

# Caso de estudio experimental para el desarrollo de módulos basados en el software libre “Unicenta”

Urrego León, Carlos Enrique, Martínez Hincapié, Andrés Mauricio, Yepes Arbeláez, Leonardo Fabio

<sup>1</sup>Facultad de ciencias básicas e ingenierías, Universidad Católica de Pereira

Carrera 21 No. 49-95 Av. de las Américas

Pereira, Colombia.

carlos.urrego@ucp.edu.co

**Recibido:** 6 de junio de 2019

**Aceptado:** 20 de junio de 2019

## RESUMEN

El principal objetivo del caso de estudio es contribuir a la comunidad de software libre Unicenta, a través de la creación de una branch que permita la gestión de un software de comercio electrónico y atención a las mesas. La metodología empleada para el desarrollo de los módulos fue una combinación entre el modelo de ciclo de vida en espiral y el prototipado de software, esta elección fue tomada por la naturaleza de los módulos, y además porque era necesario una metodología confiable para la elicitación de requerimientos sin la presencia de un cliente en específico, esta combinación dio como resultado una metodología con 3 fases y cinco sub fases. Los resultados finales del caso de estudio son un documento con la ingeniería inversa del software Unicenta en el cual jugo un papel importante el apoyo de la comunidad gracias a que era la principal fuente de información, además de un nuevo proyecto derivado de Unicenta denominado OctoPOS el cual posee dos nuevas características las cuales son la sincronización con Woocommerce y una aplicación para dispositivos móviles que permite la toma de pedidos, desarrollos en los cuales el uso de la ingeniería de software y las buenas practicas fueron aspectos fundamentales para llegar al buen término de los mismos.

**Palabras clave:** Unicenta, NodeJS, Angular, Software libre.

## ABSTRACT

The main objective of the study case is to contribute to the free software community Unicenta, through the creation of a branch that allows the management of e-commerce software and attention to the tables. The methodology used for the development of the modules was a combination between the spiral life cycle model and software prototyping, this choice was taken by the nature of the modules, and also because a reliable methodology for elicitation was necessary of requirements without the presence of a client in specific, this combination resulted in a methodology with 3 phases and five sub phases. The final results of the case study are a document with the reverse engineering of Unicenta software in which the community support played an important role thanks to the fact that it was the main source of information, as well as a new project derived from Unicenta called OctoPOS. which has two new features which are the synchronization with Woocommerce and an application for mobile devices that allows the taking of orders, developments in which the use of software engineering and good practices were fundamental aspects to reach the successful completion of the same.

**Keywords:** Unicenta, NodeJS, Angular, Free Software

## 1. INTRODUCCIÓN

Las motivaciones que suscita el desarrollo del proyecto, atienden a razones de interés general de la comunidad de software libre, es decir, la actual rama de Unicenta no cuenta con algunas funcionalidades tales como la

sincronización del programa con un comercio electrónico y el desarrollo de una aplicación para dispositivos móviles, además, no posee un canal de contribución directa la cual facilite la creación de una comunidad que soporte e impulse el desarrollo del proyecto.

La oportunidad que se resalta en mayor medida en el desarrollo del proyecto, es la posibilidad de proporcionar a las empresas una herramienta con funcionalidades necesarias en la actualidad, distribuida bajo la licencia GPL V3.

La contribución a comunidades de software libre es una actividad que, al menos las personas o empresas involucradas en el sector de las tecnologías deberían realizar, ya que esta actividad proporciona grandes beneficios devuelta, tales como conocimiento profundo sobre un proyecto, lo cual beneficia demasiado a la hora de resolver posibles errores que surjan con el uso del mismo; reconocimiento dentro de la comunidad, ya que al ser reconocido dentro de las comunidades, se va creando una reputación, la cual servirá como apoyo a la hora de postularse a ofertas laborales o licitaciones en una empresa. Entre otros beneficios.

Las comunidades de software libre se vienen dando desde la década de los 80's, gracias a Richard Stallman creador del movimiento de software libre.

A partir de este momento, han surgido infinidad de proyectos de esta índole alrededor del mundo, uno de estos es Unicenta, un software desarrollado para suplir las necesidades de los pequeños negocios dedicados a la venta minorista. Este software es distribuido bajo la licencia GNU General Public License versión 3.0 (GPLv3), la cual convierte el proyecto en uno de software libre.

Las motivaciones que suscitaron el desarrollo del proyecto, atienden a razones de interés general de la comunidad de software libre, es decir, la actual rama de Unicenta no cuenta con algunas funcionalidades tales como la sincronización del programa con un comercio electrónico y el desarrollo de una aplicación para dispositivos móviles, además, no posee un canal de contribución directa la cual facilite la creación de una comunidad que soporte e impulse el desarrollo del proyecto.

La oportunidad que se resalta en mayor medida en el desarrollo del proyecto, es la posibilidad de proporcionar a las empresas una herramienta con funcionalidades necesarias en la actualidad, distribuida bajo la licencia GPL V3, por ende se decidió crear un proyecto derivado de Unicenta denominado OctoPOS.

## **1.1 Marco teórico**

### **1.1.1 Antecedentes**

Unicenta es un proyecto derivado del software OpenBravo Pos el cual es propiedad de la compañía OpenBravo, este proyecto comenzó siendo software libre y como una comunidad, pero al pasar el tiempo fue privatizado y explotado de forma comercial por la compañía; debido a esto surgieron derivados como lo es Unicenta. Así como Unicenta es un proyecto derivado, existen otros más que también son derivados de OpenBravo, pero además, gracias a la gran acogida de Unicenta por parte de la comunidad y a la falta de integración de las nuevas funcionalidades aportadas por ésta al proyecto, han ido surgiendo otros proyectos derivados de Unicenta tales como:

#### **1.1.1.1 Openbravo Java POS**

Openbravo POS fue creado en 2008 por Adrián Romero, es un punto de venta diseñado para pantallas táctiles, soporta impresoras de tickets, pantalla de cliente y lector de códigos de barras. Es multiusuario a la hora de proveer los permisos para agregar productos, mostrar reportes y gráficos (Openbravo POS, 2018). Además, este es el proyecto base de Unicenta y de muchos otros puntos de venta. Esta distribuido bajo la licencia GNU General Public License version 3.0 (GPLv3).

#### **1.1.1.2 Chromis POS**

Este proyecto se basa en Unicenta, fue creado por John Lewis en el año 2012, y es de origen inglés. Este proyecto fue creado básicamente porque John Lewis realizaba nuevas funcionalidades al código fuente de Unicenta las cuales no eran incluidas en las nuevas versiones, así que Lewis decidió comenzar su propio proyecto y agregó las nuevas

funcionalidades desarrolladas por él, cabe resaltar que la funcionalidad más importante en este proyecto y por la cual surgió, es el módulo para la pantalla de la cocina. El proyecto es distribuido bajo la licencia GNU General Public License versión 3.0 (GPLv3).

### **1.1.1.3 Nord POS**

Nord POS es un proyecto que toma como código base el de OpenBravo POS, fue desarrollado por Andrey Svininykh en el año 2014, y es de origen kazajo. A diferencia con OpenBravo y sus otras derivaciones, Nord POS fue desarrollado con soporte web y con compatibilidad para dispositivos móviles, debido a esto Nord POS posee una interfaz gráfica diferente a la de OpenBravo POS y brinda mayor funcionalidad a los usuarios. El desarrollo es distribuido bajo la licencia Apache License V2.0, GNU General Public License versión 3.0 (GPLv3)

### **.1.2.1 Conceptos teóricos**

A continuación, se dará la definición de algunos de los conceptos necesarios para la comprensión del artículo.

#### **1.2.1.1 Software libre**

Es aquel software que no vulnera las libertades básicas de un usuario, las cuales son:

- Libertad 0: La libertad de ejecutar el programa sea cual sea nuestro propósito
- Libertad 1: La libertad de estudiar el código del programa y poderlo adaptar a las necesidades específicas del usuario, el código abierto es un requisito fundamental para esto.
- Libertad 2: Es la libertad de redistribuir el código ya sea de forma gratuita o paga y además también se refiere a la libertad de poder ayudar al prójimo
- Libertad 3: Es la libertad de modificar el programa y poder redistribuir el código con el fin de ayudar a la comunidad. (Stallman, 2004).

#### **1.2.1.2 Unicenta oPOS**

Es un software dedicado a suplir las necesidades encontradas en los puntos de venta. Se enfoca en los mercados minoristas tales como restaurantes, hostelería, almacenes, supermercados entre otros (Manjacavas, 2016). Algunas de las funciones que más se destacan en este software son:

- Conectividad con impresoras fiscales de 1ra y 2da generación, comanderas, scanners entre otras.
- Manejo de sesiones de usuario, distinción de roles.
- Gestión de artículos.
- Gestión de inventario.
- Gestión de caja.
- Gestión de proveedores.
- Generación de informes.

Este programa es desarrollado en el lenguaje de programación Java y posee una fuerte orientación a objetos dentro de su código y utiliza como base de datos principal MySQL. Estas dos características en su desarrollo permiten su fácil portabilidad entre sistemas operativos y por esta razón puede ser ejecutado en Linux, Windows y Mac OS X 10.6 o posterior.

Durante el desarrollo se decidió que sería software libre y se encontraría bajo la licencia GNU GPL V3, además empezó a ser distribuido a través de canales casuales como lo son su sitio web, la página por excelencia de software libre sourceforge, entre otros métodos, en ninguno de los cuales se fuerza al usuario a registrarse, características que según Jack Gerrard su fundador, han sido la clave fundamental para el éxito del proyecto.

Como lo expresa Gerrard el crecimiento y la popularidad del proyecto han ido creciendo de boca en boca, además, el dinero no fue un motivo para la existencia de UniCenta. Esto se ve reflejado en la posición que ocupa Unicenta como una de las soluciones POS más usadas alrededor del mundo, con más de un millón de descargas desde su inicio, gracias a lo antes mencionado el proyecto ocupa la cuarta posición, en la división de puntos de venta, en el ranking de la página Capterra, la cual se dedica a calificar software para distintas aplicaciones.

### **1.2.1.3 Node.js**

Es un framework de desarrollo basado en el motor de JavaScript V8 de google, y es usado para desarrollar aplicaciones de red escalables, gracias a que logra un alto rendimiento a través de E/S sin bloqueo y un bucle de eventos de un sólo hilo. El código de NodeJS está escrito en JavaScript y es compilado en el motor V8, de esta manera, se puede escribir el código tanto del servidor como del cliente en JavaScript, del lado del servidor se puede escribir desde un servidor web hasta los scripts necesarios para soportar el servidor (Dayley, 2014) (Node JS Foundation , 2018) (Tilkov et al, 2010).

### **1.2.1.4 REST**

REST es una arquitectura de software desarrollada a partir de varias arquitecturas de red, que combinada con algunas restricciones adicionales define una interface de conexión uniforme. Esta arquitectura se enfatiza en la escalabilidad de los componentes, generalidad de las interfaces y un desarrollo independiente de componentes. Además, intenta minimizar la latencia y la comunicación de la red, al mismo tiempo que maximiza la independencia y la escalabilidad de las implementaciones de componentes. Esto se logra al colocar restricciones en la semántica del conector, donde otros estilos se han centrado en la semántica de los componentes (R. Fielding, 2000).

### **1.2.1.5 Esquema de autenticación básica:**

Este esquema, es definido en el RFC 2617 como un esquema de autenticación básico, basado en que el usuario debe identificarse a través de un usuario y una contraseña para cada dominio, en donde el servidor compara la igualdad de estos parámetros con los almacenados en él, por tanto el servidor sólo procesa las peticiones si pudo realizar la autenticación del usuario, es de aclarar que este esquema de autenticación no es seguro, debido a que el nombre de usuario y la contraseña viajan sin encriptar por la red (J. Franks et al, 1999).

### **1.2.1.6 Comunidad**

Aquel grupo de individuos que poseen una causa común, en el caso de la comunidad de un proyecto, son todos aquellos interesados en el mismo (Apache Foundation).

### **1.2.1.7 Web server**

Equipo computacional, comúnmente de altas prestaciones, dedicado a ofrecer servicios a los diferentes requerimientos vía redes y telecomunicaciones. Todo se logra procesando peticiones HTTP, las cuales, en la mayoría de casos espera respuestas. Un servidor web dentro de su lógica, implementa el protocolo HTTP, administra los recursos web y proporciona capacidades administrativas de servidor [10] [11].

### **1.2.1.8 Identificador**

Un identificador es el atributo de un objeto el cual incorpora toda la información necesaria, para distinguir este objeto de cualquier otro objeto que habite dentro del mismo espacio. Aunque se debe tener en cuenta, que un identificador en muchos casos no define la identidad del objeto que se desea distinguir (T. Berners-Lee, 2005).

### **1.2.1.9 HTTP**

El protocolo de transferencia de hipertexto es definido en el RFC 2616 de la siguiente manera: es un protocolo de petición/respuesta, el cual se ubica en la capa de aplicación para sistemas de información distribuidos, colaborativos e hipermedia. El protocolo HTTP ha sido usado por la iniciativa de información World Wide Web desde 1990. La primera versión de HTTP fue un simple protocolo encargado de transferir datos en bruto a través de Internet (R. Fielding , 1999).

#### **1.2.1.10 Source forge**

Es el proveedor almacenamiento de proyectos de código abierto más antiguo, fue lanzado en 1999, aloja más de 500000 proyectos y tiene más de 4 millones de descargas diarias (Source Forge).

#### **1.2.1.11 MySQL**

MySQL es un sistema de administración de base de datos, sus bases de datos son relacionales. Además, es software de código abierto, el cual tiene diferentes proyectos uno de estos es “MySQL database server”, el cual es rápido, confiable, escalable y fácil de usar, lo que lo convierte en una opción perfecta para ser usado en un sistema de información (Oracle, 2019).

#### **1.2.1.12 SQL**

El lenguaje estructurado de consultas o SQL, es un lenguaje de base de datos diseñado para administrar datos en DBMS relacionales y originalmente basado en álgebra relacional. Su alcance incluye consultas y actualizaciones de datos, creación y modificación de esquemas y control de acceso a datos. Este fue desarrollado por Donald D. Chamberlin y Raymond F. Boyce en la compañía IBM a comienzos de los años 70's, esta primera versión fue llamada SEQUEL (R. Elmasri et al, 2001).

#### **1.2.1.13 Query**

El término query o consulta es una jerga usada para referirse a todo tipo de interacción con una base de datos incluyendo la modificación de datos (H. Garcia-Molina, 2009).

#### **1.2.1.14 UUID**

Los UUID (Universally unique identifier) son una cadena de 16 octetos, la cual es un identificador único en el espacio tiempo, con respecto al espacio de todos los UUID. Los UUID tiene múltiples usos desde identificar objetos persistentes en una red, hasta identificar objetos de corta vida. La representación interna de un UUID es una secuencia de bits descrita en la sección 4 del rfc 4122 (International Telecommunication Union [ITU]) (P. Leach, 2005).

#### **1.2.1.15 Postman**

Es un ambiente de desarrollo de APIs, usado por los desarrolladores para realizar pruebas de funcionamiento de sus propias APIs y de esta forma agilizar sus procesos de desarrollo y prueba. Algunas de las funcionalidades de Postman son: enviar una petición HTTP, poder escribir pruebas en la plataforma, crear y correr 38 una colección de peticiones o pruebas, entre otras. Lo que hace de Postman una herramienta esencial a la hora de desarrollo APIs (Postman).

#### **1.2.1.16 TypeScript**

Es un lenguaje de programación creado con el propósito de satisfacer las necesidades de los equipos de desarrollo de JavaScript, gracias a que TypeScript permite la definición de interfaces entre componentes y usar la ya conocida programación orientada a objetos, permite dar orden y estructura a los proyectos de JavaScript. Debido a que TypeScript no es más que un mejoramiento sintáctico de JavaScript, todos los proyectos existentes de JavaScript también son un proyecto de TypeScript (G. Bierman, 2014) (Microsoft, 2019).

### **1.1.3 Problema**

Los directores del proyecto Unicenta no agregan las nuevas funcionalidades desarrolladas por la comunidad, por tanto, su crecimiento no es potenciado con la rapidez que la comunidad requiere. Adicionalmente, debido a que los proyectos de software libre no son patrocinados directamente por una persona o empresa en particular, y en su mayoría se distribuyen de forma gratuita, sin generar ganancia por el uso particular de este, necesitan de una comunidad de personas que tengan un ideario similar a las metas del proyecto, las cuales sean capaz de sostenerlo, ya sea con su conocimiento y/o tiempo.

## **2 METODOLOGÍA**

Debido a que el proyecto presenta fases de desarrollo, es necesario adoptar metodologías que faciliten la gestión del proceso del mismo, así pues, debido a la naturaleza de los módulos y el tiempo que se tenía para el desarrollo, se optó por realizar la combinación del modelo de desarrollo en espiral y el paradigma de prototipado, combinación que permite un desarrollo ágil y una elicitación de requisitos confiable. El resultado de esta combinación se especifica a continuación.

Las fases usadas en el desarrollo de los módulos fueron las siguientes:

**Planeación:** en esta fase se planea el contenido de la iteración y el cronograma de la misma, es de tener en cuenta, que, en el caso específico del proyecto, cada una de las iteraciones corresponde a un módulo.

**Modelado:** En esta fase se realiza el análisis y el diseño del desarrollo, debido a que en esta fase se realiza la elicitación de requisitos, se usa el paradigma de hacer prototipos para este objetivo. Los prototipos realizados son prototipos no funcionales y realizados con una herramienta que facilita su desarrollo. Gracias a que se usa el paradigma de prototipos la fase de modelado posee unas sub fases, las cuales son:

- **Comunicación:** Es esta fase el equipo se reúne para definir los objetivos e identificar requerimientos.
- **Plan rápido:** Se planea una iteración para realizar el prototipo
- **Modelado:** Se modela el prototipo en forma de diseño rápido
- **Construcción del prototipo:** Se construye el prototipo
- **Despliegue:** Se entrega el prototipo y es evaluado por los participantes, retroalimentando el equipo, con el fin de mejorar los requisitos

Ahora, ya teniendo los requisitos se realiza el proceso de diseño, haciendo un diagrama de casos de uso y un modelo relacional de la base de datos.

Construcción: Esta fase pertenece a la codificación de los módulos y a la realización de pruebas de los mismos. Para el desarrollo se usaron IDE's como Netbeans y VisualCode, cada uno de los cuales usado para la codificación en los diferentes lenguajes usados en los módulos. Además, se realizaron algunas pruebas unitarias a los módulos con el fin de verificar su funcionamiento.

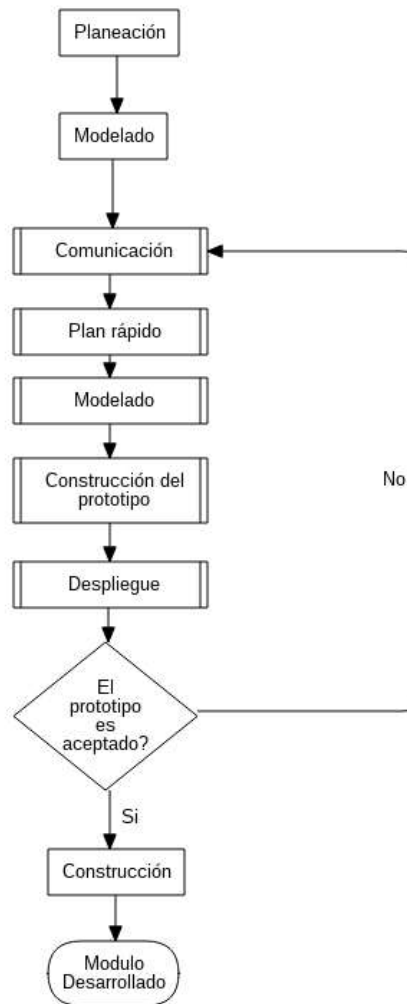


Fig. 1. Diagrama de flujo de la metodología.

### 3 RESULTADOS

El resultado final de todo este proceso es un proyecto derivado de Unicenta denominado OctoPOS, con dos nuevas funcionalidades, la sincronización con Woocommerce y una app móvil para la toma de pedidos, además de esto, un documento en donde se encuentra plasmado el proceso de ingeniería inversa realizada a Unicenta, el cual puede ser usado como una valiosa herramienta por cualquier desarrollador deseoso de contribuir a la comunidad.

Aunque Unicenta es un software maduro, con más de 7 años de trabajo, el mejoramiento del mismo difícilmente acabará, gracias a que se van creando nuevas necesidades en los usuarios. Gracias a esto se tomó la iniciativa del desarrollo.

Una parte fundamental para el desarrollo de estos módulos anteriormente mencionados, es entender el funcionamiento del software base, en este caso Unicenta oPos, para esto se realizó un proceso de ingeniería inversa el cual se dividió en dos partes, el entendimiento de la base de datos a través del análisis del modelo entidad relación de la base de datos de Unicenta y el entendimiento de su código fuente.

Para poder entender la base de datos del proyecto el primer paso realizado fue saber sobre que motor de base datos estaba funcionando el proyecto, debido a que al tener esta información se creaba una idea de cómo serían los tipos de datos utilizados en esta base de datos, las restricciones que tendría y la sintaxis SQL que se utiliza para su gestión, además es un dato esencial para comenzar la búsqueda de una herramienta que facilite el proceso de obtener el modelo ER de la base de datos del proyecto.

Al realizar la búsqueda de una herramienta para el propósito anteriormente mencionado, se eligió el software DBEaver para lleva a cabo el cometido.

Es de resaltar que el proyecto contiene más de 40 tablas, las cuales se encuentran en su tercera forma normal lo cual garantiza consistencia en los datos almacenados, además se puede observar que las llaves primarias de todas las tablas son de tipo varchar, esto se debe a que los desarrolladores optaron por usar el método de identificación UUID, ya que este método se adapta muy bien a la necesidad de identificación que se genera con el uso de los tickets de venta, los cuales tienen un tiempo de vida de corta duración.

Ya entendiendo la base de datos del proyecto se inició con el entendimiento de su código el cual fue escrito en el lenguaje de programación Java usando como entorno de desarrollo integrado Netbeans, así que el primer paso para poder entender el código del proyecto fue montar el código en dicha plataforma, después de esto se comenzó con un proceso de depuración del proyecto, el cual permitió seguir paso a paso los procesos realizados por el software, ver la interacción entre clases y los tipos de dato que manejaba el programa y de esta manera se fue construyendo un conocimiento frente a cómo funcionaba el proyecto.

Adicionalmente otro factor fundamental para entender la estructura del proyecto y su funcionamiento, fue el diagrama de clases del proyecto.

Ya conociendo el diagrama de clases, entendiendo los procesos y algoritmos del proyecto se generó la documentación de las clases más influyentes dentro del proyecto y que fueron necesarias para el desarrollo de los otros módulos. Esta documentación, como fue mencionado anteriormente, se generó por cada clase y consiste en: El diagrama de clases de la clase documentada, una descripción de la funcionalidad de la clase con sus respectivos parámetros de entrada y salida, y por último un fragmento del código de la clase como se puede apreciar a continuación.

Class SaveProvider: Esta clase se encarga de almacenar las acciones de los botones guardar, agregar y eliminar de cada una de las interfaces de gestión del sistema, su constructor es polimorfo pero el constructor que se usa con más frecuencia es *public SaveProvider(SentenceExec sentupdate, SentenceExec sentinsert, SentenceExec sentdelete)*, el cual recibe tres parámetros , de tipo *SentenceExec*, cada uno de los cuales representa la acción que tomará uno de los tres botones anteriormente mencionados. El diagrama de esta clase se puede observar en la Fig. 2.

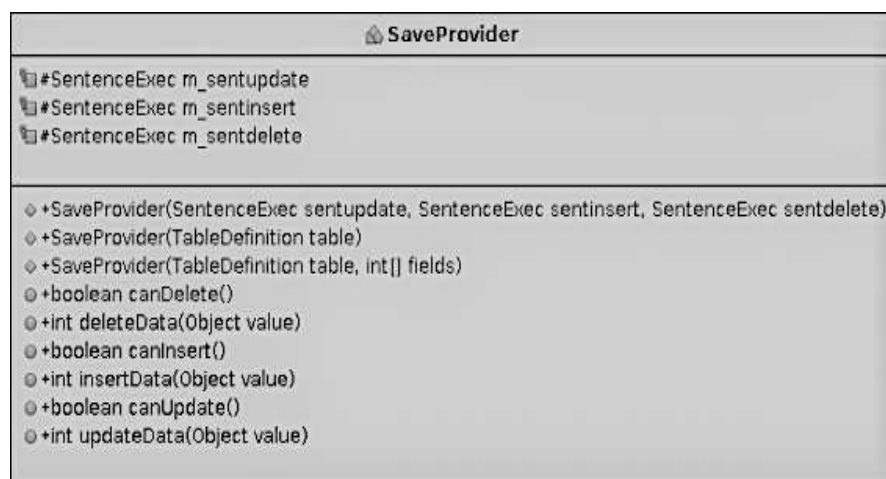


Fig. 2. Clase SaveProvider



```
public class SaveProvider {  
  
    protected SentenceExec m_sentupdate;  
    protected SentenceExec m_sentinsert;  
    protected SentenceExec m_sentdelete;  
    /** Creates a new instance of SavePrSentence  
     * @param sentupdate  
     * @param sentdelete  
     * @param sentinsert */  
    public SaveProvider(SentenceExec sentupdate, SentenceExec  
sentinsert, SentenceExec sentdelete) {  
        m_sentupdate = sentupdate;  
        m_sentinsert = sentinsert;  
        m_sentdelete = sentdelete;  
    }  
}
```

Fig. 3. Código: Clase SaveProvider

Adicionalmente se documentaron también los scripts SQL de la base de datos, la documentación de estos se realizó dando una descripción de lo que hace el script y un fragmento de código el cual se puede apreciar en la Fig. 4.

**Script MySQL-create.sql** Este script es el encargado de crear las tablas de la base de datos por primera vez, es decir, que, si se desea adicionar otra tabla al proyecto, el query debería de ir dentro de este archivo.

```
/* Header Line. Object: applications. Script date: 27/08/2015  
08:42:37. */  
CREATE TABLE `applications` (  
    `id` varchar(255) NOT NULL,  
    `name` varchar(255) NOT NULL,  
    `version` varchar(255) NOT NULL,  
    PRIMARY KEY ( `id` )  
) ENGINE = InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT = Compact;  
  
/* Header Line. Object: attribute. Script date: 27/08/2015  
08:42:37. */  
CREATE TABLE `attribute` (  
    `id` varchar(255) NOT NULL,  
    `name` varchar(255) NOT NULL,  
    PRIMARY KEY ( `id` )  
) ENGINE = InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT = Compact;
```

Fig. 4. Script: MySQL-create.sql

Ya habiendo realizado el proceso de ingeniería inversa se inició con el desarrollo del módulo de sincronización con un e-commerce en el cual se realizó la selección de cuál sería la plataforma de comercio electrónico con la que se realizará la sincronización, después de una revisión de las plataformas e-commerce que son distribuidas bajo una licencia de software libre se eligió la plataforma Woocommerce, la cual brinda una muy buena documentación para los desarrolladores, además, ya que es un plugin de Wordpress cuenta con el respaldo de la comunidad de Wordpress, la cual es bastante grande y activa en la red, lo que brinda un canal de soporte adicional a la documentación proporcionada por el proyecto.

También cabe resaltar que Woocommerce es la plataforma más usada para soportar tiendas virtuales, esta es usada por alrededor del 28% de todas las tiendas virtuales en internet (WooCommerce), con lo cual al hacer esta sincronización impactaría de manera positiva un gran sector de las ventas electrónicas.

Por estas razones anteriormente expuestas se seleccionó la plataforma Woocommerce como la candidata ideal para llevar a cabo la sincronización.

Adicionalmente se seleccionó también cual sería la plataforma usada como intermediaria entre Unicenta oPos y Woocommerce, y se llegó a la conclusión que se desarrollaría una API REST en NodeJS gracias a sus características, algunas de estas son: Es asíncrono y basado en eventos, es muy rápido, posee un solo hilo, pero es altamente escalable, su consumo de máquina es bajo, entre otras. Lo que convierte a NodeJS la herramienta ideal para realizar el desarrollo. Al tener seleccionadas las herramientas con las cuales se iba a trabajar, también se realizó una revisión de las metodologías de desarrollo existentes y con este conocimiento, se optó por realizar la combinación del modelo de desarrollo en espiral y el paradigma de prototipado, gracias a que esta combinación permite un desarrollo ágil y una elicitación de requisitos confiable. En este momento se dio inicio a la etapa de análisis y diseño del módulo, en este momento el conocimiento adquirido con la ingeniería inversa del proyecto y con la documentación proporcionada por la página de Woocommerce, jugaron un papel fundamental, ya que se debió realizar un proceso comparativo minucioso, entre las tablas de las bases de datos de ambos proyectos para de esta forma relacionarlas de manera correcta y eficiente, de forma que se mantuviera la integridad de los datos en ambas plataformas. Una vez realizado el análisis y diseño se procedió a la codificación del módulo, en la cual se desarrolló la REST API para que actuara como intermediaria, pero adicionalmente a esto, se debió desarrollar una serie de funciones en el código de Unicenta para permitir realizar las peticiones HTTP a los end-points de la REST API. Como evidencia de estos desarrollos, a continuación, se puede observar la documentación de una clase de cada uno de los proyectos (Unicenta – REST API).

Clase ProductAPI (Unicenta): Esta clase es la encargada de realizar las peticiones a la API. Todos los métodos de esta clase corresponden a un servicio de la API, los métodos de esta clase reciben como parámetro un String en donde viene el JSON con el cual se va a realizar la petición. El diagrama de esta clase se puede observar en la Fig. 5.

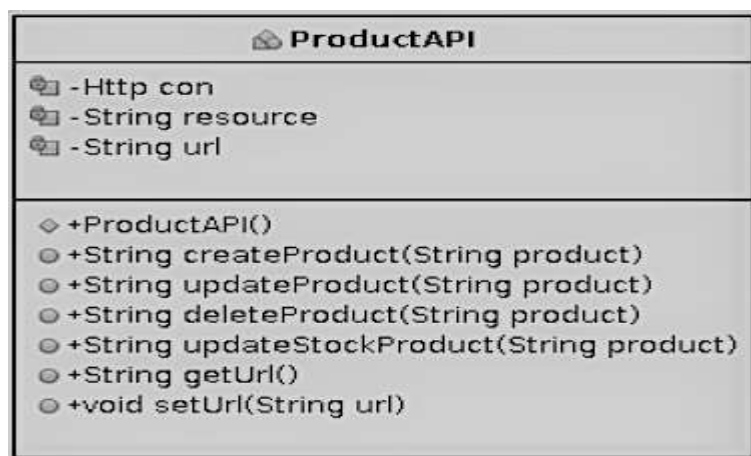


Fig. 5. Clase ProductAPI

Clase Routes (TypeScript): En esta clase se crean las rutas que va a aceptar la API, es decir, se exhiben los servicios a los clientes, para cada uno de estos servicios se define el método por el cual se va a acceder a él, el URL y cuál va a ser el proceso a realizar después de recibida la petición.

```
export class Routes {  
  
    public productController: ProductController = new ProductController();  
  
    public routes(app): void {  
  
        app.route('/')  
        .get((req: Request, res: Response) => {  
            res.status(200).send({  
                message: 'GET request successfull!!!!'  
            })  
        })  
  
        app.route('/product')  
        .post(this.productController.getProducts)  
  
        app.route('/product/read/:productId')  
        .post(this.productController.getProductWithID)  
  
        app.route('/product/create')  
        .post(this.productController.createProduct)  
  
        app.route('/product/update')  
        .post(this.productController.updateProduct)  
  
        app.route('/stock/update')  
        .post(this.productController.updateStockProduct)  
  
    }  
}
```

Fig. 6. Código: Clase ProductAPI

Después de realizar todo el desarrollo del módulo la interfaz resultante se puede apreciar en la Fig. 7.

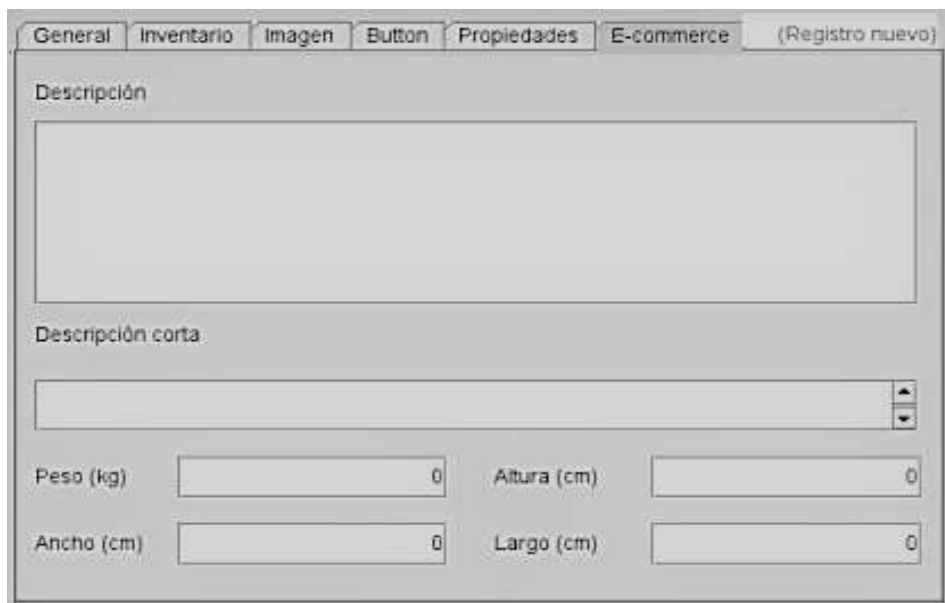


Fig. 7. Interfaz: Sincronización con WooCommerce

Posterior a esto se comenzó con el desarrollo de una aplicación para realizar pedidos desde dispositivos móviles con sistema operativo android, la cual tiene como propósito facilitar los procesos de toma de pedidos en restaurantes, cafés y bares.

La aplicación se limita a realizar pedidos y que estos queden registrados en la base de datos. Debido a que se tomó la decisión de desarrollar una aplicación híbrida, con el fin que la aplicación sea la base para un proyecto web, se optó por desarrollar el front-end en el framework de desarrollo para JavaScript, Angular ya que este tiene cierta compatibilidad con los dispositivos móviles a la hora de crear una aplicación basada en el mismo código. Para el back-end se desarrollaron nuevos end-points en la REST API desarrollada anteriormente, para crear la apk en base al código Angular se hizo uso de Phonegap, esta decisión fue tomada gracias a la facilidad que presenta esta plataforma y su buena compatibilidad con los dispositivos móviles. Como prueba del desarrollo del módulo se puede apreciar la documentación de las algunas clases tanto del front-end como del back-end y un pantallazo de una de las ventanas de la aplicación se puede observar en la Fig. 10.

Clase Routes (Back-end): Esta clase es la encargada de exponer los servicios de la API, para que puedan ser consumidos, en el siguiente código se puede apreciar los servicios que consume la aplicación para su funcionamiento.

```
import {Request, Response, NextFunction} from "express";
import { ProductController } from "../controllers/productController";
var cors = require('cors');
export class Routes {
    public productController: ProductController = new ProductController();
    public routes(app): void
        app.route('/product/get-catalog')
            .post(this.productController.getProductCatalog)
        app.route('/categories/get-root')
            .post(this.productController.getRootCategories)
        app.route('/categories/get-sub')
            .post(this.productController.getSubCategories)
        app.route('/sharedticket/get')
            .post(this.productController.getSharedTicket)
        app.route('/sharedticket/create')
```

Fig. 8. Clase Routes (Back-end)

Clase PlacesService (Front-end): Esta clase es el servicio que se encarga de realizar las peticiones a la API, relacionadas con los “lugares” de la aplicación, en el método *getPlaces*, se aprecia que usa el método “*consumeservice*”, el cual recibe como parámetro la URL del servicio de la API que se desea consumir y adicionalmente, pero no obligatorio, recibe los datos que sean necesarios para el consumo del servicio. Así como esta clase existen más las cuales permiten la interacción con la API.

```
@Injectable()
export class PlacesService extends BaseService {

  constructor(public http: HttpClient) {
    super(http);
  }

  getPlaces(floor: string) {
    try{
      return this.consumeService('/place/get',{floor});
    }
    catch (error) {
      console.log(error);
    }
  }
}
```

Fig. 9. Clase PlacesService (Front-end)



Fig. 10. Interfaz: Pedido de productos (App)

Cabe resaltar que a ambos módulos se le realizaron pruebas de caja negra y de caja blanca. Para realizar las pruebas a la REST API se usó Postman con el fin de facilitar y agilizar el proceso de pruebas, además en el desarrollo de la REST API se usó TypeScript, con el fin de darle orden y manejar un desarrollo con orientación a objetos, teniendo en cuenta igualmente que el framework Angular también trabaja con TypeScript.

Ya realizado el desarrollo de los módulos, se decidió bajo que licencia se iba a distribuir el software, así pues, se entró en el proceso de revisión de las licencias de software disponibles, con lo cual se encontró un amplio listado de licencias, debido a esto se procedió a clasificar las licencias, para de esta forma hacer el trabajo más fácil. Un factor fundamental en el momento de seleccionar la licencia era que el proyecto Unicenta está bajo la Licencia Pública General de GNU (GPL) versión 3, lo que limita la búsqueda a todas aquellas licencias compatibles con la licencia GPL versión 3, lo que significa que no pueden incumplir ninguna de las cláusulas de esta licencia. Adicional a esto como lo expresa Laurent, los factores que deben incluir la elección de una licencia son: que tan usada y conocida es la licencia, que tan comprensible es, y por ultimo cual es la filosofía de la licencia, particularmente su compatibilidad con otras (Free Software Foundation [FSF]). Una vez comparadas algunas de las licencias disponibles, lo que dio como resultado la tabla 1, e incluyendo los factores anteriormente mencionados y siguiendo el siguiente consejo de la Free Software Foundation, “*Cuando se contribuye a un proyecto existente, la versión modificada se debe publicar bajo la misma licencia que la obra original.*” (A. ST. LAURENT, 2016, p 174-178), se decidió que mantener la Licencia Publica General de GNU (GPL) versión 3 era la mejor elección.

Tabla. 1. Comparación de licencias de software libre.

Nombre	Autor	Compatible con GPL	Software libre
Licencia Pública General de GNU (GPL) versión 3	Free Software Foundation	X	X
Licencia Apache, versión 2.0	Apache Software Foundation	X	X
MIT license	MIT	X	X
Licencia Pública de Mozilla (MPL), versión 2.0	Mozilla Foundation	X	X
Licencia BSD Modificada	Regents of the University of California	X	X
Licencia BSD Original	Regents of the University of California		X
CeCILL-B versión 1	CEA-CNRS-INRIA		X
Licencia Pública de la Unión Europea (EURL) versión 1.1	European Commission		X
Academic Free License, todas las versiones hasta la 3.0	Lawrence E. Rosen		X
Apple Public Source License (APSL), versión 2	Apple Computer		X
Apple Public Source License (APSL), versión 1.x	Apple Computer		
Licencia Pública de AT&T	AT&T		
The JSON License	JSON.org		

Adicionalmente a esta decisión, se debió buscar el medio de divulgación del trabajo, para lo cual se seleccionaron las plataformas SourceForge y Gitlab.

#### 4 CONCLUSIONES

El proceso de desarrollar nuevos módulos para un software libre, requiere de tiempo y esfuerzo para entender su funcionamiento, el apoyo de la comunidad es fundamental durante todo el proceso, ya que esta se convierte en la fuente principal de información y conocimiento del tema, lo que permite mejor fluidez durante el desarrollo. Aunque el conocimiento de la comunidad es muy amplio, el proceso de ingeniería inversa, permite ahondar mucho más en el funcionamiento del programa, lo cual permite empoderarse más del proyecto para que de esta forma se facilite el desarrollo de los módulos.

En cuanto al desarrollo de los módulos se puede concluir que el uso de buenas prácticas de programación tales como la documentación, el manejo de versiones, entre otras; y adicionalmente la aplicación de la ingeniería del software y una buena planeación del proyecto, proporcionan unas herramientas y una organización al mismo, lo cual asegura el buen término del desarrollo. Así también la elección correcta de las herramientas a usar en el desarrollo permite agilizar los procesos, e igualmente llevar a buen término el desarrollo.

#### 5 REFERENCIAS BIBLIOGRAFICAS

Openbravo POS - Openbravo Forge. Retrieved from <http://centralrepository.openbravo.com/openbravo/org.openbravo.forge.ui/ForgeProjectDetail/openbravopos>.

Stallman, R. (2004). *Software libre para una sociedad libre* (1st ed., p. 45). Madrid: Traficantes de Sueños.

Manjavacas Santos, A. et al. (2016). *Adaptación e implementación de mejoras en un proyecto de software libre a un sistema cash-flow muy eficiente y seguro*. Madrid, España: Universidad Complutense de Madrid.

Dayley, B. (2014). *Node.js, MongoDB and AngularJS web development*. Upper Saddle River: Addison-Wesley.

Foundation, N. (2019). About | Node.js. Retrieved from <https://nodejs.org/en/about/>

Tilkov, S. et al. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, 14(6), 80-83. doi: 10.1109/mic.2010.145

Fielding (2000). Dissertation: CHAPTER 5: Representational State Transfer (REST). Retrieved from [https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)

Franks, J. et al. (1999). HTTP Authentication: Basic and Digest Access Authentication. Retrieved from <https://www.ietf.org/rfc/rfc2617.txt>

Glossary of Apache-Related Terms. (2017). Retrieved from <https://www.apache.org/foundation/glossary.html>

Totty, B. et al. (2009). *HTTP: The Definitive Guide*. Beijing: O'Reilly Media.

Berners-Lee, T. et al. (2005). Uniform Resource Identifier (URI): Generic Syntax. Retrieved from <https://www.ietf.org/rfc/rfc3986.txt>

Fielding, R. et al. (1999). RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1. Retrieved from <https://tools.ietf.org/html/rfc2616>

About. (2019). Retrieved from <https://sourceforge.net/about>

Oracle. (2019). *MySQL 8.0 Reference Manual*. [Ebook]. Retrieved from <https://downloads.mysql.com/docs/refman-8.0-en.pdf>

Elmasri, R. et al. (2011). *Fundamentals of database systems* (6th ed.). 1272 Seiten: Pearson Education.

Garcia-Molina, H. et al. (2009). *Database systems* (2nd ed.). New Jersey: : Pearson Education.

Universally Unique Identifiers (UUIDs). Retrieved from <https://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx>

Leach, P. et al. (2005). RFC 4122 - A Universally Unique Identifier (UUID) URN Namespace. Retrieved from <https://tools.ietf.org/html/rfc4122>

Postman API. Retrieved from [https://docs.api.getpostman.com/?\\_ga=2.62019052.1355013722.1538345190-1742112975.1538345190#intro](https://docs.api.getpostman.com/?_ga=2.62019052.1355013722.1538345190-1742112975.1538345190#intro)

Bierman, G. et al. (2014). Understanding TypeScript. *ECOOP 2014 – Object-Oriented Programming*, 257-281. doi: 10.1007/978-3-662-44202-9\_11

Microsoft. (2016). *TypeScript Language Specification* [Ebook]. Retrieved from <https://github.com/microsoft/TypeScript/blob/master/doc/TypeScript%20Language%20Specification.pdf>

WooCommerce - eCommerce for WordPress. Retrieved from <https://woocommerce.com>

Free Software Foundation . How to choose a license for your own work- GNU Project - Free Software Foundation. Retrieved from <https://www.gnu.org/licenses/license-recommendations.html>

ST. LAURENT, A. (2016). *Understanding open source and free software licensing* (pp. 174-178). SHROFF Publishers & DISTR.