PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

CARRERA DE: TECNOLOGÍAS DE LA INFORMACIÓN

Trabajo de Titulación

Tema: Desarrollo de Aplicación Prototipo Web para la Gestión Integral de Restaurantes

AUTOR:

EMILIO CHASIPANTA

TUTOR:

ARCOS VILLAGOMEZ SUYANA FABIOLA

QUITO DM, JULIO DE 2024

DEDICATORIA

El presente trabajo está dedicado a mi familia por su continuo apoyo durante mi trayectoria, por estar presente a todo momento de mi vida, gracias por estar presente en cada meta que me he propuesto.

EMILIO CHASIPANTA

AGRADECIMIENTO

Un agradecimiento a los docentes de la Facultad de Ingeniería que aportaron con sus conocimientos y experiencia para el desarrollo del presente proyecto, a la Pontificia Universidad Católica del Ecuador, por su adecuada gestión en la formación de profesionales competitivos, como también a mis compañeros quienes han sido parte de este proceso.

EMILIO CHASIPANTA

RESUMEN

El presente proyecto está enfocado en el desarrollo de una Aplicación Web para la Gestión Integral de Restaurantes, caso de estudio Restaurante Tonys, considerando que los avances tecnológicos aportan a los procesos de cualquier empresa, microempresa o emprendimiento, haciéndole más eficientes y efectivos. La competitividad entre restaurantes hace que se deba mejorar el servicio percibido por el cliente, por ello es importante contar con indicadores KPI (Indicador Clave de Desempeño o Medidor de Desempeño), que permiten controlar el rendimiento de los procesos, para el monitoreo de cada uno de los procesos que encierra la cadena de valor de los centros culinarios.

La aplicación del sistema web se basa en un modelo iterativo cumpliendo etapas como el registro de pedidos, envió de pedidos, registro de transacciones y la definición de roles y control. Además, se emplea la metodología Agile que centra la implementación rápida de un equipo eficientemente flexible para concebir el flujo de trabajo, se finaliza el proyecto con conclusiones y recomendaciones.

ÍNDICE

Capítulo 1: Introducción

1. Problematización

1.1 Planteamiento del problema

El Restaurante Tonys se encuentra participando en el mercado alrededor de 4 años, es un emprendimiento gastronómico que ofrece platos atractivos basados en diferentes carnes, cuenta con 10 colaboradores en el área operativa y 4 personas en el área administrativa, su facturación mensual es alrededor de 15.000 dólares americanos. Cuenta con una infraestructura de 600 metros cuadrados distribuidos en sus diferentes áreas que acogen a sus clientes siendo importante ofrecer un servicio eficiente y de calidad.

Al analizar una evaluación de las necesidades en el Restaurante Tonys e identificar las áreas que requieren mejoras, se puede apreciar la pérdida de tiempo en la recepción y atención a los clientes, generando pérdida de clientes por no tener una gestión integral del personal. Con el desarrollo del prototipo de la aplicación web se plantea agilitar los procesos con información verídica y confiable en función de los requerimientos de los clientes y los recursos que cuenta el restaurante.

1.2 Objetivos de la investigación

1.2.1 Objetivo General

Desarrollar un software prototipo para el Restaurante Tonys que permita una gestión integral, haciendo que las varias áreas de trabajo estén controladas mejorando la productividad y reduciendo costos para su propio beneficio.

1.2.2 Objetivo Específico

- Levantar la información de los requerimientos necesarios para la comunicación entre las áreas, mediante la conformación de reuniones focalizadas con los colaboradores y la utilización de una encuesta que con lleve a la definición de las necesidades del

cliente.

- Diseñar el software enfocado a la gestión integral para el restaurante mediante el desarrollo de la metodología Agile, enfocada a cumplir las necesidades personalizadas del restaurante.
- Desarrollar el software prototipo del restaurante para que pueda ejecutarse la gestión pertinente y todo esté controlado en un ambiente tecnológico.

1.3 Justificación de la investigación

El desarrollo de un software de gestión integral para aplicarlo en un restaurante permite mejorar la eficiencia operativa y reducir los costos, además proporciona herramientas analíticas y estratégicas que son cruciales para competir en un mercado cada vez más exigente y dinámico. Al abarcar desde la gestión de comandas hasta la administración del inventario y el control financiero.

La utilización del software de gestión integral se convierte en un aliado fundamental para garantizar un servicio de calidad y una gestión financiera efectiva, haciendo que las varias áreas de trabajo estén controladas y que la productividad del restaurante sea mejor para su propio beneficio.

Se justifica porque en este tipo de emprendimientos familiares se desconoce de la gestión integral mediante el uso de software, puesto que en el mercado actual se cuenta con módulos básicos que satisfacen las necesidades de la administración de manera general, por lo que la propuesta plantea diseñar la aplicación en función de los requerimientos que demanda la oferta de servicio.

1.4 Alcances y limitaciones de la investigación

El alcance del presente proyecto se enfoca en la creación de módulos para la aplicación web que permitan un control en la gestión integral para la toma de decisiones de la gerencia en lo referente al entorno de trabajo, considerando que la aplicación web permite

comunicar los pedidos de los clientes con todo el entorno del restaurante de manera más rápida y eficaz. El desarrollo de funcionalidades para la gestión de inventarios, gestión de pedidos, gestión de personal, gestión de reservas y la integración con sistemas de pago y plataformas de entrega. Sus limitaciones se basan en que a medida que el restaurante crece, la aplicación debe ser capaz de escalar para manejar mayores volúmenes de datos y transacciones.

Capítulo 2: Marco teórico

2. Revisión de la literatura sobre desarrollo de software

2.1 Modelos iterativos

Los modelos iterativos son una metodología de desarrollo de software que enfatiza la repetición de ciclos (iteraciones) de trabajo, permitiendo refinamientos incrementales del producto. Este enfoque se diferencia de los modelos tradicionales secuenciales (como el modelo en cascada) por su capacidad de adaptarse y evolucionar con base en el feedback recibido a lo largo del proceso. Aquí te explico en detalle las fases de un modelo iterativo:

- Planificación:
    a. Definición de Objetivos: En esta fase se establecen los objetivos específicos de la iteración. Esto puede incluir la identificación de requisitos específicos que se desean abordar en el ciclo actual.
    b. Estimación de Recursos: Se estiman los recursos necesarios, incluyendo tiempo, personal, y herramientas.
    c. Riesgos: Identificación y evaluación de posibles riesgos y preparación de estrategias de mitigación.
    d. Cronograma: Elaboración de un cronograma detallado que incluya tareas y hitos.
- Desarrollo:
    a. Diseño: Creación o actualización del diseño del software basado en los requisitos establecidos para la iteración. Esto puede incluir diagramas de arquitectura, modelos de datos y esquemas de interacción.
    b. Codificación: Escribir el código fuente siguiendo las especificaciones del diseño. El código debe ser modular y fácil de mantener para facilitar futuras iteraciones.

c. Integración: Integración del nuevo código con el sistema existente. Es crucial mantener una integración continua para identificar problemas de integración tempranamente.

- Pruebas:
  a. Pruebas Unitarias: Pruebas de los componentes individuales para asegurar que funcionan correctamente.
  b. Pruebas de Integración: Verificación de que los componentes interactúan correctamente entre sí.
  c. Pruebas de Sistema: Evaluación del sistema en su conjunto para asegurar que cumple con los requisitos especificados.
  d. Pruebas de Aceptación: Realizadas con la participación de los usuarios finales para asegurar que el software cumple con sus necesidades y expectativas.
- Revisión:
  a. Revisión del Desempeño: Evaluación del desempeño del software desarrollado durante la iteración.
  b. Revisión de Calidad: Aseguramiento de la calidad del software, incluyendo revisiones de código y auditorías de calidad.
  c. Feedback: Recopilación de feedback de todas las partes interesadas (usuarios, desarrolladores, testers) para identificar áreas de mejora.
  d. Planificación de la Próxima Iteración: Basado en el feedback recibido, se planifican las mejoras y los nuevos objetivos para la siguiente iteración.

Los modelos iterativos en el desarrollo de software ofrecen varias ventajas significativas en comparación con enfoques tradicionales como:

- Flexibilidad: Permiten adaptarse rápidamente a los cambios en los requisitos o en el entorno.

- Feedback Continuo: Facilitan la obtención de feedback temprano y frecuente de los usuarios, lo que mejora la calidad del producto final.
- Riesgo Reducido: Permiten identificar y mitigar riesgos en etapas tempranas del desarrollo.
- Mejora Continua: Cada iteración ofrece la oportunidad de mejorar y refinar el producto.

Sin embargo, las ventajas que representa este tipo de modelos Iterativos se puntualizan en lo siguiente:

- Gestión Compleja: Requiere una gestión cuidadosa para coordinar las múltiples iteraciones y asegurar que se mantenga el rumbo.
- Esfuerzo de Planificación: La planificación detallada y la re-evaluación frecuente pueden consumir tiempo y recursos.
- Sobrecarga de Documentación: La documentación debe actualizarse constantemente para reflejar los cambios realizados en cada iteración.

Se presenta a continuación algunos de los Modelos Iterativos de importancia en el presente estudio.

- Modelo en Espiral: Combina elementos de diseño y prototipos en una serie de iteraciones basadas en la evaluación del riesgo.
- Desarrollo Ágil: Enfoques como Scrum y Kanban que enfatizan iteraciones cortas, la colaboración continua con el cliente y la entrega frecuente de incrementos de producto funcionales.

Por lo tanto, se puede concluir que los modelos iterativos son fundamentales en el desarrollo de software moderno debido a su capacidad para manejar la incertidumbre y adaptarse rápidamente a los cambios. Proveen un marco estructurado pero flexible que

facilita la entrega de productos de alta calidad que satisfacen las necesidades de los usuarios. (Castro, Solarte, & Muñoz, 2019)

2.2 Tipos de software

Según (Pardo & Rodil, 2022) los tipos de software de acuerdo con su función, se puede clasificar en los siguientes tipos:

- Software de sistema: El software de sistema es software que proporciona un entorno para ejecutar otros programas. Ejemplos de software de sistema incluyen el sistema operativo, el administrador de archivos y el compilador.
- Software de aplicación: El software de aplicación es software que realiza una tarea específica para el usuario. Ejemplos de software de aplicación incluyen el procesador de texto, la hoja de cálculo y el navegador web.

Según su tamaño, el software se puede clasificar en los siguientes tipos:

- Software grande: El software grande es software que es complejo y requiere un equipo de desarrollo grande. Ejemplos de software grande incluyen los sistemas operativos, los sistemas de bases de datos y los sistemas de información empresarial.
- Software pequeño: El software pequeño es software que es relativamente simple y puede ser desarrollado por una sola persona o un pequeño equipo. Ejemplos de software pequeño incluyen los programas de escritorio, los juegos y las aplicaciones web.

Según su complejidad, el software se puede clasificar en los siguientes tipos:

- Software sencillo: El software sencillo es software que es relativamente simple y no requiere mucho esfuerzo para desarrollarlo. Ejemplos de software sencillo incluyen los programas de calculadora, los programas de ajedrez y los programas de dibujo.
- Software complejo: El software complejo es software que es difícil de desarrollar y requiere mucho esfuerzo. Ejemplos de software complejo

incluyen los sistemas operativos, los sistemas de bases de datos y los sistemas de información empresarial.

2.3 Procesos de desarrollo de software

Los diferentes procesos de desarrollo de software son guías que describen las actividades y los artefactos involucrados en el desarrollo de un sistema de software, se pueden clasificar en dos categorías principales:

- Los procesos formales son procesos que están bien definidos y documentados. Los procesos formales suelen ser utilizados por organizaciones que desarrollan software crítico o que necesitan cumplir con estándares específicos.
- Los procesos informales son procesos que no están bien definidos y documentados. Los procesos informales suelen ser utilizados por organizaciones que desarrollan software de bajo riesgo o que no necesitan cumplir con estándares específicos. (Sommerville, 2011)

2.4 Base de datos en la nube

2.4.1 Base de datos NoSQL

La base de datos NoSQL, no utiliza el modelo relacional tradicional, puesto que utilizan modelos de datos alternativos, como el modelo clave-valor, el modelo de documentos o el modelo grafos. En estos últimos años se han vuelto muy populares, puesto que ofrece una serie de ventajas sobre las bases de datos tradicionales, como:

- Flexibilidad: Las bases de datos NoSQL son más flexibles que las bases de datos relacionales tradicionales, ya que no requieren que los datos se almacenen en una estructura de datos predefinida. Esto hace que las bases de datos NoSQL sean una buena opción para aplicaciones que requieren un almacenamiento de datos flexible.

- Escalabilidad: Las bases de datos NoSQL son más escalables que las bases de datos relacionales tradicionales, ya que pueden crecer y adaptarse fácilmente a las necesidades cambiantes de las aplicaciones.
- Eficiencia: Las bases de datos NoSQL pueden ser más eficientes que las bases de datos relacionales tradicionales, ya que utilizan modelos de datos más simples. (Gartner, 2023)

En el contexto de la nube, las bases de datos NoSQL ofrecen una serie de ventajas adicionales, como:

- Facilidad de uso: Ya que son más fáciles de usar que las bases de datos relacionales tradicionales, porque no requieren un conocimiento nativo de los modelos de datos relacionales.
- Coste: Pueden ser más rentables que las bases de datos relacionales, ya que se pueden escalar y pagar según sea necesario. (Gartner, 2023)

2.4.2 Tipos de bases de datos no SQL en la nube

Existen una variedad de tipos de bases de datos no SQL disponibles en la nube, algunos de los tipos más populares incluyen:

- Bases de datos clave-valor: Las bases de datos clave-valor almacenan los datos en pares de claves y valores. Las claves son únicas y se utilizan para acceder a los valores. Las bases de datos clave-valor son una buena opción para aplicaciones que requieren un acceso rápido y sencillo a los datos.
- Bases de datos de documentos: Las bases de datos de documentos almacenan los datos en documentos. Los documentos pueden contener cualquier tipo de datos, incluidos texto, números, imágenes y archivos. Las bases de datos de documentos son una buena opción para aplicaciones que requieren un almacenamiento de datos flexible y escalable.

- Bases de datos de grafos: Las bases de datos de grafos almacenan los datos en forma de grafos. Los grafos son estructuras de datos que representan relaciones entre objetos. Las bases de datos de grafos son una buena opción para aplicaciones que requieren un análisis de datos relacional.

"La adopción de las bases de datos no SQL en la nube está creciendo rápidamente. Según un informe de Gartner, el mercado de las bases de datos no SQL en la nube crecerá a una tasa anual compuesta del 27,9 % entre 2022 y 2027" (Addison-Wesley, 2019)

2.5 Microframework de Python Flask

Flask es un Microframework web de Python que se desarrolló inicialmente en 2010. Es un framework ligero y flexible que se puede utilizar para crear una amplia gama de aplicaciones web. Flask se ha vuelto cada vez más popular en los últimos años, especialmente entre los desarrolladores de Python que buscan un framework simple y fácil de usar. Según el sitio web de PyPI, Flask es el segundo framework web más popular de Python, después de Django. Lo que más destaca es definir rutas URL fácilmente con decoradores, mapeando URLs especificas a funciones Python, facilitando un Routing sencillo y ayudando a proporcionar un servidor de desarrollo integrado y recarga automática del código. (Ronacher, 2015)

2.5.1 Características de Flask

Flask ofrece una serie de características que lo hacen un framework atractivo para los desarrolladores web:

- Ligereza: Flask es un framework ligero que requiere pocas dependencias. Esto lo hace una buena opción para aplicaciones web que necesitan ser rápidas y eficientes.

- Flexibilidad: Flask es un framework flexible que permite a los desarrolladores personalizar sus aplicaciones web según sus necesidades.
- Facilidad de uso: Flask es un framework fácil de usar que no requiere un conocimiento profundo de Python o de los frameworks web. (Ronacher, 2015)

2.5.2 Ventajas de Flask

Flask ofrece una serie de ventajas sobre otros frameworks web:

- Rendimiento: Flask es un framework ligero que puede ofrecer un rendimiento superior a otros frameworks web más complejos.
- Escalabilidad: Flask es un framework escalable que se puede adaptar a las necesidades cambiantes de las aplicaciones web.
- Seguridad: Flask ofrece una serie de funciones de seguridad integradas que ayudan a proteger las aplicaciones web de los ataques.

2.5.3 Desventajas de Flask

Flask también tiene algunas desventajas que los desarrolladores deben tener en cuenta:

- Falta de características: Flask es un microframework, lo que significa que no ofrece todas las características que se encuentran en los frameworks web más completos.
- Soporte de terceros: Flask tiene un ecosistema de soporte de terceros más pequeño que otros frameworks web más populares.

2.5.4 Aplicaciones de Flask

Flask se puede utilizar para crear una amplia gama de aplicaciones web, incluyendo:

- Aplicaciones web simples: Flask es una buena opción para crear aplicaciones web simples, como sitios web de una página o aplicaciones web con un número limitado de funciones.
- Aplicaciones web complejas: Flask también se puede utilizar para crear aplicaciones web complejas, como aplicaciones web empresariales o aplicaciones web con una gran cantidad de funciones.
- Aplicaciones web en la nube: Flask es una buena opción para crear aplicaciones web en la nube, ya que es un framework ligero y escalable.

2.5.5 Casos de uso de Flask

Flask se utiliza en una amplia gama de aplicaciones web, incluyendo:

- Sitios web de comercio electrónico: Flask se utiliza en sitios web de comercio electrónico como Shopify y Etsy.
- Aplicaciones de redes sociales: Flask se utiliza en aplicaciones de redes sociales como X y Reddit.
- Aplicaciones de mensajería: Flask se utiliza en aplicaciones de mensajería como Slack y WhatsApp.
- Aplicaciones de análisis de datos: Flask se utiliza en aplicaciones de análisis de datos como Datadog y Grafana. (Grinberg, 2018)

2.6 Revisión de la literatura sobre restaurantes de pizza y parrillada

2.6.1 El funcionamiento de los restaurantes de pizza y parrillada

El Restaurante Tonys se encuentra en la línea de restaurantes ofrece en su gama de productos pizzas y parrilladas como platos principales, el funcionamiento de los restaurantes de pizza y parrillada suele ser similar al de otros tipos de restaurantes de cadenas gastronómicas, en donde el personal se encarga de atender a los clientes, tomar sus pedidos, preparar los alimentos y servirlos.

En el caso de los restaurantes de pizza y parrillada, la cocina suele estar dividida en dos secciones: la sección de pizzas y la sección de parrilladas. La sección de pizzas se encarga de preparar las pizzas, mientras que la sección de parrilladas se encarga de preparar las parrilladas, dichos productos suelen estar disponibles en una variedad de tamaños y sabores, mientras que las parrilladas suelen estar disponibles en una variedad de carnes, pescados y verduras.

Los diferentes tipos de restaurantes de pizza y parrillada pueden clasificarse en diferentes tipos según una serie de factores, como la ubicación, el tamaño, el servicio y el menú.

- Según la ubicación: los restaurantes de pizza y parrillada pueden clasificarse en restaurantes de servicio completo, restaurantes de servicio rápido y restaurantes de entrega. Los restaurantes de servicio completo ofrecen un servicio completo, que incluye camareros, mesas y una carta de vinos. Los restaurantes de servicio rápido ofrecen un servicio rápido, que suele incluir autoservicio y una carta limitada. Los restaurantes de entrega ofrecen comida para llevar o a domicilio.
- Según el tamaño: los restaurantes de pizza y parrillada pueden clasificarse en restaurantes pequeños, medianos y grandes. Los restaurantes pequeños suelen tener menos de 50 asientos, mientras que los restaurantes medianos suelen tener entre 50 y 100 asientos. Los restaurantes grandes suelen tener más de 100 asientos.
- Según el servicio: los restaurantes de pizza y parrillada pueden clasificarse en restaurantes informales y restaurantes formales. Los restaurantes informales suelen tener un ambiente relajado y un servicio informal, mientras que los restaurantes formales suelen tener un ambiente elegante y un servicio formal.
- Según el menú: los restaurantes de pizza y parrillada pueden clasificarse en restaurantes especializados y restaurantes con un menú variado. Los restaurantes especializados ofrecen una selección limitada de platos,

principalmente pizzas y parrilladas. Los restaurantes con un menú variado ofrecen una selección más amplia de platos, que incluyen pizzas, parrilladas, ensaladas, pastas, etc.

2.6.2 Las tendencias en el sector de la pizza y parrillada

El sector de la pizza y parrillada está experimentando una serie de tendencias, como:

- El aumento de la demanda de pizzas y parrilladas saludables.
- El aumento de la demanda de pizzas y parrilladas personalizadas.
- El aumento de la demanda de pizzas y parrilladas preparadas con ingredientes frescos.

En el caso del Restaurante Tonys, se puede observar que el restaurante se adapta a estas tendencias. Por ejemplo, el restaurante ofrece una variedad de pizzas con ingredientes saludables, como la pizza de verduras y la pizza de pollo a la parrilla. También ofrece la posibilidad de personalizar las pizzas, permitiendo a los clientes elegir los ingredientes que quieran. (Allen, 2019)

Capítulo 3: Desarrollo del software

3. Desarrollo del software

3.1 Funcionalidades

Las funcionalidades del software se caracterizan por las siguientes etapas para su desarrollo:

- Registro de pedidos: El software puede ayudar a los camareros a registrar los pedidos de los clientes de manera rápida y precisa. Esto puede ayudar a evitar errores y garantizar que los pedidos se envíen a la cocina correctamente.
- Envió de pedidos: El software puede enviar los pedidos directamente a la cocina en tiempo real, lo que permite a garantizar que los pedidos se cocinen rápidamente y que los clientes reciban su comida lo antes posible.
- Registro de transacciones: El software puede ayudar a los empleados a registrar las transacciones de manera rápida y precisa, que conlleva a evitar errores y garantizar que el dinero se registre correctamente.
- Definición de roles y control: El software puede ayudar a los gerentes a programar a los empleados de manera eficiente, esto puede ayudar a reducir la inactividad de cocina.

3.2 Tecnología

Python es uno de los lenguajes de programación más populares del mundo, y se utiliza en una amplia gama de aplicaciones. Flask es un microframework web de Python que proporciona una estructura básica para el desarrollo de aplicaciones web, fácil de aprender y usar, y es adecuado para una amplia gama de proyectos web.

MongoDB es una base de datos NoSQL documentada que utiliza un modelo de documentos JSON, al ser una base de datos escalable y flexible adecuada para una amplia gama de aplicaciones, incluyendo aplicaciones web, aplicaciones móviles e IoT. De esta

manera ayuda a cada una de las tecnologías a desarrollar una aplicación para el restaurante de pizzas y parrilladas Tonys de las siguientes maneras:

- Python es un lenguaje de programación potente y versátil que puede utilizarse para crear una aplicación web completa.
- Python permite gestionar los pedidos de los clientes, calcular el precio de los pedidos y realizar los cobros.
- Flask es un microframework web fácil de aprender y usar que puede utilizarse para crear la aplicación web rápida y eficiente que van a utilizar los usuarios con sus diferentes roles.
- MongoDB al ser una base de datos NoSQL escalable y flexible que puede utilizarse para almacenar los datos de la aplicación, como podría utilizarse para almacenar información sobre las comandas, acceso de usuarios, tiempos de preparación, inventario, entre otros.

3.3 Metodología Agile

3.3.1 Etapa Evaluar

Se realiza un diagnóstico de la situación actual con la finalidad de evaluar los procesos del restaurante para poder recopilar información de la manera como se maneja actualmente la empresa, mediante una encuesta dirigida a los clientes internos y externos. De donde se interpreta que no se tiene un buen control de las comandas tanto de parte del mesero, como la cocina y la caja, además se indica por parte de los colaboradores que en varias ocasiones se han perdido varias comandas, lo que ocasiona un descuadre al momento de cerrar el día.

También se puede observar que el principal problema es que no se tiene evaluado cuanto tiempo se demoran en realizar una comanda los chicos de cocina, con ello se va a servir para sacar un KPI de la “Entrega de órdenes a tiempo”, tampoco se sabe el KPI de “Periodo de Inactividad” y el KPI de “Facturación por meta semanal”. A continuación, se evidencia como se controla las comandas.

Figura 1

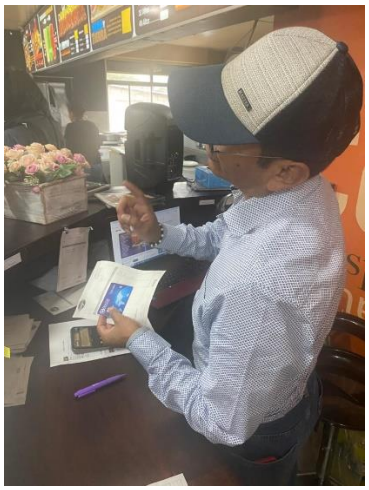
Evidencia de comandas

Nota: Tomado del Restaurante Tonys

Se puede evidenciar como se receptan las comandas en la actualidad, es un sistema tradicional de comandas en papel, siendo poco eficiente para el tema de cocina y administrativo, ya que se pueden perder las comandas.

Figura 2

Evidencia de recepción de pedidos

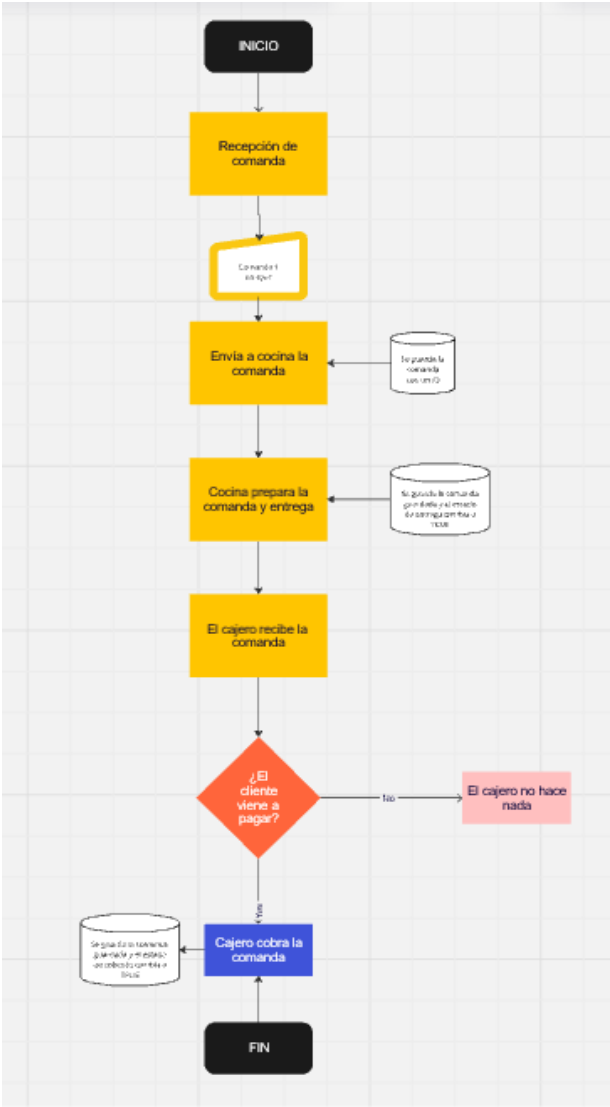


Nota: Tomado del Restaurante Tonys

Se puede evidencia el proceso de cobro, el cual se tiene un conflicto al momento de establecer que mesa es la que se tiene que cobrar, ya que las mesas pueden cambiar de lugar según los eventos que existan. Esto es un conflicto también al momento de hacer un cierre de caja ya que tienen que estar todas las comandas, pero suelen perderse mucho y no se tiene un valor exacto de cuanto se ha vendido, a continuación, se presenta el diagrama de flujo del proceso de la atención al cliente.

Figura 3

Proceso de atención al cliente

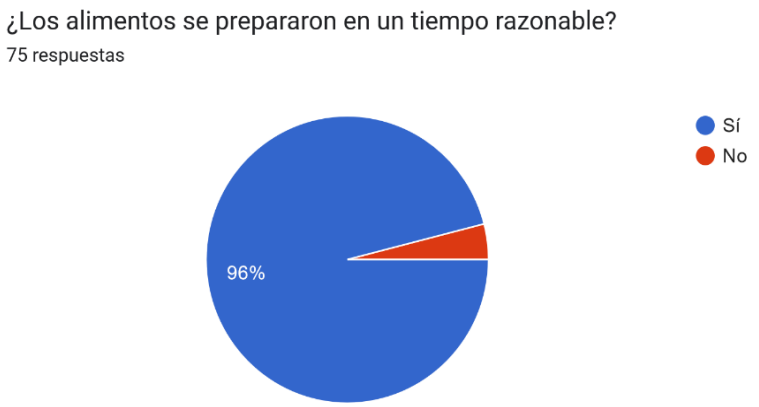


Nota: Tomado del Restaurante Tonys

Se realizó una encuesta para medir la satisfacción de los clientes con respecto al tiempo y eficiencia del restaurante, en base a esta información se pudo conocer los requerimientos básicos que debe contener el software para la gestión integral. Los resultados de la aplicación de la encuesta se presentan a continuación:

Figura 4

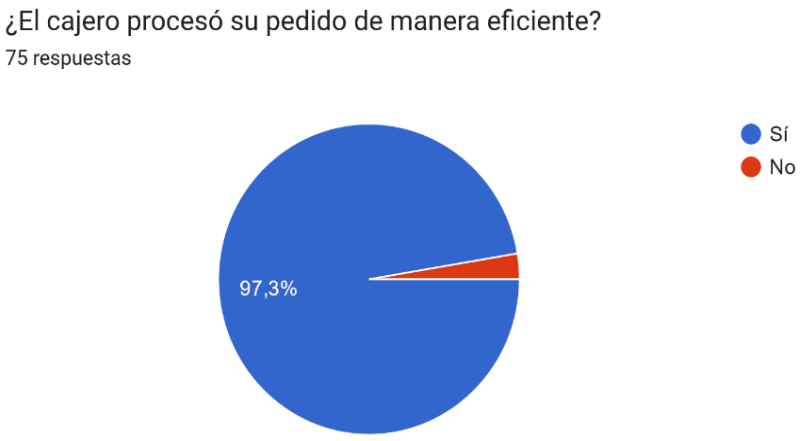
Encuesta Pregunta 1



Nota: Tomado de la encuesta aplicada a los clientes del Restaurante Tonys

Figura 5

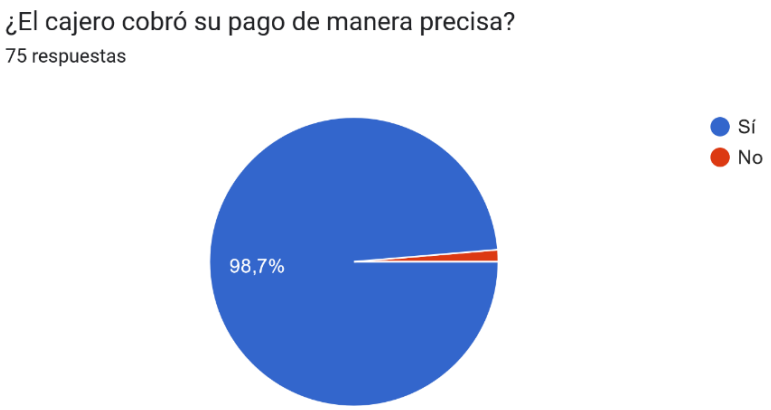
Encuesta Pregunta 2



Nota: Tomado de la encuesta aplicada a los clientes del Restaurante Tonys

Figura 6

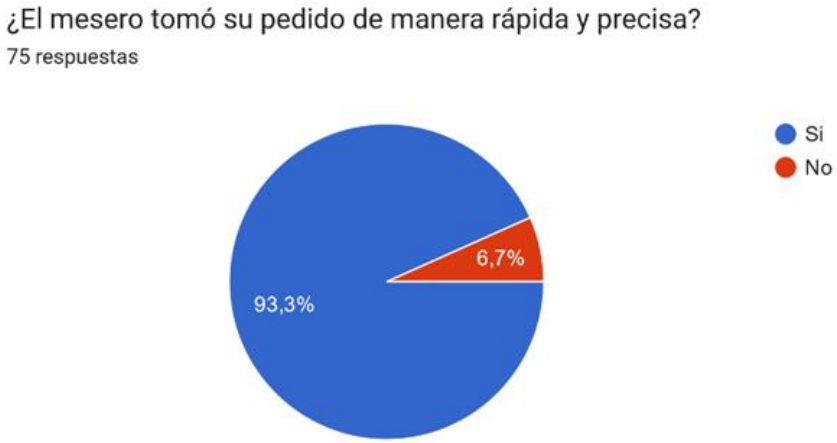
Encuesta Pregunta 3



Nota: Tomado de la encuesta aplicada a los clientes del Restaurante Tonys

Figura 7

Encuesta Pregunta 4



Nota: Tomado de la encuesta aplicada a los clientes del Restaurante Tonys

3.3.2 Etapa Sugerir

En la etapa de sugerir se establece la necesidad de desarrollar un software que permita mejorar los procesos del Restaurante Tonys pizza y parrillada, partiendo de la identificación del problema que contempla la falta de control de las comandas, la cual

ocasiona una serie de problemas, como la pérdida de las mismas, el descuadre de caja y la imposibilidad de medir los KPIs de entrega de órdenes a tiempo. Se plantea, por lo tanto, para solucionar este problema, el desarrollo de un software que automatice los procesos de registro, envío y seguimiento de las comandas. El software debería cumplir con los siguientes requisitos:

- Registro de comandas: El software debe permitir a los meseros registrar las comandas de manera rápida y sencilla. Para ello, debe contar con una interfaz intuitiva y fácil de usar.
- Envío de comandas: El software debe enviar las comandas a la cocina de manera automática. Esto evitará que las comandas se pierdan o se confundan.
- Seguimiento de comandas: El software debe permitir a los meseros y a la cocina realizar un seguimiento del estado de las comandas. Esto ayuda a garantizar que las órdenes se entreguen a tiempo.

Además de estos requisitos básicos, el software debería permitir medir los KPIs mencionados anteriormente. Para ello, el software debería registrar el tiempo que tarda la cocina en preparar las órdenes, el tiempo que los clientes esperan a que se les entregue su comida y la facturación del restaurante.

Para lo cual, el software podría dividirse en dos módulos principales: un módulo para meseros y un módulo para cocina. El módulo para meseros sería responsable de registrar las comandas y de enviarlas a la cocina. El módulo para cocina sería responsable de preparar las órdenes y de notificar a los meseros cuando las órdenes estén listas.

El módulo para meseros podría implementarse como una aplicación web. Esto permitiría a los meseros registrar las comandas desde cualquier lugar del restaurante. El módulo para cocina podría implementarse como una aplicación web o como una aplicación de escritorio. La sugerencia para la implementación del software es la siguiente:

- Fase 1: En esta fase, se implementaría el módulo para meseros. Esto permitiría a los meseros comenzar a registrar las comandas de manera digital.
- Fase 2: En esta fase, se implementaría el módulo para cocina. Esto permitiría completar la automatización de los procesos de registro, envío y seguimiento de las comandas.

La implementación del software proporcionaría los siguientes beneficios al restaurante:

- Mejor control de las comandas: El software ayudaría a evitar la pérdida de comandas y el descuadre de caja.
- Mejora de la eficiencia: El software automatizaría los procesos de registro, envío y seguimiento de las comandas, lo que liberaría tiempo a los empleados para otras tareas.
- Mejora de la experiencia del cliente: El software ayudaría a garantizar que las órdenes se entreguen a tiempo, lo que mejoraría la satisfacción de los clientes.

3.3.3 Etapa Diseñar

El uso de prototipos de baja fidelidad en el desarrollo de aplicaciones web y móviles es una práctica ampliamente adoptada por varias razones clave

- Rapidez en la Creación:

Los prototipos de baja fidelidad se pueden crear rápidamente, permitiendo a los diseñadores y desarrolladores visualizar y experimentar con diferentes ideas sin invertir mucho tiempo ni recursos.

- Facilidad de Modificación:

Al ser simples y económicos de producir, los prototipos de baja fidelidad pueden modificarse y ajustarse fácilmente en respuesta al feedback de los usuarios y las partes interesadas. Esto fomenta un proceso iterativo de diseño y mejora continua.

- Fomento del Feedback:

Al presentar un prototipo de baja fidelidad, es más probable que los usuarios y las partes interesadas se sientan cómodos proporcionando críticas y sugerencias, ya que estos prototipos no tienen el aspecto de un producto finalizado. Esto ayuda a identificar problemas y oportunidades de mejora en las primeras etapas del desarrollo.

- Enfoque en la Funcionalidad:

Los prototipos de baja fidelidad permiten a los equipos concentrarse en la estructura y funcionalidad básica de la aplicación sin distraerse con detalles estéticos. Esto asegura que los aspectos fundamentales del diseño y la usabilidad se aborden antes de pasar a fases más detalladas y costosas.

- Coste Eficiente:

La creación de prototipos de baja fidelidad requiere menos recursos financieros y técnicos en comparación con prototipos de alta fidelidad. Esto es particularmente beneficioso en las fases iniciales del proyecto, donde las ideas y los conceptos aún están en desarrollo y pueden cambiar significativamente.

- Clarificación de Requisitos:

Permiten a los equipos de desarrollo y a las partes interesadas clarificar y acordar los requisitos del proyecto de manera visual y tangible. Esto reduce el riesgo de malentendidos y asegura que todos los involucrados compartan una visión común del producto final.

El uso de prototipos de baja fidelidad es una estrategia esencial en el diseño y desarrollo de aplicaciones web y móviles. Facilitan la iteración rápida, la recolección de feedback valioso, y aseguran que los aspectos fundamentales de la aplicación se aborden desde el principio. Esto resulta en un proceso de desarrollo más eficiente y en productos finales que satisfacen mejor las necesidades de los usuarios y las partes interesadas. A continuación, las figuras de baja fidelidad del prototipo de software. (Snyder, 2003)

Figura 8

Prototipo de baja fidelidad

La figura muestra el diseño de la pantalla de inicio de sesión de la aplicación prototipo web para la gestión integral de restaurantes. Esta pantalla es fundamental para garantizar la seguridad y el acceso controlado a la plataforma, a continuación, se detalla cada componente de la interfaz:

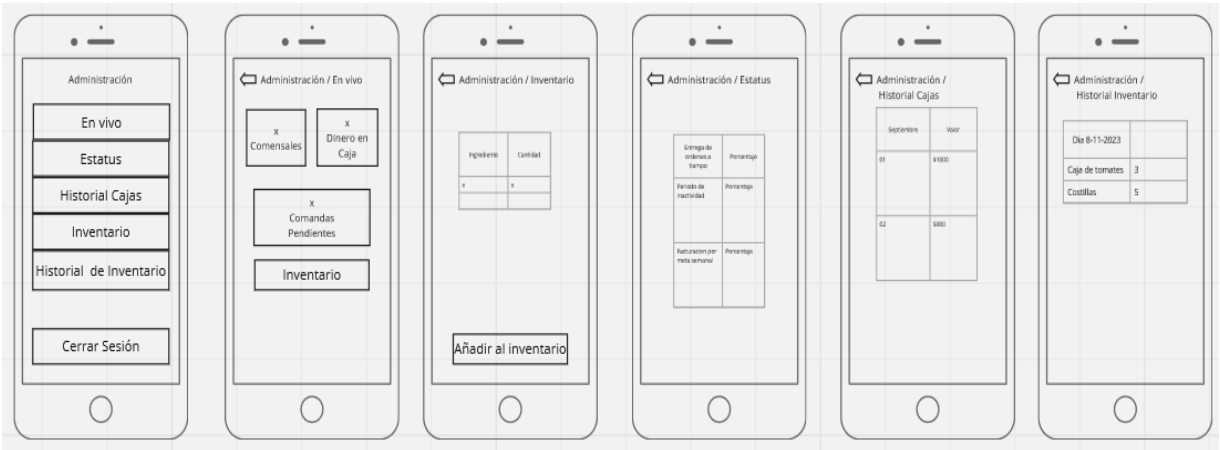
- Campo de Usuario: El campo superior está destinado a la entrada del nombre de usuario o dirección de correo electrónico del usuario. Este campo es esencial para identificar a los usuarios registrados y vincular sus credenciales con sus perfiles en el sistema.

- Campo de Contraseña: Justo debajo del campo de usuario, se encuentra el campo para la contraseña. Este campo permite a los usuarios ingresar sus contraseñas seguras, necesarias para autenticar su identidad y acceder a la plataforma.
- Enlace de Recuperación de Contraseña: Debajo de los campos de usuario y contraseña, se incluye un enlace con el texto "¿Olvidó su contraseña?". Este enlace es crucial para la funcionalidad de recuperación de contraseña, ofreciendo a los usuarios una manera sencilla de recuperar el acceso en caso de olvidar sus credenciales. Al hacer clic en este enlace, los usuarios son redirigidos a un flujo de recuperación de contraseña, que generalmente implica la verificación de identidad a través del correo electrónico o preguntas de seguridad.

El diseño de esta pantalla sigue principios de usabilidad y accesibilidad, asegurando que los usuarios puedan iniciar sesión de manera rápida y segura. La disposición vertical de los campos de entrada y el enlace de recuperación de contraseña facilita una navegación intuitiva, especialmente en dispositivos móviles. Además, el uso de etiquetas claras y concisas ayuda a minimizar la confusión y los errores de entrada, mejorando la experiencia del usuario.

Figura 9

Prototipo de baja fidelidad



La figura presenta una serie de pantallas de la interfaz de usuario de la aplicación prototipo web para la gestión integral de restaurantes. Estas pantallas reflejan diferentes funcionalidades clave de la aplicación, diseñadas para optimizar la administración y operación de un restaurante. A continuación, se describe cada pantalla en detalle:

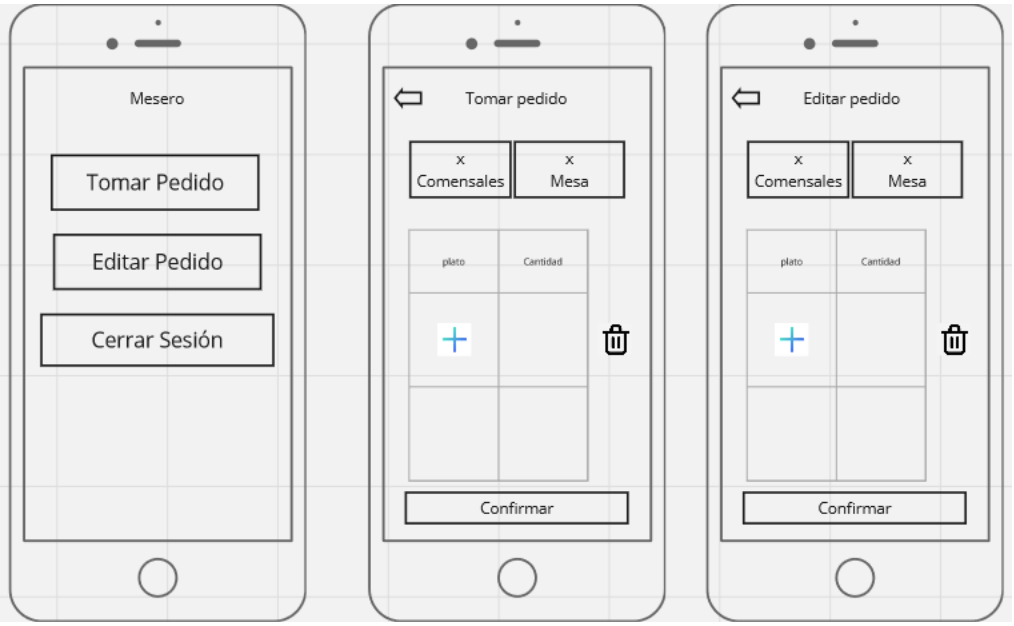
- Pantalla 1: Menú Principal de Administración
    a. Componentes: La primera pantalla muestra el menú principal de administración, que incluye opciones como "En vivo", "Estatus", "Historial Cajas", "Inventario", "Historial de Inventario" y "Cerrar Sesión".
    b. Funcionalidad: Este menú permite a los administradores acceder rápidamente a diversas secciones de la aplicación, facilitando la navegación y gestión de las operaciones del restaurante.
- Pantalla 2: Administración/En Vivo
    a. Componentes: La segunda pantalla muestra la sección "En vivo", con opciones para ver el estado de las comandas en curso, los ingresos en caja y el inventario disponible.
    b. Funcionalidad: Esta pantalla proporciona una vista en tiempo real de las operaciones del restaurante, ayudando a los administradores a tomar decisiones informadas rápidamente.
- Pantalla 3: Administración/Inventario
    a. Componentes: La tercera pantalla muestra el módulo de gestión de inventario, con opciones para ver y añadir elementos al inventario.
    b. Funcionalidad: Los administradores pueden gestionar y actualizar el inventario de manera eficiente, asegurando que siempre haya suficiente stock de los productos necesarios.
- Pantalla 4: Administración/Estatus
    a. Componentes: La cuarta pantalla presenta una vista de "Estatus", mostrando información sobre los pedidos en curso y su estado actual.

b. Funcionalidad: Esta pantalla permite a los administradores y personal del restaurante monitorear el progreso de los pedidos, asegurando una entrega oportuna y eficiente.

- Pantalla 5: Administración/Historial Cajas
  a. Componentes: La quinta pantalla muestra el historial de transacciones en caja, con detalles sobre las ventas realizadas en diferentes días.
  b. Funcionalidad: Los administradores pueden revisar y analizar el historial de ventas, lo que es crucial para la contabilidad y la planificación financiera.
- Pantalla 6: Administración/Historial de Inventario
  a. Componentes: La sexta pantalla muestra el historial de cambios en el inventario, con detalles sobre los productos añadidos o retirados en fechas específicas.
  b. Funcionalidad: Este módulo permite un seguimiento detallado de los movimientos de inventario, ayudando a prevenir pérdidas y gestionar el stock de manera más efectiva.

Figura 10

Prototipo de baja fidelidad



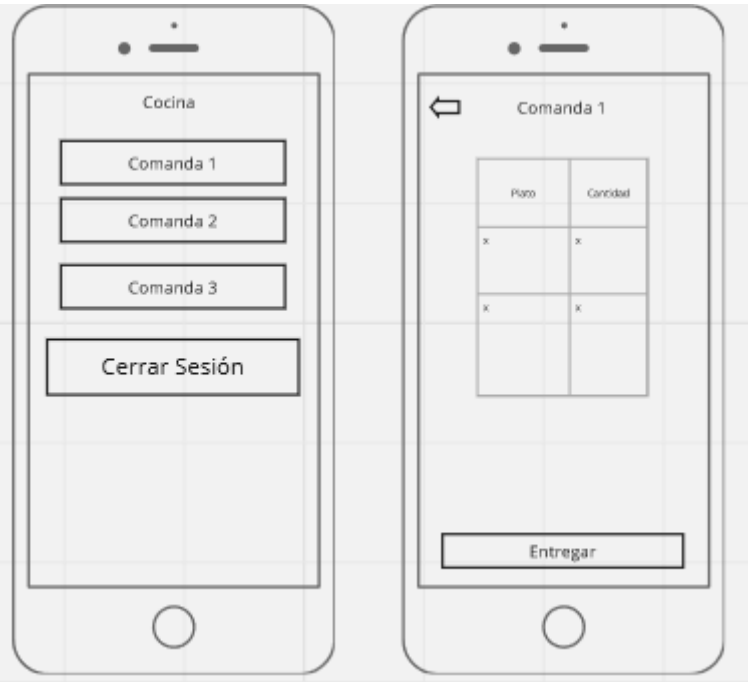
La figura presenta una serie de pantallas de la interfaz de usuario diseñadas específicamente para el mesero en la aplicación prototipo web para la gestión integral de restaurantes. Estas pantallas permiten a los meseros tomar y gestionar pedidos de manera eficiente, a continuación, se describe cada pantalla en detalle:

- Pantalla 1: Menú Principal del Mesero
    a. Componentes: La primera pantalla muestra el menú principal del mesero, que incluye opciones como "Tomar Pedido", "Editar Pedido" y "Cerrar Sesión".
    b. Funcionalidad: Este menú permite a los meseros acceder rápidamente a las funciones principales necesarias para gestionar los pedidos de los clientes. La opción de cerrar sesión asegura que las cuentas de los meseros se mantengan seguras cuando no están en uso.
- Pantalla 2: Tomar Pedido
    a. Componentes: La segunda pantalla muestra la interfaz para tomar un pedido. Incluye campos para ingresar el número de comensales y el número de mesa, así como una lista editable donde se pueden añadir platos y cantidades.
    b. Funcionalidad: Los meseros pueden añadir platos al pedido utilizando el botón de añadir (representado por el signo "+") y eliminar elementos utilizando el icono de la papelera. Una vez que se ha completado el pedido, pueden confirmarlo mediante el botón "Confirmar".
- Pantalla 3: Editar Pedido
    a. Componentes: La tercera pantalla es similar a la de tomar pedido, pero está diseñada para editar un pedido existente. Los campos de comensales y mesa se mantienen, y los platos y cantidades pueden ser modificados.
    b. Funcionalidad: Esta pantalla permite a los meseros ajustar los pedidos existentes de acuerdo a las solicitudes de los clientes, asegurando que cualquier cambio o adición se refleje en el sistema de manera precisa y rápida.

Al igual que en la pantalla de toma de pedidos, se pueden añadir y eliminar elementos, y el pedido modificado se confirma con el botón "Confirmar".

Figura 11

Prototipo de baja fidelidad



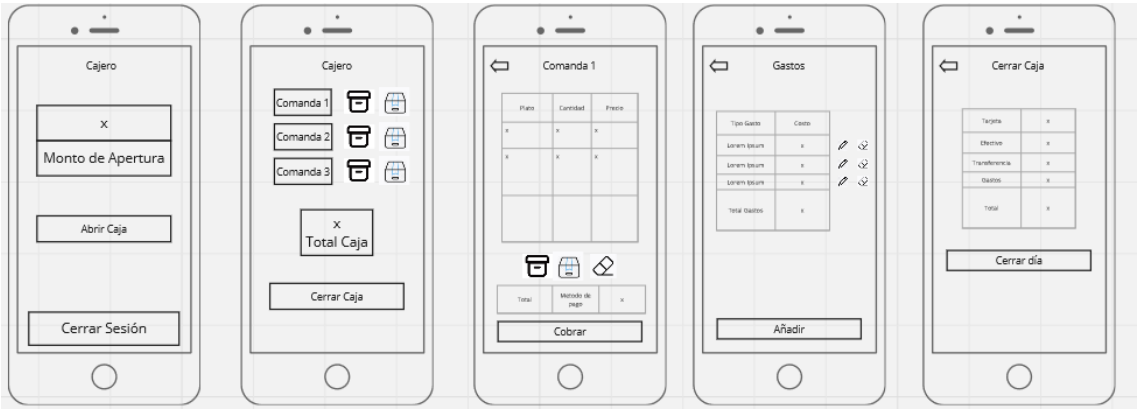
La figura presenta la interfaz de usuario diseñada para el personal de cocina, permitiendo una gestión eficaz de las comandas. A continuación, se describe cada pantalla:

- Pantalla de Inicio para Cocina:
  a. Componentes: La pantalla incluye botones para "Comanda 1", "Comanda 2", "Comanda 3" y "Cerrar Sesión".
  b. Funcionalidad: El personal de cocina puede acceder a las diferentes comandas en curso y cerrar sesión al finalizar su turno. Esta disposición permite un acceso directo a las comandas que requieren atención.
- Pantalla de Detalle de Comanda:
  a. Componentes: Muestra los detalles de una comanda específica, incluyendo los platos y cantidades.

b. Funcionalidad: Permite al personal de cocina ver el detalle de cada pedido y marcar los platos como entregados una vez que están listos para ser servidos. Un botón de "Entregar" confirma la finalización de la preparación de la comanda.

Figura 12

Prototipo de baja fidelidad cajero



La figura presenta la interfaz de usuario diseñada específicamente para los cajeros, facilitando la gestión de transacciones y la caja del restaurante. A continuación, se describe cada pantalla:

- Pantalla 1: Inicio de Turno del Cajero
  a. Componentes: Incluye un campo para ingresar el "Monto de Apertura" de la caja, un botón para "Abrir Caja" y una opción para "Cerrar Sesión".
  b. Funcionalidad: Esta pantalla permite al cajero iniciar su turno ingresando el monto inicial de efectivo en la caja. La opción de cerrar sesión asegura que solo el personal autorizado tenga acceso a las funciones de la caja.
- Pantalla 2: Gestión de Comandas y Caja
  a. Componentes: Muestra las comandas activas (Comanda 1, Comanda 2, Comanda 3), un campo que muestra el "Total Caja" y un botón para "Cerrar Caja".

b. Funcionalidad: El cajero puede ver y seleccionar las comandas activas para procesarlas. También puede monitorear el total acumulado en la caja y cerrar la caja al finalizar el turno o al final del día.

- Pantalla 3: Detalle de Comanda
  a. Componentes: Presenta el detalle de una comanda específica, con los platos y cantidades, y un botón para "Cobrar".
  b. Funcionalidad: Permite al cajero revisar los detalles de la comanda y proceder al cobro, asegurando que los pedidos sean pagados correctamente. La opción de cobro facilita la transición de la comanda a una transacción finalizada.
- Pantalla 4: Gestión de Gastos
  a. Componentes: Incluye una lista de gastos registrados, con campos para descripción y monto, y un botón para "Añadir" nuevos gastos.
  b. Funcionalidad: Esta pantalla permite al cajero registrar y monitorear los gastos realizados durante el turno, asegurando una contabilidad precisa y transparente de los costos operativos.
- Pantalla 5: Cierre de Caja
  a. Componentes: Muestra un resumen de la caja con detalles de ingresos, egresos y el total final, junto con un botón para "Cerrar Día".
  b. Funcionalidad: Permite al cajero realizar el cierre de caja, proporcionando un resumen detallado de las transacciones del día y asegurando que todos los registros estén completos antes de finalizar el turno.

La implementación de estas interfaces responde a la necesidad de un manejo integrado y simplificado de las tareas diarias, que incluyen la toma y edición de pedidos, la gestión del inventario, el control de las comandas en la cocina y la administración de las transacciones en caja. La alta fidelidad de estas figuras permite una representación detallada y precisa del aspecto final de la aplicación, proporcionando una visión clara de cómo interactuarán los usuarios con el sistema. (Snyder, 2003)

A continuación, se presentan las figuras de alta fidelidad para las diferentes secciones de la aplicación, cada una acompañada de una descripción detallada de su diseño y funcionalidad. Estas figuras son el resultado de un proceso iterativo de diseño centrado en el usuario, que incorpora feedback continuo de las partes interesadas para garantizar que la aplicación satisfaga las necesidades específicas de los administradores, meseros, cocineros y cajeros.

Figura 13

Prototipo Administrador

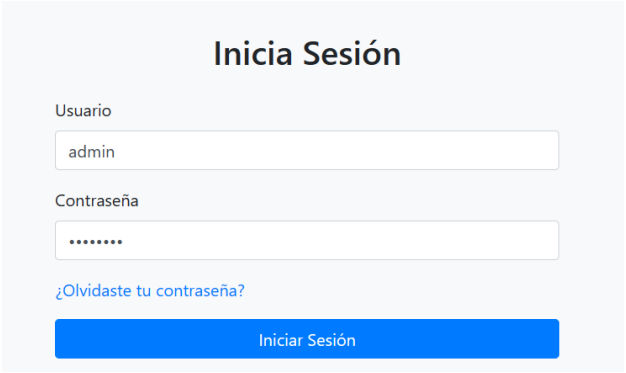


Figura 14

Panel de Administración

Un panel administrativo en un entorno del restaurante es una herramienta esencial que facilita la gestión y el control de las operaciones diarias. Este panel tiene cinco funciones principales, cada una diseñada para proporcionar información clave y mejorar la eficiencia operativa, a continuación, se describen estas funciones.

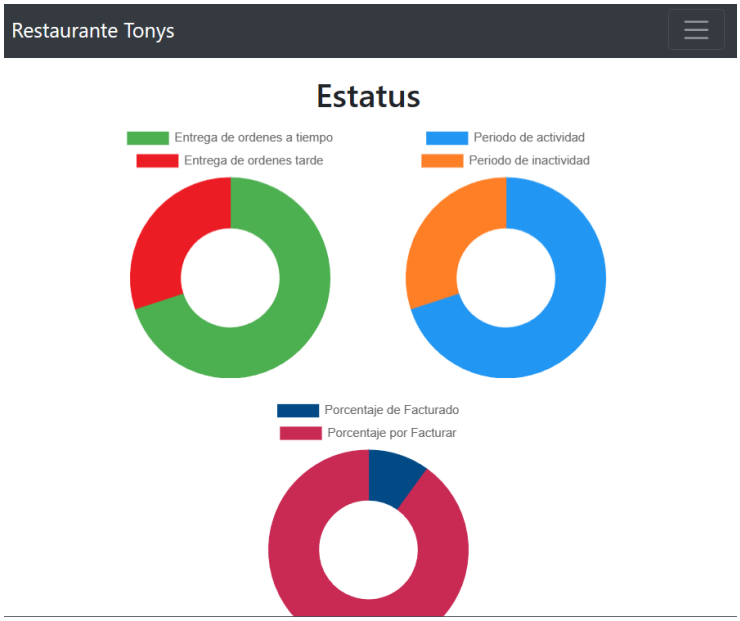
Figura 15

Estadísticas en vivo Administración

En la opción de **En Vivo,** permite visualizar en tiempo real el estado actual del restaurante. Los administradores pueden ver el flujo de comensales, el total del dinero en la caja, el estado de los pedidos, lo que facilita a la toma de decisiones rápidas y eficientes para mejorar el servicio y la experiencia del cliente, como por ejemplo llamar a más camareros o cocineros si amerita la ocasión, como en el día de las madres, del padre o del trabajador.

Figura 16

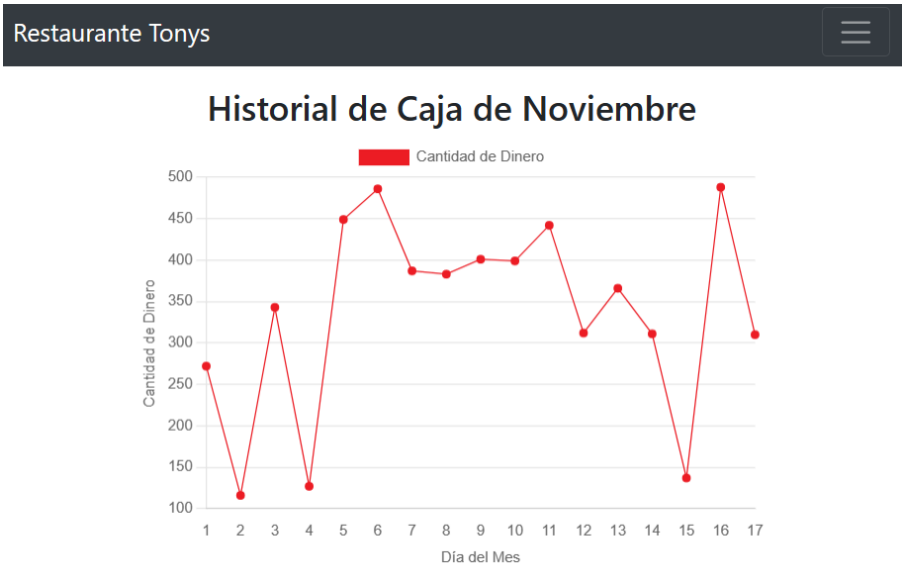
Estatus Administración



En la sección del panel de estatus, se muestra los indicadores clave de rendimiento (KPIs) de los cuales se definen 3, el porcentaje de una entrega de órdenes a tiempo, el porcentaje del periodo de actividad e inactividad y el porcentaje de facturación semanal.

Figura 17

Historial Administración



El Historial Caja muestra revisar los totales de las cajas pasadas en base a los días del mes. Con esto se sabe cuándo hay un buen mes de venta y cuando es uno más bajo. Dando así a tomar decisiones a futuro para sobrellevar esos meses.

Figura 18

Inventario Administración

Figura 19

Registro de Inventario Administración

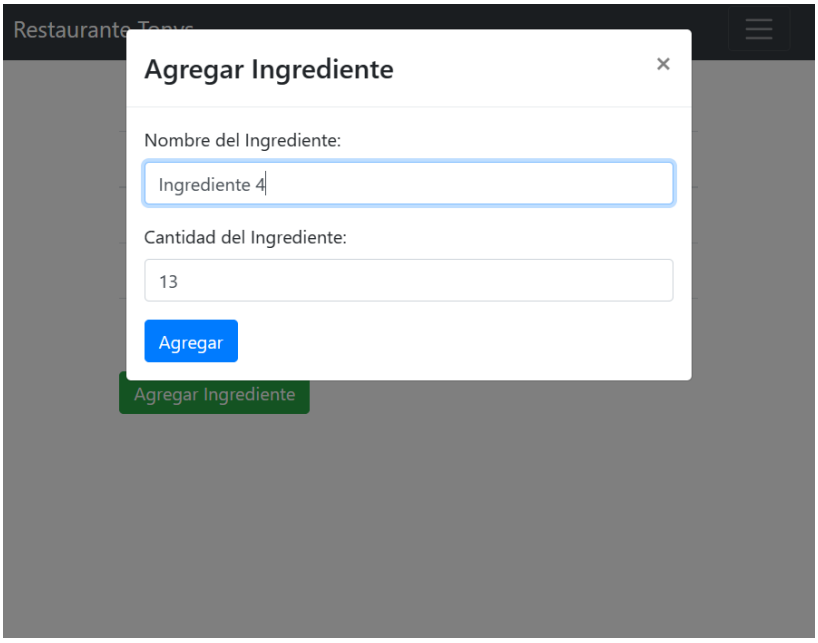


Figura 20

Resultado de Inventario Administración

Para el inventario se realizó un requerimiento diferente a los sistemas tradicionales de inventario, este panel no reduce automáticamente las cantidades en función de las ventas, ya que esto permite a los administradores realizar un seguimiento preciso de los niveles de inventario diarios y ajustar manualmente según sea necesario.

Figura 21

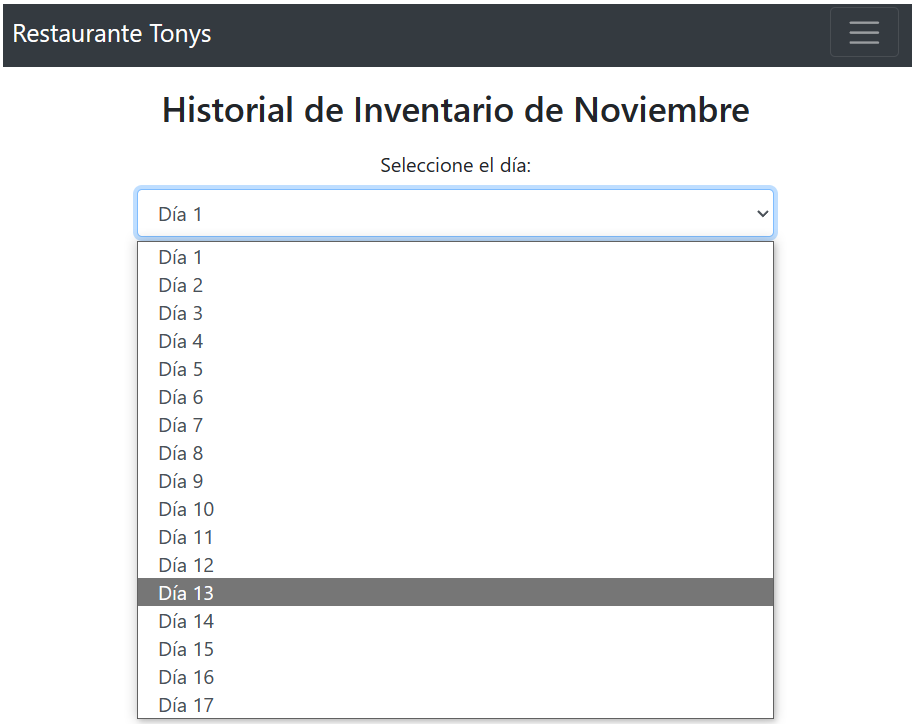
Historial de Inventario Administración

La figura muestra la interfaz de la sección "Historial de Inventario de Noviembre" de la aplicación prototipo web para la gestión integral de restaurantes. En la parte superior, se

encuentra el nombre del restaurante, "Restaurante Tonys", seguido del título de la sección. Debajo del título, un menú desplegable permite seleccionar el día específico del mes de noviembre para el cual se desea consultar el inventario. Una vez seleccionado el día, se presenta una "Tabla de Cantidades por Ingrediente" que muestra dos columnas: "Ingrediente" y "Cantidad". En el ejemplo proporcionado, se muestran las cantidades de dos ingredientes: 44 cajas de tomates y 38 chuletas. Esta pantalla es fundamental para que los administradores puedan revisar y controlar el historial del inventario de manera diaria, facilitando la toma de decisiones informadas sobre las compras y el uso de los ingredientes en el restaurante.

Figura 22

Historial de Inventario Administración



Las tres imágenes presentadas corresponden a la interfaz de usuario de la sección "Historial de Inventario de Noviembre" de la aplicación prototipo web para la gestión integral de restaurantes del "Restaurante Tonys". Esta interfaz permite a los administradores del restaurante revisar y gestionar el inventario de ingredientes de manera diaria durante el mes de noviembre.

- Imagen 1: Selección del Día

    a. Descripción: La primera imagen muestra el menú desplegable para seleccionar un día específico del mes de noviembre. Este menú desplegable ofrece una lista de los días del mes, permitiendo al usuario elegir el día para el cual desea consultar el historial de inventario.

    b. Funcionalidad: Facilita la navegación y la selección rápida del día específico, proporcionando flexibilidad en la consulta del historial diario de inventario.

- Imagen 2: Visualización del Inventario para el Día Seleccionado

    a. Descripción: La segunda imagen muestra la interfaz después de seleccionar el "Día 13". La "Tabla de Cantidades por Ingrediente" se actualiza para reflejar los datos del inventario del día seleccionado, mostrando los ingredientes y sus cantidades correspondientes.

    b. Funcionalidad: Proporciona una visión detallada y específica del inventario disponible para el día seleccionado, permitiendo una gestión precisa y eficiente de los recursos.

- Imagen 3: Interfaz Completa con Opciones Adicionales

    a. Descripción: La tercera imagen incluye un menú adicional en la parte superior izquierda, con opciones para acceder al "Panel de Administración" y "Cerrar Sesión". Este menú permite a los administradores navegar fácilmente a otras secciones de la aplicación o cerrar sesión cuando sea necesario.

    b. Funcionalidad: Añade funcionalidad adicional de navegación, mejorando la accesibilidad y la usabilidad de la aplicación al permitir a los usuarios moverse entre diferentes secciones del sistema sin problemas.

Figura 23

Prototipo Mesero

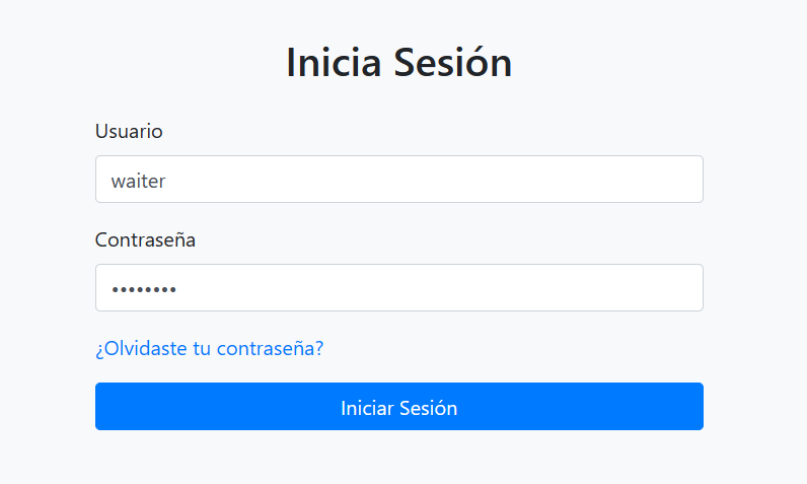


Figura 24

Panel Mesero

Figura 25

Panel Mesero Opciones

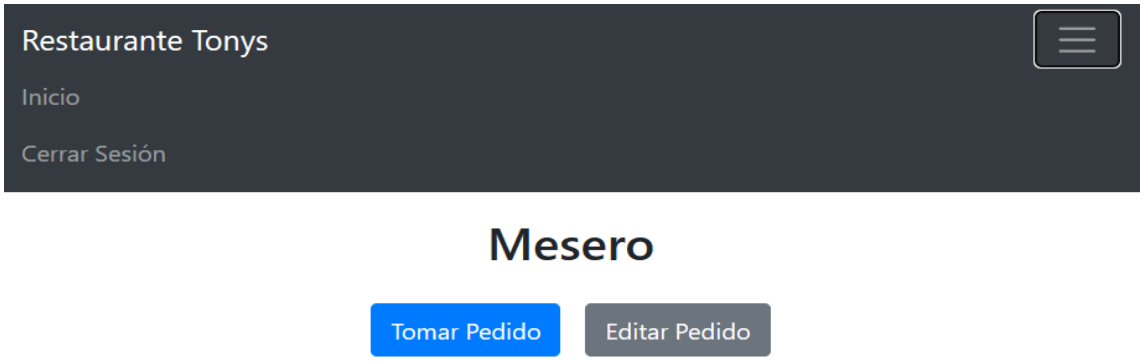


Figura 26

Prototipo Administrador Toma de Pedido

Figura 27

Prototipo Administrador Editar de Pedido

Figura 28

Prototipo Cocina

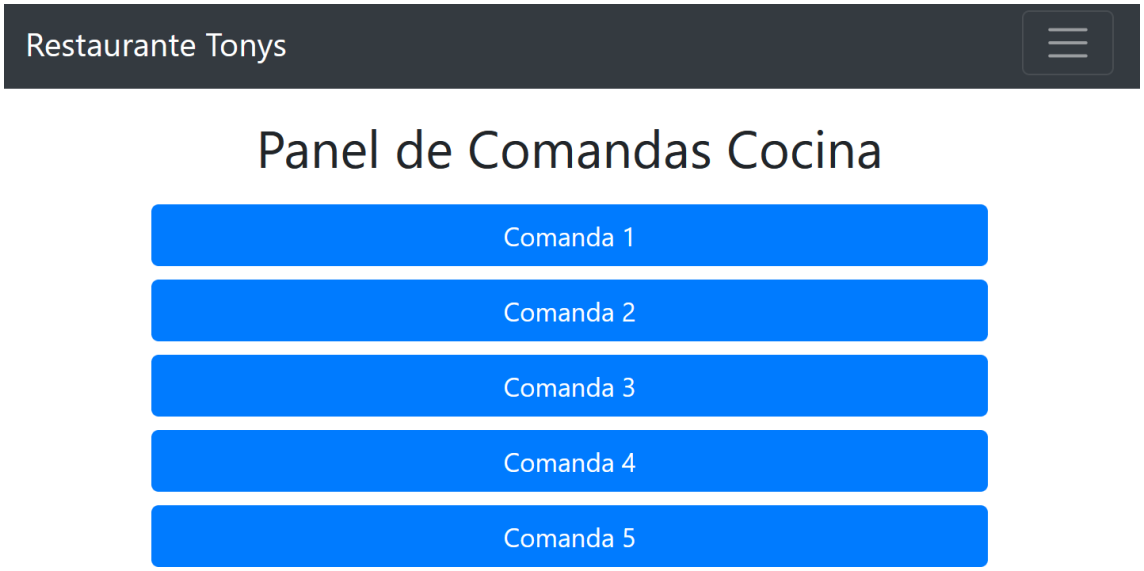


Figura 29

Panel de Comandas Mesa 1

Figura 30

Prototipo Cajero

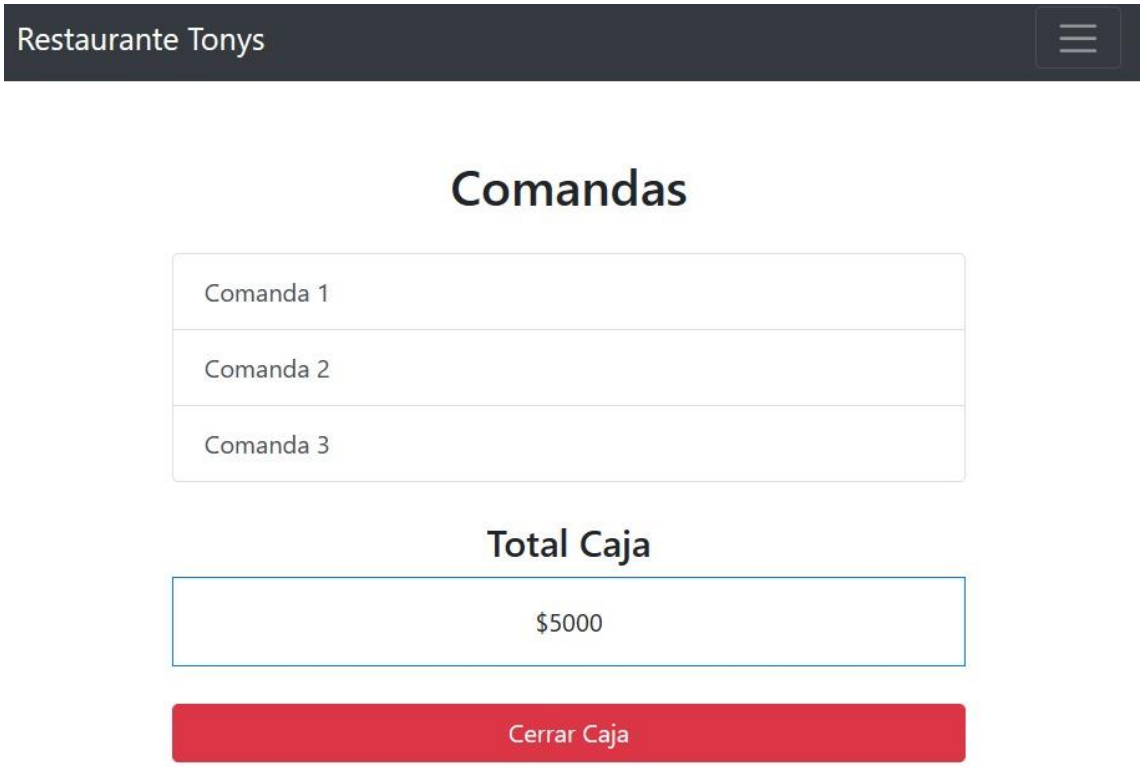


Figura 31

Panel de Comanda

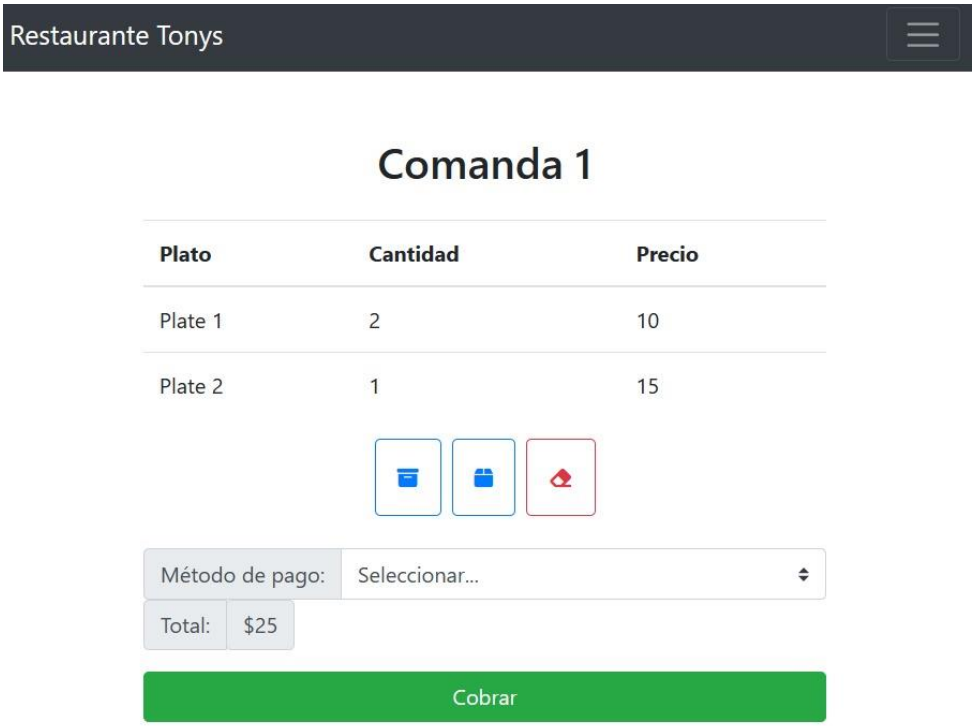


Figura 32

Panel de Comanda Pago

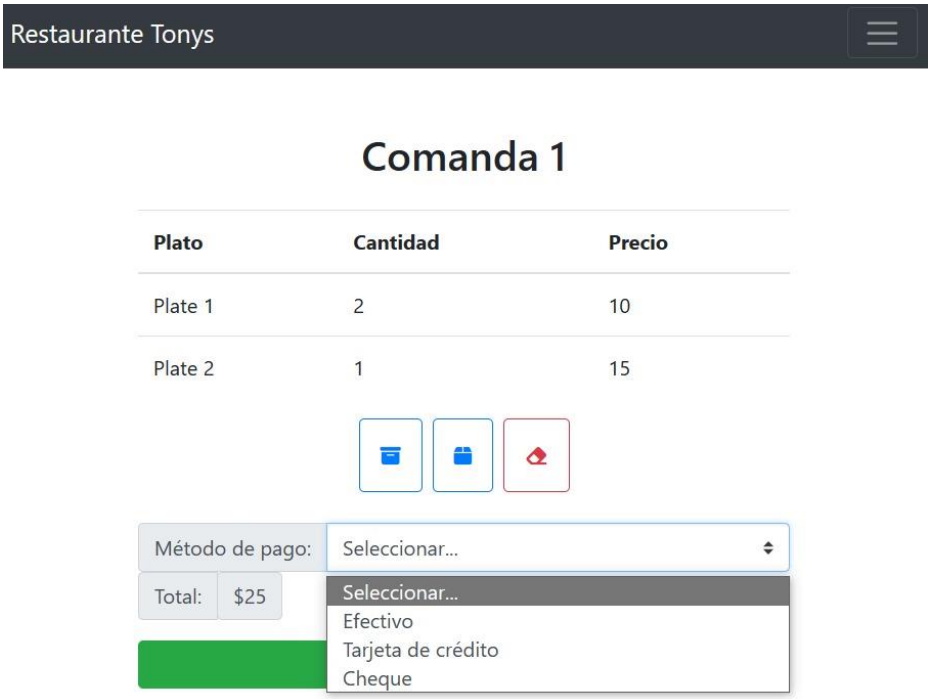


Figura 33

Panel de Comanda Cobro

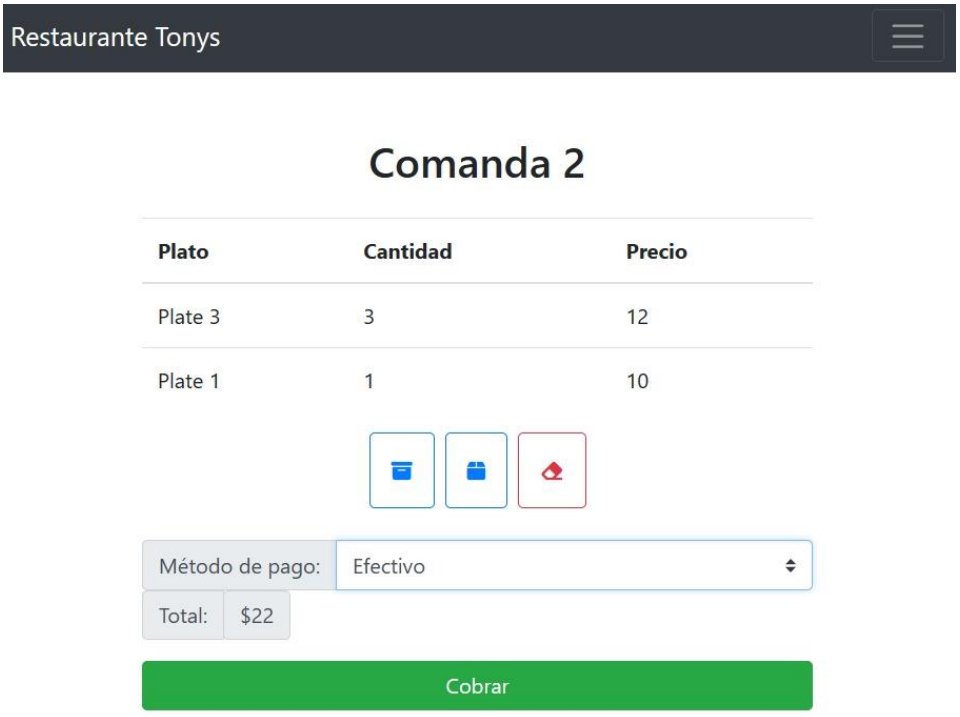


Figura 34

Panel de Comanda Cierre de Caja

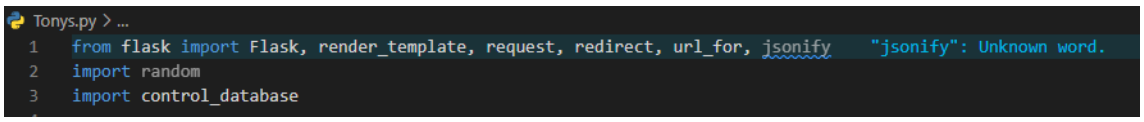
3.3.4 Etapa Codificación

En la etapa de codificación se presenta las pantallas de la programación del desarrollo del software, mediante el Desarrollo con Python y Flask Diseño y Construcción de base de datos.

- Desarrollo con Python y Flask

Figura 35

Importación de librerías utilizadas



Los siguientes códigos están comentados por uso, todos estos van a pertenecer al archivo TONYS.PY

Figura 36

Códigos

```
app = Flask(__name__)

# Instanciamos el control de la base de datos    "Instanciamos": Unknown word.
control = control_database.controldb()    "controldb": Unknown word.
control.preparacion_db()    "preparacion": Unknown word.
```

Figura 37

Código Inicio de Sesión

```
##########################################
# Sección: Inicio de sesión    "Sección": Unknown word.
##########################################

# Valida el inicio de sesión y redirecciona páginas según el rol
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        rol = control.autenticacion(username, password)
        if rol is not False:
            return redirect(url_for(rol))
    return render_template('login.html')
```

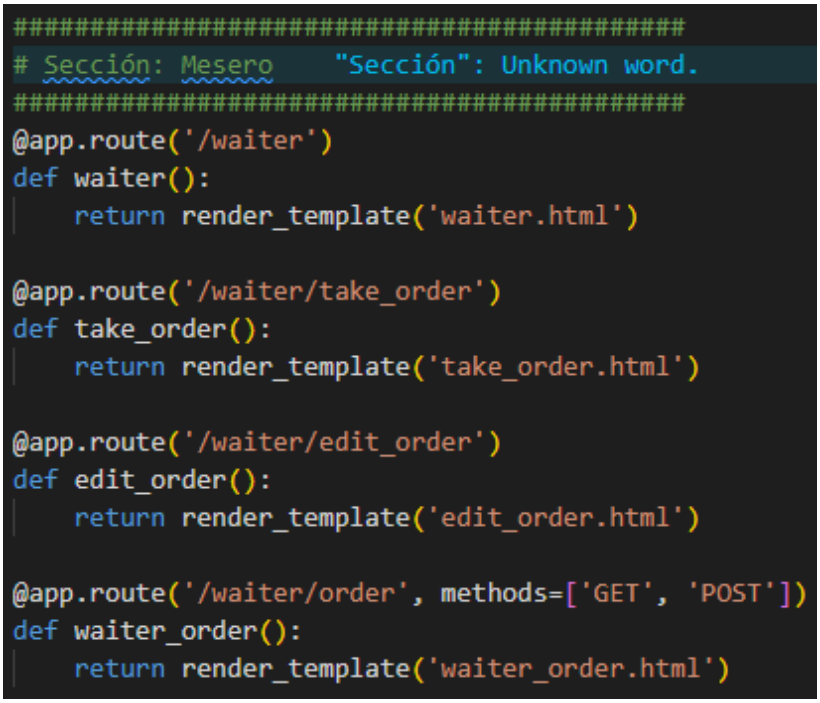
Figura 38

Código Administración

```
##########################################
# Sección: Administración    "Sección": Unknown word.
##########################################

# Panel de administración    "administración": Unknown word.
@app.route('/administration')
def administration():
    return render_template('administration.html')

# En vivo
@app.route('/administration/live')
def administration_live():
    # Solicitamos los datos para las estadísticas en vivo    "Solicitamos": Unknown
    estadisticas = control.get_envivo()    "estadisticas": Unknown word.
    return render_template('administration_live.html', estadisticas=estadisticas)

# Estatus    "Estatus": Unknown word.
@app.route('/administration/status')
def administration_status():
    # Solicitamos los datos para el estatus    "Solicitamos": Unknown word.
    estatus = control.get_status()    "estatus": Unknown word.
    return render_template("administration_status.html", estatus=estatus)    "estatu

# Inventario    "Inventario": Unknown word.
@app.route('/administration/inventory', methods=['GET', 'POST'])
def administration_inventory():
    # Solicitamos los datos de inventario    "Solicitamos": Unknown word.
    inventario = control.get_inventario()    "inventario": Unknown word.

    # Permite agregar ingredientes    "Permite": Unknown word.
    if request.method == 'POST':
        ingrediente = request.form['ingrediente']    "ingrediente": Unknown word.
        cantidad = request.form['cantidad']    "cantidad": Unknown word.
        control.agregar_ingrediente(ingrediente, cantidad)    "agregar": Unknown wo
        inventario = control.get_inventario()    "inventario": Unknown word.

    return render_template('administration_inventory.html', inventario=inventario)
```

Figura 39

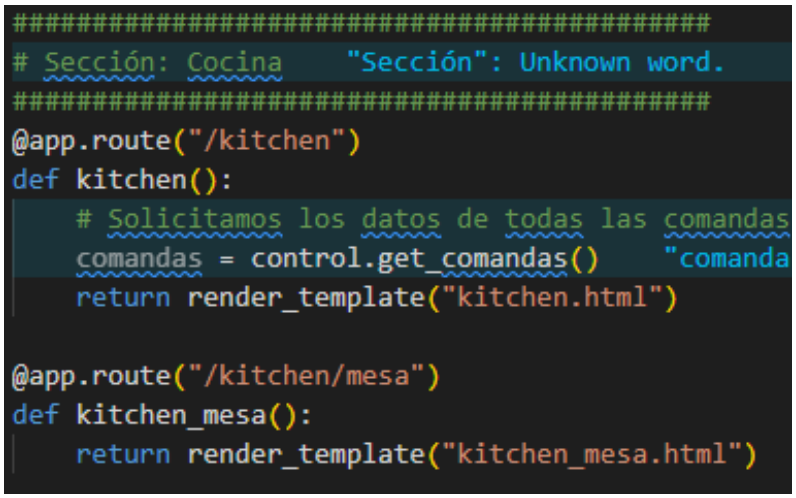
Código Mesero



```
##########################################
# Sección: Mesero    "Sección": Unknown word.
##########################################
@app.route('/waiter')
def waiter():
    return render_template('waiter.html')

@app.route('/waiter/take_order')
def take_order():
    return render_template('take_order.html')

@app.route('/waiter/edit_order')
def edit_order():
    return render_template('edit_order.html')

@app.route('/waiter/order', methods=['GET', 'POST'])
def waiter_order():
    return render_template('waiter_order.html')
```

Figura 40

Código Cocina



```
##########################################
# Sección: Cocina    "Sección": Unknown word.
##########################################
@app.route("/kitchen")
def kitchen():
    # Solicitamos los datos de todas las comandas
    comandas = control.get_comandas()    "comanda
    return render_template("kitchen.html")

@app.route("/kitchen/mesa")
def kitchen_mesa():
    return render_template("kitchen_mesa.html")
```

Figura 41

Código Caja

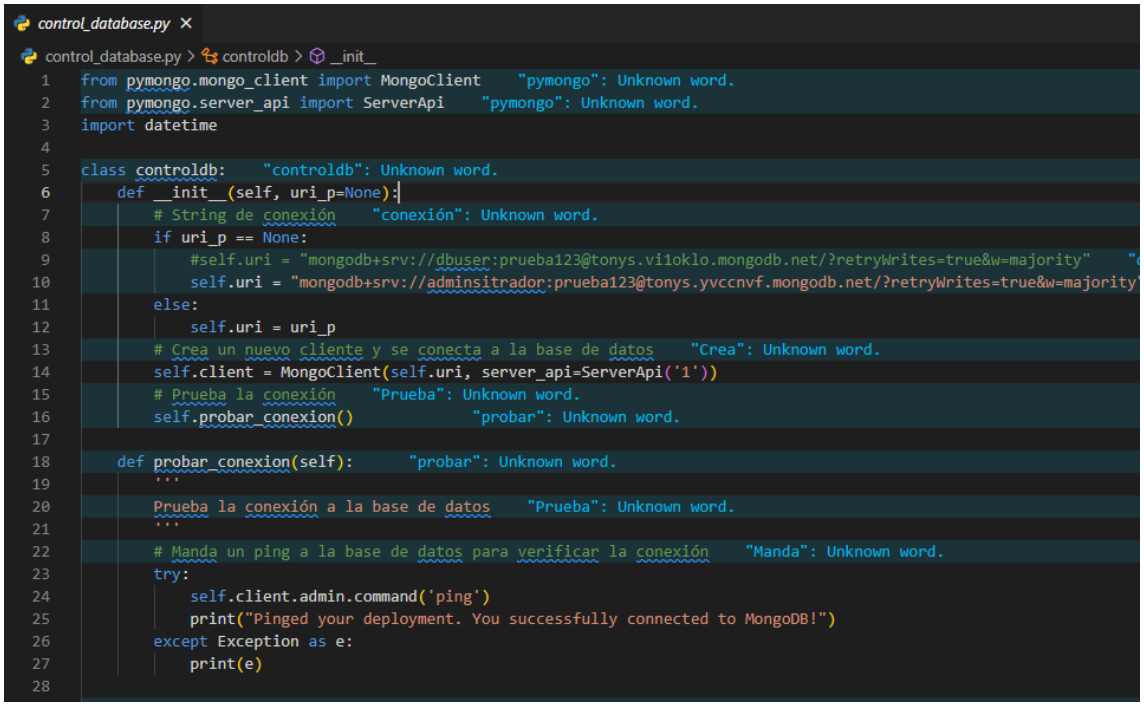
```
##########################################
# Sección: Caja    "Sección": Unknown word.
##########################################

# Página de Monto de apertura    "Página": Unknown word.
@app.route("/cashier")
def cashier():
    return render_template("cashier.html")

# Proceso la cantidad de monto de apertura    "Proceso": Unknown word.
@app.route("/process_opening", methods=["POST"])
def process_opening():
    opening_amount = request.form.get("openingAmount")

    return "Opening amount processed successfully"

# Muestra todas las comandas registradas    "Muestra": Unknown word.
@app.route("/cashier/commands")
def commands():
    # Solicitamos las comandas existentes    "Solicitamos": Unknown word.
    id_comandas = control.get_idcomandas()        "idcomandas": Unknown word.
    return render_template("commands.html", id_comandas=id_comandas)

# Muestra los detalles de una comanda y registra el pago    "Muestra": Unknown word
@app.route("/commands/<comanda_id>")    "comanda": Unknown word.
def comanda(comanda_id):        "comanda": Unknown word.
    # Solicitamos los detalles    "detalles": Unknown word.
    comandas = control.get_comandasdetails(comanda_id)    "comandasdetails": Unknow
    if comandas is None:
        return render_template("comanda_not_found.html")
    return render_template("comanda_details.html", comanda_details=comandas)

# Procesa y registra el pago    "Procesa": Unknown word.
@app.route("/process_payment/<comanda_id>", methods=['POST'])
def process_payment(comanda_id):
    payment_method = request.form.get('payment_method')
    control.procesar_pago(comanda_id, payment_method)    "procesar": Unknown word.
    return redirect(url_for('commands'))
```

```python
# Cierra la caja    "Cierra": Unknown word.
@app.route("/cashier/close")
def close():
    # Solicitamos todos los pagos registrados    "todos": Unknown word.
    pagos = control.get_pagos()    "pagos": Unknown word.

    # Inicializa totales
    total_card = 0
    total_cash = 0
    total_transfer = 0
    total_expenses = 0

    # Recorrer los pagos y actualizar los totales
    for pago in pagos:
        payment_method = pago.get("payment_method")
        total_amount = pago.get("total")

        if payment_method == "tarjeta":
            total_card += total_amount
        elif payment_method == "efectivo":
            total_cash += total_amount
        elif payment_method == "transferencia":
            total_transfer += total_amount
        elif payment_method == "gastos":
            total_expenses += total_amount

    # Calcula todo el total
    total_all = total_card + total_cash + total_transfer - total_expenses

    return render_template("close.html", total_card=total_card, total_cash=total_cash,
                           total_transfer=total_transfer, total_expenses=total_expenses,
                           total_all=total_all)

if __name__ == '__main__':
    app.run(debug=True)
```

Es importante resaltar que se realiza dos códigos aparte para tener una buena gestión de código, es así como una hace los procesos de cada uno de los usuarios, otra permite el control de los procesos de la base de datos.

Figura 42

Conexión a la Base de Datos

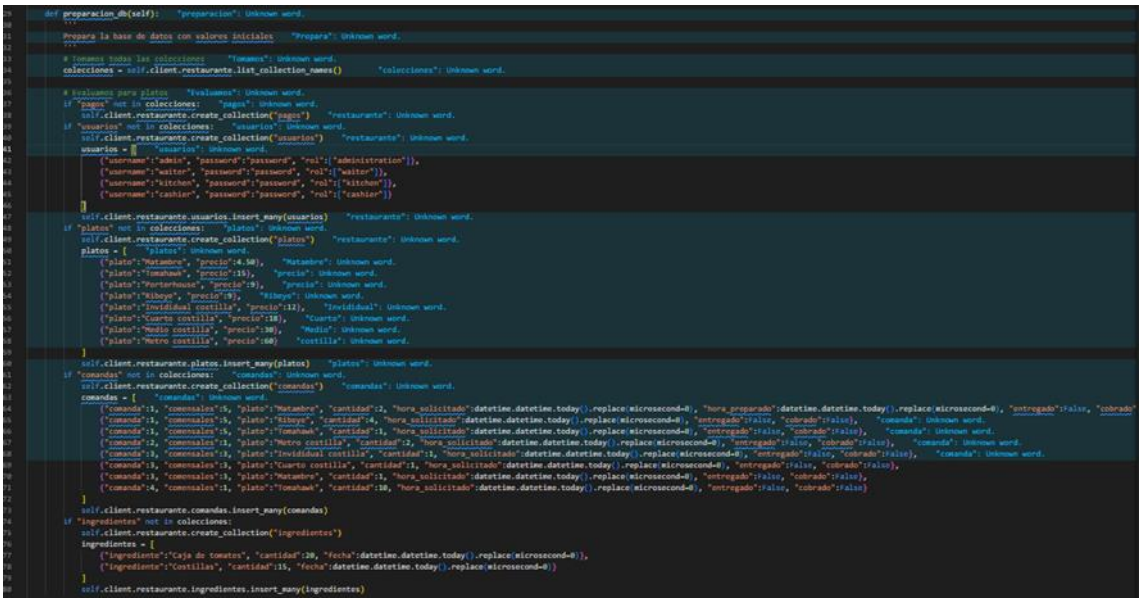


```python
# control_database.py
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
import datetime

class controldb:
    def __init__(self, uri_p=None):
        # String de conexión
        if uri_p == None:
            #self.uri = "mongodb+srv://dbuser:prueba123@tonys.vi1oklo.mongodb.net/?retryWrites=true&w=majority"
            self.uri = "mongodb+srv://adminsitrador:prueba123@tonys.yvccnvf.mongodb.net/?retryWrites=true&w=majority"
        else:
            self.uri = uri_p
        # Crea un nuevo cliente y se conecta a la base de datos
        self.client = MongoClient(self.uri, server_api=ServerApi('1'))
        # Prueba la conexión
        self.probar_conexion()

    def probar_conexion(self):
        '''
        Prueba la conexión a la base de datos
        '''
        # Manda un ping a la base de datos para verificar la conexión
        try:
            self.client.admin.command('ping')
            print("Pinged your deployment. You successfully connected to MongoDB!")
        except Exception as e:
            print(e)
```

En la función preparación, si no existen las colecciones en la base de datos se las crea y también se agregan datos esenciales, como, por ejemplo, los platos, ingrediente y usuarios, algunos datos de comandas para la versión demostrativa.

Figura 43

Conexión a la Base de Datos



```python
    def quediaeshoy(self):
        '''
        Retorna el día de hoy
        '''
        return datetime.datetime.today().replace(microsecond=0)
```

```python
def autenticacion(self, usuario, contraseña):
    '''
    Valida la autenticación y devuelve el rol del usuario
    '''
    user = self.client.restaurante.usuarios.find_one({'username': usuario, 'password': contraseña})
    if user is not None:
        return user.get("rol")[0]
    else:
        return False
```

```python
def get_envivo(self):
    '''
    Recupera los siguientes datos:
        - Cantidad de comensales
        - Cantidad de dinero en caja
        - Cantidad de comandas pendientes
        - Cantidad de dinero por cobrar
    '''
    # Total de comensales y dinero por cobrar
    comensales_cursor = self.client.restaurante.comandas.find({"cobrado":False})
    cantidad_comensales = 0
    comandas_consideradas = []
    for comensal in comensales_cursor:
        if comensal.get("comanda") not in comandas_consideradas:
            cantidad_comensales += comensal.get("comensales")
            comandas_consideradas.append(comensal.get("comanda"))
    # Dinero en caja (cobrado)
    pagos_cursor = self.client.restaurante.pagos.find()
    dinero_en_caja = 0
    if pagos_cursor is not None:
        for pago in pagos_cursor:
            dinero_en_caja += pago.get("total")
    # Comandas pendientes
    comandas_pendientes = self.client.restaurante.comandas.find({"entregado":False, "cobrado":False})

    # Devuelve lo requerido
    return {
        "cantidad_comensales":cantidad_comensales,
        "dinero_en_caja" : dinero_en_caja,
        "comandas_pendientes" : len(comandas_pendientes.distinct("comanda"))
    }
```

```python
def get_status(self):
    '''
    Devuelve el estatus de facturación, tiempo de actividad y entrega de ordenes a tiempo
    '''
    return {
        "entregas_bien" : 70,
        "entregas_mal" : 30
    }
```

```python
    def get_inventario(self):
        '''
        Devuelve todo el inventario de ingredientes
        '''
        inventario_cursor = self.client.restaurante.ingredientes.find()
        hoy = str(self.quediaeshoy()).split(" ")[0]
        inventario = {}
        for ingrediente in inventario_cursor:
            # Valida que sea el inventario actualizado del día
            if str(ingrediente.get("fecha")).split(" ")[0] == hoy:
                inventario[ingrediente.get("ingrediente")] = ingrediente.get("cantidad")
        return inventario
```

```python
    def agregar_ingrediente(self, ingrediente, cantidad):
        '''
        Agrega un nuevo ingrediente
        '''
        nuevo_ingrediente = {
            "ingrediente" : ingrediente,
            "cantidad" : cantidad,
            "fecha" : datetime.datetime.today().replace(microsecond=0)
        }
        self.client.restaurante.ingredientes.insert_one(nuevo_ingrediente)
```

```python
    def get_comandas(self):
        '''
        Retorna todas las comandas
        '''
        return ""
```

```python
    def get_idcomandas(self):
        '''
        Retorna todos los ids de las comandas
        '''
        return self.client.restaurante.comandas.distinct("comanda")

```

```python
    def get_comandasdetails(self, comanda_id):
        '''
        Retorna todos los detalles de las comandas
        '''
        comandas_cursor = self.client.restaurante.comandas.find({"comanda":int(comanda_id)})
        comandas = []
        for doc in comandas_cursor:
            precio_plato = self.client.restaurante.platos.find_one({"plato":doc.get("plato")})
            precio = precio_plato.get("precio",0)
            total_precio = doc.get("cantidad") * precio
            datos = {
                "comanda" : doc.get("comanda"),
                "plato" : doc.get("plato"),
                "cantidad" : doc.get("cantidad"),
                "precio" : total_precio
            }
            comandas.append(datos)

        total = 0
        for comanda in comandas:
            total += comanda["precio"]
        comandas.append(total)

        return comandas
```

```python
    def procesar_pago(self, id_comanda, payment_method):
        '''
        Procesa y registra el pago en la base de datos
        '''
        comandas_cursor = self.client.restaurante.comandas.find({"comanda":int(id_comanda)})
        total = 0
        # Permite calcular el total de la comanda (hay que mejorar, no esta optimizado)
        for doc in comandas_cursor:
            precio_plato = self.client.restaurante.platos.find_one({"plato":doc.get("plato")})
            precio = precio_plato.get("precio",0)
            total += doc.get("cantidad") * precio

        # Añade a la base de datos la información del pago
        payment_data = {
            "payment_method": payment_method,
            "total_amount": total
        }
        self.client.restaurante.pagos.insert_one(payment_data)

        # Editamos el estado de la comanda
        operacion = {
            "$set" : {
                "cobrado":True
            }
        }
        self.client.restaurante.comandas.update_many({"comanda":id_comanda}, operacion)
```

```python
    def get_pagos(self):
        '''
        Retorna todos los pagos
        '''
        return self.client.restaurante.pagos.find()
```

```
234    if __name__ == "__main__":
235        control = controldb()
236        control.eliminar_coleccion("comandas")
237        control.preparacion_db()
```

A continuación, se presenta las pantallas del contenido del aplicativo Web en HTML5 que funciona en un navegador web y utiliza las capacidades y características de HTML5, CSS y JavaScript para ofrecer una experiencia interactiva y funcional al usuario.

Figura 44

Login

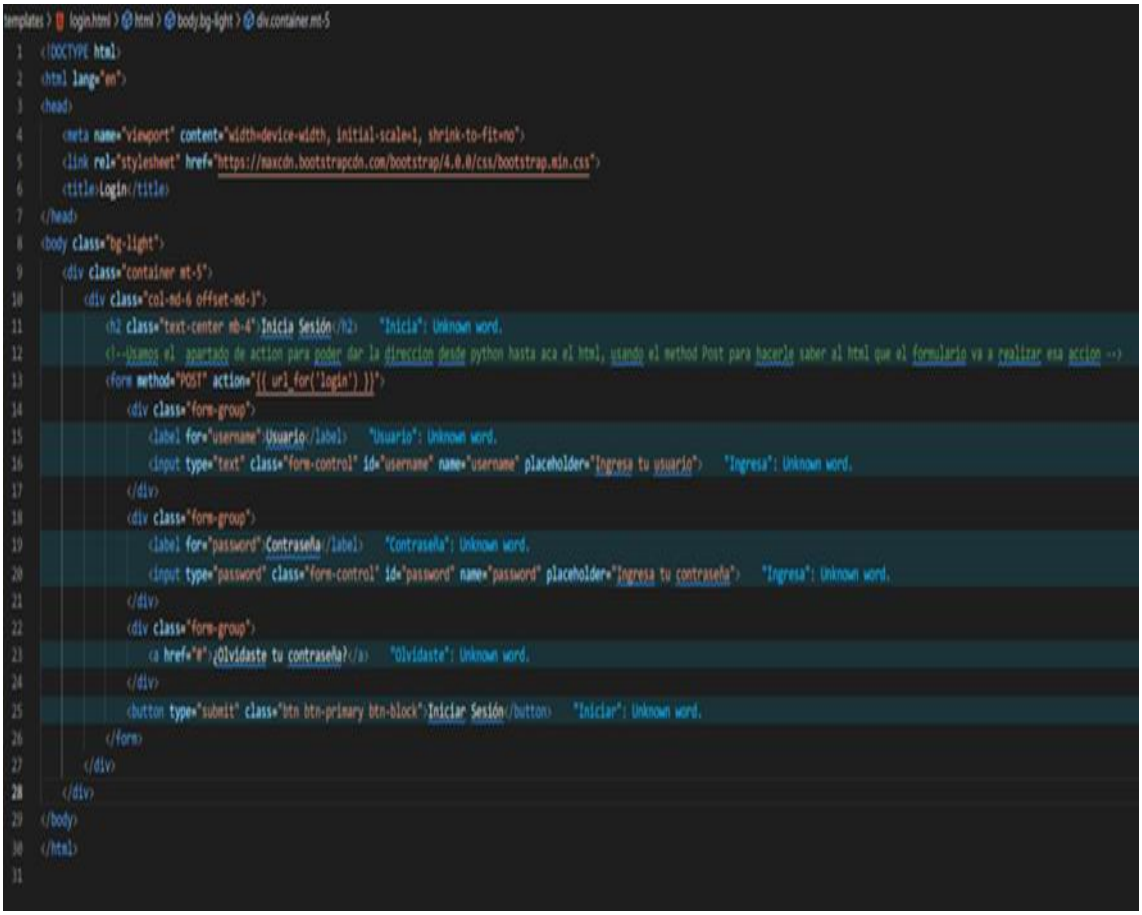
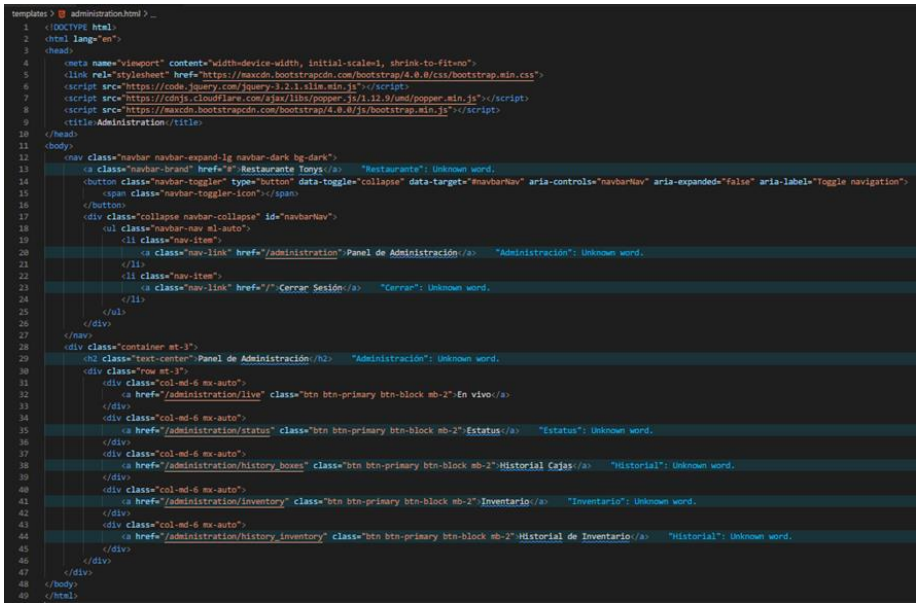


Figura 45

Administrador

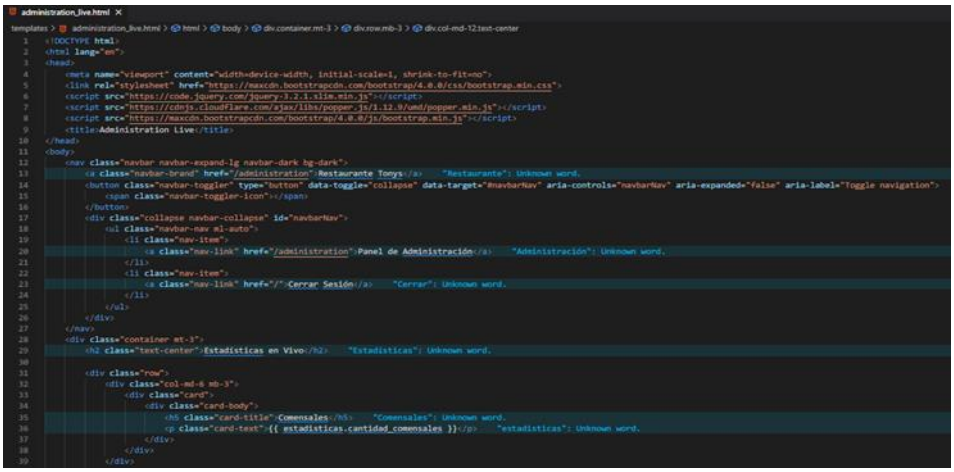


templates > administration.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Panel de Administración</h2>    "Administración": Unknown word.
        <div class="row mt-3">
            <div class="col-md-6 mx-auto">
                <a href="/administration/live" class="btn btn-primary btn-block mb-2">En vivo</a>
            </div>
            <div class="col-md-6 mx-auto">
                <a href="/administration/status" class="btn btn-primary btn-block mb-2">Estatus</a>    "Estatus": Unknown word.
            </div>
            <div class="col-md-6 mx-auto">
                <a href="/administration/history_boxes" class="btn btn-primary btn-block mb-2">Historial Cajas</a>    "Historial": Unknown word.
            </div>
            <div class="col-md-6 mx-auto">
                <a href="/administration/inventory" class="btn btn-primary btn-block mb-2">Inventario</a>    "Inventario": Unknown word.
            </div>
            <div class="col-md-6 mx-auto">
                <a href="/administration/history_inventory" class="btn btn-primary btn-block mb-2">Historial de Inventario</a>    "Historial": Unknown word.
            </div>
        </div>
    </div>
</body>
</html>
```

Figura 46

Administrador_live



administration_live.html

templates > administration_live.html > html > body > div.container.mt-3 > div.row.mb-3 > div.col-md-12.text-center

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration Live</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Estadísticas en Vivo</h2>    "Estadísticas": Unknown word.

        <div class="row">
            <div class="col-md-6 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Comensales</h5>    "Comensales": Unknown word.
                        <p class="card-text">{{ estadisticas.cantidad_comensales }}</p>    "estadisticas": Unknown word.
                    </div>
                </div>
            </div>
```

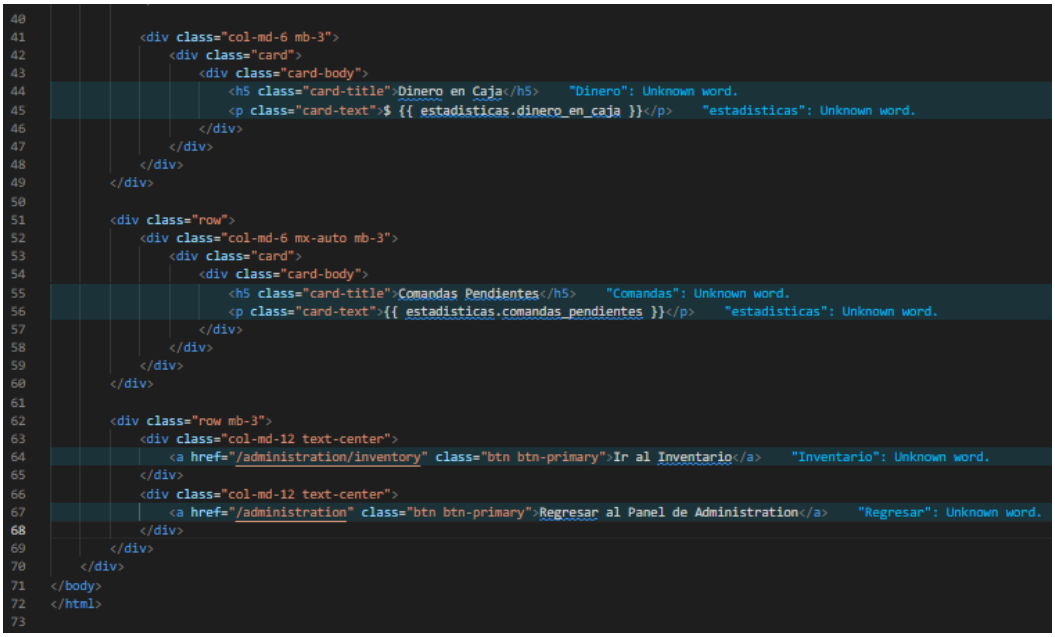


Figura 47

Administrador_inventory

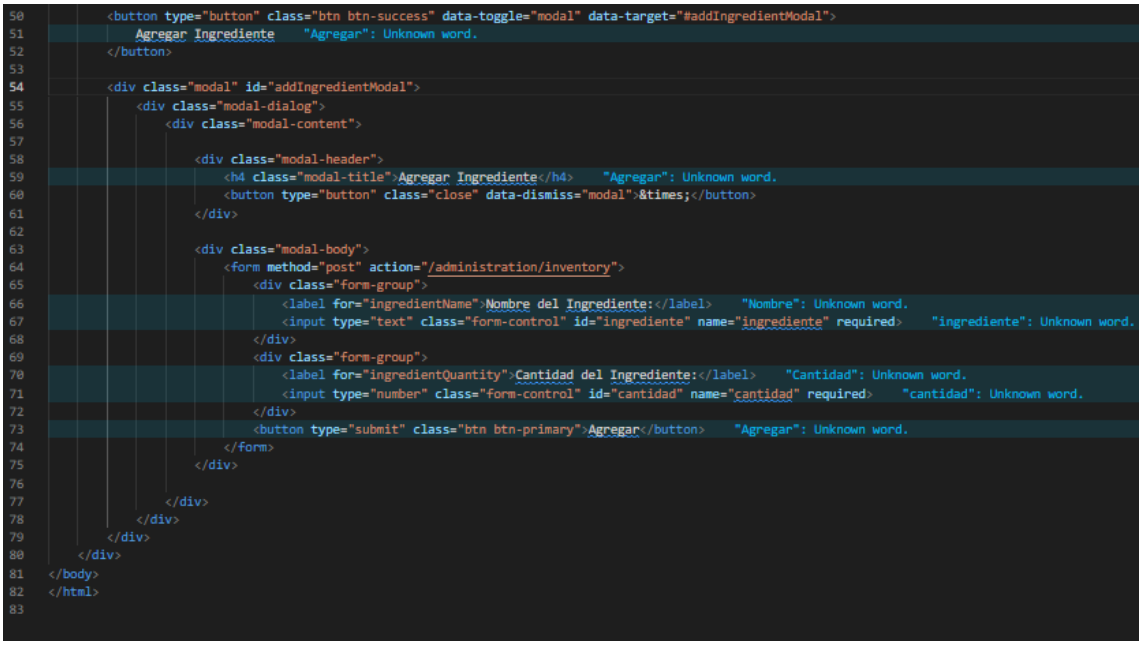
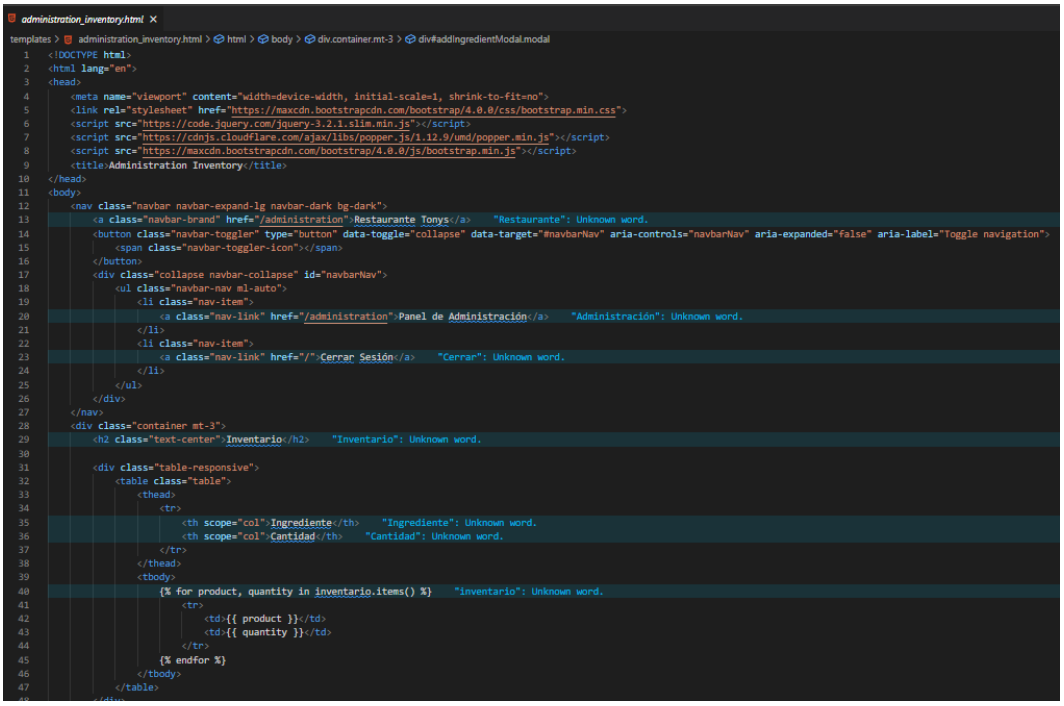


Figura 48

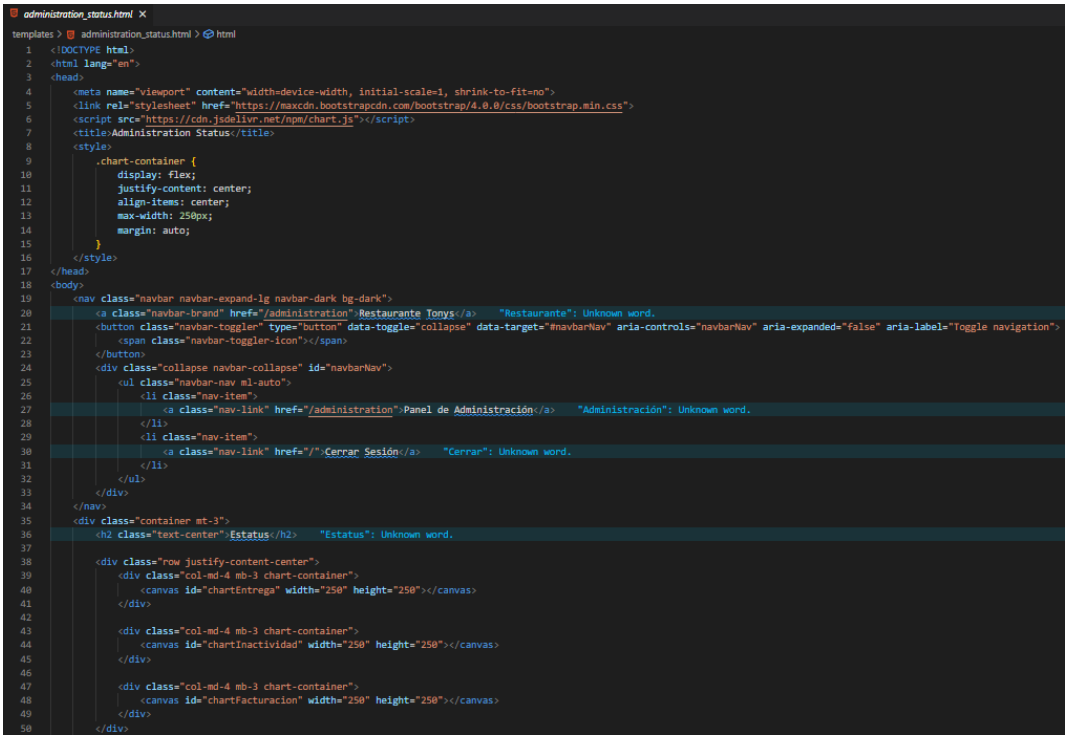
Administration_status



administration_inventory.html

templates > administration_inventory.html > html > body > div.container.mt-3 > div#addIngredientModal.modal

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration Inventory</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Inventario</h2>    "Inventario": Unknown word.

        <div class="table-responsive">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">Ingrediente</th>    "Ingrediente": Unknown word.
                        <th scope="col">Cantidad</th>    "Cantidad": Unknown word.
                    </tr>
                </thead>
                <tbody>
                    {% for product, quantity in inventario.items() %}    "inventario": Unknown word.
                        <tr>
                            <td>{{ product }}</td>
                            <td>{{ quantity }}</td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
```



administration_status.html

templates > administration_status.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <title>Administration Status</title>
    <style>
        .chart-container {
            display: flex;
            justify-content: center;
            align-items: center;
            max-width: 250px;
            margin: auto;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Estatus</h2>    "Estatus": Unknown word.

        <div class="row justify-content-center">
            <div class="col-md-4 mb-3 chart-container">
                <canvas id="chartEntrega" width="250" height="250"></canvas>
            </div>

            <div class="col-md-4 mb-3 chart-container">
                <canvas id="chartInactividad" width="250" height="250"></canvas>
            </div>

            <div class="col-md-4 mb-3 chart-container">
                <canvas id="chartFacturacion" width="250" height="250"></canvas>
            </div>
        </div>
```

```
    <!-- Genera los gráficos -->
    <script>
        var chartDataEntrega = {
            labels: ['Entrega de ordenes a tiempo','Entrega de ordenes tarde'],
            datasets: [{
                data: [100,0],
                backgroundColor: [
                    '#4CAF50', // verde
                    '#ec1c24'  // rojo
                ],
                borderColor: [
                    '#4CAF50', // verde
                    '#ec1c24'  // rojo
                ],
                borderWidth: 1
            }]
        };

        var chartDataInactividad = {
            labels: ['Periodo de actividad','Periodo de inactividad'],
            datasets: [{
                data: [100, 0],
                backgroundColor: [
                    '#2196F3', // azul
                    '#ff7f27'  // naranja
                ],
                borderColor: [
                    '#2196F3', // azul
                    '#ff7f27'  // naranja
                ],
                borderWidth: 1
            }]
        };

        var chartDataFacturacion = {
            labels: ['Porcentaje Facturado', "Porcentaje por Facturar"],
            datasets: [{
                data: [10,90],
                backgroundColor: [
                    '#024a86', // azul oscuro
                    '#c82a54'  // rojo
                ],
                borderColor: [
                    '#024a86', // azul oscuro
                    '#c82a54'  // rojo
                ],
                borderWidth: 1
            }]
        };
```

```
            var ctxEntrega = document.getElementById('chartEntrega').getContext('2d');    "Entrega": Unknown word.
            var ctxInactividad = document.getElementById('chartInactividad').getContext('2d');    "Inactividad": Unkn
            var ctxFacturacion = document.getElementById('chartFacturacion').getContext('2d');    "Facturacion": Unkn

            // Crea los gráficos    "Crea": Unknown word.
            var chartEntrega = new Chart(ctxEntrega, {
                type: 'doughnut',
                data: chartDataEntrega,
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                }
            });

            var chartInactividad = new Chart(ctxInactividad, {    "Inactividad": Unknown word.
                type: 'doughnut',
                data: chartDataInactividad,
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                }
            });

            var chartFacturacion = new Chart(ctxFacturacion, {    "Facturacion": Unknown word.
                type: 'doughnut',
                data: chartDataFacturacion,
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                }
            });
        </script>
    </div>

    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Figura 49

Historial inventario

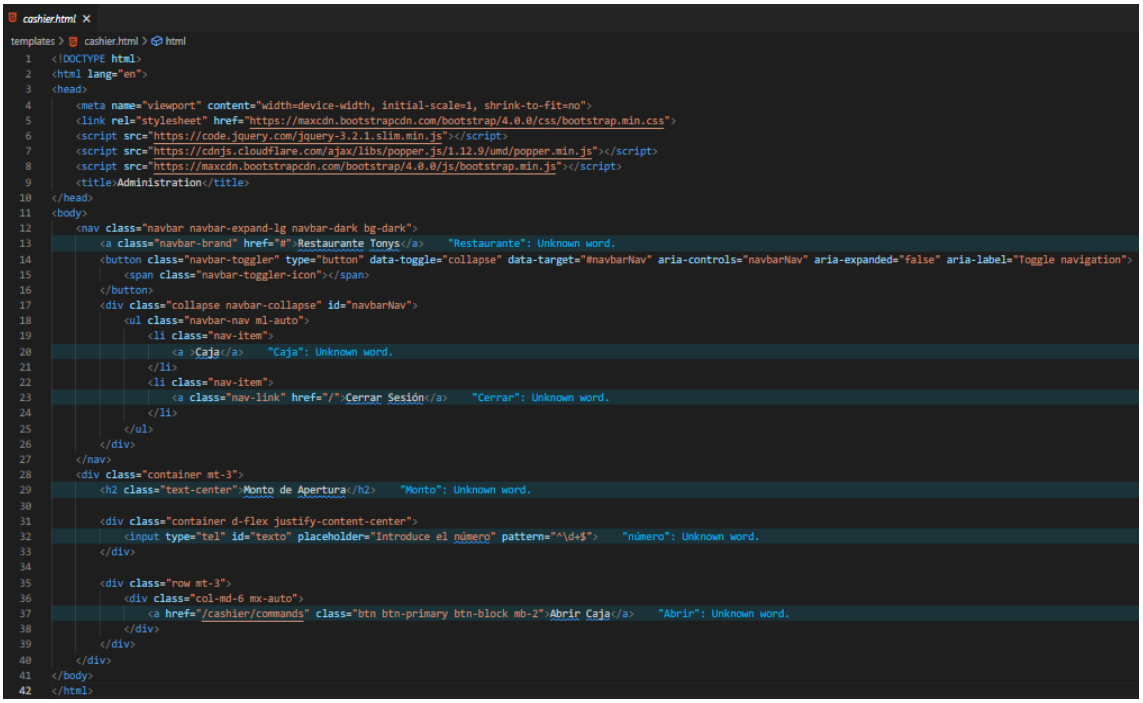
```
history_inventory.html 4 ×
templates > history_inventory.html > html > body > div.container.mt-3.text-center > div.mt-3 > div.table-responsive > table.table > tbody#tableBody
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <title>Administration Historial Inventario</title>    "Historial": Unknown word.
    <style>
        .chart-container {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3 text-center">
        <h3>Historial de Inventario de Noviembre</h3>          "Historial": Unknown word.
        <div class="mt-3">
            <label for="dayFilter">Seleccione el día:</label>    "Seleccione": Unknown word.
            <select id="dayFilter" class="form-control" onchange="updateChart()">
                {% for day in data_for_inventory.keys() %}
                    <option value="{{ day }}">{{ day }}</option>
                {% endfor %}
            </select>
```

```
            <h4 class="mt-3">Tabla de Cantidades por Ingrediente</h4>    "Cantidades": Unknown word.
            <div class="table-responsive">
                <table class="table">
                    <thead>
                        <tr>
                            <th scope="col">Ingrediente</th>    "Ingrediente": Unknown word.
                            <th scope="col">Cantidad</th>    "Cantidad": Unknown word.
                        </tr>
                    </thead>
                    <tbody id="tableBody">

                    </tbody>
                </table>
            </div>
        </div>

        <script>
            var dataForInventory = {{ data_for_inventory|tojson|safe }};    Property assignment expected.
            var currentDay = 'Día 1';

            function updateChart() {
                currentDay = document.getElementById('dayFilter').value;
                updateTable();
            }

            function updateTable() {
                var tableBody = document.getElementById('tableBody');
                tableBody.innerHTML = '';

                Object.entries(dataForInventory[currentDay]).forEach(([ingredient, quantity]) => {
                    var ingredientRow = document.createElement('tr');
                    var ingredientCell = document.createElement('td');
                    var quantityCell = document.createElement('td');

                    ingredientCell.textContent = ingredient;
                    quantityCell.textContent = quantity;

                    ingredientRow.appendChild(ingredientCell);
                    ingredientRow.appendChild(quantityCell);

                    tableBody.appendChild(ingredientRow);
                });
            }

            document.addEventListener('DOMContentLoaded', function () {
                updateTable();
            });
        </script>
    </div>
```

```
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Figura 50

Cashier

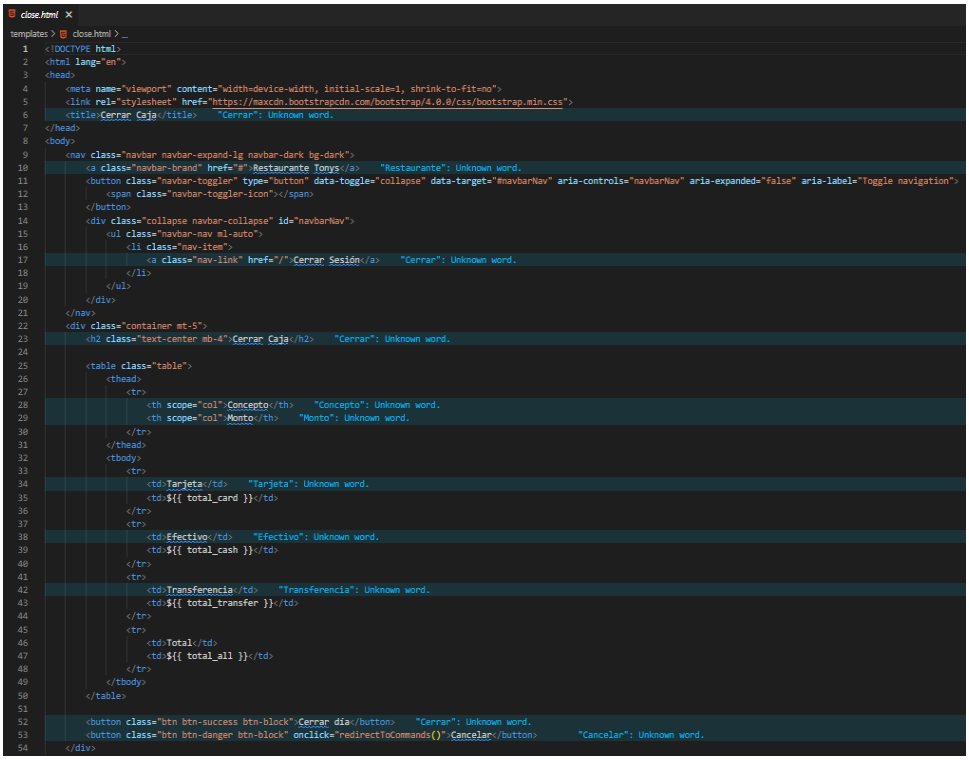


templates > cashier.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a >Caja</a>    "Caja": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Monto de Apertura</h2>    "Monto": Unknown word.

        <div class="container d-flex justify-content-center">
            <input type="tel" id="texto" placeholder="Introduce el número" pattern="^\d+$">    "número": Unknown word.
        </div>

        <div class="row mt-3">
            <div class="col-md-6 mx-auto">
                <a href="/cashier/commands" class="btn btn-primary btn-block mb-2">Abrir Caja</a>    "Abrir": Unknown word.
            </div>
        </div>
    </div>
</body>
</html>
```

Figura 51

Close (para el cajero)



templates > close.html > _

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <title>Cerrar Caja</title>    "Cerrar": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Cerrar Caja</h2>    "Cerrar": Unknown word.

        <table class="table">
            <thead>
                <tr>
                    <th scope="col">Concepto</th>    "Concepto": Unknown word.
                    <th scope="col">Monto</th>    "Monto": Unknown word.
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Tarjeta</td>    "Tarjeta": Unknown word.
                    <td>${{ total_card }}</td>
                </tr>
                <tr>
                    <td>Efectivo</td>    "Efectivo": Unknown word.
                    <td>${{ total_cash }}</td>
                </tr>
                <tr>
                    <td>Transferencia</td>    "Transferencia": Unknown word.
                    <td>${{ total_transfer }}</td>
                </tr>
                <tr>
                    <td>Total</td>
                    <td>${{ total_all }}</td>
                </tr>
            </tbody>
        </table>

        <button class="btn btn-success btn-block">Cerrar día</button>    "Cerrar": Unknown word.
        <button class="btn btn-danger btn-block" onclick="redirectToCommands()">Cancelar</button>    "Cancelar": Unknown word.
    </div>
```

```
    <script>
        function redirectToCommands() {
            window.location.href = '/cashier/commands';
        }
    </script>
</body>
</html>
```

Figura 52

Comandas x Cobrar

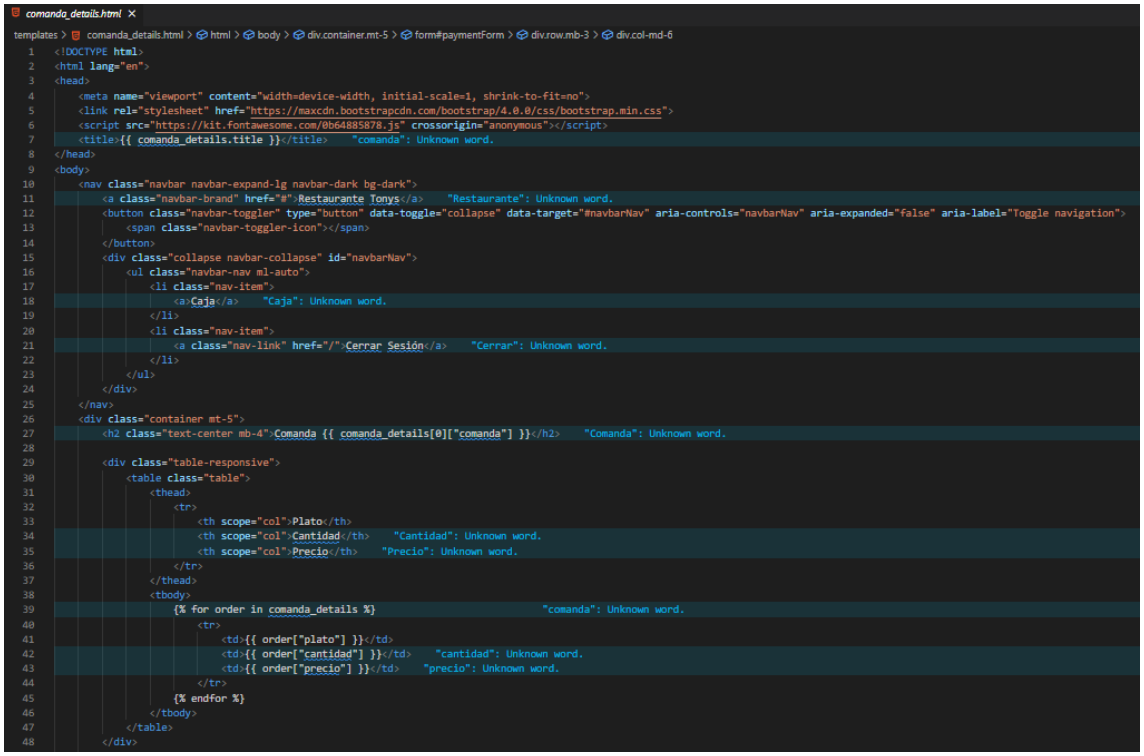
commands.html

templates > commands.html > html > body > div.container.mt-5 > div.mt-4.mb-4

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://kit.fontawesome.com/0b64885878.js" crossorigin="anonymous"></script>
    <title>Comandas</title>    "Comandas": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a >Caja</a>    "Caja": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Comandas</h2>    "Comandas": Unknown word.

        <ul class="list-group">
            {% for command_number in id_comandas %}    "comandas": Unknown word.
                <button class="list-group-item list-group-item-action d-flex justify-content-between align-items-center" onclick="handleButtonClick('{{ command_number }}')">
                    Comanda {{ command_number }}    "Comanda": Unknown word.
                </button>
            {% endfor %}
        </ul>

        <div class="mt-4 mb-4">
            <h4 class="text-center">Total Caja</h4>    "Caja": Unknown word.
            <div class="border border-primary p-3 text-center">
                $5000
            </div>
        </div>

        <button class="btn btn-danger btn-block" onclick="handleCloseButtonClick()">Cerrar Caja</button>    "Cerrar": Unknown word.
    </div>

    <script>
        function handleButtonClick(comandaId) {              "comanda": Unknown word.
            window.location.href = '/commands/' + comandaId;    "comanda": Unknown word.
        }

        function handleCloseButtonClick() {
            window.location.href = '/cashier/close';
        }
    </script>
</body>
</html>
```

Figura 53

Detalle de comandas



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://kit.fontawesome.com/0b64885878.js" crossorigin="anonymous"></script>
    <title>{{ comanda_details.title }}</title>    "comanda": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a>Caja</a>    "Caja": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Comanda {{ comanda_details[0]["comanda"] }}</h2>    "Comanda": Unknown word.

        <div class="table-responsive">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">Plato</th>
                        <th scope="col">Cantidad</th>    "Cantidad": Unknown word.
                        <th scope="col">Precio</th>    "Precio": Unknown word.
                    </tr>
                </thead>
                <tbody>
                    {% for order in comanda_details %}                    "comanda": Unknown word.
                        <tr>
                            <td>{{ order["plato"] }}</td>
                            <td>{{ order["cantidad"] }}</td>    "cantidad": Unknown word.
                            <td>{{ order["precio"] }}</td>    "precio": Unknown word.
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
```

```
        <div class="d-flex justify-content-center mb-4">
            <button class="btn btn-outline-primary p-3 mr-2"><i class="fa-solid fa-box-archive"></i></button>
            <button class="btn btn-outline-primary p-3 mr-2"><i class="fas fa-box"></i></button>
            <button class="btn btn-outline-danger p-3"><i class="fas fa-eraser"></i></button>
        </div>

        <form id="paymentForm" method="POST" action="{{ url_for('process_payment', comanda_id=comanda_details[0]['comanda']) }}">
            <div class="row mb-3">
                <div class="col-md-6">
                    <div class="input-group">
                        <div class="input-group-prepend">
                            <label class="input-group-text" for="paymentMethodSelect">Método de pago:</label>    "Método": Unknown
                        </div>
                        <select class="custom-select" id="paymentMethodSelect" name="payment_method">
                            <option selected>Seleccionar...</option>    "Seleccionar": Unknown word.
                            <option value="efectivo">Efectivo</option>    "efectivo": Unknown word.
                            <option value="tarjeta">Tarjeta de crédito</option>    "tarjeta": Unknown word.
                            <option value="transferencia">Transferencia</option>    "transferencia": Unknown word.
                        </select>
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="input-group">
                        <div class="input-group-prepend">
                            <span class="input-group-text">Total:</span>
                        </div>
                        <span class="input-group-text">${{ comanda_details[-1] }}</span>
                    </div>
                </div>
            </div>

            <button type="submit" class="btn btn-success btn-block" id="cobrarButton">Cobrar</button>    "Cobrar": Unknown word.
        </form>

        <button class="btn btn-danger btn-block" onclick="redirectToCommands()">Cancelar</button>    "Cancelar": Unknown word.

    </div>

    <script>
        function redirectToCommands() {
            window.location.href = '/cashier/commands';
        }
    </script>
</body>
</html>
```

Figura 54

Cocina Comanda detalle

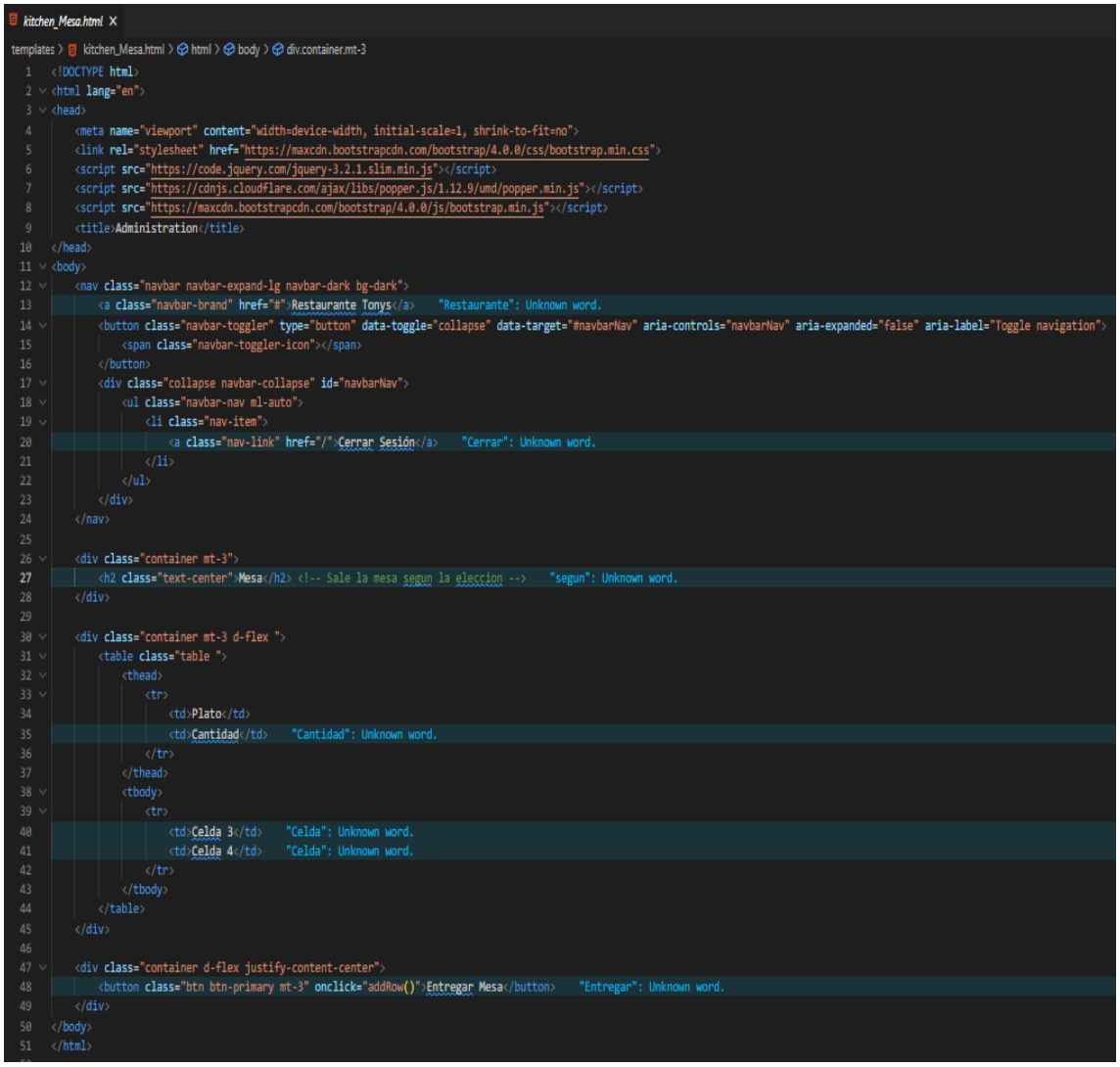
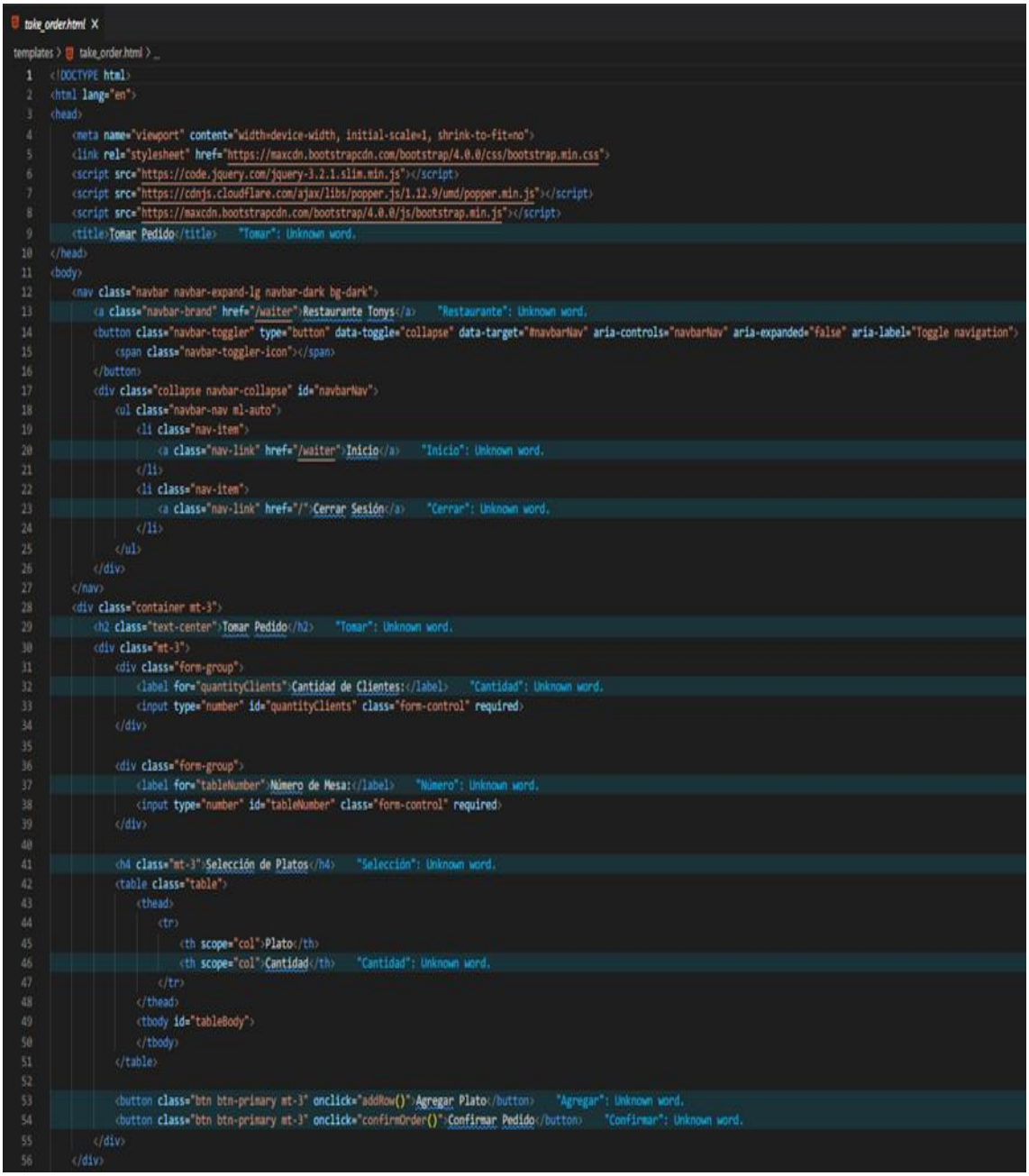


Figura 55

Cocina

Figura 56

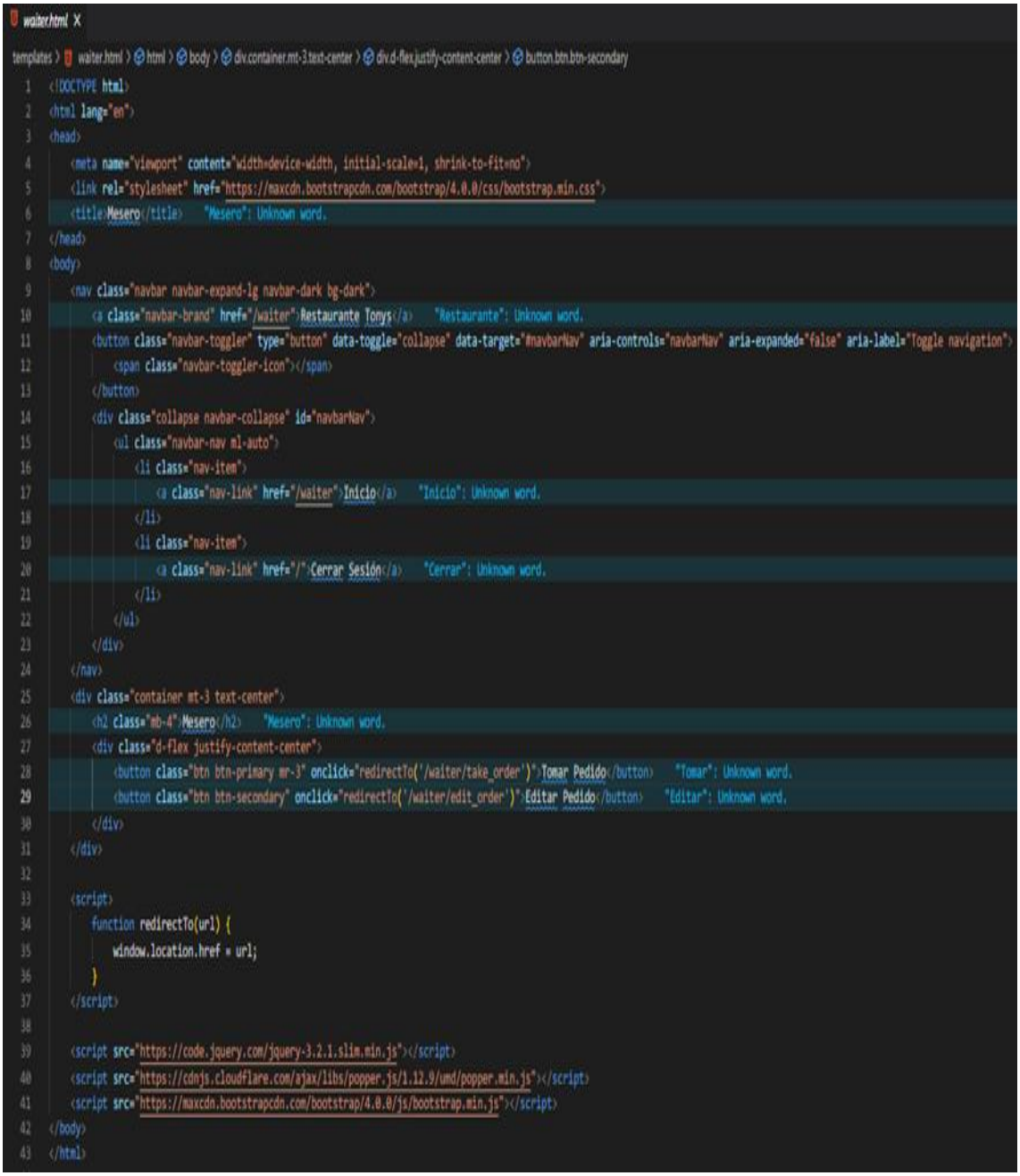
Mesero Tomar Orden



```
57
58      <script>
59          function confirmOrder() {
60              var tableData = [];
61              var tableBody = document.getElementById('tableBody');
62              var rows = tableBody.getElementsByTagName('tr');
63
64              for (var i = 0; i < rows.length; i++) {
65                  var cells = rows[i].getElementsByTagName('td');
66                  var plateName = cells[0].textContent;
67                  var quantity = cells[1].textContent;
68                  tableData.push({ plate: plateName, quantity: quantity });
69              }
70
71              var quantityClients = document.getElementById('quantityClients').value;
72              var tableNumber = document.getElementById('tableNumber').value;
73
74              console.log("Table Data:", tableData);
75              console.log("Quantity of Clients:", quantityClients);
76              console.log("Table Number:", tableNumber);
77
78              alert("Pedido confirmado. ¡Buen provecho!");    "Pedido": Unknown word.
79          }
80
81          function addRow() {
82              var plateName = prompt("Ingrese el nombre del plato:");    "Ingrese": Unk
83              var quantity = prompt("Ingrese la cantidad:");    "Ingrese": Unknown word
84
85              var tableBody = document.getElementById('tableBody');
86              var newRow = tableBody.insertRow();
87              var plateCell = newRow.insertCell(0);
88              var quantityCell = newRow.insertCell(1);
89
90              plateCell.textContent = plateName;
91              quantityCell.textContent = quantity;
92          }
93      </script>
94  </body>
95  </html>
96
```

Figura 57

Mesero

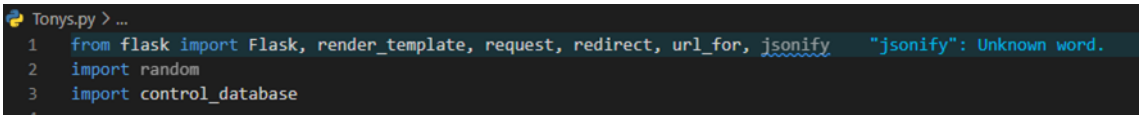


```
waiter.html
templates > waiter.html > html > body > div.container.mt-3.text-center > div.d-flex.justify-content-center > button.btn.btn-secondary
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <title>Mesero</title>    "Mesero": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/waiter">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/waiter">Inicio</a>    "Inicio": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3 text-center">
        <h2 class="mb-4">Mesero</h2>    "Mesero": Unknown word.
        <div class="d-flex justify-content-center">
            <button class="btn btn-primary mr-3" onclick="redirectTo('/waiter/take_order')">Tomar Pedido</button>    "Tomar": Unknown word.
            <button class="btn btn-secondary" onclick="redirectTo('/waiter/edit_order')">Editar Pedido</button>    "Editar": Unknown word.
        </div>
    </div>

    <script>
        function redirectTo(url) {
            window.location.href = url;
        }
    </script>

    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

- Diseño y Construcción de base de datos.

Figura 58

Importación de librerías utilizadas

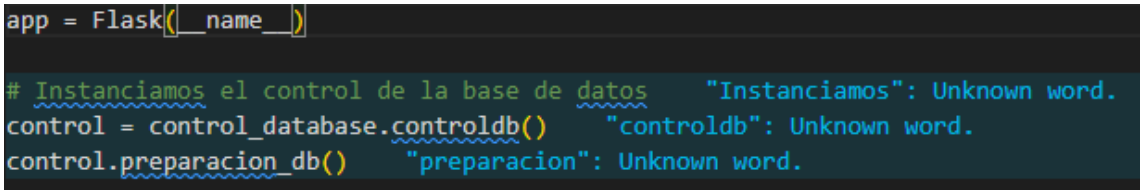


```
Tonys.py > ...
from flask import Flask, render_template, request, redirect, url_for, jsonify    "jsonify": Unknown word.
import random
import control_database
```

Los siguientes códigos están comentados por uso, todos estos van a pertenecer al archivo TONYS.PY.

Figura 59

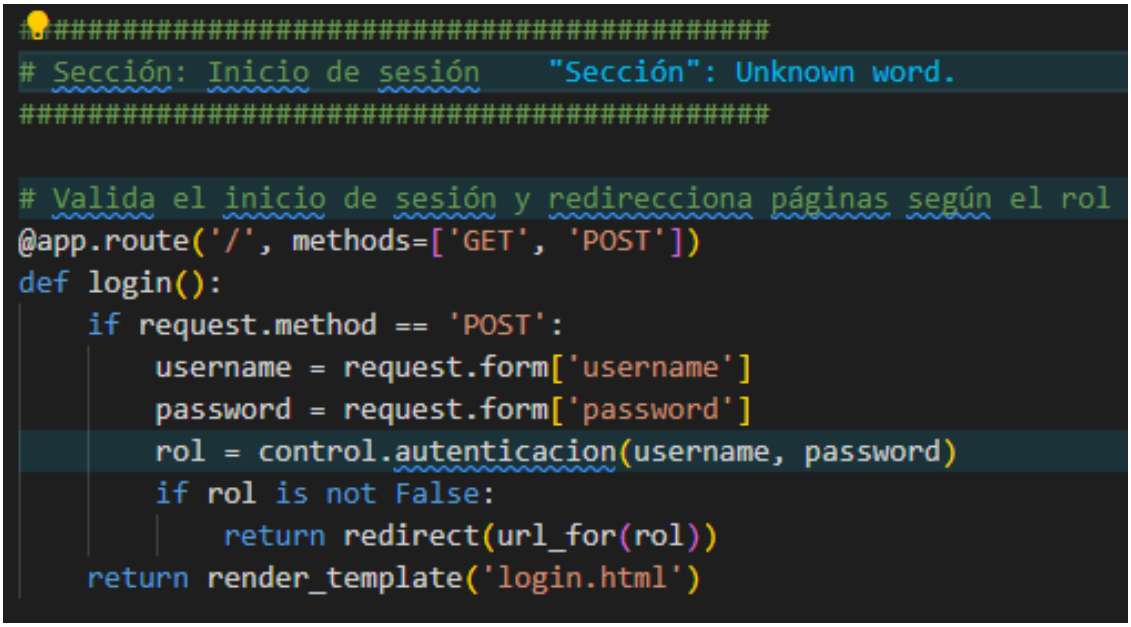
Código Tonys.py



```
app = Flask(__name__)

# Instanciamos el control de la base de datos    "Instanciamos": Unknown word.
control = control_database.controldb()    "controldb": Unknown word.
control.preparacion_db()    "preparacion": Unknown word.
```

Figura 60

Inicio de Sesión



```
##############################################
# Sección: Inicio de sesión    "Sección": Unknown word.
##############################################

# Valida el inicio de sesión y redirecciona páginas según el rol
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        rol = control.autenticacion(username, password)
        if rol is not False:
            return redirect(url_for(rol))
    return render_template('login.html')
```

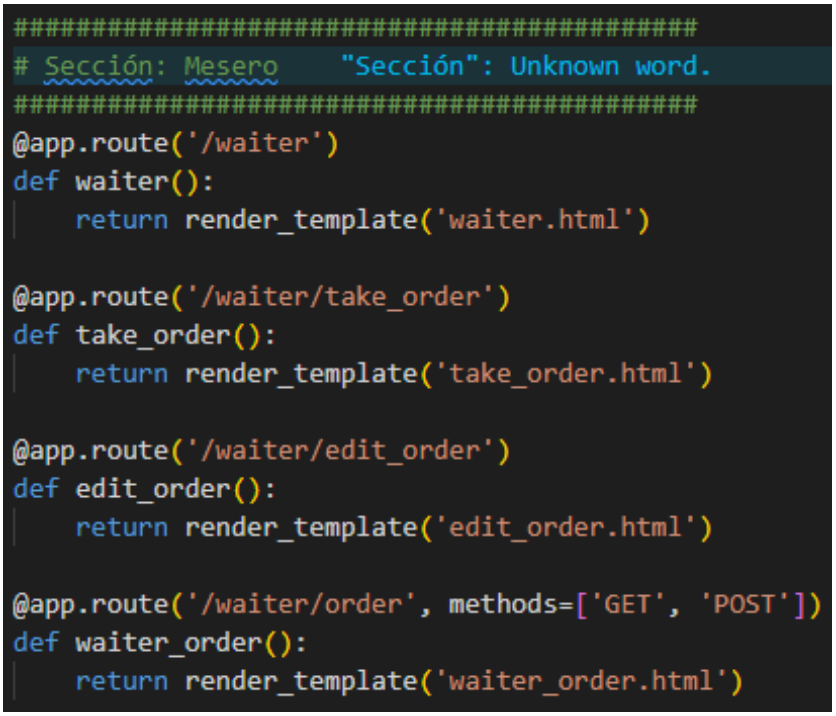
Figura 61

Administración

```
##########################################
# Sección: Administración    "Sección": Unknown word.
##########################################

# Panel de administración    "administración": Unknown word.
@app.route('/administration')
def administration():
    return render_template('administration.html')

# En vivo
@app.route('/administration/live')
def administration_live():
    # Solicitamos los datos para las estadísticas en vivo    "Solicitamos": Unknown
    estadisticas = control.get_envivo()        "estadisticas": Unknown word.
    return render_template('administration_live.html', estadisticas=estadisticas)

# Estatus    "Estatus": Unknown word.
@app.route('/administration/status')
def administration_status():
    # Solicitamos los datos para el estatus    "Solicitamos": Unknown word.
    estatus = control.get_status()    "estatus": Unknown word.
    return render_template("administration_status.html", estatus=estatus)    "estatu

# Inventario    "Inventario": Unknown word.
@app.route('/administration/inventory', methods=['GET', 'POST'])
def administration_inventory():
    # Solicitamos los datos de inventario    "Solicitamos": Unknown word.
    inventario = control.get_inventario()        "inventario": Unknown word.

    # Permite agregar ingredientes    "Permite": Unknown word.
    if request.method == 'POST':
        ingrediente = request.form['ingrediente']    "ingrediente": Unknown word.
        cantidad = request.form['cantidad']                "cantidad": Unknown word.
        control.agregar_ingrediente(ingrediente, cantidad)    "agregar": Unknown wor
        inventario = control.get_inventario()    "inventario": Unknown word.

    return render_template('administration_inventory.html', inventario=inventario)
```

Figura 62

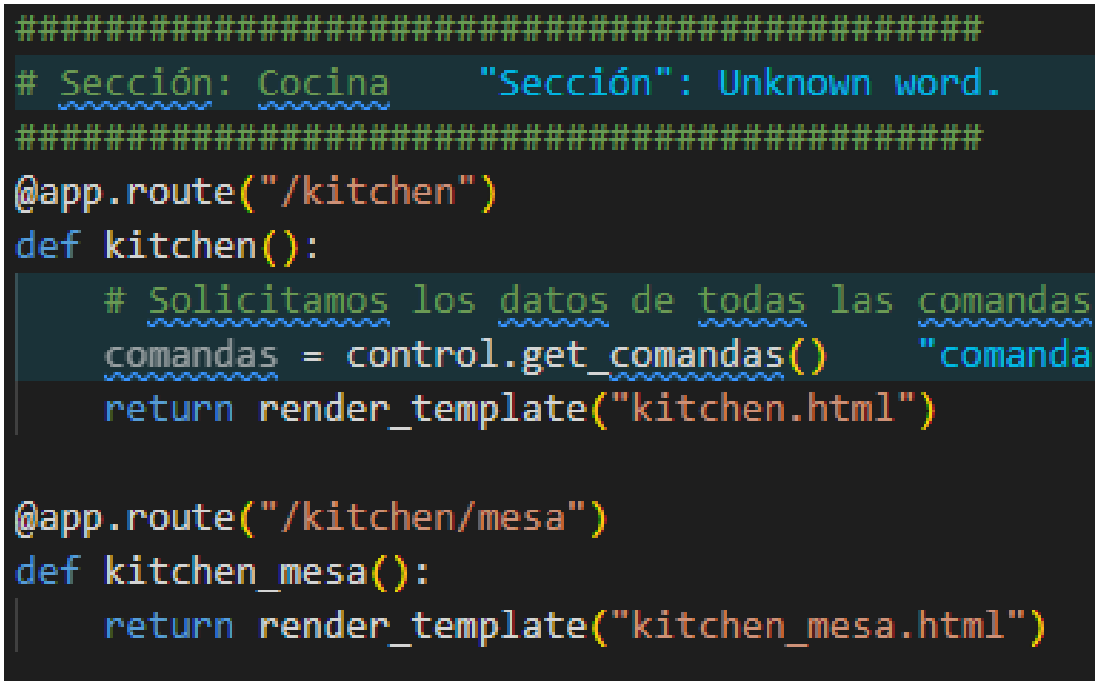
Mesero



```
##########################################
# Sección: Mesero    "Sección": Unknown word.
##########################################
@app.route('/waiter')
def waiter():
    return render_template('waiter.html')

@app.route('/waiter/take_order')
def take_order():
    return render_template('take_order.html')

@app.route('/waiter/edit_order')
def edit_order():
    return render_template('edit_order.html')

@app.route('/waiter/order', methods=['GET', 'POST'])
def waiter_order():
    return render_template('waiter_order.html')
```

Figura 63

Cocina



```
################################################
# Sección: Cocina    "Sección": Unknown word.
################################################
@app.route("/kitchen")
def kitchen():
    # Solicitamos los datos de todas las comandas
    comandas = control.get_comandas()    "comanda
    return render_template("kitchen.html")

@app.route("/kitchen/mesa")
def kitchen_mesa():
    return render_template("kitchen_mesa.html")
```

Figura 64

Caja

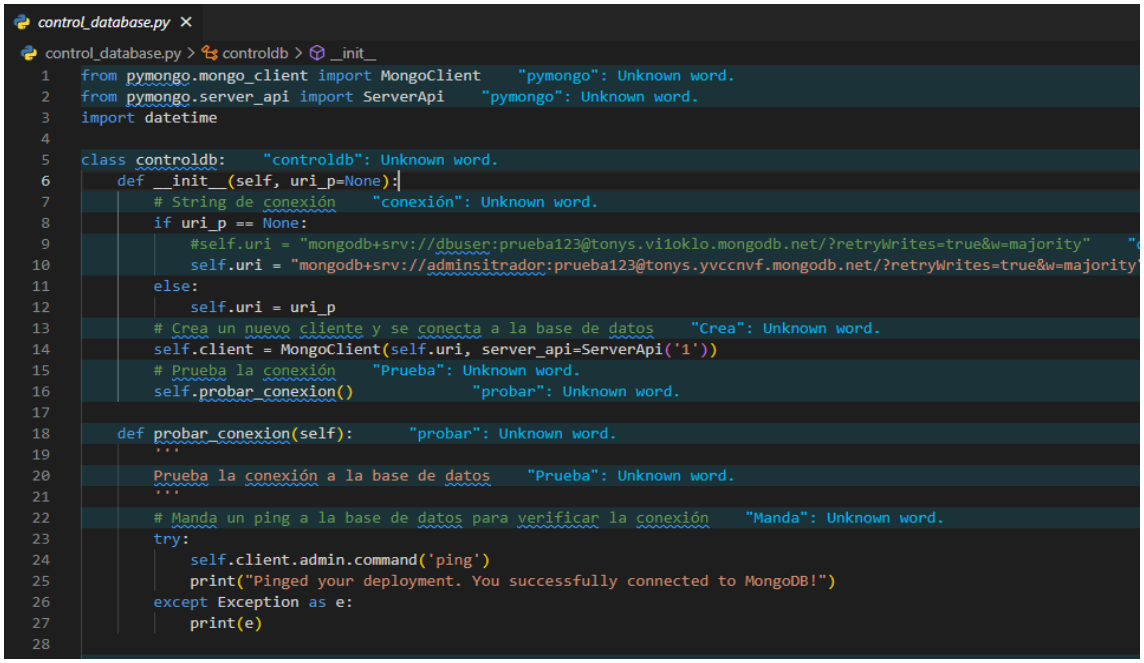
```
##########################################
# Sección: Caja    "Sección": Unknown word.
##########################################

# Página de Monto de apertura    "Página": Unknown word.
@app.route("/cashier")
def cashier():
    return render_template("cashier.html")

# Proceso la cantidad de monto de apertura    "Proceso": Unknown word.
@app.route("/process_opening", methods=["POST"])
def process_opening():
    opening_amount = request.form.get("openingAmount")

    return "Opening amount processed successfully"

# Muestra todas las comandas registradas    "Muestra": Unknown word.
@app.route("/cashier/commands")
def commands():
    # Solicitamos las comandas existentes    "Solicitamos": Unknown word.
    id_comandas = control.get_idcomandas()        "idcomandas": Unknown word.
    return render_template("commands.html", id_comandas=id_comandas)

# Muestra los detalles de una comanda y registra el pago    "Muestra": Unknown word
@app.route("/commands/<comanda_id>")    "comanda": Unknown word.
def comanda(comanda_id):        "comanda": Unknown word.
    # Solicitamos los detalles    "detalles": Unknown word.
    comandas = control.get_comandasdetails(comanda_id)    "comandasdetails": Unknow
    if comandas is None:
        return render_template("comanda_not_found.html")
    return render_template("comanda_details.html", comanda_details=comandas)

# Procesa y registra el pago    "Procesa": Unknown word.
@app.route("/process_payment/<comanda_id>", methods=['POST'])
def process_payment(comanda_id):
    payment_method = request.form.get('payment_method')
    control.procesar_pago(comanda_id, payment_method)    "procesar": Unknown word.
    return redirect(url_for('commands'))
```

```python
# Cierra la caja    "Cierra": Unknown word.
@app.route("/cashier/close")
def close():
    # Solicitamos todos los pagos registrados    "todos": Unknown word.
    pagos = control.get_pagos()    "pagos": Unknown word.

    # Inicializa totales
    total_card = 0
    total_cash = 0
    total_transfer = 0
    total_expenses = 0

    # Recorrer los pagos y actualizar los totales
    for pago in pagos:
        payment_method = pago.get("payment_method")
        total_amount = pago.get("total")

        if payment_method == "tarjeta":
            total_card += total_amount
        elif payment_method == "efectivo":
            total_cash += total_amount
        elif payment_method == "transferencia":
            total_transfer += total_amount
        elif payment_method == "gastos":
            total_expenses += total_amount

    # Calcula todo el total
    total_all = total_card + total_cash + total_transfer - total_expenses

    return render_template("close.html", total_card=total_card, total_cash=total_cash,
                           total_transfer=total_transfer, total_expenses=total_expenses,
                           total_all=total_all)

if __name__ == '__main__':
    app.run(debug=True)
```

Se realizo dos códigos aparte para tener una buena gestión de código, una que hace los procesos de cada uno de los usuarios, otra que está a continuación que son los procesos del control de base de datos.

Figura 65

Conexión a la Base de datos



```python
from pymongo.mongo_client import MongoClient    "pymongo": Unknown word.
from pymongo.server_api import ServerApi    "pymongo": Unknown word.
import datetime

class controldb:    "controldb": Unknown word.
    def __init__(self, uri_p=None):
        # String de conexión    "conexión": Unknown word.
        if uri_p == None:
            #self.uri = "mongodb+srv://dbuser:prueba123@tonys.vi1oklo.mongodb.net/?retryWrites=true&w=majority"    "
            self.uri = "mongodb+srv://adminsitrador:prueba123@tonys.yvccnvf.mongodb.net/?retryWrites=true&w=majority"
        else:
            self.uri = uri_p
        # Crea un nuevo cliente y se conecta a la base de datos    "Crea": Unknown word.
        self.client = MongoClient(self.uri, server_api=ServerApi('1'))
        # Prueba la conexión    "Prueba": Unknown word.
        self.probar_conexion()            "probar": Unknown word.

    def probar_conexion(self):      "probar": Unknown word.
        '''
        Prueba la conexión a la base de datos    "Prueba": Unknown word.
        '''
        # Manda un ping a la base de datos para verificar la conexión    "Manda": Unknown word.
        try:
            self.client.admin.command('ping')
            print("Pinged your deployment. You successfully connected to MongoDB!")
        except Exception as e:
            print(e)
```

En la función preparación, si no existen las colecciones en la base de datos se las crea y también se agregan datos esenciales, como, por ejemplo, los platos, ingredientes y usuarios, algunos datos de comandas para la versión demostrativa.

Figura 66

Conexión a la Base de datos

```
def preparacion_db(self):
    '''
    Prepara la base de datos con valores iniciales
    '''
    # Tomamos todas las colecciones
    colecciones = self.client.restaurante.list_collection_names()

    # Evaluamos para platos
    if "pagos" not in colecciones:
        self.client.restaurante.create_collection("pagos")
    if "usuarios" not in colecciones:
        self.client.restaurante.create_collection("usuarios")
        usuarios = [
            {"username":"admin", "password":"password", "rol":["administration"]},
            {"username":"waiter", "password":"password", "rol":["waiter"]},
            {"username":"kitchen", "password":"password", "rol":["kitchen"]},
            {"username":"cashier", "password":"password", "rol":["cashier"]}
        ]
        self.client.restaurante.usuarios.insert_many(usuarios)
    if "platos" not in colecciones:
        self.client.restaurante.create_collection("platos")
        platos = [
            {"plato":"Matambre", "precio":4.50},
            {"plato":"Tomahawk", "precio":15},
            {"plato":"Porterhouse", "precio":9},
            {"plato":"Ribeye", "precio":9},
            {"plato":"Invididual costilla", "precio":12},
            {"plato":"Cuarto costilla", "precio":18},
            {"plato":"Medio costilla", "precio":30},
            {"plato":"Metro costilla", "precio":60}
        ]
        self.client.restaurante.platos.insert_many(platos)
    if "comandas" not in colecciones:
        self.client.restaurante.create_collection("comandas")
        comandas = [
            {"comanda":1, "comensales":5, "plato":"Matambre", "cantidad":2, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "hora_preparado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado"
            {"comanda":1, "comensales":5, "plato":"Ribeye", "cantidad":4, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False},
            {"comanda":1, "comensales":5, "plato":"Tomahawk", "cantidad":1, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False},
            {"comanda":2, "comensales":1, "plato":"Metro costilla", "cantidad":2, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False},
            {"comanda":3, "comensales":3, "plato":"Invididual costilla", "cantidad":1, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False},
            {"comanda":3, "comensales":3, "plato":"Cuarto costilla", "cantidad":1, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False},
            {"comanda":3, "comensales":3, "plato":"Matambre", "cantidad":1, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False},
            {"comanda":4, "comensales":1, "plato":"Tomahawk", "cantidad":10, "hora_solicitado":datetime.datetime.today().replace(microsecond=0), "entregado":False, "cobrado":False}
        ]
        self.client.restaurante.comandas.insert_many(comandas)
    if "ingredientes" not in colecciones:
        self.client.restaurante.create_collection("ingredientes")
        ingredientes = [
            {"ingrediente":"Caja de tomates", "cantidad":20, "fecha":datetime.datetime.today().replace(microsecond=0)},
            {"ingrediente":"Costillas", "cantidad":15, "fecha":datetime.datetime.today().replace(microsecond=0)}
        ]
        self.client.restaurante.ingredientes.insert_many(ingredientes)
```

```
    def quediaeshoy(self):
        '''
        Retorna el día de hoy
        '''
        return datetime.datetime.today().replace(microsecond=0)
```

```
    def autenticacion(self, usuario, contraseña):
        '''
        Valida la autenticación y devuelve el rol del usuario
        '''
        user = self.client.restaurante.usuarios.find_one({'username': usuario, 'password': contraseña})
        if user is not None:
            return user.get("rol")[0]
        else:
            return False
```

```python
    def get_envivo(self):
        '''
        Recupera los siguientes datos:
            - Cantidad de comensales
            - Cantidad de dinero en caja
            - Cantidad de comandas pendientes
            - Cantidad de dinero por cobrar
        '''
        # Total de comensales y dinero por cobrar
        comensales_cursor = self.client.restaurante.comandas.find({"cobrado":False})
        cantidad_comensales = 0
        comandas_consideradas = []
        for comensal in comensales_cursor:
            if comensal.get("comanda") not in comandas_consideradas:
                cantidad_comensales += comensal.get("comensales")
                comandas_consideradas.append(comensal.get("comanda"))
        # Dinero en caja (cobrado)
        pagos_cursor = self.client.restaurante.pagos.find()
        dinero_en_caja = 0
        if pagos_cursor is not None:
            for pago in pagos_cursor:
                dinero_en_caja += pago.get("total")
        # Comandas pendientes
        comandas_pendientes = self.client.restaurante.comandas.find({"entregado":False, "cobrado":False})

        # Devuelve lo requerido
        return {
            "cantidad_comensales":cantidad_comensales,
            "dinero_en_caja" : dinero_en_caja,
            "comandas_pendientes" : len(comandas_pendientes.distinct("comanda"))
        }
```

```python
    def get_status(self):
        '''
        Devuelve el estatus de facturación, tiempo de actividad y entrega de ordenes a tiempo
        '''
        return {
            "entregas_bien" : 70,
            "entregas_mal" : 30
        }
```

```python
    def get_inventario(self):
        '''
        Devuelve todo el inventario de ingredientes
        '''
        inventario_cursor = self.client.restaurante.ingredientes.find()
        hoy = str(self.quediaeshoy()).split(" ")[0]
        inventario = {}
        for ingrediente in inventario_cursor:
            # Valida que sea el inventario actualizado del día
            if str(ingrediente.get("fecha")).split(" ")[0] == hoy:
                inventario[ingrediente.get("ingrediente")] = ingrediente.get("cantidad")
        return inventario
```

```python
    def agregar_ingrediente(self, ingrediente, cantidad):
        '''
        Agrega un nuevo ingrediente
        '''
        nuevo_ingrediente = {
            "ingrediente" : ingrediente,
            "cantidad" : cantidad,
            "fecha" : datetime.datetime.today().replace(microsecond=0)
        }
        self.client.restaurante.ingredientes.insert_one(nuevo_ingrediente)
```

```python
    def get_comandas(self):
        '''
        Retorna todas las comandas
        '''
        return ""
```

```python
    def get_idcomandas(self):
        '''
        Retorna todos los ids de las comandas
        '''
        return self.client.restaurante.comandas.distinct("comanda")

```

```python
    def get_comandasdetails(self, comanda_id):
        '''
        Retorna todos los detalles de las comandas
        '''
        comandas_cursor = self.client.restaurante.comandas.find({"comanda":int(comanda_id)})
        comandas = []
        for doc in comandas_cursor:
            precio_plato = self.client.restaurante.platos.find_one({"plato":doc.get("plato")})
            precio = precio_plato.get("precio",0)
            total_precio = doc.get("cantidad") * precio
            datos = {
                "comanda" : doc.get("comanda"),
                "plato" : doc.get("plato"),
                "cantidad" : doc.get("cantidad"),
                "precio" : total_precio
            }
            comandas.append(datos)

        total = 0
        for comanda in comandas:
            total += comanda["precio"]
        comandas.append(total)

        return comandas
```

```python
    def procesar_pago(self, id_comanda, payment_method):
        '''
        Procesa y registra el pago en la base de datos
        '''
        comandas_cursor = self.client.restaurante.comandas.find({"comanda":int(id_comanda)})
        total = 0
        # Permite calcular el total de la comanda (hay que mejorar, no esta optimizado)
        for doc in comandas_cursor:
            precio_plato = self.client.restaurante.platos.find_one({"plato":doc.get("plato")})
            precio = precio_plato.get("precio",0)
            total += doc.get("cantidad") * precio

        # Añade a la base de datos la información del pago
        payment_data = {
            "payment_method": payment_method,
            "total_amount": total
        }
        self.client.restaurante.pagos.insert_one(payment_data)

        # Editamos el estado de la comanda
        operacion = {
            "$set" : {
                "cobrado":True
            }
        }
        self.client.restaurante.comandas.update_many({"comanda":id_comanda}, operacion)
```

```python
    def get_pagos(self):
        '''
        Retorna todos los pagos
        '''
        return self.client.restaurante.pagos.find()

```

```python
if __name__ == "__main__":
    control = controldb()
    control.eliminar_coleccion("comandas")
    control.preparacion_db()
```

Figura 67

Contenido del aplicativo Web en HTML5 Login

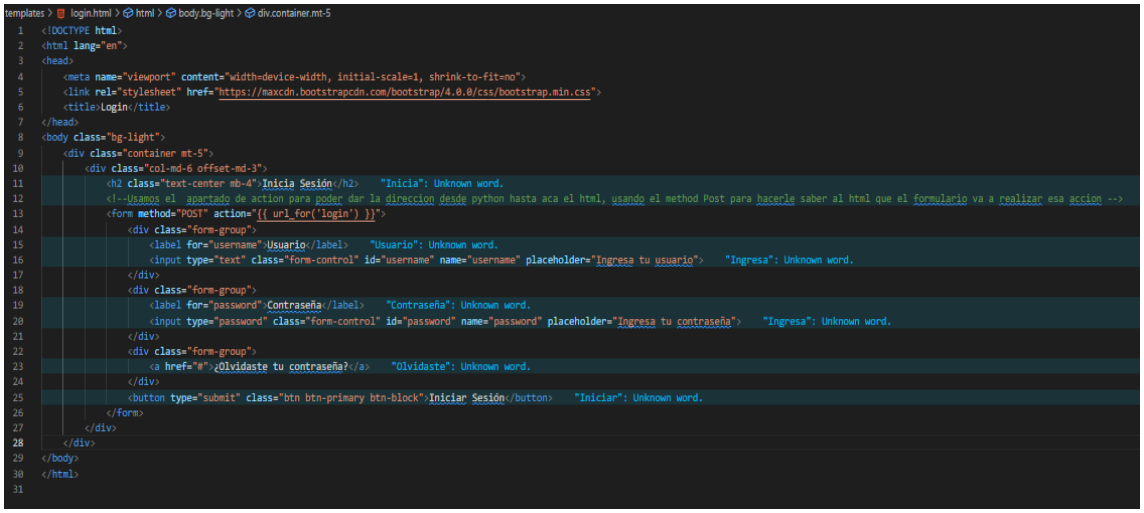


Figura 68

Administrador

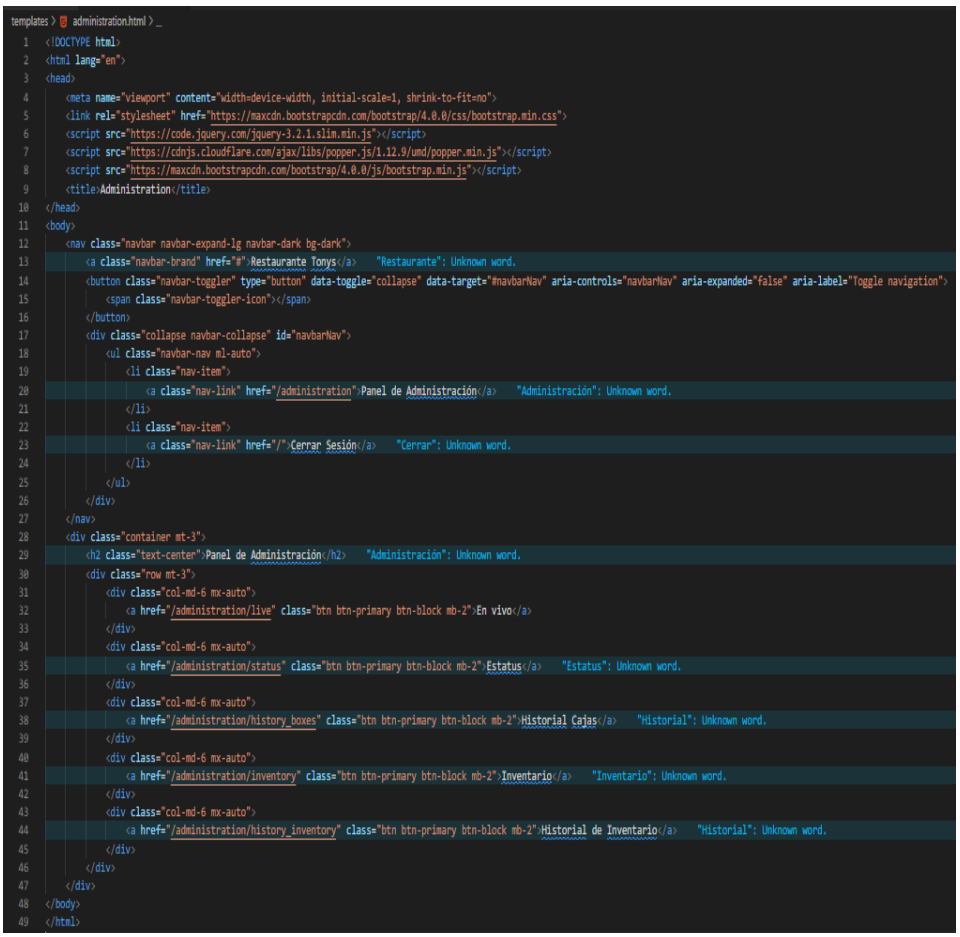
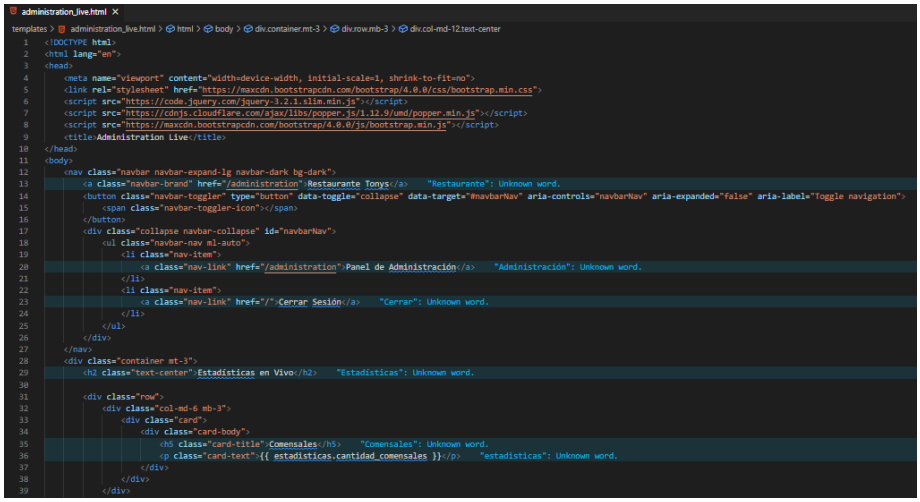


Figura 69

Administrador_live




```
40
41          <div class="col-md-6 mb-3">
42              <div class="card">
43                  <div class="card-body">
44                      <h5 class="card-title">Dinero en Caja</h5>    "Dinero": Unknown word.
45                      <p class="card-text">$ {{ estadisticas.dinero_en_caja }}</p>    "estadisticas": Unknown word.
46                  </div>
47              </div>
48          </div>
49      </div>
50
51      <div class="row">
52          <div class="col-md-6 mx-auto mb-3">
53              <div class="card">
54                  <div class="card-body">
55                      <h5 class="card-title">Comandas Pendientes</h5>    "Comandas": Unknown word.
56                      <p class="card-text">{{ estadisticas.comandas_pendientes }}</p>    "estadisticas": Unknown word.
57                  </div>
58              </div>
59          </div>
60      </div>
61
62      <div class="row mb-3">
63          <div class="col-md-12 text-center">
64              <a href="/administration/inventory" class="btn btn-primary">Ir al Inventario</a>    "Inventario": Unknown word.
65          </div>
66          <div class="col-md-12 text-center">
67              <a href="/administration" class="btn btn-primary">Regresar al Panel de Administration</a>    "Regresar": Unknown word.
68          </div>
69      </div>
70  </div>
71  </body>
72  </html>
73
```

Figura 70

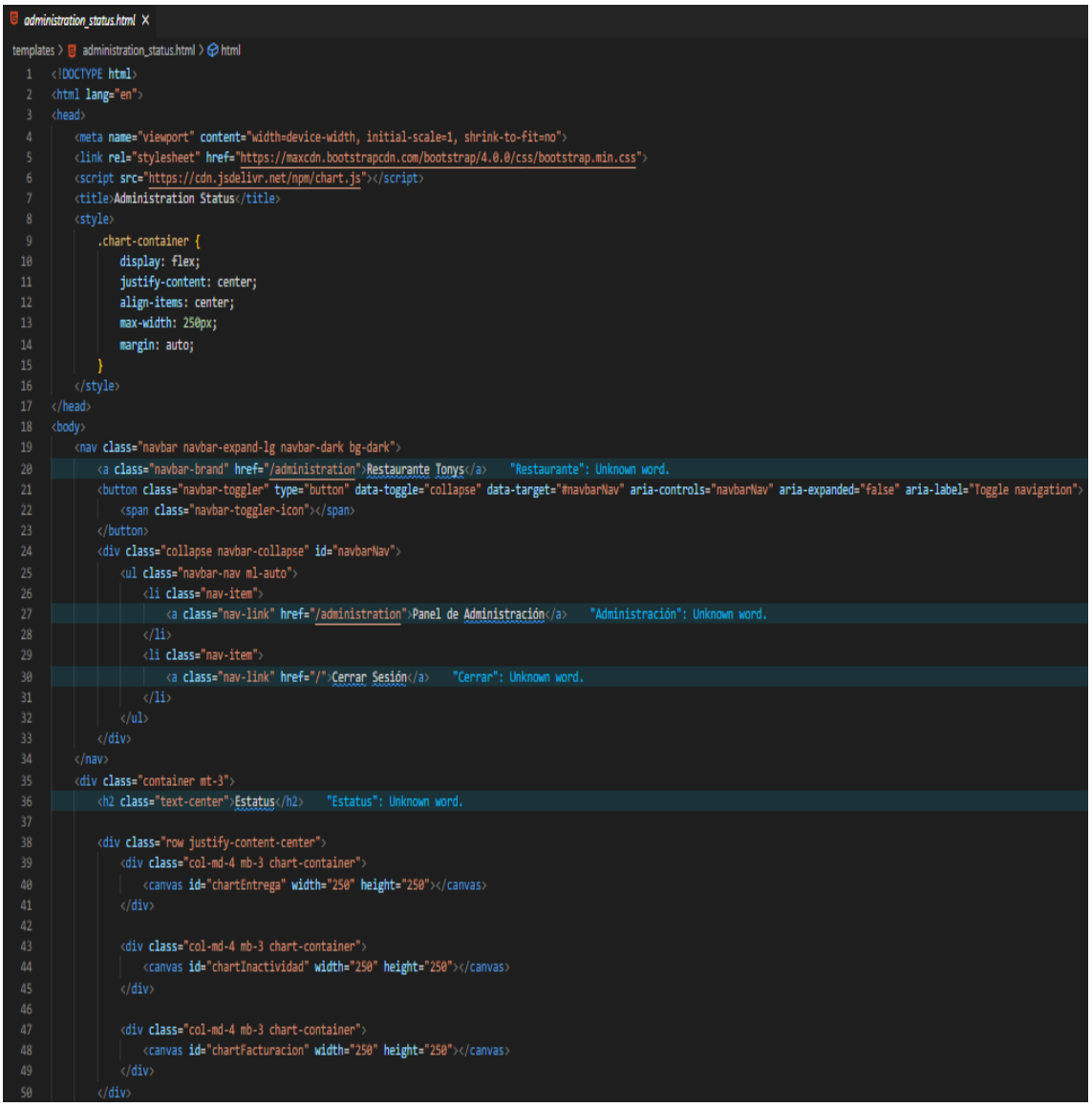
Administration_inventory

```
administration_inventory.html ×
templates > administration_inventory.html > html > body > div.container.mt-3 > div#addIngredientModal.modal
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration Inventory</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Inventario</h2>    "Inventario": Unknown word.

        <div class="table-responsive">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">Ingrediente</th>    "Ingrediente": Unknown word.
                        <th scope="col">Cantidad</th>    "Cantidad": Unknown word.
                    </tr>
                </thead>
                <tbody>
                    {% for product, quantity in inventario.items() %}    "inventario": Unknown word.
                        <tr>
                            <td>{{ product }}</td>
                            <td>{{ quantity }}</td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
```

```
        <button type="button" class="btn btn-success" data-toggle="modal" data-target="#addIngredientModal">
            Agregar Ingrediente    "Agregar": Unknown word.
        </button>

        <div class="modal" id="addIngredientModal">
            <div class="modal-dialog">
                <div class="modal-content">

                    <div class="modal-header">
                        <h4 class="modal-title">Agregar Ingrediente</h4>    "Agregar": Unknown word.
                        <button type="button" class="close" data-dismiss="modal">&times;</button>
                    </div>

                    <div class="modal-body">
                        <form method="post" action="/administration/inventory">
                            <div class="form-group">
                                <label for="ingredientName">Nombre del Ingrediente:</label>    "Nombre": Unknown word.
                                <input type="text" class="form-control" id="ingrediente" name="ingrediente" required>    "ingrediente": Unknown word.
                            </div>
                            <div class="form-group">
                                <label for="ingredientQuantity">Cantidad del Ingrediente:</label>    "Cantidad": Unknown word.
                                <input type="number" class="form-control" id="cantidad" name="cantidad" required>    "cantidad": Unknown word.
                            </div>
                            <button type="submit" class="btn btn-primary">Agregar</button>    "Agregar": Unknown word.
                        </form>
                    </div>

                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Figura 71

Administration_status



```
administration_status.html X
templates > administration_status.html > html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <title>Administration Status</title>
    <style>
        .chart-container {
            display: flex;
            justify-content: center;
            align-items: center;
            max-width: 250px;
            margin: auto;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Estatus</h2>    "Estatus": Unknown word.

        <div class="row justify-content-center">
            <div class="col-md-4 mb-3 chart-container">
                <canvas id="chartEntrega" width="250" height="250"></canvas>
            </div>

            <div class="col-md-4 mb-3 chart-container">
                <canvas id="chartInactividad" width="250" height="250"></canvas>
            </div>

            <div class="col-md-4 mb-3 chart-container">
                <canvas id="chartFacturacion" width="250" height="250"></canvas>
            </div>
        </div>
```

```
    <!-- Genera los gráficos -->
    <script>
        var chartDataEntrega = {
            labels: ['Entrega de ordenes a tiempo','Entrega de ordenes tarde'],
            datasets: [{
                data: [100,0],
                backgroundColor: [
                    '#4CAF50', // verde
                    '#ec1c24'  // rojo
                ],
                borderColor: [
                    '#4CAF50', // verde
                    '#ec1c24'  // rojo
                ],
                borderWidth: 1
            }]
        };

        var chartDataInactividad = {
            labels: ['Periodo de actividad','Periodo de inactividad'],
            datasets: [{
                data: [100, 0],
                backgroundColor: [
                    '#2196F3', // azul
                    '#ff7f27'  // naranja
                ],
                borderColor: [
                    '#2196F3', // azul
                    '#ff7f27'  // naranja
                ],
                borderWidth: 1
            }]
        };

        var chartDataFacturacion = {
            labels: ['Porcentaje Facturado', "Porcentaje por Facturar"],
            datasets: [{
                data: [10,90],
                backgroundColor: [
                    '#024a86', // azul oscuro
                    '#c82a54'  // rojo
                ],
                borderColor: [
                    '#024a86', // azul oscuro
                    '#c82a54'  // rojo
                ],
                borderWidth: 1
            }]
        };
```

```
            var ctxEntrega = document.getElementById('chartEntrega').getContext('2d');    "Entrega": Unknown word.
            var ctxInactividad = document.getElementById('chartInactividad').getContext('2d');    "Inactividad": Unkn
            var ctxFacturacion = document.getElementById('chartFacturacion').getContext('2d');    "Facturacion": Unkn

            // Crea los gráficos    "Crea": Unknown word.
            var chartEntrega = new Chart(ctxEntrega, {
                type: 'doughnut',
                data: chartDataEntrega,
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                }
            });

            var chartInactividad = new Chart(ctxInactividad, {    "Inactividad": Unknown word.
                type: 'doughnut',
                data: chartDataInactividad,
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                }
            });

            var chartFacturacion = new Chart(ctxFacturacion, {    "Facturacion": Unknown word.
                type: 'doughnut',
                data: chartDataFacturacion,
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                }
            });
        </script>
    </div>

    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Figura 72

Historial inventario

history_inventory.html 4 X

templates > history_inventory.html > html > body > div.container.mt-3.text-center > div.mt-3 > div.table-responsive > table.table > tbody#tableBody

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <title>Administration Historial Inventario</title>    "Historial": Unknown word.
    <style>
        .chart-container {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/administration">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/administration">Panel de Administración</a>    "Administración": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3 text-center">
        <h3>Historial de Inventario de Noviembre</h3>          "Historial": Unknown word.
        <div class="mt-3">
            <label for="dayFilter">Seleccione el dia:</label>    "Seleccione": Unknown word.
            <select id="dayFilter" class="form-control" onchange="updateChart()">
                {% for day in data_for_inventory.keys() %}
                    <option value="{{ day }}">{{ day }}</option>
                {% endfor %}
            </select>
```

```
<h4 class="mt-3">Tabla de Cantidades por Ingrediente</h4>    "Cantidades": Unknown word.
<div class="table-responsive">
    <table class="table">
        <thead>
            <tr>
                <th scope="col">Ingrediente</th>    "Ingrediente": Unknown word.
                <th scope="col">Cantidad</th>    "Cantidad": Unknown word.
            </tr>
        </thead>
        <tbody id="tableBody">

        </tbody>
    </table>
</div>
</div>

<script>
    var dataForInventory = {{ data_for_inventory|tojson|safe }};    Property assignment expected.
    var currentDay = 'Dia 1';

    function updateChart() {
        currentDay = document.getElementById('dayFilter').value;
        updateTable();
    }

    function updateTable() {
        var tableBody = document.getElementById('tableBody');
        tableBody.innerHTML = '';

        Object.entries(dataForInventory[currentDay]).forEach(([ingredient, quantity]) => {
            var ingredientRow = document.createElement('tr');
            var ingredientCell = document.createElement('td');
            var quantityCell = document.createElement('td');

            ingredientCell.textContent = ingredient;
            quantityCell.textContent = quantity;

            ingredientRow.appendChild(ingredientCell);
            ingredientRow.appendChild(quantityCell);

            tableBody.appendChild(ingredientRow);
        });
    }

    document.addEventListener('DOMContentLoaded', function () {
        updateTable();
    });
</script>
</div>
```

```
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Figura 73

Cashier

cashier.html

templates > cashier.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a >Caja</a>    "Caja": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-3">
        <h2 class="text-center">Monto de Apertura</h2>    "Monto": Unknown word.

        <div class="container d-flex justify-content-center">
            <input type="tel" id="texto" placeholder="Introduce el número" pattern="^\d+$">    "número": Unknown word.
        </div>

        <div class="row mt-3">
            <div class="col-md-6 mx-auto">
                <a href="/cashier/commands" class="btn btn-primary btn-block mb-2">Abrir Caja</a>    "Abrir": Unknown word.
            </div>
        </div>
    </div>
</body>
</html>
```

Figura 74

Close (para el cajero)

close.html

templates > close.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <title>Cerrar Caja</title>    "Cerrar": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Cerrar Caja</h2>    "Cerrar": Unknown word.

        <table class="table">
            <thead>
                <tr>
                    <th scope="col">Concepto</th>    "Concepto": Unknown word.
                    <th scope="col">Monto</th>    "Monto": Unknown word.
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Tarjeta</td>    "Tarjeta": Unknown word.
                    <td>${{ total_card }}</td>
                </tr>
                <tr>
                    <td>Efectivo</td>    "Efectivo": Unknown word.
                    <td>${{ total_cash }}</td>
                </tr>
                <tr>
                    <td>Transferencia</td>    "Transferencia": Unknown word.
                    <td>${{ total_transfer }}</td>
                </tr>
                <tr>
                    <td>Total</td>
                    <td>${{ total_all }}</td>
                </tr>
            </tbody>
        </table>

        <button class="btn btn-success btn-block">Cerrar día</button>    "Cerrar": Unknown word.
        <button class="btn btn-danger btn-block" onclick="redirectToCommands()">Cancelar</button>        "Cancelar": Unknown word.
    </div>
```

```
    <script>
        function redirectToCommands() {
            window.location.href = '/cashier/commands';
        }
    </script>
</body>
</html>
```

Figura 75

Comandas x Cobrar

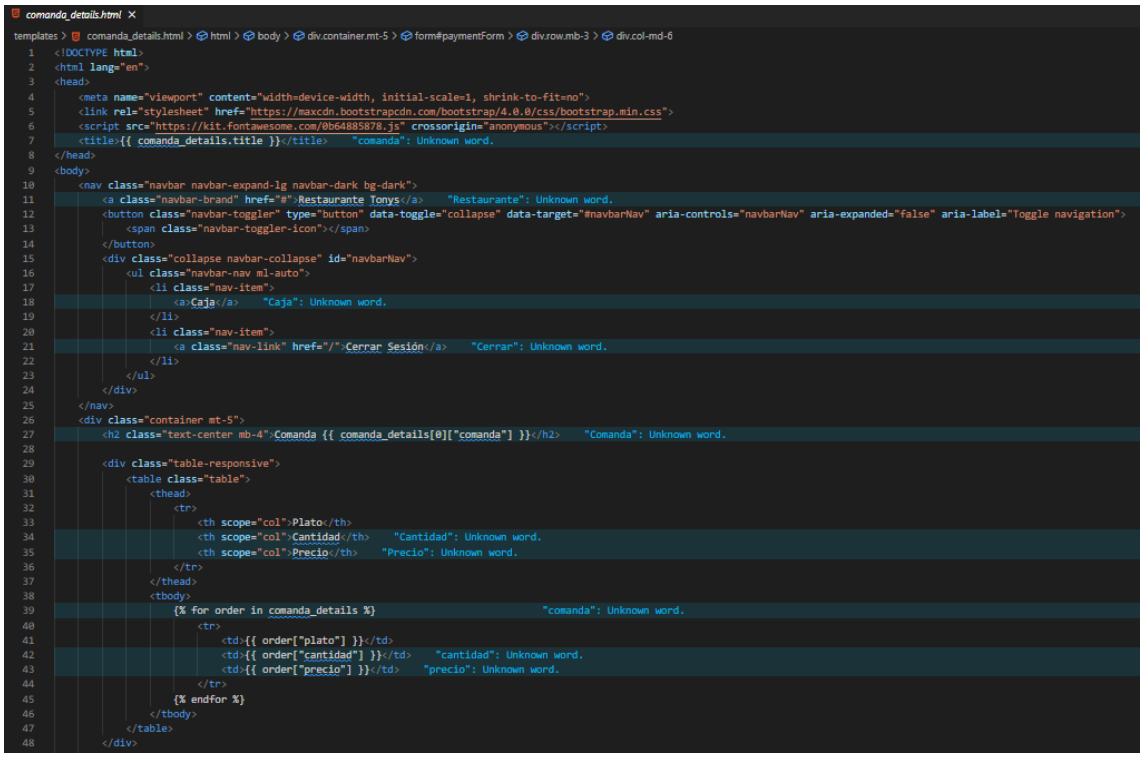
commands.html

templates > commands.html > html > body > div.container.mt-5 > div.mt-4.mb-4

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://kit.fontawesome.com/0b64885878.js" crossorigin="anonymous"></script>
    <title>Comandas</title>    "Comandas": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a >Caja</a>    "Caja": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Comandas</h2>    "Comandas": Unknown word.

        <ul class="list-group">
            {% for command_number in id_comandas %}    "comandas": Unknown word.
                <button class="list-group-item list-group-item-action d-flex justify-content-between align-items-center" onclick="handleButtonClick('{{ command_number }}')">
                    Comanda {{ command_number }}    "Comanda": Unknown word.
                </button>
            {% endfor %}
        </ul>

        <div class="mt-4 mb-4">
            <h4 class="text-center">Total Caja</h4>    "Caja": Unknown word.
            <div class="border border-primary p-3 text-center">
                $5000
            </div>
        </div>

        <button class="btn btn-danger btn-block" onclick="handleCloseButtonClick()">Cerrar Caja</button>    "Cerrar": Unknown word.
    </div>

    <script>
        function handleButtonClick(comandaId) {                "comanda": Unknown word.
            window.location.href = '/commands/' + comandaId;    "comanda": Unknown word.
        }

        function handleCloseButtonClick() {
            window.location.href = '/cashier/close';
        }
    </script>
</body>
</html>
```

Figura 76

Detalle de comandas



```
comanda_details.html
templates > comanda_details.html > html > body > div.container.mt-5 > form#paymentForm > div.row.mb-3 > div.col-md-6
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://kit.fontawesome.com/0b64885878.js" crossorigin="anonymous"></script>
    <title>{{ comanda_details.title }}</title>     "comanda": Unknown word.
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>     "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a>Caja</a>     "Caja": Unknown word.
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>     "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Comanda {{ comanda_details[0]["comanda"] }}</h2>     "Comanda": Unknown word.

        <div class="table-responsive">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">Plato</th>
                        <th scope="col">Cantidad</th>     "Cantidad": Unknown word.
                        <th scope="col">Precio</th>     "Precio": Unknown word.
                    </tr>
                </thead>
                <tbody>
                    {% for order in comanda_details %}                          "comanda": Unknown word.
                        <tr>
                            <td>{{ order["plato"] }}</td>
                            <td>{{ order["cantidad"] }}</td>     "cantidad": Unknown word.
                            <td>{{ order["precio"] }}</td>     "precio": Unknown word.
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
```

```
        <div class="d-flex justify-content-center mb-4">
            <button class="btn btn-outline-primary p-3 mr-2"><i class="fa-solid fa-box-archive"></i></button>
            <button class="btn btn-outline-primary p-3 mr-2"><i class="fas fa-box"></i></button>
            <button class="btn btn-outline-danger p-3"><i class="fas fa-eraser"></i></button>
        </div>

        <form id="paymentForm" method="POST" action="{{ url_for('process_payment', comanda_id=comanda_details[0]['comanda']) }}">
            <div class="row mb-3">
                <div class="col-md-6">
                    <div class="input-group">
                        <div class="input-group-prepend">
                            <label class="input-group-text" for="paymentMethodSelect">Método de pago:</label>     "Método": Unknown
                        </div>
                        <select class="custom-select" id="paymentMethodSelect" name="payment_method">
                            <option selected>Seleccionar...</option>     "Seleccionar": Unknown word.
                            <option value="efectivo">Efectivo</option>     "efectivo": Unknown word.
                            <option value="tarjeta">Tarjeta de crédito</option>     "tarjeta": Unknown word.
                            <option value="transferencia">Transferencia</option>     "transferencia": Unknown word.
                        </select>
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="input-group">
                        <div class="input-group-prepend">
                            <span class="input-group-text">Total:</span>
                        </div>
                        <span class="input-group-text">${{ comanda_details[-1] }}</span>
                    </div>
                </div>
            </div>

            <button type="submit" class="btn btn-success btn-block" id="cobrarButton">Cobrar</button>     "Cobrar": Unknown word.
        </form>

        <button class="btn btn-danger btn-block" onclick="redirectToCommands()">Cancelar</button>     "Cancelar": Unknown word.

    </div>

    <script>
        function redirectToCommands() {
            window.location.href = '/cashier/commands';
        }
    </script>
</body>
</html>
```

Figura 77

Cocina Comanda detalle

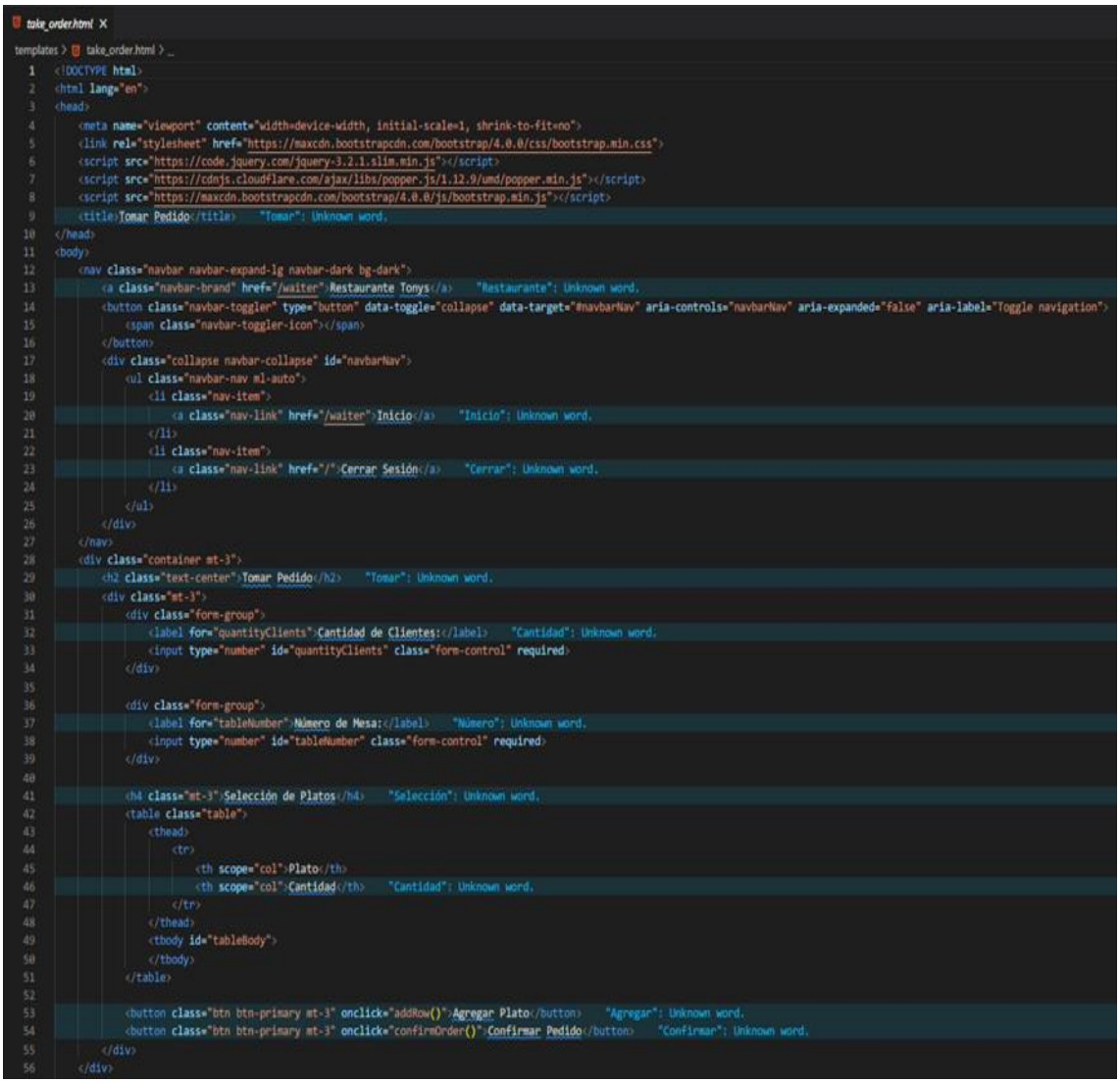
```
templates > kitchen_Mesa.html > html > body > div.container.mt-3
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <title>Administration</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Restaurante Tonys</a>    "Restaurante": Unknown word.
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
                </li>
            </ul>
        </div>
    </nav>

    <div class="container mt-3">
        <h2 class="text-center">Mesa</h2> <!-- Sale la mesa segun la eleccion -->    "segun": Unknown word.
    </div>

    <div class="container mt-3 d-flex ">
        <table class="table ">
            <thead>
                <tr>
                    <td>Plato</td>
                    <td>Cantidad</td>    "Cantidad": Unknown word.
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Celda 3</td>    "Celda": Unknown word.
                    <td>Celda 4</td>    "Celda": Unknown word.
                </tr>
            </tbody>
        </table>
    </div>

    <div class="container d-flex justify-content-center">
        <button class="btn btn-primary mt-3" onclick="addRow()">Entregar Mesa</button>    "Entregar": Unknown word.
    </div>
</body>
</html>
```

Figura 78

Cocina

Figura 79

Mesero Tomar Orden



```
57
58      <script>
59          function confirmOrder() {
60              var tableData = [];
61              var tableBody = document.getElementById('tableBody');
62              var rows = tableBody.getElementsByTagName('tr');
63
64              for (var i = 0; i < rows.length; i++) {
65                  var cells = rows[i].getElementsByTagName('td');
66                  var plateName = cells[0].textContent;
67                  var quantity = cells[1].textContent;
68                  tableData.push({ plate: plateName, quantity: quantity });
69              }
70
71              var quantityClients = document.getElementById('quantityClients').value;
72              var tableNumber = document.getElementById('tableNumber').value;
73
74              console.log("Table Data:", tableData);
75              console.log("Quantity of Clients:", quantityClients);
76              console.log("Table Number:", tableNumber);
77
78              alert("Pedido confirmado. ¡Buen provecho!");    "Pedido": Unknown word.
79          }
80
81          function addRow() {
82              var plateName = prompt("Ingrese el nombre del plato:");    "Ingrese": Unk
83              var quantity = prompt("Ingrese la cantidad:");    "Ingrese": Unknown word
84
85              var tableBody = document.getElementById('tableBody');
86              var newRow = tableBody.insertRow();
87              var plateCell = newRow.insertCell(0);
88              var quantityCell = newRow.insertCell(1);
89
90              plateCell.textContent = plateName;
91              quantityCell.textContent = quantity;
92          }
93      </script>
94  </body>
95  </html>
96
```

Figura 80

Mesero codificación

waiter.html

templates > waiter.html > html > body > div.container.mt-3.text-center > div.d-flex.justify-content-center > button.btn.btn-secondary

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
5       <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
6       <title>Mesero</title>    "Mesero": Unknown word.
7   </head>
8   <body>
9       <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
10          <a class="navbar-brand" href="/waiter">Restaurante Tonys</a>    "Restaurante": Unknown word.
11          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
12              <span class="navbar-toggler-icon"></span>
13          </button>
14          <div class="collapse navbar-collapse" id="navbarNav">
15              <ul class="navbar-nav ml-auto">
16                  <li class="nav-item">
17                      <a class="nav-link" href="/waiter">Inicio</a>    "Inicio": Unknown word.
18                  </li>
19                  <li class="nav-item">
20                      <a class="nav-link" href="/">Cerrar Sesión</a>    "Cerrar": Unknown word.
21                  </li>
22              </ul>
23          </div>
24      </nav>
25      <div class="container mt-3 text-center">
26          <h2 class="mb-4">Mesero</h2>    "Mesero": Unknown word.
27          <div class="d-flex justify-content-center">
28              <button class="btn btn-primary mr-3" onclick="redirectTo('/waiter/take_order')">Tomar Pedido</button>    "Tomar": Unknown word.
29              <button class="btn btn-secondary" onclick="redirectTo('/waiter/edit_order')">Editar Pedido</button>    "Editar": Unknown word.
30          </div>
31      </div>
32
33      <script>
34          function redirectTo(url) {
35              window.location.href = url;
36          }
37      </script>
38
39      <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
40      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
41      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
42  </body>
43  </html>
```

3.3.5 Etapa Elección

Se eligió MongoDB por su flexibilidad con los datos, ya que utiliza un modelo de datos de tipo documento y colecciones. Esto permite una mejor facilidad de uso, ya que no requiere realizar muchos cambios técnicos para manejar información que varía cada día. Al momento de la creación, se eligió que la base de datos se alojara en AWS, la empresa más grande del mundo en alojamiento de servidores web, que ofrece respaldo las 24 horas del día.

De donde se resalta que este servicio no tiene costo alguno mientras el uso del espacio principal no supere los 541 MB, siendo adecuado para el prototipo. Una vez que se finalice el prototipo en el futuro, el costo será de $0,17 por cada 1.000.000 de peticiones, para el plan básico, eligiendo el servidor de São Paulo por su mejor respuesta y mayor conectividad.

Figura 81

AWS

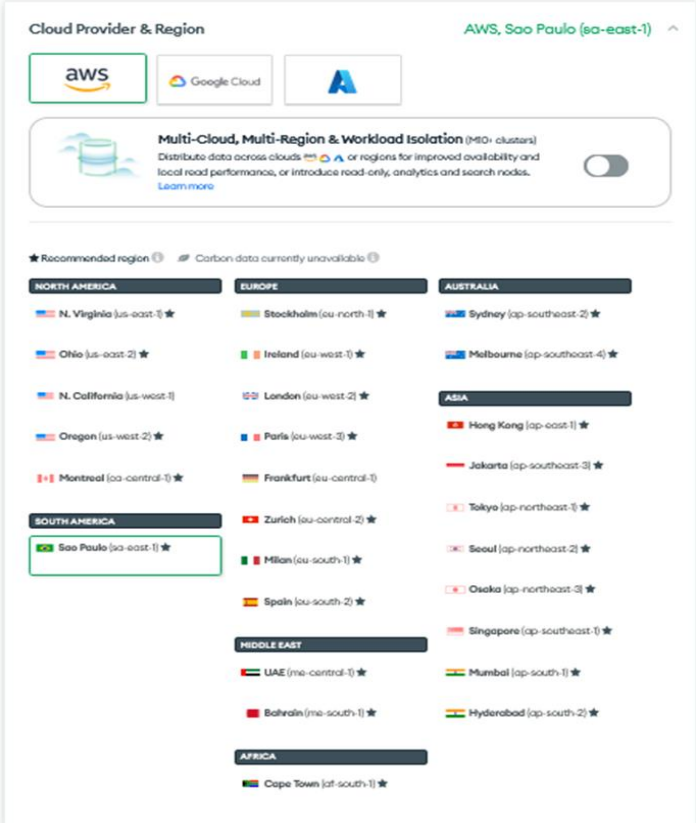
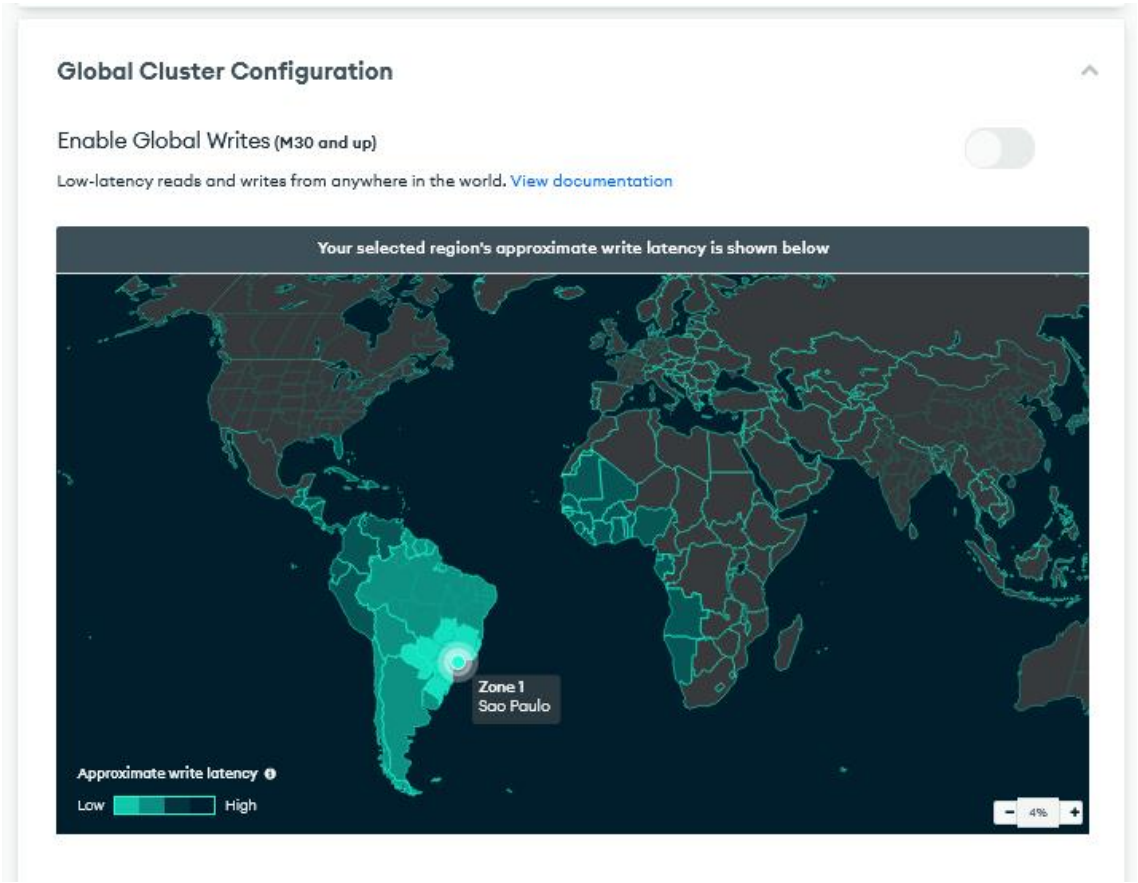
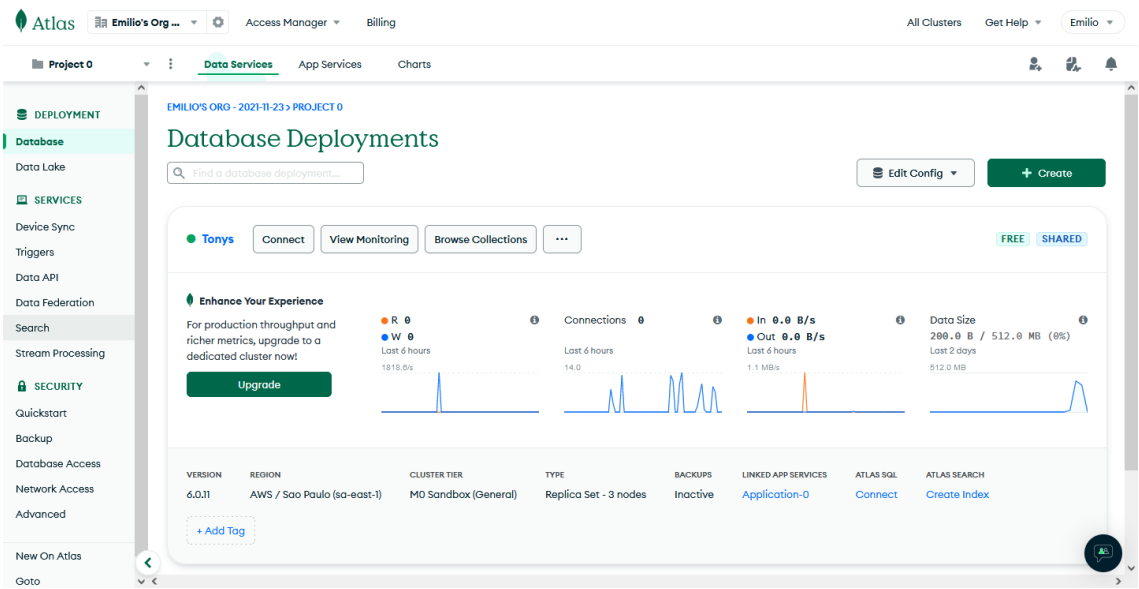


Figura 82

Global Cluster

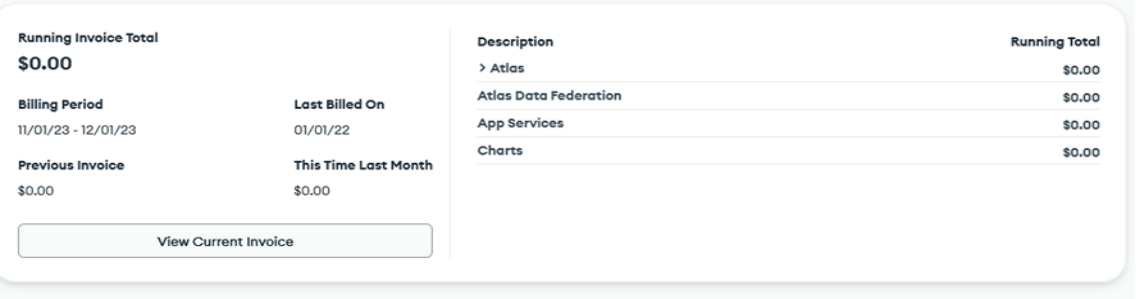




Las imágenes presentadas ilustran dos aspectos clave de la administración de bases de datos en la plataforma MongoDB Atlas, la primera muestra la "Configuración Global del Cluster", destacando la opción de habilitar escrituras globales y proporcionando un mapa mundial que indica la latencia aproximada de escritura para la región seleccionada, en este caso, São Paulo. La segunda imagen detalla la sección de "Despliegues de Base de Datos", donde se visualizan las métricas y el estado de un despliegue de base de datos específico. Aquí se incluyen gráficos de conexiones, uso de CPU, y almacenamiento de datos, así como opciones para conectar, monitorear y gestionar colecciones. Juntas, estas imágenes subrayan la capacidad de MongoDB Atlas para ofrecer una gestión eficiente y global de bases de datos, facilitando la optimización del rendimiento y la disponibilidad a nivel mundial

Figura 83

Costo

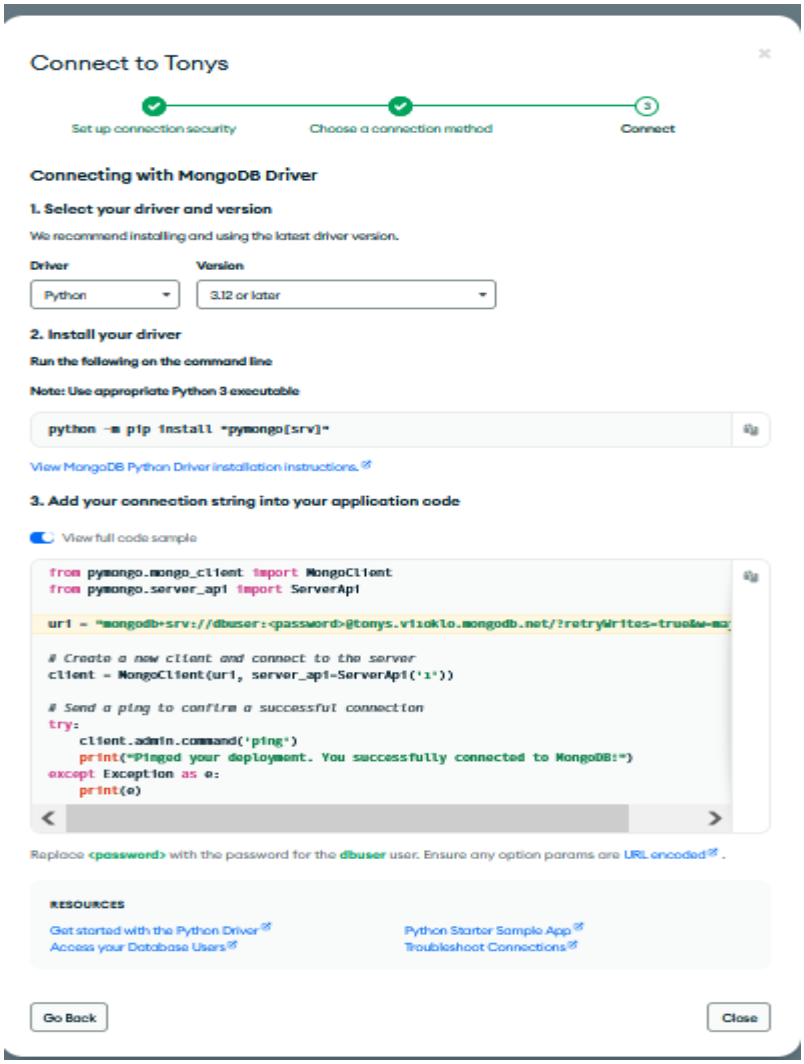


La imagen muestra la interfaz de resumen de facturación de la plataforma MongoDB Atlas. En la sección izquierda, se presenta un desglose de la facturación total acumulada, que en este caso es de $0.00. Se especifican el período de facturación actual (del 01/11/23 al 01/12/23), la fecha de la última factura emitida (01/01/22), el monto de la factura anterior ($0.00) y el monto facturado en el mismo período del mes anterior ($0.00). Un botón permite ver la factura actual. En la sección derecha, se detallan las descripciones de los servicios facturados, como Atlas, Atlas Data Federation, App Services y Charts, todos con un costo de $0.00, esta pantalla proporciona una vista clara y concisa del estado de facturación,

facilitando la gestión financiera y el monitoreo de los costos asociados con el uso de los servicios de MongoDB Atlas.

Figura 84

Conectividad



La imagen muestra una guía paso a paso para conectar la aplicación del restaurante "Tonys" a su base de datos MongoDB utilizando un controlador (driver), de donde se resalta cada paso:

- Seleccionar el Driver y la Versión: Se elige el driver de Python, versión 3.12 o superior.

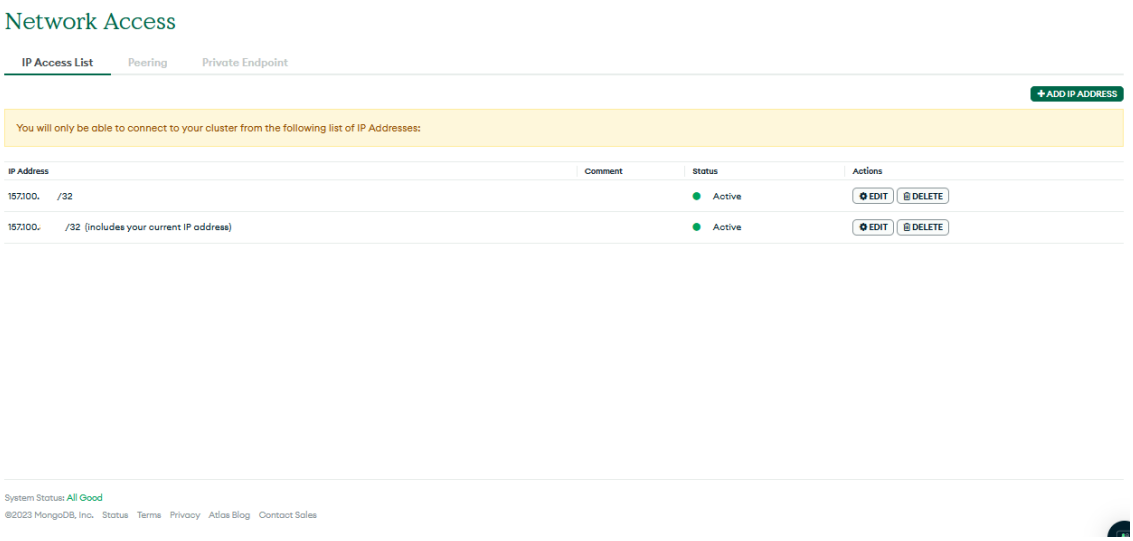
- Instalar el Driver: Se proporciona un comando que los desarrolladores deben ejecutar para instalar el driver necesario.
- Agregar la Cadena de Conexión al Código: Instrucciones claras sobre cómo incluir la cadena de conexión en la aplicación para establecer la conexión con la base de datos.

Los recursos adicionales están disponibles para más detalles, incluyendo documentación y guías de solución de problemas. Esta guía facilita la configuración rápida y precisa de la conexión entre la aplicación y la base de datos MongoDB.

Además, en el apartado de acceso a la red, se eligió permitir solo a las direcciones IP públicas establecidas por el administrador. Estas direcciones IP fueron las de la computadora y la laptop del administrador, las cuales se utilizaron para realizar pruebas de conexión. Esto permite tener un control similar a una lista blanca sobre quién tiene acceso a la red de la base de datos. La conexión con MongoDB Atlas fue muy sencilla, ya que la plataforma proporcionaba el código necesario para el programa en Python, el cual debía estar instalado en la computadora.

Figura 85

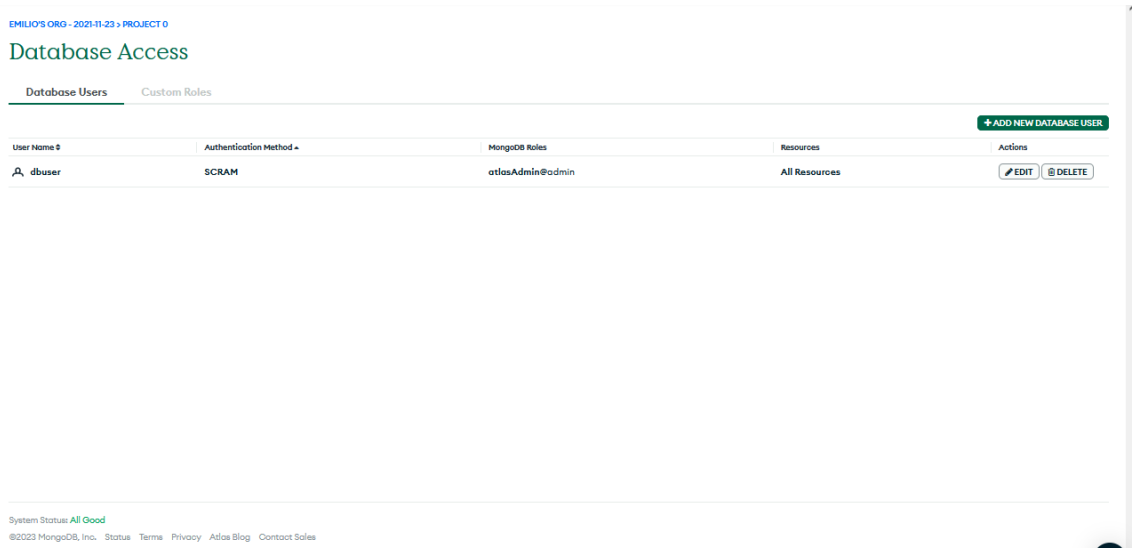
Acceso a la red



Para el acceso a la base de datos, no es necesario realizar ninguna acción compleja, basta con crear un usuario y una contraseña que serán el super-usuario de toda la base de datos.

Figura 86

Acceso a la base de datos



La imagen muestra la interfaz de "Database Access" en MongoDB Atlas, donde se gestionan los usuarios de la base de datos, de donde se explica cada componente a continuación:

- Usuarios de la Base de Datos: Lista los usuarios con sus nombres de usuario, métodos de autenticación, permisos, roles asignados y opciones de acción (editar o eliminar).
- Añadir Nuevo Usuario: Botón para agregar un nuevo usuario de la base de datos.

Esta interfaz permite a los administradores gestionar de manera eficiente los permisos y el acceso de los usuarios a la base de datos, asegurando que solo el personal autorizado pueda interactuar con los datos.

Figura 87

Modelo de documentos

La imagen muestra la estructura de la base de datos del restaurante en MongoDB Atlas, representada en un menú de navegación. Los documentos se organizan en las siguientes colecciones:

- restaurante: Colección principal que agrupa todas las demás colecciones relacionadas con la gestión del restaurante.
- comandas: Documentos que almacenan información sobre los pedidos realizados por los clientes.
- ingredientes: Documentos que detallan los ingredientes disponibles y sus cantidades.
- pagos: Documentos que registran las transacciones y pagos realizados.
- platos: Documentos que contienen información sobre los platos ofrecidos en el menú.
- temp: Colección temporal para datos intermedios o de uso provisional.
- usuarios: Documentos que gestionan la información de los usuarios que acceden al sistema.

Esta estructura de documentos facilita una organización eficiente y acceso rápido a diferentes tipos de datos esenciales para la operación diaria del restaurante.

Figura 88

Query Results

3.3.6 Monitorear

Para monitorear se presenta a continuación el proceso desde el inicio que se recepta la comanda.

Figura 89

Toma de pedido

Todos los pedidos se guardan con un ID que corresponde al número de comanda, este caso los 3 platos pertenecen a la comanda 1, cuando confirma el pedido el mesero pone la fecha y hora en la base de datos de cuando se receptó.

Figura 90

ID Toma de pedido

Al momento que el mesero recepta la comanda, pasa a cocina y se muestra primero en el panel de comandas, después cocina tiene que elegir el orden de comandas que se está ejecutando. Cuando entrega a la mesa se detiene el tiempo por medio de código y le guarda la base de datos cuanto tiempo se demoró.

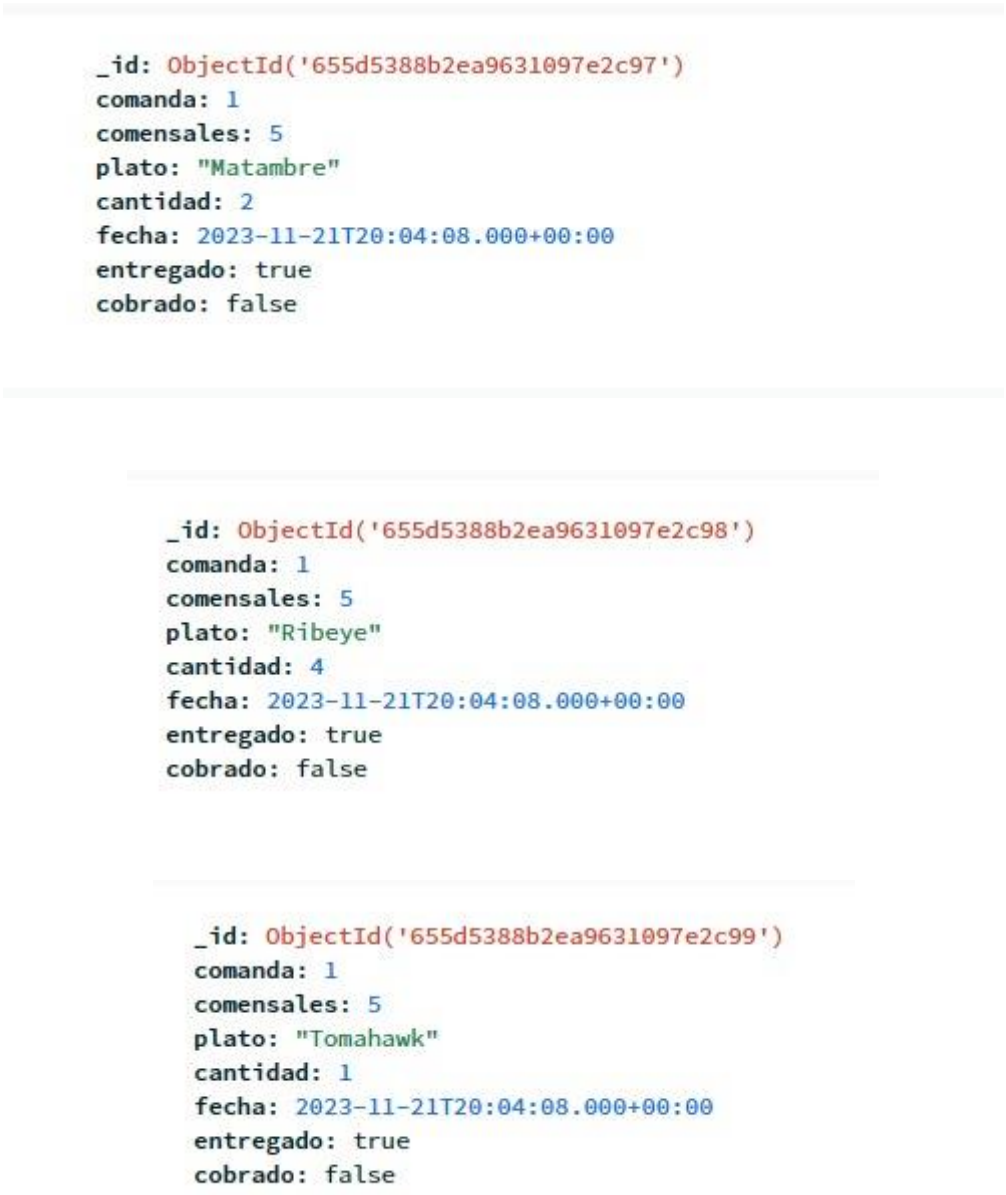
Figura 91

Panel de Comandas Cocina

Cuando se entrega a la mesa en la base de datos la variable de entregado pasa a un estado de TRUE.

Figura 92

ID Panel de Comandas Cocina



Una vez entregada la mesa pasa a Caja, en la caja se tienen todas las comandas por cobrar.

Figura 93

Panel de Comandas Cobro

Se puede verificar la comanda 1 y ver el total. Luego, puede elegir el método de pago y decidir si el cliente quiere un empaque pequeño o grande.

Figura 94

Panel de Comandas sin empaque para llevar

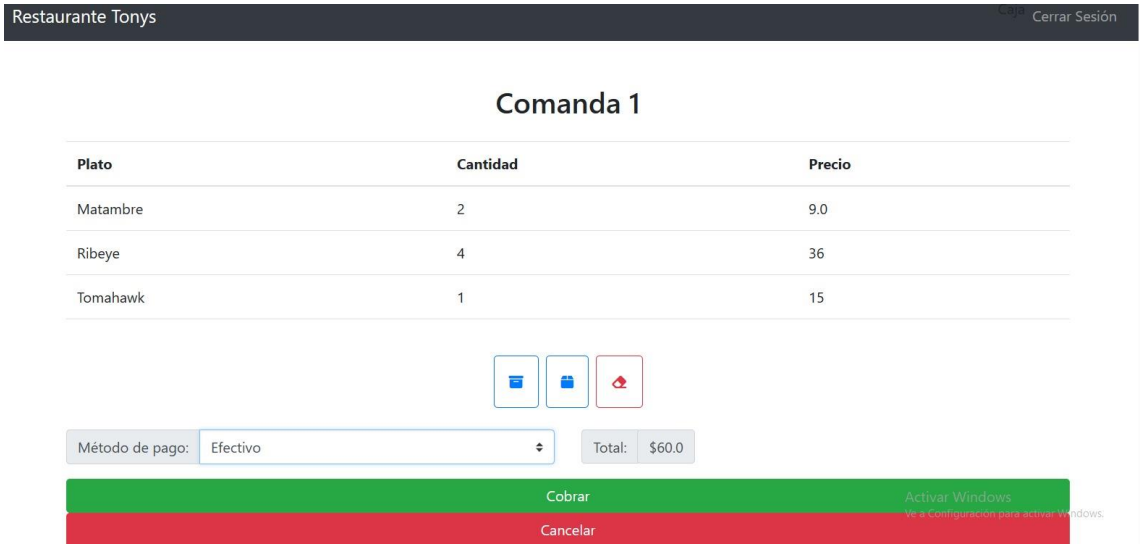


Figura 95

Panel de Comandas con empaque para llevar

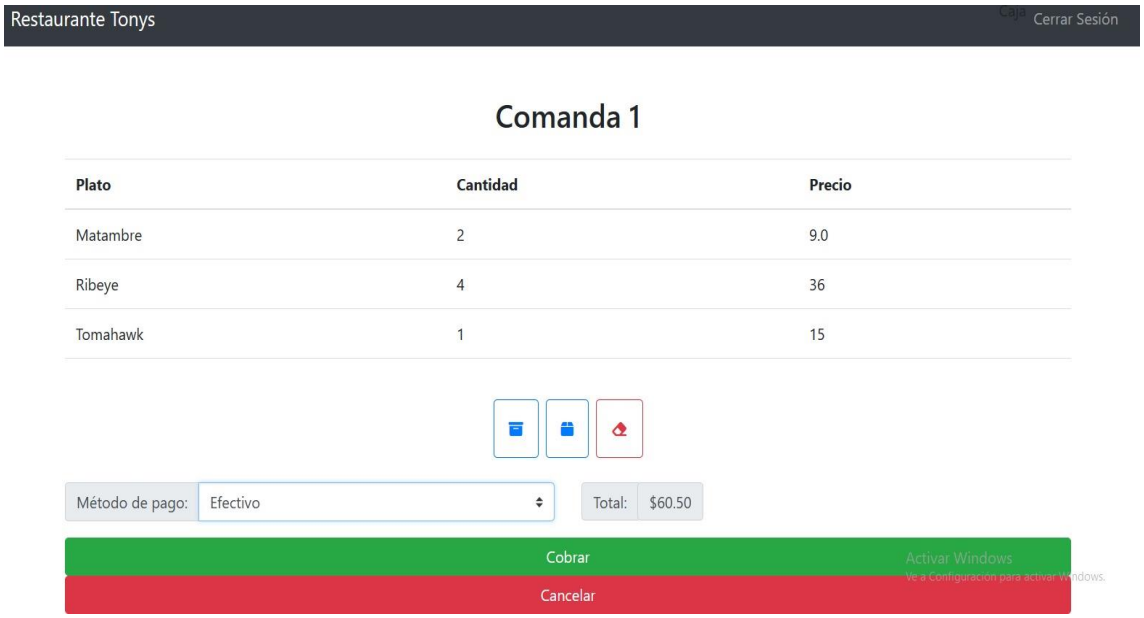
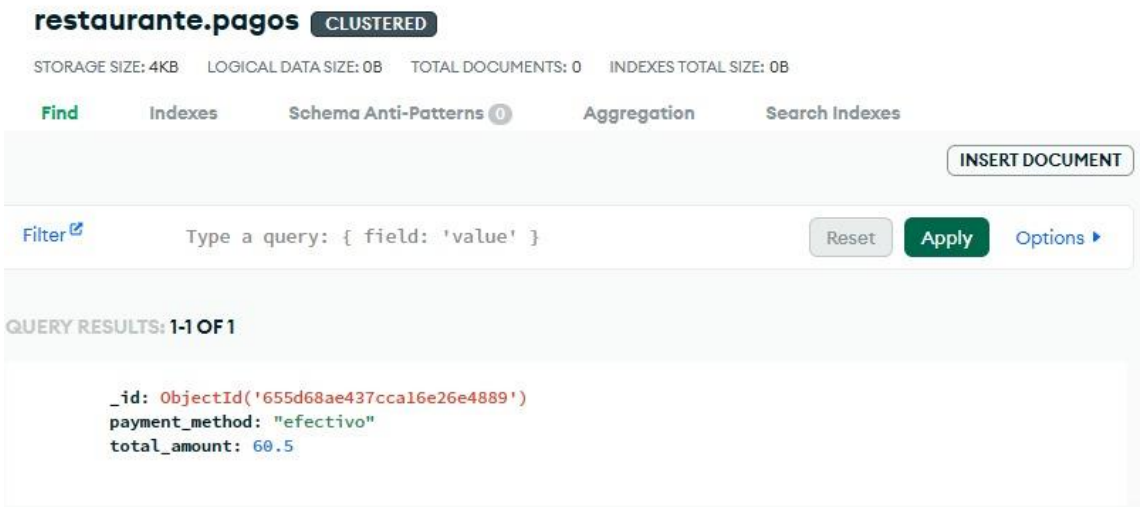


Figura 96

Pago cobrado



Una vez agregado el cobro la comanda tambien pasa de a un estado TRUE de cobrado.

Figura 97

ID pago cobrado

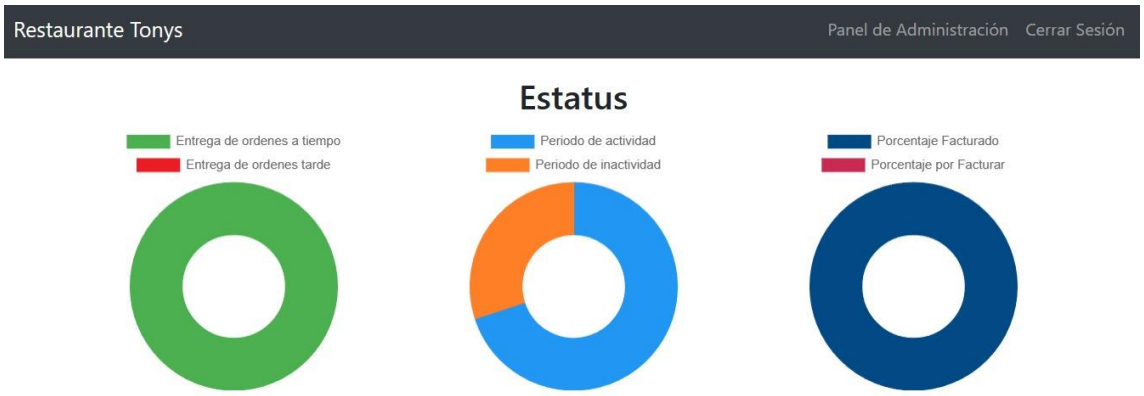

```
_id: ObjectId('655d5388b2ea9631097e2c97')
comanda: 1
comensales: 5
plato: "Matambre"
cantidad: 2
fecha: 2023-11-21T20:04:08.000+00:00
entregado: true
cobrado: true
```

```
_id: ObjectId('655d5388b2ea9631097e2c98')
comanda: 1
comensales: 5
plato: "Ribeye"
cantidad: 4
fecha: 2023-11-21T20:04:08.000+00:00
entregado: true
cobrado: true
```

```
_id: ObjectId('655d5388b2ea9631097e2c99')
comanda: 1
comensales: 5
plato: "Tomahawk"
cantidad: 1
fecha: 2023-11-21T20:04:08.000+00:00
entregado: true
cobrado: true
```


Figura 98

Estatus



Una vez cobrado se puede visualizar en el total en la pantalla para proceder a notificar al cliente.

Figura 99

Caja total

Se puede elegir la opción de cerrar el día el cual el cajero no puede modificar el dinero, de esta manera tiene que cuadrarle la caja con lo que tenga el sistema.

Figura 100

Cierre de Caja

En el panel del Administrador se puede observar opciones como: En vivo, Estatus, Historial de Cajas, Inventario e Historial de Inventario.

Figura 101

Panal de Administración

En el apartado de “En Vivo” se puede observar antes del cobro los 5 clientes que recepto el mesero, las comandas pendientes por cobrar y el dinero en caja.

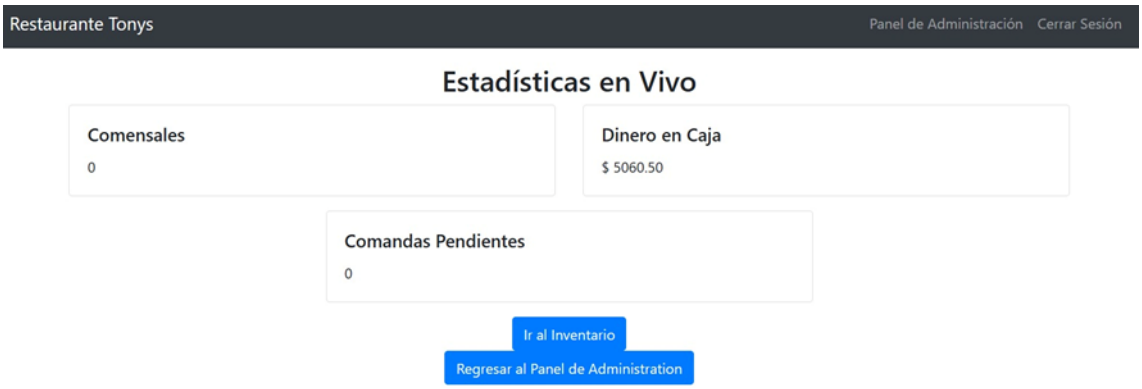
Figura 102

Estadísticas en vivo

Una vez cobrado se quitan los comensales, las comandas pendientes y se aumentaría la caja.

Figura 103

Estadísticas en vivo



En el apartado “Estatus” se puede observar los KPI que tiene por el momento la cocina, caja y periodo de inactividad.

- Cocina: Se pone la entrega de órdenes a tiempo que sean de 15 min a menos tiempo, esa es la bandera de tiempo que eligió el Restaurante Tonys para poder trabajar con esta medida resaltando que cualquier restaurante puede definir estas banderas según sus necesidades.
- Inactividad: En este apartado se visualiza el tiempo de inactividad que tienen los empleados al no recibir una comanda por cierto tiempo. De esta manera puede ver cuánto tiempo pueden usar los de la cocina para hacer producción o mise en place.
- Porcentaje facturado: Esta es una constante la cual el administrador pone una meta fija por semana, por ejemplo, la constante de esta semana es facturar 5.000 dólares, al momento que se va pagando va comenzando a subir el porcentaje, de esta manera el administrador tiene en cuenta cómo está desempeñándose semanalmente.

Todos estos KPIs son establecidos por el administrador de TONYS ya que es la necesidad que por el día de hoy requieren cubrir y saber cómo están yendo, se podría decir que son sus pilares para el manejo de su restaurante.

Figura 104

Estatus

En el apartado de historial cajas muestra una estadística de cómo van las cajas por día y si es que están llegando a la meta correspondiente, se va a hacer una simulación de cómo se vería desde el día 1 de noviembre hasta el 21 que se cobró los $60,50.

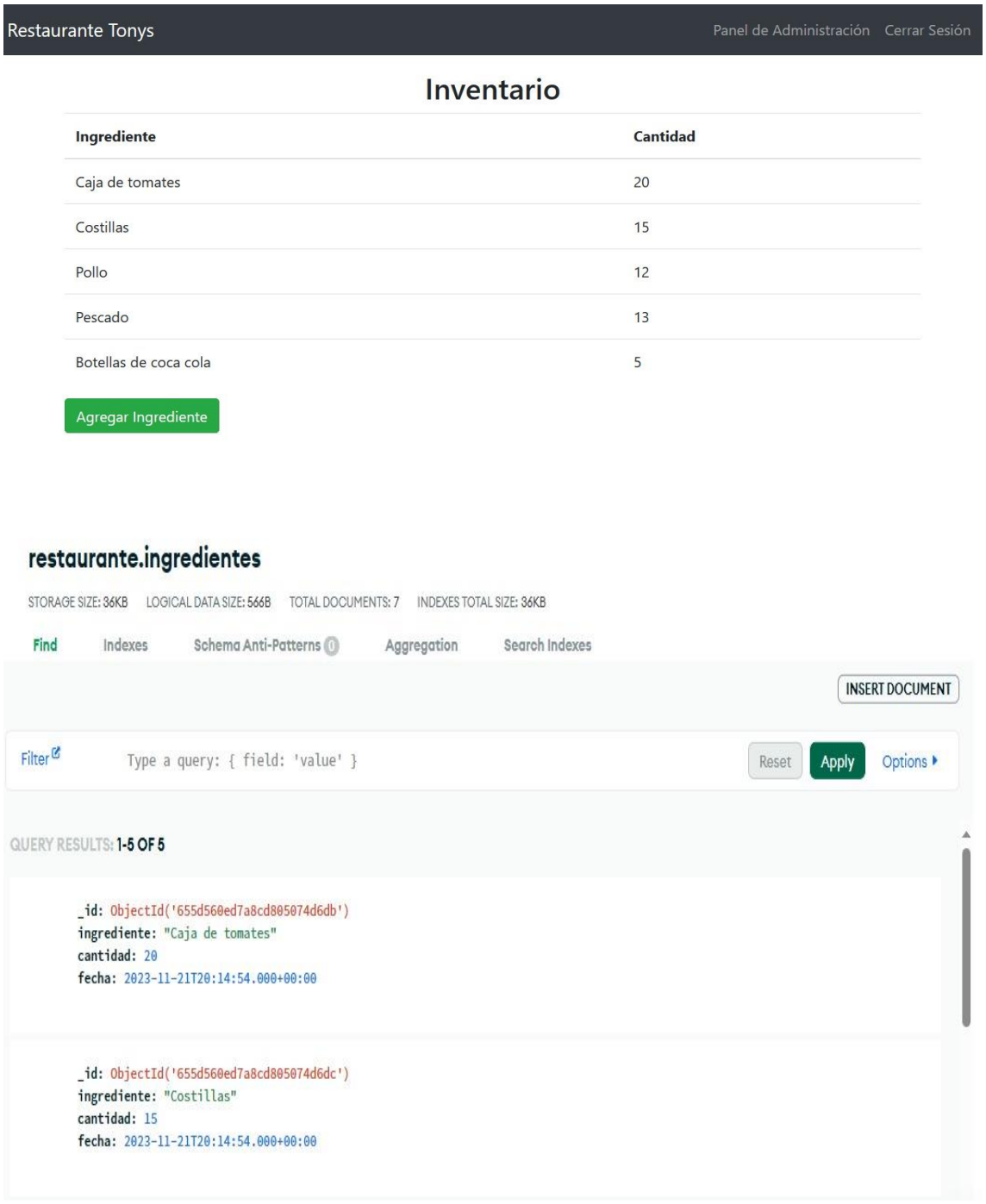
Figura 105

Historial de caja

Para el apartado del inventario se observa los ingredientes que se van a entregar ya que el administrador es el mismo proveedor, de esta manera tiene el control de cuanto se usa semanalmente.

Figura 106

Inventario



```
_id: ObjectId('655d5641ceb96fdd87d5d0e3')
ingrediente: "Pollo"
cantidad: "12"
fecha: 2023-11-21T20:15:45.000+00:00
```

```
_id: ObjectId('655d5659ceb96fdd87d5d0e5')
ingrediente: "Pescado"
cantidad: "13"
fecha: 2023-11-21T20:16:09.000+00:00
```

```
_id: ObjectId('655d5697809e4495569ebd6b')
ingrediente: "Botellas de coca cola"
cantidad: "5"
fecha: 2023-11-21T20:17:11.000+00:00
```

Capítulo 4: Conclusiones y Recomendaciones

4. Conclusiones y Recomendaciones

4.1 Conclusiones

- El proyecto demuestra la factibilidad de utilizar tecnologías web modernas para la gestión eficiente de restaurantes, incluyendo bases de datos en la nube, frameworks de desarrollo web, y aplicaciones móviles complementarias. El prototipo muestra flexibilidad para adaptarse a diferentes tipos y tamaños de restaurantes, lo que sugiere una posible escalabilidad para integrar más funcionalidades y manejar una mayor carga de trabajo en el futuro.

- El desarrollo de Aplicación Prototipo Web para la Gestión Integral de Restaurantes se basó en la automatización de los procesos clave como la gestión de pedidos, inventario, reservas y pagos. Por lo tanto, la aplicación podría simplificar tareas administrativas, permitiendo una atención más enfocada en la experiencia del cliente.

- Flask y MongoDB Atlas son una combinación adecuada para desarrollar el software para el restaurante, por la flexibilidad, la escalabilidad, la velocidad de desarrollo y la capacidad de manejar datos no estructurados y semiestructurados de manera eficiente como es la información requerida en el negocio en cuanto a menús, pedidos, clientes e inventario.

- Finalizando puedo decir que la retroalimentación constante de los usuarios del software va a permitir una mejora continua del sistema. Esto asegura que el software se mantenga relevante y útil para los restaurantes, adaptándose a las necesidades cambiantes del mercado y de los clientes, por el cual se eligió la metodología AGILE el cual nos permite hacer cambios rápidos de acuerdo a lo que se necesita al momento.

4.2 Recomendaciones

- Es importante recopilar datos de forma precisa y consistente para que los análisis sean precisos. Esto se puede hacer utilizando dispositivos de medición, como cámaras, o mediante la recopilación manual de datos de la entrada y salida de los clientes y como esta su satisfacción.
- Es importante analizar las áreas de mejora. Esto se puede hacer utilizando herramientas de análisis de datos o mediante la creación de informes personalizados por cada uno de los usuarios de cada área.
- Se recomienda que el personal del restaurante reciba capacitación continua en el uso del nuevo software. Esto no solo garantizará una utilización óptima de todas las funcionalidades del sistema, sino que también facilitará la adaptación a futuras actualizaciones o cambios en el software.
- Se recomienda explorar la implementación de módulos adicionales que puedan integrarse con el software principal. Por ejemplo, módulos de gestión de inventarios, gestión de reservas y análisis de satisfacción del cliente, los cuales podrían ofrecer una visión más integral y detallada del rendimiento y las áreas de mejora del restaurante.
- Es recomendable establecer un protocolo de monitoreo continuo y mantenimiento regular del software para asegurar su óptimo funcionamiento. Esto incluye actualizaciones periódicas, corrección de errores y ajustes necesarios según las necesidades cambiantes del restaurante. Un sistema bien mantenido puede prevenir problemas técnicos y mejorar la eficiencia a largo plazo

Bibliografía

Addison-Wesley. (2019). *Cloud Databases: A Practical Guide.*

Allen, J. (2019). *The Foodservice Industry: A Guide to the Business of Foodservice de James E. Allen. .*

Castro, Y., Solarte, G., & Muñoz, L. (2019). Planificación, Gestión y Control de la Calidad del Software. *Scientia Et Technica*, 611-617.

Gartner. (2023). *The State of the Cloud Database Market 2023.*

Grinberg, M. (2018). *Flask Web Development.*

Pardo, C., & Rodil, I. (2022). *Operaciones auxiliares con tecnologías de la información y la comunicación.* Madrid: Ediciones Paraninfo, S.A.

Ronacher, A. (2015). *Flask: The Python Microframework.*

Snyder, C. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* Morgan Kaufmann.

Sommerville, I. (2011). *Software engineering.* Addison-Wesley. Obtenido de https://www.cgn.gub.uy/innovaportal/file/83018/1/material_concurso_r14_cgn_2017.pdf