**ador de tienda**

**Modelo de Construcción y Pruebas de Software**

**Versión [1.0]**

**Historia de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| [dd/mm/aaaa] | [x.x] | [detalles] | [nombre] |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contenido**

**[Nombre del proyecto]** [**1**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.30j0zll)

**Modelo de Diseño Construcción de Software** [**1**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1fob9te)

**Versión [1.0]** [**1**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2et92p0)

**Historia de revisiones** [**1**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.tyjcwt)

**Contenido** [**2**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3dy6vkm)

**1.** **Introducción** [**3**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1t3h5sf)

1.1. Propósito [3](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.4d34og8)

1.2. Alcance [3](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2s8eyo1)

1.3. Definiciones, siglas y abreviaturas. [3](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.17dp8vu)

1.4. Referencias [3](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3rdcrjn)

1.5. Visión general [3](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.26in1rg)

**2.** **Diseño de Casos de Uso** [**3**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.lnxbz9)

2.1. Diseño del Caso de Uso [nombre del caso de uso 1] [3](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.35nkun2)

*2.1.1.* *Diagrama de paquetes* [*3*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1ksv4uv)

*2.1.2.* *Diagrama de Interacción* [*3*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.44sinio)

*2.1.3.* *Diseño de Flujo de eventos* [*3*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2jxsxqh)

*2.1.4.* *Requerimientos especiales o de implementación* [*4*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.z337ya)

2.2. Diseño del Caso de Uso [nombre del caso de uso 2] [4](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3j2qqm3)

**3.** **Diseño de Objetos** [**4**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1y810tw)

3.1. [Objeto 1] [4](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.4i7ojhp)

3.2. [Objeto 2] [4](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2xcytpi)

**4.** **Diseño de Subsistemas** [**4**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1ci93xb)

4.1. Subsistemas Específicos [4](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3whwml4)

*4.1.1.* *[Nombre del Subsistema Específico 1]* [*4*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2bn6wsx)

*4.1.2.* *[Nombre del Subsistema Específico 2]* [*5*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.23ckvvd)

4.2. Subsistemas de Soporte [5](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.ihv636)

*4.2.1.* *[Nombre del Subsistema de soporte 1]* [*5*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.32hioqz)

*4.2.2.* *[Subsistema de soporte 2]* [*5*](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1hmsyys)

**5.** **Revisión de la Interfaz de Usuario** [**5**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.41mghml)

5.1. Formatos individuales de interfaz de pantalla [6](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2grqrue)

5.2. Controles y elementos de diseño de interfaz de pantalla [6](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.vx1227)

5.3. Forma de navegación de interfaz de pantalla [6](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3fwokq0)

5.4. Formatos de impresión [6](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1v1yuxt)

**6.** **Diseño de Datos** [**6**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.4f1mdlm)

6.1. Modelo de Datos del Núcleo [6](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2u6wntf)

6.2. Modelo de Datos General [6](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.19c6y18)

6.3. Especificación de la Distribución de Datos [7](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3tbugp1)

**7. PLAN DE PRUEBAS DE SOFTWARE**…………………………………………………………     8

**Alcance de las Pruebas 8**

Elementos de Pruebas 8

Nuevas Funcionalidades a Probar 8

Pruebas de Regresión 9

Funcionalidades a No Probar 9

Enfoque de Pruebas (Estrategia) 9

**Criterios de Aceptación o Rechazo 9**

Criterios de Aceptación o Rechazo 9

Criterios de Suspensión 9

Criterios de Reanudación 9

**Entregables** [**9**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1egqt2p)

**Recursos** [**10**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3ygebqi)

Requerimientos de Entornos – Hardware [10](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2dlolyb)

Requerimientos de Entornos – Software [10](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.sqyw64)

Herramientas de Pruebas Requeridas [11](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3cqmetx)

Personal [11](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1rvwp1q)

Entrenamiento [12](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.4bvk7pj)

**Planificación y Organización** [**12**](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.2r0uhxc)

Procedimientos para las Pruebas [12](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.1664s55)

Matriz de Responsabilidades [13](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.3q5sasy)

Cronograma [13](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.25b2l0r)

Premisas [14](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.kgcv8k)

Dependencias y Riesgos [14](https://docs.google.com/document/d/1LzpokuRFgyde2ekk9MuTyX2gcZiSOLfg/edit#heading=h.34g0dwd)

1. **Introducción**

El presente documento explica qué es la construcción del software, sus actividades principales, su ejecución mediante RUP y SWEBOK, en qué consisten las interfaces, el desarrollo centrado en el usuario, desarrollo de prototipos, características del proceso de desarrollo de la interfaz y las fuentes de información de su etapa de diseño, los estándares del diseño de la interfaz y qué herramientas lo representan, la implementación de de la base de datos, etc.

1. **Propósito**

El propósito es desarrollar el sistema hasta el punto en que esté listo para la pre-producción de pruebas. Es el proceso en el cual se genera el programa que pueda ejecutarse, esta construcción está altamente ligada al diseño del sistema y a su vez estará muy relacionado con las prueba que validen la correctitud del sistema. Si es necesario, las primeras versiones del sistema se desarrollan, ya sea interna o externamente, para obtener los comentarios de los usuarios.

1. **Alcance**

Se ejecuta la creación detallada de software operativo y significativo, por medio de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración. Produce el mayor número de elementos de configuración que se necesitan gestionar en un proyecto de software (archivos de código fuente, contenido, casos de pruebas, etc.).

1. **Definiciones, siglas y abreviaturas.**

**Software**. Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

**Inventario**. Lista ordenada de bienes y demás cosas valorables que pertenecen a una persona, empresa o institución.

**Base de datos**. Recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático.

**Stock**. Conjunto de mercancías o productos que se tienen almacenados en espera de su venta o comercialización.

**Cmd**. Es el intérprete de comandos en OS/2 y sistemas basados en Windows

**X**. Cualquier cantidad o cosa.

**Alta**. Registro de nuevo producto en el sistema

**Baja**. Eliminación de un producto del sistema.

**Consulta**. Visualización de la información del producto buscado

**Modificación**. Actualización de la información del producto.

**Memoria secundaria**. Disco duro u otro tipo de almacenamiento independiente del sistema para guardar información para utilizarla después.

**Virus**. Software que tiene por objetivo alterar el funcionamiento normal de cualquier tipo de dispositivo informático, sin el permiso o el conocimiento del usuario principalmente para lograr fines maliciosos sobre el dispositivo.

**Hardware**. Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

**Sistema operativo**. Conjunto de programas que permite manejar la memoria, disco, medios de almacenamiento de información y los diferentes periféricos o recursos de nuestra computadora.

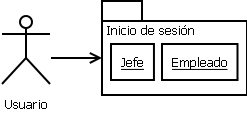
1. **Referencias**

|  |  |  |  |
| --- | --- | --- | --- |
| **Título** | **Número informe** | **Fecha** | **Organización** |
| No aplica | No aplica | No aplica | No aplica |
| No aplica | No aplica | No aplica | No aplica |

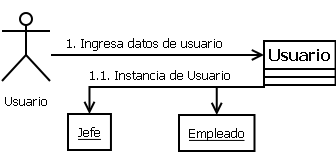
1. **Visión general**

Los fundamentos de construcción de software consisten en minimizar la complejidad, anticipar los cambios, construir para verificar y en estándares de construcción.

1. **Diseño de Casos de Uso**
2. **Diseño del Caso de Uso “login de usuario”**
3. **Diagrama de paquetes**

****

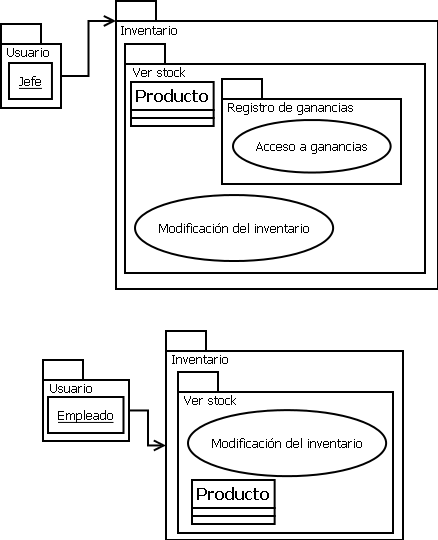
1. **Diseño de flujo de eventos**
2. El usuario configura el nivel de acceso como Jefe o empleado seleccionando una de las dos opciones.
3. Se crea el objeto Jefe o Empleado a partir de la clase Usuario.
4. **Diagrama de interacción**

****

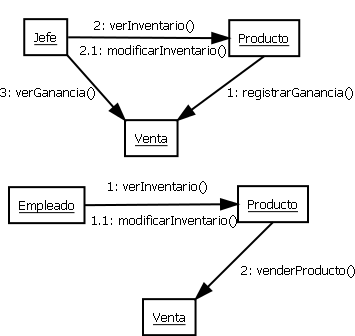
1. **Requerimientos especiales o de implementación**

* El usuario especifica tipo y nivel de acceso, los cuales determinan la forma de interactuar con el sistema posteriormente, además de los datos de login de seguridad.
* Se crea la instancia de la clase Usuario, se les da valor a sus atributos mediante sus métodos y un condicional determina si dicha instancia es de un usuario jefe o empleado.

1. **Diseño del Caso de Uso “visualizar inventario”**
2. **Diagrama de paquetes**

****

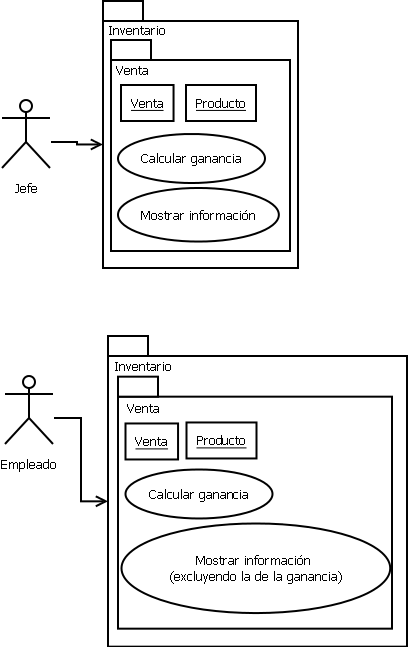
1. **Diseño de flujo de eventos**
2. Una vez creado el objeto usuario ya sea jefe o empleado, elige del menú principal la opción para visualizar el inventario, y dependiendo de su tipo de usuario, se visualiza el inventario de con o sin la información de ganancias.
3. Si se accedió al programa como empleado, se verán los productos y la posibilidad de modificar el inventario. Si se accede como jefe, se verán los productos, se podrá modificar el inventario y se podrá acceder a las ganancias.
4. **Diagrama de interacción**

****

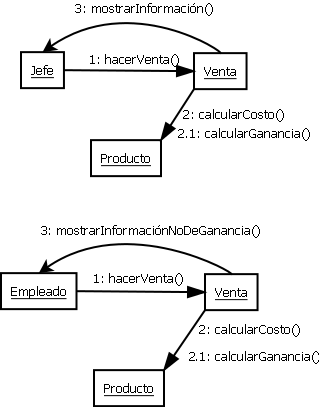
1. **Requerimientos especiales o de implementación**

* El usuario Jefe puede visualizar en el inventario la ganancia pero el Empleado no.
* El Empleado solo puede modificar el inventario y hacer ventas.

1. **Diseño del caso de uso “hacer una venta”**
   1. **Diagrama de paquetes**

****

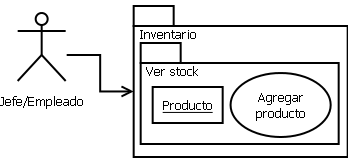
1. **Diseño de flujo de eventos**
2. El usuario selecciona la opción “Hacer una venta” del menú principal.
3. Se crea un objeto de la clase Venta.
4. Se calcula el costo.
5. Se calcula la ganancia.
6. El programa le muestra la información de la venta.
7. **Diagrama de interacción**

****

1. **Requerimientos especiales o de implementación**

* Que el sistema haga un objeto de la clase Venta, llenando sus atributos en base a pedirle al usuario el código del producto que se va a vender.
* Calcular el costo y ganancia del producto.
* Mostrar la información de la venta

1. **Diseño del caso de uso “alta de producto”**
   1. **Diagrama de paquetes**

****

1. **Diseño de flujo de eventos**
2. El usuario elige la opción “Añadir productos” del menú principal.
3. Entra al inventario y elige el stock en el cual añadir el producto.
4. Agrega el producto.
5. El stock registra el producto.

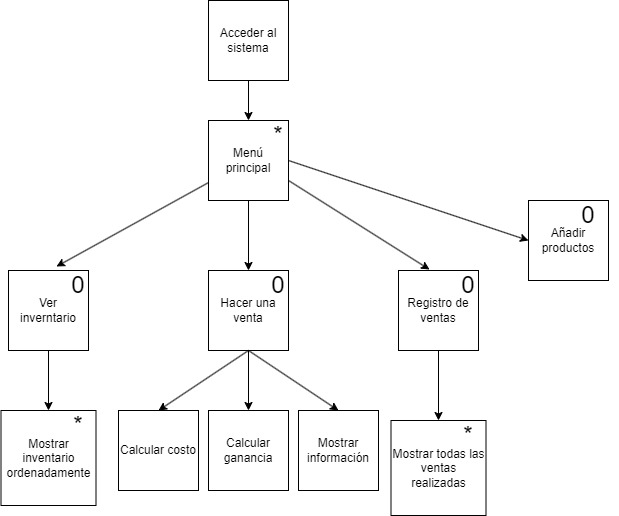
1. **Diagrama de interacción**

**https://lh5.googleusercontent.com/p-TeXUb4Lsp3iaVueLyhiv6FXGRF7X1qFr9X2CrwUE5Zwe8G6oxuq_yK5iRPZqIsCWXBfibux506rllEBl3EnVnWOHNn0m-o_p2ig7PBGgDMarXiFnxJcvQ7Hnt0unQCfxgXpCrtoNRqFKgUZcHbM4BqPM46IgE_J5SpY6uaGiFJRREtBnXO-_RZJFr0DbMo**

1. **Requerimientos especiales o de implementación**

* Para que el usuario agregue el producto al stock se usan los métodos setters de la clase Producto.
* Para que el stock registre dicho producto usa los getters de la clase Producto.

1. **Diagrama de Interacción**



1. **Diseño de Flujo de eventos**
2. El usuario accede al sistema e inicia sesión en uno de los dos usuarios (Jefe o Empleado), creándose así un objeto usuario de la clase Usuario.
3. Si en el menú principal el usuario selecciona “Ver inventario”, el inventario se muestra con los productos ordenados.
4. Si selecciona “Hacer una venta”, se crea un objeto de la clase venta y se calcula el costo y ganancia y se muestra la información.
5. Si se selecciona “Registro de ventas”, se muestra un listado de los objetos de la clase Ventas.
6. Si el usuario selecciona “Añadir producto”, se crea un objeto de la clase producto y el usuario le proporciona la información para llenar sus atributos mediante sus métodos setters, para luego registrarse en el inventario.
7. **Requerimientos especiales o de implementación**

* El sistema debe reconocer al tipo de usuario.
* Los cálculos, la creación de objetos y las relaciones entre estos se deben dar en el momento indicado.

**Entidades**

Usuario: Persona que estará manejando el software y que atiende a los clientes de la tienda.

Producto: Objeto que está a la venta.

Cliente: persona que compra los productos.

Pedidos: Conjunto de productos que un cliente apartó previamente.

**Relaciones**

Venta: Cuando un usuario registra que un cliente compró x producto,

Tienda: Los usuario atienden a los clientes.

Almacén: Se guardan los productos de algún pedido.

1. **Diseño del Caso de Uso [nombre del caso de uso 2]**
2. **Diseño de Objetos**
3. **Producto**

class Producto{

    private:

        string codigo;

        string nombre;

        float precio;

        string descripcion;

        int stock;

        float descuentos;

        float costoPorSurtir;

    public:

        Producto(string c, string n, float p, string d, int s, float dc, float cS){

            setCodigo(c);

            setNombre(n);

            setPrecio(p);

            setDescripcion(d);

            setStock(s);

            setDescuentos(dc);

            setCostoPorSurtir(cS);

        }

        void setCodigo(string c){

            this->codigo=c;

        }

        string getCodigo(){

            return this->codigo;

        }

        void setNombre(string n){

            this->nombre=n;

        }

        string getNombre(){

            return this->nombre;

        }

        void setPrecio(float p){

            this->precio=p;

        }

        float getPrecio(){

            return this->precio;

        }

        void setDescripcion(string d){

            this->descripcion=d;

        }

        string getDescripcion(){

            return this->descripcion;

        }

        void setStock(int s){

            this->stock=s;

        }

        int getStock(){

            return this->stock;

        }

        void setDescuentos(float dc){

            this->descuentos=dc;

        }

        float getDescuentos(){

            return this->descuentos;

        }

        void setCostoPorSurtir(float cS){

            this->costoPorSurtir=cS;

        }

        float getCostoPorSurtir(){

            return this->costoPorSurtir;

        }

};

1. **[Usuario]**

class Usuario{

    private:

        char nombre[TAMANIO\_NOMBRE\_USUARIO+1];

        char clave[TAMANIO\_CLAVE\_USUARIO+1];

        char puesto[TAMANIO\_PUESTO\_USUARIO+1];

        int nivelAcceso;

    public:

        Usuario(char \*n, char\* c, char\* p, int nA){

            setNombre(n);

            setClave(c);

            setPuesto(p);

            setNivelAcceso(nA);

        }

        void setNombre(char\* n){

            strcpy(this->nombre, n);

        }

        char\* getNombre(){

            return this->nombre;

        }

        void setClave(char\* c){

            strcpy(this->clave, c);

        }

        char\* getClave(){

            return this->clave;

        }

        void setPuesto(char\* p){

            strcpy(this->puesto, p);

        }

        char\* getPuesto(){

            return this->puesto;

        }

        void setNivelAcceso(int nA){

            this->nivelAcceso=nA;

        }

        int getNivelAcceso(){

            return this->nivelAcceso;

        }

};

1. **Diseño de Subsistemas**
2. **Subsistemas Específicos**
3. **[Archivo txt]**

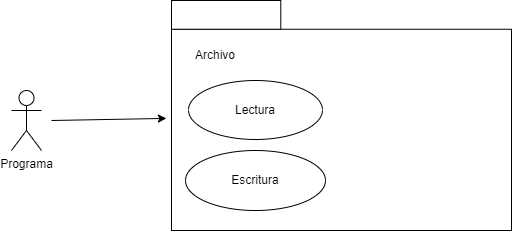
**Propósito**

El problema reside en que el software no guarda la información almacenada en la memoria dinámica al momento de cerrarse, por lo que se necesita una manera de poder guardar y leer esa información para utilizarla en futuras instancias.

**Función**

Lograr que el software pueda guardar  información en un archivo de texto para que al momento de cerrarlo no se pierda dicha información utilizando un archivo , y conseguir cargarla en la siguiente ejecución.

**Subordinados**



**Dependencias**

|  |  |  |
| --- | --- | --- |
| **Subsistema del que depende** | **Naturaleza de interacción** | **Características** |
| Escritura en archivo | Tener el permiso necesario, tener el archivo abierto. | Guarda la información dependiendo de la acción que se realiza, se escribe una coma después de cada dato guardado |
| Lectura de archivo | Debe existir un archivo, tener el permiso necesario, tener el archivo abierto. | Lee la información de un archivo ya existente al momento de ejecutar el programa. |

**Recursos**

**-Sistema operativo windows**

**-Espacio en memoria**

**Interfaces**

|  |  |  |
| --- | --- | --- |
| **Interacción** | **Método de interacción** | **Reglas de la interacción** |
| Ejecución del programa | Se invoca a la función de lectura y se capturan los datos en memoria temporal | Debe existir un archivo |
| Realizar una venta | Se invoca a la función de escritura y se añaden los datos de la venta | Se coloca al final del archivo dependiendo de la cantidad de ventas. |
| Actualizar el inventario | Se invoca a la función de escritura y se cambian ciertos datos sobre el inventario | EL usuario decide que producto va  a actualizar |

...

1. **Subsistemas de Soporte**

**NA**

1. **Revisión de la Interfaz de Usuario**

La pantallas se mostrarán en tal como viene la consola de Windows, además el usuario deberá verificar que la información que introduzca sea la correcta y que sea conforme al tipo de dato que se le solicita

1. **Formatos individuales de interfaz de pantalla**

Todas las pantallas serán mediante la consola, lo que significa que será solamente texto en fuente Terminal y fondo negro. El acomodo del texto será diferente dependiendo de la pantalla en el que se encuentre.

1. **Controles y elementos de diseño de interfaz de pantalla**

Se navegará mediante el uso del teclado, ya sea eligiendo una opción, agregando información o presionando la tecla enter, siempre que se presione la tecla adecuada se actualizará la pantalla.

1. **Forma de navegación de interfaz de pantalla**

Dentro de las pantallas solo se podrá utilizar el teclado (el programa indica que deberá presionar), algunas pantallas solamente muestran información y en otras se deberá elegir una opción.

1. **Formatos de impresión**

Todos los textos en pantalla tendrán el mismo formato: fuente Terminal tamaño 14px. En lagunas partes aparecerá el título/nombre de la pantalla en la parte superior, y dependiendo de la pantalla, se mostrará un listado o un menú.

1. **Diseño de Datos**

Se usan principalmente clases para definir los datos importantes:

Clase producto:

      string codigo

        string nombre

        float precio

        string descripcion

        int stock

        float descuentos

        float costoPorSurtir

Clase usuario:

       char nombre[TAMANIO\_NOMBRE\_USUARIO+1]

        char clave[TAMANIO\_CLAVE\_USUARIO+1]

        char puesto[TAMANIO\_PUESTO\_USUARIO+1]

        int nivelAcceso

También hay otras variables de tipo int externas a las clases para facilitar el manejo de los datos

1. **Modelo de Datos del Núcleo**

Nucleo producto:

class Producto{

    private:

        string codigo;

        string nombre;

        float precio;

        string descripcion;

        int stock;

        float descuentos;

        float costoPorSurtir;

    public:

        Producto(string c, string n, float p, string d, int s, float dc, float cS){

            setCodigo(c);

            setNombre(n);

            setPrecio(p);

            setDescripcion(d);

            setStock(s);

            setDescuentos(dc);

            setCostoPorSurtir(cS);

        }

        void setCodigo(string c){

            this->codigo=c;

        }

        string getCodigo(){

            return this->codigo;

        }

        void setNombre(string n){

            this->nombre=n;

        }

        string getNombre(){

            return this->nombre;

        }

        void setPrecio(float p){

            this->precio=p;

        }

        float getPrecio(){

            return this->precio;

        }

        void setDescripcion(string d){

            this->descripcion=d;

        }

        string getDescripcion(){

            return this->descripcion;

        }

        void setStock(int s){

            this->stock=s;

        }

        int getStock(){

            return this->stock;

        }

        void setDescuentos(float dc){

            this->descuentos=dc;

        }

        float getDescuentos(){

            return this->descuentos;

        }

        void setCostoPorSurtir(float cS){

            this->costoPorSurtir=cS;

        }

        float getCostoPorSurtir(){

            return this->costoPorSurtir;

        }

};

Núcleo usuario:

class Producto{

    private:

        string codigo;

        string nombre;

        float precio;

        string descripcion;

        int stock;

        float descuentos;

        float costoPorSurtir;

    public:

        Producto(string c, string n, float p, string d, int s, float dc, float cS){

            setCodigo(c);

            setNombre(n);

            setPrecio(p);

            setDescripcion(d);

            setStock(s);

            setDescuentos(dc);

            setCostoPorSurtir(cS);

        }

        void setCodigo(string c){

            this->codigo=c;

        }

        string getCodigo(){

            return this->codigo;

        }

        void setNombre(string n){

            this->nombre=n;

        }

        string getNombre(){

            return this->nombre;

        }

        void setPrecio(float p){

            this->precio=p;

        }

        float getPrecio(){

            return this->precio;

        }

        void setDescripcion(string d){

            this->descripcion=d;

        }

        string getDescripcion(){

            return this->descripcion;

        }

        void setStock(int s){

            this->stock=s;

        }

        int getStock(){

            return this->stock;

        }

        void setDescuentos(float dc){

            this->descuentos=dc;

        }

        float getDescuentos(){

            return this->descuentos;

        }

        void setCostoPorSurtir(float cS){

            this->costoPorSurtir=cS;

        }

        float getCostoPorSurtir(){

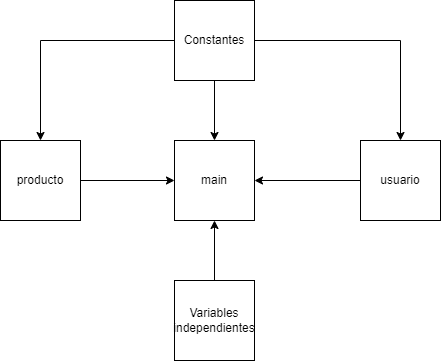
            return this->costoPorSurtir;

        }

};

1. **Modelo de Datos General**

Se cuenta con muchas constantes y variables aparte de las clases, estas son para facilitar la edición del código y para que el programa sea más óptimo. Éstas se encuentran tanto en el main como en el archivo de clases.



1. **Especificación de la Distribución de Datos**

Se cuenta con funciones de lectura y escritura de archivos, las cuales se encuentran en el main.

La función de lectura de archivos se ejecuta siempre que se abre el programa, y la de escritura se ejecuta siempre que se cierre el programa mientras el usuario haya salido mediante el uso del teclado en el menú principal, de otra manera no se habrá guardado.

**7. PLAN DE PRUEBAS DE SOFTWARE**

**Resumen ejecutivo**

Las pruebas intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo, consisten en verificar el comportamiento de un programa dinámicamente a través de un grupo finito de casos de prueba, debidamente seleccionados del, típicamente, ámbito de ejecuciones infinito, en relación al comportamiento esperado.

**Alcance de las pruebas**

La planificación de las pruebas y las propias actividades de diseño constituyen una información muy útil que ayuda a los diseñadores de software a identificar debilidades potenciales, hacer pruebas puede descubrir fallos, pero es el error el que se puede, y se debe, eliminar.

***Elementos de pruebas***

Se comprobará el guardar la información de los productos en el almacenamiento secundario(archivo txt) se verá que la información sea la misma que registramos y aparece en el programa.

***Nuevas funcionalidades a probar***

Se comprobará que el programa recupere correctamente la información guardada anteriormente de forma correcta.

***Pruebas de regresión***

La generación del código ya que se debe comprobar muchos tipos de combinaciones ya que al ser generado de la forma que lo hacemos puede tener problemas a lo largo del tiempo.

***Funcionalidades a no probar***

Información demasiada extensa, no vamos a comprobar esta parte ya que estamos asumiendo un caso muy exagerado en cuanto esto por lo que al menos por ahora no es necesario revisarlo.

***Enfoque de pruebas (estrategia)***

Emplearemos pruebas unitarias en las que nos centraremos en las funcionalidades del programa en el que emplearemos las pruebas antes mencionadas

**Criterios de aceptación o rechazo**

***Criterios de aceptación o rechazo***

El criterio de aceptación será una vez que comprobemos que todas las pruebas que hemos mencionadas estén funcionando en su totalidad y mientras tanto no serán aceptadas.

***Criterios de suspensión***

Si las pruebas tienen conflicto entre sí o el fallo puede estar relacionado entonces nos centraremos en las pruebas que veamos más importantes o que nos puedan servir más

***Criterios de reanudación***

Se reanudan cuando hayamos solucionado las pruebas seleccionadas anteriormente y luego de haber comprobado que no causen ningún otro conflicto.

**Entregables**

Se entregará un documento de plan de pruebas y casos de pruebas.

**Recursos**

***Requerimientos de entornos – Hardware***

Los requerimientos que se utilizaran son una computadora que permita correr el programa de manera óptima y nos permita realizar las pruebas necesarias.

No se necesitará red en el equipo pues el programa se ejecuta de manera local en el sistema, pero puede ser opcional para investigar programas que lleguen a surgir.

***Requerimientos de entornos – Software***

Dentro del software es necesario tener instalado windows como sistema operativo, además de tener codeblocks, programa que nos permitirá actualizar el código en caso de ser necesario.

***Herramientas de pruebas requeridas***

Como herramientas de software por el momento en el que se realiza el documento solamente manejaremos gráficas que muestran los datos que se manejan en un archivo de Excel (software de Office de Microsoft). Si se necesita un análisis más extenso, se utilizará un software gratuito para el análisis del proyecto.

***Personal***

Un líder de pruebas, dos especialistas de pruebas, dos diseñadores de software y cinco analistas de pruebas.

***Entrenamiento***

Las herramientas de prueba que se van a utilizar son el compilador de CodeBlocks en conjunto a los especialistas de prueba que primero ejecutarán el código del programa y darán su aprobación, después pasarán a los analistas de pruebas y aprobaran el proyecto corregido y sin fallas.

**Planificación y organización**

***Procedimientos para las pruebas***

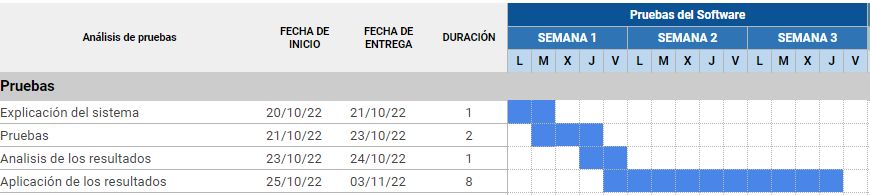
Intentar ingresar usuarios y contraseñas incorrectos al iniciar sesión para verificar la seguridad del sistema, usar información referente a varios productos en archivos de excel para introducirla al momento de hacer altas de productos y generar los objetos de la clase Producto e introducir un producto con datos incorrectos para verificar que las operaciones para realizar ventas detecten errores.

***Matriz de responsabilidades***



* Responsible (Ejecutador)
* Accountable (Responsable)
* Consulted (Consultado)
* Informed (Informado)

***Cronograma (Gantt)***



***Premisas***

Limitaciones de tiempo: tenemos para probar el funcionamiento del sistema a partir de acabar este documento hasta la fecha límite para entregar el sistema para su evaluación.

Metodología de pruebas: Utilizar un conjunto finito de casos de prueba de todo tipo para verificar el funcionamiento del sistema y ver su reacción ante datos intencionalmente erróneos.También para probar el almacenamiento en memoria secundaria.

Herramientas: Gráficas que muestran los datos que se manejan en un archivo de Excel.

***Dependencias y Riesgos***

Se depende de la objetividad en la opinión de las personas que hagan las pruebas (poco probable), en caso de que alguna sea poco creíble, se tendrá que volver a hacer la prueba con otra persona.

Se depende de la capacidad de la computadora para realizar las tareas (poco probable), si ocurre alguna falla y no se pueden continuar las pruebas, se necesitará esperar hasta que vuelva a funcionar, una forma de evitar esto es teniendo una computadora que pueda asegurarse que funcione perfectamente ante escenarios como cortes de energía o cargas pesadas de procesamiento de información.

**Referencias**

* I5898 - Ingeniería de Software I\_UNIDAD III Diseño Arquitectonico del Software
* I5898 - Ingeniería de Software I\_UNIDAD IV CONSTRUCCIÓN y PRUEBAS de Software.
* Descripción de Arquitectura de Sotware\_2