UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1

MANUAL TÉCNICO

INTEGRANTES
201700857 Daniel Arturo Alfaro Gaitan
201709073 Walter Alexander Guerra Duque

Locust

Locust es una aplicación que se utiliza para realizar pruebas. Lo utilizamos para crear tráfico a la GoApi que funciona como producer de kafka. Se implementó utilizando python y se ejecuta con locust -f traffic.py. Nuestro archivo python está configurado para enviar en formato json a nuestro endpoint obteniendo los parametros del archivo traffic.json. El formato json es el siguiente:

```
{
"team1": "Guatemala",
"team2": "Argentina",
"score":"2-5",
"phase":<1,2,3,4>
}
```

Kafka

Kafka consiste en 3 partes: Consumer, Broker y Producer desplegado en google cloud. El producer se encarga de escribir en el broker la información recibida, el broker sólo almacena las entradas y el consumer distribuye la información según se necesita. El kafka broker es accedido por medio da la ip 34.135.161.214:9092.

Kafka Producer (Go Api)

Imagen: alexingguerra/producer

Se implementó una API utilizando go y gorillamux para recibir información por medio del endpoint /input. Utilizando el siguiente struct para almacenar las request.

```
type Info struct {
	Team1 string `json:"team1"`
	Team2 string `json:"team2"`
	Score string `json:"score"`
	Phase string `json:"phase"`
}
```

Luego se escribe en formato json la entrada al kafka broker. El producer se despliega utilizando kubernetes. Para acceder a la go api es por la siguiente dirección: http://35.223.89.127.nip.io:5010.

Kafka Consumer (Go app)

Imagen: alexingguerra/consumer
Es una aplicación implementada en go desplegada en kubernetes. Esta lee lo escrito en el kafka broker y lo guarda en un struct al igual que el consumer.

```
type Info struct {
	Team1 string `json:"team1"`
	Team2 string `json:"team2"`
	Score string `json:"score"`
	Phase string `json:"phase"`
}
```

Por cada elemento escrito se envia los datos al GRPC cliente y a una base de datos en Azure de Mongo usando Cosmosdb.

Mongo

Elemento de base de datos alojado en Azure, desplegado con Cosmos DB. Y esta almacenará la información con clave - valor
Únicamente almacenaremos los logs, estos registrados con la clave **Log** y en la base de datos **goDB.**
El acceso hacia la base de datos es a través de: mongodb://dbmong-g6:n2hUsQ1MC6Py4xkZOJ9zwSwJKlUa2vsgaUX6qvDVqaOZ4dUmw1SSfCQTTvQx4ONBm3lH9c4OxTTnWHlZTO7vkQ==@dbmong-g6.mongo.cosmos.azure.com:10255/?ssl=true&retrywrites=false&maxIdleTimeMS=120000&appName=@dbmong-g6@

GRPC

Nombre de imagen: danielalfaro1/servergrpc
GRPC fue diseñado en 2 partes, una que es la que enviará la información hacia un servidor, la cual llamaremos Cliente y el otro lado que recibirá la información le llamaremos Server.

El Cliente y Server comparten un archivo comunicación.proto, archivo que se compiló para el lenguaje de GoLang debido a que cliente y servidor están hechos en este lenguaje.

Server está alojado en: 20.120.51.0:50051

El archivo comunicacion.proto contiene los modelos base en los cuales el cliente y server para comunicarse. Los modelos son los siguientes:

```
message IngresoSolicitud {
  string team1 = 1;
  string team2 = 2;
  string score = 3;
  string phase = 4;
}
```
Este mensaje contendrá los datos necesarios para ingresar una nueva entrada en la base de datos de Redis.

```
message Respuesta {
  string codigo = 1;
  string mensaje = 2;
}
```
Este mensaje servirá para poder indicar al usuario si la operación se ha realizado con éxito o si hubo algún error

Redis

Nombre de imagen: redis:7.0.5

Redis se encuentra alojado en los clusters en Azure. De tal manera que este es una imagen pública registrada por los creadores de Redis. En esta imagen se almacenarán los datos con clave - valor. Estos datos están almacenados de esta manera:

Clave: Paises
Valor: [“Pais-Pais”, “Pais-Pais”,...]

Clave: Pais-Pais,[1-4]
Valor: {Pais: “Pais-Pais,[1-4], Predicas: [{Punteo: “1-2”, Votos: 1}, …]}

De esta manera Grcp server se encarga de registrar esta información en la base de Redis, la cual con un intérvalo propio de tiempo eliminará esta información almacenada en caché.

Go API

Nombre de imagen: danielalfaro1/gowebapi

Este es el nombre con el que se le conoce a un web service o una Rest API desarrollada en Go, para hacer comunicación entre las bases de datos de mongo y Redis hacia el frontend de react.
Alojado en https://34.135.93.12.nip.io:8000

Se tienen los endpoint necesarios para las transacciones de información y operación de limpieza que consume el frontend.

/GetPaises : Este endpoint permite al usuario obtener una lista de países registrados en redis.
/GetPaisFase: Este endpoint permite al usuario obtener la información de las quinielas registradas para una fase en específico
/obtenerLogs : este endpoint permite al usuario obtener los logs registrados en la base de datos de mongo
/deleteLogs : este endpoint permite al usuario eliminar todos los registros actuales de la base de datos de mongo

Front de React

Nombre de imagen: danielalfaro1/frontreact
Esta es la página en la cual se va a proyectar nuestro sistema, esto está alojado en https://frontreact-iqkcivgykq-uc.a.run.app, a través de esta página el usuario podrá observar los votos hecho sobre cada partido, así como observar los logs ingresados en la base de datos de mongo. También se podrá borrar la información registrada en mongo.

La página se divide en 3 partes:
App.js : este se encarga de realizar el ruteo entre /live y /logs
Live:

- Live.jsx : componente que despliega la página y la información en pantalla que obtiene del servicio
- ServicioLive.js : componente exclusivo para uso desde Live.jsx, permite obtener la información desde la api de Go, y lo traslada al componente ya mencionado.

Logs:

- Logs.jsx : componente que despliega la página y la información en pantalla que obtiene del servicio.
- ServicioLogs.js : componente exclusivo para uso desde Logs.jsx permite obtener la información desde la api de Go y traslada al componente ya mencionado. Así como permite ejecutar la eliminación de logs almacenados en Mongo.