

**Universidad Americana UAM**

**Facultad de Ingeniería y Arquitectura / Ing. en Sistemas de la Información**



***Introducción a la Programación***

***Gestión de órdenes para el restaurante de comida rápida Pollos “El buen Sabor”***

**Integrantes:**

- ❖ Antonio Esteban Vargas Ramírez.
- ❖ Jhesly Lisset Castillo Martines.
- ❖ Joshua Romeo López Chávez.
- ❖ Patricia del Carmen Oquist Talavera.
- ❖ Rodolfo Alfredo Ramírez Collado.

**Docente:**

- ❖ Lic. Silvia Gigdalia Ticay López

08 de Julio, 2025

## Índice:

<b>1. Introducción.....</b>	<b>1</b>
<b>2. Definición y alcance del caso de estudio.....</b>	<b>3</b>
<b>3. Descripción del problema/necesidad o caso de estudio.....</b>	<b>4</b>
<b>4. Análisis del problema.....</b>	<b>4</b>
4.1. Lista de requerimientos.....	5
4.1.1. Requerimientos funcionales.....	5
4.1.2. Requerimientos no funcionales.....	6
4.2. Diagrama de Estructura.....	7
<b>5. Diseño del Algoritmo.....</b>	<b>8</b>
5.1. Imágenes del Pseudocódigo (PSeInt).....	9
5.2. Ejemplo práctico de ejecución del pseudocódigo.....	12
<b>6. Codificación, ejecución, verificación y depuración.....</b>	<b>13</b>
6.1. Imágenes del código Python.....	14
6.2. Ejemplo práctico de ejecución del código en VS Code.....	19
<b>7. Documentación del proyecto.....</b>	<b>20</b>
<b>8. Conclusiones.....</b>	<b>21</b>
<b>9. Recomendaciones.....</b>	<b>22</b>
<b>10. Bibliografía.....</b>	<b>23</b>
<b>11. Anexos.....</b>	<b>24</b>

## **1. Introducción**

La asignatura Introducción a la Programación es un componente esencial en la formación profesional dentro de la carrera de Ingeniería, especialmente en desarrollo de software. Su objetivo es proporcionar a los estudiantes una base sólida en conocimientos técnicos y tecnológicos, contribuyendo a generar ventajas competitivas y bienestar social mediante la innovación. Además, impulsa el desarrollo del pensamiento crítico y analítico, preparando a los estudiantes para enfrentar desafíos dinámicos en el entorno laboral actual.

Antes de abordar la solución propuesta, es importante destacar las herramientas de programación utilizadas a lo largo del desarrollo del proyecto. Durante el curso se emplearon entornos accesibles y didácticos que permitieron a los estudiantes comprender la lógica de la programación de manera progresiva. Herramientas como los entornos de pseudocódigo y los lenguajes de alto nivel jugaron un papel clave en el proceso de aprendizaje, facilitando la transición desde la teoría hacia aplicaciones prácticas concretas.

La combinación del pseudocódigo (PSeInt) con la programación estructurada (Python) es efectiva para fortalecer habilidades técnicas desde etapas tempranas del aprendizaje. De este modo, los estudiantes conectan conceptos teóricos con ejercicios prácticos, desarrollando habilidades de lógica, resolución eficiente de problemas y manejo de estructuras de datos. Esta integración permite abordar desafíos complejos sistemáticamente, fortaleciendo su confianza, creatividad y motivación hacia el aprendizaje continuo.

El proyecto desarrollado consiste en un sistema de gestión de pedidos para un restaurante de comida rápida de nombre “El Buen Sabor”, cuyo propósito es automatizar la toma de pedidos y generar facturas de forma ordenada y eficiente. Este sistema permite al usuario seleccionar productos desde un menú, visualizar precios, calcular el tiempo estimado de entrega, editar o eliminar pedidos y generar un comprobante de compra. La

problemática que se busca resolver está relacionada con la gestión manual de órdenes, que en pequeños negocios puede generar errores, demoras y desorganización, afectando la experiencia del cliente y la productividad del negocio.

Este software originalmente está pensado para el restaurante antes mencionado, pero puede ser adaptado por pequeños emprendedores o negocios locales que no cuentan con sistemas avanzados de gestión, ofreciéndoles una alternativa funcional, accesible y fácil de implementar. Además de automatizar tareas básicas, el sistema también fomenta una experiencia de usuario clara y ordenada, facilitando el control de las ventas diarias. Como estudiantes, este tipo de proyecto nos permite comprender cómo los conocimientos adquiridos pueden aplicarse directamente para resolver problemas reales, reforzando tanto nuestras capacidades técnicas como nuestra visión crítica y práctica del desarrollo de software.

Este programa de gestión de órdenes es una solución a uno de los problemas más presentes en los negocios pequeños, la automatización de procesos repetitivos que entorpecen el funcionamiento del negocio como tal. El software propuesto tiene como objetivo principal el potenciar la eficiencia de los negocios pequeños de comida rápida con la finalidad de mejorar el servicio al cliente, disminuir la presión sobre los trabajadores y maximizar las ganancias ofreciendo una solución tecnológica.

## **2. Definición y alcance del caso de estudio**

La práctica desarrollada se basa en la identificación y resolución de una problemática concreta observada en el entorno local: la gestión ineficiente de pedidos en pequeños negocios de comida rápida. El sistema diseñado para “El Buen Sabor” responde a esta necesidad mediante una aplicación de consola escrita en Python, que utiliza acceso a ficheros para mantener la persistencia de datos. Esta solución busca reemplazar procesos manuales propensos a errores por un flujo automatizado, claro y organizado.

El ámbito del problema se enmarca dentro del contexto local y comunitario, considerando que muchos negocios pequeños enfrentan limitaciones tecnológicas que afectan su eficiencia operativa y calidad de atención al cliente. Aunque el proyecto está dirigido específicamente a un restaurante ficticio, puede ser escalado o adaptado fácilmente a otros pequeños comercios de características similares dentro del mismo sector económico.

El alcance de esta solución se limita a la gestión básica de órdenes, incluyendo funciones como: visualización del menú, selección de productos, edición y eliminación de pedidos, cálculo de tiempo estimado de entrega y generación de facturas. No contempla aspectos como integración con plataformas digitales, pagos electrónicos o sistemas en red. A pesar de estas limitaciones, se presenta como un prototipo funcional que permite demostrar de manera efectiva cómo los conocimientos adquiridos pueden traducirse en herramientas útiles y aplicables en la vida real. La tecnología utilizada, al ser accesible y versátil, no representa una restricción sino una oportunidad de implementación práctica y didáctica.

### **3. Descripción del problema/necesidad o caso de estudio**

El equipo identificó una problemática frecuente en pequeños negocios de comida rápida: la toma y gestión manual de pedidos. Esta situación suele generar errores en el registro, confusión con los productos, desorganización en la atención al cliente y pérdida de tiempo tanto para el personal como para los clientes. Para abordar este problema, se planteó desarrollar un sistema informático sencillo que automatizara estas funciones básicas. La necesidad surgió de la observación directa de cómo funcionan estos negocios, tanto de forma interna como externa, así como de la falta de optimización durante el proceso de recibir la orden, presentársela a los cocineros y finalizar con el llevar dicho platillo a los clientes.

A partir de esta necesidad, se construyó un caso de estudio centrado en el restaurante ficticio “El Buen Sabor”, que representa un modelo común de negocio sin recursos tecnológicos para gestión operativa. El objetivo principal fue crear un software que permitiera registrar los pedidos, calcular el total a pagar, estimar el tiempo de entrega y generar un comprobante de compra. Para sustentar la existencia del problema, se utilizaron técnicas de observación directa (como el registro de los tiempos de atención y los errores comunes). La recopilación de datos evidenció que una solución automatizada no solo mejoraría la eficiencia, sino también la percepción del servicio ofrecido.

### **4. Análisis del problema**

El análisis del problema reveló que el sistema debía aceptar entradas como selección de productos desde un menú (por número), modificación y eliminación de órdenes, junto a la confirmación final del pedido. Las salidas requeridas eran: listado de productos en la orden, precio total, tiempo estimado de entrega y la generación de un recibo que facilite la lectura de las órdenes del cliente hacia el cocinero (de este modo solo tiene que leer el nombre del platillo y bebidas junto al número de orden).

Se identificaron cuatro subproblemas principales:

1. Mostrar menú al usuario
2. Agregar productos a la orden
3. Editar y eliminar productos de la orden
4. Generar factura con total y tiempo de entrega

Cada subproblema fue analizado por separado, detallando entradas (número de producto, texto, etc.), salidas esperadas (texto con formato), y restricciones (valores inválidos, estructuras de control). Este análisis permitió construir una solución dividida en módulos, lo que facilitó su implementación.

#### **4.1. Lista de requerimientos**

##### **4.1.1. Requerimientos funcionales**

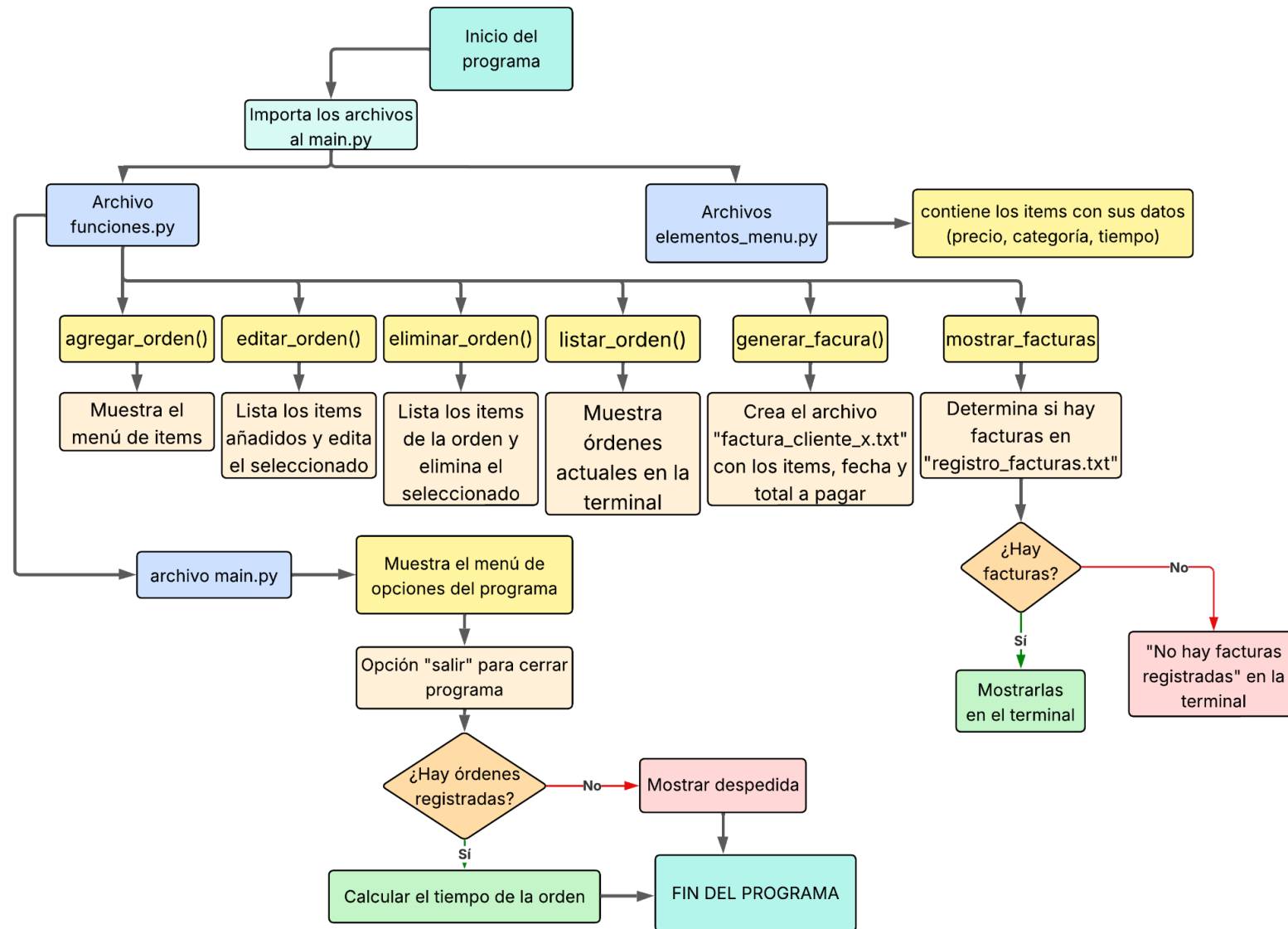
- El sistema debe mostrar un menú con los productos disponibles, incluyendo precio, categoría y tiempo de preparación.
- El usuario debe poder seleccionar uno o varios productos para agregarlos a una orden.
- El sistema debe permitir eliminar productos previamente seleccionados.
- El sistema debe mostrar un listado de los productos actuales en la orden.
- El sistema debe calcular automáticamente el total a pagar.
- El sistema debe calcular el tiempo total estimado de preparación de la orden.
- El sistema debe generar una factura en un archivo de texto con los productos pedidos, su precio y el total final.
- El sistema debe guardar los datos de las órdenes generadas.
- El sistema debe permitir cerrar la aplicación de forma ordenada, mostrando un mensaje de despedida con la información del pedido si existe.

#### **4.1.2. Requerimientos no funcionales**

- Los datos deben almacenarse de forma persistente en archivos de texto (.txt) accesibles.
- La estructura de los archivos generados (factura, registro) debe ser clara y legible para cualquier usuario.
- La interfaz debe ser completamente textual para facilitar su uso en consola.
- El sistema debe ejecutarse correctamente en cualquier dispositivo que tenga Python instalado.
- El código debe estar comentado de forma adecuada para facilitar su comprensión y mantenimiento.
- El tiempo de respuesta del sistema ante una acción del usuario debe ser inmediato (interacción fluida).
- El sistema debe utilizar nombres descriptivos para los archivos y funciones.



## 4.2. Diagrama de Estructura



## 5. Diseño del Algoritmo

El diseño inicial del sistema fue realizado en pseudocódigo utilizando la herramienta PSeInt (Pseudo Intérprete). Esta aplicación educativa fue desarrollada con el propósito de ayudar a estudiantes de programación a comprender la lógica de algoritmos sin necesidad de dominar un lenguaje de programación desde el inicio. PSeInt permite escribir instrucciones en pseudocódigo utilizando un lenguaje estructurado similar al español, lo que facilita el enfoque lógico de los problemas antes de implementarlos en un lenguaje formal como Python (Novara, 2023).

PSeInt no solo permite representar algoritmos de forma textual, sino también mediante diagramas de flujo, permitiendo visualizar la secuencia de operaciones y el flujo de decisiones dentro del programa. Entre sus principales aplicaciones se encuentran: el diseño de algoritmos paso a paso, el análisis previo a la codificación real, la práctica de estructuras de control (como bucles y condicionales), y la validación de ideas antes de su implementación en lenguajes compilados o interpretados (Novara, 2023).

En este proyecto, el uso de PSeInt resultó útil para estructurar el sistema de gestión de pedidos de forma modular, descomponiendo el problema en partes más simples y organizadas. Cada “caso” dentro de una estructura de control según sea representa una función encargada de resolver un aspecto específico del sistema. Por ejemplo, se diseñó un caso para el ingreso de productos, otro para la edición de órdenes, uno para el cálculo del total, y otro para la generación de la factura.

Durante el diseño se utilizaron estructuras fundamentales como:

- **Condicionales** (si, sino, según) para controlar las decisiones del usuario.
- **Bucles** (mientras, repetir) para permitir acciones repetitivas como la selección de productos.
- **Vectores o arreglos** para almacenar la información de los pedidos realizados.

El pseudocódigo escrito permitió tener una visión general y organizada del funcionamiento del sistema. Esto facilitó la transición a la codificación en Python, ya que gran parte de la lógica ya estaba probada y estructurada, reduciendo errores y mejorando la eficiencia del proceso de desarrollo.

### 5.1. Imágenes del Pseudocódigo (PSeInt)

**Imagen 01:** Definición de arreglos y variables auxiliares utilizadas en el pseudocódigo.

```
1 Algoritmo menu_pollos_el_buen_sabor
2     // definicion de los arreglos
3     Dimension descripcion[15]
4     Dimension precio[15]
5     Dimension tiempo[15]
6     Dimension categoria[15]
7
8     Dimension orden_descripcion[100]
9     Dimension orden_precio[100]
10    Dimension orden_tiempo[100]
11    Dimension orden_categoria[100]
12
13    // variables auxiliares que permiten en desplazarse por las listas
14    Definir orden_contador, i, eleccion, indice, nueva_eleccion Como Entero
15    Definir opcion Como Cadena
16
```

**Imágen 02:** Ejemplo de definición de ítems del restaurante con sus respectivas categorías.

```
17 // Elementos del menu de pollos
18 descripcion[1]<"pollo asado"
19 precio[1]<200
20 tiempo[1]<20
21 categoria[1]<"PTF"
22
23 descripcion[2]<"pollo frito"
24 precio[2]<150
25 tiempo[2]<15
26 categoria[2]<"PTF"
27
28 descripcion[3]<"pollo rostisado"
29 precio[3]<180
30 tiempo[3]<30
31 categoria[3]<"PTF"
32
```

**Imágen 03:** Menú de opciones que le indica al usuario las acciones que puede realizar.

```
95 // Bucle principal de comandos para determinar la accion del usuario
96 Repetir
97     Escribir ""
98     Escribir "Gestión de Órdenes"
99     Escribir "1. Agregar Orden"
100    Escribir "2. Editar Orden"
101    Escribir "3. Eliminar Orden"
102    Escribir "4. Listar Órdenes"
103    Escribir "5. Salir"
104    Escribir "6. Imprimir factura"
105    Escribir "Seleccione una opción:"
106    Leer opcion
107
```

**Imágen 04:** Bucle “segun sea” para determinar la función que se debe ejecutar en dependencia del número que ingrese el usuario desde el teclado. Adicionalmente se presenta la función “agregar\_orden()” asignada al caso “1”.

```

111      Segun opcion
112
113      // agregar_orden()
114      Caso "1":
115
116          Para i<1 Hasta 15
117              Escribir i, ". ", descripcion[i], " - ", precio[i], " Córdoba$ "
118          FinPara
119
120          Escribir "Ingrese número de producto:"
121          Leer eleccion
122
123          Si eleccion>1 Y eleccion<15 Entonces
124              orden_contador<orden_contador+1
125              orden_descripcion[orden_contador] < descripcion[eleccion]
126              orden_precio[orden_contador] < precio[eleccion]
127              orden_tiempo[orden_contador] < tiempo[eleccion]
128              orden_categoria[orden_contador] < categoria[eleccion]
129              Escribir descripcion[eleccion], " agregado."
130          Sino
131              Escribir "Selección inválida."
132          FinSi
133
134      FinSegun
135

```

**Imágen 05:** Función para imprimir la factura del cliente tras realizar su orden junto al fin de la ejecución del bucle principal “repetir” si se marcan las opciones “5” o “6” .

```

227      //imprimir_factura()
228      Caso "6":
229
230          Escribir "FACTURA DE LA ORDEN:"
231          Escribir "-----"
232
233          suma < 0
234          tiempo_total < 0
235
236          Si orden_contador = 0 Entonces
237              Escribir "No hay productos en la orden."
238          Sino
239              Para i < 1 Hasta orden_contador Con Paso 1
240                  Escribir "Producto ", i, ": ", orden_descripcion[i]
241                  Escribir "Precio: C$ ", orden_precio[i]
242                  Escribir "Tiempo estimado: ", orden_tiempo[i], " minutos"
243                  Escribir "Categoría: ", orden_categoria[i]
244                  Escribir ""
245                  suma < suma + orden_precio[i]
246                  tiempo_total < tiempo_total + orden_tiempo[i]
247              FinPara
248
249              Escribir "-----"
250              Escribir "TOTAL A PAGAR: C$ ", suma
251              Escribir "TIEMPO TOTAL ESTIMADO: ", tiempo_total, " minutos"
252              Escribir "Gracias por su compra. ¡Vuelva pronto!"
253          FinSi
254      FinSegun
255
256      Hasta Que opcion="5" o opcion = "6"
257

```

## 5.2. Ejemplo práctico de ejecución del pseudocódigo

```
*** Ejecución Iniciada. ***

Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Imprimir factura
Seleccione una opción:
> 1
1. pollo asado - 200 Córdoba
2. pollo frito - 150 Córdoba
3. pollo rostizado - 180 Córdoba
4. pollo a la plancha - 120 Córdoba
5. alitas de pollo - 220 Córdoba
6. limonada - 30 Córdoba
7. te frio - 25 Córdoba
8. cerveza - 60 Córdoba
9. jamaica - 40 Córdoba
10. horchata - 50 Córdoba
11. pay de manzana - 115 Córdoba
12. pay de limon - 100 Córdoba
13. pastel de chocolate - 90 Córdoba
14. flan casero - 120 Córdoba
15. helado de coco - 80 Córdoba
Ingrese número de producto:
> 1
pollo asado agregado.

Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Imprimir factura
Seleccione una opción:
> 6
FACTURA DE LA ORDEN:
-----
Producto 1: pollo asado
Precio: C$ 200
Tiempo estimado: 20 minutos
Categoría: PTF
-----
TOTAL A PAGAR: C$ 200
TIEMPO TOTAL ESTIMADO: 20 minutos
Gracias por su compra. ¡Vuelva pronto!
*** Ejecución Finalizada. ***
```

## 6. Codificación, ejecución, verificación y depuración

Para la implementación del sistema se utilizó el lenguaje de programación Python, uno de los lenguajes más populares y accesibles tanto para estudiantes como para desarrolladores profesionales. Python se caracteriza por su sintaxis clara y legible, lo que lo convierte en una excelente opción para trasladar algoritmos diseñados previamente en pseudocódigo hacia una versión funcional. Su enfoque multi paradigma permite trabajar con estructuras modulares, condicionales, funciones y listas de manera sencilla y eficiente (Weisheim & Weisheim, 2025).

El entorno de desarrollo elegido fue Visual Studio Code (VS Code), un editor de código liviano y multiplataforma que ofrece soporte avanzado para Python mediante extensiones como Python by Microsoft. Este entorno permite trabajar con varios archivos organizados en carpetas, acceder a resaltado de sintaxis, autocompletado, depuración integrada y ejecución directa del código, lo que mejora significativamente el flujo de trabajo durante el desarrollo (*Visual Studio: IDE y Editor de Código Para Desarrolladores de Software y Teams*, 2025).

La estructura del proyecto se dividió en tres archivos principales:

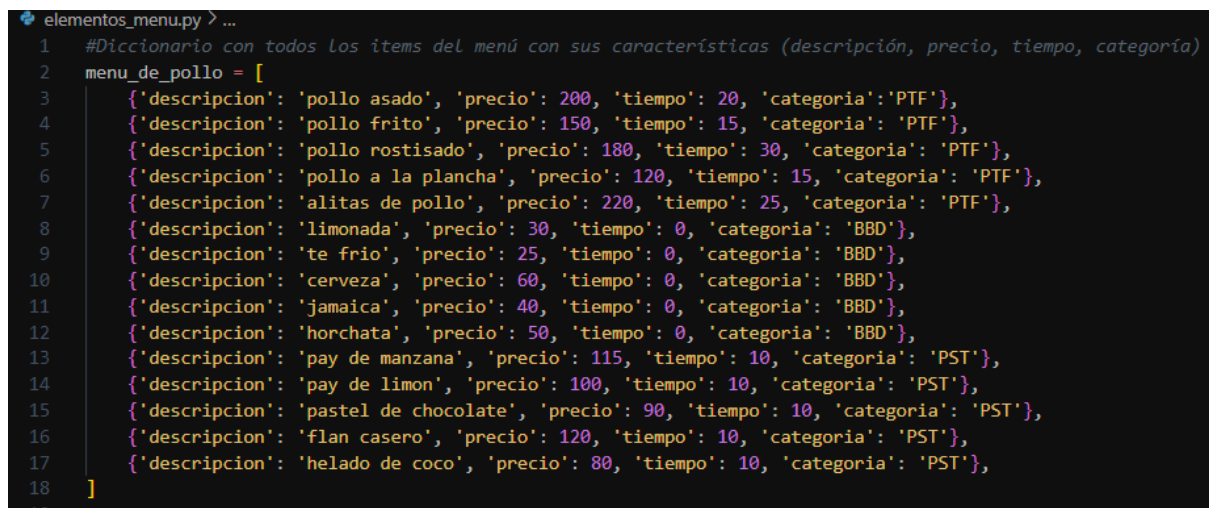
- Un archivo principal encargado de ejecutar el programa y gestionar el flujo general.
- Un archivo que contiene las funciones para manipular órdenes: agregar, editar, eliminar y generar facturas.
- Un archivo adicional con la lista de productos del menú, organizados por categorías: plato fuerte, postres y bebidas.

Gracias al diseño previo en pseudocódigo, el proceso de codificación fue fluido y sin errores graves. Las pruebas iniciales permitieron identificar detalles menores como errores de entrada del usuario, cálculos incompletos o mejoras en el formato de impresión de la factura. Estas situaciones fueron resueltas a través de un proceso de verificación constante y depuración utilizando la consola de salida y herramientas básicas de VS Code como los breakpoints y el panel de errores.

En conjunto, Python y Visual Studio Code ofrecieron un entorno robusto, accesible y eficiente para llevar a cabo este proyecto, permitiendo una implementación modular, bien documentada y fácil de mantener.

### 6.1. Imágenes del código Python

**Imagen 01:** Módulo “elementos\_menu.py”. Alberga el diccionario con todos los ítems del menú del restaurante Pollos “El buen Sabor” con sus respectivas categorías.



```
1 #Diccionario con todos los items del menú con sus características (descripción, precio, tiempo, categoría)
2 menu_de_pollo = [
3     {'descripcion': 'pollo asado', 'precio': 200, 'tiempo': 20, 'categoria': 'PTF'},
4     {'descripcion': 'pollo frito', 'precio': 150, 'tiempo': 15, 'categoria': 'PTF'},
5     {'descripcion': 'pollo rostizado', 'precio': 180, 'tiempo': 30, 'categoria': 'PTF'},
6     {'descripcion': 'pollo a la plancha', 'precio': 120, 'tiempo': 15, 'categoria': 'PTF'},
7     {'descripcion': 'alitas de pollo', 'precio': 220, 'tiempo': 25, 'categoria': 'PTF'},
8     {'descripcion': 'limonada', 'precio': 30, 'tiempo': 0, 'categoria': 'BBD'},
9     {'descripcion': 'te frio', 'precio': 25, 'tiempo': 0, 'categoria': 'BBD'},
10    {'descripcion': 'cerveza', 'precio': 60, 'tiempo': 0, 'categoria': 'BBD'},
11    {'descripcion': 'jamaica', 'precio': 40, 'tiempo': 0, 'categoria': 'BBD'},
12    {'descripcion': 'horchata', 'precio': 50, 'tiempo': 0, 'categoria': 'BBD'},
13    {'descripcion': 'pay de manzana', 'precio': 115, 'tiempo': 10, 'categoria': 'PST'},
14    {'descripcion': 'pay de limon', 'precio': 100, 'tiempo': 10, 'categoria': 'PST'},
15    {'descripcion': 'pastel de chocolate', 'precio': 90, 'tiempo': 10, 'categoria': 'PST'},
16    {'descripcion': 'flan casero', 'precio': 120, 'tiempo': 10, 'categoria': 'PST'},
17    {'descripcion': 'helado de coco', 'precio': 80, 'tiempo': 10, 'categoria': 'PST'},
18 ]
```



**Imágen 02:** Módulo “funciones.py”. Se importan las funciones como “datetime” para determinar la fecha y hora de la creación de las facturas de cada cliente junto al módulo “elementos\_menu” para importar el diccionario con todos los ítems y sus categorías. Además se crean los arreglos “ordenes”, “facturas” y se establece el n° del cliente en 1.

```
funciones.py > ...
1  import datetime
2  from elementos_menu import menu_de_pollo
3
4  ordenes = []
5  facturas = [] # Registro de facturas
6  numero_cliente = 1 # Número de cliente único
7
8  #Función para mostrar el menú con todos los items y características albergadas en el diccionario
9  > def mostrar_menu():...
13
14  #Función para agregar orden del menú anteriormente mostrado
15  > def agregar_orden():...
26
27  #Función para reemplazar un item ya añadido al arreglo "ordenes"
28  > def editar_orden():...
44
45  #Función para eliminar un item dentro del arreglo "ordenes"
46  > def eliminar_orden():...
57
58  #Función para listar los items dentro del arreglo "ordenes"
59  > def listar_orden():...
66
67  #Generador de facturas para los clientes que han completado su orden
68  > def generar_factura():...
98
99  #Función para registrar las facturas en un archivo .txt con la fecha en la que
100 #fue generada, el monto total y el número de cliente
101 > def mostrar_facturas():...
108
```

**Imágen 03:** Función “mostrar\_menu()” para mostrar todos los ítems en la terminal.

```
8  #Función para mostrar el menú con todos los items y características albergadas en el diccionario
9  def mostrar_menu():
10     print("\nMenú de Pollo:")
11     for i, item in enumerate(menu_de_pollo, 1):
12         print(f"{i}. {item['descripcion']} - {item['precio']} Córdoba")
```

**Imágen 04:** Función “agregar\_orden()” para añadir los ítems al arreglo “ordenes”.

```
14  #Función para agregar orden del menú anteriormente mostrado
15  def agregar_orden():
16     mostrar_menu()
17     try:
18         eleccion = int(input("Ingrese el número del producto que desea pedir: ")) - 1
19         if 0 <= eleccion < len(menu_de_pollo):
20             ordenes.append(menu_de_pollo[eleccion])
21             print(f"\n{menu_de_pollo[eleccion]['descripcion']} agregado a la orden.")
22         else:
23             print("\nSelección inválida.")
24     except ValueError:
25         print("\nPor favor, ingrese un número válido.")
```

**Imágen 05:** Función “editar\_orden()” para cambiar las órdenes del usuario en el arreglo.

```
27 #Función para reemplazar un ítem ya añadido al arreglo "ordenes"
28 def editar_orden():
29     listar_orden()
30     try:
31         index = int(input("Ingrese el número de la orden que desea modificar: ")) - 1
32         if 0 <= index < len(ordenes):
33             mostrar_menu()
34             nueva_eleccion = int(input("Ingrese el nuevo producto: ")) - 1
35             if 0 <= nueva_eleccion < len(menu_de_pollo):
36                 ordenes[index] = menu_de_pollo[nueva_eleccion]
37                 print("\nOrden actualizada correctamente.")
38             else:
39                 print("\nSelección inválida.")
40         else:
41             print("\nNúmero de orden inválido.")
42     except ValueError:
43         print("\nIngrese un número válido.")
44
```

**Imágen 06:** Función “eliminar\_orden()” para descartar un elemento elegido por el usuario.

```
45 #Función para eliminar un ítem dentro del arreglo "ordenes"
46 def eliminar_orden():
47     listar_orden()
48     try:
49         index = int(input("Ingrese el número de la orden que desea eliminar: ")) - 1
50         if 0 <= index < len(ordenes):
51             eliminado = ordenes.pop(index)
52             print(f"\n{eliminado['descripcion']} ha sido eliminado de la orden.")
53         else:
54             print("\nNúmero de orden inválido.")
55     except ValueError:
56         print("\nIngrese un número válido.")
```

**Imágen 07:** Función “listar\_orden()” para mostrar los elementos del arreglo que ha añadido el usuario previamente junto su precio en córdobas.

```
58 #Función para listar los ítems dentro del arreglo "ordenes"
59 def listar_orden():
60     print("\nÓrdenes actuales:")
61     if not ordenes:
62         print("No hay órdenes registradas.")
63     else:
64         for i, item in enumerate(ordenes, 1):
65             print(f"{i}. {item['descripcion']} - {item['precio']} Córdoba")
66
```

**Imágen 08:** Función “mostrar\_facturas()” para indicar todas las órdenes realizadas en el día diferenciadas por el número del cliente, la fecha en la que fue generada y el total a pagar en córdobas. También guarda cada factura en “registro\_facturas.txt” en caso de reiniciar el programa dentro de la terminal al usar la función “salir” del menú.

```
99 #Función para registrar las facturas en un archivo .txt con la fecha en la que
100 #fue generada, el monto total y el número de cliente
101 def mostrar_facturas():
102     print("\nRegistro de Facturas:")
103     if not facturas:
104         print("No hay facturas registradas.")
105     else:
106         with open("registro_facturas.txt", "a", encoding = "utf-8") as archivo:
107             for factura in facturas:
108                 print(f"Cliente Nº: {factura['numero_cliente']} - Fecha: {factura['fecha']} - Total: {factura['total']} Córdobas")
109                 archivo.write(f"Cliente Nº: {factura['numero_cliente']} - Fecha: {factura['fecha']} - Total: {factura['total']} Córdobas")
```

**Imágen 09:** Función “generar\_factura()” para crear un archivo “factura\_Cliente\_X.txt” que muestre los datos como el número de cliente, la fecha en la que fue generada, los elementos del arreglo “ordenes” y el total a pagar. El contador del cliente va sumando 1 por cada factura generada.

```
67 #Generador de facturas para los clientes que han completado su orden
68 def generar_factura():
69     global numero_cliente
70
71     if not ordenes:
72         print("\nNo hay órdenes registradas.")
73         return
74
75     fecha_actual = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
76     nombre_archivo = f"factura_cliente_{numero_cliente}.txt"
77
78     with open(nombre_archivo, "w", encoding="utf-8") as archivo:
79         archivo.write("----- FACTURA ----- \n")
80         archivo.write(f"Fecha: {fecha_actual} \n")
81         archivo.write(f"Cliente Nº: {numero_cliente} \n \n")
82
83         total = 0
84         for i, item in enumerate(ordenes, 1):
85             archivo.write(f"{i}. {item['descripcion']} - {item['precio']} Córdobas \n")
86             total += item['precio']
87         archivo.write(f"\nTotal a pagar: {total} Córdobas \n")
88         archivo.write("----- \n")
89
90     facturas.append({
91         'numero_cliente': numero_cliente,
92         'fecha': fecha_actual,
93         'total': total
94     })
95     numero_cliente += 1
96
97     print(f"\nFactura generada exitosamente en el archivo: {nombre_archivo}")
98
```

**Imagen 10:** Archivo “main.py” donde se importan todas las funciones del módulo

“funciones.py”. Acá se muestra el bucle principal que contiene el menú de opciones, el cual se ejecutará hasta que se marque la opción “salir” al escribir el número 5. Cada función es utilizada en dependencia de la opción marcada por el usuario. Si se marca “salir” el programa guarda un registro de todas las facturas generadas en el archivo “registro\_facturas.py” para llevar un seguimiento de todos los ingresos del restaurante en según qué fechas o horas del día, lo que permitirá mejorar el servicio a largo plazo.

```
main.py > ...
1  from funciones import agregar_orden, editar_orden, eliminar_orden, listar_orden, ordenes, generar_factura, mostrar_facturas
2
3  #Función principal con el menú de opciones que verá el cliente al momento de gestionar su orden
4  def main():
5      while True:
6          print("\nGestión de Órdenes")
7          print("1. Agregar Orden")
8          print("2. Editar Orden")
9          print("3. Eliminar Orden")
10         print("4. Listar Órdenes")
11         print("5. Salir")
12         print("6. Generar Factura")
13         print("7. Mostrar Registro de Facturas")
14
15         opcion = input("Seleccione una opción: ")
16
17         if opcion == "1":
18             agregar_orden()
19         elif opcion == "2":
20             editar_orden()
21         elif opcion == "3":
22             eliminar_orden()
23         elif opcion == "4":
24             listar_orden()
25         elif opcion == "5":
26             if ordenes:
27                 tiempo_total = sum(item['tiempo'] for item in ordenes)
28                 print(f"\nGracias por su pedido. Su comida estará lista en aproximadamente {tiempo_total} minutos.")
29             else:
30                 print("\nNo hay órdenes registradas. ¡Hasta pronto!")
31             break
32         elif opcion == "6":
33             generar_factura()
34         elif opcion == "7":
35             mostrar_facturas()
36         else:
37             print("\nOpción inválida. Intente de nuevo.")
38
39 if __name__ == "__main__":
40     main()
```

## 6.2. Ejemplo práctico de ejecución del código en VS Code

(En caso de imprimir la factura y mostrar el registro de facturas en la terminal)

```
Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Generar Factura
7. Mostrar Registro de Facturas
Seleccione una opción: 1

Menú de Pollo:
1. pollo asado - 200 Córdobas
2. pollo frito - 150 Córdobas
3. pollo rostisado - 180 Córdobas
4. pollo a la plancha - 120 Córdobas
5. alitas de pollo - 220 Córdobas
6. limonada - 30 Córdobas
7. te frio - 25 Córdobas
8. cerveza - 60 Córdobas
9. jamaica - 40 Córdobas
10. horchata - 50 Córdobas
11. pay de manzana - 115 Córdobas
12. pay de limon - 100 Córdobas
13. pastel de chocolate - 90 Córdobas
14. flan casero - 120 Córdobas
15. helado de coco - 80 Córdobas
Ingrese el número del producto que desea pedir: 1

pollo asado agregado a la orden.
```

```
Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Generar Factura
7. Mostrar Registro de Facturas
Seleccione una opción: 6

Factura generada exitosamente en el archivo: factura_cliente_1.txt

Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Generar Factura
7. Mostrar Registro de Facturas
Seleccione una opción: 7

Registro de Facturas:
Cliente Nº: 1 - Fecha: 2025-07-03 23:47:18 - Total: 200 Córdobas

Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Generar Factura
7. Mostrar Registro de Facturas
Seleccione una opción: 5

Gracias por su pedido. Su comida estará lista en aproximadamente 20 minutos.
```

```
factura_cliente_1.txt
1  ----- FACTURA -----
2  Fecha: 2025-07-03 23:47:18
3  Cliente Nº: 1
4
5  1. pollo asado - 200 Córdobas
6
7  Total a pagar: 200 Córdobas
8  -----
```

```
registro_facturas.txt
1  Cliente Nº: 1 - Fecha: 2025-07-03 23:47:18 - Total: 200 Córdobas
2
```

(En caso de editar y eliminar la orden)

```
Órdenes actuales:
1. pollo asado - 200 Córdobas
Ingrese el número de la orden que desea modificar: 1

Menú de Pollo:
1. pollo asado - 200 Córdobas
2. pollo frito - 150 Córdobas
3. pollo rostisado - 180 Córdobas
4. pollo a la plancha - 120 Córdobas
5. alitas de pollo - 220 Córdobas
6. limonada - 30 Córdobas
7. te frio - 25 Córdobas
8. cerveza - 60 Córdobas
9. jamaica - 40 Córdobas
10. horchata - 50 Córdobas
11. pay de manzana - 115 Córdobas
12. pay de limon - 100 Córdobas
13. pastel de chocolate - 90 Córdobas
14. flan casero - 120 Córdobas
15. helado de coco - 80 Córdobas
Ingrese el nuevo producto: 2

Orden actualizada correctamente.
```

```
Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Generar Factura
7. Mostrar Registro de Facturas
Seleccione una opción: 3

Órdenes actuales:
1. pollo frito - 150 Córdobas
Ingrese el número de la orden que desea eliminar: 1

pollo frito ha sido eliminado de la orden.
```

(En caso de listar todos lo elementos que ya se han añadido)

```
Gestión de Órdenes
1. Agregar Orden
2. Editar Orden
3. Eliminar Orden
4. Listar Órdenes
5. Salir
6. Generar Factura
7. Mostrar Registro de Facturas
Seleccione una opción: 4

Órdenes actuales:
1. pollo frito - 150 Córdobas
2. pollo rostizado - 180 Córdobas
3. pollo a la plancha - 120 Córdobas
4. alitas de pollo - 220 Córdobas
```

## 7. Documentación del proyecto

La documentación interna se integró en el código fuente mediante comentarios explicativos para cada función. La documentación externa incluye el pseudocódigo, descripciones de cada módulo y el presente informe como recopilación final.

El proyecto fue entregado con una estructura clara, con todas las funciones alojadas en un archivo separado que luego se importa dentro del archivo principal. Esto facilita la comprensión del programa, su mantenimiento y futuras mejoras. También se anexaron ejemplos de ejecución y resultados esperados como respaldo del funcionamiento correcto del sistema.

## **8. Conclusiones**

El desarrollo del sistema de gestión de órdenes para el restaurante “El Buen Sabor” permitió aplicar de forma práctica los conocimientos adquiridos en la asignatura Introducción a la Programación. A través del uso de PSeInt para el diseño y Python para la implementación, se logró construir un sistema funcional, modular y adaptable a las necesidades del entorno.

El uso del pseudocódigo facilitó la organización lógica de las funciones antes de su implementación, y el empleo de Python y Visual Studio Code permitió traducir esa lógica en un sistema accesible y eficiente. Esto redujo errores y optimizó el tiempo de desarrollo.

El sistema automatiza la toma de pedidos, calcula totales y tiempos de preparación, y genera facturas en archivos de texto, contribuyendo a mejorar la gestión en pequeños negocios. Además, este proyecto fortaleció habilidades técnicas como el uso de estructuras de control, validaciones, modularidad, y manejo de archivos.



En síntesis, se demostró que la programación estructurada y el diseño algorítmico permiten resolver problemáticas reales con soluciones viables. Asimismo, se desarrollaron competencias esenciales como el trabajo en equipo, la planificación, la depuración y la documentación, fundamentales en la formación profesional en ingeniería de sistemas.

## **9. Recomendaciones**

- Se recomienda continuar empleando herramientas de diseño algorítmico como PSeInt en etapas tempranas del desarrollo, ya que permiten visualizar la lógica del programa y anticipar posibles errores antes de la codificación formal.
- Es recomendable fomentar una planificación previa al desarrollo que contemple la asignación de responsabilidades, cronograma de trabajo y seguimiento de avances, con el objetivo de mejorar la organización del equipo y optimizar los tiempos de ejecución del proyecto.
- Se sugiere mantener una estructura modular del código fuente, separando claramente las funcionalidades en distintos archivos, lo cual facilita el mantenimiento, la reutilización del código y su escalabilidad.
- La documentación interna mediante comentarios detallados y descriptivos debe ser promovida como una práctica estándar, ya que contribuye a la comprensión del sistema por parte de terceros y garantiza la continuidad del desarrollo.
- Se recomienda explorar otras tecnologías complementarias como interfaces gráficas o versiones web y móviles, para mejorar la experiencia del usuario y aumentar la accesibilidad del sistema en diversos entornos operativos.
- Finalmente, se recomienda que, en futuras versiones del sistema, se implemente una base de datos relacional como SQLite o MySQL para sustituir el uso de archivos de texto. Esto permitiría una gestión más eficiente y segura de la información, facilitando consultas, reportes y organización de datos en escenarios más complejos.



## 10. Bibliografía

 *El libro de Python* . (s. f.). El Libro de Python. <https://ellibrodepython.com/>

colaboradores de Wikipedia. (2025a, marzo 3). *PSEInt*. Wikipedia, la Enciclopedia Libre. <https://es.wikipedia.org/wiki/PSEInt>

colaboradores de Wikipedia. (2025b, junio 18). *Python*. Wikipedia, la Enciclopedia Libre. <https://es.wikipedia.org/wiki/Python>

colaboradores de Wikipedia. (2025c, junio 25). *Visual Studio Code*. Wikipedia, la Enciclopedia Libre. [https://es.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://es.wikipedia.org/wiki/Visual_Studio_Code)

*El tutorial de Python*. (s. f.). Python Documentation.  
<https://docs.python.org/es/3.13/tutorial/index.html>

Novara, P. (2023). *Características y funcionalidades - PSEInt*. SourceForge.  
Recuperado 3 de julio de 2025, de <https://pseint.sourceforge.net/features.php>

Soy Dalto. (2023, 22 enero). *Curso de PYTHON desde CERO (Completo)* [Video].  
YouTube. <https://www.youtube.com/watch?v=nKPbfIU442g>

*Visual Studio: IDE y Editor de código para desarrolladores de software y Teams*.  
(2025, 24 marzo). Visual Studio. <https://visualstudio.microsoft.com/es/>

Weisheim, R., & Weisheim, R. (2025, 19 febrero). *Qué es Python: conoce uno de los lenguajes de programación más populares*. ES Tutoriales.

<https://www.hostinger.com/es/tutoriales/que-es-python>

## 11. Anexos

### Anexo 01: El Libro De Python



### Anexo 02: Curso de Soy Dalto en YouTube // Python desde cero (completo)



### Anexo 03: Ficha de Observación

#### Ficha de Observación // I Semestre III CE

**Nombre del lugar observado:** Restaurante de comida rápida "El Buen Sabor"

**Dirección:** Km 3½ Carretera al Mercado Iván Montenegro, 50 metros al sur, Managua, Nicaragua

**Fecha de observación:** 25 de junio de 2025

**Hora de observación:** 11:30 a.m. – 1:00 p.m.

**Observadores:**

- Antonio Esteban Vargas Ramírez
- Jhesly Lisset Castillo Martines
- Joshua Romeo López Chávez
- Patricia del Carmen Oquist Talavera
- Rodolfo Alfredo Ramírez Collado

**Objetivo de la observación:**

Registrar de forma directa las principales fallas en el proceso de gestión de órdenes y atención al cliente, con el fin de sustentar el desarrollo de una solución automatizada.

**Aspectos Observados:**

Aspecto Evaluado	Descripción de lo observado
Tiempos de espera	Se registraron tiempos de espera de entre 7 a 15 minutos desde la orden hasta la entrega.
Registro de pedidos	El personal anotaba manualmente las órdenes en papel; se observaron errores en 2 pedidos.
Comunicación entre personal	Hubo confusión entre cocina y caja por pedidos mal interpretados (se cambió bebida por error).
Atención al cliente	La atención fue cordial pero demorada cuando el local se llenó.
Organización del flujo de pedidos	No existe un sistema visible de seguimiento de las órdenes; los pedidos se amontonaban.
Facturación	La cuenta era dada en papel a mano; el cálculo del total tomó más de 1 minuto en 3 ocasiones.
Repetición de errores	Un cliente recibió un plato incorrecto y tuvo que esperar nuevamente su comida.

**Herramientas de observación utilizadas:**

- Bitácora escrita (cuaderno de apuntes)
- Reloj digital con cronómetro (para medir tiempos de espera)
- Ficha de evaluación preestablecida (para comparar aspectos clave)
- Grabadora de audio en el celular (uso interno, sin registrar al personal directamente).

**Conclusión del equipo observador:**

Durante la visita se evidenciaron varias deficiencias en la gestión manual de pedidos, entre ellas: errores humanos frecuentes, demoras en la atención, y desorganización interna en el proceso de entrega. Esto valida la necesidad de un sistema digital como el que proponemos, que mejore la eficiencia, reduzca errores y optimice la experiencia del cliente.