Universidad de Matanzas

Facultad de Ciencias Técnicas

Departamento de Informática

TRABAJO DE DIPLOMA EN OPCIÓN DEL TÍTULO DE INGENIERO INFORMÁTICO

SISTEMA DE ANÁLISIS DE VENTAS SOPORTADO EN ALGORITMOS DE MINERÍA DE DATOS

Autor: Omar Sarmiento Rolo

Tutores: Dr.C. Liz Pérez Martínez

Ing. Ramsey Ricardo Busto Martínez

Matanzas, 2023

Declaración de Autoría y Nota Legal

Yo, Omar Sarmiento Rolo, declaro que soy el único autor de la investigación titulada Sistema informático para la gestión y el análisis de las ventas para el bar Bacardí y, en virtud de tal, cedo los derechos sobre la misma a la Universidad de Matanzas, bajo la licencia Creative Commons de tipo Reconocimiento No Comercial Sin Obra Derivada, con lo cual se permite su copia y distribución por cualquier medio siempre que mantenga el reconocimiento de sus autores, no haga usocomercial de la obra y no realice ninguna modificación de ella.

Para que así conste firmo la presente a los_____días del mes de_____del año_____.

_____

Omar Sarmiento Rolo

Opinión del tutor

Resumen

Los bares constituyen una parte de la vida cultural de nuestro país, y se han convertido en lugares muy populares para los turistas y los habitantes locales por igual, llegando a fomentarse como una de las fuentes económicas más importantes que en los últimos años ha experimentado una expansión significativa. La presente investigación tiene como objetivo desarrollar un sistema de gestión y análisis de ventas para el bar Bacardí, utilizando el marco de trabajo Django, lenguaje de programación Python y MySQL como gestor de base de datos, guiado por la metodología de desarrollo XP. En primer lugar, se realiza un análisis detallado de los requerimientos del negocio para determinar las características necesarias del sistema, posteriormente se lleva a cabo el diseño de la arquitectura del sistema y se implementa utilizando el marco de trabajo Django y el lenguaje de programación Python. La metodología de desarrollo XP se utiliza para garantizar el trabajo en equipo y la calidad del software. Se obtuvo un sistema que permite la gestión y el análisis de las ventas del bar Bacardí utilizando diferentes criterios, como el tipo de producto vendido, el horario de venta y la frecuencia de compra de los clientes. El sistema también genera informes y estadísticas útiles para la toma de decisiones del negocio.

Palabras claves: Sistemas de gestión, sistemas de análisis de ventas, bares, análisis de datos, Django, Python.

Abstract

Bars are a part of the cultural life of our country, and have become very popular places for tourists and local residents alike, even becoming an important driver of the economy and experiencing significant expansion in recent years. The aim of this research is to develop a sales analysis system for the bar Bacardí using the Django framework, Python programming language, and MySQL as the database manager, guided by the XP development methodology. Firstly, a detailed analysis of the business requirements is carried out to determine the necessary characteristics of the system, followed by the design of the system architecture and its implementation using the Django framework and Python programming language. The XP development methodology is used to ensure teamwork and software quality. In conclusion, a system was obtained that allows the management and analysis of the sales of the bar Bacardí using different criteria, such as the type of product sold, the time of sale and the frequency of purchase of customers. The system also generates useful reports and statistics for business decision-making.

Keyworks: Management systems, sales analysis systems, bars, data analysis, Django, Python.

Tabla de Contenido

Tabla de Contenido

Tabla de Contenido

Introducción

Los bares son establecimientos que se encuentran en todos los lugares del mundo y son un elemento común de la cultura en muchos países. Han existido durante siglos y han evolucionado junto con las sociedades en las que se encuentran. Los bares tienen una función social, ofreciendo lugares donde las personas pueden socializar, relajarse y disfrutar. Los bares también son importantes desde un punto de vista económico, debido que generan tanto empleo directo como indirecto y contribuyen significativamente a la economía local. Además, su presencia puede tener un efecto positivo en la vida nocturna y el turismo de una ciudad.

Estos también son una parte de la vida cultural de nuestro país, y se han convertido en lugares muy populares para los turistas y los habitantes locales por igual. Los bares cubanos ofrecen una amplia variedad de bebidas, así como comidas típicas y espectáculos en vivo. La industria de los bares en Cuba constituye una de las importantes para la economía del país y en los últimos años ha experimentado una expansión significativa.

Sin embargo, muchos de los propietarios de los bares suelen llevar a cabo sus operaciones comerciales sin una adecuada gestión y análisis de sus ventas.

La gestión de las ventas en los bares es esencial para garantizar el éxito y la rentabilidad del negocio. Para lograrlo, es importante maximizar los ingresos mediante estrategias de fijación de precios adecuadas, promociones efectivas y venta cruzada de productos. Al mismo tiempo, es fundamental controlar los costos asociados, como los ingredientes, suministros e inventario, para mejorar larentabilidad general del negocio.

Además, analizar las ventas brinda información valiosa sobre la demanda y las preferencias de

los clientes. Esto permite adaptar el menú, agregar nuevos productos o ajustar las ofertas según los gustos y necesidades de los clientes, mejorando así la satisfacción del cliente y fomentando su fidelidad.

La gestión de las ventas también implica identificar tendencias y oportunidades en el mercado. Al observar los patrones de consumo y las preferencias de los clientes, se pueden descubrir nuevas oportunidades de venta, como la introducción de eventos temáticos, promociones especiales o la ampliación de la oferta de productos para satisfacer nuevas demandas. Asimismo, contribuye a optimizar la eficiencia operativa del bar. Al analizar las ventas por hora, día de la semana o temporada, se puede determinar cuándo se necesita más personal, ajustar los horarios de apertura y cierre, y optimizar la distribución de recursos para atender la demanda de manera más eficiente.

Sin embargo, a pesar de la importancia que conllevan en el éxito de cualquier negocio de servicios, incluidos los bares, en el Bar Bacardí no es nada habitual la realización de esta práctica, lo que trae como consecuencia la afectación de la rentabilidad del negocio, no se tiene una comprensión clara de cuáles son sus productos más vendidos y cuáles no, en ocasiones se enfrentan a una gestión deficiente del inventario, al lento movimiento de productos menos populares y al desperdicio de recursos. También pueden incurren en gastos excesivos en la compra de suministros innecesarios, disminuyendo su margen de beneficio además de afectar la calidad del servicio si no se sabe lo que sus clientes consumen y cuáles son sus preferencias, lo que se resume en un servicio deficiente. Los clientes pueden sentirse insatisfechos con la selección de bebidas y comidas, la calidad del servicio y la experiencia en general del bar. La falta de atención a la satisfacción del cliente puede resultar en una caída de las ventas y, por tanto, una disminución en la reputación y popularidad delbar.

Por esta razón, es crucial analizar el modelo de negocio del bar y su relación con el análisis de las ventas, especialmente en lo referente a conceptos como la rentabilidad, la satisfacción del cliente, la calidad del servicio y la gestión de inventario.

Teniendo en cuenta esta situación se define el siguiente **problema de investigación**: ¿cómo desarrollar un sistema informático o software que contribuya en el proceso de gestión y análisis de las ventas del bar Bacardí?

Como **hipótesis** se plantea: mediante el uso de las herramientas y tecnologías actuales, es posible desarrollar un sistema informático que contribuya en el proceso de gestión y análisis de las ventas del bar Bacardí.

Se asume como **objeto de estudio**: Los sistemas informáticos para la gestión y el análisis de las ventas en los bares.

De acuerdo con lo planteado anteriormente, se establece como **objetivo general**: desarrollar un sistema informático que contribuya en el proceso de gestión y análisis de las ventas del bar Bacardí.

Para dar cumplimiento al objetivo de esta investigación se definieron los siguientes **objetivos específicos:**

1. Analizar los referentes bibliográficos que permitan el establecimiento de las bases teóricas de la investigación.

2. Diseñar el sistema informático para la gestión y análisis de las ventas en el bar Bacardí a partir de requisitos funcionales y no funcionales.

3. Implementar el sistema informático para la gestión y análisis de las ventas en el bar Bacardí a partir de requisitos funcionales y no funcionales.

4. Realizar las pruebas de validación al sistema informático para la gestión y análisis de las ventas en el bar Bacardí a partir de requisitos funcionales y no funcionales.

Para el desarrollo de la investigación se utilizaron diversos métodos tales como:

Métodos Teóricos:

- **Analítico-sintético:** para el análisis de la bibliografía necesaria y el estudio de los principales elementos que componen los sistemas de análisis de ventas, lo cual, permite una mejor comprensión del problema a resolver y así arribar a conclusiones que sustenten la necesidadde realizar esta investigación,

- **Inductivo-deductivo:** para realizar el estudio de las principales herramientas y tecnologías existentes para los sistemas de ventas, revisando sus características propias, y según las mismas, definir las que debe cumplir el sistema que se propone en el presente trabajo de diploma,

- **Histórico-lógico:** para realizar un análisis de las tendencias actuales y soluciones similares en los sistemas informáticos para la gestión y el análisis de ventas,

- **Modelación:** para la elaboración de múltiples diagramas, lo que permite un mejor entendimiento del problema y solución que se propone.

Métodos Empíricos:

- **Consulta bibliográfica:** para consultar las fuentes de información relacionadas con los diversos tipos de sistemas informáticos para la gestión y el análisis de ventas,
- **Generalización:** permite estructurar los elementos más significativos en cada capítulo de la investigación y arribar a conclusiones más objetivas y explícitas.

Técnicas para la obtención de información:

- **La entrevista:** para realizar encuentros planificados con el cliente y de esta forma obtener información referente a la gestión de las ventas del bar y los diferentes tipos análisis de ventas que se quieren desarrollar, fomentando una buena comunicación con el equipo de desarrollo.

Atendiendo a lo planteado anteriormente el trabajo de diploma quedó estructurado como sigue:

Capítulo 1. Fundamentación teórica: en este capítulo se muestra la elaboración del marco teórico de la investigación. Se analizan y se exponen las características principales que argumentan la propuesta y permiten un acercamiento al objeto de estudio como las principales tendencias, tecnologías, metodologías y *softwares* utilizados en la actualidad para el desarrollo de sistemas de gestión y análisis de ventas.

Capítulo 2. Análisis y diseño del sistema: en este capítulo se presentan los fundamentos que sostienen al sistema propuesto, se describen sus funcionalidades y características esenciales, proceso que será guiado por la metodología de desarrollo seleccionada. En el mismo se realiza el modelo de dominio para describir las principales entidades que intervienen, se obtienen los requisitos funcionales y no funcionales, se define además la arquitectura y los patrones de diseño que seutilizan para el desarrollo del sistema informático.

Capítulo 3. Implementación y pruebas: en este capítulo se describe la fase de implementación del sistema, según la metodología propuesta. Se realizan además una serie de pruebas con el objetivo de entregarle al cliente un producto funcional, que cumpla con los requisitos demandados, verificándose así, el cumplimiento de los objetivos trazados al inicio de la investigación.

Finalmente, se presentan las conclusiones y las recomendaciones de la investigación. De igual forma, quedan recogidas las bibliografías y los anexos.

Capítulo 1: Fundamentación Teórica

En este capítulo se muestra la elaboración del marco teórico de la investigación. Se analizan y se exponen las características principales que argumentan la propuesta y permiten un acercamiento al objeto de estudio como las principales tendencias, tecnologías, metodologías y *softwares* utilizados en la actualidad para el desarrollo de sistemas de análisis de ventas.

1.1 Principales elementos para el análisis de ventas del bar Bacardí

El análisis de ventas es un proceso utilizado en la gestión empresarial para estudiar y evaluar el rendimiento de las ventas de una empresa en un período de tiempo determinado. Según (Kotler y Keller, 2016), el análisis de ventas permite a las empresas identificar las oportunidades y amenazas del mercado, así como evaluar la efectividad de sus estrategias de marketing y ventas en relación con los objetivos de la empresa.

- **Ventas:** Es el proceso de persuadir y convencer a los clientes para que adquieran un producto o servicio ofrecido por una empresa, mediante la comunicación efectiva de sus beneficios y características (Kotler y Armstrong, 2018),
- **Registro de ventas:** Llevar un registro detallado de todas las transacciones de ventas realizadas en el bar, incluyendo información como la fecha, hora, productos vendidos y precios,
- **Segmentación de productos:** Clasificar los productos del bar en diferentes categorías, como bebidas alcohólicas, bebidas sin alcohol, alimentos, aperitivos, etc. Esto facilita el análisis de las ventas por categoría y ayuda a identificar los productos más populares,
- **Análisis de ventas por período de tiempo:** Evaluar las ventas en períodos específicos, como diario, semanal, mensual o estacional. Esto permite identificar patrones de demanda y ajustar la oferta en consecuencia,

- **Análisis de clientes:** Registrar información sobre los clientes, como datos demográficos, preferencias de compra, frecuencia de visita, etc. Esto permite identificar a los clientes leales, comprender sus necesidades y adaptar las estrategias de ventas y marketing en consecuencia,
- **Comparación con períodos anteriores:** Realizar comparaciones con los datos de ventas de períodos anteriores para identificar tendencias de crecimiento, estacionalidad u otros cambios significativos en las ventas. Esto ayuda a evaluar el progreso y ajustar las estrategias en consecuencia.

1.2 Antecedentes

PowerBI

Power BI es la solución más novedosa de Microsoft en el mundo del *Business Intelligence* en *Cloud* a la vez que una potente herramienta de análisis, con la que puede analizar los datos necesarios en cada momento y obtener las mejores conclusiones para una toma de decisiones rápida y eficaz. También tiene una versión de escritorio (*Desktop*) que se trata de un servicio en línea de *Business Intelligence* con el que se puede:

- Crear paneles
- Compartir informes
- Conectarse directamente a todos los datos

Incluye las funcionalidades para conectar, dar forma y compartir perspectivas gracias a un contenedor flexible que permite arrastrar y soltar contenidos. *Power BI Desktop* permite transformar datos, crear informes y visualizaciones eficaces (Power y col., 2021).

GMDH Streamline

Con sede en Nueva York, GMDH Streamline es una solución local de planificación automática de reabastecimiento de inventario y pronóstico de la demanda en memoria. Un sólido enfoque de

descomposición en series temporales permite crear un pronóstico de gran precisión estadística que proporciona una base sólida para futuros procesos de planificación de la demanda. GMDH Streamline incorpora efectivamente tecnologías y estrategias modernas de planificación con herramientas de optimización de inventario y proporciona información crucial y oportuna para la toma de decisiones. Es ideal para comercios minoristas, mayoristas, distribuidores, fabricantes y de comercio electrónico (Capterra, 2023).

Looker Studio

Es una plataforma de inteligencia empresarial y análisis de datos desarrollada por *Looker*. Según su sitio web, *Looker Studio* ofrece un entorno de autoservicio que permite a los usuarios de negocios crear y compartir sus propios informes y paneles de control, al mismo tiempo que proporciona una sólida capacidad de modelado y análisis de datos para los profesionales de datos.
La plataforma incluye una variedad de herramientas de visualización de datos, descubrimiento y exploración de datos, informes y paneles personalizados, segmentación y filtración de datos, y herramientas de colaboración y compartición. También se integra con otras herramientas de análisis de datos y se puede personalizar para satisfacer las necesidades específicas de la empresa (Looker, 2021).

Monday Sales CRM

Es una solución personalizable sin código desarrollada dentro del sistema operativo de trabajo monday.com que permite a los gerentes y equipos de ventas controlar todo el ciclos de ventas, desde la captura de leads y la gestión del flujo de ventas hasta la gestión de posventa, todo en una plataforma centralizada. Los gerentes tienen una vista general completa de la capacidad de su equipo de ventas y del progreso mediante paneles sin código que pueden crear ellos mismos fácilmente. La interfaz es agradable y muy intuitiva de usar. Aparte de las operaciones habituales de ventas, monday sales CRM permite a los equipos de ventas gestionar operaciones de posventa. Esto incluye la incorporación de clientes nuevos, la gestión de proyectos de clientes, el seguimiento de los cobros y la recopilación de conocimientos sobre las necesidades y los tiempos de los clientes (GetApp, 2023).

De forma general cada uno de estos sistemas presentan una versión gratuita que ofrece diversas funciones muy limitadas y básicas para el usuario final, dicha versión carece de soporte técnico y del resto de las funciones avanzadas. Para acceder a estas funciones avanzadas, es necesario realizar el pago por algún paquete de funciones y el coste varía en dependencia, además, este pago puede no corresponderse con la cantidad de funciones o se incluyen funciones en el paquete que no son necesarias para el cliente, por lo cual, la empresa corre el riesgo de pagar por lo que no va a consumir. Por tanto, se decide desarrollar un sistema adaptado a las necesidades del cliente tomando las mejores experiencias y características de las soluciones previamente detectadas.

1.3 Metodologías para el desarrollo de software

Según (Pressman, 2015), la metodología de desarrollo de software se refiere a un marco de trabajo utilizado para planificar, estructurar y controlar el proceso de desarrollo de software. Se trata de un conjunto de prácticas y procedimientos sistemáticos que se utilizan en el proceso de desarrollo de software para garantizar la calidad del producto final y cumplir con los objetivos del proyecto.
Las metodologías de desarrollo de software se clasifican en dos grupos:

- Tradicionales como RUP (*Rational Unified Process*) y MSF (*Microsoft Solution Framework*),
- Ágiles: como Programación Extrema (eXtreme Programming o XP) y SCRUM.

Tabla 1.1: Fuente: Tomada de (García Navarro, 2018)

Tradicionales	Ágiles
Orientada a proyectos de cualquier tamaño	Orientada a proyectos pequeños
Equipos grandes y dispersos	Equipos pequeños, sobre 10 personas
Proyectos de media / larga duración	Proyectos de corta duración
Proyecto cerrado	Proyecto abierto a cambios

Continúa en la siguiente página

Tabla 1.1 Continuación en la página anterior

El cliente mantiene reuniones con la dirección	El cliente está integrado en el equipo
Arquitectura prefijada	Arquitectura se va mejorando
Documentación rigurosa	Poca documentación
Roles específicos	Roles genéricos
Roles no intercambiables	Roles flexibles
Centrada en los procesos	Centrada en las personas
Gestión dirigida	Gestión colaborativa
Alto coste de prototipado	Bajo coste de prototipado
Planificación inicial alta	Planificación inicial baja
Basada en estándares de desarrollo	Basadas en heurísticas
Poco *feedback*	Continuo *feedback*
Proceso lineal	Proceso iterativo
El coste se acerca a lo estimado	El coste puede dispararse

Para el desarrollo de la presente investigación se opta por utilizar una metodología de desarrollo ágil porque, en primer lugar, son más flexibles y adaptables a cambios, lo que significa que pueden responder de una forma eficaz a requisitos nuevos o cambiantes del cliente. Además, las metodologías ágiles promueven una colaboración más estrecha entre el equipo de desarrollo y el cliente, lo que permite una mejor comprensión de las necesidades y requerimientos del proyecto. Esto, a su vez, conduce a un producto final más útil y satisfactorio para el cliente. Otra ventaja es que permiten un desarrollo más rápido y frecuente de prototipos y versiones del software. Esto significa que los problemas o errores pueden detectarse y corregirse más temprano en el ciclo de vida del proyecto, lo que ahorra tiempo y dinero en correcciones tardías.

Metodologías ágiles de desarrollo

En la actualidad existen varios tipos metodologías ágiles, cada una aportando al desarrollo ágil distintos métodos que ayudan a mejorar de una manera eficaz la calidad del software. Entre las metodologías ágiles más utilizadas se encuentran Scrum y XP.

Scrum es un marco que se utiliza para desarrollar, entregar y mantener productos complejos (Sutherland y Schwaber, 2020). Algunas de las características claves de Scrum son:

- Equipos autoorganizativos: los equipos de desarrollo Scrum organizan su propio trabajo, se auto-dirigen y deciden cómo cumplir los objetivos de los sprints,
- Orden: El dueño del producto de Scrum prioriza la pila del producto, pero el equipo determinael orden en que se desarrollan,
- Sesiones de tiempo fijo: Scrum trabaja en sprints de tiempo fijo (de dos semanas a un mes), lo que significa que los equipos de desarrollo y grupos interesados tienen predictibilidad en cuanto a los plazos de entrega,
- Reuniones: Scrum tiene diferentes tipos de reuniones diarias de manera rutinaria, como Scrum diario, Revisión del Sprint, Planificación del Sprint y Retrospectivas del Sprint,
- Enfoque empírico: el equipo Scrum implanta un proceso de retroalimentación y adaptación constante para maximizar el valor entregado,
- Pruebas de software: se realizan al final de cada Sprint, estas se componen de pruebas unitarias, pruebas de integración, pruebas de aceptación.

Por otro lado, XP es un enfoque ágil para la gestión de proyectos de software que se centra en la entrega rápida y la adaptabilidad a los cambios en los requisitos del cliente. Algunas de sus características son las siguientes (Beck, 2000; Jeffries, Anderson y Hendrickson, 2001; Larman, 2003):

- Equipo: todos los miembros trabajan juntos para lograr un objetivo común,
- Comunicación frecuente: es un pilar fundamental, fomentando la colaboración y la transparencia entre los miembros del equipo y con el cliente,
- Tiempo: se trabaja en iteraciones muy cortas de una o dos semanas
- Simplicidad: prioriza la entrega de pequeñas funcionalidades que agreguen valor al usuario de forma regular,

- Cambios: XP busca una gestión de proyectos flexible, adaptándose a las necesidades del cliente a lo largo del desarrollo,
- Orden de trabajo: trabajan en un orden estricto, el orden de desarrollo de los requerimientos los determina el cliente junto con el equipo de desarrollo,
- Pruebas: la calidad del software se garantiza mediante la realización de pruebas continuas, incluyendo la programación en parejas y la revisión de código.

Elección de la metodología de desarrollo de software

A partir de las características mencionadas anteriormente se opta por la elección de la metodología de desarrollo ágil XP, debido a que el cliente tiene una participación activa y directa en la solución propuesta. Además, la entrega temprana y regular de funcionalidades permite al cliente tener un mayor control sobre el resultado final del proyecto, y debido a esto es necesario una mayor flexibilidad a los cambios de requisitos, lo que reduce el riesgo de desarrollar un producto que no satisfaga las necesidades del cliente. Por último, como el desarrollo es llevado a cabo por un equipo es necesario la comunicación y la transparencia entre todos los miembros del equipo fomentandoel trabajo colaborativo y la creatividad.

Metodología de desarrollo XP

Programación Extrema (XP) propone cuatro fases de desarrollo (Valladarez, Gaitan y Reyes, 2016):

Planificación: se realiza el análisis del negocio y el levantamiento de requisitos, se elabora un plan de entrega y su estimación.

- **Las Historias de Usuarios:** son descritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar,
- **El Plan de Entregas (*Release Plan*):** las historias de usuarios serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reuniónentre todos los actores del proyecto,

- **El Plan de Iteraciones (*Iteration Plan*):** las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido,
- **Reuniones Diarias de Seguimiento (*Stand – Up Meeting*)**: el objetivo es mantener la comunicación entre el equipo y compartir problemas y soluciones.

Diseño: en esta fase se diseña el sistema, se realiza el modelado de las clases y se define la arquitectura del sistema.

- **Simplicidad:** un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione,
- **Soluciones *"Spike"*:** cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados *"Spike"*), para explorar diferentes soluciones,
- **Recodificación *("Refactoring")*:** consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de crearlo más simple, conciso y entendible. Las metodologías de XP sugieren recodificar cada vez que sea necesario,
- **Metáforas:** XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, así como guiar la estructura del mismo. Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

Implementación: se desarrolla el sistema a partir del diseño realizado.

- **Disponibilidad del Cliente:** uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. Al comienzo del proyecto, este debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de "alto nivel", no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles

deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo,

- **Uso de Estándares:** XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación,
- **Programación Dirigida por las Pruebas (*"Test-Driven Programming"*):** primero se escriben los *tests* que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a los que se refiere esta práctica, son las pruebas unitarias, realizadas por los desarrolladores. La definición de estos *tests* al comienzo, condiciona o "dirige" el desarrollo,
- **Programación en Pares:** XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que esta práctica duplica el tiempo asignado al proyecto (y, por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales,
- **Integraciones Permanentes:** todos los desarrolladores necesitan trabajar siempre con la "última versión". Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez,
- **Propiedad Colectiva del Código:** en un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, una pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o recodificar,
- **Ritmo Sostenido:** la Metodología XP indica que debe llevarse un ritmo sostenido de trabajo. El concepto que se desea establecer con esta práctica es planificar el trabajo de forma a mantener un ritmo constante y razonable, sin sobrecargar al equipo.

Pruebas: se prueba y se verifica el correcto funcionamiento del sistema. Luego de realizadas las pruebas y la corrección de los errores detectados, se comienza una nueva iteración.

- **Pruebas Unitarias:** todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código (*"Test-Driven Programmming"*). Que todo código libe- rado pase correctamente las pruebas unitarias, es lo que habilita que funcione la propiedad colectiva del código,

- **Detección y Corrección de Errores:** cuando se encuentra un error, éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto,

- **Pruebas de Aceptación:** son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El Cliente debe especificar uno o diversos escenarios para compro- bar que una historia de usuario ha sido correctamente implementada. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación (Joskowicz, 2008).

XP define varios roles clave en su metodología de desarrollo ágil. Es importante destacar que los roles en XP pueden variar según el contexto y las necesidades del proyecto (Beck, 2004):

- **Cliente** (***Customer***): El cliente es la parte interesada que representa a los usuarios o la organización que solicita el desarrollo del software. El cliente proporciona los requisitos, establecelas prioridades y toma decisiones sobre las funcionalidades del producto,

- **Programador** (***Programmer***): Los programadores son los miembros del equipo de desarrollo responsables de escribir el código del software. Trabajan en estrecha colaboración con los clientes para comprender los requisitos y desarrollar las funcionalidades,

- ***Tester***: Los *testers* son responsables de llevar a cabo pruebas en el software para identificar

errores y problemas. Realizan pruebas unitarias, de integración y de aceptación para garantizar la calidad del producto,

- **Entrenador** (***Coach***): El entrenador, también conocido como "*coach*", es un facilitador externo que guía y apoya al equipo de desarrollo en la implementación de XP. Proporciona orientación en cuanto a las prácticas y valores de XP, y ayuda a resolver conflictos o desafíos,

- ***Tracker***: El *tracker* es el encargado de llevar un registro de las historias de usuario, las tareas y el progreso del proyecto. Mantiene actualizada la lista de tareas pendientes y ayuda a coordinar y visualizar el avance del equipo,

- **Líder del equipo** (***Team Leader***): El líder del equipo tiene la responsabilidad de facilitar la comunicación y la colaboración dentro del equipo. Ayuda a tomar decisiones técnicas, supervisa el progreso del proyecto y actúa como un enlace entre el equipo de desarrollo y los clientes.

1.4 Tecnologías y herramientas a utilizar

1.4.1 Lenguaje de modelado

El Lenguaje de Modelado Unificado (UML, por sus siglas en inglés) es un lenguaje visual utilizado para modelar sistemas de software. Se utiliza para representar gráficamente diferentes aspectos de un sistema de software, como su estructura, comportamiento y relaciones. El propósito principal de UML es facilitar la comunicación y comprensión entre los diferentes stakeholders y miembros del equipo de desarrollo de software. Al utilizar un lenguaje visual común, se pueden evitar malentendidos y ambigüedades en la especificación y diseño de un sistema de software. UML consta de diferentes tipos de diagramas, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, entre otros, cada uno con su propia sintaxis y semántica. Estos diagramas se pue- den utilizar para modelar diferentes aspectos del sistema, como su arquitectura, comportamiento ointeracciones entre los componentes (Budgen, 2013).

1.4.2 Herramienta CASE

Una herramienta CASE (Computer-Aided Software Engineering) es un software utilizado para apoyar el desarrollo de software de manera automatizada. Estas herramientas se utilizan en ca-

da una de las etapas del ciclo de vida del software, desde la especificación de requisitos hasta la implementación y mantenimiento. Pueden incluir diferentes funcionalidades, como la gestión de requisitos, el modelado de sistemas, la generación de código, la depuración y el control de versiones. Estas permiten una mayor eficiencia y calidad en el desarrollo de software, reduciendo los errores y el tiempo necesario para el desarrollo y mantenimiento del software (Sommerville, 2016). Existen varias herramientas CASE en el mercado, entre ellas ArgoUML, Visual Paradigm, Rational Rose.

ArgoUML

ArgoUML es una herramienta CASE gratuita y de código abierto que se utiliza para modelar sistemas de software utilizando UML (Lenguaje de Modelado Unificado). La herramienta fue creada en 1998 por Jason Robbins y ha sido desarrollada y actualizada por un equipo de desarrolladores de software de todo el mundo.

ArgoUML es compatible con diferentes plataformas y permite la creación de diferentes tipos de diagramas UML, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, entre otros. La herramienta también permite la generación de código a partir de los modelos creados y la importación y exportación de modelos en diferentes formatos, como XMI y SVG (ArgoUML, 2023).

A pesar de que ArgoUML es una herramienta gratuita y de código abierto que ofrece muchas funcionalidades útiles, también tiene algunas desventajas que la desechan como alternativa para la presente investigación, entre estas se encuentran:

- Problemas de compatibilidad: puede presentar problemas de compatibilidad con algunas versiones de sistemas operativos o con otros programas de software, lo que puede dificultar su uso,
- Interfaz desactualizada: Aunque cuenta con una interfaz gráfica de usuario intuitiva, esta se encuentra desactualizada en comparación con otras herramientas CASE disponibles en el mercado,
- Problemas con el guardado de archivos: En ocasiones los archivos se corrompen producto a fallas del Sistema Operativo y no pueden ser recuperados.

Rational Rose

Rational Rose es una herramienta CASE utilizada para modelar sistemas de software utilizando UML. La herramienta fue desarrollada inicialmente por Rational Software Corporation y posteriormente adquirida por IBM. Rational Rose permite la creación de diferentes tipos de diagramas UML, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, entre otros. La herramienta también permite la generación de código a partir de los modelos creados y la importación y exportación de modelos en diferentes formatos.
Rational Rose es compatible con diferentes plataformas y cuenta con una interfaz gráfica de usuario intuitiva y fácil de usar. La herramienta es ampliamente utilizada en la industria de software para el modelado de sistemas de software complejos y para la generación de código a partir de los modelos creados (IBM, 2023).
Sin embargo esta herramienta no posee licencia gratuita, por lo que se descarta como alternativa para el desarrollo de la presente investigación.

Visual Paradigm

Visual Paradigm es una herramienta potente, multiplataforma y fácil de usar para el diseño y la gestión de sistemas informáticos. Cuenta con una amplia gama de funcionalidades en su versión gratuita, incluyendo la creación de diferentes tipos de diagramas UML, la generación de código a partir de los modelos creados, la importación y exportación de modelos en diferentes formatos. Proporciona a los desarrolladores de software una plataforma de desarrollo innovadora para construir aplicaciones de calidad más rápido, mejor y más barato. Además, también cuenta con una comunidad activa de usuarios y desarrolladores que pueden proporcionar soporte y actualizaciones para la herramienta (Paradigm, 2023a; b), por lo que teniendo en cuenta sus características se selecciona como herramienta CASE para la presente investigación.

1.4.3 Marco de trabajo (*Framework*)

Según Fowler y Parsons (2019), un framework es un conjunto de clases y otros tipos de código que proporcionan una estructura para el desarrollo de aplicaciones y componentes". Constituye un conjunto de herramientas, bibliotecas y estándares que proporcionan una estructura común y reutilizable para el desarrollo de software. Está diseñado para acelerar el proceso de desarrollo y

minimizar la necesidad de escribir código desde cero (Fowler y Parsons, 2019). La elección de un framework para depender de varios factores, como los conocimientos y experiencia en programación, la naturaleza del proyecto y las necesidades específicas del mismo.

1.4.3.1 Lado del servidor

Teniendo en cuenta que en la presente investigación será necesario desarrollar algoritmos relacionados con Inteligencia Artificial se hará un análisis de marcos de trabajos por el lado del servidor basados en el lenguaje de programación Python. Esta elección se debe a que si se emplea un marco de trabajo que no es basado en Python, se hace necesario investigar cómo realizar las integraciones correspondientes de las librerías necesarias al marco de trabajo seleccionado, con lo cual, se pueden presentar dificultades durante este proceso, retrasando así el desarrollo de la investigación. Entre los diferentes marcos de trabajos por el lado del servidor basados en Python *Django*, *Flask* y *Pyramid*. A continuación, se describen de manera breve algunas de sus características:

Django

Django es un framework de desarrollo web en Python que facilita la creación de aplicaciones seguras, escalables y mantenibles. Proporciona un conjunto de herramientas para el manejo de bases de datos, el manejo de URLs, la administración de sesiones, el caché y la seguridad, las mismas aceleran y facilitan el desarrollo de aplicaciones web (Foundation, 2023b). Es gratuito y de código abierto. Este marco de trabajo se caracteriza porque permite construir aplicaciones de forma rápida, segura y con menos código (en comparación con otros marcos de trabajos). Además, cuenta con una comunidad de colaboradores muy grande a nivel mundial que se encarga de realizar las debidas actualizaciones tanto del marco de trabajo como de su documentación de forma diaria (Gómez García, 2018).

Pyramid

Pyramid es un framework web de Python que facilita el desarrollo de aplicaciones web de forma flexible y escalable. Se basa en el enfoque minimalista y permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus necesidades. Pyramid es conocido por su simplicidad y su capacidad para construir aplicaciones web robustas. Es

gratuito y de código abierto, orientado a objetos y usa un sistema de configuración basado en decoradores que permite registrar las vistas, los eventos, los permisos y otros componentes de forma clara y modular. Además, es compatible con varios sistemas de plantillas y se puede integrar con diferentes sistemas degestión de bases de datos (Team, 2023b).

Flask

Flask es un framework web de Python que permite crear aplicaciones web de manera rápida y sencilla. Proporciona una estructura minimalista pero poderosa que se centra en la simplicidad y la extensibilidad. Flask se utiliza ampliamente en el desarrollo de aplicaciones web y servicios API. Este se caracteriza por ser ligero, sencillo y flexible (Pallets, 2023). Algunas de sus características clave incluyen: enrutamiento y manejo de URL sencillo y flexible, soporte para el uso de plantillas para la generación dinámica de contenido, integración con bases de datos mediante extensiones como SQLAlchemy, manejo de sesiones y cookies, y soporte para la creación de APIs

RESTful. Flask se basa en el principio de "hacerlo sencillo" permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus necesidades (Grinberg, 2018).

Elección del framework de desarrollo por el lado del servidor

Tabla 1.2: Comparativa Django, Pyramid, Flask. Fuente: Elaboración propia

Aspecto	Django	Pyramid	Flask
Tipo de marco	Desarrollo web escalado	Micro-framework de aplicaciones web	Micro-framework
Base de datos	ORM integrado	SQLAlchemy, requiere extensiones para integrar con otros SGBD	SQLAlchemy, requiere extensiones para integrar con otros SGBD

Continúa en la siguiente página

Autenticación	Implementado	Implementado	Requiere extensiones
Administración	Avanzado	Requiere extensiones	Básico
Alcance	Proyectos grandes	Sitios web de pequeño a mediano alcance	Sitios web estáticos
Soporte REST	Soportado. Avanzado	Soportado. Básico	Soportado. Básico

Por las características que se mencionan en la tabla anterior se opta por seleccionar a Django como marco de trabajo por el lado del servidor.

1.4.3.2 Lado del cliente

Un framework de desarrollo por el lado del cliente es un conjunto de herramientas, bibliotecas y estructuras que facilitan la creación de aplicaciones web interactivas en el navegador del usuario. Estos se centran en la programación del lado del cliente, es decir, en el desarrollo de la interfaz de usuario y la lógica que se ejecuta en el navegador (Mikowski y Powell, 2013). Entre estos se encuentran *Vue.js*, *React* y *Angular*. A continuación, se describen de manera breve algunas de sus características:

Vue.js

Vue.js es un framework de JavaScript de código abierto utilizado para construir interfaces de usuario interactivas y reactivas en aplicaciones web. Se caracteriza por ser progresivo, lo que significa que puede ser adoptado gradualmente en proyectos existentes sin necesidad de reescribir el código por completo. Sus principales características incluyen su enfoque progresivo, lo que permite su adopción gradual en proyectos existentes; su sistema de reactividad, que actualiza automáticamente la interfaz de usuario cuando los datos cambian; el uso de componentes reutilizables que facilitan la creación de interfaces modulares; un conjunto de directivas que amplían la funcionalidad HTML; y una comunidad activa que proporciona soporte, complementos y recursos adicionales. Vue.js se destaca por su simplicidad, flexibilidad y rendimiento, lo que lo convierte en una opción popular para el desarrollo ágil de aplicaciones web modernas y reactivas (Team, 2023d).

React

React es una biblioteca de JavaScript desarrollada por Facebook que se utiliza ampliamente para crear interfaces de usuario interactivas y eficientes en aplicaciones web. Se basa en un modelo de programación declarativo y componentes reutilizables. Sus principales características incluyen el uso de componentes reutilizables, que permiten la creación de interfaces modulares y escalables; el concepto de Virtual DOM, que mejora el rendimiento al aplicar solo los cambios necesarios en el DOM del navegador; la unidireccionalidad de datos, que simplifica el seguimiento de los cambios y el mantenimiento del estado de la aplicación; JSX, una sintaxis que combina JavaScript y HTML para facilitar la composición de componentes; y una comunidad activa que proporciona soporte, bibliotecas y recursos adicionales (Team, 2023c).

Angular

Angular es un framework de desarrollo de aplicaciones web de código abierto creado por Google. Ofrece un enfoque completo para construir aplicaciones web robustas y escalables. Sus características principales incluyen una arquitectura MVC que permite una separación clara de la lógica de negocio y la representación visual, el uso de componentes reutilizables para construir interfaces modulares, un sistema de inyección de dependencias que mejora la modularidad y testabilidad, directivas integradas que facilitan la manipulación del DOM de manera declarativa, y un enrutador que permite la navegación fluida entre diferentes vistas de la aplicación (Team, 2023a).

Elección del framework de desarrollo por el lado del cliente

Tabla 1.3: Comparativa Vue.js, React, Angular. Fuente: Elaboración propia

Aspecto	Vue.js	React	Angular
Curva de aprendizaje	Baja	Moderada	Alta
Continúa en la siguiente página			

Rendimiento y recursos	Alto rendimiento, bajos recursos	Alto rendimiento, bajos recursos	Rendimiento medio-alto, Consumo de recursos en aumento según crece la aplicación
Document-Object-Model (DOM)	Virtual	Virtual	Regular
Flexibilidad	Amplia gama de herramientas y librerías	Amplia gama de herramientas y librerías	Amplia gama de herramientas y librerías, aunque presenta una estructura más rigurosa, lo que lo hace en ocasiones poco flexible
Documentación y comunidad	Alta documentación y comunidad. En aumento constante	Alta documentación y comunidad	Documentación completa en su sitio oficial, aunque se aprecia un gran descenso en su comunidad

Por las características que se mencionan en la tabla anterior se opta por seleccionar a Vue.js como marco de trabajo por el lado del cliente.

1.4.4 Lenguajes de programación

Un lenguaje de programación es un conjunto de reglas y estructuras sintácticas utilizadas para escribir programas de computadora. Proporciona un conjunto de instrucciones que pueden ser ejecutadas por una computadora para realizar tareas específicas (Sebesta, 2020).

Python

Es el lenguaje seleccionado para el desarrollo del sistema debido a que es el lenguaje que maneja el marco de trabajo *Django*. es un lenguaje de programación interpretado, orientado a objetos y de alto nivel, con una semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con tipado dinámico y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de script o de unión para conectar componentes existentes entre sí. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo del mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización de código. El intérprete de Python y la extensa biblioteca estándar están disponibles en forma de código fuente o binario sin cargo para todas las plataformas principales, y se pueden distribuir libremente (Foundation, 2023a).

HTML

HTML (HyperText Markup Language) es un lenguaje de marcado utilizado para estructurar y presentar contenido en la web. Es la base fundamental de la mayoría de los sitios web y se utiliza para definir la estructura y el formato del contenido, como textos, imágenes, enlaces y otros elementos multimedia (Docs, 2023b). A lo largo de los años las versiones de HTML han evolucionado con el objetivo de adaptarse a los nuevos tiempos y así dar soporte a nuevas necesidades (estandarizaciónde los sistemas de audio, vídeo) (Jiménez Ortega, 2017).

CSS

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para describir la presentación visual y el diseño de un documento HTML. Permite definir estilos, como colores, fuentes, márgenes y posicionamiento, para aplicar a elementos individuales o conjuntos de elementos en una página web (Docs, 2023a).

JavaScript

JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente en el desarrollo web. Es conocido por su capacidad para agregar interactividad y dinamismo a las páginas web, permitiendo la manipulación de elementos

HTML, gestión de eventos, acceso a API del navegador y comunicación con servidores (c).

Typescript

TypeScript es un lenguaje de programación de código abierto desarrollado por *Microsoft* que se basa en el lenguaje *JavaScript*. *TypeScript* se considera un "superset"de *JavaScript*, lo que significa que es compatible con *JavaScript*, pero también agrega una serie de características adicionales que lo hacen más seguro, legible y mantenible.

Entre las características más destacadas de *TypeScript* se incluyen la tipificación estática, la inferencia de tipos, las clases, los módulos y la compatibilidad con las últimas especificaciones de ECMAScript (Freeman, 2020).

1.4.5 Servidor Web

Un servidor web es un software o un sistema informático que procesa las solicitudes de los clientes y les entrega los recursos web solicitados. Actúa como un intermediario entre los navegadores web de los clientes y los archivos o servicios que forman parte de un sitio web. El servidor web es responsable de recibir las solicitudes HTTP o HTTPS, procesarlas y enviar las respuestas correspondientes, que generalmente incluyen páginas web, imágenes, archivos CSS, scripts y otros recursos (Mozilla, 2023). A partir de la elección del marco de trabajo *Django*, servidores web como *Apache o Nginx* son alternativas viables para el despliegue de la solución.

1.4.6 Sistema gestor de base de datos

Un sistema de gestión de base de datos (SGBD) es un conjunto de software que permite administrar, organizar y manipular de manera eficiente una base de datos. Proporciona una interfaz para crear, modificar y consultar la base de datos, garantizando la integridad, seguridad y disponibilidad de los datos almacenados. Además, ofrece herramientas para realizar copias de seguridad, recuperación ante fallos y optimización del rendimiento de las consultas (Connolly y Begg, 2014). Entrelos gestores de base de datos se encuentran *PostgreSQL*, *SQLite y MySQL* :

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto y robusto. Proporciona un entorno de base de datos completo con soporte para consultas SQL avanzadas, transacciones, integridad de datos, replicación y escalabilidad. PostgreSQL se destaca por su estabilidad, rendimiento y capacidad para manejar grandes volúmenes de datos. Es ampliamente utilizado en diversas aplicaciones y entornos, desde pequeñas empresas hasta grandes organizaciones. Ofrece un completo soporte SQL, transacciones ACID y una gran escalabilidad y rendimiento. Además, se destaca por su capacidad para garantizar la integridad referencial y aplicar restricciones, así como por su flexibilidad en la definición de funciones almacenadas y procedimientos. Con un amplio conjunto de tipos de datos avanzados, PostgreSQL se adapta a diversas necesidades. Su enfoque en la seguridad y el control de acceso, junto con su reputación como una de las bases de datos más avanzadas, lo convierten en una opción sólida para proyectos de todos los tamaños y complejidades (Group, 2023).

MySQL

Se ha convertido en el SGBD de código abierto más MySQL es un sistema de gestión de bases de datos relacional de código abierto. Es ampliamente utilizado en el desarrollo de aplicaciones web y es conocido por su rendimiento, confiabilidad y facilidad de uso. Destaca por su escalabilidad, rendimiento y soporte multiplataforma. Con amplio respaldo para diferentes lenguajes de programación y características de seguridad avanzadas, MySQL se adapta a diversos entornos de desarrollo y garantiza la integridad de los datos. Además, ofrece funcionalidad extensible a través de complementos y una eficiente gestión de transacciones con soporte para transacciones ACID. Con estas características, MySQL se posiciona como una opción confiable y flexible para aplicaciones de alto rendimiento y requerimientos específicos. (Corporation, 2023).

SQLite

SQLite es un sistema de gestión de bases de datos relacional de código abierto que se caracteriza por ser ligero, autónomo y de uso sencillo. A diferencia de otros sistemas de gestión de bases de datos, SQLite se implementa como una biblioteca embebida en la aplicación, lo que facilita su in-

tegración en una amplia variedad de proyectos y plataformas. SQLite es compatible con la mayoría de las características estándar de SQL y es utilizado en una amplia gama de aplicaciones, desde dispositivos móviles hasta aplicaciones de escritorio. Ofrece soporte para transacciones ACID, lo que garantiza la integridad de los datos, y permite la ejecución eficiente de consultas y operaciones en entornos con grandes volúmenes de datos. (Consortium, 2023).

Elección del sistema gestor de base de datos

Al analizar las características de los diferentes gestores de bases de datos anteriormente mencionados se determina que *PostgreSQL* posee un elevado consumo de memoria RAM, este aspecto en ocasiones reduce los tiempos de respuesta del sistema y en computadoras de prestaciones limitadas, admeás, para realizar su despliegue de forma local se requieren de conocimientos avanzados, por lo cual no es una opción viable, Por otro lado *SQLite* tiene limitaciones en cuanto al tamaño de datos, variables y su rendimiento puede verse afectado negativamente a medida que la base de datos crece en tamaño y complejidad, además, carece de funciones de seguridad y administración de usuarios, con lo cual se desecha como alternativa viable. Por último, *MySQL* es fácil de usar, es ligero, es potente y viene integrado en los servidores libres. Posee mayor rendimiento, buenas utilidades de administración y está preparado para manejar grandes volúmenes de datos, además, es altamente personalizable por lo que su despliegue es fácil de realizarse en entornos locales. Por estos motivos se decide seleccionar a *MySQL* como el gestor de base de datos de la presente investigación.

1.4.7 Entornos de desarrollo

Un Entorno de Desarrollo Integrado (*IDE*) es un conjunto de herramientas y funcionalidades que facilitan el desarrollo de software. Proporciona un entorno centralizado donde los desarrolladores pueden escribir, depurar y probar su código de manera eficiente. Un *IDE* generalmente incluye un editor de código, un compilador o intérprete, herramientas de depuración, un sistema de gestión de versiones, un explorador de archivos y otras características que mejoran la productividad del desarrollador (Jetbrains, 2023a). Para la elaboración de la solución se utilizará PyCharm como *IDE*.

PyCharm

PyCharm es un IDE (Entorno de Desarrollo Integrado) específicamente diseñado para el desarrollo de aplicaciones Python. Proporciona un conjunto de herramientas y características que ayudan a los desarrolladores a escribir, depurar y ejecutar código Python de manera eficiente. Algunas de las características de PyCharm son (b):

- **Editor de código inteligente:** PyCharm ofrece un editor de código inteligente con resaltado de sintaxis, finalización automática de código, refactorización y navegación avanzada,
- **Depurador integrado:** Permite depurar paso a paso el código Python, establecer puntos de interrupción y realizar inspecciones para detectar y solucionar errores,
- **Administración de proyectos:** PyCharm facilita la creación y gestión de proyectos Python, con herramientas para la organización de archivos, administración de dependencias y configuración de entornos virtuales,
- **Integración con herramientas de control de versiones:** Soporta sistemas de control de versiones como Git, SVN y Mercurial, lo que facilita el seguimiento y la colaboración en proyectos de desarrollo,
- **Testing y cobertura de código:** PyCharm ofrece soporte integrado para la ejecución de pruebas unitarias y pruebas de integración, así como para el análisis de cobertura de código,
- **Análisis estático y detección de errores:** Proporciona herramientas de análisis estático que ayudan a identificar posibles errores, problemas de rendimiento y prácticas de codificación no óptimas.

1.5 Conclusiones del capítulo

En el presente capítulo se identificaron soluciones anteriores que se encuentran dentro del campo de acción de la investigación, pero al analizarlos se confirma que no son los adecuados para la entidad por sus características y necesidades particulares. Se analizaron los diferentes grupos de metodologías de desarrollo de software y se optó por emplear la metodología ágil, específicamente, Programación Extrema (XP). De acuerdo al estudio realizado se decide utilizar

las siguientes tecnologías y herramientas: el lenguaje de modelado UML y el *Visual Paradigm* como herramienta CASE para visualizar, construir y documentar los artefactos del sistema, *Django* como marco de trabajo de desarrollo, *Python* como lenguaje de programación, *MySql* como Sistema de Gestión de Bases de Datos y *PyCharm* como entorno de desarrollo integrado. En sentido general se contribuye a mejorar la comprensión del objeto de estudio y se establecen las bases para las siguientes fases de la investigación.

Capítulo 2: Análisis y diseño del sistema

En el presente capítulo se reflejan las actividades realizadas en los procesos de análisis y diseño de la solución propuesta guiado por la metodología de desarrollo XP, donde se recogen las historias de usuario, costos, plan de iteraciones y plan de entrega, se realiza una descripción de la solución propuesta, planteándose los requisitos funcionales y no funcionales y se define la arquitectura.

2.1 Propuesta del sistema

Se propone un sistema llamado Sistema Análisis-Bacardí, utilizando para su desarrollo herramientas libres y que sea capaz de brindar las funcionalidades requeridas para la gestión y análisis de ventas del bar Bacardí.

2.2 Roles del sistema

Administrador: encargado de gestionar toda la información referente al sistema de gestión de ventas, incluida la gestión de usuarios y logs.
Dependiente: encargado de gestionar los módulos de pedidos y productos No tiene acceso a los reportes de ventas.
Analista: encargado de visualizar y realizar el análisis de las ventas.

2.3 Modelo de negocio

Un diagrama de proceso de negocio es una representación gráfica que muestra el flujo completo de un proceso de negocio, ayudando a visualizar la relación secuencial de las actividades mediante descripciones y símbolos. Estos diagramas son fundamentales para analizar y ver en qué aspectos se pueden introducir mejoras, especialmente para aumentar la productividad de los empleados, delimitar la responsabilidad de cada tarea y, en general, aclarar el propio flujo de trabajo. Permiten conocer los procesos empresariales dentro de un único documento y sus relaciones, identificar

puntos de mejora y, en general, dar importancia a todos los procesos de una compañía (Equipo Ekon, 2020).

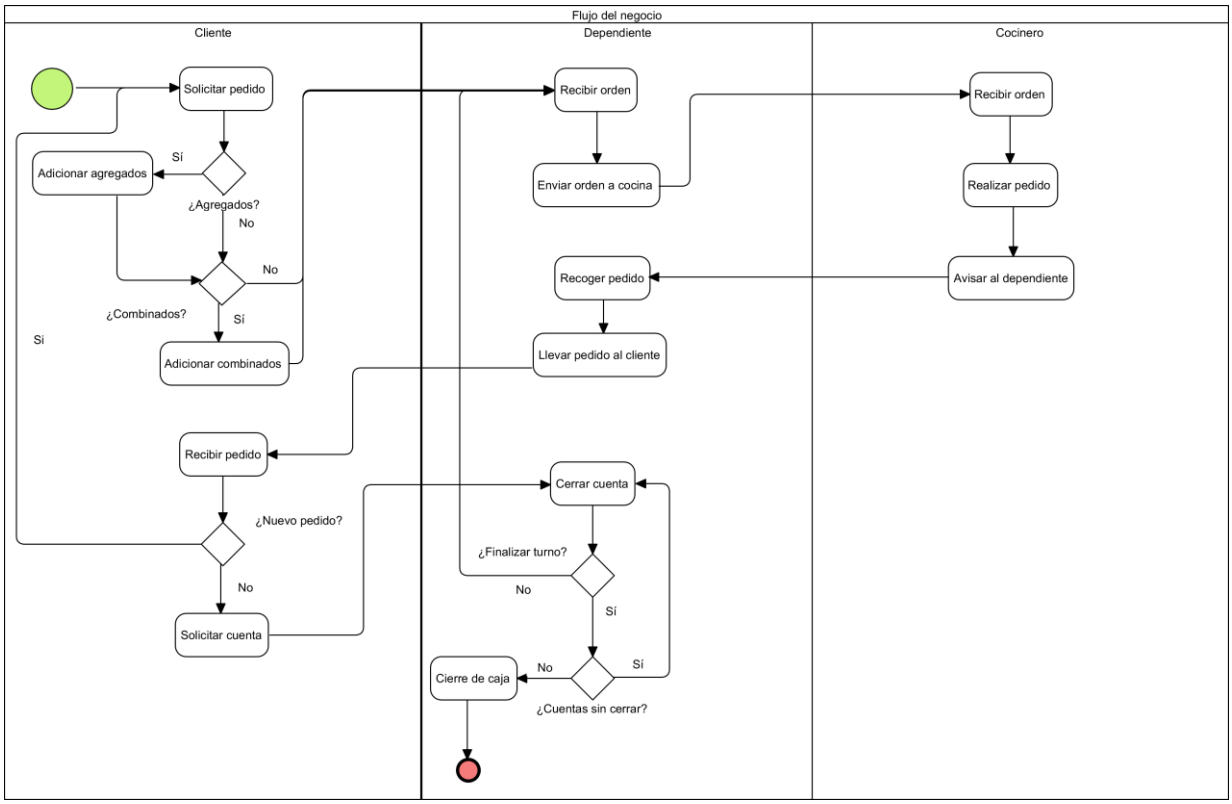

Figura 2.1: Diagrama de proceso del negocio

2.4 Fase de exploración y planificación

Es la fase inicial de la metodología XP, donde se establece una comunicación continua entre el equipo de desarrollo y el cliente, para obtener principalmente los requisitos del sistema. Además permite establecer el alcance del proyecto y fechas de entrega del sistema, tomando en cuenta en la prioridad y tiempo estimado para el desarrollo de cada historia de usuario (Valladarez, Gaitan y Reyes, 2016).

2.4.1 Requisitos funcionales

Los requisitos funcionales son aquellos que describen qué debe hacer el sistema, desde el punto de vista de las necesidades del usuario, son capacidades o condiciones que debe cumplir el sistema

y que están fuertemente ligados a las opciones del programa (Cardozzo y Academy, 2016). A continuación, se muestran los requisitos funcionales del sistema,

1. **RF-1:** Autenticar usuario en el sistema:
    a) **RF-1.1:** Iniciar sesión,
    b) **RF-1.2:** Cambiar clave de acceso,
    c) **RF-1.3:** Cerrar sesión.
2. **RF-2:** Gestionar usuarios en el panel de administración:
    a) **RF-2.1:** Listar usuarios existentes en el sistema,
    b) **RF-2.2:** Registrar un usuario en el sistema,
    c) **RF-2.3:** Actualizar un usuario en el sistema,
    d) **RF-2.4:** Eliminar un usuario en el sistema.
3. **RF-3:** Gestionar pedidos en el sistema:
    a) **RF-3.1:** Listar pedidos existentes en el sistema,
    b) **RF-3.2:** Adicionar pedido en el sistema,
    c) **RF-3.3:** Actualizar pedido en el sistema,
    d) **RF-3.4:** Visualizar detalles del pedido en el sistema.
4. **RF-4:** Gestionar productos en el sistema:
    a) **RF-4.1:** Listar productos existentes en el sistema,
    b) **RF-4.2:** Adicionar producto en el sistema,
    c) **RF-4.3:** Actualizar producto en el sistema,
    d) **RF-4.4:** Eliminar producto en el sistema,
    e) **RF-4.5:** Visualizar detalles del producto en el sistema.
5. **RF-5:** Gestionar categorías en el sistema:

a) **RF-5.1:** Listar categorías existentes en el sistema,

b) **RF-5.2:** Adicionar categoría en el sistema,

c) **RF-5.3:** Actualizar categoría en el sistema,

d) **RF-5.4:** Eliminar categoría en el sistema.

6. **RF-6:** Gestionar mesas en el sistema:

a) **RF-6.1:** Listar mesas existentes en el sistema,

b) **RF-6.2:** Adicionar mesa en el sistema,

c) **RF-6.3:** Actualizar mesa en el sistema,

d) **RF-6.4:** Eliminar mesa en el sistema,

e) **RF-6.5:** Visualizar detalles de la mesa en el sistema.

7. **RF-7:** Gestionar clientes en el sistema:

a) **RF-7.1:** Listar clientes en el sistema,

b) **RF-7.2:** Adicionar cliente en el sistema,

c) **RF-7.3:** Actualizar cliente en el sistema,

d) **RF-7.4:** Eliminar cliente en el sistema,

e) **RF-7.5:** Visualizar detalles del cliente en el sistema.

8. **RF-8:** Gestionar pagos en el sistema:

a) **RF-8.1:** Listar pagos en el sistema,

b) **RF-8.2:** Adicionar pago en el sistema,

c) **RF-8.3:** Cerrar pago en el sistema.

9. **RF-9:** Gestionar zonas en el sistema:

a) **RF-9.1:** Listar zonas en el sistema,

b) **RF-9.2:** Adicionar zona en el sistema,

c) **RF-9.3:** Actualizar zona en el sistema,

d) **RF-9.4:** Eliminar zona en el sistema.

10. **RF-10:** Gestionar comandas en el sistema:

a) **RF-10.1:** Listar comandas en el sistema,

b) **RF-10.2:** Actualizar estado de comanda en el sistema,

c) **RF-10.3:** Actualizar mesa de comanda en el sistema,

d) **RF-10.4:** Invalidar comanda en el sistema.

e) **RF-10.5:** Cerrar comanda en el sistema.

11. **RF-11:** Generar reportes en el sistema:

a) **RF-11.1:** Cierre de caja,

b) **RF-11.2:** Reporte de ventas,

c) **RF-11.3:** Historial de pedidos.

1. **RF-12:** Generar reportes estadísticos:

a) **RF-12.1:** Reporte de predicción de ventas,

b) **RF-12.2:** Reporte de predicción de ventas por productos,

c) **RF-12.3:** Reporte de compras asociativas entre productos,

d) **RF-12.4:** Reporte de probabilidad de compra de productos cuando se ha comprado otro.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son criterios o restricciones que se aplican al sistema o software, y no se relacionan directamente con las funcionalidades específicas del sistema. Estos requisitos se centran en atributos de calidad, restricciones técnicas y características generales que se esperan del sistema, como el rendimiento, la seguridad, la usabilidad, la escalabilidad, la disponibilidad, entre otros (Sommerville, 2016). Teniendo en cuenta las características del sistema se definieron los siguientes requerimientos no funcionales:

1. **RNF-1: Interfaz:**

    a) Clara y concisa. No debe dar lugar a la confusión del usuario y debe seguir los estándares de diseño,

    b) La interfaz de usuario del sistema deberá ser diseñada de forma tal que permita el aprovechamiento del espacio,

    c) Los componentes de la interfaz de usuario para la entrada de datos, de ser posible, deberán ser configurados de manera que el riesgo de entrada de datos inválidos por parte del usuario disminuya,

    d) La interfaz debe ser responsiva, es decir, debe adaptarse automáticamente al tamaño de la pantalla del dispositivo en el que se esté visualizando, garantizando así una buena experiencia de usuario en cualquier dispositivo.

2. **RNF-2: Estabilidad:** el sistema debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma y avisando de la naturalidad del error,

3. **RNF-3: Rendimiento:** el sistema debe desempeñar su función de una manera fluida. Se debe buscar la experiencia de uso más agradable para el usuario,

4. **RNF-4: Usabilidad:**

    a) El sistema debe visualizarse correctamente en dispositivos móviles y computadoras,

    b) La interfaz visual del sistema debe ser atractiva y sencilla, permitiendo al usuario facilidad de uso y entrenamiento,

5. **RNF-5: Ayuda y documentación:** se brindarán manuales de ayuda que documenten cómo trabajar de forma adecuada con el sistema,

6. **RNF-6: Hardware:** para el servidor como mínimo una computadora con un procesador de 2.0 GHz o más rápido de x64 bits, 4 GB de memoria RAM y al menos 50 GB de espacio disponible en el disco duro. Para el cliente como mínimo una computadora con un procesador de 1.0 GHz o más rápido de x64 bits, 2 GB de memoria RAM y al menos 30 GB de espacio disponible en el disco duro,

7. **RNF-7: Software:** el sistema se ejecutará en los navegadores web *Mozilla Firefox*, *Google Chrome*, *Opera* y *Microsoft Edge*. Es recomendable actualizar los navegadores a la versión más reciente,

8. **RNF-8: Seguridad:** el sistema debe garantizar la confidencialidad, integridad, disponibilidad y autenticidad de los datos y recursos.

2.4.3 Historias de Usuario

Una historia de usuario es una técnica utilizada en el desarrollo de software para capturar los requisitos funcionales desde la perspectiva del usuario. Una historia de usuario describe una funcionalidad o característica específica del sistema en un formato simple y comprensible, generalmente narrado desde el punto de vista del usuario final (Cohn, 2004b). A continuación se muestra un ejemplo de las Historias de Usuarios del sistema:

Tabla 2.1: Historia de usuario

HISTORIA DE USO			
Orden	HU_3	**Nombre**	Gestionar usuarios
Riesgo	Bajo	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	0.5
Descripción	El sistema debe brindar la funcionalidad de listar los usuarios, además, adicionar, actualizar y eliminar un usuario en el sistema, todo esto a través del panel de administración del sistema. Se realizará un sistema para la búsqueda de usuarios por nombre de usuario y correo electrónico.		
Observación			

Descripción de los campos que componen las Historias de Usuario:

- **Orden:** está constituido por dos partes. La primera está referido al nomenclador HU (Historia de Usuario) y la segunda corresponde el número de la funcionalidad que representa,

- **Nombre:** nombre que identificará a la Historia de Usuario,

- **Riesgo:** es el grado de incertidumbre en el desarrollo que se asocia a la Historia de Usuario. Determina la posibilidad real de implementarse o no con las condiciones previstas por el equipo de desarrollo (tiempo, recursos, personal). Puede ser Bajo, Medio o Alto,
- **Prioridad:** la prioridad la define el cliente, y es el grado de importancia que le concede a la funcionalidad,
- **Iteración:** es el número de la fase en la cual se define la Historia de Usuario,
- **Puntos estimados:** es un número entero que representa la cantidad de semanas que se dispone para el desarrollo de la Historia de Usuario. Las Historias de Usuario con altos puntos estimados deben ser separadas en varias tareas. Un punto es una semana efectiva de desarrollo,
- **Descripción:** se escribe una fundamentación de lo que hace la funcionalidad,
- **Observación:** se escribe los elementos o detalles que se deben tener en cuenta para la implementación de la misma.

A partir de la solución propuesta se identificaron 12 requisitos funcionales agrupados en las siguientes Historias de Usuario:

Tabla 2.2: Historias de usuario

Número	Historia de usuario
1	Diseño y creación de la base de datos
2	Sistema de autenticación
3	Gestionar usuarios
4	Gestionar zonas
5	Gestionar mesas
6	Gestionar clientes
7	Gestionar categorías
8	Gestionar productos

Continúa en la siguiente página

Tabla 2.2 Continuación de la página anterior

9	Gestionar comandas
10	Gestionar pedidos
11	Gestionar pagos
12	Generar reportes
13	Generar reportes estadísticos

El resto de las Historias de Usuarios que definen el sistema están adjuntas en los anexos del documento.

2.4.4 Estimación de esfuerzo por Historias de Usuario y plan de iteraciones

La estimación de esfuerzo por historia de usuario en XP (Extreme Programming) se refiere al proceso de asignar un nivel de esfuerzo o tamaño relativo a cada historia de usuario como una medida de la complejidad y el trabajo requerido para su implementación. Esta estimación se utiliza para planificar y asignar tareas durante el desarrollo del proyecto (b).

Para lograr una mejor organización del trabajo y proporcionar un desarrollo iterativo e incremental, se crea el plan de iteraciones donde se planifica el orden de desarrollo de las Historias de Usuario. Se definió realizar 7 iteraciones, su orden está determinada según las prioridades de las Historias de Usuario y las dependencias existentes entre ellas. La duración total de cada iteración dependerá de los puntos estimados de las Historias de Usuario que en él se desarrollan. Teniendo en cuenta lo expuesto anteriormente la estimación de esfuerzo de las Historias de Usuario y el plan de iteraciones queda como se muestra en la siguiente tabla:

Tabla 2.3: Plan de iteraciones

Iteración	Historia de Usuario	Estimación de esfuerzo
1	Diseño y creación de la base de datos Sistema de autenticación Gestionar usuarios	3 semanas
2	Gestionar zonas	4 semanas

Continúa en la siguiente página

Tabla 2.3 Continuación de la página anterior

	Gestionar mesas	
3	Gestionar clientes Gestionar categorías	4 semanas
4	Gestionar productos	2 semanas
5	Gestionar pedidos Gestionar comandas	4 semanas
6	Gestionar pagos Generar reportes	4 semanas
7	Generar reportes estadísticos	3 semanas
Total: 7	**Total: 13**	**Total: 24**

A partir de la suma de los puntos de estimación de esfuerzo por cada Historia de Usuario, se calcula que el desarrollo del sistema tendrá una duración de 24 semanas.

2.4.5 Plan de entrega

El plan de entrega en la metodología XP (*Extreme Programming*) se refiere a la planificación de los incrementos de funcionalidad y las fechas de entrega durante el desarrollo del proyecto. Es un componente esencial para gestionar el flujo de trabajo y garantizar la entrega de software funcional de manera continua (Beck, 2004).

Tabla 2.4: Plan de entrega

Iteración	Historia de Usuario	Fecha de entrega
1	3	16 de junio del 2023
2	2	14 de julio del 2023
3	2	11 de julio del 2023
4	1	25 de agosto del 2023
5	2	22 de septiembre del 2023

6	2	20 de octubre del 2023
7	1	10 de noviembre del 2023

2.5 Estimación del costo

Se realiza un desglose del coste de los elementos necesarios en esta investigación. Dichos elementos incluyen costes de personal, de hardware y de software. La investigación se realizará entre el 29 de mayo del 2023 al 10 de noviembre de 2023, por lo tanto, han sido 24 semanas de trabajo. Teniendo en cuenta una jornada laboral de 8 horas se tiene un total de 960 horas de trabajo, distribuidas entre diferentes tareas y diferentes roles profesionales que las llevana cabo.

2.5.1 Coste de personal

La metodología de software escogida propone un equipo de desarrollo pequeño donde cada integrante tiene su rol y funciones bien definidas. Para determinar el coste del personal involucrado se va desglosar el equipo de acuerdo a la categoría de cada uno, así como en la fase donde participa quedando el desglose del coste como se aprecia en la siguiente tabla:

Tabla 2.5: Coste de personal

Fase	Categoría	Horas	Coste/hora	Coste
Documentación	Analista	106	100.00 CUP	10600.00 CUP
Análisis	Analista	115	100.00 CUP	11500.00 CUP
Diseño	Diseñador	173	75.00 CUP	12975.00 CUP
Diseño gráfico	Diseñador gráfico	67	75.00 CUP	5025.00 CUP
Implementación	Programador	432	50.00 CUP	21600.00 CUP
Pruebas	Programador	67	50.00 CUP	3350.00 CUP
Total		**960**		**65050.00 CUP**

2.5.2 Coste de hardware

Para el hardware calcularemos el coste según el período de amortización teniendo en cuenta una duración del proyecto de 28 semanas. El equipo está formado por computadoras portátiles, teléfonos celulares y tablets, necesitando uno de cada uno de estos dispositivos para el desarrollo.

Tabla 2.6: Coste de hardware

Equipo	Coste	Coste de amortizado
Computadora portátil	43260.00 CUP	25015.40 CUP
Tablet	9888.00 CUP	5921.70 CUP
Teléfono celular	18540.00 CUP	11712.34 CUP
Total	71688.00 CUP	42649.44 CUP

2.5.3 Coste de software

En la realización de esta investigación se ha optado por utilizar software libre por lo que no hay ningún coste asociado al software.

2.5.4 Coste total

A partir del coste de cada uno de los elementos necesarios para la investigación se puede llegar al coste total, como se aprecia en la siguiente tabla:

Tabla 2.7: Coste total

Tipo de coste	Total
Coste de personal	65050.00 CUP
Coste de hardware	42649.44 CUP
Coste de software	0 CUP
Total	107699.44 CUP

Por tanto el coste total para la presente investigación asciende a: 118499.44 CUP.

2.6 Diagrama de paquetes

Este diagrama es el encargado de representar las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre ellas (Jiménez, 2016).

Toda esta estructura es contenida dentro de la carpeta del proyecto que en este caso es **Bacardi**. Enla siguiente imagen se muestra la estructura correspondiente a la solución:

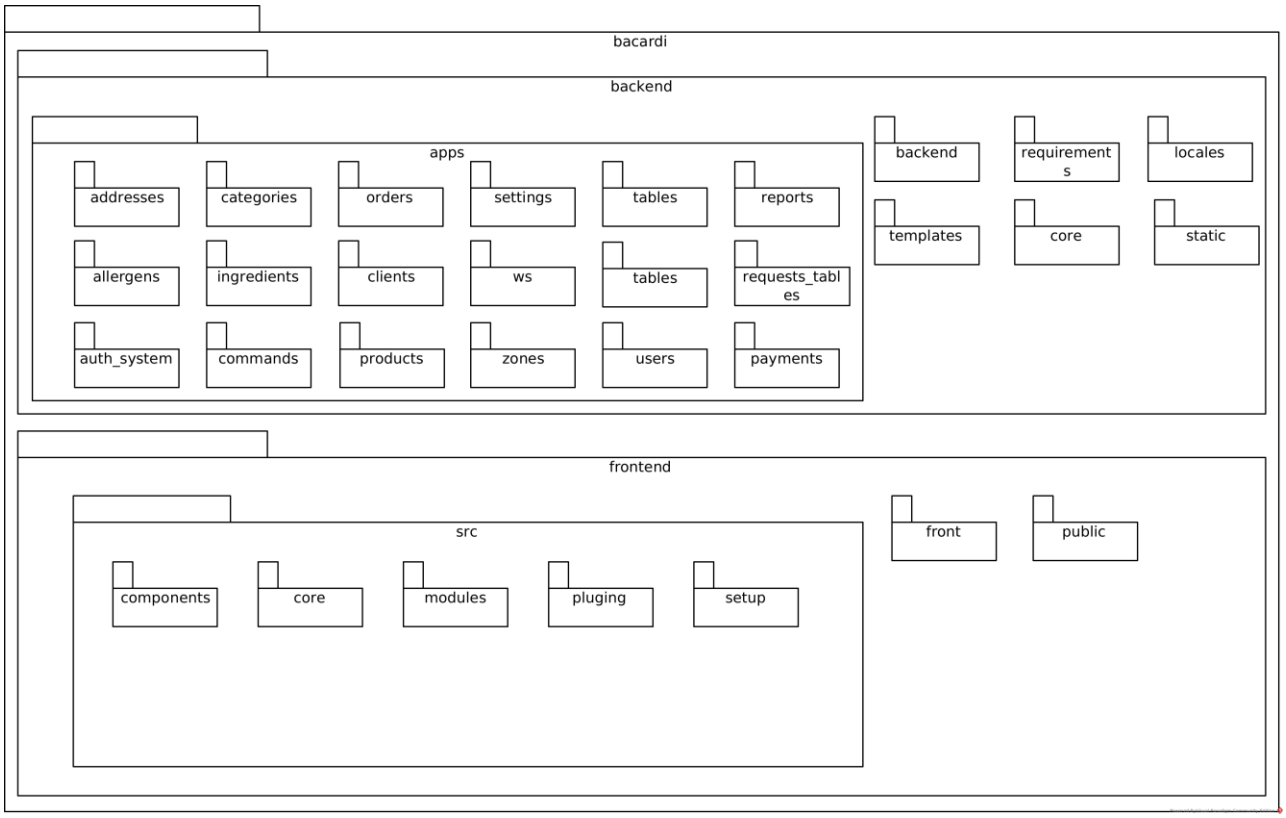

Figura 2.2: Estructura de carpetas de la solución

1. ***Backend:***

   - ***apps:*** cada uno de los paquetes contenidos dentro de este paquete contiene los modelos, la lógica y las validaciones correspondientes a la ***app*** con el mismo nombre,
   - ***backend:*** es la carpeta raíz del proyecto. Se encuentran los ficheros que manejan toda la configuración del proyecto,

- ***templates:*** se encuentran las plantillas visuales de los correos electrónicos
- ***requirements:*** se encuentran todas las dependencias utilizadas para el desarrollo del sistema,
- ***locales:*** contiene archivos de traducción para el sitio o aplicación web,
- ***static:*** contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, cadenas de texto, css y algunos ficheros que contienen la programación lógica de las vistas,
- ***core:*** se almacenan las bases para la arquitectura principal del proyecto.

2. Frontend:

- ***public:*** se almacenan los archivos estáticos del sitio,
- ***front:*** donde se almacenará el sitio compilado y listo para producción,
- ***src:*** es la carpeta raíz del proyecto,
- ***components:*** se almacenan los componentes genéricos y compartidos del proyecto,
- ***core:*** se almacenan las bases para la arquitectura principal del proyecto,
- ***modules:*** contiene cada módulo del sistema, contiene sus componentes, estilos y sus archivos de typescript,
- ***pluing:*** carpeta donde se instalan los pluings del proyecto,
- ***setup:*** configuraciones globales del proyecto y de los pluings.

2.7 Patrones de Arquitectura

Para el desarrollo de un software, se hace necesario seleccionar diferentes patrones, estos contribuyen a mantener una buena estructura y organización, lo que hace eficiente su funcionamiento. Estos patrones son una guía para cometer una determinada acción. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones, para organizar los distintos componentes.

2.7.1 Patrón de arquitectura cliente-servidor

El patrón de arquitectura cliente-servidor es un enfoque de diseño ampliamente utilizado en el desarrollo de sistemas distribuidos. En este patrón, la aplicación se divide en dos componentes principales:

- **Cliente:** Es la parte de la aplicación que interactúa directamente con el usuario. El cliente se encarga de enviar solicitudes al servidor y presentar los resultados al usuario de manera adecuada. Puede ser una interfaz de usuario, una aplicación móvil, un navegador web, entre otros,
- **Servidor:** Es la parte centralizada de la aplicación que procesa las solicitudes del cliente y envía las respuestas correspondientes. El servidor gestiona la lógica de negocio, el acceso a los datos y otros aspectos relacionados con el funcionamiento de la aplicación. Puede ser un servidor físico, una máquina virtual o un servicio en la nube.

El patrón cliente-servidor permite una arquitectura escalable y flexible, ya que varios clientes pueden conectarse simultáneamente a un servidor centralizado. Además, permite la separación de preocupaciones y la reutilización de la lógica de negocio en el servidor, lo que facilita el mantenimiento y la evolución del sistema (Thies y Fuhrmannek, 2019).

2.7.2 Patrón de arquitectura de la solución

El patrón de arquitectura MVC (Modelo-Vista-Controlador) es un enfoque de diseño utilizado en el desarrollo de aplicaciones de software para separar las responsabilidades de presentación, lógica de negocio y manipulación de datos. MVC proporciona una estructura organizada y modular que facilita la flexibilidad, mantenibilidad y reutilización del código.
En el patrón MVC, el modelo representa los datos y la lógica de negocio de la aplicación. La vista se encarga de la presentación visual de los datos y la interacción con el usuario. El controlador actúa como intermediario entre el modelo y la vista, gestionando las solicitudes del usuario, actualizando el modelo y coordinando la actualización de la vista (Gammack y Kopec, 2017).

2.7.3 Arquitectura basada en capas

La arquitectura basada en capas es un patrón de diseño arquitectónico que organiza una aplicación en capas o niveles lógicos separados. Cada capa se enfoca en una responsabilidad específica y se comunica con capas adyacentes a través de interfaces bien definidas. Esta arquitectura promueve la modularidad, el acoplamiento débil y la reutilización de componentes.
Las capas típicas en una arquitectura basada en capas incluyen:

- **Capa de presentación:** Es la capa más cercana al usuario y se encarga de la interacción con la interfaz de usuario. Puede incluir componentes como interfaces gráficas, páginas web o servicios de API,
- **Capa de lógica de negocio:** Contiene la lógica de negocio y las reglas que gobiernan el funcionamiento de la aplicación. Esta capa se encarga del procesamiento de datos, las operaciones comerciales y las reglas de validación,
- **Capa de acceso a datos:** Es responsable de acceder y manipular los datos de la aplicación. Puede incluir componentes como bases de datos, servicios web o repositorios.

La arquitectura basada en capas facilita la escalabilidad y el mantenimiento del sistema, ya que permite la sustitución o actualización de una capa sin afectar a las demás. También permite la reutilización de componentes, ya que las capas están bien definidas y se pueden utilizar en diferentes contextos (Buschmann y col., 2020).

2.8 Patrones de diseño

Los patrones de diseño son soluciones probadas y comprobadas para problemas comunes en el diseño de software. Son enfoques estructurados y reutilizables que proporcionan una guía para resolver problemas recurrentes en el desarrollo de software. A continuación se describen los patrones de diseño utilizados:

Método de plantilla (*Template Method*)

El patrón de diseño *Template Method* es un patrón de comportamiento que se utiliza en el desarrollo de software para definir un esqueleto de algoritmo en una clase base, mientras se

permiten que las subclases implementen partes específicas del algoritmo según sea necesario. En este patrón, se define una clase base que contiene un método principal llamado "*template method*". Este define la estructura general del algoritmo y utiliza métodos abstractos o "*hooks*"(métodos que pueden ser sobrescritos por las subclases) para permitir la personalización del algoritmo en partes específicas. Promueve la reutilización del código y permite que las subclases tengan comportamientos específicos dentro de un esquema común. Esto evita la duplicación de código y facilita la extensibilidaddel diseño.

Método de la fábrica (*Factory Method*)

El patrón de diseño *Factory Method* es un patrón creacional que proporciona una interfaz para crear objetos, pero permite a las subclases decidir qué clase concreta instanciar. Este encapsula la creación de objetos y promueve la flexibilidad al permitir que las subclases tomen decisiones sobre la creación de objetos. En el patrón, se define un método abstracto en una clase base (conocido como el "*factory method*") que las subclases deben implementar. Este método es responsable de crear y devolver una instancia de una clase concreta, que se ajuste a una interfaz común. La idea principal del patrón es delegar la creación de objetos a las subclases, lo que permite extender o personalizar el proceso de creación sin modificar el código cliente. El cliente simplemente invoca el *factory method* de la clase base y recibe una instancia del objeto deseado sin tener conocimiento de la clase concreta que se está creando.

Fábrica abstracta (*Abstract Factory*)

El patrón de diseño *Abstract Factory* es un patrón creacional que proporciona una interfaz para crear familias de objetos relacionados sin especificar sus clases concretas. Este permite encapsular la creación de objetos y proporcionar una abstracción sobre las clases concretas involucradas. se define una interfaz abstracta (conocida como la fábrica abstracta) que declara los métodos para crear diferentes tipos de objetos. Luego, se implementan las clases concretas (conocidas como fábricas concretas) que implementan la interfaz abstracta y son responsables de crear objetos específicos de una familia. La idea principal del patrón es proporcionar un mecanismo para crear objetos relacionados sin acoplar el código cliente a clases concretas. Esto permite que el código cliente sea independiente de la implementación concreta y se pueda intercambiar fácilmente entre

diferentes familias de objetos (Gamma y col., 1994).

Repositorio (*Repository*)

El patrón de diseño *Repository* es un patrón de diseño arquitectónico que se utiliza en el desarrollo de software para gestionar el acceso y la persistencia de los datos. Proporciona una capa de abstracción entre la lógica de negocio de una aplicación y el almacenamiento de datos, permitiendo un acoplamiento más débil y una mayor flexibilidad. Se basa en la idea de tener una interfaz o clase abstracta llamada "*Repository*" que define métodos para acceder, crear, actualizar y eliminar objetos de un origen de datos, como una base de datos. Luego, se implementan clases concretas que utilizan tecnologías específicas para interactuar con el almacenamiento de datos, ya sea mediante consultas SQL, llamadas a API u otras formas de acceso. Proporciona una abstracción de alto nivel para el acceso a los datos, lo que permite a la lógica de negocio trabajar con entidades y objetos en lugar de tener que lidiar directamente con los detalles de almacenamiento y persistencia. Además, permite centralizar la lógica de acceso a los datos, lo que facilita su mantenimiento y evita laduplicación de código (Eisenberg, 2020).

2.9 Diseño de base de datos

Para el desarrollo del sistema se modeló el siguiente esquema de base de datos:

2.10 Tarjetas CRC

Las tarjetas CRC (Clase-Responsabilidad-Colaboradoras) son una técnica utilizada en la metodología XP para el análisis y diseño orientado a objetos. Se utilizan para identificar las clases en un sistema, así como sus responsabilidades y las colaboraciones con otras clases.
Cada tarjeta CRC representa una clase y se utiliza durante las sesiones de diseño colaborativo, donde los miembros del equipo escriben en la tarjeta el nombre de la clase, sus responsabilidades y las clases con las que interactúa. Esto permite una comprensión clara de las interacciones y responsabilidades de cada clase en el sistema (Cohn, 2004a).

- **Clase:** es cualquier persona, evento, concepto, pantalla o reporte,

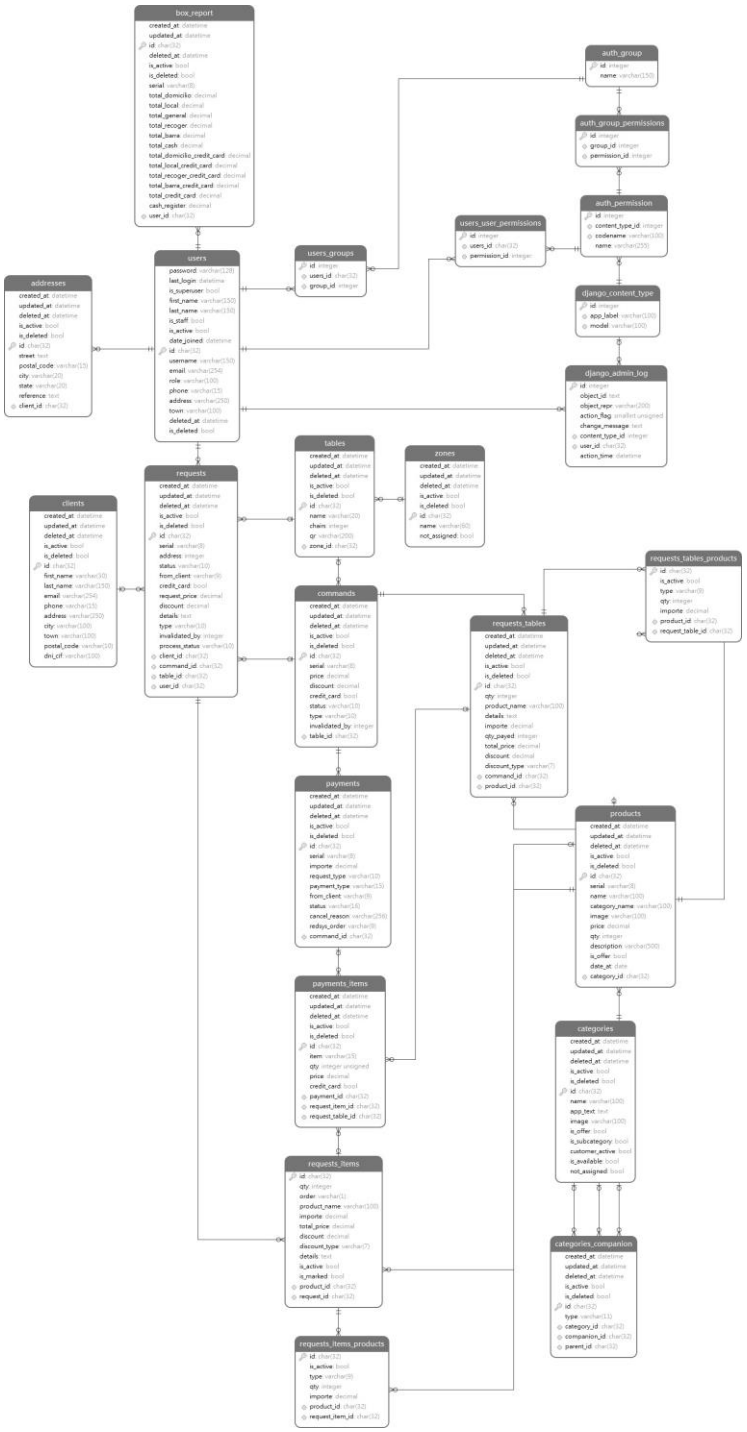

Figura 2.3: Modelo de datos

- **Responsabilidades:** las responsabilidades de una clase son las entidades que conoce y las que realizan sus atributos y métodos,

- **Colaboradoras:** las colaboradoras de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

A continuación se muestra unas de las tarjetas CRC obtenidas del sistema:

Nombre de Clase: ***Users***	
Superclase: *AbstractUser*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *ForeignKey*, *EmailField*, *CharField*, *DateTimeField*, *BooleanField*, *Meta*.

El resto de tarjetas CRC que definen el sistema están adjuntas en los anexos del documento.

2.11 Conclusiones del capítulo

En el presente capítulo se definió el modelo del negocio, el cual refleja el punto de partida de la solución propuesta. Se especificaron además los requisitos del sistema que permitieron identificar las funcionalidades con las que este contará y que darán respuesta a las necesidades del usuario. Se elaboraron los principales artefactos que guían la metodología de desarrollo, lo que permitió definir los aspectos necesarios para el desarrollo del sistema y se delimitó la arquitectura que tendrá el sistema guiado por el patrón arquitectónico MVC, así como los patrones de diseño los cuales posibilitaron definir una mejor arquitectura.

Capítulo 3: Implementación y pruebas

En el presente capítulo se abordan las Tareas de Ingeniería. Se presentan los diagramas de despliegue y de componentes y se confeccionan los casos de prueba y se muestran sus resultados para lavalidación de la misma.

3.1 Tarea de Ingeniería

Las Tareas de Ingeniería son actividades específicas que los equipos de desarrollo realizan para construir y entregar un software de alta calidad de manera iterativa e incremental. Deben ser estimables, su tiempo de implementación debe ser corto, aproximadamente entre uno y tres días, su objetivo es resolver las Historias de Usuario. Una Historia de Usuario puede tener una o varias Tareas de Ingeniería en dependencia de la funcionalidad a desarrollar (Winesett, 2010).
En la siguiente tabla se muestra el formato utilizado para la confección de las tareas de ingeniería:

Tabla 3.1: Ejemplo de Tarea de Ingeniería

Tarea de Ingeniería	
Número tarea: 1	**Número de Historia de Usuario:** 1
Nombre tarea: Diseño de base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.0
Fecha inicio: 29 de mayo del 2023	**Fecha fin:** 30 de mayo del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Realizar el diseño de la base de datos del sistema.	

Los campos de la tarjeta de las Tareas de Ingeniería reflejan lo siguiente:

- **Número tarea:** representa el número por el que se identifica a la tarea. Cada tarea tiene un único número que la identifica,

- **Número Historia de Usuario:** es el número de la Historia de Usuario a la que responde la tarea,
- **Nombre de tarea:** define el nombre o funcionalidad concreta a la que se dedica la tarea, debe estar expresado en forma infinitiva,
- **Tipo de tarea:** información del tipo de tarea a realizar, la misma puede ser:
  - **Desarrollo:** tarea que se realizará por primera vez,
  - **Corrección:** tarea que se realiza a partir de una anterior que no se realizó correctamente, es decir, que no pasó los casos de prueba satisfactoriamente,
  - **Mejora:** tarea que se realiza a partir de una anterior incorporándole nuevos requerimientos,
  - **Otra:** tarea que no corresponde con una de las anteriores, en este caso es necesario especificar el tipo de tarea o realizar una descripción más profunda de esta.
- **Puntos estimados:** tiempo de duración de la tarea. El tiempo estimado es reflejado en días. La suma de los puntos estimados de las tareas de ingeniería de una Historia de Usuario no puede superar la cantidad de puntos estimados definidos para la Historia de Usuario,
- **Fecha inicial:** fecha en la que se inicia el desarrollo de la tarea de ingeniería,
- **Fecha final:** fecha en la que se concluye el desarrollo de la tarea de ingeniería,
- **Programador responsable:** nombre del responsable de la realización de la tarea,
- **Descripción:** es una breve descripción sobre lo que la tarea debe hacer o resolver.

El resto de Tareas de Ingeniería que definen el sistema están adjuntas en los anexos del documento. Igualmente se encuentra el resumen de las tareas de ingenierías implementadas por cada Historia de Usuario.

3.2 Modelo de implementación

Un modelo de implementación es una representación conceptual de cómo se llevará a cabo la construcción y despliegue de un sistema de software. Se centra en proporcionar una visión clara de cómo se estructurará y desplegará el sistema en un entorno de producción. Puede incluir detalles sobre la infraestructura física necesaria, como servidores, redes, bases de datos y dispositivos de almacenamiento. Para su conformación se realizan dos diagramas fundamentales, el diagrama de componentes y el diagrama de despliegue.

3.2.1 Diagrama de componentes

El diagrama de componentes es un tipo de diagrama utilizado en ingeniería de software para representar la estructura y las interacciones entre los componentes de un sistema. Se utiliza para visualizar la arquitectura de software y la organización de los módulos o bloques funcionales que componen el sistema (OMG, 2017).

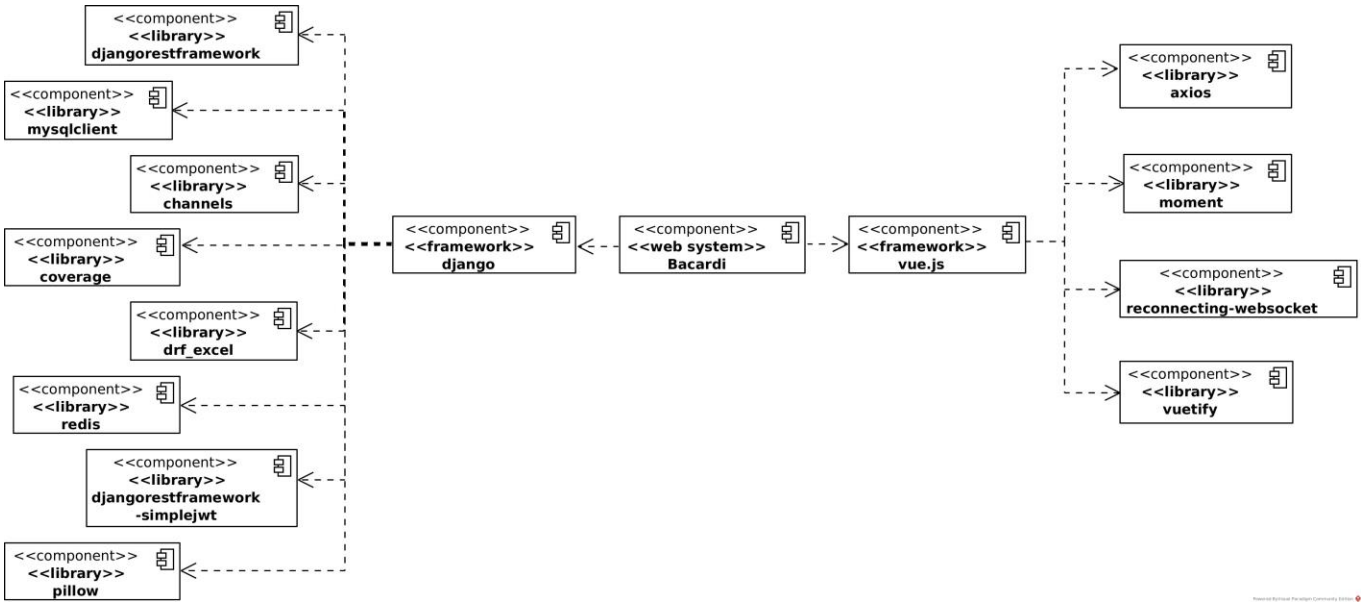

Figura 3.1: Diagrama de componentes del sistema Bacardí

- **Bacardi:** es la carpeta donde se empaqueta el sistema que se propone como solución de esta investigación,
- ***django**:* es el marco de trabajo utilizado para el lado del servidor,

- ***djangorestframework**:* biblioteca de código abierto para Django que permite la creación de APIs RESTful,
- ***mysqlclient**:* proporciona una interfaz para interactuar con una base de datos MySQL,
- ***channels**:* permite la creación de aplicaciones web en tiempo real,
- ***coverage**:* es una herramienta para medir la cobertura de código de programas Python,
- ***drf_excel**:* proporciona un renderizador de hojas de cálculo de Excel (xlsx) para Django REST Framework,
- ***redis**:* cliente de Python para almacenamiento de datos,
- ***djangorestframework-simplejwt**:* un complemento de autenticación basado en JSON Web Token para Django REST Framework,
- ***pillow**:* utilizado para el procesamiento de imágenes.

- ***vue.js**:* es el marco de trabajo utilizado para el lado del cliente,
  - ***axios**:* se utiliza para realizar peticiones HTTP al servidor,
  - ***moment**:* se utiliza para manipular y formatear fechas y horas,
  - ***reconnecting-websocket**:* proporciona una implementación de WebSocket que se autoreconecta en caso de desconexión,
  - ***vuetify**:* para la creación de componentes gráficos de material desing.

3.2.2 Diagrama de despliegue

El diagrama de despliegue en ingeniería de software es una representación visual que muestra la configuración física de un sistema y la distribución de sus componentes en entornos de hardware. Se utiliza para comprender cómo los componentes de un sistema se despliegan en diferentes nodos o dispositivos físicos, como servidores, computadoras O dispositivos móviles (). A continuación, se muestra la distribución de los nodos del sistema:
Descripción de los nodos:

- **Cliente:** el nodo representa un dispositivo cliente desde el cual se podrá acceder al sistema por medio de un navegador web e interactuar con todas las funcionalidades que este brinda,

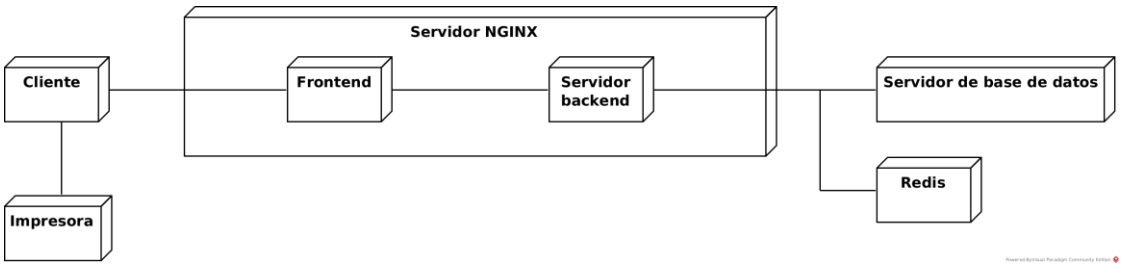

Figura 3.2: Diagrama de despliegue

- **Servidor NGINX:** el nodo representa el servidor web donde estarán alojados:
  - ***Frontend*:** servidor donde se encuentra la interfaz que interactúa con los usuarios,
  - **Servidor *Backend*:** servidor que se encarga de procesar toda la lógica que se encuentra en la página web,
- **Servidor de base de datos:** el nodo representa el servidor de base de datos *MySQL* en el que estará alojada la base de datos del sistema,
- **Redis:** el nodo representa un motor de base de datos en memoria,
- **Impresora:** el nodo representa la impresora a través de la cual se imprimen los reportes generados por el sistema.

Descripción de los protocolos utilizados:

- **HTTPS:** Protocolo de Transferencia de Hipertexto Seguro (*HyperText Transfer Protocol Secure*). Protocolo de comunicación que permite las transferencias de datos y recursos,
- **USB Tipo B:** Bus Universal en Serie Tipo B (*Universal Serial Bus Type B*). Es un cable USB que permite la transmisión de datos desde una computadora hacia la impresora y viceversa,
- **WIFI:** tecnología que permite la interconexión inalámbrica de dispositivos electrónicos.

3.3 Pruebas

La metodología XP se enfoca en la realización de pruebas durante todo el proceso de desarrollo de software, con el propósito de lograr un producto final de alta calidad, disminuyendo la cantidad

de errores no detectados y reduciendo el tiempo entre la identificación y corrección de errores. En este proceso, no solo participan los desarrolladores, sino también es fundamental la colaboración del cliente, especialmente en las pruebas de aceptación.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación son un proceso clave en el desarrollo de software que permite validar que un sistema o aplicación cumple con los requisitos y expectativas del usuario final. Estas pruebas son realizadas por el equipo de desarrollo o por los usuarios finales con el objetivo de asegurar que el software es funcional, confiable y cumple con los estándares de calidad. Las pruebas de aceptación son importantes para garantizar que el software cumpla con las expectativas del usuario final y para minimizar los problemas y errores que puedan surgir en la producción. Estas pruebas se deben realizar en una etapa posterior del ciclo de vida del software (Pressman y Maxim, 2015).

3.3.2 Casos de prueba

Los casos de prueba son evidencias de pruebas funcionales o unitarias que se realizan al sistema para comprobar su funcionamiento. Las pruebas funcionales son validaciones escritas desde la perspectiva del cliente, y las pruebas unitarias son validaciones desde la perspectiva del programador. El objetivo general es tener una forma para decirle al cliente que la Historia de Usuario está lista. Las pruebas funcionales o pruebas de aceptación, son las más importantes, ya que re- presentan la medida de satisfacción del cliente para una funcionalidad que el sistema debe tener. Los casos de pruebas fueron definidos para cada Historia de Usuario establecida. Se comprueba el funcionamiento de cada una de las funcionalidades implementadas que responden a la misma. El formato utilizado para la confección de casos de pruebas se muestra a continuación:

Tabla 3.2: Ejemplo de Caso de prueba de aceptación

Caso de prueba de aceptación	
Código: PA01_HU2	**Historia de Usuario:** HU_2
Nombre: Autenticar al sistema con errores I	

Descripción: Se intentará autenticar al sistema dejando los campos de correo y clave vacíos.
Condiciones de ejecución: El sistema no debe haber iniciado sesión o tenerla cerrada en el servidor.
Pasos de ejecución: Se ingresa a la pantalla de autenticación mediante la url en el navegador. Una vez que aparezca la vista inicial, se presiona el botón Aceptar del cuadro de diálogo con los campos correo y clave vacíos.
Resultados esperados: El sistema debe mostrar que el correo y la clave son requeridas.
Evaluación de la prueba: Satisfactoria

Campos del caso de prueba

- **Código:** identificador del caso de prueba. Dividido en dos partes. La primera representa la inicial del artefacto y la segunda representa el número con que se identifica la prueba,
- **Historia de Usuario:** es el número de la Historia de Usuario a la que responde el caso de prueba,
- **Descripción:** es una breve descripción del propósito de la prueba,
- **Condiciones de ejecución:** condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba,
- **Entradas / pasos de ejecución:** entradas o funciones que deben ejecutarse para realizar el caso de prueba,
- **Resultado esperado:** salida u objetivo que debe cumplir la funcionalidad a la que se le realiza el caso de prueba,
- **Evaluación:** evaluación de éxito del caso de prueba. Prueba satisfactoria en caso de éxito o prueba insatisfactoria en caso de fallo.

Los casos de prueba son agregados a los artefactos de entrega que se realiza al cliente al terminar cada fase o iteración del proyecto. Las Historias de Usuario con evaluación insatisfactoria, serán corregidas en la próxima iteración a partir de nuevas tareas de ingeniería. Los casos de pruebas realizados al sistema se encuentran adjuntos en los anexos del documento.

3.3.3 Pruebas de compatibilidad

Las pruebas de compatibilidad en ingeniería de software se enfocan en asegurar que un sistema de software sea compatible con diferentes configuraciones de hardware, sistemas operativos, navegadores web, etc. Estas pruebas verifican que el software funcione correctamente en los entornos previstos y no fallen o tengan comportamientos inesperados. Algunos enfoques comunes son probar en múltiples navegadores y sistemas operativos, probar interfaces de programación de aplicaciones (APIs) o probar compatibilidad hacia atrás con versiones anteriores (González-Herrera y col., 2022). Para realizar dichas pruebas existen diversas técnicas ya definidas, entre estas se encuentran:

- **Verificación del cumplimiento de estándares:** consiste en analizar los componentes gráficos del sitio web en diferentes navegadores, para verificar que sigan los lineamientos y especificaciones de los estándares (Mascheroni, Cogliolo e Irrazabal, 2016),
- **Pruebas de interfaz de usuario:** es la más común de las técnicas. Puede ser realizada de forma manual o con software especializado (pruebas automatizadas). Su objetivo es revisar el contenido visual del sitio web a través de la navegación de sus páginas en los diferentes navegadores (Memon, Banerjee y Nagarajan, 2003), (Xie y Memon, 2007),
- **Análisis del modelo de objetos del documento (*Document Object Model o DOM*):** es una técnica dinámica que consiste en comparar el comportamiento de una aplicación web en diferentes navegadores, identificando las diferencias como defectos (Choudhary, Versee y Orso, 2010),
- **Comparación de imágenes:** se basa en tomar una captura de pantalla del sitio en un tipo de navegador, y compararla con otra captura del sitio en otro navegador diferente del primero. Si ambas imágenes coinciden, entonces el sitio será compatible entre ambos navegadores (Hori y col., 2015).

3.3.4 Pruebas de usabilidad

Las pruebas de usabilidad son una técnica utilizada para evaluar la facilidad de uso y la eficacia de un producto o sistema por parte de los usuarios finales. El objetivo de estas pruebas es identificar problemas o áreas de mejora en la experiencia del usuario y, por lo tanto, mejorar la calidad del producto o sistema.

Una de las pruebas de usabilidad más frecuentemente realizada, son las *heurísticas de Nielsen* (Nielsen, 1995). Nielsen, tras analizar varios artículos y libros plantea diez heurísticas que permiten evaluar la usabilidad de un producto. Dichas heurísticas plantean:

1. **Visibilidad del estado del sistema:** el sistema siempre debe mantener a los usuarios informados sobre lo que está pasando, a través de una retroalimentación adecuada dentro de un tiempo razonable,

2. **Correspondencia entre el sistema y el mundo real:** el sistema debe hablar el idioma de los usuarios, con palabras, frases y conceptos familiares para el usuario, en lugar de términos orientados al sistema. Haciendo que la información aparezca en un orden natural y lógico,

3. **Control del usuario y libertad:** los usuarios suelen elegir funciones del sistema por error y necesitarán una **salida de emergencia** claramente marcada para dejar el estado no deseado sin tener que pasar por un diálogo extendido. La posibilidad de deshacer y rehacer acciones,

4. **Consistencia y estándares:** los usuarios no deberían preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Siga las convenciones de la plataforma,

5. **Prevención de errores:** incluso mejor que los buenos mensajes de error, es un diseño cuidadoso que impide que se produzca un problema en primer lugar. Elimine las condiciones propensas a errores o compruébelo y presente a los usuarios una opción de confirmación antes de comprometerse con la acción,

6. **Reconocimiento en lugar de recordar:** minimice la carga de memoria del usuario haciendo visibles los objetos, las acciones y las opciones. El usuario no debe tener que recordar la información de una parte del diálogo a otra. Las instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado,

7. **Flexibilidad y eficiencia de uso:** los aceleradores no vistos por el usuario principiante pueden a menudo acelerar la interacción para el usuario experto, de manera que el sistema pueda atender a usuarios inexpertos y experimentados. Permite a los usuarios adaptar acciones frecuentes,

8. **Diseño estético y minimalista:** los diálogos no deben contener información irrelevante o raramente necesaria. Cada unidad extra de información compite con las unidades relevantes de información y disminuye la visibilidad relativa,

9. **Ayuda a los usuarios a reconocer, diagnosticar y recuperar errores:** los mensajes de error deben expresarse en lenguaje sencillo (sin códigos), indicar con precisión el problema y sugerir constructivamente una solución,

10. **Ayuda y documentación:** a pesar de que es mejor si el sistema puede utilizarse sin documentación, puede ser necesario proporcionar ayuda y documentación. Cualquier información de este tipo debe ser fácil de buscar, centrada en la tarea del usuario, enumerar los pasosconcretos que se deben llevar a cabo, y no ser demasiado grande.

3.4 Resultados de las pruebas

3.4.1 Pruebas de aceptación

Como resultado de las pruebas de aceptación se detectaron un total de 32 no conformidades. A medida que se avanzó en las iteraciones se disminuyeron los números de no conformidades hasta ser cero. De esta manera el sistema queda listo para ser utilizado. En la siguiente figura se resumen las no conformidades detectadas en cada una de las iteraciones:

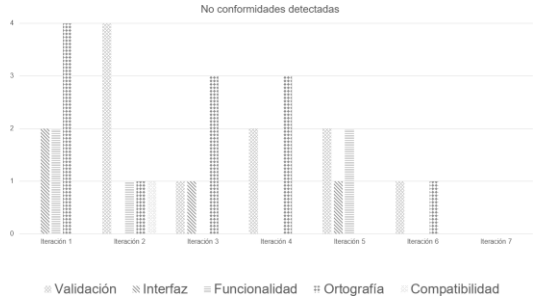

Figura 3.3: Resultados de las no conformidades detectadas

3.4.2 Pruebas de compatibilidad

El sistema fue ejecutado y probado en diferentes navegadores webs como Opera, Firefox, Chrome y Edge, en sus diferentes versiones, obteniéndose resultados positivos. Ni el diseño ni el rendimiento de la aplicación se ven afectados en ninguno de los navegadores. Tampoco se vieron afectadaslas funcionalidades del sistema.

3.4.3 Pruebas de usabilidad

Para llegar a uno o varios resultados en las pruebas de usuabilidad, se debe partir del cumplimiento en el sistema de las diez heurísticas que permiten evaluar la usabilidad de un producto según Nielsen:

1. Este primer aspecto es cumplido por el sistema el cual muestra en todo momento en las tablas los listados de los diferentes datos actualizados, igualmente se le implementó *WebSocket* para el manejo de información en tiempo real,

2. Los mensajes de error aparecen en el momento que el usuario realiza alguna operación, estos son concisos y no se emplean palabras técnicas o ambiguas que puedan confundir al usuario final,

3. En el sistema existen botones para cancelar o cerrar las diferentes operaciones, un botón atrás, un panel de navegación, con lo cual, si el usuario quiere deshacer su acción puede hacerlo,

4. El sistema utiliza componentes estándares a los que los usuarios ya están acostumbrados al interactuar con ellos y a su vez estos responden de manera intuitiva a posibles interacciones.

5. El sistema utiliza cuadros de diálogos en los que el usuario tiene que ingresar datos en varios campos de texto, los cuales pueden contener errores y el sistema debe ser capaz de señalarlos. Para evitar esto, en el momento donde el usuario envíe los datos al servidor, si estos son incorrectos, serán señalados en dicho cuadro de diálogo,

6. El sistema cuenta con varios componentes interactivos, todos están visibles y situados de

forma que se minimice el riesgo de que el usuario pueda realizar acciones erróneas accidentalmente,

7. Este aspecto no se aprecia, ya que el sistema no hace uso de aceleradores, accesos directos o atajos del teclado,

8. El diseño de la aplicación está basado en las peticiones del cliente, garantizando de esta forma uniformidad y ajuste al negocio. En cuanto a la estética se tuvo en cuenta los tamaños de las fuentes utilizadas en correspondencia con la función que el texto desempeña, los controles alineados y uniformes que se adapten de igual forma para cualquier navegador,

9. Los mensajes de error son precisos sin llegar a una formalidad que desinterese a los usuarios, pero tampoco con una informalidad que los incomode; e indicando qué hacer para solucionar el problema,

10. Se ofrece un manual de usuario final que servirá de guía para el trabajo en el sistema.

3.5 Conclusiones del capítulo

En el presente capítulo se definieron todas la tareas de ingenierías para solucionar cada una de las Historias de Usuario definidas. Se determinó la manera en que se implantará el sistema propuesto en un ambiente real. Se definieron las pruebas a que se someterá el software, según la metodología de software escogida en conjunto con otros tipos de pruebas para la validación del sistema desarrollado y gracias a estas se detectaron y se solucionaron las fallas existentes. Por tanto, el sistemaestá listo para ser desplegado y utilizado por el Bar Bacardí.

Conclusiones

Con la realización de la presente investigación se establecieron los fundamentos teórico-metodológicos de los principales elementos que componen la gestión y el análisis de ventas para el bar Bacardí.

Se definió la metodología de desarrollo ágil XP para el desarrollo del sistema. El estudio de las bases teóricas permitió, además, la elección de las herramientas adecuadas para el desarrollo de este tipo de sistemas, se especificaron los requisitos funcionales y no funcionales, lo que permitió identificar las funcionalidades que dan respuesta a las necesidades del usuario.

Se implementó el sistema web Análisis-Bacardí según los presupuestos y requisitos previamente definidos, haciendo uso de herramientas de software libre y código abierto.

Se validó su funcionamiento mediante la realización de las pruebas necesarias, según la metodología XP y posteriormente se realizó el análisis de los resultados que arrojaron las mismas.

Recomendaciones

Dentro de las posibles mejoras y recomendaciones que se le hacen al sistema desarrollado están:

1. Continuar con la investigación con el objetivo de perfeccionar o adicionar nuevas funcionalidades al sistema Análisis-Bacardí.

Referencias Bibliográficas

ArgoUML (2023). *ArgoUML: Open source UML modeling tool*. Consultado: 2023-05-15. URL: http://argouml.org/.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Addison-Wesley Professional.

Beck, K. (2004). *Extreme Programming Explained: Embrace Change*. 2nd. Addison-Wesley Professional.

Budgen, D. (2013). *Software design*. Pearson Education.

Buschmann, F. y col. (2020). *Pattern-Oriented Software Architecture: A System of Patterns*. Hoboken, NJ: John Wiley y Sons. ISBN: 978-0471958697.

Capterra (2023). *Software para pronóstico de ventas*. Consultado: 2023-03-23. URL: https://www.capterra.es/directory/30930/sales-forecasting/software.

Cardozzo, D. e I. Academy (2016). *Desarrollo de Software: Requisitos, Estimaciones y Análisis. 2 Edición*. CreateSpace Independent Publishing Platform. ISBN: 9781530088614.

Choudhary, S. R., H. Versee y A. Orso (2010). "WEBDIFF: Automated identification of cross-browser issues in web applications". En: *2010 IEEE International Conference on Software Maintenance*. IEEE, págs. 1-10.

Cohn, M. (2004b). *User stories applied: For agile software development*. Addison-Wesley Professional.

Cohn, M. (2004a). "An Introduction to CRC Cards". En: *Better Software* 6.1, págs. 19-22. URL: https://www.mountaingoatsoftware.com/papers/crc-cards.pdf (visitado 21-05-2023).

Connolly, T. M. y C. E. Begg (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th. Pearson. ISBN: 978-0-273-79364-4.

Consortium, S. (2023). *SQLite*. Consultado: 2023-05-17. URL: https://www.sqlite.org/index.html.

Corporation, O. (2023). *MySQL*. Consultado: 2023-05-17. URL: https://www.mysql.com/.

Docs, M. W. (2023a). *CSS (Cascading Style Sheets)*. Consultado: 2023-05-17. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

Docs, M. W. (2023b). *HTML (HyperText Markup Language)*. Consultado: 2023-05-17. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

Docs, M. W. (2023c). *JavaScript*. Consultado: 2023-05-17. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

Eisenberg, J. (2020). "Repository Design Pattern". En: *Microsoft Docs*. URL: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design (visitado 20-05-2023).

Equipo Ekon (2020). *Diagrama de procesos y su importancia para tu empresa*. URL: https://www.ekon.es/blog/diagrama-procesos-empresa/.

Foundation, P. S. (2023a). *What is Python? Executive Summary*. Consultado: 2023-05-17. URL: https://www.python.org/doc/essays/blurb/.

Foundation, T. D. S. (2023b). *Django Web Framework*. Consultado: 2023-05-15. URL: https://www.djangoproject.com/.

Fowler, M. y R. Parsons (2019). *Domain-specific languages*. Addison-Wesley Professional.

Freeman, A. (2020). *Programming TypeScript: Making Your JavaScript Applications Scale*. O'Reilly Media. ISBN: 978-1-492-05103-2.

Gamma, E. y col. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley Professional. ISBN: 978-0201633610.

Gammack, J. y D. Kopec (2017). *Learning MVC Architecture with ASP.NET Core 2*. Birmingham, UK: Packt Publishing. ISBN: 978-1786465354.

García Navarro, J. (2018). "Estudio comparativo de metodologías, herramientas y wiki de soporte para la gestión de proyectos de desarrollo software". En.

GetApp (2023). *Monday sales CRM*. Consultado: 2023-03-23. URL: https://www.getapp.es/software/2056613/monday-sales-crm.

Gómez García, D. E. (2018). “Desarrollo del sistema de requisiciones para la empresa hidroeléctrica Abanico SA Aplicando el entorno de programación Node. js”. B.S. thesis. Escuela Superior Politécnica de Chimborazo.

González-Herrera, I. y col. (2022). “Software compatibility testing: a systematic mapping study”. En: *Information and Software Technology* 144, pág. 106760.

Grinberg, M. (2018). *Flask Web Development with Python Tutorial*. O’Reilly Media. ISBN: 978-1449372620.

Group, P. G. D. (2023). *PostgreSQL: The world’s most advanced open source database*. Consultado: 2023-05-17. URL: https://www.postgresql.org/.

Hori, A. y col. (2015). “An Oracle based on Image Comparison for Regression Testing of Web Applications.” En: *SEKE*, págs. 639-645.

IBM (2023). *IBM Rational Rose Enterprise*. Consultado: 2023-05-15. URL: https://www.ibm.com/products/rational-rose-enterprise.

Jeffries, R., A. Anderson y C. Hendrickson (2001). *Extreme programming installed*. Addison-Wesley Professional.

Jetbrains (2023a). *Integrated Development Environment (IDE)*. Consultado: 2023-05-17. URL: https://www.jetbrains.com/lp/definition-of-ide/.

Jetbrains (2023b). *PyCharm Features*. Consultado: 2023-05-17. URL: https://www.jetbrains.com/pycharm/features/.

Jiménez, J. (2016). *UF2406 - El cliclo de vida del desarrollo de aplicaciones*. Editorial Elearning, S.L.

Jiménez Ortega, R. (2017). *Curso de HTML5 desde cero (HTML5, CSS3, JavaScript)*.

Joskowicz, J. (2008). “Reglas y prácticas en eXtreme Programming”. En: *Universidad de Vigo* 22.

Kotler, P. y G. Armstrong (2018). *Fundamentos de marketing*. Pearson Educación.

Kotler, P. y K. L. Keller (2016). *Dirección de marketing*. Pearson educación.

Larman, C. (2003). *Agile and iterative development: A manager’s guide*. Pearson Education.

Looker (2021). *Looker Studio*. Consultado: 2023-03-23. URL: https://www.looker.com/products/looker-studio.

Mascheroni, M. A., M. K. Cogliolo y E. Irrazabal (2016). "Automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de software". En: *Simposio Argentino de Ingeniería de Software (ASSE 2016)-JAIIO 45 (Tres de Febrero, 2016)*.

Memon, A., I. Banerjee y A. Nagarajan (2003). "GUI ripping: Reverse engineering of graphical user interfaces for testing". En: *10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings.* Citeseer, págs. 260-269.

Mikowski, T. y J. Powell (2013). *Single Page Web Applications: JavaScript end-to-end*. Manning Publications. ISBN: 978-1617290756.

Mozilla (2023). *How does the Internet work?* Consultado: 2023-05-17. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

Nielsen, J. (1995). "10 usability heuristics for user interface design". En: *Nielsen Norman Group* 1.1.

OMG (2017). *Unified Modeling Language (UML) - Version 2.5.1*. Object Management Group. URL: https://www.omg.org/spec/UML/2.5.1 (visitado 21-05-2023).

Pallets (2023). *Flask*. Consultado: 2023-05-16. URL: https://flask.palletsprojects.com/.

Paradigm, V. (2023a). *Visual Paradigm*. Consultado: 2023-05-15. URL: https://www.visual-paradigm.com/.

Paradigm, V. (2023b). *Visual Paradigm Product Overview*. Consultado: 2023-03-24. URL: https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

Power, B. y col. (2021). "Microsoft power bi". En: *Available here: https://powerbi. microsoft. com/en-us* 130.

Pressman, R. (2015). *Ingeniería del software: un enfoque práctico*. McGraw-Hill.

Pressman, R. S. y B. R. Maxim (2015). *Software engineering: A practitioner's approach*. New York: McGraw-Hill.

Sebesta, R. W. (2020). *Concepts of Programming Languages*. 12th. Pearson. ISBN: 978-0134984163.

Sommerville, I. (2016). *Software Engineering*. 10th. Harlow, UK: Pearson. ISBN: 978-0133943030.

Sutherland, J. y K. Schwaber (2020). *The Scrum Guide™. The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org.

Team, T. A. (2023a). *Angular*. Consultado: 2023-05-16. URL: https://angular.io/.

Team, T. P. (2023b). *Welcome to Pyramid, a Python Web Framework*. Consultado: 2023-05-16. URL: https://trypyramid.com/.

Team, T. R. (2023c). *React*. Consultado: 2023-05-16. URL: https://reactjs.org/.

Team, T. V. (2023d). *Vue.js*. Consultado: 2023-05-16. URL: https://vuejs.org/.

Thies, F. y R. Fuhrmannek (2019). *Software Architecture for Developers*. Heidelberg, Germany: dpunkt.verlag. ISBN: 978-3864906468.

Valladarez, B. S. M. M., B. M. E. Gaitan y B. N. N. P. Reyes (2016). "Universidad Nacional Autonoma de Nicaragua, Managua UNAN-MANAGUA Recinto Universitario Rubén Darío (RURD) Facultad de Ciencias e Ingeniería Departamento de Computación". En.

Winesett, J. (2010). *Agile web application development with Yii1. 1 and PHP5*. Packt Publishing Ltd.

Xie, Q. y A. M. Memon (2007). "Designing and comparing automated test oracles for GUI-based software applications". En: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 16.1, 4-es.

Anexo A: Glosario de términos

App: una *App* (Aplicación) es un conjunto portable de una funcionalidad de Django que incluye modelos, vistas, formularios y otros ficheros que conviven en un solo paquete de Python.

API: una *API* (Application Programming Interface) es un conjunto de reglas, protocolos y herramientas que se utilizan para crear software y aplicaciones. En términos simples, una API es un conjunto de definiciones y protocolos que se utilizan para comunicarse con una aplicación o servicio web.

Anexo B: Historias de Usuarios

Descripción de las Historias de Usuarios definidas para el sistema.

HISTORIA DE USO			
Orden	HU_1	**Nombre**	Diseño y creación de la base de datos
Riesgo	Alto	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	Se realizará el diseño de la base de datos y será definida por medio de de migraciones con el ORM del marco de trabajo.		
Observación	El modelo de base de datos será escrito en el idioma inglés. El nombre de las tablas será en plural y el nombre de los atributos en singular		

HISTORIA DE USO			
Orden	HU_2	**Nombre**	Sistema de autenticación
Riesgo	Bajo	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	El sistema debe brindar la funcionalidad de acceso por medio de correo electrónico y clave de acceso. Las claves de acceso podrán ser modificadas solamente por el administrador del sistema. El sistema deberá permitir al usuario cerrar su sesión una vez el mismo lo decida. Si el usuario olvida su clave de acceso se le generará una de forma automática y se le enviará a su correo electrónico.		
Observación	El sistema no debe permitir autenticar o cerrar sesión al usuario más una vez. El sistema debe permitir eliminar y restaurar todos los usuarios. El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_3	**Nombre**	Gestionar usuarios
Riesgo	Bajo	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	El sistema debe brindar las funcionalidades de listar los usuarios, además, adicionar, actualizar y eliminar un usuario en el sistema, todo esto a través del panel de administración del sistema. Se realizará un sistema para la búsqueda de usuarios por nombre de usuario y correo electrónico.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_4	**Nombre**	Gestionar zonas
Riesgo	Bajo	**Prioridad**	Media
Iteración	2	**Puntos estimados**	1
Descripción	El sistema debe brindar las funcionalidades de listar las zonas, además, adicionar, actualizar y eliminar una zona en el sistema. Las zonas tendrán el nombre y si están asignadas o no.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_5	**Nombre**	Gestionar mesas
Riesgo	Bajo	**Prioridad**	Alta
Iteración	2	**Puntos estimados**	1
Descripción	El sistema debe brindar las funcionalidades de listar las mesas, además, adicionar, actualizar y eliminar una mesa en el sistema. Las mesas tendrán el nombre, la cantidad de sillas, un código qr y la zona a la que pertenecen. El sistema debe brindar una funcionalidad que permita mostrar las mesas según su zona.		

Observación	El listado debe contener una paginación de registros.

HISTORIA DE USO			
Orden	HU_6	**Nombre**	Gestionar clientes
Riesgo	Bajo	**Prioridad**	Alta
Iteración	3	**Puntos estimados**	1
Descripción	El sistema debe brindar las funcionalidades de listar los clientes, además, adicionar, actualizar y eliminar un cliente en el sistema. Los clientes tendrán el nombre y apellidos, correo, teléfono, dirección, provincia y municipio, código postal y carnet de identidad.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_7	**Nombre**	Gestionar categorías
Riesgo	Bajo	**Prioridad**	Alta
Iteración	3	**Puntos estimados**	2
Descripción	El sistema debe brindar las funcionalidades de listar las categorías, además, adicionar, actualizar y eliminar una categoría en el sistema. Las categorías tendrán nombre, descripción, imagen, compañera o agregado, si es oferta, si es subcategoría, si es visible para los clientes y si está disponible.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_8	**Nombre**	Gestionar productos
Riesgo	Bajo	**Prioridad**	Alta
Iteración	4	**Puntos estimados**	2

Descripción	El sistema debe brindar las funcionalidades de listar los productos, además, adicionar, actualizar y eliminar un producto en el sistema. Los productos tendrán número de serie, nombre, descripción, imagen y precio, categoría, cantidad, si es oferta y su fecha, alérgenos e ingredientes.
Observación	El listado debe contener una paginación de registros.

HISTORIA DE USO			
Orden	HU_9	**Nombre**	Gestionar pedidos
Riesgo	Alto	**Prioridad**	Alta
Iteración	5	**Puntos estimados**	2
Descripción	El sistema debe brindar las funcionalidades de listar los pedidos, además, adicionar, actualizar su estado y visualizar los detalles del pedido. El pedido tendrá un número de serie, la mesa, la comanda, los productos, el cliente, dirección, estado, si se usa tarjeta de crédito, precio, descuento, descripción y tipo.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_10	**Nombre**	Gestionar comandas
Riesgo	Medio	**Prioridad**	Alta
Iteración	5	**Puntos estimados**	2
Descripción	El sistema debe brindar las funcionalidades de listar las comandas, además, cambiar su estado, su mesa, invalidarlas y cerrar la comanda. Las comandas tendrán número de serie, mesa, precio, descuento, si se usa tarjeta de crédito, estado, tipo, por quien se invalidó y los productos.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_11	**Nombre**	Gestionar pagos

Riesgo	Alto	**Prioridad**	Alta
Iteración	6	**Puntos estimados**	2
Descripción	El sistema debe brindar las funcionalidades de listar los pagos, además, de adicionar y cerrar el pago. Los pagos tendrán un número de serie, la comanda, importe, tipo de orden y tipo de pago.		
Observación			

HISTORIA DE USO			
Orden	HU_12	**Nombre**	Generar reportes
Riesgo	Alto	**Prioridad**	Alta
Iteración	6	**Puntos estimados**	2
Descripción	El sistema debe brindar las funcionalidades de generar el reporte de cierre de caja, reporte de ventas y un reporte de historial de pedidos.		
Observación			

HISTORIA DE USO			
Orden	HU_13	**Nombre**	Generar reportes estadísticos
Riesgo	Alto	**Prioridad**	Alta
Iteración	7	**Puntos estimados**	3
Descripción	El sistema debe brindar las funcionalidades de generar los siguientes reportes estadísticos: ▪ Reporte de predicción de ventas, ▪ Reporte de predicción de ventas por productos, ▪ Reporte de compras asociativas entre productos, ▪ Reporte de probabilidad de compra de productos cuando se ha comprado otro.		

Observación	

Anexo C: Tarjetas CRC

Nombre de Clase: ***Time***	
Superclase: *Model*	**Subclases:** *Categories, CategoriesCompanion, Clients, Commands, Requests, Payments, Products, Tables, Zones*
Responsabilidades	**Colaboradoras**
Clase abstracta para generalizar los atributos comunes en los modelos del sistema.	*DateTimeField, BooleanField, Meta*

Nombre de Clase: ***Users***	
Superclase: *AbstractUser*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField, ForeignKey, EmailField, CharField, DateTimeField, BooleanField, Meta*.

Nombre de Clase: ***Categories***	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField, ForeignKey, TextField, CharField, ImageField, ManyToManyField, BooleanField, Meta*.

Nombre de Clase: ***CategoriesCompanion***

Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *ForeignKey*, *CharField*, *Categories*, *Meta*.

Nombre de Clase: ***Clients***	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *CharField*, *EmailField*, *Meta*.

Nombre de Clase: ***Ingredients***	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *CharField*, *ImageField*, *DecimalField*, *Meta*.

Nombre de Clase: ***Requests***	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *CharField*, *ForeignKey*, *ManyToManyField*, *IntegerField*, *BooleanField*, *DecimalField*, *TextField*, *Products*, *Commands*, *Tables*, *Meta*.

Nombre de Clase: *RequestsItems*	
Superclase: *Model*	**Subclases:**
Responsabilidades	**Colaboradoras**
Clase modelo que agrupa los productos del pedido.	*UUIDField, CharField, ForeignKey, IntegerField, DecimalField, TextField, BooleanField, Products, Requests, Decimal, int, Meta*.

Nombre de Clase: *RequestsItemsProducts*	
Superclase: *Model*	**Subclases:**
Responsabilidades	**Colaboradoras**
Clase modelo que agrupa los productos de varios pedidos.	*UUIDField, CharField, ForeignKey, IntegerField, DecimalField, BooleanField, Products, RequestsItems, Decimal, Meta*.

Nombre de Clase: *Payments*	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField, ForeignKey, CharField, DecimalField, Commands, Meta*.

Nombre de Clase: *Products*	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**

Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *ForeignKey*, *CharField*, *ImageField*, *DecimalField*, *IntegerField*, *BooleanField*, *DateField*, *ManyToManyField*, *Ingredients*, *Categories*, *Meta*.

Nombre de Clase: *Recipes*	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *ForeignKey*, *IntegerField*, *Ingredients*, *Products*, *Meta*.

Nombre de Clase: *BoxReport*	
Superclase: *Model*	**Subclases:**
Responsabilidades	**Colaboradoras**
Clase encargada de generar el reporte general.	*UUIDField*, *DateTimeField*, *BooleanField*, *CharField*, *DecimalField*, *ForeignKey*, *Users*, *Meta*.

Nombre de Clase: *Tables*	
Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *ForeignKey*, *CharField*, *IntegerField*, *Zones*, *Meta*.

Nombre de Clase: *Zones*

Superclase: *Time*	**Subclases:**
Responsabilidades	**Colaboradoras**
Representar conceptualmente las tuplas almacenadas en la tabla con igual nombre en la base de datos.	*UUIDField*, *CharField*, *BooleanField*, *Meta*.

Anexo D: Tareas de Ingeniería

A continuación la descripción de cada una de las Tareas de Ingeniería definidas para el sistema.

Tarea de Ingeniería	
Número tarea: 1	**Número de Historia de Usuario:** 1
Nombre tarea: Diseño de base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.0
Fecha inicio: 29 de mayo del 2023	**Fecha fin:** 30 de mayo del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Realizar el diseño de la base de datos del sistema.	

Tarea de Ingeniería	
Número tarea: 2	**Número de Historia de Usuario:** 1
Nombre tarea: Implementación de la base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3.0
Fecha inicio: 31 de mayo del 2023	**Fecha fin:** 2 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la base de datos diseñada con el ORM del marco de trabajo.	

Tarea de Ingeniería	
Número tarea: 3	**Número de Historia de Usuario:** 2
Nombre tarea: Iniciar sesión	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 5 de junio del 2023	**Fecha fin:** 5 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	

Descripción: Implementar un mecanismo que permita al usuario autenticarse en el sistema.

Tarea de Ingeniería	
Número tarea: 4	**Número de Historia de Usuario:** 2
Nombre tarea: Cambiar clave de acceso	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 6 de junio del 2023	**Fecha fin:** 6 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita al usuario cambiar su clave de acceso al sistema.	

Tarea de Ingeniería	
Número tarea: 5	**Número de Historia de Usuario:** 2
Nombre tarea: Cerrar sesión	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 6 de junio del 2023	**Fecha fin:** 6 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita al usuario cerrar la sesión en el sistema.	

Tarea de Ingeniería	
Número tarea: 6	**Número de Historia de Usuario:** 2
Nombre tarea: Diseño de la iconografía referente a la HU_2	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 7 de junio del 2023	**Fecha fin:** 7 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_2.	

Tarea de Ingeniería	
Número tarea: 7	**Número de Historia de Usuario:** 2
Nombre tarea: Diseño visual de la interfaz de autenticación	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 8 de junio del 2023	**Fecha fin:** 8 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden el sistema de autenticación.	

Tarea de Ingeniería	
Número tarea: 8	**Número de Historia de Usuario:** 2
Nombre tarea: Implementación de la interfaz de autenticación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha inicio: 9 de junio del 2023	**Fecha fin:** 9 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran el sistema de autenticación.	

Tarea de Ingeniería	
Número tarea: 9	**Número de Historia de Usuario:** 3
Nombre tarea: Listar usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 12 de junio del 2023	**Fecha fin:** 12 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar los usuarios existentes en la base de datos desde el panel de administración.	

Tarea de Ingeniería	
Número tarea: 10	**Número de Historia de Usuario:** 3

Nombre tarea: Adicionar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 12 de junio del 2023	**Fecha fin:** 12 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar un usuario en la base de datos desde el panel de administración.	

Tarea de Ingeniería	
Número tarea: 11	**Número de Historia de Usuario:** 3
Nombre tarea: Modificar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 13 de junio del 2023	**Fecha fin:** 13 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar los usuarios existentes en la base de datos desde el panel de administración.	

Tarea de Ingeniería	
Número tarea: 12	**Número de Historia de Usuario:** 3
Nombre tarea: Eliminar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 13 de junio del 2023	**Fecha fin:** 13 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita eliminar los usuarios existentes en la base de datos desde el panel de administración.	

Tarea de Ingeniería	
Número tarea: 13	**Número de Historia de Usuario:** 3
Nombre tarea: Diseño de la iconografía referente a la HU_3	

Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 14 de junio del 2023	**Fecha fin:** 14 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_3.	

Tarea de Ingeniería	
Número tarea: 14	**Número de Historia de Usuario:** 3
Nombre tarea: Diseño visual de la gestión de usuarios	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 15 de junio del 2023	**Fecha fin:** 15 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de usuarios en el sistema.	

Tarea de Ingeniería	
Número tarea: 15	**Número de Historia de Usuario:** 3
Nombre tarea: Implementación de la interfaz de gestión de usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha inicio: 16 de junio del 2023	**Fecha fin:** 16 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de usuarios en el sistema.	

Tarea de Ingeniería	
Número tarea: 16	**Número de Historia de Usuario:** 4
Nombre tarea: Listar zonas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 26 de junio del 2023	**Fecha fin:** 26 de junio del 2023

Programador responsable: Omar Sarmiento Rolo
Descripción: Implementar un mecanismo que permita listar las zonas existentes en la base de datos.

Tarea de Ingeniería	
Número tarea: 17	**Número de Historia de Usuario:** 4
Nombre tarea: Adicionar zona	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 26 de junio del 2023	**Fecha fin:** 26 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar una zona a la base de datos.	

Tarea de Ingeniería	
Número tarea: 18	**Número de Historia de Usuario:** 4
Nombre tarea: Modificar zona	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 27 de junio del 2023	**Fecha fin:** 27 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar las zonas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 19	**Número de Historia de Usuario:** 4
Nombre tarea: Eliminar zona	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 27 de junio del 2023	**Fecha fin:** 27 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	

Descripción: Implementar un mecanismo que permita eliminar las zonas existentes en la base de datos.

Tarea de Ingeniería	
Número tarea: 20	**Número de Historia de Usuario:** 4
Nombre tarea: Diseño de la iconografía referente a la HU_4	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 28 de junio del 2023	**Fecha fin:** 28 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_4.	

Tarea de Ingeniería	
Número tarea: 21	**Número de Historia de Usuario:** 4
Nombre tarea: Diseño visual de la gestión de zonas	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 29 de junio del 2023	**Fecha fin:** 29 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de zonas en el sistema.	

Tarea de Ingeniería	
Número tarea: 22	**Número de Historia de Usuario:** 4
Nombre tarea: Implementación de la interfaz de gestión de zonas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha inicio: 30 de junio del 2023	**Fecha fin:** 30 de junio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de usuarios en el sistema.	

Tarea de Ingeniería	
Número tarea: 23	**Número de Historia de Usuario:** 5
Nombre tarea: Listar mesas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 3 de julio del 2023	**Fecha fin:** 3 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar las mesas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 24	**Número de Historia de Usuario:** 5
Nombre tarea: Adicionar mesa	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 3 de julio del 2023	**Fecha fin:** 3 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar una mesa a la base de datos.	

Tarea de Ingeniería	
Número tarea: 25	**Número de Historia de Usuario:** 5
Nombre tarea: Modificar mesas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 3 de julio del 2023	**Fecha fin:** 4 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar las mesas existentes en la base de datos.	

Tarea de Ingeniería

Número tarea: 26	**Número de Historia de Usuario:** 5
Nombre tarea: Eliminar mesa	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 4 de julio del 2023	**Fecha fin:** 4 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita eliminar las mesas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 27	**Número de Historia de Usuario:** 5
Nombre tarea: Visualizar detalles de una mesa	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 4 de julio del 2023	**Fecha fin:** 4 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita ver los detalles de las mesas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 28	**Número de Historia de Usuario:** 5
Nombre tarea: Listar mesas por zona	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 5 de julio del 2023	**Fecha fin:** 5 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar las mesas existentes en la base de datos según su zona.	

Tarea de Ingeniería	
Número tarea: 29	**Número de Historia de Usuario:** 5

Nombre tarea: Diseño de la iconografía referente a la HU_5	
Tipo de tarea: Diseño	**Puntos estimados:** 0.8
Fecha inicio: 5 de julio del 2023	**Fecha fin:** 5 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_5.	

Tarea de Ingeniería	
Número tarea: 30	**Número de Historia de Usuario:** 5
Nombre tarea: Diseño visual de la interfaz de gestión de mesas	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 6 de julio del 2023	**Fecha fin:** 6 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de mesas en el sistema.	

Tarea de Ingeniería	
Número tarea: 31	**Número de Historia de Usuario:** 5
Nombre tarea: Implementación de la interfaz de gestión de mesas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha inicio: 7 de julio del 2023	**Fecha fin:** 7 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de mesas en el sistema.	

Tarea de Ingeniería	
Número tarea: 32	**Número de Historia de Usuario:** 6
Nombre tarea: Listar clientes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 10 de julio del 2023	**Fecha fin:** 10 de julio del 2023

Programador responsable: Omar Sarmiento Rolo
Descripción: Implementar un mecanismo que permita listar los clientes existentes en la base de datos.

Tarea de Ingeniería	
Número tarea: 33	**Número de Historia de Usuario:** 6
Nombre tarea: Adicionar cliente	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 10 de julio del 2023	**Fecha fin:** 10 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar un cliente a la base de datos.	

Tarea de Ingeniería	
Número tarea: 34	**Número de Historia de Usuario:** 6
Nombre tarea: Modificar cliente	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 10 de julio del 2023	**Fecha fin:** 11 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar los clientes existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 35	**Número de Historia de Usuario:** 6
Nombre tarea: Eliminar cliente	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 11 de julio del 2023	**Fecha fin:** 11 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	

Descripción: Implementar un mecanismo que permita eliminar los clientes existentes en la base de datos.

Tarea de Ingeniería	
Número tarea: 36	**Número de Historia de Usuario:** 6
Nombre tarea: Visualizar detalles de un cliente	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 11 de julio del 2023	**Fecha fin:** 11 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita ver los detalles de los clientes existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 37	**Número de Historia de Usuario:** 6
Nombre tarea: Diseño de la iconografía referente a la HU_6	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 12 de julio del 2023	**Fecha fin:** 12 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_6.	

Tarea de Ingeniería	
Número tarea: 38	**Número de Historia de Usuario:** 6
Nombre tarea: Diseño visual de la interfaz de gestión de clientes	
Tipo de tarea: Diseño	**Puntos estimados:** 1.0
Fecha inicio: 13 de julio del 2023	**Fecha fin:** 13 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de clientes en el sistema.	

Tarea de Ingeniería	
Número tarea: 39	**Número de Historia de Usuario:** 6
Nombre tarea: Implementación de la interfaz de gestión de clientes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha inicio: 14 de julio del 2023	**Fecha fin:** 14 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de clientes en el sistema.	

Tarea de Ingeniería	
Número tarea: 40	**Número de Historia de Usuario:** 7
Nombre tarea: Listar categorías	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17 de julio del 2023	**Fecha fin:** 17 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar las categorías existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 41	**Número de Historia de Usuario:** 7
Nombre tarea: Adicionar categoría	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 18 de julio del 2023	**Fecha fin:** 19 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar una categoría a la base de datos.	

Tarea de Ingeniería

Número tarea: 42	**Número de Historia de Usuario:** 7
Nombre tarea: Modificar categoría	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 20 de julio del 2023	**Fecha fin:** 21 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar las categorías existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 43	**Número de Historia de Usuario:** 7
Nombre tarea: Eliminar categoría	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24 de julio del 2023	**Fecha fin:** 24 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita eliminar las categorías existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 44	**Número de Historia de Usuario:** 7
Nombre tarea: Diseño de la iconografía referente a la HU_7	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 25 de julio del 2023	**Fecha fin:** 25 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_7.	

Tarea de Ingeniería	
Número tarea: 45	**Número de Historia de Usuario:** 7
Nombre tarea: Diseño visual de la interfaz de gestión de categorías	

Tipo de tarea: Diseño	**Puntos estimados:** 1.5
Fecha inicio: 26 de julio del 2023	**Fecha fin:** 27 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de categorías en el sistema.	

Tarea de Ingeniería	
Número tarea: 46	**Número de Historia de Usuario:** 7
Nombre tarea: Implementación de la interfaz de gestión de categorías	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 27 de julio del 2023	**Fecha fin:** 28 de julio del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de categorías en el sistema.	

Tarea de Ingeniería	
Número tarea: 47	**Número de Historia de Usuario:** 8
Nombre tarea: Listar productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14 de agosto del 2023	**Fecha fin:** 14 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar los productos existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 48	**Número de Historia de Usuario:** 8
Nombre tarea: Adicionar producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5

Fecha inicio: 15 de agosto del 2023	**Fecha fin:** 16 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar un producto a la base de datos.	

Tarea de Ingeniería	
Número tarea: 49	**Número de Historia de Usuario:** 8
Nombre tarea: Modificar producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 16 de agosto del 2023	**Fecha fin:** 17 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar los productos existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 50	**Número de Historia de Usuario:** 8
Nombre tarea: Eliminar producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 18 de agosto del 2023	**Fecha fin:** 18 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita eliminar los productos existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 51	**Número de Historia de Usuario:** 8
Nombre tarea: Visualizar detalles de un producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21 de agosto del 2023	**Fecha fin:** 21 de agosto del 2023

Programador responsable: Omar Sarmiento Rolo
Descripción: Implementar un mecanismo que permita visualizar los detalles de los productos existentes en la base de datos.

Tarea de Ingeniería	
Número tarea: 52	**Número de Historia de Usuario:** 8
Nombre tarea: Diseño de la iconografía referente a la HU_8	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 22 de agosto del 2023	**Fecha fin:** 22 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_8.	

Tarea de Ingeniería	
Número tarea: 53	**Número de Historia de Usuario:** 8
Nombre tarea: Diseño visual de la interfaz de gestión de productos	
Tipo de tarea: Diseño	**Puntos estimados:** 1.5
Fecha inicio: 23 de agosto del 2023	**Fecha fin:** 24 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de productos en el sistema.	

Tarea de Ingeniería	
Número tarea: 54	**Número de Historia de Usuario:** 8
Nombre tarea: Implementación de la interfaz de gestión de productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 24 de agosto del 2023	**Fecha fin:** 25 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	

Descripción: Implementar la lógica programable de los componentes que integran la gestión de productos en el sistema.

Tarea de Ingeniería	
Número tarea: 55	**Número de Historia de Usuario:** 9
Nombre tarea: Listar pedidos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11 de septiembre del 2023	**Fecha fin:** 11 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar los pedidos existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 56	**Número de Historia de Usuario:** 9
Nombre tarea: Adicionar pedido	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 12 de septiembre del 2023	**Fecha fin:** 13 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar un pedido a la base de datos.	

Tarea de Ingeniería	
Número tarea: 57	**Número de Historia de Usuario:** 9
Nombre tarea: Modificar pedido	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13 de septiembre del 2023	**Fecha fin:** 15 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	

Descripción: Implementar un mecanismo que permita modificar los pedidos existentes en la base de datos.

Tarea de Ingeniería	
Número tarea: 58	**Número de Historia de Usuario:** 9
Nombre tarea: Visualizar detalles del pedido	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 18 de septiembre del 2023	**Fecha fin:** 18 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita visualizar los detalles de los pedidos existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 59	**Número de Historia de Usuario:** 9
Nombre tarea: Diseño de la iconografía referente a la HU_9	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 19 de septiembre del 2023	**Fecha fin:** 19 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_9.	

Tarea de Ingeniería	
Número tarea: 60	**Número de Historia de Usuario:** 9
Nombre tarea: Diseño visual de la interfaz de gestión de pedidos	
Tipo de tarea: Diseño	**Puntos estimados:** 1.5
Fecha inicio: 20 de septiembre del 2023	**Fecha fin:** 21 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de pedidos en el sistema.	

Tarea de Ingeniería	
Número tarea: 61	**Número de Historia de Usuario:** 9
Nombre tarea: Implementación de la interfaz de gestión de pedidos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 21 de septiembre del 2023	**Fecha fin:** 22 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de pedidos en el sistema.	

Tarea de Ingeniería	
Número tarea: 62	**Número de Historia de Usuario:** 10
Nombre tarea: Listar comandas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 28 de agosto del 2023	**Fecha fin:** 28 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar las comandas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 63	**Número de Historia de Usuario:** 10
Nombre tarea: Modificar estado de comanda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 29 de agosto del 2023	**Fecha fin:** 30 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar el estado de las comandas existentes en la base de datos.	

Tarea de Ingeniería

Número tarea: 64	**Número de Historia de Usuario:** 10
Nombre tarea: Modificar mesa de comanda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 30 de agosto del 2023	**Fecha fin:** 31 de agosto del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita modificar las mesas de las comandas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 65	**Número de Historia de Usuario:** 10
Nombre tarea: Invalidar comanda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1 de septiembre del 2023	**Fecha fin:** 1 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita invalidar las comandas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 66	**Número de Historia de Usuario:** 10
Nombre tarea: Cerrar comanda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 4 de septiembre del 2023	**Fecha fin:** 5 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita cerrar las comandas existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 67	**Número de Historia de Usuario:** 10

Nombre tarea: Diseño de la iconografía referente a la HU_11	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 6 de septiembre del 2023	**Fecha fin:** 6 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar la iconografía referente a la HU_11.	

Tarea de Ingeniería	
Número tarea: 68	**Número de Historia de Usuario:** 10
Nombre tarea: Diseño visual de la interfaz de gestión de comandas	
Tipo de tarea: Diseño	**Puntos estimados:** 1.5
Fecha inicio: 7 de septiembre del 2023	**Fecha fin:** 7 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de comandas en el sistema.	

Tarea de Ingeniería	
Número tarea: 69	**Número de Historia de Usuario:** 10
Nombre tarea: Implementación de la interfaz de gestión de comandas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 8 de septiembre del 2023	**Fecha fin:** 8 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de comandas en el sistema.	

Tarea de Ingeniería	
Número tarea: 70	**Número de Historia de Usuario:** 11
Nombre tarea: Listar pagos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2

Fecha inicio: 25 de septiembre del 2023	**Fecha fin:** 26 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita listar los pagos existentes en la base de datos.	

Tarea de Ingeniería	
Número tarea: 71	**Número de Historia de Usuario:** 11
Nombre tarea: Adicionar pago	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 27 de septiembre del 2023	**Fecha fin:** 29 de septiembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita adicionar un pago a la base de datos.	

Tarea de Ingeniería	
Número tarea: 72	**Número de Historia de Usuario:** 11
Nombre tarea: Cerrar pago	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 2 de octubre del 2023	**Fecha fin:** 3 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita cerrar los pagos que se generen en el sistema.	

Tarea de Ingeniería	
Número tarea: 73	**Número de Historia de Usuario:** 11
Nombre tarea: Diseño de la iconografía referente a la HU_11	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 4 de octubre del 2023	**Fecha fin:** 4 de octubre del 2023

Programador responsable: Omar Sarmiento Rolo
Descripción: Diseñar la iconografía referente a la HU_11.

Tarea de Ingeniería	
Número tarea: 74	**Número de Historia de Usuario:** 11
Nombre tarea: Diseño visual de la interfaz de gestión de pagos	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 5 de octubre del 2023	**Fecha fin:** 5 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la gestión de pagos en el sistema.	

Tarea de Ingeniería	
Número tarea: 75	**Número de Historia de Usuario:** 11
Nombre tarea: Implementación de la interfaz de gestión de pagos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 6 de octubre del 2023	**Fecha fin:** 6 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la gestión de pagos en el sistema.	

Tarea de Ingeniería	
Número tarea: 76	**Número de Historia de Usuario:** 12
Nombre tarea: Implementación del reporte de cierre de caja	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 9 de octubre del 2023	**Fecha fin:** 10 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de cierre de caja.	

Tarea de Ingeniería	
Número tarea: 77	**Número de Historia de Usuario:** 12
Nombre tarea: Implementación del reporte de ventas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 10 de octubre del 2023	**Fecha fin:** 11 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de ventas.	

Tarea de Ingeniería	
Número tarea: 78	**Número de Historia de Usuario:** 12
Nombre tarea: Implementación del reporte de historial de pedidos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 12 de octubre del 2023	**Fecha fin:** 13 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de historial de pedidos.	

Tarea de Ingeniería	
Número tarea: 79	**Número de Historia de Usuario:** 12
Nombre tarea: Diseño visual de los reportes del sistema	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 16 de octubre del 2023	**Fecha fin:** 17 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la generación de reportes en el sistema.	

Tarea de Ingeniería	
Número tarea: 80	**Número de Historia de Usuario:** 12

Nombre tarea: Implementación de la interfaz de generación de reportes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 18 de octubre del 2023	**Fecha fin:** 20 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la generación de reportes en el sistema.	

Tarea de Ingeniería	
Número tarea: 81	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación del reporte de predicción de ventas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.5
Fecha inicio: 23 de octubre del 2023	**Fecha fin:** 25 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de predicción de ventas.	

Tarea de Ingeniería	
Número tarea: 82	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación del reporte de predicción de ventas por productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.5
Fecha inicio: 25 de octubre del 2023	**Fecha fin:** 27 de octubre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de predicción de ventas por productos.	

Tarea de Ingeniería	
Número tarea: 83	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación del reporte de compras asociativas entre productos	

Tipo de tarea: Desarrollo	**Puntos estimados:** 2.5
Fecha inicio: 30 de octubre del 2023	**Fecha fin:** 1 de noviembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de compras asociativas entre productos.	

Tarea de Ingeniería	
Número tarea: 84	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación del reporte de probabilidad de compra de productos cuando se ha comprado otro	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.5
Fecha inicio: 1 de noviembre del 2023	**Fecha fin:** 3 de noviembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar un mecanismo que permita generar el reporte de probabilidad de compra de productos cuando se ha comprado otro.	

Tarea de Ingeniería	
Número tarea: 85	**Número de Historia de Usuario:** 13
Nombre tarea: Diseño visual de los reportes estadísticos del sistema	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 6 de noviembre del 2023	**Fecha fin:** 7 de noviembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Diseñar las vistas que comprenden la generación de reportes estadísticos en el sistema.	

Tarea de Ingeniería	
Número tarea: 86	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación de la interfaz de generación de reportes estadísticos	

Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 8 de noviembre del 2023	**Fecha fin:** 10 de noviembre del 2023
Programador responsable: Omar Sarmiento Rolo	
Descripción: Implementar la lógica programable de los componentes que integran la generación de reportes estadísticos en el sistema.	

Anexo E: Resumen de las Tareas de Ingeniería por Historias de Usuarios

A continuación el resumen de las tareas de ingenierías implementadas por cada Historia de Usuario.

Tabla E.1: Resumen de Tareas de Ingeniería por Historias de Usuario

Historia de Usuario	No. Tarea	Tarea de Ingeniería
Diseño y creación de la base de datos	2	1, 2
Sistema de autenticación	6	3, 4, 5, 6, 7, 8
Gestionar usuarios	7	9, 10, 11, 12, 13, 14, 15
Gestionar zonas	7	23, 24, 25, 26, 27, 28, 29
Gestionar mesas	9	30, 31, 32, 33, 34, 35, 36, 37, 38
Gestionar clientes	8	39, 40, 41, 42, 43, 44, 45, 46
Gestionar categorías	7	47, 48, 49, 50, 51, 52, 53
Gestionar productos	8	62, 63, 64, 65, 66, 67, 68, 69
Gestionar comandas	4	70, 71, 72, 73, 74, 75, 76, 77
Gestionar pedidos	7	78, 79, 80, 81, 82, 83, 84

Continúa en la siguiente página

Tabla E.1 Continuación de la página anterior

Gestionar pagos	6	85, 86, 87, 88, 89, 90
Generar reportes	5	91, 92, 93, 94, 95
Generar reportes estadísticos	6	96, 97, 98, 99, 100, 101

Anexo F: Casos de pruebas

A continuación la descripción de cada uno de los casos de pruebas realizados al sistemas.

Caso de prueba de aceptación	
Código: PA01_HU2	**Historia de Usuario:** HU_2
Nombre: Autenticar al sistema con errores I	
Descripción: Se intentará autenticar al sistema dejando los campos de correo y clave vacíos.	
Condiciones de ejecución: El sistema no debe haber iniciado sesión o tenerla cerrada en el servidor.	
Pasos de ejecución: Se ingresa a la pantalla de autenticación mediante la url en el navegador. Una vez que aparezca la vista inicial, se presiona el botón Aceptar del cuadro de diálogo con los campos correo y clave vacíos.	
Resultados esperados: El sistema debe mostrar que el correo y la clave son requeridas.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA02_HU2	**Historia de Usuario:** HU_2
Nombre: Autenticar al sistema con errores II	
Descripción: Se intentará autenticar al sistema con el correo y/o clave incorrectos.	
Condiciones de ejecución: El sistema no debe haber iniciado sesión o tenerla cerrada en el servidor.	

Pasos de ejecución: Se ingresa a la pantalla de autenticación mediante el navegador. Una vez que aparezca la vista inicial, se escribirá un correo y contraseña incorrectos y se dará clic en el botón Aceptar.
Resultados esperados: El sistema debe mostrar una alerta indicando que no es posible iniciar sesión con las credenciales proporcionadas.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA03_HU3	**Historia de Usuario:** HU_3
Nombre: Gestionar usuarios	
Descripción: Se adicionará, modificará y se eliminará un usuario del sistema	
Condiciones de ejecución: Debe existir al menos un usuario administrador en la base de datos.	
Pasos de ejecución: El usuario ingresa y se autentica en el sistema de administración a través del navegador, dará clic en el módulo de "Usuarios". Para adicionar un usuario, se dará clic en el botón "Adicionar", rellenará el formulario con los datos y adicionará el usuario. Para modificar un usuario, se dirigirá al listado de usuarios y se dará clic al nombre del usuario que se desea modificar, una vez modificado el usuario se dará clic en guardar. Para eliminar un usuario, se dirigirá al listado de usuarios y se dará clic al nombre del usuario que se desea eliminar, al final de la vista estará el botón eliminar, se dará clic y aparecerá un cuadro de confirmación de eliminación, clic en sí y se eliminará el usuario.	
Resultados esperados: Adicionar, modificar y eliminar un usuario del sistema.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA04_HU4	**Historia de Usuario:** HU_4
Nombre: Gestionar zonas	
Descripción: Se adicionará, modificará y se eliminará una zona del sistema	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos.	
Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de "Zonas". Para adicionar una zona, se dará clic en el botón "Adicionar", rellenará el formulario con los datos y adicionará la zona. Para modificar una zona, se dirigirá al listado de zonas y se dará clic al botón de modificar de la zona que se desee, una vez modificada la información se dará clic en salvar. Para eliminar una zona, se dirigirá al listado de zonas y se dará clic al botón de "Eliminar" de la zona que se desee. Aparecerá un mensaje de confirmación y se dará clic en "Eliminar".	
Resultados esperados: Adicionar, modificar y eliminar una zona del sistema.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA05_HU5	**Historia de Usuario:** HU_5
Nombre: Gestionar mesas	
Descripción: Se adicionará, modificará y se eliminará una zona del sistema. Se visualizará la mesa según la zona asignada	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos.	

Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de “Mesas”. Para adicionar una mesa, se dará clic en el botón “Adicionar”, rellenará el formulario con los datos y adicionará la mesa. Para modificar una mesa, se dirigirá al listado de mesas y se dará clic al botón de modificar de la mesa que se desee, una vez modificada la información se dará clic en salvar. Para eliminar una mesa, se dirigirá al listado de mesas y se dará clic al botón de “Eliminar” de la mesa que se desee. Aparecerá un mensaje de confirmación y se dará clic en “Eliminar”.
Resultados esperados: Adicionar, modificar y eliminar una mesa del sistema.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA06_HU6	**Historia de Usuario:** HU_6
Nombre: Gestionar clientes	
Descripción: Se adicionará, modificará y se eliminará un cliente del sistema	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos.	
Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de “Clientes”. Para adicionar un cliente, se dará clic en el botón “Adicionar”, rellenará el formulario con los datos y adicionará el cliente. Para modificar un cliente, se dirigirá al listado de clientes y se dará clic al botón de modificar del cliente que se desee, una vez modificada la información se dará clic en salvar. Para eliminar un cliente, se dirigirá al listado de clientes y se dará clic al botón de “Eliminar” del cliente que se desee. Aparecerá un mensaje de confirmación y se dará clic en “Eliminar”.	

Resultados esperados: Adicionar, modificar y eliminar un cliente del sistema.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA07_HU7	**Historia de Usuario:** HU_7
Nombre: Gestionar categorías	
Descripción: Se adicionará, modificará y se eliminará una categoría del sistema	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos.	
Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de "Categorías". Para adicionar una categoría, se dará clic en el botón "Adicionar", rellenará el formulario con los datos y adicionará la categoría. Para modificar una categoría, se dirigirá al listado de categorías y se dará clic al botón de modificar de la categoría que se desee, una vez modificada la información se dará clic en salvar. Para eliminar una categoría, se dirigirá al listado de categorías y se dará clic al botón de "Eliminar" de la categoría que se desee. Aparecerá un mensaje de confirmación y se dará clic en "Eliminar".	
Resultados esperados: Adicionar, modificar y eliminar una categoría del sistema.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA8_HU8	**Historia de Usuario:** HU_8
Nombre: Gestionar productos	

Descripción: Se adicionará, modificará y se eliminará un producto del sistema
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos. Una categoría en el sistema.
Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de "Productos". Para adicionar un producto, se dará clic en el botón "Adicionar", rellenará el formulario con los datos y adicionará el producto. Para modificar un producto, se dirigirá al listado de productos y se dará clic al botón de modificar del producto que se desee, una vez modificada la información se dará clic en salvar. Para eliminar un producto, se dirigirá al listado de productos y se dará clic al botón de "Eliminar" del producto que se desee. Aparecerá un mensaje de confirmación y se dará clic en "Eliminar".
Resultados esperados: Adicionar, modificar y eliminar un producto del sistema.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA9_HU9	**Historia de Usuario:** HU_9
Nombre: Gestionar pedidos	
Descripción: Se adicionará, se actualizará el estado y se visualizarán los detalles de un pedido	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos. Deben de existir productos, mesas y clientes en el sistema.	

Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de “Pedidos”. Para adicionar un pedido dará clic en el botón “Nuevo”, rellenará el formulario con los datos y adicionará el pedido. En caso de que el pedido sea de tipo Domicilio o Recoger debe seleccionar el cliente que lo solicite en vez de la mesa. Para adicionar un pedido de tipo barra se hará clic en el botón “Barra”, se seleccionará el producto y se cerrará el pedido directamente. Para que el estado del pedido se actualice basta con cerrar el pedido y pasará a estado cerrado. Para visualizar los detalles de un pedido el usuario hará clic en el botón de pedidos donde se muestran todas las tarjetas con los detalles de cada pedido realizado.
Resultados esperados: Adicionar, se actualizará el estado y se visualizarán los detalles de un pedido.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA10_HU10	**Historia de Usuario:** HU_10
Nombre: Gestionar comandas	
Descripción: Se cambiará el estado de una comanda, su mesa y se invalidará la comanda	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos. Debe de haber un pedido realizado.	

Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de "Pedidos". Para cambiar el estado de la comanda se hará clic en la mesa con color naranja y basta con cerrar el pedido y la comanda pasará a estado cerrado. Para cambiar la mesa de la comanda se hará clic en el botón cambiar, se seleccionará una mesa vacía (color gris) y se hará clic en editar. Para invalidar la comanda el usuario deberá dirigirse al módulo "Pedidos", seleccionará la mesa que desee invalidar, clic en ajustes y dará clic al botón de invalidar mesa y la comanda cambiará su estado a invalidada.
Resultados esperados: Cambiar el estado de una comanda, su mesa y se invalidará la comanda.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA11_HU11	**Historia de Usuario:** HU_11
Nombre: Gestionar pagos	
Descripción: Se adicionará y se cerrará un pago	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos. Debe de existir un pedido previamente realizado.	
Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de "Pedidos". Para adicionar un pago, el usuario deberá hacer clic en la mesa que se desee cerrar y dar clic en el botón de cerrar y se adicionará un nuevo pago. Para cerrar los pagos el usuario deberá dirigirse al módulo de "Cierre de Caja" y hacer clic en cerrar caja y todos los pagos realizados hasta la fecha se cerrarán.	
Resultados esperados: Adicionar y se cerrará un pago.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA12_HU12	**Historia de Usuario:** HU_12
Nombre: Generar reportes	
Descripción: Se generará el reporte de cierre de caja, de ventas e historial de pedidos	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos. Deben existir pedidos cerrados y al menos un cierre de caja realizado.	
Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de "Reportes", a la pantalla de ventas. Para generar el reporte de ventas se seleccionará el reporte de tipo ventas, el rango de fecha que se desee y clic en generar reporte. Para el historial de pedidos seguir los pasos anteriores seleccionando el tipo de reporte de historial de ventas. Para generar el reporte de cierre de caja el usuario deberá dirigirse al módulo de "Cierre de Caja", en la parte superior de la pantalla se mostrará el último cierre de caja realizado, dará clic en el botón de generar reporte.	
Resultados esperados: Generar el reporte de cierre de caja, de ventas e historial de pedidos.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA13_HU13	**Historia de Usuario:** HU_13
Nombre: Generar reportes estadísticos	
Descripción: Se generarán todos los tipos de reportes estadísticos	
Condiciones de ejecución: Debe existir al menos un usuario administrador o dependiente en la base de datos. Deben existir un volumen considerable de datos que permitan realizar estos reportes.	

Pasos de ejecución: El usuario ingresa y se autentica en el sistema a través del navegador, dará clic en el módulo de “Reportes”, a la pantalla de predicciones. Seleccionará el tipo de reporte que desee y a continuación clic en generar reporte.
Resultados esperados: Generar los diferentes tipos de reportes estadísticos del sistema.
Evaluación de la prueba: Satisfactoria