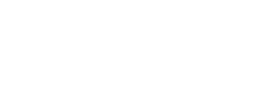
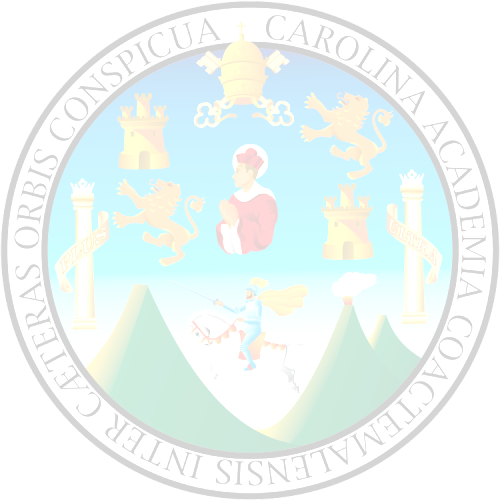
Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Sistemas Operativos 1

Laboratorio



**Proyecto**



Grupo 17

# Introducción

Se requiere la construcción una arquitectura de sistema distribuido genérica que muestre las predicciones de las personas sobre los partidos de futbol del mundial de Qatar 2022. Estos datos se deben procesan mediante una conexión multicluster, uno de Google Cloud y el otro de Microsoft Azure. Esta conexión divide el procesamiento de datos en datos y registros en vivo. Para este proyecto se utilizará Linkerd para interconectar los clusters y Chaos Mesh para Ingeniería de Caos.

# Herramientas Utilizadas para el Desarrollo

## React

Es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

## Go Language

Go es un lenguaje de programación concurrente y compilado con tipado estático inspirado en la sintaxis de C, pero con seguridad de memoria y recolección de basura.

## Ubuntu

Ubuntu es una distribución Linux basada en Debian GNU/Linux, que incluye principalmente software libre y de código abierto. Puede utilizarse en ordenadores y servidores. Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario.

## Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

**Kubernetes**

Kubernetes es una plataforma de sistema distribuido de código libre para la automatización del despliegue, ajuste de escala y manejo de aplicaciones en contenedores​ que fue originalmente diseñado por Google y donado a la Cloud Native Computing Foundation.

## MongoDB en CosmosDB

Es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

**Locust**

Es una herramienta de fácil uso para el testing del performance de una aplicación. El comportamiento se define en lenguaje Python y ofrece un atractivo UI al ser levantado su servidor.

**Go y Locust**

Se utilizó Go para crear un pequeño servidor capaz de ejecutar un script que nos permite levantar locust y cuyo host es un servicio expuesto en el clúster de k8s en GCP que manda el tráfico generado hacia otro servicio que inserta directamente en MongoDB.

Graphical user interface, text, website

Description automatically generated

# Kafka y MongoDB

Se utilizó una instancia de CosmosDB de lado de Azure para la implementación de una base de datos MongoDB.

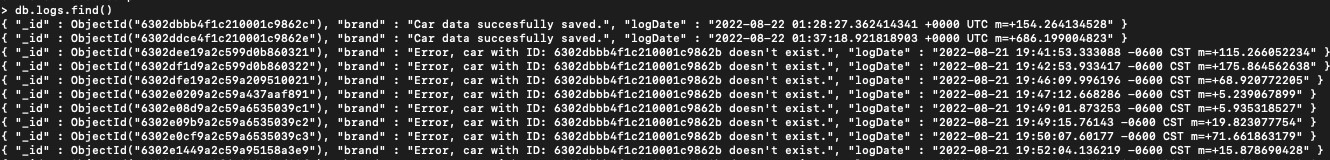
Graphical user interface, text, application, email

Description automatically generated

Mediante colas de Kafka esta información es transmitida y al momento de ser consumido se insertan automáticamente cada dato a la base de datos.

# Imagenes y Contenedores

Con ayuda de Docker se generaron dos imágenes: la del frontend y servidor, y por aparte, se utilizó la imagen de mongo ya existente. Se le definió un volumen para la persistencia de datos.



**gRPC**

gRPC es un sistema de llamada a procedimiento remoto de código abierto desarrollado inicialmente en Google. Utiliza como transporte HTTP/2 y Protocol Buffers como lenguaje de descripción de interfaz.

Se crearon dos aplicaciones en Go, uno maneja el cliente y otro el servidor. El cliente básicamente realiza la inserción a la base de datos no relacional.

**Clústers K8s**

Se hicieron uso de dos proveedores en la nube: Google Cloud y Azure. Ambos con sus respectivas capas gratuitas. En el clúster de GKE se concentran los servicios para obtener el tráfico, el manejo de la cola de kafka y el cliente de gRPC.

Text

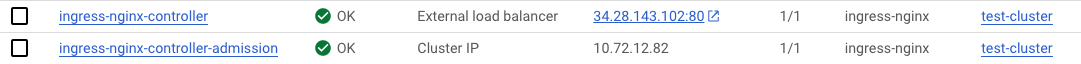
Description automatically generated

El clúster de Azure contiene el servidor de gRPC y el proveedor como tal contiene la base de datos de mongo. Así como el servidor de Go al cual la aplicación web consultará.

Graphical user interface, text, application, email

Description automatically generated

Ambas instancias cuentan con un ingress controller y sus respectivos recursos. Del cual se utilizó uno basado en Nginx.

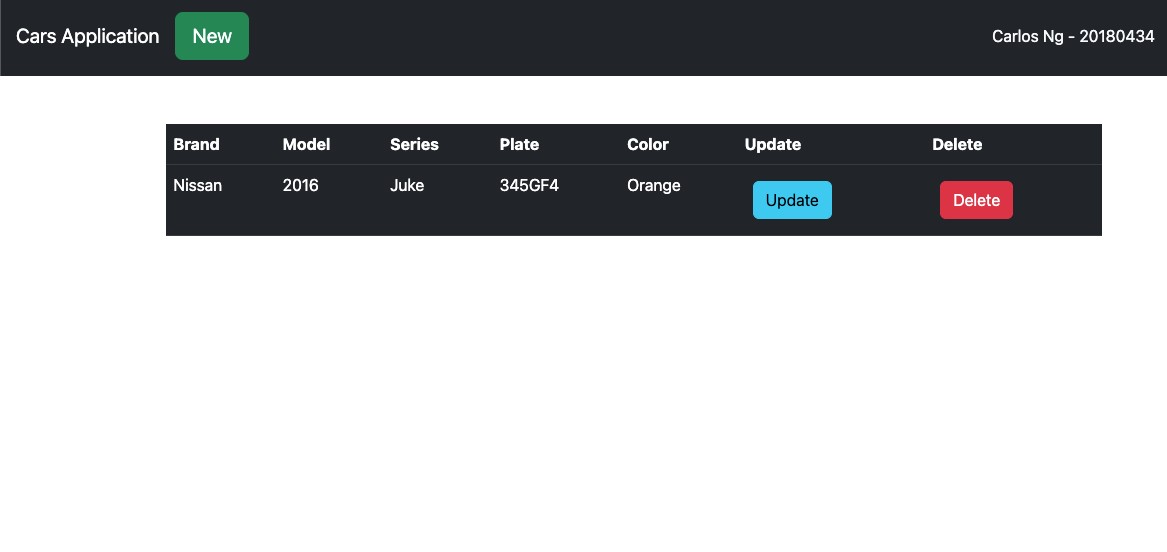


# Aplicación

La aplicación cuenta con una interfaz sencilla y amigable para el usuario. Se distribuye de la siguiente manera:

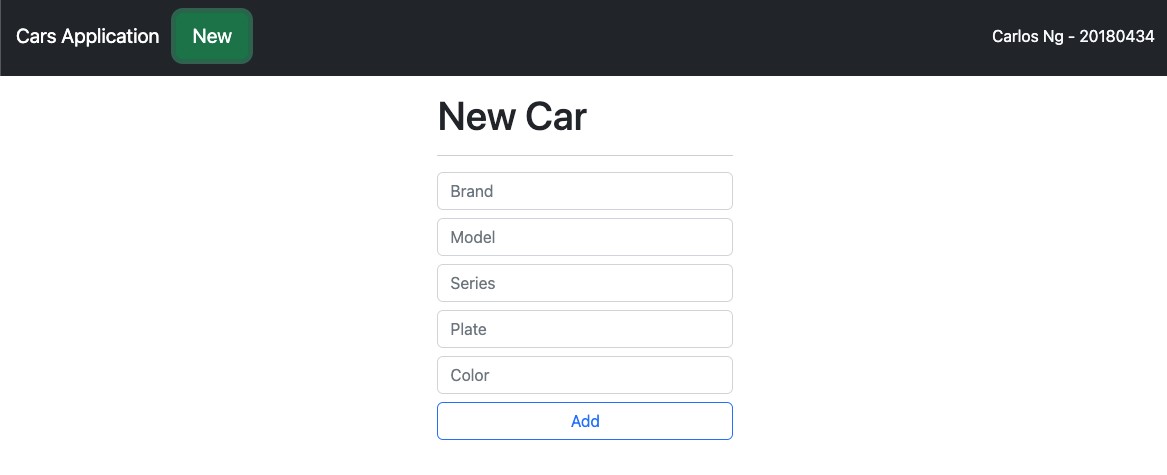
## Pantalla de Inicio

Esta pantalla lista los carros que ya se encuentran dentro de la base de datos.



## Pantalla de Agregar Carro

Formulario que permite al usuario ingresar los datos necesarios para guardar los datos de un nuevo carro.



## Pantalla de Actualizar Carro

Formulario que le permite al usuario ingresar cuáles datos del carro actualizar.

