Arellano Granados Angel Mariano

218123444

Computación Tolerante a Fallas

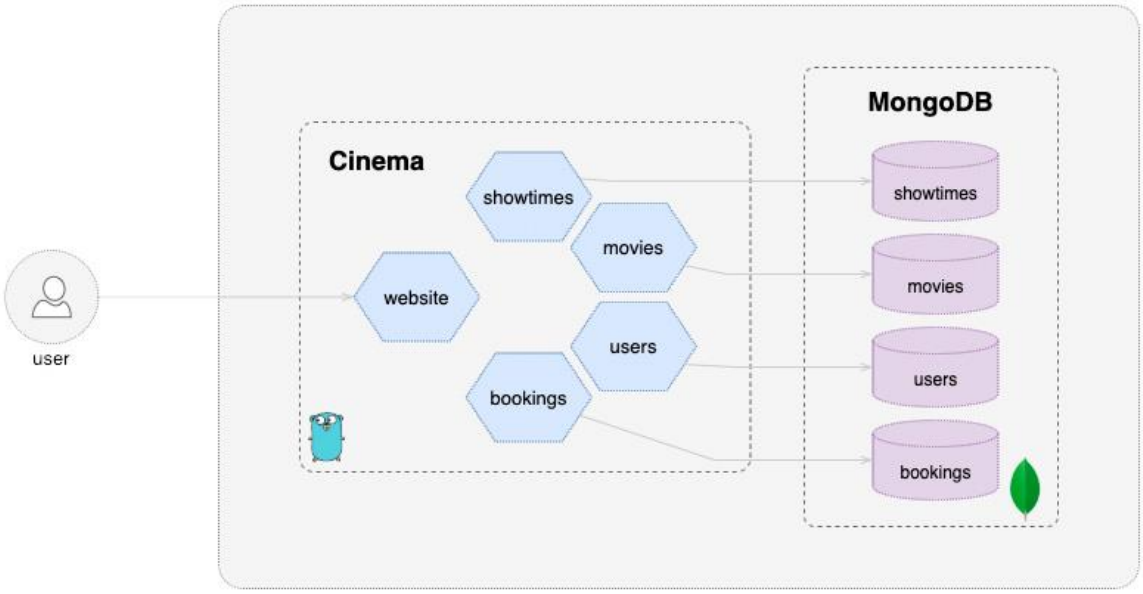
D06 2023B

Microservicios

Introducción

Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de API bien definidas.

Para esta práctica usaremos el código y procedimientos del directorio publico de GitHub microservices-docker-go-mongodb del autor mmorejon el cual es un ejemplo de una aplicación de Multiprocesos que se puede ejecutar localmente con ayuda de Docker, esta emula la gestión de una pagina web de un cine con la siguiente arquitectura.



Cinema es un proyecto de ejemplo que demuestra el uso de microservicios para una sala de cine ficticia. El backend de Cinema funciona con 4 microservicios, todos los cuales están escritos en Go, utilizando MongoDB para administrar la base de datos y Docker para aislar e implementar el ecosistema.

Servicio de películas: proporciona información como clasificaciones de películas, títulos, etc.

Servicio de horarios de espectáculos: proporciona información sobre horarios de espectáculos.

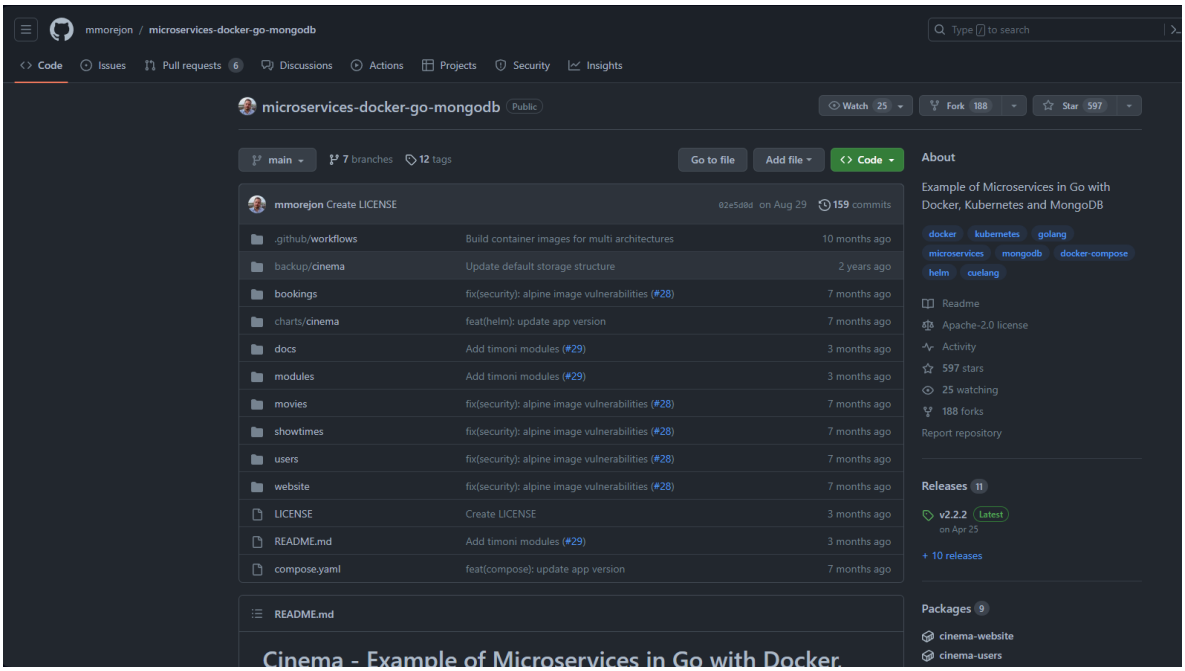
Servicio de reservas: proporciona información de reservas.

Servicio de usuarios: proporciona sugerencias de películas para los usuarios comunicándose con otros servicios.

Desarrollo

Instalación:

Para instalar la aplicación en nuestra computadora de manera local primero entramos al GitHub y descargamos los archivos locales.

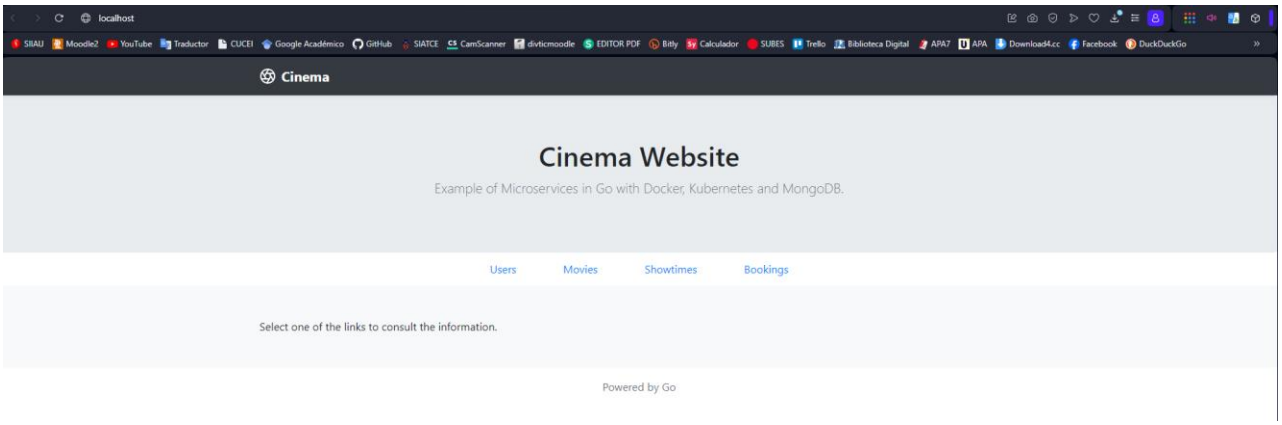


En una consola desde el directorio donde están todos los archivos ejecutamos el comando “docker compose up --detach" para iniciar el servidor.

Una vez termina de subir la aplicación podremos ver que todo salió bien en la salida de la consola.

Para confirmar que la aplicación está en un servidor local podemos ver los procesos corriendo simultaneamente con el comando “docker compose ps”

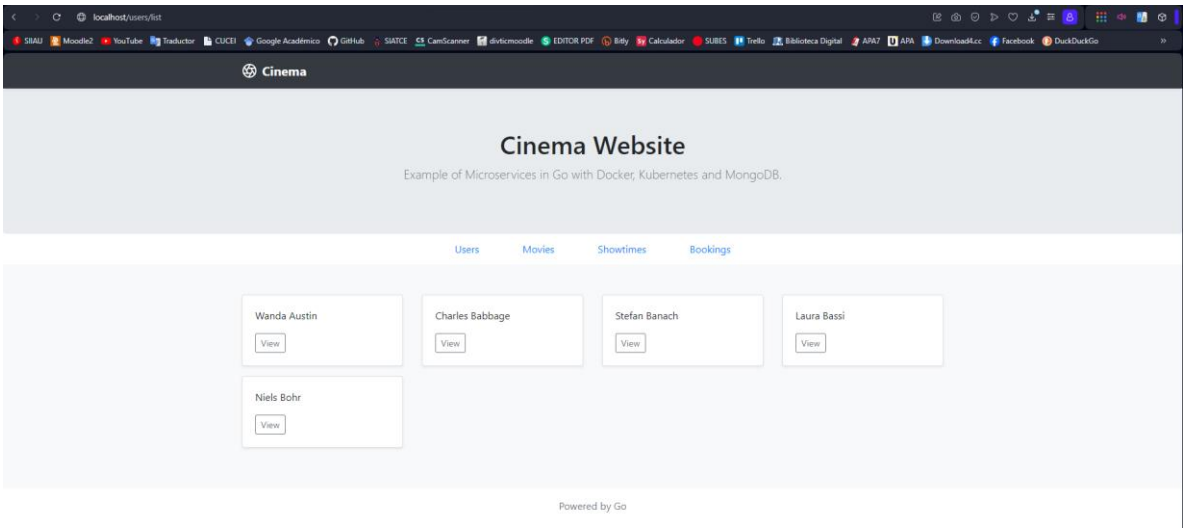
Visitando el link en cualquier navegador http://localhost podemos ver la pagina central de la aplicación, sin embargo, esta no cuenta con datos, pues la acabamos de crear.



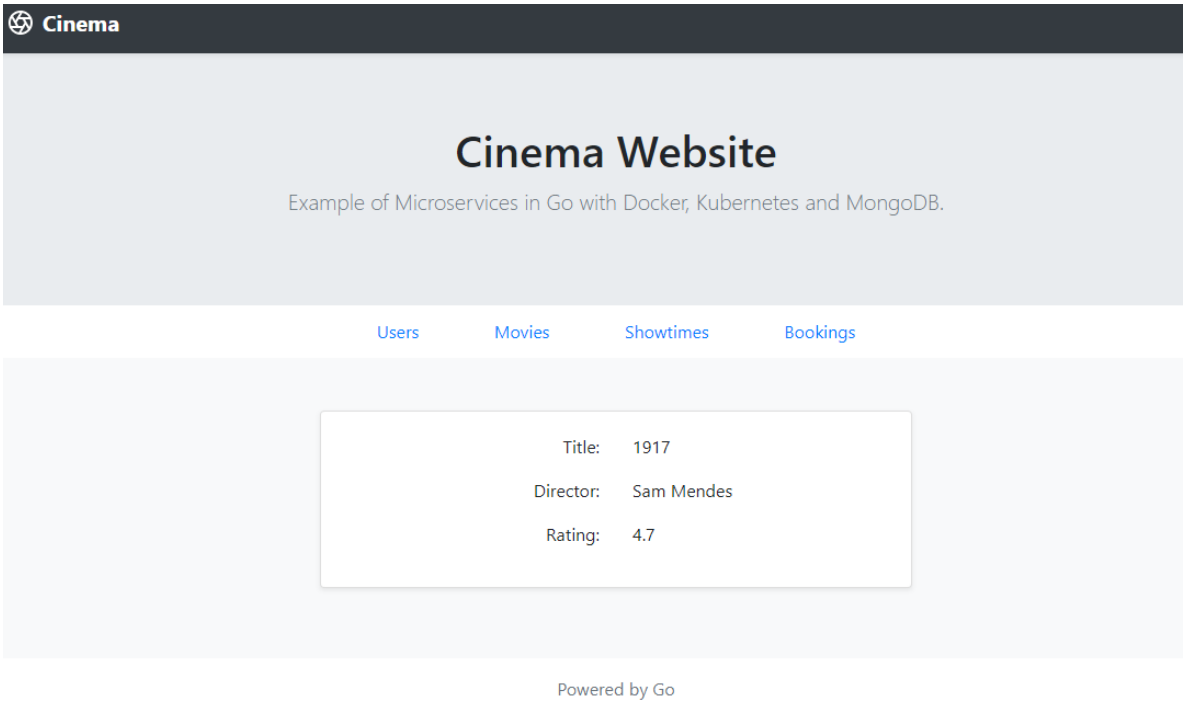
Podemos recuperar todos los datos de una base de datos que nos proporciona el autor original usando el siguiente comando.

```
docker compose exec db mongorestore \
  --uri mongodb://db:27017 \
  --gzip /backup/cinema
```

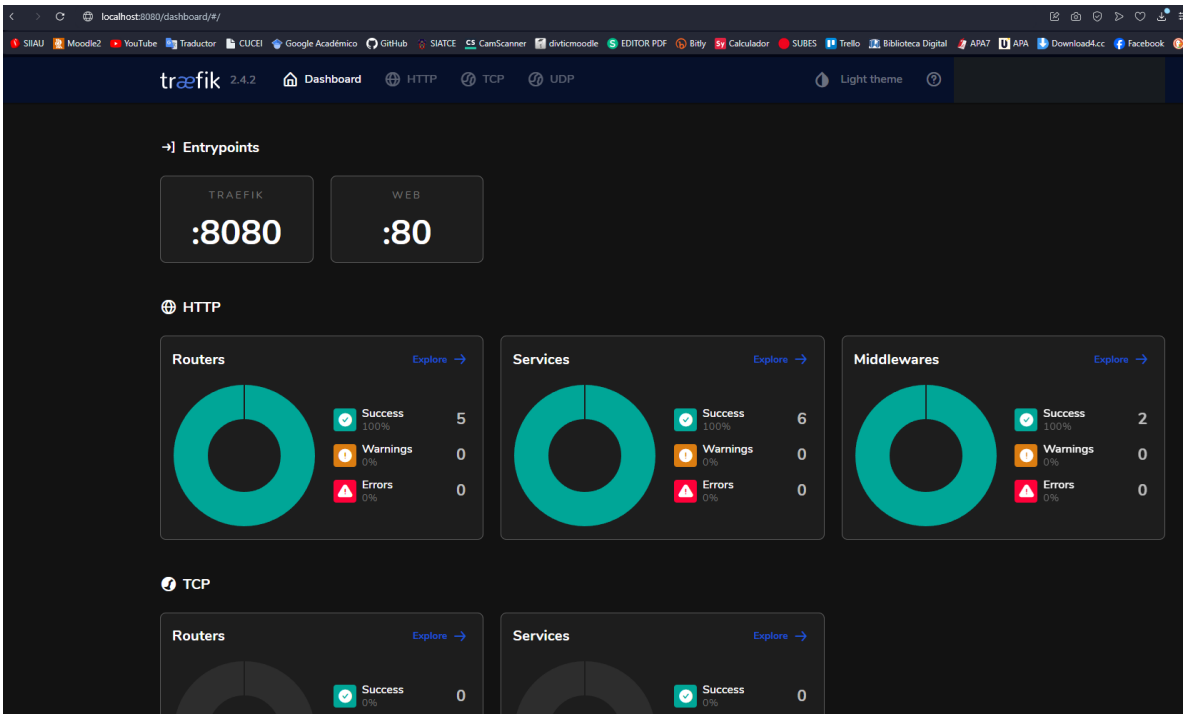
Entonces al entrar una vez más a la aplicación ya tendrá datos en cada uno de los apartados como usuarios, películas, horarios y reservas.

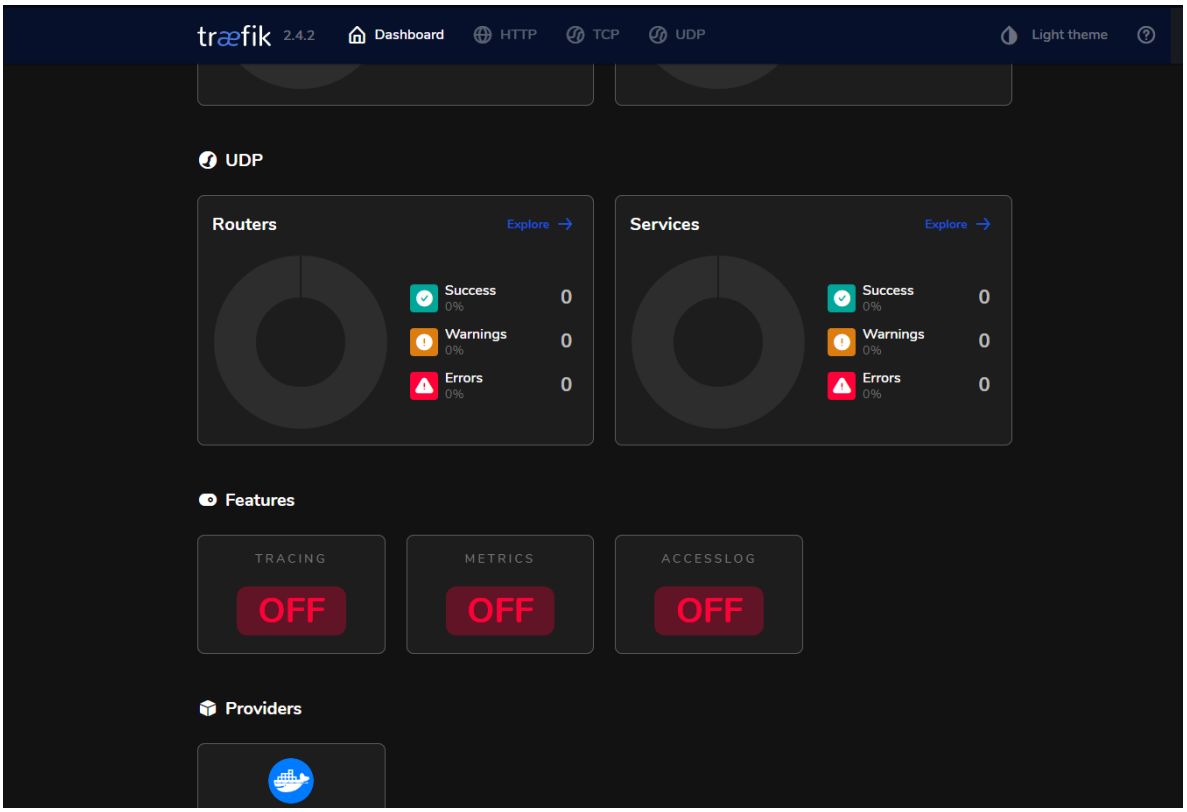


También podemos ver los detalles de cada uno de los datos en la aplicación como esta que es de una de las películas.



El servidor viene integrado con un método para traquear los errores en cada uno de los servicios con el gestor Traefik Proxy v2.4.2 al cual podemos acceder en el link http://localhost:8080 el cual cuenta con una interfaz bastante agradable.





Ya con todo esto solo nos queda agregar o cambiar algo a la API, desgraciadamente no pude implementar los comandos por mi falta de

conocimiento de los códigos de Linux y mas ejecutados desde Windows con WSL pues los comandos eran los siguientes.

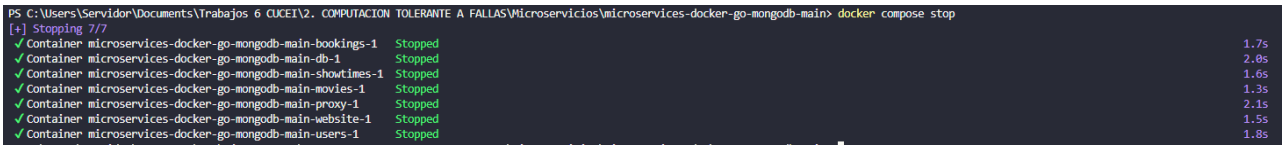
Service	Method	Endpoint
List users	`GET`	`/api/users/`
Get user by Id	`GET`	`/api/users/{id}`
Insert user	`POST`	`/api/users/`
Delete user	`DELETE`	`/api/users/{id}`

Service	Method	Endpoint
List movies	`GET`	`/api/movies/`
Get movie by Id	`GET`	`/api/movies/{id}`
Insert movie	`POST`	`/api/movies/`
Delete movie	`DELETE`	`/api/movies/{id}`

Service	Method	Endpoint
List showtimes	`GET`	`/api/showtimes/`
Get showtime by Id	`GET`	`/api/showtimes/{id}`
Get showtime by date	`GET`	`/api/showtimes/filter/date/{date}`
Insert showtime	`POST`	`/api/showtimes/`
Delete showtime	`DELETE`	`/api/showtimes/{id}`

Service	Method	Endpoint
List bookings	`GET`	`/api/bookings/`
Get booking by Id	`GET`	`/api/bookings/{id}`
Insert booking	`POST`	`/api/bookings/`
Delete booking	`DELETE`	`/api/bookings/{id}`

Por último, cerramos el servidor y la aplicación con el comando “Docker compse stop”



Conclusión

La arquitectura de Multiservicios es una estrategia de trabajo bastante útil , pues nos permite escalar nuestra aplicación según la demanda de cada módulo independiente, así como garantizar la integridad y seguridad de los módulos y el sistema entero pues si aparece un error fatal este solo afectara a una pequeña sección del programa.

Aun así esta practica fue extremadamente complicada para mi pues al no estar familiarizado con nada de estos temas cometí cientos de errores que no me permitían siquiera empezar la actividad, pues sufrí muchos errores en la instalación de WSL para poder siquiera abrir Docker, sim embargo tras bastante tiempo e insistir mucho logre echar a andar la aplicación en mi computadora personal y entregar la actividad.