Universidad de las Ciencias Informáticas

Facultad 4

Título: "Capa de servicios web para la gestión de la información generada en un punto de venta"

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Janderd Pérez Pérez

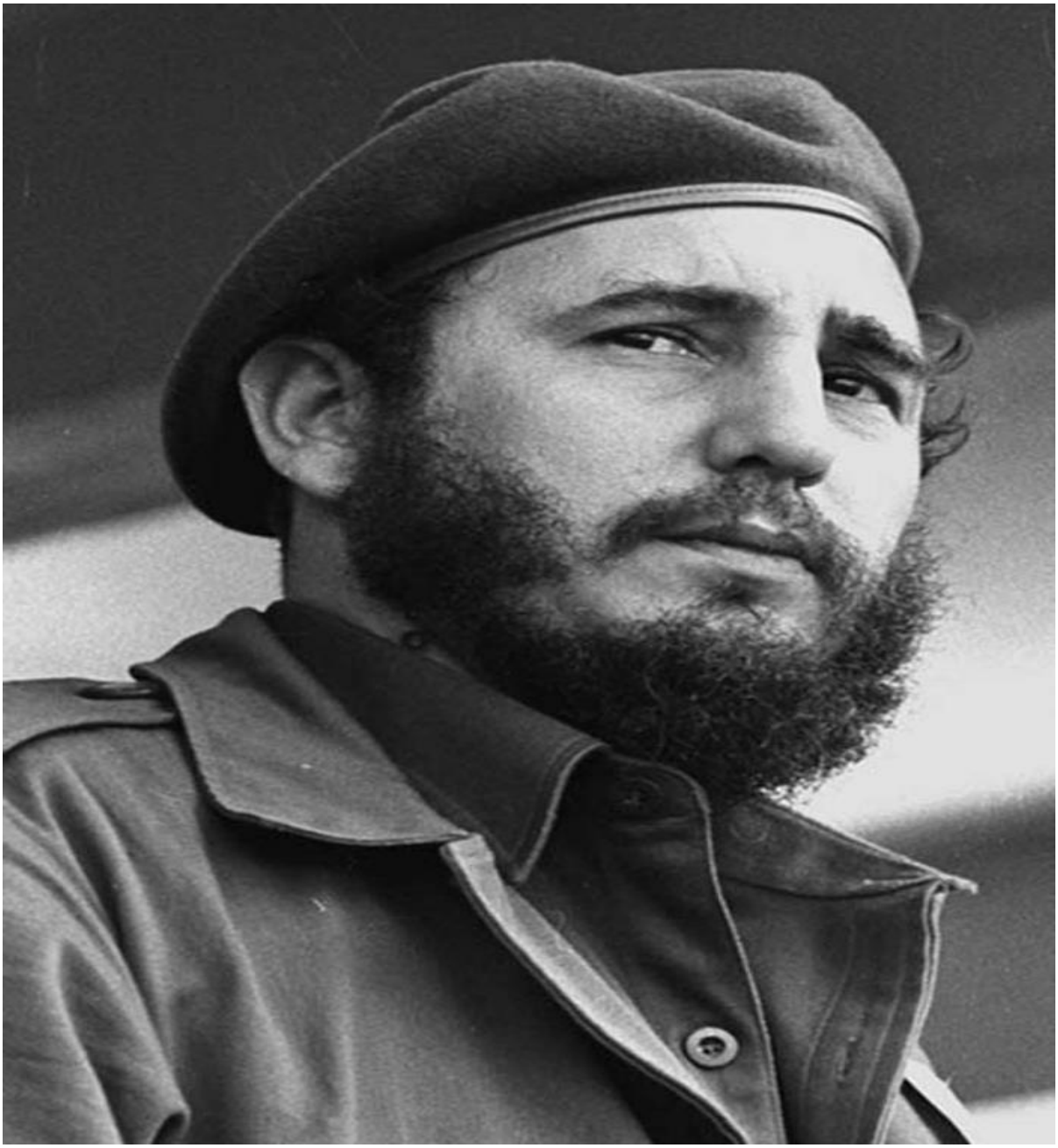
Tutor(es): Ing. Diana Tahirí Galí Ramírez

Ing. Yubismel Perdomo Velázquez

Cotutor: Ing. Frank Geiler Vega Duverger

La Habana, Cuba

2017



¨..... creer en los jóvenes es ver en ellos además de entusiasmo, capacidad; además de energía, responsabilidad; además de juventud, ¡pureza, heroísmo, carácter, voluntad, amor a la patria, fe en la patria! ¡Amor a la Revolución, fe en la Revolución, confianza en sí mismos!, convicción profunda de que la juventud puede, de que la juventud es capaz, convicción profunda de que sobre los hombros de la juventud se pueden depositar grandes tareas."

DECLARACIÓN DE AUTORÍA

Declaro ser el 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 se firma la presente a los _____ días del mes de _____ del año _____.

Firma del Autor: Janderd Pérez Pérez

Firma del Tutor: Diana Tahirí Galí Ramírez

Firma del Tutor: Yubismel Perdomo Velázquez

Firma del Cotutor: Frank Geiler Vega Duverger

DATOS DE CONTACTO

Tutor: Ing. Yubismel Perdomo Velázquez. Profesor graduado de Ingeniero Informático en el año 2006. Ha impartido asignaturas como Introducción a la Programación, Programación II, Programación V, Sistemas de Bases de Datos I, Sistemas de Bases de Datos II. Posee categoría docente de asistente. Dirección de correo electrónico: yubismel@uci.cu

Tutora: Ing. Diana Tahirí Galí Ramírez. Profesora graduada en Ciencias Informáticas en el año 2012. Posee categoría docente de Instructor. Dirección de correo electrónico: dtgali@uci.cu

Cotutor: Ing. Frank Geiler Vega Duverger. Graduado en ingeniería en Ciencias Informáticas en el año 2013. Dirección de correo electrónico: fgvega@uci.cu

AGRADECIMIENTOS

Agradezco a todas aquellas personas que me apoyaron e hicieron posible la realización del presente trabajo.

Agradezco especialmente a mi mamá, mi papá, a mis tías, a mis hermanos, en general a toda mi familia que siempre confiaron en mí, me guiaron por el camino correcto.

A mi novia por tener toda la paciencia del mundo en este tiempo, por su amor y comprensión en todos estos años.

A los tutores Diana, Frank, Yubismel, por su paciencia y dedicación.

Gracias a todos, Mil gracias.

RESUMEN

El propósito de la presente investigación consiste en la realización de una capa de servicios web que permita establecer la comunicación entre un cliente y un servidor web para gestionar la información que se concibe en un punto de venta, generando a su vez consultas a una base de datos.

En el desarrollo del presente trabajo se profundizó en el estudio de los servicios web, y gestores de base de datos utilizados internacionalmente. Una vez que se finiquitó la investigación se procedió a desarrollar las especificaciones del sistema. Para la creación de la capa de servicios se trabaja con el lenguaje de programación PHP, como gestor de base de datos PostgreSQL. Se cuenta con una base de datos para almacenar la información persistentemente para que pueda ser consultada en todo momento, y se hizo uso de los servicios basados en Rest, por su alta flexibilidad a los recursos. La fase de desarrollo y ejecución de la aplicación finalizó con un ciclo de pruebas para la validación del correcto funcionamiento de la capa de servicios web.

Palabras claves: base de datos, comunicación, servicios web, sistemas informáticos.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN **1**

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA **4**

1. Introducción 4

1.1 Punto de Ventas 4

1.2 Terminal de Punto de Ventas 4

1.2.1 Componentes y Tecnologías de una TPV 5

1.2.2 Tipos de TPV 6

1.2 Sistemas de Gestión 7

1.2.1 Sistemas de Gestión utilizados en los PDV 8

1.3 Persistencia de Datos 11

1.3.1 Persistencia 11

1.3.2 Base de Datos 12

1.4 Servicios WEB 14

1.4.1 Características, Ventajas y Desventajas 14

1.4.2 Tecnologías para la comunicación usadas en los Servicios WEB 15

1.4.3 Comparación entre los servicios existentes 16

1.4.4 Tipo de servicios web a utilizar 21

1.5 Sistemas informáticos que poseen TPV 21

1.6 Metodologías de desarrollo de software 25

1.6.1 Selección de la metodología a utilizar 26

1.7 Herramientas y Tecnologías a utilizar 26

1.7.1 Raspberry Pi 27

1.7.2 Servidor web 27
1.7.2 Sistema de Gestor de Base de Datos 27
1.7.3 Lenguaje de Programación 30
1.7.4 Framework 31
1.7.5 Herramienta CASE 32
1.7.6 Entorno de Desarrollo Integrado 32
1.7.7 Herramienta para probar servicios web 33
1.8 Conclusiones Parciales 33
CAPÍTULO 2. ANÁLISIS Y DISEÑO 34
2. Introducción 34
2.1 Descripción general de la propuesta de solución 34
2.2 Funcionalidades del Sistema 35
2.2.1 Requisitos Funcionales 35
2.2.2 Requisitos no Funcionales 36
2.3 Modelo de Datos 37
2.4 Historias de Usuario 38
2.5 Diseño 40
2.5.1 Arquitectura del Software 40
2.5.2 Patrones de Diseño 42
2.6 Conclusiones parciales 43
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA 44
3. Introducción 44
3.1 Modelo de Implementación 44

3.1.2 Diagrama de Despliegue 44

3.2 Estándares de código 44

3.3 Tratamiento de Errores 45

3.4 Pantalla Principal de la Aplicación 46

3.5 Modelo de Pruebas 47

3.5.1 Pruebas Funcionales 48

3.5.2 Pruebas de Rendimiento 52

3.5.3 Pruebas de Seguridad 53

3.6 Conclusiones Parciales 54

CONCLUSIONES 55

RECOMENDACIONES 56

REFERENCIAS BIBLIOGRÁFICAS 57

ÍNDICE DE FIGURAS

Ilustración 1 TPV Dial Up 6

Ilustración 2 TPV LAN 6

Ilustración 3 TPV Inalámbrico 7

Ilustración 4 TPV VPOS Merchant. 7

Ilustración 5 Funcionamiento de las bases de datos dinámicas. 12

Ilustración 6 Funcionamiento de las bases de datos estáticas. 13

Ilustración 7 Funcionamiento de SOAP. (Weerawarana, Curbera, & Leymann, 2005) 17

Ilustración 8 Funcionamiento REST. (Álvarez, 2014) 18

Ilustración 9 Uso general de protocolos de servicio web en aplicaciones. (Programmableweb, 2016) 18

Ilustración 10 Propuesta de Solución. 34

Ilustración 11 Modelo de Datos. 37

Ilustración 12 Patrón arquitectónico MVC. Contenido de cada capa. 41

Ilustración 13 Diagrama de Despliegue. 44

Ilustración 14 Vista del fronted de la capa de servicios web. 46

Ilustración 15 Resultados de las Pruebas Funcionales. 51

ÍNDICE DE TABLAS

Tabla 1 Comparación JSON – XML. (Boci, 2012) 15

Tabla 2 Características, ventajas y desventajas de los Servicios Web REST/SOAP. 19

Tabla 3 Diferencias entre los servicios Web REST/SOAP. 20

Tabla 4 Comparación de los sistemas de gestión de TPV.(Elaboración propia) 24

Tabla 5 Comparación de los SGBD. 28

Tabla 6 HU_1 Obtener todos los productos. 38

Tabla 7 HU_7 Crear nuevo producto. 38

Tabla 8 HU_3 Eliminar una categoría. 38

Tabla 9 HU_17 Autenticar Usuario. 39

Tabla 10 HU_11 Listar productos más vendidos en el día. 39

Tabla 11 Posibles errores que contienen las respuestas de los servicios. 45

Tabla 12 Caso de Prueba: Obtener Productos. 47

Tabla 13 Descripción de variables. Caso de Prueba: Obtener Productos. 47

Tabla 14 Diseño de caso de prueba: Obtener Producto. 47

Tabla 15 Caso de Prueba: Autenticar Usuario. 48

Tabla 16 Descripción de variables. Caso de Prueba: Autenticar usuario. 48

Tabla 17 Diseño de caso de prueba: Autenticar usuario. 48

Tabla 18 No Conformidades por Iteración 49

Tabla 19 Resultados de las pruebas con JMeter. 51

INTRODUCCIÓN

En el proceso de informatización de la sociedad se encuentran involucradas principalmente las denominadas Tecnologías de la Información y las Comunicaciones (TIC), las cuales ocupan un lugar importante en el desarrollo de un país, lo que ha conllevado a una proliferación de la información en cada uno de los procesos vinculados a empresas e industrias. Al aumentar la cantidad de datos almacenados se hace necesaria la búsqueda de alternativas que permitan tener una mejor gestión de los mismos.

Las empresas para garantizar el éxito y la estabilidad de su negocio dependen de la seguridad, control y disponibilidad de sus volúmenes de datos. Una forma de acreditar lo anteriormente planteado es con la existencia de los sistemas informáticos, que permiten almacenar y procesar la información. Gracias a los cambios continuos en la esfera de la informática, específicamente en la realización de sistemas informáticos, las instituciones globales han alcanzado mejorar la eficiencia de sus procesos: productivos, logísticos, de ventas, financieros y de administración; logrando así reducir costos y elevar su competitividad, contribuyendo al aumento progresivo de la economía.

De los sistemas informáticos se emanan las Terminales de Punto de Ventas (TPV) que son sistemas que ayudan en las tareas de gestión de negocio, de inventario y de productos. Diversas instituciones se han visto en la necesidad de informatizar sus distintos puntos de ventas con dichos medios para el buen desarrollo de su negocio.

En Cuba existen empresas y unidades presupuestadas como Havanatur, ETECSA, TRD Caribe, tiendas panamericanas, restaurantes, centros turísticos y tiendas minoristas de Comercio Interno que presentan TPV como parte fundamental de su objeto social. Muchos puntos de ventas no cuentan con TPV para el control de sus ventas, lo que hace que se cometan errores a la hora de anotar las ventas realizadas, provocando desorganización y pérdida de la integridad en la información que se procesa. Existen otras instituciones que poseen aplicaciones informáticas para el control de los procesos de ventas, que en su mayoría son desarrolladas con tecnologías privativas a un costo muy alto para la adquisición de las licencias y mantenimiento del software.

La Universidad de las Ciencias Informáticas (UCI) desde su fundación en el año 2002 ha desempeñado un rol importante en la informatización de todos los campos de la economía del país. En este centro universitario, se encuentra el Centro de Informática Industrial (CEDIN), que se plantea desarrollar un sistema que permita gestionar servicios de ventas, el mismo deberá consultar y generar grandes cantidades de información, donde debido al gran volumen de datos, se deberá almacenar de forma organizada toda la información, y a su vez el sistema deberá adaptarse a diferentes entornos.

A partir de la situación anteriormente expuesta surge el siguiente **problema a resolver:** ¿Cómo garantizar el almacenamiento de datos generados en sistemas de puntos de ventas (POS) y la posterior consulta de los mismos en diferentes escenarios de aplicación?

Se precisó como **objeto de estudio:** Servicios de gestión de información para puntos de ventas.

Enmarcado en el **campo de acción**: Servicios web y persistencia de datos en sistemas de gestión de puntos de ventas.

Definiendo como **objetivo general:** Desarrollar una capa de servicios web capaz de administrar, proveer datos generados, además la interoperabilidad en el sistema para la gestión de servicios de puntos de ventas.

Teniendo en cuenta el objetivo antes planteado se derivan las siguientes **tareas de investigación.**

1. Elaboración del marco teórico a partir del estudio del estado del arte del tema.
2. Valoración sobre la posible reutilización de componentes ya existentes que puedan enriquecer la solución propuesta.
3. Argumentación de tecnologías, metodologías, herramientas de software para el desarrollo de la solución propuesta.
4. Generación los artefactos relacionados con el Análisis y Diseño de la solución propuesta.
5. Análisis, Diseño e implementación del sistema para lograr el objetivo propuesto.
6. Desarrollo de pruebas para garantizar la calidad del sistema.

Los **métodos de investigación científica** que serán utilizados son los siguientes:

- **Teóricos:**
    - **Histórico - Lógico:** Se utilizó para realizar un estado del arte de la problemática analizada con el fin de conocer cómo ha evolucionado la utilización de servicios web hasta la actualidad.
    - **Analítico – Sintético:** Se aplicó al realizar un estudio acerca de los aspectos relacionados con los servicios web y su implementación, mediante el análisis de la documentación existente.

Esta investigación constará de 3 Capítulos, donde se describe el trabajo realizado y los resultados obtenidos en el mismo.

Capítulo 1 Fundamentación Teórica: En este capítulo se describen los conceptos fundamentales que están relacionados con la gestión de punto de venta. También se hace énfasis en el empleo de la metodología, herramientas y tecnologías a utilizar en el trabajo.

Capítulo 2 Análisis y Diseño: En este capítulo se realiza la modelación detallada y construcción de la estructura de la aplicación. Se definen los elementos del diseño, así como el modelo de datos. Además, se realiza el levantamiento de los requisitos funcionales y no funcionales, al mismo tiempo se formalizan los artefactos derivados de la metodología de desarrollo de software que se seleccione.

Capítulo 3 Implementación y Prueba: Se implementa la solución propuesta, dándole cumplimiento a cada uno de los requerimientos pactados con el cliente y se muestran los resultados obtenidos de las pruebas realizadas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1. Introducción

En este capítulo se hace mención a conceptos fundamentales que están relacionados con los procesos que se realizan en un Punto de Venta (PDV), para lograr una mejor comprensión de la investigación. Se realiza un análisis de la gestión de punto de venta en diferentes sistemas informáticos nacionales y extranjeros. Se justifica la utilización de las tecnologías, metodología, técnicas y herramientas definidas para el desarrollo de la investigación.

1.1 Punto de Ventas

El punto de venta (PDV) es el lugar que gestiona el proceso de salida y cobro de la mercancía en tiendas, comercios, restaurantes y otras instituciones, en las que se organiza el proceso de venta de artículos en un mostrador de manera fácil, ágil y cómoda para vendedores y compradores, contando con una caja registradora, un cobrador o cajero y emitiéndose un comprobante de venta como evidencia de la actividad realizada. De poseer un programa para la gestión de la venta facilita la creación e impresión del ticket de venta mediante la información del stock. Para realizar una transacción comercial de compra-venta como el uso de tarjetas bancarias, el PDV debe contar con una terminal de punto de venta. (Nuñez Horta & García Pérez, 2014)

1.2 Terminal de Punto de Ventas

Un Terminal de Punto de Venta puede definirse como el sistema que está compuesto por dos partes fundamentales, la primera es el hardware (dispositivos físicos), como un escáner de código de barras, una impresora de recibos, lectores de banda magnética y monitores. El *software* está compuesto por un programa de gestión de venta de productos o servicios creado específicamente para agilizar los procesos de ventas en los PDV. Dicho sistema se encarga de realizar los procesos de la venta de un centro de ofertas públicas a través de una interfaz accesible para los compradores y vendedores relacionados en la ejecución de la actividad, dígase la captura de los productos en la base de datos, imprimir un *ticket* o gestionar la factura de venta, lectura de la información mediante dispositivos externos para el cobro a través de tarjetas bancarias, emisión de comprobantes de venta, revisión de reportes mensuales, así como llevar el control de inventarios y operaciones comerciales determinadas. (Nuñez Horta & García Pérez, 2014)

Ventajas de una TPV

- Comunicación: Aumentar la comunicación en un negocio puede ayudar a reducir las posibilidades de error cuando se trata de la orden de un cliente.
- Registro y organización de los negocios: Un TPV puede configurar y registrar todas las transacciones financieras de una empresa. La nómina, el inventario, los costos de alimentos

y los artículos más populares del menú pueden organizarse en el sistema de la computadora, permitiendo que los negocios funcionen de forma más eficiente. (Shires, 2016)

Desventajas de una TPV

- Problemas en el sistema: Las TPV, al igual que cualquier sistema de computadora, pueden fallar provocando que las compañías pierdan información guardada.
- Costo: La instalación de una Terminal de Punto de Venta puede ser costosa, con el precio mínimo alrededor de los 4000 USD. (Shires, 2016)

1.2.1 Componentes y Tecnologías de una TPV

Las TPV están compuestas por *hardware* y *software*, los cuales son: (MBCEStore, 2013)

- *Software* de Punto de Venta o *Software* POS: Este es el sistema encargado de realizar todo el proceso de venta desde la captura de los productos en su base de datos, lectura de la información mediante dispositivos externos, emisión de comprobantes de compra, emisión de reportes de ventas mensuales, entre muchas funciones más.

- Escáneres: Es el dispositivo encargado de interpretar la información codificada en un código de barras y transformarla en información que la computadora pueda procesar.

- Impresora de Recibos: Es uno de los componentes indispensables para un punto de venta, es el encargado de emitir los comprobantes de ventas, *boucher* y reportes como los son corte de caja y más.

- Monitor *Touch-Screen*: Dispositivo que permite al usuario interactuar visualmente con la computadora y el *software* en tiempo real, siguiendo procesos necesarios para completar una venta o introducir información al sistema.

- Lector de bandas magnéticas: Este accesorio sirve para convertir la información codificada en una banda magnética y así transformarla en información útil para el sistema.

- Unidad Central de Procesamiento (del inglés: CPU): Es una de las partes centrales pues en este va integrado el sistema operativo, y el sistema punto de venta a utilizar en las ventas. Es compuesto por la tarjeta madre, disco duro, memoria, unidad de cd, opcionalmente algún otro accesorio como lo son las tarjetas de red.

- Lector de código de barras: Generalmente se utilizan los de tipo pistola o semi-fijos en comercios pequeños como tiendas de conveniencia o boutiques, y los fijos empotrados en el

mostrador en supermercados, pudiendo estos últimos incluso contar con una báscula para el pesaje y cobro de productos vendidos a granel. (SEO, 2015)

1.2.2 Tipos de TPV

Existen distintos TPV, como muestra de ellos están los siguientes: (BEExterior, 2016)

- Equipo Dial Up: Funciona con líneas telefónicas que se pueden adaptar a líneas centrales, siempre y cuando existan más de dos líneas telefónicas. Es ideal para clientes que posean de una a dos tarjetas.

Ilustración 1 TPV Dial Up

- Equipo LAN: A diferencia de los Diap Up, estos usan como puente de transmisión el Internet. Ideal para clientes que posean más de dos tarjetas.

Ilustración 2 TPV LAN

- Equipo Inalámbrico: Funciona con líneas telefónicas de móviles. Ideal para clientes que no tengan telefonía fija. Es bastante usado en eventos y ferias.

Ilustración 3 TPV Inalámbrico

- Equipo VPOS Merchant: Son un conjunto de componentes que hacen el funcionamiento de una TPV. Tiene la posibilidad de integrarse con la funcionalidad de su caja registradora mediante el Internet, de manera rápida y segura. Es utilizado en los comercios que poseen un sistema administrativo centralizado y una gran cantidad de cajas registradoras.

Ilustración 4 TPV VPOS Merchant.

Las TPV son considerados sistemas de gestión, ya que los mismos ayudan a los usuarios con las tareas de gestión de negocio, de inventario y de producto.

1.2 Sistemas de Gestión

Sistema de Gestión

Un sistema de gestión permite que una organización mediante una serie de estrategias que incluyen la optimización de los procesos, logre los objetivos principales de su negocio. Es un conjunto de etapas unidas en un proceso continuo, que permite trabajar ordenadamente una idea hasta lograr mejoras. Se establecen cuatro etapas en este proceso, que hacen de este sistema, un proceso circular virtuoso: (Vergara, 2014)

- Etapa de Ideación: El objetivo de esta etapa es trabajar en la idea que guiará los primeros pasos del proceso de creación que se logra con el sistema de gestión propuesto.

- Etapa de Planeación: En esta etapa, se definen las estrategias que se utilizarán, la estructura organizacional que se requiere, el personal que se asigna, el tipo de tecnología que se necesita, el tipo de recursos que se utilizan y la clase de controles que se aplican en todo el proceso.

- Etapa de Implementación: En esta etapa la implementación se entiende por gestión, la acción y efecto de administrar. Pero, en el contexto empresarial, esto se refiere a la dirección que toman las decisiones y las acciones para alcanzar los objetivos trazados.

- Etapa de Control: El control es una función administrativa, esencialmente reguladora, que permite verificar (o también constatar, palpar, medir o evaluar), si el elemento seleccionado, es decir, la actividad, proceso, unidad o sistema está cumpliendo sus objetivos o alcanzando los resultados que se esperan.

El desarrollo de un sistema de gestión proporciona ventajas y desventajas.

Ventajas:

- Control de las actividades de la organización.
- Disponibilidad de información para los usuarios en tiempo real.
- Elimina la barrera de la distancia trabajando con un mismo sistema en puntos distantes.
- Disminuye errores, tiempo y recursos.

Desventajas:

- La resistencia al cambio de los usuarios.

- La inadecuada implementación de las funcionalidades para el apoyo de actividades de organización. (González Guadalupe & León Pérez, 2015)

1.2.1 Sistemas de Gestión utilizados en los PDV

Sistemas de Gestión Empresarial

Es el conjunto de aplicaciones que se utilizan en las empresas para realizar cada uno de los pasos de administración de la misma, desde la producción, pasando por la logística, hasta la entrega del producto en el punto de venta. Con el fin de lograr una eficaz productividad, y debido la importancia que posee el manejo de información en las empresas, se utilizan las herramientas propias de los

sistemas de gestión empresarial, que permiten controlar, planificar, organizar y dirigir cada uno de los eslabones de la cadena productiva.

Debido a la relevancia que posee la información real y a tiempo en las empresas, uno de los aspectos fundamentales en los que se basan los sistemas de gestión empresarial radican puntualmente en dicha información, la cual debe responder a una serie de cuatro puntos básicos para que logre ser útil en el desarrollo de la productividad de cada organización. (INFORMATICAHOY, 2013)

- Los datos utilizados deben ser totalmente verdaderos, y ofrecer un fiel reflejo de la realidad, por lo cual es imprescindible contar con información de calidad.
- Si se habla de calidad también es necesario hablar de cantidad, ya que el personal que se encuentra a cargo de una empresa precisa obtener toda la información necesaria para poder tomar una decisión, por ello a mayor información disponible mejor será el resultado.
- Es útil aquella información de real relevancia, ya que los datos irrelevantes no sólo son innecesarios, sino que también pueden llegar a provocar una mala decisión.
- El cuarto aspecto responde a la premisa de la oportunidad, es decir, que para que las respuestas a las necesidades planteadas lleguen en el momento justo, logrando con esto un verdadero control eficaz de la producción, se debe obtener información precisa en tiempo real. Este hecho permite realizar tomas de decisiones adecuadas, incluso cuando se hace necesario modificar planificaciones anteriores.

Para lograr sus metas estos tipos de sistemas cuando son automatizados deberían presentar las siguientes características: (INFORMATICAHOY, 2013)

- Orientación del negocio.
- Integración con los sistemas existentes.
- Flexibilidad y facilidad de uso.

Ventajas de los Sistemas de Gestión Empresarial.

Un sistema de gestión empresarial automatizado, basa su funcionamiento en datos. Los recoge, los almacena, los procesa y los analiza; las ventajas de utilizar estos sistemas son: (INFORMATICAHOY, 2013)

- Hacer que sea posible almacenar inmensas cantidades de información.

- Evitar los errores típicos de los registros manuales.
- Multiplicar la profundidad del análisis, mediante cálculos y comparaciones que serían prácticamente imposibles en un sistema manual.

La aportación de los sistemas de gestión empresarial es insustituible, sin ellos no sería posible evaluar todas las variables que pueden afectar a la resolución de un problema o tomar en consideración todos los datos que se deben tener en cuenta para una visión completa y de calidad de la realidad.

Sistemas de Gestión de la Información

La gestión de la información es el proceso de analizar y utilizar la información que se ha recopilado y registrado. Permite a los directivos de las empresas crear informes que les proporcione una visión completa de toda la información que necesitan para tomar decisiones que van desde pequeños detalles diarios hasta una estrategia de nivel superior. (Ingram, 2016)

Muchas empresas utilizan de alguna forma un sistema de gestión de información (SGI). Algunas pequeñas empresas hacen esto a través de una base de datos. Otras compran SGI externos e integran estos sistemas con el *software* que utilizan actualmente. El tipo de sistema de gestión de información que una empresa elige depende de cuánto valor puede traer el sistema a la compañía. Entre sus principales características se pueden definir: (Ingram, 2016)

- Administración de Base de Datos: La característica principal de un SGI es su capacidad para almacenar datos y facilitar información para ser recuperada por los usuarios del sistema. El tipo de base de datos utilizada determina cómo el sistema de gestión de información responde a las peticiones o consultas de información.
- Informes: La siguiente característica se basa en la forma de cómo genera informes. La capacidad de producir información que ayuda en el proceso de toma de decisiones es un atributo clave para este tipo de sistema.
- Acceso Abierto: Un SGI que consiente el acceso abierto a la arquitectura del sistema permite a una empresa cumplir más fácilmente con las regulaciones externas y requisitos internos. El acceso abierto significa que la empresa puede integrar más fácilmente el SGI con los sistemas existentes. También reduce los gastos de mantenimiento debido a que los recursos internos pueden gestionar el mantenimiento del sistema.
- Integración: Los sistemas de gestión de información típicamente se integran con los sistemas existentes de la empresa. Un buen SGI proporciona facilidad de integración con sistemas

heredados, permitiendo así que una empresa mantenga las inversiones de equipo que ya ha hecho.

- Escalabilidad: Las empresas más pequeñas pueden requerir una versión reducida de un SGI al momento, pero dentro de unos años necesitan características adicionales y funciones mayores de administración de bases de datos. Comprar un sistema escalable proporciona espacio de crecimiento para la compañía sin que tenga una pérdida de inversión inicial.

Gestionar la información de manera correcta, implica, según la referencia: (Reyez Velázquez & Jiménez Andarcio, 2010)

- Determinar la información que se precisa.
- Recoger y analizar la información.
- Registrarla y recuperarla cuando sea necesaria.

Por lo tanto, un buen SGI ayuda a los usuarios a estar documentados, teniendo siempre datos persistentes, actualizados y disponibles en el momento de la necesidad.

1.3 Persistencia de Datos

1.3.1 Persistencia

Se le llama persistencia a la capacidad de guardar la información de un programa para poder volver a utilizarla en otro momento. Suele involucrar un proceso de serialización de los datos a un archivo o a una base de datos (BD) o a algún otro medio similar, y el proceso inverso de recuperar los datos a partir de la información serializada. (Wachenchauzer & Manterola, 2014)

Tipos de Persistencia de Datos

En la actualidad se consideran varios tipos de persistencias:

- Persistencia en Memoria: La persistencia en memoria es la capacidad de un dato u objeto para seguir existiendo tras determinadas operaciones. La operación más común que se presta a la persistencia en memoria es la asignación.
- Persistencia de Aplicación: Es la capacidad para que los datos sobrevivan a la ejecución del programa que los ha creado. Sin esta capacidad, los datos solo existen en memoria RAM, y se pierden cuando la memoria pierde energía, como cuando se apaga el computador.

- Persistencia de Objetos: En el caso de persistencia de objetos la información que persiste en la mayoría de los casos son los valores que contienen los atributos en ese momento, no necesariamente la funcionalidad que proveen sus métodos. La persistencia de objetos puede ser fácilmente confundida con la persistencia en memoria; incluso con la persistencia de aplicación.

1.3.2 Base de Datos

Es un contenedor que permite almacenar la información de forma ordenada con diferentes propósitos y usos**.** Está constituida por cierto conjunto de datos persistentes utilizados por los sistemas de aplicaciones de una empresa o cualquier organización determinada. Presentan unas series de características tales como seguridad (solo las personas autorizadas tendrán acceso a la información), integridad (la información se mantendrá sin pérdidas de datos), e independencia (esta característica es fundamental ya que una buena base de datos debe ser independiente del sistema operativo o programas que interactúen en ellas). (Sierra, 2006)

Tipos de Base de Datos

- OLTP (*On Line Transaction Processing*): También son llamadas bases de datos dinámicas. Los sistemas OLTP son bases de datos orientadas al procesamiento de transacciones. Una transacción genera un proceso atómico y que puede involucrar operaciones de inserción, modificación y borrado de datos. El proceso transaccional es típico de las bases de datos operacionales. (Sinnexus, 2016)

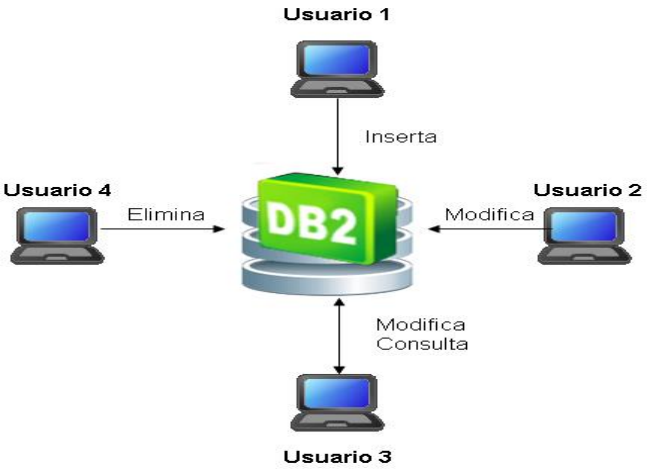


Ilustración 5 Funcionamiento de las bases de datos dinámicas.

- OLAP (*On Line Analytical Processing*): También son llamadas bases de datos estáticas. Los sistemas OLAP son bases de datos orientadas al procesamiento analítico. Este análisis suele implicar, generalmente, la lectura de grandes cantidades de datos para llegar a extraer algún

tipo de información útil: tendencias de ventas, patrones de comportamiento de los consumidores y elaboración de informes complejos. (Sinnexus, 2016)

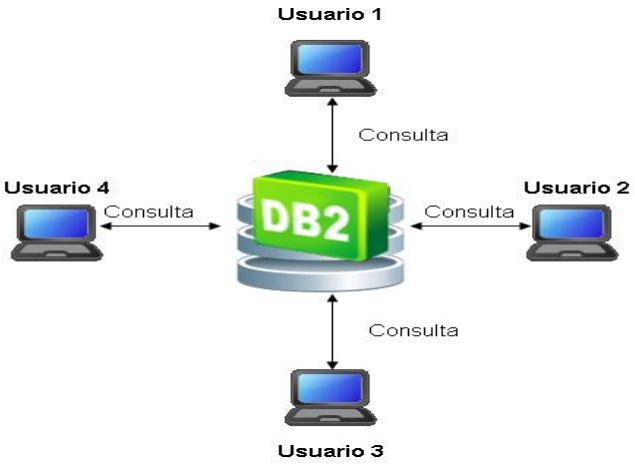


Ilustración 6 Funcionamiento de las bases de datos estáticas.

Ventajas de las Base de Datos

Las bases de datos son de vital importancia en la dinámica del mundo actual. Se verán algunos de sus beneficios:

- Control sobre la redundancia de datos.
- Consistencia de datos.
- Compartición de datos.
- Mantenimiento de estándares.
- Mejora en la integridad de datos.

Desventajas de las Base de Datos

Con todo el potencial que poseen las bases de datos, tienen algunas cosas y aspectos que no siempre las harán estables, se analizarán a continuación.

- Coste del equipamiento adicional.
- Vulnerabilidad a los fallos.

- Los Sistemas Gestores de Base de Datos (SGBD) son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.

- Cuando la BD crece mucho puede llegar a ponerse lenta, lo que afecta las búsquedas y la recuperación de información.

La persistencia en las Base de Datos permite preservar la información de forma permanente, de manera que pueda ser nuevamente utilizada.

1.4 Servicios WEB

La necesidad de que los sistemas informáticos intercambiaran datos entre sí facilitó el surgimiento de los servicios web. Un servicio Web es una tecnología que utiliza diversos estándares y protocolos para intercambiar datos entre aplicaciones. Estas aplicaciones de *software* se pueden programar en distintos lenguajes de programación y acceder a distintas bases de datos a la vez, ya sea desde una red local o desde Internet por medio de computadoras o cualquier otro dispositivo que sea compatible con dicha aplicación. (Castro, y otros, 2013)

Funciones de los servicios web: (W3C, 2014)

- Proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones.

- Proporcionan interoperabilidad y extensibilidad.

- Posibilidad de combinación entre las aplicaciones para realizar operaciones complejas.

- Se pueden utilizar con HTTP (del inglés: *Hypertext Transfer Protocol*) sobre TCP (del inglés: *Transmission Control Protocol*) en el puerto 80, lo que evita que sean bloqueados por los firewalls de la mayoría de los sistemas operativos existentes.

1.4.1 Características, Ventajas y Desventajas

Las principales características de los servicios Web, son las siguientes: (Castro, y otros, 2013)

- Utilización de estándares de internet. La única forma para que los servicios web sean utilizados por la cantidad de sistemas heterogéneos existentes en Internet es el empleo del protocolo de transferencia de datos HTTP utilizado por todos los navegadores Web y el lenguaje de etiquetado XML.

- Basados en tecnologías de paso de mensajes. La interacción entre el cliente y el proveedor del servicio es empaquetada en unidades denominadas mensajes. Dicha interacción se describe en función de los mensajes intercambiados.

- Combinan lo mejor de la tecnología de componentes y de la tecnología web. Los servicios web presentan una funcionalidad de caja negra que puede ser reutilizada sin preocuparse de cómo es implementada y ello proporciona interfaces bien definidas.

Ventajas:

- Permiten que servicios y *software* de diferentes compañías ubicadas en distintos lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Aportan interoperabilidad entre aplicaciones de *software* independientemente de sus propiedades o de las plataformas sobre las que se instalen.

Desventajas:

- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en *firewall* cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

1.4.2 Tecnologías para la comunicación usadas en los Servicios WEB

Es relevante señalar que son los estándares abiertos los que emergen como una clave importante para lograr la interoperabilidad (Martin, 2011). Entre los estándares de intercambio de datos destacan por su nivel de preferencia entre desarrolladores: XML (del inglés: *Extensible Markup Language*) y JSON (del inglés: *Java Script Object Notation*).

En la siguiente tabla se muestra un resumen de las principales características de los formatos de intercambios mencionados anteriormente: (Boci, 2012)

Referencia	JSON	XML
	Tiene una gramática mucho más pequeña que XML y mapea	XML ha sido regularmente criticado por su nivel de detalle y

Simplicidad	directamente sobre las estructuras de datos utilizadas en el lenguaje de programación.	complejidad. El mapeo del modelo de árbol básico de XML hacia los sistemas de tipos de lenguajes de programación o bases de datos puede ser difícil, especialmente cuando se utiliza XML para el intercambio de datos altamente estructurados entre aplicaciones, lo que no era su objetivo primario de diseño.
Extensibilidad	No es extensible, ya que JSON no es un lenguaje de marcado de documentos, por lo que no es necesario definir nuevas etiquetas o atributos para representar los datos en ella.	Si es extensible ya que es un lenguaje de marcado de documentos.
Interoperabilidad	Equivalente entre los dos lenguajes.	Equivalente entre los dos lenguajes.
Código Abierto	Equivalente entre los dos lenguajes.	Equivalente entre los dos lenguajes.
Procesamiento	JSON se procesa más fácil ya que su estructura es más simple.	Los analizadores sintácticos de XML no son tan sencillos de crear debido a que la estructura de los datos es compleja.

Tabla 1 Comparación JSON – XML. (Boci, 2012)

Por lo anteriormente planteado se escoge como lenguaje de intercambio de datos el formato JSON dada su ligereza, fácil procesamiento y su poca extensibilidad para así optimizar el volumen de intercambio de datos entre las aplicaciones.

1.4.3 Comparación entre los servicios existentes

- SOAP (del inglés: *Simple Object Access Protocol*): Es un protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes de computadoras. Permite

utilizar lenguajes de alto nivel para llamar e implementar el servicio web. Es básicamente un paradigma de una sola vía, pero con la ayuda de las aplicaciones se puede llegar a crear patrones más complejos. Está constituido por: un marco que describe el contenido del mensaje e instrucciones de proceso; un conjunto de reglas para representar los tipos de datos definidos; convenciones para representar llamadas a procedimientos remotos y respuestas. (Weerawarana, Curbera, & Leymann, 2005)

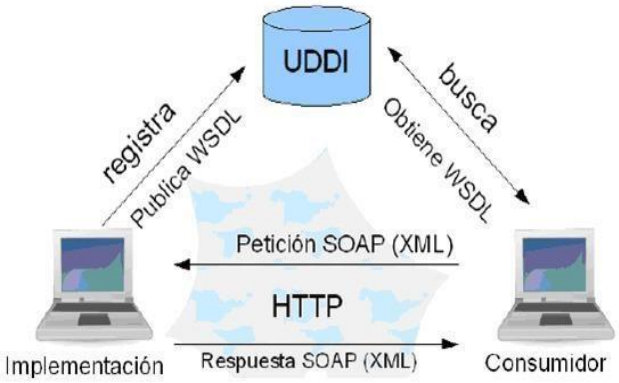


Ilustración 7 Funcionamiento de SOAP. (Weerawarana, Curbera, & Leymann, 2005)

- REST (del inglés: *Representational State Transfer*): Existen varios conceptos que lo definen como:

  - ✓ "Estilo de arquitectura de *software* para sistemas hipermedias distribuidos tales como la *World Wide Web*". (Fielding, 2000)
  - ✓ "Conjunto de principios para el diseño de redes, utilizado comúnmente para definir una interfaz de transmisión sobre HTTP". (Machuca, 2010)

  En resumen, REST es considerado un conjunto de principios de arquitectura que usa directamente el protocolo HTTP(s) (del inglés: *Hypertext Transfer Protocol Secure*) para obtener datos o indicar la ejecución de operaciones sobre los datos en cualquier formato (JSON, XML). Los principios de REST son: (Álvarez, 2014)

  - ✓ *Stateless* o sin estado: Se refiere a que cada mensaje que viaja a través de HTTP(s) lleva toda la información necesaria para realizar una petición, es decir, que el servidor no guarda datos referentes a otras comunicaciones con el cliente (*cookies* o sesiones).
  - ✓ Utilizar las operaciones o verbos ya definidos por HTTP: *POST, GET, PUT y DELETE*.
  - ✓ Utilización de URL (del inglés: *Uniform Resource Identifier*): Se busca la utilización de URL para identificar de manera precisa un recurso a través de red.

- ✓ Hipermedias: El concepto básicamente es que los resultados de los llamados a un sistema basado en REST permitan navegar a otros recursos sin necesidad de hacer alguna transformación extra.
- ✓ Escalabilidad: Se puede crecer todo lo que se necesite en cualquier momento. La API REST puede responder a otros tipos de operaciones o puede versionarse tanto como se desee.
- ✓ Fiabilidad: Solo hay que tener en cuenta que el nexo cliente/servidor esté correcto. Se pueden hacer cambios en el servidor, lenguajes, bases de datos, etc. y mientras devuelvas los datos que espera el cliente todo funcionará correctamente.

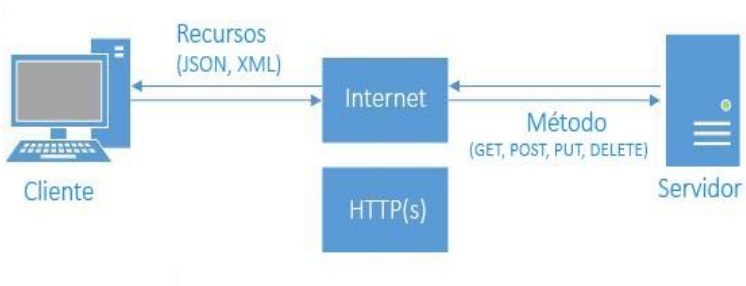


Ilustración 8 Funcionamiento REST. (Álvarez, 2014)

La mayor competencia se centra actualmente entre *REST* y *SOAP* por ser considerados los estilos arquitectónicos de comunicación más usados. La siguiente imagen tomada del directorio de servicios web muestra el nivel de uso general de los principales protocolos de mensajería para el desarrollo de servicios web a nivel global.

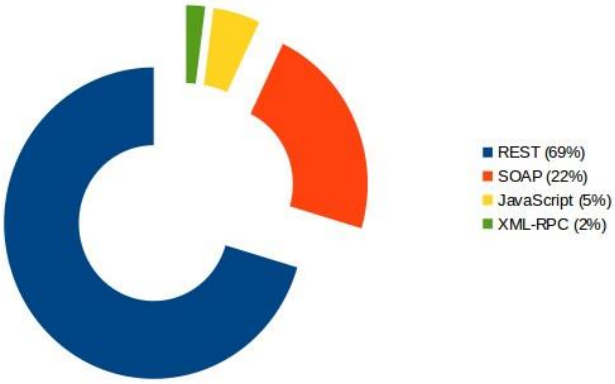


Ilustración 9 Uso general de protocolos de servicio web en aplicaciones. (Programmableweb, 2016)

El principal beneficio de SOAP recae en que es fuertemente acoplado, lo que permite poder ser testado y depurado antes de poner en marcha la aplicación. En cambio, las ventajas de los servicios basados en REST recaen en la potencial escalabilidad de este tipo de sistemas, así como el acceso con escaso consumo de recursos a sus operaciones. A modo de resumen, se mencionan las características de ambos servicios en la siguiente tabla: (Marset, Rest vs Web Services, 2007)

	REST	SOAP
Características	**1.** Las operaciones se definen en los mensajes. **2.** Una dirección única para cada instancia del proceso. **3.** Cada objeto soporta las operaciones estándares definidas. **4.** Componentes débilmente acoplados	**1.** Las operaciones son definidas como puertos WSDL (*Web Services Description Language)*. **2.** Dirección única para todas las operaciones. **3.** Múltiples instancias del proceso comparten la misma operación. **4.** Componentes fuertemente acoplados.
Ventajas	**1.** Bajo consumo de recursos. **2.** Las instancias del proceso son creadas explícitamente. **3.** El cliente no necesita información de enrutamiento a partir de la URI inicial**.** **4.** Los clientes pueden tener una interfaz "listener" (escuchadora) genérica para las notificaciones. **5.** Generalmente fácil de construir y adoptar. **6.** Visibilidad, escalabilidad y rendimiento.	**1.** Fácil (generalmente) de utilizar. **2.** La depuración es posible. **3.** Las operaciones complejas pueden ser escondidas detrás de una fachada. **4.** Envolver APIs existentes es sencillo. **5.** Incrementa la privacidad. **6.** Herramientas de desarrollo.

Desventajas	**1.** Gran número de objetos. **2**. Manejar el espacio de nombres (URIs) puede ser engorroso. **3.** La descripción sintáctica/semántica muy informal (orientada al usuario). **4.** Pocas herramientas de desarrollo.	**1.** Los clientes necesitan saber las operaciones y su semántica antes del uso. **2.** Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. **3.** Las instancias del proceso son creadas implícitamente.

Tabla 2 Características, ventajas y desventajas de los Servicios Web REST/SOAP.

Las principales diferencias entre REST y SOAP desde varios puntos de vistas:

	REST	**SOAP**
Tecnología	**1.** Interacción dirigida por el usuario por medio de formularios. **2.** Pocas operaciones con muchos recursos. **3.** Mecanismo consistente de nombrado de recursos (URL). **4.** Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	**1.** Flujo de eventos orquestados. **2.** Muchas operaciones con pocos recursos. **3.** Falta de un mecanismo de nombrado. **4.** Se centra en el diseño de aplicaciones distribuidas.
Seguridad	**1.** HTTP(s). **2.** Implementado desde hace muchos años. **3.** Comunicación punto a punto segura.	**1.** WS-Security (Seguridad en Servicios Web). **2.** Las implementaciones están comenzando a aparecer ahora. **3.** Comunicación origen a destino segura.
	1. Identificar recursos a ser expuestos como servicios.	**1.** Listar las operaciones del servicio en el documento WSDL.

Metodología de diseño	**2.** Definir URLs para direccionarlos. **3.** Distinguir los recursos de solo lectura (GET) de los modificables (POST, PUT, DELETE). **4.** Implementar e implantar el servidor web.	**2.** Definir un modelo de datos para el contenido de los mensajes. **3.** Elegir un protocolo de transporte apropiado y definir las correspondientes políticas QoS (Calidad de Servicios), de seguridad y transaccional. **4.** Implementar e implantar el contenedor del servicio Web.

Tabla 3 Diferencias entre los servicios Web REST/SOAP.

WSDL (del inglés: *Web Services Description Language*): Es un tipo de documento XML que describe lo que hace un servicio web, donde se encuentra y la forma de ser invocado. Este provee información muy importante para los desarrolladores, este lenguaje describe el formato de los mensajes que utiliza y a cuáles puede responder. (Snell, Tidwell, & Kulchenko, 2002)

1.4.4 Tipo de servicios web a utilizar

Después de haber analizado las características, ventajas, desventajas, tecnologías y seguridad de los tipos de servicios web anteriores se decide usar REST, debido a que los sistemas basados en este estilo ofrecen un bajo uso de recursos, alta escalabilidad y rendimiento, así como bajo número de operaciones con muchos recursos.

1.5 Sistemas informáticos que poseen TPV

Numerosas empresas tienen sistemas para la gestión de ventas, facilitando el manejo de toda la información relacionada. Estos sistemas tienen características en común, como son: (Nuñez Horta & García Pérez, 2014)

- Se adaptan a diversos tipos de establecimientos.
- Las gestiones se almacenan en la base de datos del sistema.
- La facturación de las salidas.

Golden.NeT

Golden.NET es una aplicación diseñada para adaptarse a cualquier tipo de empresa con tiempos y costos de implantación reducidos, también para el control de los inventarios, los cobros y pagos en

las Bases Centrales de Almacenes (BCA) o distribuidoras de mercancías, así como para un PV (tiendas, quioscos, uniones de quioscos). (GoldenSoft, 2013)

Terminal Punto de Venta en Golden.NET:

El software Terminal Punto de Venta para Comercios es un módulo integrado con Gestión Comercial Golden.NET y ha sido ideado para un amplio abanico de negocios: tiendas de ropa, perfumerías, librerías, zapaterías. Adaptado tanto a monitores con pantalla táctil como con teclado y conexión a los distintos periféricos: lector de código de barras, visor, impresora de *tickets*. El software terminal punto de venta para hostelería es un módulo integrado con Gestión Comercial Golden.NET y ha sido especialmente diseñado para la gestión de bares y negocios de hostelería. Se distingue por una entrada ágil de tickets adaptado a monitores con pantalla táctil y elección de los pedidos seleccionando la imagen del artículo. El usuario podrá configurar la ubicación de salones, terrazas, número de comensales por mesa y si la mesa está libre u ocupada.

Además, la aplicación conserva la orden del cliente en el cambio de la barra a una mesa**.** TPV Hostelería y el TPV Comercial de Golden.NET está desarrollado para sistemas de arquitectura de 64 y de 32 bits, confiriéndole como una de las aplicaciones tecnológicamente más avanzadas en el mercado. Esta aplicación permite de manera operativa efectuar las ventas en cadenas de tiendas, el mismo está constituido por diferentes menús y opciones que posibilitan una correcta ejecución, pero dentro de sus desventajas está que necesita una versión actualizada del programa que se utiliza para los reportes comerciales a nivel de División y Cadena y de manera respectiva sus dos tipos de reportes comerciales, los personalizados y los temporales, ya que para el caso de los personalizados el sistema guarda los grupos de estudios de las diferentes dimensiones que un usuario utiliza, de manera que sólo tiene que personalizar una vez el sistema, y periódicamente ir variándolo según sus comodidades, por lo que al no estar actualizado imposibilita la eficiencia de la aplicación. (GoldenSoft, 2013)

Odoo

Conocido anteriormente como OpenERP es un sistema de planificación de recursos empresariales que cubre las necesidades de las áreas de contabilidad, ventas, compras, almacén e inventario de una empresa. Soporta múltiples monedas, compañías y contabilidades; además incorpora funcionalidades de gestión de documentos para agilizar la colaboración entre departamentos y equipos en la empresa; y permite trabajar remotamente mediante una interfaz web desde una computadora conectada a Internet. Tiene componentes separados en esquema cliente-servidor. Dispone de interfaces XML-RPC (*Extensible Markup Language Remote Procedure Call*) (protocolo

de llamada a procedimiento remoto) y SOAP (*Simple Object Access Protocol*) (Protocolo de Acceso de simple Objeto). (Odoo, 2016)

Terminal Punto de Venta en Odoo:

Permite gestionar las ventas de una forma muy sencilla, está basado en tecnología web por lo que no se necesita instalar ningún *software* y todas las tiendas de venta pueden ser fácilmente consolidadas. Permite trabajar con y sin conexión por lo que se puede seguir vendiendo, aunque se pierda la conexión a Internet. Dentro de las distintas opciones para seleccionar productos, posee un lector de código de barras, también logra realizar búsqueda de texto en caso que las otras funcionalidades no estén disponibles. Se puede realizar en el TPV múltiples *tickets* sin tener que esperar a realizar una transacción al mismo tiempo. Así mismo, para facilitar los pagos, la aplicación permite múltiples métodos de pago. El TPV es simple y accesible para el uso en tiendas o restaurantes. El objetivo de ello es una solución empaquetada para mejorar la productividad de los negocios de ventas en las distintas empresas de una forma ágil y segura para el control de las ventas realizadas en los TPV. (Odoo, 2016)

OpenBravo

OpenBravo es un sistema de gestión empresarial en software libre, ha sido específicamente diseñado para ayudar a las empresas a mejorar su rendimiento. La cobertura funcional del producto incluye todas las áreas típicas de un sistema de gestión integrado, destacando la solución de PV. Este sistema posee entre sus módulos: facturación, cobros y pagos, contabilidad, estadísticas, productos, recursos humanos, control de inventario, gestión de atención a clientes y proveedores, gestión de compras, gestión de almacenes, *workflow*[1] de procesos, gestión de proyectos, planificación de proyectos, gestión de producción/fabricación, gestión de ventas, facturación, gestión de informes, gestor documental y una terminal de punto de venta, logrando con ello simplificar y automatizar los procesos empresariales de gestión de entidades.

Dentro de sus principales ventajas está su cobertura funcional que incluye la gestión financiera donde enmarca la gestión de caja. Se puede destacar que permite gestionar varias cajas y soporta múltiples monedas. Además, esta aplicación se integra de manera natural con otras áreas como la gestión de relaciones con clientes, inteligencia de negocio y TPV. (Openbravo, 2016)

Terminal de Punto de Venta de OpenBravo:

Se manifiestan las siguientes características:

[1] **Workflow: flujo de trabajo.**

- Zonas de ventas.
- Pedidos de ventas.
- Auto-venta.
- Preventa.
- Tele-venta.
- Reserva de género en almacén para pedidos no servidos.
- Corrección de pedidos.

Donde cada una de estas quedan estrechamente relacionada con el módulo sistema de punto de venta denominado Openbravo WEB POS, en el cual se realizan cada una de ellas de forma ágil, flexible y autónoma.

A continuación, la tabla comparativa:

Sistemas	Plataforma Tecnológica	Licencia	Sistema Operativo	Soporte y Actualización
Golden.NET	-Gestor de BD: SQL Server Compact 3.5. -Framework.net: 3.5 -JavaPos	Privada	Windows	Si
Odoo	-OpenERP Server 6.0.3 -OpenERP web 6.0.3 -Gestor de BD: PostgreSQL	*License* AGPL, Odoo Enterprise *Edition License* v1.0	Windows, MacOSX, Linux, Unix, Android	Si

	-Lenguaje: Python			
OpenBravo	-*Server*: Apache y Tomcat -BD: PostgreSQL y Oracle -Lenguaje: JavaPos	Openbravo *Public License*,GNU (GPL)	Windows, Linux, Unix, Solaris, freeBSD	Si

Tabla 4 Comparación de los sistemas de gestión de PDV. (Elaboración propia)

En el análisis a los sistemas similares anteriormente comparados se resume que:

- Golden es uno de los sistemas que atiende la planificación de recursos empresariales, el mismo no se ajusta a las normativas del Ministerio de Finanzas y Precio, además su licencia es privada, como también la adquisición de su gestor de base de datos.
- Las TPV Odoo y Openbravo son sistemas multientidades, cuentan con funcionalidades como el uso de servicios web para la interoperabilidad de sus módulos TPV, pero según la bibliografía consultada no aparecen referencias de su explotación en las entidades cubanas, y a pesar de que ambos tienen licencia GPL, no se puede pagar por su soporte y actualización por su elevado costo.
- Las TPV Golden y OpenBravo fueron desarrolladas con tecnología JavaPOS (abreviatura de Java para dispositivos de punto de venta), que es un estándar para interfaz de POS escrito en Java. Dicha tecnología no se encuentra explotada en entidades de nuestro país.

Con el estudio de los sistemas expuestos anteriormente, se concluye que estos programas informáticos no constituyen una elección factible, pues algunos representan gastos muy excesivos a instituciones del país, ya que algunos fueron desarrollados con tecnologías privativas principalmente en el uso de sus gestores de base datos, sin embargo, se tendrán en cuenta algunas de sus funcionalidades para el desarrollo del sistema de gestión de venta a implementar en el CEDIN.

1.6 Metodologías de desarrollo de software

La metodología de desarrollo de *software* se encarga de estructurar, planificar y controlar el proceso de desarrollo del *software* a través de un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a la realización del mismo.

Las metodologías de desarrollo de software surgieron por la necesidad de la industria del software de guiar dicho proceso y eliminar el caos que existía a la hora de creación del mismo. Existen metodologías tradicionales y ágiles, las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto, mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y mantener una relación fuerte con el cliente. (Rodríguez & Almaguer Guerra, 2016)

1.6.1 Selección de la metodología a utilizar

PROCESO UNIFICADO ÁGIL (AUP)

Es una versión simplificada de RUP. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (del inglés: *Rational Unified Process*). El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (*test driven development* - TDD), Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad. Está basada en disciplinas entregables e incrementales con el tiempo, las cuales son: modelado, implementación, prueba, despliegue, administración de la configuración, administración o gerencia de proyecto y entorno. (Rodríguez & Almaguer Guerra, 2016)

AUP-UCI

Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. (Aguilar & Maure Morejón, 2013)

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, llamada Ejecución y se agrega la fase de Cierre.

1.7 Herramientas y Tecnologías a utilizar

Existe un gran avance tecnológico en el mundo donde van apareciendo nuevas tecnologías y herramientas, con el fin de hacer más reales las soluciones de los clientes. Para lograr un sistema acorde a lo esperado, luego de un gran análisis y basándose en la facilidad de uso, documentación existente y las políticas de migración de *software* libre que se desarrolla en el país, se analizan las siguientes tecnologías y herramientas que a continuación serán detalladas:

1.7.1 Raspberry Pi

La Raspberry Pi es un micro ordenador o una placa de computadora SBC[2] de bajo costo desarrollada en el Reino Unido por la fundación Raspberry Pi. Algunos modelos poseen un micro procesador ARM con potencia de hasta 1GHz, integrado en un chip Broadcom BCM2835, todo lo necesario para poder ejecutar programas básicos, navegar por internet y por supuesto programar. Para trabajar con un Raspberry Pi se requiere almacenamiento que en este caso específico debe ser una tarjeta de memoria SD o microSD. Entre los sistemas operativos disponibles para Raspberry Pi se encuentran: Raspbian, Arch Linux, RaspBMC, Pidora u OpenELEC. (ComputerHoy, 2016)

1.7.2 Servidor web

Apache: en su versión 2.2, es un servidor W*eb* de código abierto, altamente configurable y modular. Utiliza Perl, PHP y otros lenguajes scripts. Su función principal es analizar cualquier archivo solicitado por un navegador y mostrar resultados correctos de acuerdo con el código del archivo. Permite configurar los informes de errores, presenta visualización de códigos en numerosos niveles y la capacidad de determinar qué nivel del navegador puede aceptar el contenido. Es uno de los primeros servidores en soportar hosts basados en direcciones IP (por sus siglas en inglés *Internet Protocol*, Protocolo de Internet) y hosts virtuales. Tiene un elaborado índice de directorios, un directorio de alias, informe de errores HTTP y HTML (por sus siglas en inglés *Hypertext Transfer Protocol*, Protocolo de Transferencia de Hipertexto) configurable, gestión de recursos para procesos hijos, reescritura de las URL (por sus siglas en inglés *Uniform Resource Locator*, Localizador de Recurso Uniforme), comprobación de ortografía de las URL y manuales *online.* (Cabrera & Ricardo González, 2015)

1.7.2 Sistema de Gestor de Base de Datos

Microsoft SQL Server: Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ (potente motor de bases de datos), capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. (González, Junio, 2009)

PostgreSQL: Es un sistema gestor de bases de datos relacionales orientadas a objetos con cerca de una década de desarrollo. Soporta casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto

[2] ***Single Board Computer*** **(ordenador de placa reducida)**

de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl, Python, entre otros). (González, Junio, 2009)

MySQL: Es uno de los gestores de bases de datos, más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. (González, Junio, 2009)

Oracle: Es un sistema de gestión de base de datos relacional, es básicamente una herramienta que utiliza arquitectura cliente-servidor, fabricado por Oracle Corporation. Está considerado como uno de los SGBD más completos, destacando sus soportes de transacciones, estabilidad, escalabilidad y que es un sistema multiplataforma.

A continuación, se muestra una información general de los SGBD expuestos:

GESTORES DE BASE DE DATOS	LICENCIA	SIST. OPERAT	VENTAJAS	DESVENTAJAS
Oracle	Privada	Windows, Mac OS X, Linux, Unix	1. Motor de BD relacional más usado del mundo. 2. El software del servidor puede ejecutarse en diferentes SO. 3. Más orientación hacia Internet.	1. Es el SGBD más caro. 2. El costo de mantenimiento es muy alto.
	Privada	Windows	1. Permite la adición de funciones al lenguaje SQL por los mismos usuarios. 2. Lleva a cabo la creación, administración y distribución de las	1. Las funciones definidas por los usuarios tienen algunas restricciones. 2. No todas las sentencias son

Microsoft SLQ Server			aplicaciones de la forma más sencilla. 3. Es ideal para sistemas de almacenamiento de datos y OLTP de la gama alta.	válidas dentro de una función. 3. Es un software propietario.
MySQL	GPL o propietario	Windows, Mac OS X, Linux, Unix	1. Acceso a la base de datos de forma simultánea por varios usuarios y/o aplicaciones. 2. Seguridad, en forma de permisos y privilegios. 3. Potencia, portabilidad, escalabilidad, conectividad.	1. No es viable para su uso con grandes bases de datos. 2. Carece de soporte para transacciones, *rollback´s* y subconsultas.
PostgreSQL	GPL	Windows, Mac OS X, Linux, Unix	1. Posee una gran escalabilidad 2. Es muy extensible ya que soporta distintos tipos de bases de datos. 3. Lenguajes procedurales.	1. Consume muchos recursos y carga con mucha facilidad el sistema. 2. Velocidad de respuesta un poco deficiente al gestionar base de datos pequeñas.

Tabla 5 Comparación de los SGBD.

Sistema de Gestor de Base de Datos a utilizar

Se escoge PostgreSQL en su versión 9.3 por brindar las siguientes características para el desarrollo del sistema de gestión de información para un punto de venta:

- Aislamiento: Asegura que una operación no puede afectar a otras.
- Durabilidad: Asegura que, una vez realizada la operación, esta persistirá y no se podrá deshacer.
- Documentación: Posee una vasta documentación bien organizada, pública y libre.
- Es un potente sistema de gestión de bases de datos objeto-relacional liberado bajo licencia BSD (*Berkeley Software Distribution*).
- Posee una arquitectura probada que garantiza una elevada confiabilidad e integridad en los datos, está basado en una arquitectura cliente-servidor.
- Es multiplataforma. (PostgreSQL, 2013)

1.7.3 Lenguaje de Programación

PHP 5

PHP (*Hypertext Pre-Processor*) es un lenguaje interpretado de alto nivel que permite intercalar las sentencias en las páginas HTML y ejecutado en el servidor, se utiliza para crear sitios Web de características dinámicas. Una de las características más potente y destacable de PHP es su soporte para una gran cantidad de manejadores de bases de datos. (Programmableweb, 2016)

Las ventajas son:

- Es un lenguaje multiplataforma.
- Tiene capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Es capaz de leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Tiene la capacidad de expandir su potencial debido a su enorme cantidad de módulos.
- Posee una amplia documentación en su Web oficial de Internet. En la misma se encuentran muy bien explicadas todas las funciones del sistema, contando con ejemplos detallados.
- Pertenece a la alternativa de código abierto (*Open Source*), por lo que se presenta como una elección de fácil acceso para todos los desarrolladores.

- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la Web.
- Cuenta con una biblioteca sumamente amplia por defecto de funciones.
- No requiere definición de tipos de variables ni manejo detallado de bajo nivel.

¿Por qué utilizar PHP?

Se escogió PHP para la implementación de la aplicación ya que pertenece al grupo de software libre y open source, es un lenguaje multiplataforma, de fácil manejo y aprendizaje porque utiliza instrucciones sencillas. Posee librerías de funciones incorporadas, que permiten realizar cualquier tipo de operación, como el fácil manejo de bases de datos. Debido a sus características hace posible que el cliente interactúe con una página rápida, eficiente y segura, capaz de mostrar y procesar información. Es un lenguaje bastante utilizado por lo que se puede encontrar mucha documentación sobre él.

1.7.4 Framework

Symfony 2.7

Symfony es un *Framework* de desarrollo escrito en PHP, para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Adopta buenas ideas de otros proyectos, reutilizando el código existente cuando está disponible. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, utilizando el patrón arquitectónico modelo-vista-controlador, además de apoyarse en otros patrones de diseño. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL *Server* de *Microsoft* y puede ser ejecutado tanto en plataformas Unix y Linux como en plataformas *Windows*. (Symfony, 2017)

Características de Symfony: (Symfony, 2017)

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la Web.

- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.

1.7.5 Herramienta CASE

Visual Paradigm

Es una herramienta CASE para el desarrollo de aplicaciones utilizando modelado UML (Lenguaje Unificado de Modelado) que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una rápida construcción de aplicaciones de calidad y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación. (Paradigm, 2013)

En este caso Visual Paradigm, se presenta como la mejor opción para el modelado del problema teniendo en cuenta que es una herramienta en que el diseñador posee cierta experiencia previa. Además, ofrece sencillez y claridad en su trabajo.

1.7.6 Entorno de Desarrollo Integrado

PHP Storm

Es un IDE de programación desarrollado por Jetbrains. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación PHP como lo indica su nombre. Actualmente es compatible con sistemas operativos Windows, Linux y Mac OS X. (Código, 2014)

Estas son sólo algunas de las características principales: (Código, 2014)

- Permite la gestión de proyectos fácilmente.

- Proporciona un fácil autocompletado de código.

- Soporta el trabajo con PHP 5.5.

- Sintaxis abreviada.

1.7.7 Herramienta para probar servicios web

JMeter 2.8

Es una herramienta de código abierto creada por ASF (del inglés: *Apache Software Foundation*). Originalmente fue diseñada para probar aplicaciones web, pero se ha extendido a otras funciones de prueba. La función básica de JMeter es cargar pruebas de aplicaciones cliente/servidor, pero también se puede utilizar para la medición del rendimiento. Además, JMeter es también útil en las pruebas de regresión, facilitando la creación de *scripts* de prueba y comprobando que los cambios realizados no afecten los resultados esperados. Soporta multi-hilos lo que permite lograr concurrencia a la hora de hacer peticiones simultáneas. JMeter ofrece una alta extensibilidad debido al uso de componentes acoplables y ofrece al usuario una amigable interfaz gráfica (GUI). La configuración y creación de planes de pruebas requiere poco esfuerzo y ofrece una serie de informes estadísticos, así como análisis gráfico. (Rodríguez & Almaguer Guerra, 2016)

1.8 Conclusiones Parciales

- A través del análisis de los principales conceptos se logró elaborar una panorámica en cuanto al objeto de estudio de la investigación donde se observa principalmente la estructura de los servicios Web y los elementos que lo componen.

- Los servicios Web REST son la forma escogida para lograr la interoperabilidad en el sistema de gestión de ventas a desarrollar por el CEDIN.

- Se eligió AUP-UCI como metodología para garantizar la completitud y calidad del *software* a la hora de guiar eficientemente no solo el proceso de desarrollo de *software*, sino también la organización, gestión, configuración, cambios y el soporte de la capa de servicios web.

- El análisis de las principales características de las herramientas y tecnologías adoptadas, enfocadas en las ventajas y facilidades de cada una de estas, permitió crear un marco de trabajo adecuado.

CAPÍTULO 2. ANÁLISIS Y DISEÑO

2. Introducción

En el capítulo se exponen las principales características del sistema a implementar. Se define el modelo de datos, se describen las historias de usuarios asociadas a las funcionalidades y los requisitos no funcionales relacionados con la propuesta de solución. También se describe la arquitectura por la cual se regirá la construcción de la solución, los patrones de diseño y los artefactos generados.

2.1 Descripción general de la propuesta de solución

Se accederá a través de la Wi-Fi a una tarjeta SBC (*Single Board Computer*) llamada Raspberry Pi desde cualquier dispositivo móvil que se encuentre en el alcance de la red Wi-Fi de dicha tarjeta; la cual poseerá un Servidor Rest que brindará los datos de los productos, categorías y servicios que se ofertan en el sistema, este es el encargado de manejar la información resultante de las ventas.

La capa de servicios web estará enfocada en permitirle a las aplicaciones que interactúen con ella, realizar servicios de comunicación entre un cliente y el servidor web, a través de peticiones REST mediante el protocolo HTTP, ofreciendo como resultado un fichero JSON con la respuesta de la petición, va estar empalmado en una Raspberry Pi, haciendo uso de un servicio Rest que permita actualizar, guardar y modificar la información que se genere.

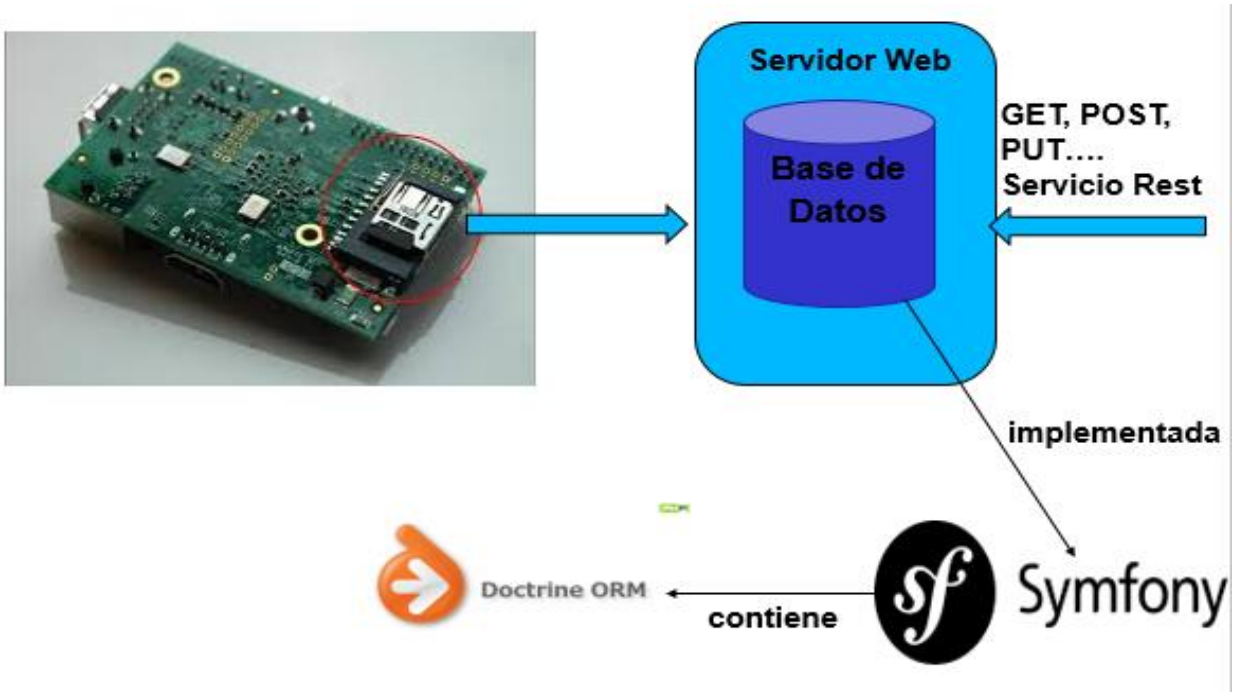


Ilustración 10 Propuesta de Solución.

2.2 Funcionalidades del Sistema

Los requisitos del sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos suelen clasificarse en requisitos funcionales o no funcionales. Los funcionales declaran los servicios que debe brindar el sistema, la manera que éste debe reaccionar y funcionar ante una entrada o situación en particular y los no funcionales son las restricciones de los servicios o funciones ofrecidas por el sistema. (Pressman, 2012)

2.2.1 Requisitos Funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema de manera que este reaccione a entidades particulares y de cómo se debe comportar en situaciones específicas. (Rodríguez & Almaguer Guerra, 2016)

RF1. Obtener todas las categorías.

RF2. Crear nueva categoría.

RF3. Eliminar una categoría solicitada por el cliente.

RF4. Obtener una categoría solicitada por el cliente.

RF5. Editar una categoría solicitada por el cliente.

RF6. Obtener todos los productos.

RF7. Obtener todos los productos por categoría.

RF8. Comprar cierta cantidad de un producto.

RF9. Crear nuevo producto.

RF10. Eliminar un producto solicitado por el cliente.

RF11. Obtener un producto solicitado por el cliente.

RF12. Editar un producto solicitado por el cliente.

RF13. Listar los productos más vendidos en el día.

RF14. Listar los productos más vendidos en la semana.

RF15. Listar los productos más vendidos en el mes.

RF16. Listar las categorías más vendidas en el día.

RF17. Listar las categorías más vendidas en la semana.

RF18. Listar las categorías más vendidas en el mes.

RF19. Listar el producto más vendido globalmente.

RF20. Autenticar Usuario.

RF21. Obtener la configuración del sitio.

RF22. Establecer Configuraciones del sitio.

2.2.2 Requisitos no Funcionales

Son propiedades o cualidades que el producto debe tener que lo hacen atractivo, usable, rápido o confiable. Estos requisitos pueden marcar la diferencia entre un producto aceptado y otro con poca aceptación, no definen el éxito general del producto, pero si influyen en la evaluación del cliente. (Sommervilles, 2005)

- **Software del Servidor:**

RNF1. GNU/Linux como sistema operativo.

- **Hardware de Servidor:**

RNF2. El servidor web requiere como mínimo 512 MB (Mega Bytes) de RAM (*Random Access Memory*).

RNF3. El disco duro requiere, como mínimo, 4 GB para almacenar la Base de Datos.

RNF4. El servidor debe estar instalado en una Raspberry Pi.

- **Seguridad:**

RNF5. Los métodos que requieren de autentificación en la capa de servicios deben estar disponibles solo para aquellos usuarios correctamente autenticados.

RNF6. Se ha definido que se devuelva un token al autenticar un usuario y que el tiempo límite de demora que tiene el usuario para solicitar un servicio antes de que expire su token es de 2 horas, pasado ese tiempo tendría que volver a obtener uno nuevo.

- **Rendimiento**

RNF7. La capa de servicios debe ser capaz de responder una petición en el menor tiempo posible dependiendo del tipo de petición y los datos que se manejan en esta.

2.3 Modelo de Datos

Un modelo de datos es una colección de herramientas conceptuales para la descripción de datos, relaciones entre datos, semántica de los datos y restricciones de consistencia. (Silberschatz, Korth, & Sudarshan)

Es el lenguaje orientado a describir la Base de Datos, permite almacenar, organizar, manipular cantidades de datos con cierta facilidad y describir los elementos de la realidad que intervienen en el problema dado y la forma en que se relacionan estos elementos entre sí. Cuando se utiliza una base de datos para gestionar información, se está plasmando una parte del mundo real en una serie de tablas, registros y campos ubicados en un ordenador. (Rodríguez & Almaguer Guerra, 2016)

En el modelo de datos de la solución se muestran tres tablas que están relacionadas entre si, la tabla Products que contiene todos lo relacionado con los productos, la tabla Category que contiene todo lo relacionado con las categorías y una tabla estadísticas que contiene las estadísticas brindadas por el sistema

Seguidamente se muestra el modelo de base de datos que se gestionan en el sistema propuesto.

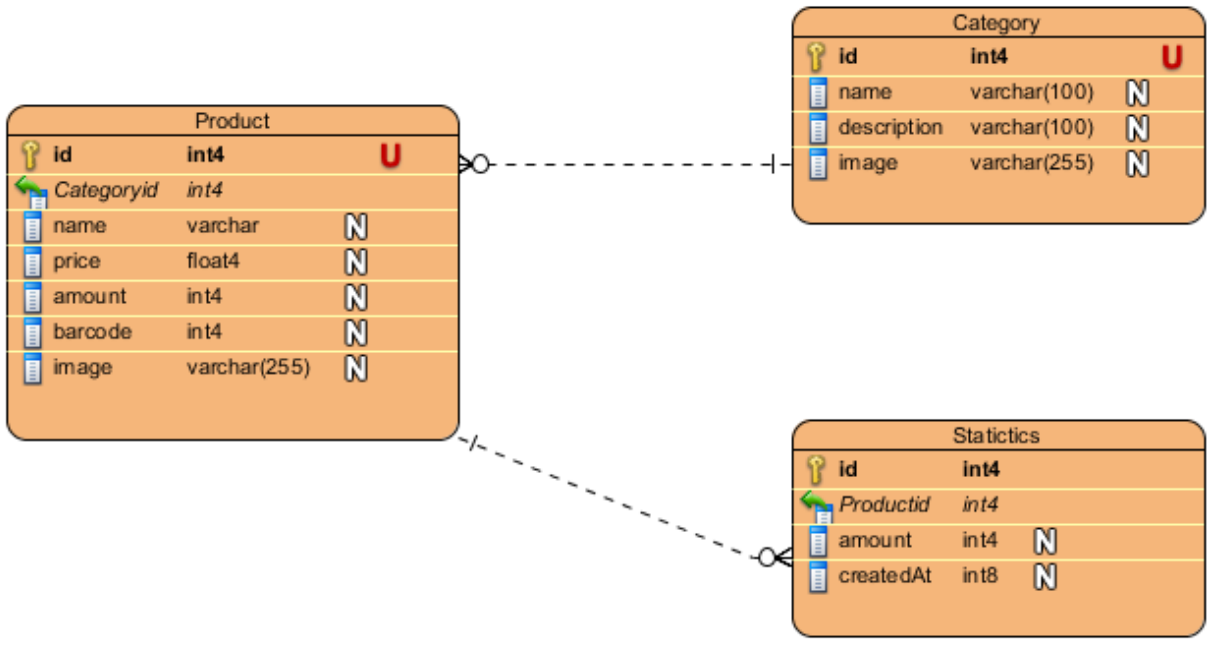


Ilustración 11 Modelo de Datos.

2.4 Historias de Usuario

Las Historias de Usuario (HU) son herramientas que dan a conocer los requisitos del sistema al equipo de desarrollo. En ellas el cliente describe la actividad que se realizará en el sistema, muestran la silueta de una tarea a realizar, deben ser claras, sencillas y sin profundizar en detalles. También, son utilizadas para estimar el tiempo que el equipo de desarrollo tomará para realizar las entregas, para estas estimaciones se le asignan unidades de medida a las HU que representen magnitud y no días reales de duración, tales como: puntos de estimación, días ideales u otra unidad de medida. (Cabrera & Ricardo González, 2015)

Seguidamente se detallan los campos que componen las HU en función:

- **Número de HU:** Este campo contiene el número que le es asignado a las HU consecutivamente.
- **Nombre de HU:** Identifica la HU a la que se hace alusión.
- **Usuario:** Usuario del sistema que interactúa o realiza la funcionalidad.
- **Prioridad en el negocio:** Prioridad que le es asignada a la HU en el negocio de acuerdo a las necesidades del usuario. Los valores que puede tomar son Alta, Media o Baja.
- **Riesgo en el desarrollo:** Riesgo que se corre en el desarrollo de la aplicación en caso de no realizarse la HU. Los valores que puede tomar son Alto, Medio o Bajo.
- **Tiempos estimados:** Esta estimación se realiza en semanas trabajando 8 horas diarias de lunes a viernes, la misma es formalizada por el equipo de desarrollo evaluando el tiempo que incurre en la realización de esta HU.
- **Iteración asignada:** Este campo es en dependencia de la prioridad que se establece a la HU, lo cual determina en la iteración que se desarrolla la HU.
- **Descripción:** Se realiza una pequeña descripción de la función de la HU.
- **Observación:** Datos adicionales de la HU en cuestión, resalta datos que son necesarios cumplir, para una mejor utilización y entendimiento de la funcionalidad que se brinda.

Se muestran a continuación algunas de las HU:

Historia de Usuario	
Número: HU_6	**Nombre Historia de Usuario:** Obtener todos los productos

Modificación de Historia de Usuario Número: *6*	
Usuario: Janderd Pérez Pérez	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Alto	
Descripción: Lista todos los productos existentes en la base de datos con sus respectivas categorías.	

Tabla 6 HU_1 Obtener todos los productos.

Historia de Usuario	
Número: HU_7	**Nombre Historia de Usuario:** Crear nuevo producto
Modificación de Historia de Usuario Número: *7*	
Usuario: Janderd Pérez Pérez	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	
Descripción: Permite al usuario crear un nuevo producto. Para ello se tienen los siguientes campos: • Nombre • Precio • Cantidad • Categoría • Código de barras • Imagen	
Observaciones: El usuario debe estar autenticado. El sistema devuelve todos los datos del producto si se suministró correctamente el token de usuario.	

Tabla 7 HU_7 Crear nuevo producto.

Historia de Usuario	
Número: HU_3	**Nombre Historia de Usuario:** Eliminar una categoría
Modificación de Historia de Usuario Número: *3*	
Usuario: Janderd Pérez Pérez	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	
Descripción: Permite eliminar una categoría. Se eliminarán todos los datos de la categoría en la base de datos como también todos los productos asociados a la misma. También se eliminan las estadísticas sobre los productos eliminados.	
Observaciones: El usuario debe estar autenticado. El sistema elimina la categoría si se suministró correctamente el token de usuario.	

Tabla 8 HU_3 Eliminar una categoría.

Historia de Usuario

Número: HU_17	**Nombre Historia de Usuario:** Autenticar Usuario
Modificación de Historia de Usuario Número: *17*	
Usuario: Janderd Pérez Pérez	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alta	
Descripción: Permitir autenticar al usuario en el sistema y devolver un token para poder acceder a los servicios.	
Observaciones: • El usuario se autentica con una contraseña y el sistema le devuelve un token a utilizar por 2 horas.	

Tabla 9 HU_17 Autenticar Usuario.

Historia de Usuario	
Número: HU_11	**Nombre Historia de Usuario:** Listar productos más vendidos en el día
Modificación de Historia de Usuario Número: *11*	
Usuario: Janderd Pérez Pérez	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	
Descripción: Listar los productos que más se venden en el día.	

Tabla 10 HU_11 Listar productos más vendidos en el día.

2.5 Diseño

El diseño de una solución en cuanto a funcionamiento e implementación es una de las pautas seguidas mundialmente para el desarrollo de *software* de alta calidad, del cual se verifica su éxito una vez superadas todas las pruebas realizadas al sistema. Un diseño favorable consiste en minimizar el esfuerzo de trabajo contando con la menor cantidad de métodos y clases a implementar. (González Guadalupe & León Pérez, 2015)

2.5.1 Arquitectura del Software

La arquitectura del software determina la estructura general del mismo y las formas en que proporciona una integridad conceptual para un sistema. En su forma más simple, la arquitectura es la organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan. En un sentido más amplio, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones. (Sommervilles, 2005)

Patrón Arquitectónico Modelo Vista Controlador

El patrón Modelo Vista Controlador separa los datos y la lógica del negocio de una aplicación de la interfaz de usuarios y el módulo encargado de gestionar los eventos y las comunicaciones. Permite además separar cada una de las lógicas del módulo en archivos independientes permitiendo hacerlo

mucho más flexible y sencillo de mantener (Sommervilles, 2005). El Framework de desarrollo definido, en este caso Symfony2, asigna el uso del patrón arquitectónico MVC.

Modelo: El modelo administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista). (Reynoso & Kicillof, 2004) Es decir, contiene la información básica del módulo. Esto incluye reglas de validación, acceso a datos y lógica de agregación.

Vista: Maneja toda la visualización de la información. Genera una representación visual del modelo y muestra los datos al usuario. (Reynoso & Kicillof, 2004)

Controlador: Es el componente que da soporte a las funcionalidades de la capa de negocio y que se encuentra relacionado con la fuente de datos. La principal función de esta capa es realizar una implementación de las funcionalidades definidas en las interfaces de la capa de negocio y al mismo tiempo trabajar directamente con la fuente de datos (Reynoso & Kicillof, 2004). Se utilizan diferentes clases controladoras conteniendo disímiles servicios web con similitudes en cuanto a los recursos que son mostrados.

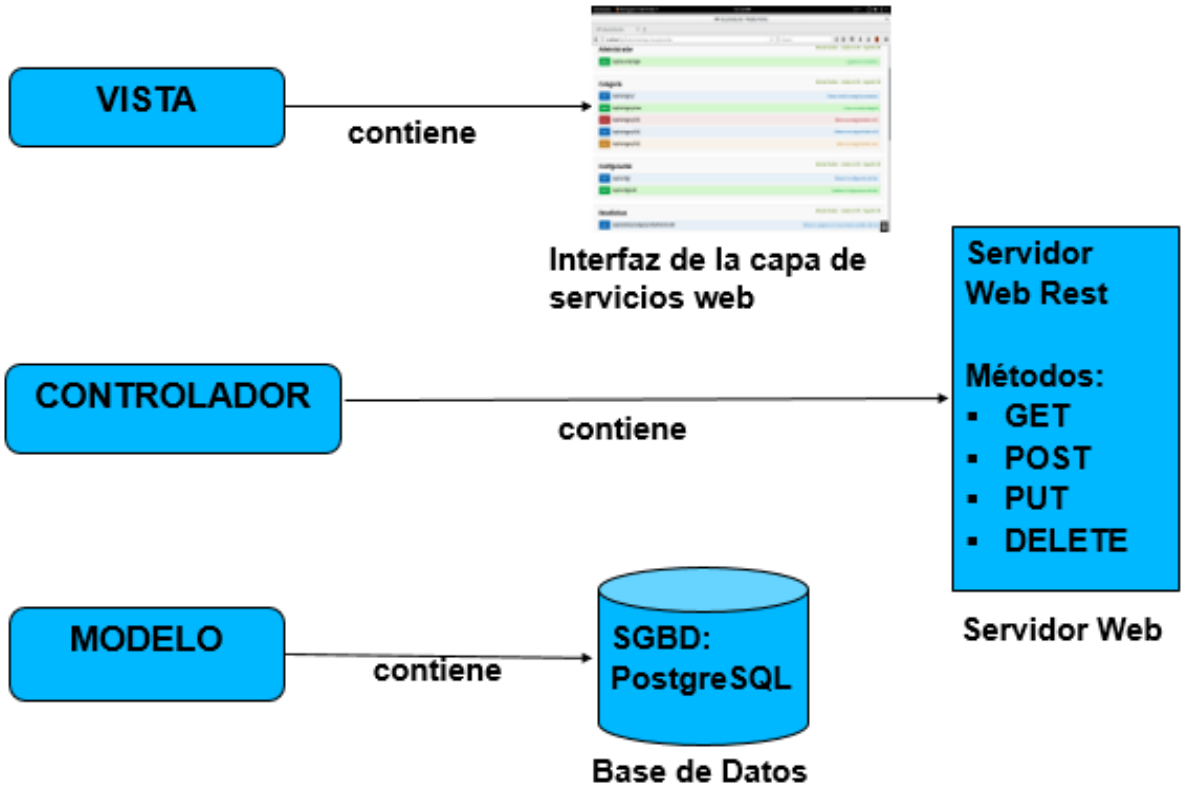


Ilustración 12 Patrón arquitectónico MVC. Contenido de cada capa.

2.5.2 Patrones de Diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema de diseño dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y utiliza el patrón. (Larman, 2001)

Patrones GRASP

Los patrones GRASP (del inglés: *General Responsibility Assignment Software Patterns*) son patrones generales de software para asignación de responsabilidades.

Controlador: Se utilizan diferentes clases controladoras conteniendo diferentes servicios web con similitudes en cuanto a los recursos que son mostrados. (*ProductController.php, CategoryController.php, SattiticsController.php*).

Bajo Acoplamiento: Es el patrón que se encarga de medir el nivel en que una clase depende o está conectada a otra, basándose en la idea de desligar la relación entre las clases de manera que, si se tiene que hacer alguna modificación o cambio en una de ellas, se tenga la mínima repercusión posible en las clases que dependen de la clase modificada, es decir, que no se afecte el diseño por cambios en otros componentes (*ApiController extends ForRestController)*.

Alta Cohesión: Con el uso de este patrón, se resuelve el problema de saber cómo mantener la complejidad dentro de límites manejables, pues es el encargado de asignar una responsabilidad de modo que la cohesión siga siendo alta. Su puesta en práctica permite lograr un diseño claro y fácil de comprender. Además de que aumenta la capacidad de reutilización.

Patrones GOF

Los patrones GOF (del inglés: *Gang of Four*) son patrones de diseño software que solucionan problemas de creación de instancias.

Contrato Uniforme: Es un patrón utilizado exclusivamente en servicios web, ya que plantea el uso de los cuatro verbos principales del protocolo HTTP en tareas específicas. Se pone de manifiesto en la capa de servicios debido al uso de estos verbos de forma estándar para las siguientes tareas:

- POST: Crear recursos.
- GET: Servicios que devuelven el estado de los recursos.
- PUT: Para actualizar recursos dado la URI del mismo.

- DELETE: Eliminar un recurso y después la URI de la misma no es válida.

Fachada: Este patrón es estructural y es el encargado de proporcionar una interfaz unificada. Define una interfaz de alto nivel que hace que el sistema sea más fácil de usar.

Observador: Este patrón es de comportamiento, define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y se actualicen automáticamente todos los objetos que dependen de él.

2.6 Conclusiones parciales

- La modelación mediante historias de usuario para el encapsulamiento de los requisitos resultó ser un método eficaz para la descripción y especificación de los mismos.

- El diseño de la arquitectura MVC y el uso de los patrones seleccionados garantizó la compatibilidad entre las capas (Modelo, Vista, Controlador) dentro del sistema.

- La modelación y el diseño crearon la base para la implementación, como siguiente fase de desarrollo, que logra dar cumplimiento a los requisitos tanto funcionales como no funcionales descritos

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3. Introducción

El período de implementación y pruebas es el momento en el que luego de haber realizado el diseño del sistema, se comienza la elaboración de los elementos de código que convertirán lo planificado en un producto real. Posteriormente se realizan las pruebas al software que permitan verificar la correcta implementación de los requisitos establecidos.

3.1 Modelo de Implementación

En el modelo de implementación se describe como los elementos del modelo del diseño se realizan en términos de componentes y como estos se organizan de acuerdo a los nodos específicos en el diagrama de despliegue.

3.1.2 Diagrama de Despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra como los elementos y artefactos del software se trazan en esos nodos. (SparxSystems, 2007)

El diagrama de despliegue de la capa de servicios web para un sistema de punto de venta que se muestra en la figura, se encuentra estructurado de la siguiente forma: el dispositivo externo (Aplicación Móvil) se conecta al servidor mediante el protocolo de comunicación HTTP(s) y este a su vez a la base de datos a través de TCP/IP.

Ilustración 13 Diagrama de Despliegue.

3.2 Estándares de código

Estándar de código, convención de código, o estilo de programación es un término que describe convenciones para escribir código fuente en cierto lenguaje de programación. Las convenciones de código son importantes para los programadores debido a que el 80% del coste del código de un programa va a su mantenimiento. Casi ningún software lo mantiene toda su vida el autor original. Las convenciones de código mejoran la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo. Para que funcionen las convenciones, cada persona que escribe software debe seguir la convención. (Rodríguez & Almaguer Guerra, 2016)

Para la implementación de la capa de servicios web, se seleccionó el lenguaje de programación PHP. La propuesta de solución cumple con el documento “Convenciones del Código PHP”.

Convenciones de Código PHP utilizadas

- Sangría y espacio en blanco: Utilizar un espaciado doble, sin tabulación. Las líneas no deben tener espacios en blanco detrás al final.
- Operadores: Todos los operadores binarios (operadores que se encuentran entre dos valores), como +, -, =,>! =, ==, deben tener un espacio antes y después de que el operador.
- Estructuras de control: Estructuras de control, que incluyen *if, for, while, switch.*
- Declaración de funciones: Cuando se llama a los constructores de clase sin argumentos, siempre incluyen entre paréntesis
- Funciones y variables: Funciones y variables deben ser nombrados con minúsculas, y las palabras deben estar separadas por un guión.

3.3 Tratamiento de Errores

Durante el desarrollo del sistema, se han definido una serie de acciones que pueden provocar un mal funcionamiento, para cada una de ellas se definió un tratamiento específico, para asegurar una respuesta adecuada que garantice cierto grado de estabilidad.

El tratamiento de errores debe estar enfocado tanto a los errores que se producen a la hora de los usuarios introducir datos en la pantalla principal de la aplicación, como a los errores que puedan ser generados por el comportamiento incorrecto de los componentes internos. (Rodríguez & Almaguer Guerra, 2016)

En la capa de servicios web el seguimiento que se le da a los datos introducidos por los usuarios es mediante validaciones que se les asignan a los parámetros de entrada, garantizando que se muestre el tipo de error sobre el parámetro donde se introdujeron los datos incorrectos. Una ventaja de darle seguimiento a los errores es poder señalar y separar el tratamiento de los errores mediante excepciones y poder cumplir una respuesta a errores particulares. A continuación, se muestra una tabla que evidencia el tratamiento de errores.

#error HTTP	Error	Razón

400	Petición Incorrecta	Incluye una serie de errores relacionado con las peticiones que hace el usuario hacia la capa de servicio si no cumplen estas con las especificaciones requeridas.
403	No autorizado	Incluye una serie de errores que indican que no está autorizado el usuario a ejecutar esa petición en el sistema. Ya sea porque no está autentificado o expiró el tiempo de uso del token.
404	No encontrado	Incluye una serie de errores en las peticiones del usuario a la hora de pedir un recurso que no existe en el sistema.

Tabla 11 Posibles errores que contienen las respuestas de los servicios.

3.4 Pantalla Principal de la Aplicación

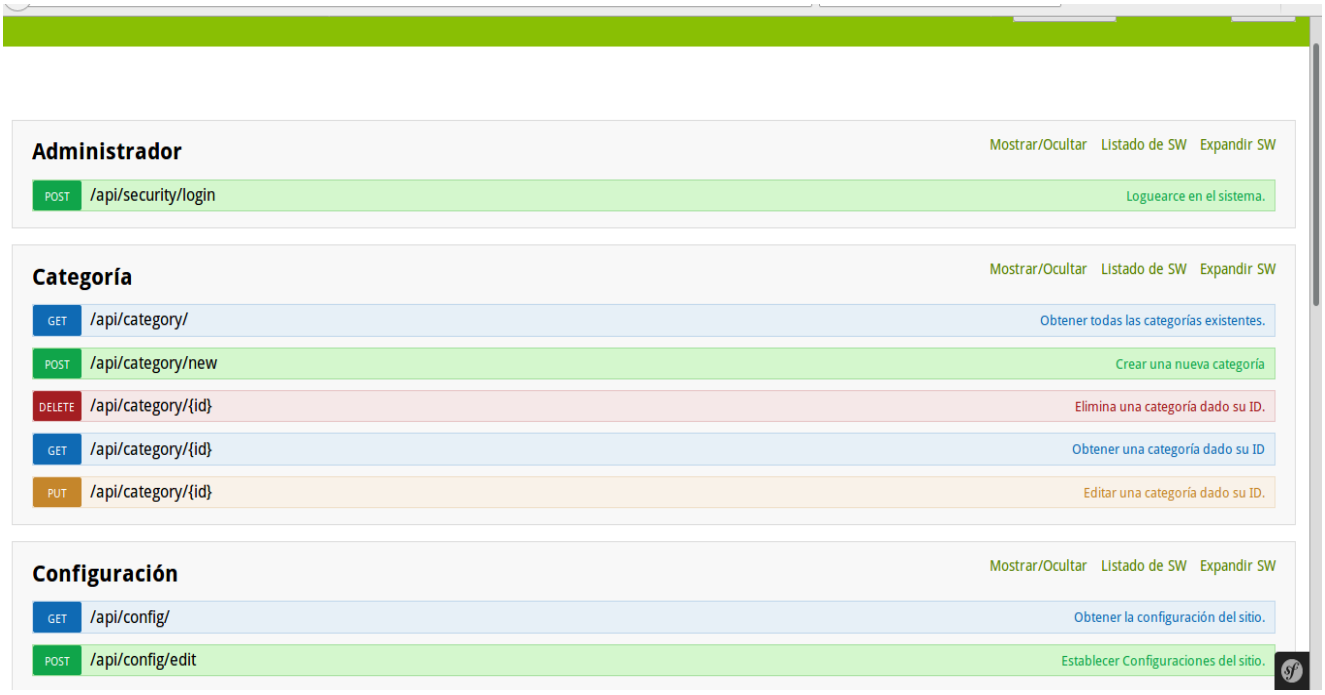


Ilustración 14 Vista del fronted de la capa de servicios web.

3.5 Modelo de Pruebas

Para que la capa de servicios Web se considere lista para ser desplegada debe someterse previamente a una etapa de pruebas rigurosas, con el fin de analizar si cumple con las funcionalidades requeridas. Las pruebas que se le realizarán a la aplicación son un elemento crítico para la garantía de la calidad, representan además la revisión final de las especificaciones, los requerimientos, el análisis, diseño e implementación. El principal objetivo de estas pruebas es descubrir errores; estos serán corregidos posteriormente para que la aplicación final cumpla con lo previsto y tenga la eficiencia y la calidad requerida. Aunque las pruebas no pueden asegurar la ausencia de defectos; sí pueden demostrar la existencia de estos. (Pressman, 2012)

El modelo de prueba describe principalmente como se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. El modelo de pruebas puede describir también como han de ser probados aspectos específicos de sistemas, por ejemplo, si el manual de usuario del sistema cumple con su cometido. El modelo de pruebas es una colección de casos de prueba, procedimientos de prueba y componentes de prueba. (Vivv, 2008)

Descripción de los tipos de prueba realizadas

Para la realización de las pruebas al sistema, se definieron una serie de pruebas fundamentales para la validación de los servicios web:

Pruebas Funcionales: Tiene como objetivo asegurar el cumplimiento apropiado de los requisitos funcionales, entrada de datos, procesamiento y obtención de resultados. La principal intención de estas pruebas es medir la correspondencia entre la arquitectura de información propuesta y las funciones que realmente fueron implementadas. Para realizar este tipo de pruebas es necesario tener definidos los requerimientos a verificar con los casos de prueba apuntando a cada uno de los requisitos. Estos serán los encargados de verificar la aplicación implementada. Se utilizará el método de caja negra.

Pruebas de Rendimiento: La prueba de rendimiento de *software* se enfoca en la capacidad de recibir peticiones mediante la utilización de alguna herramienta, verificando cuantas peticiones pueda sostener el sistema sin que este se vea afectado, así como la velocidad de respuesta del mismo. Las pruebas de rendimiento para la capa de servicios web serán realizadas utilizando la herramienta JMeter.

Pruebas de Seguridad: La prueba de seguridad y control de acceso se enfocará a la seguridad en el ámbito de la aplicación, asegurando que los mecanismos de protección incorporados en el sistema

lo protegerán de accesos no autorizados y que los usuarios solo accedan a los servicios que requieran autentificación cuando estén debidamente registrados y autentificados en el sistema.

3.5.1 Pruebas Funcionales

Las pruebas se le realizaron a cada una de las historias de usuarios de la capa de servicios web. En el desarrollo de la capa de servicios Web se diseñaron un conjunto de casos de prueba de las diferentes funcionalidades del sistema. A continuación, se describen algunos de los casos de prueba utilizados.

Historia de Usuario	HU_6 Obtener todos los productos.
Requisito	RF6.
Condición	Deben existir productos registrados en la tabla Products de la Base de Datos del sistema.

Tabla 12 Caso de Prueba: Obtener Productos.

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Id	Campo de texto	No	Solo números.
2	page	Campo de texto	No	Entero que indica el número de página.
3	size	Campo de texto	No	Entero que indica el tamaño de página.

Tabla 13 Descripción de variables. Caso de Prueba: Obtener Productos.

Escenario	Descripción	id	size	page	Respuesta del sistema	Flujo Central

EC 1.1 Obtener todos los Productos	Lista todos los productos existentes en la base de datos con sus respectivas categorías.	N/A	V	N/A	Devuelve todos los productos que están almacenados. Se muestra los siguientes datos: Identificador del producto, nombre, precio, cantidad, categoría, estadística.	/products
EC1.2 Obtener un producto solicitado por el cliente	Devuelve al cliente el producto que solicito a través del id del mismo.	V	N/A	N/A	Devuelve un producto solicitado por el cliente: Se muestra los siguientes datos: Identificador del producto, nombre, precio, cantidad, categoría, estadística.	/products/{id}
EC1.3 Obtener los 100 primeros productos.	Devuelve los 100 primeros productos existentes en la base de datos.	N/A	V	V	Devuelve los 100 primeros productos que están almacenados. Se muestra los siguientes datos: Identificador del producto, nombre, precio, cantidad, categoría, estadística.	/products

Tabla 14 Diseño de caso de prueba: Obtener Producto.

Historia de Usuario	HU_17 Autenticar Usuario
Requisito	RF17
Condición	Debe existir un usuario registrado en el sistema.

Tabla 15 Caso de Prueba: Autenticar Usuario.

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	usuario	Campo de texto	No	Admite solo letras
2	contraseña	Campo de texto	No	Admite cualquier carácter

Tabla 16 Descripción de variables. Caso de Prueba: Autenticar usuario.

Escenario	Descripción	usuario	contraseña	Respuesta del sistema	Flujo Central
Ec 1.1 Autenticar usuario	Autentica al usuario y devuelve un token de usuario.	V	V	El usuario es autenticado y puede hacer uso de los demás servicios web.	/security/login
EC 1.2 Datos Incorrectos	Datos incorrectos	V I	I V	Devuelve un error HTTP 403.	/security/login

Tabla 17 Diseño de caso de prueba: Autenticar usuario.

Resultados de las pruebas funcionales

La implementación de las pruebas funcionales muestra una visión de la calidad del *software*. En una primera iteración, se obtuvo como resultado del análisis de los escenarios de pruebas, un conjunto de no conformidades correspondientes a los servicios web implementados. Las no conformidades fueron corregidas posteriormente, dándole paso a la segunda iteración de pruebas, en las cuales se evaluaron nuevamente los servicios web hasta no quedar ninguna no conformidad. A continuación, se muestran los principales datos de cada iteración.

Iteración	No Conformidades	Descripción (No conformidades más significativas)

1	7	• Respuestas incorrectas de algunas funcionalidades como el obtener producto, y el obtener categoría. • El formato de los errores en las respuestas no estaban en JSON.
2	3	• Algunos servicios no cumplían con los estándares REST a la hora de consumirlos. • Errores ortográficos en la descripción de los servicios.
3	0	

Tabla 18 No Conformidades por Iteración.

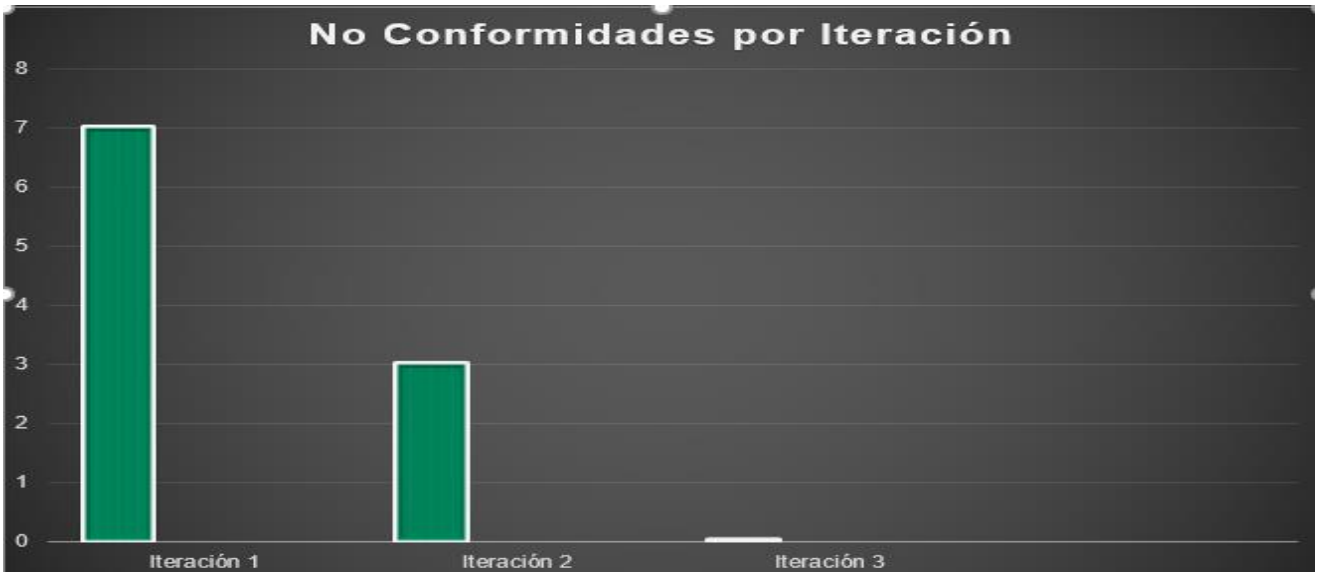


Ilustración 15 Resultados de las Pruebas Funcionales.

Se validó en sentido general el correcto funcionamiento y cumplimiento de los requisitos funcionales de la capa de servicios, de esta forma el desarrollo de las pruebas funcionales muestra una visión de la calidad del *software*. Para una total verificación de las funcionalidades de la capa de servicios web se utilizó la interfaz de prueba generada con el NelmioApiDocBundle, el cual es un bundle de Symfony, que ayuda en la documentación, a su vez cuenta con la herramienta “Sandbox” (Symfony, 2017) para probar cada una de las funcionalidades.

3.5.2 Pruebas de Rendimiento

Para probar la escalabilidad del sistema es necesario conocer acerca de la carga que puede soportar. Para ello se implementaron las pruebas de rendimiento de software, las cuales se apoyaron en el uso de la herramienta JMeter, que ofrece datos importantes acerca del *stress* que puede tolerar el sistema.

Para la realización de las pruebas se utilizó el entorno de hardware y software del servidor:

- Sistema Operativo Ubuntu 16.04
- Servidor Web Apache en su versión 2.2
- PostgreSQL en su versión 9.4

Resultados de las pruebas de rendimiento

Se probó el sistema con un total de 2200 peticiones enviadas en 100 hilos de ejecución concurrentes en once servicios web.

Label	**#Muestras**	**%Error**	**Rendimiento**	**Kb/sec**	**Avg. Bytes**
Obtener todas las categorías	200	11.50%	2.4/sec	587.69	255709.6
Obtener una categoría solicitada por el cliente	200	0.50%	2.6/sec	15.05	5910.4
Obtener todos los productos	200	8.50%	2.5/sec	202.84	83631.8
Obtener un producto solicitado por el cliente	200	1.50%	2.7/sec	9.63	3715.1
Autenticar Usuario	200	4.00%	2.7/sec	2.34	901.2
Listar Productos más vendido en el día	200	0.50%	2.9/sec	4.68	1650.8

Crear un nuevo producto	200	0.50%	3.0/sec	1.85	641.8
Crear una nueva categoría	200	1.50%	3.0/sec	206.16	70301.0
Comprar X cantidad de un producto	200	0.00%	3.4/sec	0.01	2.0
Obtener la configuración del sitio	200	0.00%	3.4/sec	3.80	1154.0
Establecer configuraciones del sitio	200	0.00%	3.4/sec	10.83	3295.0
TOTAL	2200	2.59%	**23.7/sec**	899.39	38810.3

Tabla 19 Resultados de las pruebas con JMeter.

El significado de algunos campos se describe a continuación:

- **Label:** el nombre de la petición HTTP(s) o nombre del servicio.
- **# Muestras:** el número de muestras para cada petición.
- **% Error:** porcentaje de las peticiones con errores.
- **Rendimiento:** rendimiento medido en peticiones por segundo/minuto/hora.
- **Kb/segundos:** rendimiento medido en kilobytes por segundo.
- **Avg. Bytes:** Este es el promedio de datos de la aplicación que obtenemos durante la ejecución.

Por lo tanto, se puede decir que la capa de servicios web cumple con un rendimiento esperado (**23.7** peticiones por segundo según la herramienta) para atender la carga y *stress* que se genere de atender las solicitudes de los diferentes clientes que pueda conectarse a la capa.

3.5.3 Pruebas de Seguridad

Se probó manualmente un token de usuario devuelto por el servicio de autenticar usuario y se verificó que las funcionalidades que lo requerían funcionan correctamente, proporcionando dicho token.

Además, se probó que el token solo fuera válido en el tiempo asignado y expirará después de cumplido el mismo.

3.6 Conclusiones Parciales

- El tratamiento de errores del sistema se concentró en aquellos que se pudieran producir a la hora de los usuarios introducir datos en la pantalla principal (“sandbox”) de la aplicación, y los que puedan ser generados por el comportamiento incorrecto de los componentes internos. Por lo que se decidió darle un seguimiento a los datos introducidos por los usuarios mediante validaciones que se les asignan a los parámetros de entrada. Lo que permitió señalar y separar dicho tratamiento mediante excepciones para poder dar una respuesta a errores específicos.

- A través del modelo de pruebas se describieron las pruebas funcionales, de rendimiento y seguridad realizadas en el sistema. Se utilizó la validación de la aplicación mediante pruebas de caja negra. Se obtuvieron resultados satisfactorios y se demostró el correcto desarrollo de la capa de servicios web.

CONCLUSIONES

- A lo largo de la investigación se han ido describiendo todos los pasos seguidos para darle cumplimiento al objetivo de desarrollar una capa de servicios web que permita ofrecer información para el sistema de gestión de ventas a implementar por el CEDIN.

- Se analizó un estudio del arte basado en los servicios web, así como el impacto que tienen estos sobre las TPV, donde a partir de ahí se identificaron algunas funcionalidades que por sus características pudieron ser implementados como servicios.

- Para comenzar la implementación se definieron los requisitos funcionales y no funcionales, además se realizó la modelación y el diseño los cuales dieron el punto de partida y apoyo para lograr el objetivo final: poner en funcionamiento la capa de servicios web.

- Por otra parte, se propuso la realización del diagrama de despliegue en el cual se expone la ubicación física de cada componente hardware y software.

- Se empleó como metodología de desarrollo AUP-UCI, así como herramientas y tecnologías multiplataforma, según las políticas de la universidad.

- A través de las diferentes pruebas de software ejecutadas en la capa de servicios web se validó el correcto funcionamiento del sistema, permitiendo demostrar la interoperabilidad entre el sistema.

- Por tanto, se concluye que los objetivos propuestos para el presente proyecto, han sido cumplidos satisfactoriamente. Se dan una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

RECOMENDACIONES

Con el objetivo de mejorar y seguir perfeccionando la capa de servicios web, se recomienda la implementación de más funcionalidades como servicios para el control de inventario y operaciones comerciales determinadas.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar, Y. D., & Maure Morejón, A. R. (2013). *Gestión de tipos documentales del Sistema para la obtención de documentos digitales con valor legal.* Universidad de las Ciencias Informáticas (UCI), La Habana. Recuperado el 11 de marzo de 2017, de http://publicaciones.uci.cu

Álvarez, M. A. (19 de diciembre de 2014). *Desarrollo Web.* Recuperado el 1 de diciembre de 2016, de Ventajas e inconvenientes de API REST para el desarrollo: http://www.desarrolloweb.com/articulos/ventajas-inconvenientes-apirest-desarrollo.html

BEExterior. (2016). *Tipos de TPV.*

Boci, L. (2012). *Comparison between JSON and XML in Applications on AJAX. CSSS.* Recuperado el 20 de junio de 2017

Cabrera, Y. C., & Ricardo González, G. M. (2015). *Informatización del procedimiento de Revisiones Técnicas Formales en el área Arquitectura del desarrollo de software para el centro FORTES.* Universidad de las Ciencias Informáticas, La Habana. Recuperado el 11 de marzo de 2017

Castro, M., Sánchez Rivero, D., Farfán, J., Castro, D., Cándido, A., & Vargas, A. (2013). *Aplicación de Servicios Web SOAP/REST para funcionalidades existentes en sistemas informáticos existentes.* Universidad Nacional de Jujuy, Jujuy. Recuperado el 28 de noviembre de 2016

Código, E. d. (junio de 2014). *Editores de Código*. Recuperado el 16 de junio de 2017, de PhpStorm: http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html

ComputerHoy. (2016). *ComputerHoy*. Recuperado el 13 de marzo de 2017, de Que es raspberry pi: http://computerhoy.com/noticias/hardware/que-es-raspberry-pi-donde-comprarla-como-usarla-8614

deSymfony. (2016). *Symfony.es*. Recuperado el 15 de junio de 2017, de http://symfony.es/pagina/que-es-symfony/

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architecture. PhD Thesis.* Recuperado el 15 de junio de 2017, de http://roy.gbiv.com/pubs/dissertation/top.html

GoldenSoft. (2013). *GoldenSoft*. Recuperado el 17 de junio de 2017, de http://www.goldensoft.com.

González Guadalupe, E. A., & León Pérez, M. A. (2015). *Sistema de Gestión de la Guardia Obrera Estudiantil en la Fac 2.* Ciudad Habana. Recuperado el 6 de diciembre de 2016

González, M. H. (Junio, 2009). *Diseño de la Base de Datos del Sistema de Información de Perforación de Pozos.* Universidad de Ciencias Informáticas, La Habana. Recuperado el 20 de noviembre de 2016

i2factory. (26 de febrero de 2016). *i2factory.* Recuperado el 29 de noviembre de 2016, de Que es un servicio Restful: http://www.i2factory.com/es/integracion/qué-es-un-servicio-restful

INFORMATICAHOY. (2013). *INFORMATICAHOY.* Obtenido de Que son los sistemas de gestión empresarial: www.informatica-hoy.com.ar/.../Que-son-los-sistemas-de-gestion-empresarial.php

Ingram, D. (2016). *La Voz.* Recuperado el noviembre de 2016, de Que es un sistema de gestión de la información: http://pyme.lavoztx.com

Larman, C. (2001). *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Recuperado el 15 de junio de 2017

Machuca, C. A. (2010). *Estado del Arte: Servicios Web.* Universidad Nacional de Colombia, Bogotá. Recuperado el 15 de junio de 2017

Marset, R. N. (2007). *Modelado, Diseño e Implementacion de Servicios Web. .* ELP-DSIC-UPV. Recuperado el 28 de noviembre de 2016

Marset, R. N. (2007). *REST vs Web Services.* ELP-DSIC-UPV. Recuperado el 20 de junio de 2017

MBCEStore. (2013). *MBCEStore.* Recuperado el 5 de diciembre de 2016, de www.mbcestore.com

Nuñez Horta, E., & García Pérez, I. (2014). *Subsistema Punto de Venta del Sistema de Gestión Integral de Organizaciones Xedro-ERP.* Universidad de las Ciencias Informáticas, Ciudad Habana. Recuperado el 5 de diciembre de 2016

Odoo. (2016). *Odoo*. Recuperado el 16 de junio de 2017, de https://www.odoo.com/es_ES/

Openbravo. (2016). *Openbravo*. Obtenido de www.openbravo.com/es/about/company/

Paradigm, V. (2013). *Visual Paradigm.* Recuperado el 13 de marzo de 2017, de http://www.visualparadigm.com

PostgreSQL. (2013). *Postgre SQL*. Recuperado el 2 de junio de 2017, de www.postgresql.org.es.

Pressman, R. S. (2012). *Ingienería del Software, Un Enfoque Práctico.* Recuperado el 16 de mayo de 2017

Programmableweb. (2016). *Programmableweb*. Recuperado el 15 de junio de 2017, de http://programmableweb.com

QODE. (28 de noviembre de 2013). *Web Services - REST vs SOAP.* Recuperado el 28 de noviembre de 2016, de http://qode.pro/blog/web-services-rest-vs-soap/

Reyez Velázquez, Y., & Jiménez Andarcio, O. (2010). *Sistema de Gestión de la Información Docente de la Fac 10.* Universidad de las Ciencias Informátcias, La Habana. Recuperado el 7 de diciembre de 2016

Reynoso, C., & Kicillof, N. (2004). *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires. Recuperado el 20 de junio de 2017

Rodríguez, C. A., & Almaguer Guerra, O. (2016). *Capa de servicios web para el Juez en Línea Caribeño.* Universidad de las Ciencias Informáticas. Recuperado el 5 de junio de 2017

SEO. (2015). *TEC ELECTRONICA.* Obtenido de https://tec-mex.com.mx/

Shires, Q. (2016). *eHow.* Obtenido de Ventajas y desventajas de los sistemas de puntos de ventas: http://www.ehowenespanol.com/ventajas-desventajas-sistemas-puntos-venta-lista_528507/

Sierra, M. (2006). Recuperado el 2016, de Aprender a programar: http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=57&Itemid=86

Silberschatz, A., Korth, F. H., & Sudarshan, S. (s.f.). *Fundamentos de Bases de Datos 4ta Edición.* Instituto Indio de Tecnología, Bombay. Recuperado el 13 de marzo de 2017

Sinnexus. (2016). *Sinnexus.* Obtenido de Bussiness Intelligence, Informatica Estrategica: http://www.sinnexus.com/business_intelligence/

Snell, J., Tidwell, D., & Kulchenko, P. (2002). *Programming Web Services with SOAP.* Oreally.

Sommervilles, I. (2005). *Ingeniería de software 7ma edición.* Recuperado el 15 de junio de 2017

SparxSystems. (2007). *SparxSystems.* Recuperado el 16 de junio de 2017, de http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

Symfony. (2017). *Symfony*. Recuperado el 28 de junio de 2017, de Sandbox: http://symfony.com/doc/current/bundles/NelmioApiDocBundle/sandbox.html

Vergara, G. (19 de agosto de 2014). *Mejora Tu Gestión.* Recuperado el 5 de diciembre de 2016, de http://mejoratugestion.com/mejora-tu-gestion/que-es-un-sistema-de-gestion/

Vivv, V. (2008). *Ingienería del Software.* Recuperado el 16 de mayo de 2017, de http://clases3gingsof.wikifoundry.com/page/Pruebas.

W3C. (2014). *W3C.* Recuperado el 11 de junio de 2017, de Servicios Web: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

Wachenchauzer, R., & Manterola, M. (2014). *Libros Web.* Obtenido de 11.7. Persistencia de datos (Algoritmos de Programación con Python): http://librosweb.es/

Weerawarana, S., Curbera, F., & Leymann, F. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL,WS-Reliable Messaging, and More.* Recuperado el 15 de junio de 2017