FINNEGANS

Construcción Backend del Pedido de Venta V1

Sprint deadline	03/11/2023
Tarjeta	SCRUM-23
Responsable	Iván Bilvinas

Tabla 1 Detalle de la tarjeta correspondiente en Jira

1. Objetivos. Contenido de la tarjeta

- Controller: Definir API.
- Model:
  - Definir constraints.
  - Definir DTO para request y response.
- Service:
  - Validación de información recibida.
  - Validaciones previas a interactuar con la BD.
  - Lógica de negocio.
- IRepository.

FINNEGANS

Dependencias

En primer lugar, dependeremos de las siguientes tarjetas:

- SCRUM-21 – Construcción Backend de Productos / Servicios.
- SCRUM-12 – Construcción Backend de Usuarios.
- SCRUM-20 – Construcción Backend de Clientes.

En segundo lugar, dependerán de esta tarjeta las siguientes tarjetas:

- SCRUM-19 – Construcción Frontend de Pedido de Venta V1

Entregable

- API bajo la especificación OpenAPI.
- Migración de la base de datos completa para la tabla Order.
- Migración de la base de datos completa para la tabla ServiceDetail.
- Migración de la base de datos completa para la tabla ProductDetail.
- Método de trabajo reutilizable en los próximos endpoints que el proyecto requiera.
- Definición de los DTOs para Request y Response de la entidad Order.
- Definición de los DTOs para Request y Response de la entidad ServiceDetail.
- Definición de los DTOs para Request y Response de la entidad ProductDetail.
- Fábrica para la creación de los DTO y entidades Order.
- Fábrica para la creación de los DTO y entidades ServiceDetail.
- Fábrica para la creación de los DTO y entidades ProductDetail.
- Mecanismo que permita informar un conjunto de errores encontrados al procesar una solicitud (y que minimice el acceso a la base de datos).
- Set granular de excepciones con su correspondiente mapeo a los códigos de respuesta HTTP.
- Sistema diseñado con un mecanismo de bajas lógicas para conservar información de las ordenes eliminadas.
- Endpoins funcionales con todos los puntos detallados anteriormente para: “Registrar orden”, “Listar todas las órdenes”, “Eliminar una orden”.

FINNEGANS

Modelos

Se establecieron tres modelos necesarios.

1. **Order**.
    - Id: Entero.
    - Date: String.
    - Client: Client.
    - User: User.
    - Enabled: Boolean.
    - ServiceDetails: List<ServiceDetail>.
    - ProductDetails: List<ProductDetail>.
    - Total: Decimal.

   Relación con Id: Cada orden tendrá un Id único, generado automáticamente cuando se crea.

   Relación con Date: Cada orden tendrá una fecha con formato “dd/MM/yyyy”, generado automáticamente cuando se crea.

   Relación con Client: Un cliente tendrá muchas ordenes, pero cada orden podrá tener solo un cliente.

   Relación con User: Un usuario podrá cargar muchas ordenes, pero cada orden deberá ser cargada por solo un usuario.

   Relacion con Enabled: Será true o false, en un principio será true, hasta que se deshabilite la orden.

   Relación con ServiceDetails: Una orden podrá tener muchos detalles de servicio guardados en una lista, y cada detalle de servicio pertenecerá a una orden.

   Relación con ProductDetails: Una orden podrá tener muchos detalles de producto guardados en una lista, y cada detalle de producto pertenecerá a una orden.

   Relación con Total: El total se calculará mediante la función “calculateTotal”, donde iteraremos cada Lista de ProductDetail y ServiceDetail sumando los respectivos subtotales.

2. **ServiceDetail**.
    - Id: Entero.
    - TaxesApplied: String.
    - TaxCharges: Decimal.
    - Subtotal: Decimal.
    - Order: Order.
    - Service: Service.

Relacion con Id: Cada detalle de servicio tendrá un Id único.

Relación con TaxesApplied: Cada detalle de servicio tendrá un String que provenga de la entidad Servicio relacionada, que contenga los Impuestos que se le aplican al mismo.

Relacion con TaxCharges: Sera un porcentaje según los impuestos que se le aplican.

Relacion con Subtotal: En esta V1 de pedido de venta, el subtotal solo estará compuesto por el atributo Precio Base heredado del servicio relacionado.

Relación con Order: Cada detalle de servicio pertenecerá a una orden.

Relación con Service: Cada detalle de servicio pertenecerá a un servicio.

3. **ProductDetail**.
    - Id: Entero.
    - TaxesApplied: String.
    - TaxCharges: Decimal.
    - Quantity: Entero.
    - Warranty: Decimal.
    - Subtotal: Decimal.
    - Order: Order
    - Product: Product.

Relacion con Id: Cada detalle de producto tendrá un Id único.

Relacion con TaxesApplied: Cada detalle de producto tendrá un String que provenga de la entidad Producto relacionada, que contenga los Impuestos que se le aplican al mismo.

Relacion con TaxCharges: Sera un porcentaje según los impuestos que se le aplican.

Relacion con Quantity: Sera la cantidad de producto que se compra, llegará desde la petición que hará el Frontend.

Relacion con Warranty: Serán los años de garantía.

Relación con Subtotal: En esta V1 de pedido de venta, el subtotal estará compuesto por una multiplicación entre Precio base del producto y la cantidad comprada.

Relacion con Order: Cada detalle de servicio pertenecerá a una orden.

Relacion con Product: Cada detalle de servicio pertenecerá a un producto.

DTOs.

1. Order.
    - OrderRequest
        - ClientId: Integer.
        - Services: List<ServiceDetailRequest>
        - Products: List<ProductDetailRequest>
    - OrderResponse
        - Id: Integer.
        - Date: String.
        - Total: Decimal.
        - Client: ClientResponse.
        - Services: List<ServiceDetailResponse>
        - Products: List<ProductDetailResponse>
2. ServiceDetail.
    - ServiceDetailRequest.
        - ServiceId: Integer
    - ServiceDetailResponse.
        - Id: Integer.
        - ServiceId: Integer.
        - Subtotal: Decimal.
        - TaxesApplied: String.
        - TaxCharges: Decimal.
3. ProductDetail.
    - ProductDetailRequest.
        - ProductId: Integer.
        - Quantity: Integer.
        - Warranty: Decimal.
    - ProductDetailResponse.
        - Id: Integer.
        - ProductId: Integer
        - Quantity: Integer.
        - Subtotal: Decimal.
        - TaxesApplied: String.
        - TaxCharges: Decimal.

FINNEGANS

Controladores

Se establecieron las siguientes rutas.

- Petición POST a /order

En esta petición se registrará la orden en la Base de Datos, al igual que sus respectivos ServiceDetail y ProductDetail.

- Petición GET a /order/list

En esta petición se retornará todas las ordenes que se encuentren en la Base de Datos.

- Petición GET a /order/{id}

En esta petición se retornará la orden que cumpla con el Id que se recibe por parámetros.

- Petición DELETE a /order/{id}

En esta petición se hará un borrado lógico de la orden, deshabilitándola, poniendo su atributo “enabled” en false.

Servicios

Order.

- Función “registerOrder”, se la invocará en el caso de la petición POST a /order. Recibirá dos parámetros: una OrderRequest y un nombre de usuario.
    1) Validaremos los parámetros.
        - Servicios y productos. Vamos a validar que las listas de servicios y productos que nos llegan del OrderRequest no estén vacías.
        - Nombre de usuario. Validaremos que sea un usuario existente en la Base de Datos
        - Cliente. Vamos a validar que el ClientId que nos llega por la OrderRequest represente a un cliente verdadero y que se encuentre en la Base de Datos.
    2) Crearemos la Orden y le insertaremos el cliente, usuario, detalles de servicios y detalles de productos.
    3) Guardaremos la Orden en la Base de Datos.
    4) Retornaremos la OrderResponse.
- Función “get” se la invocará en el caso de la petición GET a /order. Recibirá un parámetro: Id.
    1) Validaremos que el id enviado por parámetro represente a una orden verdaderamente creada y almacenada en la base de datos.
    2) Retornaremos la OrderResponse de la orden encontrada.

- Función “get” se la invocará en el caso de la petición GET a /order/list. No recibirá parámetros.
    1) Guardaremos en una lista de ordenes todas las ordenes encontradas en Base de Datos.
    2) Validaremos que la lista no esté vacía.
    3) Iteraremos la lista convirtiendo cada orden en una OrderResponse.
    4) Retornaremos una Lista de OrderResponse.
- Función “delete” se la invocará en el caso de la petición DELETE a /order/list. Recibirá por parámetro un Id.
    1) Validaremos que el id enviado por parámetro represente a una orden verdaderamente creada y almacenada en la Base de Datos.
    2) Haciendo uso del borrado lógico, no eliminaremos la orden, sino que modificaremos su atributo “enabled” a false.
    3) Retornaremos una OrderResponse de la orden modificada.