

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 su integración en 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:

- [RF1] *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).
- [RF2] *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.
- [RF3] *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.)
- [RF4] *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.
- [RF5] *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.
- [RF6] *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*.

Análisis de Riesgos:

Asumiendo que los riesgos se pueden categorizar en las siguientes tres categorías, encontramos:

a) Riesgos de Planificación o Recursos:

- i) Los integrantes del equipo tienen un conocimiento bastante similar, por lo que la salida de alguno de ellos puede no ser significativa respecto a los conocimientos necesarios para continuar con el proyecto.
- ii) Una vez definida la planificación, es seguro que una iteración completa se dedique al desarrollo de las Bases de Datos. Ninguno de los integrantes tiene experiencia (real) implementando BDs, por lo que no conseguir los conocimientos necesarios puede retrasar esa iteración. Es por esto que se considerará como la iteración de más duración (4 semanas como mínimo).
- iii) Dado que estamos en un proceso formativo y supervisado, puede que se nos propongan cambios importantes durante la documentación o la implementación, lo que puede retrasar el desarrollo. Sin embargo, se espera que este riesgo sea bajo, y el equipo pueda superar oportunamente las asignaciones.
- iv) Considerando que somos estudiantes, también es bueno tener en cuenta nuestra carga académica. Sin embargo, no compartimos la totalidad de ramos inscritos, por lo que pueden haber márgenes que permitan que al menos todos podamos trabajar una misma semana, pero distintos días. Es decir, existe un bajo riesgo que esto retrase el proyecto.
- v) Un riesgo medianamente alto puede ser un Paro Estudiantil, el cual no impedirá el desarrollo por parte del equipo, pero si la revisión con los Profesores monitores.

b) Riesgos de Calidad o Performance:

- i) Nuestra poca experiencia en Bases de Datos también pueden implicar resultados precarios de usabilidad, que podrían ser bastante optimizados. Sin embargo, este riesgo puede ser reducido pues el ramo contempla una sección de Refactorización.
- ii) La carga académica de los integrantes además de afectar la planificación, también puede afectar en la calidad del software, resultando en una documentación incompleta o un código mal diseñado o poco optimizado.
- iii) En caso de implementarse la autorización vía Google, existe un pequeño riesgo de que el usuario no tenga una cuenta creada, y le parezca molesto crearla.

c) Riesgos del Negocio:

- i) No se identifican riesgos claros para el Negocio, pues el resto de equipos se encuentran desarrollando proyectos de ámbitos muy diferentes a nuestro, como lo son restaurantes y hospedaje, por lo que no habrían problemas de competencia. Además, dentro del mercado existen software con un propósito similar al nuestro, pero son alternativas mucho menos asequibles.

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.

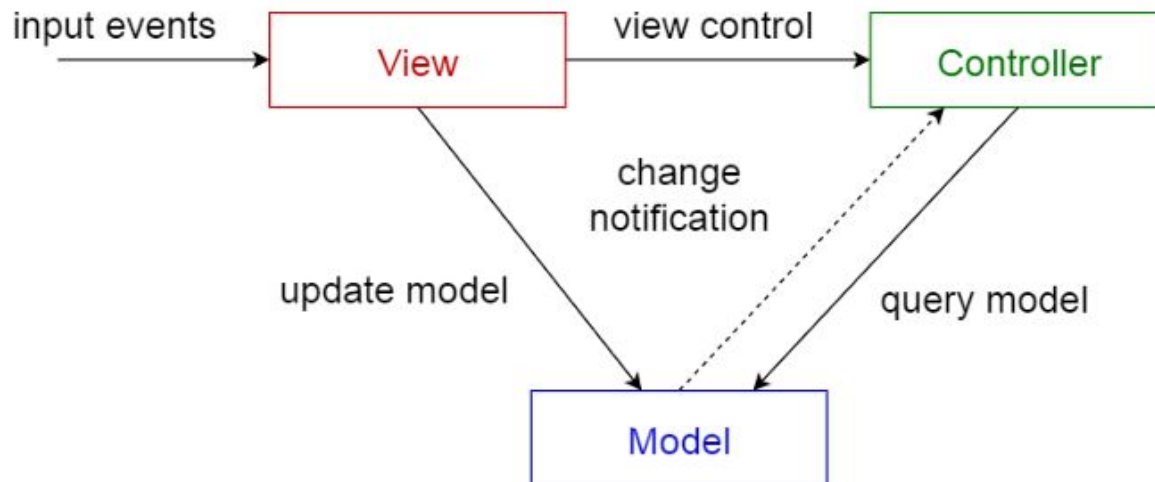


Diagrama MVC, Patrones comunes de arquitectura

- **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 queries.

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.

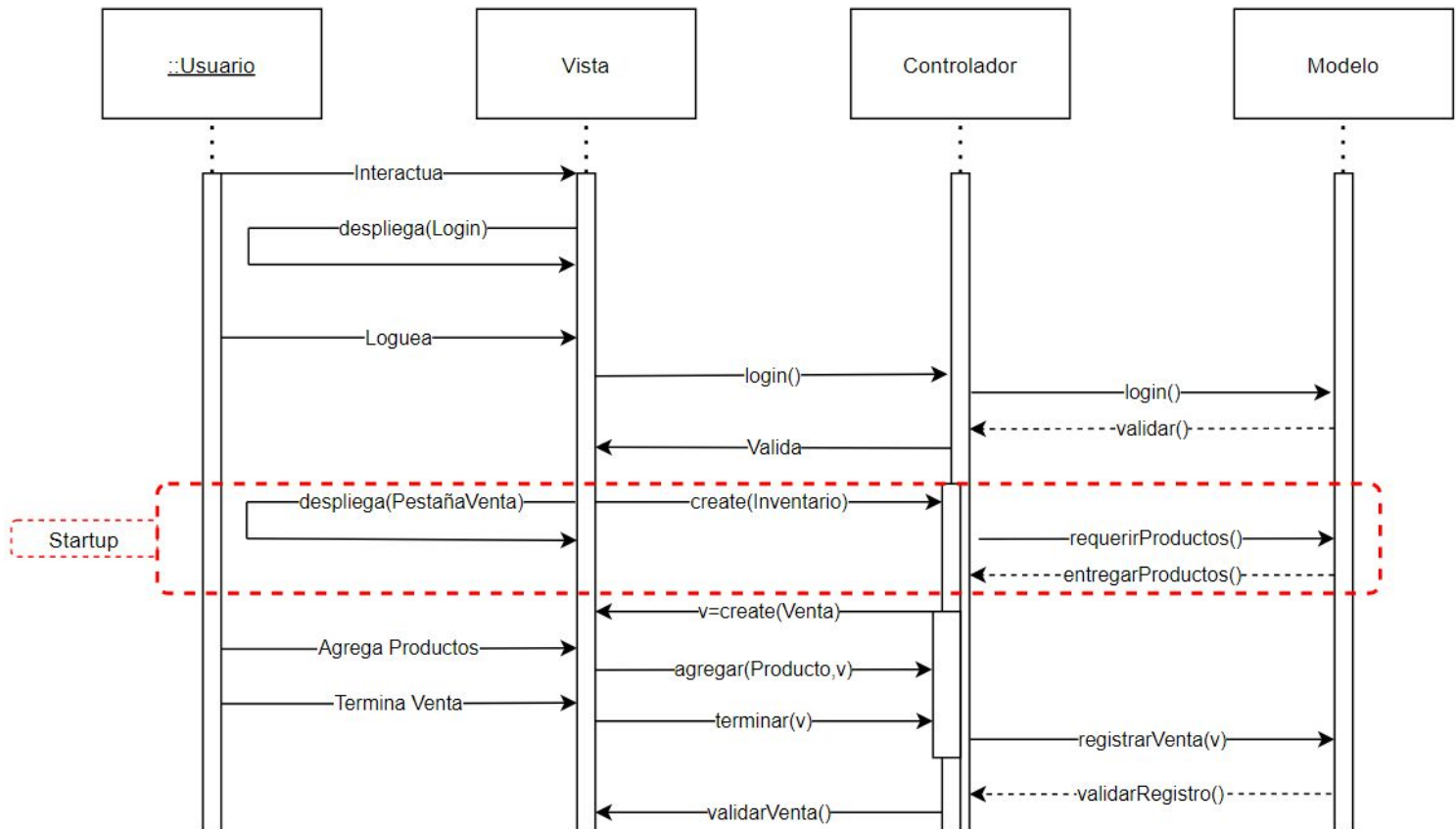


Diagrama de Secuencia "Caso de Uso Básico de Venta"

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**.

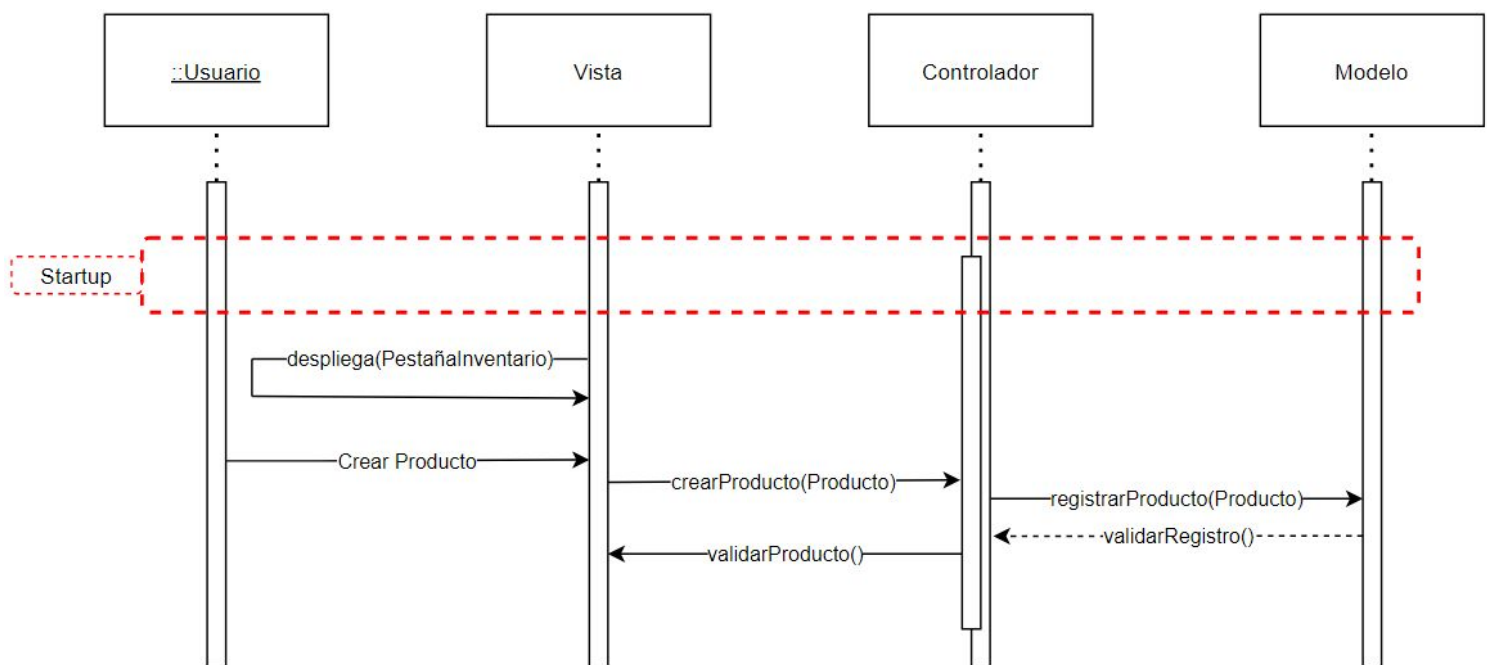


Diagrama de Secuencia "Caso de Uso Básico de Inventario"

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:

<i>Pestaña Venta</i>	<i>Pestaña Resumen</i>	<i>Pestaña Inventario</i>
<i>Inventario</i>	<i>Producto</i>	<i>Venta</i>
<i>Cajero</i>	<i>Usuario</i>	<i>Merkit</i>
<i>Descripción de Producto</i>	<i>Stock</i>	<i>Pago</i>

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:	<ul style="list-style-type: none">● <u>PestañaVenta:</u> 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.● <u>PestañaResumen:</u> 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.● <u>PestañaInventario:</u> 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:	<ul style="list-style-type: none">● <u>Usuario:</u> 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.● <u>Producto:</u> Clase que necesariamente encapsula los detalles de cada Producto, según los datos recibidos del Modelo. Además tendrá funciones que faciliten su despliegue en Vista.● <u>Inventario:</u> 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.● <u>Venta:</u> Clase que encapsula los detalles de cada Venta, y reflejarán los Productos y detalles monetarios de una transacción.● <u>HistorialVentas:</u> 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	<table> <tr> <td>Hacer:</td><td>Mostrar todos los aspectos gráficos que permitirán el proceso de una Venta. También debe incluir un despliegue que facilite el cálculo de los vueltos para el Usuario.</td></tr> <tr> <td>Conocer:</td><td>Deberá conocer el Inventario, las Imágenes de los Productos, y debe tener acceso al HistorialVentas.</td></tr> <tr> <td>Patrones:</td><td>Experto respecto a clases Galería, Inventario e HistorialVentas, ya que necesita listar los Productos con su respectiva Imagen, además de finalizar una Venta (archivarla).</td></tr> </table>	Hacer:	Mostrar todos los aspectos gráficos que permitirán el proceso de una Venta. También debe incluir un despliegue que facilite el cálculo de los vueltos para el Usuario.	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).
Hacer:	Mostrar todos los aspectos gráficos que permitirán el proceso de una Venta. También debe incluir un despliegue que facilite el cálculo de los vueltos para el Usuario.						
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	<table> <tr> <td>Hacer:</td><td>Mostrar todos los aspectos gráficos que permitan la visualización de resúmenes.</td></tr> <tr> <td>Conocer:</td><td>Deberá conocer el HistorialVentas y tendrá acceso al Inventario para identificar los Productos de las Ventas.</td></tr> <tr> <td>Patrones:</td><td>Experto respecto a clases HistorialVentas e Inventario, ya que necesita conocer los detalles de las Ventas, además de identificar a los Productos involucrados.</td></tr> </table>	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.
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	<table> <tr> <td>Hacer:</td><td>Mostrar todos los aspectos gráficos que permitan la visualización y edición del Inventario (y sus Productos).</td></tr> </table>	Hacer:	Mostrar todos los aspectos gráficos que permitan la visualización y edición del Inventario (y sus Productos).				
Hacer:	Mostrar todos los aspectos gráficos que permitan la visualización y edición del Inventario (y sus Productos).						

	<table> <tr> <td>Conocer:</td><td>Deberá conocer el Inventario y la Galería, pues permitirá crear nuevos Productos e integrarles alguna imagen, la cual necesariamente será <i>normalizada</i> para optimizar el uso del espacio.</td></tr> <tr> <td>Patrones:</td><td>Experto respecto a las clases Inventario y Galería, ya que necesita conocer los detalles de los Productos, y de sus Imágenes correspondientes.</td></tr> </table>	Conocer:	Deberá conocer el Inventario y la Galería, pues permitirá crear nuevos Productos e integrarles alguna imagen, la cual necesariamente será <i>normalizada</i> 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.		
Conocer:	Deberá conocer el Inventario y la Galería, pues permitirá crear nuevos Productos e integrarles alguna imagen, la cual necesariamente será <i>normalizada</i> 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	<table> <tr> <td>Hacer:</td><td>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.</td></tr> <tr> <td>Conocer:</td><td>Deberá conocer (y contener) los objetos tipo Imagen, y conoce su relación individual con cada Producto.</td></tr> <tr> <td>Patrones:</td><td>Creador de objetos de clase Imagen.</td></tr> </table>	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.
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	<table> <tr> <td>Hacer:</td><td>Permite la modificación de sí mismo.</td></tr> <tr> <td>Conocer:</td><td>Conocerá su propio ID y la ruta que lleva al archivo correspondiente.</td></tr> <tr> <td>Patrones:</td><td>Bajo acoplamiento entre objetos de tipo Imagen, y Alta cohesión con clase Galeria.</td></tr> </table>	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 de tipo Imagen, y Alta cohesión con clase Galeria.
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 de tipo Imagen, y Alta cohesión con clase Galeria.						
Inventario	<table> <tr> <td>Hacer:</td><td>Permite crear, modificar y filtrar nuevos objetos de tipo Producto.</td></tr> <tr> <td>Conocer:</td><td>Conoce todos los Productos instanciados.</td></tr> <tr> <td>Patrones:</td><td>Creador de objetos de clase Producto.</td></tr> </table>	Hacer:	Permite crear, modificar y filtrar nuevos objetos de tipo Producto.	Conocer:	Conoce todos los Productos instanciados.	Patrones:	Creador de objetos de clase Producto.
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	<table> <tr> <td>Hacer:</td><td>Permite la modificación de sí mismo.</td></tr> <tr> <td>Conocer:</td><td>Conocerá su ID, nombre, precio, stock, descripción, código de barra, marca, ID de imagen y tipo unitario (si es por unidad ó gramaje).</td></tr> <tr> <td>Patrones:</td><td>Bajo acoplamiento entre objetos tipo Producto, y Alta cohesión con</td></tr> </table>	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 tipo Producto, y Alta cohesión con
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 tipo Producto, y Alta cohesión con						

	<table> <tr> <td></td><td>clase Inventario</td></tr> </table>		clase Inventario				
	clase Inventario						
HistorialVentas	<table> <tr> <td>Hacer:</td><td>Permite crear, modificar y filtrar objetos de tipo Venta.</td></tr> <tr> <td>Conocer:</td><td>Conoce todas las Ventas ya creadas o instanciadas.</td></tr> <tr> <td>Patrones:</td><td>Creador de objetos de clase Venta.</td></tr> </table>	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.
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	<table> <tr> <td>Hacer:</td><td>Permite la modificación de sí mismo.</td></tr> <tr> <td>Conocer:</td><td>Conocerá su relación con objetos tipo Producto ya creados, cantidades y la fecha en la que se realizó la Venta.</td></tr> <tr> <td>Patrones:</td><td>Bajo acoplamiento entre objetos tipo Ventas, y Alta cohesión entre con la clase HistorialVentas. También es Creador de Productos, ya que utiliza estrechamente objetos de este tipo.</td></tr> </table>	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 realizó la Venta.	Patrones:	Bajo acoplamiento entre objetos tipo Ventas, y Alta cohesión entre con la clase HistorialVentas. También es Creador de Productos, ya que utiliza estrechamente objetos de este tipo.
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 realizó la Venta.						
Patrones:	Bajo acoplamiento entre objetos tipo Ventas, y Alta cohesión entre con la clase HistorialVentas. También es Creador de Productos, ya que utiliza estrechamente objetos de este tipo.						
Usuario	<table> <tr> <td>Hacer:</td><td>Validar la cuenta del usuario con un identificador Google que cargue sus preferencias.</td></tr> <tr> <td>Conocer:</td><td>Deberá (en caso de aplicarse) conocer la ruta contenedora de los Productos y/o Imágenes.</td></tr> <tr> <td>Patrones:</td><td>Experto respecto a conocer la ruta contenedora de los datos.</td></tr> </table>	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.
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	<table> <tr> <td>Hacer:</td><td>Alimentará las clases creadoras Inventario, HistorialVentas y Galería. Se comunicará directamente con la Base de Datos SQL.</td></tr> <tr> <td>Conocer:</td><td>Conoce y almacena toda la información del negocio, como Productos, Imágenes y Ventas.</td></tr> <tr> <td>Patrones:</td><td>Experto respecto a los datos guardados.</td></tr> </table>	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.
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:

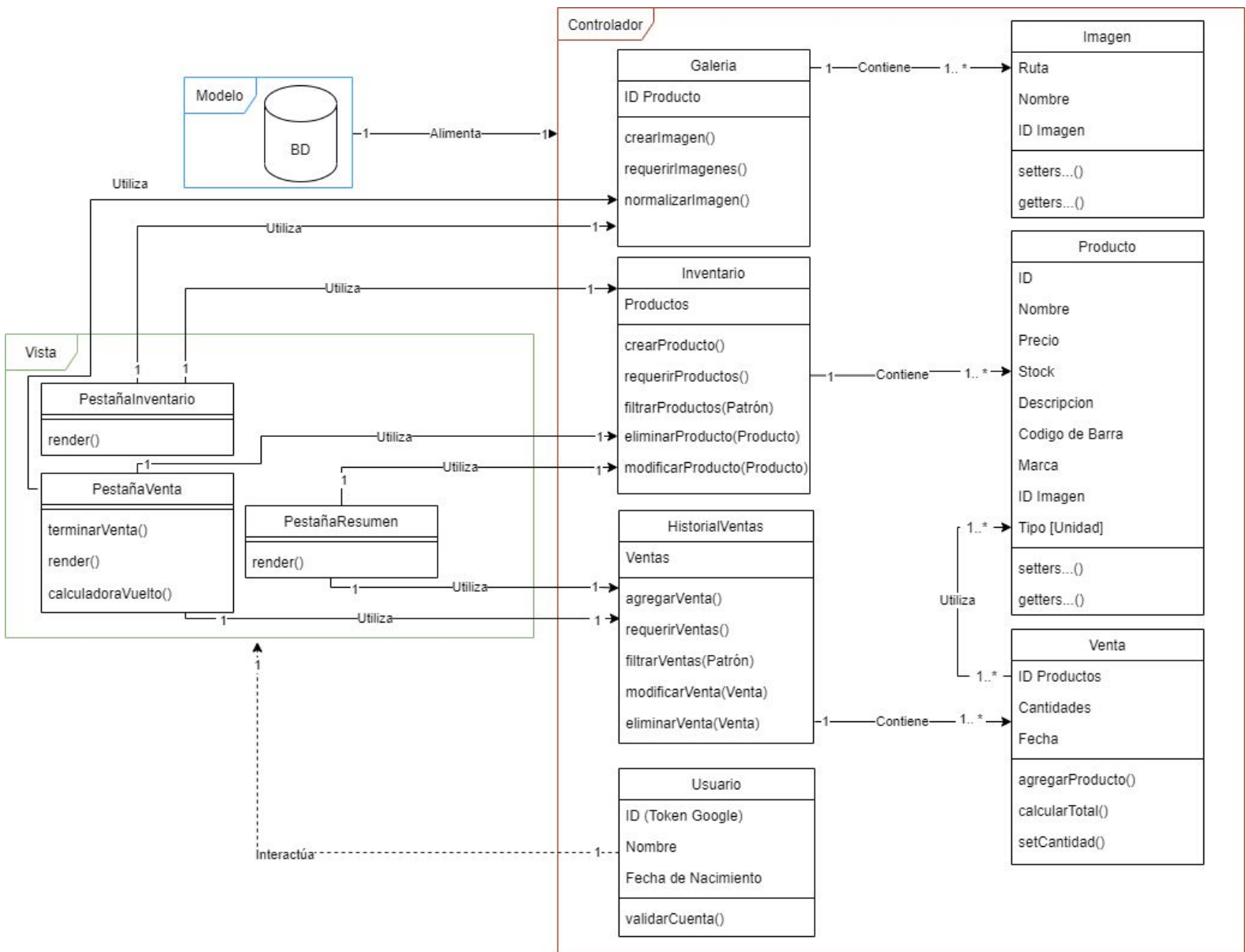
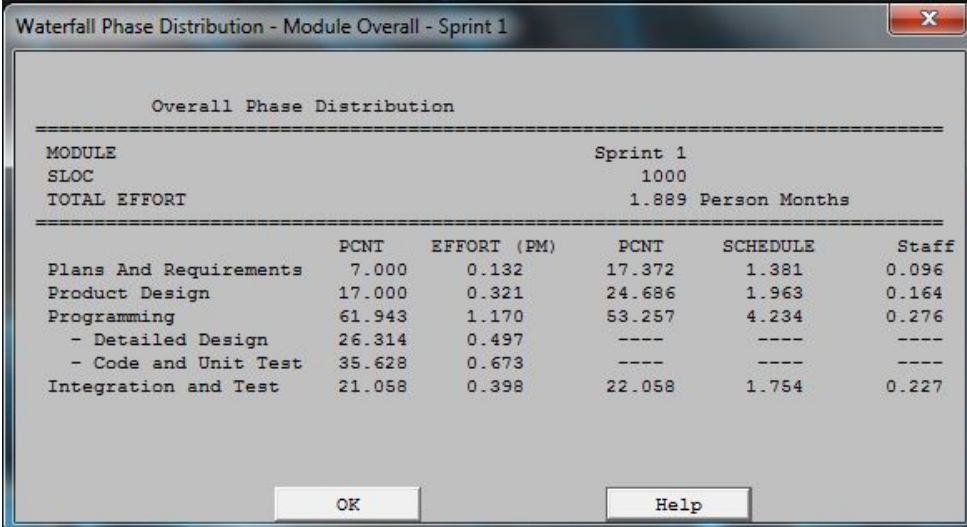


Diagrama General de Clases

Estimación Primer Sprint

En resumen, para el presente Sprint seguiremos la siguiente planificación:



Waterfall Phase Distribution - Module Overall - Sprint 1

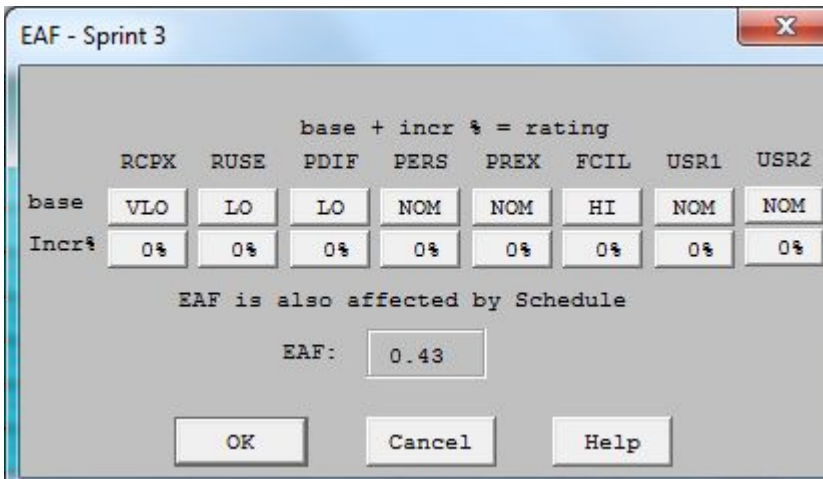
Overall Phase Distribution

MODULE	Sprint 1				
SLOC	1000				
TOTAL EFFORT	1.889 Person Months				
	PCNT	EFFORT (PM)	PCNT	SCHEDULE	Staff
Plans And Requirements	7.000	0.132	17.372	1.381	0.096
Product Design	17.000	0.321	24.686	1.963	0.164
Programming	61.943	1.170	53.257	4.234	0.276
- Detailed Design	26.314	0.497	----	----	----
- Code and Unit Test	35.628	0.673	----	----	----
Integration and Test	21.058	0.398	22.058	1.754	0.227

OK Help

Estimación de duración:

Para lo anterior, los parámetros EAF utilizados fueron:



EAF - Sprint 3

base + incr % = rating

	RCPX	RUSE	PDIF	PERS	PREX	FCIL	USR1	USR2
base	VLO	LO	LO	NOM	NOM	HI	NOM	NOM
Incr%	0%	0%	0%	0%	0%	0%	0%	0%

EAF is also affected by Schedule

EAF: 0.43

OK Cancel Help

Con todo esto, nuestra estimación para el proyecto completo se refleja en la siguiente tabla:

The screenshot shows the USC-COCOMO II 2000.4 software interface. The project name is 'MerkitWeb' and the scale factor is 17.95. The development model is set to 'Early Design'. The main table lists six sprints with their respective module sizes, labor rates, effort, and risk. Below the table, summary statistics are provided for the entire project.

X	Module Name	Module Size	LABOR Rate (\$/month)	EAF	Language	NCM Effort DEV	EST Effort DEV	PROD	COST	INST COST	Staff	RISK
	Sprint 1	S:1000	0.00	0.43	USR 1	4.4	1.9	529.3	0.00	0.0	0.2	0.0
	Sprint 2	F:1375	0.00	0.43	USR 1	6.0	2.6	529.3	0.00	0.0	0.3	0.0
	Sprint 3	F:1100	0.00	0.43	USR 1	4.8	2.1	529.3	0.00	0.0	0.3	0.0
	Sprint 4	F:1045	0.00	0.43	USR 1	4.6	2.0	529.3	0.00	0.0	0.2	0.0
	Sprint 5	F:825	0.00	0.43	USR 1	3.6	1.6	529.3	0.00	0.0	0.2	0.0
	Sprint 6	F:770	0.00	0.43	USR 1	3.4	1.5	529.3	0.00	0.0	0.2	0.0

	Estimated	Effort Sched	PROD	COST	INST	Staff	RISK
Total Lines of Code:	6115						
Hours/PM:	120.00						
Optimistic	7.7	7.0	790.0	0.00	0.0	1.1	
Most Likely	11.6	8.0	529.3	0.00	0.0	1.5	0.0
Pessimistic	17.3	9.0	352.9	0.00	0.0	1.9	

Es importante recalcar que el archivo de planificación COCOMO II se encuentra en [GitHub/EstimacionMerkit.est](https://github.com/SebastianMerkit/EstimacionMerkit.est).

Con estos antecedentes, refinamos las fechas, y decidimos concretar las siguientes conversiones de los plazos:

Nombre	EFFORT	SCHEDULE
Sprint 1	1.889	19 días -> 3 semanas
Sprint 2	2.598	26 días -> 4 semanas
Sprint 3	2.078	21 días -> 3 semanas
Sprint 4	1.974	20 días -> 3 semanas
Sprint 5	1.559	16 días -> 3 semanas
Sprint 6	1.455	15 días -> 2 semanas

Planificación

Se realizó una planeación de acuerdo a lo que se iba a desarrollar en Merkit, ésta se dividió en seis sprints en total, descritos de manera general en una carta Gantt dentro del proyecto en Redmine (http://146.83.216.195:8080/projects/web_merkit).

Luego de realizar esta planificación, se especificó de un modo más profundo el primer y segundo sprint, esto ya que estos se deben ir construyendo a medida que se lleva a cabo el proyecto.

