación flask hemos utilizado los parámetros determinados, estableciendo una serie de parámetros:



*Configuración de la aplicación Flask*

Le estamos proporcionando una configuración con la cual podamos hacer debug y añadirle una cabecera para autentificar nuestro contenido, ya que vamos a realizar una comunicación bidireccional con nuestra aplicación de React y para ello necesitaremos añadir ‘CORS\_HEADERS’.

Una vez configurada nuestra aplicación, necesitaremos instalar las librerías y dependencias de nuestro proyecto, para ello será necesario movernos a la carpeta de nuestro controlador y ejecutar el siguiente comando:

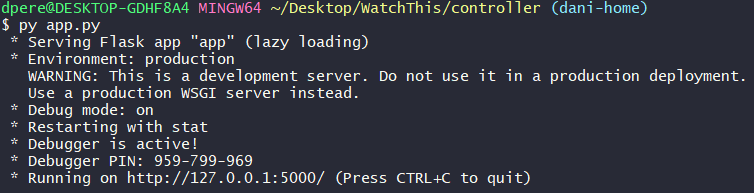
* *pip install -r requirements.txt*



*Dependencias de la aplicación Flask*

Nuestra aplicación ya está lista para ser ejecutada, para ello ejecutaremos el archivo “app.py” como cualquier archivo normal de Python:

* *py app.py*



*Ejecución de la aplicación Flask*

## **NodeJS**

NodeJS es un entorno en tiempo de ejecución multiplataforma de código abierto, el cual actúa para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos.

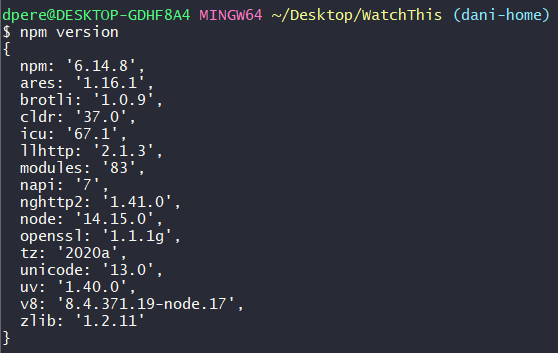
El servidor de la aplicación de FrontEnd se ha desplegado bajo el entorno de NodeJS, por lo tanto, es necesario descargarnos el driver correspondiente para poder instalar todas las dependencias del proyecto y poder ejecutarlo, para ello accederemos a este [**enlace**](https://nodejs.org/es/download/):



*Instalación de NodeJS*

La instalación se realizará tras la descarga del instalador del driver en el sistema operativo seleccionado, incluyendo las operaciones dentro de nuestro PATH de Windows al igual que hicimos con Python para evitar cualquier tipo de conflicto y permitir ejecutar las operaciones desde cualquier ruta en nuestro sistema.

Para comprobar que se ha instalado correctamente simplemente tendremos que ejecutar el comando de versionado al igual que hicimos con Python, devolviéndonos un resultado parecido al siguiente:



*Comprobación de versión NodeJS*

## **Yarn**

Yarn es un instalador de paquetes de JavaScript y gestor de dependencias, el cual introduce una gestión durante la ejecución de tareas y algunas mejoras de rendimiento, enfocado en la descarga e instalación de los paquetes, así como la gestión de las dependencias aportando una mayor fiabilidad.

Para poder lanzar nuestra aplicación React necesitaremos instalar Yarn en nuestro sistema, aprovechando que hemos instalado NodeJS podremos instalarlo de forma global ejecutándolo como un paquete npm:



*Instalación Yarn vía NodeJS*

Para comprobar que se ha instalado correctamente simplemente tendremos que ejecutar el comando de versionado al igual que hemos hecho con Python y con NodeJS, devolviéndonos un resultado parecido al siguiente:



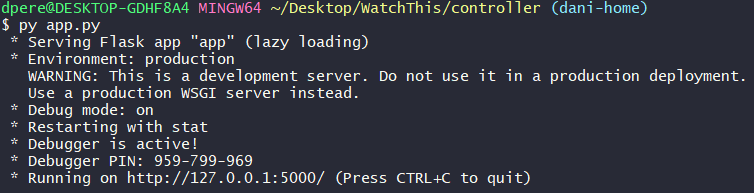
*Comprobación de versión Yarn*

## **Despliegue de aplicaciones**

Una vez que ya hemos instalado todas las tecnologías y drivers del proyecto, podremos ejecutar ambas aplicaciones (Python y React):

* **Ejecución Flask app:**

Tal y como habíamos comentado anteriormente, tendremos que desplazarnos a la carpeta “controller” de nuestro proyecto y ejecutar el comando “py app.py”

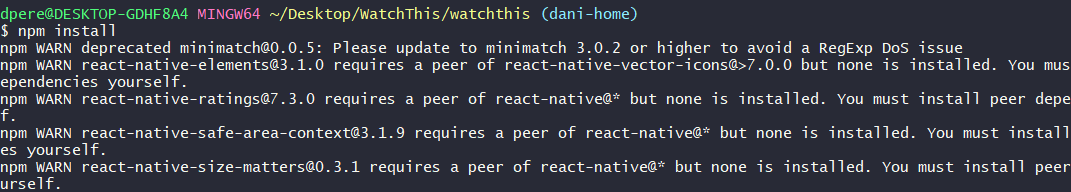


*Ejecución de la aplicación Flask*

* **Ejecución React app:**

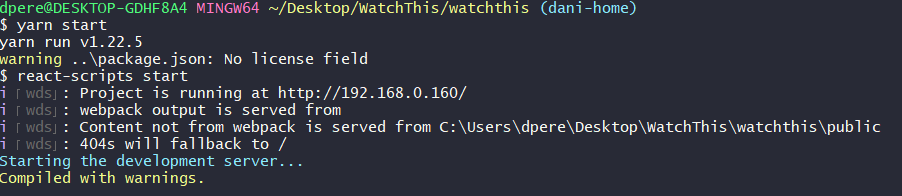
Para poder ejecutar nuestra aplicación React primero tendremos que instalar todas las dependencias del proyecto, para ello tenemos que ejecutar un instalado de los paquetes del proyecto.

Para ello, nos tendremos que desplazar a la carpeta raíz del proyecto y ejecutar un instalado de paquetes, obteniendo una salida de pantalla como la siguiente:



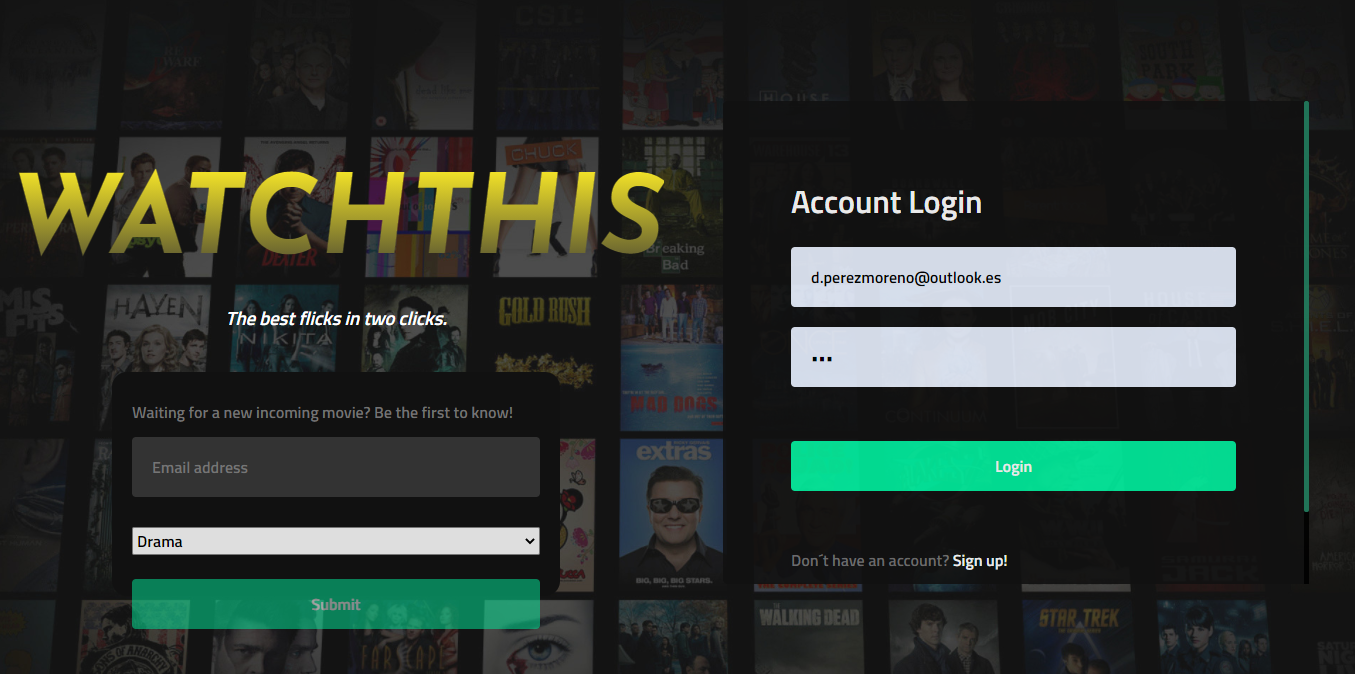
*Instalación de dependencias React*

Una vez instaladas todas las dependencias podremos poner nuestro servidor en marcha y ejecutar nuestra aplicación mediante Yarn, desplegando nuestro servicio en “localhost:3000”:



*Despliegue del servidor y ejecución de la aplicación React*

Si se han seguido todos los pasos de instalación y configuración, al abrir el navegador con la url “localhost:3000” nos tendría que salir el resultado final con el cual podremos interactuar y acceder a los servicios del proyecto:



*Proyecto Final Watchthis*