UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Automatización de una pequeña empresa mediante un proceso de ciencia de datos.

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

AUTOR/A: Lee Kâm Châk, Gustavo

Tutor/a: Ramírez Quintana, María José

CURSO ACADÉMICO: 2021/2022

A mi tutora, por la gran paciencia que tiene

A toda mi familia, por el apoyo incondicional

A mis amigos, por sacarme de casa ante el mínimo inconveniente

A mi juventud, por permitirme trabajar hasta tan tarde

Y a María Molas, por ser la persona más amable del mundo

Resum

Des de fa bastants anys s'està parlant de la importància de la dada per a la gestió eficient de les organitzacions. Mitjançant la dada, les organitzacions són capaces de comprendre qualsevol aspecte de les seues transaccions, productes, clients, empleats, proveïdors i processos de negoci. En aquest àmbit, la ciència de dades utilitzada amb objectius clars, proporciona una base sistemàtica per a la presa de decisions en el procés de crear valor afegit en les empreses. Aquest treball de fi de màster té com a objectiu principal optimitzar i automatitzar els processos clau del dia a dia d'un xicotet comerç situat a València, fent ús de tecnologies basades en les dades. La meta final és proporcionar un conjunt d'eines que faciliten el treball dels empleats d'aquesta empresa, a més d'augmentar la visibilitat de la informació interna del negoci i millorar la presa de decisions. Per a arribar a aquest fi és necessari d'una banda ocupar-se de les dades (internes i externes) rellevants per al propòsit de negoci, i d'altra banda, aplicar les tecnologies que permetran generar els models analítics. Per a aconseguir aquests objectius, s'ha triat una arquitectura basada en microserveis per al desplegament desacoblat de tots els components de l'aplicació juntament amb un enfocament rigorós de la metodologia CRISP-DM per a l'anàlisi de les dades.

Resumen

Desde hace bastantes años se está hablando de la importancia del dato para la gestión eficiente de las organizaciones. Mediante el dato, las organizaciones son capaces de comprender cualquier aspecto de sus transacciones, productos, clientes, empleados, proveedores y procesos de negocio. En este ámbito, la ciencia de datos utilizada con objetivos claros, proporciona una base sistemática para la toma de decisiones en el proceso de crear valor añadido en las empresas. Este trabajo fin de máster tiene como objetivo principal optimizar y automatizar los procesos clave del día a día de un pequeño comercio ubicado en Valencia haciendo uso de tecnologías basadas en los datos. La meta final es proporcionar un conjunto de herramientas que faciliten el trabajo de los empleados de dicha empresa además de aumentar la visibilidad de la información interna del negocio y mejorar la toma de decisiones. Para llegar a ese fin es necesario por un lado ocuparse de los datos (internos y externos) relevantes para el propósito de negocio, y por otro lado, aplicar las tecnologías que van a permitir generar los modelos analíticos. Para alcanzar dichos objetivos, se ha elegido una arquitectura basada en microservicios para el despliegue desacoplado de todos los componentes de la aplicación junto con un enfoque riguroso de la metodología CRISP-DM para el análisis basado en los datos.

Abstract

The importance of the data to efficiently manage organizations has been for many years a subject of discuss. Through the data, organizations are capable of understanding every detail of their transactions, products, clients, employees, suppliers and business processes. In this context, data science used with clear objectives provides a sistematic basis for decision making in the process of creating added value for companies. This Master's thesis' main goal is to optimize and automate the day by day key processes of a small business located in Valencia through the use of data based technologies. The ultimate objective is to provide a set of tools that facilitate the workflow of the employees in addition to increasing the visibility of internal business information and improve the business's decision making. To achieve this goal, it is necessary to deal with the relevant external and internal data for business purposes, and also, to apply the technologies that will allow the generation of the analytical models. For these purposes, an architecture based on microservices for the decoupled use of all the components of the application has been chosen alongside the use of a rigorous approach to the methodology CRISP-DM for data-driven analysis.

Tabla de contenidos

Capítulo 1

Introducción

El avance tecnológico de los últimos años junto a las nuevas facilidades que nos proporcionan la computación en la nube hace que cada vez sea más fácil crear soluciones complejas en poco tiempo. Este trabajo de fin de máster tiene como objetivo relatar los procesos y decisiones tomadas en el diseño e implementación de una solución web para una pequeña empresa valenciana. El desarrollo de esta solución se realizó durante este último año. El contenido de este proyecto está dividido en dos partes:

- Desarrollo de la solución utilizando una arquitectura basada en microservicios.
- Uso de CRISP-DM como modelo analítico para el estudio y minado de los datos internos de la empresa.

En la primera mitad de este trabajo se verán todos los elementos que han formado parte del diseño y desarrollo del sistema. Desde el apartado técnico como el conjunto de herramientas utilizados para la implementación de cada microservicio, hasta temas de ámbito empresarial como los costos mensuales, ventajas y desventajas de esta nueva solución, y la satisfacción general de los empleados.

Por otro lado, en la segunda mitad de este proyecto se verá el uso de *CRISP-DM* [1] para cumplir uno de los objetivos de negocio propuesto por el cliente. CRISP-DM está compuesto por seis fases donde trataremos de comprender el negocio, comprender los datos del negocio, preparar dichos datos, realizar un modelo, evaluar dicho modelo y finalmente añadirlo a la arquitectura basada en microservicios y desplegarlo en producción.

[1] Cross Industry Standard Process for Data Mining

Capítulo 2

Caso de estudio

El caso de estudio gira alrededor de un pequeño establecimiento de venta al por menor localizado en Valencia. Dicho establecimiento formaba parte de una franquicia valenciana cuyo negocio se centra en la venta de productos de alimentación, bebidas, droguería, bollería y panadería al consumidor final. Debido a un cambio de dirección y otros problemas que no son relevantes para este caso de estudio, el franquiciador no realizó los pagos necesarios para el correcto funcionamiento del comercio. Algunos de dichos pagos fueron los pedidos a proveedores o las licencias de las aplicaciones.

El problema más urgente que tuvo que afrontar este establecimiento ha sido la inhabilidad de seguir trabajando, ya que era imposible hacer uso de su aplicación de venta sin una licencia válida.

Este contratiempo empujó a los dueños del local a buscar otra herramienta que pudiera sustituir su sistema actual. Los requisitos básicos del nuevo sistema fueron los siguientes:

- Coste anual bajo, inferior a cien euros al mes. Cuanto más bajo, mejor.
- Intuitivo y fácil de utilizar.
- Disponible en español para facilitar el trabajo a los empleados.
- Alto control sobre los datos internos de la empresa.

Además de todo lo mencionado, la nueva aplicación ha de ser capaz de hacer lo siguiente:

- Generar ventas con sus respectivos recibos.
- Leer, crear, modificar y eliminar toda la información relacionada con los productos de la tienda.
- Llevar un control sobre las existencias del local.
- Proporcionar un «POS» [1] donde los trabajadores puedan interactuar con el sistema y generar nuevas ventas.
- Proporcionar un medio donde los empleados puedan imprimir las etiquetas de los productos. Dicha etiqueta debe contener el nombre del producto y su PVP.

[1] Point of sale. Punto de venta en español

Opcionalmente puede contener el código de barras.

- Debe permitir abrir y cerrar caja. Cada cierre de caja debe proporcionar un documento donde se resume dicho cierre con los valores iniciales, finales y reales de la caja registradora.
- Debe permitir hacer reembolsos y devoluciones a los clientes.
- Opcionalmente, debe mostrar información relevante para los usuarios de la aplicación como un análisis descriptivo de los datos (estadísticas, resumen diario...).
- Debe proporcionar una herramienta que facilite contar el dinero de la caja.
- Debe permitir la exportación de los datos en algún tipo de documento editable como hojas de cálculo o CSV.
- El sistema ha de responder rápidamente, especialmente en las funciones relacionadas con la TPV. Por tanto, las llamadas al servidor deben durar menos de tres segundos.

Además de un nuevo conjunto de herramientas, es necesario crear y desplegar el servicio web resultante del proceso de minería de datos. Este servicio ha de ser capaz de dar una solución a algunos de los problemas de negocio típicos de un pequeño comercio.

Capítulo 3

Herramientas en el mercado

A continuación se hablará de las opciones valoradas para la sustitución el antiguo sistema que proporcionaba la franquicia. Se verán los costes de dichos sistemas, ventajas y desventajas y qué pueden llegar a ofrecer.

3.1. Ekon ERP

Ekon ERP es un software de gestión de empresas cuyo objetivo es mejorar la operatividad a través de la automatización de los procesos de negocio. Proporciona herramientas básicas para la gestión de un comercio minorista. Desde puntos de venta, inicio sesión hasta gestión de contabilidad y RRHH.

Figura 3.1: Logotipo de Ekon

3.1.1. Ventajas

- Todo el personal de la tienda conoce esta aplicación ya que era la aplicación con la que trabajaban.
- Aporta un control sencillo sobre los datos de la tienda.
- Se trata de una aplicación de escritorio, lo cual brinda mayor control sobre el hardware disponible en la tienda como la caja registradora.

3.1.2. Desventajas

- Hacer algunas operaciones sobre este sistema se vuelve bastante pesado y tedioso (modificar el tipo de cobro de una venta en caso de error).
- Los costos anuales sobrepasan los deseados por parte del cliente.

Se trata de una herramienta bastante completa, especialmente teniendo en cuenta la experiencia previa del personal, por desgracia cuenta con un alto coste anual.

3.2. ERPNext

ERPNext es un software de planificación de recursos empresariales integrado gratuito y de código abierto desarrollado por Frappe Technologies. Se trata de una aplicación web clara y fácil de utilizar. Frappe proporciona las herramientas necesarias para desplegar su aplicación en un servidor (o varios si se necesita) mediante Docker o Kubernetes.

3.2.1. Ventajas

- Código y aplicación totalmente gratuitos.
- Interfaz limpia y baja curva de aprendizaje.
- El personal de Frappe está dispuesta a ofrecer ayuda a cualquiera que la necesite, a cambio de una remuneración económica.
- Frappe ofrece planes de pago para despliegues de su aplicación.

3.2.2. Desventajas

- Está mayoritariamente en inglés. Si bien Frappe proporciona soporte a varios idiomas, la calidad de ERPNext en español deja mucho que desear.
- La cuota más económica del plan de despliegue de Frappe es de trescientos euros al mes.
- El despliegue por cuenta propia puede llegar a ser complicado incluso con la ayuda de Docker, y más si se tiene en cuenta el versionado del sistema.

ERPNext es una buena solución ya que es gratis, salvo por los costes de los servidores. Por desgracia el despliegue es algo complejo en caso de querer un bajo coste mensual. En caso contrario, Frappe puede alojar todo el sistema, facilitando todo este trabajo inicial. Por desgracia el precio mensual por alojar el sistema con Frappe asciende a unos trescientos euros como mínimo.

Figura 3.2: Logotipo de ERPNext

3.3. Odoo

Odoo es un conjunto de aplicaciones de código abierto que cubren muchos casos de uso muy frecuentes en una empresa: CRM, comercio electrónico, contabilidad, inventario, punto de venta, gestión de proyectos, etc.

Odoo, muy similar a ERPNext, también ofrece opciones de despliegue de pago. A diferencia de ERPNext, Odoo es mucho más personalizable ya que cuenta con diversos módulos o «plug-ins» que mejoran la funcionalidad de la herramienta y la adapta a todos los posibles casos de uso.

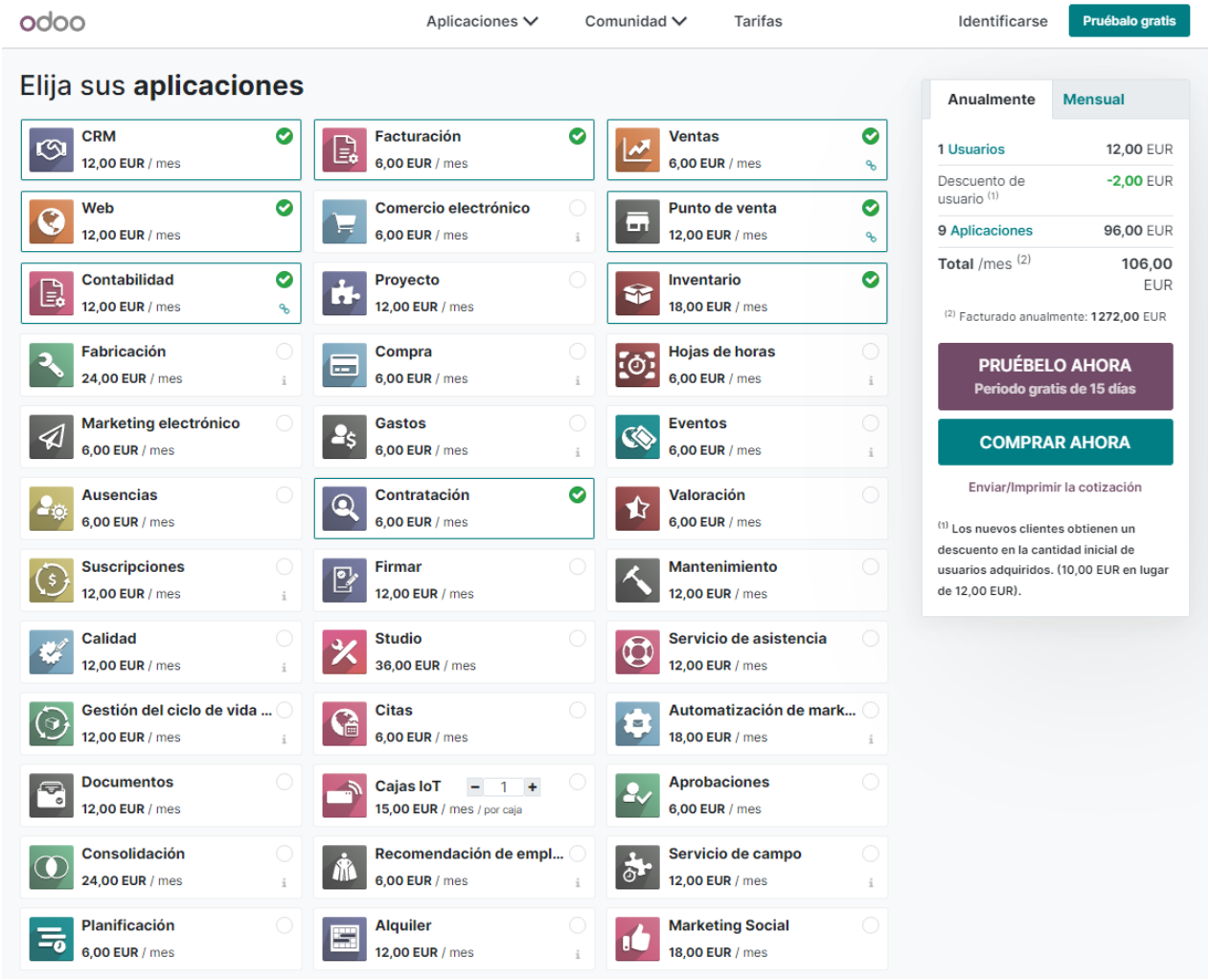


Figura 3.3: Plugins de Odoo junto a sus precios

3.3.1. Ventajas

- Código abierto.
- Herramienta basada en plugins. Se adapta a las necesidades del cliente.
- En español.
- Aprendizaje sencillo.

3.3.2. Desventajas

- Coste variable, en función de las necesidades del cliente podría o no superar su cuota mensual.
- Despliegue complejo en caso de hacerlo solo.

Se obtuvieron resultados muy similares al trabajar con ERPNext. Se desplegó Odoo sobre una instancia de EC2 de AWS con algo de dificultad. Existen muchos detalles que se deben tener en cuenta en toda la instalación, lo cual puede resultar tedioso a la vez que consume mucho tiempo.

Figura 3.4: Logotipo de Odoo

3.4. Vend

«Vend» es una empresa de software para minorista que proporciona varios terminales de punto de venta para sus clientes. El coste de la solución aumenta en función del número de terminales contratadas. Su sistema consta de una aplicación web donde los trabajadores pueden realizar actividades esenciales como vender productos usando una «TPV» (terminal punto de venta) a los consumidores finales, pueden acceder a una lista de productos y modificar la información asociada a cada producto, entre muchas otras cosas.

Figura 3.5: Logotipo de Vend

Esta fue la primera solución usada por los dueños del local. En agosto de dos mil veintiuno, cuando caducaron las licencias de Ekon, se usó la prueba gratuita de catorce días de Vend como remedio momentáneo para "salir del paso". Dicho periodo de pruebas brindó algo de tiempo para buscar otro sistema más acorde a las necesidades de la tienda.

Vend proporciona una interfaz amigable y fácil de entender, el sistema está bien diseñado e incluso facilita las tareas monótonas. A pesar de ello, Vend no proporciona dichas interfaces en español, lo cual supone un problema ya que la mayoría de los empleados no conocen el inglés. Por otra parte, los precios que proponen no satisfacen las necesidades de los dueños del pequeño comercio. Para disponer de dos terminales de punto de venta, es necesario pagar mensualmente ciento cincuenta y ocho euros, lo cual supera el presupuesto máximo.

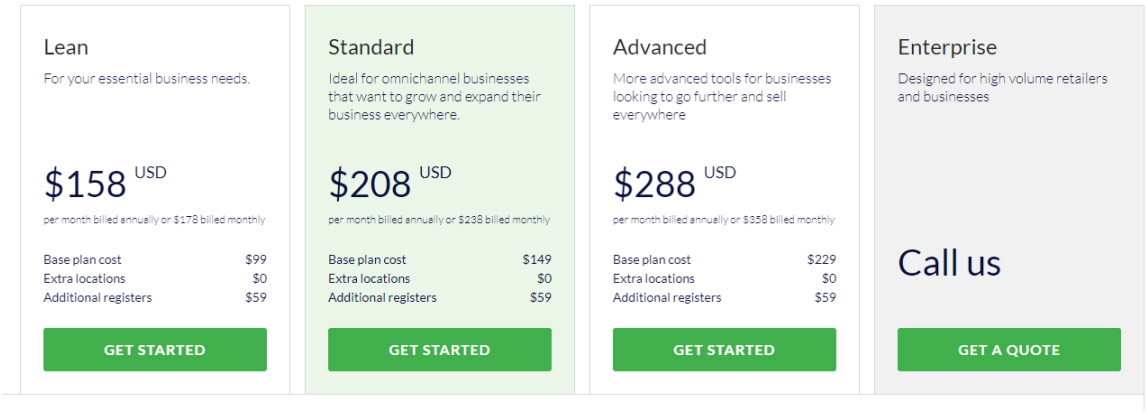


Figura 3.6: Precios de Vend en 2022

3.4.1. Ventajas

- Dispone de una prueba gratuita de catorce días.
- Fácil de desplegar y empezar a usar. Se puede poner en marcha con unos pocos clics.
- Aprendizaje sencillo.
- Proporciona un apartado de «estadísticas» donde se pueden ver datos internos como el número de ventas por hora o el beneficio bruto facturado.

3.4.2. Desventajas

- Código propietario (no se trata de código abierto).
- El coste de Vend supera el presupuesto del local.
- Totalmente en inglés, supone una barrera lingüística para los trabajadores.
- Poca funcionalidad. Vend encaja más como un «POS» de mucha calidad que un sistema ERP.

3.5. ClassicGes 6

ClassicGes 6 es un motor de administración con herramientas que se adaptan a todo tipo de empresa. Se trata de una aplicación de escritorio monolítica para Windows, donde cada ordenador es responsable de la integridad de sus datos. ClassicGes 6 está totalmente en español, lo cual es una ventaja. Además de eso, el precio anual de las licencias no superan los ciento cincuenta euros. También se puede contratar asistencia técnica para todo un año por trescientos euros.

A diferencia de las demás herramientas, ClassicGes 6 no es una aplicación web sino que es una aplicación instalable. Esto es bueno ya que cuenta con mejor rendimiento que las herramientas web y tiene acceso a bajo nivel de componentes hardware como el escáner de código de barras, la impresora y la caja registradora. Por desgracia, este sistema trae una interfaz gráfica poco agradable con multitud de botones redundantes y graves fallos de seguridad. La curva de aprendizajes es mayor que las demás opciones en el mercado por la burda interfaz gráfica, la excesiva cantidad de información que muestran algunas porciones del sistema y lo complejo que puede llegar a ser realizar tareas sencillas.

Además de lo mencionado, cada monolito tiene su propia capa de datos, lo cual supone un problema al trabajar con dos o más ordenadores al mismo tiempo. Por ejemplo: si hay un producto nuevo en la tienda, es necesario rellenar el formulario del nuevo producto (nombre, código EAN, precio de compra, familia...) dos o más veces, una para cada ordenador. Como solución a este problema, el equipo de AIG recomienda crear un canal de red local en el que todos los monolitos consuman de la misma carpeta de datos.

La seguridad que aporta esta solución es baja, cualquiera con acceso a los ordenadores es capaz de entrar a la carpeta de datos y modificar (incluso borrar) información delicada acerca de la empresa como el registro de ventas, los cierres contables e incluso la información acerca de los productos.

Por otro lado, se trata de la opción más económica que se ha podido encontrar. Por cuatrocientos cincuenta euros al año (o treinta y siete con cincuenta euros al mes), se tiene un sistema que cumple con todas las necesidades de la tienda a cambio de una mayor curva de aprendizaje y una baja seguridad acerca de los datos.

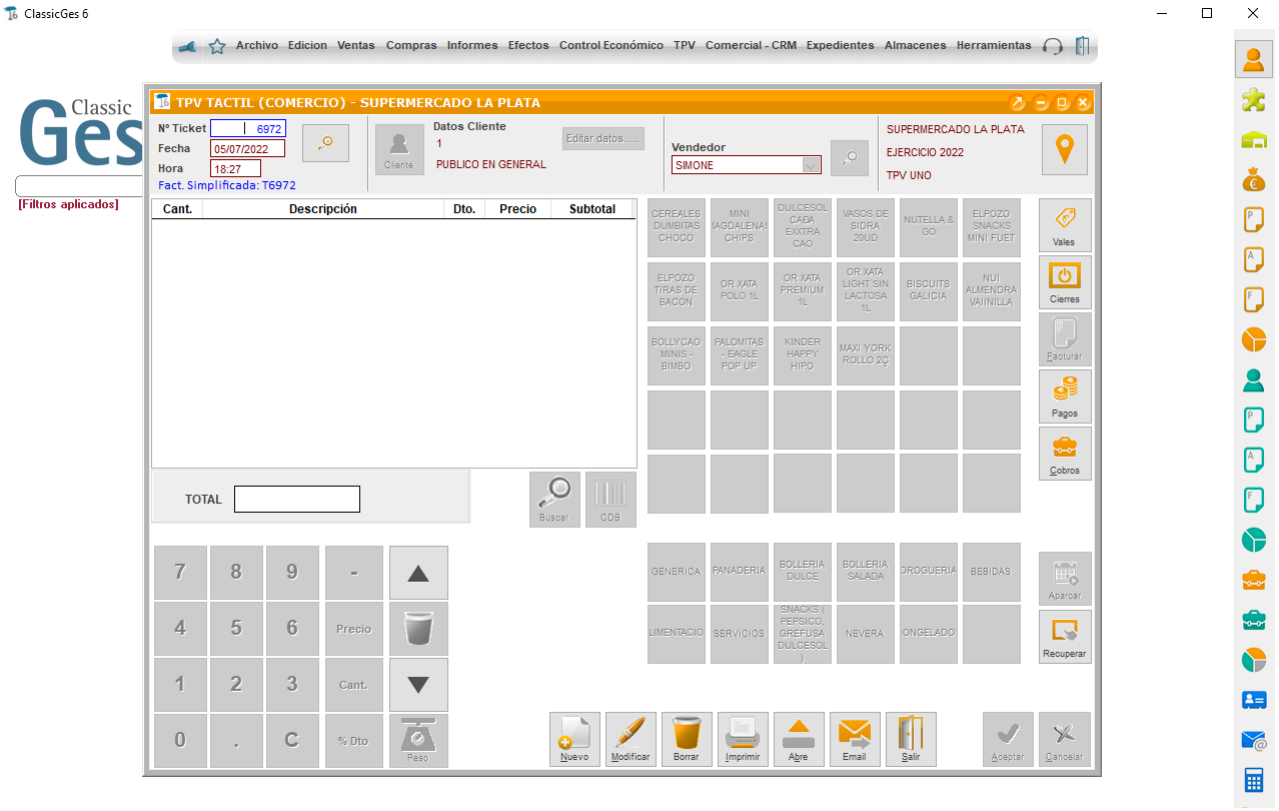


Figura 3.7: Pantalla de inicio de ClassicGes 6

3.5.1. Ventajas

- Muy económico.
- Fácil de instalar.
- Totalmente en español.
- Aplicación de escritorio.
- Puede funcionar sin conexión a internet.

3.5.2. Desventajas

- Código propietario (no se trata de código abierto).
- Interfaz gráfica poco amigable.
- Difícil de aprender ya que puede causar mucha confusión al principio.

- Se trata de un monolito donde cada instancia de la aplicación es responsable de la integridad de su capa de datos.
- Requiere configuración adicional para crear el canal de red compartida y distribuir la misma carpeta de datos.
- Requiere la instalación de una nueva aplicación en caso de requerir alguna actualización o parche.
- Poco seguro. Los datos están a disposición de cualquiera que tenga acceso a los ordenadores (cualquier trabajador).

A diferencia de las demás opciones mencionadas hasta ahora, ClassicGes 6 tiene un coste mensual que se adapta al presupuesto de la empresa. Por otro lado, esta está completamente en español, lo cual es una gran ventaja sobre herramientas como Vend. Si bien es cierto que ClassicGes 6 cuenta con problemas de seguridad importantes, los franquiciados se encuentran en una situación delicada con su franquiciador y no esperan expandir su negocio brevemente. Este es un detalle a tener en cuenta porque ClassicGes 6 no escala bien con muchos ordenadores. Disponer de más de una tienda supone realizar mucho trabajo adicional para mantener la integridad de los datos.

3.6. Desarrollar el software a medida

Sin duda se trata de la opción más barata una vez desplegada. Por desgracia, es necesario desarrollar el sistema, lo cual implica un gran coste temporal. Realizar el desarrollo de un sistema ERP (o cualquier sistema software no trivial) requiere de muchas horas de trabajo y reuniones con los clientes para verificar y validar el software. A diferencia de las otras opciones en el mercado, esta aplicación cumple con todas las expectativas del cliente, ya que el software en cuestión está diseñado en base a sus necesidades. Otra gran ventaja es la facilidad de ampliar y modificar el software en función de los futuros requerimientos del comercio.

Con tal de seguir las mejores prácticas y tendencias actuales del desarrollo de software, se ha optado por construir una solución siguiendo una arquitectura basada en microservicios y desplegarlo en la nube. Esta aproximación nos permite desarrollar los distintos componentes del sistema con bajo acoplamiento y poca dependencia entre los servicios.

En internet existen multitud de herramientas que permiten calcular el coste aproximado del uso de sus máquinas virtuales. Según algunos cálculos iniciales, se ha obtenido un costo estimado de doce euros al mes para una instancia EC2 de AWS (t3a.micro). Teniendo en cuenta que muchas empresas de computación en la nube ofrecen planes de uso bajo demanda (es decir, se paga por la capacidad informática por unidad de tiempo que se usa), se puede estimar un coste mensual inferior a dichos doce euros. Esto se debe a que no se usaría la instancia al cien por cien durante todo el día, sino que se haría en horario laboral.

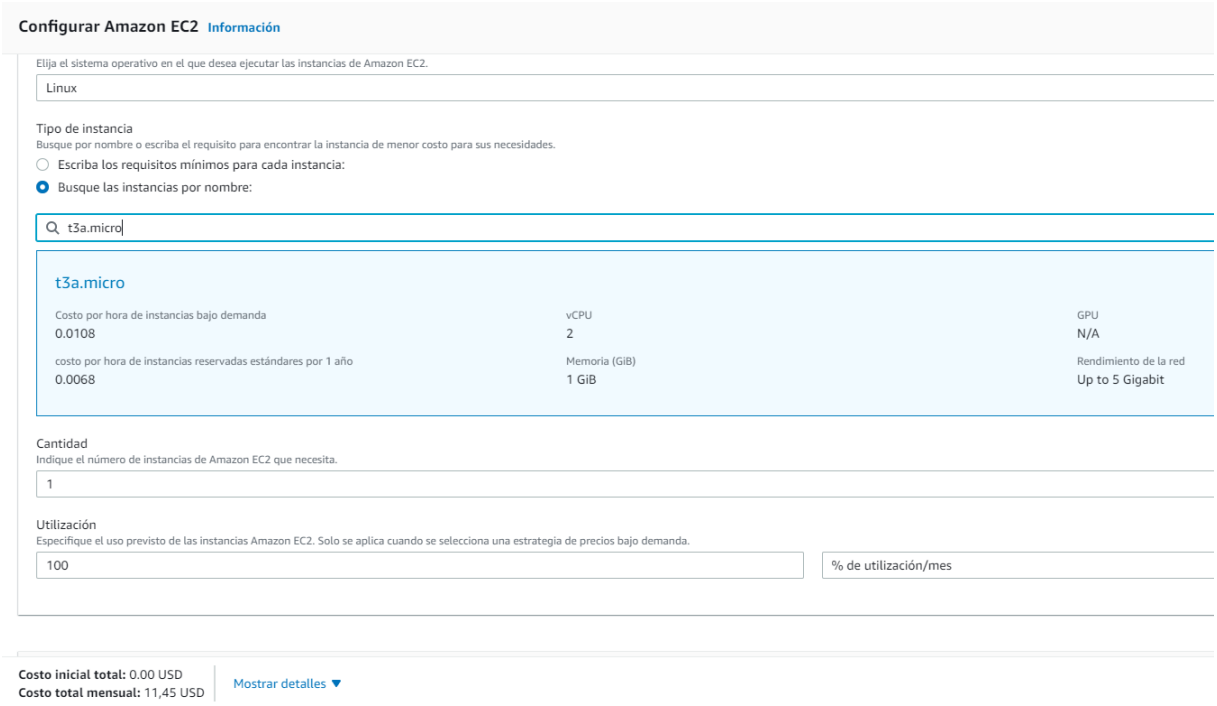


Figura 3.8: Calculadora de precios de Amazon Web Services

3.6.1. Ventajas

- Modificable y adaptable.
- Totalmente en español.
- Intuitivo y fácil de utilizar.
- Coste mensual bajo.
- Claridad de los datos internos de la empresa.
- Código abierto.

3.6.2. Desventajas

- Requiere de mucho tiempo de desarrollo.
- Se necesita desarrollar el software antes de utilizarlo. No sirve hasta que no esté terminado o sea un «MVP» (producto mínimo viable).
- El despliegue puede resultar complicado ya que el sistema puede estar compuesto por diversos microservicios que necesitan un canal de comunicación y una buena coordinación.
- No funciona sin internet, lo cual puede causar muchos inconvenientes si el establecimiento llega a tener problemas de conexión.

Por desgracia, los franquiciados necesitan una herramienta funcional cuanto antes. No disponen del tiempo necesario para esperar el desarrollo de un primer MVP, lo cual les obliga a contratar una de las aplicaciones mencionadas previamente.

3.7. Comparación de las herramientas

En esta sección se muestra una tabla comparativa de las diferentes características que soportan las herramientas mencionadas previamente. La finalidad de esta tabla es ayudar a la toma de decisiones y establecer de forma gráfica las mejores opciones para este caso de estudio.

Los requisitos marcados con un asterisco son considerados necesarios y los que llevan dos asterisco son considerados como imprescindibles.

Soluciones ERP						
	Ekon	ERPNext	Odoo	Vend	ClassicGes6	A medida
Open-source	✗	✓	✓	✗	✗	✓
Coste asumible**	✗	✗	✗	✗	✓	✓
Aplicación de escritorio	✓	✗	✗	✗	✓	✗
Fácil de usar*	✓	✓	✓	✓	✗	✓
En español**	✓	✗	✓	✗	✓	✓
Seguro y fiable*	✓	✓	✓	✓	✗	✓
Funciona sin conexión	✗	✗	✗	✗	✓	✗
Exportación de datos*	✓	✓	✓	✓	✗	✓
Contador de dinero	✓	✗	✗	✗	✗	✓
Permite reembolsos*	✓	✓	✓	✓	✗	✓
Estadísticas	✗	✓	✓	✓	✗	✓
Control de inventario*	✓	✓	✓	✓	✓	✓
Registro de albaranes	✓	✓	✓	✓	✗	✗
Login de usuarios	✓	✓	✓	✓	✗	✓
Expansible	✗	✗	✓	✗	✗	✓
Disponibilidad inmediata**	✗	✓	✓	✓	✓	✗
Modelo de ML	✗	✗	✗	✗	✗	✓

Cuadro 3.1: Tabla comparativa de las diferentes soluciones ERP

En la tabla previa podemos observar algunas de las propiedades principales que reúnen los diferentes sistemas. No todas las características tienen el mismo peso, el coste mensual de las herramientas es sin duda el factor más importante para los clientes, pero en conjunto esta tabla pretende reflejar la calidad de cada una de ellas y también pretende ayudar al cliente en su decisión.

Como se puede comprobar, ClassicGes 6 es la herramienta con peor desempeño de las seis expuestas en este trabajo, pero a diferencia de todas las demás opciones (salvo el desarrollo del software para el cliente), ClassicGes 6 es la única que cumple con las propiedades «coste asumible», «en español» y «disponibilidad inmediata». Cubiertos estos tres requisitos, los demás siguen siendo importantes en caso de necesitar una herramienta de mejor calidad o más completa en un futuro.

3.8. Solución planteada

Ante este escenario, los dueños del pequeño comercio decidieron hacer uso de ClassicGes 6 ya que cuenta con el precio más económico de todos los mencionados y también está completamente en español. Por desgracia, el uso de este sistema no es viable a largo plazo ya que no escala muy bien con más de un ordenador (por no decir más de un establecimiento) ni facilita la extracción de los propios datos de la empresa. Además de eso, cuenta con numeroso fallos de seguridad que pueden llegar a costar mucho dinero a la empresa.

Para evitar el uso a largo plazo de una herramienta carente de funcionalidad, se ha planteado desarrollar, durante un periodo de un año, una aplicación más acorde a las necesidades de la empresa. La idea consiste en usar ClassicGes 6 durante todo un año, y mientras tanto, desarrollar un sistema software de mejor calidad que eventualmente sustituya ClassicGes 6. Esta idea fue bien recibida por parte de los franquiciados porque les permitía seguir adelante con el curso de sus actividades sin muchos inconvenientes a cambio de, en un futuro próximo, adaptarse a una nueva solución que les ofrecería muchas más herramientas útiles para el día a día.

Asimismo, desarrollar una solución desde cero brinda la posibilidad de ampliar la funcionalidad del sistema con tecnologías muy útiles e interesantes que se adaptan todavía más a las necesidades de la empresa. En este caso de estudio, los franquiciados ya no disponían del acceso a la lista de ofertas y descuentos que ofrecía la franquicia en su catálogo de productos. Con tal de recuperar a los clientes que buscan este tipo de descuentos, los dueños del local desean que la nueva aplicación pueda ofrecer algún tipo de descuento a los clientes de la tienda en base a lo que esté comprando en ese momento, similar a un sistema de recomendación. Para lograr este objetivo, se ha llegado a la conclusión de que es necesario abordar este problema mediante un método de minería de datos. Dichos datos son todos aquellos relacionados con las ventas, los productos y los clientes. Además, esta información es proporcionada por la propia empresa. Para ello, se seguirá una metodología conocida como CRISP-DM y se desplegará el modelo resultante como un microservicio más del sistema final.

Capítulo 4

Desarrollo

Este capítulo tiene como objetivo reflejar todo el proceso de desarrollo del nuevo sistema ERP que tuvo lugar desde el día 16/08/2021 hasta el 20/07/2022, dicho sistema se pasará a llamar «ERPSolution». Los dueños del pequeño comercio, de ahora en adelante los clientes del proyecto software, sustituyeron con éxito ClassicGes 6 por ERPSolution.

También se verá, paso a paso, todo el proceso de minería de datos que se ha realizado siguiendo la metodología CRISP-DM. Este proceso incluye todas las seis fases propuestas por el método, desde la comprensión del negocio hasta el despliegue final del modelo y cómo se integra con ERPSolution.

4.1. ERPSolution

Se trata de la solución ERP de código abierto creada para mejorar y automatizar los procesos diarios de un pequeño establecimiento ubicado en Valencia. Esta solución fue diseñada para ser barata y fácil de utilizar. Este proyecto prima la sencillez y el desarrollo rápido, es por ello que no se hablará del despliegue de este sistema en servidores propios ni tampoco en ordenadores locales, toda la solución está pensada para ser desplegada en alguna infraestructura en la nube donde todo el contenido relacionado con la administración y control de sistemas será obviado por cuestiones de simplicidad.

4.1.1. Arquitectura

La historia de la informática está llena de grandes eventos, desde la creación de UNIX[1] o el auge del World Wide Web (WWW) hasta la aparición de la inteligencia artificial y las nuevas tendencias del desarrollo de software. No es de extrañar que hayan miles de nuevas páginas web diariamente.

Crear una nueva aplicación no es tarea sencilla. Desde aplicaciones «CLI» hasta aplicaciones móviles, es necesario tener en cuenta una infinidad de detalles sobre el proyecto, estudiar la mejor forma de abordarlo, desarrollarlo, distribuirlo y mantenerlo. Es por eso que existen multitud de arquitecturas, patrones de diseño y buenas prácticas sobre de este tema.

[1] Unix es un sistema operativo portable, multitarea y multiusuario que fue desarrollado en 1969

Actualmente existen dos arquitecturas comúnmente conocidas: arquitecturas basadas en monolitos y arquitecturas basadas en microservicios. Cada una con sus ventajas y desventajas.

4.1.1.1. Monolitos

Se trata de la más antigua de las dos y consiste en un sistema software en la que la capa de presentación, la capa de negocios y la capa datos están combinadas en un mismo programa y sobre una misma plataforma. Refleja el método de desarrollo que apareció naturalmente a finales del siglo XX y principios del siglo XXI.

Consiste en una única aplicación donde todos los componentes necesarios están en un mismo proyecto. Este modelo puede resultar cómodo y rápido de desarrollar en las primeras etapas de un proyecto, pero escala bastante mal. La alta dependencia entre los componentes puede resultar en dificultades a la hora de mantener, probar y modificar el programa. Debido a estos fallos, esta arquitectura puede resultar muy cara de mantener a largo plazo.

Muchas empresas transformaban su solución en un fichero «jar» o «war» y lo desplegaban mediante algún contenedor de servelts (como Apache Tomcat). Los ficheros resultantes podrían llegar a ocupar varios megabytes (incluso gigabytes). Si se hacía algún cambio en la aplicación, era necesario volver a desplegar todo el monolito.

Algunas de sus ventajas son:

- Permite un rápido desarrollo en las etapas iniciales del proyecto.
- No requiere la creación de varios proyectos (servicios) conectados mediante un API, lo cual simplifica el desarrollo.
- Costo inicial barato.

En cambio, sus desventajas incluyen:

- La complejidad del proyecto crece constantemente ya que, en un mismo proyecto, es necesario tener en cuenta una funcionalidad cada vez más grande.
- Una aplicación monolítica puede aportar serios problemas de escalabilidad. Al depender de una instancia donde toda la aplicación lo forma un mismo código, toda la funcionalidad debe ser escalada al mismo tiempo. Eso dificulta mucho el escalado horizontal y obliga a los desarrolladores a escalar verticalmente, lo cual es mucho más caro.
- La mantenibilidad es otro problema ya que el código se vuelve cada vez más complicado y arduo de mantener, el número de entregas se reduce y la aplicación se vuelve menos consistente y estable.

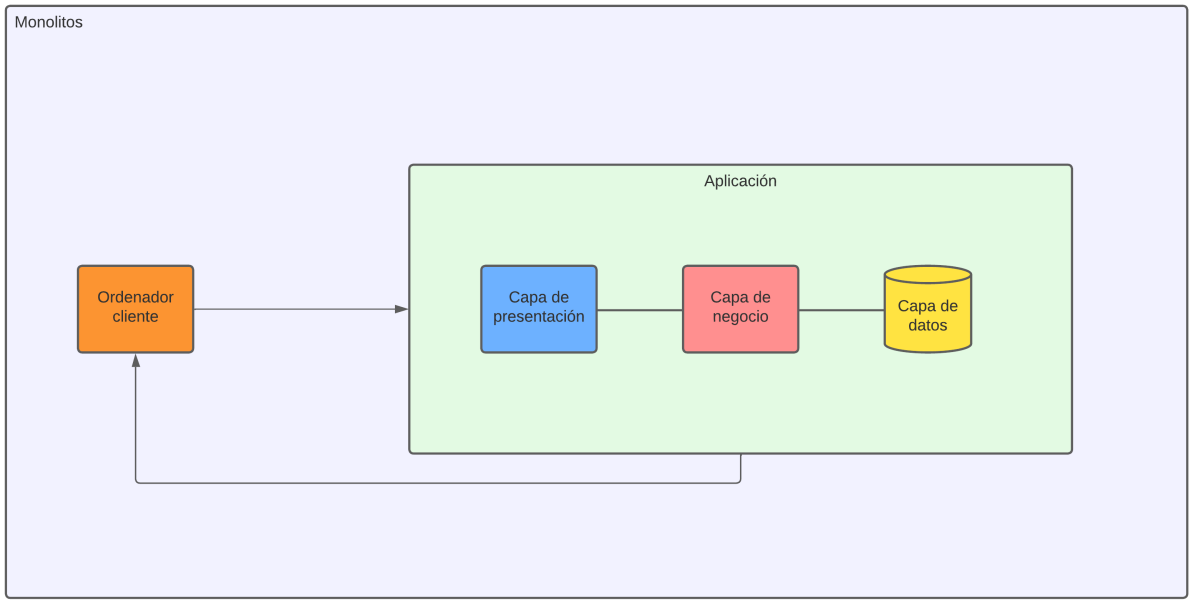


Figura 4.1: Arquitectura monolítica

4.1.1.2. Microservicios

Arquitectura más reciente cuya aparición se remonta a finales de la primera década del siglo XXI. Aparece como respuesta a los problemas que trae la arquitectura basada en monolitos. La idea es dividir el software en pequeños servicios independientes que se comunican mediante APIs bien definidas. Esta filosofía de trabajar con sistemas pequeños, manejables e independientes mejoran notablemente la mantenibilidad. También cabe recalcar que los servicios resultantes son más pequeños y están desacoplados, lo cual abarata la escalabilidad porque es mucho más sencillo escalar horizontalmente en una arquitectura basada en microservicios que una basada en monolito.

La parte positiva de esta arquitectura incluye:

- Más desacoplamiento entre los diferentes componentes que forman la solución.
- Mejora en la escalabilidad.
- Mejora en la mantenibilidad. Desarrollar, mantener y testear un microservicio supone una menor carga cognitiva para el desarrollador que hacerlo sobre un proyecto grande con muchas dependencias.
- Permite el uso de la integración continua.

Pero existen algunos inconvenientes como los siguientes:

- Requiere de un esfuerzo adicional y más «boilerplate» que una arquitectura monolítica ya que es necesario añadir a cada servicio un canal de comunicación para enviar y recibir datos. La forma más frecuente de hacerlo es convertir los microservicios en APIs REST.

- Es necesario contenerizar los diferentes servicios y añadir algún tipo de orquestador para coordinarlos, lo cual podría tomar algo de tiempo.
- Supone una mayor complejidad que en el caso anterior ya que es necesario gestionar la creciente cantidad de servicios.
- Es necesario tener en cuenta temas como la latencia en la red.

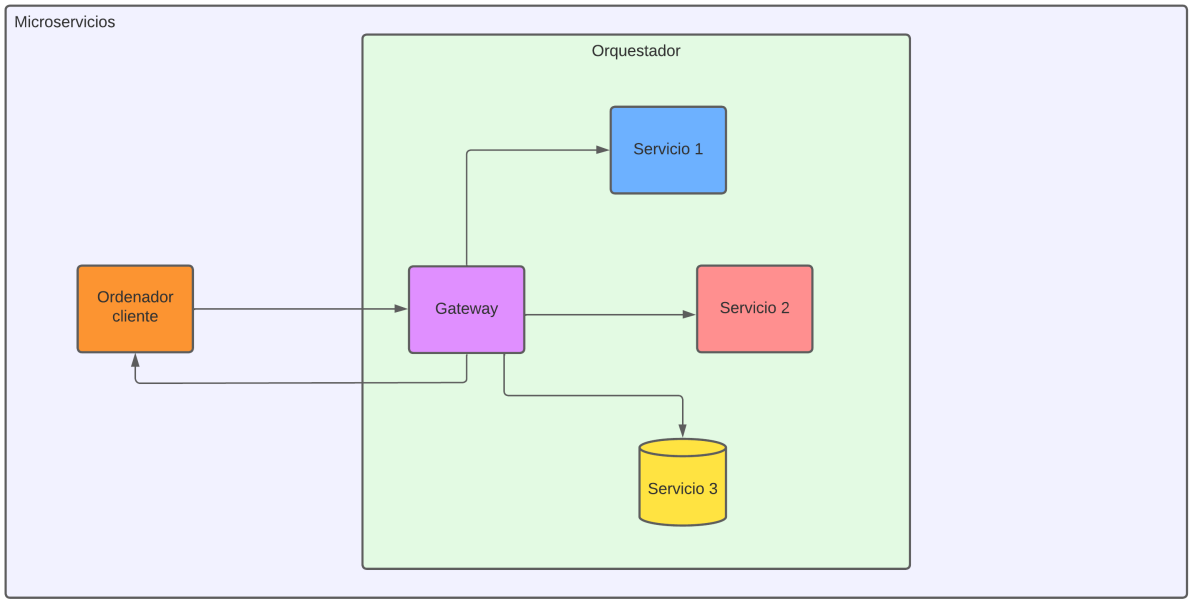


Figura 4.2: Arquitectura basada en microservicios

Para esta solución y, como se ha mencionado previamente, se hará uso de una arquitectura basada en microservicios desplegada en la nube. Esta decisión se ha tomado teniendo en cuenta que un sistema ERP es una aplicación medianamente grande que podría requerir de futuras actualizaciones, además de saber que no es muy recomendable desarrollar aplicaciones monolíticas sabiendo que sus desventajas podría llegar a costar mucho tiempo y dinero a la empresa.

4.1.1.3. Arquitectura final

La arquitectura web de ERPSolution se divide en dos componentes, tres en caso de contar el ordenador del cliente.

- El ordenador del cliente: se trata del punto de acceso de los usuarios con la herramienta.
- La página web, alojada en Vercel: la aplicación web está alojada en la infraestructura de Vercel. Desde allí puede hacer peticiones al API de ERPSolution y acceder a la funcionalidad del sistema.
- El backend en AWS: el servidor final, donde se almacenan los datos y se hacen los cómputos más caros, es accesible mediante un API REST. La aplicación web alojada con Vercel llama a los servicios de esta parte de la infraestructura y esta devuelve información a la página web que eventualmente se renderiza en el ordenador del cliente.

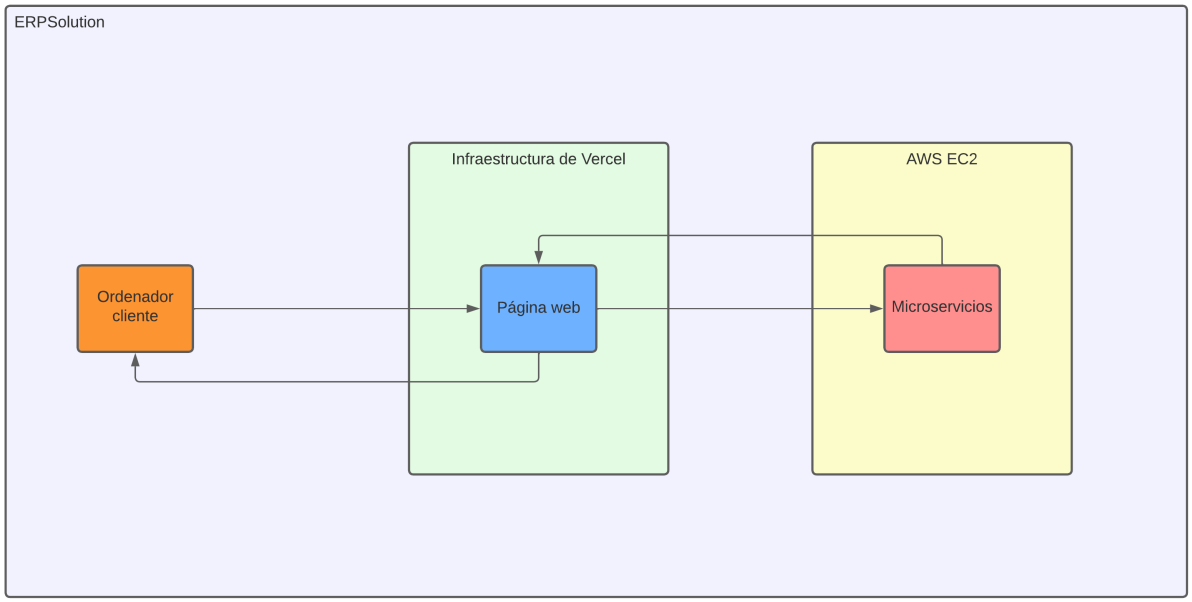


Figura 4.3: Arquitectura de ERPSolution

4.1.2. Stack tecnológico

Se han utilizado diferentes herramientas a lo largo del desarrollo. Desde frameworks de JavaScript como ReactJS (NextJS) hasta orquestadores de servicios como docker-compose. En esta sección se hablará de la composición del proyecto, por qué se han elegido ciertas tecnologías, sus ventajas y desventajas, y cómo se conectan todas ellas.

- NextJS: se trata de un framework de código abierto para JavaScript. Funciona sobre NodeJS y es muy popular entre los programadores front-end por su uso

de ReactJS. Ofrece tanto «client-side rendering» como «server-side rendering». Se ha elegido NextJS por su consistencia y la experiencia previa con ReactJS.

- NodeJS y ExpressJS: runtime de JavaScript, muy popular ya que permite el uso de dicho lenguaje de programación en entornos diferentes al navegador. ExpressJS, en cambio, es una librería de NodeJS que facilita el desarrollo de APIs. Famosa por su sencillez y robustez. Se ha elegido NodeJS por su gran ecosistema.
- MongoDB: uno de los sistemas de gestión de base de datos más populares del mundo. A diferencia de MySQL o PostgreSQL, MongoDB es una base de datos basada en documentos, no una base de datos relacional. Se ha elegido MongoDB por la facilidad que aporta trabajar con ficheros en formato JSON.
- Go y Gin-Gonic: consiste en un lenguaje de programación creado por Google. Este lenguaje es compilado y, a nivel de rendimiento, es mucho más rápido que un lenguaje interpretado como JavaScript sobre NodeJS. Gin-Gonic es una librería de Go que ayuda a crear APIs REST.
- Apollo (GraphQL): Apollo es un conjunto de herramientas que permiten y simplifican todo el proceso de comunicación entre servicios mediante el uso de GraphQL. Se ha elegido trabajar con GraphQL por la facilidad que supone trabajar con estructuras de datos bien definidas, algo que no proporciona REST.
- Docker: herramienta mundialmente conocida de código abierto que permite la automatización del despliegue de aplicaciones mediante contenedores. Docker asegura que un servicio pueda funcionar correctamente en cualquier máquina, independientemente de su sistema operativo. Otra ventaja de Docker es su ligereza ya que funciona como una máquina virtual pero no virtualiza ni simula los componentes hardware de un ordenador.
- REST: este ha sido el método elegido para la comunicación y transferencia de datos entre todos los componentes. Esta elección se debe a diversas ventajas que aporta REST. Si bien el estándar SOAP ofrece mejores resultados en cuanto a seguridad y estandarización, REST es muy útil en proyectos de pequeña y mediana escala ya que permite un desarrollo más ágil, contiene una menor curva de aprendizaje, es ligero y mucho menos estricto.

4.1.3. ERPWeb

ERPWeb es la parte del sistema con la cual interactuarán todos los empleados del pequeño comercio. Consta de una página web hecha con NextJS (un framework de JavaScript sobre ReactJS). Esta aplicación consta de varias pantallas que dividen todas las secciones del sistema.

Requisitos esenciales:

- Inicio y cierre de sesión.
- Creación de nuevas cuentas.
- Acceso de lectura y escritura sobre los datos relacionados a las ventas, productos, cierres, empleados y devoluciones.
- Acceso y uso del punto de venta.
- Dashboard.
- Herramienta para contar el dinero de la caja.
- Impresión de etiquetas y recibos.

Esta aplicación contiene todas las herramientas necesarios para llevar a cabo las tareas del día a día de la tienda. Desde cobrar por caja a los clientes hasta la devolución de una venta.

Lo primero que ven los empleados es la pantalla de inicio donde pueden navegar a la pantalla de «Iniciar sesión». Esta página es de acceso público, tanto trabajadores como terceros pueden acceder al contenido que aquí aparece.

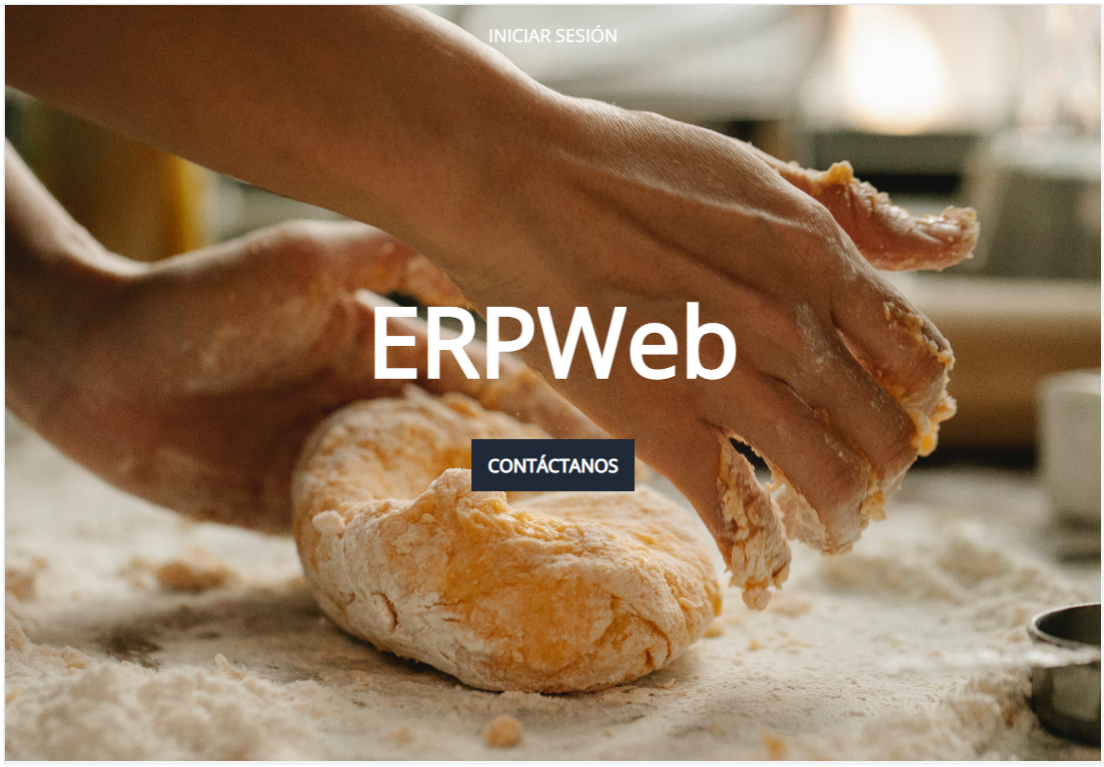


Figura 4.4: Página inicial de ERPWeb

Si la persona que ha accedido a la página cliquea en el botón de iniciar sesión, será llevado a la siguiente página. En esta sección del sistema, el personal con cuenta registrada podrá introducir sus credenciales para acceder a las distintas herramientas de ERPWeb una vez sea autenticado correctamente.

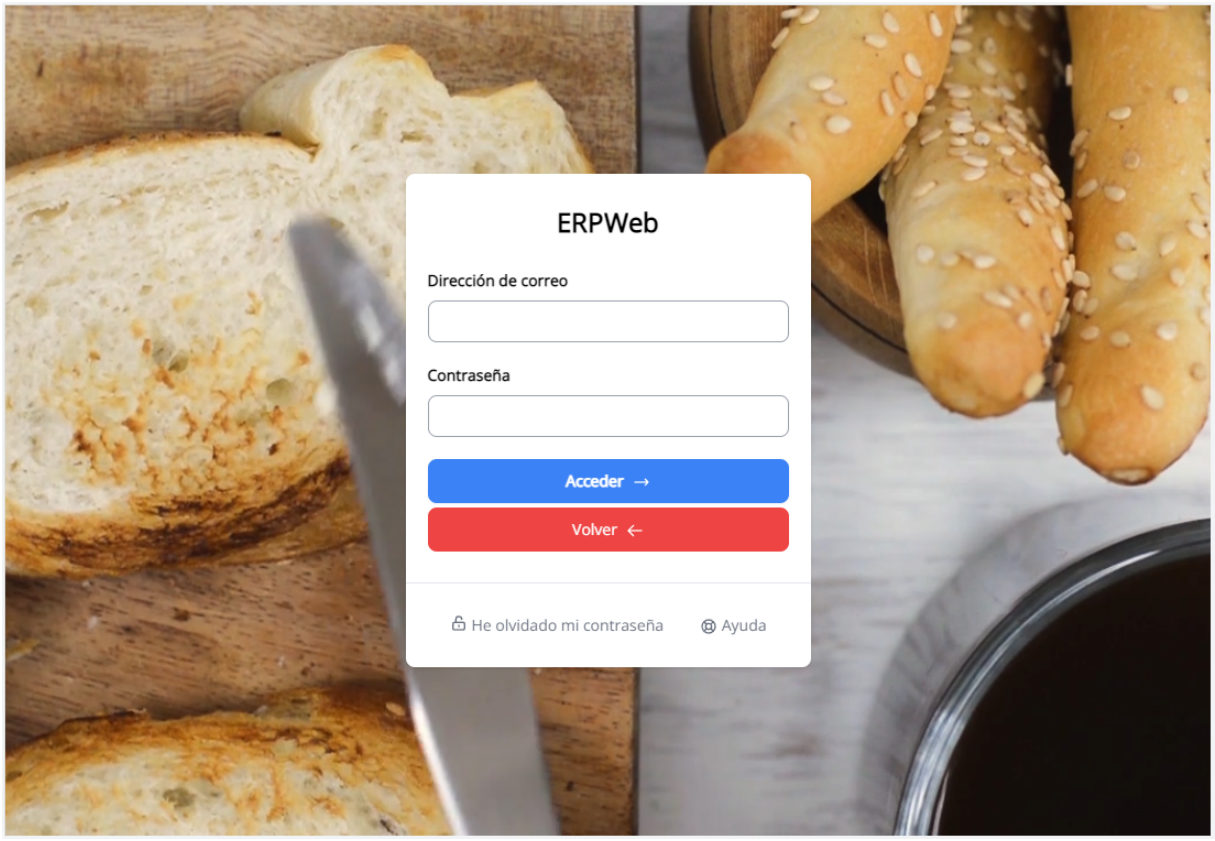


Figura 4.5: Inicio de sesión

Una vez autenticado, el empleado podrá acceder a todos los recursos de la herramienta en función de su rol en la empresa. Los empleados con el rol de «cajero» tienen acceso a todas las funcionalidades necesarias para llevar a cabo su trabajo correctamente, pero no disponen de acceso a las secciones que tratan con datos los más sensibles de la empresa. Un ejemplo de dichas secciones es el apartado de estadísticas. Los empleados con el rol de «gerente» o «administrador» tienen acceso a todas las demás funcionalidades del sistema.

- **Inicio:** aquí se verá un breve desglose de la facturación de ese día y un gráfico comparando las ventas de dicho día y del anterior, separado por horas. Desde esta parte, el usuario puede optar por acceder a muchos otros puntos del sistema.

Figura 4.6: Captura de pantalla de la sección «Home» de ERPWeb

- **TPV:** aquí es donde el usuario podrá abrir y cerrar caja. Además de eso, podrá realizar las ventas a los clientes. La TPV cuenta con una barra de búsqueda de artículos que también funciona como lector de código de barras. A la derecha está la cesta de la compra del cliente donde el empleado puedo apuntar las cantidades vendidas, el precio de venta y los descuentos. Una vez el cliente esté satisfecho, el trabajador podrá cerrar la venta y registrarla en el servidor de base de datos. Un pequeño diálogo de impresora aparecerá en la pantalla para imprimir el tique de la venta.

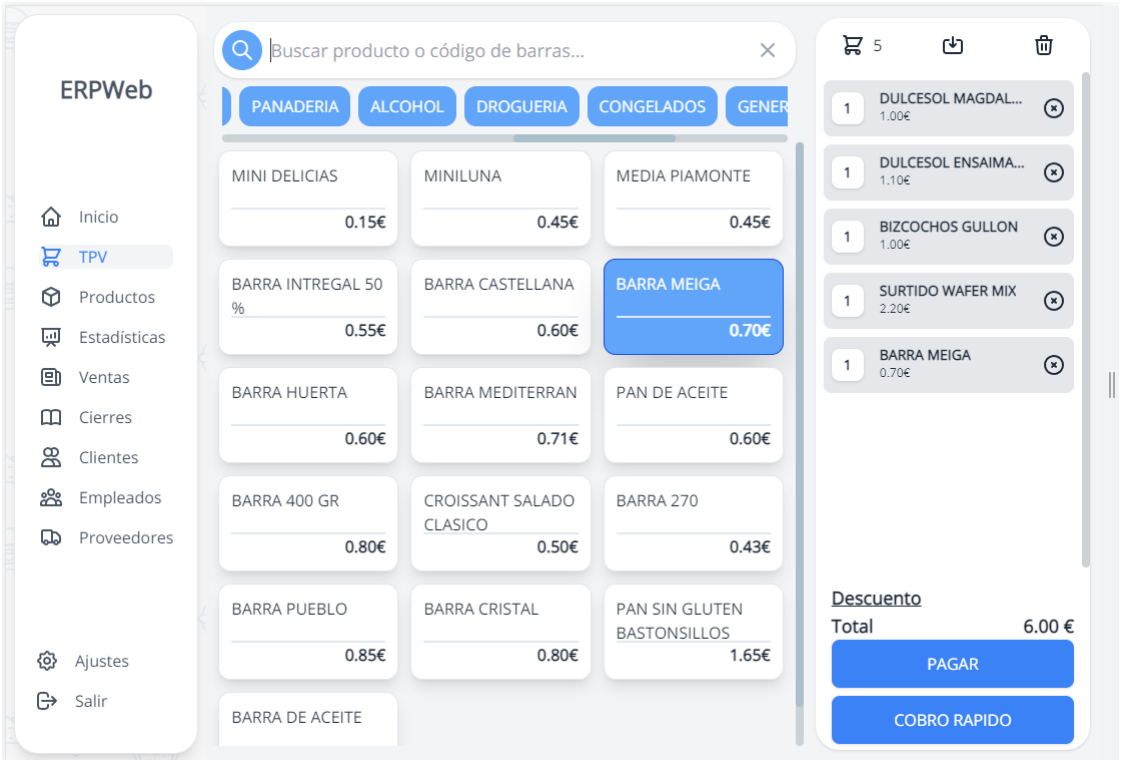


Figura 4.7: Sección de TPV

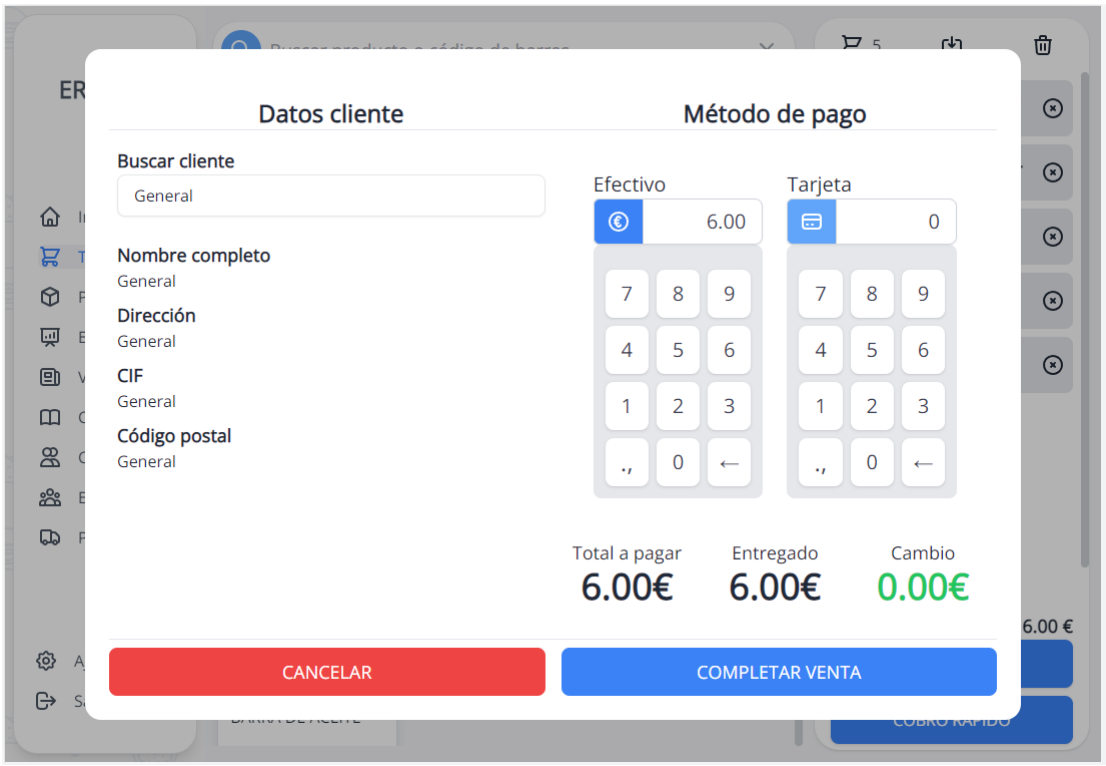


Figura 4.8: Sección para completar las ventas

- **Productos:** en este apartado del programa, los empleados pueden acceder a toda la información relacionada con los productos. Además de eso, los empleados pueden añadir, borrar, modificar y dar de baja a los artículos de la tienda. También incluye la capacidad de imprimir las etiquetas. Dichas etiquetas incluyen el nombre, el código EAN y el precio de venta. Los productos creados aquí pueden ser vendidos en la sección TPV.

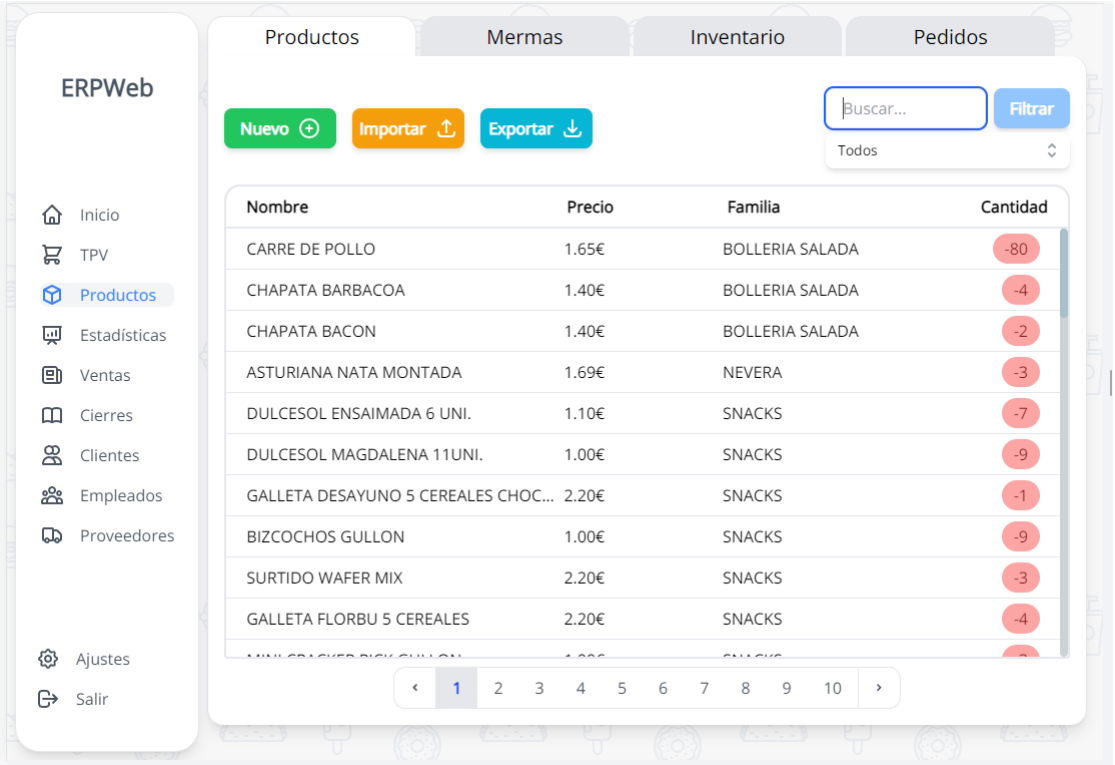


Figura 4.9: Sección de productos

- **Estadísticas:** la sección de estadísticas incluye información asociada a las ventas y los productos. Se puede buscar los datos asociados a las transacciones pasadas mediante un selector de rango de fechas. En la página de estadísticas también se puede consultar el valor de las ventas totales desglosadas por horas, los artículos más vendidos y el beneficio recaudado.

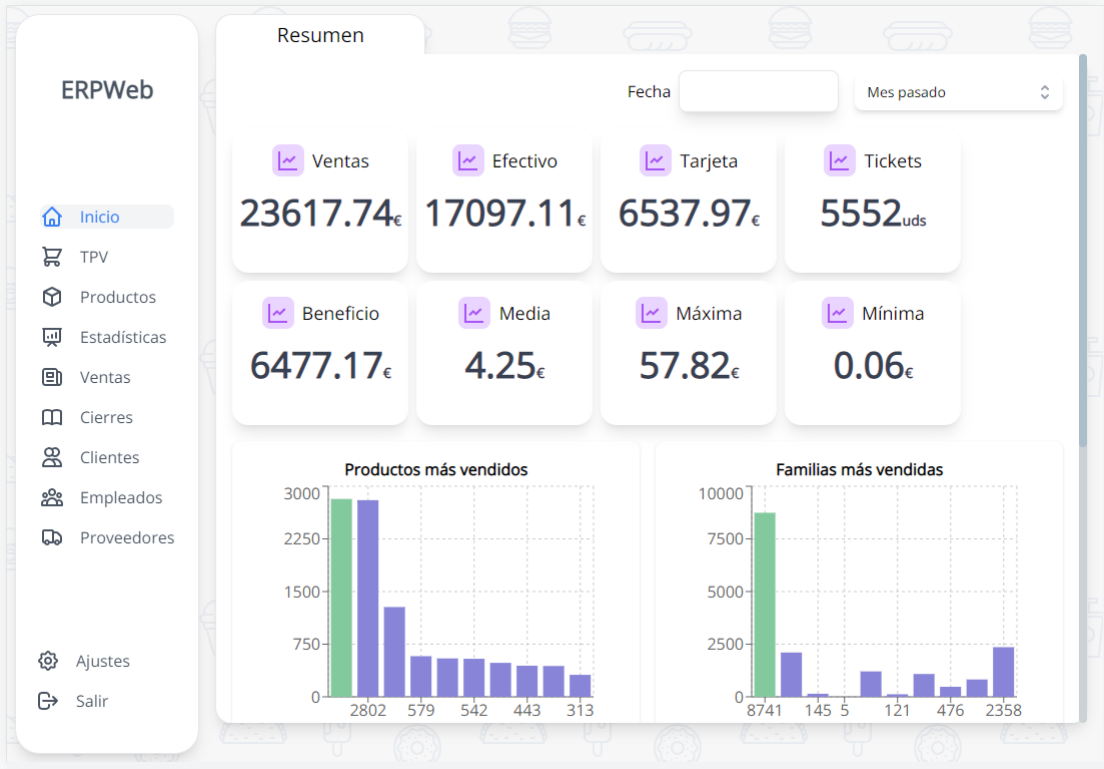


Figura 4.10: Sección de estadísticas parte 1

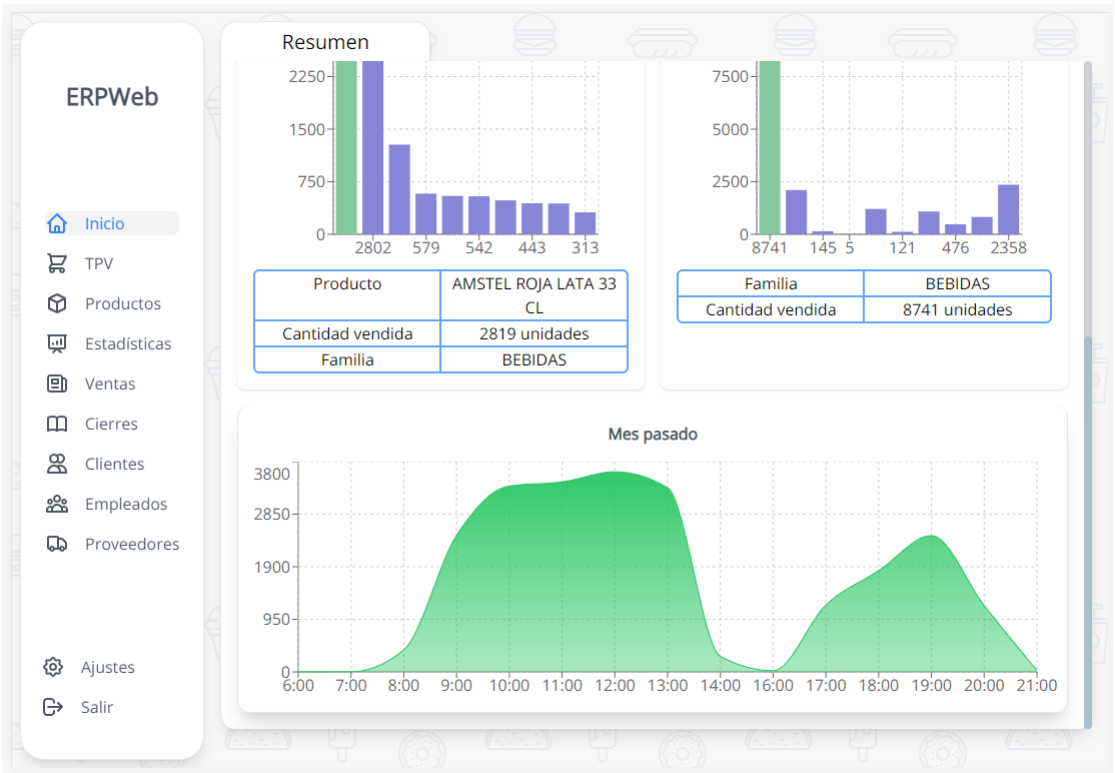


Figura 4.11: Sección de estadísticas parte 2

- **Ventas:** esta página recoge toda la información asociada a las ventas. La fecha y hora de la transacción, el valor final, el método de pago, el trabajador que la efectuó y también los artículos que se han vendido. Además de lo mencionado, las ventas se pueden devolver o reembolsar. Una devolución es llevada a cabo cuando un cliente quiere devolver algún o todos los productos. Una vez devuelto, se imprime un recibo donde se especifica el dinero a devolver al cliente y los objetos devueltos a la tienda.

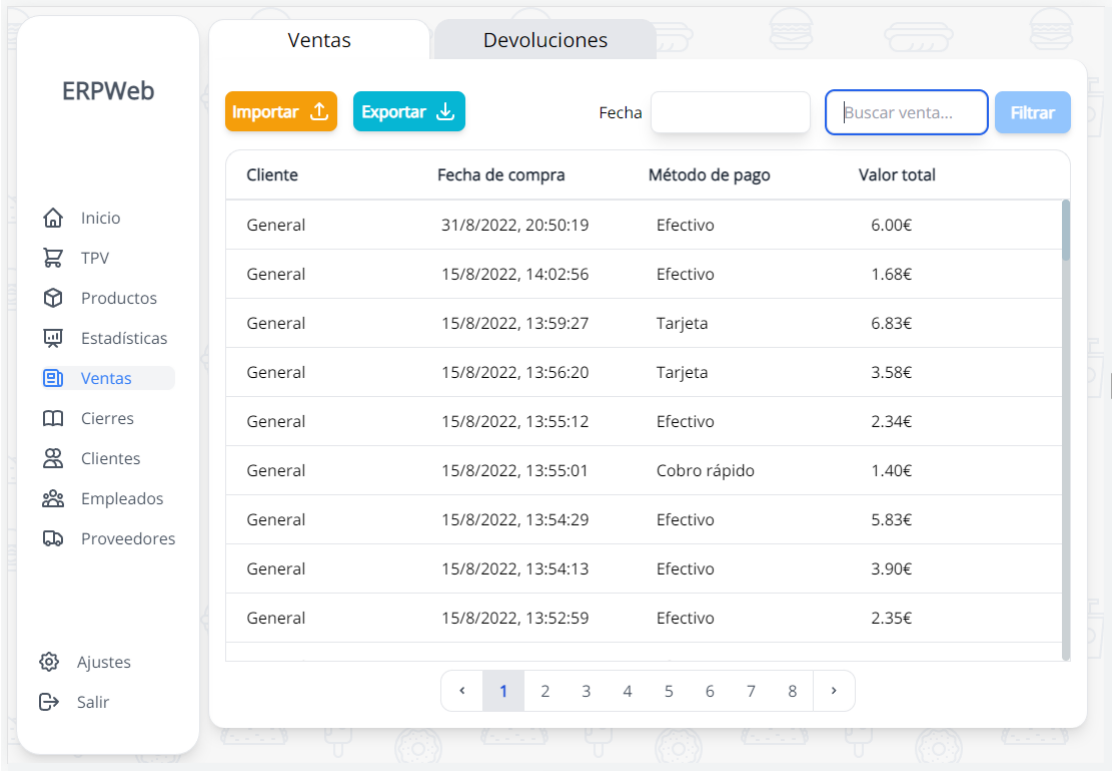


Figura 4.12: Sección de ventas y devolución

- **Clientes:** al tratarse de un pequeño comercio, los clientes mayoritarios de este establecimiento son personas que van físicamente al local. Por este motivo, la mayor parte de las ventas no llevan un cliente asociado. Por cuestiones relacionadas al pago de impuestos y Hacienda, alguno de los clientes sí necesitan estar registrados en el sistema. Estos clientes suelen ser autónomos que requieren de una factura que incluyan sus datos personales y los datos del establecimiento. Este apartado muestra a todos los clientes registrados en la base de datos. Cuando alguno de ellos realiza una compra en el local, una factura es creada automáticamente. Además de eso, los trabajadores pueden crear, borrar y modificar los datos asociados a los clientes.

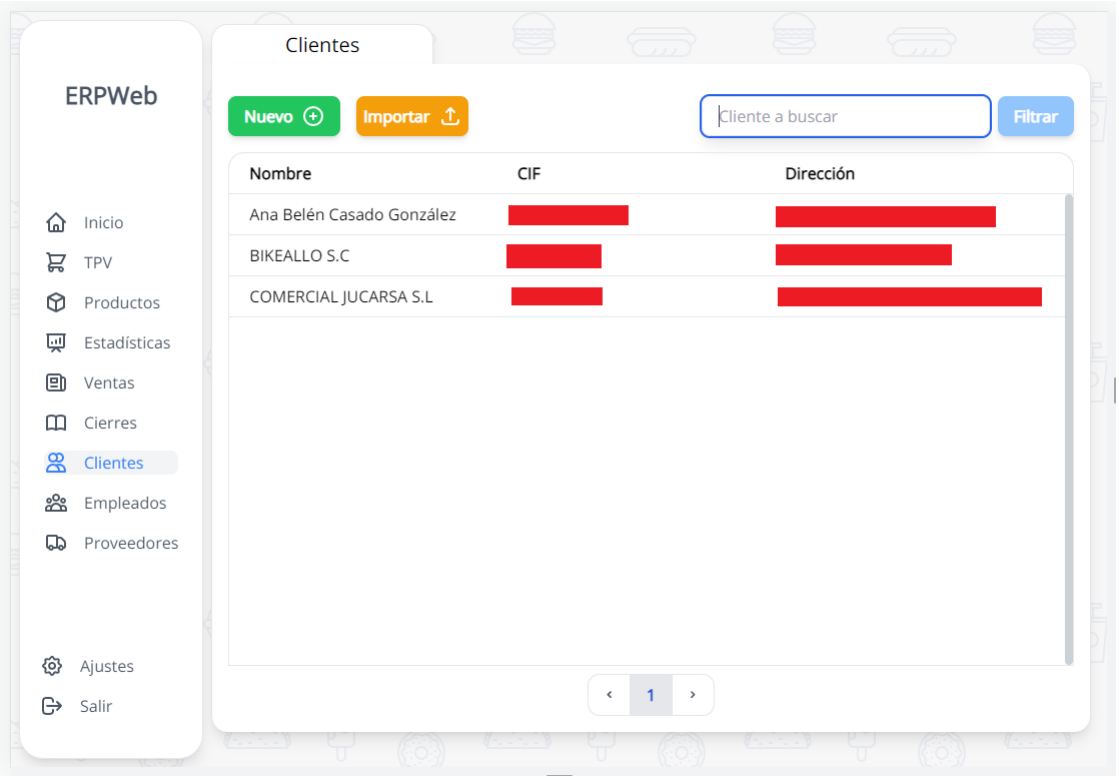


Figura 4.13: Sección de clientes

- **Cierres:** cuando la jornada laboral termina, los empleados deben contar el dinero de las cajas registradoras, recogerlo y guardarlo en la caja fuerte. Por último, para registrar el cierre, los empleados deben cerrar la TPV que hayan utilizado, apuntando el dinero real contado dentro de la caja registradora. Esta sección permite mantener un control sobre las aperturas y los cierres de las TPV.

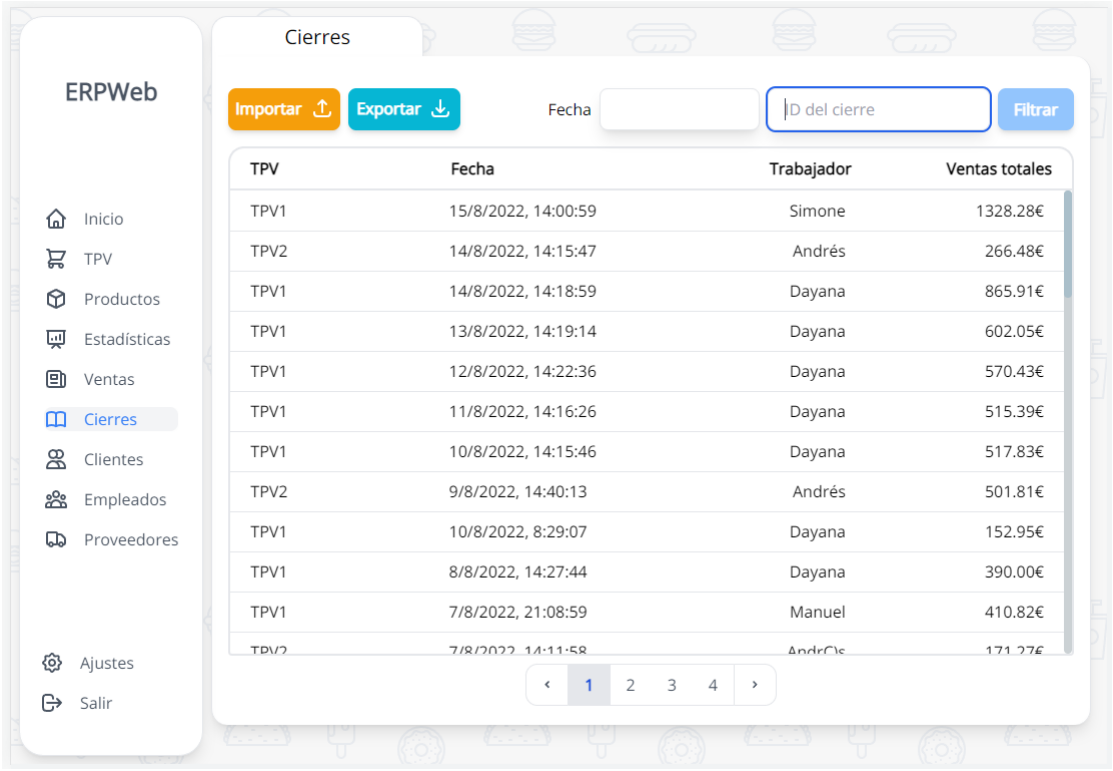


Figura 4.14: Sección de cierres

- **Empleados:** esta parte de la herramienta recoge la información de todos los empleados del sistema. Los empleados que aparecen en este apartado son todos aquellos que tienen acceso a la herramienta. Esta sección no está disponible para todos los empleados, únicamente para los gerentes y administradores. Si se desea añadir un nuevo empleado, simplemente se cliquea en el botón «Nuevo» y se rellena el formulario. Una vez rellenado, el servicio de «ERPRegistration» enviará un correo electrónico al nuevo empleado y este podrá crear la contraseña con la que podrá acceder a la herramienta.

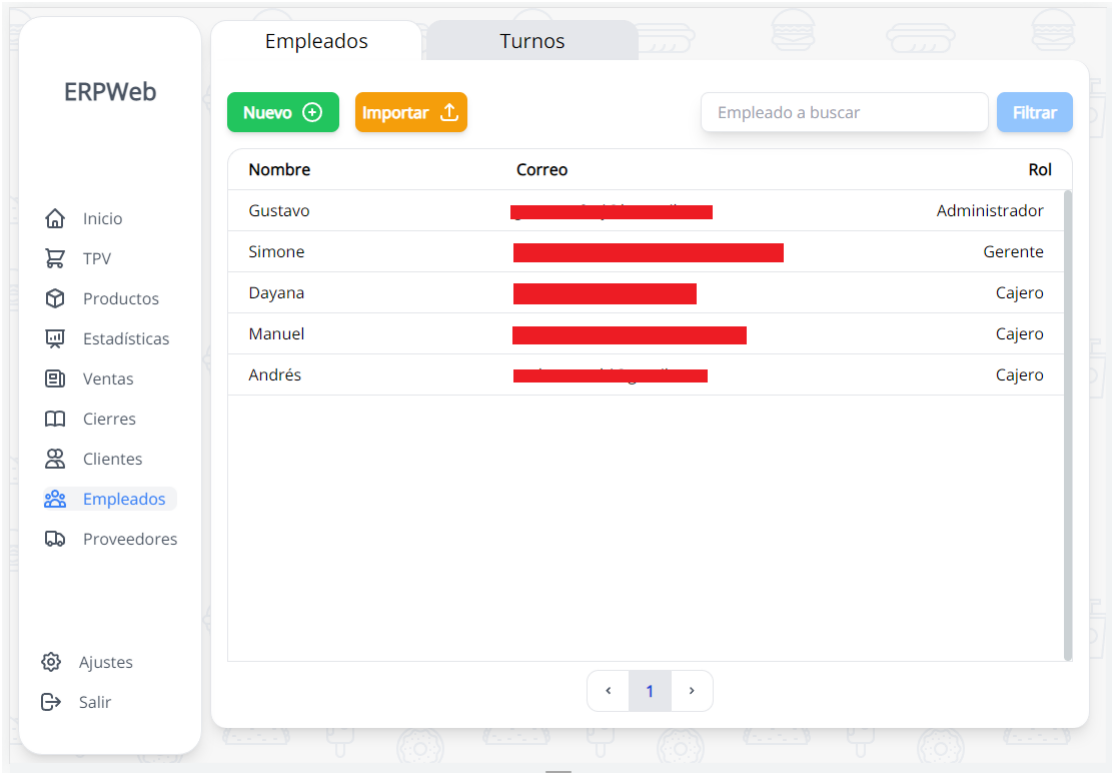


Figura 4.15: Sección de empleados

En ERPWeb se hace uso de una librería de ReactJS llamada «Toastify». Esta librería tiene como finalidad proporcionar a los usuarios de un rico sistema de notificaciones que permita notificar en pantalla todo lo que los desarrolladores necesiten. Toastify se usa a lo largo de toda la aplicación para proporcionar notificaciones a los usuarios y permitir que estos entiendan bien el contexto de la herramienta y dónde están en cada momento.

4.1.4. ERPGateway

Este es el servicio encargado de mejorar la escalabilidad y reducir la complejidad de la comunicación entre los microservicios. Está programado en Go y usa una librería llamada «Gin» para manejar las peticiones REST recibidas. La idea detrás de esta puerta de enlace es aumentar el desacople e independencia entre los microservicios, haciendo que ninguno de ellos conozca la existencia de los demás. Si una aplicación necesita de datos que están alojados en otra parte, esta lo pregunta al Gateway y este se encarga de buscar la información donde corresponda.

Asimismo, los únicos puertos abierto en toda la orquestación de este sistema es el 443 para HTTPS y el 80 para HTTP. Esto garantiza que ninguna máquina pueda hacer peticiones a los demás servicios sin pasar por ERPGateway. Esto añade una capa de seguridad en el sistema porque nadie puede acceder a información delicada de la empresa preguntando directamente al servidor de base de datos.

Requisitos esenciales:

- Request forwarding. Es necesario reenvíar las peticiones a los servicios oportunos.
- Consistencia en las respuestas. Cabe asegurar que todos los servicios respondan a la petición inicial usando un mismo formato.

4.1.5. ERPBack

Esta aplicación, hecha en NodeJS y TypeScript, se encarga de recibir peticiones por parte de la puerta de enlace. Es la encargada de realizar todas las modificaciones en la base de datos y mantener su integridad. Almacena información acerca de los productos, las ventas, las devoluciones, los empleados, los cierres, las TPV y los clientes. Una futura ampliación de este servicio incluirá la posibilidad de realizar operaciones CRUD sobre nuevos modelos de datos como los proveedores, los inventarios y los albaranes.

ERPBack crea y se conecta a una instancia de MongoDB por medio de la librería «Mongoose». Dicha instancia puede estar alojada en la nube (mediante MongoDB Atlas) o puede ser local de la propia máquina. Cabe recalcar que, al trabajar con aplicaciones desplegadas en contenedores, los datos alojados no son persistentes y dejan de existir en el momento en que el contenedor se pare. Para evitar este problema de persistencia, se pueden crear «volúmenes» en Docker donde almacenar los datos.

Requisitos esenciales:

- Consistencia de datos. Asegura que todas las inserciones, modificación y borrados de información mantengan la consistencia y sentido de los datos. ERPBack es el servicio responsable de mantener la integridad de los datos.
- Almacenar las colecciones de datos relacionadas con las ventas, los productos, los empleados, los cierres de caja y las devoluciones.
- Encriptar las contraseñas de todos los nuevos usuarios.
- Devolver información de autenticación en un JWT.

- Mantener un registro temporal. Almacenar una fecha (timestamp) en todos los documentos en el momento que sean creados y modificados.

4.1.6. ERPRegistration

A diferencia de los servicios mencionados previamente, el desarrollo de ERPRegistration fue mucho más corto y sencillo. Esta aplicación solo tiene una finalidad: ofrecer la funcionalidad de registro de usuario y cambio de contraseñas. Se usa NodeJS junto a librería «Nodemailer» para hacer uso de un servidor SMTP de pago que permite el envío y la recepción de correos electrónicos. Los correos electrónicos contienen enlaces que permiten acceder a las funcionalidades de este servicio. Por ejemplo: un nuevo trabajador necesita de un usuario y una contraseña para acceder a la herramienta. Para ello, los gerentes necesitan crear su usuario en ERPWeb. Para evitar que los gerentes pidan alguna contraseña a los empleados o creen una nueva que pueda ser fácilmente olvidada, ERPSolution se encarga de enviar un correo electrónico al nuevo empleado con un enlace que le permite confirmar su nueva cuenta y añadir una contraseña válida.

Requisitos esenciales:

- Capacidad de registrar nuevos usuarios.
- Envío de correos electrónicos.
- Capacidad de modificar contraseñas (en caso de que uno la olvide)

4.1.7. ERPAnalytics

ERPAnalytics trata de dar una solución al problema de la falta de visibilidad de los datos. ERPWeb pide al servidor de base de datos la información asociada a las ventas y los producto, dichos datos se muestran en forma de filas y columnas en ERPWeb. Para aumentar la visibilidad de los datos de la empresa y mejorar la toma de decisiones, ERPWeb necesita procesar y mostrar este tipo de información en un «dashboard». Este dashboard necesita mostrar valores importantes para la empresa, como por ejemplo, el beneficio en tiempo real o el valor medio de las ventas en un periodo de tiempo.

Requisitos esenciales:

- Calcular un resumen de ventas en función de la fecha.
- El resumen debe incluir información de las ventas y productos vendidos separada por horas.
- El resumen debe calcular máximos, mínimos, medias, número de productos vendidos, beneficios, IVA pagado y otros datos de relevancia.

Este servicio accede a los datos guardados en la base de datos y aplica diversas funciones para calcular los valores necesarios para el dashboard. Ya que el tiempo de ejecución de una petición a este servicio depende mucho del tamaño de los datos a analizar, se ha optado por desarrollarlo usando un lenguaje compilado y de alto rendimiento. Para cumplir con este requisito, se ha usado el lenguaje de programación Go en lugar de JavaScript con NodeJS. Esto se debe a que Go, además de ser un

lenguaje tipado y compilado, es capaz de ofrecer mucho mejores prestaciones que NodeJS.

La estructura del JSON devuelto al llamar el «summary» endpoint es el siguiente:

```
You, hace 2 semanas | 1 author (You)
type Summary struct {
    VentasPorHora             []VentasPorHora      `json:"ventasPorHora"`
    ProductosMasVendidos      []ProductoMasVendido `json:"productosMasVendidos"`
    FamiliasMasVendidas       []FamiliaMasVendida  `json:"familiasMasVendidas"`
    Beneficio                 float64              `json:"beneficio"`
    TotalVentas               float64              `json:"totalVentas"`
    TotalEfectivo             float64              `json:"totalEfectivo"`
    TotalTarjeta              float64              `json:"totalTarjeta"`
    NumVentas                 int                  `json:"numVentas"`
    MediaVentas               float64              `json:"mediaVentas"`
    VentaMinima               float64              `json:"ventaMinima"`
    VentaMaxima               float64              `json:"ventaMaxima"`
    MediaCantidadVenida       float64              `json:"mediaCantidadVenida"`
    CantidadProductosVendidos int                  `json:"cantidadProductosVendidos"`
    DineroDescontado          float64              `json:"dineroDescontado"`
    IVAPagado                 float64              `json:"ivaPagado"`
}
```

Figura 4.16: Estructura de summary JSON

Como se puede observar, el tipo Summary contiene tres atributos (VentasPorHora, ProductosMasVendidos, FamiliasMasVendidas) de tipo array.

«VentasPorHora» , como su nombre indica, recoge información acerca de las transacciones hechas a lo largo del día. Dichas transacciones son agrupadas por horas y muestran los datos acerca del beneficio de las ventas en una hora determinada, el total de esas ventas y otros elementos de interés.

```
You, hace 2 meses | 1 author (You)
type VentasPorHora struct {
    Hora                 string  `json:"hora"`
    BeneficioHora        float64 `json:"beneficioHora"`
    TotalVentaHora       float64 `json:"totalVentaHora"`
    TotalEfectivoHora    float64 `json:"totalEfectivoHora"`
    TotalTarjetaHora     float64 `json:"totalTarjetaHora"`
    ProductosVendidosHora int    `json:"productosVendidosHora"`
    DineroDescontadoHora float64 `json:"dineroDescontadoHora"`
}
```

Figura 4.17: Estructura de «ventas por hora» JSON

«ProductosMasVendidos» es una lista de productos vendidos a lo largo del día. Estos productos son los que más salidas han tenido hasta el momento. Por defecto esta lista contiene los diez productos más vendidos.

«FamiliasMasVendidas» es similar a «ProductosMasVendidos», salvo que en lugar de recoger información de los productos vendidos, recoge información acerca de las familias (o categorías) más vendidas. También informa de la cantidad vendida de cada familia, lo cual se muestra en el dashboard de ERPWeb.

```
You, hace 2 meses | 1 author (You)
type ProductoMasVendido struct {
    ID              string `json:"_id"`
    Nombre          string `json:"nombre"`
    Ean             string `json:"ean"`
    Familia         string `json:"familia"`
    CantidadVendida int    `json:"cantidadVendida"`
}

You, hace 2 meses | 1 author (You)
type FamiliaMasVendida struct {
    Familia         string `json:"familia"`
    CantidadVendida int    `json:"cantidadVendida"`
}
```

Figura 4.18: Estructura de «producto más vendido» y «familia más vendida» JSON

4.1.8. Desarrollo

El desarrollo de toda la solución tuvo lugar en un plazo de once meses, con un total estimado de mil doscientas horas invertidas. La mayor parte del esfuerzo en este proyecto ha sido trasladado a la página web (ERPWeb) con más de quinientos commits realizados y en producción. La mayor motivación detrás de este proyecto ha sido proporcionar herramientas claras, fáciles de aprender y usar a los trabajadores del comercio.

Desgraciadamente, al ser un proyecto desarrollado por una persona, muchos de los elementos necesarios para el correcto funcionamiento de la solución han sido aprendidos y estudiados durante el desarrollo, lo cual ha implicado mucho tiempo dedicado al aprendizaje e investigación de librerías y metodologías en lugar del desarrollo de la propia herramienta.

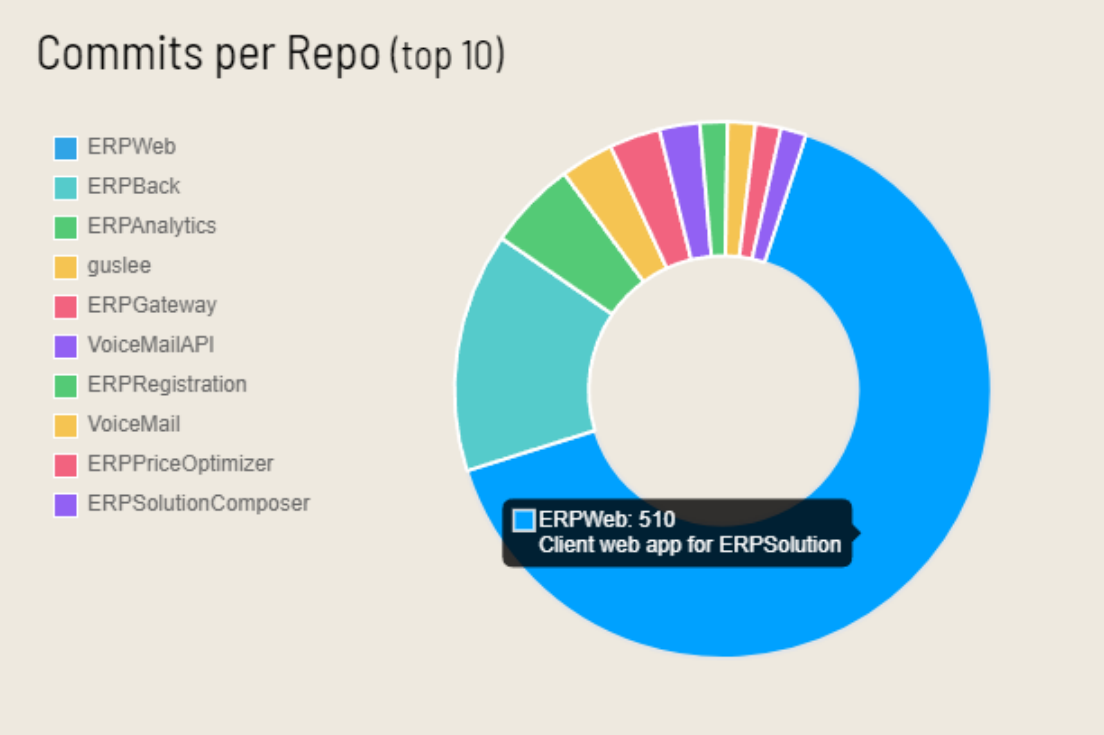


Figura 4.19: Resumen de los repositiorios en Github

A diferencia de la mayoría de los proyectos actuales, especialmente las start-ups, esta solución no ha seguido ninguna metodología de desarrollo ágil (como SCRUM o Kanban) tal y como están planteadas. Ya que el tiempo disponible del programador para desarrollar este producto era muy variable e inconsistente, se ha decidido no hacer uso de sprints con fechas fijas. Al no tener que presentar un producto mínimo viable en las etapas tempranas del proyecto, se ha decidido integrar las distintas funcionalidades de forma gradual. Una vez terminada alguna nueva funcionalidad, se pedía a los clientes del proyecto que hicieran uso del sistema y aportasen comentarios y opiniones acerca del trabajo. De esta forma, a lo largo de esos once meses, se comprobó que el software cumplía con los requisitos del proyecto y que esta aplicación era lo que necesitaban los clientes.

A continuación, el diagrama de Gantt relacionado con el proceso de desarrollo. Como se puede observar, ERPWeb y ERPBack fueron los dos repositorios que más tiempo de desarrollo han necesitado. Esto se debe al hecho de que ERPWeb necesita nuevas páginas y pestañas cada poco tiempo para reflejar los datos proporcionados por todos los nuevos servicios. ERPBack sustituyo el uso de REST por GraphQL y se tuvo que modificar mucho código. ERPGateway y ERPAnalytics fueron las dos aplicaciones más cortas de programar. Por otro lado, ERPRegistration es una aplicación sencilla con poco código detrás, pero desarrollarla supuso mucho tiempo porque gran parte de ese tiempo se fue al intentar implementar un servicio mediante el uso SMTP. Finalmente esa idea se desechó y se hizo uso de un servicio adicional de cloud computing.

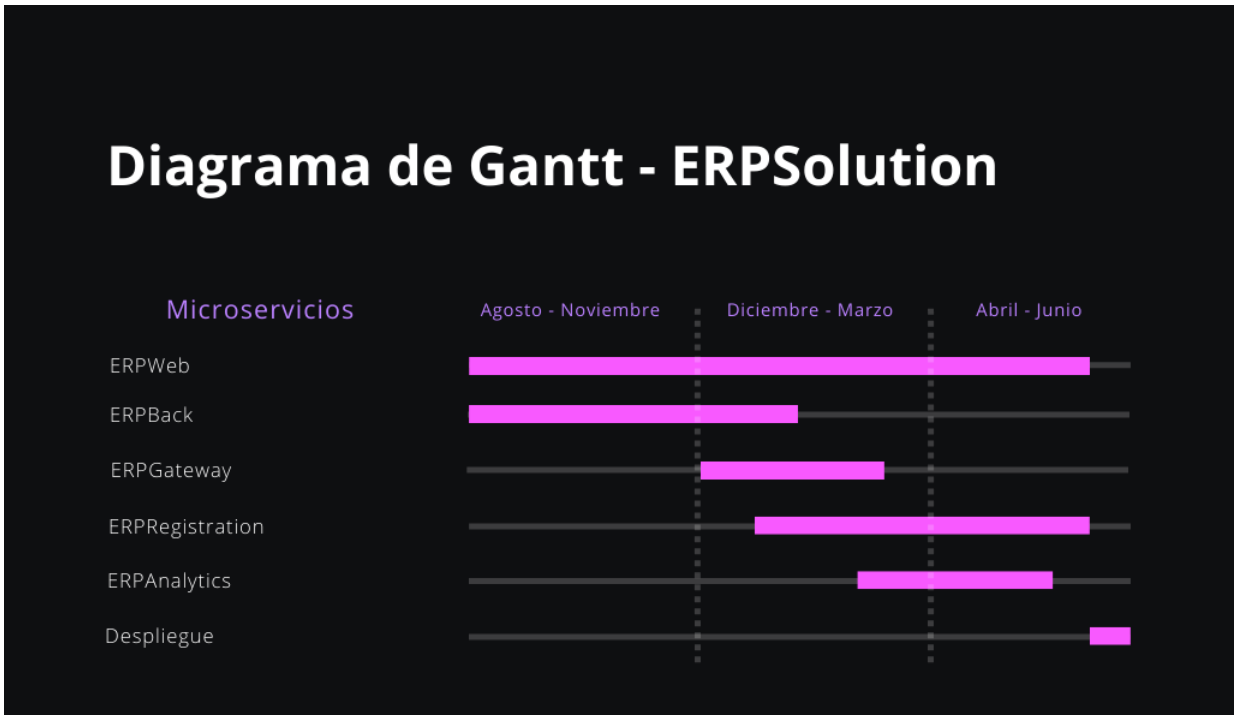


Figura 4.20: Diagrama de Gantt de ERPSolution

4.1.9. Seguridad

Se ha tenido en cuenta muchas herramientas, metodologías, tecnologías y buenas prácticas con tal de mantener la integridad y anonimato de los datos personales de los trabajadores, clientes y proveedores relacionados con este sistema. A continuación se enumerarán todas los posibles fallos de seguridad que podrían existir y cómo se han solucionado para ERPSolution.

- **JWT:** muchas páginas webs necesitan almacenar información para mejorar la experiencia de los usuarios o aportar algún tipo de servicio. Desde el modo oscuro en Github hasta la autenticación en sitios webs como Twitter. Los navegadores ofrecen tres herramientas que ayudan a cumplir esta finalidad. Las «cookies», el «sessionStorage» y el «localStorage». Para este proyecto se ha elegido almacenar la información en los navegadores mediante el uso de cookies. Una

de las desventajas que tiene esta opción es la facilidad que ofrece a cualquier usuario almacenar y modificar la información asociada a cada cookie. Esto puede conllevar a diversos fallos de seguridad. Para evitar este problema, se ha decidido transformar toda la información de las cookies en un JWT (JSON Web Tokens). Esto permite que, en caso de que alguien modifique una cookie de autenticación, el servidor la rechace porque esta no estaría firmada por el servidor.

- **Man in the middle attack:** también conocido como «ataque de intermediario», suele ocurrir en sitios web poco seguros. Los navegadores modernos suelen advertir a los usuarios cuando visitan un sitios web de este estilo. Normalmente la información del cliente viaja directamente al servidor, desde envíos de formularios como inicios de sesión. El ataque de intermediario consiste en interceptar el tráfico proveniente desde alguno de estos dos puntos, leer y/o modificar la información en dicha comunicación y entregar el paquete interceptado al destinatario final. Como se puede observar, este es un grave fallo de seguridad. Si un usuario intenta iniciar sesión en un sitio web poco seguro, tanto el nombre de usuario como la contraseña podrían verse expuestos a cualquier persona que intercepte ese tráfico. Para solventar este problema, la comunicación entre el cliente y el servidor necesita estar encriptada. Para ello, los servidores deben instalar un certificado SSL que aseguran dicho canal de comunicación, habilitando el tráfico HTTPS en lugar de HTTP. En el caso de ERPWeb, al estar alojado en Vercel, esta empresa asegura la comunicación entre el cliente y el sitio web instalando un certificado SSL automáticamente. Para el resto de ERPSolution, al estar alojado en Amazon Web Services (AWS) en una instancia de máquina virtual (EC2), se ha necesitado hacer uso de uno de sus servicios para permitir instalar un certificado SSL y permitir el tráfico sobre HTTPS.
- **Contraseñas:** al trata con inicios de sesión, es necesario almacenar y leer información sensible acerca de los empleados. Siempre que un trabajador inicia sesión en ERPWeb, la herramienta debe comprobar la validez de la contraseña introducida haciendo peticiones contra el servidor de base de datos. Esto implica que todas las contraseñas deben de almacenarse en la base datos de MongoDB. Estas contraseñas no se pueden almacenar en texto plano porque, en caso de haber un fallo de seguridad capaz de proporcionar una entrada a algún atacante, este tendría acceso a los correos y contraseñas de todos los usuarios. Para minimizar los daños en caso de que eso ocurra, las contraseñas de todos los empleados deben estar encriptadas. Cuando alguno de los usuarios intenta iniciar sesión, la contraseña que ha escrito se encripta y se compara con la contraseña encriptada en la base de datos. Si son iguales, se le concede acceso, en caso de que no lo sea, se le niega el acceso.

4.1.10. Testing

Para el testing de la aplicación web se han elegido estas herramientas:

- **Vitest:** librería similar y también compatible a «Jest» para las comprobaciones mediante test unitarios. Vitest es popular por estar bien integrada con la herramienta «Vite». Vitest permite el desarrollo de pruebas unitarias de cualquier función, clase o componente dentro del proyecto. No se puede usar en todos los microservicios porque solo funciona sobre NodeJS.

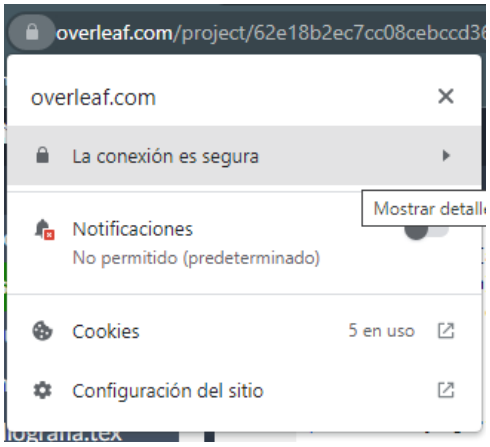


Figura 4.21: Sitio seguro

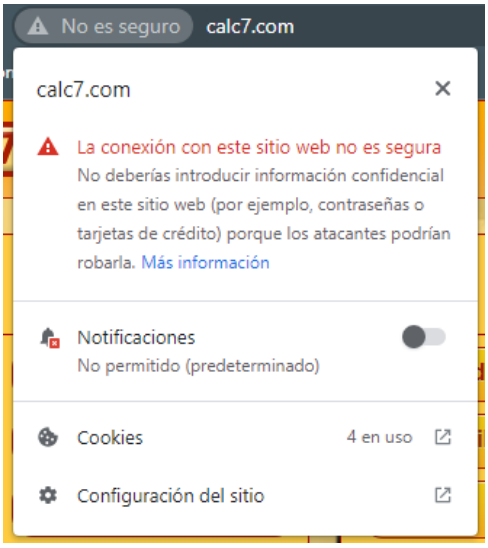


Figura 4.22: Sitio no seguro

- **Cypress:** para las pruebas «End-to-end» (o E2E) de la página web. Cypress es una herramienta muy útil que permite a los desarrolladores probar los componentes de interfaz de usuario y también casos de usos por completo. Al ser una herramienta E2E, el programador escribe las acciones que realizan los usuarios y Cypress se encarga de ejecutarlas. En caso de haber algún error, Cypress lo muestra junto a una grabación de dicho error.
- **Postman:** aplicación web y de escritorio que permite realizar llamadas HTTP y HTTPS de forma muy cómoda. No es una herramienta de pruebas como tal, pero se hizo uso de esta aplicación para comprobar el correcto envío de datos entre las distintas APIs REST.

4.1.11. Despliegue y costes

Una vez terminado el desarrollo de la mayor parte de la solución, era hora de desplegar el sistema y hacerlo accesible a la web. Con tal de lograr este objetivo, se ha considerado tres formas de despliegue. A continuación se hará una breve comparación entre dichas opciones y la conclusión final.

1. **Servidor propio:** sin duda la forma más barata a largo plazo. Tiendas como PCComponentes o Amazon ofrecen servidores por precios que, al cabo de dos años aproximadamente, acaban siendo más económicos que usar una infraestructura de computación en la nube. Por desgracia, montar un servidor desde cero supone mucho trabajo relacionado con la instalación y el mantenimiento. Otro punto negativo es la dificultad de escalado. Si hace falta más memoria principal, es necesario comprarla y ampliarla manualmente. Otro aspecto negativo está en el hecho de que toda la infraestructura pertenece a la red de la tienda. Cualquier acceso desde fuera no sería posible.
2. **Servidor propio junto a una IP abierta:** muy similar al apartado anterior, salvo por el hecho de que se dispondría de una IP pública abierta. Para ello sería necesario hablar con los proveedores de conexión a internet del local, lo cual podría cambiar con el tiempo. Supondría más trabajo añadido sobre el tratamiento

del servidor pero a cambio el servicio sería accesible desde cualquier parte del mundo.

3. **Computación en la nube:** la opción más económica a corto plazo. A día de hoy, hacer uso de servicios en la nube agiliza mucho el despliegue de las soluciones. Esta opción permite obviar muchos aspectos esenciales de la administración de sistemas que, de otra forma, habrían consumido mucho tiempo del desarrollo. Muchos proveedores ofrecen planes y descuentos, en caso de AWS, el primer de uso de sus máquinas virtuales salen gratis (para instancias del tipo t2.micro)

ERPSolution ha estado en producción durante tres meses. Los costos por uso de los servicios de AWS ascienden a 0.61 dólares al mes. Los costos por uso de los servicios de Vercel para el hosting de la página web ascienden a 20 dólares al mes.

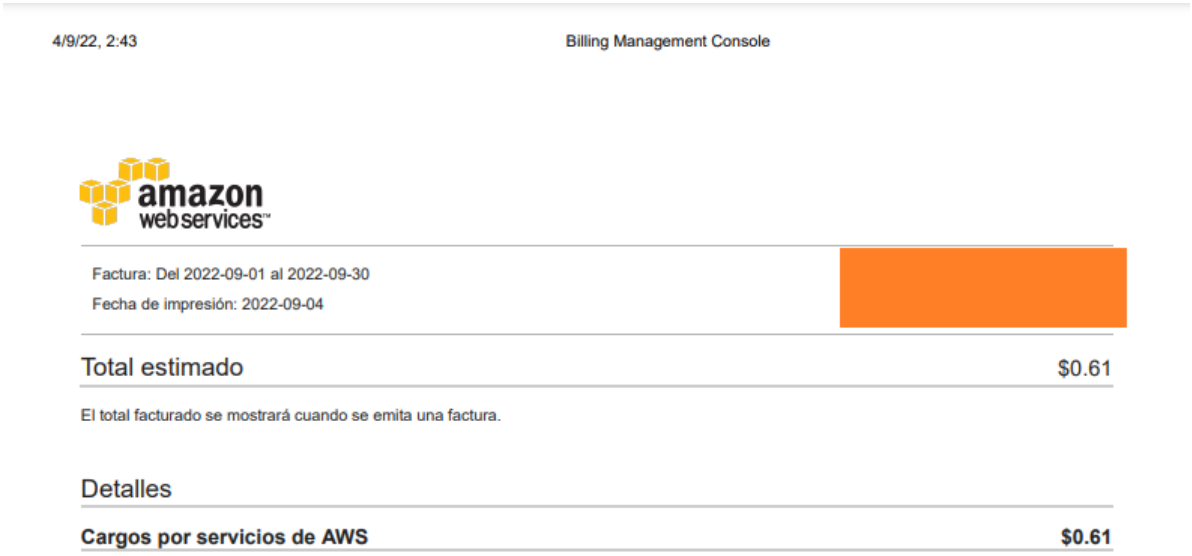


Figura 4.23: Factura de agosto de AWS

Figura 4.24: Factura de agosto de AWS

4.1.12. Resultado final

Como resultado final, se ha obtenido un sistema robusto y bien conectado. La aplicación web es fácil de usar y tiene una baja curva de aprendizaje. Además de eso, la herramienta es rápida y funciona bien en ordenadores poco potentes. Se siguen buenas prácticas de protección de datos que permiten mantener la seguridad de la información sensibles de los empleados. Se ha conseguido crear una herramienta económica cuyo bajo precio cumple con los requisitos de los dueños de la pequeña empresa. Además de eso, la herramienta está completamente en español, lo cual es otro punto a favor.

Al disponer de un API independiente de la aplicación cliente, es posible generar nuevas herramientas para automatizar y mejorar la calidad del trabajo y la eficiencia de la empresa. Un trabajo futuro será el de crear una aplicación móvil para los registros del inventario.

4.2. CRISP-DM

CRISP-DM es una metodología estándar para extraer valor de los datos mediante enfoques bien definidos por expertos en el campo. Su finalidad es la de orientar los trabajos de minería de datos haciendo una división del proyecto en seis pasos. Además, CRISP-DM es una metodología cíclica. Una de las ideas detrás de esta metodología es la de iterar sobre fases previas del proyecto con tal de modificar, mejorar o hasta replantear la preparación de los datos, el modelado o el problema de negocio.

Para ampliar este trabajo de fin de máster, se ha optado por complementar el sistema ERP con un modelo de minería de datos. La idea es integrar nueva funcionalidad a la página web que solucione alguno de los problemas de negocio existentes.

4.2.1. Fase I. Comprensión del negocio

Para este caso, es necesario comprender todos los aspectos relevantes de la mejor forma posible. La tienda se sitúa en una gran avenida de Valencia, cerca de la Ciudad de las Artes y las Ciencias. Los clientes suelen ser turistas que simplemente pasan por el local, personas que vienen por vacaciones, los ciudadanos del propio barrio y también los estudiantes de un instituto cercano que vienen para comprar el almuerzo.

El establecimiento cuenta con dos cajas registradoras, dos hornos, cinco neveras (dos para congelar y tres para enfriar) y diversas estanterías que contienen los siguientes tipos de productos: bollería salada, bollería dulce, snacks, alimentación, bebidas, vinos, alcohol y droguería.

Tras una entrevista con los dueños del local, los problemas de negocio que les gustaría resolver son los siguientes:

1. **¿Qué nuevos productos podrían aumentar las ventas del negocio?:** este problema requiere de datos que no se poseen actualmente. Además de eso, se necesitaría mucha cantidad de tiempo de desarrollo y tiempo para saber si la inclusión de los nuevos productos han aumentado o no el número de ventas.
2. **¿Qué ofertas personalizadas les puede interesar a los clientes habituales?:** este problema requiere de datos acerca de las compras de los clientes y de ofertas usadas en el pasado. Debido a la naturaleza del negocio, son pocos los clientes registrados. Por tanto, este es un problema complicado de resolver dada la falta de datos.
3. **¿Qué productos se les puede ofrecer a los clientes que estén pasando por caja?:** este problema es abordable porque se tienen los datos de las ventas con sus respectivos productos vendidos. Se podría crear algún tipo de sistema de recomendación que funcione en base a los productos de su cesta de la compra. Es similar a la sección de «productos que te pueden interesar» que tiene Amazon.
4. **¿Qué tipo de ofertas se pueden ofrecer para aumentar el número de clientes y el valor de las ventas?:** este caso es bastante interesante e interesa mucho a los dueños de la tienda, por desgracia no se disponen de datos acerca de las ofertas pasadas, por tanto no se puede trabajar con este modelo.
5. **¿Qué margen deberían de tener los nuevos productos de la tienda para mantener un precio competitivo y que genere buenos beneficios?:** este caso conlleva varias dificultades. La primera la falta de datos acerca de los precios

de la competencia. Para mantener precios competitivos, es necesario tener precios más bajos que los mercados y supermercados cercanos. Pero también es necesario tener un margen suficiente para poder generar beneficios. Este es un problema demasiado complejo para los pocos datos y el poco tiempo de los que se disponen.

6. **¿Cuánto facturará la tienda la próxima semana o el próximo mes?:** esta es una pregunta muy importante que se hacen los trabajadores de este sector. También conocido como «sales forecasting», la idea consiste en usar datos del pasado para intentar predecir las ventas que tendrá lugar en un futuro. Este tipo de información es muy importante para las empresas ya que permite que estas planifiquen adecuadamente los presupuestos para futuras contrataciones, nuevos equipamientos, gastos de mantenimiento y muchas otras cosas. Este problema no es difícil de resolver, se trata de un modelo de regresión ya que se intenta predecir un valor continuo.

7. **¿Cuánto se venderá de cada producto esta semana? ¿Cuál es el mejor momento para realizar un pedido de un determinado producto?:** similar al problema anterior, se trata de un modelo de regresión donde se intenta predecir la cantidad vendida de un determinado producto de cara al futuro. Disponer de este tipo de información es de gran ayuda para la persona encargada de mantener el almacén. Saber en qué momento se puede agotar un producto es vital para realizar un pedido a los proveedores adecuadamente. Esto evita que los clientes intenten comprar un artículo agotado y que los trabajadores tengan problemas para ubicar las unidades sobrantes en el almacén.

El problema número **seis** y número **siete** tienen mucho valor para los empresarios ya que aportan información relevante que puede ayudar en la toma de decisiones. Lamentablemente, se requieren muchos datos que actualmente no están disponibles. Esto se debe al hecho de que ambos son problemas de estimación de futuros y para ello es necesario una cantidad de datos que permita encontrar patrones cíclicos en la recaudación. Al disponer de once meses de ventas, no se pueden encontrar patrones ni tendencias significativas en los ingresos del negocio. Para obtener este tipo de información, sería necesario disponer de, como mínimo, dos años de ventas. Esto permitiría determinar con más precisión los valores por meses, detectar tendencias y patrones temporales. Un modelo que no disponga de datos suficientes dará resultados con precisiones muy bajas y que reflejen erróneamente la realidad. De todos los problemas de negocio mencionados previamente, el más viable es el número tres.

El problema número tres se suele plantear por dos caminos diferentes:

- Se puede ofrecer productos a los clientes en base a la información de sus antiguas compras. Para ello es necesario segregar a los clientes registrados en diferentes grupos. Este caso es modelado mediante algoritmos de aprendizajes no supervisados, por ejemplo el método de clustering «K-Means».

- Se puede buscar patrones frecuentes de ventas y, en base a esa información y los productos en el carro de la compra del cliente, llegar a ofrecer artículos que le podría interesar. Este problema se suele plantear mediante reglas de asociación. Este caso en particular es conocido como «MBA» (Market basket analysis).

Por tanto, en este proyecto de minería de datos se abordará el problema «¿Qué productos se puede ofrecer a los clientes que estén pasando por caja?». La idea consiste

en aumentar el grueso de las ventas diarias realizando una técnica de marketing conocida como «cross-selling» (o venta cruzada). El cross-selling pretende vender productos complementarios a los que esté comprando el cliente en ese momento. Un ejemplo muy sencillo es ofrecer tomate frito a un cliente que esté comprando macarrones, a sabiendas de que ambos productos suelen comprarse juntos.

A nivel de aplicación, se añadirá un botón que permita a los empleados ofrecer productos a los clientes en base al contenido actual de su cesto de la compra.

4.2.2. Fase II. Estudio y comprensión de los datos

La finalidad de esta segunda fase es realizar un estudio inicial de los datos con tal de establecer una primera toma de contacto con el problema de negocio y la información disponible. En esta sección se hará un análisis exploratorio de los datos con tal de encontrar patrones y comprobar la validez del conjunto de datos disponibles.

4.2.2.1. Estructura de datos

A continuación, se mostrará el esquema de los datos almacenados en el sistema. Todas las ventas se almacenan como un documento en este formato:

- **Id:** se trata de una cadena única de 24 caracteres. Sirve para identificar los documentos. En un contexto relacional, este identificador sería equivalente a una clave primaria. MongoDB es el encargado de asignar dicha cadena a cada nuevo documento insertado, asegurando que sea único en toda la colección.
- **Productos:** lista de productos vendidos al cliente. Cada producto vendido almacena información acerca de su nombre, su precio de compra y de venta, el margen aplicado a la venta, la familia del producto (o categoría), el nombre del proveedor, la cantidad vendida, el código de barras (EAN) y el IVA de dicho producto.
- **DineroEntregadoEfectivo:** representa la cantidad de dinero en efectivo entregado por parte del cliente final.
- **DineroEntregadoTarjeta:** representa la cantidad de dinero en tarjeta entregado por parte del cliente final.
- **PrecioVentaTotalSinDto:** el valor final de la venta antes de aplicar descuentos.
- **PrecioVentaTotal:** el valor final de la venta después de aplicar los distintos descuentos.
- **Cambio:** el dinero en efectivo a devolver al cliente.
- **VendidoPor:** muestra los datos asociados al trabajador que ha realizado la venta. Contiene su nombre, apellidos, DNI, rol y correo electrónico.
- **ModificadoPor:** muestra los datos del trabajador que haya modificado la venta.
- **tipo:** representa la forma de pago elegida por el cliente. Son cadenas de texto que pueden tomar los valores siguientes Çobro rápido", .Efectivo", "Tarjetaz "Fraccionado".
- **DescuentoEfectivo:** cantidad de dinero efectivo descontado del valor de la venta.

- **DescuentoPorcentaje:** cantidad porcentual descontado del valor de la venta.
- **Tpv:** representa el identificador de la TPV donde se ha realizado la venta.
- **CreatedAt:** contiene una cadena numérica que representa la fecha y hora de la venta. Está en formato «epoch» (los milisegundos que han pasado desde el 1 de enero de 1970).
- **UpdatedAt:** contiene una cadena numérica que representa la fecha y hora de la última modificación de la venta.

Figura 4.25: Esquema de las ventas

En cambio, los productos almacenados en el sistema siguen el siguiente esquema:

- **Id:** identificador de cada producto en la base de datos. se trata de una cadena de 24 caracteres única en toda la colección. Como se ha mencionado en el caso de las ventas, esta cadena es creada automáticamente por MongoDB, asegurando así su unicidad.
- **Nombre:** cadena de texto que contiene un nombre descriptivo del producto en cuestión.
- **Proveedor:** nombre descriptivo del proveedor del producto.
- **Familia:** categoría del producto. Un ejemplo de categoría son las bebidas donde entrarían productos como el agua o la cerveza.
- **PrecioVenta:** número en coma flotante que representa el precio de venta final. Lo que pagan los clientes por unidad.
- **PrecioCompra:** número en coma flotante que representa el coste de compra al proveedor.

- **Iva:** valor porcentual en coma flotante. Representa el impuesto sobre el valor añadido del producto.
- **Ean:** cadena de texto que contiene el código de barras del producto. El código de barras suele estar en formato EAN-13.
- **Margen:** valor porcentual en coma flotante. Representa el margen de beneficio aplicado al producto.
- **Alta:** valor booleano que indica si el producto se encuentra o no a la venta.
- **Cantidad:** número entero que representa la cantidad en almacén de dicho producto.
- **CantidadRestock:** número entero que representa la cantidad de producto que se considera necesario para realizar el siguiente pedido.
- **CreatedAt:** valor numérico entero que contiene la fecha y la hora del momento en el que se añadió el producto al sistema. La fecha está en formato «epoch»
- **UpdatedAt:** valor numérico entero que contiene la fecha y la hora del momento en el que se modificó el producto. La fecha está en formato «epoch»

Figura 4.26: Esquema de los productos

4.2.2.2. Exploración de los datos

A continuación se hará un análisis exploratorio de los datos. El objetivo detrás de este análisis es observar y entender mejor la relación entre los datos, buscar anomalías y comprobar que la información reflejada en el análisis representa la información del mundo real. Para ello haremos uso de gráficas que puedan ser de ayuda para entender la información.

Primero veremos a qué horas suelen venderse los artículos. Con esta información podremos saber si los datos concuerdan con el horario de apertura y cierra del establecimiento.

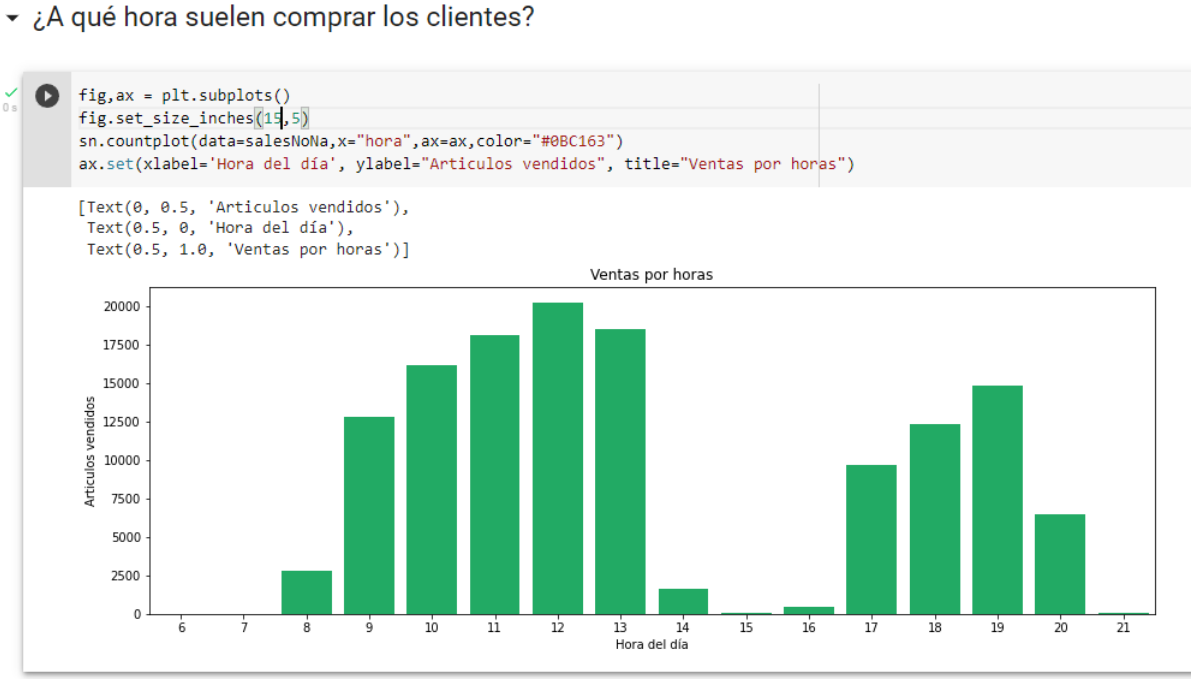


Figura 4.27: Ventas por hora

Como se puede observar en el gráfico, existen ventas desde las ocho de la mañana hasta las dos del mediodía. Luego hay un bajón de dos del mediodía hasta las cuatro de la tarde donde no existen ventas (o son muy pocas). Por último, de cinco de la tarde hasta las ocho suele haber otro pico de ventas.

Estos datos concuerdan con los datos reales del comercio. El primer turno del día empieza a las 08:30 y termina a las 14:00. Luego de 14:00 hasta las 17:00 el establecimiento se encuentra cerrado. Finalmente, el segundo turno del día empieza a las 17:00 y termina a las 20:30. El gráfico obtenido concuerda con los turnos de la tienda, por tanto podemos afirmar que no hay ninguna anomalía relacionada con las horas de las ventas.

Ahora veremos la cantidad de artículos vendidos en función del día de la semana. Cabe recalcar que la tienda funciona con dos turnos diferentes: los turnos de mañana y los turnos de tarde. Esto es así para cada día de la semana salvo los sábados porque el establecimiento cierra únicamente por las tardes. De lunes a viernes las ventas suelen mantenerse estables, los sábados los ingresos caen bastante y los domingos son los días con más ingresos de la semana. Esto se debe a que la competencia cierra los domingos.

Figura 4.28: Artículos vendidos por día de la semana

El gráfico de arriba refleja la situación mencionada previamente sobre el movimiento de la tienda. Se puede observar que las ventas suelen ser estables de lunes a viernes, los sábados suponen una caída en la facturación y que los domingos son los días de más rentabilidad de la semana. Por tanto, se puede afirmar que no existen incoherencias a simple vista sobre las fechas de las ventas.

Seguidamente, veremos una gráfica muy similar a la anterior. Esta nueva gráfica refleja los días de la semana según su ingreso bruto en euros. Esta gráfica se hace para comprobar que, los días especiales como el sábado o el domingo tienen un nivel de ingresos similar al nivel de productos vendidos por día de la semana.

Días con más ingresos

```
fig,ax = plt.subplots()
fig.set_size_inches(15,5)

income = salesNoNa.groupby(["diaSemana"])[["precioFinal"]].sum().reset_index()
ordersDay = income[["diaSemana"]].replace(
    {0:"Lunes",1:"Martes",2:"Miércoles",3:"Jueves",4:"Viernes",5:"Sábado",6:"Domingo"})
ordersDay["precioFinal"] = income["precioFinal"]
sn.barplot(color="#8332AC",data=ordersDay,x="diaSemana",y="precioFinal",
           ax=ax,order=["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"])
ax.set(xlabel='Día de la semana', ylabel="Ingresos brutos",title="Ingresos por días de la semana")
```

```
[Text(0, 0.5, 'Ingresos brutos'),
 Text(0.5, 0, 'Día de la semana'),
 Text(0.5, 1.0, 'Ingresos por días de la semana')]
```

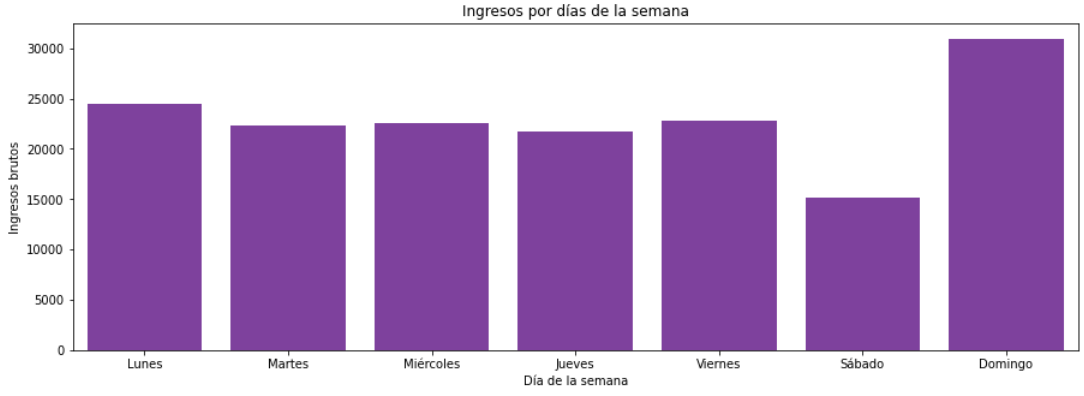


Figura 4.29: Ingresos por días de la semana

Como se puede ver, los ingresos brutos de lunes a viernes suelen ser similares igual que en el caso anterior. Los ingresos de los sábados descienden en comparación a los días anteriores, de la misma forma que ocurría en la gráfica previa. Y por último, los domingos se suele generar muchos más ingresos que ningún otro día. Todos estos datos concuerdan con lo que ya sabíamos acerca de la empresa y su modelo de negocio. Podemos afirmar que no hay ninguna anomalía presente en la facturación de las ventas.

En este caso, indagaremos más en los datos de los productos vendidos. A continuación se verá un gráfico que refleje los treinta productos más vendidos del establecimiento. Dicha información está ordenada de más vendido a menos vendido.

▾ Productos más vendidos

```
productsCount = salesNoNa.groupby(["nombreProd"])[["cantidadVendida"]].sum().reset_index()

pAux = pd.DataFrame(productos_nombre, columns=["nombreProd"])
mergedData = pd.merge(productsCount, pAux, how="left", on="nombreProd").sort_values(
    by="cantidadVendida",ascending=False)

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="nombreProd",y="cantidadVendida",ax=ax,orient="v",color="#5AA9E6")
ax.set(xlabel='Nombre de los productos',ylabel="Cantidad vendida",title="Productos más vendidos")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 <a list of 30 Text major ticklabel objects>)
```

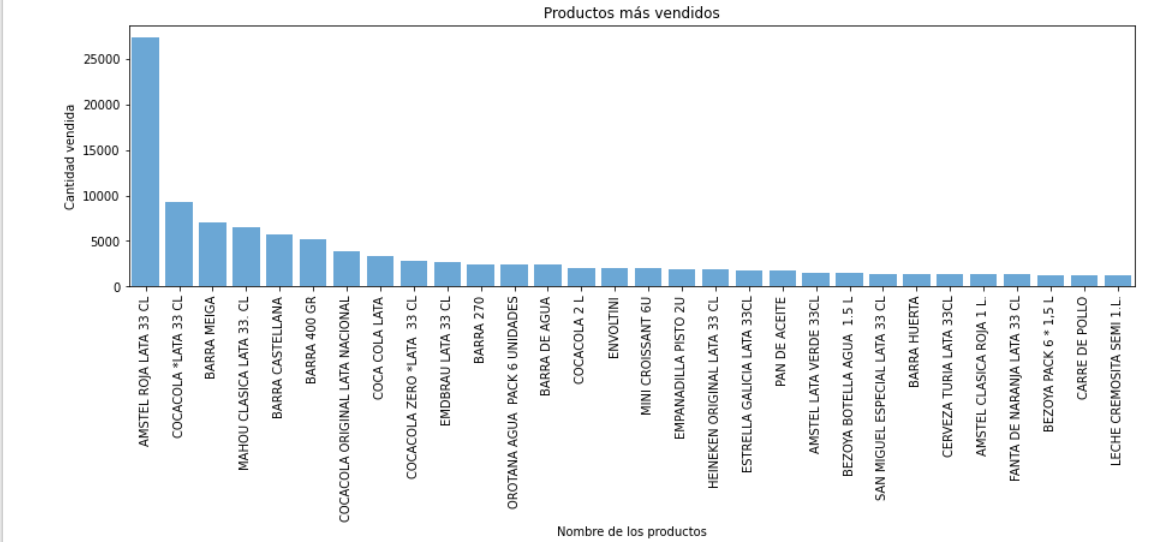


Figura 4.30: Productos más vendidos

Se puede observar que el producto más vendido, con diferencia, es la cerveza Amstel de lata. Esto era esperable ya que los dueños del local tienen dos clientes que compran dicha cerveza para venderlas en sus propios establecimientos. Cuando van a comprar, suelen cargar cinco o seis packs de veinticuatro latas. Como se puede observar, al vender cada semana cantidades superiores a doce packs, la cantidad de latas vendidas ascienden mucho.

Ahora se mostrará una gráfica de productos más frecuentes en las ventas. A diferencia de la gráfica anterior, esta no suma la cantidad vendida de cada producto, simplemente suma el número de ventas que contiene cada producto. Esto nos permite diferenciar qué productos se suelen comprar con frecuencia de los que no. También nos resuelve el problema del caso anterior donde la cerveza Amstel de lata se vende mucho más que los otros productos porque se venden en packs.

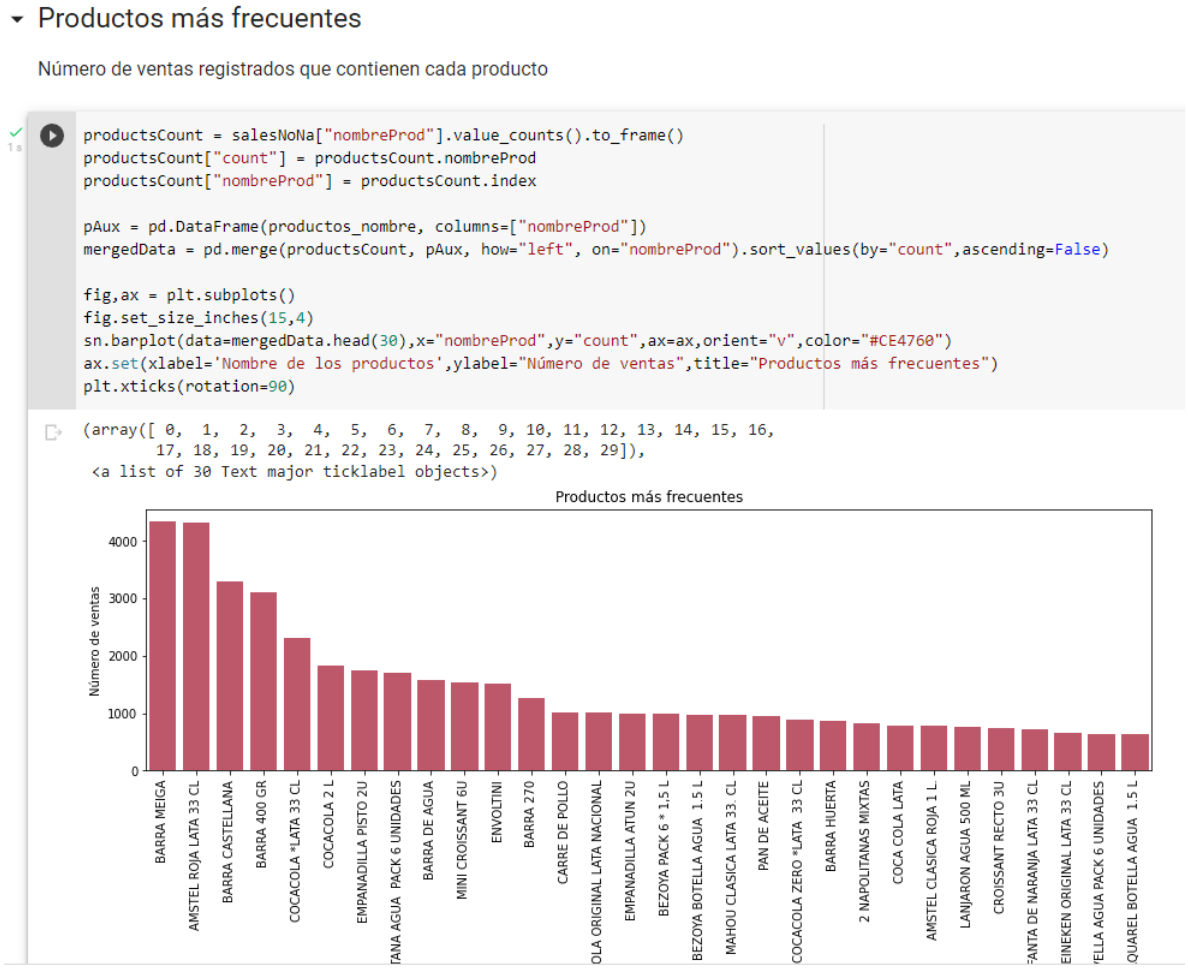


Figura 4.31: Productos más frecuentes

Esta nueva gráfica nos muestra una información muy acertada con la realidad. Al ser una tienda con sección de bollería y panadería, no es de extrañar que productos como las barras de pan o las empanadillas aparezcan con tanta frecuencia en las ventas. También se puede ver que los packs de agua y las cervezas se suelen vender con bastante frecuencia, lo cual encaja con lo vivido por los dueños del establecimiento. Podemos afirmar que, en base a estos datos recogidos y mostrados, la información de las ventas reflejan la realidad del pequeño comercio.

A continuación se mostrarán dos gráficas de frecuenta y cantidad vendida. Los datos en cuestión agrupan categorías de los productos en lugar de nombres. El objetivo de estas dos gráficas es comprobar si tienen coherencia y cuadran con la información vista acerca de los productos.

▾ Categorias más vendidas

```
categoriaCount = salesNoNa.groupby(["categoria"])[["cantidadVendida"]].sum().reset_index()

pAux = pd.DataFrame(productos_nombre, columns=["categoria"])
mergedData = pd.merge(categoriaCount, pAux, how="left", on="categoria").sort_values(
    by="cantidadVendida",ascending=False)

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="categoria",y="cantidadVendida",ax=ax,orient="v",color="#97CC04")
ax.set(xlabel='Categoría de los productos',ylabel="Cantidad vendida",title="Categorías más vendidas")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 <a list of 16 Text major ticklabel objects>)
```

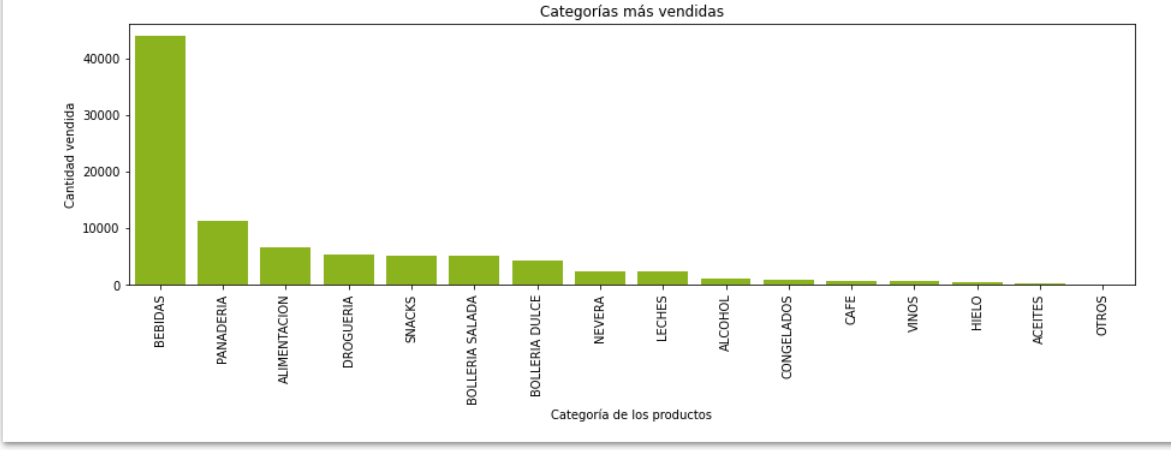


Figura 4.32: Categorías más vendidas

Claramente la gráfica de arriba muestra un alto valor de ventas para los productos de la categoría «bebidas». Esto encaja con la información vista previamente porque algunos de los productos más vendidos y con más frecuencia de compra eran las cervezas, las bebidas azucaradas y las aguas. También se puede observar que la panadería y la alimentación son las siguientes dos categorías más vendidas.

En esta gráfica de frecuencia podemos ver que las diferencias entre categorías se han suavizado. De la misma forma, también podemos observar que el orden de las categorías más frecuentes coincide con el orden de las categorías más vendidas, lo cual no es de extrañar.

Categorías más frecuentes

```
categoriaFreqCount = salesNoNa["categoria"].value_counts().to_frame()
categoriaFreqCount["count"] = categoriaFreqCount.categoria
categoriaFreqCount["categoria"] = categoriaFreqCount.index

pAux = pd.DataFrame(productos_nombre, columns=["categoria"])
mergedData = pd.merge(categoriaFreqCount, pAux, how="left", on="categoria").sort_values(by="count",ascending=False)

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="categoria",y="count",ax=ax,orient="v",color="#0FA3B1")
ax.set(xlabel='Categoría de los productos',ylabel="Número de ventas",title="Categorías más frecuentes")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 <a list of 16 Text major ticklabel objects>)
```

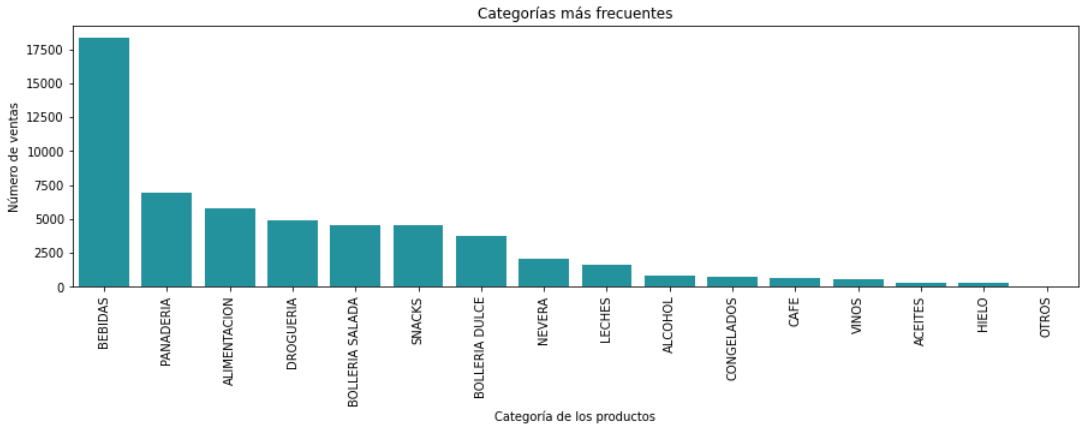


Figura 4.33: Categorías más frecuentes

4.2.3. Fase III. Preparación de los datos

En esta tercera fase de CRISP-DM se preparan los datos con el objetivo de poder aplicar correctamente las distintas técnicas de minería de datos. Para ello es necesario limpiar los datos disponibles y darles una forma fácil de manipular. En total hay aproximadamente 60.000 documentos asociados a las ventas en los registros del negocio. Estos registros recogen valores de ventas desde agosto de 2021 hasta julio de 2022. La colección de datos está almacenada en un fichero JSON. Todas las ventas siguen el esquema mencionado previamente.

En este proyecto se trabajó en Python haciendo uso de diversas librerías de ciencias de datos que están disponibles. Debido a esta elección de lenguaje de programación, fue necesario transformar el fichero exportado en MongoDB a un documento más fácil de manipular en Python. Con tal de cumplir ese requisito, se escribió un pequeño programa en JavaScript para leer y transformar ficheros de datos a un archivo CSV con filas y columnas.

No se añadieron al fichero CSV las filas que incluyen productos irrelevantes para las reglas de asociación. Dichas filas incluían productos que los clientes no eligieron comprar intencionadamente ya que tales productos son considerados servicios del establecimiento. Un ejemplo de servicio del establecimiento son las bolsas de plástico que compran los clientes para cargar la compra. Fue necesario no incluir esos servicios porque podrían sesgar los resultados del modelo. Los servicios en cuestión son todos los tipos de bolsas (de plástico y cartón) y los servicios de bebida fría (una bebida es veinte céntimos más cara si se desea comprarla fría). El número de filas resultante es de 135.518 con 10 columnas.

La colección de productos no fue utilizada para el desarrollo de este proyecto, pero podría tener utilidad en futuros modelos que pueda necesitar la empresa. Esta decisión se debe al hecho de que se están buscando patrones de productos frecuentes en las ventas pasadas. La colección de productos es irrelevante ya que la propia colección de ventas incluye información acerca de los productos vendidos. Esto reduce la cantidad de datos con los que se trabaja a un solo fichero. El fichero en cuestión se llamará «ventasExtended.csv» (porque el fichero cuenta con una lista de productos vendidos) y ocupa 14MB.

Una vez los datos hayan sido tratados y transformados, este es el formato final de la colección de ventas:

```
display(salesCsv.head(10))
```

	ventaID	nombreProd	ean	cantidadVendida
0	62fa35f0b82ab4eabc25f35e	PAPEL HIGIENICO COLHOGAR ROSA PROTECT	7322540670608	1
1	62fa351fb82ab4eabc25f349	OR XATA PREMIUM 1L	8410331300069	1
2	62fa351fb82ab4eabc25f349	HORCHATA PREMIUM DULCE SOL	8410087976587	1
3	62fa351fb82ab4eabc25f349	DULCESOL VALENCIANAS 150 GR	8410087013404	1
4	62fa351fb82ab4eabc25f349	DULCESOL PALMERITAS 16 UDS.	8410087041230	1
5	62fa351fb82ab4eabc25f349	FARTONS POLO	8410331000020	1
6	62fa3464b82ab4eabc25f33e	HORCHATA PREMIUM DULCE SOL	8410087976587	2
7	62fa3420b82ab4eabc25f331	AMSTEL ROJA LATA 33 CL	8410569005651	3
8	62fa3420b82ab4eabc25f331	MAHOU SIN LATA 33 CL	8411327722018	2
9	62fa3415b82ab4eabc25f326	HAPPYPLUS TOALLITAS BABY 72 S	8410800053908	1

Figura 4.34: Esquema de las ventas en CSV - Parte izquierda

createdAt	categoria	iva	margen	precioCompra	precioFinal
1660564976398	DROGUERIA	21.0	41.676505	0.980	1.68
1660564767056	BEBIDAS	10.0	28.131711	1.270	1.79
1660564767056	BEBIDAS	10.0	44.010000	1.130	1.79
1660564767056	ALIMENTACION	10.0	37.320000	0.662	1.00
1660564767056	ALIMENTACION	10.0	37.741047	0.660	1.00
1660564767056	CONGELADOS	10.0	11.111111	0.900	1.10
1660564580644	BEBIDAS	10.0	44.010000	1.130	1.79
1660564512030	BEBIDAS	21.0	13.990000	0.290	0.40
1660564512030	BEBIDAS	21.0	47.210000	0.320	0.57
1660564501671	DROGUERIA	21.0	30.000000	0.890	1.40

Figura 4.35: Esquema de las ventas en CSV - Parte derecha

Como se puede observar, cada fila del conjunto de datos representa un producto vendido. Las columnas del conjunto de datos son las siguientes: «ventaID» (identificador de la venta), «nombreProd» (el nombre del producto vendido), «ean» (código de barras

del producto), «cantidadVendida» (representa cuantas unidades del producto fueron vendidas), «createdAt» (fecha de la transacción), «categoria» (representa la familia del producto), «iva», «margen», «precioCompra» y «precioFinal» (el precio que ha pagado el cliente por el producto).

Si hacemos una comprobación de la validez de los datos, podemos observar que el número de filas con al menos un NA es de 1387. Para este tipo de modelo, no todos los datos son necesarios. Toda la información acerca de los precios, márgenes e IVA se pueden descartar. Podemos comprobar más abajo que el número de filas donde la categoría contiene un NA es de 1120.

```
import pandas as pd
import numpy as np
from IPython.display import display
import sys
from itertools import combinations, groupby
from collections import Counter

# Importamos los datos de las ventas
salesCsv = pd.read_csv("ventasExtended.csv")
salesNoNa = salesCsv.dropna(subset=['categoria'])

print("Tamaño del Dataset inicial:", len(salesCsv.index))

rows_nan = salesCsv.isna().any(axis=1).sum()
print("Filas con al menos un NA:", rows_nan)

print("Tamaño final sin las filas con NA en categoria:", len(salesNoNa.index))
print("Número de filas eliminadas: ", (len(salesCsv.index) - len(salesNoNa.index)))

display(salesCsv.head(10))
```

```
Tamaño del Dataset inicial: 135518
Filas con al menos un NA: 1387
Tamaño final sin las filas con NA en categoria: 134398
Número de filas eliminadas:  1120
```

Figura 4.36: Dimensiones del conjunto de datos

4.2.4. Fase IV. Modelado - Parte uno. Patrones entre productos

En esta cuarta fase de CRISP-DM se hablará de las dos técnica de modelado posibles para resolver este problema de negocio, el significado de algunas reglas de asociación y el desarrollo del propio modelo.

4.2.4.1. Reglas de asociación

Para el trabajo asociado a las reglas de asociación es necesario explicar brevemente el significado de tres reglas. Dichas reglas son el **«support»**, **«confidence»** y **«lift»**

- **Support:** el support de un producto consiste en el porcentaje de ventas que contienen dicho producto. Si un producto aparece en la mitad de las ventas, ese producto tendrá un support de 0.5 que equivalen a un 50 %

  La fórmula para calcular el support de un producto, siendo N el número total de transacciones, es la siguiente.

$$Support(A) = \frac{Frecuencia(A)}{N} \tag{4.1}$$

  Y para un conjunto de productos *A* y *B*, la fórmula es la siguiente.

$$Support(A \longrightarrow B) = \frac{Frecuencia(A \cap B)}{N} \tag{4.2}$$

- **Confidence:** valor numérico que relacionan dos artículos. Dados dos productos, *A* y *B*, esta regla de asociación mide el porcentaje de confianza en que un producto *B* se compre una vez se haya comprado un producto *A*. Su fórmula es la siguiente.

$$Confidence(A \longrightarrow B) = \frac{Support(A \longrightarrow B)}{Support(A)} \tag{4.3}$$

- **Lift:** Dado dos productos, *A* y *B*, el lift indica si hay o no alguna relación entre ambos productos. Cabe recalcar que el lift de dos productos es independiente del orden de compra, eso quiere decir que el lift de *A* y *B* es igual al lift de *B* y *A*. Esta es la fórmula que lo define.

$$Lift(A \longrightarrow B) = \frac{Support(A \longrightarrow B)}{Support(A) \times Support(B)} \tag{4.4}$$

  A diferencia de las reglas vistas anteriormente, el lift no se trata de un porcentaje. Si el valor del lift es igual a 1, significa que no hay relación entre ambos productos. Si el valor del lift es mayor que 1, quiere decir que el lift es positivo y los productos se compran juntos con frecuencia. En cambio, si el lift es menor que 1, eso implica que el lift es negativo y que las compras conjuntas de ambos productos son resultado de la casualidad.

4.2.4.2. Algoritmos disponibles

El problema de negocio requiere de la creación de un sistema de recomendación de productos para la página web ERPWeb que pueda aumentar el número de ventas del local. Existen diversas formas de crear un sistema de recomendación. Desde agrupar y analizar los gustos de los clientes del establecimiento hasta buscar patrones de compras y asociaciones entre artículos. Como ya se ha mencionado previamente en este documento, se ha elegido, en base a la cantidad y calidad de los datos disponibles, realizar un análisis y modelado mediante reglas de asociación.

Existen dos algoritmos muy conocidos y utilizados para esta problemática:

1. **Algoritmo Apriori:** este es un algoritmo muy usado para el minado de patrones frecuentes. Apriori se centra en seleccionar artículos y generar conjuntos y filtrar aquellos que sean más frecuentes en función del *support*, *confidence* o *lift*.
2. **Algoritmo FP Growth:** este algoritmo es popularmente conocido como un reemplazo de del algoritmo Apriori. Tanto es así que, en un proyecto de minería de datos que use Apriori, la función encargada de ejecutar dicho algoritmo se puede sustituir totalmente por una nueva función que implemente FP Growth. La ventaja principal de este nuevo algoritmo es lo bien que escala con conjuntos de datos grandes. El algoritmo Apriori necesita más tiempo de ejecución que FP Growth para generar un conjunto de artículos frecuentes.

4.2.4.3. Construir el Modelo

La implementación del modelo se hará en Python, mediante una libreta de *Google Colaboratory* [2].

Se implementó una función llamada *CalcularFP* con dos parámetros de entrada: el conjunto de datos y el soporte mínimo. Dicha función se ha implementado haciendo uso de una librería de Python llamada «mlxtend».

▾ FP Growth

```
[20] import mlxtend
     from mlxtend.preprocessing import TransactionEncoder
     from mlxtend.frequent_patterns import association_rules, apriori as ap, fpgrowth

     def CalcularFP(dataset, min_supp):
       con = TransactionEncoder()
       con_arr = con.fit(dataset).transform(dataset)
       df = pd.DataFrame(con_arr, columns = con.columns_)

       # Ejecución del algoritmo
       resFP=fpgrowth(df, min_support=min_supp, use_colnames=True)
       association_resFP = association_rules(resFP, metric="lift", min_threshold=1.2)

       association_resFP["antecedents_len"] = association_resFP["antecedents"].apply(
           lambda x: len(x))

       # display(association_resFP[ (association_resFP['antecedents_len'] <= 1) &
       #        (association_resFP['confidence'] > 0.3) &
       #        (association_resFP['lift'] > 1.2) ])

       display(association_resFP)
       return association_resFP
```

[2] Google Colaboratory, también conocido como Colab, es una herramienta en línea que permite trabajar con un intérprete de Python de forma gratuita.

Figura 4.37: Función CalcularFP

4.2.4.4. Ejecución del modelo

Para ejecutar el modelo es necesario alimentarlo con los datos necesarios y un valor de soporte adecuado. Hay que tener en cuenta que un valor muy elevado de soporte (por ejemplo un 99 %) podría devolver un resultado vacío, mientras que un valor muy bajo (0.01 %) podría devolver demasiados valores de poca utilidad.

El modelo se ejecuta junto al conjunto de datos mencionado en las fases anteriores. El algoritmo en cuestión consume una array de arrays con los códigos EAN de los productos vendidos.

Por ejemplo: [["5449000164957", "5449000131836"], ["5449000131836"]] es un array de arrays. Cada subarray es una venta del sistema. El primer subarray es una venta con dos elementos, el segundo subarray es una venta con solo producto.

Otro valor que necesita el algoritmo es el valor del soporte mínimo. Esto ayuda al algoritmo a mejorarlos tiempos de ejecución y filtrar los resultados no deseados. Cualquier conjunto de datos que no cumpla con el valor del soporte mínimo es descartado.

Haciendo una primera ejecución sobre los datos de las ventas y un soporte del 30 % el algoritmo nos devuelve una lista vacía. Este es un indicador de que no existen conjuntos que aparezcan en un 30 % de las ventas. A continuación se prueba a ejecutar el mismo algoritmo con un soporte de 5 %. El resultado de dicha ejecución también es una lista vacía. Esto significa que el algoritmo no ha encontrado ningún par de artículos que se compren juntos con una frecuencia igual o superior al 5 %. Si analizamos el conjunto de datos, podemos ver que hay en total 2018 productos distintos. Son demasiados productos para tan pocas ventas, por tanto es difícil encontrar un conjunto de datos con un alto soporte. No solo eso, otro problema que podemos encontrar al mirar el conjunto de datos es que existen muchos artículos diferentes para un mismo tipo de producto. En un supuesto de que el pan de molde se compre mucho con la mantequilla, disponer de diez tipos diferentes de pan de molde y diez tipos diferentes de mantequillas hacen muy difícil que el algoritmo pueda encontrar algún par de productos con un soporte elevado. Aunque las diez referencias de pan de molde son esencialmente el mismo tipo de producto, disponer de diez códigos de barras diferentes hacen que el algoritmo no pueda encontrar relación en dichos productos. Para encontrar algún conjunto de datos, necesitamos bajar mucho el valor del soporte. Al ejecutar una vez más el algoritmo con un soporte del 0.2 %, podemos encontrar un conjunto con nueve pares de artículos relacionados.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(MINI CROISSANT)	(CASTELLANA)	0.025884	0.055753	0.002103	0.081258	1.457456
1	(CASTELLANA)	(MINI CROISSANT)	0.055753	0.025884	0.002103	0.037724	1.457456
2	(MINI CROISSANT)	(MEIGA)	0.025884	0.073580	0.002324	0.089777	1.220125
3	(MEIGA)	(MINI CROISSANT)	0.073580	0.025884	0.002324	0.031581	1.220125
4	(5740700988349)	(5449000011527)	0.039334	0.012128	0.002476	0.062958	5.191275
5	(5449000011527)	(5740700988349)	0.012128	0.039334	0.002476	0.204196	5.191275

Figura 4.38: Cinco primeros resultados obtenidos con un soporte del 0.2 %

Como se puede observar en la imagen anterior, los valores de soporte y la confianza son bastante bajos. Si un par de artículos tienen un soporte muy bajo, significa que no disponemos de información suficiente acerca de la relación entre los productos. Por tanto, no se pueden sacar conclusiones satisfactorias sobre esa regla de asociación. Al ver que todos los pares obtenidos tienen valores tan bajos, ninguna conclusión se puede sacar acerca de esta ejecución.

▾ Patrones frecuentes en los productos

```
[19] association_results_products = CalcularFP(dataset=datasetEan, min_supp=0.002)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents_len
0	(MINI CROISSANT)	(CASTELLANA)	0.025884	0.055753	0.002103	0.081258	1.457456	0.000660	1.027761	1
1	(CASTELLANA)	(MINI CROISSANT)	0.055753	0.025884	0.002103	0.037724	1.457456	0.000660	1.012305	1
2	(MINI CROISSANT)	(MEIGA)	0.025884	0.073580	0.002324	0.089777	1.220125	0.000419	1.017794	1
3	(MEIGA)	(MINI CROISSANT)	0.073580	0.025884	0.002324	0.031581	1.220125	0.000419	1.005883	1
4	(5740700988349)	(5449000011527)	0.039334	0.012128	0.002476	0.062958	5.191275	0.001999	1.054246	1
5	(5449000011527)	(5740700988349)	0.012128	0.039334	0.002476	0.204196	5.191275	0.001999	1.207163	1
6	(5449000009067)	(BARRA 400 GR)	0.031108	0.052683	0.002409	0.077426	1.469656	0.000770	1.026820	1
7	(BARRA 400 GR)	(5449000009067)	0.052683	0.031108	0.002409	0.045718	1.469656	0.000770	1.015310	1
8	(5449000009067)	(MEIGA)	0.031108	0.073580	0.002765	0.088877	1.207888	0.000476	1.016789	1
9	(MEIGA)	(5449000009067)	0.073580	0.031108	0.002765	0.037575	1.207888	0.000476	1.006719	1

Figura 4.39: Ejecución del algoritmo con el conjunto de datos de los productos

Si se quiere crear un sistema de recomendación mediante reglas de asociación, es necesario entender qué problemas han llevado a unos resultados de baja calidad y generar un nuevo modelo que proporcione mejores resultados.

4.2.5. Fase IV. Modelado - Parte dos. Patrones entre categorías

Para resolver este problema y obtener mejores resultados, es necesario plantear el modelado de otra forma. Dado que el conjunto de datos tiene muchos productos similares pero de otras marcas y formatos, es difícil trazar una relación y encontrar patrones entre los distintos artículos. Para resolver este problema, se hará un nuevo modelado usando las categorías de los productos en lugar de su código EAN. La idea detrás de esta decisión está en buscar patrones entre los tipos de productos en lugar de patrones entre los propios productos.

4.2.5.1. Ejecución del modelo

Para esta nueva ejecución solo es necesario cambiar uno de los argumentos pasados a la función «CalcularFP». En lugar de usar los códigos EAN de los productos, se usan las categorías de dichos productos. Un array de ejemplo sería: [['BOLLERIA SALADA', 'ALIMENTACION'], ['DROGUERIA']].

Existen 2018 códigos EAN en todo el conjunto de datos, mientras que el número de categorías solo es de 16. Esta reducción ayudará a obtener mejores valores de soporte, lo cual puede conllevar la implementación de un sistema de recomendación de mejor calidad.

▾ Patrones frecuentes en las categorias

```
[28] association_results_categorias = CalcularFP(dataset=dfCategorias, min_supp=0.02)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(BOLLERIA SALADA)	(BOLLERIA DULCE)	0.160560	0.147211	0.028869	0.179801	1.221382
1	(BOLLERIA DULCE)	(BOLLERIA SALADA)	0.147211	0.160560	0.028869	0.196106	1.221382
2	(DROGUERIA)	(ALIMENTACION)	0.119004	0.171586	0.031532	0.264966	1.544219
3	(ALIMENTACION)	(DROGUERIA)	0.171586	0.119004	0.031532	0.183768	1.544219
4	(BEBIDAS, PANADERIA)	(ALIMENTACION)	0.093476	0.171586	0.025443	0.272183	1.586281
5	(BEBIDAS, ALIMENTACION)	(PANADERIA)	0.078347	0.262433	0.025443	0.324746	1.237442
6	(PANADERIA)	(BEBIDAS, ALIMENTACION)	0.262433	0.078347	0.025443	0.096949	1.237442
7	(ALIMENTACION)	(BEBIDAS, PANADERIA)	0.171586	0.093476	0.025443	0.148280	1.586281
8	(ALIMENTACION)	(SNACKS)	0.171586	0.141360	0.033618	0.195927	1.386020
9	(SNACKS)	(ALIMENTACION)	0.141360	0.171586	0.033618	0.237821	1.386020
10	(BOLLERIA DULCE)	(SNACKS)	0.147211	0.141360	0.025663	0.174329	1.233229
11	(SNACKS)	(BOLLERIA DULCE)	0.141360	0.147211	0.025663	0.181545	1.233229
12	(ALIMENTACION)	(LECHES)	0.171586	0.070951	0.023560	0.137307	1.935234
13	(LECHES)	(ALIMENTACION)	0.070951	0.171586	0.023560	0.332058	1.935234
14	(NEVERA)	(ALIMENTACION)	0.062386	0.171586	0.026206	0.420065	2.448138
15	(ALIMENTACION)	(NEVERA)	0.171586	0.062386	0.026206	0.152728	2.448138
16	(NEVERA)	(PANADERIA)	0.062386	0.262433	0.024917	0.399402	1.521919
17	(PANADERIA)	(NEVERA)	0.262433	0.062386	0.024917	0.094946	1.521919

Figura 4.40: Ejecución del algoritmo con el conjunto de categorías

Tras la ejecución, se puede ver que los valores de soporte para antecedentes y consecuentes son mayores. También es mayor el valor de soporte para la pareja (o trío) de

productos. Los valores de confianza también son más elevados, llegando a indicar un 39.94 % para el par NEVERA ->PANADERIA. Este es un buen indicativo de que este segundo modelo es mejor que el primero.

4.2.6. Fase V. Evaluación

En esta quinta fase del proyecto se evalúan los modelos creados y los resultados obtenidos. Una vez terminada la evaluación, se decide si el modelo cumplen los objetivos del negocio y se puede proseguir con la fase de despliegue.

Se crearon dos modelos para el sistema de recomendación. El primero es un modelo de reglas de asociación que busca patrones frecuentes en los productos vendidos (se usaban los códigos EAN para distinguir los distintos productos), mientras que el segundo modelo de reglas de asociación consiste en buscar patrones frecuentes entre los distintos tipos de categorías de los productos.

Del primer modelo no podemos sacar muchas conclusiones ya que el soporte del par de productos es muy bajo y apenas aporta información. Un ejemplo es el par *mini croissant* y *barra de pan castellana* que tienen una frecuencia de compra del 0.21 %, es decir, que de 58.000 ventas, solo aproximadamente 122 tienen ese par de productos. Al ser tan poco frecuente, no podemos sacar ninguna conclusión significativa.

Del segundo modelo se obtienen muchos mejores resultados acerca de la frecuencia de los productos y sus pares de compra. Podemos observar que los productos del tipo *nevera* están fuertemente relacionados con los productos del tipo *alimentación* con una frecuencia de compra del 2.6 %. Parece un valor bajo pero es casi 13 veces más alto que los valores del modelo anterior, lo cual es una gran mejora.

Si se quisiera obtener un modelo de mejor calidad se necesitarían dos elementos de los que actualmente no se dispone: un volumen de datos más elevado que incluyan ventas de al menos un par de años (sería interesante segregar los distintos patrones de compra por estaciones), y datos de mejor calidad. Este último elemento se debe a la ambigüedad de algunas categorías. Por ejemplo: la categoría *bebidas* incluyen artículos muy dispares. Desde cervezas y zumos hasta bebidas energéticas y diferentes marcas de agua. También es necesario recalcar que esta es la categoría más vendida de la tienda, la gran descompensación de ventas puede generar un sesgo en los datos.

4.2.7. Fase VI. Despliegue

En esta última fase del proceso de minería de datos se hablará del despliegue del modelo en un entorno de producción. Se repasará la mejor forma de desplegarlo y cómo encaja con toda la arquitectura del sistema.

El despliegue de este modelo se hará aprovechando el amplio ecosistema alrededor de Python y su comunidad. Ya que el proyecto de ERPSolution está diseñado para facilitar la incorporación de nuevos componentes y servicios, podemos desplegar este modelo como una pequeña API REST. La idea es crear una pequeña API que pueda responder a peticiones por parte de ERPGateway. Una vez desarrollada la API, se creará una imagen de Docker que eventualmente se inicializará en un contenedor. Dicho contenedor formará parte de la totalidad de la arquitectura como un servicio más.

Esta aplicación se creó usando una librería de Python llamada «Flask». Este nuevo servicio se llama «ERPRecommendation» y acepta una única petición POST por el endpoint raíz.

▾ Recomendador

```
def Recomendar(association_result: pd.DataFrame, categorias_carrito, confidence_min: float, lift_min: float):
  recomendador: pd.DataFrame = association_result[ (association_result['antecedents_len'] <= 1) &
                                    (association_result['confidence'] >= confidence_min) &
                                    (association_result['lift'] >= lift_min) ]

  indice = -1
  indiceRecomendador = -1
  reco = {}

  for index, recomendacion in recomendador.iterrows():
    indiceRecomendador += 1
    confianza = -1

    for indexCategorias, categoria in enumerate(categorias_carrito):
      if categoria in recomendacion["antecedents"]:
        if confianza < recomendacion["confidence"]:
          confianza = recomendacion["confidence"]
          indice = indiceRecomendador

  if indice >= 0:
    recomendacionFinal = recomendador.iloc[[indice]]
    res = list(list(recomendacionFinal["consequents"])[0])[0]
    return res

  return None
```

Figura 4.41: Función encargada de recomendar una categoría

El servicio de recomendación necesita el contenido de la cesta de la compra actual del cliente que esté pasando por caja. Dicho contenido no es más que un array de las categorías a las que pertenecen los productos de su cesta. Una vez recibida la llamada, el servidor redirige el cuerpo de la petición a la función «Recomendar» que se encarga de buscar la categoría a recomendar que más confianza tenga. Una vez encontrada, se devuelve a la aplicación cliente un JSON cuyo cuerpo incluye un campo llamado «recomendación» y el nombre de la categoría a recomendar. En cuanto a ERPWeb: una

vez recibida la respuesta de la petición de recomendación, la aplicación web ofrece de una oferta al cliente. Esta oferta le da a elegir cualquier producto de la categoría recomendada con un descuento máximo de un 5 %. La idea es incentivar al cliente a comprar un artículo más que le pueda interesar y así aumentar la facturación del establecimiento.

Por tanto, este es el diagrama de la arquitectura final de la solución una vez añadido este nuevo servicio de recomendación.

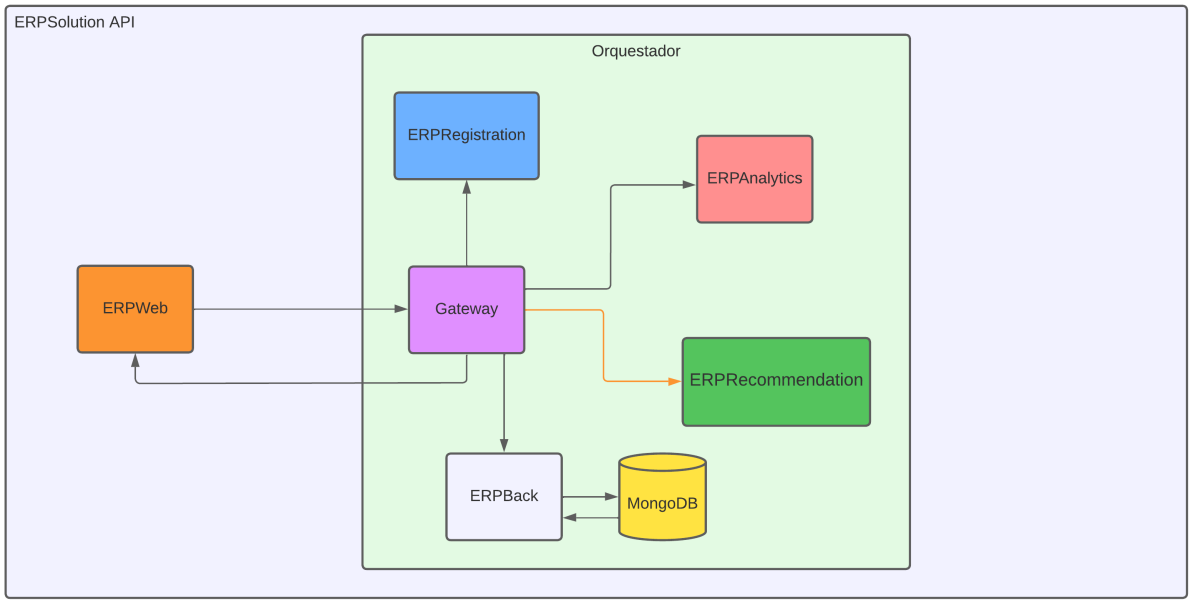


Figura 4.42: Diseño final del API de ERPSolution

Capítulo 5

Resultados y conclusiones

Ya que este proyecto se divide en dos partes, se podría decir que la primera parte está relacionada con el ámbito ingenieril y el desarrollo de software mientras que la segunda parte está relacionado con el ámbito científico y estadístico.

De la primera parte se puede afirmar que el proyecto ha sido un éxito ya que se ha cumplido con la mayor parte de los requisitos. Sin duda lo aspectos más importante para el nuevo sistema software del establecimiento eran el bajo coste mensual, el hecho de que todos los componentes de la página web estuvieran en español, la simplificación de procesos cotidianos del local y, por último, una mejoría en la forma de acceder y entender los datos internos de la empresa. Si bien aún quedan muchas herramientas por implementar, el funcionamiento básico y esencial está implementado y probado. De los futuros trabajos para este proyecto, es necesario incluir nuevos componentes típicos de un sistema ERP. Lo más urgente en este sentido serían las herramientas de gestión de inventario, procesamiento de albaranes y gestión de pérdidas, robos y mermas.

Otro motivo para afirmar el éxito de este proyecto de software es el interés que ha despertado en otros antiguos franquiciados. Muchos de ellos se vieron en esta misma situación (la pérdida de su herramienta de negocio) y, para sustituir su antiguo sistema informático, optaron por usar aplicaciones baratas y de baja calidad. Realizar un despliegue de ERPSolution para estos establecimientos, junto a la migración de sus bases de datos, puede resultar en una buena forma de rentabilizar el trabajo de todo un año.

De la segunda parte de este trabajo de fin de máster, la parte relacionada con la ciencia de datos, se puede afirmar que se ha cumplido el objetivo principal de desarrollar y desplegar un modelo de minería de datos que sirva para recomendar artículos a los clientes y fomentar la venta cruzada. Si bien los valores de soporte y confianza de ambos modelos se pueden mejorar con un mejor conjunto de datos, es necesario recalcar que, según pase el tiempo y se obtengan más datos y de mejor calidad, la precisión del modelo también aumentará. Los resultados obtenidos de este modelo también pueden ser usados fuera del ámbito del software. En un futuro se podrá analizar mejor el resultado del modelo y, en base a esa información, se podrá cambiar la distribución de las estanterías para ubicar mejor los artículos del establecimiento. Este tipo de información es vital para una empresa, además de aumentar todavía más el número de ventas cruzadas.

Bibliografía

[1] CHAPMAN, Pete, et al. The CRISP-DM user guide. En 4th CRISP-DM SIG Workshop in Brussels in March. sn, 1999.

[2] DRAGONI, Nicola, et al. Microservices: yesterday, today, and tomorrow. Present and ulterior software engineering, 2017, p. 195-216.

[3] NEWMAN, Sam. Building microservices. ."O'Reilly Media, Inc.", 2021.

[4] HARTIG, Olaf; PÉREZ, Jorge. Semantics and complexity of GraphQL. En Proceedings of the 2018 World Wide Web Conference. 2018. p. 1155-1164.

[5] MASSE, Mark. REST API design rulebook: designing consistent RESTful web service interfaces. ."O'Reilly Media, Inc.", 2011.

[6] DOGLIO, Fernando. Pro REST API Development with Node. js. Apress, 2015.

[7] OSMAN, Abdullahi Sidow. Data mining techniques. 2019.

[8] ROMERO, Cristobal; VENTURA, Sebastian. Data mining in education. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2013, vol. 3, no 1, p. 12-27.

[9] AZEVEDO, Ana; SANTOS, Manuel Filipe. KDD, SEMMA and CRISP-DM: a parallel overview. IADS-DM, 2008.

[10] ESPINOSA-ZÚÑIGA, Javier Jesús. Aplicación de metodología CRISP-DM para segmentación geográfica de una base de datos pública. Ingeniería, investigación y tecnología, 2020, vol. 21, no 1.

[11] PORTUGAL, Ivens; ALENCAR, Paulo; COWAN, Donald. The use of machine learning algorithms in recommender systems: A systematic review. Expert Systems with Applications, 2018, vol. 97, p. 205-227.

[12] ANANDARAJ, A., et al. Book Recommendation System with TensorFlow. En 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, 2021. p. 1665-1669.

[13] KOTSIANTIS, Sotiris; KANELLOPOULOS, Dimitris. Association rules mining: A recent overview. GESTS International Transactions on Computer Science and Engineering, 2006, vol. 32, no 1, p. 71-82.

[14] KODINARIYA, Trupti M.; MAKWANA, Prashant R. Review on determining number of Cluster in K-Means Clustering. International Journal, 2013, vol. 1, no 6, p. 90-95.

[15] DA'U, Aminu; SALIM, Naomie. Recommendation system based on deep learning methods: a systematic review and new directions. Artificial Intelligence Review, 2020, vol. 53, no 4, p. 2709-2748.

[16] BORGELT, Christian; KRUSE, Rudolf. Induction of association rules: Apriori implementation. En Compstat. Physica, Heidelberg, 2002. p. 395-400.

[17] BORGELT, Christian. An Implementation of the FP-growth Algorithm. En Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations. 2005. p. 1-5.

[18] KAMAL, Parves, et al. Security of password hashing in cloud. Journal of Information Security, 2019, vol. 10, no 02, p. 45.

[19] CHEN, Yen-Liang, et al. Market basket analysis in a multiple store environment. Decision support systems, 2005, vol. 40, no 2, p. 339-354.

Anexo I - Notebook de Google Colaboratory

El anexo de abajo es el fichero notebook resultante de este trabajo de minería de datos.

TFM Gustavo Lee - Association Rules Mining

Importación de datos

```
!pip install mlxtend --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.7/dist-packages (0.14.0)
Collecting mlxtend
  Downloading mlxtend-0.20.0-py2.py3-none-any.whl (1.3 MB)
ent already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.7.3)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.3.5
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.21.6
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from mlxtend) (57.4.0)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.1.0
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (3.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotli
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.2->m
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1-
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-le
Installing collected packages: mlxtend
  Attempting uninstall: mlxtend
    Found existing installation: mlxtend 0.14.0
    Uninstalling mlxtend-0.14.0:
      Successfully uninstalled mlxtend-0.14.0
Successfully installed mlxtend-0.20.0
```

```
import pandas as pd
import numpy as np
from IPython.display import display
import sys
from itertools import combinations, groupby
from collections import Counter
import pylab
import calendar
import seaborn as sn
from scipy import stats
import missingno as msno
from datetime import datetime
import matplotlib.pyplot as plt
import warnings
pd.options.mode.chained_assignment = None
warnings.filterwarnings("ignore", category=DeprecationWarning)
%matplotlib inline
sn.set_palette(palette="OrRd")

# Importamos los datos de las ventas
salesCsv = pd.read_csv("ventasDataset.csv")

timeStamp_column = pd.to_datetime(salesCsv['createdAt'], unit='ms', utc=True).map(lambda x: x.tz_convert('Afr
salesCsv['createdAt'] = timeStamp_column
salesCsv['hora'] = timeStamp_column.dt.hour
salesCsv['diaSemana'] = timeStamp_column.dt.day_of_week

salesNoNa = salesCsv.dropna(subset=['categoria'])

print("Tamaño del Dataset inicial:", len(salesCsv.index))

rows_nan = salesCsv.isna().any(axis=1).sum()
```

```
print("Filas con al menos un NA:", rows_nan)

print("Tamaño final sin las filas con NA en categoria:", len(salesNoNa.index))
print("Número de filas eliminadas: ", (len(salesCsv.index) - len(salesNoNa.index)))

display(salesCsv.head(10))
```

```
Tamaño del Dataset inicial: 135518
Filas con al menos un NA: 1387
Tamaño final sin las filas con NA en categoria: 134398
Número de filas eliminadas:  1120
```

	ventaID	nombreProd
0	62fa35f0b82ab4eabc25f35e	PAPEL HIGIENICO COLHOGAR ROSA PROTECT
1	62fa351fb82ab4eabc25f349	OR XATA PREMIUM 1L
2	62fa351fb82ab4eabc25f349	HORCHATA PREMIUM DULCE SOL
3	62fa351fb82ab4eabc25f349	DULCESOL VALENCIANAS 150 GR
4	62fa351fb82ab4eabc25f349	DULCESOL PALMERITAS 16 UDS.
5	62fa351fb82ab4eabc25f349	FARTONS POLO
6	62fa3464b82ab4eabc25f33e	HORCHATA PREMIUM DULCE SOL
7	62fa3420b82ab4eabc25f331	AMSTEL ROJA LATA 33 CL
8	62fa3420b82ab4eabc25f331	MAHOU SIN LATA 33 CL
9	62fa3415b82ab4eabc25f326	HAPPYPLUS TOALLITAS BABY 72 S

	ean	cantidadVendida	createdAt
0	7322540670608	1	2022-08-15 14:02:56.398000+02:00
1	8410331300069	1	2022-08-15 13:59:27.056000+02:00
2	8410087976587	1	2022-08-15 13:59:27.056000+02:00
3	8410087013404	1	2022-08-15 13:59:27.056000+02:00
4	8410087041230	1	2022-08-15 13:59:27.056000+02:00
5	8410331000020	1	2022-08-15 13:59:27.056000+02:00
6	8410087976587	2	2022-08-15 13:56:20.644000+02:00
7	8410569005651	3	2022-08-15 13:55:12.030000+02:00
8	8411327722018	2	2022-08-15 13:55:12.030000+02:00
9	8410800053908	1	2022-08-15 13:55:01.671000+02:00

	categoria	iva	margen	precioCompra	precioFinal	hora	diaSemana
0	DROGUERIA	21.0	41.676505	0.980	1.68	14	0
1	BEBIDAS	10.0	28.131711	1.270	1.79	13	0
2	BEBIDAS	10.0	44.010000	1.130	1.79	13	0
3	ALIMENTACION	10.0	37.320000	0.662	1.00	13	0
4	ALIMENTACION	10.0	37.741047	0.660	1.00	13	0
5	CONGELADOS	10.0	11.111111	0.900	1.10	13	0
6	BEBIDAS	10.0	44.010000	1.130	1.79	13	0
7	BEBIDAS	21.0	13.990000	0.290	0.40	13	0
8	BEBIDAS	21.0	47.210000	0.320	0.57	13	0
9	DROGUERIA	21.0	30.000000	0.890	1.40	13	0

Limpieza de datos

```
ventas = salesNoNa.groupby(["ventaID"])[["nombreProd","ean", "cantidadVendida", "categoria"]].agg(lambda x: l

# Obtenemos una lista de todos los productos por EAN
productos_dict = salesNoNa[['ean']].drop_duplicates()
productos_dict = {name: np.array(value) for name, value in productos_dict.items()}
productos = productos_dict["ean"]

# Obtenemos una lista de todos los productos por nombre
productos_nombre_dict = salesNoNa[['nombreProd']].drop_duplicates()
productos_nombre_dict = {name: np.array(value) for name, value in productos_nombre_dict.items()}
productos_nombre = productos_nombre_dict["nombreProd"]

# Obtenemos una lista de todas las categorias
categorias_dict = salesNoNa[['categoria']].drop_duplicates()
```

```
categorias_dict = {name: np.array(value) for name, value in categorias_dict.items()}
categorias = categorias_dict["categoria"]

datasetEan = list(ventas["ean"])
datasetCategoria = list(ventas["categoria"])

dfCategorias = []
for lista in datasetCategoria:
  mylist = list(dict.fromkeys(lista))
  dfCategorias.append(mylist)

salesNoNa.describe()
```

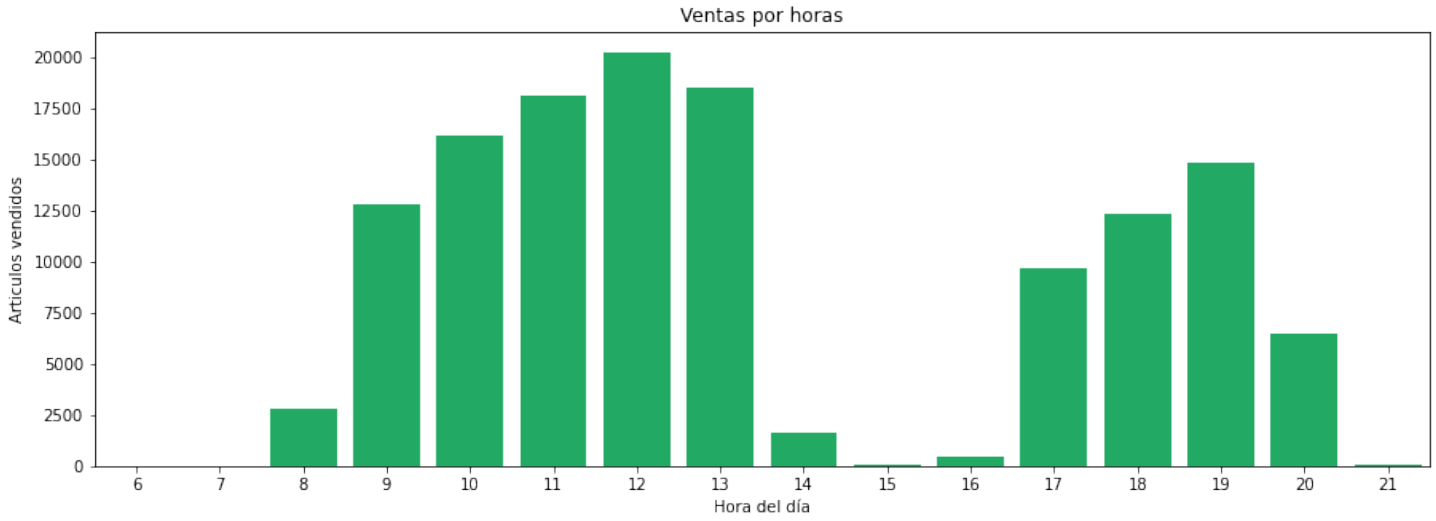
```
       cantidadVendida            iva         margen   precioCompra  \
count    134398.000000  134131.000000  134131.000000  134131.000000
mean          1.671773      12.948021      16.267967       0.874965
std           3.360436       6.202472     119.598336       0.723032
min           0.000000       4.000000     -98.200000       0.000000
25%           1.000000      10.000000       6.389163       0.460000
50%           1.000000      10.000000      11.244898       0.700000
75%           1.000000      21.000000      16.753623       1.020000
max         192.000000      21.000000   14990.909090      13.900000

         precioFinal           hora      diaSemana
count  134131.000000  134398.000000  134398.000000
mean        1.194178      13.503177       3.017790
std         0.958019       3.665026       2.081611
min        -1.950000       6.000000       0.000000
25%         0.600000      11.000000       1.000000
50%         1.000000      12.000000       3.000000
75%         1.360000      18.000000       5.000000
max        19.900000      21.000000       6.000000
```

Exploratory Data Analysis

¿A qué hora suelen comprar los clientes?

```
fig,ax = plt.subplots()
fig.set_size_inches(15,5)
sn.countplot(data=salesNoNa,x="hora",ax=ax,color="#0BC163")
ax.set(xlabel='Hora del día', ylabel="Articulos vendidos", title="Ventas por horas")
```

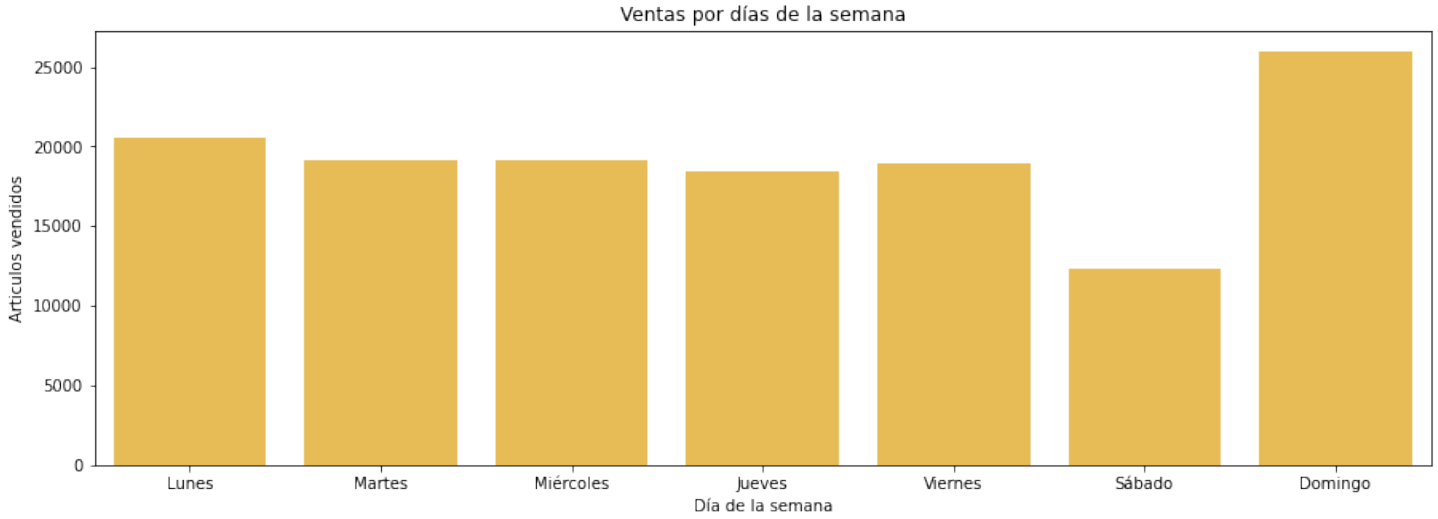
```
[Text(0, 0.5, 'Articulos vendidos'),
 Text(0.5, 0, 'Hora del día'),
 Text(0.5, 1.0, 'Ventas por horas')]
```



¿En qué días de la semana suelen comprar los clientes?

```
fig,ax = plt.subplots()
fig.set_size_inches(15,5)

ordersDay = salesNoNa[["diaSemana"]].replace(
    {0:"Lunes",1:"Martes",2:"Miércoles",3:"Jueves",4:"Viernes",5:"Sábado",6:"Domingo"})
sn.countplot(color="#FFC43D",data=ordersDay,x="diaSemana",
             ax=ax,order=["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"])
ax.set(xlabel='Día de la semana', ylabel="Articulos vendidos",title="Ventas por días de la semana")
```

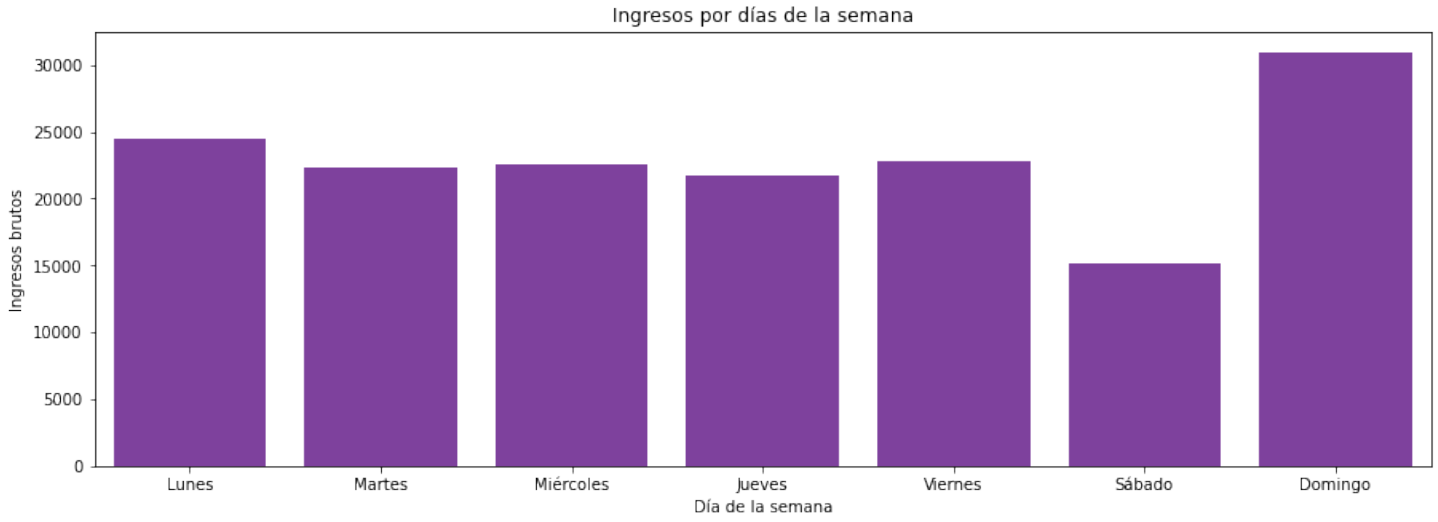
```
[Text(0, 0.5, 'Articulos vendidos'),
 Text(0.5, 0, 'Día de la semana'),
 Text(0.5, 1.0, 'Ventas por días de la semana')]
```



Días con más ingresos

```
fig,ax = plt.subplots()
fig.set_size_inches(15,5)

income = salesNoNa.groupby(["diaSemana"])[["precioFinal"]].sum().reset_index()
ordersDay = income[["diaSemana"]].replace(
    {0:"Lunes",1:"Martes",2:"Miércoles",3:"Jueves",4:"Viernes",5:"Sábado",6:"Domingo"})
ordersDay["precioFinal"] = income["precioFinal"]
sn.barplot(color="#8332AC",data=ordersDay,x="diaSemana",y="precioFinal",
           ax=ax,order=["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"])
ax.set(xlabel='Día de la semana', ylabel="Ingresos brutos",title="Ingresos por días de la semana")
```

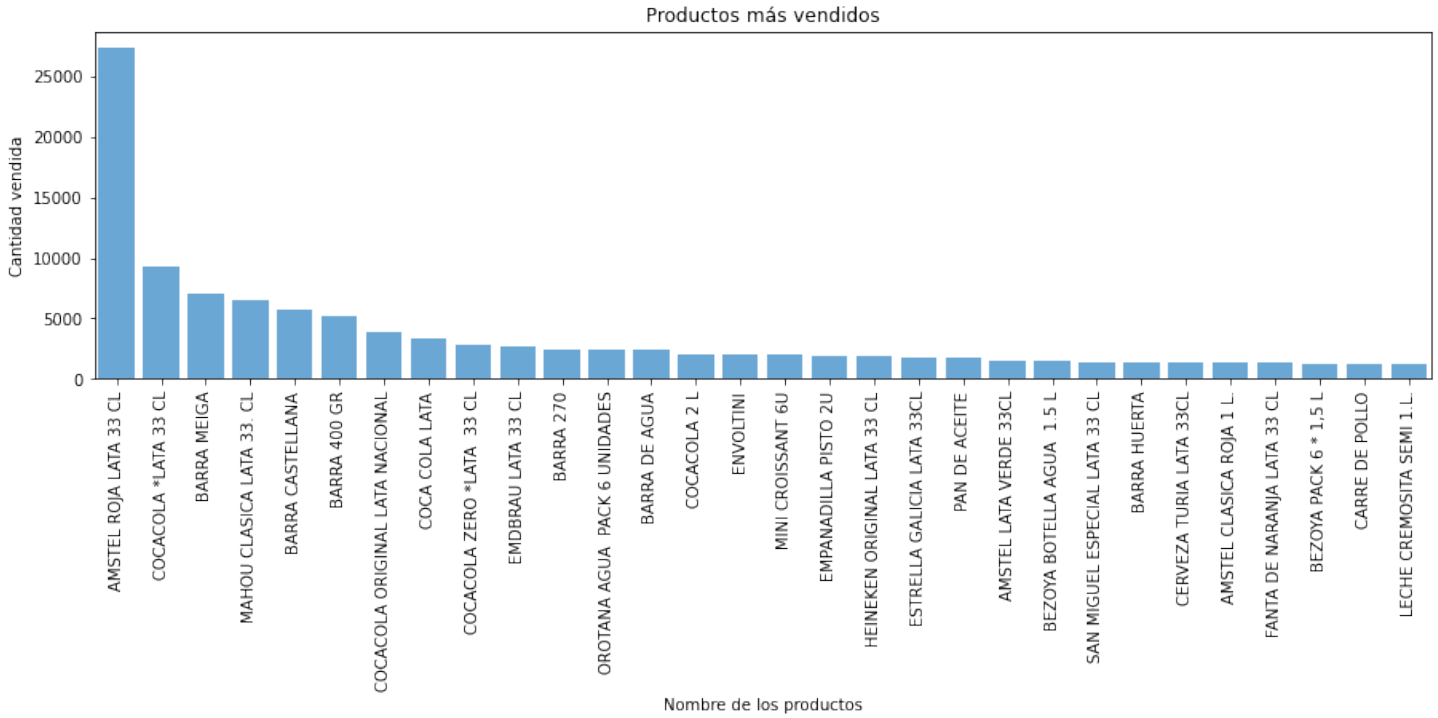
```
[Text(0, 0.5, 'Ingresos brutos'),
 Text(0.5, 0, 'Día de la semana'),
 Text(0.5, 1.0, 'Ingresos por días de la semana')]
```



Productos más vendidos

```
productsCount = salesNoNa.groupby(["nombreProd"])[["cantidadVendida"]].sum().reset_index()

pAux = pd.DataFrame(productos_nombre, columns=["nombreProd"])
mergedData = pd.merge(productsCount, pAux, how="left", on="nombreProd").sort_values(
    by="cantidadVendida",ascending=False)

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="nombreProd",y="cantidadVendida",ax=ax,orient="v",color="#5AA9E6")
ax.set(xlabel='Nombre de los productos',ylabel="Cantidad vendida",title="Productos más vendidos")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 <a list of 30 Text major ticklabel objects>)
```

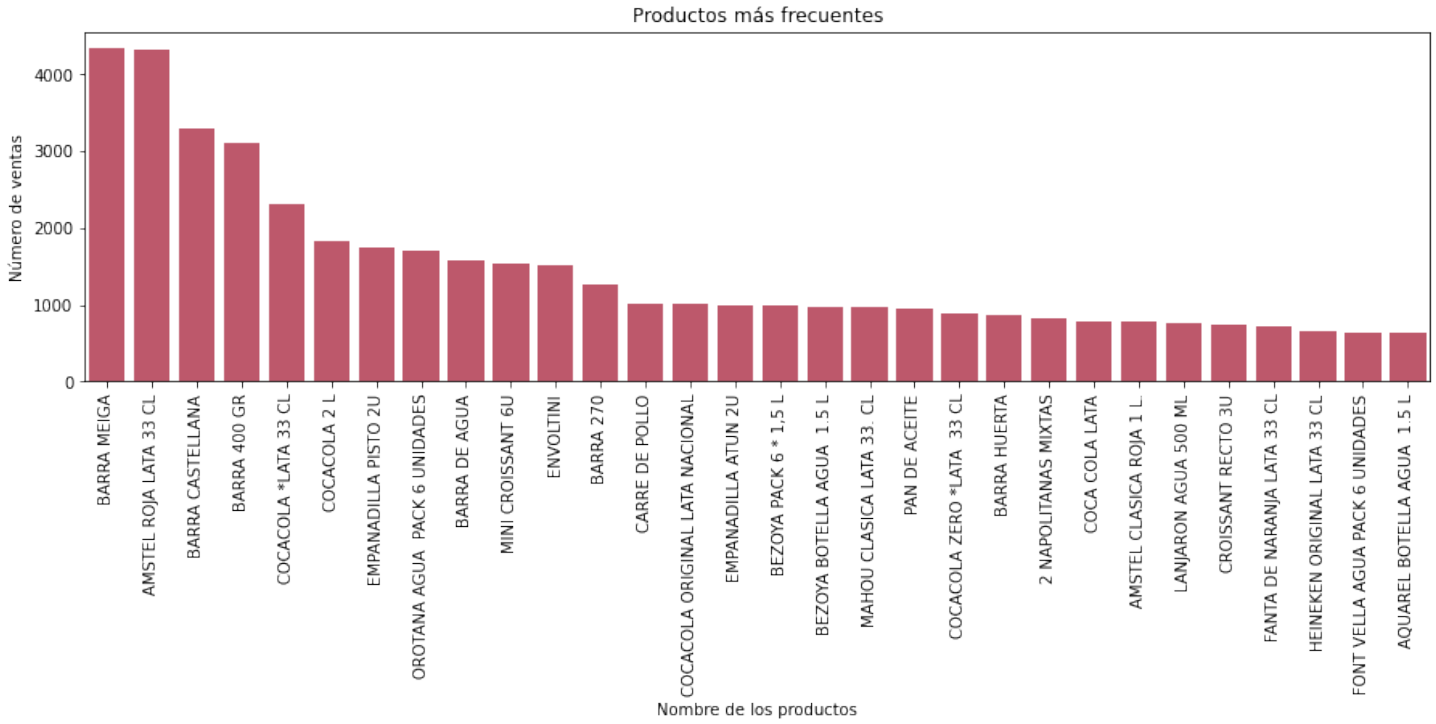


Productos más frecuentes

Número de ventas registrados que contienen cada producto

```
productsCount = salesNoNa["nombreProd"].value_counts().to_frame()
productsCount["count"] = productsCount.nombreProd
productsCount["nombreProd"] = productsCount.index

pAux = pd.DataFrame(productos_nombre, columns=["nombreProd"])
mergedData = pd.merge(productsCount, pAux, how="left", on="nombreProd").sort_values(by="count",ascending=Fals

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="nombreProd",y="count",ax=ax,orient="v",color="#CE4760")
ax.set(xlabel='Nombre de los productos',ylabel="Número de ventas",title="Productos más frecuentes")
plt.xticks(rotation=90)
```

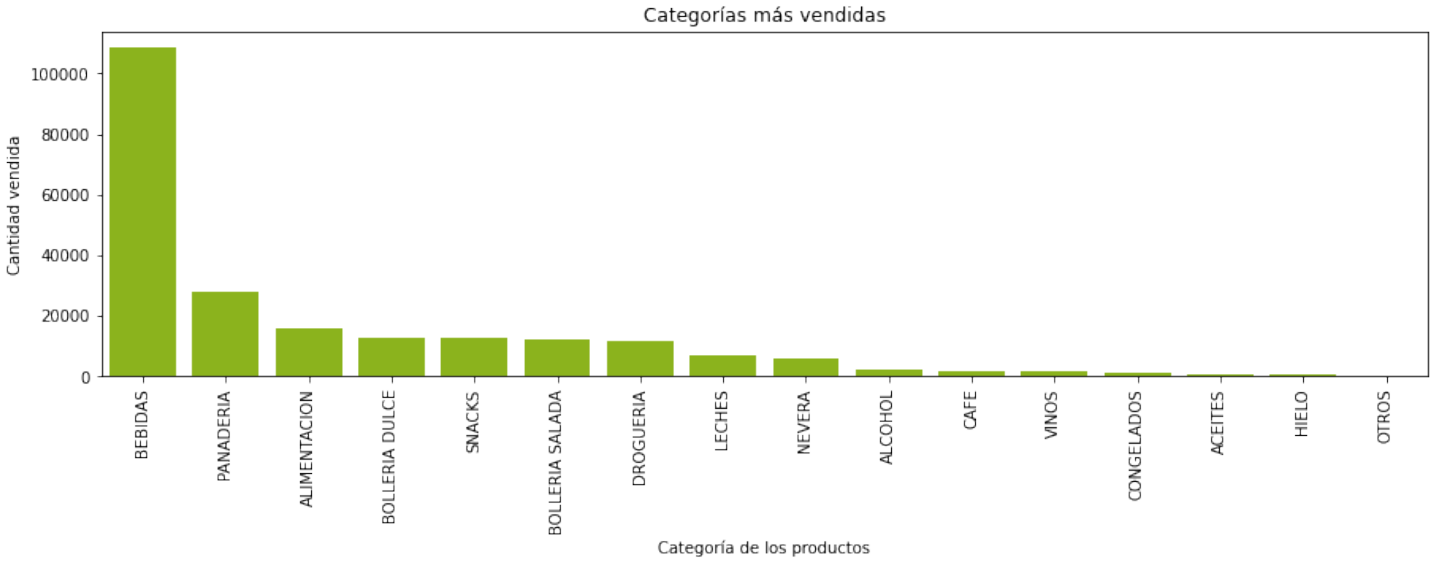
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 <a list of 30 Text major ticklabel objects>)
```



Categorias más vendidas

```
categoriaCount = salesNoNa.groupby(["categoria"])[["cantidadVendida"]].sum().reset_index()

pAux = pd.DataFrame(productos_nombre, columns=["categoria"])
mergedData = pd.merge(categoriaCount, pAux, how="left", on="categoria").sort_values(
    by="cantidadVendida",ascending=False)

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="categoria",y="cantidadVendida",ax=ax,orient="v",color="#97CC04")
ax.set(xlabel='Categoría de los productos',ylabel="Cantidad vendida",title="Categorías más vendidas")
plt.xticks(rotation=90)
```

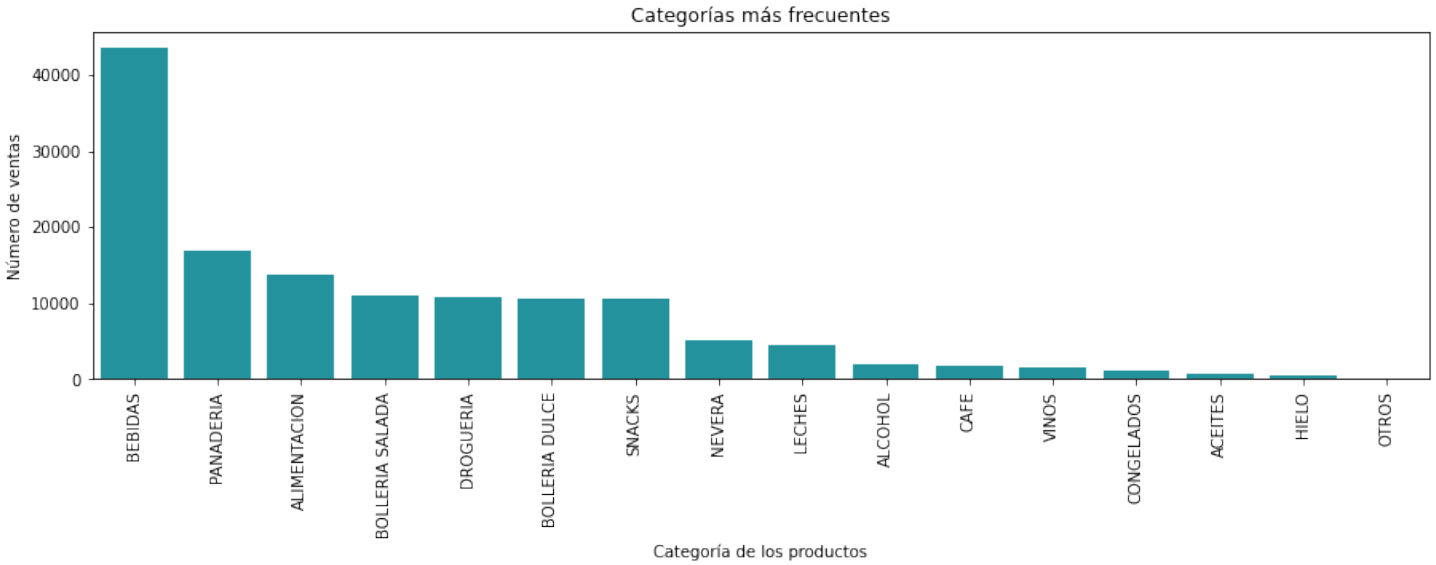
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 <a list of 16 Text major ticklabel objects>)
```



Categorías más frecuentes

```
categoriaFreqCount = salesNoNa["categoria"].value_counts().to_frame()
categoriaFreqCount["count"] = categoriaFreqCount.categoria
categoriaFreqCount["categoria"] = categoriaFreqCount.index

pAux = pd.DataFrame(productos_nombre, columns=["categoria"])
mergedData = pd.merge(categoriaFreqCount, pAux, how="left", on="categoria").sort_values(by="count",ascending=

fig,ax = plt.subplots()
fig.set_size_inches(15,4)
sn.barplot(data=mergedData.head(30),x="categoria",y="count",ax=ax,orient="v",color="#0FA3B1")
ax.set(xlabel='Categoría de los productos',ylabel="Número de ventas",title="Categorías más frecuentes")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 <a list of 16 Text major ticklabel objects>)
```



FP Growth

```
import mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import association_rules, apriori as ap, fpgrowth

def CalcularFP(dataset, min_supp):
  con = TransactionEncoder()
  con_arr = con.fit(dataset).transform(dataset)
```

```
  df = pd.DataFrame(con_arr, columns = con.columns_)

  # Ejecución del algoritmo
  resFP=fpgrowth(df, min_support=min_supp, use_colnames=True)
  association_resFP = association_rules(resFP, metric="lift", min_threshold=1.2)

  association_resFP["antecedents_len"] = association_resFP["antecedents"].apply(
      lambda x: len(x))

  # display(association_resFP[ (association_resFP['antecedents_len'] <= 1) &
  #        (association_resFP['confidence'] > 0.2) &
  #        (association_resFP['lift'] > 1.2) ])

  display(association_resFP)
  return association_resFP
```

#Patrones frecuentes en los productos

```
association_results_products = CalcularFP(dataset=datasetEan, min_supp=0.002)
```

	antecedents	consequents	antecedent support	consequent support	\
0	(MINI CROISSANT)	(CASTELLANA)	0.025884	0.055753	
1	(CASTELLANA)	(MINI CROISSANT)	0.055753	0.025884	
2	(MINI CROISSANT)	(MEIGA)	0.025884	0.073580	
3	(MEIGA)	(MINI CROISSANT)	0.073580	0.025884	
4	(5740700988349)	(5449000011527)	0.039334	0.012128	
5	(5449000011527)	(5740700988349)	0.012128	0.039334	
6	(5449000009067)	(BARRA 400 GR)	0.031108	0.052683	
7	(BARRA 400 GR)	(5449000009067)	0.052683	0.031108	
8	(5449000009067)	(MEIGA)	0.031108	0.073580	
9	(MEIGA)	(5449000009067)	0.073580	0.031108	

	support	confidence	lift	leverage	conviction	antecedents_len
0	0.002103	0.081258	1.457456	0.000660	1.027761	1
1	0.002103	0.037724	1.457456	0.000660	1.012305	1
2	0.002324	0.089777	1.220125	0.000419	1.017794	1
3	0.002324	0.031581	1.220125	0.000419	1.005883	1
4	0.002476	0.062958	5.191275	0.001999	1.054246	1
5	0.002476	0.204196	5.191275	0.001999	1.207163	1
6	0.002409	0.077426	1.469656	0.000770	1.026820	1
7	0.002409	0.045718	1.469656	0.000770	1.015310	1
8	0.002765	0.088877	1.207888	0.000476	1.016789	1
9	0.002765	0.037575	1.207888	0.000476	1.006719	1

#Patrones frecuentes en las categorias

```
association_results_categorias = CalcularFP(dataset=dfCategorias, min_supp=0.02)
```

	antecedents	consequents	antecedent support	\
0	(BOLLERIA SALADA)	(BOLLERIA DULCE)	0.160560	
1	(BOLLERIA DULCE)	(BOLLERIA SALADA)	0.147211	
2	(DROGUERIA)	(ALIMENTACION)	0.119004	
3	(ALIMENTACION)	(DROGUERIA)	0.171586	
4	(ALIMENTACION, BEBIDAS)	(PANADERIA)	0.078347	
5	(BEBIDAS, PANADERIA)	(ALIMENTACION)	0.093476	
6	(ALIMENTACION)	(BEBIDAS, PANADERIA)	0.171586	
7	(PANADERIA)	(ALIMENTACION, BEBIDAS)	0.262433	
8	(ALIMENTACION)	(SNACKS)	0.171586	
9	(SNACKS)	(ALIMENTACION)	0.141360	
10	(SNACKS)	(BOLLERIA DULCE)	0.141360	
11	(BOLLERIA DULCE)	(SNACKS)	0.147211	
12	(ALIMENTACION)	(LECHES)	0.171586	
13	(LECHES)	(ALIMENTACION)	0.070951	
14	(ALIMENTACION)	(NEVERA)	0.171586	
15	(NEVERA)	(ALIMENTACION)	0.062386	
16	(NEVERA)	(PANADERIA)	0.062386	

```
17              (PANADERIA)                (NEVERA)           0.262433

    consequent support   support  confidence      lift  leverage  conviction  \
0             0.147211  0.028869    0.179801  1.221382  0.005233    1.039734
1             0.160560  0.028869    0.196106  1.221382  0.005233    1.044216
2             0.171586  0.031532    0.264966  1.544219  0.011113    1.127042
3             0.119004  0.031532    0.183768  1.544219  0.011113    1.079345
4             0.262433  0.025443    0.324746  1.237442  0.004882    1.092280
5             0.171586  0.025443    0.272183  1.586281  0.009403    1.138218
6             0.093476  0.025443    0.148280  1.586281  0.009403    1.064344
7             0.078347  0.025443    0.096949  1.237442  0.004882    1.020600
8             0.141360  0.033618    0.195927  1.386020  0.009363    1.067864
9             0.171586  0.033618    0.237821  1.386020  0.009363    1.086903
10            0.147211  0.025663    0.181545  1.233229  0.004853    1.041950
11            0.141360  0.025663    0.174329  1.233229  0.004853    1.039930
12            0.070951  0.023560    0.137307  1.935234  0.011386    1.076917
13            0.171586  0.023560    0.332058  1.935234  0.011386    1.240250
14            0.062386  0.026206    0.152728  2.448138  0.015502    1.106628
15            0.171586  0.026206    0.420065  2.448138  0.015502    1.428461
16            0.262433  0.024917    0.399402  1.521919  0.008545    1.228054
17            0.062386  0.024917    0.094946  1.521919  0.008545    1.035976

    antecedents_len
0                 1
1                 1
2                 1
3                 1
4                 2
5                 2
6                 1
7                 1
8                 1
9                 1
10                1
11                1
12                1
13                1
14                1
15                1
16                1
17                1
```

##Recomendador

```python
def Recomendar(association_result: pd.DataFrame, categorias_carrito, confidence_min: float, lift_min: float):
  recomendador: pd.DataFrame = association_result[ (association_result['antecedents_len'] <= 1) &
                                    (association_result['confidence'] >= confidence_min) &
                                    (association_result['lift'] >= lift_min) ]

  indice = -1
  indiceRecomendador = -1
  reco = {}

  for index, recomendacion in recomendador.iterrows():
    indiceRecomendador += 1
    confianza = -1

    for indexCategorias, categoria in enumerate(categorias_carrito):
      if categoria in recomendacion["antecedents"]:
        if confianza < recomendacion["confidence"]:
          confianza = recomendacion["confidence"]
          indice = indiceRecomendador

  if indice >= 0:
```

```
    recomendacionFinal = recomendador.iloc[[indice]]
    res = list(list(recomendacionFinal["consequents"])[0])[0]
    return res

  return None

# Ejecución del modelo
# Recomendar(association_result=association_results_categorias, categorias_carrito=["NEVERA", "DROGUERIA", "
```