Informe del Proyecto Integrador y Aula UPBFOOD*2023-2*

Fernando Javier Vega Sabino [1] [000501008], and Juan Nicolás Rey Vásquez [2] [000500772].

Instructor: Lenin Javier Serrano Gil
Materia: Estructura De Datos

Universidad Pontificia Bolivariana, Campus Bucaramanga, Colombia

Abstract. The scenarios detailed in this document transpired within the framework of the pedagogical strategy of integrative and classroom projects implemented by the Faculty of Systems Engineering and Informatics at the Pontifical Bolivarian University.

Across the document's various sections, it delineates the challenging scenario of the UPB FOOD restaurant, as posed in the academic exercise. It further encompasses the research process, development, results, and conclusions of the proposed solution, along with the progress achieved up to the present moment.

Resumen. Las situaciones descritas en este documento tuvieron lugar en el contexto de la estrategia pedagógica de proyectos integradores y de aula llevadas a cabo por la Facultad de Ingeniería de Sistemas e Informática de la Universidad Pontificia Bolivariana.

A lo largo de las diferentes secciones del documento, se presenta la situación problemática del restaurante UPB FOOD, planteada en el ejercicio académico, junto con el proceso de investigación, desarrollo, resultados y conclusiones de la solución proporcionada, así como los avances realizados hasta la fecha.

Keywords: UPB FOOD, progreso acdémico, enfoque integrador, avances.

1. Introducción

El restaurante Food UPB desea sistematizar su proceso de atención telefónica para pedidos de comida a domicilio en el área metropolitana de Bucaramanga. [1].

Dado lo expuesto previamente, la pregunta central que orienta el desarrollo del proyecto es la siguiente: ""¿Cómo implementar un sistema de gestión de pedidos a domicilio para el restaurante UPB FOOD, ubicado en el área metropolitana de Bucaramanga, implementando tres módulos de gestión: Administrador, Operador y Entrega?"

A partir de esta estructura, se establecen cinco ciclos, cada uno compuesto por cuatro fases específicas:

-Definición de objetivos: En esta etapa, se establecen de manera clara y precisa los

objetivos que se deben alcanzar en el ciclo, así como los pasos concretos que se deben seguir para lograrlos.

-Evaluación y mitigación de riesgos: Durante esta fase, se lleva a cabo una evaluación inicial de los posibles riesgos asociados al ciclo en curso. Se desarrolla un plan estratégico para reducir y solucionar cualquier debilidad o amenaza identificada.

-Desarrollo y validación: Aquí se procede a la creación del prototipo o desarrollo correspondiente al alcance definido en la fase anterior. Se verifica y valida el funcionamiento del trabajo realizado en relación con los objetivos establecidos.

-Planificación: En la última fase de cada ciclo, se establecen las funciones y requisitos necesarios que deben estar en marcha para el inicio del ciclo siguiente. Se planifican las actividades y recursos requeridos para garantizar una transición fluida al siguiente ciclo.

Estos cinco ciclos están enfocados en áreas específicas del proyecto: Ciclo 1 se centra en el "Módulo de gestión de usuarios". Ciclo 2 está dedicado al "Desarrollo de la base de datos". Ciclo 3 se enfoca en el "Tipo de usuario administrador". Ciclo 4 se ocupa de la "Implementación del módulo operador". Ciclo 5 se dedica al "Usuario de entrega". Cada uno de estos ciclos sigue el mismo patrón de fases con el propósito de avanzar gradualmente en el proyecto y asegurar el cumplimiento de los objetivos y la gestión apropiada de los riesgos.

2. Estado del arte y marco conceptual

Los temas tratados a lo largo del proyecto son: 1) Estructuras de datos, 2) Modelo vista controlador, 3) Metodología espiral, 4) Diseño de software, 5) Prueba de software, 6) Control de versiones, 7) Remote database connection, 8) SQL, 9) Base de datos, 10)Archivos jar, 11) Java swing (Intelillij), ,13)Ubuntu, 14) Máquinas virtuales.

2.1. Estructuras de datos.

Arrays (Arreglos)

Un array es una colección de elementos del mismo tipo de datos dispuestos en una secuencia contigua en la memoria. Los elementos de un array se pueden acceder mediante su índice. Son eficientes para el acceso aleatorio, pero no son eficientes para la inserción o eliminación de elementos en el medio de la estructura.

b. Listas Enlazadas

Una lista enlazada es una colección de nodos, donde cada nodo contiene un elemento de datos y un puntero (referencia) al siguiente nodo. Pueden ser de tipo "simples" (un solo enlace al siguiente nodo), "dobles" (un enlace al siguiente y otro al anterior) o circulares (el último nodo enlaza con el primero). Son eficientes para la inserción y eliminación de elementos, pero no son tan eficientes para el acceso aleatorio.

c. Pilas (Stacks)

Una pila es una estructura de datos que sigue el principio "LIFO" (Last-In-First-Out), lo que significa que el último elemento en ser insertado es el primero en ser eliminado. Se utiliza en tareas como seguimiento de llamadas de funciones y gestión de historiales de navegación.

d. Colas (Queues)

Una cola es una estructura de datos que sigue el principio "FIFO" (First-In-First-Out), lo que significa que el primer elemento en ser insertado es el primero en ser eliminado. Se utiliza para gestionar tareas en las que el orden de llegada es importante, como la gestión de pedidos en un restaurante.

Estas estructuras de datos pueden mejorar la eficiencia y la velocidad de la gestión de pedidos y entregas en nuestro restaurante UPB FOOD. Al implementarlas de manera adecuada, podremos optimizar la experiencia del cliente al garantizar entregas rápidas y precisas.

e. Colas de Prioridad

Una cola de prioridades es una estructura de datos donde los elementos tienen una importancia o prioridad asignada.

Los elementos con mayor prioridad se manejan primero, y si dos elementos tienen la misma prioridad, se tratan en el orden en que llegaron.

Las dos operaciones principales en una cola de prioridades son: Agregar con Prioridad: Permite añadir un elemento a la cola junto con su nivel de importancia. Eliminar el Elemento de Mayor Prioridad: Devuelve y saca de la cola el elemento más importante y antiguo que aún no ha sido eliminado. [1]

2.2. Modelo Vista Controlador

Modelo-vista-controlador es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. [2]

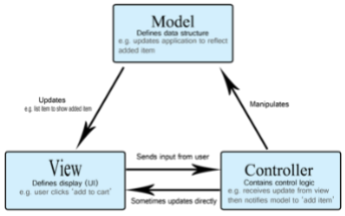


imagen 1. Patrón de modelo vista controlador. Tomada de [https://developer.mozilla.org/es/docs/Glossary/MVC]

2.3. Metodología

El modelo de desarrollo en Espiral combina elementos del enfoque "cascada" y un enfoque iterativo. En lugar de seguir un camino único y lineal, este modelo se divide en ciclos o "espirales" que abarcan distintas etapas, desde la planificación hasta el mantenimiento.

En cada ciclo, el equipo de desarrollo comienza con una idea básica y trabaja en una pequeña porción del proyecto. Luego, evalúan lo que han logrado y analizan los riesgos antes de avanzar. Después, agregan más funcionalidad al proyecto en cada ciclo subsiguiente hasta que la aplicación esté completa.

Este enfoque permite una adaptación continua a medida que se descubren nuevos requisitos o riesgos, lo que hace que el proceso sea más flexible y reduce las sorpresas desagradables en etapas posteriores del proyecto. En resumen, el modelo en Espiral combina planificación, desarrollo iterativo y evaluación constante para lograr un proceso de desarrollo más controlado y adaptable. [3]

2.4. Diseño de software

Se refiere al proceso de crear una estructura detallada y planificada para un programa informático antes de escribir el código real. Este proceso define cómo funcionará y se verá el software, incluyendo aspectos como la arquitectura, la interfaz de usuario y otros componentes esenciales. El diseño de software es esencial para asegurar que el desarrollo del programa sea eficiente, organizado y cumpla con los requisitos y expectativas del proyecto.

2.5. Prueba de software

Son un conjunto de procesos y actividades que tienen como objetivo evaluar y verificar la calidad y el funcionamiento correcto de un programa o aplicación informática. Estas pruebas se realizan para identificar errores, defectos, vulnerabilidades y problemas en el software antes de su lanzamiento o implementación en un entorno de producción. Las pruebas de software son una parte esencial del ciclo de desarrollo de software y contribuyen a garantizar que el software cumpla con los estándares de calidad y las expectativas del usuario final.

2.6. Metodología Espiral

La metodología espiral es un enfoque iterativo y flexible para el desarrollo de software. Se basa en la idea de que el desarrollo de software es un proceso de aprendizaje continuo, y que cada iteración de desarrollo debe incluir un ciclo de planificación, análisis de riesgos, desarrollo y evaluación. Esto permite una mayor adaptación a los cambios en los requisitos y una mayor calidad del software final.[11]

2.7. Control de versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación. [4]

2.7.1. Control de Versiones GIT

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones. [5]

2.9. REMOTE DATABASE CONNECTION:

Una remote database connection o conexión a una base de datos remota es un proceso que permite a un sistema o programa acceder a los datos almacenados en una base de datos que se encuentra en un sistema diferente al que está realizando la solicitud.

En esencia, cuando se trabaja con una base de datos remota, se establece una conexión entre el sistema local (donde se encuentra el programa que desea acceder a los datos) y el sistema remoto (donde reside la base de datos). Esto permite que el sistema local pueda enviar consultas, comandos y solicitudes de datos a la base de datos remota y recibir resultados o datos específicos de esa base de datos. . [7]

Una conexión a una base de datos remota permite acceder a datos distribuidos en diferentes sistemas y trabajar con ellos como si estuvieran localmente disponibles, facilitando así la gestión y el análisis de datos en entornos distribuidos o donde la información se almacena en ubicaciones geográficamente separadas. [8]

VIDEO EXPLICACIÓN CONEXIÓN A UNA BASE DE DATOS DE LOS MODULOS:
https://youtu.be/GQgcjPHKZAQ

2.10. SQL:

Es un lenguaje de programación utilizado para administrar y consultar bases de datos relacionales. Se utiliza para realizar tareas como la creación, modificación, recuperación y eliminación de datos en bases de datos, así como para definir la estructura y las relaciones de los datos en una base de datos.

SQL es una herramienta esencial para interactuar con bases de datos en la mayoría de los proyectos que involucran la gestión de datos.

Los sistemas de administración de bases de datos relacionales utilizan un lenguaje de consulta estructurada (SQL) para almacenar y administrar datos. El sistema almacena varias tablas de bases de datos que se relacionan entre sí. MS SQL Server, MySQL o MS Access son ejemplos de sistemas de administración de bases de datos relacionales. Los siguientes son los componentes de un sistema de este tipo. [9]

2.11. BASES DE DATOS:

Una base de datos es un conjunto de información organizada que se utiliza para gestionar datos de manera electrónica. En lugar de tener datos dispersos, una base de datos los relaciona y almacena en un formato digital.

Las bases de datos deben estructurarse de forma lógica y relacionar los datos entre sí. Esto evita la duplicación innecesaria y asegura que los datos se almacenen de manera eficiente.

Las bases de datos se utilizan ampliamente en la gestión empresarial, instituciones públicas y la investigación científica para almacenar información valiosa de manera eficiente. [10]

2.12. ARCHIVOS *JAR:*

Imagen 2. Archivo.Jar UPB FOOD (Imágen propia)

Un archivo JAR (Java Archive) es un formato de archivo utilizado en programación Java para agrupar y distribuir clases, recursos y bibliotecas. Este formato es especialmente útil en proyectos de desarrollo de software en Java, ya que permite empaquetar todos los componentes necesarios en un solo archivo comprimido. De esta manera, se simplifica la distribución y ejecución de aplicaciones Java, ya que todos los elementos se almacenan de manera organizada en un único archivo.

En un proyecto, un archivo JAR se utiliza para:

13.1. Empaquetar clases Java y recursos relacionados en un solo archivo.

13.2. Facilitar la distribución y la instalación de una aplicación Java, ya que se puede entregar como un único archivo descargable.

13.3. Garantizar la portabilidad, ya que las aplicaciones empaquetadas en JAR pueden ejecutarse en diferentes plataformas sin modificar el código fuente.

Características atractivas de los archivos JAR

- Un archivo JAR se basa en un archivo zip.
- Este formato de archivo puede manejar archivos de clase, audio e imagen almacenados en un solo lugar.
- El archivo JAR es el único formato de archivo multiplataforma.
- Es compatible con versiones anteriores del código del applet.
- Escrito en Java, es un estándar abierto y totalmente extensible.
- Perfecto para empaquetar piezas de un applet de Java.
- Contiene un archivo de manifiesto y potencialmente un archivo de firma. [11]

Video propio de como crear un Archivo.Jar: https://youtu.be/Se5GjbJYcCU

2.13. JAVA SWING (Intelillij):

El "GUI Designer" en IntelliJ IDEA es una herramienta que te permite crear interfaces gráficas de usuario para tus aplicaciones utilizando los elementos de la biblioteca Swing. Esta herramienta hace que sea más fácil realizar tareas comunes, como crear ventanas de diálogo y grupos de controles que se pueden utilizar en una ventana principal, como un JFrame. Al diseñar un formulario con GUI Designer, estás creando un panel en lugar de una ventana .

El GUI Designer en IntelliJ IDEA es una herramienta que facilita la creación de interfaces gráficas para aplicaciones Java utilizando componentes Swing, pero no se encarga de crear la ventana principal de la aplicación ni de los menús, y solo es compatible con componentes Swing. [12]

2.14. UBUNTU:

Ubuntu es un sistema operativo basado en Linux que destaca por ser gratuito y de código abierto. Está diseñado para su uso en computadoras de escritorio, desarrollo de software, servidores, entornos educativos y nubes. Se diferencia de Linux, que es el núcleo del sistema, ya que Ubuntu es una distribución específica de Linux que incluye un conjunto de aplicaciones y un entorno de escritorio. Su objetivo principal es proporcionar un sistema de fácil uso y promover el software libre. En resumen, Ubuntu es una opción versátil de sistema operativo con múltiples aplicaciones y usos. [13]

2.15. MÁQUINAS VIRTUALES:

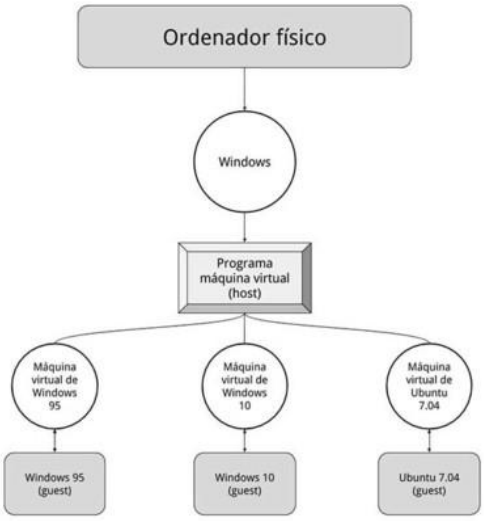


Imagen 3. Maquinas virtuales de sistema. Extraído de [https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas]

Las máquinas virtuales son software capaces de emular o imitar a un ordenador físico. Su función principal es permitir la ejecución de un sistema operativo completo en un entorno virtual. Estas máquinas virtuales se dividen en dos tipos principales: las de sistema y las de proceso. Las máquinas virtuales de sistema emulan un ordenador completo con sus propios recursos virtuales, como disco duro, memoria y tarjeta gráfica.

Las máquinas virtuales permiten ejecutar sistemas operativos y aplicaciones en un entorno aislado, lo que significa que el sistema operativo y las aplicaciones que se ejecutan en una máquina virtual no pueden acceder directamente a los recursos del sistema anfitrión (el sistema físico en el que se encuentra la máquina virtual). Esto proporciona una forma segura de ejecutar diferentes sistemas operativos y aplicaciones en una misma máquina física. [14]

3. Objetivos

3.1. Objetivo General

Desarrollar un sistema de gestión de pedidos a domicilio para el restaurante FOOD UPB, utilizando la metodología en espiral mediante el lenguaje de programación Java, con el objetivo de automatizar el proceso de toma pedidos, preparación, registro y entrega del producto.

3.2. Objetivos Específicos

- Analizar los requerimientos funcionales y no funcionalidades del sistema de pedidos a domicilio para Food UPB.
- Diseñar el sistema utilizando especificaciones UML basados en los requerimientos funcionales y no funcionales
- Diseñar las interfaces de usuario, conforme a los requerimientos establecidos y aplicando conceptos de UX .
- Hacer uso de algoritmos y estructuras de datos de autoría propia para las diferentes funcionalidades de los módulos de gestión.
- Evaluar el funcionamiento del sistema asegurándose que se cumplan los requerimientos

acordados con el cliente.

- Utilizar repositorios para el control de versiones y avance de iteraciones.
- Implementar una base de datos centralizada utilizando MySQL que permita almacenar, organizar y exportar la información del sistema

4. Metodología

La metodología para el desarrollo del sistema de gestión de pedidos a domicilio es la metodología en espiral. Esta metodología se basa en una planificación detallada del proyecto, la identificación y evaluación de riesgos, un enfoque iterativo de desarrollo y una evaluación continua del sistema de gestión.

Imagen 4. Metodología espiral. Tomada de [https://www.lifeder.com/modelo-espiral/]

El sistema de esta metodología permitirá crear un sistema de gestión de pedidos a domicilio de calidad y minimizar los riesgos potenciales durante todo el proceso de creación del sistema.

A. Fase de planificación:

Se realizará una planificación detallada del sistema de pedidos a domicilio. Se definirán los objetivos del sistema, las funcionalidades que debe incluir y los requisitos necesarios para su correcto funcionamiento

Imagen 5. Necesidades del software. Imagen propia

B. Fase de análisis de riesgos:

Se identificaron los riesgos asociados al desarrollo e implementación del sistema de pedidos a domicilio. Entre los riesgos potenciales se pueden incluir problemas de seguridad de datos,

retrasos en la entrega, fallos en la integración con sistemas de pago, entre otros. Se evaluó el impacto y la probabilidad de ocurrencia de cada riesgo y se diseñaron planes de contingencia y mitigación para reducir su impacto en el proyecto.

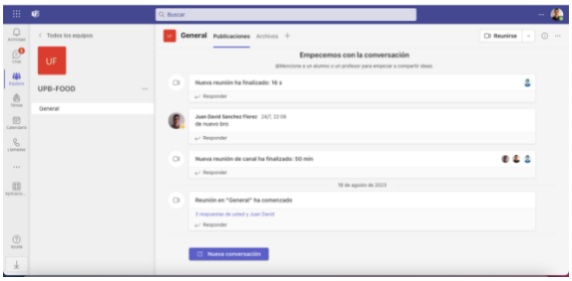


Imagen 6. Reuniones semanales. Imagen propia

C. Fase de desarrollo:

Se llevó a cabo el desarrollo del sistema de pedidos a domicilio. Se llevó a cabo un enfoque iterativo, comenzando con la definición de los requisitos del usuario y la especificación de las funcionalidades clave del sistema. Se diseñó un prototipo inicial del sistema, y se hicieron ciclos de refinamiento y mejora para asegurar la calidad del producto final.

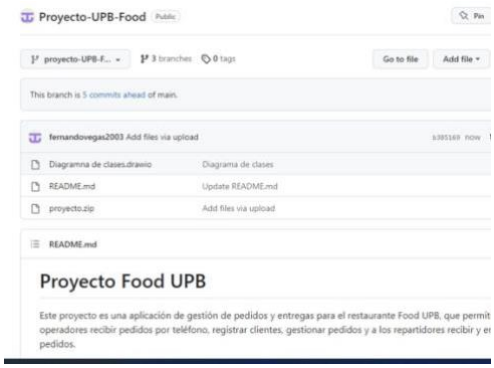


Imagen 7. Repositorio GitHub. Imagen propia.

D. Fase de evaluación:

Se llevaron a cabo pruebas y evaluaciones exhaustivas. Se probaron todas las funcionalidades del sistema para identificar problemas y oportunidades de mejora. Se implementaron las correcciones y mejoras necesarias para asegurar que el sistema funcionara de manera óptima y cumpliera con los requisitos establecidos en la fase de planificación.

5. RESULTADOS DEL PRODUCTO Y PROYECTO

Para medir el progreso del proyecto, se llevará a cabo un seguimiento y evaluación de acuerdo con los objetivos específicos previamente establecidos. Cada uno de estos objetivos representa una meta concreta que se debe lograr durante el desarrollo del proyecto.

A. RESULTADOS DE LOS OBJETIVOS ESPECÍFICOS.

1) Objetivo Específico 1: Identificación de requerimientos.

Se desarrolló un documento que contiene los requisitos funcionales y no funcionales identificados a partir del documento de definición del proyecto.(este documento será anexado en referencias).

2) Objetivo Específico 2: Diseño del sistema.

Para dar cumplimiento a este objetivo, se realizaron, en primer lugar, los diagramas de casos de uso del sistema. Los diagramas de casos de uso presentan las posibles acciones de los usuarios o roles identificados, los cuales son:

- - Administrador
- - Operador
- - Domiciliario
- - Cocina

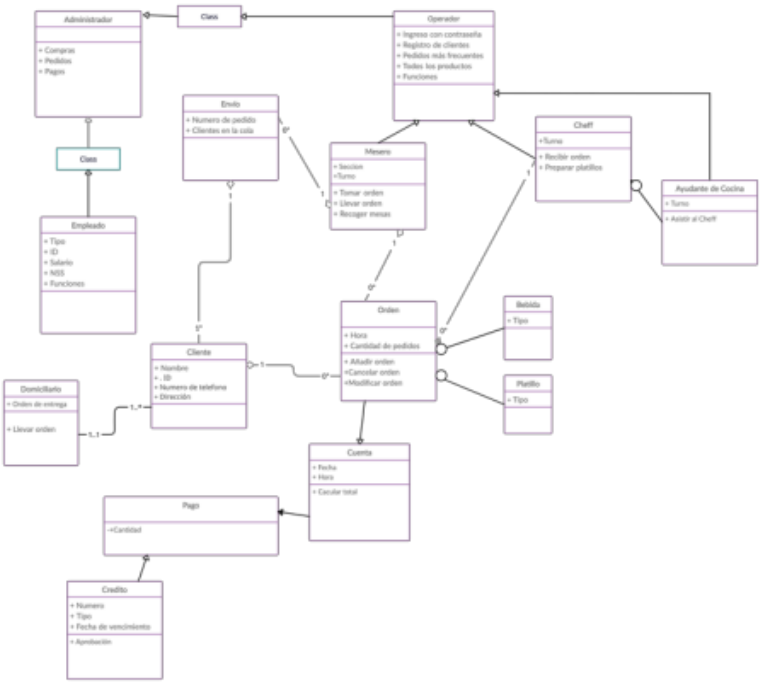


Imagen 8. Diagrama casos de uso. Imagen propia.

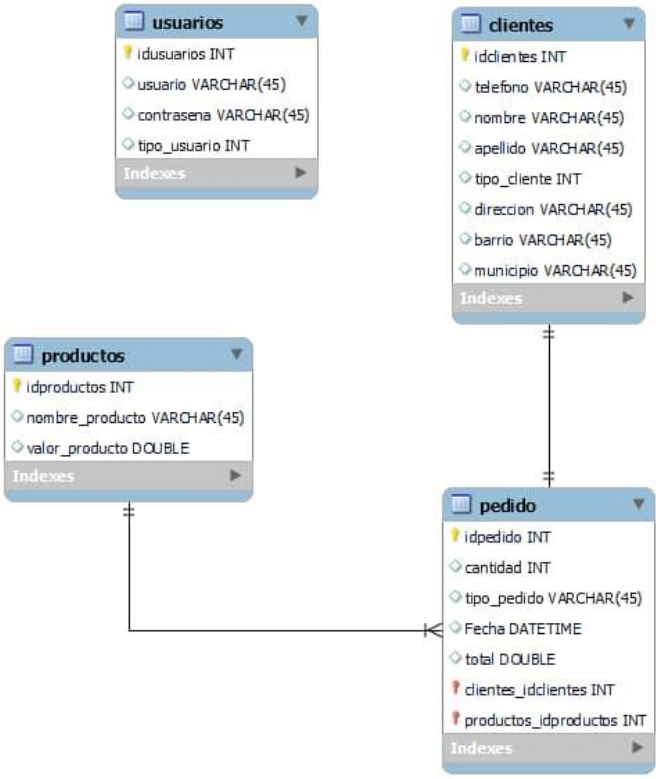


Imagen 9. Modelo de identidad relacion de la base de datos MYSQL. Imagen propia.

3) Objetivo Específico 3: Diseño de interfaces deusuario.

3.1 Modulo Admin.

Imagen 10. Vista de Usuarios. Imagen propia.

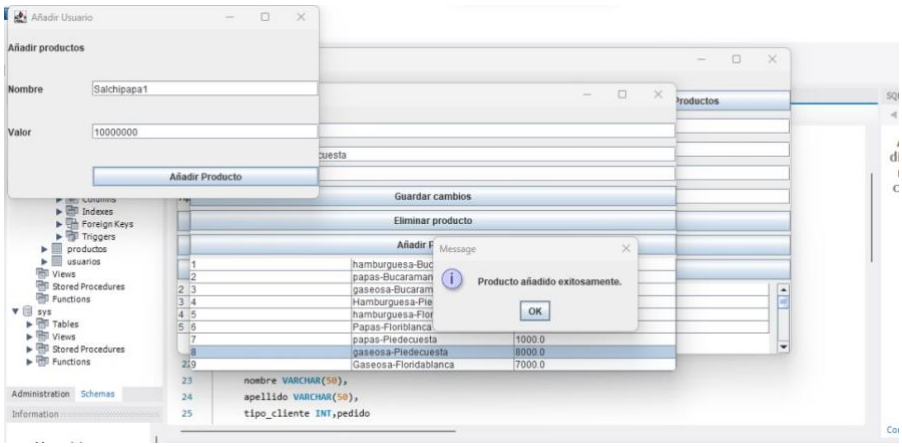


Imagen 11. Administración de Productos. Imagen propia.

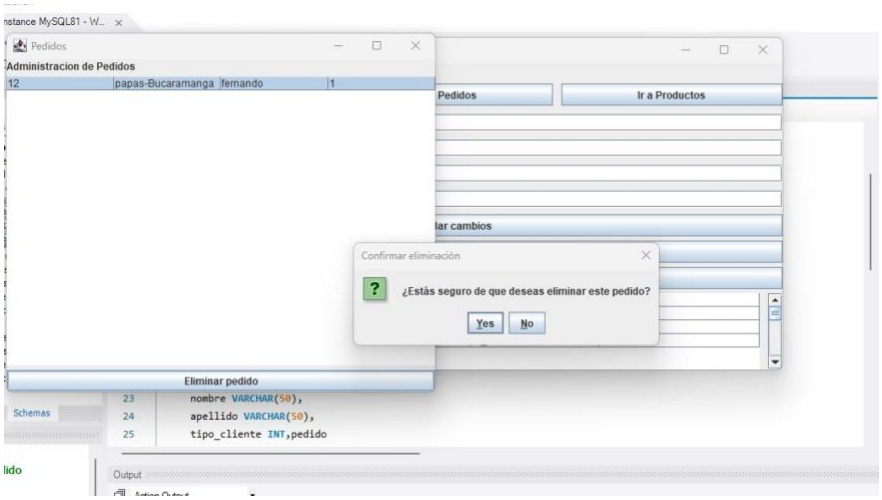


Imagen 12. Administración de pedidos. Imágen propia.

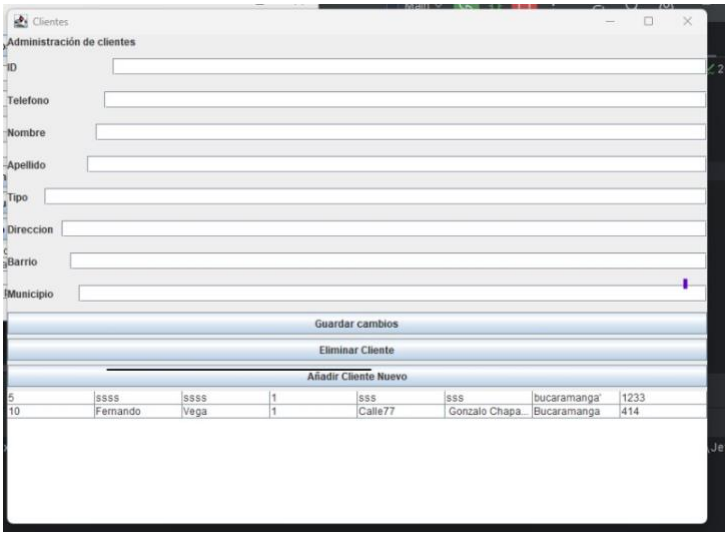


Imagen 13. Administración de clientes. Imagen propia

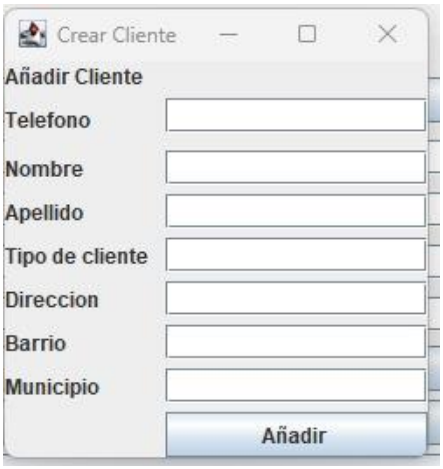


Imagen 14. Añadir clientes. Imagen propia.

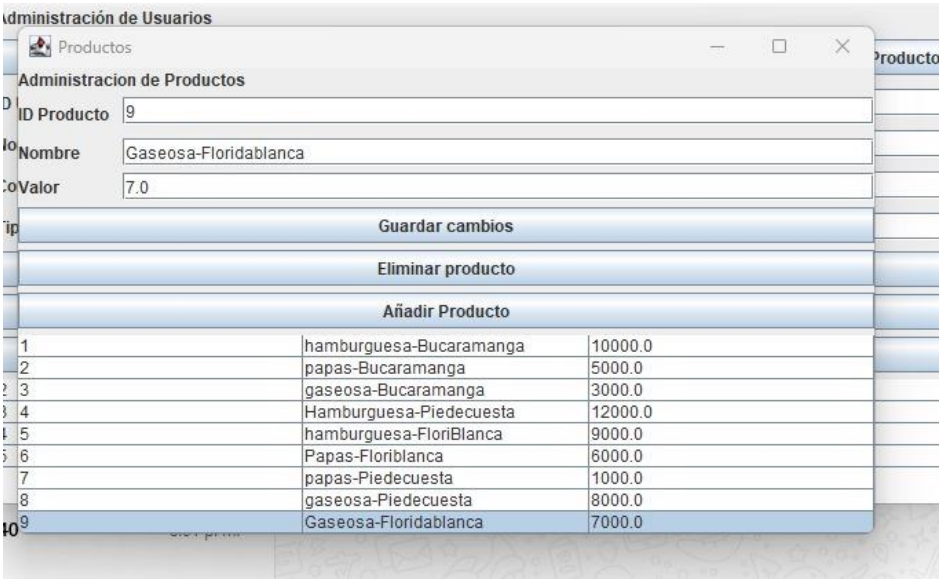


Imagen 15. Productos. Imagen propia.

3.2. Modulo Operador.

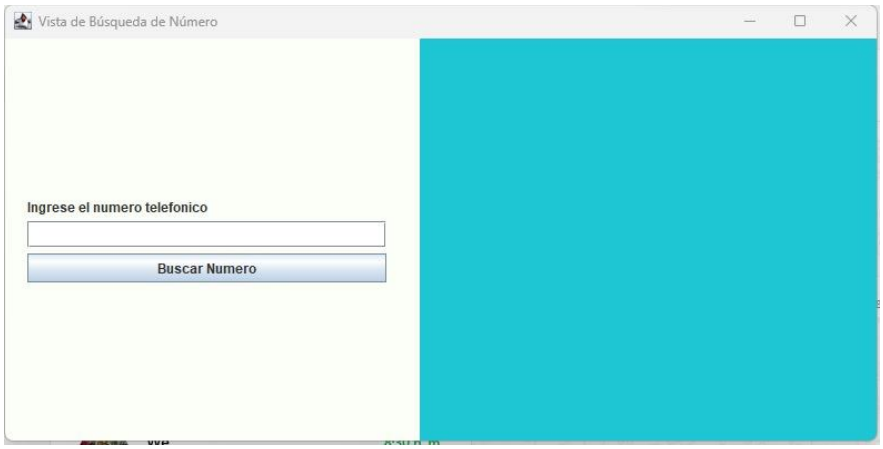


Imagen 16. Vista de busqueda de Numero telefonico. Imagen propia

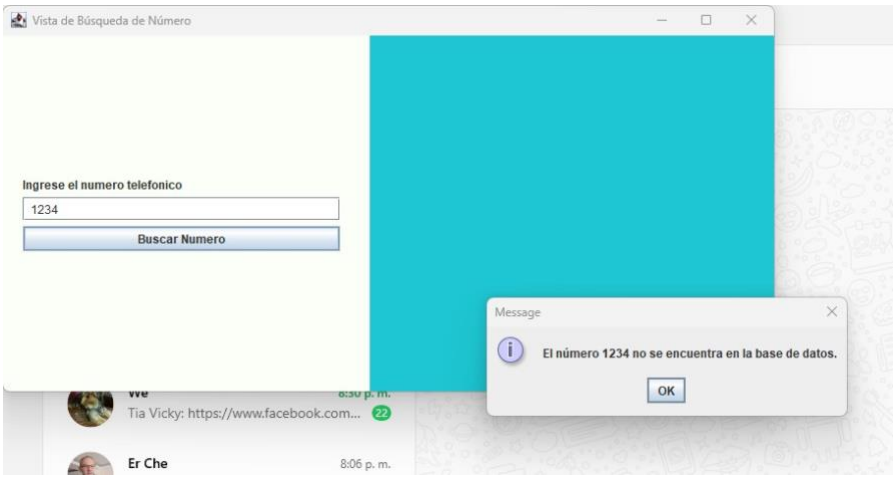


Imagen 17. Comprobación del número en la base de datos. Imagen propia

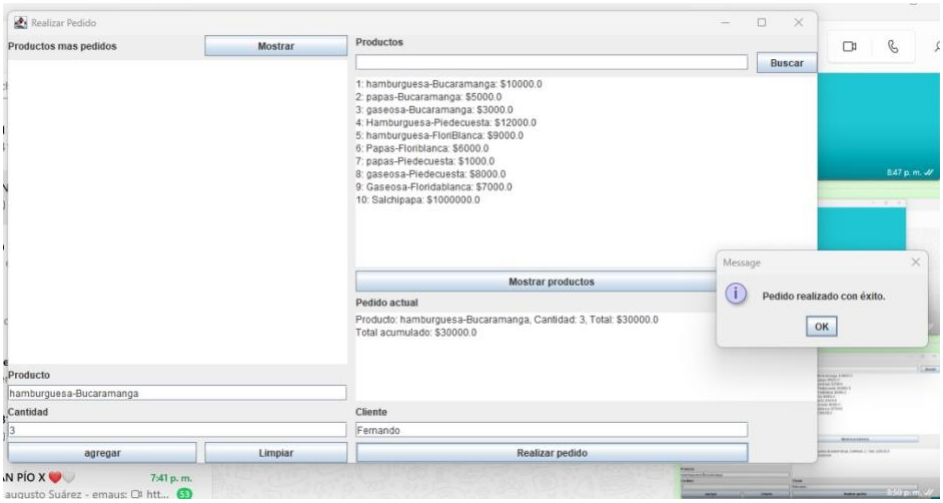


Imagen 18. Realizar pedido. Imagen propia.

3.3 Modulo domicilio.

Imagen 19. Vista de Domicilio. Imagen propia

3.4 Modulo cocina.

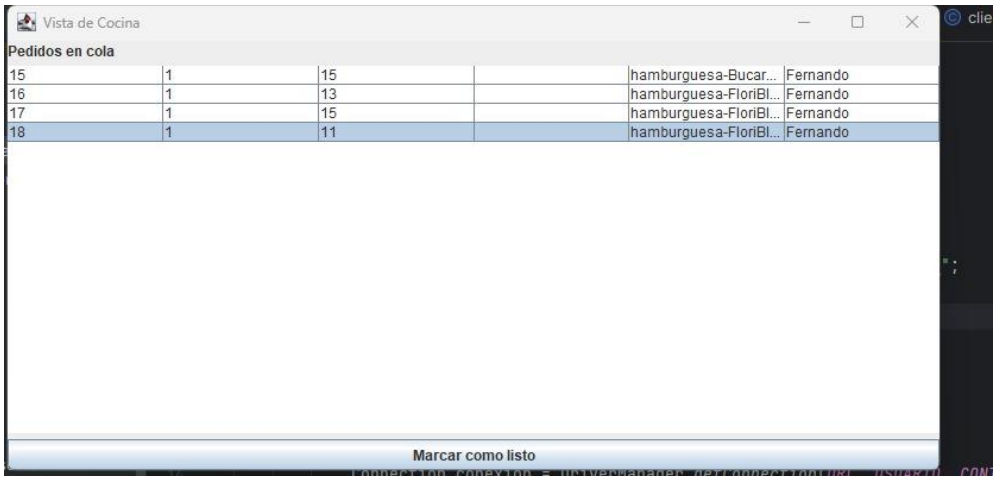
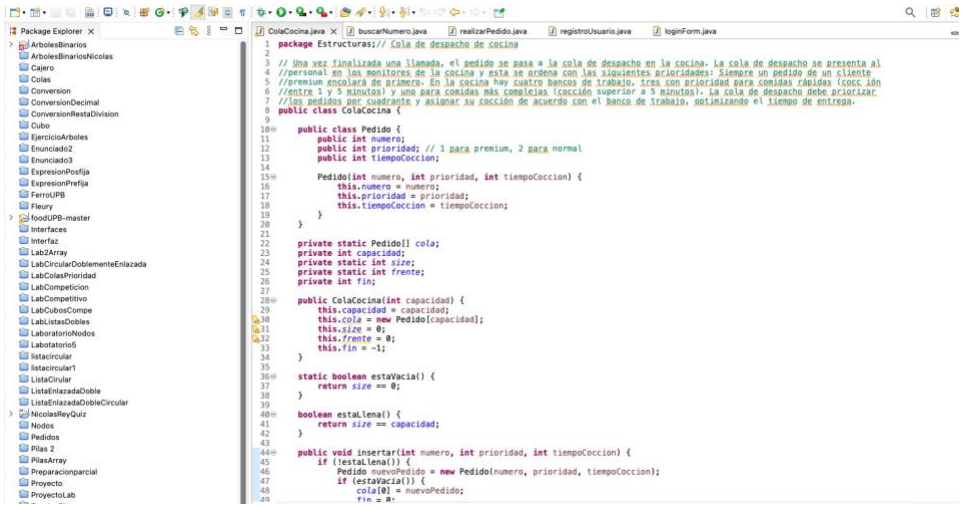


Imagen 20. Vista de cocina. Imagen propia.

Para el cumplimiento de este objetivo se diseñaron modelos con ayuda de la herramienta Adobe XD para la interfaz de usuario del restaurante con algunas opciones a implementar y fueron implementados en el lenguaje de programación Java.

4) Objetivo específico 4: Uso de estructuras de datos.



```
package Estructuras;// Cola de despacho de cocina

// Una vez finalizada una llamada, el pedido se pasa a la cola de despacho en la cocina. La cola de despacho se presenta al
//personal en los monitores de la cocina y esta se ordena con las siguientes prioridades: Siempre un pedido de un cliente
//premium encolará de primero. En la cocina hay cuatro bancos de trabajo, tres con prioridad para comidas rápidas (cocción
//entre 1 y 5 minutos) y uno para comidas más complejas (cocción superior a 5 minutos). La cola de despacho debe priorizar
//los pedidos por cuadrante y asignar su cocción de acuerdo con el banco de trabajo, optimizando el tiempo de entrega.
public class ColaCocina {

    public class Pedido {
        public int numero;
        public int prioridad; // 1 para premium, 2 para normal
        public int tiempoCoccion;

        Pedido(int numero, int prioridad, int tiempoCoccion) {
            this.numero = numero;
            this.prioridad = prioridad;
            this.tiempoCoccion = tiempoCoccion;
        }
    }

    private static Pedido[] cola;
    private int capacidad;
    private static int size;
    private static int frente;
    private int fin;

    public ColaCocina(int capacidad) {
        this.capacidad = capacidad;
        this.cola = new Pedido[capacidad];
        this.size = 0;
        this.frente = 0;
        this.fin = -1;
    }

    static boolean estaVacia() {
        return size == 0;
    }

    boolean estaLlena() {
        return size == capacidad;
    }

    public void insertar(int numero, int prioridad, int tiempoCoccion) {
        if (!estaLlena()) {
            Pedido nuevoPedido = new Pedido(numero, prioridad, tiempoCoccion);
            if (estaVacia()) {
                cola[0] = nuevoPedido;
```

Imagen 21. Estructura de datos, cola despacho de cocina. Imagen propia.

Imagen 22. Cola de pedidos. Imagen propia.

Se implementó una cola de pedidos con prioridad para un restaurante utilizando estructuras de datos. Esta cola permite gestionar los pedidos de los clientes de manera eficiente, asignando prioridades a los pedidos de clientes premium. Esto garantiza que los pedidos urgentes se procesen antes que los de menor importancia, mejorando la eficiencia en la entrega de los alimentos y la satisfacción del cliente.

5) Objetivo específico 5: *Funcionamiento del sistema*

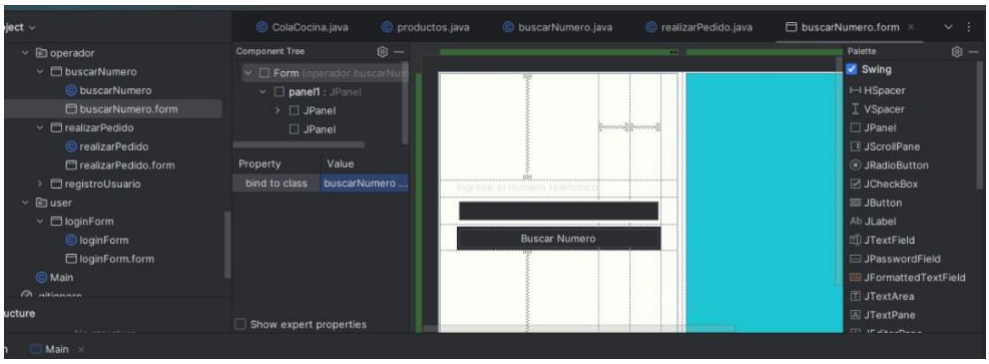


Imagen 23. Form Operador. Imagen propia.

Se ha desarrollado el codigo basado en Java Swing en IntelliJ IDEA, donde cada módulo cumple una función específica. El módulo de Operador permite tomar pedidos y gestionamiento del restaurante, el modulo Cocina muestra los pedidos entrantes y su estado de preparación, el de Domicilio administra entregas a domicilio y el módulo de Administrador proporciona un panel de control con información clave para la gestión del restaurante. Cada módulo tiene su propia interfaz gráfica diseñada con Java Swing para facilitar su funcionamiento y mejorar la experiencia del usuario.

6) Objetivo Especifico 6: Utilizar Repositorios

Se hace uso de un repositorio GITHUB creando el proyecto en elsistema y organizando las actividades a realizar a partir de su importancia y necesidad para el desarrollo del software

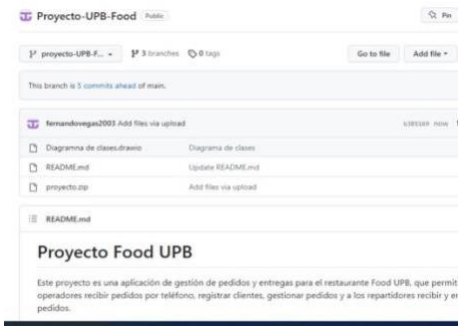


Imagen 24. Repositorio GitHub. Imagen propia

Se han incorporado historias de usuarios que permiten cumplir con los requisitos establecidos en la bitácora de desarrollo. Esto no solo facilita la asignación de tareas, sino que también proporciona una visión general de las metas semanales, permitiendo un seguimiento preciso del progreso y la capacidad de realizar ajustes según sea necesario. El uso de GitHub y la integración de historias de usuarios son prácticas esenciales para garantizar una gestión efectiva del proyecto, lo que resulta en un desarrollo de software más eficiente y satisfactorio.

Se propuso la implementación de una base de datos como parte integral de las operaciones diarias. Esta base de datos sirvió como un sistema centralizado para almacenar y gestionar todos los aspectos relacionados con el restaurante, desde la gestión de pedidos de administrador y de administrador hasta el sistema de domicilio.

La propuesta de implementar una base de datos en MySQL como parte integral de las operaciones diarias de un restaurante marcó un avance significativo en la eficiencia y la gestión de este establecimiento. MySQL, como un sistema de gestión de bases de datos de código abierto ampliamente utilizado, se convirtió en la columna vertebral de la organización y el almacenamiento centralizado de información.

Esta base de datos MySQL permitió al restaurante administrar de manera efectiva la información crucial, desde la gestión de pedidos, la contabilidad y la administración de recursos, hasta la coordinación del sistema de entrega a domicilio.

7) Objetivo específico 7: Bases de datos

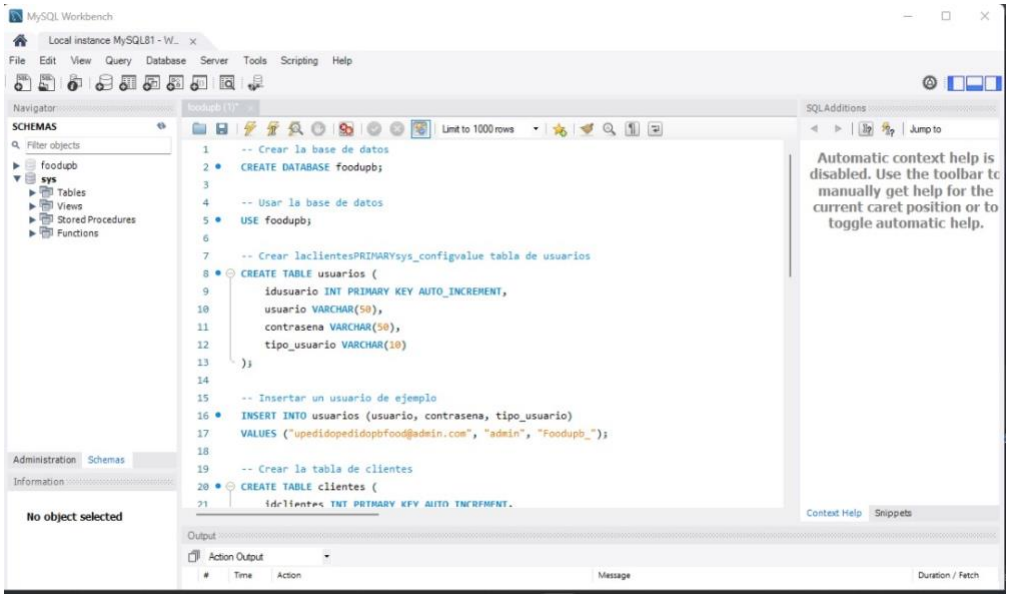


Imagen 25. Bases de datos MYSQL. Imagen propia

Se propuso la implementación de una base de datos como parte integral de las operaciones diarias. Esta base de datos sirvió como un sistema centralizado para almacenar y gestionar todos los aspectos relacionados con el restaurante, desde la gestión de pedidos de administrador y de administrador hasta el sistema de domicilio.

La propuesta de implementar una base de datos en MySQL como parte integral de las operaciones diarias de un restaurante marcó un avance significativo en la eficiencia y la gestión de este establecimiento. MySQL, como un sistema de gestión de bases de datos de código abierto ampliamente utilizado, se convirtió en la columna vertebral de la organización y el almacenamiento centralizado de información.

Esta base de datos MySQL permitió al restaurante administrar de manera efectiva la información crucial, desde la gestión de pedidos, la contabilidad y la administración de recursos, hasta la coordinación del sistema de entrega a domicilio.

VIDEO EXPLICACION DE LA BASE DE DATOS: https://youtu.be/aOGMPri0VEA

6. RESULTADOS DEL PROYECTO

a) Modulo operador

Este código permite a un operador interactuar con una base de datos para buscar clientes, registrar nuevos clientes y realizar pedidos de productos. También maneja la interfaz gráfica de la aplicación utilizando la biblioteca Swing en Java.

El código del módulo operador de una aplicación Java se encarga de gestionar la interacción con una base de datos para realizar pedidos y registrar clientes.

1. buscarNumero (Clase para buscar números de teléfono):

- Permite al operador buscar un número de teléfono en la base de datos de clientes.
- Si se encuentra el número, muestra el nombre del cliente y abre la ventana de ***realizarPedido.***
- Si no se encuentra el número, muestra un mensaje de error.

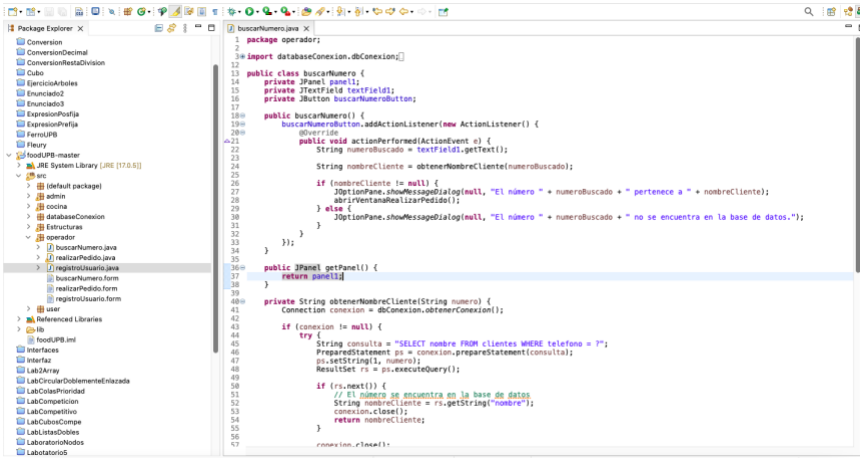


Imagen 26. Buscar número de telefono. Imagen propia

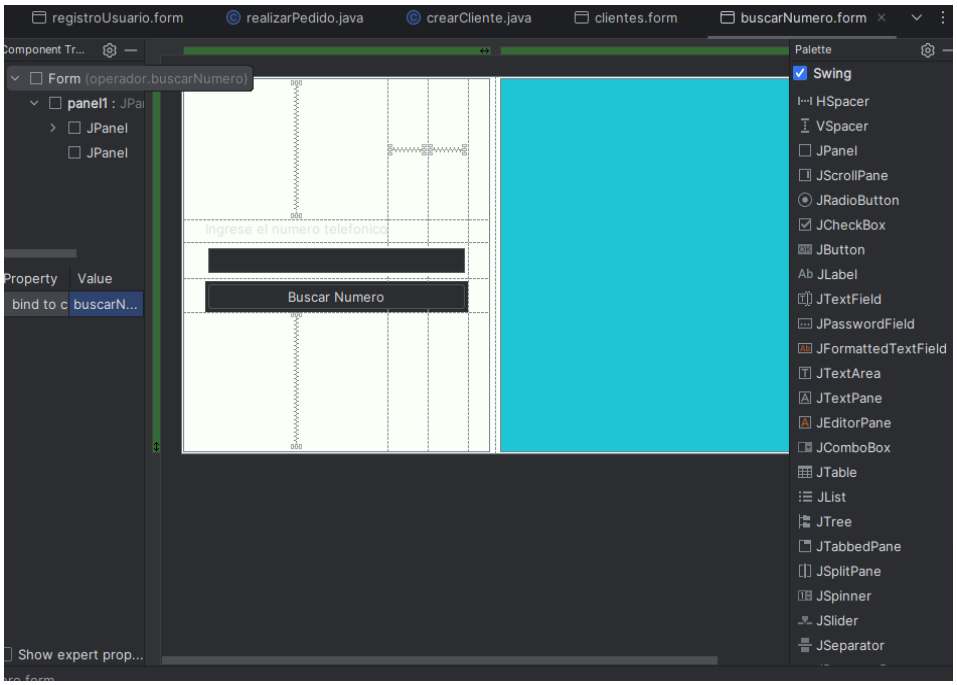


Imagen 27. Buscar Número de telefono Form. Imagen propia

2. realizarPedido (Clase para realizar pedidos):

- Carga la lista de productos desde la base de datos y permite buscar productos por nombre.
- Permite agregar productos a un pedido, mostrando el total acumulado.
- Al realizar un pedido, se registra en la base de datos la información del cliente y los productos seleccionados.
- Permite limpiar la lista de productos agregados y el total acumulado.
- Muestra mensajes de error en caso de problemas con la base de datos o campos vacíos.
- Contiene una clase interna ***ProductoPedido*** para representar productos en el pedido.

Imagen 28. Realizar pedido. Imagen propia.

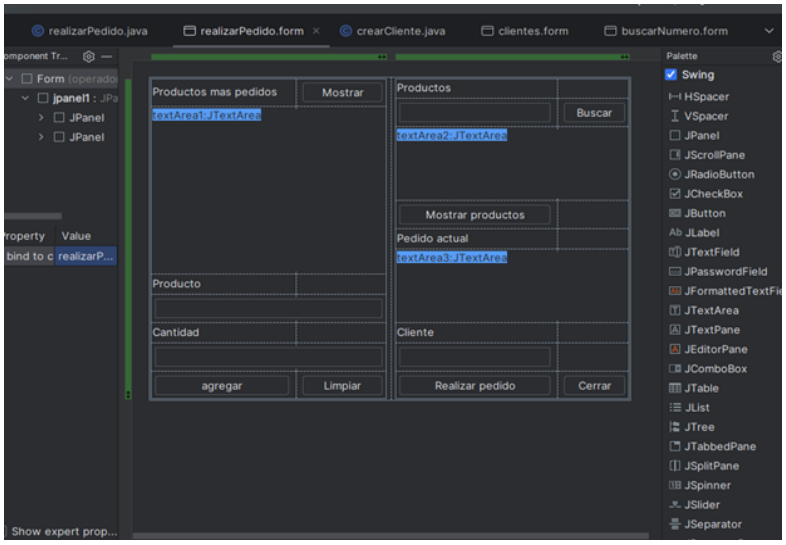
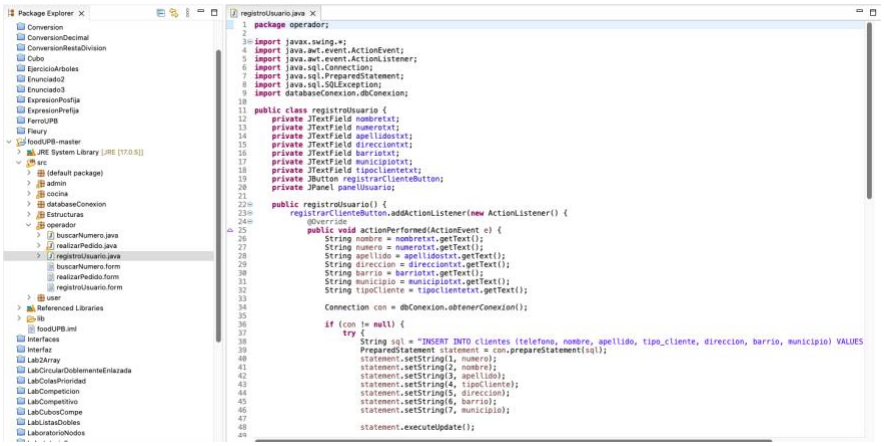


Imagen 29. Realizar pedido Form. Imagen propia.

3. registroUsuario (Clase para registrar clientes):

Permite al operador registrar nuevos clientes en la base de datos.

- Los detalles del cliente, como nombre, número de teléfono, dirección, etc., se ingresan en campos de texto.
- Al registrarse, se valida y se inserta la información del cliente en la base de datos.
- Después de registrar al cliente, redirige a la ventana de ***realizarPedido***.



```
package operador;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import databaseConexion.dbConexion;

public class registroUsuario {
    private JTextField nombretxt;
    private JTextField numerotxt;
    private JTextField apellidostxt;
    private JTextField direcciontxt;
    private JTextField barriotxt;
    private JTextField municipiotxt;
    private JTextField tipoclientetxt;
    private JButton registrarClienteButton;
    private JPanel panelUsuario;

    public registroUsuario() {
        registrarClienteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String nombre = nombretxt.getText();
                String numero = numerotxt.getText();
                String apellido = apellidostxt.getText();
                String direccion = direcciontxt.getText();
                String barrio = barriotxt.getText();
                String municipio = municipiotxt.getText();
                String tipoCliente = tipoclientetxt.getText();

                Connection con = dbConexion.obtenerConexion();

                if (con != null) {
                    try {
                        String sql = "INSERT INTO clientes (telefono, nombre, apellido, tipo_cliente, direccion, barrio, municipio) VALUES
                        PreparedStatement statement = con.prepareStatement(sql);
                        statement.setString(1, numero);
                        statement.setString(2, nombre);
                        statement.setString(3, apellido);
                        statement.setString(4, tipoCliente);
                        statement.setString(5, direccion);
                        statement.setString(6, barrio);
                        statement.setString(7, municipio);

                        statement.executeUpdate();
```

Imagen 30. Registro de usuario. Imagen propia.

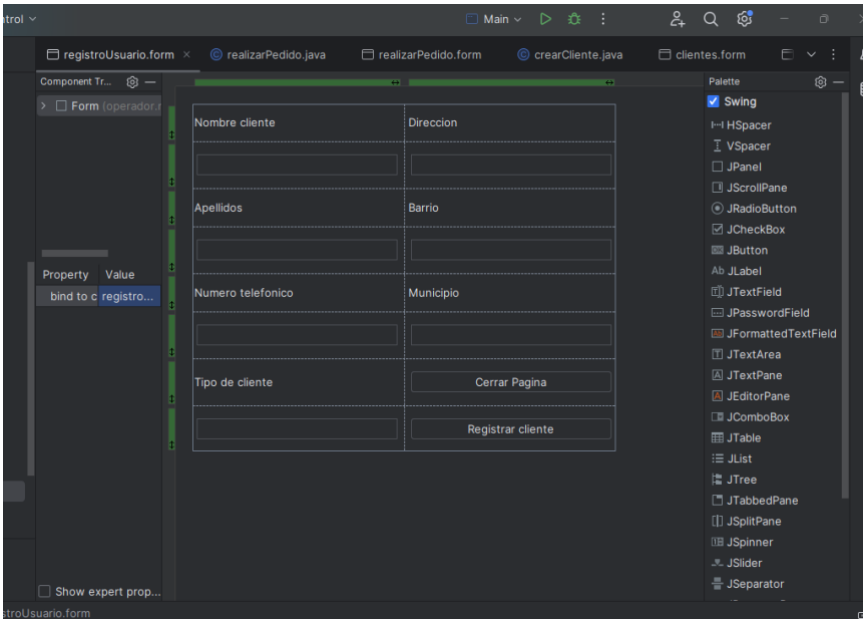


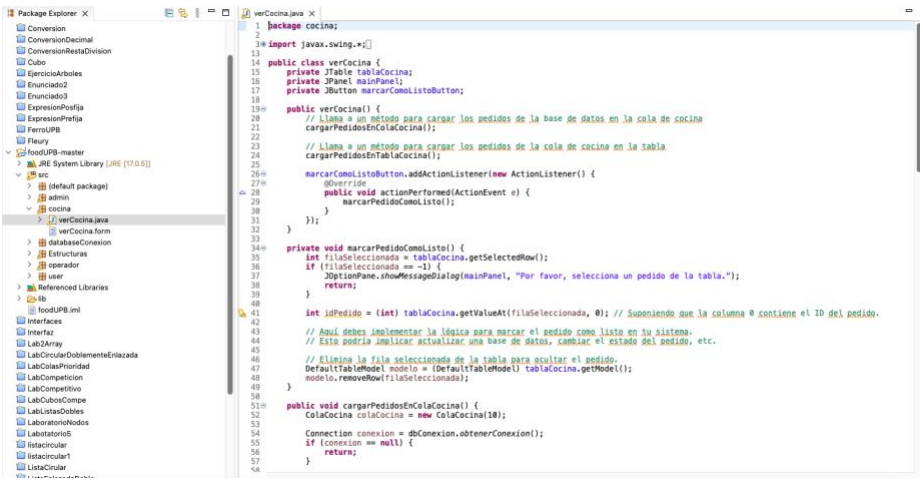
Imagen 31 . Registro de usuario Form. Imagen propia

b) Modulo cocina.

Se utiliza en el módulo de cocina de una aplicación de restaurante para visualizar y gestionar los pedidos pendientes. Carga los pedidos desde la base de datos en una cola de cocina, muestra los detalles de los pedidos en una tabla y permite a los cocineros marcar los pedidos como listos cuando están preparados

1. verCocina (Clase para ver los pedidos en la cocina):

- Esta clase se encarga de mostrar una interfaz gráfica que permite a los chefs y cocineros ver los pedidos pendientes en la cocina.
- Carga los pedidos de la base de datos en una cola de cocina.
- Muestra los detalles de los pedidos, como el ID del pedido, prioridad, tiempo de cocción, tipo de pedido, producto y cliente en una tabla.
- Permite a los cocineros marcar un pedido como listo una vez que se ha preparado.
- La función ***cargarPedidosEnColaCocina()*** carga los pedidos de la base de datos en una cola de cocina con prioridades según el tipo de cliente y el tiempo de cocción.
- La función ***cargarPedidosEnTablaCocina()*** muestra los detalles de los pedidos en una tabla, incluyendo información de productos y clientes.



```java
package cocina;

import javax.swing.*;

public class verCocina {
    private JTable tablaCocina;
    private JPanel mainPanel;
    private JButton marcarComoListoButton;

    public verCocina() {
        // Llama a un método para cargar los pedidos de la base de datos en la cola de cocina
        cargarPedidosEnColaCocina();

        // Llama a un método para cargar los pedidos de la cola de cocina en la tabla
        cargarPedidosEnTablaCocina();

        marcarComoListoButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                marcarPedidoComoListo();
            }
        });
    }

    private void marcarPedidoComoListo() {
        int filaSeleccionada = tablaCocina.getSelectedRow();
        if (filaSeleccionada == -1) {
            JOptionPane.showMessageDialog(mainPanel, "Por favor, selecciona un pedido de la tabla.");
            return;
        }

        int idPedido = (int) tablaCocina.getValueAt(filaSeleccionada, 0); // Suponiendo que la columna 0 contiene el ID del pedido.

        // Aquí debes implementar la lógica para marcar el pedido como listo en tu sistema.
        // Esto podría implicar actualizar una base de datos, cambiar el estado del pedido, etc.

        // Elimina la fila seleccionada de la tabla para ocultar el pedido.
        DefaultTableModel modelo = (DefaultTableModel) tablaCocina.getModel();
        modelo.removeRow(filaSeleccionada);
    }

    public void cargarPedidosEnColaCocina() {
        ColaCocina colaCocina = new ColaCocina(10);

        Connection conexion = dbConexion.obtenerConexion();
        if (conexion == null) {
            return;
        }
```

Imagen 32. Ver cocina. Imagen propia

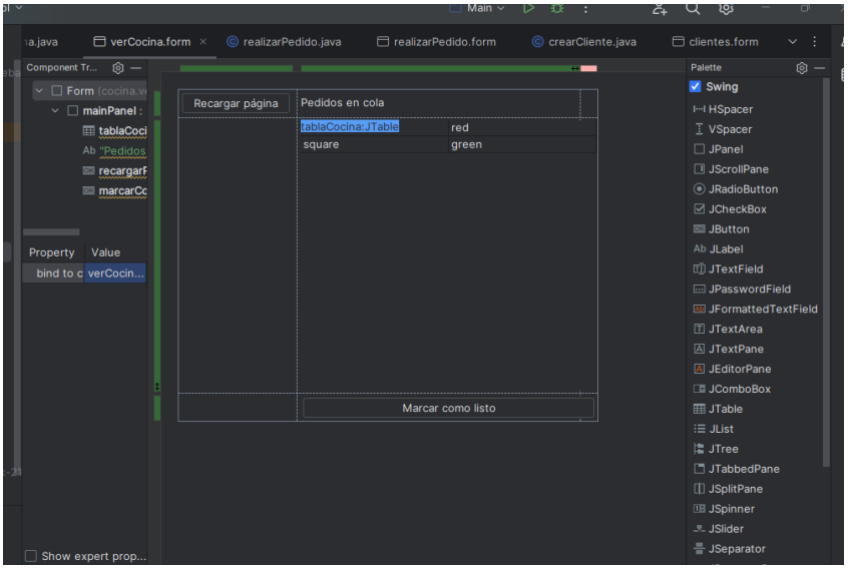


Imagen 33. Ver Cocina Form. Imagen propia.

c) Modulo administrador

Se proporciona una interfaz para administrar datos de clientes en una aplicación de administración. Permite ver, editar y eliminar clientes, así como agregar nuevos clientes. Los datos se almacenan y recuperan desde una base de datos.

1. Clientes:

1.1. Interfaz de Administración de Clientes:

- La clase `clientes` se encarga de la administración de clientes, lo que incluye la visualización de una lista de clientes, la edición de sus datos y la eliminación de clientes.

1.2. Carga de Datos en la Tabla:

- El método `cargarDatosEnJTable()` se utiliza para cargar los datos de los clientes desde

una base de datos en una tabla (JTable) en la interfaz gráfica.
- Se crea un modelo de tabla que incluye columnas para el ID, nombre, apellido, tipo de cliente, dirección, barrio, municipio y teléfono de los clientes.
- Los datos se obtienen de la base de datos y se muestran en la tabla.

1.3. Selección de Clientes en la Tabla:

- Se utiliza un ListSelectionListener para detectar la selección de un cliente en la tabla.
- Cuando un cliente se selecciona, sus datos se muestran en campos de texto en la interfaz gráfica, lo que permite su edición.

1.4. Edición de Clientes:

- El botón "Guardar Cambios" permite guardar los cambios realizados en los datos de un cliente seleccionado en la base de datos.
- Se obtienen los datos editados de los campos de texto y se actualiza el registro del cliente en la base de datos.

1.5. Eliminación de Clientes:

- El botón "Eliminar Cliente" permite eliminar un cliente seleccionado.
- Se muestra un mensaje de confirmación antes de realizar la eliminación.
- Se elimina el cliente de la base de datos si se confirma la eliminación.

1.6. Creación de Cliente Nuevo:

- El botón "Añadir Cliente Nuevo" abre una ventana de creación de un nuevo cliente utilizando la clase `crearCliente`.
- La nueva ventana permite ingresar datos para crear un cliente nuevo.

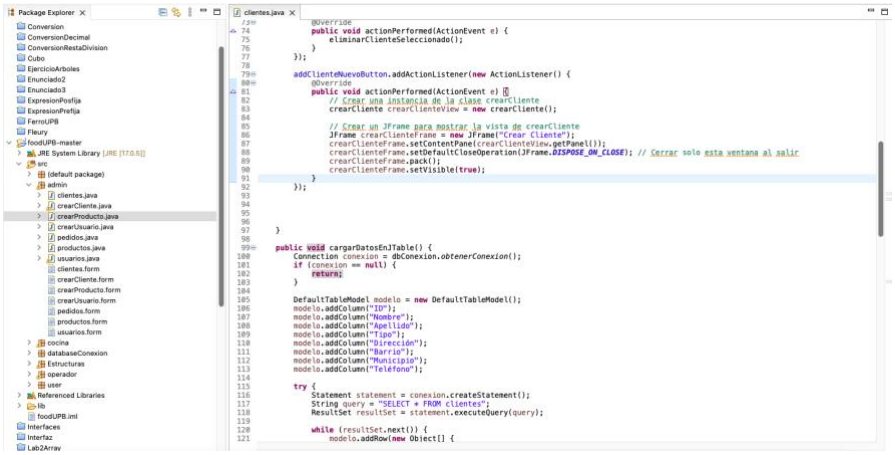


Imagen 34. Clientes. Imagen propia.

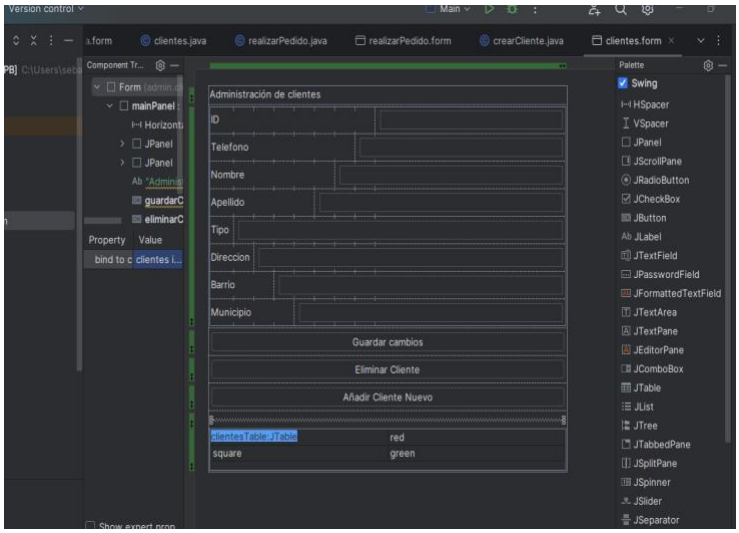


Imagen 35. Clientes Form. Imagen propia.

2. Crear Cliente

Este código proporciona una interfaz gráfica para ingresar datos de un nuevo cliente y agregarlo a la base de datos. Permite la inserción de información del cliente, incluyendo su nombre, dirección, tipo de cliente, etc. Los datos ingresados se utilizan para construir y ejecutar una consulta SQL de inserción en la base de datos.

La clase `crearCliente` se encarga de la creación de un nuevo cliente, lo que incluye la recopilación de datos del cliente a través de campos de entrada de texto y la inserción de estos datos en la base de datos.

2.1. Recopilación de Datos del Nuevo Cliente:

- En la interfaz gráfica, hay campos de texto para ingresar los siguientes datos del nuevo cliente:
  - Teléfono
  - Nombre
  - Apellido
  - Tipo de cliente
  - Dirección
  - Barrio
  - Municipio

2.2. Añadir Cliente a la Base de Datos:

- Cuando se presiona el botón "Añadir Cliente," se activa el evento `actionPerformed`, y se llama al método `addClienteABaseDeDatos()`.

2.3. Inserción en la Base de Datos:

- El método `addClienteABaseDeDatos()` se encarga de insertar los datos del nuevo cliente en la base de datos.
- Se obtienen los datos ingresados en los campos de texto y se utilizan para construir una consulta SQL de inserción.
- La consulta SQL se ejecuta en la base de datos para agregar un nuevo cliente.
- Se muestra un mensaje de éxito o error según el resultado de la inserción.

2.4. Limpieza de Campos de Texto:

- Después de agregar exitosamente un cliente, los campos de texto se pueden limpiar para permitir la entrada de datos de un nuevo cliente.

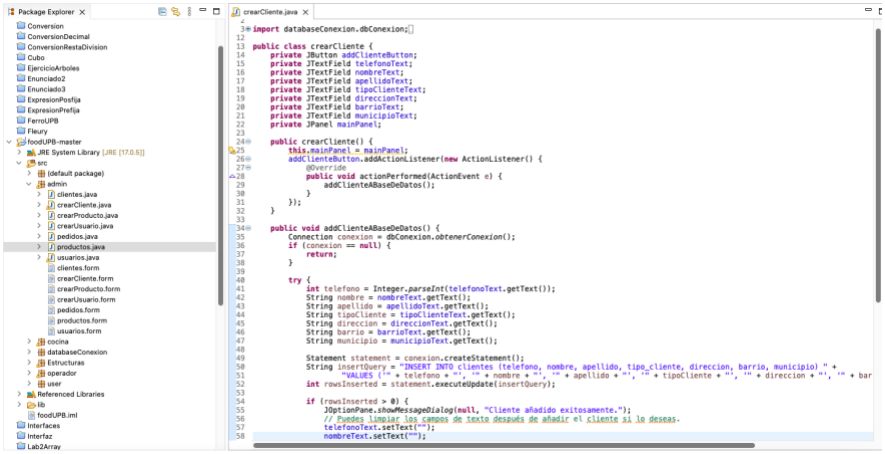


Imagen 36. Crear cliente. Imagen propia.

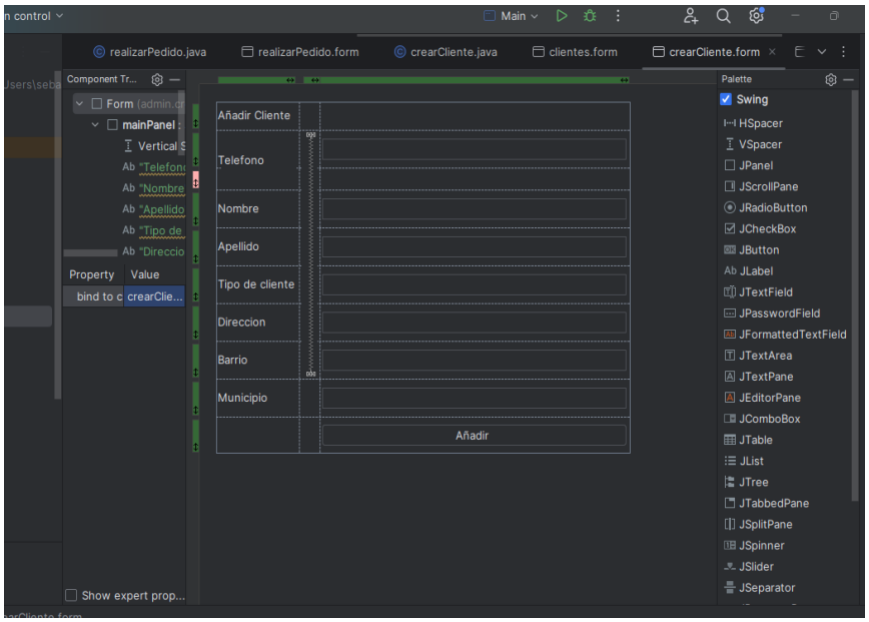


Imagen 37. Crear cliente Form. Imagen propia.

3. Crear Producto

Este código proporciona una interfaz gráfica para ingresar datos de un nuevo producto y agregarlo a la base de datos. Los datos ingresados se utilizan para construir y ejecutar una consulta SQL de inserción en la base de datos.

La clase `crearProducto` se encarga de crear un nuevo producto, lo que incluye la recopilación de datos del producto a través de campos de entrada de texto y la inserción de estos datos en la base de datos.

3.1. Recopilación de Datos del Nuevo Producto:

- En la interfaz gráfica, hay campos de texto para ingresar los siguientes datos del nuevo producto:
  - Nombre del producto.

- Valor del producto.

3.2. Añadir Producto a la Base de Datos:

- Cuando se presiona el botón "Añadir Producto," se activa el evento `actionPerformed`, y se llama al método `addProductoABaseDeDatos()`.

3.3. Inserción en la Base de Datos:

- El método `addProductoABaseDeDatos()` se encarga de insertar los datos del nuevo producto en la base de datos.
- Se obtienen los datos ingresados en los campos de texto y se utilizan para construir una consulta SQL de inserción.
- La consulta SQL se ejecuta en la base de datos para agregar un nuevo producto.
- Se muestra un mensaje de éxito o error según el resultado de la inserción.

3.4. Limpieza de Campos de Texto:

Después de agregar exitosamente un producto, los campos de texto se pueden limpiar para permitir la entrada de datos de un nuevo producto.

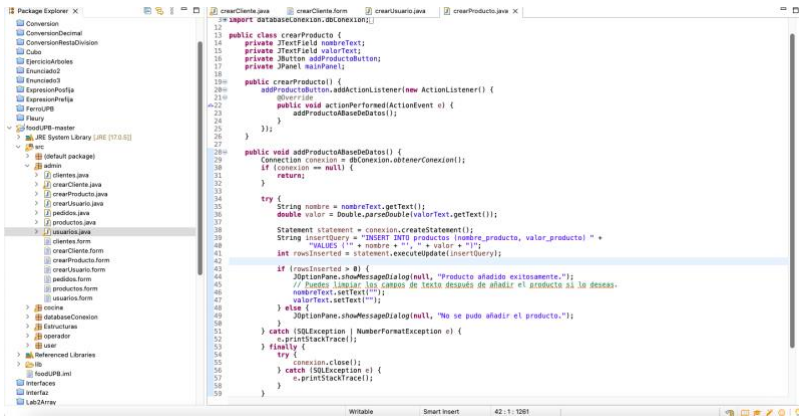


Imagen 38. Crear Producto. Imagen propia.

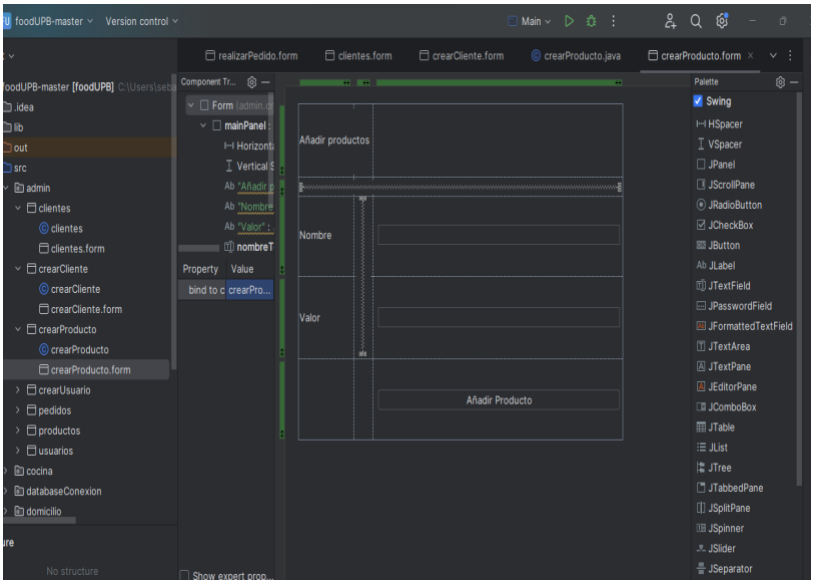


Imagen 39. Crear producto Form. Imagen propia.

4. Crear Usuario.

Este código proporciona una interfaz gráfica para ingresar datos de un nuevo usuario y

agregarlo a la base de datos. Los datos ingresados se utilizan para construir y ejecutar una consulta SQL de inserción en la base de datos.

La clase `crearUsuario` se encarga de crear un nuevo usuario, lo que implica recopilar los datos del usuario a través de campos de entrada de texto y la inserción de estos datos en la base de datos.

4.1. Recopilación de Datos del Nuevo Usuario:

- En la interfaz gráfica, hay campos de texto para ingresar los siguientes datos del nuevo usuario:
  - Nombre de usuario.
  - Tipo de usuario.
  - Contraseña.

4.2. Agregar Usuario a la Base de Datos:

- Cuando se presiona el botón "Añadir", se activa el evento `actionPerformed`, y se llama al método `añadirUsuarioABaseDeDatos()`.

4.3. Inserción en la Base de Datos:

- El método `añadirUsuarioABaseDeDatos()` se encarga de insertar los datos del nuevo usuario en la base de datos.
- Se obtienen los datos ingresados en los campos de texto y se utilizan para construir una consulta SQL de inserción.
- La consulta SQL se ejecuta en la base de datos para agregar un nuevo usuario.
- Se muestra un mensaje de éxito o error según el resultado de la inserción.

4.4. Limpieza de Campos de Texto:

- Después de agregar exitosamente un usuario, los campos de texto se pueden limpiar para permitir la entrada de datos de un nuevo usuario.

4.5. Interfaz Gráfica Principal:

- El método `main` se utiliza para crear una ventana de la interfaz gráfica. Cuando ejecutas esta clase, se mostrará una ventana que permite agregar usuarios a la base de datos.

Imagen 40. Crear Usuario. Imagen propia.

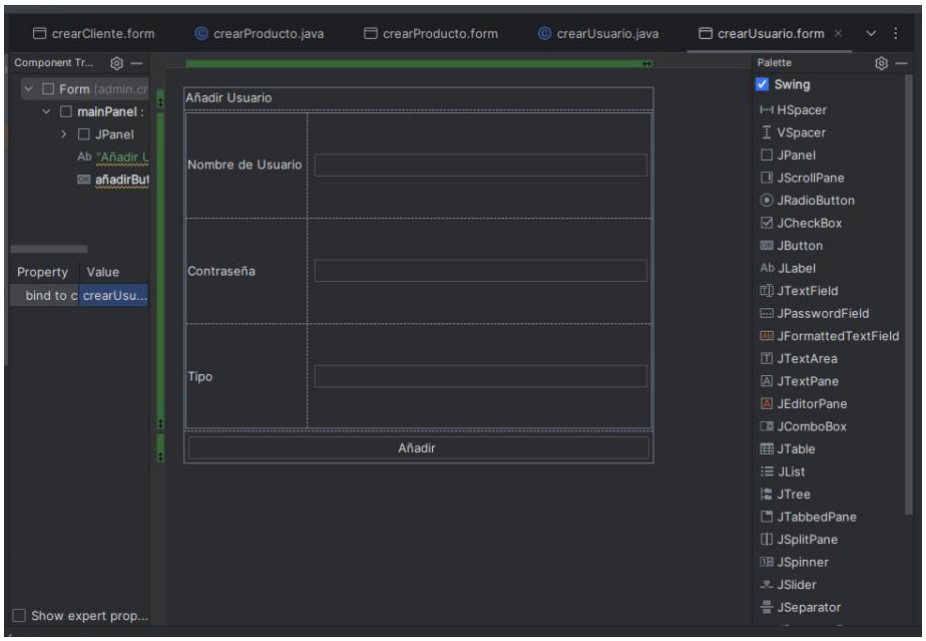


Imagen 41. Crear Usuario Form. Imagen propia.

5 Pedidos.

La clase `pedidos` crea una interfaz para mostrar y gestionar pedidos.
Incluye una tabla para mostrar información de los pedidos y un botón para eliminar pedidos seleccionados.

Esta clase proporciona una interfaz para ver y eliminar pedidos de una base de datos. Los datos se muestran en una tabla y se pueden eliminar pedidos seleccionados.

5.1. Cargar Datos en la Tabla:

- El método `cargarDatosEnJTable()` se encarga de cargar los datos de los pedidos desde la base de datos y mostrarlos en la tabla.
- Se realiza una consulta SQL para obtener información de los pedidos, productos y clientes.
- Los datos se insertan en un `DefaultTableModel` que se establece en la tabla.

5.2. Eliminar Pedido Seleccionado:

- El método `eliminarPedidoSeleccionado()` permite eliminar un pedido seleccionado de la base de datos.
- Primero, se obtiene el pedido seleccionado en la tabla y se muestra una confirmación.
- Si se confirma la eliminación, se ejecuta una consulta SQL para eliminar el pedido.
- Se muestra un mensaje de éxito o error y se actualiza la tabla.

5.3. Interfaz Gráfica Principal:

- El método `getPanel()` devuelve el panel principal que contiene la tabla y el botón.

Imagen 42. Pedidos. Imagen propia.

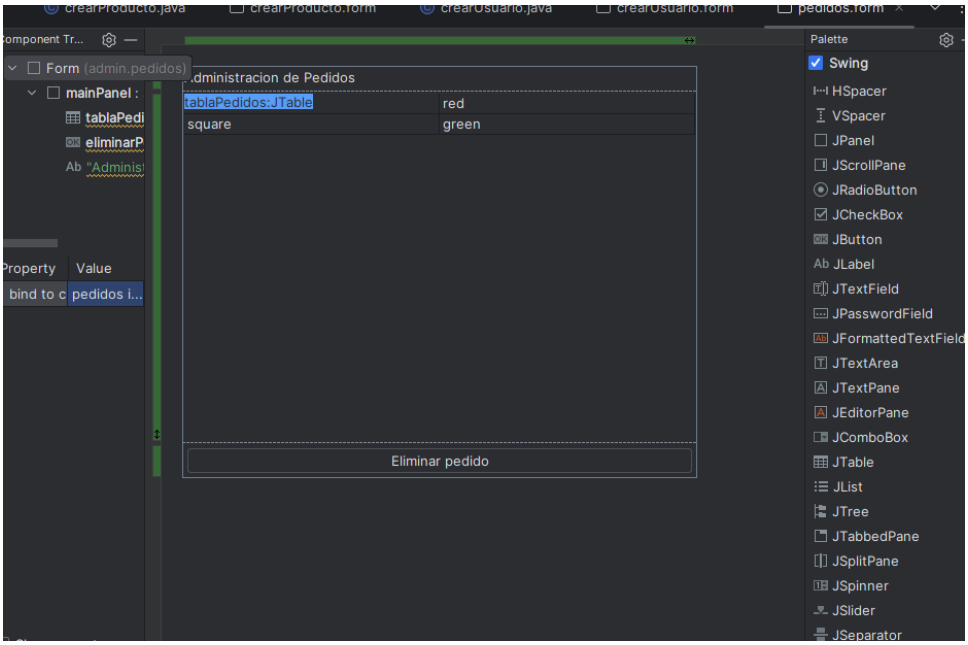


Imagen 43. Pedidos Form. Imagen propia

6 Productos

Se proporciona una interfaz para ver, editar, eliminar y añadir productos en una base de datos. Los datos se muestran en una tabla y se pueden editar y eliminar productos seleccionados. También permite añadir nuevos productos mediante una ventana emergente.

- La clase `productos` crea una interfaz para mostrar y gestionar productos.
- Incluye una tabla para mostrar información de los productos, campos para editar productos y botones para guardar cambios, eliminar productos y añadir nuevos productos.

6.1. Cargar Datos en la Tabla:

- El método `cargarDatosEnJTable()` se encarga de cargar los datos de los productos desde la base de datos y mostrarlos en la tabla.
- Se realiza una consulta SQL para obtener información de los productos.
- Los datos se insertan en un `DefaultTableModel` que se establece en la tabla.

6.2. Editar y Guardar Cambios:

- Los campos de texto permiten editar el nombre y el valor del producto.
- El botón "Guardar Cambios" guarda los cambios en la base de datos.

6.3. Eliminar Producto Seleccionado:

- El botón "Eliminar Producto" permite eliminar un producto seleccionado de la base de datos.
- Se muestra una confirmación antes de la eliminación.

6.4. Añadir Nuevo Producto:

- El botón "Añadir Producto" abre una nueva ventana para crear un nuevo producto utilizando la clase `crearProducto`.

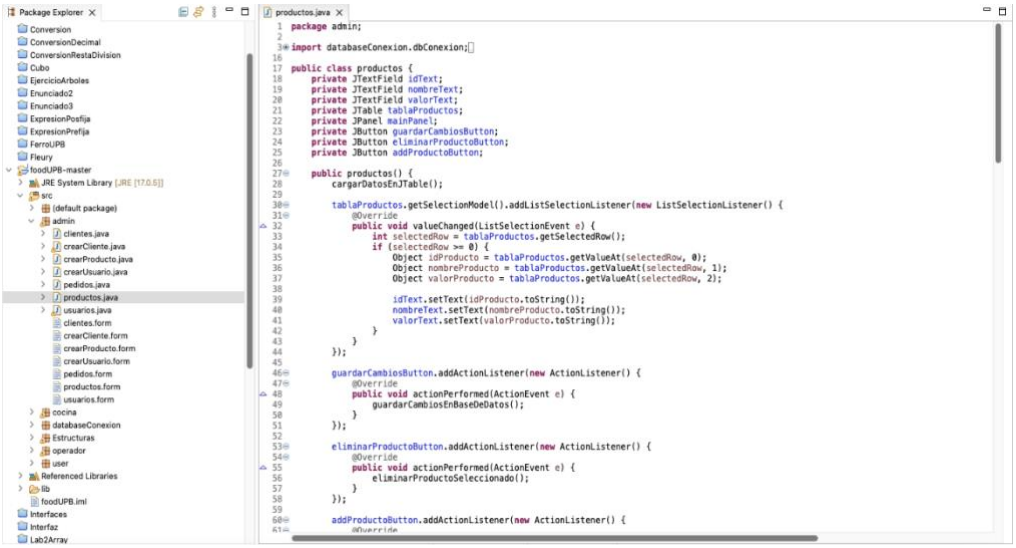


Imagen 44. Productos. Imagen propia.

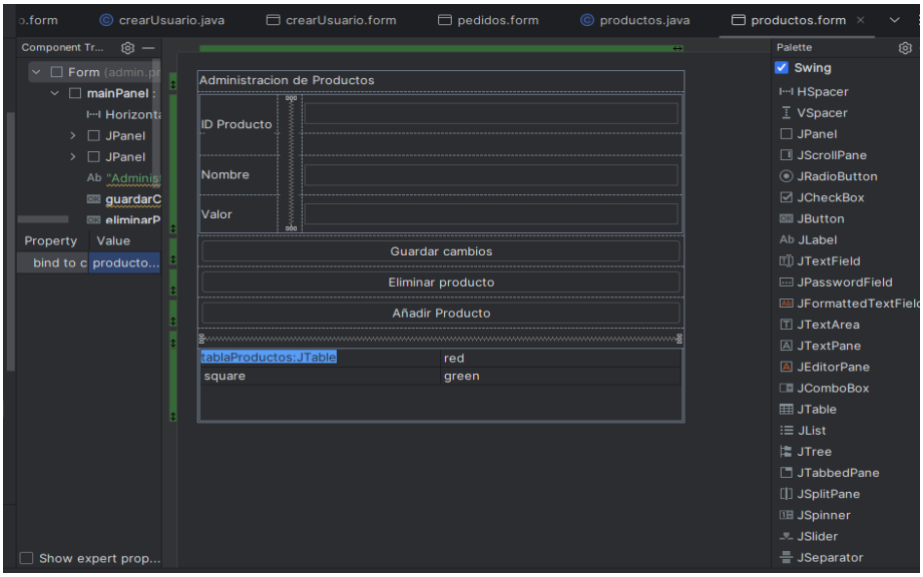


Imagen 45. Productos Form. Imagen propia

Se proporciona una interfaz para ver, editar, eliminar y añadir usuarios en una base de datos. Los datos se muestran en una tabla y se pueden editar y eliminar usuarios seleccionados. También permite navegar a otras secciones de la aplicación.

- La clase `usuarios` crea una interfaz para mostrar y gestionar usuarios.
- Incluye una tabla para mostrar información de los usuarios, campos para editar usuarios y botones para guardar cambios, eliminar usuarios y añadir nuevos usuarios.

7.1. Cargar Datos en la Tabla:

- El método `cargarDatosEnJTable()` se encarga de cargar los datos de los usuarios desde la base de datos y mostrarlos en la tabla.
- Se realiza una consulta SQL para obtener información de los usuarios.
- Los datos se insertan en un `DefaultTableModel` que se establece en la tabla.

7.2. Editar y Guardar Cambios:

- Los campos de texto permiten editar el nombre, tipo de usuario y contraseña del usuario.
- El botón "Guardar Cambios" guarda los cambios en la base de datos.

7.3. Eliminar Usuario Seleccionado:

- El botón "Eliminar Usuario" permite eliminar un usuario seleccionado de la base de datos.
- Se muestra una confirmación antes de la eliminación.

7.4. Añadir Nuevo Usuario:

- El botón "Añadir Usuario" abre una nueva ventana para crear un nuevo usuario utilizando la clase `crearUsuario`.

7.5. Navegación entre Secciones:

- Los botones "Clientes", "Pedidos" y "Productos" permiten navegar a otras secciones de la aplicación, como la gestión de clientes, pedidos y productos.

Imagen 46. Usuarios. Imagen propia.

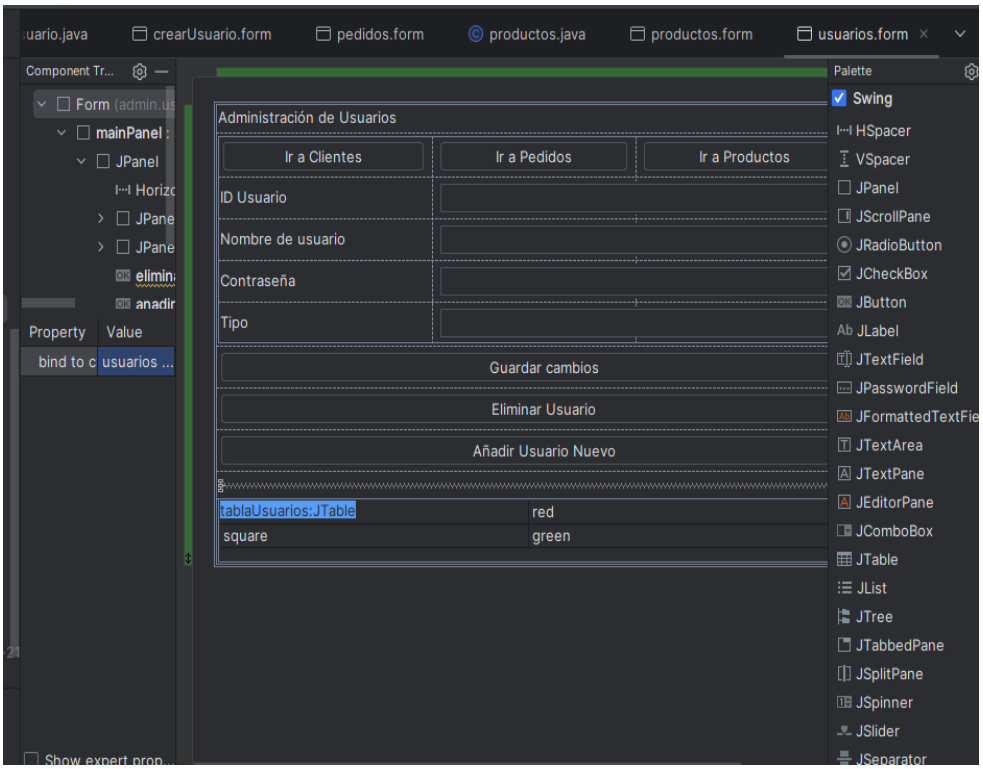


Imagen 47. Uusarios Form. Imagen propia.

d) DataBase Conexión.

Se proporciona una forma de obtener una conexión a una base de datos MySQL utilizando los detalles de conexión proporcionados. Es importante asegurarse de que los detalles de conexión sean correctos para que la conexión se establezca correctamente.

1.1. Conexión a la Base de Datos:

- La clase tiene una función estática `obtenerConexion` que devuelve un objeto `Connection`. Esta función es utilizada para establecer una conexión a una base de datos MySQL.

1.2. Detalles de Conexión:

- Los detalles de la conexión a la base de datos, como la URL de la base de datos, el nombre de usuario y la contraseña, están definidos como constantes en la clase:
  - URL: La URL de la base de datos MySQL, que generalmente incluye el servidor y el puerto, seguido del nombre de la base de datos.

  - USUARIO: El nombre de usuario para autenticarse en la base de datos.
  - CONTRASEÑA: La contraseña del usuario.

1.3. Establecer Conexión:

- En el método `obtenerConexion`, se intenta establecer una conexión utilizando los detalles proporcionados.
- Si la conexión se establece con éxito, se devuelve el objeto `Connection`.

1.4. Manejo de Errores:

- Si ocurre algún error durante la conexión, se captura la excepción `SQLException` y se muestra un mensaje de error en la salida estándar de error.

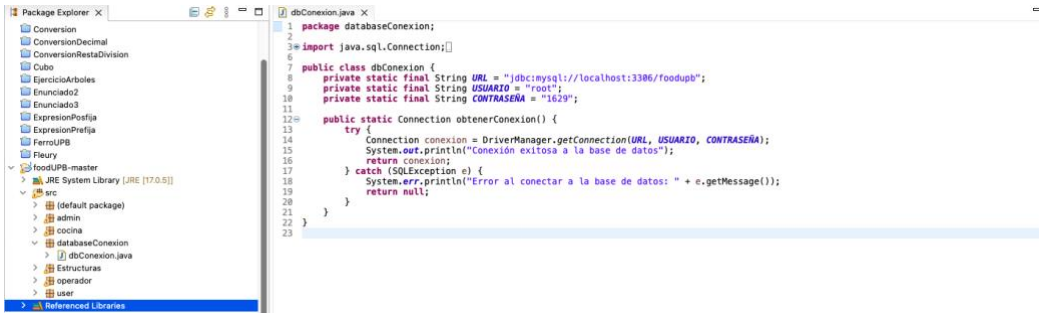


Imagen 48. DB conexión. Imagen propia.

e) Estructuras

Este código es una implementación simplificada de una cola de despacho de cocina que podría ser parte de un sistema de gestión de pedidos de un restaurante o un servicio de entrega de comida. Ayuda a organizar los pedidos según la prioridad y facilita la preparación y entrega eficiente de la comida.

La clase `Pedido` se utiliza para representar un pedido. Cada pedido tiene un número, una prioridad (1 para premium y 2 para normal) y un tiempo de cocción.

1. Clase ColaCocina:

- La clase `ColaCocina` representa la cola de despacho de cocina.
- Tiene un arreglo de pedidos llamado `cola` y un conjunto de atributos para realizar un seguimiento del tamaño, la capacidad y las posiciones de frente y fin de la cola.

1.2. Funciones de la Clase:
- `estaVacia()`: Verifica si la cola está vacía.
- `estaLlena()`: Verifica si la cola está llena.
- `insertar(int numero, int prioridad, int tiempoCoccion)`: Inserta un nuevo pedido en la cola.
- `extraer()`: Extrae y devuelve el pedido en la parte frontal de la cola.
- `imprimirCola()`: Muestra los pedidos en la cola de despacho.

1.3. Inserción de Pedidos:

- Cuando se inserta un pedido, se ordena en la cola según su prioridad. Los pedidos premium tienen prioridad sobre los pedidos normales.
- Se realiza una inserción ordenada para garantizar que los pedidos premium se atiendan primero.

1.4. Extracción de Pedidos:
- Los pedidos se extraen de la parte frontal de la cola. Los pedidos premium se atienden primero.
- La función `extraer` devuelve el pedido que se va a preparar.

1.5. Impresión de la Cola:
- La función `imprimirCola` muestra los pedidos en la cola, incluyendo su número, prioridad y tiempo de cocción.

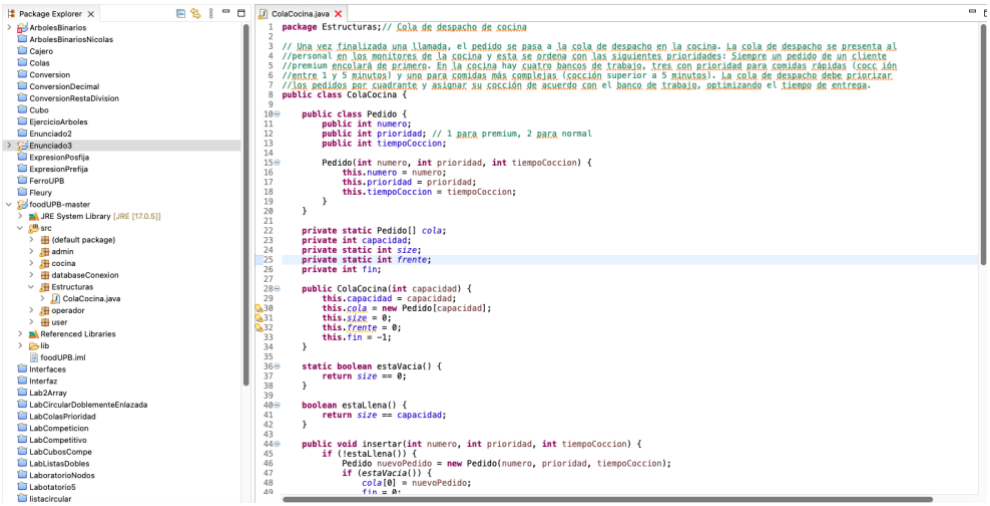


Imagen 49. Estructuras cola. Imagen propia.

2. Clase Node< T > :

La clase **Node** desglosa un nodo en una estructura de datos enlazada. Cada instancia de esta clase almacena dos componentes clave: el campo data, que guarda un dato de cualquier tipo, y el campo next, que apunta al siguiente nodo en la secuencia. Esto permite la construcción de listas enlazadas, donde los nodos se conectan entre sí a través de las referencias next, formando una secuencia lineal de datos. Cuando se desea acceder a los elementos en la lista, se sigue el enlace next de un nodo al siguiente, lo que permite la iteración a través de los datos de manera eficiente. Esta estructura es fundamental para la implementación de diversas estructuras de datos, como listas enlazadas, pilas y colas, y proporciona una forma flexible de organizar y acceder a los datos en un orden específico.

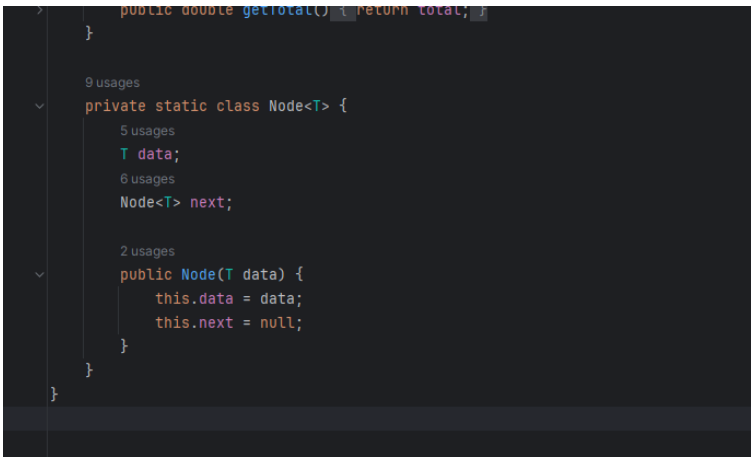


Imagen 50. Estructuras Node. Imagen propia.

f) User

Se implementa una funcionalidad de inicio de sesión que verifica las credenciales del usuario en la base de datos y redirige al usuario a vistas específicas según su rol. Las vistas específicas se abren en nuevas ventanas, y se asegura que las ventanas anteriores se cierren al abrir una nueva vista.

1. Clase loginForm:
- Representa un formulario de inicio de sesión.
- Contiene campos para ingresar un usuario y una contraseña, así como un botón para iniciar sesión.

1.2. Verificación de Credenciales:
- Cuando se hace clic en el botón de inicio de sesión (`button1`), se verifica si el usuario y la contraseña coinciden en la base de datos utilizando la función `verificarCredenciales`.
- La función `verificarCredenciales` realiza una consulta SQL para verificar la autenticación del usuario en la base de datos. Si las credenciales son correctas, el usuario obtiene acceso.

1.3. Redirección según el Rol:
- Si las credenciales son correctas, el código comprueba el dominio del usuario para determinar su rol.
- Si el usuario es un operador (`@operador.com`), se abre una vista de búsqueda de número (`buscarNumero`).
- Si el usuario es un administrador (`@admin.com`), se abre una vista de usuarios (`usuarios`).
- Si el usuario es personal de cocina (`@cocina.com`), se abre una vista de cocina (`verCocina`).
- En otros casos, se muestra un mensaje de "Acceso concedido".

1.4. Métodos de Apertura de Vistas:
- Para cada rol, se proporcionan métodos (`abrirVistaBuscarNumero`, `abrirVistaUsuarios`, `abrirVistaVerCocina`) para abrir las vistas correspondientes después de la autenticación.

1.5. Cierre de Ventanas Anteriores:
- Antes de abrir una nueva vista, el código cierra la ventana actual en la que se encuentra el formulario de inicio de sesión.

1.6. Aspectos Visuales:
- Se ajustan propiedades visuales de las ventanas, como el tamaño, el cierre de ventanas y el contenido a mostrar en función del rol del usuario.

Imagen 51. Login de la app. Imagen propia.

VIDEO EXPLICACIÓN DE LA APLICACIÓN: https://youtu.be/kitg4sOvrPI

COMPATIBILIDAD CON UBUNTU 22.04.

Desde el inicio de nuestro proyecto, el cliente se presentó un requisito claro y fundamental: nuestra aplicación debe funcionar sin problemas en Ubuntu 22.04, una de las últimas versiones de este sistema operativo ampliamente utilizado. Para asegurar el cumplimiento de este requisito, seguimos un enfoque ordenado y meticuloso:

1. Optimización Técnica: Realizamos una revisión exhaustiva de todo nuestro código, lo que resultó en ajustes técnicos y optimizaciones necesarios para garantizar la ejecución sin contratiempos en Ubuntu 22.04.

2. Pruebas Rigurosas: Sometimos nuestra aplicación a pruebas minuciosas en un entorno Ubuntu 22.04. Cada aspecto de la aplicación fue sometido a una evaluación detallada, y se abordaron de manera proactiva cualquier incompatibilidad potencial.

3. Seguridad y Privacidad: Confirmamos que nuestra aplicación cumple con los rigurosos estándares de seguridad y privacidad de Ubuntu 22.04 para proteger la integridad de los datos y la información del usuario.

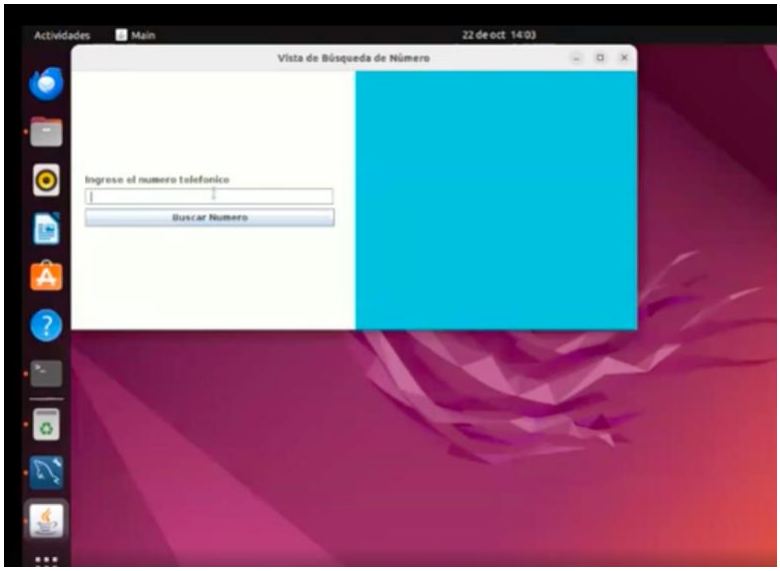


Imagen 52. Implementación proyecto upb-food en ubuntu 22.04. Imagen propia

VIDEO EXPLICACIÓN DE LA IMPLEMENTACIÓN DE UBUNTU 22.04 AL PROYECTO: https://youtu.be/rdJM_DvRh5M

7. Cronograma

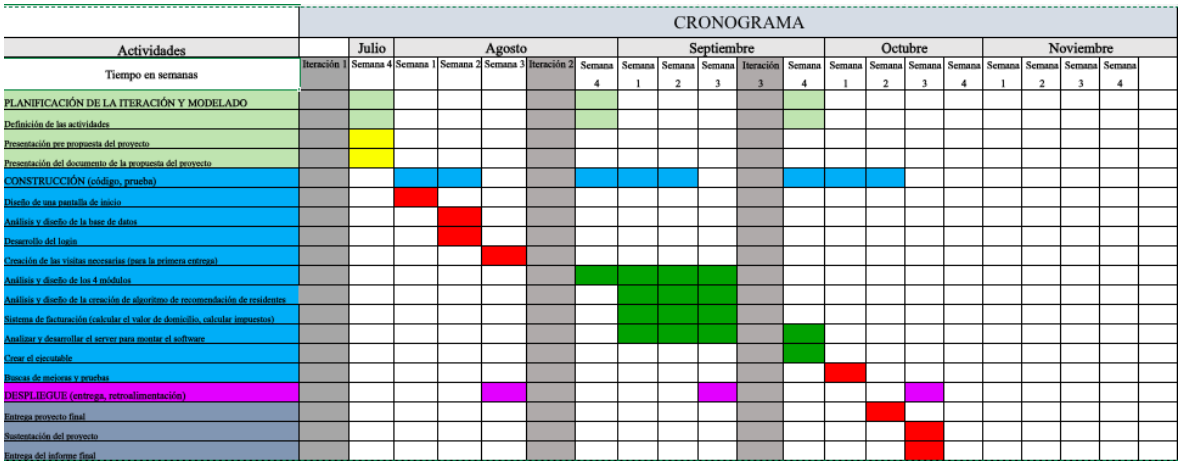


Imagen 53. Cronograma de actividades semanales. Imagen propia

8. DISCUSIONES Y CONCLUSIONES

A lo largo de este proyecto, se han empleado una serie de estructuras y técnicas fundamentales para lograr el éxito en el desarrollo de la aplicación. La implementación de un sistema de autenticación y la diferenciación de roles de usuario se basaron en estructuras de control de acceso y gestión de sesiones, lo que garantizó la seguridad y la efectividad en la asignación de funciones específicas a distintos usuarios, como operadores, administradores y personal de cocina.

La conexión a una base de datos MySQL se llevó a cabo mediante el diseño de estructuras de datos que permitieron el almacenamiento y la recuperación eficientes de información vital para el funcionamiento del restaurante. La modularidad del código se logró mediante la creación de módulos independientes y la aplicación de principios de diseño de software, lo que facilitó la expansión y el mantenimiento del sistema en el futuro.

Para abordar desafíos técnicos y desconocidos, se llevaron a cabo investigaciones exhaustivas. Esto implicó la consulta de documentación, tutoriales y fuentes en línea para comprender y aplicar conceptos como SQL, gestión de ventanas, optimización de la interfaz de usuario y el manejo seguro de información sensible, como contraseñas.

La adaptación de tecnologías avanzadas, como la implementación de una base de datos MySQL y la integración con un proyecto en Java, se basó en investigaciones detalladas para garantizar la compatibilidad y el rendimiento adecuado.

El proyecto fue un ejercicio de integración de estructuras de datos, investigaciones rigurosas y adaptación tecnológica para lograr una aplicación exitosa en la gestión de un restaurante de comida rápida. Las estructuras de control, diseño modular y la investigación desempeñaron un papel crucial en la consecución de los objetivos del proyecto.

Finalmente, todas las experiencias, desafíos y extensas investigaciones realizadas en este proyecto no solo han contribuido al éxito de la aplicación, sino que también representan un valioso aprendizaje que nos prepara para nuestro futuro como profesionales en el campo de la ingeniería de sistemas e informática. La adquisición de conocimientos técnicos, la capacidad de resolver problemas y la adaptabilidad a nuevas tecnologías son habilidades cruciales que nos servirán como ingenieros en sistemas e informática. Además, el trabajo en equipo, la gestión de proyectos y la comunicación efectiva son destrezas que continuarán siendo fundamentales en nuestro camino profesional. Este proyecto ha sido una sólida base para el crecimiento y la preparación de cara a los desafíos y oportunidades que nos esperan en nuestra futura carrera y en nuestro futuro como profesionales.

Conclusiones adicionales:

1. La conexión a una base de datos (base de datos MySQL) es fundamental para almacenar y gestionar la información de la aplicación.

2. La modularidad del código facilita la expansión y el mantenimiento del sistema.

3. La optimización de la interfaz de usuario (UI) mejora la experiencia del usuario y la usabilidad.

4. La gestión de errores y excepciones es crucial para prevenir fallos inesperados y mantener la estabilidad del sistema.

5. La comunicación con la base de datos debe manejarse de manera segura para evitar vulnerabilidades de seguridad.

6. La adaptación de la aplicación a diferentes roles de usuario permite una experiencia personalizada y brinda una muy buena experiencia de usuario.

7. El seguimiento constante y la comunicación efectiva entre el equipo de desarrollo ayudaron a mantener un flujo de trabajo productivo.

8. La colaboración estrecha entre los otros miembros del equipo genera una sinergia en el código garantizando la entrega de un buen producto.

9. La división del proyecto en módulos y tareas específicas facilitó la asignación de responsabilidades y la supervisión del progreso.

10. La gestión de versiones y la documentación constante garantizaron la trazabilidad de los cambios y la integridad del código.

11. La colaboración en equipo permitió corregir problemas y mejorar la funcionalidad del producto.

12. La iteración y la retroalimentación constante del equipo permitieron refinar y mejorar el programa en cada etapa.

13. La entrega de un programa funcional en el plazo establecido y dentro del presupuesto es el resultado de una gestión y un trabajo en equipo eficaces.

pon coclisones extras

9.REFERENCIAS BIBLIOGRÁFICAS

-Wikipedia contributors. (s/f). Cola de prioridades. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Cola_de_prior idades&oldid=145477000 [1]

-Wikipedia contributors. (s/f). Modelo–vista–controlador. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Modelo%E2% 80%93vista%E2%80%93controlador&oldid=138615253 [2]

-¿Qué es el desarrollo en Espiral? (2020, abril 8). Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/article s/que-es-el-desarrollo-en-espiral.html [3]

-Wikipedia contributors. (s/f). Control de versiones. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Control_de_v ersiones&oldid=152003367 [4]

-¿Qué es Git? (s/f). Microsoft.com. Recuperado el 3 de septiembre de 2023, de https://learn.microsoft.com/es- es/devops/develop/git/what-is-git[5]

-Git - Acerca del Control de Versiones. (s/f). Git-scm.com. Recuperado el 23 de octubre de 2023, de https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones
[6]

-IBM Documentation. (2022, septiembre 20). Ibm.com. https://www.ibm.com/docs/es/qmf/12.1.0?topic=access-accessing-data-remote-database
[7]

-IBM Documentation. (2022, septiembre 20). Ibm.com. https://www.ibm.com/docs/es/qmf/12.1.0?topic=access-accessing-data-remote-database
[8]

-S/f). Amazon.com. Recuperado el 23 de octubre de 2023, de https://aws.amazon.com/es/what-is/sql/
[9]

-Wikipedia contributors. (s/f). Base de datos. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Base_de_datos&oldid=154597027
[10]

-Recoverit. (2022, agosto 9). ¿Qué es un archivo JAR? Wondershare Recoverit. https://recoverit.wondershare.es/file-tips/what-is-jar-file.html
[11]

-GUI designer basics. (s/f). IntelliJ IDEA Help. Recuperado el 23 de octubre de 2023, de https://www.jetbrains.com/help/idea/gui-designer-basics.html.
[12]

-Martinez, D. G. (2023, junio 1). ¿Qué es Ubuntu y para qué sirve? Blog; GoDaddy. https://es.godaddy.com/blog/que-es-ubuntu-y-para-que-sirve/.
[13]

-Ramírez, I. (2016, julio 25). Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas. Xataka.com; Xataka. https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-

funcionan-y-como-utilizarlas.
[14]

-Design patterns: Elements of reusable object-oriented software. (1995). Addison-Wesley Professional.

-Software development kits and command line interface. (s/f). Oracle.com. Recuperado el 9 de mayo de 2023, de https://docs.oracle.com/en-us/iaas/Content/API/Concepts/sdks.htm

-RMI. (2015, febrero 25). RMI; Rocky Mountain Institute. https://rmi.org/

-Trail: RMI. (s/f). Oracle.com. Recuperado el 3 de septiembre de 2023, de https://docs.oracle.com/javase/tutorial/rmi/index.html

-Follow, Y. Y. (2020, julio 18). *Difference between AWT and swing in java*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-awt-and-swing-in-java/

- Hasenmueller, A. (s/f). *Oracle VM VirtualBox*. Virtualbox.org. Recuperado el 23 de octubre de 2023, de https://www.virtualbox.org/

- *JAR file overview*. (s/f). Oracle.com. Recuperado el 23 de octubre de 2023, de https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jarGuide.html

-*JAR files in java*. (2016, diciembre 16). GeeksforGeeks. https://www.geeksforgeeks.org/jar-files-java/

- Noviantika, G. (2022, abril 1). *What is Ubuntu? A quick beginner's guide*. Hostinger Tutorials; Hostinger. https://www.hostinger.com/tutorials/what-is-ubuntu

- *Remote database servers*. (s/f). Embarcadero.com. Recuperado el 23 de octubre de 2023, de https://docwiki.embarcadero.com/InterBase/2020/en/Remote_Database_Servers

- *SQL introduction*. (s/f). W3schools.com. Recuperado el 23 de octubre de 2023, de https://www.w3schools.com/sql/sql_intro.asp

- *What is a virtual machine?* (2022, agosto 4). VMware. https://www.vmware.com/content/vmware/vmware-published-sites/us/topics/glossary/content/virtual-machine.html.html.html

- *What is Software Testing and How Does it Work?* (s/f). Ibm.com. Recuperado el 23 de octubre de 2023, de https://www.ibm.com/topics/software-testing

-(S/f-a). Mysql.com. Recuperado el 23 de octubre de 2023, de https://www.mysql.com/downloads/

-(S/f-b). Mysql.com. Recuperado el 23 de octubre de 2023, de https://www.mysql.com/products/workbench/

-(S/f). Amazon.com. Recuperado el 23 de octubre de 2023, de https://aws.amazon.com/es/what-is/structured-data/#:~:text=Structured%20data%20is%20data%20that,due%20to%20its%20quantitative%20nature.

-Weiss, M. A. (1998). Data structures and problem solving using Java. *ACM SIGACT News*, *29*(2), 42–49. https://doi.org/10.1145/288079.288084

-Imieliński, T., & Virmani, A. (1999). *Data mining and knowledge discovery*, *3*(4), 373–408. https://doi.org/10.1023/a:1009816913055

-What is Software Testing? Definition, Types, and Tools. (2023, febrero 17). *Katalon.com*. https://katalon.com/resources-center/blog/software-testing

-*What is a database?* (s/f). Oracle.com. Recuperado el 23 de octubre de 2023, de https://www.oracle.com/database/what-is-database/

-Deacon, J. (s/f). *Model-view-controller (MVC) architecture*. Cloudfront.net. Recuperado el 23 de octubre de 2023, de https://d1wqtxts1xzle7.cloudfront.net/50526307/MVC-libre.pdf?1480020702=&response-content-disposition=inline%3B+filename%3DModel_View_Controller_MVC_Architecture.pdf&Expires=1698033292&Signature=dx8Vn~0bqsIQv7qWJ6yBoeJFMxTJqoLla1gl1i3vbW1gHyfSzffTYJgh0fCgAopOFEFrcTgcDdjc0ic8c3VFau0RwBfXddWfsWea3TlFCgXc8M1pA2vvlKnL1BP7psU47kun5D0Um~dj3clKAuMuDuxG1~cJ5iKjTAlSRg7dtNJ8yQ5TAMmOrEW~uzibS7ef2NCaZQ1hgNBLXFRcRb0BTRxbcqoxlyGarVtsLvS3LF9eqFS3qNqkDWIw~mO4b-RxUL5HZzMyHQx8hdl8HOkjBDB9iO1aO6nwLJXmaPv~BicMOZafSrpxweYhnM2-8Ack3t9t9C23K-FRD29MyeW1Bw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

-Bishop, J., & Bishop, N. (2000). Object-orientation in Java for scientific programmers. *SIGCSE Bulletin*, *32*(1), 357–361. https://doi.org/10.1145/331795.331885

-Dale. (2001). *Data Structures in Java*. Jones and Bartlett
.