Sprint #1

Merkit

Taller de ingeniería de software
21/10/2020

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, por lo que en este documento se encuentran los requisitos funcionales, no funcionales y la arquitectura que se utilizará.

Requerimientos

Funcionales

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

- ➔ *Login:* La aplicación web cuenta con un inicio de sesión vinculado al correo, a través

- ➔ *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 productos según lo que se requiera (orden alfabético, mayor a, etc.)

➔ *Sistema de calculadora de vuelto:* La página debe entregar el valor del vuelto en monedas/billetes según lo requiera para facilitarle al cajero(a) este proceso.

➔ *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.
- Botón de finalizar venta que modifica la BD ventas y BD inventario.
- Vista de elementos a agregar y elementos agregados.

➔ *Configuración de usuario:* el sistema debe permitir al usuario modificar sus datos, tales como su foto de perfil, su correo, su contraseña, etc.

No Funcionales

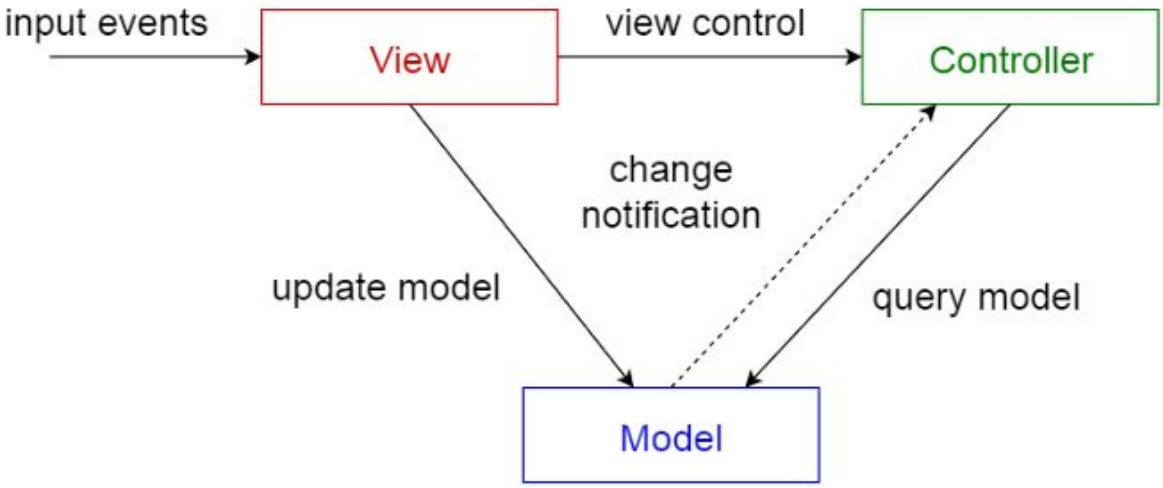
Por el contrario, los requisitos no funcionales que se detectaron para construir el sistema son:

➔ *Usabilidad:* Si bien es esperado que la página web esté disponible para usuarios de todo tipo, el cliente específico que se tiene en mente al construir el sistema es alguien que no posee mucha experiencia con la tecnología, es por esto que la usabilidad es muy importante. Se piensa llevar esto a cabo mediante un seguimiento cercano a los usuarios.

➔ *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.

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. Tendrá una copia local del inventario de la empresa, además de un respaldo en la nube (Google Drive), y tendrá la capacidad de requerir los datos a conveniencia.

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 a implementar 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.

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** 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**.

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 y Selección:

Listado de Frases Nominales:

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

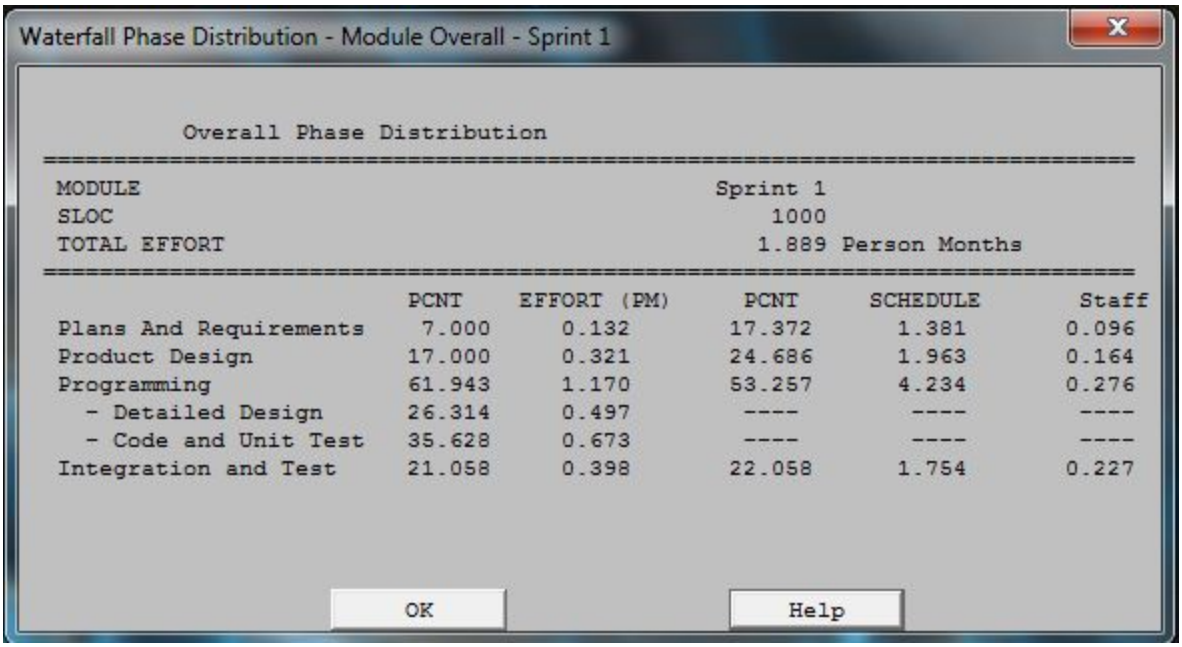
Selección y desambiguado:

Mertkit: 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, o modificar detalles del listado. ● 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) o algunas opciones de la aplicación. ● Producto: Clase que necesariamente encapsulará 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. ● 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.

Planificación



Planificación Redmine: