PROYECTO INTEGRADOR “mateAR”

DISEÑO DE SISTEMAS – Año 2021

Ingeniería en Sistemas de Información

Alumna

Filippa, Rebeca (14833)

Team 1

Profesores

Ferreyra, Juan Pablo

Pioli, Pablo

Fecha límite de entrega

19/11/2021

Universidad Tecnológica Nacional, Facultad Regional San Francisco

ÍNDICE

Título del trabajo

INTRODUCCIÓN

El siguiente trabajo integrador se presenta a pedido de los docentes a cargo de la cátedra de Diseño de Sistemas, de la Universidad Tecnológica Nacional, Facultad Regional San Francisco. El mismo se repartió en teams, cada uno con una implementación particular, pero apuntando al mismo propósito y trabajando en forma conjunta.

Su finalidad es, poder llevar a cabo el desarrollo de un sistema, en este caso bajo el dominio particular de "Venta de Mates y complementos", con el desafío de experimentar un camino similar que requerirá el trabajo final de carrera, pero con un menor nivel de dificultad.

En este caso, se discutieron puestas en común con los compañeros de cátedra para definir el dominio particular al que apunto el proyecto, que luego fue la base para la búsqueda de información respecto de la competencia del medio y así, poder definir las implementaciones con las que se trabajarán.

Seguidamente, se representó mediante un diagrama BPMN, que en secciones posteriores se desarrollará con mayor nivel de detalle, el recorrido que realizaron las actividades que hacen a la vida de nuestro sistema, para posteriormente tener una mejor visualización del comportamiento que tendrá el mismo y la forma en que se interrelacionan las actividades que lo conforman.

Además, se trabajó con una representación de la arquitectura, mediante un modelo arquitectónico en capas, dejando en vista cuales son los niveles que logramos identificar de acuerdo a la complejidad de nuestro sistema.

Finalmente, se realizaron las pruebas correspondientes para evaluar que el sistema realmente funcione como es de esperar, dejando constancia detallada de ello.

DESARROLLO

- MODELO DE NEGOCIO
- DISEÑO ORIENTADO A OBJETOS
- PROTOTIPO
- DISEÑO DE ARQUITECTURA
- PERSISTENCIA
- VERIFICACIÓN Y VALIDACIÓN

MODELO DE NEGOCIO

Teniendo como base las competencias del medio que se lograron identificar, se estableció una lista de los requerimientos que hacen a la vida del sistema en sí.

- Categorías de productos
- Mostrar los productos
- Mostrar combos que incluyen el producto seleccionado
- Actualizar el stock y los nuevos modelos
- Contar con un carrito de venta
- Cobrar la venta
- Envío de productos y combos
- Cuenta personal
- Comentarios y preguntas
- Estadísticas

A partir de esto, se englobaron en una serie de teams implementaciones en común, que terminaron de definir las acciones que desempeñará el sistema.

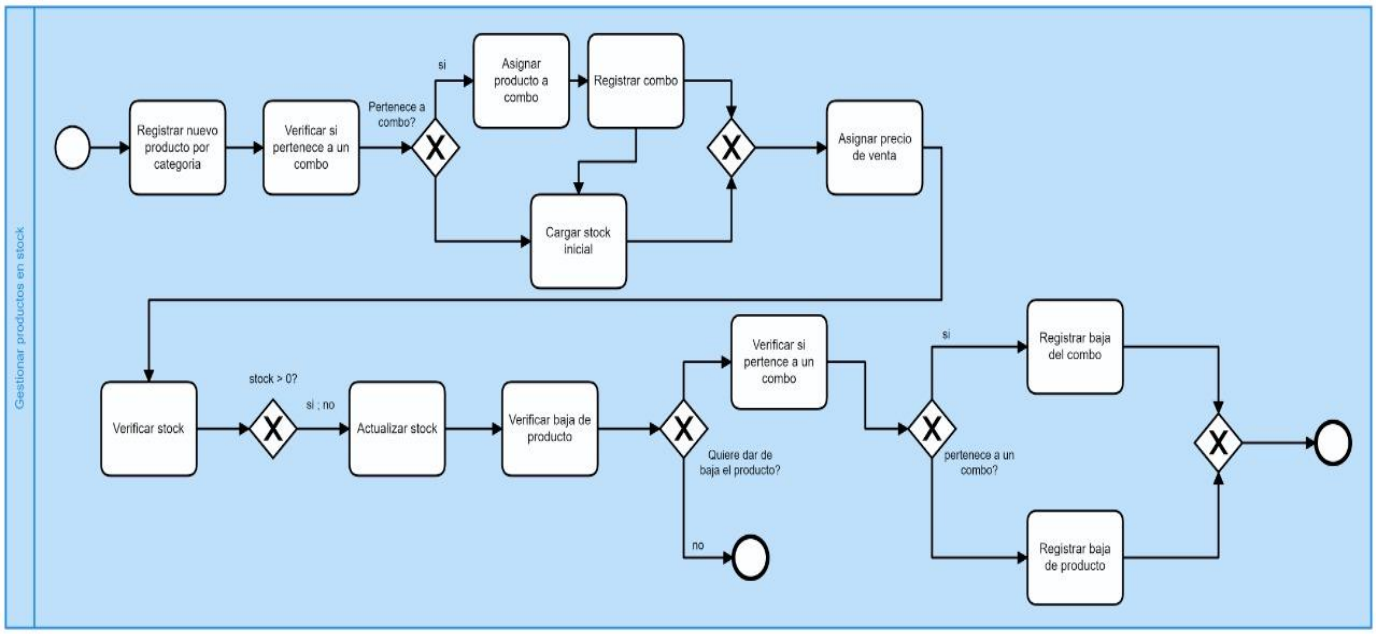
Team 1, Definir los productos por categorías y actualizar su stock.
Team 2, Mostrar los productos y sus respectivos combos.
Team 3, Carrito de compra.
Team 4, Realizar el cobro de los productos vendidos y métodos de envío.
Team 5, Cuenta personal.

En este caso, se desarrolló particularmente la implementación correspondiente a “Definir los productos por categorías y actualizar su stock”.

El problema a solucionar está enfocado en lo que el usuario quiere que haga el sistema, centrándose en la carga de los productos que estarán disponibles para su venta, así como también de su actualización y la baja de los mismos.

La solución que se implementó para ello, requiere definir una lista de los productos con los que se cuenta, detallando precio, color, tamaño, modelo y disponibilidad de cada uno. Además, poder identificar el stock actual y realizar las correspondientes actualizaciones cuando así se lo requiera.

A continuación, se adjunta una captura que corresponde al diagrama BPMN, donde queda plasmada la secuencia de actividades que le añaden valor a nuestro producto final.



Cerca de las etapas finales se incorporaron requerimientos simulando la cruda realidad en la que el usuario puede pretender añadir nuevas funcionalidades al sistema en cualquier etapa de la vida del mismo. Estos requisitos se nombran a continuación.

- El usuario pretende conocer los productos que están en oferta, se debería permitir cargar una oferta con una fecha de vigencia.
- Permitir definir precio en función de la cantidad a comprar.

A partir de ello, se debió integrar tales pretensiones del usuario, sin afectar las implementaciones ya trabajadas.

DISEÑO ORIENTADO A OBJETOS

En esta etapa se trabajó con tres tipos de diagramas UML, uno correspondiente a la vista de diseño estático “Diagrama de Clases “ y los otros dos de comportamiento “Diagrama de CU” y “Diagrama de Secuencia”.

Mediante el *Diagrama de Clases*, se buscó emplear un esquema lógico de los datos, para modelar el vocabulario del sistema. Mientras que, en el *Diagrama de CU*, se describe un grupo de secuencias de acciones que detallan el comportamiento que se espera tenga el sistema, y en el *Diagrama de Secuencia*, mostramos las interacciones entre objetos mediante el envío de mensajes entre ellos.

En primer lugar, se diseñó el *Diagrama de Casos de Uso*, dejando plasmadas las actividades principales a realizar por el sistema.

Luego, en segundo lugar, se trabajó con el *Diagrama de Clases* para tener un pantallazo de cuáles son los bloques de construcción más importantes, con sus respectivos atributos, operaciones y relaciones entre los mismos.

Con el agregado de los nuevos requerimientos antes nombrados, que se incorporaron más tardíamente al proyecto, se vio afectado el esquema del *Diagrama de Clases*, debiendo ser modificado para que contemple las adiciones que se dieron a causa de tales implementaciones.

Por último, en tercer lugar, se esbozó el *Diagrama de Secuencia*, donde se reflejó el camino que recorren los mensajes para intercambiar información permitiendo que las clases interaccionen entre ellas.

A continuación, se adjuntan las capturas correspondientes a los diagramas antes nombrados.

Diagrama de Casos de Uso

En este caso particular, se pensó en el actor con un perfil de *Administrador*, el cual es en definitiva un trabajador del sistema. El mismo, es quien realiza la alta y baja de los productos y combos, como así también, el encargado de registrar el stock disponible de cada uno y las actualizaciones correspondientes cuando así se lo requiera.

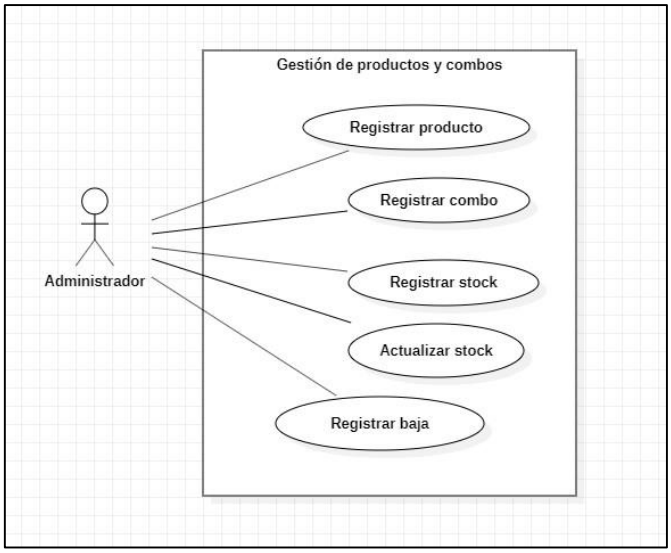


Diagrama de Clases

Descripción breve de las clases consideradas.

Producto, representa los elementos que estarán disponible para su comercialización, considerando características de su esencia (código, descripción, modelo, color, entre otros), stock y datos extras.

ProductoCombo, es la clase intermedia que representa información relevante para las otras dos clases.

Combo, representa los elementos en conjunto que estarán disponible para su comercialización, considerando características de la esencia del combo en sí.

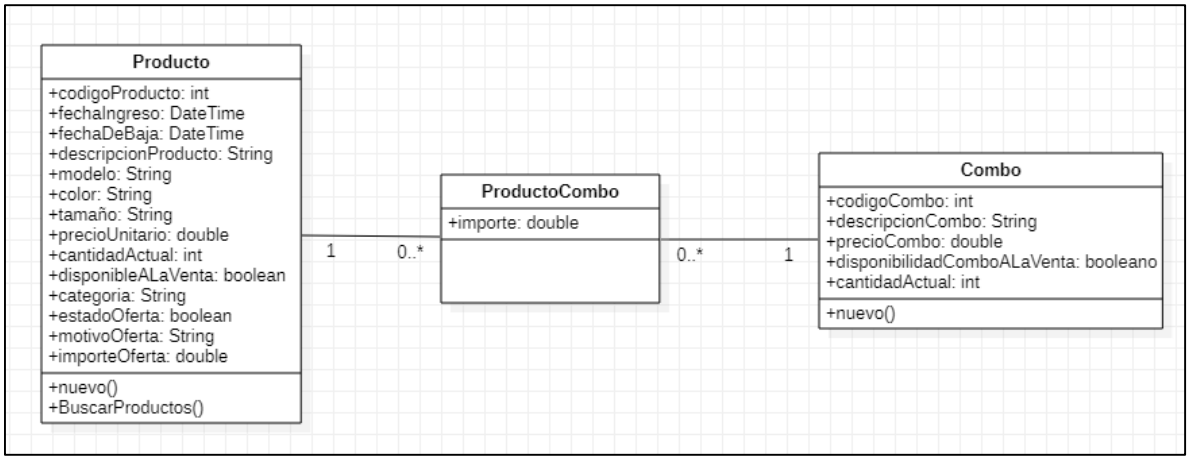
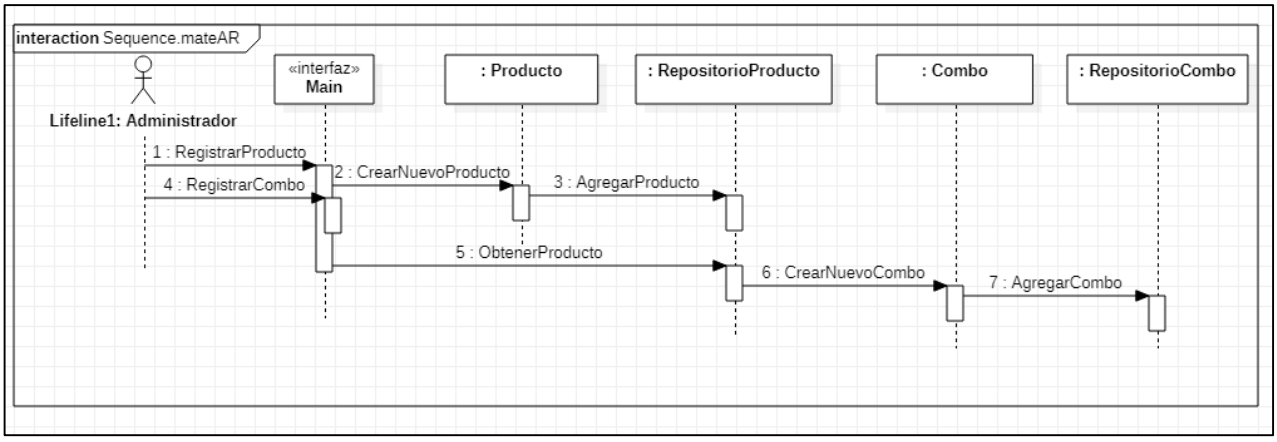


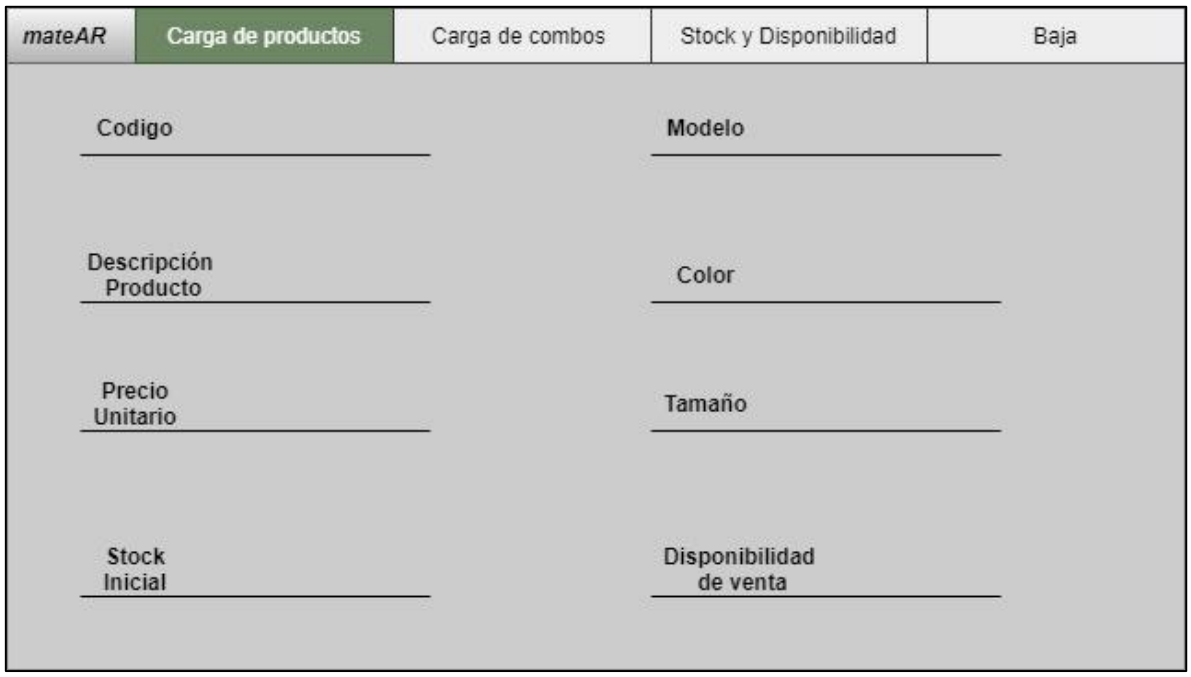
Diagrama de Secuencia

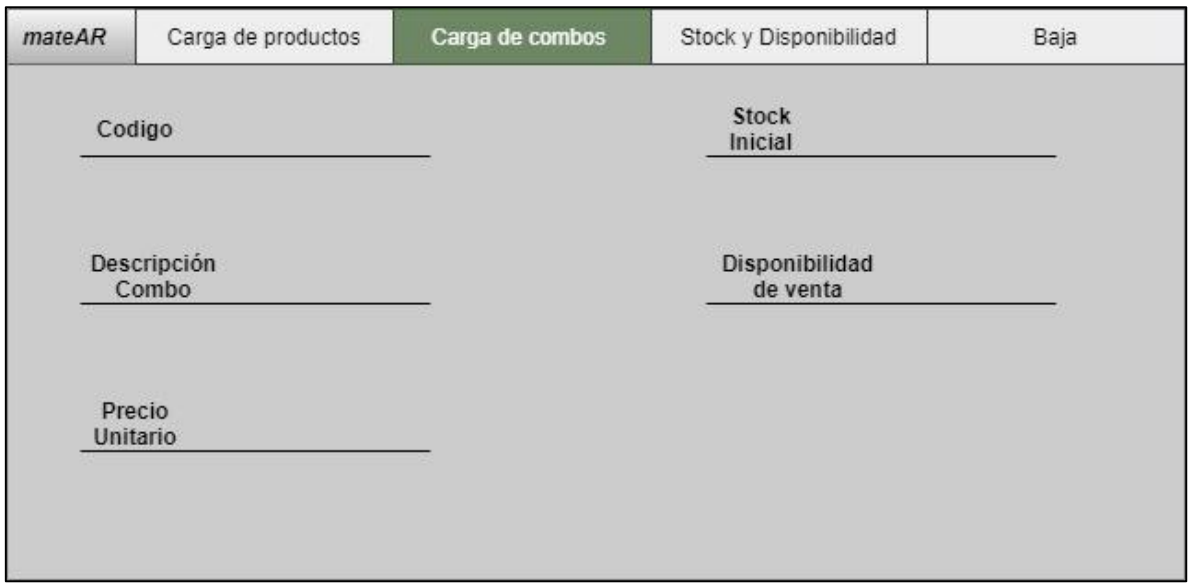


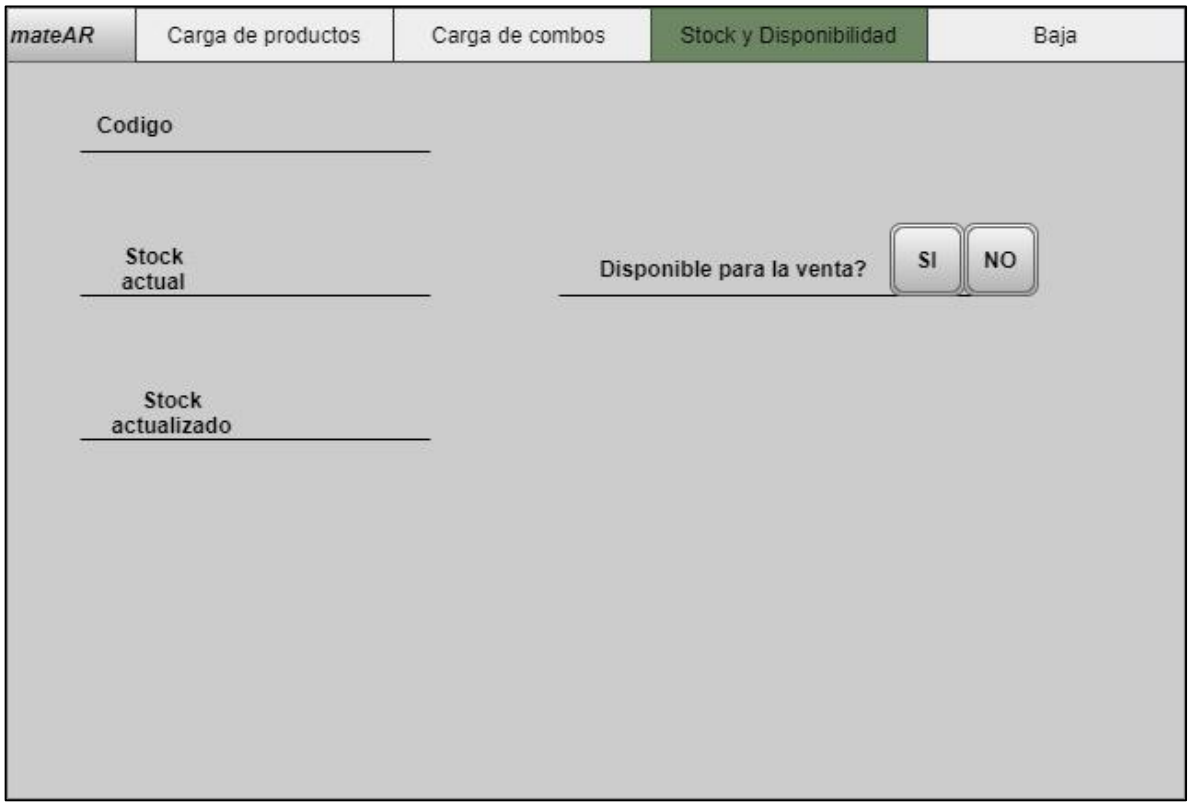
PROTOTIPO

En esta sección buscamos como finalidad simular el producto final, considerando su interfaz como así también su funcionalidad de entradas y salidas, definiendo los componentes que implementan la interacción con los usuarios, a través de una representación visual del sistema.

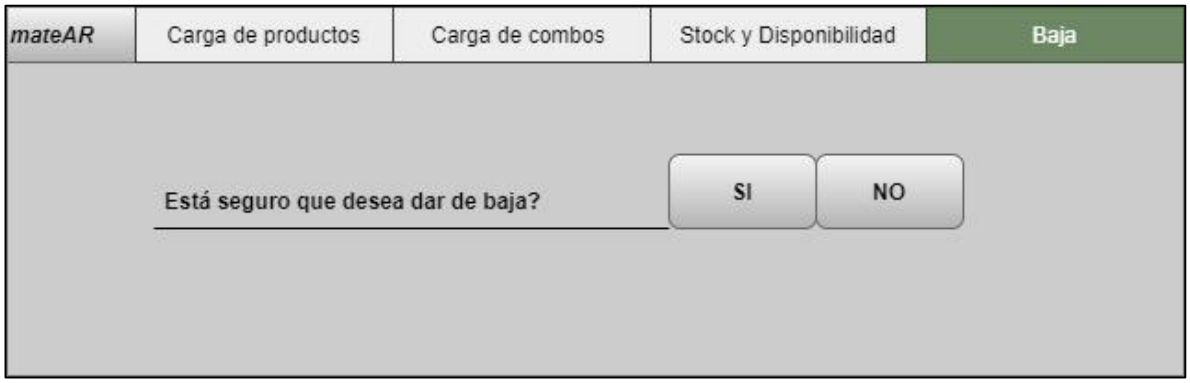
Mediante esto, se busca representar cómo el usuario podrá navegar por las distintas páginas para poder obtener la información que desee o aportarla dependiendo de su necesidad, con el propósito de obtener una evaluación sobre los requerimientos funcionales y la usabilidad del sistema.






mateAR
Carga de productos
Carga de combos
Stock y Disponibilidad
Baja
Codigo
Stock actual
Disponible para la venta?
SI
NO
Stock actualizado


mateAR
Carga de productos
Carga de combos
Stock y Disponibilidad
Baja
Codigo
DAR BAJA


mateAR
Carga de productos
Carga de combos
Stock y Disponibilidad
Baja
Está seguro que desea dar de baja?
SI
NO

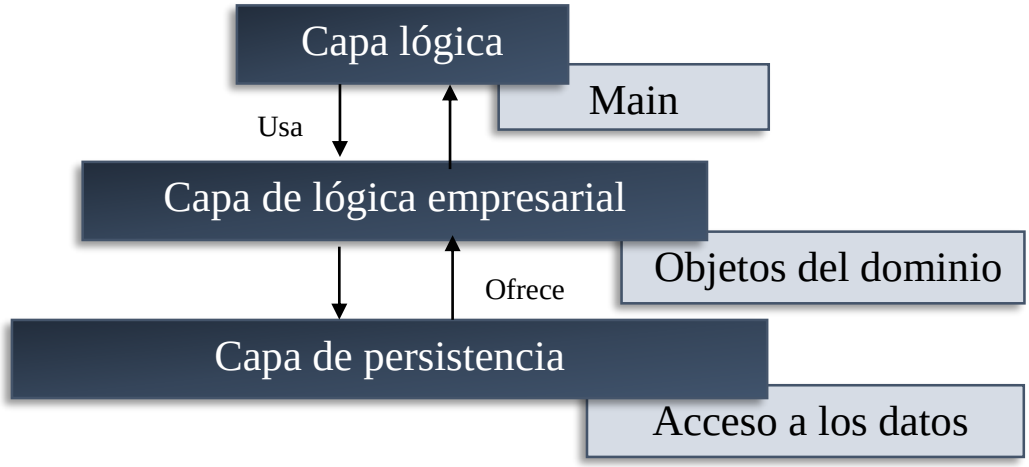

mateAR
Carga de productos
Carga de combos
Stock y Disponibilidad
Baja
Baja realizada con éxito, el día 2021-09-30

DISEÑO DE ARQUITECTURA

En esta sección, se pensó en el patrón de arquitectura en capas.

En este caso, la funcionalidad del sistema está organizada en capas separadas, de manera que la forma de interacción y las responsabilidades se reparten jerárquicamente y cada una se apoya en los servicios ofrecidos por la capa inmediatamente debajo de ella.

A continuación, se ilustra un boceto de una arquitectura en capas, en este caso con 3 capas.



Capa lógica, proporciona a los usuarios una forma de acceder y controlar los datos y servicios de los objetos.

Capa de dominio, se define el conjunto de componentes que implementan el comportamiento de las clases del dominio, es decir, su funcionalidad. Es esta capa la que se comunica directamente con la *Capa de persistencia* para recuperar los datos requeridos, solo recibe datos o los procesa. Toma peticiones o eventos del usuario, las procesa y luego envía la respuesta de vuelta a la *Capa lógica.*

En este caso, la *Capa de negocio* se corresponde con las clases que están representadas en el *Diagrama de Clases*.

Capa de persistencia, se define el conjunto de componentes que proporcionan servicios a los objetos, permitiendo que los mismos interactúen con su repositorio asociado. En este caso, contamos con el *RepositorioCombo*, *RepositorioProducto* y el *DTO*, que nos permitirá intercambiar la información.

PERSISTENCIA

En esta sección, se desarrolló como se preservan los datos de nuestro sistema. Existen numerosas alternativas a la hora de seleccionar un mecanismo para acceder a los datos y posteriormente manipularlos.

En este caso, se utilizaron estructuras de datos dinámicas que fueron implementadas en las clases *RepositorioCombo* y *RepositorioProducto*, definidas en nuestro código. Se trabajó con dos listas, una correspondiente a los productos y otra a los combos, en las cuales se adicionan los elementos que se ingresan por teclado y representan aquellos que estarán disponible para su venta, para posteriormente hacer uso de ellas y poder recuperar los datos a fin de manipularlos para propósitos específicos.

Además, se utilizaron *Patrones DTO* para intercambiar información con el resto de los teams. Estos *Patrones DTO*, tienen como finalidad crear un objeto plano con una serie de atributos que pueden ser enviados o recuperados y contener la información de múltiples clases para concentrarla en una única.

En este caso, se trabajó con cuatro listas correspondientes al *DTO*, *ProductoTeamDos*, *ComboTeamDos*, *ProductoTeamTres*, *ProductoTeamCinco*, en cada una de las cuales se incorporaron solo los atributos requeridos por cada team particular, por lo cual se necesitó que nos comunicáramos para acordar cuáles y con qué formato pretendían los atributos para su trabajo.

VERIFICACIÓN Y VALIDACIÓN

En esta sección, nos centramos en procesos de análisis y pruebas, donde buscamos comprobar que el sistema que estamos desarrollando satisface las funcionalidades esperadas por quien pretende el software.

La Validación, se refiere a si estamos construyendo el producto correcto, si se cumple con los requerimientos funcionales y las restricciones que afectan a los mismos. En definitiva, es comprobar que el sistema cumple con las expectativas del cliente.
La Verificación, si lo estamos construyendo correctamente.

En conclusión, el objetivo fundamental de esta etapa es establecer la seguridad de que el sistema es lo suficientemente bueno para el uso que se pretende, implantando defectos existentes en el sistema.

En este proyecto, trabajamos con *Pruebas unitarias* y *Pruebas de aceptación*. En el primer caso, probamos los módulos por separado, logrando utilizar una o más de las clases implementadas como objeto de las pruebas a desarrollar. En el caso de las *Pruebas de aceptación*, se comprobó que el sistema cumple con los requisitos previstos, encontrando la mayor cantidad de errores, pensando de antemano que es lo que se busca probar. Cada prueba es documentada con la finalidad de que pueda ser probada en cualquier momento. Tal documentación es escrita en el lenguaje coloquial del cliente, pero que puede ser ejecutado por la máquina.

A continuación, se presenta la documentación de las *Pruebas de aceptación* realizadas.

#	Nombre (Función que se prueba)	Valores ingresados	Resultado esperado
1	Registrar nuevo producto con código únicamente numérico.	**Código:** F36V1 **Descripción:** Mate camionero **Modelo:** Original térmico **Color:** Verde **Precio Unitario:** 5900 **Cantidad Actual:** 50 **Disponibilidad:** True	Debería devolver error de Código numérico.
2	Registrar nuevo producto con disponibilidad dicotómica (true o false).	**Código:** 125 **Descripción:** Termo Stanley **Modelo:** Clásico **Color:** Rojo **Precio Unitario:** 12100 **Cantidad Actual:** 120 **Disponibilidad:** DISPONIBLE	Debería devolver error de Disponibilidad distinta de true o false.

3	Registrar nuevo producto con precio unitario sin signo monetario.	**Código:** 75 **Descripción:** Bombilla Pico de Loro **Modelo:** Alpaca **Color:** Plateado **Precio Unitario:** $660 **Cantidad Actual:** 14 **Disponibilidad:** False	Debería devolver error de Precio unitario sin signo monetario.
4	Registrar nuevo producto con formato de separador de coma en decimales para el precio.	**Código:** 409 **Descripción:** Mate Pampa **Modelo:** Cuero térmico **Color:** Blanco **Precio Unitario:** 5248.50 **Cantidad Actual:** 28 **Disponibilidad:** True	Debería devolver error de Precio unitario con separador de coma en decimales.
5	Registrar nuevo producto con cantidad actual numérica.	**Código:** 8407 **Descripción:** Mate Acero Inoxidable **Modelo:** Doble capa **Color:** Negro **Precio Unitario:** 2600,50 **Cantidad Actual:** SIN STOCK **Disponibilidad:** True	Debería devolver error de Cantidad actual con cantidad numérica.
6	Registrar nuevo producto con cantidad actual mayor o igual a cero.	**Código:** 810 **Descripción:** Mate Torpedo **Modelo:** Original térmico **Color:** Rojo **Precio Unitario:** 5230,60 **Cantidad Actual:** -74 **Disponibilidad:** True	Debería devolver error en Cantidad actual mayor o igual a cero.
7	Registrar nuevo producto con precio unitario mayor a cero.	**Código:** 7814 **Descripción:** Bombilla Plana **Modelo:** Alpaca **Color:** Plateado **Precio Unitario:** 0 **Cantidad Actual:** 8 **Disponibilidad:** True	Debería devolver error de Precio unitario mayor a cero.
8	Registrar nuevo producto con precio unitario mayor a cero.	**Código:** 812 **Descripción:** Bombilla Chata **Modelo:** Acero boca ancha **Color:** Plateado **Precio Unitario:** -432 **Cantidad Actual:** 3 **Disponibilidad:** False	Debería devolver error de Precio unitario mayor a cero.

9	Registrar nuevo producto con cantidad actual que no supere los diez dígitos.	**Código:** 1094 **Descripción:** Mate Uruguayo **Modelo:** Imperial **Color:** Negro **Precio Unitario:** 5690 **Cantidad Actual:** 10000000000 **Disponibilidad:** True	Debería devolver error de Cantidad actual que no supere los diez dígitos.
10	Registrar nuevo producto con código que no supere los diez dígitos.	**Código:** 19278064305 **Descripción:** Mate Uruguayo **Modelo:** Trenzado con tiento **Color:** Blanco **Precio Unitario:** 6780,99 **Cantidad Actual:** 6 **Disponibilidad:** True	Debería devolver error de Código que no supere los 10 dígitos.
11	Registrar nuevo combo con código únicamente numérico.	**Código:** 12D8S **Descripción:** Combo Completo **Precio Unitario:** 3860 **Cantidad Actual:** 5 **Disponibilidad:** True	Debería devolver error de Código numérico.
12	Registrar nuevo combo con precio unitario sin signo monetario.	**Código:** 410 **Descripción:** Combo Semi-Completo **Precio Unitario:** $2970 **Cantidad Actual:** 10 **Disponibilidad:** False	Debería devolver error de Precio unitario sin signo monetario.
13	Registrar nuevo combo con precio unitario mayor a cero.	**Código:** 102 **Descripción:** Combo Completo **Precio Unitario:** 0 **Cantidad Actual:** 8 **Disponibilidad:** False	Debería devolver error de Precio unitario mayor a cero.
14	Registrar nuevo combo con precio unitario mayor a cero.	**Código:** 612 **Descripción:** Combo Completo **Precio Unitario:** -3912 **Cantidad Actual:** 6 **Disponibilidad:** True	Debería devolver error de Precio unitario mayor a cero.
15	Registrar nuevo combo con formato de separador de coma en decimales para el precio.	**Código:** 314 **Descripción:** Combo Mini-Completo **Precio Unitario:** 1072.41 **Cantidad Actual:** 6 **Disponibilidad:** True	Debería devolver error de Precio unitario con separador de coma en decimales.

16	Registrar nuevo combo con cantidad actual numérica.	**Código:** 1325 **Descripción:** Combo Extra-Completo **Precio Unitario:** 7321,42 **Cantidad Actual:** SIN STOCK **Disponibilidad:** True	Debería devolver error de Cantidad actual con cantidad numérica.
17	Registrar nuevo combo con cantidad actual mayor o igual a cero.	**Código:** 1406 **Descripción:** Combo Completo **Precio Unitario:** 3970,15 **Cantidad Actual:** -31 **Disponibilidad:** False	Debería devolver error de Cantidad actual mayor o igual a cero.
18	Registrar nuevo combo con cantidad actual que no supere los diez dígitos.	**Código:** 250 **Descripción:** Combo Completo **Precio Unitario:** 4016,10 **Cantidad Actual:** 10000000000 **Disponibilidad:** True	Debería devolver error en Cantidad actual que no supere los diez dígitos.
19	Registrar nuevo combo con código que no supere los diez dígitos.	**Código:** 62460780169 **Descripción:** Combo Mini-Completo **Precio Unitario:** 1350 **Cantidad Actual:** 8 **Disponibilidad:** True	Debería devolver error de Código que no supere los diez dígitos.
20	Registrar nuevo combo con disponibilidad dicotómica (true o false).	**Código:** 316 **Descripción:** Combo Extra-Completo **Precio Unitario:** 7412,50 **Cantidad Actual:** 8 **Disponibilidad:** DISPONIBLE	Debería devolver error de Disponibilidad distinta de true o false.
21	Actualizar stock con stock actual mayor o igual a cero.	**Código:** 153 **Stock actual:** -187 **Cantidad a reponer:** 26	Debería devolver error de Stock actual mayor o igual a cero.
22	Actualizar stock con código únicamente numérico.	**Código:** 5DE3 **Stock actual:** 32 **Cantidad a reponer:** 68	Debería devolver error de Código numérico.
23	Actualizar stock con cantidad a reponer mayor a cero.	**Código:** 6204 **Stock actual:** 6 **Cantidad a reponer:** 0	Debería devolver error de Cantidad a reponer mayor a cero.
24	Actualizar stock con cantidad a reponer mayor a cero.	**Código:** 7401 **Stock actual:** 9 **Cantidad a reponer:** -12	Debería devolver error de Cantidad a reponer mayor a cero.
25	Registrar baja de producto con código	**Código:** 5F2S	Debería devolver error de Código

	únicamente numérico.		únicamente numérico.
26	Registrar baja de producto con código que no supere los diez dígitos.	**Código:** 69804376015	Debería devolver error de Código que no supere los diez dígitos.
27	Registrar baja de combo con código únicamente numérico.	**Código:** 3E9W	Debería devolver error de Código únicamente numérico.
28	Registrar baja de combo con código que no supere los diez dígitos.	**Código:** 98703164285	Debería devolver error de Código que no supere los diez dígitos.
29	Verificar el registro de las ofertas vigentes con código únicamente numérico.	**Código:** 5S9X	Debería devolver error de Código únicamente numérico.
30	Registrar nuevo producto con cantidad de compra para definir el precio únicamente numérico.	**Código:** 12 **Descripción:** Combo Completo **Precio Unitario:** 3880,50 **Cantidad Actual:** 5 **Disponibilidad:** True **Cantidad de compra p/definir precio:** 62X	Debería devolver error de Cantidad de compra para definir precio únicamente numérico.

CONCLUSIÓN

El desarrollo de este trabajo permitió tener una experiencia del largo proceso, con las dificultades que acarrea, que implica el desarrollo de un sistema, poniendo en juego los conocimientos adquiridos durante estos años, como así también los incorporados en esta cátedra.

Se logró comprender, como todo resulto tener un mayor nivel de dificultad y complejidad de lo que parecía ser en un principio, a medida que se pensaba más en la cruda realidad que nos rodea.

Desde mi punto de vista personal, la dificultad del trabajo estuvo reflejada más que todo en relación al código, ya que, al no contar con un buen nivel de programación, me impidió poder dejar expresado en el código otras ideas que me hubieran permitido explotar mejor las funcionalidades planteadas en relación al dominio.

Como crítica personal y constructiva, fue un trabajo provechoso que ayudo a visualizar y comprender como todos los aprendizajes que venimos adquiriendo, se lograron relacionar y confabularse en un desarrollo global, que apunto a un propósito específico. El hecho de haber trabajado en teams, más allá de ser un trabajo individual, ayudo mucho a poder apoyarse en otros compañeros para solventar dudas o inconvenientes que se presentaban en el camino. Si creo, que temas particulares, que hacen y afectan a la parte del código, deberían de instruirse desde algún caso práctico más amplio, parecido al nivel de dificultad que implica este proyecto, para poder comprender y tener una mejor visión de cómo aplicarlo posteriormente.

BIBLIOGRAFÍA

Jacobson, Ivar; Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software.

Ian Sommerville. Ingeniería de Software (9na edición).

Carpeta compartida con el material de la clase de Diseño de Sistemas.