Buenas tardes

Para esta prueba se propondra una situacion la cual seria resulta a traves de una API, que satisfaga los casos de uso descritos mas adelante, cabe destacar que para la solucion de dicho problema NO se espera contar con un UI que implemente la API solicitida

Situacion propuesta :

Daniela, tiene una tienda de barrio que con los años ha crecido, actualmente cuenta con 5 empleados, y esta presentando problemas al hacer las cuentas finalizando los dias de trabajo, en donde mas ha presentado problemas es en el cruce de las ventas que se realizaron en el dia junto al dinero que deberia estar al final del dia.

Actualmente Daniela y sus empleados, cada que realizan una venta escriben cual fue el producto que se vendio y valor correspondiente a dicho producto; ocasionando que uno de los problemas mas frecuentes sea que algun empleado o ella misma escriban mal el producto o su costo.

Requerimientos :

- Administrar las ventas

- Administrar los productos

- Administrar los usuarios

- Hacer cierres diarios

- Hacer un balance mensual

Como requerimiento para la aplicacion habran ciertas urls que cubriran los casos de uso descritos mas adelante, hay que tener en cuenta que en los casos de uso se especifica el tipo de usuario que tiene permitido realizar cada operacion.

Para satisfacer que cada usuario solo tenga acceso a las urls que se requieren se debe enviar el tipo de usuario en un header 'Auth: <id\_usuario>', el servidor web debe poder tomar esta info y determinar si el usuario tiene o no acceso a esta informacion

retornando codigos http 401 o 403 segun sea el caso.

Ejemplo de solicitud con auth header

HTTP FORMAT

GET /api/v1/sales HTTP/1.1

Host: localhost:3000

Auth: 123456789

CURL

curl --location --request GET 'localhost:3000/api/v1/sales' --header 'Auth: 123456789'

Tipos de usuario que tienen permitido hacer las operaciones, basado en el rol que tenga el usuario

admin = (\*)

employee = (+)

everyone = (')

Casos de uso

- Administrar las ventas (Por facilidad una compra siempre va a implicar a solo un producto)

(') Crear venta [5pt]

(\*) Actualizar venta [5pt]

(\*) Eliminar venta [2pt]

(') Listar ventas [1pt]

- Administrar los productos

(\*) Crear productos [5pt]

(') Listar productos [5pt]

- Administrar los usuarios

(\*) Crear usuarios [5pt]

(\*) Listar usuarios [1pt]

(\*) Borrar usuarios [3pt]

(\*) Asignarle un rol a un usuario [3pt]

(\*) Crear roles [5pt]

- Hacer cierres diarios

(\*) Entregar el valor total de ventas [5pt]

en un dia especifico

- Hacer un balance mensual

(\*) Entregar el valor total de ventas en un mes [5pt]

----------------------------------------------------------

Total [50pt]

La base de datos que se usara para la resolucion de este problema, sera PostgresSQL 12 y el diagrama se entrega adjunto en el presente correo

La base de datos debe ser inicializada con los productos adjuntos en el archivo products.json

#CONSIDERACIONES DE LA PRUEBA

Criterios de evaluacion:

manejo del lenguaje 3pt

manejo del framework web 3pt

manejo del repositorio (hacer commits para cada caso de uso) 3pt

uso de herramientas externas (postman/swagger) 1pt

cumplimiento de requerimientos 5 pt

patrones de diseno de la aplicacion (se sugiere clean arquitecture con SOLID) 5pt (adicionales)

Puntaje maximo (Sin adicionales): 15

Opcional: Puedes dejar un Dockerfile para deplegar tu aplicacion aplicacion en un contenedor.

Al final se requiere que en el repositorio (urls a repositorio publico bit bucket, gitlab o github)

Se agregue una coleccion exportada desde postman con las diferentes urls que contiene tu servicio o el archivo oas en el caso de usar swagger,

Tambien se deben agregar las diferentes sentencias DDL y DML que se haya usado para la creacion de la Base de datos y para creacion de los productos

La fecha máxima de entrega es el martes 21 de junio a las 08:00 am.

Exitos