Trabajo Práctico 1

Parte 2 - API Restful

Una tienda de comercio contrata su servicio para desarrollar el ecommerce de la tienda, dicha web debe permitir a los clientes registrarse y llevar a cabo sus compras, asi como también permitir que el personal del local pueda ver y gestionar las ventas.

La entrega de esta app se realiza en tres etapas:

1. Entrega de prototipo con lógica de base de datos (TP 1)
2. Entrega de Backend con la lógica de negocios (TP 2)
3. Entrega de Página web (TP 3)

Por lo tanto, Los **criterios de aceptación son**:

1. Debe permitir registrar a los clientes.
2. Debe permitir ver todos los productos y realizar búsquedas por filtrado de nombre y ordenamiento por precio.
3. Debe permitir ver el detalle de un producto.
4. Debe permitir agregar un producto al carrito de compra.
5. Debe permitir modificar el carrito de compra.
6. Debe permitir eliminar un producto del carrito.
7. Debe permitir efectuar una compra.
8. Agregar un reporte de balance por día, debe devolver la recaudación del día con las ventas realizadas y sus respectivos productos.

Consigna:

Realizar una aplicación con la arquitectura API Rest, que exponga los servicios necesarios para cumplir con los criterios de aceptación.
Se deberá reescribir la aplicación de consola realizada en el TP 1 y adaptarla a los nuevos requerimientos.

- La aplicación se debe ajustar al estándar de REST. Tanto los métodos de petición como los de respuesta.
- Las URL de los endpoint deben ser las que se detallan en este documento.
- Los datos que se devuelven deben estar en formato JSON
- Se debe documentar las API con la librería con OpenApi

EndPoints:

1. **[*]** >/api/clientes

   BODY

   ```
   {
      "dni": "123456789",
      "name": "Señor",
      "lastname": "X",
      "address": "Calle falsa 123",
      "phoneNumber": "123456789"
   }
   ```

2. **[*]** >/api/productos

   QueryParams

   ```
   {
         name : EJ. Mochila
         sort : EJ. true (ASC = true | DESC = false)
   }
   ```

3. **[GET]** >/api/productos/:id
4. **[*]** >/api/carrito

   Body

   ```
   {
      "clientId": 123,
      "productId": 123,
      "amount": 123
   }
   ```

5. **[*]** >/api/carrito

   Body

   ```
   {
      "clientId": 123,
      "productId": 123,
      "amount": 123
   }
   ```

6. **[DELETE]** >/api/carrito/:clientId/:productId
7. **[POST]** >/api/orden/:clientId
8. **[GET]** >/api/orden

   QueryParams

   ```
   {
         from: 2022-09-19 (Fecha desde)
         to: 2022-09-20 (Fecha hasta)
   }
   ```

Aclaración:

- Las url, parámetros y body definidos previamente deben respetarse.
- [*] : el método HTTP a utilizar en este endpoint debe ser seleccionado por el estudiante según el estándar REST. - En la collection de Postman [Recursos] los request que tienen método “LOCK” son los que deben seleccionar
- Las respuestas de las Apis deben respetar el estándar de REST y la información que presenta deben seleccionarlo el alumno.

Entrega:
El código debe ser subido al campus virtual o enviado por mail al docente en formato ZIP o RAR con el nombre: “TP1-REST-Apellido_Nombre”

Pautas de evaluación:

1. **Criterios de aceptación [4 pts]**
2. **Calidad del código [4 pts]**
    a. **Clean Code**
    b. **Clean Architecture**
    c. **Solid**
    d. **Funcionalidad y usabilidad**
3. **Estándares API Restful [2 pts]**

RECURSOS:

1. Collection de Postman: https://drive.google.com/file/d/1Sb0BzVxzGKzmeUCqnbm2pH3JjUbECkJN/view?usp=sharing
2. Tutorial API Net Core: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio
3. Tutorial API Spring Boot: https://www.nigmacode.com/java/Crear-API-REST-con-Spring
4. Tutorial API Flask: https://flask-restful.readthedocs.io/en/latest/quickstart.html