ST0246 BASES DE DATOS

Profesor: Edwin Montoya, [emontoya@eafit.edu.co](mailto:emontoya@eafit.edu.co)

2022-2

(versión 1.0)

Fecha máxima de entrega del proyecto: 24 de noviembre de 2022

Reto3 final:

Objetivo:

* Realizar el diseño de una base de datos relacional para un sitio web de ecommerce, de venta de diferentes tipos de productos y entrega en domicilio.
  + Entregable actualizado del Modelo Entidad-Relación
  + Entregable actualizado del diseño lógico de la base de datos (definición de tablas y relaciones -> entregar el diseño mysql-workbench de la ingeniería inversa).
* Realizar la implementación de la base de datos, con tablas y registros en el servidor de amazon.
  + Entregable actualizado de la definición de las tablas e ingreso de registros ejemplo: reto3ecommerce.sql.
  + Base de datos actualizada e implementada en mysql en el servidor ec2 aws en nube (volver a entregar la IP elastica y vockey.pem)
* Realizar el diseño e implementación de los servicios web mediante API REST de diferentes operaciones y transacciones en el sitio de ecommerce.

Diseño e implementación de servicios web mediante API REST de operaciones y transacciones del sitio web de ecommerce:

Realizar:

1. El diseño de los servicios Web mediante API REST, para ello deberá especificar y responder las siguientes preguntas:
   1. Definir las n consultas que realizará sobre el sistema de ecommerce, deberá definir AL MENOS 10 CONSULTAS sobre productos, ordenes, carrito, facturas, clientes, etc.

* **Consulta 1:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | consultar todos los productos |
| METODO: | GET |
| URI: | URI:/Productos/nombre\_producto |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos?nombre\_producto |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 2:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar el stock de cada producto |
| METODO: | GET |
| URI: | URI: /Productos/stock |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos?stock |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 3:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar la descripción de los productos que son perfumes |
| METODO: | GET |
| URI: | /Productos/nombre\_producto/descripcion\_producto |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos/nombre\_producto/descripcion\_producto?nombre\_producto=”Perfume” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 4:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar las categorías |
| METODO: | GET |
| URI: | /Categoria/nombre\_categoria |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Categoria?nombre\_categoria |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 5:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar la descripción de la categoría de mujer y hombre |
| METODO: | GET |
| URI: | /Categoria/nombre\_categoria/descripción\_categoria |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Categoria/nombre\_categoria/descripción\_categoria?nombre\_categoria=”Hombre”&nombre\_categoria=”Mujer” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 6:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar el precio de los productos que sean “jean” |
| METODO: | GET |
| URI: | /Productos/nombre\_producto/precio\_producto |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos/nombre\_producto/precio\_producto?nombre\_producto=”Jean” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 7:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar la descripción del producto con id = “29697311” |
| METODO: | GET |
| URI: | /Productos/sku/descripcion\_producto |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos/sku/descripcion\_producto?sku=”29697311” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 8:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar el nombre del producto con precio = “2.649.900” |
| METODO: | GET |
| URI: | /Productos/precio\_producto/nombre\_producto/ |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos/precio\_producto/nombre\_producto?precio\_producto=”2.649.900” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 9:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar el id de los productos de nombre “Camiseta” |
| METODO: | GET |
| URI: | /Producto/nombre\_producto/id |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos/nombre\_producto/sku?nombre\_producto=”Camiseta” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* **Consulta 10:**

|  |  |
| --- | --- |
| # Breve descripción de la consulta | Consultar el stock del producto con id = “0123456789012” |
| METODO: | GET |
| URI: | /Producto/id/stock |
| Parámetros:  ya sean con formato ….. uri/:parametro o con formato uri?var1=val&var2=val&etc | localhost:3000/Productos/id/stock?id=”0123456789012” |
| Datos y formato de respuesta:  Normalmente nodejs formatea los datos de respuesta en JSON, pero debe colocar uno o dos ejemplos de datos de salida. |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* 1. Al menos un CRUD (Create, Retrieve, Update, Delete) para al menos una tabla maestra de productos o clientes (menos Retrieve que está en el numeral 1.1)

Seleccionar la tabla: COSTUMERS

Definir al menos 3 API REST para este CRUD así:

|  |  |
| --- | --- |
| CREAR REGISTRO | Permite guardar un nuevo documento en la tabla ‘customer’ |
| METODO: | POST |
| URI: | /customer/create |
| Parámetros en el BODY formato JSON. |  |
| Datos y formato de respuesta: |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

|  |  |
| --- | --- |
| ACTUALIZAR REGISTRO | Permite actualizar un documento que coincida con la ‘id’ dada |
| METODO: | PUT |
| URI: | /customer/update/:id |
| Parámetros en el BODY formato JSON. |  |
| Datos y formato de respuesta: |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

|  |  |
| --- | --- |
| BORRAR REGISTRO | Permite eliminar un documento que coincida con la ‘id’ dada |
| METODO: | DELETE |
| URI: | /customer/delete/:id |
| Parámetros en el BODY formato JSON. |  |
| Datos y formato de respuesta: |  |
| Alguna restricción o lógica de negocio que se requiera en la consulta: |  |

* 1. transacciones:

Realizar el diseño del API rest para las siguientes transacciones:

* + 1. login
    2. logout
    3. crear\_carrrito (un cliente solo puede tener un carrito, se recomienda crear una tabla aparte de carrito)
    4. adicionar producto al carrito (tener en cuenta las reglas para poder adicionar un producto al carrito)
    5. consultar productos de un carrito (debe estas definida en la sección 1.1)
    6. modificar datos de un producto en el carrito
    7. borrar un producto del carrito
    8. realizar la compra (en este caso, se debe crear la orden, actualizar todos los productos y cantidades para esta orden, realizar el borrado de los productos del carrito, PERO ojo: mantener la relación de los productos ya con esta nueva orden de compra.
    9. Una transacción que considere que falta o una definida por el grupo
    10. Otra transacción opcional – por si quiere retarse, que sea compleja.

Realizar el diseño en las siguientes tablas:

|  |  |
| --- | --- |
| 1.3.1 login | Permite ingresar a la api usando el nombre y la contraseña ya registradas |
| METODO: | GET |
| URI: | /login |
| Parámetros: |  |
| Datos y formato de respuesta: |  |
| Restricciones: |  |

|  |  |
| --- | --- |
| 1.3.2 logout | Permite cerrar la sesión solo escribiendo la URI |
| METODO: | GET |
| URI: | /logout |
| Parámetros: |  |
| Datos y formato de respuesta: |  |
| Restricciones: |  |

|  |  |
| --- | --- |
| 1.3.3 crear carrito | Crear nuevo carrito |
| METODO: | ? |
| URI: | /Kart/CreateKart/:CustomerId |
| Parámetros: | http://localhost:3000/Kart/Createkart/:CustomerId |
| Datos y formato de respuesta: |  |
| Restricciones: | res.status(500).send("No confirmaste la creacion del carrito") |

|  |  |
| --- | --- |
| 1.3.4 adicionar producto al carrito | agregar un producto al carrito |
| METODO: | ? |
| URI: | Kart/DeleteProduct/:CustomerKartId |
| Parámetros: | http://localhost:3000/Kart/DeleteProduct/:CustomerKartId |
| Datos y formato de respuesta: |  |
| Restricciones: | res.status(500).send("Error eliminando") |

|  |  |
| --- | --- |
| 1.3.5 consultar productos del carrito | actualizar cantidad de un producto en el carrito |
| METODO: | ? |
| URI: | /Kart/UpdateQuantity/:CustomerKartId |
| Parámetros: | http://localhost:3000/Kart/UpdateQuantity/:CustomerKartId |
| Datos y formato de respuesta: |  |
| Restricciones: | res.status(500).send(err) |

|  |  |
| --- | --- |
| 1.3.6 modificar datos de un producto en el carrito | <descripción> |
| METODO: | ? |
| URI: |  |
| Parámetros: |  |
| Datos y formato de respuesta: |  |
| Restricciones: |  |

|  |  |
| --- | --- |
| 1.3.7 borrar un producto del carrito | <descripción> |
| METODO: | ? |
| URI: | /Kart/DeleteProduct/:CustomerKartId |
| Parámetros: | http://localhost:3000/Kart/DeleteProduct/:CustomerKartId |
| Datos y formato de respuesta: |  |
| Restricciones: | res.status(500).send("Error eliminando") |

|  |  |
| --- | --- |
| 1.3.8 realizar la compra | <descripción> |
| METODO: | ? |
| URI: | / Kart/MakePurchase1/:CustomerKartId |
| Parámetros: | http://localhost:3000/Kart/MakePurchase1/:CustomerKartId |
| Datos y formato de respuesta: |  |
| Restricciones: | res.status(500).send("No agregaste algun campo requerido") |

|  |  |
| --- | --- |
| 1.3.9. Una transacción que considere que falta o una definida por el grupo -> registro | Le permite al usuario registrarse con su nombre y contraseña, verifica que no exista alguien ya registrado con el mismo nombre y por ultimo envía a la parte de login |
| METODO: | POST |
| URI: | // URI: /register |
| Parámetros: |  |
| Datos y formato de respuesta: |  |
| Restricciones: |  |

1. Realizar la implementación de todos los métodos API REST diseñados en el numeral anterior, implementados en nodejs y mysql.

Todos los métodos deben ser implementados en el servidor desplegado en amazon, con la IP elástica y clave vockey.pem, recordar abrir el puerto 3000 en el Security Group de la máquina virtual en amazon.

Cada estudiante debe crear un repositorio github para entregar el reto3ecommerce, con todos los códigos fuente, y toda la especificación de los servicios API REST.

Ver ejemplos de códigos similares en:

Basado en los ejemplos de las aplicaciones en el repositorio github de la materia:

* <https://github.com/st0246eafit/st0246-2266.git>
* Ejemplo básico nodejs + mysql de tienda de productos:
  + <https://github.com/st0246eafit/st0246-2266/tree/main/nodejs-mysql-tienda-productos>
* Ejemplo base para el reto3final: ndoejs+mysql:
  + <https://github.com/st0246eafit/st0246-2266/tree/main/nodejs-mysql-reto3-ecommerce>