**Proyecto final inventarios con sistema de notificaciones para productos**

**Tipos de usuarios:**

**Usuario Principal**

El Usuario se encarga de entrar al sistema, verificar las existencias de los productos y ver cuales están disponibles y cuáles no, si hay productos faltantes, generara una orden que será enviada al proveedor.

**Proveedor**

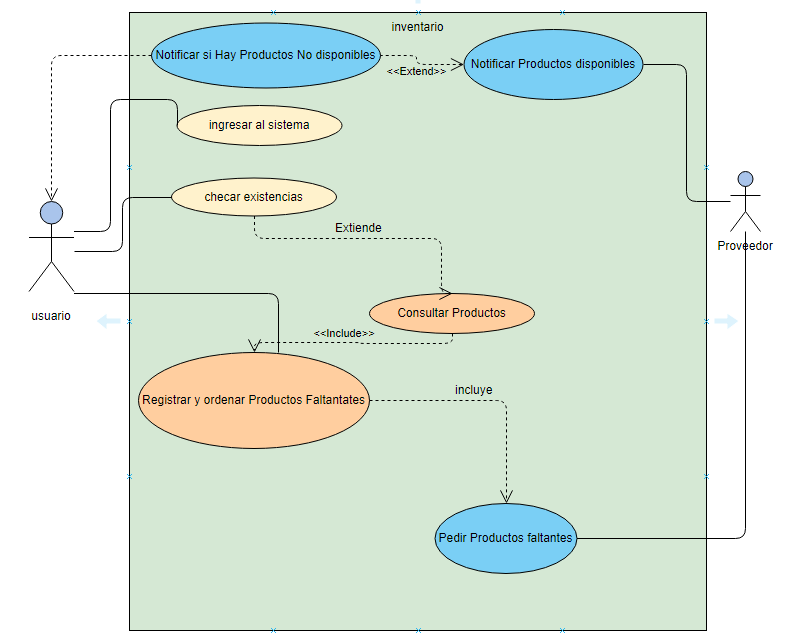
el proveedor recibe la orden y en base a lo recibido, deberá checar los productos en existencia, de no haber algún producto, se le notificará de vuelta al usuario antes de que se realice un pedido para confirmar si está seguro.

**Requerimientos funcionales:**

* El sistema mostrará al usuario una pantalla en donde podrá ver cuatro columnas, las cuales serán un código de barra, proveedor, nombre del producto, cantidad disponible.
* Se mostrará la cantidad de cada producto y en rojo se mostrará el producto que está por agotarse o ya no sea existente.
* El usuario tendrá en pantalla una notificación de los productos que tiene disponible el proveedor para el envío para que de esta manera no haya malos entendidos a la hora de la entrega.

**Requerimientos no funcionales:**

* El software de inventario por el momento solo podrá ser utilizado por el usuario (tienda) y por un proveedor, se planea que más adelante pueda haber más proveedores.
* La precisión y rapidez del sistema depende de la persona que lo usé y que tan rápido verifiqué el sistema.
* Para ingresar se necesita el nombre de la tienda y del proveedor al igual que registrar los productos que tengan en común.
* El sistema necesita conexión wi-fi para funcionar.



**Definición de estándar de codificación de java**

**Nombres de las Interfaces**

Los nombres de interfaces utilizarán el sufijo Interface y estarán compuestos por palabras con la primera letra en mayúscula (CamelCase). Se debe evitar el uso de abreviaciones que dificulten la comprensión del código, esto debido a la .

**Funciones**

Las funciones deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase).

No se permiten caracteres especiales debido a las limitaciones de java.

**Variables**

Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos.

**Constantes**

Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("\_"). Todas serán declaradas como public static final.

**Comentarios**

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma.

**Plantilla para el control de versiones**

|  |  |
| --- | --- |
| Codificador/a: |  |
| Responsable |  |
| Proyecto |  |
| Modulo |  |
| Funciones realizadas |  |
| Resumen de la prueba |  |
| Comentarios |  |
| Versión |  |
| Aprobado |  |
| Fecha de realización |  |

**Herramientas a utilizar**

**Eclipse**

Como se sabe es un conjunto de herramientas de programación, por lo tanto, este software libre será utilizado para la elaboración de código, así como la edición del mismo código, además cuenta con un compilador, así que se realizaran las pruebas de validación correspondiente en el mismo entorno. Aunque es posible utilizar otras fuentes.

**Google hangouts**

Esta aplicación se usará para poder entablar reuniones con el profesor y entre los integrantes para poder tener una organización y planificación del trabajo mas eficientemente. Además de entablar conversaciones para la comprobación de avances del proyecto, esto apoyando los avances entre cada integrante. Incluso en la realización de pequeñas pruebas en vivo, para poder solucionar errores de codificación y documentación. Además de comentar los posibles requerimientos que pueden ser agregados mejorados o quitados.

**GitHub**

Se usará un repositorio en donde estarán como colaboradores cada uno de los integrantes del equipo. En el cual se estarán subiendo las modificaciones que se realicen en el código del programa, esto para asegurar que el código este siempre guardado para cualquier percance. Además, encontraremos los cambios hechos por el equipo con mayor facilidad gracias al uso de las ramas.

**Diseño.**

En esta fase se utiliza toda la información recogida en la fase de especificación y se propone una solución que responda a las necesidades del usuario y se pueda desarrollar. En esta fase se define la arquitectura de la aplicación. Es necesario detallar la estructura y la organización del programa y cómo se relacionan las distintas partes de la aplicación.

El diseño conlleva:

​​- Decidir la estructura de la interfaz

- Qué componentes gráficos se van a utilizar, y cómo se van a relacionar estos componentes

-​​Decidir la disposición (layout) de los componentes

- sirven para organizar los componentes contenidos en los mismos.

​​- Decidir el comportamiento de la interfaz: gestión de eventos

- Algunos componentes son controles: permiten reaccionar ante eventos del usuario. El comportamiento se especifica programando las respuestas a dichos eventos.

**Codificación.**

Esta fase consiste en la programación en Java de las especificaciones de diseño de la fase anterior. Durante esta fase de codificación o implementación se aplican normas de programación para facilitar la legibilidad de los programas Java.

**Prueba.**

En esta fase se compila y se ejecuta la aplicación para verificar que cumple con los requisitos funcionales y técnicos definidos en la fase de especificación. Si el programa no cumple con todos los requisitos, se puede deber a errores de diseño o de programación. En tal caso, es necesario corregir los errores que se hayan detectado y repetir de nuevo el proceso de diseño y codificación. Durante la fase de prueba se verifica que la aplicación cumple con los criterios de calidad establecidos en el proyecto: facilidad de uso, corrección, eficiencia, integridad, fiabilidad, flexibilidad, facilidad de mantenimiento, facilidad de prueba, portabilidad, capacidad de reutilización e interoperabilidad.

Una vez que la aplicación se ha probado y cumple con los requisitos establecidos, entonces se pone en funcionamiento y comienza la fase de operación para que sea utilizada para el fin con el que ha sido desarrollada.

**Mantenimiento.**

Una vez que la aplicación se ha puesto en operación da comienzo la fase de mantenimiento. En esta fase se corrigen errores de funcionamiento de la aplicación, se modifica la funcionalidad o se añaden las nuevas funcionalidades que demandan los usuarios. La fase de mantenimiento es la de mayor duración, pues pueden pasar muchos años desde el inicio de la operación hasta que el producto es retirado.