UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Desarrollo de un módulo de un ERP de una tienda online dedicada a la venta de accesorios y recambios de moto

MEMORIA PRESENTADA POR:

Álvaro Llopis Samper

TUTOR/A:

Vicente Guerola Navarro

GRADO EN INGENIERÍA INFORMÁTICA

Convocatoria de defensa: *06/2021*

Resumen

Las tiendas online están cogiendo mucho protagonismo en estos últimos años debido a la pandemia, las tiendas físicas han sufrido mucho más en comparación con las digitales, por el simple hecho del confinamiento. El e-commerce se está renovando y cada año que pasa se pueden encontrar más y más tiendas online.

Toda empresa que se dedica a la venta online sabe que el paso de información entre departamentos de la empresa es primordial, y sobretodo saber cuando no hay stock de un producto, notificarlo y comprar al proveedor que, teniendo en cuenta distintas variables, salga más económico para la empresa, lo que se conoce como un ERP (entre otras funcionalidades).

Este trabajo está centrado en el mundo del motociclismo, ya que, la empresa en la que se va a desarrollar se llama Import Export Unobike, tienda de accesorios y recambios de moto.

Pues, esta empresa no tiene un ERP como tal, el paso de información se realiza a través de las observaciones y notificaciones que se escriben manualmente y todos los integrantes de la empresa tienen acceso a esa información, no es un mensaje instantáneo, ni objetivo para un departamento de la empresa en específico.

Pero el problema lo encuentra la empresa cuando no hay stock de un producto y tiene distintos proveedores, necesitan saber el mejor “postor” para comprar y si llegan a portes en el momento de realizar el pedido. Esta tarea la realiza una sola persona y le lleva mucho tiempo elegir el proveedor.

En este Trabajo de Fin de Grado, de Ingeniería Informática, se plantea una lógica automática que seguiría el pedido realizado por un cliente de un producto del cual no hay stock, en el que se escoge el mejor proveedor disponible y se realiza la compra en un momento del día, de acuerdo con las necesidades del proveedor, con la condición de que sea el más barato y el pedido llegue a portes gratuitos. Teniendo en cuenta que todos los proveedores que trabajan con esta empresa disponen de unos productos de muy buena calidad.

Para realizar este algoritmo, se ha utilizado un software libre conocido como XAMPP el cual utiliza servidor de apache, sistema de gestión de base de datos MySQL- MariaDB, e intérprete de lenguaje PHP, XAMPP permite realizar tus propias pruebas y comprobaciones vía web en servidor local.

Palabras clave

ERP

Venta Online
Automatización
XAMPP

Abstract

Online stores are taking on a lot of prominence in recent years due to the pandemic, physical stores have suffered much more compared to digital stores, simply because of the confinement. E-commerce is being renewed and every year you can find more and more online stores.

Every company engaged in online sales knows that the passage of information between departments of the company is paramount, and obviously know when there is no stock of a product, notify it and buy it from the supplier who, taking into account different variables, is cheaper for the company, which is known as an ERP, but it has other functionalities.

This work is focused on the world of motorcycling, because the company in which it is going to develop is called Import Export Unobike, motorcycle accessories and spare parts store.

Well, this company does not have an ERP as such, the passage of information is done through the observations and notifications that are written manually, and everyone in the company has access to that information, it's not an instant message, nor objective for a specific company department.

But the problem is found by the company when there is no stock of a product and it has different suppliers, they need to know the best "bidder" to buy and if they arrive at the time of placing the order. This task is performed by a single person and takes a long time to choose the provider.

In this End-of-Degree Work of Computer Engineering, there is an automatic logic that would follow the order placed by a customer of a product of which there is no stock, where the best available supplier is chosen and the purchase is made at one time of the dat, according to the needs of the supplier, provided that he is the cheapest and the order arrives at free shipping. Considering that all suppliers working with this company have very good quality products.

To perform this algorithm, a free software known as XAMPP has been used which uses Apache server, database management system MySQL-MariaDB and PHP language interpreter. XAMPP allows you to perform your tests and checks on the web on a local server.

Key words

ERP

Online sales
Automation
XAMPP

Resum

Les tendes en línia estan agafant molt protagonisme en aquests últims anys a causa de la pandèmia, les botigues físiques han patit molt més en comparació amb les digitals, pel simple fet del confinament. L'e-commerce s'està renovant i cada any que passa es poden trobar més i més botigues en línia.

Tota empresa que es dedica a la venda en línia sap que el pas d'informació entre departaments de l'empresa és primordial, i sobretot saber quan no hi ha estoc d'un producte, notificar-lo i comprar al proveïdor que, tenint en compte diferents variables, isca més econòmic per a l'empresa, la qual cosa es coneix com un ERP (a part d'altres funcionalitats).

Aquest treball està centrat en el món del motociclisme, ja que, l'empresa en la qual es desenvoluparà es diu Import Export Unobike, tenda d'accessoris i recanvis de moto.

Perquè, aquesta empresa no té un ERP com a tal, el pas d'informació es realitza a través de les observacions i notificacions que s'escriuen manualment i tots els integrants de l'empresa tenen accés a aqueixa informació, no és un missatge instantani, ni objectiu per a un departament de l'empresa en específic.

Però el problema el troba l'empresa quan no hi ha estoc d'un producte i té diferents proveïdors, necessiten saber el millor “postor” per a comprar-li i si arriben a ports en el moment de realitzar la comanda. Aquesta tasca la realitza una sola persona i li porta molt temps triar el proveïdor.

En aquest Treball de Fi de Grau, d'Enginyeria Informàtica, es planteja una lògica automàtica que seguiria la comanda realitzada per un client d'un producte del qual no hi ha estoc, en el qual es tria el millor proveïdor disponible i es realitza la compra en un moment del dia, d'acord amb les necessitats del proveïdor, amb la condició que siga el més barat i la comanda arribe a ports gratuïts. Tenint en compte que tots els proveïdors que treballen amb aquesta empresa disposen d'uns productes de molt bona qualitat.

Per a realitzar aquest algorisme, s'ha utilitzat un programari lliure conegut com XAMPP el qual utilitza servidor d'apatxe, sistema de gestió de base de dades MySQL-MariaDB, i intèrpret de llenguatge PHP, XAMPP permet realitzar les teues pròpies proves i comprovacions via web en servidor local.

Paraules clau

ERP

Venta en línia
Automatització
XAMPP

Tabla de Contenidos

Tabla de Contenido

Introducción **6**
1.1 Motivación 7
1.2 Objetivos 7
1.3 Estructura 8

Situación actual **9**
2.1 Tecnología y medios digitales 10
2.2 Módulos más destacados actualmente 11
2.3 Módulo Pedidos a Proveedor introducción 12
2.4 Antecedentes 12
2.4.1 ERP “ODOO” 12

Especificación de requisitos **13**
3.1 Diagrama de flujo 13
3.2 Descripción del diagrama de flujo 14
3.3 Componentes de desarrollo del módulo 16

Desarrollo XAMPP **17**
4.1 Base de datos phpMyAdmin 17

Desarrollo de la lógica **19**
5.1 Constructor 21
5.2 Stock Unobike 21
5.3 Diferentes proveedores 23
5.4 Pedidos al mismo proveedor 24
5.5 Comprobación de portes 25
5.6 Tiempo de envío 26
5.7 Mejor precio 26
5.8 Talla y color del producto 27
5.9 Marca y proveedor del producto 28
5.10 Convertir en un único array 29
5.11 Lógica completa 30

AUTOMATIZACIÓN **31**
6.1 CURL 31
6.2 PROGRAMADOR DE TAREAS 32

CONCLUSIONES Y MEJORAS FUTURAS **34**
7.1 Aspectos a mejorar 34

Bibliografía **35**

Tabla de Figuras

Figura 1:Organigrama Empresarial, Fuente: Elaboración propia *6*

Figura 2:Instantánea de Tienda Unobike, Fuente: Búsqueda de Google *10*

Figura 3:Diagrama de Flujo, Fuente: Elaboración propia *13*

Figura 4:Conexiones BBDD, Fuente: Elaboración propia *17*

Figura 5:Constructor, Fuente: Elaboración propia *20*

Figura 6:Comprobación de stock en Unobike, Fuente: Elaboración propia *21*

Figura 7:Comprobación de distintos proveedores, Fuente: Elaboración propia *22*

Figura 8:Lista de pedidos a proveedor, Fuente: Elaboración propia *23*

Figura 9:Comprobación de portes, Fuente: Elaboración propia *24*

Figura 10:Servicio del proveedor, Fuente: Elaboración propia *25*

Figura 11:Mejor Precio, Fuente: Elaboración propia *26*

Figura 12:Talla y color del producto, Fuente: Elaboración propia *27*

Figura 13:Marca y proveedor del producto, Fuente: Elaboración propia *27*

Figura 14:Conversión a un único array, Fuente: Ejemplo de php.net *28*

Figura 15:Lógica completa, Fuente:Elaboración propia *29*

Figura 16:Instalación y uso de Curl, Fuente: Elaboración propia *30*

Figura 17:Formato de tiempo de Crontab, Fuente: Crontab.guru *30*

Figura 18:Ejecución automática, Fuente: Elaboración propia *31*

Figura 19:Acción de programación de tareas, Fuente: Elaboración propia *31*

Figura 20:Desencadenante de programación de tareas, Fuente: Elaboración propia *32*

Figura 21:JSON de detalles de proveedor, Fuente: Elaboración propia *32*

Figura 22:PHPMailer, Fuente: Estructura usada por UnoBike *33*

Figura 23:Automatización de envíos, Fuentes: Elaboración propia *34*

1. Introducción

En este Trabajo de Fin de Grado se va a colaborar con una empresa dedicada a la venta de accesorios y recambios de moto de forma online, para desarrollar un módulo de ERP el cual realiza pedidos a proveedores. Se pretende que el pedido realizado por el cliente que no está en stock en la empresa, se demore lo menos posible y el cliente lo tenga en su casa lo antes posible. Para ello, se va automatizar el proceso de selección del proveedor con el mejor precio y que se llegue a portes gratuitos a la hora de realizar el pedido.

La empresa que va dirigido este trabajo, se fundó en el 2006 y está compuesta por un almacén y una oficina y ha ido obteniendo más reputación y cada vez más marcas trabajan con ellos. Se compone de la siguiente forma:

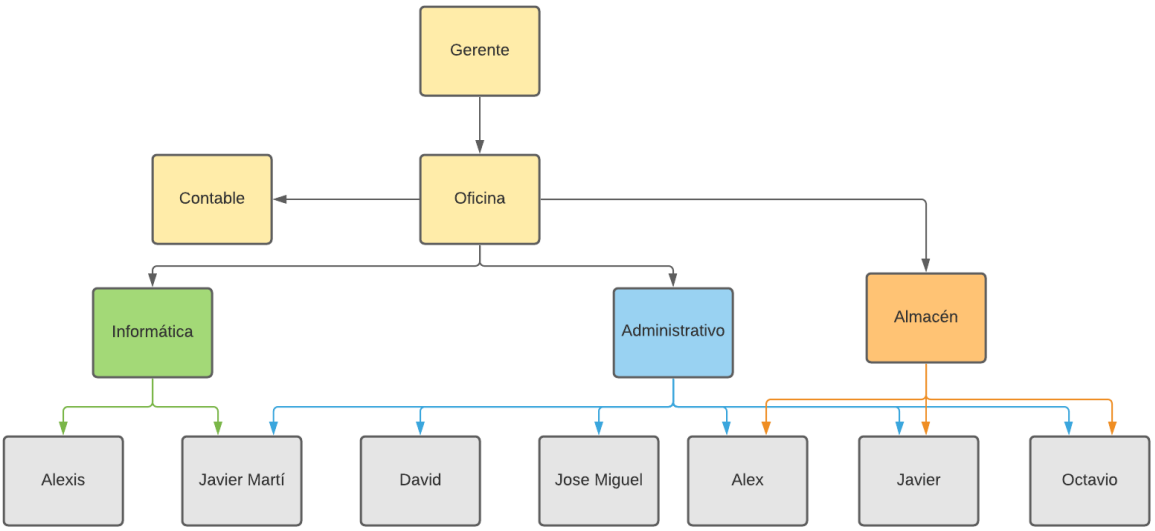


Figura 1: Organigrama empresarial
Fuente: Elaboración propia

Como se puede observar prácticamente todos los empleados pueden acceder a la misma información, además que en la misma página web, a no ser que tenga un rol distintivo, todos tienen acceso y pueden modificarla.

Desde un inicio el negocio ha sido digital, es decir, que todas las ventas se realizaban a través de su página web. Por lo tanto la finalidad de los informáticos de la empresa es hacer la web cada vez más atractiva y manejable para que los clientes encuentren los productos que necesitan rápidamente.

El objetivo principal del gerente, aparte de llegar a más gente organizando campañas de marketing y ofreciendo los productos más novedosos en su tienda, es la fidelización de los clientes. Para ello busca que los clientes tengan la mejor experiencia a la hora de comprar en su tienda y que vuelvan a comprar en el futuro.

Por eso lo más importante para la empresa es conseguir los productos que no tienen stock que han sido pedidos por algún cliente lo antes posible. Este trabajo lo realiza el gerente manualmente, de todos los pedidos que no tienen stock, agrupa los que son del mismo proveedor y realiza el pedido a este, en el momento en el que se llega a portes gratuitos dentro del margen de tiempo que utiliza el proveedor para realizar pedidos. Por ejemplo en Givi(Fábrica de accesorios de motos) se pueden realizar pedidos hasta las 15:00 porque envían los productos esa misma tarde, si el pedido se realiza con retraso, el envío será pospuesto a la tarde del día siguiente, lo que supone una demora en la entrega del pedido de cara al cliente.

1.1 Motivación

El motivo por el cual se ha decidido realizar este trabajo es la importancia que están teniendo las empresas digitales y de venta online actualmente, ya que, la digitalización de las empresas está a la orden del día, debido a la facilidad de los clientes de acceder a los artículos a través de la web de la empresa. La competencia entre distintas empresas del mismo sector viene dada por la diferencia de precios y por el tiempo de entrega de los productos. Es por eso que la necesidad de tener stock de todos los productos que los clientes compran es primordial.

1.2 Objetivos

La finalidad de este Trabajo de Fin de Grado (TFG) es proporcionar a la empresa un módulo que automatice los pedidos a proveedor de los productos que no se encuentran en stock, reduciendo al máximo el tiempo dedicado específicamente a esta tarea.

- Identificar los pedidos de clientes que no están en stock.
- Tener acceso al precio de los productos de los proveedores.
- Facilitar el proceso de agrupar pedidos a un mismo proveedor.
- Considerar los términos de los proveedores: hora máxima para realizar el pedido y cantidad necesaria para llegar a portes gratuitos.
- Conocer en todo momento el estado del pedido. “Enviado”, “En reparto”, “Recibido” y si ha habido alguna incidencia que suponga un retraso en la recepción del pedido.

- Proponer una solución sencilla de bajo coste, que solicite pedidos al mejor proveedor.

1.3 Estructura

Capítulo 1. Introducción

Se trata de una introducción, donde se detalla los objetivos y los motivos de porque se va a realizar este trabajo.

Capítulo 2. Situación actual

En este segundo capítulo se va a hablar de la situación que se encontraba la empresa en el momento de realizar este trabajo, qué tecnologías y medios utilizaban y cómo estaban conectadas.

Capítulo 3. Especificación de requisitos

A través de diagramas se van a definir en el tercer capítulo los requisitos funcionales necesarios para la realización del módulo.

Capítulo 4. Desarrollo XAMPP

Este capítulo se va a centrar en la preparación del entorno de trabajo y los componentes que se van a utilizar para llevar a cabo la implementación del módulo.

Capítulo 5. Desarrollo de la lógica

En el capítulo 5 se mostrará cómo ha sido la programación y qué importancia ha tenido el desempeño de otros compañeros para que el módulo funcione perfectamente.

Capítulo 6. Automatización

En este sexto capítulo se va a centrar en la forma de automatizar la ejecución del módulo desarrollado.

Capítulo 7. Conclusiones y mejoras futuras

Se revisarán los objetivos previstos y se comprobará el grado de cumpliemiento de los mismos, así como, aspectos que podrían mejorarse en el futuro

Capítulo 8. Bibliografía

Contiene las fuentes consultadas para la elaboración de este Trabajo de Fin de Grado.

2. Situación actual

La empresa se encuentra en un estado bastante avanzado en lo que sería la implementación de un ERP ad hoc. Dispone de un sistema de gestión de pedidos de cliente y gestión de almacenamiento, pero la información se encuentra descentralizada y poco accesible. Sin embargo, cualquier integrante de la plantilla puede acceder a ella.

Es poco accesible porque si un cliente realiza consultas por distintos medios, es decir, whatsapp, email, facebook, msn y llamada telefónica; no están protocolizadas las respuestas, por lo que se pregunta constantemente en la oficina que se le ha dicho por email o por whatsapp, para decirle lo mismo.

El gerente ya está trabajando en suplir estas carencias protocolarias, pero le consume mucho trabajo ir pedido a pedido que no se tiene en stock y añadirlo al carrito de cada proveedor para su posterior compra.

La empresa además de la web, cuenta con espacios sociales en Instagram, Facebook y Whatsapp, con más de 700 interacciones diarias entre las tres plataformas. En Instagram cuentan con más de 2.400 seguidores y en Facebook han superado los 4.000. Si comparamos con su competencia más directa NilMoto los números son bastante parecidos.

Actualmente intentan mejorar la posición de la página con respecto a distintas marcas y en búsquedas de google. Para ello una persona está dedicada al “Search Engine Optimization” (SEO) de la empresa, analizando estadísticas y optimizando rutas de búsqueda para que el usuario tenga a su disposición los productos más buscados.

Esto ha llevado a que se vendan más de 100 productos al día y que las críticas de los clientes sean muy positivas y cuentan con una nota media de 4,8 sobre 5 en internet. Los únicos que ponen una reseña negativa es porque su pedido no se encontraba en stock y se tuvo que pedir a proveedor y tardó en recibir su pedido más de lo esperado, pero a favor de la web se deja claro tanto las condiciones de envío como los plazos de entrega de cada producto.

Figura 2: Instantánea de tienda Unobike
Fuente: Búsqueda por Google

2.1 Tecnología y medios digitales

Toda la web la tienen subida a un proyecto en GitLab en el que pueden gestionar tanto la página de la empresa como la visibilidad que tiene y planificar tareas y proyectos dentro de la web. Con Git a través de comandos Bash van subiendo todas las actualizaciones a GitLab.

La programación se realiza por medio de un paquete XAMPP el cual hace uso de servidor web Apache, un sistema de gestión de bases de datos MySQL e intérprete de comandos PHP y Perl, siempre utilizando un sistema operativo Linux, Kubuntu 18.04.

El servidor web está conectado a un editor de código fuente como puede ser Visual Studio Code, el cual permite programar en JavaScript, CSS y PHP además de Java, C++, C#, etc.

El propio XAMPP hace uso de phpMyAdmin para permitir la creación y modificación de las bases de datos. Cualquier modificación y/o actualización, la prueban antes en un servidor local/localhost para evitar errores y cuando esté correcta la suben al GitLab para que se integre con el resto de la web.

Cada uno de los trabajadores tiene su propio ordenador en la oficina, la mayoría con sistema operativo Windows, menos los programadores que utilizan Linux.

Para atención al cliente han conseguido distintas vías de comunicación, como pueden ser las llamadas telefónicas que atienden a través de una centralita la cual están todos conectados, por whatsapp el cual se encarga una persona y que contesta por orden de entrada, e interacciones y mensajes por Facebook o Instagram, en donde son remitidos la mayoría de veces a Whatsapp o atendidos por telefono.

A pesar de ser solamente una oficina con un almacén bastante grande, la gente local de Alcoy y de ciudades cercanas se acercan al almacén para realizar su pedido desde la misma oficina o para recoger el paquete y evitar portes. Pero el pedido se realiza siempre a través de la página web, así que no deja de ser una empresa digital.

2.2 Módulos más destacados actualmente

No puede considerarse que la empresa tenga un ERP, porque realmente no tienen ninguno aplicado, pero las funciones que cumplen sus trabajadores y el software implementado cumple perfectamente muchos aspectos de uno.

Las funcionalidades del “ERP” de la empresa se pueden agrupar en:

- Gestión financiera, un contable se encarga de todo una vez por semana:
    - Contabilidad
    - Tesorería
    - Conciliaciones Bancarias
    - Activos Fijos

- Ventas y marketing, una persona se encarga de hacer todas las campañas de marketing, entre otras cosas hace de Community Manager:
    - Creación de campañas de márketing
    - Seguimiento activo de los clientes
    - Uso de redes sociales

- Almacén el cual se encuentra limitado pero bien organizado:
    - Ubicación de existencias
    - Picking de inventario
    - Ventas diarias de stockaje

- Servicios de atención al cliente:
    - Gestión de llamadas
    - Consultas via whatsapp o email
    - Seguimiento de incidencias de envío

- Recursos Humanos con la finalidad de gestionar información de los trabajadores:
    - Registros de empleados
    - Informes y nóminas
    - Seguimiento de días laborales

- Mantenimiento de la web:
    - Actualizaciónes de la web
    - Incorporación de nuevas marcas y productos
    - Prevención de fallos informáticos

2.3 Módulo Pedidos a Proveedor introducción

Para desarrollar los Pedidos a Proveedor en el ERP de la empresa se va a ser fiel a la programación ya utilizada en otros módulos, es decir, con la misma metodología y en el mismo ámbito.

Para ello se ha instalado un paquete XAMPP (X sistema operativo, Apache, MySQL, PHP, Perl), que es la principal característica de conexión de bases de datos y servidor, en un sistema operativo LINUX, porque la instalación del servidor Apache y de base de datos da menos problema de compatibilidad.

Se ha escogido XAMPP porque se trata de un software libre y las conexiones del servidor con la BBDD se hacen automáticamente al instalarse, podría haberse usado LAMPP o WAMP que se tratan de paquetes bastante similares. LAMP es compatible con sistemas operativos tipo LINUX y WAMP compatible con Windows.

Una vez conectada la conexión del servidor con el editor de código fuente (Visual Code), se ha realizado una copia de la base de datos de la empresa en phpMyAdmin para poder hacer consultas durante la programación de la lógica del módulo.

2.4 Antecedentes

2.4.1 ERP “ODOO”

Como ya hemos visto, en un ERP este módulo ya se encontraría implementado y realizar pedidos a proveedor estaría a un click del administrador. Por ejemplo ODOO se trata de un software ERP integrado, desarrollado por una empresa belga llamada “odoo”.

Este software obviamente cumple con todos los objetivos de un ERP y la conexión con el proveedor es una de ellas. Simplemente con crear al proveedor dentro del software relacionarlo con sus productos y añadirle variables de hora de realización de pedido y coste mínimo para portes gratuitos, cuando un pedido pase el filtro de almacén y no se encuentre en stock se advierte al administrador y se crea el pedido al proveedor, pero es el administrador quien tiene que aceptar el pedido.

El motivo por el que se desarrolla este módulo en específico es replicar la funcionalidad que tendría en un ERP pero que el pedido se realice automático a una hora determinada para cada proveedor.

3. Especificación de requisitos

En este capítulo veremos los requisitos para cumplir con las necesidades de los proveedores.

3.1 Diagrama de flujo

Mediante el diagrama de flujo se va a definir la lógica que va a seguir un pedido en el momento de ser recibido.

Para la realización del diagrama se ha usado Lucidchart que es una herramienta muy útil para realizar este tipo de diagramas.

Los dos finales posibles son:

- El pedido se encuentra en stock y se envía o no se encuentra a stock, se recibe del proveedor y se envía

- El pedido no se encuentra en stock y ningún proveedor tiene stock del producto, se propone una alternativa al cliente.

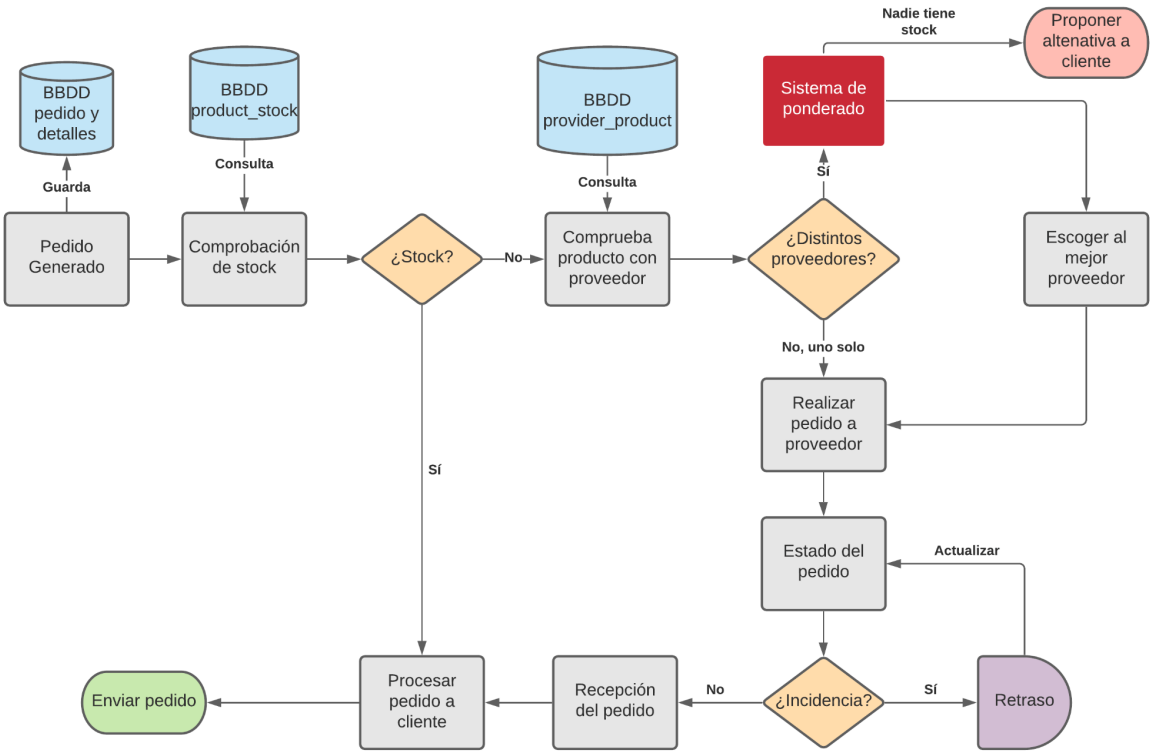
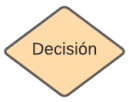


Figura 3: Diagrama de Flujo
Fuente: Elaboración propia

3.2 Descripción del diagrama de flujo

A continuación se va a describir las formas y el color del diagrama y posteriormente se va seguir el flujo explicando cada tarea.

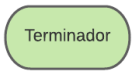


El rombo se trata de una decisión de carácter booleano, es decir, como respuestas Sí o No, pero puede haber excepciones.

El cilindro significa que se ha realizado una consulta o se ha guardado información en la BBDD.

El cuadrado hace referencia al proceso o tarea que se tiene que realizar en ese momento. Podemos encontrar algún cuadrado de un rojo intenso, indicando que es una tarea realmente importante.

Esta especie de D se utiliza cuando se produce algún retraso o incidencia que supone un tiempo de espera.



Esta forma ovalada se usa para indicar que se ha terminado el flujo, si el final es positivo para el usuario es verde y si es negativo será rojo.

Ahora se va a analizar paso a paso los procesos del diagrama de flujo.

Pedido Generado: Llega un pedido de un cliente y se guarda en la base de datos toda la información del cliente: nombre, teléfono, email, dirección, etc. También se guardan los detalles del pedido, como los productos y su precio.

Comprobación de stock: Automáticamente se realiza una consulta a la BBDD para comprobar si existe stock de los productos solicitados por el cliente. Si hay stock el flujo pasa directamente al procesado del pedido del cliente, si no hay stock continuamos con el siguiente proceso.

Comprueba el producto con el proveedor: Este proceso hace una consulta a la BBDD para saber si para cada producto solicitado existe más de un proveedor disponible. En el caso de que solo exista un proveedor en la BBDD de ese producto, se realizaría el pedido al proveedor; sin embargo si existen más proveedores de un mismo producto, se valorará cuál es el proveedor más rentable.

Sistema de ponderado: En este proceso encontramos el momento más crítico y más importante de la lógica a seguir, porque esta tarea determinará quién es el proveedor

más rentable analizando distintas variables como: precio, tiempo de entrega, portes y si hay más productos de ese proveedor para pedir. En el caso de que ningún proveedor tenga stock disponible del producto en cuestión, se le ofrecerá al cliente una alternativa y tendrá que realizar otro pedido de ese producto por lo que el flujo terminaría.

Escoger al mejor proveedor: Una vez realizado el sistema de ponderado y obtenido unos resultados, se escoge al mejor proveedor.

Realizar pedido al proveedor: En el momento en el que el pedido al proveedor llegue a portes, justo a la hora establecida por el proveedor, se realizará el pedido.

Estado del pedido: En este proceso se va a realizar un seguimiento del pedido, para comprobar que no se produce ninguna incidencia, en caso de producirse una, supondría un retraso en la entrega por lo que se actualizará el estado del pedido.

Recepción del pedido: Una vez pasado el tiempo de espera de entrega del producto por parte del proveedor, el pedido es enviado al departamento de almacén.

Procesar pedido al cliente: Los encargados de almacén preparan el envío y esa misma tarde sale de la empresa con dirección al domicilio o puesto de recogida administrado por el cliente al realizar el pedido.

La única intervención manual durante el flujo se encuentra en la generación del pedido, donde el encargado de realizar el pedido es el propio cliente o si lo ha realizado por teléfono el personal de atención al cliente.

Seguidamente el resto de tareas se ejecutan automáticamente hasta el estado del pedido, que es a medias. Porque el estado se actualiza automáticamente por el sistema, pero el personal de administración estará atento del estado por si ha habido una incidencia , o si ha sido recibido en la empresa para enviar un email al cliente y avisar en cualquier caso.

Las dos últimas tareas son completamente tareas manuales en las que este sistema ya no interviene.

3.3 Componentes de desarrollo del módulo

Antes de comenzar con los componentes de nuestro módulo, hay que hacer hincapié en que el módulo primeramente se va a desarrollar en una copia instantánea de la web de un momento dado, para evitar errores o problemas en la web durante el desarrollo del módulo.

Para la realización del módulo se van a usar los siguientes componentes de XAMPP, ya comentados pero que se van a describir con más detalle:

- Servidor → Apache servidor libre y gratuito que conecta el servidor web de Chrome, mozilla y otros navegadores con el cliente.

- BBDD → MySQL/MariaDB cuenta con un sistema de gestión de BBDD bastante sencillo y con una estructura que puede modificarse por local a través de phpMyAdmin o por líneas de comando en PHP una vez conectados todos los servicios.

- Servidor Web → En este caso el servidor web va a ser localhost para evitar problemas en la web.

- Entorno de desarrollo PHP y CSS→ Visual Studio Code, herramienta de programación libre de Visual Studio, el cual cuenta con compatibilidad con numerosos lenguajes de programación.

Para el correcto funcionamiento del módulo deben estar conectados y funcionando todos los servicios.

4. Desarrollo XAMPP

4.1 Base de datos phpMyAdmin

Para poder entender el funcionamiento de la empresa y cómo debe adaptarse el módulo a esta, hay que familiarizarse con la construcción de la BBDD, al tenerla montada en el phpMyAdmin, se va a analizar las tablas que se utilizarán tanto para consultar como para guardar información durante el proceso de pedidos a proveedores.

Encontramos una BBDD muy extensa, con 85 tablas, no se me permite hacer capturas de las tablas debido a un contrato que se firmó al trabajar con la empresa de protección de datos, pero sí que puedo comentarlas por encima.

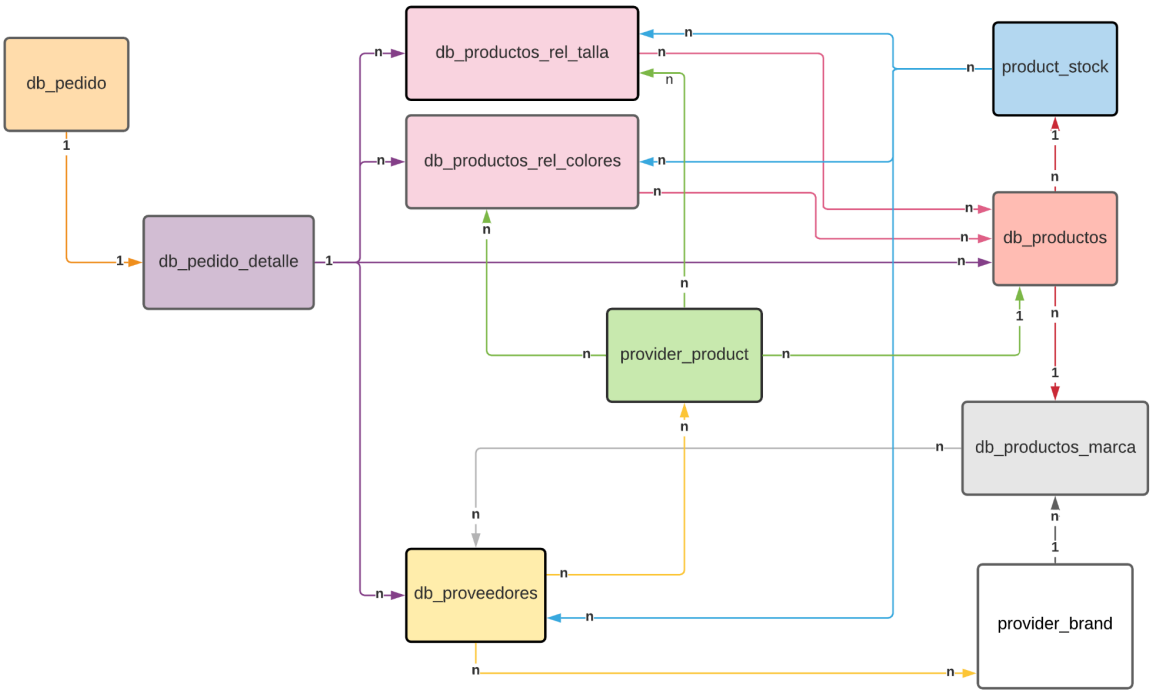


Figura 4: Conexiones BBDD
Fuente: Elaboración propia

Esta imagen representa las tablas que van a ser consultadas durante el desarrollo del trabajo y como están conectadas entre ellas. No he podido realizar una demostración de cómo serían las conexiones con sus atributos de todas las tablas de la BBDD, porque visualmente sería imposible de entender algo, ya que, se trata de una empresa en la que la información se encuentra muy repartida entre las distintas tablas.

En la representación anterior se muestran las conexiones entre tablas, de 1 si se trata de uno solo o “n” si es 1 o muchos. Por ejemplo, de un db_pedido se genera un db_pedido_detalle y a su vez un db_pedido_detalle hace referencia a un db_pedido, sin embargo un db_pedido_detalle puede hacer referencia a 1 o muchos db_productos, mientras que 1 o muchos productos hacen referencia a 1 solo pedido detalle.

A continuación se van a desarrollar los atributos de las distintas tablas que se han usado en el desarrollo del módulo:

db_pedido → Contiene 57 columnas, en las que se guarda toda la información del cliente y del pedido. Las columnas que más se van a consultar en este trabajo son las siguientes:

- npedido : Se trata del identificador del pedido.
- idmarca: Distingue la marca del producto.
- idmodelo: Disguinte el modelo dentro de la propia marca.
- total : Precio por el cual se vendió al cliente.
- dscr: Discriminatorio que determina el estado del pedido.

db_pedidos_detalle → Cuenta con 21 columnas y especifica los datos del pedido, las columnas más importantes son las siguientes:

- npedido: Identificador del pedido y enlaza con npedido de la tabla db_pedidos.
- idtalla: Si se trata de alguna prenda identifica la talla.
- idcolor: Si el producto permite distintos colores indica el color seleccionado por el cliente.
- product_id: Identificador del producto que enlaza con la tabla db_productos.
- provider_id: Indica el proveedor que se une con la tabla db_proveedores.

db_productos → Con 59 columnas describe toda la información del producto. En el trabajo nos vamos a centrar en las siguientes columnas:

- id: Identificador del producto.
- idmarca: Indica la marca del producto y es clave ajena de la tabla db_productos_marcas.
- precio: precio del producto.

db_productos_marcas → Con 34 columnas pero las realmente importantes son las siguientes:

- idmarca: identificador de la marca.
- idproveedor: Clave ajena de la tabla db_proveedores e indentifica al proveedor

db_proveedores → Contiene 13 columnas que describen todos los datos almacenados del proveedor.

- codproveedor: Identificador del proveedor.
- email: indica el email del proveedor.

db_productos_rel_colores → Se trata de una tabla de 10 columnas que solamente sirve para conocer el color.

- idcolor → Identificador del color.
- product_id → Se trata de la clave ajena de la tabla db_productos

db_productos_rel_tallas → Como la tabla anterior con 10 tablas y también sirve únicamente para relacionar la talla con el producto.

- idtalla: Indica la talla del producto.
- product_id: También es la clave ajena de la tabla db_productos.

product_stock → Con 10 columnas y aporta información sobre el almacén de la empresa y contiene cuatro claves ajenas:

- id: Identificador del producto en la tabla.
- product_id: Identifica el producto con la tabla db_productos.

- provider_id: Une el proveedor con la tabla db_proveedores.
- size_id: Inidica la talla con la tabla db_productos_rel_tallas.
- colour_id: Indica el color con la tabla db_productos_rel_colores.
- stock: Da a conocer el stockaje del producto en la empresa.
- price: Precio del producto

provider_brand → Esta tabla consta de 7 columnas y principalmente une los proveedores con las marcas. Puede que un proveedor tenga más de una marca, únicamente nos centraremos en las claves ajenas.

- provider_id: Id del proveedor de la tabla db_proveedores.
- brand_id: Indica la marca de la tabla db_productos_marcas.

provider_product → Tabla que tenía pendiente de rellenar un informático de la empresa, el cual tenía que hacer una subida masiva de productos enlazados con proveedor, esta tarea se completó al final del trabajo, dando como resultado 10 columnas de las cuales las que se utilizan son:

- product_id: Indica el producto con la tabla db_productos.
- provider_id: Identifica el proveedor con la tabla db_proveedores.
- price_cost: El precio de compra del producto que ofrece el proveedor.
- stock: Stockaje del proveedor respecto a dicho producto.
- size_id: Si tiene identificador de talla se une con la tabla db_productos_rel_tallas.
- color_id: Si tiene identificador de color se une con la tabla db_productos_rel_colores.

Las relaciones entre las tablas son muy liosas pero es imprescindible conocerlas para realizar el trabajo sin problemas. Una vez entendido el funcionamiento de la empresa y las conexiones de los servidores establecidas, se ha programado la lógica del módulo.

5. Desarrollo de la lógica

Como ya se ha comentado la lógica se ha desarrollado en Visual Studio Code, editor redefinido y optimizado para construir webs, que permite la programación en múltiples lenguajes.

El código ha sido desarrollado en PHP y al contar con innumerables variables a tener en cuenta, la estructura if/else está presente constantemente. También al trabajar con arrays multidimensionales, la estructura foreach es frecuente.

Antes de empezar a describir el código me gustaría comentar algunas funciones que se utilizan constantemente en el desarrollo de la lógica que fueron programadas por los informáticos de la empresa para hacer más sencillas las consultas y que la devolución sea un array ya preparado. Estas funciones son las siguientes:

- fetchAll → Permite realizar una consulta que devuelve como resultado todos los atributos de la tabla consultados en un array multidimensional.

- fetchOne → Devuelve solamente un atributo de la tabla consultada.
- fetchRow → Solamente devuelve como resultado un array simple con los atributos consultados de la tabla.

No siempre encontramos todos los datos necesarios en las tablas, al ser programada y desarrollada desde cero manualmente, hay valores muy importantes para esta lógica que se encuentran en null o están vacíos y hay que conseguirlos a través de otras tablas. Podrían surgir problemas en un futuro si no se arreglan las tablas o se centraliza toda la información. Es por eso que hay que estar preguntando constantemente si el valor existe en la tabla con la función empty, si no se usa esta función las probabilidades de que te de errores por null son bastante altas.

Para que la información del módulo pueda ser consultada en cualquier momento, se ha estructurado en forma de funciones que devuelven un valor booleano 1 o 0 si cumplen o no las distintas condiciones. De esta forma para comprobar que funciona todo correctamente al final se crea una instancia de esta clase y se pueden realizar cualquier consulta a través de sus funciones. Más adelante se mostrará como se comprueba el código. De momento se va a ir función a función explicando su funcionamiento y si se usan nuevas estructuras de programación.

El motivo por el cual toda la lógica está separada por funciones que devuelven valores booleanos, o lista en algún caso, es por petición expresa del jefe de informáticos de la empresa, quiere tener la opción de consultar cualquier valor sin la necesidad de realizar todos los cálculos cada vez. Es por eso que vamos a encontrar funciones bastante simples que indican una sola característica del producto como por ejemplo veremos más adelante las funciones “tallaProducto” o “colorProducto”.

5.1 Constructor

```
function __construct(){

    $this->orders=fetchAll("SELECT db_pedidos_detalle.*
                            FROM db_pedidos
                            LEFT JOIN db_pedidos_detalle
                            ON db_pedidos.npedido = db_pedidos_detalle.npedido
                            WHERE db_pedidos.fecha BETWEEN'2021-02-11 10:00:00' AND '2021-02-12 10:00:00'
                            ORDER BY number DESC
                            LIMIT 20");
}
```

Figura 5: Constructor
Fuente: Elaboración propia

Al trabajar con una instantánea de la web y de la base de datos, no se podía acceder a los pedidos realizados en el momento, así que, en un constructor se ha creado un array “orders” donde se encuentran los pedidos de las últimas 24h desde la entrada del último pedido de la instantánea, es decir, en el array se encontrarán todos los pedidos realizados desde las 10:00 del 11 de febrero hasta las 10:00 del 12.

Para hacer las comprobaciones más rápido y llevaderas, se encuentran en orden descendente y se ha limitado el array en 20. En la función final los valores del día se

obtendrán dinámicamente a través de la función date().

5.2 Stock Unobike

```php
function stockUnobike($product_id){
    $stock = fetchAll("SELECT * FROM product_stock WHERE product_id =".$product_id);
    if(!empty($stock)){
        foreach($this->orders as $details){
            if($details['product_id']==$product_id){
                foreach($stock as $clave){
                    if(!empty($details['idtalla'])){
                        if($clave['size_id']==$details['idtalla']){
                            if(!empty($details['idcolor'])){
                                if($clave['colour_id']==$details['idcolor']){
                                    if(!empty($clave['stock'])){
                                        return 1;
                                    }else{
                                        return 0;
                                    }
                                }
                            }if(!empty($clave['stock'])){
                                return 1;
                            }else{
                                return 0;
                            }
                        }
                    }else if(!empty($details['idcolor'])){
                        if($clave['colour_id'==$details['idcolor']]){
                            if(!empty($clave['stock'])){
                                return 1;
                            }else{
                                return 0;
                            }
                        }
                    }else{
                        if(!empty($clave['stock'])){
                            return 1;
                        }else{
                            return 0;
                        }
                    }
                }
            }
        }
    }
}
```

Figura 6: Comprobación de stock en Unobike
Fuente: Elaboración propia

La siguiente función es“stockUnobike” y se le pasa como parámetro el identificador del producto. Esta función te indica si se encuentra el producto en stock dependiendo de la talla y el color, para ello, se guarda en un array todos los productos en stock independientemente de su color y talla. Para posteriormente compararlas y comprobar si es el color o la talla que el cliente ha pedido. En el caso de que el producto solo tenga un color

y no sea una prenda, es decir, que no tenga distintas tallas ni colores, solamente se comprueba si está en stock.

Esta función devuelve 1 si se encuentra en stock y 0 en caso contrario. Si devuelve 1 seguiría con el procesado del pedido para su posterior envío, pero en caso de devolver 0 se continúa con la lógica.

5.3 Diferentes proveedores

```php
function diferentProvider($product_id){
    $proveedores =[];
    $providersPorMarca =[];
    if(!$this->stockUnobike($product_id)){
        foreach($this->orders as $details){
            if($details['product_id']==$product_id){
                if(!empty($details['provider_id'])){
                    $marcas = fetchRow("SELECT id, idmarca FROM db_productos WHERE id=".$product_id);
                    $providersPorMarca = fetchAll ("SELECT provider_id, brand_id
                                                    FROM provider_brand WHERE brand_id=".$marcas['idmarca']);
                    foreach($providersPorMarca as $clave){
                        $proveedores []= $clave['provider_id'];
                    }
                    if(!empty($proveedores[1])){
                        return 1;
                    }else{
                        return 0;
                    }
                }else{
                    $marcas = fetchRow("SELECT id, idmarca FROM db_productos WHERE id =".$product_id);
                    $providersPorMarca = fetchAll("SELECT provider_id
                                                    FROM provider_brand WHERE brand_id=".$marcas['idmarca']);
                    foreach($providersPorMarca as $clave){
                        $proveedores[] = $clave['provider_id'];
                    }
                    if(!empty($proveedores[1])){
                        return 1;
                    }else{
                        return 0;
                    }
                }
            }
        }
    }else{
       return 0;
    }
}
```

Figura 7: Comprobación de diferentes proveedores
Fuente: Elaboración propia

Esta función es “diferentProvider” y también se le pasa como parámetro el identificador del producto. Esta función comprueba el proveedor de todos los productos que dieron como resultado 0 en la función anterior.

Para comprobarlo se saca la marca del producto y los proveedores de la marca se guardan en un array, si el array solamente contiene un valor devuelve 0, y si contiene más

valores devuelve 1 . Si el producto se encontraba en stock se devuelve 0 y no se hace la comprobación de proveedores.

5.4 Pedidos al mismo proveedor

```
function listaPedidoProveedor($provider_id){
    $lista = [];
    $a= true;
    foreach($this->orders as $details){
        if(!empty($details['product_id'])){
            if(!empty($details['provider_id'])){
                if($details['provider_id']==$provider_id){
                    if($this->orderSameProvider($provider_id)){
                        if($a){
                            $producto = array($details['product_id']);
                            $lista = [$provider_id=> $producto];
                            $a=False;
                        }else{
                            array_push($producto,$details['product_id']);
                            $lista =[$provider_id=> $producto];
                        }
                    }else{
                        $lista = [$provider_id => $details['product_id']];
                    }
                }
            }else{
                $proveedor = $this->proveedorMarca($details['product_id']);
                foreach($proveedor as $provs){
                    if($provs['provider_id']==$provider_id){
                        if($this->orderSameProvider($provider_id)){
                            if($a){
                                $producto = array($details['product_id']);
                                $lista = [$provider_id=> $producto];
                                $a=False;
                            }else{
                                array_push($producto,$details['product_id']);
                                $lista =[$provider_id=> $producto];
                            }
                        }else{
                            $lista = [$provider_id => $details['product_id']];
                        }
                    }
                }
            }
        }
    }
return $lista;
}
```

Figura 8: Lista de pedidos a proveedor
Fuente: Elaboración propia

En esta función se hace uso de un nuevo recurso informático como son los semáforos, el cual indican cuándo se puede acceder a ciertas partes del código y cuándo no. En este caso la variable “$a” será el semáforo que siempre será positivo hasta que durante la ejecución de las siguientes líneas de código cambie de valor y bloquee parte del

código para el resto de la ejecución. De esta forma se mantiene la variable “lista” limpia y ordenada.

Para devolver el listado de todos los proveedores que tienen pedidos solicitados, se ha comprobado si en los datos de la tabla existe tanto el identificador del producto como del proveedor. En el caso de que no exista el id del proveedor, que es muy frecuente, su identificador se consigue a través de la marca del producto.

La función devuelve una lista con los pedidos que se han solicitado a ese proveedor. Se muestra incluso cuando solamente tiene un pedido. Con el array_push lo que se consigue es que el siguiente producto del mismo proveedor se inserte al array de producto como un nuevo índice, es una forma de mantener el array multidimensional como índice el identificador del producto y dentro de cada índice las características de cada pedido.

5.5 Comprobación de portes

```php
function portesComprobation($provider_id, $portes){
    $comp =[];
    foreach($this->orders as $details){
        if(!empty($details['provider_id'])){
            if($details['provider_id']==$provider_id){
                $precio = fetchRow("SELECT price FROM product_stock WHERE product_id =". $details['product_id']);
                $repeat =false;
                for($i=0;$i<count($comp);$i++){
                    if($comp[$i]['provider_id'] == $provider_id){
                        $comp[$i]['precio']+=$precio['price'];
                        $repeat = true;
                        break;
                    }
                }
                if($repeat==false){
                    $comp[]=array('provider_id'=>$details['provider_id'], 'precio'=>$details['precio']);
                }
            }
        }else{
            $marca = fetchRow("SELECT idmarca FROM db_productos WHERE id =".$details['product_id']);
            $proveedorMarca = fetchRow("SELECT provider_id, brand_id FROM provider_brand WHERE brand_id =". $marca['idmarca']);
            if($proveedorMarca['provider_id']==$provider_id){
                if(!empty($details['product_id'])){
                    if(!$this->stockUnobike($details['product_id'])){
                        $precio = fetchRow("SELECT price_cost FROM la_tabla_que_necesito WHERE product_id =". $details['product_id']);
                        $repeat =false;
                    }else{
                        $precio = fetchRow("SELECT price FROM product_stock WHERE product_id =". $details['product_id']);
                        $repeat =false;
                    }
                }
                for($i=0;$i<count($comp);$i++){
                    if($comp[$i]['provider_id'] == $provider_id){
                        $comp[$i]['precio']+=$precio['price_cost'];
                        $repeat = true;
                        break;
                    }
                }
                if($repeat==false){
                    $comp[]=array('provider_id'=>$proveedorMarca['provider_id'], 'precio'=>$precio['price_cost']);
                }
            }
        }
    }
    foreach($comp as $clave){
        if($clave['precio']>=$portes){
            return 1;
        }else{
            return 0;
        }
    }
}
```

Figura 9: Comprobación de portes
Fuente: Elaboración propia

Para la ejecución correcta de esta función llamada “portesComprobation” se necesita el id del proveedor y los portes del mismo. Para conocer si se llega a portes primero se necesita la ejecución de anteriores funciones por ejemplo de si se encuentra en stock y si son del mismo proveedor. Una vez conocemos esa información y está guardada en variables, se suma el precio de los productos del mismo proveedor y se compara con “portes”, que como hemos nombrado es un atributo de la función.

5.6 Tiempo de envío

```php
function providerService($provider_id){
    foreach($this->orders as $details){
        if(!empty($details['product_id'])){
            if(!$this->stockUnobike($details['product_id'])){
                if(!empty($details['provider_id'])){
                    if($details['provider_id']== $provider_id){
                        $service = fetchRow("SELECT * FROM la_tabla_que_necesito
                                             WHERE provider_id =". $details['provider_id']);
                        if($service['delivery_days']<=5){
                            return 1;
                        }else{
                            return 0;
                        }
                    }
                }
            }
        }
    }
}
```

Figura 10: Servicio del proveedor
Fuente: Elaboración propia

Para conocer el tiempo de envío que va a llevar el pedido que le hagamos al proveedor se realiza una consulta a “la_tabla_que_necesito” es la que más adelante se convertirá en “provider_product”. De esta tabla se consulta los días de envío del proveedor y se devuelve 1 si es menor o igual a 5 días, o devolverá 0 si es mayor.

5.7 Mejor precio

```php
function bestPrice($product_id){
    $proveedores = [];
    $precio = [];
    $precioDetalles=[];
    foreach($this->orders as $details){
        if(!empty($details['product_id'])){
            if($details['product_id']==$product_id){
                if(!$this->stockUnobike($product_id)){
                    if($this->diferentProvider($product_id)){
                        $provs = fetchAll("SELECT provider_id
                                FROM la_tabla_que_necesito WHERE product_id =". $product_id);
                        foreach($provs as $clave){
                            $proveedores[]= ['provider_id' =>$clave['provider_id']];
                        }
                    }else{
                        $bestPrice = fetchOne ("SELECT provider_id
                                        FROM la_tabla_que_necesito WHERE product_id =". $product_id);
                        return $bestPrice;
                    }
                }
            }
        }
    }
    foreach($proveedores as $valor){
        $precioDetalles = fetchRow("SELECT price_cost
                        FROM la_tabla_que_necesito
                        WHERE provider_id =" . $valor['provider_id'] . "AND product_id = $product_id" );
        $precio [] = $precioDetalles['price_cost'];
    }
    $mejor=min($precio);
    $bestPrice = fetchOne("SELECT provider_id FROM la_tabla_que_necesito
                WHERE product_id =". $product_id ." AND price_cost =". $mejor);
    return $bestPrice;
}
```

Figura 11: Mejor precio
Fuente: Elaboración propia

Se trata de una de las funciones más importantes de la lógica, específicamente la más importante dentro del sistema de ponderación. Para calcular el mejor precio, se necesita conocer también el resultado de otras funciones y a partir de ahí conocer el mejor precio del producto de “la_tabla_que_necesito” como ya se ha comentado se trata de la tabla“provider_product”.

Esta función devuelve el proveedor que tiene el mejor precio de ese producto, siempre y cuando tenga más de un proveedor y no se encuentre en stock, si no se encuentra en stock y solo hay un proveedor del producto, el precio dictado por este será el resultado.

5.8 Talla y color del producto

```php
function tallaProducto($product_id){
    foreach($this->orders as $details){
        if(!empty($details['product_id'])){
            if($details['product_id'] == $product_id){
                if(!empty($details['idtalla'])){
                    return $details['idtalla'];
                }
            }
        }
    }
}

function colorProducto($product_id){
    foreach($this->orders as $details){
        if(!empty($details['product_id'])){
            if($details['product_id']==$product_id){
                if(!empty($details['idcolor'])){
                    return $details['idcolor'];
                }
            }
        }
    }
}
```

Figura 12: Talla y color del producto
Fuente: Elaboración propia

Simples funciones que permiten la consulta de información sobre la talla o sobre el color del producto. Estas funciones no eran necesarias pero el jefe de informáticos de la empresa pidió expresamente que se pudiese consultar cualquier dato del producto.

5.9 Marca y proveedor del producto

```php
function marcaProducto($product_id){

    $marca = fetchOne("SELECT idmarca
            FROM db_productos WHERE id=". $product_id);
    return $marca;
}

function proveedorMarca($product_id){
    $proveedor = fetchAll("SELECT provider_id
                FROM provider_brand WHERE brand_id=". $this->marcaProducto($product_id));
    return $proveedor;
}
```

Figura 13: Marca y proveedor del producto
Fuente: Elaboración propia

Al igual que las dos anteriores, estas funciones solamente detallan una información del producto.

5.10 Convertir en un único array

```php
function super_unique($array)
{
  $result = array_map("unserialize", array_unique(array_map("serialize", $array)));

  foreach ($result as $key => $value)
  {
    if ( is_array($value) )
    {
      $result[$key] = $this->super_unique($value);
    }
  }

  return $result;
}
```

Figura 14: Conversión a un único array
Fuente: Ejemplo de php.net

La intención de esta función es como su nombre indica convertir los arrays en uno solo, se usará solamente al final de la lógica. Para unir en un único array todos los pedidos realizados a todos los proveedores. Este código es el único que no se trata de una elaboración propia. Buscando funcionalidades que pudiesen servir para el desarrollo del módulo en la api de php, me encontré con la estructura array_unique y en uno de los ejemplos que mostraban para emplearlo, unificaban multi arrays que era justo lo que yo necesitaba.

Se trata de una iteración de la función siempre y cuando se encuentren más valores dentro del foreach. Los valores se van guardando en cada iteración en un único array que es el que finalmente se devuelve como resultado.

5.11 Lógica completa

```
function logica(){
    $portes = 400;
    $ponderado = null;
    $final = [];
    $to = 'rubiallopis2@gmail.com';
    foreach($this->orders as $details){
        if(!empty($details['product_id'])){
            $proveedor = $this->proveedorMarca($details['product_id']);
            $ponderado = 0;
            if(!$this->stockUnobike($details['product_id'])){
                if(!$this->diferentProvider($details['product_id'])){
                    if(!empty($details['provider_id'])){
                        //$this->sendMail($to, $details['product_id'], $details['provider_id']);//proveedor
                        $final[] = $this->listaPedidoProveedor($details['provider_id']);
                    }else{
                        foreach($proveedor as $provi){
                            //$this->sendMail($to, $details['product_id'], $prov['provider_id']);
                            $final[] =  $this->listaPedidoProveedor($provi['provider_id']);
                        }
                    }
                }else{
                    foreach($proveedor as $provs){
                        $prov = $provs['provider_id'];
                        if(empty($details['provider_id'])){
                            if($this->bestPrice($details['product_id'])==$prov){
                                $ponderado += 10;
                            }
                            if($this->orderSameProvider($prov)){
                                $ponderado += 3;
                            }
                            if($this->portesComprobation($prov, $portes)){
                                $ponderado += 5;
                            }
                            if($this->providerService($prov)){
                                $ponderado += 2;
                            }
                            if($ponderado >= 10){
                                $final[] = $this->listaPedidoProveedor($prov);
                            }
                        }
                    }
                }
            }

        }
    }
    $res = $this->super_unique($final);
    print_r($res);
    exit();
}
```

Figura 15: Lógica completa
Fuente: Elaboración propia

Función que ejecuta la lógica completa utilizando las funciones creadas anteriormente. El código que se muestra es el que se ejecuta en la fase de pruebas del código el cual todavía mantiene unos portes seleccionados manualmente y la variable “$to” que sería el proveedor en cuestión, es un correo creado para realizar las pruebas de envío.

6. AUTOMATIZACIÓN

Al no tratarse de una aplicación convertir a este módulo en automático es realmente sencillo. Encontramos distintas formas de hacerlo para distintos sistemas operativos

6.1 CURL

La automatización se ha llevado a cabo a través de la herramienta “curl”, el cual se tiene que instalar previamente en el equipo si se trata de sistema operativo tipo Linux, Windows ya lo tiene instalado por defecto.

```
sudo apt install curl
curl -I https://crontab.guru
crontab -e
crontab -l
```

Figura 16:Instalación y uso de Curl
Fuente: Elaboración propia

Para instalar curl en tu sistema Linux se hace uso de la figura sudo para realizarlo como administrador. Posteriormente se enlaza curl con la web la cual vamos a utilizar para que nos reconozca la hora a la que queremos ejecutar el programa.

La línea de código “crontab -e” sirve para poder editar el crontab. Nos aparecerá como un nano simple con los elementos ya incluidos que necesita para ejecutarse automáticamente y solamente tenemos que añadir al final la hora con el formato que nos muestra la siguiente imagen.

Figura 17: Formato de tiempo de crontab
Fuente: Crontab.guru

Seguidamente introducimos la ruta del archivo que queremos que se ejecute a esa hora por lo que en un documento distinto al módulo marcaremos lo siguiente.

```php
<?php
include('adm_provider_order_logic.php');

$timer = new ProviderOrderLogic;

$timer->logica();
```

Figura 18: Ejecución automática
Fuente: Elaboración propia

Este nuevo fichero llamado, “adm_provider_order_timer.php” solamente sirve para ejecutar la función de lógica del módulo para ello se ha creado una instancia de la clase de “adm_provider_order_logic.php” para tener acceso a la ejecución de la función. En este momento obtenemos una lista con todos los proveedores con su propia lista de pedidos para posteriormente, valga la redundancia, pedirselos.

6.2 PROGRAMADOR DE TAREAS

En el momento de rellenar esta memoria se está utilizando Windows y en este sistema operativo encontramos otra forma de automatizar las tareas.

Desde el buscador de Windows buscamos “Programador de tareas”, en él indicamos el nombre de la tarea y se le puede añadir una descripción. Posteriormente en la pestaña de acciones añadimos la ruta del documento que queremos ejecutar. Como vemos en la siguiente imagen.

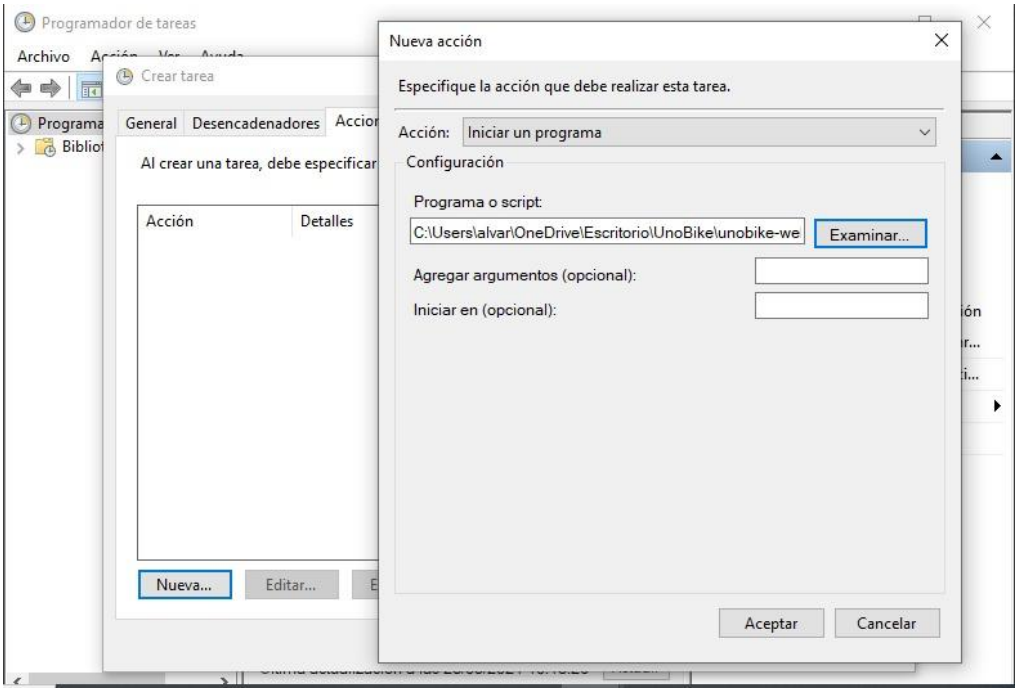


Figura 19: Acción de programador de tareas
Fuente: Elaboración propia

Seguidamente en la pestaña de “Desencadenadores” añadimos los términos de ejecución, es decir, cuando y cuantas veces. Nosotros queremos que se ejecute a las 10 todos los días. De este modo se crea una ejecución automática desde windows.

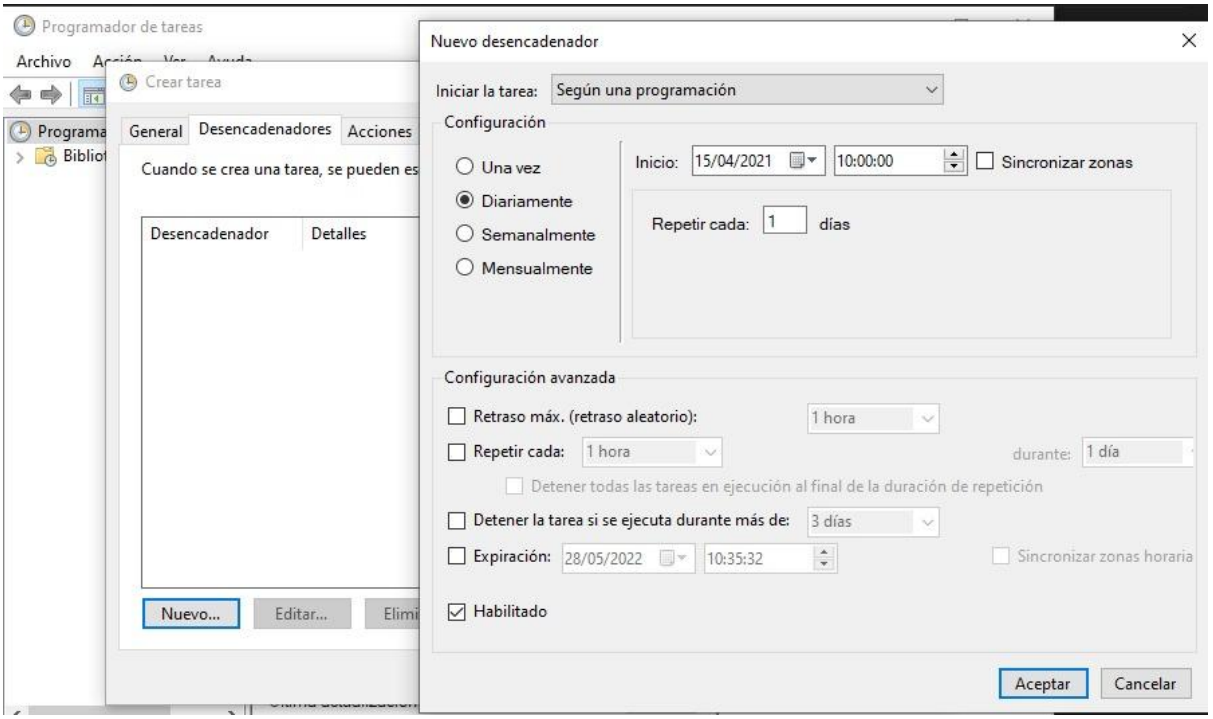


Figura 20: Desencadenante de programador de tareas
Fuente: Elaboración propia

6.3 Envío automático a proveedor

Para automatizar los envíos a los proveedores se ha hecho uso de un recurso que usan mucho en la empresa que es JSON, que convierte en una cadena de caracteres (String) cualquier array con formato compatible. En un documento nuevo se ha creado un array con los proveedores, su email, portes y horarios. Ese array se ha codificado con json_encode y guardado en la tabla “db_config” en la BBDD de unobike.

```
foreach($lista as $key => $provider_id){
    $mail=fetchOne("SELECT email FROM db_proveedores WHERE codproveedor =". $provider_id);
    $hora= fetchOne("SELECT hora FROM provider_product WHERE provider_id =". $provider_id);

    $array [$provider_id]=['email'=>$mail, 'hora'=>$hora];


}
$jsonProvider = json_encode($array);

$values = ['cod'=>'detallesProveedor', 'dato'=>$jsonProvider];
executeInsert("db_config",$values);
```

Figura 21: JSON de detalles de Proveedor
Fuente: Elaboración propia

Como esta información es estática y muy pocas veces va tener alguna actualización, es mejor mantenerla en una tabla accesible, a tener que ejecutar las consultas constantemente. Si estos datos de los proveedores no fuesen constantes no habría más remedio que consultarlos cada vez. Es por eso que se usa una estructura como JSON para guardar las referencias.

Para acceder a esta información consultamos la tabla para sacar el valor con fetchOne y se le hace un json_decode añadiendo un true como parámetro para que devuelva un array bien formado y poder tratar con él. De esta forma podemos consultar todos los portes y horarios de los distintos proveedores y en el archivo "adm_provider_order_time.php" se automatiza el envío.

El envío necesita de una función extra llamada "sendMail", en esta función como nos encontramos en periodo de pruebas, los emails se envían desde el correo creado anteriormente, a través del PHPMailer se realiza el envío.

```php
function sendMail($to, $product, $provider){//de momento envia correo a un particular

    $mail = new PHPMailer();
    $mail -> isSMTP();

    //servidor mail
    $mail->SMTPDebug = 0;
    $mail->From = "rubiallopis2@gmail.com";//remitente
    $mail->FromName = "Unobike";
    $mail->SMTPAuth = true;
    $mail->SMTPSecure = 'tls';//seguridad
    $mail->Host = "smtp.gmail.com"; //servidor smtp
    $mail->Port = 587; //puerto
    $mail->Username = "rubiallopis2@gmail.com";//usuario
    $mail->Password = 'Monosilaba1';//contraseña

    //mensaje al destinatario
    $mail->AddAddress($to);
    $mail->Subject='Prueba de mail';
    $mail->Body='Pedido del producto '. $product . ' a el proveedor '. $provider;
    $mail->WordWrap = 50;
    $mail->IsHTML(true);

    if($mail->Send()){
        echo 'Enviado correctamente';
    }else{
        echo 'No enviado, intentalo de nuevo crack';
    }
}
```

Figura 22: PHPMailer
Fuente: Estructura usada por UnoBike

Para la ejecución de la función se necesita de 3 atributos, "$to" que es el email del proveedor, "$product" que es el listado de productos a pedir y "$provider" que es el identificador. Y en la figura 21 se ve el formato típico que PHPMailer necesita para que el envío se produzca sin problemas.

```php
$datos = fetchOne("SELECT dato FROM db_config WHERE cod = detallesProveedor");

$arrayDatos = json_decode($datos,true);

foreach($arrayDatos as $clave => $proveedor){
    foreach($lista as $key => $provider_id){
        if($provider_id == $proveedor){
            sendMail($clave['email'], $key ,$proveedor);
        }
    }
}
```

Figura 23: Automatización de envíos
Fuente: Elaboración propia

Tanto la figura 22 como la 23 están dentro del fichero “adm_provider_order_time.php” y son los que se ejecutan a una hora concretada anteriormente en el Curl o programador de tareas para realizar el pedido al proveedor.

7. CONCLUSIONES Y MEJORAS FUTURAS

En este punto se va a ir objetivo a objetivo para comprobar si se cumplen y posteriormente proponer unas mejoras del código o de forma de trabajo en la empresa.

El primer objetivo era identificar el stock de los productos en la empresa, es decir, de un pedido saber si existe o no en el almacén. Este punto se cumple con creces con las consultas en la base de datos que se realizan de forma automática, en el que si se encuentra en stock el pedido pasa a ser facturado y enviado al cliente. En caso contrario continuaría con la lógica del módulo desarrollado.

El segundo objetivo se trataba de tener acceso al precio del producto al que lo vende el proveedor. Al subir y tener acceso a la tabla “provider_product” ya se cuenta con el precio de los productos de cada proveedor.

Siempre se sigue un protocolo para realizar los pedidos a los proveedores por lo que conocer su horario de pedidos ha resultado realmente sencillo generando un crontab para ejecutar el pedido una vez al día siempre a la misma hora.

Para conocer el estado del pedido se ha incorporado nuevos discriminatorios que identifican si el pedido se encuentra “Enviado” en “Reparto” o “Recibido”.

El último punto se cumple con creces porque es la lógica entera, una solución de bajo coste, ya que solamente se trata de consultas a la BBDD y operaciones simples, que se finaliza realizando el pedido al proveedor.

7.1 Aspectos a mejorar

La empresa entiende la importancia de implementar un ERP y un CRM en la empresa, porque es verdad que la información se encuentra descentralizada, por eso en un futuro próximo a poder ser, yo intentaría introducir un CRM sobretodo para suplir esa carencia, y en el caso de que se introduzca también un ERP muchos de los módulos del software que se utilizan dejarían de ser útiles, pero ya no depende de la utilidad o no de los módulos si no de la rentabilidad de la empresa en gastarse mucho dinero en licencias para instalar un ERP o mantener este software programado ad-hoc y añadir pequeñas mejoras para que se parezca aún más a uno.

La decisión no será fácil, pero lo más optimizable para la empresa sería mejorar su software y evitar los gastos en las licencias y mantenimiento de un ERP.

8. Bibliografía

-U. (2021, 1 junio). *Tienda de Accesorios y Ropa para Moto | Unobike*. unobike.com. https://www.unobike.com/

-Oltra-Badenes, R., Gil-Gomez, H., & Guerola-Navarro, V. (2018). METODOLOGÍA PARA LA SELECCIÓN DE SISTEMAS ERP PARA PYMES. *3C Empresa : Investigación y pensamiento crítico*, *7*(4), 10–33. https://doi.org/10.17993/3cemp.2018.070436.10-33/

-Vicedo, P., Gil-Gómez, H., Oltra-Badenes, R., & Guerola-Navarro, V. (2020). A bibliometric overview of how critical success factors influence on enterprise resource planning implementations. *Journal of Intelligent & Fuzzy Systems*, *38*(5), 5475–5487. https://doi.org/10.3233/jifs-179639

-*El auge y desarrollo del comercio electrónico*. (2020, 10 agosto). Lancelot Digital. https://www.lancelotdigital.com/otras-noticias-de-interes/el-auge-y-desarrollo-del-comercio-electronico

-Computing, R. (2019, 30 enero). *En qué consiste realmente la digitalización de una empresa*. Computing. https://www.computing.es/mundo-digital/noticias/1109916046601/consiste-realmente-digitalizacion-de-empresa.1.html

-Socialmood. (2021, 16 marzo). *Guía SEO: Cómo dominar a Google en 10 pasos*. 40deFiebre. https://www.40defiebre.com/guia-seo

-*Unify the DevOps lifecycle with*. (2020, 1 junio). GitLab. https://about.gitlab.com/stages-devops-lifecycle/

-*XAMPP Installers and Downloads for Apache Friends*. (2020, 1 junio). XAMPP. https://www.apachefriends.org/es/index.html

-Microsoft. (2016, 14 abril). *Visual Studio Code - Code Editing. Redefined*. https://code.visualstudio.com/

-*ERP y CRM de código abierto | Odoo*. (2020, 1 junio). Odoo S.A. https://www.odoo.com/es_ES/

-L. (2020, 1 junio). *Online Diagram Software & Visual Solution*. Lucidchart. https://www.lucidchart.com/

-Webempresa. (2021, 18 mayo). *Qué es Apache y cómo funciona*. https://www.webempresa.com/hosting/que-es-servidor-apache.html

-*PHP: Hypertext Preprocessor*. (2021, 6 mayo). PHP. https://www.php.net/

-*Crontab.guru - The cron schedule expression editor*. (2020, 1 junio). Crontab. https://crontab.guru/