Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Manejadores de entrada/salida para los periféricos de un Sistema POS de hardware y software abierto

Autor: Oscar Luis Núñez Mansanet

Tutor: Ing. Jordanis Viltres Chávez

Cotutor: Ing. Julio Alberto Leyva Durán

La Habana, Cuba 2017

“Año 59 de la Revolución”

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma del autor Firma del tutor

_____

Firma Cotutor

Agradecimientos

Agradezco de primero y especialmente a mi madre que siempre ha estado ahí para mí en todo momento y en todo lo que necesité, para poder lograr este sueño, que se convirtió en el suyo. A mi adorable hijo, quien inconscientemente fue mi más grande fuerza de empuje, y mi deseo de ser un gran ejemplo y orgullo para él. A mi familia, que constituye una parte importante de mi vida. A mi compañera, que vivió todo lo que viví, bueno y malo, y me levantó cuando me caí, me impulsó cuando perdí la fuerza y me ayudó a tomar mi rumbo, como quien a la deriva encuentra a lo lejos un faro. A mi tutor que a pesar de los altibajos nunca perdió la convicción de que yo pudiera lograr mis metas, y siempre tuvo la mejor disposición de transmitir sus conocimientos y contribuir en mi preparación como profesional. A mis amigos, los mejores que he tenido y que estoy seguro siempre tendré, por pasar a ser parte de mi familia y como tal acogerme en sus vidas, y batallar contra todo junto a mí, incondicionalmente. A todos los que de alguna manera contribuyeron a materializar lo que un día pudo sonar a utopía: Gracias.

Oscar Luis

Dedicatoria

A mi hijo, a mis padres, a mi familia, a mi compañera y pareja, a mis amigos, a todos los que confiaron en mí hasta el último momento.

Resumen

Los sistemas de punto de venta (POS), constituyen un gran avance en el proceso de las ventas a nivel mundial. En el transcurso de los años la informática ha sido una potente herramienta para automatizar este tipo de procedimiento, eliminando cada vez más la complejidad de los trámites, la pérdida y duplicación de la información.

Con el objetivo de lograr la independencia tecnológica el Centro de Informática Industrial (CEDIN), desarrolla un sistema POS que utilice las tecnologías actuales como dispositivos móviles, *hardware* y *software* libre, para satisfacer necesidades de este tipo de gestión, para el cual se necesitan los manejadores para los dispositivos vinculados al proceso de venta y un servidor que permita el acceso a ellos de forma remota a través de un protocolo para el intercambio de la información.

La presente investigación se basa en el estudio del estado del arte de los Sistemas POS a nivel mundial, las tecnologías y herramientas utilizadas. Se describe la solución propuesta para los manejadores de un sistema POS y una interfaz de acceso, apoyándose en los artefactos que propone la metodología de desarrollo seleccionada, se realiza un análisis de los diseños y modelos obtenidos y se describe el proceso real. Por último, se termina el proceso de implementación del sistema a partir del diseño resultante y la realización de las pruebas correspondientes a las funcionalidades requeridas.

Palabras claves: Punto de Venta (POS), manejadores, dispositivos móviles, servidor, interfaz.

Índice

Resumen IV

Índice de figuras VIII

Índice de tablas VIII

Introducción 1

Capítulo 1: Fundamentación teórica 5

Introducción 5

1.1 Punto de Venta (POS) 5

1.1.2 *Software* POS 6

1.1.3 Dispositivos HID 7

1.2 Manejadores de dispositivos 7

1.3 Bus Universal en Serie (USB) 8

1.3.1 Versiones del estándar USB 8

1.4 Estado del arte de los sistemas POS 9

1.5 Herramientas y tecnologías para el Desarrollo 12

1.5.1 Lenguajes de programación 12

1.5.2 Lenguaje de Modelado 12

1.5.3 Herramientas de Modelado 12

1.5.4 Bibliotecas 12

1.5.5 Protocolos y Tecnologías 13

1.5.5.1 Ventajas de WebSocket 13

1.5.6 *Hardware* 13

1.5.7 Entorno Integrado de Desarrollo (IDE) 14

1.5.8 Metodología de desarrollo de *software* 15

1.5.8.1 Metodología AUP-UCI 15
1.5.8.2 Fases de AUP-UCI 15
1.5.8.3 Escenarios de AUP-UCI 16
1.5.8.4 Selección de la metodología 16
Conclusiones parciales: 16
Capítulo 2: Propuesta de solución 18
Introducción 18
2.1 Búsqueda de información útil para desarrollar el sistema 18
2.2 Especificación de requisitos 18
2.2.1 Descripción de las Historias de Usuario 19
2.2.2 Estimación de tiempo de desarrollo por HU 22
2.2.3 Plan de iteraciones 23
2.2.4 Plan de entregas 24
2.3 Arquitectura del Sistema 25
2.4 Intercambio de Mensajes 25
2.5 Diagrama de Clases 27
2.6 Patrones de diseño 28
2.6.1 Patrones GRASP 28
2.6.2 Patrones GoF 29
Conclusiones parciales 30
Capítulo 3: Implementación y Pruebas 31
Introducción 31
3.1 Tareas de ingeniería 31
3.2 Descripción del Sistema 35

3.2.1 Descripción de WebServer 35
3.2.2 Descripción de POSDriverInterface 35
3.2.3 Biblioteca de manejadores: 35
3.3 Manejo de los dispositivos con la biblioteca libusb-1.0 36
3.4 Estandarización de la implementación 37
3.5 Diagrama de despliegue 37
3.6 Pruebas de Aceptación 38
3.6.1 Resultados de las pruebas de aceptación 46
Conclusiones parciales 46
Conclusiones Generales 47
Recomendaciones 48
Referencias Bibliográficas 49

Índice de figuras

Figura 1. Terminal de punto de venta 6
Figura 2. Conector USB 8
Figura 3. Quorion QMP 9
Figura 4. Interfaz de POS Tasven 10
Figura 5. Interfaz de POS Vend 10
Figura 6. Interfaz de POS InfoTouch 11
Figura 7. Arquitectura Cliente-Servidor de un sistema POS aplicado a la solución propuesta 25
Figura 8. Relación entre las clases de la solución propuesta 27
Figura 9. Clase WebServer 29
Figura 10. Clase POSDriverInterface 30
Figura 11. Diagrama de Clases donde se evidencia el patron Factoría 30
Figura 12. Diagrama de despliegue 37

Índice de tablas

Tabla 1. Requisitos funcionales 18
Tabla 2. Historia de Usuario 1 19
Tabla 3. Historia de Usuario 2 20
Tabla 4. Historia de Usuario 3 20
Tabla 5. Historia de Usuario 4 21
Tabla 6. Historia de Usuario 5 21
Tabla 7. Historia de Usuario 6 22
Tabla 8. Historia de Usuario 7 22
Tabla 9. Estimación por esfuerzo de las HU 23
Tabla 10. Plan de iteraciones 24
Tabla 11. Plan de entregas 24
Tabla 12. Tareas de Ingeniería 32
Tabla 13. Tarea de Ingeniería 1 32

TABLA 14. TAREA DE INGENIERÍA 2 33
TABLA 15. TAREA DE INGENIERÍA 3 33
TABLA 16. TAREA DE INGENIERÍA 4 33
TABLA 17. TAREA DE INGENIERÍA 5 33
TABLA 18. TAREA DE INGENIERÍA 6 34
TABLA 19. TAREA DE INGENIERÍA 7 34
TABLA 20.TAREA DE INGENIERÍA 8 34
TABLA 21. TAREA DE INGENIERÍA 9 35
TABLA 22. PRUEBA DE ACEPTACIÓN 2 39
TABLA 23. PRUEBA DE ACEPTACIÓN 4 40
TABLA 24. PRUEBA DE ACEPTACIÓN 3 41
TABLA 25. PRUEBA DE ACEPTACIÓN 8 41
TABLA 26 PRUEBA DE ACEPTACIÓN 9 42
TABLA 27. PRUEBA DE ACEPTACIÓN 10 42
TABLA 28. PRUEBA DE ACEPTACIÓN 11 43
TABLA 29 PRUEBA DE ACEPTACIÓN 1 44
TABLA 30 PRUEBA DE ACEPTACIÓN 3 44
TABLA 31. PRUEBA DE ACEPTACIÓN 5 45
TABLA 32. PRUEBA DE ACEPTACIÓN 7 46

Introducción

La evolución del hombre, producto a su propio desarrollo científico y tecnológico, se ha visto marcada en las últimas décadas por el rápido avance de la ciencia, siendo la informática, a pesar de ser pionera en cuanto a su existencia, una de las más reconocidas a nivel mundial. Esto se pone de manifiesto en la automatización de procesos, la digitalización de información, la realización de cálculos a gran velocidad, por encima de la capacidad humana, y también al acelerado proceso evolutivo de esta rama.

Cuba, un país sub-desarrollado y bloqueado, se ha lanzado a promover el desarrollo de la informática llegando a alcanzar buenos resultados dentro y fuera del ámbito nacional. Como muestra de ello es necesario mencionar la Universidad de las Ciencias Informáticas (UCI), escuela de nivel superior que ha graduado a miles de ingenieros, preparados para llevar la informática de manera profesional a la sociedad cubana.

La UCI además de su proceso docente cuenta con centros de desarrollo en los que se llevan a cabo proyectos nacionales e internacionales, en los que trabajan profesionales y estudiantes, contribuyendo así a la economía del país y a su vez al desarrollo tecnológico nacional. Dentro de estas infraestructuras se encuentra el Centro de Informática Industrial (CEDIN). Este centro como su nombre lo indica se encarga de informatizar procesos industriales a menor o mayor escala, dándole solución a diversas necesidades de la sociedad, dentro o fuera del ámbito nacional.

En Cuba muchos son los puntos de venta que cuentan con terminales de modelos comerciales para leer códigos de barras, leer tarjetas magnéticas, abrir o cerrar cajas de dinero, imprimir vales en una impresora, etc. Otros muchos puntos de venta estatal o particular no disponen de dichos sistemas, teniendo que llevar constancia de los artículos vendidos de forma escrita. A dichas terminales se les conoce como sistemas POS, los cuales en el mercado internacional tienen un valor considerable y no existen modelos cubanos. Estos sistemas cuentan con *hardware* y *software* privativo, de difícil integración con soluciones informáticas realizadas por terceros, como por ejemplo para el manejo de los periféricos de un sistema POS sobre tecnologías actuales de intercambio de información como redes *Ethernet*, *Wi-fi* o *Bluetooth*. No se pueden implementar aplicaciones que realicen las funciones mencionadas de un sistema POS si no se cuentan con los *drivers* o manejadores de los periféricos que más comúnmente se utilizan en los mismos. El uso potencial que pudieran tener los *drivers* proporcionados por los fabricantes de estos

periféricos, tendrían sus limitaciones. Por otro lado, el auge de los dispositivos móviles con la posibilidad de implementar aplicaciones de diversa índole para los mismos, la diversidad de herramientas y tecnologías para el desarrollo de sistemas sobre *hardware* y *software* abierto, y la diversidad de plataformas de bajo costo para el diseño de prototipos integrales de dispositivos con *software* embebido para el control de periféricos, permitiría obtener una solución cubana de mucho más bajo costo que en el mercado internacional y adaptable a diversos entornos de ventas estatales o particulares, contribuyendo a la sustitución de importaciones.

La **situación problemática** planteada con anterioridad permite formular el siguiente **problema científico**: ¿Cómo permitir el uso de periféricos de sistemas POS, desde posibles aplicaciones de usuario de un sistema de este tipo?

El problema planteado permite definir el **objeto de estudio**: Proceso de desarrollo de manejadores de código abierto para dispositivos HID, funcionamiento de sistemas POS, estándar USB 2.0 y protocolos para transferencia de datos.

Para solucionar el problema se definió como **objetivo general**: Desarrollar los manejadores de los periféricos de un sistema POS y la interfaz que permita el acceso a los mismos.

Por todo lo anterior queda definido como **campo de acción**: Mecanismo de transferencia de datos entre aplicaciones de usuario y dispositivos HID en Sistemas POS utilizando el estándar USB 2.0.

Para darle cumplimiento al objetivo definido fueron confeccionadas las siguientes **tareas de la investigación:**

- Caracterización del estándar USB 2.0.
- Caracterización de los protocolos de comunicación con dispositivos de interfaz humana (*Human Interface Device* HID) para sistemas POS, sobre el estándar USB2.0.
- Caracterización de las tecnologías o protocolos para el intercambio de información entre aplicaciones de *software*.
- Identificación de los requisitos funcionales para el desarrollo de la solución propuesta.
- Elección de los estándares, protocolos de comunicación, metodologías, herramientas y tecnologías a usar para el desarrollo de la solución propuesta.

- Generación de los artefactos relacionados con el análisis y diseño de la solución propuesta.
- Propuesta de la arquitectura de la solución propuesta cumpliendo con la política de *software* libre.
- Diseño e implementación de la solución propuesta.
- Pruebas y corrección de errores de la solución propuesta.
- Elaboración del documento del Trabajo de Diploma.

Durante el desarrollo de la investigación se utilizaron varios métodos científicos en la búsqueda y procesamiento de la información, los cuales fueron:

Métodos Empíricos

- Experimental: Para verificar el funcionamiento de los procesos que intervienen en el sistema mediante pruebas realizadas al sistema desarrollado durante la investigación.

Métodos Teóricos

- Histórico-lógico: Para estudiar aplicaciones de la misma índole, investigar el estado actual de la tecnología utilizada y las herramientas que permiten desarrollar el sistema propuesto
- Modelación: Para facilitar la comprensión de los procesos que se ejecutan en el sistema y esquematizar los procedimientos desarrollados
- Analítico-sintético: Para identificar los aspectos teóricos principales de la investigación y la implementación del proceso llevado a cabo por la aplicación.

Posibles resultados: Biblioteca de manejadores para periféricos de un sistema POS y la interfaz para el acceso a los periféricos a través de un protocolo de comunicación entre aplicaciones.

La estructura de este documento se presenta de la siguiente forma:

Capítulo 1: Fundamentación teórica: Este capítulo trata acerca de los fundamentos teóricos en los que se basa la investigación, y se describen las tecnologías y herramientas a utilizar.

Capítulo 2: Propuesta de Solución: Aquí se realiza una descripción de la solución apoyándose en los artefactos requeridos por la metodología de desarrollo seleccionada, se realiza un análisis de los diseños y modelos obtenidos y se describe el proceso real vinculado al campo de acción.

Capítulo 3: Implementación y pruebas: El tercer capítulo aborda sobre la implementación del sistema a partir del diseño resultante y la realización de las pruebas correspondientes a las funcionalidades requeridas.

Capítulo 1: Fundamentación teórica

Introducción

Actualmente los Puntos de Venta y la venta de productos han ido evolucionando aprovechando las facilidades que brindan las nuevas tecnologías. De este modo ha sido notable el cambio y los beneficios que genera la automatización de procesos en esta esfera al igual que en muchas otras. Éste capítulo trata sobre los fundamentos teóricos, definiciones y características relacionados con los Sistemas POS y sus similares en el mercado, el estándar USB, los dispositivos HID, los periféricos de un Terminal de Punto de Venta, la transferencia de datos entre manejadores de dispositivos y sistemas de este tipo. También se describirán las herramientas y técnicas utilizadas para la implementación de la aplicación.

1.1 Punto de Venta (POS)

El *Cambridge Business English Dictionary* define **punto de venta** como: 1- Un lugar como una tienda donde se vende algo. 2- Un lugar donde la gente paga en una tienda. Por otro lado define **punto de venta electrónico** como: Un sistema informático que registra las ventas de una tienda y los productos que tiene en *stock*, y proporciona a los clientes su información de pago.(1)

Punto de venta (*Point of Sale*, POS) es la frase utilizada para referirse al punto o ubicación donde tiene lugar una transacción de venta, como una línea de pago o un contador de venta al por menor. Un sistema de punto de venta es el término utilizado para la combinación de *hardware* y *software* que gestiona la transacción de ventas. Hay muchas ventajas de usar un sistema de punto de venta sobre una caja registradora tradicional, ya que una computadora es capaz de capturar, almacenar, compartir y reportar datos (como ventas, pagos o información de clientes). Un sistema POS ahorra tiempo y duplicación de trabajo, e incrementa la eficiencia y exactitud en inventario, informes, pedidos y un excelente servicio al cliente.(2)

Un terminal de punto de venta es un reemplazo computarizado para una caja registradora, aunque mucho más complejo que las cajas registradoras de años pasados. Un sistema POS puede incluir la capacidad de almacenar y realizar el seguimiento de pedidos de clientes, procesar tarjetas de crédito y débito, conectarse a otros sistemas en una red y administrar el inventario. Un terminal POS tiene como núcleo un microprocesador especializado, que está provisto de programas específicos de aplicación y dispositivos

de E/S para el entorno particular en el que servirá. Los terminales de punto de venta también están cada vez más habilitados para la *Web*, lo que hace posible la capacitación remota y la operación, así como el seguimiento de inventarios en lugares geográficamente dispersos. Terminales de punto de venta se utilizan en la mayoría de las industrias que tienen un punto de venta, como un servicio de mesa, incluyendo restaurantes, alojamiento, entretenimiento y museos.(3)

En un POS existen varios componentes principales como el *hardware* y el *software*. De la parte del *hardware* se encuentra la computadora, y el terminal de POS con sus dispositivos, que generalmente son lector de tarjeta magnética, lector de código de barras, impresora de comprobantes de venta, caja registradora entre otros dispositivos HID que pueden utilizar el estándar USB. De la parte del *software* se encuentran los manejadores de los dispositivos y el sistema que controla el POS.

Figura 1. Terminal de punto de venta

1.1.2 *Software* POS

Estos programas son los encargados de realizar todo el proceso de venta, desde dar de alta en su base de datos cada producto manejado en un comercio, la lectura del código de barras en ellos, hasta la emisión del *ticket* o comprobante de compra, así como de la emisión de reportes de inventarios, proveedores, estadísticas, etc.(4)

1.1.3 Dispositivos HID

Los dispositivos HID (*Human-Interface-Device* en inglés), o Dispositivos de Interfaz Humana, son aquellos periféricos que se conectan al ordenador de forma externa, a través de un puerto generalmente USB. En el caso del POS pueden ser lector de código de barras, lector de tarjeta magnética, impresora de *tickets*, etc., con los que el usuario interactúa físicamente.

1.2 Manejadores de dispositivos

Según la Real Academia de Ingeniería, manejador, controlador o *driver* es un componente *software* que permite que un dispositivo se entienda con el sistema operativo y pueda ser utilizado por las aplicaciones.(5)

Manejador o controlador de dispositivo (*Device Driver):* Pequeño programa que permite que una computadora y un dispositivo se comuniquen entre sí. Los sistemas operativos de la computadora usualmente vienen con controladores de dispositivos 'preinstalados' para los modelos actuales de los dispositivos populares. Los controladores de dispositivo más antiguos o más recientes (que usualmente vienen en un disco con el dispositivo o se descargan desde el sitio *Web* del fabricante) deben ser instalados por el usuario.(6) Estos programas realizan una abstracción del *hardware* y proporcionan una interfaz para utilizar el dispositivo.

Interfaz: Límite común donde se produce el contacto directo entre dos culturas diferentes, dispositivos, entidades, entornos, sistemas, etc., y donde se intercambia energía, información y / o material. Un enchufe eléctrico o una toma telefónica son ejemplos comunes de interfaz de dispositivo. En contextos técnicos, los requisitos de interfaz se rigen por convenciones estándar llamadas protocolos que podrían ser tan complejos que la interfaz en sí se considera un dispositivo o sistema independiente. En el contexto de las computadoras, tres tipos comunes de interfaz son: **interfaz de *hardware***, que consiste en cables, conectores y puertos que conectan dispositivos como teclados, ratones, impresoras, unidades de almacenamiento, etc., al ordenador; **Interfaz de *software***, que consta de comandos, códigos y mensajes (denominados interfaz de programa de aplicación) que permiten a diferentes programas comunicarse entre sí y con el sistema operativo; e **interfaz de usuario**, que consta de línea de comandos, menús, avisos, cuadros de diálogo, iconos, asistentes, etc., que permiten a un usuario y una computadora comunicarse entre sí.(7)

1.3 Bus Universal en Serie (USB)

Figura 2. Conector USB

Puerto USB o *Universal Serial Bus* es un bus universal en serie o Conductor Universal en Serie CUS, es un puerto que sirve para conectar periféricos a un ordenador. Una característica importante es que permite a los dispositivos trabajar a velocidades mayores, en promedio a unos 12 Mbps, esto es más o menos de 3 a 5 veces más rápido que un dispositivo de puerto paralelo y de 20 a 40 veces más rápido que un dispositivo de puerto serial.(8) (9)

1.3.1 Versiones del estándar USB

USB 1.0 Tasa de transferencia de hasta 1,5 Mbps (192 KB/s). Utilizado en su mayor parte por HID como los teclados, los ratones, las cámaras *Web*, etc.

USB 1.1 Tasa de transferencia de hasta 12 Mbps (1,5 MB/s), según este estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones. Ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias LIFO.

USB 2.0 Tasa de transferencia de hasta 480 Mbps (60 MB/s) pero por lo general de hasta 25Mbps (16MB/s). Está presente casi en el 99% de los ordenadores actuales. El cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y una de toma de tierra.

USB 3.0 Tiene una tasa de transferencia de hasta 4.8 Gbps (600 MB/s). Esta especificación es diez veces más veloz que la anterior 2.0 y se lanzó a mediados de 2009 por Intel, según se estima, o por otra empresa de *hardware*. La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 conectores extra, desechando el conector de fibra óptica propuesto inicialmente, y será

compatible con los estándares anteriores. La principal característica es la multiplicación por 10 de la velocidad de transferencia, que pasa de los 480 Mbps a los 4,8 Gbps (600 MB/s).(9)

Se selecciona el estándar USB 2.0 porque es el más común para los dispositivos HID actuales, la tasa de transferencia es eficiente, estable y el sistema no maneja grandes cantidades de datos por lo que no es necesario la versión 3.0 del estándar.

1.4 Estado del arte de los sistemas POS

Durante la investigación se produjeron varios resultados sobre algunos de los *software*s para POS a nivel nacional e internacional. A continuación, se analizan algunas de sus características e interfaces.

QMP POS: El proceso de las ventas en las tiendas recaudadoras de divisa (TRD) en Cuba es llevado a cabo mediante la Caja Registradora Quorion QMP y utiliza el *software* QMP POS. Es fácil de manejar e intuitivo y brinda funcionalidades y operaciones programables. Utiliza un software basado en una arquitectura de 32 bits. Presenta una impresora integrada y puertos USB para otros dispositivos.

Figura 3. Quorion QMP

Tasven: Controla y administra el negocio de una manera rápida y amigable. No requiere conocimientos en computación. Reduce la fuga de mercancía y mermas. Altas, bajas, modificaciones, ajustes, ingreso rápido, ofertas, imagen en los productos. Exporta a Excel todos los reportes. Entrega comprobantes de pago. Utiliza Windows. Los periféricos se encuentran en una caja registradora con dispositivos integrados (10)

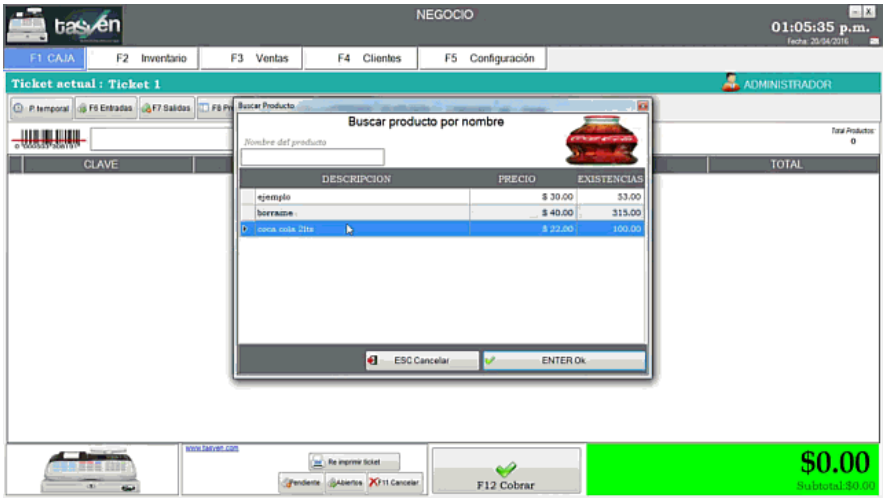


Figura 4. Interfaz de POS Tasven

Vend: Ofrece un fácil manejo, basado en *Web* sensible a Mac o PC, o la aplicación de registro de Vend Register. Posibilita crear una tienda en línea y administrar todo de forma centralizada. Se puede añadir rápidamente productos, realizar el seguimiento de las ventas desde una sola cuenta. Soportado en iOS, Mac y Windows. Los periféricos se conectan por *Bluetooth* y por puerto RJ12 (11)

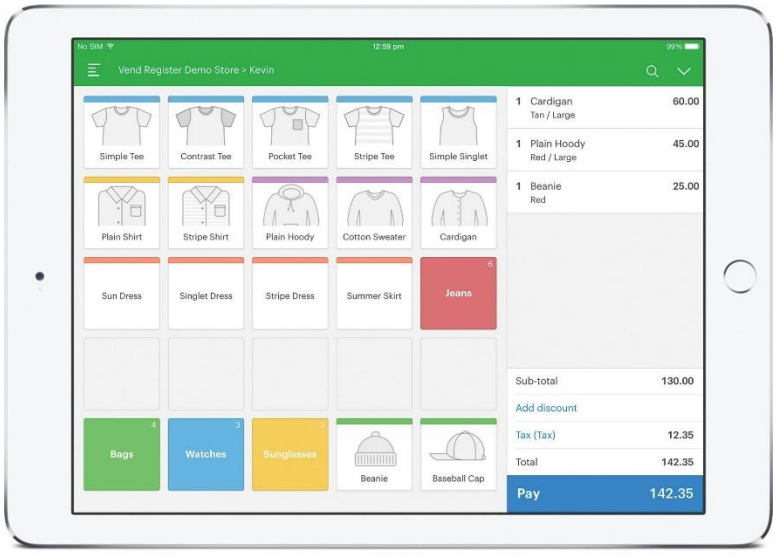


Figura 5. Interfaz de POS Vend

InfoTouch POS: Es una solución robusta, escalable, estable. El sistema ofrece escalabilidad para usuarios de una sola tienda y cadenas de tiendas múltiples con módulos de tiempo y asistencia, seguimiento de inventario completo, seguridad robusta. Utiliza Windows. Los dispositivos se conectan por USB. (12)

Figura 6. Interfaz de POS InfoTouch

Después de realizado un análisis de los sistemas anteriormente expuestos se pueden identificar las dificultades que representa su adquisición y adaptación para el país: Los sistemas Tasven, InfoTouch y Vend presentan costos elevados, el *software* es privativo y algunos no admiten *hardware* diferente al que trae por defecto. La Quorion, presenta desventajas similares en cuanto a la adaptabilidad a pesar de sus facilidades de uso, y resulta un gasto adicional. Todo lo anterior demuestra la necesidad de desarrollar los manejadores para un sistema POS de *hardware* y *software* abierto y la interfaz que permita el acceso a los periféricos de manera remota, posibilitando de esta manera satisfacer necesidades en el entorno nacional, que sea rentable y adaptado al comercio interno, estatal o cuentapropista, contribuyendo a la independencia tecnológica.

1.5 Herramientas y tecnologías para el Desarrollo

Para dar solución al problema planteado se seleccionaron herramientas libres que brindan facilidades para el desarrollo y son utilizadas por el centro CEDIN. A continuación, se describen algunas características de las mismas.

1.5.1 Lenguajes de programación

Lenguaje de Programación C++

C++ es un lenguaje de programación orientado a objetos que toma la base del lenguaje C y le agrega la capacidad de abstraer tipos como en *Smalltalk.* C++ es un lenguaje de programación diseñado a mediados de los años 80 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multi-paradigma.(13)

1.5.2 Lenguaje de Modelado

UML: El Lenguaje Unificado de Modelado (*UML* en inglés) es un lenguaje para especificar, visualizar construir y documentar los artefactos de los sistemas *software*, así como para el modelado del negocio y otros sistemas no *software*. (14)

1.5.3 Herramientas de Modelado

Visual Paradigm: Para el modelado de la solución se escogió como herramienta CASE (*Computer Aided Software Engineering* o Ingeniería de *Software* Asistida por Ordenador) el *software* Visual Paradigm. Es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de *Software*, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.(15)

1.5.4 Bibliotecas

Libusb-1.0: es una biblioteca C que proporciona acceso genérico a dispositivos USB. Está destinado a ser utilizado por los desarrolladores para facilitar la producción de aplicaciones que se comunican con *hardware* USB. Es portable: Utiliza una única API multiplataforma, proporciona acceso a dispositivos USB

en Linux, OS X, Windows, Android, OpenBSD, etc. Es modo de usuario: No se requiere privilegio o elevación especial para que la aplicación se comunique con un dispositivo.(16)

1.5.5 Protocolos y Tecnologías

Para el desarrollo del sistema se utilizaron tecnologías tales como el protocolo *WebSocket* para la comunicación con terceras aplicaciones.

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único *socket* TCP. Está diseñada para ser implementada en navegadores y servidores *Web*, pero puede utilizarse por cualquier aplicación cliente/servidor.(17)

1.5.5.1 Ventajas de WebSocket

- Reduce el uso de la red, ya que evita la necesidad de utilizar los paquetes HTTP que contienen grandes cantidades de datos de cabecera, lo que recae, además, en un mayor trabajo de procesamiento.
- Se reduce la latencia en las conexiones ya que ponen menos carga en los servidores, y esto permite que estos equipos atiendan más conexiones simultáneas.
- Permite atravesar firewalls y servidores proxy. En este caso si un WebSocket detecta la presencia de un servidor proxy, solicita una conexión TCP/IP usando una instrucción Connect HTTP, luego de lo cual se puede pasar por el proxy sin problemas.
- Facilita una mayor escalabilidad en la Web debido a su eficiencia al momento de mantener conexiones persistentes con los servidores.
- Es más rápido que el protocolo HTTP(18)

1.5.6 *Hardware*

Raspberry Pi: Es una computadora de tamaño reducido y placa única, de bajo costo y uso libre. Puede ser utilizado en proyectos de electrónica, y para muchas de las cosas que hace una PC de escritorio. El *software* que utiliza es de código abierto siendo su sistema operativo oficial una versión adaptada a Debian llamada Raspbian.

Lector de código de barras: Un lector de códigos de barras es un dispositivo HID que puede conectarse por USB y permite escanear códigos de barras emitiendo la información que muestra el mismo.

Lector de tarjetas magnéticas: Un lector de tarjeta magnética es un dispositivo HID que puede conectarse por USB y permite leer tarjetas magnéticas convirtiendo la información almacenada en la banda magnética en datos digitales.

Impresora de recibos: Una impresora de recibos es un dispositivo que utiliza un terminal de punto de venta para la impresión de los recibos, pueden ser de varios tipos, térmicas, matriciales o láser, que pueden variar en la calidad de impresión o el ruido y pueden ser conectadas por USB. El sistema es compatible con impresoras Epson basadas en el protocolo ESC/POS.

1.5.7 Entorno Integrado de Desarrollo (IDE)

Para la implementación del sistema se seleccionó el *framework Qt*, porque es un marco de trabajo hecho sobre C++, que posibilita el rendimiento en aplicaciones que necesiten acceder a los dispositivos e interfaces de una computadora.

Framework Qt: Es un completo marco de trabajo de C++ para el desarrollo de aplicaciones de interfaz gráfica de usuario multiplataforma utilizando un enfoque de "escribir una vez, compilar en cualquier lugar". Qt permite a los programadores utilizar un solo árbol de código fuente para las aplicaciones que se ejecutarán en Windows, Mac OS X, Linux, Solaris, HP-UX y muchas otras versiones de Unix con X11. Las bibliotecas y herramientas de Qt también forman parte de Qtopia Core, un producto que proporciona su propio sistema de ventanas encima del Linux incorporado.(19) Qt puede ser utilizado más fácilmente por los programadores mediante el IDE Qt Creator.

Qt Creator: Es un Entorno Integrado de Desarrollo (IDE) creado por Trolltech, multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil. Entre sus principales características se encuentran:

- Posee un avanzado editor de código C++.
- Soporta los lenguajes: C#/.NET Lenguajes (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto, integrado.
- Depurador visual.

- Resaltado y auto-completado de código.
- Soporte para refactorización de código.(20)

1.5.8 Metodología de desarrollo de *software*

Una metodología de desarrollo de *software* consiste en un conjunto de procedimientos y pasos a seguir para desarrollar un proyecto informático. En dependencia de las características del proyecto se escoge la metodología, con el objetivo de minimizar el tiempo, y a su vez el costo, sin perder la calidad del producto, garantizando así la eficiencia y eficacia del trabajo realizado. En los últimos años las tendencias en cuanto a metodologías de desarrollo han seguido dos líneas principales: ágiles y tradicionales. Dentro de las ágiles se encuentran XP, SCRUM, AUP, y de las tradicionales se pueden encontrar otras como RUP. Debido a que se necesitan pocos artefactos, hay pocos roles y el grupo de trabajo es pequeño se seleccionó una metodología ágil, por las facilidades que brinda para este tipo de proyecto.

1.5.8.1 Metodología AUP-UCI

La metodología AUP-UCI es una variación de la metodología AUP (*Agile Unified Process*) para la Universidad de las Ciencias Informáticas. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Con esta metodología la actividad productiva de la UCI logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a 1 la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11.(21)

1.5.8.2 Fases de AUP-UCI

1. **Inicio**: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planificación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto.
2. **Ejecución**: En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura.

Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

3. **Cierre**: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del mismo.

1.5.8.3 Escenarios de AUP-UCI

Escenario No 1: Proyectos que modelen el negocio con Casos de Uso del Negocio (CUN), solo pueden modelar el sistema con Casos de Uso del Sistema (CUS).

Escenario No 2: Proyectos que modelen el negocio con Modelo Conceptual (MC) solo pueden modelar el sistema con CUS.

Escenario No 3: Proyectos que modelen el negocio con Descripción de Proceso de Negocio (DPN), solo pueden modelar el sistema con Descripción de Requisitos por Proceso DRP.

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de usuario (HU).

1.5.8.4 Selección de la metodología

Para el desarrollo de la solución se eligió la metodología AUP-UCI y su escenario No. 4 para modelar el sistema, debido a que se ajusta a las condiciones específicas del proyecto y la universidad, y está pensada para proyectos de corta duración y para equipos pequeños. AUP-UCI constituye un eficiente modelo a seguir para el desarrollo del proyecto y con esta se espera alcanzar con éxito los objetivos planteados con anterioridad.

Conclusiones parciales:

- Se obtuvo como resultado de la investigación, mediante el análisis de algunos de los Sistemas POS existentes, que su adquisición representa un gasto adicional al país, no contribuyen a la independencia tecnológica, algunos presentan *hardware* específico o traen los dispositivos integrados, no brindan acceso al código fuente, algunos utilizan interfaces de conexión para periféricos no estandarizadas, y el software POS en algunos sistemas se encuentra embebido sin posibilidad de actualización o modificación por terceros.

- Se logró definir las tecnologías y herramientas necesarias para la implementación y la metodología de desarrollo a utilizar.
- El uso del protocolo WebSocket para la comunicación permite aumentar el rendimiento para el manejo de conexiones y el intercambio de información entre aplicaciones.
- Se logró definir la utilización del estándar USB 2.0, el cual se utiliza actualmente en la mayoría de los dispositivos HID y presenta características suficientes para el manejo de dispositivos de un Sistema POS.

Capítulo 2: Propuesta de solución

Introducción

En este capítulo se dan a conocer las características de la solución propuesta para resolver el problema de la investigación, se realiza una descripción de la arquitectura propuesta, y los requerimientos del sistema. Además, mediante la metodología de desarrollo de *software* seleccionada, se detallan los artefactos generados en sus diferentes fases.

2.1 Búsqueda de información útil para desarrollar el sistema

Para obtener información del sistema se realizaron entrevistas a trabajadores vinculados directamente a las ventas, y se investigó a través de la *Web* cómo funciona este proceso a mayor y menor escalas, partiendo de la particularidad de sus variaciones en cuanto a la ejecución según el tipo y el lugar en que se realiza.

2.2 Especificación de requisitos

La especificación de requisitos de *software* es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de historias de usuario que describen todas las interacciones que tendrán los usuarios con el *software.* Además de las historias de usuario, la especificación de requisitos de *software* también contiene requisitos no funcionales o complementarios.

Tipo	Requisito	Complejidad
RF1	Crear una conexión	Alta
RF2	Interpretar JSON	Alta
RF3	Crear interfaces de los dispositivos	Alta
RF4	Leer código de barras	Media
RF5	Leer tarjeta magnética	Media
RF6	Imprimir recibo	Media
RF7	Enviar mensaje	Baja

Tabla 1. Requisitos funcionales

2.2.1 Descripción de las Historias de Usuario

Las Historias de Usuario (HU) constituyen una manera sencilla de encapsular los requisitos del sistema, están diseñadas para un fácil entendimiento minimizando los detalles, de forma que se pueda determinar el costo de la implementación, de acuerdo con las necesidades del sistema y son utilizadas por las metodologías ágiles. Una HU está compuesta generalmente por la siguiente información:

- **Código**: Posee el número asignado a la HU.
- **Usuario**: El usuario del sistema que utiliza o protagoniza la HU.
- **Nombre de la HU:** Atributo que contiene el nombre de la HU.
- **Referencia**: Es el conjunto de identificadores de las HU de las cuales depende la historia actual que está en desarrollo.
- **Prioridad**: Evidencia el nivel de prioridad de la HU en el negocio.
- **Puntos estimados**: Este atributo es una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo.
- **Iteración Asignada**: Número de la iteración en la cual se desarrollará la HU.
- **Descripción**: Posee una breve descripción de lo que realizará la HU.
- **Observaciones**: Algunas aclaraciones que es importante señalar acerca de la HU.

A continuación, se muestran las HU definidas:

Historia de Usuario	
Código: HU-1	**Usuario:** Servidor
Nombre Historia: Crear conexión	
Referencia:	**Prioridad:** Alta
Puntos Estimados:3	**Iteración:** 1
Descripción: El sistema debe crear una conexión *Web* utilizando el protocolo *WebSocket* que permita el intercambio de mensajes entre el usuario y el sistema, y el acceso a los dispositivos.	
Observaciones:	

Tabla 2. Historia de Usuario 1

Historia de Usuario	
Código: HU-2	**Usuario:** Servidor
Nombre Historia: Interpretar JSON	
Referencia:	**Prioridad:** Media
Puntos Estimados:3	**Iteración:** 1
Descripción: El sistema debe interpretar un mensaje recibido mediante el protocolo *WebSocket*, del cual se deberán reconocer los parámetros *name, cash, total, products*, y los parámetros de cada producto. El orden en que estén los parámetros no influye en el orden en que se impriman las líneas del recibo.	
Observaciones: El mensaje puede contener un JSON que contiene los datos de un recibo de pago.	

Tabla 3. Historia de Usuario 2

Historia de Usuario	
Código: HU-3	**Usuario:** Servidor
Nombre Historia: Crear interfaces de los dispositivos	
Referencia:	**Prioridad:** Alta
Puntos Estimados:3	**Iteración:** 1
Descripción: El sistema debe utilizar una biblioteca que permita crear las interfaces de los dispositivos utilizando la biblioteca libusb-1.0. Los dispositivos deben poder ser conectados en cualquier momento.	
Observaciones:	

Tabla 4. Historia de Usuario 3

Historia de Usuario	
Código: HU-4	**Usuario:** Empleado
Nombre Historia: Leer código de barras	
Referencia:3,7	**Prioridad:** Alta
Puntos Estimados:2	**Iteración:** 2
Descripción: El sistema debe permitir leer un código de barras y enviarlo a través del protocolo *WebSocket* al dispositivo del usuario.	
Observaciones:	

Tabla 5. Historia de Usuario 4

Historia de Usuario	
Código: HU-5	**Usuario:** Empleado
Nombre Historia: Leer tarjeta magnética	
Referencia: 3,7	**Prioridad:** Alta
Puntos Estimados:2	**Iteración:** 2
Descripción: El sistema debe permitir leer una tarjeta magnética y enviar los datos a través del protocolo *WebSocket* al dispositivo del usuario.	
Observaciones:	

Tabla 6. Historia de Usuario 5

Historia de Usuario	
Código: HU-6	**Usuario:** Empleado
Nombre Historia: Imprimir recibo	

Referencia: 3,7,2	**Prioridad:** Alta
Puntos Estimados:2	**Iteración:** 3
Descripción: El sistema debe permitir imprimir un recibo de pago con los datos enviados por el usuario. Si el mensaje no tiene la estructura correcta no se imprimirá el recibo.	
Observaciones: Se debe recibir un mensaje con la información que se desee imprimir.	

Tabla 7. Historia de Usuario 6

Historia de Usuario	
Código: HU-7	**Usuario:** Servidor
Nombre Historia: Enviar mensaje	
Referencia:	**Prioridad:** Media
Puntos Estimados:1	**Iteración:** 3
Descripción: El sistema debe permitir enviar un mensaje al usuario mediante el protocolo *WebSocket* para enviar las lecturas de código de barras y tarjeta magnética.	
Observaciones: El mensaje puede contener un JSON	

Tabla 8. Historia de Usuario 7

2.2.2 Estimación de tiempo de desarrollo por HU

Los programadores estiman el esfuerzo en tiempo que necesita una Historia de Usuario para ser implementada. La estimación se realiza por semanas y de acuerdo a la complejidad es el tiempo que se propone.

Historias de usuario	Puntos de estimación (semanas)
Crear conexión	3
Interpretar JSON	3
Crear interfaces de los dispositivos	3
Leer código de barras	2
Leer tarjeta magnética	2
Imprimir recibo	2
Enviar mensaje	1

Tabla 9. Estimación por esfuerzo de las HU

2.2.3 Plan de iteraciones

Para la elaboración de un plan de Iteraciones es necesario tomar las Historias de Usuario para distribuirlas en varias iteraciones. De esta manera al convertirse las HU en tareas de programación, se le realizan pruebas de aceptación, proporcionando así que se puedan obtener resultados incrementales, y alcanzar un producto final de manera creciente.

- **Iteración 1:** En esta iteración se implementan las HU 1, 2 y 3 que es de alta prioridad para el proceso del negocio, y de mayor complejidad.
- **Iteración 2:** Tiene como objetivo la implementación de las HU 4 y 5, que constituyen funciones importantes para la comunicación Cliente-Servidor y el acceso a los servicios del sistema.
- **Iteración 3:** En esta iteración después de un desarrollo progresivo se implementan las HU: 6 y 7 que presentan menor complejidad.

Iteración	Historias de Usuario	Puntos de estimación (semanas)

1	Crear conexión Interpretar JSON Crear interfaces de los dispositivos	9
2	Leer código de barras Leer tarjeta magnética	4
3	Imprimir recibo Enviar Mensaje	3

Tabla 10. Plan de iteraciones

2.2.4 Plan de entregas

El plan de entregas se crea después de definir en qué iteración se implementarán las Historias de Usuario. A continuación se muestran las Historias de Usuario que deben ser entregadas por iteraciones:

Historias de usuario	Iteración 1	Iteración 2	Iteración 3
Crear conexión Interpretar JSON Crear interfaces de los dispositivos	Entregable 1		
Leer código de barras Leer tarjeta magnética		Entregable 2	
Imprimir recibo Enviar Mensaje			Entregable 3

Tabla 11. Plan de entregas

2.3 Arquitectura del Sistema

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.(22)

El sistema que se propone está basado en una arquitectura Cliente-Servidor, utilizando el protocolo *WebSocket*, el cual permite el intercambio de mensajes a través de un *socket* TCP. Los mensajes que enviará el cliente serán peticiones para acceder a los dispositivos del Terminal de Punto de Venta. Los mensajes pueden ser para enviar un código de barras, una tarjeta magnética o imprimir un recibo, en el caso del último deberá enviar la información en el formato que se define como plantilla. Las respuestas del servidor serán en el caso de las lecturas, el envío de la información de las mismas.

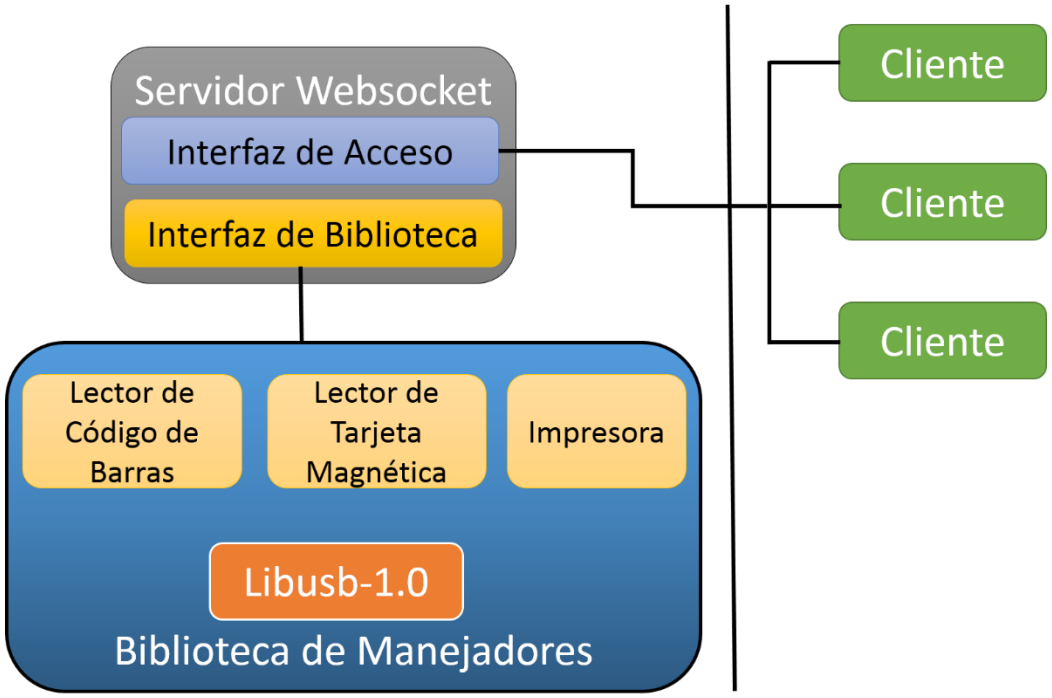


Figura 7. Arquitectura Cliente-Servidor de un sistema POS aplicado a la solución propuesta

2.4 Intercambio de Mensajes

Para el intercambio de información con posibles clientes el sistema utiliza mensajes en forma de JSON (*Java Script Object Notation*). JSON es un estándar para encapsular colecciones de objetos no ordenados

en texto plano. Cada objeto está compuesto por pares **<nombre>:<valor>** y puesto entre llaves (“**{ }**”), un objeto puede ser o contener arreglos de objetos, los arreglos pueden denotarse de la siguiente manera {“arreglo”: [“id”: 1, “id”: 2, “id”: 3,]}. Los arreglos pueden contener tipos de datos como números, booleanos y objetos.

Los mensajes que se enviarán hacia el cliente pueden ser de 2 tipos:

1. **Envío de código de barras:** Cuando el sistema captura un código de barras lo envía a los clientes conectados de la forma **{“bc”: “valor”}**.
2. **Envío de tarjeta magnética:** Cuando el sistema captura una tarjeta magnética envía la información a los clientes conectados de la forma **{“mc”: “valor”}**.

Los mensajes que recibirá el sistema por parte del cliente serán de la siguiente manera:

Imprimir recibo: Cuando el cliente desea imprimir un recibo debe enviar un mensaje al servidor en formato JSON:

1.

```
{
"name" : "tienda pepe",
"total" : 10,
 "cash" : 90,
 "change" : 78,
 "products" : [
        {
                "name" : "arroz",
                "quantity" : 10,
                "cost”: 10,
                "total" : 40
        },
        {
                "name" : "ensalada de pepino",
                "quantity: 10,
                "cost" : 10,
```

```
            "total" : 40
        },
        ]
    }
```

El sistema se encargará de convertir el mensaje en información que pueda entender la impresora mediante el requisito funcional Interpretar JSON.

2.5 Diagrama de Clases

Un diagrama de clases en UML permite describir la estructura de un sistema y la relación entre los objetos; representa generalmente relaciones, interfaces, interacciones y colaboración entre las clases, notas, paquetes, restricciones y todos los elementos que conforman un programa orientado a objetos.

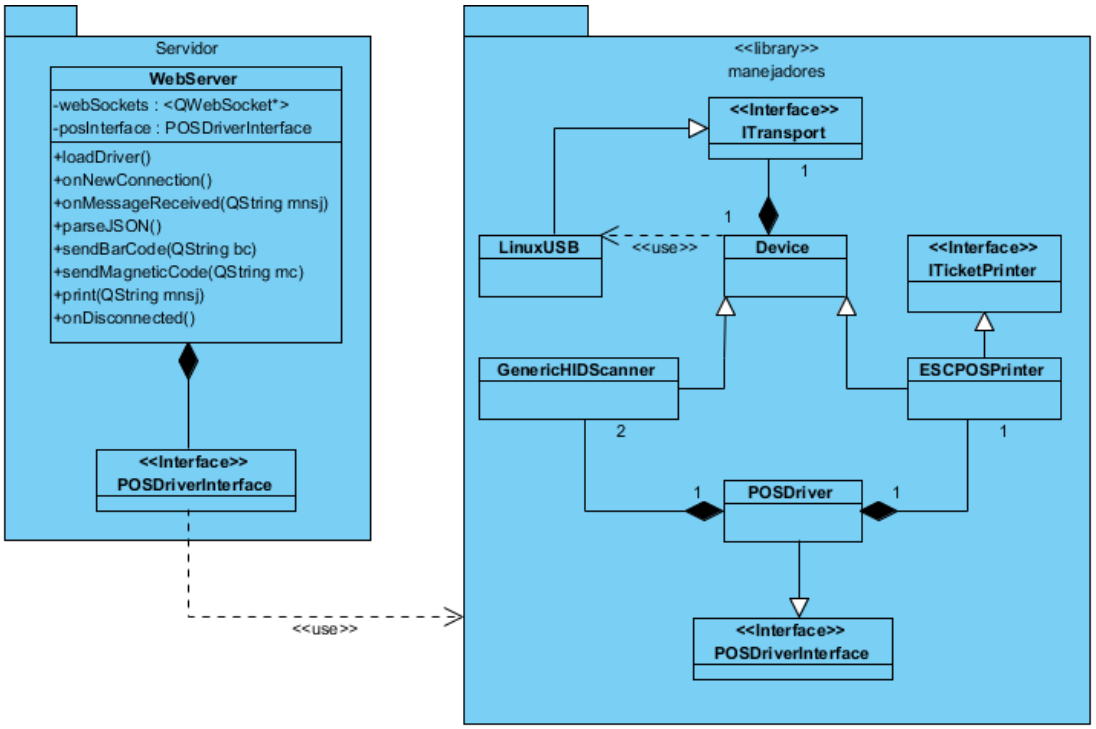


Figura 8. Relación entre las clases de la solución propuesta

2.6 Patrones de diseño

Los patrones de diseño son soluciones dadas a problemas específicos, que se basan en la aplicación de ciertos estilos para la implementación de un sistema, los cuales se codifican en un formato estructurado que logre describir la solución y el problema, permitiendo de esta manera ser reutilizable.

2.6.1 Patrones GRASP

Los patrones GRASP, acrónimo de (*General Responsibility Asignment Software Patterns*), como su nombre lo indica son patrones de asignación de responsabilidades en la programación orientada a objetos tales como **hacer** o **conocer.**

Entre las responsabilidades de **hacer** de un objeto se encuentran:

- Hacer algo él mismo, como crear un objeto o hacer un cálculo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos.

Entre las responsabilidades de **conocer** de un objeto se encuentran:

- Conocer los datos privados encapsulados.
- Conocer los objetos relacionados.
- Conocer las cosas que puede derivar o calcular.(14)

Controlador: Este patrón se pone de manifiesto en la clase **WebServer** (Figura), debido a que en ella se implementa el acceso a las operaciones principales del sistema, inicializa las instancias de los manejadores de los periféricos, gestiona las conexiones con los clientes, y el flujo de mensajes entre las partes que intervienen en el proceso.

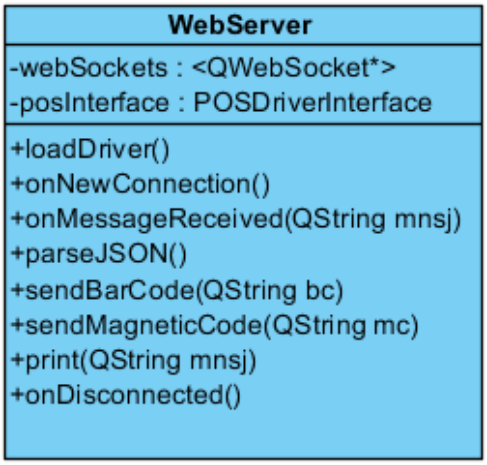


Figura 9. Clase WebServer

Alta Cohesión: Este patrón se evidencia en las clases (ver Figura 8): **LinuxUSB**, la cual se encarga de la lógica para encontrar el dispositivo conectado con determinadas características predefinidas (identificador de fabricante y de producto); **ESCPOSPrinter,** la cual se encarga de la comunicación con impresoras que utilicen el protocolo ESC/POS para impresoras Epson; **GenericHIDScanner,** responsable de reconocer dispositivos HID genéricos como lector de código de barras y lector de tarjeta magnética permitiendo la captura y reenvío de los datos que estos proporcionan.

2.6.2 Patrones GoF

Los patrones GoF (Pandilla de los Cuatro, en inglés *Gang of Four*) cuyo nombre proviene de sus 4 creadores. Estos corresponden a patrones de diseño que brindan solución a problemas de creación de instancias, ayudan a encapsular y abstraer dicha creación.

Fachada: Se utiliza en situaciones en que se puede necesitar proveer de una interfaz unificada simple para acceder a módulos de un subsistema. Se pone de manifiesto en la clase interfaz **POSDriverInterface**.

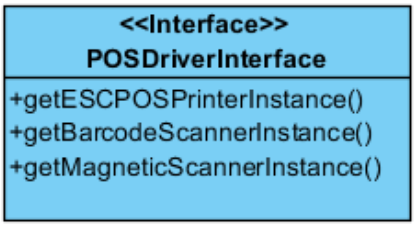


Figura 10. Clase POSDriverInterface

Factoría: Este patrón se pone de manifiesto en la clase **POSDriver** al ser la encargada de crear las instancias de los dispositivos; estas deben seguir una lógica para el reconocimiento de los periféricos conectados que corresponden a cada una.

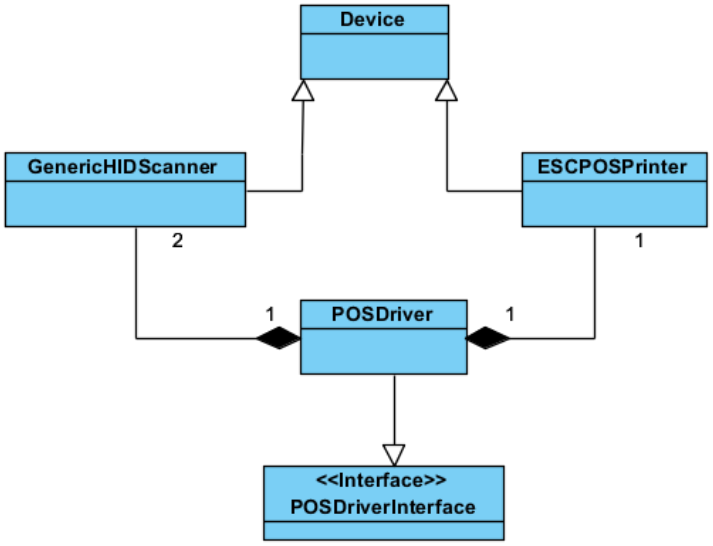


Figura 11. Diagrama de Clases donde se evidencia el patron Factoría

Conclusiones parciales

- Mediante la metodología de desarrollo de software seleccionada, se generaron los artefactos necesarios para describir la solución propuesta
- Se estableció una arquitectura aceptable utilizando patrones de diseño conocidos.
- Los mensajes definidos para la comunicación entre los clientes con la solución propuesta fueron diseñados para evitar el envío de información innecesaria.

Capítulo 3: Implementación y Pruebas

Introducción

A partir de la propuesta de solución diseñada en el capítulo anterior, se inicia la implementación del sistema, de acuerdo con las iteraciones planificadas, basándose en la metodología AUP-UCI. Se realizan las tareas de ingeniería correspondientes a las Historias de Usuario, se describe la implementación y se ejecutan las pruebas de *software*.

3.1 Tareas de ingeniería

Las tareas de ingeniería son generadas a partir de las Historias de Usuario en sus respectivas iteraciones, desglosando un problema grande en varias partes para mejor organización y planificación, a continuación, se muestra una tabla con las tareas correspondientes a las HU.

Historia de Usuario	Tareas por HU
Crear una conexión	1. Implementar servidor WebSocket para el intercambio de mensajes. 2. Definir operaciones de la clase servidor.
Interpretar JSON	3. Desarrollar funcionalidad para interpretar un JSON.
Crear interfaces de los Dispositivos	4. Implementar la biblioteca de manejadores. 5. Implementar clase constructora de las interfaces de los dispositivos.
Leer código de barras	6. Desarrollar una funcionalidad que permita capturar la lectura del lector de código de barras.
Leer tarjeta magnética	7. Desarrollar una funcionalidad que permita capturar la lectura del lector de tarjeta magnética.

Imprimir recibo	8. Desarrollar una funcionalidad que permita imprimir un recibo.
Enviar mensaje	9. Desarrollar funcionalidad que permita enviar un mensaje al cliente.

Tabla 12. Tareas de Ingeniería

Las tareas de ingeniería que se muestran a continuación tienen los siguientes campos:

- **No. de tarea:** Numeración continua que identifica a la tarea.
- **No. de HU:** Número de la HU a la cual pertenece.
- **Nombre de la tarea:** Identificación literal de la tarea.
- **Tipo de tarea:** Tipo de tarea, dígase diseño, desarrollo, prueba.
- **Puntos estimados:** Representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizará para su realización.
- **Descripción:** Se describe en que consiste la tarea y que elementos deben cumplirse para declarar la tarea terminada.

Tarea de ingeniería	
TI: 1	**HU:1**
Nombre de Tarea: Implementar servidor WebSocket para el intercambio de mensajes	
Tipo: Desarrollo	**Puntos Estimados:** 2
Descripción: Se realiza el diseño y la implementación de un servidor WebSocket para la comunicación e intercambio de mensajes con un cliente que se conectará utilizando un dispositivo móvil	

Tabla 13. Tarea de Ingeniería 1

Tarea de ingeniería	
TI: 2	**HU: 1**
Nombre de Tarea: Definir operaciones de la clase servidor	
Tipo: Desarrollo	**Puntos Estimados:** 1
Descripción: Se declaran todas las funcionalidades del servidor, y se	

implementan haciendo posible el intercambio de mensajes, y que puedan existir varias conexiones.

Tabla 14. Tarea de Ingeniería 2

Tarea de ingeniería	
TI: 3	**HU: 2**
Nombre de Tarea: Desarrollar funcionalidad para interpretar un JSON	
Tipo: Desarrollo	**Puntos Estimados:** 3
Descripción: Se implementa una funcionalidad que al recibir un JSON pueda convertirlo en texto formateado para enviar comandos a la impresora	

Tabla 15. Tarea de ingeniería 3

Tarea de ingeniería	
TI: 4	**HU: 3**
Nombre de Tarea: Implementar la biblioteca de manejadores.	
Tipo: Desarrollo	**Puntos Estimados:** 2
Descripción: Se implementa una biblioteca para manejar las interfaces de los dispositivos con libusb-1.0 basándose en una implementación realizada con la versión anterior.	

Tabla 16. Tarea de ingeniería 4

Tarea de ingeniería	
TI: 5	**HU: 3**
Nombre de Tarea: Implementar clase constructora de las interfaces de los dispositivos	
Tipo: Desarrollo	**Puntos Estimados:** 1
Descripción: Se implementa una clase que se encargue de crear las interfaces de los dispositivos y permita el manejo de ellos mediante el uso de la biblioteca de manejadores.	

Tabla 17. Tarea de ingeniería 5

Tarea de ingeniería	
TI: 6	**HU: 4**
Nombre de Tarea: Desarrollar una funcionalidad que permita capturar la lectura del lector de código de barras	
Tipo: Desarrollo	**Puntos Estimados:** 2
Descripción: Se implementa una función que mediante las interfaces creadas pueda capturar la lectura del lector de código de barras y automáticamente enviarlo en un mensaje en forma de JSON al cliente conectado.	

Tabla 18. Tarea de Ingeniería 6

Tarea de ingeniería	
TI: 7	**HU: 5**
Nombre de Tarea: Desarrollar una funcionalidad que permita capturar la lectura del lector de tarjeta magnética.	
Tipo: Desarrollo	**Puntos Estimados:** 2
Descripción: Se implementa una función que mediante las interfaces creadas pueda capturar la lectura del lector de tarjeta magnética y enviarlo en forma de JSON al cliente conectado.	

Tabla 19. Tarea de ingeniería 7

Tarea de ingeniería	
TI: 8	**HU: 6**
Nombre de Tarea: Desarrollar una funcionalidad que permita imprimir un recibo.	
Tipo: Desarrollo	**Puntos Estimados:** 2
Descripción: Se implementa una funcionalidad que permita imprimir un recibo con el texto formateado, mediante los comandos a la impresora generados al convertir el JSON recibido en órdenes entendibles por la biblioteca de manejadores.	

Tabla 20.Tarea de ingeniería 8

Tarea de ingeniería	
TI: 9	**HU: 7**
Nombre de Tarea: Desarrollar una funcionalidad que permita enviar un mensaje al cliente	
Tipo: Desarrollo	**Puntos Estimados:**0,5
Descripción: Se implementa una función que al recibir la señal de lectura del lector de código de barras o tarjeta magnética lo envíe al cliente en un mensaje conformado por un JSON con dicha información.	

Tabla 21. Tarea de ingeniería 9

3.2 Descripción del Sistema

El sistema implementado se compone de una aplicación para atender conexiones WebSocket: **WebServer**, una biblioteca de manejadores: **libpos** y una interfaz para acceder a las instancias de los dispositivos: **POSDriverInterface**

3.2.1 Descripción de WebServer

La clase **WebServer** es la que controla el acceso los servicios, carga los *drivers*, realiza las funcionalidades principales implementadas para acceder a los dispositivos y maneja el intercambio de mensajes. Cuando se inicia el sistema se el servidor comienza a escuchar las conexiones, se cargan los *drivers* y se esperan las peticiones del cliente. Cuando algún dispositivo lee un paquete lo envía al cliente conectado, cuando recibe un mensaje para imprimir, lo convierte en comandos para ejecutarlos en la impresora.

3.2.2 Descripción de POSDriverInterface

Esta clase se encarga de crear instancias de cada dispositivo y está implementada utilizando la biblioteca de manejadores implementada con libusb-1.0.

3.2.3 Biblioteca de manejadores:

La biblioteca de manejadores se compone por varias clases principales: **LinuxUSB**, la cual se encarga de la lógica para encontrar el dispositivo conectado con determinadas características predefinidas (identificador de fabricante y de producto); **ESCPOSPrinter**, la cual se encarga de la comunicación con impresoras que utilicen el protocolo ESC/POS para impresoras; **GenericHIDScanner**, responsable de

reconocer dispositivos HID genéricos como lector de código de barras y lector de tarjeta magnética permitiendo la captura y reenvío de los datos que estos proporcionan; y **POSDriver** que se encarga de crear e inicializar las instancias de los dispositivos.(ver figura 8).

3.3 Manejo de los dispositivos con la biblioteca libusb-1.0

Para poder manejar los dispositivos haciendo uso de libusb-1.0 los desarrolladores hicieron variaciones en cuanto a la versión anterior (libusb-0.1). A continuación se muestra una secuencia lógica de pasos que se deben seguir para la implementación de una aplicación que necesite acceso a dispositivos del sistema.

1. Inicializar creando una sesión con **libusb_context** y llamando a **libusb_init**.
2. Llamar a la funcionalidad **libusb_get_device_list** para obtener la lista de dispositivos conectados.
3. Recorrer los dispositivos conectados para acceder a la información de los mismos mediante los **libusb_descriptor.**
4. Obtener el dispositivo abierto mediante **libusb_open**, o **libusb_open_device_with_vid_pid**, cuando se conoce el id del producto y del fabricante.
5. Vaciar la lista de dispositivos mediante **libusb_free_device_list**.
6. Obtener la interfaz del dispositivo con **libusb_claim_interface**. (Se necesita conocer los números de las interfaces de los dispositivos).
7. Realizar la operación de E/S.
8. Liberar el dispositivo con **libusb_release_interface**.
9. Cerrar el dispositivo abierto con **libusb_close**.
10. Cerrar la sesión con **libusb_exit**.
11. El *kernel* del sistema operativo intentará proveer un manejador para el dispositivo que se conecte a la PC, por eso se necesita liberar el driver del *kernel* para que el dispositivo pueda utilizar el que escribió el programador. Para esto existe la función **libusb_detach_kernel_driver**.

Para comprobar si el *kernel* dejó de asignarle un driver al dispositivo se utiliza la función **libusb_kernel_driver_active**.

3.4 Estandarización de la implementación

La implementación del sistema fue desarrollada en el lenguaje inglés para facilitar el entendimiento fuera del ámbito nacional. Como estándar de codificación se utilizó *CamelCase* aplicando una buena práctica de programación.

3.5 Diagrama de despliegue

Un diagrama de despliegue permite modelar la arquitectura de un sistema en tiempo de ejecución, en el cual se muestra la disposición física de los artefactos en nodos. A continuación, se muestra el diagrama de despliegue para el sistema:

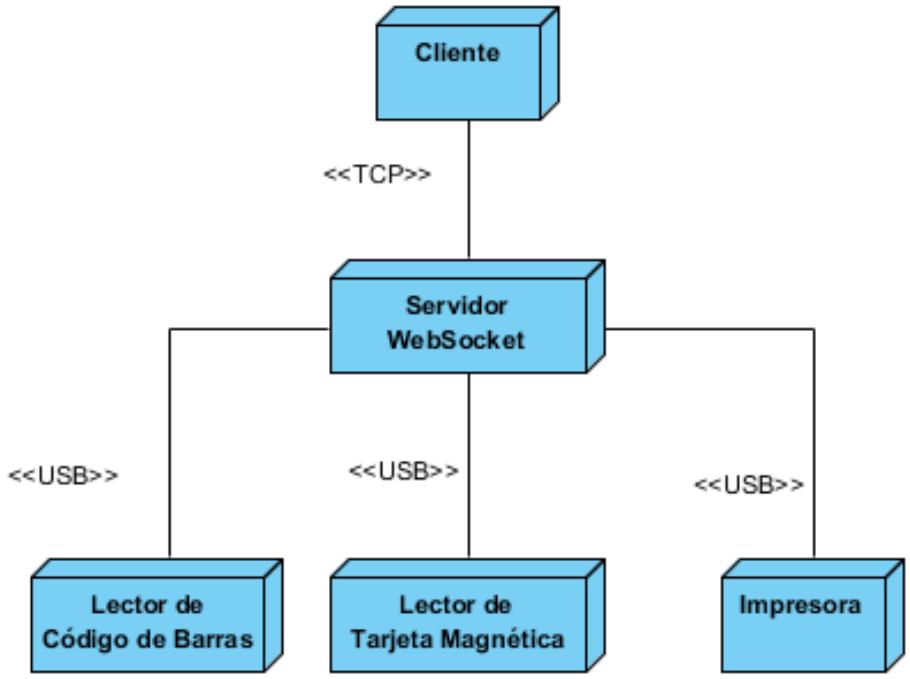


Figura 12. Diagrama de despliegue

Cliente: Representa el dispositivo móvil que consumirá los servicios brindados por el servidor.

Controlador WebSocketServer: Este nodo representa el servidor *WebSocket* que permite el intercambio de información, y el acceso a los dispositivos conectados.

Lector de Código de Barras: Este nodo representa el dispositivo conectado al nodo físico donde corre el servidor para leer códigos de barras.

Lector de Tarjeta Magnética: Este nodo representa el dispositivo conectado al nodo físico donde corre el servidor para leer tarjetas magnéticas.

Impresora: Este hace referencia al dispositivo conectado al nodo físico donde corre el servidor. para imprimir recibos.

Conexión por USB: Es la conexión que utilizan los dispositivos HID que se encuentran en el servidor.

Conexión por TCP: Es la conexión que utiliza el Servidor *WebSocket* para el intercambio de la información con el cliente.

3.6 Pruebas de Aceptación

Las Pruebas de Aceptación son pruebas formales llevadas a cabo para permitir a un usuario, a un cliente o a otra entidad autorizada, determinar si acepta un entregable.(23)

Las Pruebas de Aceptación que se describen a continuación son tablas que se basan en las Historias de Usuario, con el fin de comprobar si las funcionalidades implementadas se ejecutan correctamente y se obtienen los resultados esperados. Los campos que contienen las tablas son los siguientes:

- **Código:** Identifica la prueba en cuestión, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Nombre del caso de prueba.
- **Descripción:** Acciones que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos en general que debe tener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución:** Incluye las entradas necesarias del sistema y los pasos necesarios para realizar el caso de prueba.
- **Resultados Esperados:** Respuesta que debe dar el sistema luego de aplicar el caso de prueba.
- **Resultado Obtenido:** Respuesta visual del sistema después de realizar el caso de prueba.

- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

A continuación se muestran los resultados arrojados durante la ejecución de las pruebas realizadas al sistema, correspondientes a las Historias de Usuarios:

Pruebas Satisfactorias

Prueba de Aceptación	
Caso de Prueba: 2	**No. De Historia: 1**
Nombre: Crear conexión	
Descripción: El sistema debe crear una conexión Web utilizando el protocolo WebSocket que permita el intercambio de mensajes entre el usuario y el sistema, y el acceso a los dispositivos.	
Condiciones de Ejecución: El dispositivo cliente y el servidor deben encontrarse en una misma subred TCP.	
Entradas/Pasos de Ejecución: 1- Desde un cliente de prueba intentar una conexión WebSocket.	
Resultado esperado: El sistema debe crear una conexión.	
Resultado obtenido: El sistema permitió la conexión.	
Evaluación de Prueba: Satisfactoria	

Tabla 22. Prueba de aceptación 2

Prueba de Aceptación	
Caso de Prueba: 4	**No. De Historia: 2**
Nombre: Interpretar JSON	
Descripción: El sistema debe interpretar un mensaje recibido mediante el protocolo WebSocket.	

Condiciones de Ejecución: El mensaje recibido debe ser un JSON con una estructura determinada.
Entradas/Pasos de Ejecución: 1- Desde un cliente de prueba previamente conectado enviar al sistema un mensaje de impresión de recibo en el formato definido.
Resultado esperado: El sistema debe interpretar el mensaje recibido.
Resultado obtenido: El mensaje fue interpretado correctamente.
Evaluación de Prueba: Satisfactoria

Tabla 23. Prueba de aceptación 4

Prueba de Aceptación	
Caso de Prueba: 6	**No. De Historia: 3**
Nombre: Crear interfaces de los dispositivos	
Descripción: El sistema debe utilizar una biblioteca que permita crear las interfaces de los dispositivos utilizando la biblioteca libusb-1.0 y cargar los manejadores.	
Condiciones de Ejecución: Debe estar previamente instalada la biblioteca libusb-1.0.	
Entradas/Pasos de Ejecución: 1- Conectar los dispositivos	
Resultado esperado: El sistema debe cargar los manejadores correctamente y crear las interfaces.	
Resultado obtenido: Los controladores se cargaron correctamente	

Evaluación de Prueba: Satisfactoria

Tabla 24. Prueba de aceptación 3

Prueba de Aceptación	
Caso de Prueba: 8	**No. De Historia: 4**
Nombre: Leer código de barras	
Descripción: El sistema debe leer un código de barras y enviarlo a través del protocolo WebSocket al dispositivo del cliente.	
Condiciones de Ejecución: Debe estar conectado el dispositivo de lectura y un cliente conectado.	
Entradas/Pasos de Ejecución: 1-Escanear un código de barras.	
Resultado esperado: El sistema debe capturar el código escaneado y realizar el envío al cliente conectado.	
Resultado obtenido: El cliente de prueba recibió un JSON con el código escaneado	
Evaluación de Prueba: Satisfactoria	

Tabla 25. Prueba de aceptación 8

Prueba de Aceptación	
Caso de Prueba: 9	**No. De Historia: 5**
Nombre: Leer tarjeta magnética	
Descripción: El sistema debe leer una tarjeta magnética y enviar la lectura a través del protocolo WebSocket al dispositivo del cliente.	
Condiciones de Ejecución: Debe estar conectado el dispositivo de lectura y un cliente conectado.	

Entradas/Pasos de Ejecución: 1-Escanear una tarjeta magnética.
Resultado esperado: El sistema debe capturar los datos de la tarjeta escaneada y realizar el envío al cliente conectado.
Resultado obtenido: El cliente de prueba recibió un JSON con la información de la tarjeta magnética.
Evaluación de Prueba: Satisfactoria

Tabla 26 Prueba de aceptación 9

Prueba de Aceptación	
Caso de Prueba: 10	**No. De Historia: 6**
Nombre: Imprimir recibo	
Descripción: El sistema debe imprimir un recibo luego de que se reciba un mensaje de impresión.	
Condiciones de Ejecución: El mensaje recibido debe ser válido. La impresora debe estar conectada.	
Entradas/Pasos de Ejecución: 1- Desde un cliente conectado enviar un mensaje de impresión.	
Resultado esperado: Impresión de un recibo con texto formateado.	
Resultado obtenido: Se imprimió el recibo correctamente	
Evaluación de Prueba: Satisfactoria	

Tabla 27. Prueba de aceptación 10

Prueba de Aceptación	
Caso de Prueba: 11	**No. De Historia: 7**

Nombre: Enviar mensaje
Descripción: El sistema debe permitir el envío de mensajes en forma de JSON, al escanear una tarjeta magnética o un código de barras.
Condiciones de Ejecución: Debe existir algún cliente conectado.
Entradas/Pasos de Ejecución: 1- Escanear con algún dispositivo un código de barras o una tarjeta magnética.
Resultado esperado: La información capturada por los lectores se envía al cliente conectado.
Resultado obtenido: El cliente recibió los datos capturados con los dispositivos.
Evaluación de Prueba: Satisfactoria

Tabla 28. Prueba de aceptación 11

Pruebas no satisfactorias:

Prueba de Aceptación	
Caso de Prueba: 1	**No. De Historia: 1**
Nombre: Crear conexión	
Descripción: El sistema debe crear una conexión Web utilizando el protocolo WebSocket que permita el intercambio de mensajes entre el usuario y el sistema, y el acceso a los dispositivos.	
Condiciones de Ejecución:	
Entradas/Pasos de Ejecución: 1- Desde un cliente de prueba intentar una conexión WebSocket.	

Resultado esperado: El sistema debe crear una conexión accesible desde cualquier dirección IP de la subred
Resultado obtenido: El sistema no permitió la conexión
Evaluación de Prueba: No **s**atisfactoria

Tabla 29 Prueba de Aceptación 1

Prueba de Aceptación	
Caso de Prueba: 3	**No. De Historia: 2**
Nombre: Interpretar JSON	
Descripción: El sistema debe interpretar un mensaje recibido mediante el protocolo WebSocket, para imprimirlo en un formato que pueda entender la impresora.	
Condiciones de Ejecución: El mensaje recibido debe ser un JSON	
Entradas/Pasos de Ejecución: 1- Capturar señal de mensaje recibido	
Resultado esperado: El sistema debe interpretar el mensaje recibido y convertirlo en comandos para la impresora	
Resultado obtenido: El mensaje no fue convertido correctamente	
Evaluación de Prueba: No **s**atisfactoria	

Tabla 30 Prueba de aceptación 3

Prueba de Aceptación	
Caso de Prueba: 5	**No. De Historia: 3**
Nombre: Crear interfaces de los dispositivos	
Descripción: El sistema debe utilizar una biblioteca que permita crear las interfaces de los	

dispositivos utilizando Libusb 1.0 y cargar los controladores
Condiciones de Ejecución: Debe estar previamente instalada la biblioteca libusb-1.0.
Entradas/Pasos de Ejecución: 1- Iniciar servidor 2- Conectar dispositivos HID.
Resultado esperado: El sistema debe cargar los manejadores correctamente y crear las interfaces.
Resultado obtenido: Los controladores no se cargaron correctamente
Evaluación de Prueba: No satisfactoria

Tabla 31. Prueba de aceptación 5

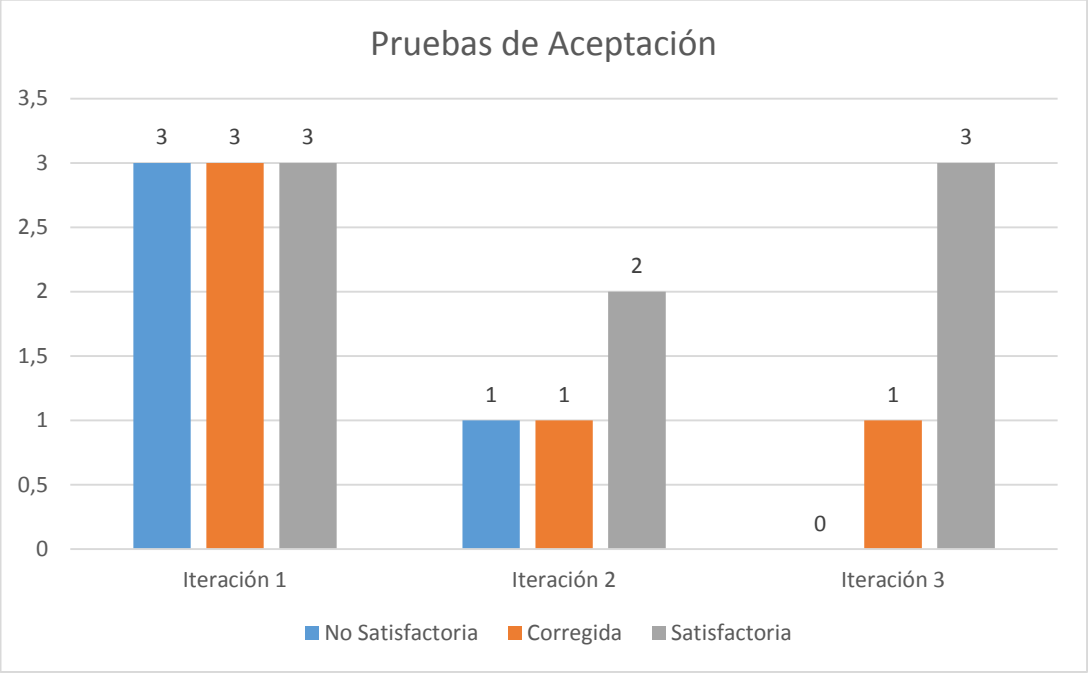
Prueba de Aceptación	
Caso de Prueba: 7	**No. De Historia: 4**
Nombre: Leer código de barras	
Descripción: El sistema debe leer un código de barras y enviarlo a través del protocolo WebSocket al dispositivo del cliente.	
Condiciones de Ejecución: Debe estar conectado el dispositivo de lectura y un cliente conectado.	
Entradas/Pasos de Ejecución: 1-Escanear un código de barras.	
Resultado esperado: El sistema debe capturar el código escaneado y realizar el envío al cliente conectado.	
Resultado obtenido: El cliente de prueba no recibió mensajes.	

Evaluación de Prueba: No satisfactoria

Tabla 32. Prueba de aceptación 7

3.6.1 Resultados de las pruebas de aceptación

Las pruebas correspondientes a las Historias de Usuario se realizaron en cada iteración hasta obtener resultados satisfactorios, en total se realizaron 11 pruebas de aceptación. A continuación se muestra un gráfico con las pruebas por iteración:



Conclusiones parciales

- Basándose en la metodología de desarrollo seleccionada se generaron los artefactos necesarios para describir la estructura y el funcionamiento del sistema.
- Mediante las pruebas realizadas se corrigieron los errores de ejecución de las funcionalidades que arrojaron resultados no satisfactorios.
- Se aportó el procedimiento a seguir para la utilización de la biblioteca libusb-1.0.
- Teniendo en cuenta el resultado de la implementación del sistema, es posible afirmar que se logró desarrollar una biblioteca de manejadores para dispositivos de un Sistema POS que utilicen el estándar USB 2.0, de manera que la conexión de éstos sea *Plug and Play.*

Conclusiones Generales

Después del cumplimiento de las tareas y objetivos planteados se puede concluir que:

- Se obtuvo como resultado de la investigación, mediante el análisis de algunos de los Sistemas POS existentes, que su adquisición representa un gasto adicional al país, no contribuyen a la independencia tecnológica, algunos presentan *hardware* específico o traen los dispositivos integrados, no brindan acceso al código fuente, algunos utilizan interfaces de conexión para periféricos no estandarizadas, y el software POS en algunos sistemas se encuentra embebido sin posibilidad de actualización o modificación por terceros.
- Se demostró la necesidad de implementar una solución cubana para los manejadores de dispositivos de un Sistema POS y una interfaz que provea el acceso a estos desde terceras aplicaciones.
- Con la realización de las pruebas al sistema se logró el correcto funcionamiento del mismo, mostrando un comportamiento aceptable.
- De manera general se puede afirmar que la solución brindada representa una posibilidad para la sustitución de importaciones de Sistemas POS y constituye una fortaleza para la economía nacional.
- La solución propuesta contribuye a la independencia tecnológica del país, al ser implementada completamente con tecnologías libres, por lo que puede ser extendida para soportar diferentes tipos de periféricos, la interfaz de acceso a los manejadores puede ser modificada para soportar otros protocolos de comunicación.

Recomendaciones

A partir de la implementación del sistema y la investigación realizada se recomienda para futuras contribuciones a la solución propuesta:

1. Adicionar funcionalidades para manejar dispositivos de más fabricantes.
2. Incorporar un motor de plantillas para la impresión de los recibos.
3. Implementar la biblioteca de manejadores que funcione asincrónicamente.

Referencias Bibliográficas

1. point of sale Significado en el diccionario Cambridge inglés. [online]. [Accessed 15 December 2016]. Available from: http://dictionary.cambridge.org/es/diccionario/ingles/point-of-sale

2. What is Point of Sale (What is POS)? | POS 101 | Learn. *The Point of Sale News* [online]. [Accessed 14 December 2016]. Available from: https://pointofsale.com/POS-101/What-is-Point-of-Sale-What-is-POS.html

3. Point-of-sale (POS) Terminal | IDT. *Integrated Device Technology* [online]. [Accessed 15 December 2016]. Available from: http://www.idt.com/application/industrial-embedded/point-sale-pos-terminal

4. *Punto de venta* [online]. [Accessed 14 December 2016]. Available from: https://tec-mex.com.mx/punto-de-venta-2/

5. driver | Real Academia de Ingeniería. *Diccionario Español de Ingeniería* [online]. [Accessed 25 January 2017]. Available from: http://diccionario.raing.es/es/lema/driver

6. What is device driver? definition and meaning - BusinessDictionary.com. [online]. [Accessed 7 May 2017]. Available from: http://www.businessdictionary.com/definition/device-driver.html

7. What is interface? definition and meaning - BusinessDictionary.com. *Business Dictionary* [online]. [Accessed 7 May 2017]. Available from: http://www.businessdictionary.com/definition/interface.html

8. Jan Axelson. *USB Complete* [online]. tercera. [no date]. ISBN ISBN13 978-1-931448-03-1 ISBN10 1-931448-03-5. Available from: www.Lvr.com

9. USB - EcuRed. *EcuRed* [online]. [Accessed 25 January 2017]. Available from: https://www.ecured.cu/USB

10. tasven - Software Punto de Venta. *Tasven* [online]. [Accessed 14 December 2016]. Available from: http://www.tasven.com/

11. POS Software - Best Retail Point of Sale Systems | Vend. *vendhq* [online]. [Accessed 15 December 2016]. Available from: https://www.vendhq.com/

12. POS Software Solutions | InfoTouch. *infoTouch* [online]. [Accessed 15 December 2016]. Available from: http://www.infotouch.com/

13. David Blanchard. Introducción a C++ ¿que es? – Blanchard Space. [online]. [Accessed 26 January 2017]. Available from: https://blanchardspace.wordpress.com/2013/05/06/introduccion-a-c-que-es/

14. CRAIG LARMAN. *UML y Patrones 2da Edicion*. [no date].

15. Visual Paradigm para UML. [online]. [Accessed 26 January 2017]. Available from: http://www.software.com.ar/p/visual-paradigm-para-uml

16. libusb. [online]. [Accessed 3 May 2017]. Available from: http://libusb.info/

17. WebSockets Firefoxmania. *Firefoxmania* [online]. Available from: developer.firefoxmania.uci.cu/2014/05/01/WebSockets-en-la-practica-2/

18. Análisis de la tecnología WebSocket – Mario Mori Acosta. [online]. [Accessed 26 June 2017]. Available from: https://mariomoriacosta.wordpress.com/2015/02/22/analisis-de-la-tecnologia-WebSocket-2/

19. Jasmin Blanchette. *C++ GUI Programming with Qt 4* [online]. [no date]. [Accessed 26 January 2017]. Available from: http://my.safaribooksonline.com/0131872494/pref04#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTAxMzE4NzI0OTQlMkZwcmVmMDImcXVlcnk9

20. Qt Creator - EcuRed. [online]. [Accessed 26 January 2017]. Available from: https://www.ecured.cu/Qt_Creator

21. Tamara Rodriguez Sánchez. *Metodología de desarrollo para la Actividad productiva de la UCI.*

22. Definición arquitectura cliente servidor - Monografias.com. [online]. [Accessed 3 May 2017]. Available from: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#estilos

23. EQUIPO DEL PRODUCTO CMMI. *CMMI para Desarrollo ,Versión 1.3* [online]. Available from: http://www.sei.cmu.edu