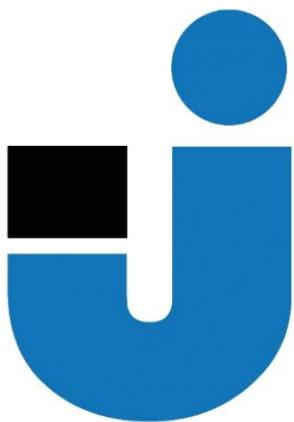


20 de julio de 2020

# Universidad Nacional Arturo Jauretche



## Trabajo Final: Metodologías de Programación II

**Profesora: Claudia Capelletti**

**Alumnos: Álvarez Marcos – Juan Roballo – Carlos Suarez**

**Comisión: N°1**

**Carrera: Ingeniería Informática**

**Correspondiente: Tercer Año**

# INDICE

1. DESCRIPCIÓN .....	3
2. IMPLEMENTACION DE CLASES .....	3
2.1. CLASE SUPERMERCADO .....	3
2.1.1. ATRIBUTOS .....	3
2.1.2. MÉTODOS DE CLASE .....	3
2.1.3. MÉTODOS DE INSTANCIAS .....	4
2.2. CLASE PRODUCTO .....	4
2.2.1. ATRIBUTOS .....	4
2.2.2. MÉTODOS DE CLASE .....	5
2.2.3. MÉTODOS DE INSTANCIAS .....	5
2.3. CLASE PRODUCTO VENDIDO .....	5
2.3.1. ATRIBUTOS .....	5
2.3.2. MÉTODOS DE CLASE .....	6
2.3.3. MÉTODOS DE INSTANCIAS .....	6
2.4. CLASE VENTA .....	6
2.4.1. ATRIBUTOS .....	6
2.4.2. MÉTODOS DE CLASE .....	6
2.4.3. MÉTODOS DE INSTANCIAS .....	7
3. IMPLEMENTACIÓN DE LA APLICACIÓN DESARROLLADA .....	7
3.1. WORKSPACE CARGA .....	7
3.2. WORKSPACE CONSULTAS-ORDENACIÓN .....	8
3.3. WORKSPACE ELIMINACIÓN-MODIFICACIÓN .....	9
3.4. WORKSPACE DICCIONARIO .....	10
4. DESCRIPCIÓN DE LA METODOLOGÍA DE TRABAJO .....	10
5. FRAMEWORKS QUE SE PODRÍAN UTILIZAR EN POO .....	10
6. USO DE PATRONES .....	11
7. REPOSITORIOS DE CÓDIGO .....	11

## **1. DESCRIPCIÓN**

Se trato de implementar el alta, baja y modificación de los productos del modelo de negocio de un supermercado. A su vez, se simulo la selección de los mismos y la introducción de la cantidad solicitada para posteriormente concretar la venta en cuestión. Por otro lado, también se procedió a representar la baja y modificación de los productos dados de alta en el supermercado.

## **2. IMPLEMENTACION DE CLASES**

Para el desarrollo y la simulación de la aplicación se utilizaron cuatro clases: Supermercado, Producto, ProductoVendido y Venta.

### **2.1. CLASE SUPERMERCADO**

Esta clase es la encargada de almacenar todos los productos dados de altas y las ventas concretadas. A su vez, tiene los métodos para consultar, eliminar, modificar y validar la selección de los productos.

#### **2.1.1. ATRIBUTOS**

- nombre: string
- productos: OrderedCollection
- ventas: OrderedCollection

#### **2.1.2. MÉTODOS DE CLASE**

- crearSupermercado(string nombre): Supermercado

### 2.1.3. MÉTODOS DE INSTANCIAS

- iniSupermercado(string nombre)
- agregarProducto(Producto objProducto)
- agregarVenta(Venta objVenta)
- eliminarProducto(Producto objProducto)
- eliminarProductoPorCodigo(int código)
- eliminarProductoPorNombre(string nombre)
- verProductos(): OrderedCollection (Producto)
- verVentas(): OrderedCollection (Venta)
- validarSeleccionProducto(int pCodigo): boolean
- consultarProductosPorCategoria(string categoria): OrderedCollection(Producto)
- consultarProductosPorCategoriaDistinta(string categoría): OrderedCollection(Producto)
- consultarTodasLasCategoriasProductos(): OrderedCollection(string)
- buscarProductoPorCodigo(int código): Producto
- buscarProductoPorNombre(string nombre): Producto
- existeProducto(Producto producto): boolean
- modificarProducto(int código, string nombre, string categoria, int precio, int stock)

## 2.2. CLASE PRODUCTO

Esta clase es la encargada de crear instancias de Producto cuando el usuario desea dar uno de alta. Posteriormente, la instancia es almacenada en una colección de productos de la clase “Supermercado”.

### 2.2.1. ATRIBUTOS

- codigo: int
- nombre: string
- precio: int
- categoria: string
- cantidad: int

## 2.2.2. MÉTODOS DE CLASE

- crearProducto(int cod, string nom, int prec, int cant, string cat): Producto

## 2.2.3. MÉTODOS DE INSTANCIAS

- iniProducto(int cod, string nom, int prec, int cant, string cat)
- agregarCantidad(int pCantidad)
- hayCantidad(): boolean
- modiCodigo(int pCodigo)
- modiNombre(string pNombre)
- modiPrecio(int pPrecio)
- modiCantidad(int pCantidad)
- modiCategoria(string pCategoria)
- verCodigo(): Number
- verNombre(): String
- verPrecio(): Number
- verCantidad(): Number
- verCategoria(): String

## 2.3. CLASE PRODUCTO VENDIDO

Esta clase tiene la utilidad cuando el usuario selecciona el producto a vender y por ende la cantidad que desea, por lo tanto, en ese punto del programa se crean instancias de esta clase para que posteriormente sea agregada a una colección de productos vendidos de la clase “Venta”.

### 2.3.1. ATRIBUTOS

- producto: Producto
- cantidad: int

### 2.3.2. MÉTODOS DE CLASE

- crearProductoVendido(Producto pProd, int pCant): ProductoVendido

### 2.3.3. MÉTODOS DE INSTANCIAS

- iniProductoVendido(Producto pProd, int pCant)
- verProducto: Producto
- verCantidad: int
- verPrecioTotal(): int

## 2.4. CLASE VENTA

Esta clase se instancia cada vez que se concreta o termina una venta, entre sus atributos se encuentra una colección donde almacena las instancias de la clase “ProductoVendido” que han sido seleccionados por el usuario e introducido su cantidad. Por otro lado, esta clase contiene un método de clase, donde recibe como parámetro una colección de instancias de “ProductoVendido” y calcula el importe total de la venta. A su vez, contienen otros dos métodos de clase, el cual uno incrementa el ID de cada venta y otro es el encargado de crear la instancia de la clase Venta, cuando está concluyendo todas las operaciones.

### 2.4.1. ATRIBUTOS

- idVenta: int
- fecha: Date
- productosVendidos: OrderedCollection (ProductoVendido)
- importe: int

### 2.4.2. MÉTODOS DE CLASE

- incrementarIdVenta(): int
- obtenerImporte(OrderedCollection pProductos)
- crearVenta(int pId, Date pFec, OrderedCollection pProd, int pImp): Venta

#### 2.4.3. MÉTODOS DE INSTANCIAS

- iniVenta(int pId, Date pFec, OrderedCollection pProd, int pImp)
- modiId(int pId)
- modi(Date pFecha)
- modi(OrderedCollection pProductos)
- modi(int pImporte)
- verId: Number
- verFecha: Date
- verProductosVendidos: OrderedCollection (ProductoVendido)
- verImporte: int

### 3. IMPLEMENTACIÓN DE LA APLICACIÓN DESARROLLADA

La implementación de la aplicación se desarrolló en diferentes espacios de trabajos (Workspaces), donde se comprobó el funcionamiento de la aplicación entre las diferentes clases que componían el programa. A continuación, se detalla el flujo de ejecución de los distintos espacios de trabajo implementados.

#### 3.1. WORKSPACE CARGA

En este espacio de trabajo se simulo la carga de productos a la colección dispuesta en la clase de “Supermercado” por parte del usuario. Posteriormente, se simulo y valido la selección de los productos cargados por parte del usuario, para poder concretar una venta, mostrando los productos seleccionados y su importe.

## 3.2. WORKSPACE CONSULTAS-ORDENACIÓN

En este espacio de trabajo, se pretendió completar varios incisos del trabajo final, haciendo uso de varios métodos para colecciones. Por lo tal, se implementaron los métodos “select”, “reject” y “collect” a modo de consultar los productos previamente cargados en la colección de la clase “Supermercado”. Para tal fin, se realizaron tres métodos en la clase “Supermercado” para efectuar diferentes consultas con los productos de la colección. Estos métodos son:

### **Utilización de select:**

- consultarProductosPorCategoria(string categoria):

### **Utilización de collect:**

- consultarTodasLasCategoriasProductos(): OrderedCollection(string)

### **Utilización de reject:**

- consultarProductosPorCategoriaDistinta(string categoría):  
OrderedCollection(Producto)

### **También se implementó la utilización de “detect” en el método:**

- buscarProductoPorNombre(string nombre) : Producto

### **3.3. WORKSPACE ELIMINACIÓN-MODIFICACIÓN**

Eliminación por nombre:

A través del ingreso del nombre de un producto por el usuario que se pasa como parámetro al mensaje “eliminarProductoPorNombre” perteneciente a la clase Supermercado, se busca el producto en la colección mediante el mensaje “buscarProductoPorNombre” que devuelve el producto de la colección y a partir de esta forma se elimina el mismo.

El programa pregunta si se desea seguir eliminando productos, en caso afirmativo se ingresa un nuevo nombre y en caso contrario este finaliza.

Eliminación por código:

Análogo al anterior pero la búsqueda se realiza a partir del código del producto.

Modificación por Nombre:

A través del ingreso del código de un producto y sus datos a modificar, se envía el mensaje “modificarProducto”, el cual halla al objeto en la colección a través de su código identificativo, esto lo logra a través del uso de un método auxiliar “detect” para colecciones. Una vez, devuelto el producto, se procede a modificar el mismo con los datos pasados como parámetros. La modificación se realiza mediante el uso de los métodos de instancias de la clase Producto los cuales permiten la alteración del estado de un objeto perteneciente a dicha clase.

Como se mencionó anteriormente el programa pregunta, si se desea seguir modificando productos, en caso afirmativo, se ingresa un nuevo código con sus respectivos datos a modificar, en caso contrario este finaliza.

### **3.4. WORKSPACE DICCIONARIO**

Implementación de la estructura del diccionario a través del alta de productos, las categorías y su ordenación.

## **4. DESCRIPCIÓN DE LA METODOLOGÍA DE TRABAJO**

Por cuestiones de tiempo y de poca comunicación con los integrantes del grupo, no se pudo establecer ninguna metodología de trabajo exacta. Por ende, los integrantes del trabajo, fueron realizando los incisos del mismo a medida que transcurría el tiempo y se acercaba la fecha de entrega.

## **5. FRAMEWORKS QUE SE PODRÍAN UTILIZAR EN POO**

Se podrían implementar diversos Frameworks en el paradigma de la POO de acuerdo a la tecnología y al tipo de desarrollo que se desea implementar. Por lo tanto, estos son algunos de los más utilizados:

.NET: es un Framework creado por la empresa Microsoft. A través, de este marco de trabajo, se pueden realizar distintos desarrollos como aplicaciones de escritorio (Windows Forms o WPF), aplicaciones móviles mediante Xamarin o desarrollos Web usando ASP.NET o ASP.NET MVC. Para programar en el entorno de .NET existen diversos lenguajes, pero los más usados son C# y VB.

Spring: es uno de los Framework más utilizados en la programación para el lenguaje Java en aplicaciones de desarrollo Web.

Laravel: es un Framework de desarrollo Web para el lenguaje PHP.

## **6. USO DE PATRONES**

En el desarrollo realizado si se hubiera tenido una importante cantidad de diferentes consultas de datos sobre la colección, se hubiera podido implementar un Factory Method en conjunto con Strategy, donde a través de la selección de una opción de consulta por parte del usuario, delegue al patrón Factory Method a crear la instancia correspondiente definida en el patrón Strategy para realizar el tipo de consulta correspondiente, elegida por el usuario. También se podrían haber utilizado otro patrones como MVC y DAO en conjunto.

## **7. REPOSITORIOS DE CÓDIGO**

Los repositorios que se podrían utilizar para el desarrollo de software colaborativo son:

- GitHub
- GitLab
- SourceForge
- GitKraken

## 8. DIAGRAMA DE CLASES DEL PROYECTO

