Diplomado de actualización en nuevas tecnologías para el desarrollo de Software

Modulo 2, Procesos ágiles en el desarrollo de software

Jhonattan Gabriel Benavides
José Alejandro Castrillón
Marlon Tenganan
David Bastidas

Universidad de Nariño
Ingeniería de Sistemas
Enero de 2024

Utilizando la metodología SCRUM realizar los siguientes puntos

1. Permitir registrar Productos (nombre, cantidad, valor unitario) en dos diferentes motores de bases de datos. El motor de base de datos 1 debe almacenar los productos con valor unitario superior a 100.000 pesos en caso contrario se debe almacenar en el motor 2. (Utilizar el patrón Sigleton y Factory Method)

2. Permitir visualizar todos los productos.

3. En el mismo programa se desea permitir crear productos utilizando registros existentes en el inventario (Utilizar patrón prototype)

4. Se quiere implementar una funcionalidad que permita visualizar la cantidad total de productos y el valor total de inventario. Cada uno debe ser un elemento de interfaz diferente, la cantidad de productos mostrarlo en un una Etiqueta (JLabel) y el total de inventario en una caja de texto (JtextField). Los valores de estos productos se deben ver refrescados instantáneamente después de ingresar un nuevo producto. (Utilizar el patrón Observar)

5. Mostrar un informe, con los Eventos realizados (Dayli, Review y Retrospective), Colocar los roles de cada miembro del equipo, realizar al menos dos sprints y en cada uno mostrar que requerimientos están en el producto Sprint, explicar porque se priorizó de esta manera. Colocar los tiempos de desarrollo de cada requerimiento.

A continuación, se comenzará con el informe y se presentarán capturas de pantalla correspondientes a cada requisito cumplido.

Roles:

- Scrum Master: Marlon Tenganan.
- Product Owner: David Bastidas.
- Team Scrum: Gabriel Benavides y Alejandro Castrillón.

Product Backlog:

David Bastidas (Product Owner):

Sprint 1:

- **Product Sprint 1-> R1, R2**

R1: Permitir registrar Productos (nombre, cantidad, valor unitario) en dos diferentes motores de bases de datos. El motor de base de datos 1 debe almacenar los productos con valor unitario superior a 100.000 pesos en caso contrario se debe almacenar en el motor 2. (Utilizar el patrón Sigleton y Factory Method).

R2: Permitir visualizar todos los productos.

Tiempos de desarrollo estimados:

R1: 2 horas.

R2: 1 horas.

Explicación de la priorización:

Se prioriza el registro de productos (R1) antes que la visualización (R2) para garantizar que haya datos disponibles para mostrar.

Sprint 2:

- **Product Sprint 2 -> R3, R4.**

R3: En el mismo programa se desea permitir crear productos utilizando registros existentes en el inventario (Utilizar patrón prototype).

R4: Incorporar a la interfaz gráfica las funcionalidades crear, clonar y eliminar productos.

Tiempos de desarrollo estimados:

R3: 1 horas

R4: 6 horas

Explicación de la priorización:

Se prioriza la creación de productos utilizando registros existentes (R3) antes que la incorporación de las funcionalidades en la interfaz gráfica (R4) para garantizar la coherencia en los datos.

Sprint 3:

- **Product Sprint 3 -> R5.**

R5: Implementar el patrón Observer para notificar a los campos que guardan la cantidad de productos y el total del inventario, esto cada vez que se realice un nuevo registro o cambios en los productos existentes.

Tiempos de desarrollo estimados:

R5: 3 horas

Eventos: **Fecha inicio: 22/01/2024 1:40pm**

Sprint 1:

- ✓ **Daily 1: 22/01/2024, 2:30pm a 7:00pm**

Alejandro Castrillón (Team Scrum):

¿Qué se ha hecho??:

1. Creación de bases de datos en MySQL y PostgreSQL con el mismo nombre, ambas tienen la tabla products y los campos iguales sin tener en cuanta algunas propiedades propias de cada motor de base de datos.

2. Creación de las clases e interfaces necesarias para implementar el patrón Singleton y Factory Method con el fin de garantizar una única conexión a las bases de datos, además de crear, obtener, actualizar y visualizar productos.

¿Qué dificultades hay?: El entorno de desarrollo no reconoció temporalmente los Drivers que permiten conectar la aplicación con las bases de datos.

¿Qué queda por hacer?: Implementar los patrones Prototype y Observer e integrarlos en la interfaz gráfica.

Gabriel Benavides (Team Scrum):

¿Qué se ha hecho?: Revisión del código de registro, actualización y eliminación de productos. Iniciar en la implementación de la visualización (interfaz) utilizando el JFrame Form que está integrado en NetBeans, además de comunicar el código con esta interfaz.

¿Qué dificultades hay?: Problemas con las conexiones a las bases de datos MySQL y PostgreSQL.

¿Qué queda por hacer?: Resolver los problemas de conexión, los cuales muy seguramente sean debidos a las dependencias de versionamiento de estos.

Marlon Tenganan (Scrum Master):

¿Qué se ha hecho?: Ayuda en la resolución de problemas de conexión.

¿Qué dificultades hay?: Problemas técnicos.

¿Qué queda por hacer?: Continuar ayudando y prepararse para la revisión.

- **Review: 22/01/2024, 8:00pm a 8:20pm**

Marlon Tenganan (Scrum Master):

¿Qué porcentaje de producto sprint se terminó?

El porcentaje de sprint terminado es de un 100%

¿Qué se realizó?

Registro y visualización completados.

- Bases de datos con sus respectivas tablas para el registro de los productos. La primera creada en MySQL para productos con precios mayores a $100.000 y la segunda en PostgreSQL para precios menores o iguales a $100.000.

```
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| id             | int(11)      | NO   | PRI | NULL    | auto_increment |
| name           | varchar(246) | NO   |     | NULL    |                |
| quantity       | int(11)      | NO   |     | NULL    |                |
| price_per_unit | float        | NO   |     | NULL    |                |
| DBMS           | varchar(10)  | NO   |     | MySQL   |                |
+----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.026 sec)

MariaDB [agileinventorydb]>
```

```
                                    Table "public.products"
     Column     |          Type          | Collation | Nullable |               Default
----------------+------------------------+-----------+----------+-------------------------------------
 id             | integer                |           | not null | nextval('products_id_seq'::regclass)
 name           | character varying(256) |           | not null |
 quantity       | integer                |           | not null |
 price_per_unit | double precision       |           | not null |
 dbms           | character varying(10)  |           | not null | 'PostgreSQL'::character varying
Indexes:
    "products_pkey" PRIMARY KEY, btree (id)

agileinventorydb=#
```

- Implementación de patrones Singleton (única conexión) y Factory Method (Creación condicional).

```
DBConnection conn = DBConnection.getInstance();
conn.resetDatabases();

ArrayList<IProduct> products = new ArrayList<IProduct>(Arrays.asList(
        ProductFactory.factory("product1", 1, 200000),
        ProductFactory.factory("product2", 2, 20000),
        ProductFactory.factory("product3", 3, 50000),
        ProductFactory.factory("product4", 4, 5000000),
        ProductFactory.factory("product5", 5, 100000)));

for (IProduct product : products)
    conn.insertProduct(product);

for (IProduct product : conn.selectProducts())
    System.out.println(product);
```

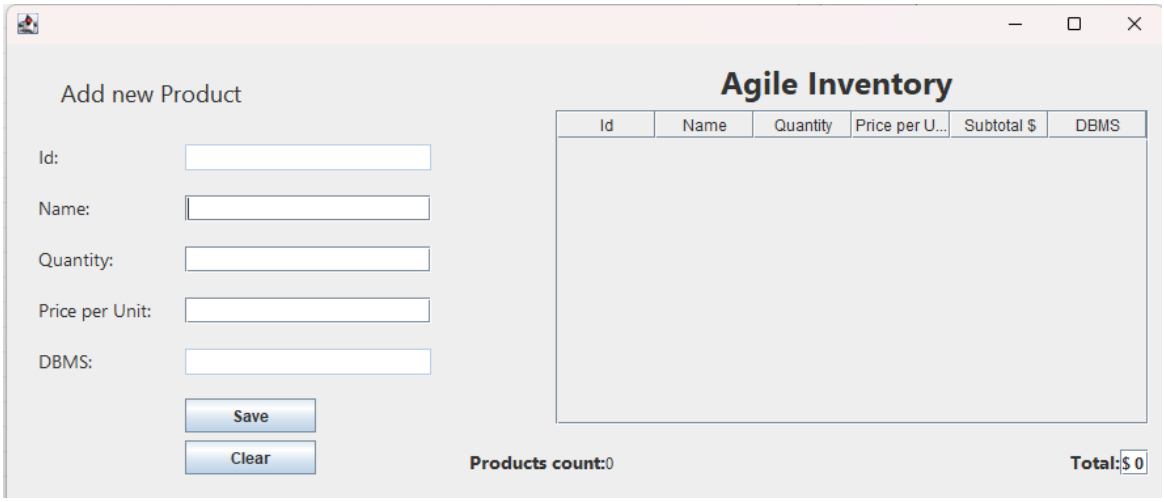
- Visualizar productos creados.

```
Product [ id: 1, name: product1, quantity: 1, pricePerUnit: 200000.0, DBMS: MySQL ]
Product [ id: 2, name: product4, quantity: 4, pricePerUnit: 5000000.0, DBMS: MySQL ]
Product [ id: 1, name: product2, quantity: 2, pricePerUnit: 20000.0, DBMS: PostgreSQL ]
Product [ id: 2, name: product3, quantity: 3, pricePerUnit: 50000.0, DBMS: PostgreSQL ]
Product [ id: 3, name: product5, quantity: 5, pricePerUnit: 100000.0, DBMS: PostgreSQL ]
```

```
+----+----------+----------+----------------+-------+
| id | name     | quantity | price_per_unit | DBMS  |
+----+----------+----------+----------------+-------+
|  1 | product1 |        1 |         200000 | MySQL |
|  2 | product4 |        4 |        5000000 | MySQL |
+----+----------+----------+----------------+-------+
2 rows in set (0.010 sec)

MariaDB [agileinventorydb]>
```

```
 id |   name   | quantity | price_per_unit |    dbms
----+----------+----------+----------------+------------
  1 | product2 |        2 |          20000 | PostgreSQL
  2 | product3 |        3 |          50000 | PostgreSQL
  3 | product5 |        5 |         100000 | PostgreSQL
(3 rows)

agileinventorydb=#
```

- Interfaz para la interacción con los productos, crear y visualizar productos.



¿Qué falta?

Nada.

¿Qué fallas se encontraron?

Problemas de integración con las bases de datos MySQL y PostgreSQL.

- **Restrospective: 22/01/2024, 8:20pm a 8:30pm**

Marlon Tenganan (Scrum Master):

Aspectos por mejorar: Manejo de problemas técnicos principalmente de código y credenciales.

Mejora continua: Planificar mejor las tareas.

Sprint 2:

- ✓ **Daily 1: 23/01/2024, 10:15am a 8:00pm**

Alejandro Castrillón (Team Scrum):

¿Qué se ha hecho?: 1) Implementar patrón prototype. 2) Usar la propiedad de clonar objetos en la funcionalidad del botón Clone. 3) Permitir la edición, clonación y eliminación de productos cargados en la interfaz sin necesidad de acceder a la base de datos.

¿Qué dificultades hay?: Ninguna.

¿Qué queda por hacer?: Avanzar en la implementación del patrón Observer.

Gabriel Benavides (Team Scrum):

¿Qué se ha hecho?: Interfaz en su mayoría completada y funcional.

¿Qué dificultades hay?: Problemas con la presentación de datos.

¿Qué queda por hacer?: Avanzar en la implementación del patrón Observar, ya que se tiene aún una lógica simple de actualización de la cantidad de productos y valor total.

Marlon Tenganan (Scrum Master):

¿Qué se ha hecho?: Ayuda en la identificación de problemas con la presentación de datos.

¿Qué dificultades hay?: Problemas técnicos persistentes.

¿Qué queda por hacer?: Continuar ayudando y buscar soluciones a los problemas técnicos.

- **Review: 22/01/2024, 8:00pm a 8:20pm**

Marlon Tenganan (Scrum Master):

¿Qué porcentaje de producto sprint se terminó?

El porcentaje de sprint terminado es de un 100%

¿Qué se realizó?

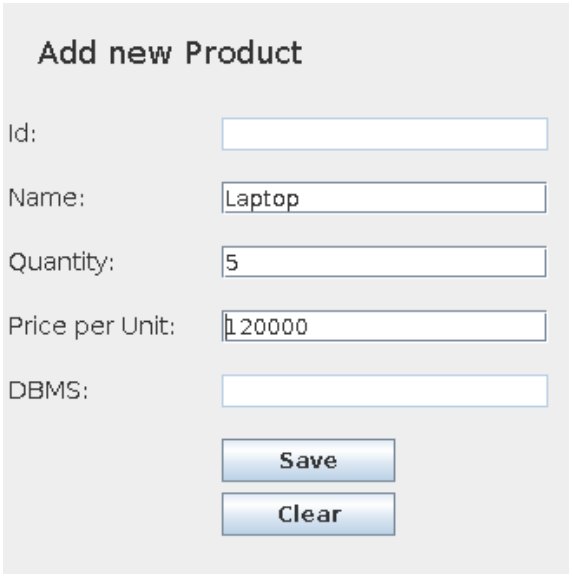
Creación de productos utilizando registros existentes completada, implementación de visualización completada.

- Implementación del patrón Observer para la clonación de objetos.

```java
IProduct p1 = ProductFactory.factory("cloneable-product", 1, 1000000);
IProduct p2 = p1.clone();
p2.setName("cloned-product");

System.out.println("Cloneable 1: " + p1);
System.out.println("Cloned 1: " + p2);
```

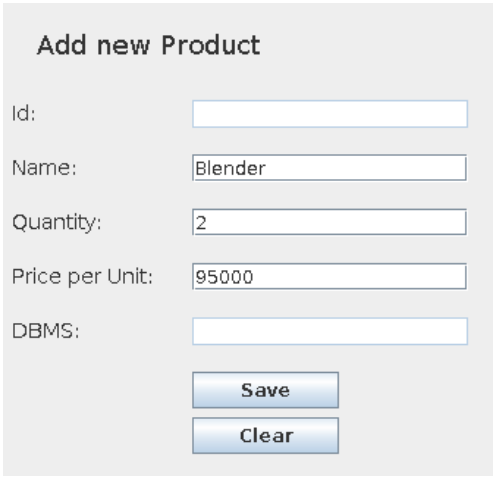
```
Cloneable 1: Product [ id: 0, name: cloneable-product, quantity: 1, pricePerUnit: 1000000.0, DBMS: MySQL ]
Cloned 1: Product [ id: 0, name: cloned-product, quantity: 1, pricePerUnit: 1000000.0, DBMS: MySQL ]
```

- Guardado y visualización de producto con valor superior a $100.000



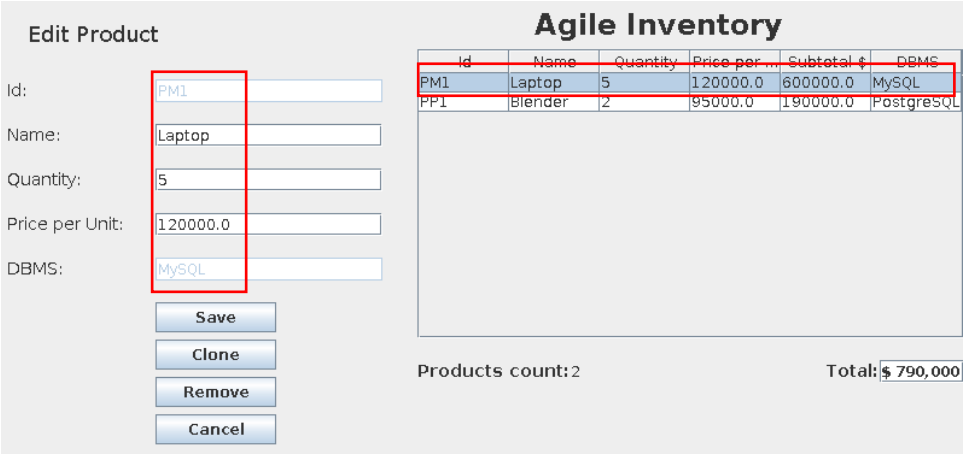
Id	Name	Quantity	Price per ...	Subtotal $	DBMS
PM1	Laptop	5	120000.0	600000.0	MySQL

- Guardado y visualización de producto con valor inferior a $100.000

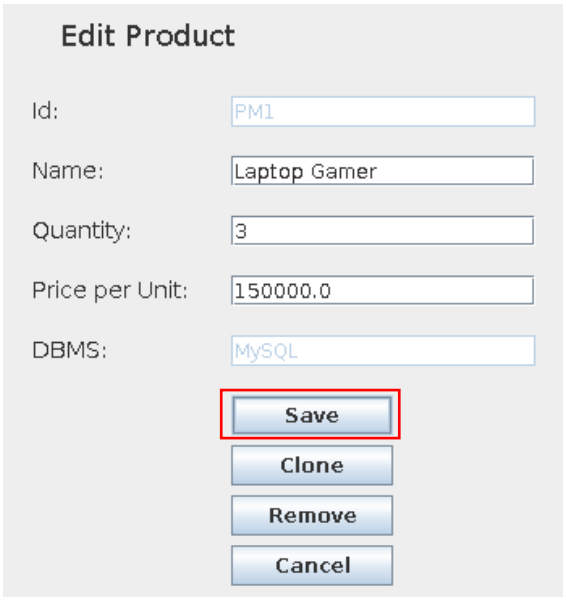


Id	Name	Quantity	Price per ...	Subtotal $	DBMS
PM1	Laptop	5	120000.0	600000.0	MySQL
PP1	Blender	2	95000.0	190000.0	PostgreSQL

- Selección de productos: Dar clic a producto en tabla.

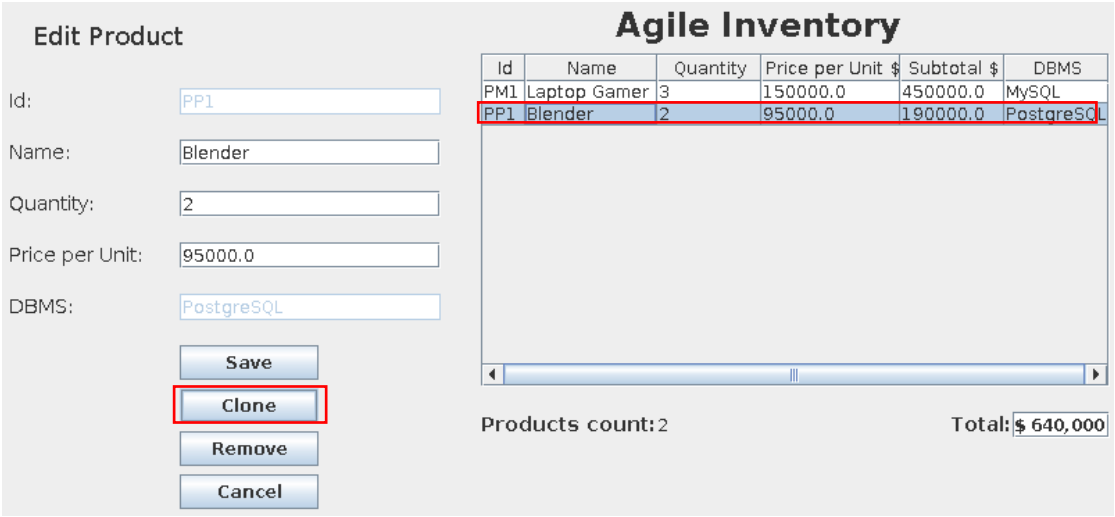


- Edición de productos: Clic en tabla, Editar campos y Clic en botón Save.



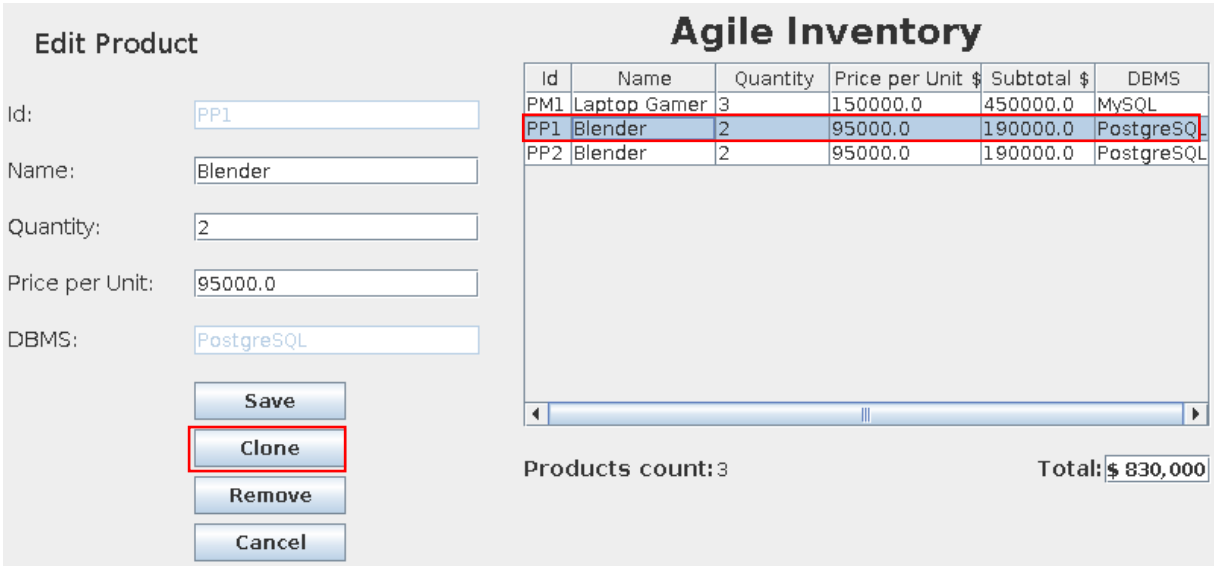
Id	Name	Quantity	Price per Unit $	Subtotal $	DBMS
PM1	Laptop Gamer	3	150000.0	450000.0	MySQL
PP1	Blender	2	95000.0	190000.0	PostgreSQL

- Clonación de producto: Seleccionar en tabla y Clic en Clone.



Id	Name	Quantity	Price per Unit $	Subtotal $	DBMS
PM1	Laptop Gamer	3	150000.0	450000.0	MySQL
PP1	Blender	2	95000.0	190000.0	PostgreSQL
PP2	Blender	2	95000.0	190000.0	PostgreSQL

- Eliminar registro de producto: Clic en tabla, y Clic en botón Remove.



Id	Name	Quantity	Price per Unit $	Subtotal $	DBMS
PM1	Laptop Gamer	3	150000.0	450000.0	MySQL
PP2	Blender	2	95000.0	190000.0	PostgreSQL

¿Qué falta?

Nada.

¿Qué fallas se encontraron?

Problemas con la presentación de datos.

- **Restrospective: 23/01/2024, 8:30pm a 8:40pm**

Marlon Tenganan (Scrum Master):

Aspectos por mejorar: Mejora del diseño y resolución de problemas técnicos.

Mejora continua: Enfocarse en el diseño eficiente y una respuesta rápida a problemas identificados.

Sprint 3:

- ✓ **Daily 1: 24/01/2024, 10:10pm a 7:15pm**

Alejandro Castrillón (Team Scrum):

¿Qué se ha hecho?:

1. Implementar interfaces y clases para manejar objetos Observadores y Observables.

2. Mover un producto al respectivo motor de base de datos al editar el valor unitario.

3. Mostrar cuadros de diálogo con información referente a las operaciones realizadas en la aplicación.

¿Qué dificultades hay?: Ninguna.

¿Qué queda por hacer?: Mejora en la visualización de la interfaz.

Gabriel Benavides (Team Scrum):

¿Qué se ha hecho?: Implementación del patrón Observer con la interfaz.

¿Qué dificultades hay?: Ninguna.

¿Qué queda por hacer?: Mejora en la eficiencia de código.

Marlon Tenganan (Scrum Master):

¿Qué se ha hecho?: Asistir en la eficiencia de código.

¿Qué dificultades hay?: Ninguna

¿Qué queda por hacer?: Finalización del documento.

- **Review: 24/01/2024, 9:30pm a 10:00pm**

Marlon Tenganan (Scrum Master):

¿Qué porcentaje de producto sprint se terminó?

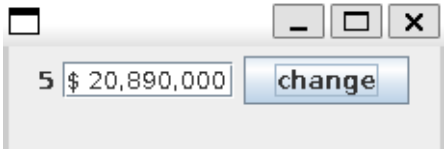
El porcentaje de sprint terminado es de un 100%

¿Qué se realizó?

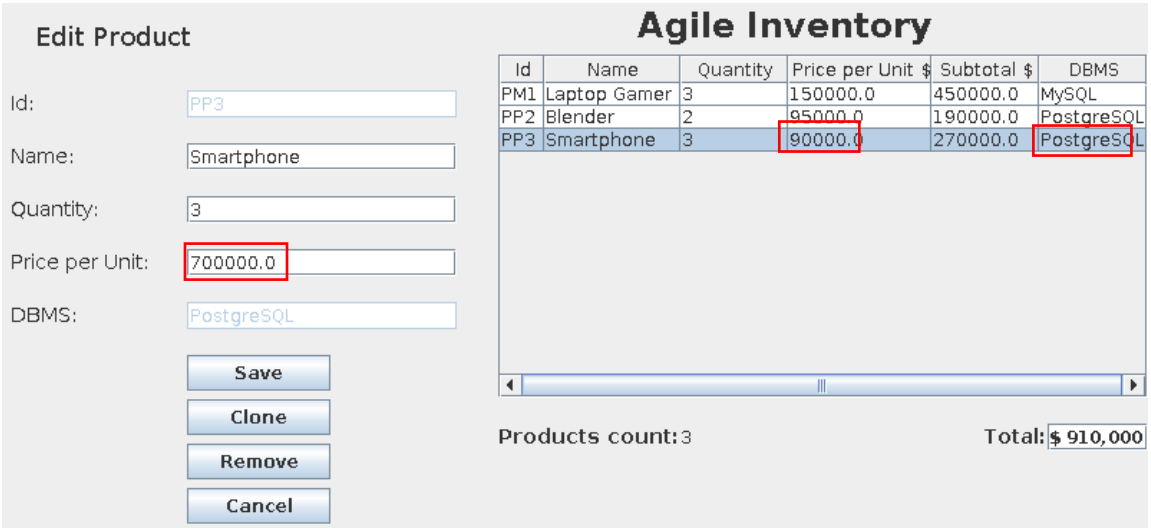
- Implementación de patrón Observer (Observables y Observadores).

```
JButton button = new JButton("change");
JLabel label = new JLabel("0");
JTextField field = new JTextField("$ 0");

CustomObservable observable = new CustomObservable(products);
QuantityObserver quantityObserver = new QuantityObserver(label);
TotalPriceObserver totalPriceObserver = new TotalPriceObserver(field);

observable.addObserver(quantityObserver);
observable.addObserver(totalPriceObserver);

button.addActionListener(e -> observable.notifyAllObservers());
```

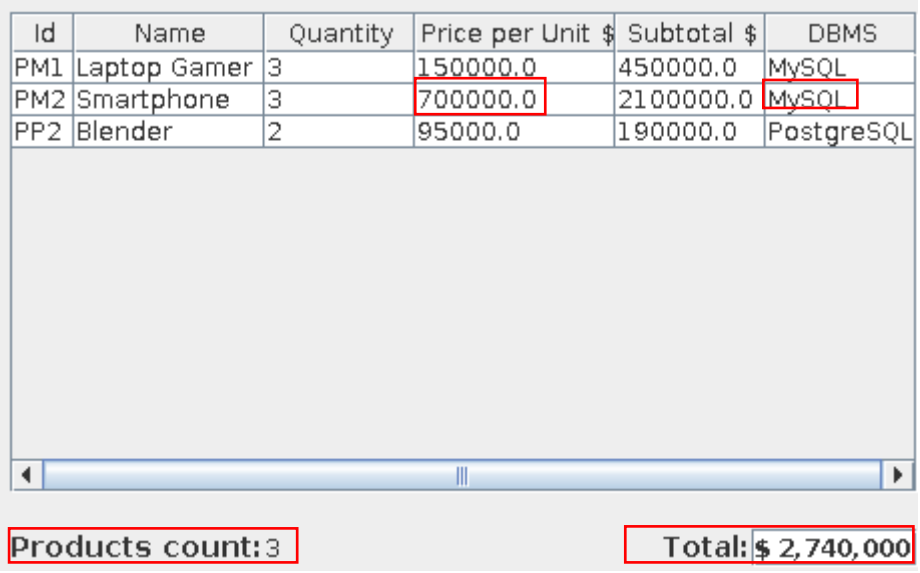
al dar clic en el botón:



- Mover el producto seleccionado al motor de base de datos correcto: Seleccionar producto en la tabla, Editar valor unitario (en este caso de $ 90.000 a $ 700.000), y Clic en botón Save.



- Al usar objetos observadores y observables personalizados en los elementos de la tabla se actualizan los valores de Cantidad de productos y Precio Total, sin usar técnicas de sumatoria en la funcionalidad del botón guardar.



¿Qué falta?

Nada.

¿Qué fallas se encontraron?

Ninguna.

- **Restrospective: 25/01/2024, 3:50pm a 4:05pm**

Marlon Tenganan (Scrum Master):

Aspectos por mejorar: Revisión más profunda para identificar oportunidades de optimización.

Mejora continua: Implementar revisiones de código entre pares de manera más regular. Esto no solo mejora la calidad del código, sino que también proporciona oportunidades para el aprendizaje mutuo y el intercambio de conocimientos.