INF-343 SISTEMAS DISTRIBUIDOS

TAREA Nº1: "Desarrollo de una aplicación distribuida simple de comercio electrónico usando servicios web"

2 de septiembre de 2022

ANTECEDENTES GENERALES

Objetivos:

- Aprender a programar con el lenguaje de programación concurrente desarrollado por Google (https://go.dev/)
- Desarrollar una aplicación con servicios Web basados en RESTful APIs, programando con lenguaje Go y usando Gin Web Framework.

Fechas importantes:

Ayudantía por Zoom: martes 6 de septiembre de 2022
Fecha de entrega de la tarea: jueves 15 de septiembre de 2022

ESPECIFICACIÓNDE LA TAREA

Descripción de la aplicación

La aplicación permite la simulación de una compra habitual en un supermercado. Por una parte, está el cliente, quien ingresa los productos que le interesa a su canasta y luego paga por ellos. Por otra parte, se encuentra el administrador, quién está interesado en la gestión de los productos y en conocer estadísticas que le permita tomar mejores decisiones.

De acuerdo con lo mencionado anteriormente, desarrolle una aplicación que simule una compra simple en un supermercado. Para esto, considere a dos tipos de usuario: cliente y administrador.

El cliente inicia sesión con un usuario y una contraseña, cuyos datos se encuentran en la base de datos. Si el inicio de sesión es satisfactorio, el cliente podrá realizar dos tipos de acciones:

- Listar productos.
- Realizar una compra.

Si el cliente selecciona la primera opción, entonces los productos se listarán línea por línea. En cambio, si decide realizar una compra, entonces primero ingresará cuántos productos desea comprar y luego por cada uno de ellos ingresará el ID y la cantidad que llevará.

Cuando la compra finalice, muestre la cantidad de productos comprados y el monto total de la compra.

Para el caso del administrador, su inicio de sesión solo consiste en ingresar la contraseña correcta: 1234. Si el valor ingresado es correcto, entonces podrá seleccionar las siguientes opciones:

- Listar productos.
- Crear producto.
- Eliminar producto.
- Ver estadísticas.

La primera opción permite listar los productos línea por línea. En tanto, la segunda y tercera opción realizan la acción de crear y eliminar respectivamente. Para la cuarta opción, la aplicación debe entregar las siguientes estadísticas:

- ID del producto más vendido.
- ID del producto menos vendido.
- ID del producto con mayor ganancia.
- ID del producto con menor ganancia.

Para una implementación general de todas las opciones antes mencionadas, utilice el despliegue de un menú (observe los ejemplos).

Con respecto al almacenamiento de los datos, se utilizará una base de datos con las siguientes tablas:

- Producto.
- Cliente.
- Compra.
- Detalle.

Para revisar las relaciones y atributos, en su cliente de base de datos cree una base de datos llamada “tarea_1_sd” y luego importe allí el archivo SQL.

Arquitectura:

La arquitectura de la aplicación es de tipo 3-Tier y consistirá en los siguientes cinco componentes:

1. Servidor de base de datos. Se usará MySQL para gestionar los datos persistentes de la aplicación.

2. Aplicación servidor. Programa que implementa los servicios Web de las APIs RESTful, recibiendo peticiones de clientes y operando sobre el servidor de base de datos para consultar o actualizar datos.
3. Cliente. Un cliente de comercio que tendrá una API RESTful para invocar los servicios que le permitan realizar operaciones como: leer y crear. Podrá haber múltiples instancias de este tipo de cliente.
4. Administrador. Un cliente que administra los datos de la aplicación mediante una API RESTful con control de acceso, incluyendo operaciones como: crear, actualizar, leer y eliminar. Existirá una única instancia de este tipo de cliente.

Especificación de las interfaces
(según tus casos de uso)

A continuación, se especifica cada *endpoint* con su método que debe implementar. Se incluye además un ejemplo para la solicitud y la respuesta.

Endpoint	**127.0.0.1:5000/api/clientes/iniciar_sesion**
method	**POST**

request	{
	"id_cliente": 890,
	"contrasena": 12345678
	}

response	{
	"acceso_valido": true
	}

Endpoint	**127.0.0.1:5000/api/compras**
method	**POST**

request	{
	"id_cliente": 123,
	"productos": [{
	"id_producto": 2,
	"cantidad": 1
	},
	{
	"id_producto": 2,
	"cantidad": 1
	},
	{
	"id_producto": 1,
	"cantidad": 3
	},
	{
	"id_producto": 3,
	"cantidad": 2
	},
	{

	"id_producto": 6,
	"cantidad": 1
	},
	{
	"id_producto": 4,
	"cantidad": 1
	}]
	}

response	{
	"id_compra": 24
	}

Endpoint	**127.0.0.1:5000/api/productos**
method	**POST**

request	{
	"nombre": "Manzana",
	"cantidad_disponible": 100,
	"precio_unitario": 150
	}

response	{
	"id_producto": 14
	}

Endpoint	**127.0.0.1:5000/api/productos**
method	**GET**

response	[
	{
	"id_producto": 14,
	"nombre": "Manzana",
	"cantidad_disponible": 100,
	"precio_unitario": 150
	},
	{
	"id_producto": 20,
	"nombre": "Jugo",
	"cantidad_disponible": 85,
	"precio_unitario": 90
	}
	]

Endpoint	**127.0.0.1:5000/api/productos/:id**
method	**PUT**

request	{
	"nombre": "Manzana",
	"cantidad_disponible": 100,
	"precio_unitario": 150
	}

response	{ "id_producto": 14

	}

Endpoint	**127.0.0.1:5000/api/productos/:id**
method	**DELETE**

response	{
	"id_producto": 14
	}

Endpoint	**127.0.0.1:5000/api/estadisticas**
method	**GET**

response	{
	"producto_mas_vendido": 20,
	"producto_menos_vendido": 20,
	"producto_mayor_ganancia": 43,
	"producto_menor_ganancia": 15
	}

Ejemplos:

Los siguientes son ejemplos de cómo un cliente o administrador debería interactuar con la aplicación a través de una terminal:

```
Bienvenido

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 1
Ingrese su id: 8080
Ingrese su contraseña: 1234567
Error, no hay ninguna coincidencia con los datos ingresados.

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 3

Hasta luego!
```

```
Bienvenido

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 1
Ingrese su id: 8080
Ingrese su contraseña: 8931441
Inicio de sesión exitoso
```

```
Opciones:
1. Ver lista de productos
2. Hacer compra
3. Salir
Ingrese una opción: 1
14;Manzana;150 por unidad;100 disponibles
20;Jugo;90 por unidad,85 disponibles

Opciones:
1. Ver lista de productos
2. Hacer compra
3. Salir
Ingrese una opción: 2
Ingrese cantidad de productos a comprar: 2
Ingrese producto 1 par id-cantidad: 14-2
Ingrese producto 2 par id-cantidad: 20-5
Gracias por su compra!
Cantidad de productos comprados: 7
Monto total de la compra: 750

Opciones:
1. Ver lista de productos
2. Hacer compra
3. Salir
Ingrese una opción: 3

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 3

Hasta luego!
```

```
Bienvenido

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 2
Ingrese contraseña de administrador: 1234
Inicio de sesión exitoso

Opciones:
1. Ver lista de productos
2. Crear producto
3. Eliminar producto
4. Ver estadísticas
5. Salir
Ingrese una opción: 2
Ingrese el nombre: pan
Ingrese la disponibilidad: 300
Ingrese el precio unitario: 100
```

```
Producto ingresado correctamente!

Opciones:
1. Ver lista de productos
2. Crear producto
3. Eliminar producto
4. Ver estadísticas
5. Salir
Ingrese una opción: 5

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 3

Hasta luego!
```

Consideraciones:

- Si un cliente para algún producto intenta comprar una cantidad que excede el *stock* disponible para tal producto, entonces ignorar ese producto dentro de la compra. En caso contrario, disminuir el *stock* del producto.
- La API debe recibir peticiones a través de la IP 127.0.0.1 y puerto 5000, mientras que el servidor de la base de datos en la IP 127.0.0.1 y puerto 3306.
- La base de datos debe llamarse "tarea_1_sd".
- El usuario que se conecta a la base de datos debe llamarse "admin" y la contraseña "12345678".
- Para conocer los formatos de entrada y salida a través de la terminal, guíese rigurosamente por los ejemplos.
- No es necesario realizar ninguna validación para los datos que se ingresan a través de la terminal, excepto los de inicio de sesión.
- La versión de MySQL es la 8.0.X.
- La versión de Go es la 1.19.X.
- Para consultas, escriba a hugo.leyton@sansano.usm.cl.

Reglas de la tarea:

- La tarea se entrega en grupos de 3 personas.
- La fecha de entrega es el día 15 de septiembre a las 23:55.
- La tarea debe ser subida a través de un repositorio en https://gitlab.inf.utfsm.cl/. Además, genere una invitación al usuario hleyton, asignando previamente un rol de *Developer*.
- El repositorio debe contener al menos dos archivos:
    - main.go para interactuar con el menú mostrado en los ejemplos.
    - server.go para habilitar el servidor local en el puerto 5000.
- En el README del repositorio, incluya el nombre y apellido de cada integrante.

- No está permitido modificar las tablas de la base de datos.

RÚBRICA

1. Aspectos generales:
   - Aprendizaje y dominio del lenguaje Go.
   - Diseño de una arquitectura orientada a servicios usando API RESTful (endpoints).
   - Documentación y calidad del código.

2. Configuración de la base de datos en MySQL
   - Correcta definición de usuarios y tablas.

3. Desarrollo de la aplicación servidor
   - Ruteo de los servicios
   - Correcta implementación de los servicios y las consultas en SQL a la base de datos.
   - Correcta verificación de las credenciales del administrador y restricción del acceso a los servicios correspondientes.

4. Desarrollo de un cliente de comercio
   - Calidad de la interfaz de presentación para el usuario
   - Correcta invocación de los servicios de la API RESTful

5. Desarrollo del cliente administrador
   - Calidad de la interfaz de presentación para el usuario
   - Correcta invocación de los servicios de la API RESTful