Sprint #1

Merkit

Taller de ingeniería de software
04/11/2020

Tamara Badilla
Sebastián Pacheco
Álvaro Valverde

“Merkit es una página web de administración para minimarket gratis, fácil y rápida de usar. Si bien la página no posee sistema de boleta, permite al usuario manejar fácilmente el stock, las ventas diarias realizadas, y acceso a la nube para que sus datos sean accesibles mediante cualquier dispositivo donde inicie sesión.”

Dentro del primer Sprint del sistema Merkit, se realizó la obtención de requisitos junto con la arquitectura del sistema, además de la identificación preliminar de Clases Conceptuales, y diseño de las Clases a utilizar (también desde una vista preliminar).

Requerimientos

Funcionales

Dentro de los requisitos funcionales de la aplicación, se encuentran:

- *Login:* La aplicación web cuenta con un inicio de sesión vinculado a un correo Gmail, que cargue sus preferencias o la ruta de los archivos del usuario (en caso de ser pertinente.

- *Sistema de inventario:* El sistema debe poseer inventario, para así obtener una lista clara de productos los disponibles con su precio y stock correspondiente. Esto integrará las siguientes funciones:
  - Acceso a la base de datos.
  - Listar todos los productos en la base de datos junto con su nombre, stock (opcional), etc.
  - Agregar/quitar y editar productos de la base de datos.

- *Resumen de Ventas:* La página web permite mostrar el total de ventas, para hacer un análisis rápido, ofreciendo al usuario acceso a este cuando lo necesite. Esto integrará:
  - Acceso al historial de ventas realizadas.

- Acceso al almacenamiento local.
- Filtro de ventas según lo que se requiera (orden alfabético, mayor a, etc.)

➔ *Calculador de vuelto:* La página debe entregar el valor del vuelto en monedas/billetes según lo requiera para facilitarle al cajero(a) este proceso, cuando éste finalice la Venta.

➔ *Sistema de ventas:* el sistema debe mostrar los elementos que se puedan vender, junto con la capacidad de añadirlos a una lista para agregarlos a una Venta. Esto integrará:
- Barra de búsqueda para acelerar el proceso.
- Un despliegue gráfico que facilite la identificación de productos, y su adición a una venta.
- Un despliegue gráfico de las ventas y sus productos agregados, permitiendo la modificación de precio y cantidad.
- Botón de finalizar venta que modifica la BD ventas y BD inventario.

➔ *Configuración de usuario:* El sistema identificará al usuario con un servicio de Google, lo que le permitirá tener un respaldo de sus datos.

No Funcionales

Por el contrario, los requisitos no-funcionales que se detectaron para construir el sistema son:

➔ *Usabilidad:* Se espera que la página web sea *intuitiva*, y permita que trabajadores sin experiencia en el local puedan realizar ventas. Se piensa llevar esto a cabo mediante un seguimiento cercano a los usuarios, y su experiencia con la aplicación.

➔ *Backup:* El sistema se conectará a la nube de Google para así tener la posibilidad de acceder a sus datos desde cualquier dispositivo, en cualquier lugar (con conexión a internet). Esto servirá tanto de *respaldo* como de *portabilidad*.

Diseño Preliminar de Arquitectura del Sistema

Se decidió construir el sistema usando el tipo de arquitectura de Modelo-Vista-Controlador, ya que por el requisito funcional de Usabilidad, se espera separar lo más posible el lado de interacción de usuario de toda la logística que hay detrás.

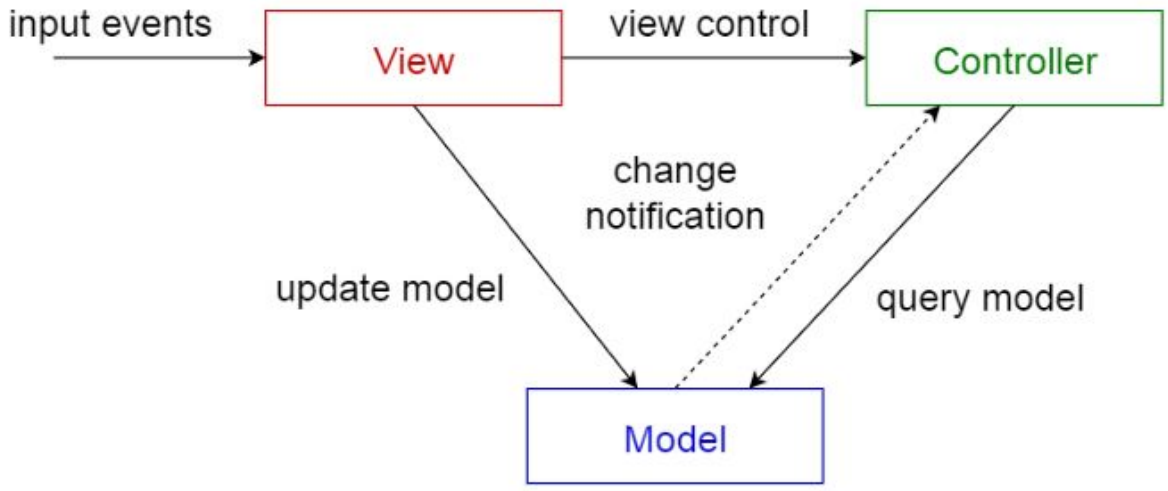


Diagrama MVC, Patrones comunes de arquitectura

➔ ***Controller:*** Corresponde a un componente centralizado que maneja la información de la aplicación. En particular, será el encargado de administrar los objetos tipo Producto que han sido creados con la información recibida del Modelo.

➔ ***View:*** Corresponde al componente de interfaz gráfica, que dispondrá de todas las funcionalidades del software, clasificado en pestañas. Cada pestaña tendrá sus propios botones y barras de búsqueda correspondientes, por lo que el Controller estará escuchando en todas las pestañas.

➔ ***Model:*** Corresponderá al componente que administrará la conexión con la Base de Datos, y entregará la información requerida al Controller. En posteriores Sprints se analizarán las siguientes opciones, y de ellas se escogerá una:

- a) El usuario guarda localmente los datos de los productos, y tiene la opción de tener un respaldo periódico en Google Drive. De ser así, el usuario tendrá opciones para determinar el directorio contenedor.
- b) El servidor del sitio web responde al cliente con todos los datos de los productos. De ser así, el usuario en su web solo guardará productos temporales, y solo recibirá la información requerida en querys.

Clases Conceptuales

Para identificar las clases involucradas, preliminarmente realizamos un Análisis Lingüístico de los Casos de Uso identificados, y de acuerdo a estos desambiguamos el lenguaje natural. Posteriormente se mostrará una lista de Clases Candidatas, y finalmente seleccionamos las Clases Conceptuales que serán parte de nuestra arquitectura MVC.

Merkit será ocupado como sinónimo para **Sistema**.

Caso de Uso Básico de Venta:

1. **Cliente** entra al minimarket que utiliza app web **Merkit**.
2. **Merkit** se abre en la **Pestaña Ventas**, y muestra todos los **Productos** disponibles.
3. **Cajero(a)** de minimarket comienza una nueva **Venta**.
4. **Cajero(a)** busca en app el **nombre del producto** ó su **ID (código de barras)** deseado por **Cliente**.
5. **Cajero(a) registra** producto en **Venta** actual.
6. **Merkit** muestra el precio final de la **Venta**, junto con las **Descripciones** y sumas parciales de cada **Producto**.
7. **Cajero(a)** informa y solicita el monto al **Cliente**.
8. **Cliente** realiza el **Pago**.
9. **Cajero(a)** recibe el dinero y **finaliza Venta** en **Merkit**.
10. **Merkit** guarda **Venta** en la **Base de Datos** del **Usuario**, en la tabla **Ventas**, junto con modificar el **Stock** del **Producto**.
11. **Cliente** se va feliz con sus **Productos** si es que pagó. Si no, se va triste.

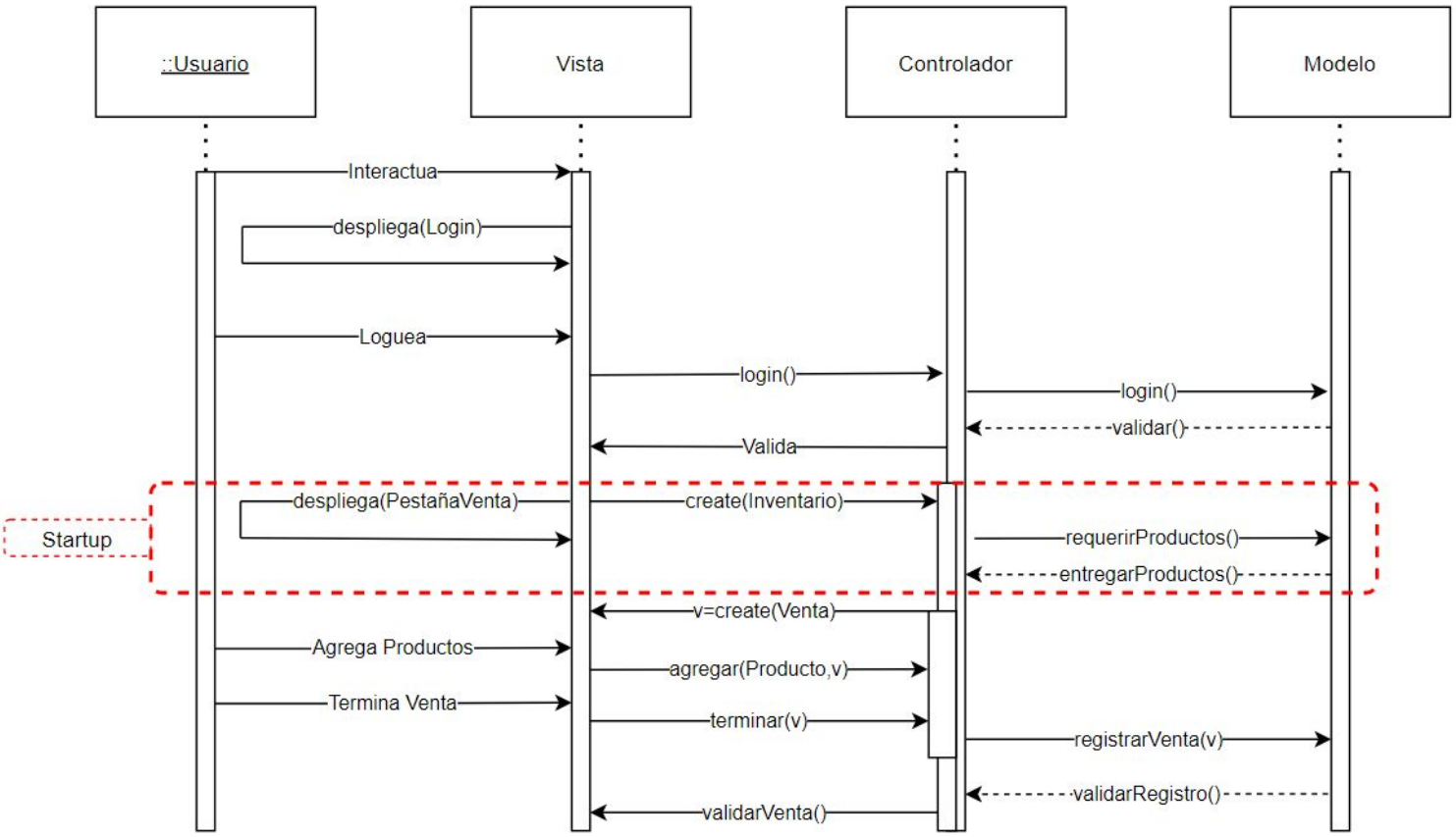


Diagrama de Secuencia "Caso de Uso Básico de Venta"

Flujo Alternativo 1:

(...)

7. **Cajero(a)** informa y solicita el monto al **Cliente**.
8. **Cliente** no puede **Pagar** por insuficiente presupuesto.
9. **Cajero(a) cancela** la **Venta**.
10. **Cliente** se va triste.

Flujo Alternativo 2:

(...)

7. **Cajero(a)** informa y solicita el monto al **Cliente**.
8. **Cliente** al ver el monto, decide llevar menos **Productos**.
9. **Cajero(a) modifica** la **Venta** y solicita el nuevo monto.
10. **Cliente** realiza el **Pago**.
11. **Cajero(a)** recibe el dinero y **finaliza Venta** en **Merkit**.

12. **Cliente** se va feliz con sus **Productos**

Caso de Uso Básico de Inventario:

1. Llega un novedoso cargamento de papas a un local que utiliza **Merkit.**
2. **Cajero(a) recibe** y **almacena Cargamento**.
3. **Cajero(a)** añade el **Producto** a su **Inventario** en **Merkit.**
4. **Merkit** añade **Producto** a **Base de Datos** de **Usuario**, en tabla **Inventario.**
5. **Cajero(a)** espera con ansias su próxima **Venta**.

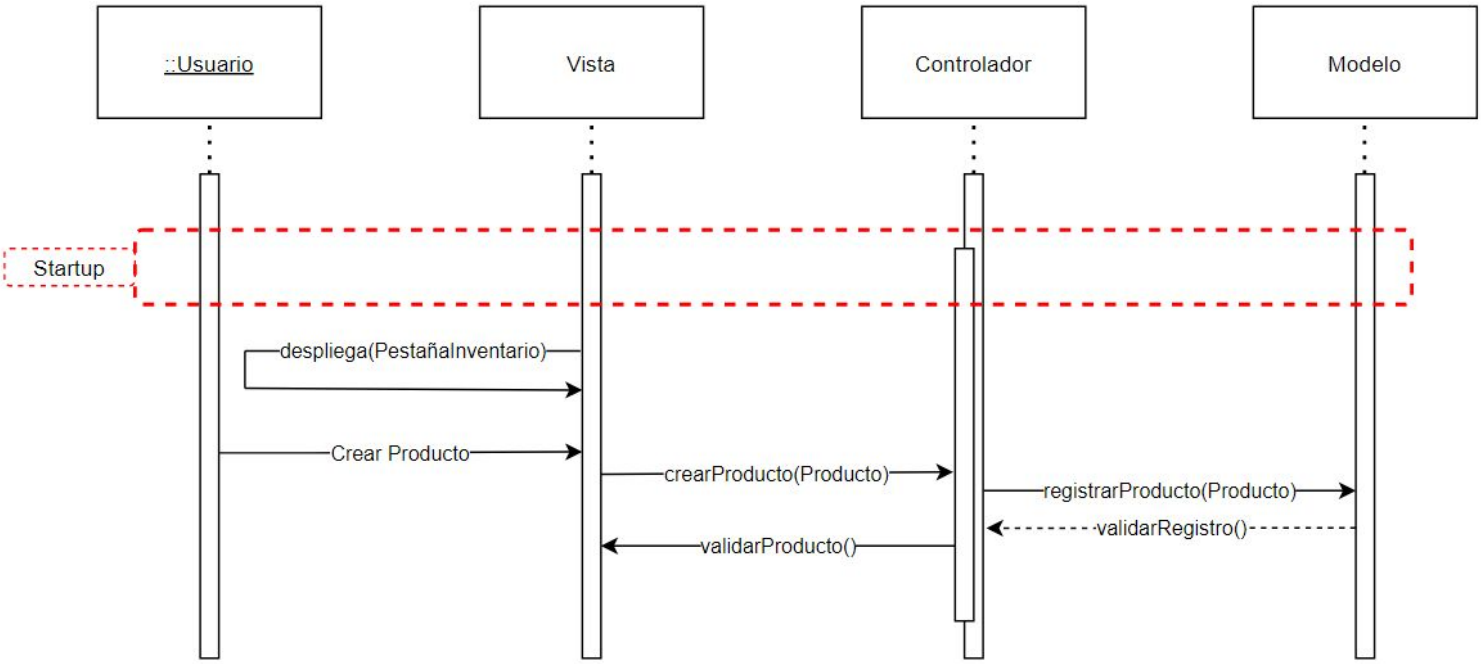


Diagrama de Secuencia "Caso de Uso Básico de Inventario"

Flujo Alternativo 1 (Modificar):

1. Llega un nuevo cargamento de zanahorias a un local que utiliza **Merkit.**
2. **Cajero(a) recibe** y **almacena Cargamento**.
3. **Cajero(a) modifica** el stock del **Producto** en su **Inventario** en **Merkit.**
4. **Merkit modifica** el **Producto** a **Base de Datos** de **Usuario**, en tabla **Inventario.**
5. **Cajero(a)** espera con ansias su próxima **Venta**.

Flujo Alternativo 2 (Eliminar):

1. **Cajero(a)** ve **Inventario Merkit**
2. **Cajero(a)** se da cuenta que existe un **Producto** duplicado
3. **Cajero(a)** procede a **eliminar Producto** duplicado
4. **Merkit elimina Producto** de la **Base de Datos**

Caso de Uso Básico de Resumen:

1. **Usuario** revisa en la **Pestaña de Resumen** el **detalle** de las **Ventas Realizadas**.
2. **Merkit** despliega solo las **Ventas del día** con los **detalles** de cada **Producto**.

Flujo Alternativo 1 (VIP):

1. **Usuario** revisa en la **Pestaña de Resumen** el **detalle** de las **Ventas Realizadas**.
2. **Usuario** ha pagado la suscripción **Merkit,** así que **Merkit** despliega cuatro opciones; **Resumen Diario**, **Semanal**, **Mensual** y **Anual**.
3. **Usuario** busca en la **Barra de Búsqueda** algún **Producto** en específico.
4. **Merkit** despliega todas **Ventas** que incluyen el **Producto** buscado, y los resultados en **orden temporal**.

Clases Candidatas:

Listado de Frases Nominales:

Pestaña Venta	*Pestaña Resumen*	*Pestaña Inventario*
Inventario	*Producto*	*Venta*
Cajero	*Usuario*	*Merkit*
Descripción de Producto	*Stock*	*Pago*

Selección de Clases Conceptuales:

El siguiente es un desarrollo preliminar de las Clases a implementar en el proyecto, las Clases Conceptuales. Se describen de acuerdo a su propósito en nuestra arquitectura MVC.
Merkit: Será el sistema en su conjunto, y es con quien interacciona el Usuario.

Vista:	● PestañaVenta: Despliegue gráfico en la web, donde se crearán las Ventas y se le añadirán Productos. También, tendrá un botón para finalizar la venta, modificar detalles del listado y calcular vuelto. ● PestañaResumen: Despliegue gráfico en la web, se desplegará información en pantalla con los datos obtenidos durante el día, la semana, mes, etc. Además de permitir un filtro de estos mismos. ● PestañaInventario: Despliegue gráfico en la web, permite el manejo y edición de los productos existentes en sistema, además de agregar nuevos a la base de datos.
Controlador:	● Usuario: Será quien manipule Merkit, es decir, el dueño del minimarket o algún trabajador. Dentro del controlador, será la clase que permitirá al Usuario configurar su cuenta (Google) y/o algunas opciones de la aplicación. ● Producto: Clase que necesariamente encapsula los detalles de cada Producto, según los datos recibidos del Modelo. Además tendrá funciones que faciliten su despliegue en Vista. ● Inventario: Será una clase contenedora de Productos, que además permitirá la filtración de Productos según sea necesario. Los detalles de los productos serán modificables, y tendrá las funciones pertinentes para comunicarse con el Modelo, y filtrar búsquedas para la Vista. ● Venta: Clase que encapsula los detalles de cada Venta, y reflejarán los Productos y detalles monetarios de una transacción. ● HistorialVentas: Será una clase contenedora de Ventas, que además permitirá la filtración de Ventas según sea necesario y esté habilitado para el tipo de usuario. Además tendrá funciones que faciliten su despliegue en Vista.

Modelo:	No se manejan clases, pues el modelo será una clase por sí misma, única y tendrá todas las responsabilidades de comunicarse con la Base de Datos, y manejar el flujo de datos hacia el Controlador del usuario. Es decir, será parte del Back-end.

Responsabilidades y Patrones:

A continuación se describen las responsabilidades (hacer/conocer) y patrones identificados de acuerdo a las Clases Conceptuales, además de otras identificadas:

PestañaVenta	**Hacer:** Mostrar todos los aspectos gráficos que permitirán el proceso de una Venta. **Conocer:** Deberá conocer el Inventario, las Imágenes de los Productos, y debe tener acceso al HistorialVentas. **Patrones:** Experto respecto a clases Galería, Inventario e HistorialVentas, ya que necesita listar los Productos con su respectiva Imagen, además de finalizar una Venta (archivarla).
PestañaResumen	**Hacer:** Mostrar todos los aspectos gráficos que permitan la visualización de resúmenes. **Conocer:** Deberá conocer el HistorialVentas y tendrá acceso al Inventario para identificar los Productos de las Ventas. **Patrones:** Experto respecto a clases HistorialVentas e Inventario, ya que necesita conocer los detalles de las Ventas, además de identificar a los Productos involucrados.
PestañaInventario	**Hacer:** Mostrar todos los aspectos gráficos que permitan la visualización y edición del Inventario (y sus Productos). **Conocer:** Deberá conocer el Inventario y la Galería, pues permitirá crear nuevos

	Productos e integrarles alguna imagen, la cual necesariamente será *normalizada* para optimizar el uso del espacio.
Patrones:	Experto respecto a las clases Inventario y Galería, ya que necesita conocer los detalles de los Productos, y de sus Imágenes correspondientes.

Galeria	
Hacer:	Deberá facilitar la creación de nuevas Imágenes, con su correspondiente ruta y/o archivo, y la posible normalización de los archivos, refiriéndonos a dimensiones y tamaño en disco.
Conocer:	Deberá conocer (y contener) los objetos tipo Imagen, y conoce su relación individual con cada Producto.
Patrones:	Creador de objetos de clase Imagen.

Imagen	
Hacer:	Permite la modificación de sí mismo.
Conocer:	Conocerá su propio ID y la ruta que lleva al archivo correspondiente.
Patrones:	Bajo acoplamiento entre objetos e Imagen, y Alta cohesión entre Imagen y Galeria.

Inventario	
Hacer:	Permite crear, modificar y filtrar nuevos objetos de tipo Producto.
Conocer:	Conoce todos los Productos instanciados.
Patrones:	Creador de objetos de clase Producto

Producto	
Hacer:	Permite la modificación de sí mismo.
Conocer:	Conocerá su ID, nombre, precio, stock, descripción, código de barra, marca, ID de imagen y tipo unitario (si es por unidad ó gramaje).
Patrones:	Bajo acoplamiento entre objetos y Producto, y Alta cohesión entre Producto e Inventario

HistorialVentas	

Hacer:	Permite crear, modificar y filtrar objetos de tipo Venta.
Conocer:	Conoce todas las Ventas ya creadas o instanciadas.
Patrones:	Creador de objetos de clase Venta

Ventas	

Hacer:	Permite la modificación de sí mismo.
Conocer:	Conocerá su relación con objetos tipo Producto ya creados, cantidades y la fecha en la que se realizan las ventas.
Patrones:	Bajo acoplamiento entre objetos y Ventas, y Alta cohesión entre Ventas y HistorialVentas, y Creador de Productos, ya que utiliza estrechamente objetos de este tipo.

Usuario	

Hacer:	Validar la cuenta del usuario con un identificador Google que cargue sus preferencias.
Conocer:	Deberá (en caso de aplicarse) conocer la ruta contenedora de los Productos y/o Imágenes.
Patrones:	Experto respecto a conocer la ruta contenedora de los datos.

Modelo	

Hacer:	Alimentará las clases creadoras Inventario, HistorialVentas y Galería. Se comunicará directamente con la Base de Datos SQL.
Conocer:	Conoce y almacena toda la información del negocio, como Productos, Imágenes y Ventas.
Patrones:	Experto respecto a los datos guardados.

Diagrama General de Clases:

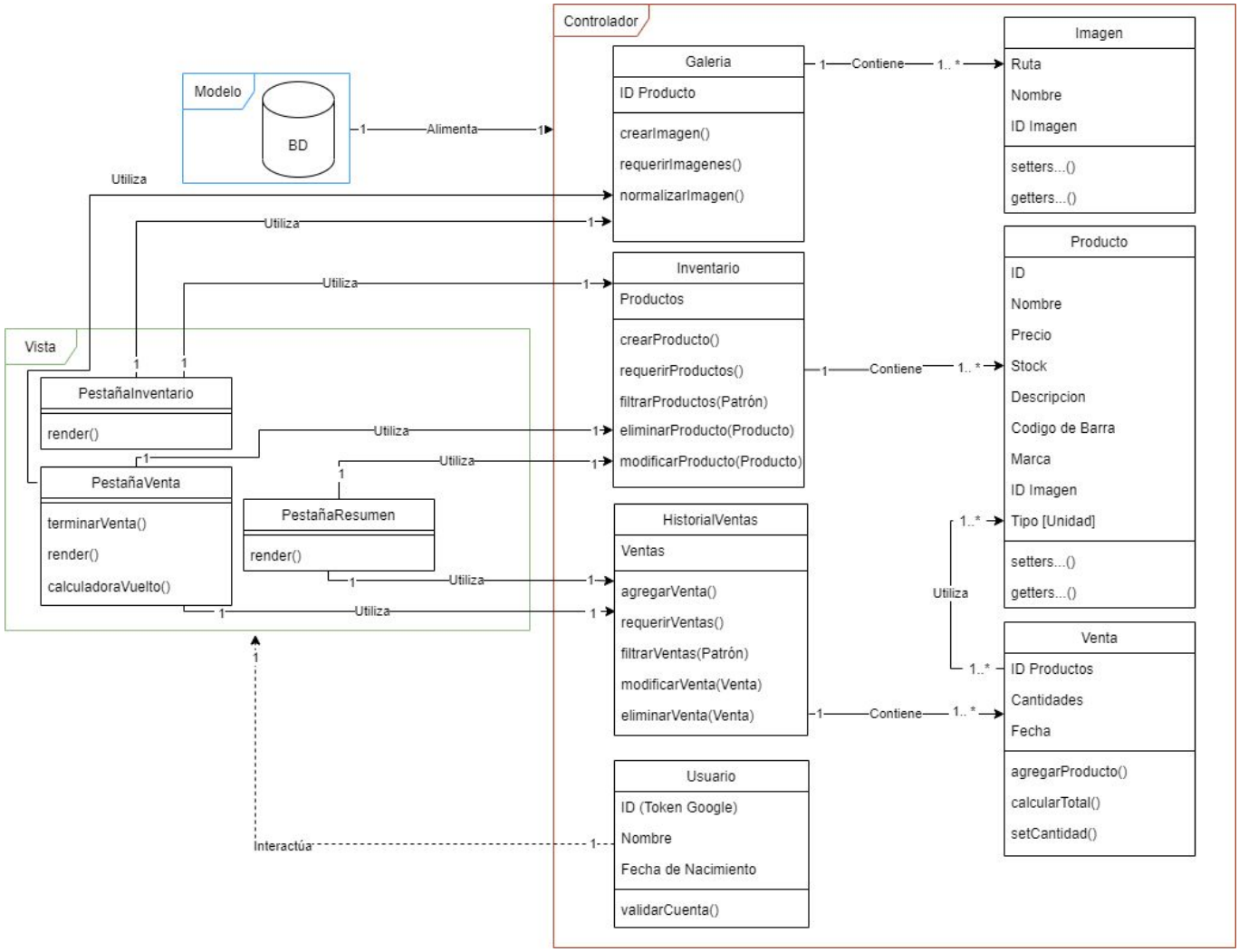
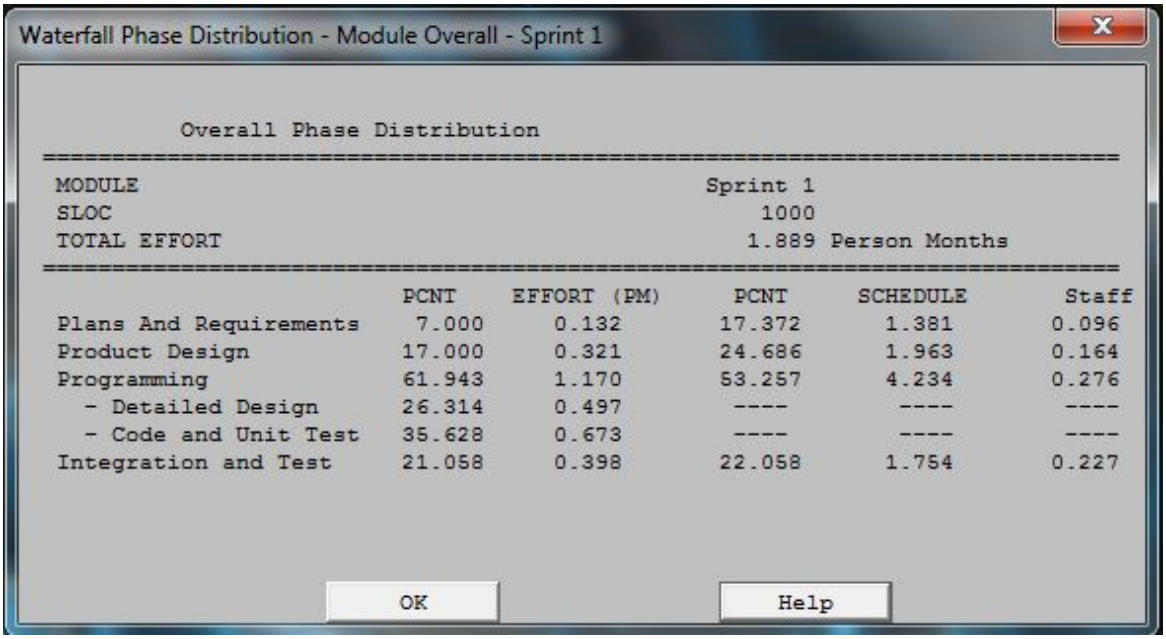


Diagrama General de Clases

Estimación Primer Sprint



Planificación Redmine

Se realizó la planificación de los 6 sprints preliminares, pero solo se muestran los primeros dos sprints, que se rigen por la siguiente carta Gantt: