AndrómedaAppDesktop

Valeria Fuentes Surco

Christian Deyvid Ibañez Plata

Rodrigo López Monasterios

Kelly Olivera Daza

Adriana Sahonero Cossio

Versión 1.0

Cochabamba, 26 de Junio de 2017

Integrantes del equipo y Roles

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Rol | e-mail | Teléfono |
| Valeria Fuentes Surco | Product Owner | vale.fuentes.97@gmail.com | 79771716 |
| Christian Deyvid Ibañez Plata | Scrum Team | lealtad313@gmail.com | 72519568 |
| Rodrigo López Monasterios | Scrum Team | [pensu15@gmail.com](mailto:pensu15@gmail.com) | 70374503 |
| Kelly Olivera Daza | Scrum Master | [kellyolid@gmail.com](mailto:kellyolid@gmail.com) | 79725239 |
| Adriana Sahonero Cossio | Scrum Team | [sahoneroa@gmail.com](mailto:sahoneroa@gmail.com) | 67683372 |

Registro Histórico de Cambios y Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Responsable(s) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabla de Contenidos

[1. Introducción 3](#_Toc486193771)

[2. Análisis de Requerimientos 3](#_Toc486193772)

[2.1 Requerimientos de Usuario 3](#_Toc486193773)

[2.2 Requerimientos del Sistema 3](#_Toc486193774)

[2.3 Requerimientos Funcionales 3](#_Toc486193775)

[2.4 Requerimientos No Funcionales 4](#_Toc486193776)

[3. Objetivo General 4](#_Toc486193777)

[4. Objetivos Específicos 4](#_Toc486193778)

[5. Límites y Alcances 4](#_Toc486193779)

[5.1 Limites: 4](#_Toc486193780)

[**5.2 Alcance:** 4](#_Toc486193781)

[6. Diagrama de Casos de Uso 5](#_Toc486193782)

[7. Diagrama de Clases 5](#_Toc486193783)

[8. Diagrama Entidad – Relación 5](#_Toc486193784)

[9. Diagrama Burn-Up 5](#_Toc486193785)

[10. Diagrama Burn-Down 5](#_Toc486193786)

[11. Pruebas y Validación 5](#_Toc486193787)

[12. Estilo de Codificación 5](#_Toc486193788)

[12.1 Clases 5](#_Toc486193789)

[12.2 Variables 5](#_Toc486193790)

[12.3 Funciones 5](#_Toc486193791)

[12.4 Importaciones 5](#_Toc486193792)

[12.5 Estructura de control 6](#_Toc486193793)

[12.6 Comentarios 6](#_Toc486193794)

[12.7 Salto de línea 6](#_Toc486193795)

[12.8 Espacios en blanco en expresiones y declaraciones 6](#_Toc486193796)

[13. Conclusiones 7](#_Toc486193797)

[14. Recomendaciones 7](#_Toc486193798)

[15. Bibliografía 7](#_Toc486193799)

[16. Anexos 7](#_Toc486193800)

Informe Final del Proyecto

# Introducción

El nivel de desarrollo de nuestra sociedad actual ha permitido que el conocimiento a nivel mundial crezca de manera exponencial. Hoy en día las cosas se manejan de manera muy distinta a como se manejaban 50 años atrás, las necesidades humanas tienen respuestas nuevas y los procesos que se involucran en el día a día de las personas se han facilitado de manera notable.

Por este motivo se dice que vivimos en la era de la información y el conocimiento, en esta era la tecnología bien sea nueva o alta ocupa un lugar relevante.

En nuestro caso la tecnología que se pretende implementar es la creación de una aplicación de escritorio para el manejo de un supermercado. Con este sistema la compra, la venta y el pedido de productos se facilitarían.

# Análisis de Requerimientos

## Requerimientos de Usuario

Se desarrolla una aplicación de escritorio la cual constará con un sistema automático de compra venta y pedido de productos, mediante un servidor.

* El cliente podrá iniciar sesión en su usuario según su cargo.
* El sistema mostrara los productos con imagen, nombre, descripción en el menú de consultar productos.
* El administrador podrá realizar la venta de los productos, habiendo iniciado sesión con nombre y contraseña.
* El usuario puede buscar los productos según categoría.

## Requerimientos del Sistema

* El sistema requiere 4GigaBytes de RAM.
* El sistema requiere un procesador i3 o mayor, caso contrario su rendimiento disminuirá.
* El sistema requiere Compatibilidad con Windows 8.1 o versiones posteriores.
* El sistema necesita un servidor de 2GigaBytes de espacio mínimo.

## Requerimientos Funcionales

* El usuario al abrir la aplicación puede iniciar sesión si tiene una cuenta.
* El sistema debe mostrar un menú de tareas para seleccionar lo que se desea realizar.
* El usuario debe poder buscar los productos según su categoría.
* El usuario puede realizar pedidos y ventas de los diferentes productos, para esto tiene que tener la conexión con el servidor.
* El usuario debe poder ver el registro de las ventas.
* El usuario debe poder ver la factura.
* El sistema debe generar el código de control y la factura.
* El sistema debe almacenar los datos de los clientes nuevos.
* El sistema debe crear el código QR de la factura, según el reglamento del Sistema de facturación virtual de Impuestos Nacionales.
* El sistema debe despegar una alerta cuando se llegue al stock mínimo de un producto.
* Un administrador puede registrar nuevos empleados.
* El sistema debe desplegar una alerta cuando se pidan más productos de los existentes en el almacén.
* El sistema debe desplegar una alerta cuando se ingresen códigos o contraseñas erróneas.
* El Usuario debe ser capaz de acceder a través de una pantalla táctil.
* Para realizar la venta el usuario debe contar con un teclado numérico en la pantalla.

## Requerimientos No Funcionales

* El nombre de usuario no debe rebasar los 25 caracteres.
* La contraseña de usuario no debe rebasar los caracteres.
* La foto no debe sobrepasar los 2 megabytes.
* La descripción no debe pasar de los 500 caracteres.
* El nombre no debe rebasar los 100 caracteres.
* Los precios deben ser numéricos.
* El stock debe ser numérico.

# Objetivo General

Crear una página aplicación de escritorio funcional para vender productos en un supermercado, utilizando el lenguaje C#.

# Objetivos Específicos

Los objetivos específicos son:

* Haciendo uso de las herramientas que brinda Visual Studio y las metodologías aprendidas, diseñar una aplicación que facilite la compra y venta de productos en un supermercado.
* Automatizar ciertos procesos o procedimientos para reducir los tiempos y la dificultad de determinadas tareas.
* Implementar una aplicación de uso sencillo, con un buen diseño tomando en cuenta las buenas prácticas del diseño de interfaces.
* Facilitar el manejo de inventario del supermercado.

# Límites y Alcances

## Limites:

* Tiempo de la ejecución de cada Sprint.
* Tiempo para la elaboración completa del Proyecto.
* La aplicación solo funcionará previa instalación de los programas correspondientes.
* Requerimiento de Visual Studio 2017.

**5.2 Alcance:**

* Realizar la venta y pedido de productos, según los requerimientos.
* Registrar nuevos usuarios.

# Diagrama de Casos de Uso

# Diagrama de Clases

# Diagrama Entidad – Relación

# Diagrama Burn-Up

# Diagrama Burn-Down

# Pruebas y Validación

A medida que se desarrolló el proyecto se realizaron las siguientes pruebas:

* Pruebas con la conexión de la base de datos
* Pruebas de enviar datos y registro correcto.
* Pruebas de funcionamiento del código QR según las normas correspondientes.
* Pruebas de la impresión de la factura.
* Pruebas del correcto funcionamiento de todos los requerimientos.
* Pruebas con la conexión Web Service.
* Pruebas de funcionamiento del código de control según las normas correspondientes.

# Estilo de Codificación

## Clases

La primera letra del nombre de todas las clases debe estar escrita con mayúscula. Si existiese la necesidad de utilizar más de una palabra para nombrar la clase, las palabras serán unidas y diferenciadas por mayúsculas. (Ej.: NombreClase)

## Variables

Las variables deben ser en minúscula y si se necesita el uso de más de una palabra, estas serán unidas y diferenciadas por mayúsculas. (Ej.: nombreFrutas)

## Funciones

Las funciones al igual que las variables deben ir en minúsculas con las palabras unidas y diferenciadas por mayúsculas. (Ej.: calcularArea)

## Importaciones

Las importaciones de librerías deben ir en líneas separadas.

Ej.:

Sí: using os

using sys

No: using sys, os

## Estructura de control

Deben presentar una identacion de 4 espacios, indicara que las instrucciones identadas, forman parte de una misma estructura de control.

Ej.:

If condición:

\_ \_ \_ Instrucciones

## Comentarios

Los comentarios deben comenzar con el doble símbolo de slash (//). El comentario debe ser realizado en una misma línea. También puede empezar con el símbolo de slash seguido de un asterisco (/\*) y terminar con el símbolo de asterisco seguido por el slash (\*/). Debe empezar con mayúsculas.

## Salto de línea

Antes de un operador binario y no después para que los operadores coincidan con los operandos.

Ej.:

ingresos = (gross\_wages

+ taxable\_interest

+ (dividendos - qualified\_dividends)

- ira\_deduction

- student\_loan\_interest)

## Espacios en blanco en expresiones y declaraciones

**Evitar** los espacios en blanco extraños en las siguientes situaciones:

* Inmediatamente dentro paréntesis, corchetes o llaves.

Yes: spam(ham[1], {eggs: 2})

No: spam( ham[ 1 ], { eggs: 2 } )

* Entre una coma final y un siguiente paréntesis cerca.

Yes: foo = (0,)

No: bar = (0, )

* Inmediatamente antes de una coma o punto y coma,:

Yes: if x == 4: print x, y; x, y = y, x

No: if x == 4 : print x , y ; x , y = y , x

* Inmediatamente antes del paréntesis de apertura que se inicia la lista de argumentos de una llamada de función:

Yes: spam(1)

No: spam (1)

* Inmediatamente antes del paréntesis de apertura que se inicia una indexación o el corte:

Yes: dct['key'] = lst[index]

No: dct ['key'] = lst [index]

* Más de un espacio alrededor de una asignación (u otro) del operador para alinear con otro.

Sí:

x = 1

y = 2

long\_variable = 3

No:

x = 1

y = 2

long\_variable = 3

* Si se utilizan los operadores con diferentes prioridades, considerar la adición de un espacio en blanco alrededor de los operadores con la prioridad más baja (es). Use su propio juicio; Sin embargo, nunca utilice más de un espacio, y siempre tienen la misma cantidad de espacio en blanco en ambos lados de un operador binario.

Sí:

i = i + 1

submitted += 1

x = x\*2 - 1

hypot2 = x\*x + y\*y

c = (a+b) \* (a-b)

No:

i=i+1

submitted +=1

x = x \* 2 - 1

hypot2 = x \* x + y \* y

c = (a + b) \* (a - b)

# Conclusiones

Las conclusiones a las que se llegaron son:

* Se logró desarrollar la aplicación en el lenguaje predeterminado, haciendo uso de las metodologías, conceptos y otros aprendidos en clase.
* Se lograron satisfacer los requerimientos previamente establecidos.
* El proyecto funciona satisfactoriamente, a pesar de que hay cosas que se pueden mejorar, y funcionalidades que se pueden optimizar con algunos cambios.

# Recomendaciones

Las recomendaciones son las siguientes:

* Eliminar posibles errores existentes en el desarrollo de la aplicación.
* Implementar más funcionalidades al proyecto.
* Se puede mejorar la función de facturar, para que no existan problemas en ningún caso.
* Implementar en más casos el uso de excepciones, para que evitar posibles errores.

# Bibliografía

* Diapositivas del curso

# Anexos

* Manual de usuario (documento adjunto)
* Youtube: Tutorial para conexión con base de datos.

<https://www.youtube.com/watch?v=_4D6_j_JGes>

* Tutoriales para el manejo de distintas herramientas (listbox, picturebox, etc).

<https://msdn.microsoft.com/es-es/library/ms172590(v=vs.90).aspx>

<http://www.forosdelweb.com/f69/ver-varias-imagenes-picturebox-621226/>

<https://www.youtube.com/watch?v=HJOKKR4qE0U>

<https://msdn.microsoft.com/es-es/library/system.drawing.bitmap(v=vs.110).aspx>

* Tutoriales para implementar código QR

<http://blog.koalite.com/2013/07/como-generar-codigos-qr-en-c/>

* Reglamento para implementación de los 3 algoritmos para el código de control.

<http://www.impuestos.gob.bo/index.php?option=com_content&view=article&id=1564&Itemid=584>

* <https://www.youtube.com/watch?v=biDHnQ7e8cA>