Instructivo para la elaboración de un Manual Técnico

Autor (s):

Nombre:
Código:

Universidad de Guadalajara
Guadalajara, Jalisco. México
Mes, Año

Índice

1. Introducción

El Manual Técnico, como su nombre lo indica, contiene las especificaciones técnicas más importantes del sistema desarrollado. Constituye una guía especializada para la realización de las operaciones de mantenimiento de la aplicación. Se encuentra dirigido fundamentalmente a la dirección de Tecnologías de la Información, al administrador del sistema, a otros desarrolladores, así como al departamento de calidad y auditoría de sistemas.

2. Confección

Para la elaboración de un manual de técnico se deben de considerar los siguientes apartados normativos.

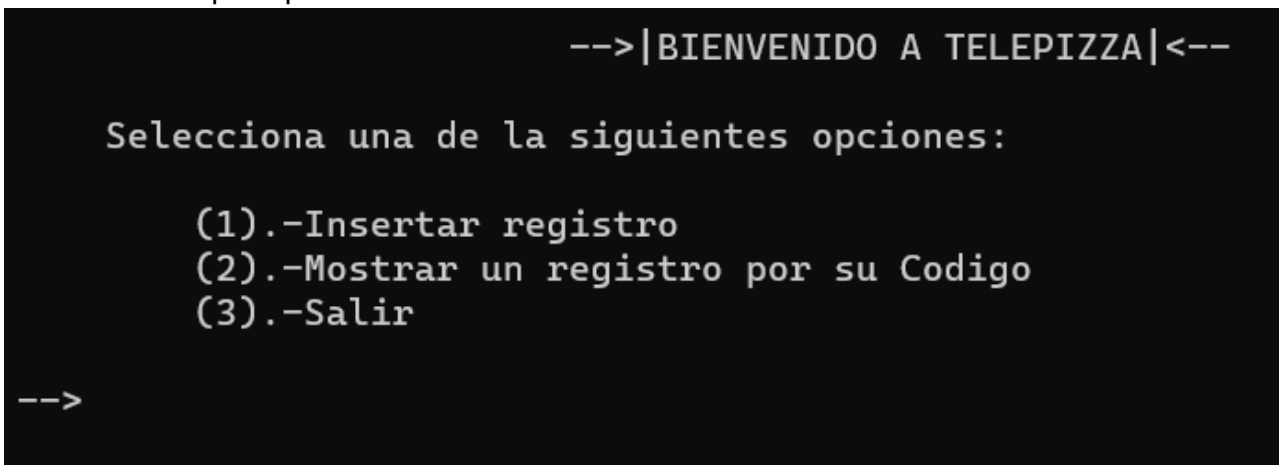
Nota: Los acápites que aparecen en color rojo y con descripción en letra cursiva son de carácter opcional.

2.1 Nombre del Sistema: Software de Gestión y manejo de productos.

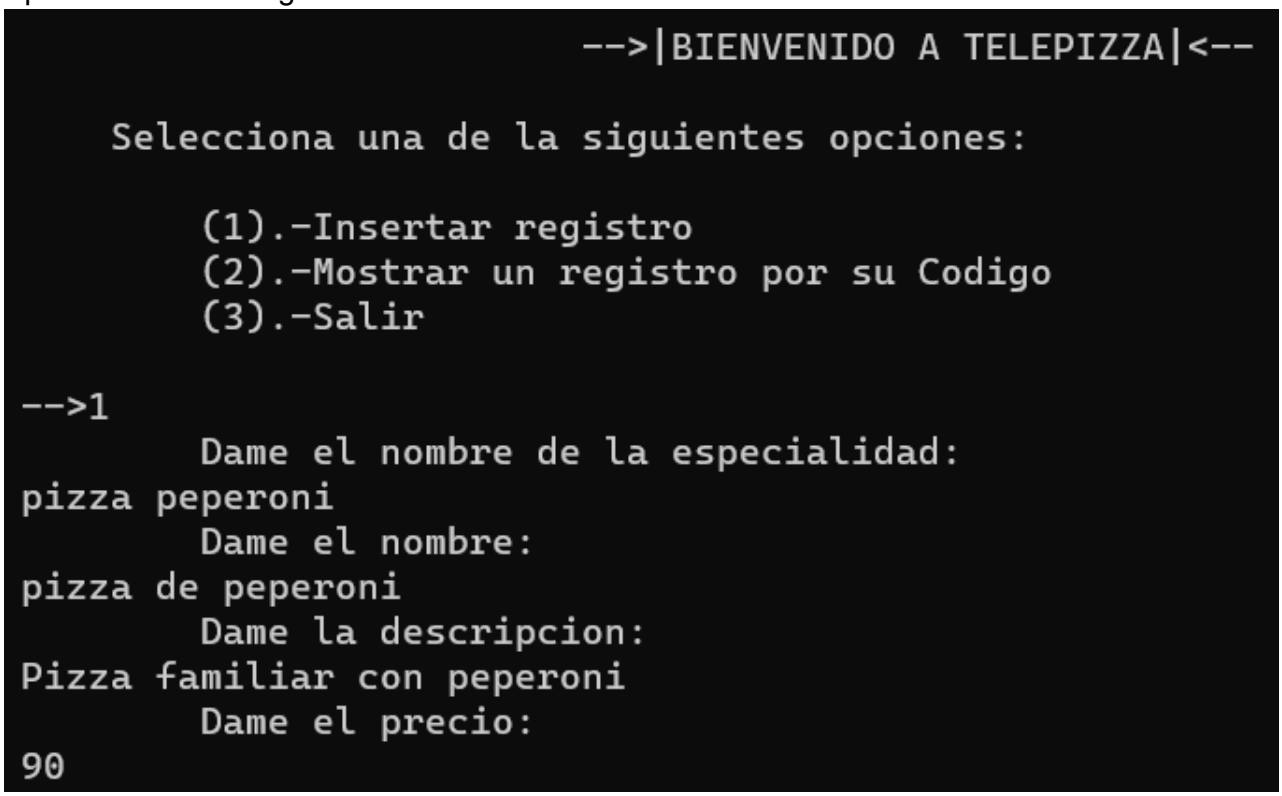
2.2 Versión del Sistema: 0100 versión inicial.

2.3 Tipo de Manual: Manual técnico.

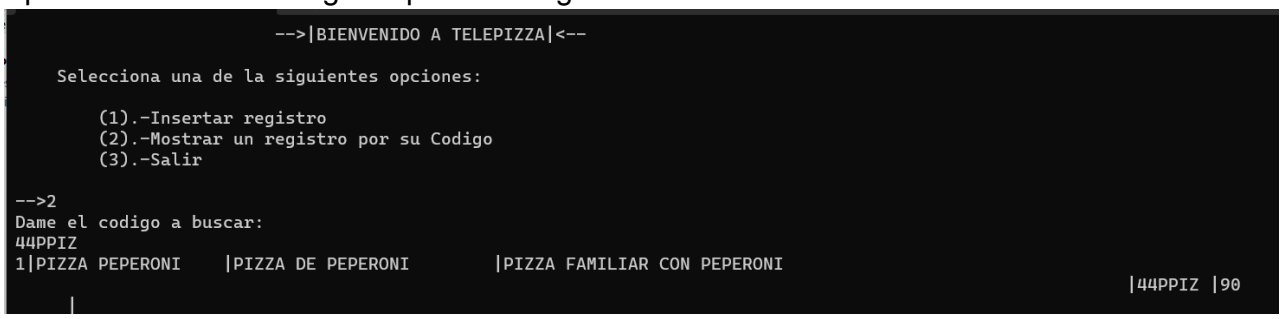
2.4 Poner una Imagen:
Vista del menú principal.



Opción 1: insertar registro.



Opcion 2: Mostrar un registro por su codigo.



2.5 Fecha de Elaboración: Resulta importante el incluir la fecha de elaboración, pues representa un punto de referencia y control.

14/04/2023

2.6 Área donde fue elaborado: Incluir el nombre del área en donde fue elaborado el manual.

CUCEI

2.7 Índice del Contenido del Manual:

Índice:

2.7.1. ***Introducción***

Este código implementa un sistema de menú para una pizzería utilizando C + +. El programa utiliza estructuras para almacenar los datos de los diferentes elementos del menú, y permite ingresar nuevos elementos, buscar elementos existentes y ordenarlos por índice.

El programa comienza con la definición de las librerías necesarias y las estructuras para almacenar los datos. La estructura Menú tiene los siguientes campos: índice (el índice del elemento), código (el código del elemento), especialidad (la especialidad del elemento), nombre (el nombre del elemento), descripción (la descripción del elemento) y precio (el precio del elemento).

La estructura st Índice código es utilizada para almacenar la llave primaria (el código del elemento) y el índice correspondiente.

La función Rellenar es una función auxiliar que se utiliza para rellenar con espacios los campos de texto para asegurarse de que todos tengan la misma longitud.

La función Menú::Ingresar se encarga de agregar un nuevo elemento al archivo MENU.txt. Primero abre el archivo en modo de escritura y lee el último índice existente para asignar uno nuevo. Luego genera un código para el nuevo elemento utilizando la forma canónica, que utiliza algunos caracteres de la especialidad y el nombre del elemento para generar un código único.

Finalmente, se solicita al usuario que ingrese los datos del nuevo elemento (especialidad, nombre, descripción y precio), se rellenan los campos con espacios utilizando la función Rellenar, y se escriben en el archivo MENU.txt

2.7.2. ***Objetivos Generales y Específicos del Sistema***

Breve descripción de los objetivos generales y específicos que se cumplieron con el desarrollo del sistema.

Objetivos generales:

- Presentar de manera clara y organizada las opciones de comida y bebida disponibles para los clientes.
- Ayudar a los clientes a elegir lo que desean comer o beber al proporcionar información detallada sobre los platos, ingredientes, precios, etc.
- Facilitar el trabajo de los empleados al proporcionar información clara y detallada sobre las opciones disponibles y los precios.

Objetivos específicos:

- Actualizar y cambiar el menú periódicamente para mantener la oferta fresca y atractiva para los clientes.
- Mejorar el sistema de menú anterior de la pizzeria.

2.7.3. ***Normas, Políticas y Procedimientos***

Normas:

- Todos los platos del menú deben cumplir con los estándares de calidad y sabor establecidos por la empresa.
- Los precios de los platos deben ser fijados de manera coherente y justa, y deben ser aprobados por la gerencia antes de ser incluidos en el menú.
- El menú debe ser actualizado regularmente para reflejar los cambios en los productos, los precios y las temporadas.
- Los platos del menú deben ser presentados y servidos siguiendo los estándares de higiene y seguridad alimentaria establecidos por las autoridades sanitarias locales.

Políticas:

- El menú debe incluir opciones vegetarianas y/o veganas para satisfacer las necesidades de los clientes con diferentes dietas.
- Los ingredientes de los platos del menú deben ser seleccionados de proveedores de confianza y calidad.
- Los platos del menú deben ser preparados y cocinados siguiendo las recetas y técnicas establecidas por la empresa.
- Los platos del menú no deben ser modificados o personalizados de manera significativa sin la aprobación del gerente o encargado de cocina.

Procedimientos:

- El proceso de creación y actualización del menú debe incluir la revisión y aprobación de la gerencia y del equipo de cocina.
- Los platos del menú deben ser probados y evaluados antes de ser incluidos en el menú.
- Los platos del menú deben ser presentados y servidos siguiendo los estándares de la empresa, incluyendo la estética, el sabor y la temperatura adecuados.

2.7.4. ***Definición de las Reglas del Negocio Implementadas***

Definir los lineamientos que se contemplaron durante el desarrollo de la aplicación. Descripción detallada de las reglas de negocio que debe seguir el sistema para garantizar las restricciones que existen en el negocio.

- Diseño intuitivo y fácil de usar: La aplicación debe ser fácil de usar y navegar, con un diseño atractivo e intuitivo que permita a los usuarios realizar pedidos rápidamente y sin confusiones.
- Integración de archivos: La aplicación debe estar integrada con archivos de la pizzería para mostrar los menús actualizados y los precios correctos.

2.7.5. ***Fundamentación de la Tecnología Utilizada***

Realizar una breve descripción, con sus correspondientes referencias bibliográficas, acerca de las tendencias y tecnologías actuales sobre las que se apoya la propuesta, además de incluir la justificación de las seleccionadas para el desarrollo de la aplicación.

- Eficiencia: C++ es un lenguaje de programación de bajo nivel que permite un mayor control sobre el hardware y la memoria, lo que lo hace muy eficiente. Es especialmente útil para aplicaciones que requieren un alto rendimiento, como los juegos, la animación, la simulación y el procesamiento de imágenes.

- Portabilidad: C++ es un lenguaje de programación multiplataforma que se puede utilizar en diferentes sistemas operativos y arquitecturas de procesadores. Esto permite que los programas escritos en C++ se ejecuten en diferentes plataformas sin necesidad de modificaciones adicionales.
- Orientación a objetos: C++ es un lenguaje de programación orientado a objetos que permite organizar el código de manera más estructurada y modular. Esto hace que sea más fácil de entender, mantener y actualizar el código.

En general, hacer un programa en C++ puede ser una buena opción si se busca eficiencia, portabilidad, modularidad y reutilización de código. Sin embargo, también es importante tener en cuenta que C++ es un lenguaje de programación más complejo que otros lenguajes de programación y puede requerir una mayor experiencia en programación para ser utilizado eficazmente.

Stroustrup, B. (2013). The C++ Programming Language. Addison-Wesley Professional.
Lippman, S., Lajoie, J., & Moo, B. (2012). C++ Primer (5th ed.). Addison-Wesley Professional.
Eckel, B. (2000). Thinking in C++. Prentice Hall.
Schildt, H. (2017). C++: The Complete Reference (5th ed.). McGraw-Hill Education.
Josuttis, N. M. (2012). The C++ Standard Library: A Tutorial and Reference (2nd ed.). Addison-Wesley Professional.

2.7.6. ***Descripción de los Actores del Sistema***

Tabla 1. Definición Resumida de los Actores del Sistema

Actor del Negocio	Descripción
<Usuario>	El usuario es el empleado principal encargado de modificar ingresar y eliminar platillos del menú. Este usuario accede al sistema a través de un usuario y su contraseña el cual le habilita el menú principal del programa. Este actor debe contar con un control de las computadoras básico, una capacitación mínima casi nula, y responsabilidad de mantener los platillos actualizados en el sistema.

Actor del Negocio	Descripción
<Cliente>	El cliente es el usuario encargado de visualizar el menú para pedir a posteriori sus platillos. Este usuario simplemente se encarga de visualizar los platillos.

2.7.7. ***Especificación de Requisitos***

I. *Descripción de los Requisitos Funcionales*.

- Requisito funcional 1 Poder ingresar nuevos platillos al menú para que el cliente tenga una forma más cómoda de poder administrar su propio menú.
- Requisito funcional 2 Los platillos del menú deben de estar ordenados de forma secuencial puesto que el cliente quiere esto para tenerlo de forma más organizada.
- Requisito funcional 3 El cliente deberá poder tener un apartado donde pueda ingresar un código de un producto y este saque el platillo.
- Requisito funcional 4 Deberá generar un número aleatorio de dos dígitos (45). Concatenar la primera letra de la especialidad a la que pertenece cada producto (PIZZAS, PICAR Y COMPARTIR, HELADOS,BEBIDAS). Sólo en el caso de “Picar y compartir”, considerar las dos primeras letras (P, PI, H y B). Por último concatenar

las 3 primeras letras del nombre del producto. Ejemplo: 45PTEJ 68PIALA.

- Requisito funcional 5 El software deberá poder leer archivos txt para poder generar las llaves.
- Requisito funcional 6 El software generará un archivo txt.
- Requisito funcional 7 El software deberá contar obligatoriamente con una función que rellene con el carácter " " espacio los espacios vacíos del menú para que estos salgan de forma ordenada.
- Requisito funcional 8 El software contempla que a la hora de poner el caracter “|” sea un delimitador de los campos establecidos por la pizzería los cuales son: Código, nombre, descripción, precio, especialidad.
- Requisito funcional 9 El índice primario simple, sólo mantiene registros con dos campos: llave primaria y NRR (en este orden).
- Requisito funcional 10 Documentar cada función y parte importante de su código para que el clienta en dado caso de dar mantenimiento pueda fácilmente entender el código

Requisitos no funcionales:

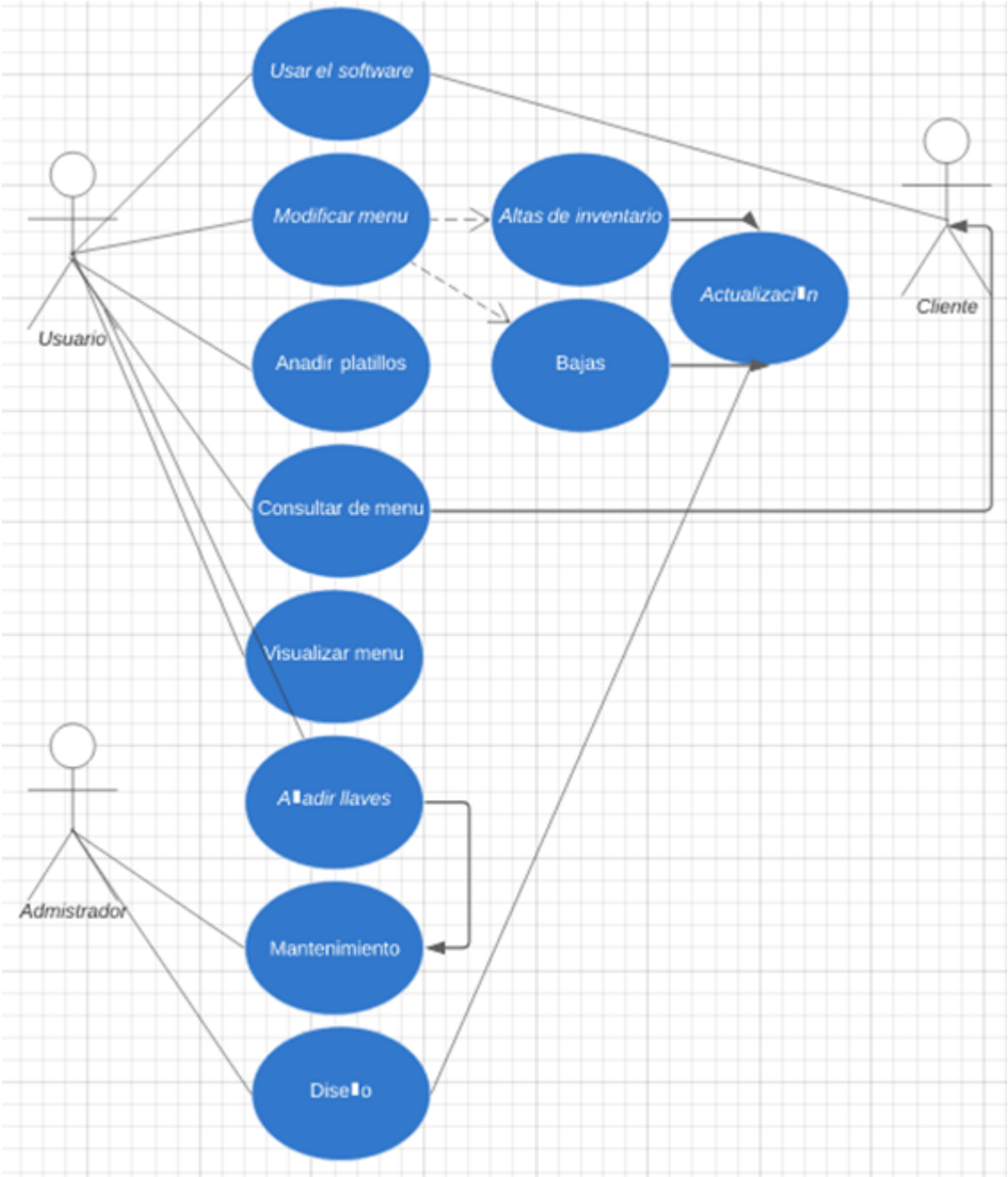
- Requisitos de rendimiento: Se espera que al agregar algo nuevo al menú no deberá pasar de un minuto para que se refleje en el programa. Así como también se espera al menos 200 productos guardados de manera simultánea el sistema no cuente con ninguna falla, también deberá soportar al menos 500 acciones por 5 minutos en el sistema, al buscar un menú no tardar más de 10 segundos en la búsqueda de dicho menú.
- Seguridad: Se mantendrá la información tanto como los menús cifrados para que ninguna persona con intenciones maliciosas pudiera acceder a esta información, garantizando la seguridad del sistema informático para los usuarios.
- Fiabilidad: El sistema contará con una interfaz sencilla y de uso práctico para el usuario, la interfaz se ajustará a las características del programa para el uso de gestión de menú.
- Disponibilidad: El sistema deberá estar disponible las 24 horas de todos los días excepto el día domingo y a excepción de caso de mantenimiento el cual será anunciado un día antes.
- Mantenibilidad: El sistema deberá tener con la documentación adecuada y fácilmente actualizable que permita realizar las operaciones de mantenimiento con una mayor facilidad así ahorrando tiempo a la persona que estará a cargo de la mantenibilidad, el cual en caso de hacerse tendrá que poner un aviso en el programa un día antes para que los usuarios estén conscientes de esto
- Portabilidad: El sistema apto para poder hacer uso óptimo del software es una versión no menos baja que Windows 10, el cual podrá ser visto desde cualquier dispositivo computacional.

Otros requisitos

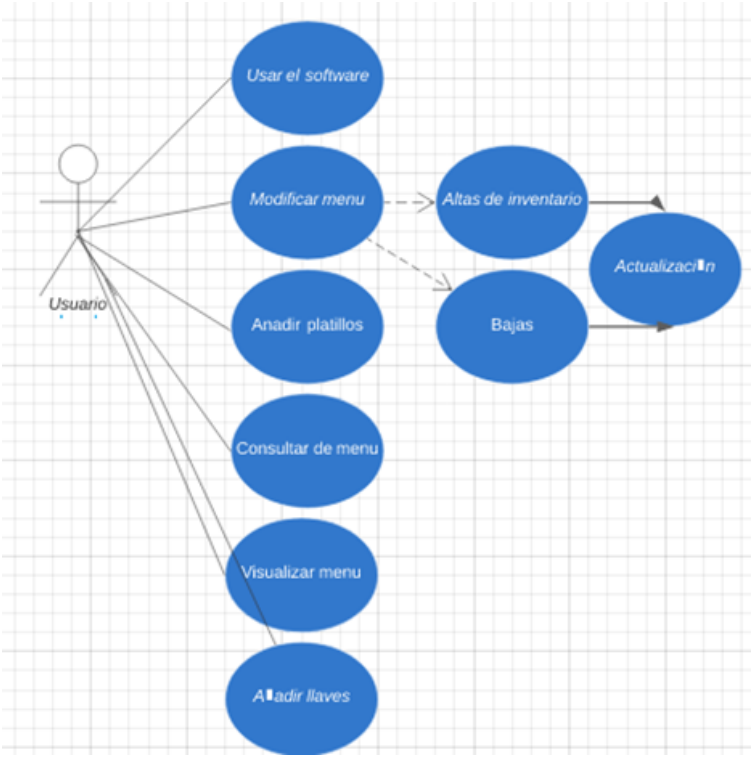
- Requisitos comunes de los interfaces: El diseño se llevará a cabo acorde al diseño que ya está establecido meses atrás por el cliente
- Interfaces de usuario: Que la interfaz sea simple e intuitiva, sin elementos innecesarios, fácil de entender solo usando el teclado. Las información impresa en pantalla ordenadamente y que todo se base en texto.
- Interfaces de hardware: Para el uso óptimo del software se necesitará al menos un sistema operativo actualizado para poder hacer que sea compatible con cualquier tipo de máquina, tener en su posición mouse, teclado.
- Interfaces de software: No aplica porque la terminal de Windows no es un producto de software.
- Interfaces de comunicación: No requiere puesto que es en un solo sistema

2.7.8. *Vista Funcional*

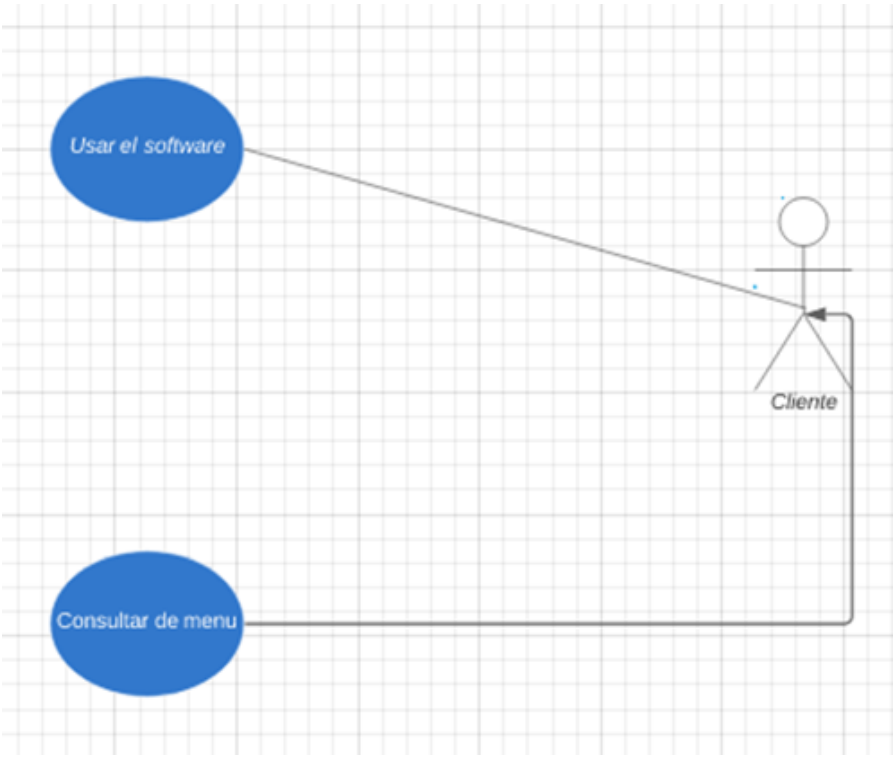
Diagrama de Casos de Uso relevantes a la Arquitectura



Caso de Uso relevante a la Arquitectura 1



Caso de Uso relevante a la Arquitectura 2



Caso de Uso relevante a la Arquitectura 3

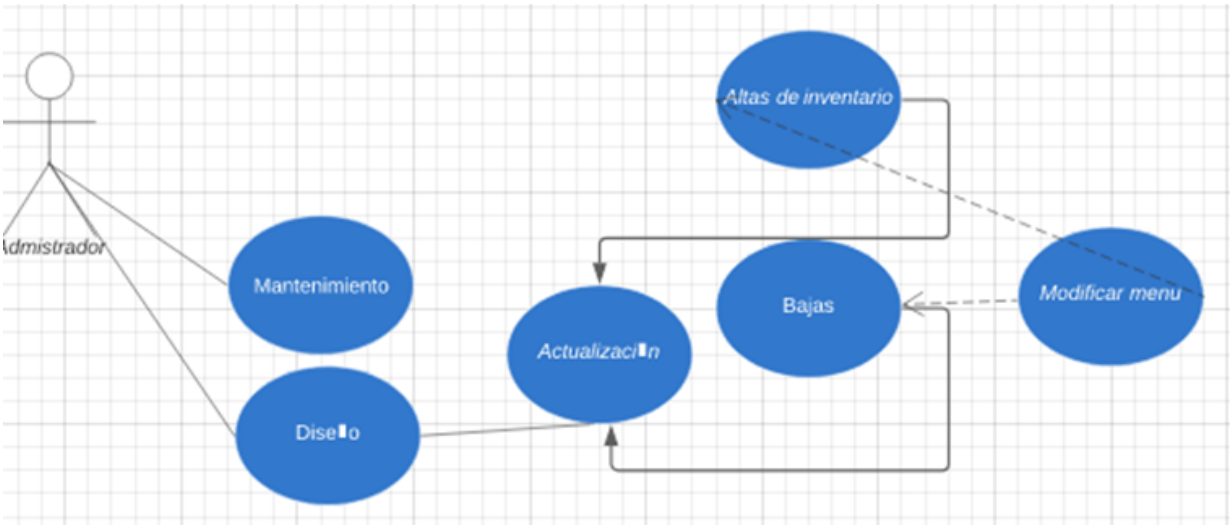


Tabla2. Descripción de Bajo Nivel de los Casos de Uso

Caso de uso relevante a la arquitectura	Interacción del usuario en el administrador, el cliente y sus funcionalidades.
Actores	Usuario, Usuario, Cliente.

Descripción del flujo de trabajo. Se debe indicar:

1. El usuario inicia sesión en el sistema.
2. El usuario selecciona la transacción añadir platillos en el sistema
3. El usuario selecciona modificar menú en caso de error para actualizar o dar de baja un platillo.
4. El usuario añade las llaves al menú.
5. El administrador le da mantenimiento al sistema.
6. El cliente ingresa en el sistema.
7. El cliente selecciona la transacción y consulta el menú.

Excepciones:

- El usuario debe estar logueado para modificar e ingresar platillos.
- El administrador le da mantenimiento fuera del horario laboral de la pizzeria.
- Debe estar agregado por lo menos un platillo en el menú para poder mostrarlo.

Relaciones:
Usuario - Consultar menu - Cliente
Administrador - actualizar - usuario

Requisitos especiales	El usuario deberá estar previamente registrado y logueado antes de hacer algún cambio. El usuario solo podrá acceder al sistema en el horario establecido de trabajo a menos que cuente con permiso del superior(admin).

2.7.9 ***Vista Lógica***

* *Estilo Arquitectónico*: Define la estructuración en capas de la aplicación.

* *Patrones de Diseño*

Patrón de diseño de fábrica: puede ser utilizado para crear diferentes tipos de pizzas con diferentes ingredientes y tamaños siendo este el diseño de fábrica del sistema Windows 10(terminal).

```
                        -->|BIENVENIDO A TELEPIZZA|<--

    Selecciona una de la siguientes opciones:

        (1).-Insertar registro
        (2).-Mostrar un registro por su Codigo
        (3).-Salir

-->
```

- Patrones de Comportamiento

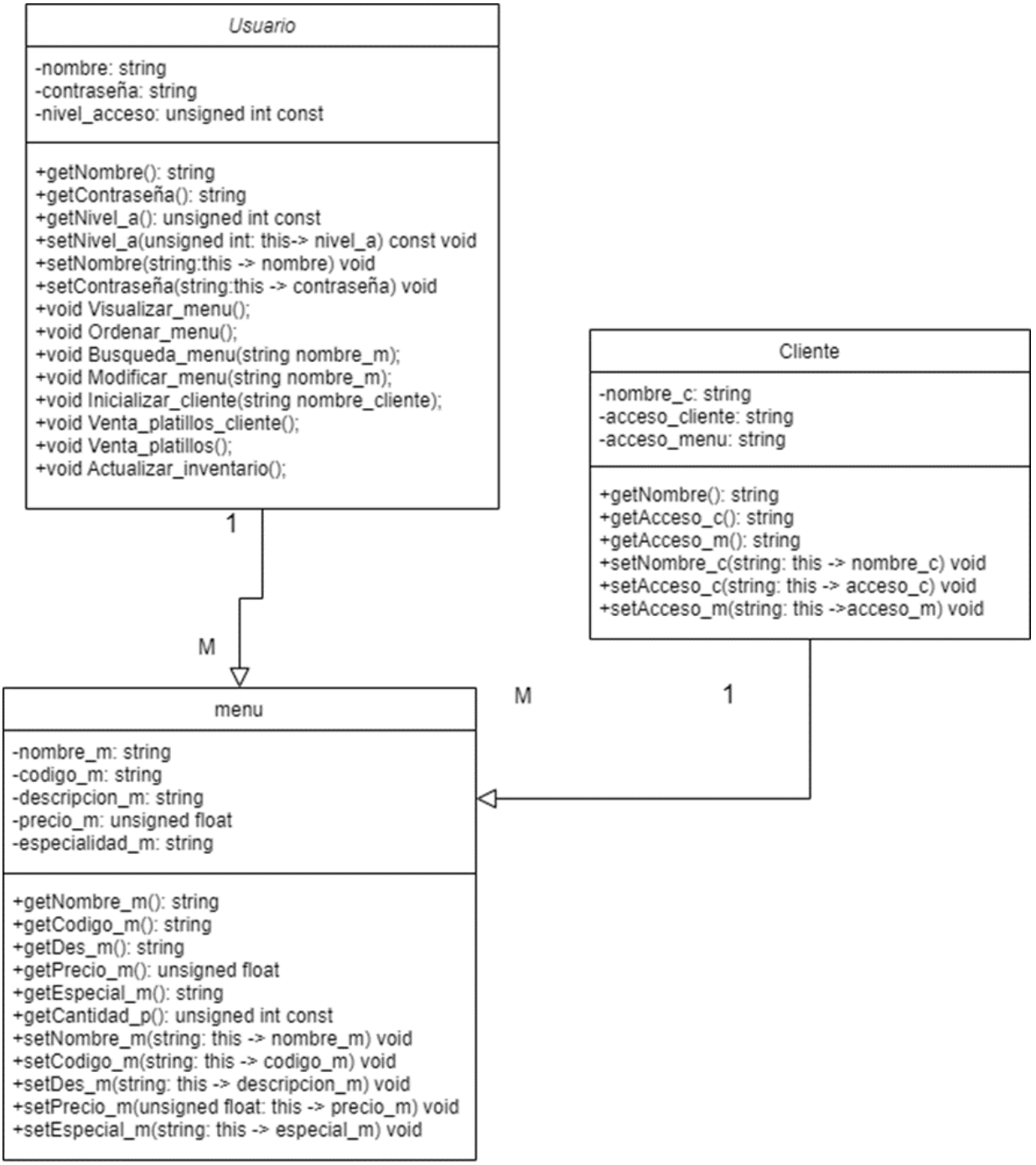
  ✔ **A nivel de objetos:** Patrón de menú: Este patrón se utiliza para diseñar la estructura del menú y cómo los diferentes elementos del menú interactúan entre sí. En el caso de una pizzería, esto puede incluir categorías de pizza, como "Margherita", "Pepperoni" o "Hawaiana", así como opciones adicionales, como "salsa extra" o "sin queso".

Patrones de Diseño Detallados

Mencionar los patrones de diseño que se utilizan, especificando para cada patrón:

a) Breve descripción del patrón.
MVC
(Modelo-Vista-Controlador).
Es un enfoque arquitectónico que separa la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Cada uno de estos componentes tiene una función específica
b) ¿Dónde y por qué lo usan?
Al utilizar el patrón de diseño MVC, se logra una separación clara de responsabilidades entre los diferentes componentes de la aplicación, lo que permite una mayor modularidad y reutilización de código. Además, esto permite que cada componente pueda ser modificado o reemplazado sin afectar a los demás componentes de la aplicación, lo que facilita el mantenimiento y la evolución de la aplicación.

c) Diagrama de clases que muestre cómo se implementa el patrón en la propuesta realizada del sistema.



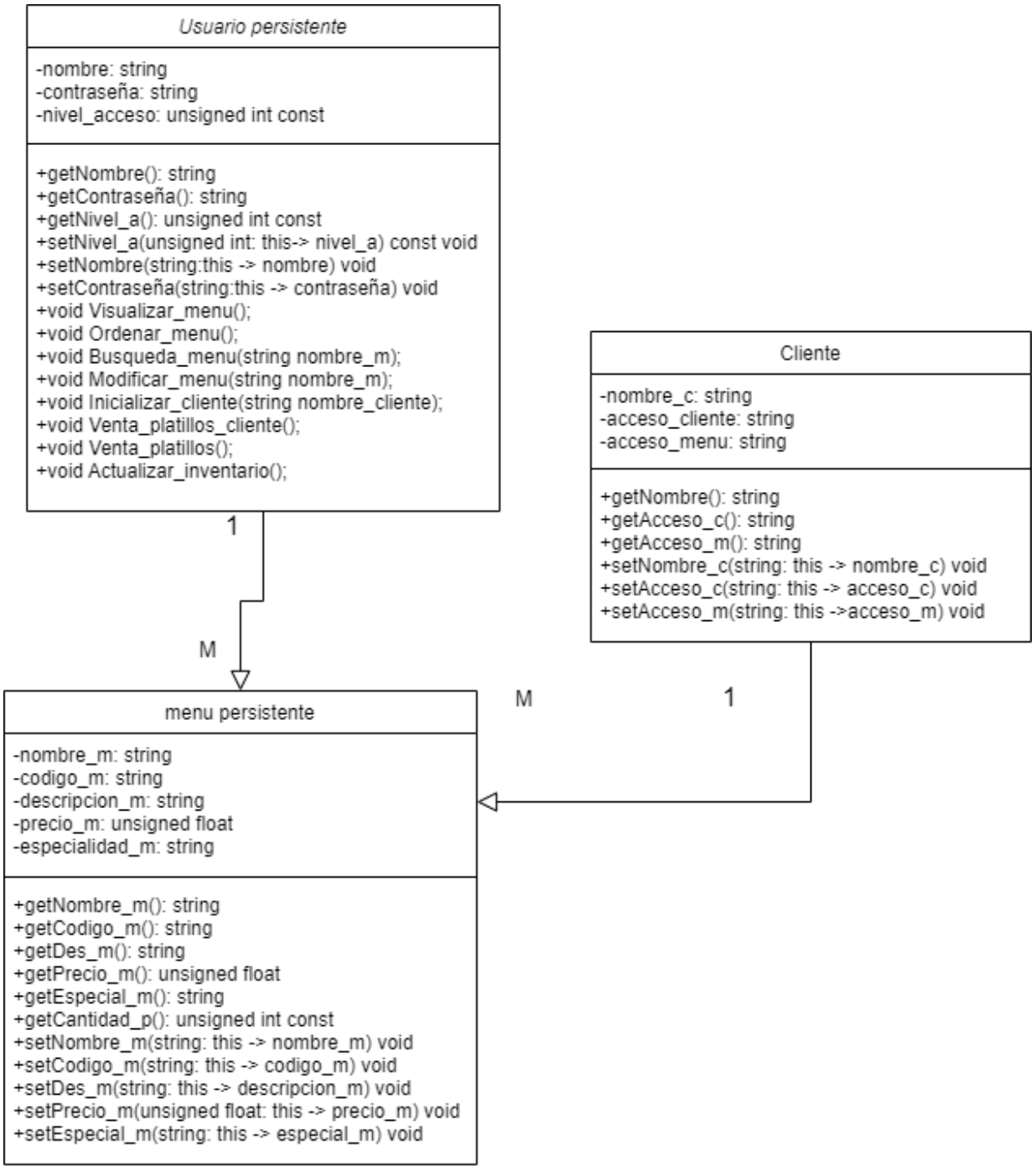
* *<u>Mecanismos de Diseño</u>*

✔ Mecanismo de Seguridad al utilizar métodos canónicos (Todo mayúsculas sin espacios vacíos etc).

✔ Mecanismo de Acceso a Datos utilizando archivos almacenados en windows.

2.7.10. ***Modelo Lógico de Datos***

I*Diagrama de Clases Persistentes:*



2.7.11. ***Modelo Físico de Datos***

Empleado para describir la estructura de la información persistente manipulada por el sistema. Se debe incluir su respectivo diccionario de datos.

n/a

2.7.12. ***Descripción Detallada de los Algoritmos***

Se deben explicar de manera exhaustiva en lenguaje natural los algoritmos y diagramas de flujo más importantes y complejos.

El programa utiliza la estructura de datos "Menú" para almacenar los detalles de cada elemento, como el índice, el código, la especialidad, el nombre, la descripción y el precio. Además, el programa utiliza la estructura de datos "stIndicecodi" para almacenar la llave primaria (el código) y el número de índice.

```
int main() {// Menu del programa
	int opc;
	do {
	// system("clear");
	cout << "\t\t\t -->|BIENVENIDO A TELEPIZZA|<-- \n\n";
	cout << "    Selecciona una de la siguientes opciones:\n\n";
	cout << "\t(1).-Insertar registro\n" << "\t(2).-Mostrar un registro por
su Codigo\n" << "\t(3).-Salir\n\n" << "-->";
	cin >> opc;
	switch(opc)
	{ //switch para entrar en cada uno de los casos del menu
		case 1:
		{
			// std::cout << "Opcion 1" << '\n';
			// sleep (1);
			me.Ingresar();//manda a llamar la funcion ingresar
			  me.Ordenamiento();//manda a llamar la funcion ordenamiento
			break;
		}
		case 2:
		{
			// std::cout << "Opcion 2" << '\n';
			me.Buscar();//manda a llamar la funcion de buscar
			// sleep (1);
			break;
		}
	}
	}
	while(opc != 3);
	return 0;
	}
```

El programa comienza abriendo el archivo "MENU.txt" en modo de apéndice (modo "append") para determinar el número actual de líneas en el archivo, que se utiliza para calcular el índice del nuevo elemento a agregar. Luego, utiliza la función srand() para inicializar el generador de números aleatorios basado en la hora actual del sistema. El programa luego utiliza la cadena de caracteres "random" para generar aleatoriamente los dos primeros bytes del código, y el tercer y cuarto bytes se toman de la especialidad. Si la especialidad es "PICAR Y COMPARTIR", el programa utiliza el primer y segundo bytes del nombre para el tercer y cuarto bytes del código.

El programa solicita al usuario que ingrese el nombre de la especialidad, que se almacena en la estructura de datos "Menú". Luego, utiliza la función Rellenar() para rellenar los espacios vacíos de la cadena de caracteres "especialidad" con espacios en blanco. Después de eso, el programa escribe la especialidad en el archivo "MENU.txt" y, si corresponde, utiliza el nombre para generar los dos últimos bytes del código.

```
void Menu::Ingresar()
{ //funcion para ingresar datos
    //indice method
      // indice+=1;
      string texto;
      int j=0;

      ifstream archivo;//abrirmos el archivo en modo lectura
      archivo.open("MENU.txt",std::ios::app);
      while(!archivo.eof())
      {
      getline(archivo,texto);
      j+=1;
      }
      archivo.close();//ceramos el archivo
      indice=j;//pasamos j a indice
      std::ofstream archivomenu;//abrimos el archivo en modo escritura
      archivomenu.open("MENU.txt",std::ios::app); //Abriendo y creando archivo
      archivomenu << indice << "|";//se escribe el indice
      char  random[10]=  {'0','1','2','3','4','5','6','7','8','9'};//cadena de
numeros a tomar aleatoreamente para la llave canonica
      fflush(stdin);
      //Forma Canonica
      //Dos bits [0:1]
      srand  (  time(NULL)  );  //  Tomar  time  como  referencia  para  generar
diferente NRG
      codigo[0]  =  random[rand()%10];//tomamos  el  cero  byte  del  codigo  y  le
aplicamos rand con nuestra cadeba ya hecha
      codigo[1]  =  random[rand()%10];//tomamos  el  primero  byte  del  codigo  y  le
aplicamos rand con nuestra cadena ya hecha
      //Dos bits [1:2|3]
      cout << "\tDame el nombre de la especialidad: " << endl;
      cin.getline(especialidad,18);//entra el nombre de la especialidad
      int i;
      for(i=0; i<=strlen(especialidad); i++)
      { // pone en mayusculas la especialidad
      if(especialidad[i]>=97 && especialidad[i]<=122)
      {
            especialidad[i]=especialidad[i]-32;
      }
      }
      aux2  =  Rellenar(especialidad,18);//rellena  con  espacios,  los  espacios
vacios
    strcpy(especialidad,aux2.c_str());//aux2 se escribe en especialidad
      archivomenu << especialidad << "|";// especialidad se escibe al archivo
"MENU.txt"
```

```
      fflush(stdin); ////////////////
      if (aux2 == "PICAR Y COMPARTIR ")
      {//if para seguir la forma canonica
      codigo[2]=especialidad[0];//tomamos el cero byte de especialidad y lo
insertamos en el codigo en el byte 2
      codigo[3]=especialidad[1];//tomamos el primer byte de especialidad y lo
insertamos en el codigo en el byte 3
      fflush(stdin);
      memset(especialidad, 0, 18);//limpia para no unirlo con basura
```

El programa continúa pidiendo al usuario que ingrese el nombre del elemento, que se almacena en la estructura de datos "Menú". Nuevamente, el programa utiliza la función Rellenar() para rellenar los espacios vacíos de la cadena de caracteres "nombre" con espacios en blanco. El programa solicita la descripción y el precio del elemento, que se almacenan en la estructura de datos "Menú". El programa utiliza la función Rellenar() para rellenar los espacios vacíos de la cadena de caracteres "descripción" y "precio" con espacios en blanco.

```
cout << "\tDame el nombre: " << endl;
      cin.getline(nombre,25);//recibe el nombre
      int i;
      for(i=0; i<=strlen(nombre); i++)
      { //pone en mayusculas el nombre
            if(nombre[i]>=97 && nombre[i]<=122)
            {
            nombre[i]=nombre[i]-32;
            }
      }
      aux2 = Rellenar(nombre,25);//rellena con espacios, los espacios vacios
en nombre
      strcpy(nombre,aux2.c_str());//aux2 se escribe en nombre
      archivomenu << nombre << "|";//nombre se escribe en el archivo
"MENU.txt"
      codigo[4] = nombre[0];//codigo toma el 4 bit de codigo y le asigna el
primer bit de nombre siguiendo la forma canonica
      codigo[5] = nombre[1];//codigo toma el 5 bit de codigo y le asigna el
segundo bit de nombre siguiendo la forma canonica
      codigo[6] = nombre[2];//codigo toma el 6 bit de codigo y le asigna el
tercero bit de nombre siguiendo la forma canonica
      memset(nombre, 0, 25);//limpia para no unirlo con la basura
      }
      else
      {
      codigo[2]=especialidad[0];//codigo toma el 0 bit de especialida y le
asigna el segundo bit siguiendo la forma canonica
      fflush(stdin);
      memset(especialidad, 0, 18);//limpia para no unirlo con basura
      cout << "\tDame el nombre: " << endl;
      cin.getline(nombre,25);//recibe el nombre
      int i;
      for(i=0; i<=strlen(nombre); i++)
      { //pone en mayusculas el nombre
            if(nombre[i]>=97 && nombre[i]<=122)
            {
            nombre[i]=nombre[i]-32;
            }
      }
      aux2 = Rellenar(nombre,25);//rellena con espacios, los espcacios vacios
en nombre
      strcpy(nombre,aux2.c_str());// aux2 se escribe en nombre
      archivomenu << nombre << "|";//nombre se escribe en el archvo "MENU.txt"
      fflush(stdin); ///////
```

```
	codigo[3] = nombre[0];//codigo toma el 3 bit de codigo y le asigna el primer bit de nombre siguiendo la forma canonica
	codigo[4] = nombre[1];//codigo toma el 4 bit de codigo y le asigna el segundo bit de nombre siguiendo la forma canonica
	codigo[5] = nombre[2];//codigo toma el 5 bit de codigo y le asigna el tercero bit de nombre siguiendo la forma canonica
	memset(nombre, 0, 25);//limpia nombre para no unirlo con basura
	}
	fflush(stdin);
	cout << "\tDame la descripcion: " << endl;
	cin.getline(descripcion,180);//recibe descripcion
	for(i=0; i<=strlen(descripcion); i++)
	{ //pone en mayusculas la descripcion
	if(descripcion[i]>=97 && descripcion[i]<=122)
	{
		descripcion[i]=descripcion[i]-32;
	}
	}
	aux2 = Rellenar(descripcion,180);//rellena de espacios, los espacios vacios de la descripcion
	strcpy(descripcion,aux2.c_str());//aux2 se escribe en descripcion
	archivomenu << descripcion << "|";//escribe el campo descripcion en el archivo "MENU,txt"
	aux2 = Rellenar(codigo,7);//rellena de espacios vacios codigo
	strcpy(codigo,aux2.c_str());//aux2 se escribe en codigo
	archivomenu << codigo << "|"; // escribe el codigo con la forma canonica descripcion en el archivo "MENU,txt"
	fflush(stdin); ////////////////////////
	memset(descripcion, 0, 180);
	//precio
	cout << "\tDame el precio: " << endl;
	cin.getline(precio,9);//recibe precio
	aux2 = Rellenar(precio,9);//rellena de espacios vacios precio
	strcpy(precio,aux2.c_str());//aux2 se escribe en precio
	archivomenu << precio << "|" << endl;//escribe el precio con la forma canonica descripcion en el archivo "MENU,txt"
	// cout<<"\nCodigo: "<< codigo << "|" << endl;//imprime el codigo
	fflush(stdin);
	archivomenu.close();//se cierra el archivo

	std::ofstream archivoindice;// se abre el archivo "INDICE.TXT" en modo escritura
	archivoindice.open("INDICE.txt",std::ios::app);
	in.indice=indice;//se mete indice en el indice del archivo "INIDICE.TXT" el cual va a ser el nrr
	strcpy(in.codigo,codigo);//se mete codigo a codgio del archivo "INDICE.TXT"
	archivoindice << in.codigo << "|"; //escribe codigo en el archivo "INDICE.TXT"
	archivoindice << in.indice << endl; //escribe indice o en este caso el nrr en el archivo "INDICE.TXT"
	archivoindice.close();// se cierrra el nrr

	memset(codigo, 0, 7);//se limpia el codigo para que no lo una con basura
	memset(descripcion, 0, 180);
	memset(nombre, 0, 25);
	memset(especialidad, 0, 18);//se limpian todos los datos para que no los una con basura
	}
```

Finalmente, el programa escribe el nuevo elemento en el archivo "MENU.txt" y actualiza el archivo de índice "indicecodi.txt" con la nueva llave primaria y el número de índice correspondiente. El programa también utiliza la función Ordenamiento()

para ordenar los elementos del archivo "MENU.txt" por orden alfabético de acuerdo con la especialidad y el nombre. Además, el programa utiliza la función Buscar() para buscar elementos en el archivo "MENU.txt" y mostrar sus detalles en pantalla.

```
void Menu::Ordenamiento()
{ //funcion que ordena el indice
	int i=0, r, j;
	string aux1;
	vector <string> v (MAX);
	ifstream archivo1 ("INDICE.txt");//se abre el archivo en modo de lectura
	if (archivo1.fail()) cout<<"El archivo no se abrio correctamente."<<endl;//en caso de el archivo no se abrio correctamente
	while (getline(archivo1, v[i]))
	{ //Guardamos las lineas en vector
	// cout << v[i] << endl;
	i++;
	}
	archivo1.close();
	//Ordenamiento de Codigo
	for(int i=0; i<indice-1; i++)
	{ //forma de ordenamiento de bubblesort
	for(int j=i+1; j<indice; j++)
	{
		if(v[i]>v[j])
		{
		aux1 = v[j];
		v[j] = v[i];
		v[i] = aux1;
		}
	}
	}
	// std::cout << "\nORDENADA: " << '\n';
	//Impresion de Ordenamiento
	std::ofstream archivo2;// se abre el archivo en escritura
	archivo2.open("INDICE.txt",std::ios::ate); // Abrir archivo en modo ios:ate para sobreescribir
	for (int i= 0 ; i < indice ; i++)
	{
	archivo2 << v[i] << endl;
	}
	archivo2.close();//se cierra el archivo

	// std::cout << "Funcion Ordenamiento" << '\n';
	// sleep (1);
}
void Menu::Buscar()
{ //funcion para buscar y recuperar el dato
	int i=0, r, j, size, ia=-1;
	string aux1, aux2;
	std::cout << "Dame el codigo a buscar: " << '\n';
	std::cin >> aux1;
	size=aux1.size();//saca el tamaÃ±o de aux1
	std::transform(aux1.begin(), aux1.end(), aux1.begin(), ::toupper);//pone todo en mayusculas
	vector <string> v (MAX);
	ifstream archivo1 ("INDICE.txt");//abre el archivo en modo lectura
	if (archivo1.fail()) cout<<"El archivo no se abrio correctamente."<<endl;// ver si el archivo falla
	while (getline(archivo1, v[i]))
	{ //Guardamos las lineas en vector
	i++;
	}
	for (size_t j = 0; j < i; j++)
```

```
	{// for para comparar codigos
	if(v[j].substr(0,size) == aux1)
	{//compara el codigo con el auxiliar1
		ia = std::stoi(v[j].substr(8,2));//convierte a entero
	}
	}
	if (ia == -1)
	{
	std::cout << "No hay match :C" << '\n';// condicional por si no hay registro
	}
	else
	{
	i=0;
	vector <string> recuperarNRR (MAX);
	ifstream archi ("MENU.txt");//abre el archivo en forma de lectura
	if (archi.fail()) cout<<"El archivo no se abrio correctamente."<<endl;
	while (getline(archi, recuperarNRR[i]))
	{ //Guardamos las lineas en vector
		i++;
	}
	archi.close();//cerrarmos el archivo
	std::cout << recuperarNRR[ia-1] << '\n';//mostramos a pantalla el registro buscado
	}
	archivo1.close();//ceramos el archivo
}
```

2.7.13. ***Diseño de Pantallas y Reportes***

Breve descripción de las consideraciones asociadas al diseño de las pantallas y reportes de la aplicación.

N/A

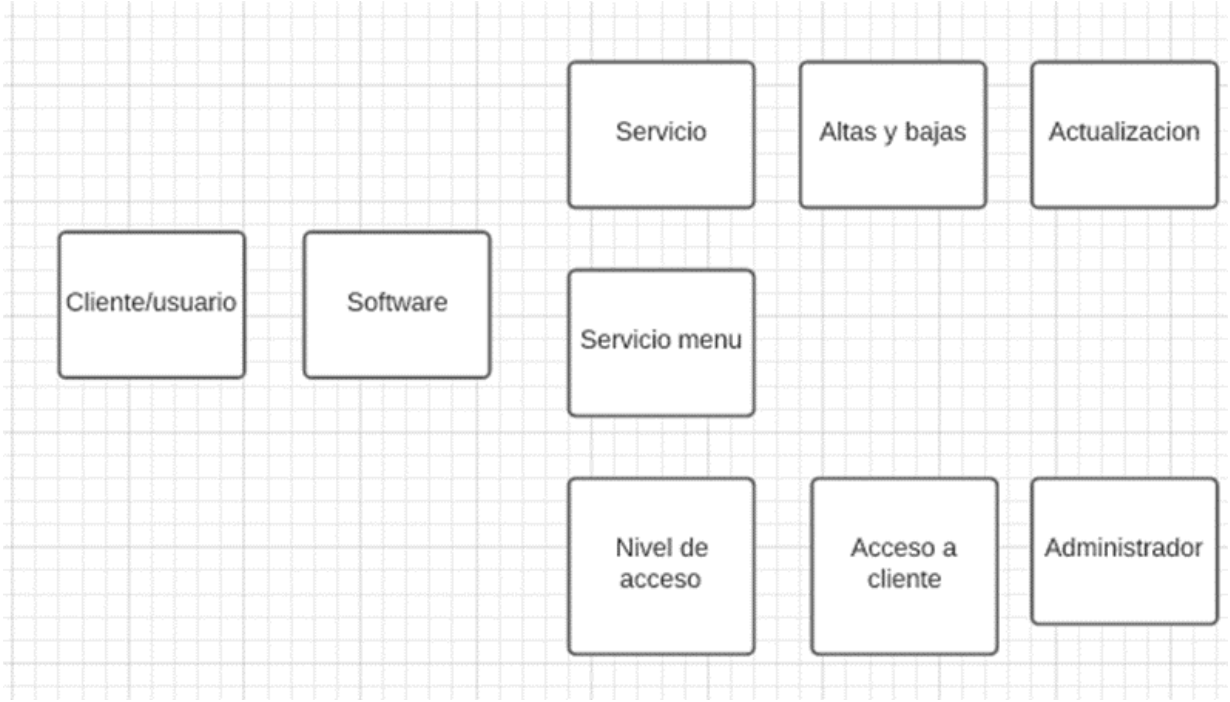
2.7.14. ***Descripción de Campos Requeridos por Pantalla***

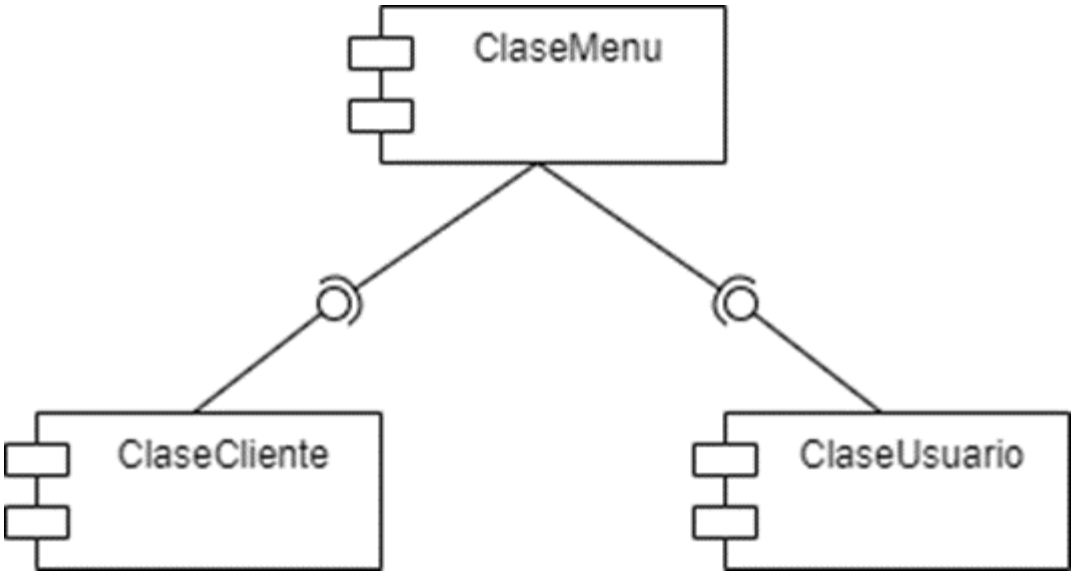
Incluir imágenes de las pantallas.

```
                        -->|BIENVENIDO A TELEPIZZA|<--

    Selecciona una de la siguientes opciones:

        (1).-Insertar registro
        (2).-Mostrar un registro por su Codigo
        (3).-Salir

-->
```

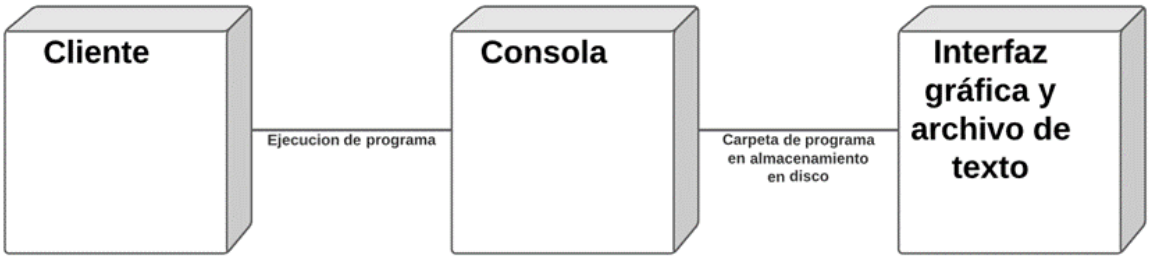
2.7.15. ***Vista de Implementación***

Compuesta fundamentalmente por un diagrama que muestra las principales piezas desde el punto de vista físico (subsistemas de implementación) que conforman el sistema. Dicho diagrama representa la estructuración física del código, los distintos directorios que organizan el código fuente, librerías, ficheros ejecutables (si existen), entre otros elementos. Se deben describir los componentes mostrados en el diagrama, incluyendo su propósito y contenido. Además, han de considerarse como parte de la vista las dependencias con aplicaciones o componentes externos.

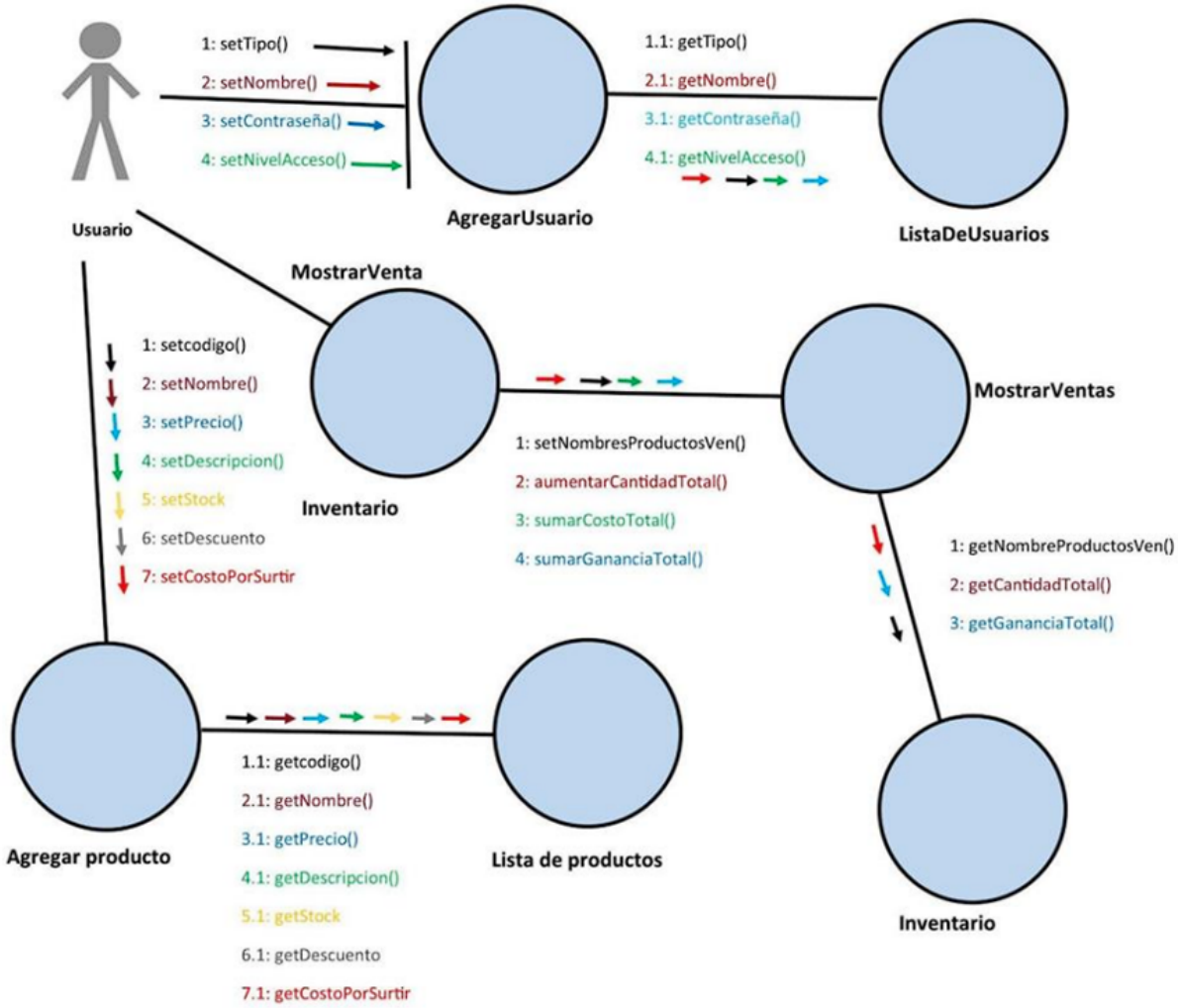



ClaseMenu
ClaseCliente
ClaseUsuario

2.7.16. *Vista de Despliegue*



2.7.17. *Diagrama de Navegación del Sistema*



2.7.18. ***Controles de Auditoría Implementados en el Sistema***

Adjuntar la documentación vinculada a los controles de auditoría implementados en el sistema.

- Control de accesos: asegurarse de que solo las personas autorizadas tienen acceso al software de menú y que sus permisos de acceso están definidos adecuadamente.
- Registro de cambios: mantener un registro de todas las modificaciones realizadas en el software de menú, incluyendo quién las hizo y cuándo.
- Validación de entradas: verificar que la información ingresada en el software de menú es válida y consistente.
- Verificación de cálculos: asegurarse de que los cálculos de precios y descuentos son precisos y están correctamente aplicados.

2.7.19. ***Glosario de Términos***

En él se definen los conceptos y términos importantes para la comprensión del problema, así como de todos los procesos descritos anteriormente.

- Manual Técnico: documento que contiene las especificaciones técnicas más importantes de un sistema desarrollado y que sirve como guía para las operaciones de mantenimiento de la aplicación.
- Dirección de Tecnologías de la Información: área encargada de la gestión y administración de la tecnología de la información en una organización.
- Administrador del sistema: persona encargada de la gestión y mantenimiento de un sistema informático.
- Departamento de calidad y auditoría de sistemas: área encargada de garantizar la calidad y seguridad de los sistemas informáticos de una organización.
- Confección: proceso de elaboración de un manual técnico.
- Normativos: aspectos que deben ser considerados y cumplidos en la elaboración de un manual técnico.
- Índice: lista de los apartados o secciones que conforman un documento, ordenados de manera sistemática.
- C++: lenguaje de programación de alto nivel, orientado a objetos y utilizado para la programación de sistemas, juegos y aplicaciones de escritorio.
- Estructuras: elementos de programación que permiten almacenar y organizar datos de diferentes tipos en una misma variable.
- Archivo: conjunto de datos almacenados en un medio de almacenamiento, como un disco duro o una memoria USB.
- Menú: Es el conjunto de opciones que se presentan al usuario para interactuar con el sistema.
- Secuencial: Se refiere a la ordenación de elementos en una secuencia específica, en este caso los platillos del menú.
- Alfanumérico: Se refiere a una combinación de letras y números utilizados para identificar un producto o registro en el sistema.
- Archivos txt: Son archivos de texto plano que se utilizan para almacenar información en el sistema.
- Delimitador: Es un carácter utilizado para separar diferentes campos de información en un archivo, en este caso el carácter "|" es utilizado como delimitador entre los campos de código, nombre, descripción, precio y especialidad de los platillos del menú.

- Llave primaria: Es un campo o conjunto de campos que identifican de manera única cada registro en una base de datos.
- NRR: Número de registro relativo, utilizado para acceder a registros en una base de datos.
- Fiabilidad: Se refiere a la capacidad del sistema para funcionar de manera correcta y confiable en todas las situaciones.
- Requisito funcional: Es una especificación de lo que el sistema debe hacer, en este caso se refiere a los requerimientos funcionales del software para la gestión del menú.
- Requisito no funcional: Es una especificación de cómo debe funcionar el sistema, en este caso se refiere a los requerimientos de rendimiento, seguridad, fiabilidad, disponibilidad, mantenibilidad y portabilidad del software.
- Sistema: Es el conjunto de componentes y subsistemas que trabajan juntos para llevar a cabo una función específica.
- Subsistemas: Son partes del sistema más grandes que se encargan de tareas específicas.
- Librerías: Son colecciones de código que se utilizan para facilitar el desarrollo de software, en este caso pueden ser utilizadas para la lectura y escritura de archivos txt.
- Ficheros: Son archivos que se utilizan para almacenar información en el sistema, en este caso los archivos txt son un ejemplo de ficheros utilizados para el almacenamiento de información del menú.

3. Estándares de Elaboración del Manual

Toda la documentación que se relacione con un sistema, ya sea impresa o digital, sencilla o compleja, debe reunir los siguientes requisitos básicos:

Debe ser rotulada con claridad y bien organizada en carpetas e índice, con secciones claramente indicadas.

Los diagramas deberán ser claros, no aglomerados y la escritura manuscrita ha de ser legible

La documentación deberá ser completa.

Se incluirá una leyenda o explicación de los términos utilizados.

La documentación siempre se conserva actualizada.

El estilo de redacción de los manuales de documentación debe:

Ser concreto.

Definir los términos utilizados.

Utilizar títulos, subtítulos y párrafos cortos.

Emplear formas activas en lugar de pasivas.

Aplicar correctamente las referencias bibliográficas.

No usar frases largas que presenten hechos distintos.