Andrómeda Desktop

Valeria Fuentes Surco

Christian Deyvid Ibañez Plata

Rodrigo López Monasterios

Kelly Olivera Daza

Adriana Sahonero Cossio

Versión 1.0

Cochabamba, 30 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 – Diseñador | [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 – DBA | [sahoneroa@gmail.com](mailto:sahoneroa@gmail.com) | 67683372 |

Registro Histórico de Cambios y Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Responsable(s)** |
| 17/05/2017 | 0.1 | Requerimientos de usuario y sistema | Dayvid |
| 18/05/2017 | 0.1.1 | Diseño de los formularios | Rodrigo |
| 23/05/2017 | 0.2 | Diseño de la base de datos | Adriana |
| 25/05/2017 | 0.2.1 | Integración de la base de datos | Adriana |
| 27/05/2017 | 0.2.2 | Menú vendedor | Kelly |
| 29/05/2017 | 0.3 | Menú administrador | Valeria |
| 04/06/2017 | 0.3.1 | Productos – Menú vendedor | Kelly |
| 08/06/2017 | 0.3.2 | Agregar empleados, consultar empleados, consultar productos | Valeria |
| 15/06/2017 | 0.3.3 | Venta – Menú vendedor | Kelly |
| 16/06/2017 | 0.4 | Código de control | Adriana |
| 19/06/2017 | 0.4.1 | Integración código QR | Rodrigo |
| 20/06/2017 | 0.4.2 | Diagrama entidad – relación | Adriana |
| 21/06/2017 | 0.5 | Mostrar productos por categoría | Kelly |
| 22/06/2017 | 0.5.1 | Factura con todos los campos necesarios | Kelly |
| 23/06/2017 | 0.5.2 | Nuevo pedido, Sincronizar usuarios, Reponer fallados, Devolver productos, Recibir pedido | Kelly y Valeria |
| 24/06/2017 | 0.6 | Imprimir factura | Adriana |
| 25/06/2017 | 0.7 | Conexión con Web Service | Deyvid |
| 26/06/2017 | 0.7.1 | Control de calidad | Grupo Zeus |
| 27/06/2017 | 0.8 | Corrección de errores | Kelly y Valeria |
| 28/06/2017 | 0.9 | Pruebas | Todo el equipo |
| 29/06/2017 | 0.1 | Integración | Todo el equipo |

Tabla de Contenidos

[1. Introducción 4](#_Toc486531067)

[2. Análisis de Requerimientos 4](#_Toc486531068)

[2.1 Requerimientos de Usuario 4](#_Toc486531069)

[2.2 Requerimientos del Sistema 4](#_Toc486531070)

[2.3 Requerimientos Funcionales 4](#_Toc486531071)

[2.4 Requerimientos No Funcionales 5](#_Toc486531072)

[3. Objetivo General 5](#_Toc486531073)

[4. Objetivos Específicos 5](#_Toc486531074)

[5. Límites y Alcances 5](#_Toc486531075)

[5.1 Limites: 5](#_Toc486531076)

[**5.2 Alcance:** 5](#_Toc486531077)

[6. Diagrama de Casos de Uso 6](#_Toc486531078)

[7. Diagrama de Clases 7](#_Toc486531079)

[8. Diagrama Entidad – Relación 8](#_Toc486531080)

[9. Diagrama Burn-Up 8](#_Toc486531081)

[9.1 Sprint 1 8](#_Toc486531082)

[9.2 Sprint 2 9](#_Toc486531083)

[9.3 Sprint 3 9](#_Toc486531084)

[9.4 Sprint total 10](#_Toc486531085)

[10. Diagrama Burn-Down 10](#_Toc486531086)

[10.1 Sprint 1 10](#_Toc486531087)

[10.2 Sprint 2 11](#_Toc486531088)

[10.3 Sprint 3 11](#_Toc486531089)

[11. Pruebas y Validación 11](#_Toc486531090)

[12. Creación de Interfaz 12](#_Toc486531091)

[13. Estilo de Codificación 12](#_Toc486531092)

[13.1 Clases 12](#_Toc486531093)

[13.2 Variables 12](#_Toc486531094)

[13.3 Funciones 12](#_Toc486531095)

[13.4 Importaciones 13](#_Toc486531096)

[13.5 Estructura de control 13](#_Toc486531097)

[13.6 Comentarios 13](#_Toc486531098)

[13.7 Salto de línea 13](#_Toc486531099)

[13.8 Espacios en blanco en expresiones y declaraciones 13](#_Toc486531100)

[14. Conclusiones 14](#_Toc486531101)

[15. Recomendaciones 14](#_Toc486531102)

[16. Bibliografía 14](#_Toc486531103)

[17. Anexos 14](#_Toc486531104)

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 usuario podrá iniciar sesión en su usuario según su cargo, ingresando su nombre y contraseña.
* El vendedor responsable podrá realizar ventas de distintos productos.
* Listar los productos que se encuentran a disposición con sus respectivos datos.
* Los productos deben contar con su imagen respectiva.
* Almacenar los datos del cliente.
* Generar una factura de la venta que cuente con un código QR en base a los reglamentos del Sistema de Facturación Virtual de Impuestos Nacionales.
* El administrador podrá realizar pedidos de productos faltantes al servidor.
* El sistema despliegue una alerta cuando uno de los productos llegue al stock mínimo.
* La base de datos se actualiza al final del día con el servidor y cada que sea necesario.
* El sistema mostrara los productos con imagen, nombre, descripción en el menú de consultar productos.
* 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 7 o versiones posteriores, caso contrario su rendimiento disminuirá.
* El sistema necesita un servidor de 2GigaBytes de espacio mínimo.
* El sistema necesita el programa “Sql Server”.

## 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.
* Al finalizar el día la sucursal debe devolver todos los productos no vendidos a la central.
* El administrador debe ser capaz de devolver los productos fallados.

## Requerimientos No Funcionales

* El código de usuario no debe rebasar los 15 caracteres.
* El nombre de usuario no debe rebasar los 30 caracteres.
* La contraseña de usuario no debe rebasar los 15 caracteres.
* El nombre del cliente no debe rebasar los 50 caracteres.
* Las cantidades de venta deben ser numéricas.

# 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 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

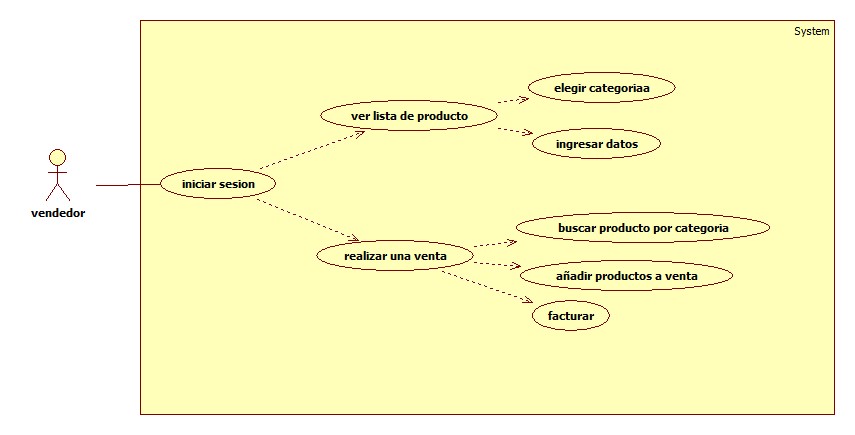
## Limites:

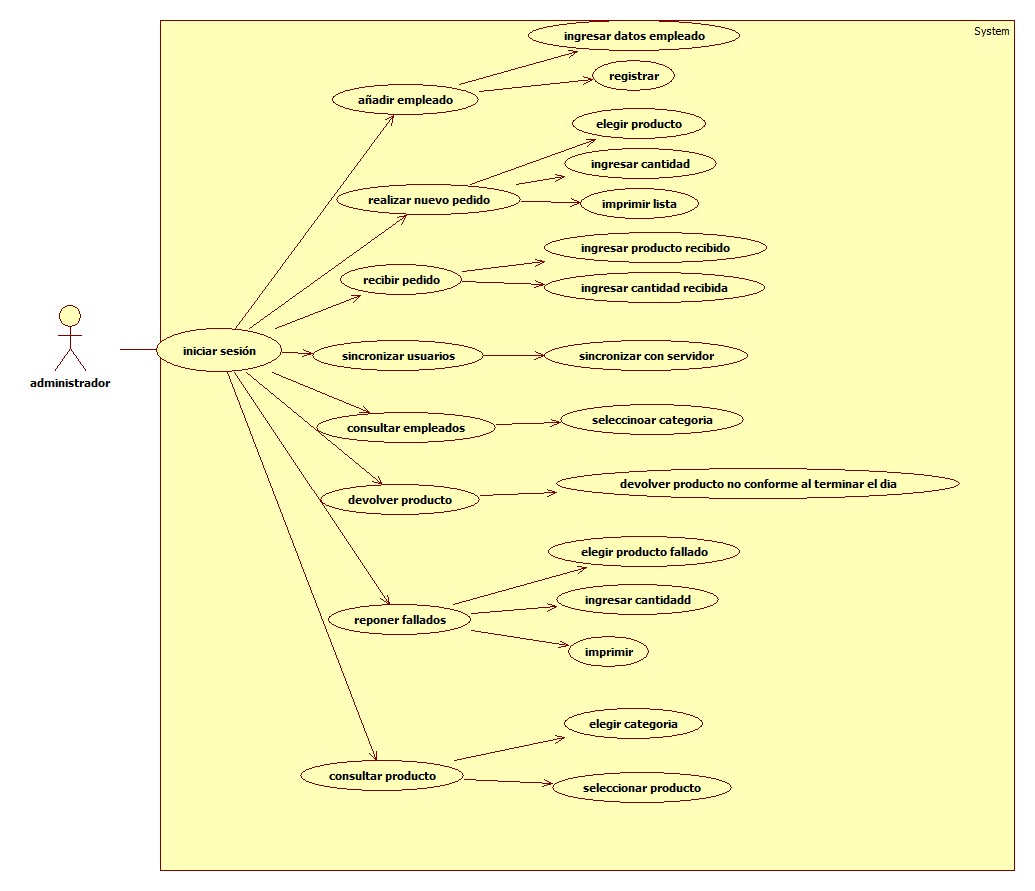
* Tiempo de la ejecución de cada Sprint.
* Tiempo para la elaboración completa del Proyecto.

## Alcance:

* Iniciar sesión según el tipo de usuario.
* Realizar la venta y pedido de productos, según los requerimientos.
* Devolver productos a la central.
* Registrar nuevos usuarios.
* Registrar nuevos clientes.
* Imprimir factura de venta, detalle de pedido y detalle de productos fallados.
* Poder acceder a todos los controles con el puntero o de manera táctil si el entorno lo permite.

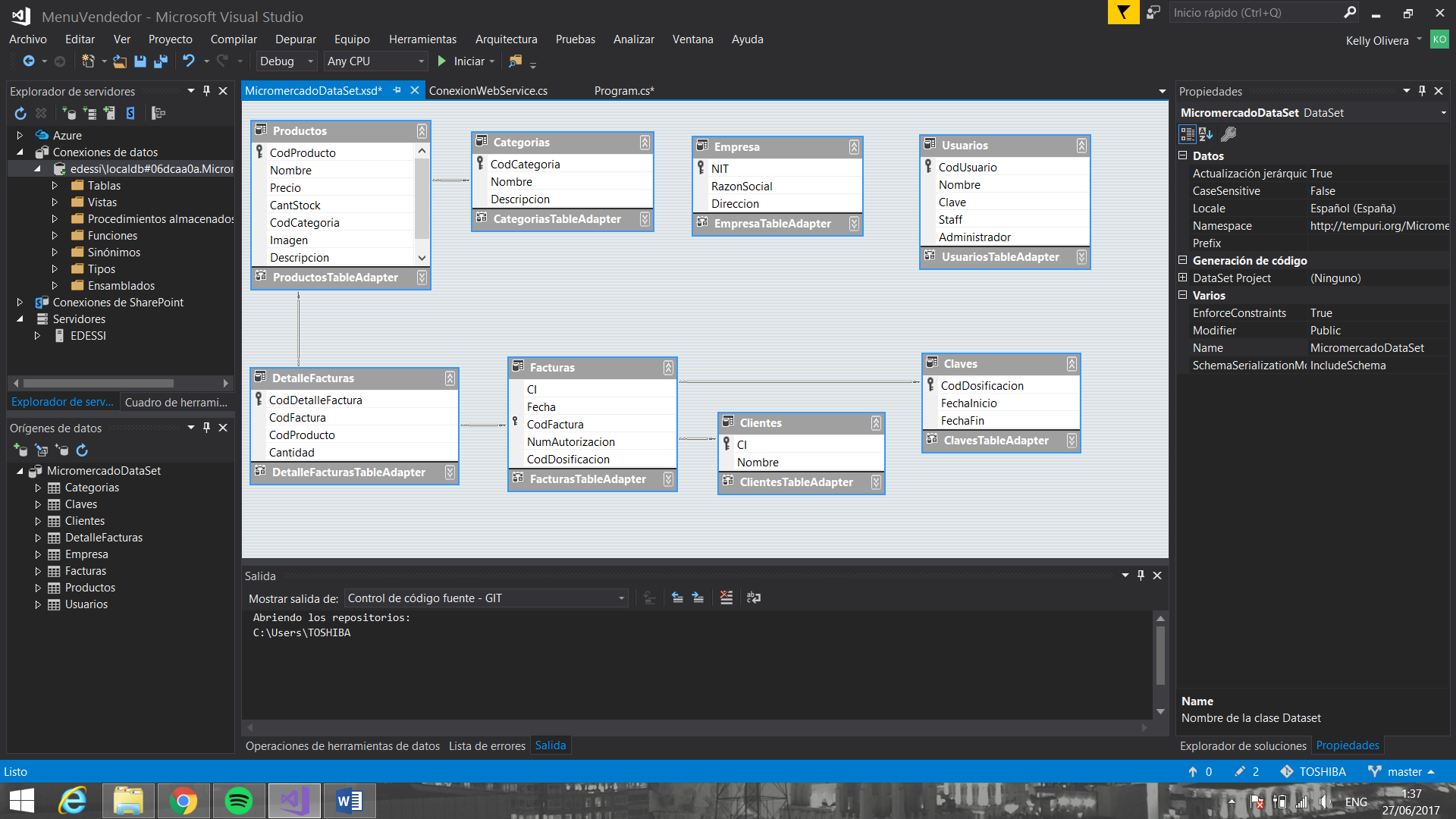
# Diagrama de Casos de Uso





# Diagrama de Clases

# Diagrama Entidad – Relación



# Diagrama Burn-Up

## Sprint 1

## Sprint 2

## Sprint 3

## Sprint total

# Diagrama Burn-Down

## Sprint 1

## Sprint 2

## Sprint 3

# Product Backlog

* El cliente podrá iniciar sesión en su usuario según su cargo, ingresando su nombre y contraseña.
* El vendedor responsable podrá realizar ventas de distintos productos.
* Listar los productos que se encuentran a disposición con sus respectivos datos.
* Los productos deben contar con su imagen respectiva.
* Almacenar los datos del cliente.
* Generar una factura de la venta que cuente con un código QR en base a los reglamentos del Sistema de Facturación Virtual de Impuestos Nacionales.
* El administrador podrá realizar pedidos de productos faltantes al servidor.
* Que el sistema despliegue una alerta cuando uno de los productos llegue al stock mínimo.
* La base de datos se actualiza al final del día con el servidor y cada que sea necesario.
* El sistema mostrara los productos con imagen, nombre, descripción en el menú de consultar productos.
* El usuario puede buscar los productos según categoría.

# Sprint Backlog

## Sprint 1

* Requerimientos de Usuario
* Requerimientos de Sistema
* Investigar y hacer pruebas con la base de datos (Visual Studio)
* Diseño de la base de datos en Visual Studio
* Inicio de sesión funcional con la base de datos
* Conexión entre inicio de sesión y menús
* Investigar los reglamentos del SVF - Impuestos Nacionales
* Interfaz del inicio de sesión
* Interfaz del menú administrador
* Ventana Productos
* Managment Studio for sqlserver
* Diagrama E-R
* Diseño de las interfaces
* Menú vendedor
* Venta
* Investigar y descargar la biblioteca de Código QR
* Pruebas funcionales de código QR
* Añadir generador de QR a la interfaz de la venta
* Investigar la conexión entre la base de datos de visual y web service
* Integración

## Sprint 2

* Añadir nuevo cliente cuando no existe
* Actualizar la base de datos
* Mostrar productos por categoria
* Estandarizar colores
* Teclado numerico
* Estandarizar colores
* Teclado numerico
* Generar los 3 algortimos de codigo de control SFV
* Agregar los 3 algoritmos a la pagina principal
* Consultar pedidos
* Consultar empleados
* Generar la factura con el codigo QR
* Generar lista de productos vendidos
* Generar factura con todos los datos de la empresa y cliente
* Conexión con servidor

## Sprint 3

* Agregar icono
* Imprimir la factura directamente
* Arreglar consultar empleados
* Agregar imagen cuando seleccionar fila en consultar productos
* Stock minimo message box de alerta
* Restar productos cuando se vende
* nuevo pedido
* Conexión de web service

# Story Board

## Sprint 1

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Not Checked Out** | **Checked Out** | **Done!** |
| Adriana  Sahonero |  |  | Investigar y hacer pruebas con la Base de Datos – Visual Studio |
|  |  | Diseño de la base de datos en Visual Studio |
|  |  | Inicio de sesión funcional con la base de datos (Vale) |
|  |  | Conexion entre Inicio de sesion y menus |
| Valeria Fuentes |  |  | Investigar los reglamentos del SFV – Impuestos Nacionales |
|  |  | Interfaz del inicio de sesión |
|  |  | Interfaz del menú administrador |
|  |  | Ventana Productos |
| Kelly Olivera |  |  | Managment Studio for sqlserver |
|  |  | Diagrama E-R |
|  |  | Diseño de las interfaces |
|  |  | Menú vendedor |
|  |  | Venta |
| Rodrigo Lopez |  |  | Investigar y descargar la biblioteca de Código QR – Visual Studio |
|  |  | Pruebas funcionales de código QR |
|  |  | Añadir generador de QR a la interfaz de venta |
| Deyvid Ibañez |  |  | Investigar la conexión entre la base de datos de Visual Studio y la base de datos del servidor (postgreSQL) |
|  |  | Requerimientos de Usuario |
|  |  | Requerimientos de Sistema |
|  | Conexión Visual Studio – PostgreSql Servidor |  |
| Todos |  |  | Integración |

## Sprint 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Not Checked Out | Checked Out | Done! |
| Rodrigo Lopez |  |  | Reporte de Factura |
|  |  | Agregar reporte a la app principal |
|  |  | Añadir nuevo cliente cuando no existe |
| Adriana Sahonero |  |  | Generar los 3 algoritmos de código de control SFV |
|  |  | Agregar los 3 algoritmos a la app principal |
|  |  | Actualizar la base de datos |
| Kelly Olivera |  |  | Mostrar productos por categoría |
|  |  | Estandarizar colores |
|  |  | Teclado Numérico |
| Valeria Fuentes |  |  | Interfaz menú administrador |
|  |  | Consultar productos, pedidos, empleados |
|  |  | Nuevo Pedido, empleado |
| Deyvid Ibañez |  |  | Restar cantidad |
|  | Conexión Visual Studio – PostgreSql Servidor |  |
| Todos |  |  | Integración |

## Sprint 3

|  |  |  |  |
| --- | --- | --- | --- |
|  | Not Checked Out | Checked Out | Done! |
| Rodrigo Lopez |  |  | Agregar icono |
| Adriana Sahonero |  |  | Imprimir la factura directamente |
| Kelly Olivera |  |  | Agregar imagen cuando seleccionar fila en consultar productos |
|  |  | Stock mínimo message box de alerta |
|  |  | Restar productos cuando se vende |
| Valeria Fuentes |  |  | Arreglar consultar empleados |
|  |  | nuevo pedido |
| Deyvid Ibañez |  |  | Conexión Visual Studio – PostgreSql Servidor |
| Todos |  |  | Integración |

# Pruebas y Validación

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

* Pruebas con la conexión de la base de datos
* Comprobar que se abra el menú correspondiente según el tipo de usuario que ingresa.
* Pruebas de envió de datos y registro correcto.
* Comprobar el funcionamiento correcto del teclado numérico.
* Pruebas de funcionamiento del código QR según las normas correspondientes.
* Pruebas de la impresión de la factura.
* Comprobar si se resta la cantidad vendida en el Stock del producto.
* Pruebas del correcto funcionamiento de todos los requerimientos.
* Pruebas con la conexión Web Service.
* Pruebas en el manejo de excepciones.
* Pruebas de funcionamiento del código de control según las normas correspondientes.

# Creación de Interfaz

Los formularios fueron creados con el objetivo de facilitar a los usuarios el uso del sistema,   
permitiendo el acceso rápido a todas sus funcionalidades. Todas fueron creadas teniendo en cuenta los principios de diseño de la interfaz de usuario:

* Familiaridad del usuario: Se aplicó mediante el uso de palabras y comandos sencillos que se aprenden intuitivamente.
* Uniformidad: Se implementó en todos los formularios según las bases de diseño que fueron acordadas entre todos los diseñadores del proyecto, utilizando el mismo fondo, logo, color de letra, estilo, etc.
* Mínima sorpresa: Los comandos fueron establecidos de manera específica y gráfica de modo que el usuario puede predecir la operación de un comando parecido.
* Recuperabilidad: El diseño de los formularios fue creado haciendo uso de advertencias cuando se quiere realizar alguna función “irreversible”, con el objetivo de alertar al usuario de todas las acciones que quiere ejecutar. Como al momento de sincronizar todos los usuarios, devolver los productos al final del día, antes de imprimir cualquier tipo de documento.
* Guía de usuario: Cada formulario tiene su respectivo punto en la guía de usuario, permitiendo al mismo utilizar de manera correcta cada comando y proporcionando retroalimentación significativa y características de ayuda sensible al contexto.
* Diversidad de usuario: En este caso se tienen dos usuarios, vendedor y administrador los cuales tienen idea de cómo funcionan las actividades que necesitan realizar, de igual manera el diseño facilita que entendimiento para ambos usuarios.

# 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 System

using System.Data

No: using System, System.Data

## 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 (exceso de productos en el detalle).
* Implementar en más casos el uso de excepciones, para que evitar posibles errores.

# Bibliografía

* Diapositivas del curso
* 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>

# Anexos

Manual de usuario (documento adjunto)