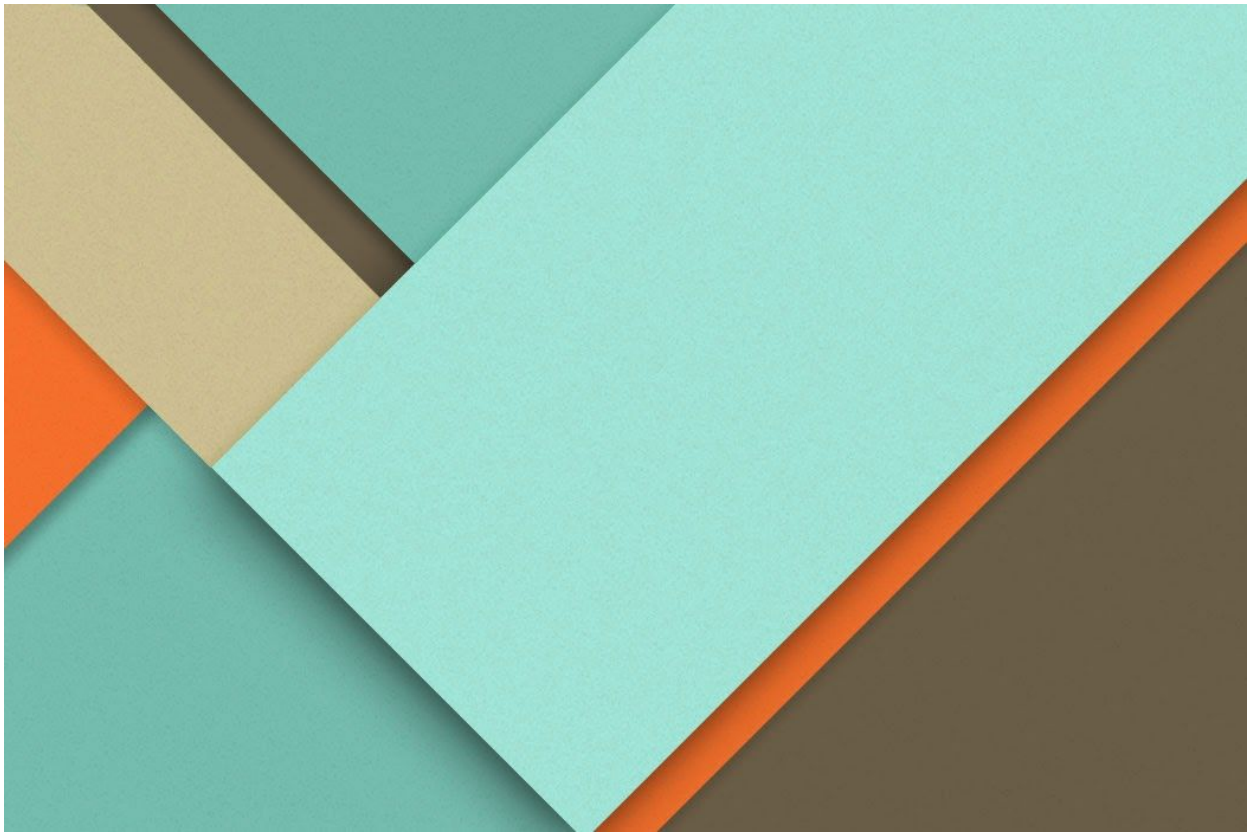


VirtualMall [Fase 1]

Primer Semestre 2021

Ing. Jesus Guzman, Ing. Alavaro Hernandez, Ing. Luis Espino

Marvin Martinez, Andree Avalos, Randolph Muy, Jorge Salazar, Josué Pérez, José Véliz

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Estructuras de Datos

Visión general

A raíz de la crisis sucedida por la pandemia y la casi imposible movilización para poder realizar cualquier tipo de compras, la empresa EDDsystems le ha encargado a usted como desarrollador crear un proyecto que sea capaz de gestionar tiendas simulando el acceso a centro comercial con los comercios que este posee de manera segura y confiable.

VirtualMall será un aplicación que le permita a los clientes interactuar con las tiendas que ellos deseen de manera eficaz y segura, para esto, el sistema debe ser capaz de manejar los distintos servicios que las tiendas brindarán a sus clientes, como lo son las ventas bajo pedido y los posibles envíos a distintas locaciones, también tener la capacidad de guardar toda la información que se necesite de las tiendas, sus productos, sus servicios y sus clientes, estos deben ser almacenados de forma segura y confiable.

Objetivos

1. Que el estudiante se familiarice con el lenguaje Go
2. Que el estudiante conozca el funcionamiento de peticiones REST mediante la implementación de un servidor web
3. Que el estudiante sepa envolverse en el ámbito de manejo de la memoria
4. Comprender y desarrollar distintos tipos de estructuras básicas
5. Familiarizarse con el manejo de lectura y escritura de archivos JSON
6. Familiarizarse con el uso de Git.

Especificaciones

Se solicita realizar una aplicación de servidor en la cual se puedan realizar la carga de múltiples tiendas que contendrá nuestra tienda en línea, para eso se deben implementar una serie de url en las que nosotros podamos subir el archivo de tiendas y se almacenen en el servidor. También se debe contar con la opción de guardar esta información con el propósito de que antes de que se cierre el servidor se puedan almacenar las tiendas en un archivo JSON.

Almacenamiento de Tiendas

Servidor REST

La aplicación contará con un servidor de tipo REST desarrollado en el lenguaje Go, este servidor será el encargado de recibir todas las peticiones necesarias, y devolver la información que el usuario espera.

Para la carga de tiendas que se describe en el módulo posterior, se tendrá un endpoint de tipo **POST**, esta información será enviada desde el body de la herramienta que sea utilizada para realizar esta petición, que contendrá toda la información relacionada con las tiendas que maneja el sistema.

El endpoint debe ser de la siguiente manera: **localhost:*puerto*/cargartienda** (El número de puerto utilizado en su defecto es el 3000.)**.**

Además de este endpoint, existirán distintos endpoints de consulta, estos endpoint permitirán obtener información acerca de las estructuras que ya se tienen almacenadas, de tal manera que se podrá proveer a los usuarios de datos importantes sobre las tiendas del sistema.

A continuación se describe la carga de tiendas, estableciendo la manera en la cual debe realizarse, luego se establecerá un listado con todos los reportes que pueden ser realizados, detallando cada uno de los endpoints y la información que debe de obtenerse.

Cargar Tiendas

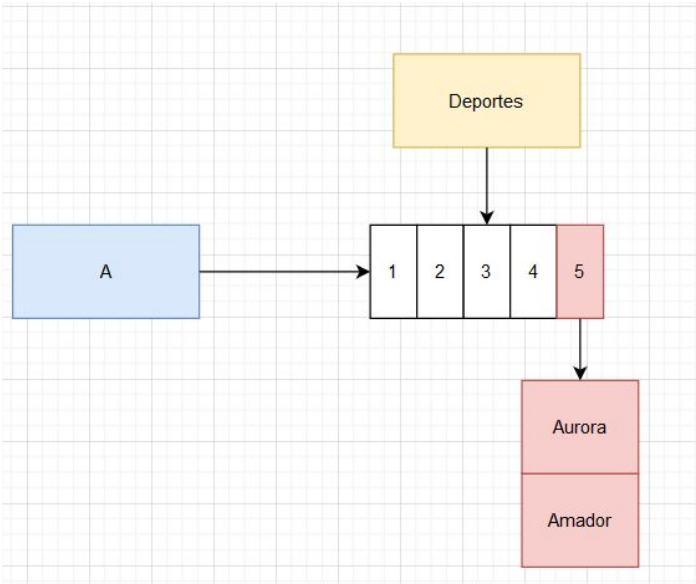
La carga de las tiendas, se realizará por medio de un archivo de entrada JSON con la siguiente estructura:
https://drive.google.com/file/d/146bILejN5GkpuNtyDhQ-fcEmnJ05Ywmt/view?usp=sharing

Estructura del archivo

El documento simula una matriz utilizando como encabezado de columnas las categorías de las tiendas, para los encabezados de las filas se utilizara las letras iniciales con las que inician las tiendas y para las celdas van a estar divididas por 5 calificaciones, ejemplo:

```
{
    "Datos": [{
            "Indice": "A",
            "Departamentos": [{
                    "Deportes": [{
                            "Nombre": "Aurora",
                            "Descripcion": "Una descripcion",
                            "Contacto": "5544-3377",
                            "Calificacion": 5
                        }
```

Esta tienda tiene columna Deportes, fila ‘A’ y en la calificación de 5. Una celda puede contener varias tiendas dentro, la vista de la matriz quedaría de la siguiente manera.



Nota: El archivo será enviado en el body de una petición post

Tiendas

Los campos de una tienda en esta fase serán los siguientes:

- Nombre
- Descripción
- Contacto
- Calificación

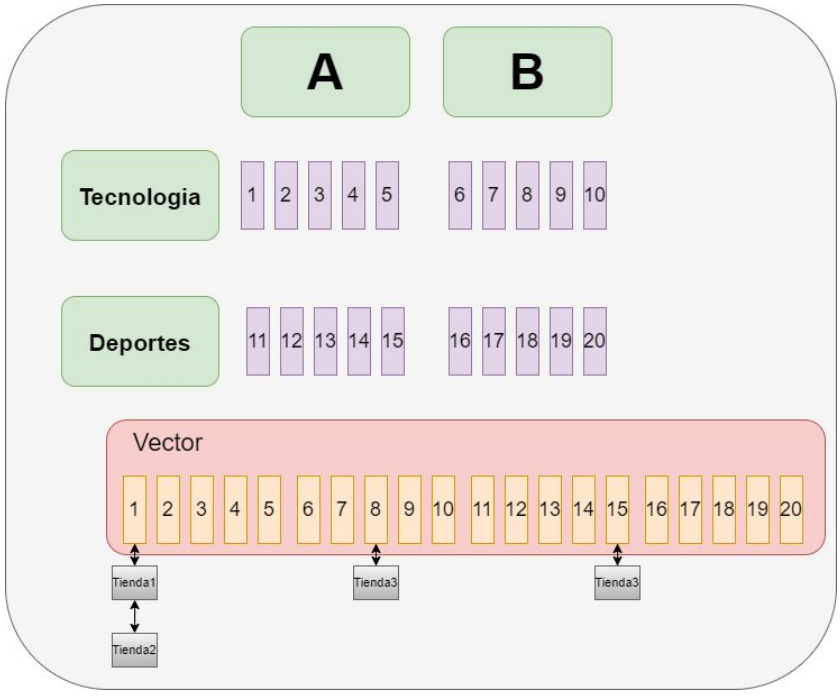
El campo calificación nos servirá para clasificar las distintas tiendas que contiene el servidor. Cada categoría puede almacenar N tiendas con la misma letra inicial para que exista un ranking de las mejores tiendas clasificadas por nuestro servidor para que al momento de que un cliente tenga la necesidad de buscar una tienda identifique el prestigio que ella tiene. La calificación puede tomar los siguientes valores:

- **Magnifica (5)**
- **Excelente (4)**
- **Muy Buena (3)**
- **Buena (2)**
- **Regular (1)**

Linealización

Al momento de leer el archivo de entrada la cual simulará una matriz de 4 dimensiones, el estudiante debe ingresar toda la información en un vector, logrando así una linealización de la matriz. La manera de linealizar la matriz será por medio de col major y row major, por lo que deben leer las consideraciones de este enunciado para saber a qué estudiante le corresponde utilizar col o row major. La manera de linealizar seria de la siguiente manera:

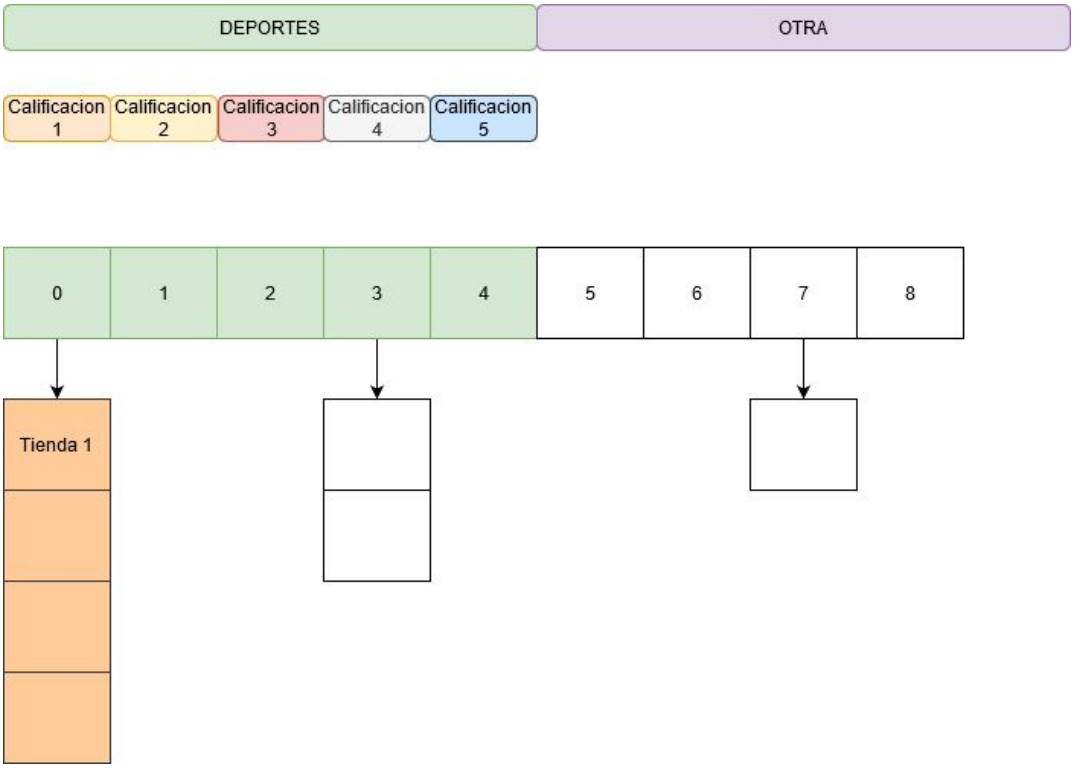
- En el archivo de entrada, la cual tiene un formato en Json , **el Índice** corresponde a filas (1era dimensión) y **departamentos** corresponde a columnas (2da dimensión), de modo que la tercera dimensión es el tipo 5 de calificaciones. Por lo tanto, si tenemos una matriz de 2 x 2 en el archivo Json, tendríamos un vector de 20 posiciones. como en el siguiente ejemplo.



- La cuarta dimensión serían todas las tiendas que se pueden observar en el archivo Json, por lo que el estudiante debe poder ubicar a las tiendas en la posición de la tercera dimensión del vector respectivamente. formando así una lista doblemente enlazada de tiendas.
- La tercera dimensión corresponde al Top 5 de calificación como se mencionó anteriormente por lo que la tienda debe ir ubicada con respecto a su calificación (la cual está como atributo en el archivo Json), de modo que habrán tiendas con la misma calificación y al momento de ubicarlas se insertaran en la posición

correspondiente del vector de modo que la lista doblemente enlazada (de tiendas) irá creciendo.

- Las tiendas tienen que estar ordenadas en la lista doblemente enlazada para que la búsqueda sea mucho más fácil. El ordenamiento se va a realizar utilizando el valor ASCII del nombre de la tienda.
- Para una mejor manera de visualizar la matriz linealizada en sus 4 dimensiones observe la siguiente imagen.



Nota: Para cada unión de categoría y letra se tendrán que crear todas las calificaciones (excelente, bueno ...) aunque no exista una tienda con esa calificación, dejando así el espacio reservado.

Reportes:

En este apartado se describirán los reportes que son necesarios para que los usuarios de la aplicación puedan conocer la información que se encuentra almacenada y consultarla, por ello a continuación, se detalla un listado con todos los reportes que se encuentran disponibles.

➔ **Grafico completo del arreglo**

Este reporte permitirá visualizar el arreglo completo con sus listas que se generará al momento de aplicar la técnica correcta de linealización, permitiendo de tal manera visualizar el gráfico de la lista doblemente enlazada, y así comprobar que la estructura creada, corresponde correctamente a lo que se requiere.

Para lograr esto se debe generar un gráfico del arreglo con sus listas y almacenarse en la misma carpeta del proyecto.

El endpoint que permitirá ejecutar dicha acción será de tipo GET, y la dirección o URL exacta es: **localhost:*puerto*/getArreglo .**

➔ **Búsqueda de posición específica**

Para este reporte el usuario realizará una solicitud especificando los tres parámetros necesarios para mostrar la información de una tienda que se encuentre dentro de la estructura del arreglo, de tal manera que se enviaran tres parámetros en formato json, estos parámetros son **el departamento, el nombre de la tienda y la calificación de la tienda,** con estos datos la aplicación devolverá la información de la tienda específica que cumpla con todos los parámetros enviados.

Un ejemplo del json que debe enviarse es el siguiente:

```
{
    "Departamento": "Vestuario",
    "Nombre": "Puma",
    "Calificacion": 4
}
```

En el momento que la aplicación reciba este json debe retornar toda la información de la tienda.

El endpoint que permitirá ejecutar dicha acción será de tipo **POST**, el contenido es como el ejemplo mostrado con anterioridad, y la dirección o URL exacta es : **localhost:*puerto*/TiendaEspecifica.**

Ejemplo de salida en formato JSON:

```
{
    "Nombre": "Puma",
    "Descripcion": "es una empresa alemana fabricante de accesorios, ropa y calzado deportivo, cuya sede central está en Herzogenaurach, Alemania",
    "Contacto": "5588-9988",
    "Calificacion": 4
}
```

➔ **Búsqueda de posición en resultado linealizado**

Para este reporte el usuario podrá realizar una petición de tipo **GET** hacia nuestro servidor donde se le enviará como parámetros en la **URL** la posición dentro de la lista, dando como retorno a la petición toda la información de las tiendas perteneciente a esta posición, en dado caso no exista ninguna tiene en esta posición deberá retorna un mensaje donde explique que no hay registro de tiendas en dicho índice.

Ejemplo:

- **localhost:*puerto:/*id/:numero**

Nota: número es el índice que se desea retornar.

Ejemplo de salida en formato JSON:

```
{
    "Nombre": "Hersheys",
    "Descripción": "Es la compañía fabricante de chocolates más grande de Estados Unidos.",
    "Contacto": "(502) 7788-6666",
```

```
    "Calificacion": 5
}
```

➔ **Eliminar Registro**

Esta función nos permite eliminar una tienda que ya no necesitamos mandando a una url el **Nombre, Categoría y Calificación.** Con los datos mandados se procederá a realizar el cálculo en la posición que debería de ir la tienda y eliminar en esa posición.

Ejemplo:

localhost:*puerto*/Eliminar.

Ejemplo del json que debe enviarse es el siguiente:

```
{
      "Nombre": "Wallmart",
      "Categoria": "Tecnologia",
      "Calificacion": 3
}
```

Nota: las calificaciones serán las siguientes:

- **5**
- **4**
- **3**
- **2**
- **1**

Se debe de obtener los índices correspondientes a cada parámetro.

➔ **Guardar Datos**

Esta función del servidor permite almacenar la información que contiene la matriz de tiendas en un nuevo archivo con formato JSON, este archivo debe ser parecido al archivo de entrada.

Nota: Se va a cargar el archivo generado por el estudiante para comprobar que el formato es el adecuado.

Consideraciones

I. Lenguaje a utilizar será **Go**

II. **Puerto** por defecto **3000**

III. Las estructuras serán realizadas por el estudiante.

IV. IDE a utilizar **Goland** o **Visual Studio Code**

V. La entrega será por medio de la plataforma de **UEDI**

VI. Los alumnos con carné par realizarán la linealización por medio de colum major, y los alumnos con carné impar realizarán la linealización por medio de row major.

VII. El estudiante debe tener un repositorio privado en github con el nombre **EDD_VirtualMall_#Carné** y agregar a su tutor como colaborador al repositorio del proyecto (Cada tutor les hará llegar su usuario)

VIII. Será calificado del último commit realizado antes de la fecha de entrega.

IX. Las copias totales o parciales **serán penalizadas con nota de 0 puntos** y reportadas ante la escuela de ciencias y sistemas.

X. Todos los reportes deben ser generados con graphviz.

XI. **Fecha de entrega: 20 febrero de 2021 antes de las 23:59 horas.**