UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1 SECCIÓN N
ING. SERGIO ARNALDO MENDEZ AGUILAR
AUX. BRYAN OTONIEL ORDOÑEZ MORALES

PROYECTO 2

MANEJO DE CONCURRENCIA

OBJETIVOS

- Poner en práctica los conocimientos adquiridos en el curso y laboratorio de sistemas operativos 1.
- Entender el concepto de concurrencia al implementar un sistema en el que este concepto es primordial.
- Resolver problemas mediante el uso de tecnología contemporánea.

DESCRIPCIÓN

El proyecto consiste en crear un sitio web que muestre los datos enviados en tiempo real con ayuda de dos balanceador de carga, Docker Compose, kubernetes y las bases de datos de Mongodb y Redis.

La aplicación utilizara el concepto de colas para el manejo de la concurrencia. Se debera de transmitir la información mediante diferentes nodos, cada uno con implementado con tecnología diferente.

Primera Parte (Envió de datos)

La primera parte consta en realizar un programa en Go el cual debe enviar los datos de un archivo de entrada a los balanceadores de carga, los balanceadores de carga serán implementados en:

- Contour
- Nginx

El archivo con los datos tendrá el siguiente formato:

```
[
    {
        "Nombre": "Neto Bran",
        "Departamento": "Guatemala",
        "Edad": 25,
        "Forma de contagio": "Comunitario"
        "Estado": "Activo"
    },
    {
        "Nombre": "Aironmen",
        "Departamento": "Zacapa",
        "Edad": 50,
        "Forma de contaio": "Comunitario",
        "Estado": "Recuperado"
    }
]
```

El tendrá N cantidad de datos (para la calificación será proporcionado por el auxiliar).

Cuando se ejecute el programa de Go se deberá de preguntar lo siguiente:

- Url del balanceador de carga que se desea enviar
- Cantidad de hilos que se desean para enviar
- Cantidad de solicitudes que tiene el archivo (la cantidad de casos que se desean enviar)
- Ruta del archivo que se desea cargar

Segunda Parte (Balanceadores de carga)

Nota: De esta parte en adelante será en la nube.

La segunda parte empieza por instalar kubernetes la cual será la base de las aplicaciones.

Balanceadores de carga (Nginx y Contour):

Será el mismo trabajo con dos balanceadores de carga distintos, se deben de instalar los balanceador de carga de forma local, los balanceadores de carga apuntará a dos contenedores que tendrán cada uno un servidor web corriendo sobre go, entonces los balanceadores de carga recibirán los datos de la aplicación local y los distribuirá entre los dos contenedores.

Nota: Los contenedores de go deberán de tener escalabilidad vertical/horizontal para cuando los hilos aumenten pueda soportar la carga, queda a discreción del alumno como implementarla.

Tercera Parte (Mensajería entre aplicaciones)

La tercera parte se necesita obtener la información que se envió y mandarla por medio de por medio de dos herramientas:

> **NATS.io:** Es un sistema de mensajería de código abierto simple, seguro y de alto rendimiento para aplicaciones nativas de la nube, mensajería IoT y arquitecturas de microservicios.
>
> **gRPC:** Es un framework moderno de llamada a procedimiento remoto (RPC) de código abierto que puede ejecutarse en cualquier lugar. Permite que las aplicaciones cliente y servidor se comuniquen de manera transparente y facilita la creación de sistemas conectados.

La información será enviada a dos contenedores que correrán una aplicación en Python, estas aplicaciones se encargarán de almacenar los datos finalmente en las dos bases de datos que se estarán manejando.

Cuarta parte (Bases de datos)

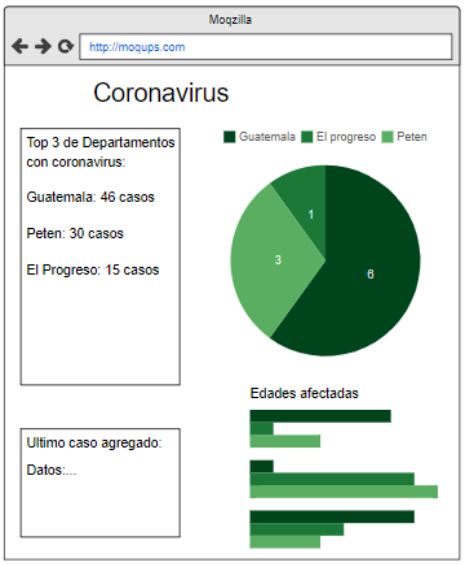
Se necesita que se implementen bases de datos no relaciones para mejorar la velocidad y rendimiento de la aplicación, las bases de datos a manejar serán:

> **MongoDB:** 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.
>
> **Redis:** Es un almacén de datos en memoria de código abierto (con licencia BSD), utilizado como base de datos, caché y agente de mensajes. Admite estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, hiperloglogs, índices geoespaciales con consultas y flujos de radio.

Estas bases de datos deberán de ser instaladas en máquinas virtuales en la nube.

Quinta parte (Pagina Web)

Por último, se debe de crear una página web corriendo sobre un servidor de node js, dentro de la página la cual tendrá una temática de coronavirus debe de tener lo siguiente:



Apartados:

Deberá mostrar una tabla con todos los datos (MongoDB).

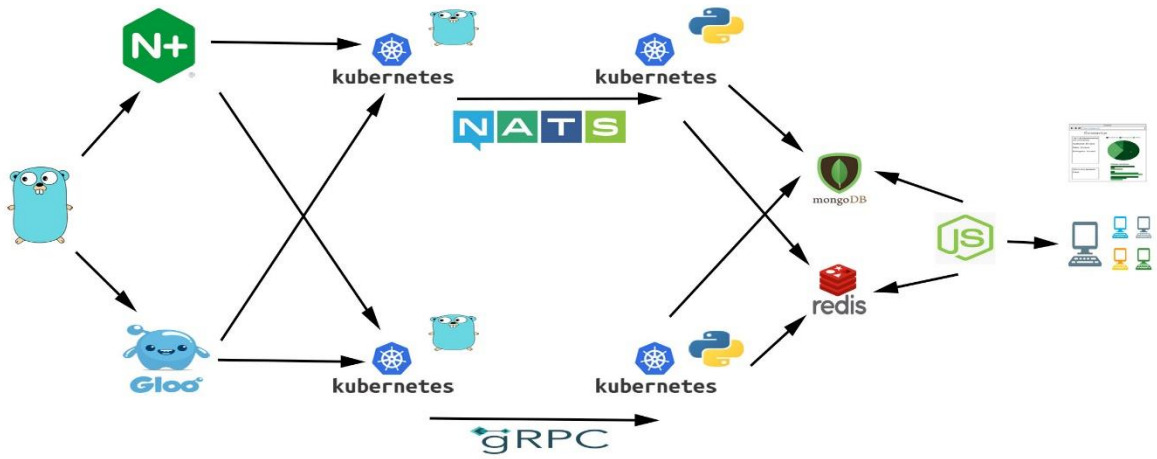
Deberá mostrar el top tres de departamentos con más casos(MongoDB).

Gráfica de Pie con todos los departamentos afectados (MongoDB).

El último caso agregado (el último del archivo) (Redis).

Una gráfica de barras que muestra la cantidad de afectados por rango de edades, ej. 0-10 años 5 casos (Redis).

Arquitectura Final



MONITOREO

Jaeger

Se debe de implementar jaeger tracing en todos los contenedores de la línea superior para poder monitorear todas las acciones que suceden en estos.

Linkerd

También se le solicita implementar Linkerd en los contenedores de la línea inferior, esta es una herramienta que trabajara en conjunto con kubernetes para el monitoreo de los contenedores.

Prometheus

Se deberá instalar el softeware de prometheus para el monitoreo de las máquinas virtuales que contienen las bases de datos. Se sugiere una instalación local bajando los binarios.

RESTRICCIONES

- El proyecto se realizará en modalidad de trios.
- Debe implementarse en lenguaje de establecido en cada sección.
- Debe realizarse manual técnico y de usuario.
- Copias totales o parciales, tendrán una nota de 0 puntos y será reportado a escuela de sistemas.
- Entregas Tarde no se aceptarán.

ENTREGABLES

- Código de los servidores .
- Manual del proceso en formato PDF.

En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará el proyecto; por lo cual, se tendrá una nota de cero puntos.

La entrega se debe realizar antes del Lunes 29 de Junio de 2020