FINNEGANS

Engine de cálculo de Pedido

Sprint deadline	15/10/2023
Tarjeta	SCRUM-39
Responsable	Martín Paez

Tabla 1 Detalle de la tarjeta correspondiente en Jira

1. Objetivos. Contenido de la tarjeta

- Implementar la lógica para calcular los subtotates de los items y el monto total del pedido en el servicio de ordenes. Tener en cuenta:
    - Impuestos
    - Garantía
    - Descuento
- Implementar el endpoint para registrar una nueva orden debe computar y retornar en la respuesta un resumen con los impuestos, garantías y descuentos aplicados.

2. Dependencias

La tarjeta presentada en este informe, “SCRUM-39: Engine de cálculo de Pedido” depende de la tarjeta “SCRUM-35 Backend de Asignación de servicios a clientes”, ya que debe destacar el servicio que generó el descuento.

Además, hay una dependencia con la tarjeta “SCRUM-43 Implementación patrón State Pedidos de venta V2”, ya que una vez tomado el pedido hay que asignarle un estado a la orden.

3. Entregable

- “Engine de cálculo” completo que cumpla la totalidad de las reglas de negocio detalladas en el enunciado.

1. Procedimientos

1.1. Manejo de las dependencias

Respecto a la dependencia con la tarjeta “SCRUM-35 Backend de Asignación de servicios a clientes”, se definió, de antemano y en equipo, la firma de un método que recupera un servicio a partir de un cliente, con lo cual se planeó entregar comentada la invocación a dicho método con la intención de que la misma sea descomentada manualmente después del merge.

En relación a la otra dependencia mencionada, o sea, la tarjeta “SCRUM-43 Implementación patrón State Pedidos de venta V2”, hay que mencionar que no es de carácter prioritario para el tercer Sprint, ya que es una característica que agregamos durante la etapa de diseño para cumplir con la solicitud de emplear al menos tres patrones de diseño.

Se plantearon dos escenarios hipotéticos: el primero, en caso de llegar a implementar la tarjeta SCRUM-43 antes del cierre del sprint, el procedimiento sería equivalente al empleado para la dependencia antes mencionada. El segundo, en caso de no incorporar para este sprint la funcionalidad completa de la tarjeta SCRUM-43, se utilizará una baja lógica para la orden estableciendo dos estados: “pagada”, ni bien se registra la orden, o “cancelada”, empleando un “endpoint delete” para persistir la acción.

1.1. Asignación de responsabilidades

Lo primero que hay que mencionar, es que la lógica de aplicación se implementó en el servicio de orden, en su totalidad. Si bien, en determinadas ocasiones, no está mal agregar lógica a las Entites, se prefirió que su responsabilidad única sea la de modelar la tabla de la base de datos. Por otro lado, la responsabilidad de implementar la lógica de negocio es de los servicios, con lo cual el servicio que recibe la orden para ser persistida es quien realiza el cómputo; o, en su defecto, informa que no se pudo registrar la orden mediante un mecanismo de excepciones, que luego serán mapeadas al código de estado HTTP según corresponda.

1.2. Cálculo a partir de factores

Al confeccionar la fórmula de cálculo se planteó resolver las características en función del cálculo de factores, o sea, valores numéricos que se multiplicarán al valor computado. La idea básica es sacar provecho de la propiedad conmutativa de la multiplicación, o sea, “el orden de los factores no altera el producto”.

De este modo no hay confusiones respecto a que sucedería si se computa primero una garantía, o un descuento, etc. Lo cual permite, en consecuencia, simplificar la interpretación del enunciado, ya que, por ejemplo, si el enunciado dijera que se aplica tal característica a tal entidad, la frase “a tal entidad” no sería un condicionante tan restrictivo. Al final de cuentas lo importante es que el cálculo sea correcto, y para ello es posible emplear equivalencias matemáticas.

Lo más importante de plantear el diseño desde este punto de vista, más allá de la gran ventaja a la hora de interpretar y aunar criterios, es que brindó flexibilidad para distribuir las responsabilidades de cada pieza de código.

Por último, mencionar que lo planteado en esta sección no fue una regla, si no un punto de vista a partir del cual se comenzó a diseñar. Por tal motivo, donde resultó conveniente se terminaron aplicando operaciones de suma o resta, siempre que ello ayudara a organizar el código de un modo más claro.

1.3. DiscountService

Se llamo discountService al servicio que fue asignado como aquel que generó el descuento. El criterio empleado fue seleccionar el primero de los servicios que figuran en el pedido, o, en caso de no haberlo, será el servicio retornado por la implementación antes mencionada, o sea, aquella de la tarjeta “SCRUM-35 Backend de Asignación de servicios a clientes”.

1.4. Tareas de refactoring

Antes de comenzar a escribir las líneas de código correspondientes a los objetivos de esta tarjeta, se realizaron tareas de refactoring sobre el código del servicio de ordenes preexistente.

Es importante destacar que el código a partir del cual se comenzó a trabajar estaba bien planteado, incluso mejor de lo que se hubiera esperado considerando que fue escrito por miembros del equipo que no tenían conocimientos sobre la tecnología en cuestión al comenzar el bootcamp.

Debido a que el “Engine de cálculo”, y, el “Pedido” en si mismo, adquirieron un carácter central en el desarrollo a partir de la opinión general del equipo, se tomó la decisión de aportar estas tareas de refactoring al comienzo del sprint.

1.5. Representación de la information

Luego de la Review del tercer Sprint, fue necesario realizar modificaciones en el Diagrama Entidad Relación planteado en el primer Sprint. El motivo fue que confundimos los requerimientos de los tutores, se debía calcular el importe que cada impuesto generó, no por producto o servicio (como se planificó en un principio), si no, el acumulado para todos los items de la orden que tuvieran asignado ese impuesto en particular. Para ello fue necesario crear un modelo más con sus DTO, fábrica, repositorio y servicio correspondientes. Además, hubo que realizar modificaciones en el engine de cálculo, en el DTO respuesta para el registro de una orden, y, por consiguiente, adaptar el frontend para mostrar la información como se solicitaba (esta última tarea excede el alcance del presente informe).

1.6. Método de cálculo

Para dar mayor claridad, se presenta el método para calcular el total a partir del cual se basó el desarrollo. Sin embargo, hay que tener en cuenta que, a la hora de la implementación se adaptó este método empleando equivalencias matemáticas para contemplar cuestiones tales como el cómputo de información derivada, irrelevante para lo que se busca describir en esta sección.

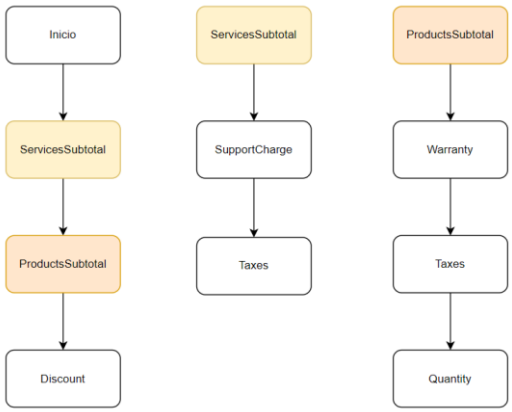


Ilustración 1 Diagrama de bloques simplificado. La intención es esquematizar únicamente el orden en que se agrega cada característica durante el proceso de cálculo.

El método comienza por realizar las conversiones que permitan para partir de la premisa de que:

$$PrecioBase > \$0$$

$$CargoSoporte >= \$0$$

$$0 < Impuesto <= 1$$

$$añosGarantía >= 0\ (Podría, incluso, ser\ un\ número\ Real)$$

$$0 <= garantíaAnual <= 1\ (Podría\ ser\ mayor\ a\ 1)$$

$$factorDescuento = 0{,}2$$

$$\text{maxDescuento} = \$2500$$

$$cantidad \geq 0\ \ , cantidad \in Z$$

Luego, se calculan los factores adimensionales para impuestos y garantías:

$$factorImpuesto = 1 + \sum_{1}^{n} impuesto_i$$

$$factorGarantía = 1 + añosGarantía * garantíaAnual$$

A continuación, se computan los subtotales en unidad monetaria usando las siguientes expresiones para cada uno de los item:

$$subtotalServicio = (\text{PrecioBase} + \text{CargoSoporte}) * \text{factorImpuesto}$$

$$subtotal\text{Producto} = \text{PrecioBase} * \text{factorGarantía} * \text{factorImpuesto} * \text{Cantidad}$$

Por último, se computa el descuento en unidad monetaria y se lo debita del total:

$$descuentoProductos = \min\{\text{factorDescuento} * \sum_{1}^{n} subtotalProducto_i, \text{maxDescuento}\}$$

$$\text{Total} = \sum_{1}^{n} subtotalServicio_i + \sum_{1}^{n} subtotalProducto_i - descuentoProductos$$

4. Resultados

4.1. Ejemplo usando Postman

```
select * from client;
```

id	adress	bussiness_name	cuit	dni	enabled	is_bussiness	last_name	name	phone	start_date
1	calle siempre viva	x.srl	27137460350	46013734	1	1	paez	diego	43512226	2023-10-30 21:00:00.0000000

Ilustración 2 Captura de pantalla de Azure de la consulta para obtener los clientes registrados. En este caso Diego Paez.

```
select * from taxes_by_services;
```

id	tax_id
1	1
1	2

Ilustración 3 Captura de pantalla de Azure de la consulta para obtener los impuestos aplicados a los servicios. Se observa que se aplican dos impuestos al mismo servicio.

```
select * from taxes_by_products;
```

id	tax_id
1	1
1	2

Ilustración 4 Captura de pantalla de Azure de la consulta para obtener los impuestos aplicados a los productos. Se observa que se aplican dos impuestos al mismo producto.

```
SELECT * from tax;
```

tax_id	enabled	name	percentage
1	1	IIBB	10
2	1	IVA	21

Ilustración 5 Captura de pantalla de Azure de la consulta para obtener los impuestos registrados. Se observa IVA del 21% eh IIBB de 10%, ambos habilitados.

```
SELECT * from service_table;
```

id	base_price	description	enabled	name	is_special	suport_charge
1	100	Un servicio de $100	1	Service 100	1	10

Ilustración 6 Captura de pantalla de Azure de la consulta para obtener los servicios ofrecidos. Se observa que hay uno solo, es un servicio especial y este habilitado.

```
SELECT * from product_table;
```

id	base_price	description	enabled	name
1	100	Un producto de $100	1	Producto 100

Ilustración 7 Captura de pantalla de Azure de la consulta para obtener los impuestos aplicados a los productos. Se observa que hay un producto habilitado con precio base de $100.

```
select * from subscription;
```

	id	is_enabled	client_id	service_id

Ilustración 8 Captura de pantalla de Azure de la consulta para obtener las suscripciones de todos los clientes. Se observa que no hay ningún cliente suscripto a ningún servicio.

Ilustración 9 Captura de pantalla del Postman mostrando la información enviada al servidor para registrar un pedido. Se esta solicitando una suscripción a un servicio y un producto con garantía de 5 años.

Ilustración 10 Captura de pantalla de Postman de una porción la respuesta obtenida del servidor al registrar la orden. Se observa que la suscripción es, como se esperaba, para el servicio que se acaba de contratar.

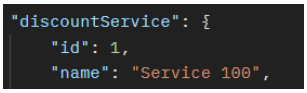


Ilustración 11 Captura de pantalla de Postman de una porción de la respuesta obtenida del servidor al registrar la orden. Se observa el descuento fue adjudicado al servicio que se acaba de contratar.

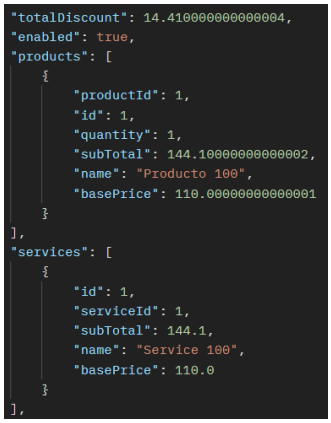
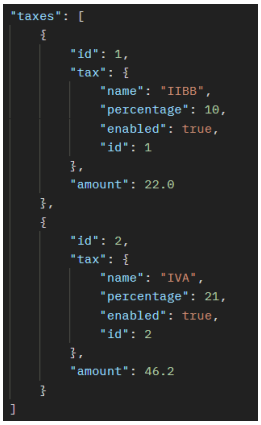
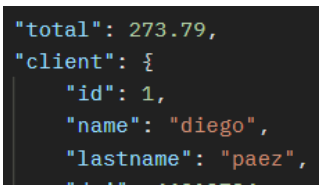


Ilustración 12 Captura de pantalla de Postman de una porción de la respuesta obtenida del servidor al registrar la orden. Se observa que el descuento total es del 10% del subtotal de productos. Además, los subtotales son correctos y contemplan, cargo de soporte, garantía, cantidades, eh impuestos (IVA 21% eh IIBB 10%).



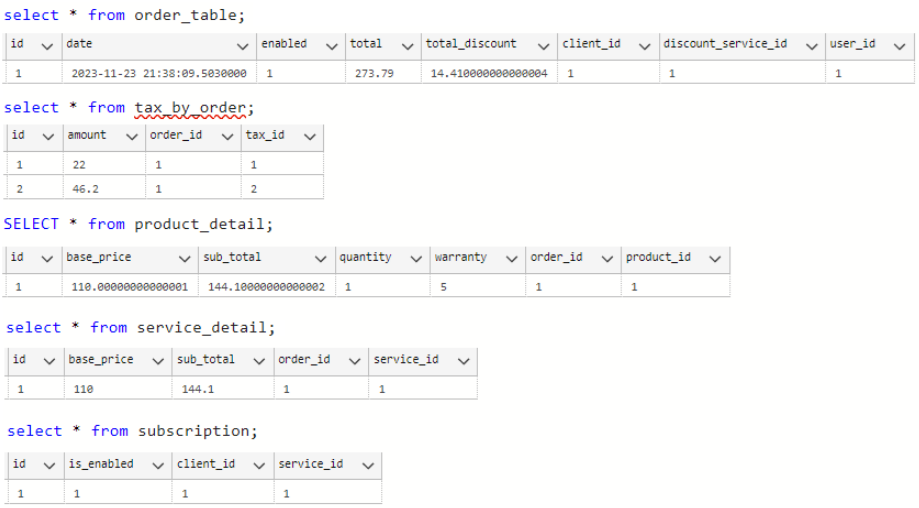
```
"taxes": [
    {
        "id": 1,
        "tax": {
            "name": "IIBB",
            "percentage": 10,
            "enabled": true,
            "id": 1
        },
        "amount": 22.0
    },
    {
        "id": 2,
        "tax": {
            "name": "IVA",
            "percentage": 21,
            "enabled": true,
            "id": 2
        },
        "amount": 46.2
    }
]
```


*Ilustración 13 Captura de pantalla de Postman de una porción de la respuesta obtenida del servidor al registrar la orden. Se observa que el monto de los impuestos es el esperado: el 10% sobre $110 para ambos casos hacen $22 de IIBB. Los mismo para el IVA = 110 * 0.21 * 2.*



```
"total": 273.79,
"client": {
    "id": 1,
    "name": "diego",
    "lastname": "paez",
```


Ilustración 14 Captura de pantalla de Postman de una porción de la respuesta obtenida del servidor al registrar la orden. Se destaca que el total fue bien calculado.



```
select * from order_table;
```

id	date	enabled	total	total_discount	client_id	discount_service_id	user_id
1	2023-11-23 21:38:09.5030000	1	273.79	14.410000000000004	1	1	1

```
select * from tax_by_order;
```

id	amount	order_id	tax_id
1	22	1	1
2	46.2	1	2

```
SELECT * from product_detail;
```

id	base_price	sub_total	quantity	warranty	order_id	product_id
1	110.00000000000001	144.10000000000002	1	5	1	1

```
select * from service_detail;
```

id	base_price	sub_total	order_id	service_id
1	110	144.1	1	1

```
select * from subscription;
```

id	is_enabled	client_id	service_id
1	1	1	1

Ilustración 15 Captura de pantalla del Azure. Se observa que los valores recibido en Postman, que son justamente los valores esperados,impactaron correctamente en la base de datos.

2. Conclusión

Se implementó el engine de cálculo en el servicio de ordenes. Además, se implementó el servicio, modelo, DTOs, fabrica, etc., relacionados al registro de cargos totales computados por impuesto para cada orden.

Hubo que hacer modificaciones debido a que el diseño propuesto en el primer Sprint no contemplaba el modo en que los tutores esperaban que se muestre la información relativa al resumen de impuestos aplicados. Queda como aprendizaje el hecho de que se debió haber indagado más respecto a dicho requerimiento, a favor de mejorar la flexibilidad a la hora de agregar funcionalidades.