Práctica 1

Gestión del restaurante Guzman’s

Programación Orientada a Objetos

Grupo 2-Equipo 1

Andrés Loaiza García, Ángel Sebastián Cuarán Cruz, Juan Camilo León Barrios, Luis Carlos Sánchez Flórez, Pablo Bernal García

Profesor encargado: Jaime Alberto Guzmán Luna

Universidad Nacional de Colombia-Sede Medellín

Fecha de entrega: 28-01-2025

Descripción General de la Solución

1. Identificación de los Problemas y Necesidades
Problemas Identificados:

1. Gestión Compleja de un Restaurante Multisucursal:
- Los restaurantes que operan con varias sucursales enfrentan desafíos significativos en la administración centralizada de recursos, personal, inventarios, pedidos y finanzas.
- La ausencia de automatización en estos procesos conduce a tareas manuales que son propensas a errores y resultan en una pérdida de eficiencia operativa.

2. Falta de Automatización en Procesos Clave:
- El cálculo de costos, como los de envíos, salarios y presupuestos, se realiza de manera manual, lo que aumenta el riesgo de errores y retrasos.
- La asignación de repartidores, mesas y otros recursos depende en gran medida de la intervención humana, lo que puede ser ineficiente y poco escalable.

3. Necesidad de Optimizar la Experiencia del Cliente y del Usuario Interno:
- Los clientes esperan un servicio rápido, confiable y sin complicaciones, tanto en pedidos a domicilio como en pedidos físicos en el restaurante.
- Los empleados, incluyendo administradores, meseros, repartidores y chefs, requieren herramientas que les permitan trabajar de manera más eficiente y efectiva, reduciendo la carga de trabajo manual y mejorando la productividad.

Necesidades del Sistema:
- Centralización: Es fundamental contar con un sistema que permita la gestión centralizada de las operaciones de todas las sucursales, facilitando el control y la coordinación desde un único punto.
- Automatización: El sistema debe automatizar procesos clave como el cálculo de costos, la asignación de recursos y la gestión de inventarios, reduciendo así los errores humanos y mejorando la eficiencia.
- Funcionalidades Específicas: El sistema debe ofrecer herramientas adaptadas a las necesidades de cada tipo de usuario, ya sean administradores o empleados, para garantizar una experiencia óptima en cada interacción.

---

2. Requisitos Funcionales y No Funcionales
Requisitos Funcionales:
1. Gestión de Pedidos:
- Permitir a los clientes realizar pedidos a domicilio, seleccionando barrios y productos, con costos calculados automáticamente.
- Gestionar pedidos físicos en mesas asignadas a sucursales.

2. Gestión de Personal:
- Administrar empleados, como meseros, repartidores y chefs, incluyendo su contratación, asignación a sucursales y evaluación de desempeño.

3. Gestión Financiera:
- Calcular presupuestos, gestionar deudas y simular préstamos para la apertura de nuevas sucursales.

4. Administración de Inventarios:
- Controlar los ingredientes necesarios para preparar los platos y actualizar automáticamente el stock al realizar pedidos.

5. Automatización de Recursos:
- Asignar automáticamente repartidores disponibles según el pedido y la ubicación.
- Optimizar el uso de mesas y otros recursos en cada sucursal.

Requisitos No Funcionales:

1. Eficiencia:
- El sistema debe responder a solicitudes de pedidos, cálculo de costos y asignaciones en menos de 1 segundo.

2. Escalabilidad:
- Debe ser capaz de soportar múltiples sucursales y grandes volúmenes de pedidos sin pérdida de rendimiento.

3. Seguridad:
- Restringir el acceso a funcionalidades administrativas mediante autenticación de usuarios.

4. Usabilidad:
- Ofrecer una interfaz simple y clara que minimice los errores tanto del usuario interno como del externo.

5. Persistencia de Datos:
- Garantizar la integridad de los datos a través de serialización y recuperación eficiente de la información.

---

3. Herramientas y Funcionalidades Clave del Sistema

1. Pedidos y Clientes:
- Clases como Cliente, Pedido y Domicilio gestionan los pedidos, el historial de compras y la asignación de repartidores.

2. Gestión de Personal:
- Clases como Empleado, Mesero, Repartidor y Chef permiten controlar el desempeño y la disponibilidad de los empleados.

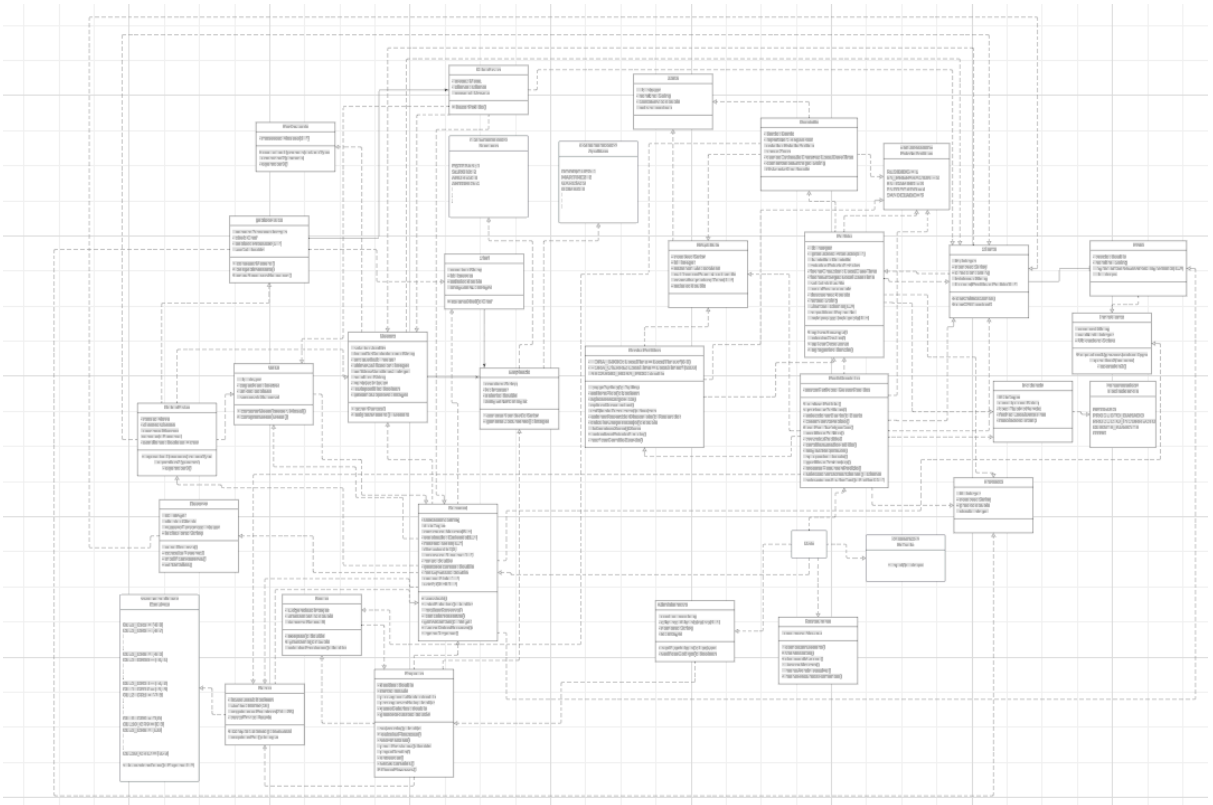
3. Administración Financiera y Sucursales:
- Clases como Empresa y Sucursal manejan las finanzas, presupuestos y recursos físicos de cada ubicación.

4. Automatización y Control:
- Clases como Barrio, Esquina y Zona definen las ubicaciones y calculan automáticamente los costos de envío .

5. Control de Inventarios:

- Clases como Ingrediente y Producto administran los recursos necesarios para el servicio.

Diseño estático del sistema



https://lucid.app/lucidchart/f3fee765-4cf5-429a-ab14-4690286e9b35/edit?viewport_loc=-4672%2C-2447%2C9522%2C3821%2C0_0&invitationId=inv_c03dc7bd-bf5d-4f4b-98bc-7a89fea24869

Main: Esta es la clase principal del programa, es la que se encarga de inicializar los procesos y dar acceso a cada una de las funcionalidades, al igual que guardar los datos que se crean al ejecutar las funciones del programa.

- Se relaciona con la interfaz entrada para poder asegurarse que los datos ingresados no generen errores no deseados.
- Se relaciona con la clase PesirDomicilio para inicializar la función con este nombre.
- Se relaciona con la clase administrativo para realizar la autenticación del usuario como administrador para así ingresar a las funciones de abrir sede y contratar personal
- Se relaciona con la clase OrdenFisica para iniciar esta función.
- Se relaciona con la clase Sucursal pues esta es la encargada de ejecutar la funció de hacer reserva.

PedirDomicilio: Clase encargada de inicializar la función de pedir un domicilio, generando la instancia necesaria para recabar la información que se necesita del usuario para poder cumplir la función de forma eficaz.

- Se relaciona con el GestoPedidos para poder pasarle la información necesaria para proseguir con la función.
- Se relaciona con Pedido a la hora de crear el objeto protagonista de la función.
- Se relaciona con Cliente, EstadoProducto, Incidencia y Producto, pues los datos que poseen estas clases son la información necesaria para poder crear el pedido

GestorPedidos: Esta es la clase en la que, como su nombre lo indica, se gestionan los pedidos, teniendo los métodos para ordenarlos y modificarlos de la manera que sea necesaria también es el que vela por que los horarios de los pedidos funcionen correctamente.

- Se relaciona con repartidor pues esta clase es la encargada de asignar los repartidores para cada pedido
- Se relaciona con EstadoPedido porque en esta puede modificarse el estado a voluntad.
- Se relaciona con Pedido, pues esta clase se encarga de modificar los pedidos para su correcto funcionamiento.

OrdenFisica: Clase destinada a iniciar la función de orden física, se encarga de recopilar la información para el desarrollo de la funcionalidad.

- Se relaciona con Mesa, Mesero, Cliente, Chef, Plato y Sucursal, pues esta es la información necesaria para poder crear la instancia del pedido.
- Se relaciona con pedidoFisico, pues hereda la información para crear la instancia que se encarga de continuar con la función.

PedidoFisico: Esta clase está encargada de organizar las órdenes físicas y realizar la facturación de estas, al igual que permitir la calificación del trabajo que ejercen los empleados.

- Se relaciona con OrdenFisica pues hereda la información que ésta reúne, y se relaciona con las mismas clases para trabajar con la información que estas poseen.

Administrativo: Es la clase que posee las instancias con las que se permite acceder a la gestión económica y humana de la empresa, verifica que el usuario de verdad es un administrador y da comienzo a las funcionalidades de abrir sucursal y contratar personal.

- Se relaciona con Empresa y Restaurante porque es la encargada de convocar los métodos de estas dos clases para iniciar las funciones ya dichas.

Apellido y Nombre: Enumeraciones creadas para facilitar la generación de nombres aleatorios para los meseros por defecto de las nuevas sucursales.

Banco: Clase que se encarga de la interacción de financiación a la hora de abrir nuevas sucursales, esta posee los métodos donde se calcula el dinero que se tendrá disponible para poder abrir una nueva sede.

Barrio: clase que posee los datos de mapeado de la ciudad, cada instancia de barrio será un posible lugar para abrir una nueva sucursal, y precisamente se encarga de explorar y designar posibles localizaciones para abrir una sucursal.

- Se relaciona con Esquina, pues esta enumeración es la que posee todas las posibles coordenadas que tiene el barrio.

Chef: Instancias que serán utilizadas para la creación de órdenes físicas y serán calificados tras estas.

- Se relaciona con sucursal pues cada chef tiene una sucursal asignada.
- Se relaciona con Empleado, pues hereda información de esta clase.

Cliente: Representa al usuario, un cliente es una instancia que se guardará con la intención de que múltiples participaciones de un mismo usuario cambien el comportamiento del programa, guardando el historial de participaciones de un usuario en el programa.

- Se relaciona con Plato, Pedido y Pedido físico, pues estas son las instancias que se guardan en el historial de un cliente.

Domicilio: Representa el hogar a donde será enviado un pedido, posee la información para calcular el precio y posible tiempo de los pedidos.

- Se relaciona con Repartidor, pues dependiendo de su ubicación se le será asignado uno para realizar la entrega.
- Se relaciona con Barrio, pues cada casa está ubicada en uno, por lo que tiene un barrio asignado.
- Se relaciona con estadoPedido para mostrar precisamente el estado del pedidoque será entregado en este domicilio.

Empleado: Clase con la información básica que tienen en común todos los integrantes de restaurante, hernando datos como la id o el nombre.

- Se relaciona con Cliente y Apellido porque en esta clase está la función de generar nombres,

Empresa: Clase encargada de administrar y actualizar las finanzas de la empresa, esta es la encargada de llamar a todos los métodos necesarios para la creación de nuevas sucursales.

- Se relaciona con Banco para poder pedir el financiamiento necesario para llevar a cabo la funcionalidad.
- Se relaciona con Barrio pues aporta el presupuesto que tendrá en cuenta el barrio para escoger la localización.
- Se relaciona con Sucursal, para poder pasar la información y crear las nuevas instancias del restaurante.

Esquina: Enumeración donde se guardan todas las coordenadas que tiene la ciudad.

EstadoPedido: Enumeración donde se presentan los diferentes estados que puede tener un pedido, como recibido, en preparación, en camino, entregado y cancelado.

Incidencia: Clase donde se representan posibles problemas que puedan suceder durante el envío de un producto.

Mesa: Entidades usadas para representar un límite de espacio en las reservas y las órdenes físicas.

Mesero: Instancias encargadas de atender clientes, son asignadas a las órdenes físicas, y al ser el puesto más volátil, hay una funcionalidad que trata precisamente de contratarlos, pues son un límite más pequeño que las mesas, pueden recibir calificaciones tras realizar un trabajo.

- Se relaciona con sucursal pues cada chef tiene una sucursal asignada.
- Se relaciona con Empleado, pues hereda información de esta clase.

Pedido: Instancias que se crean durante la función de pedir un domicilio, estas son las guardadas en el historial de los clientes, tiene la información completa referente a lo que son las órdenes a distancia, tanto precios, como información general.

Plato y Producto: Son instancias que representan cosas similares, las órdenes que se les entregarán a los clientes, cada una con sus características que las hacen más viables en pedidos a domicilio y en pedidos físicos respectivamente.

Repartidor: Entidades que son asignadas a los domicilios dependiendo de la ubicación donde serán enviados, reciben calificaciones.

- Se relaciona con Barrio, pues este tiene una serie de barrios asignados para trabajar.

Reservacion: Instancias que se guardan al realizar una reservación.

- Se relacionan con Mesa, pues estas tienen una mesa asignada.

Restaurante: Clase encargada de la gestión de recursos humanos, esta es la principal participante en el proceso de contratar meseros.

- Se relaciona con Mesero, pues se encarga de estar creando objetos de dicha clase.
- Se relaciona con Sucursal, pues esta designa los meseros a diferentes sucursales.

Sucursal: De las clases más participativas, tiene participación directa en la contratación de meseros, la creación de reservas(Función que inicia esta misma clase) y evidentemente la construcción de sucursales, al igual que indirectamente participa en los pedidos físicos, las instancias de esta clase representan los locales que tiene el restaurante, y poseen las finanzas específicas de cada una, y por ello es tan esencial en tantas funciones.

- Se relaciona con Plato, pues el menú se basa en objetos de esta clase.
- Se relaciona con Chef y Mesero, pues tiene arreglos para ambos roles.
- Se relaciona con Mesa, pues tiene una lista de objetos de esta clase.
- Se realiza con Reserva, pues crea instancias de esta clase y las guarda.

1. Clases y Objetos

- **Descripción:**
    - Las clases son la base de la POO y representan plantillas para crear objetos. Los objetos son instancias de clases que encapsulan datos y comportamientos.
- **Ejemplo en el proyecto:**
    - **Clase Cliente**: Implementa una plantilla para los clientes que realizan pedidos.
    - **Ubicación:** Archivo Cliente.java, desde la línea 1.

· **Captura:**

- **Uso:**
    - La clase Cliente permite manejar la información de los clientes (nombre, dirección, teléfono) y asociarla con pedidos específicos.

2. Encapsulamiento

- **Descripción:**
    - El encapsulamiento protege los datos al restringir el acceso a los atributos de las clases mediante modificadores de acceso (private, public) y el uso de getters y setters.
- **Ejemplo en el proyecto:**
    - Atributos privados y métodos de acceso en la clase Domicilio.
    - **Ubicación:** Archivo Domicilio.java, líneas 10 a 15 y 35 a 60.
- **Captura:**

- **Uso:**
    - Protege los datos internos del objeto Domicilio y permite acceder o modificar valores específicos solo mediante los métodos públicos, asegurando consistencia en el sistema.

3. Herencia

- **Descripción:**
    - La herencia permite que una clase (hija) extienda a otra clase (padre), reutilizando su funcionalidad y añadiendo características específicas.
- **Ejemplo en el proyecto:**
    - Clase Mesero que extiende Empleado.
    - **Ubicación:** Archivo Mesero.java, línea 5.
- **Captura:**
- **Uso:**
    - La clase Mesero hereda atributos y comportamientos de Empleado, y añade funcionalidades específicas, como la gestión de mesas.

4. Polimorfismo

- **Descripción:**
    - Permite que una misma operación se comporte de manera diferente según el objeto que la implemente.
- **Ejemplo en el proyecto:**
    - Métodos sobreescritos en la clase Domicilio.
    - **Ubicación:** Archivo Domicilio.java, método getCostoEnvio, línea 58.
- **Captura:**
- **Uso:**
    - Este polimorfismo asegura que el costo de envío se calcule dinámicamente según las propiedades del barrio asociado.

5. Abstracción

- **Descripción:**
    - La abstracción oculta detalles complejos, exponiendo solo lo necesario para interactuar con un objeto o proceso.
- **Ejemplo en el proyecto:**
    - Uso de la clase Zona para abstraer las características geográficas de los barrios.
    - **Ubicación:** Archivo Zona.java, línea 1Zona.
- **Captura:**
- **Uso:**
    - Zona permite representar de forma abstracta una región geográfica, asociándola con barrios y costos de envío.

6. Enumeraciones

- **Descripción:**
    - Las enumeraciones definen un conjunto fijo de valores constantes.
- **Ejemplo en el proyecto:**
    - EstadoPedido enumera los estados posibles de un pedido.
    - **Ubicación:** Archivo EstadoPedido.java, línea 3.
- **Captura:**
- **Uso:**

- Simplifica el manejo de estados, asegurando que un pedido solo tenga valores válidos.

Funcionalidades principales del programa

Antes de iniciar cada funcionalidad se llama a la clase DataManager, que es la base de datos en donde se encuentra la información serializada.

1. Abrir nueva sucursal.

   Ingreso:

   Primero se enseña el menú principal donde se muestra cada funcionalidad, y la opción de cerrar; al escoger la primera funcionalidad se pide un número de documento, al entregarse se inicia el método de clase “Administrativo”.

   VerificarAdmin(int cedula, dataManager): Busca en el arreglo de administradores si el documento de algunos coincide con el número ingresado, de no ser así se regresará al menú principal, si se encuentra un número coincidente, una variable empieza a apuntar al administrador encontrado y se pide que se ingrese una contraseña, si esta no coincide con la que tiene asignada el administrador se pide nuevamente, si se alcanzan tres intentos y sigue sin coincidir se cerrará el método y se retornará al menú principal, por el contrario, si la contraseña coincide se aprobará el ingreso y se enseña el menú de administración de las finanzas.

MenuFinanzas(dataManager): Esté menú se encuentra en la case Empresa, que es la clase destinada a administrar las finanzas y sucursales del restaurante, desde este menú se pueden acceder a las funciones financieras del programa, como ver las finanzas, ver las sucursales activas, o cerrar dichas sucursales, sin embargo, la función principal será abrir una nueva sucursal.

Interacción 1:

Al escoger abrir una sucursal, se inicia la primera interacción que es pedir, un préstamo, esta se hace con la intención de que no haya que arriesgar la sostenibilidad del restaurante de enfrentar un pago tan grande; para esto se inicia el método de la clase Banco, prestamo.

Antes de continuar es necesario apuntar que la clase Empresa tiene múltiples atributos estáticos que representan diferentes componentes de la actividad financiera, dos de estos son, deudas, que representa la cantidad de dinero que la empresa debe, y el otro es solvencia, que es un método, que divide el presupuesto de la empresa entre los gasto, ambos son importantes en esta interacción.

Prestamo(double deudas, double solvencia): Este método primero enseña las instancias que hay en el arreglo de bancos, enseñando cuánto es el mínimo que se garantiza que cada banco va a prestar, luego se pide que se elija uno, el banco elegido inicia el método de instancia aceptar.

Aceptar(double solvencia, double deudas): Cada banco tiene sus propias expectativas de qué tan sostenible es la entidad a la que le darán un préstamo, este método evalúa precisamente eso, evalúa si las deudas son demasiado altas, verificando que estas no sean mayores a 10 M, multiplicado por, 10 menos las expectativas del banco, estas son un entero del 1 al 9, también evalúa si la solvencia no es suficiente, asegurándose de que la solvencia no se manor a 1.1, si alguna de las evaluaciones falla, se retorna 0, en este caso se pediría que se escoja otra opción, por otro lado, si ambas son aceptadas, luego se calcula la capacidad crediticia de la empresa, que sería la solvencia menos 1, multiplicado por 10, si esta resulta siendo menor que las exigencias del banco, también se retornará 0, tras aceptar, el banco calcula cuánto será lo prestado, para esto toma el préstamo mínimo y se sacará una fracción mediante el siguiente cálculo:

valor agregado = préstamo mínimo * ((capacidad - exigencia)/10)

Y se retorna el préstamo mínimo más el valor agregado, tras esto ya no va a haber vuelta atrás, y se tendrá que culminar la función.

Prestamo(solvencia, deudas): luego se le pregunta al usuario a cuánto años desea adquirir el préstamo, este número debe estar entre 1 y 10, tras escoger, se calcula el interés del préstamo, para esto se multiplica el número de años, por el total de lo prestado, por 0.03, y se convoca al método de la clase empresa endeudar, que le suma a la deuda el valor del argumento que se pase, en este caso el total del

préstamo más el interés, y para acabar esta interacción se retorna el total de lo prestado.

Interacción 2:

Esta interacción se trata de escoger la ubicación de la nueva sucursal, asegurándose de elegir estratégicamente, para evitar hacerse autocompetencia.

ComprarTerreno(double presupuesto, dataManager): Este es un método perteneciente a la clase Barrio, este método en primer lugar, crea un arreglo temporal, llamado noHay, se recorre la lista de barrios llamada ciudad en el dataManager, para cada elemento se evalúa mediante el atributo booleano sucursal, si este ya tiene una sucursal, si el atributo es falso se añade al arreglo noHay, tras esto, se imprimen lo elementos que quedaron en este arreglo, y se piede que se escoja en cuál se abrirá la nueva sucursal, tras elegir, una variable llamada barrio empieza a apuntar al barrio elegido, luego se empieza a recorrer el arreglo esquinas, del barrio elegido, este contiene las esquinas de la enumeración con el mismo nombre, cuyas coordenadas cartesianas coinciden con las del barrio, al recorrer el arreglo por cada esquina se ejecuta el método de la clase Sucursal calcularDistancia pasando como argumentos las coordenadas de la esquina que son un arreglo de enteros, y la lista de sucursales del dataManager.

CalcularDistancia(int[] coordenadas, ArrayList<Sucursal> sucursales): este método recorre el arreglo pasado como argumento, en cada iteración toma la cada elemento del arreglo dirección, que son un par de enteros y los pone en una variable a cada uno, y hace lo mismo con las coordenadas de la esquina que se pasaron como argumento, una vez tiene los cuatro valores, se calcula la magnitud del segmento entre los puntos que generan ambas coordenadas, si en alguna iteración este valor es menor a 4, y se termina el ciclo, devolviendo falso, mientras que si en ninguna iteración este valor resulta siendo menor que 4, se retorna true.Este método se usa con la intención de que ninguna sucursal quede demasiado cerca de otra, por lo que se exige una distancia de mínimo 4 cuadras.

ComprarTerreno(presupuesto, dataManager): cada que el método devuelve falso se termina la iteración automáticamente, y cuándo devuelve verdadero, se añade a esquina correspondiente a la iteración a una arreglo temporal llamado locales, terminado el ciclo, se imprimen las direcciones de los locales obtenidos y se pide que se escoja uno, el escogido es asignado a una variable llamada esquina, y sus coordenadas a una llamada dirección, tras esto se inicia el método de la enumeración Esquina esqPer,(o esquinas permitidas).

esqPer(int[] coordenadas): Esto lo que hace es, dependiendo de las coordenadas, definir cuántos establecimientos pueden haber, juzgando si la esquina está frente al río(Representado por el eje de las y), si está en el límite entre dos barrios, en los límites de la ciudad, etc; de este modo dependiendo de estas evaluaciones se retornará 1, 2 o 4, que serán las opciones de locales ubicados en las coordenadas elegidas.

ComprarTerreno(presupuesto, dataManager): El retorno de este método se le asigna a una variable también llamada esqPer, y se crean una par de arreglos con cantidad de elementos igual a este valor, uno llamado valor y otro cantidad, después se inicia un ciclo por cada ubicación en el primer arreglo, en cada iteración se genera un número entre 100 M y la mitad del presupuesto y se asigna a una variable real, luego se genera otro número, en este caso, entre 15 y el número anterior dividido entre 150 M, y se redondea para poder ser asignado a una variable entera, normalmente se genera un número entre 25 y 50, pero como no se puede estar seguro, si el resultado es mayor a 50, se asigna a la variable el 50, antes de acabar la iteración, ambos valores serán añadidos a los arreglos valor y cantidad respectivamente, de modo que queden exactamente en la misma posición de sus respectivos arreglos, estos arreglos serán el precio del local y la cantidad de mesas que puede albergar, una vez terminado el ciclo se imprimen los elementos de ambos arreglos lado a lado, y se pide que se escoja uno, el local elegido tendrá un valor y espacio de ubicaciones similares, tras esto, el elemento de cantidad elegido se le asigna a una variable llamada espacio, se resta el valor del elemento elegido del arreglo “valor” al presupuesto(representando que se ha comprado ese local), también se restarán otros 10 M(representando que se compró lo necesario para crear la sucursal, como por ejemplo una cocina profesional), luego se cambia el atributo sucursal del barrio elegido anteriormente a verdadero, y se crea un objeto de clase Sucursal, la id será la cantidad de elementos en el arreglo de sucursales del dataManager, el nombre será el nombre del barrio seleccionado, la cantidad(o espacio) será el valor que hay en la variable espacio, la dirección serán las coordenadas de la esquina escogida, y el presupuesto lo que queda del presupuesto, para finalmente retornar esta sucursal.

Interacción 3:
La última interacción se trata de habilitar la sucursal para abrirse, en primer lugar se activa el método de la clase Sucursal comprarMesas.

ComprarMesas(): Este método inicia declarando una variable en cero, llamada compradas, e inicia un ciclo que no acaba hasta que compradas sea igual a la cantidad de elementos del arreglo mesas(el tamaño tanto de este arreglo como el de meseros es el espacio que se pasó como argumento al constructor del objeto), se imprimen las opciones de mesas que hay en el restaurante, basado en la capacidad que tienen, siendo de 4, 6 y 8, y el precio respectivo de cada opción, luego se pregunta de qué tipo se desean comprar, luego de elegir, se pregunta cuántas unidades se desean, si se escoge un número mayor a la capacidad de la sucursal, si se permite la cantidad “n” ingresada, se crean “n” mesas con la capacidad escogida, se resta al presupuesto el valor del tipo escogido por la cantidad, y se suma la cantidad elegida a la variable compradas, y termina la iteración, en cada iteración la cantidad máxima de mesas que se pueden comprar se reduce a la capacidad menos las mesas compradas, y se pregunta nuevamente por el tipo y la cantidad, una vez se hayan llenado todos los espacio se cierra el ciclo y se acaba el método.

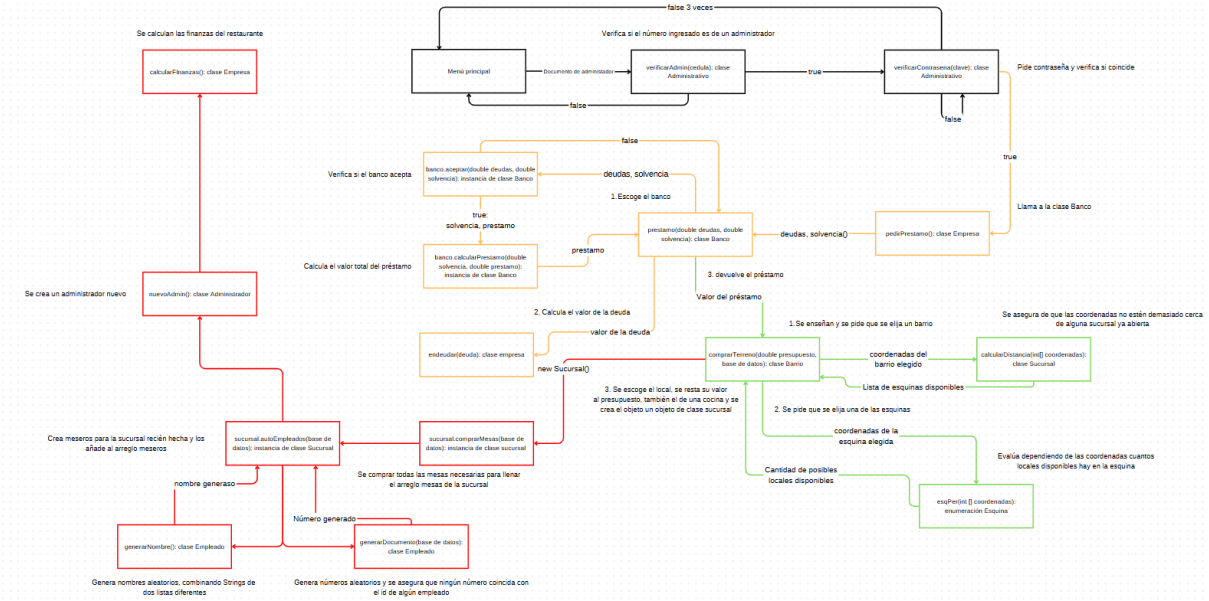
AutoEmpleado(): Este método lo que hace es generar 5 meseros básicos para la sucursal, la razón por la que se hace de forma simple es porque ya hay una función exclusivamente para contratar personal de un forma más personalizada, lo que se

hace es generar un número al azar y se recorren todas las listas del personal, si este número coincide con el id de alguno, se genera otro número y se vuelve a evaluar, una vez haya un número diferente al resto se genera un nombre, y se se crea un objeto de clase mesero, con el número generado para el id, el nombre para precisamente el nombre, su sucursal será la nueva sucursal, su antigüedad aparecerá como 1, su fecha de contratación aparecerá por defecto como 24/01/24, su última calificación 0, su disponibilidad verdadera y su sueldo será 1500000, se hace esto con cada mesero y se añaden al arreglo de meseros de la sucursal, finalmente se reduce el presupuesto de la sucursal en 90 M, que es el total del salario por un año que se le debe pagar a los meseros.

NuevoAdmin(dataManager): Se debe crear un nuevo administrador para que no hayan menos administradores que sucursales, para esto se genera un nombre y una id, de la misma forma que se hizo con los meseros, y se pide una contraseña para el nuevo administrador, se crea el objeto y se añade al arreglo de administradores del dataManager.

Tras la función.

CalcularFinanzas(): Luego de cada función se procede a calcular las finanzas de la empresa, la renta pasa a ser 10 M por la cantidad de sucursales, se recorre a todos los empleados sumando sus sueldos multiplicados por 12, y este será el valor de gastoSalarios, se suma el valor de gastoRecursos de todas las sucursales y su suma se añaden a gastoRecursos, lo mismo pasa con el presupuesto de cada sucursal, añadiendo estos valores a presupuestoBruto y el presupuestoTotal pasa a ser la resta del presupuesto bruto menos el gasto en recursos, la renta y las deudas, y tras esto se acaba la funcionalidad.



https://www.canva.com/design/DAGbwn7lvvY/m4VBn6RINOPM7L3DuQF10g/edit?utm_content=DAGbwn7lvvY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

2. Contratar Mesero.

Ingreso:

Al igual que en la función anterior, esta función pertenece al rol de administrador, por lo que también realiza la verificación con los métodos verificarAdmin y verificarContrasena.

MenuRecursosHumanos(): Esté menú perteneciente a la clase Restaurante, es el que permite gestionar al personal de los restaurantes, y realizar acciones como ver a los mesero contratados, despedir personal o contratar, en esta caso es esta última opción lo que nos interesa.

Interacción 1:

Lo primero que realiza esta funcionalidad es buscar vacantes en las diferentes sucursales del restaurante, para esto se inicia un ciclo en el que se evalúa cada sucursal del restaurante, para esto se recorre el arreglo de meseros de cada sucursal, en el momento en el que uno de los elementos sea nulo, se añade la sucursal a un arreglo llamado conEspacio.

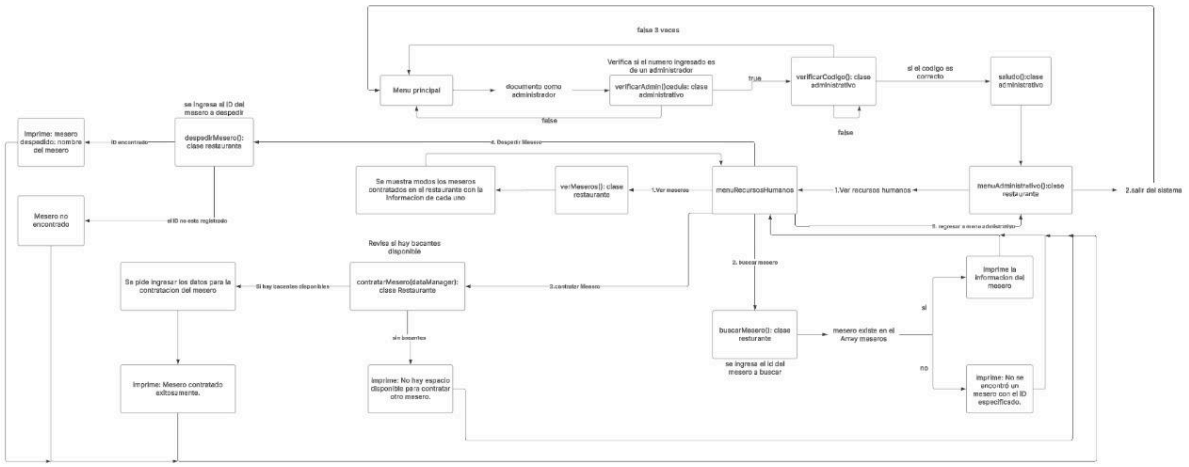
Interacción 2:

Tras hallar las posibles vacantes se procede a tomar la información del mesero, primero se pide la id, luego se evalúa entre todos los trabajadores del restaurante si la id coincide con alguno de ellos, de encontrarse alguna coincidencia se pedirá que se ingrese otra id, una vez hecho esto se pide el nombre del nuevo mesero, su dirección y su edad, luego se muestran los elementos del arreglo conEspacio, y se elija la sucursal en la que el novato va a trabajar, tras esto se piden sus años de experiencia y la fecha de contratación.

Interacción 3:

Luego de obtener todos los datos del mesero se crea un objeto de clase mesero, dependiendo de los años de experiencia que este tenga se calculará su salario y una vez hecho esto se activa el método de clase sucursal nuevoMesero.

NuevoMesero(Mesero mesero): Se resta el salario del mesero por 12 al presupuesto de la sucursal, y se define la disponibilidad como verdadera y se añade el mesero al arreglo de meseros de la sucursal.



https://lucid.app/lucidchart/2783e3ed-d550-4180-9237-8fccef8aa40f/edit?viewport_loc=-977%2C-197%2C4257%2C1710%2C0_0&invitationId=inv_85aef08b-9729-42a2-88c9-4aafbaf013be

3. Orden física.

Interacción 1:

Primero se crea un objeto de clase OrdenFisica, para esto se pide un documento y se verifica si este coincide con algún cliente que ya haya realizado órdenes en el restaurante, seguido de eso se muestran las sucursales en las que se podría estar realizando la orden y se le pide al usuario que escoja una, luego se busca en los arreglos de mesas y meseros alguno que esté disponible, y se asignan a la orden, finalmente se crea el objeto, tras esto se se activa el método hacerPedido.

HacerPedido(Sucursal sucursal): Este método primero enseña el menú disponible en la sucursal, y se le pide al usuario la cantidad de la cantidad de platos que se pedirán, una vez se enseña la cantidad que se pedirá, si este número es igual a o mayor a 6 se podrá elegir un plato, si este número es menor se inicia un ciclo con cantidad de iteraciones iguales al número ingresado , en el que en cada una se le pide al usuario que escoja uno de los platos enseñados en el menú, si en algún caso el número que se pide no coincide con la id de alguno de los platos, se pedirá que se ingrese de nuevo el número, si el número coincide se añade el plato con id coincidente un arreglo, luego se activa el método de clase Chef asignar con la sucursal del pedido como argumento.

Asignar(Sucursal sucursal): En este método se busca en el arreglo de la sucursal pasada como argumento algún chef, cuyo atributo disponible sea verdadero y se regresa a ese chef y se crea un objeto de clase pedidoFisico.

Interacción 2:

Facturación(PedidoFisico pedido): Se calcula la factura de la orden, para esto se verifica una puntuación que posee el cliente, esta puntuación es un entero que se define si el cliente tiene descuentos, luego de tener la puntuación definida se calcula cuánto se le sumará a la puntuación, esto depende de cuánto sea el valor de la compra, tras esto, se verifica si el cliente tiene suficiente puntos para obtener un descuento, si es así, reduce el precio en un porcentaje definido por la cantidad de puntos.

Interacción 3:

Se llama al método de clase Cliente darCalificacionM(), en donde se pide que se le de una calificación al mesero, esta estará entre 1 y 5, dependiendo de este numerosos se le sumarán o restarán puntos al mesero, si estos suben, se sumará un monto al salario del mesero.

Luego sucede lo mismo con el chef mediante el método darCalificacionC().

4. Pedir domicilios.
   Esta función trata de gestionar y actualizar la sección de pedidos.

   Interacción 1:

   realizarPedido(): Tras iniciar la función se dan las opciones de crear un pedido nuevo o ver alguno ya existente, en este caso lo que hay que elegir es la primera opción, al hacerlo se pide un número de identidad y esto abre paso al método seleccionarOcrearCliente.

   SeleccionarOcrearCliente(Scanner): Se busca en el historial de clientes si la id de alguno coincide con alguno, de no ser así se procede a crear un cliente nuevo, para esto se recopila información acerca del usuario, pidiendo nombre y se activa seleccionarBarrio().

   seleccionarBarrio(): Para crear un nuevo cliente es necesario verificar en qué localización va a recibir las entregas, es por esto que se pide que se ingrese uno de los barrios con cobertura del restaurante, y se retorna el barrio elegido.

   Se crea el nuevo cliente y se envía a la gestión de pedidos.

   Interacción 2:

   Una vez creado el usuario, se realiza el pedido, para esto se inicia en la gestión de pedidos, donde se realizan varias opciones, eligiendo la primera, que es la de realizar un pedido inicia esta segunda interacción.

   CrearNuevoPedido(Scanner): Tras ingresar el usuario se empieza el método

SeleccionarProductos(Scanner): Primero busca los productos disponibles para domicilios en la base de datos e imprime las opciones con sus precios, y se pide indefinidamente que se seleccione una opción, si el usuario pone 0 se termina la selección de productos, cada que el usuario elige un producto se añade a un arreglo, y si al acabar este arreglo está vacío se dice que no se pidió ningún pedido y se regresa al menú.

tras tener los datos se llama al método de la clase gestorPedidos crearPedido.

crearPedido(Cliente cliente, List<Producto> productos, Barrio barrioSeleccionado): En primer lugar se importa la hora del dispositivo y se verifica si el pedido se está realizando a una hora de servicio, entre las 8 de la mañana y las 10 de la noche, si se está fuera de hora no se permitirá continuar con la función, por otro lado, si se está en la hora correcta se procede a buscar en el arreglo de repartidores si hay alguno disponible, si ninguno lo está se interrumpe el proceso y no será posible continuar con la funcionalidad, de haber alguno, se verifica si el barrio del pedido está entre sus barrios asignados, de no ser así se continúa buscando y si ninguno lo posee se acaba la función, si se encuentra un repartidor que esté disponible y si tenga el barrio elegido asignado, se crea un objeto de clase Domicilio con el barrio y el repartidor seleccionado, el estado por defecto será RECIBIDO, y el tiempo estimado de entrega será la hora al a que se realiza el pedido más 30 minutos, este domicilio representará la casa a la que será enviado el pedido, luego se recorre el arreglo de pedidos que hay en la base de datos contando la cantidad de pedidos que se han realizado, esta cantidad más 1, será la id del nuevo pedido y se crea finalmente el objeto de clase pedido, pasando la id, el domicilio, los productos y el cliente, al inicializarse el pedido, se calculan los costos, sumando el valor de todos los productos y sumándole el costo de envío, el cuál depende del barrio al que será enviado, y su estado será por defecto, tras eso se pasa al método aplicarDescuentos.

AplicarDescuentos(Pedido pedido, Cliente cliente): En primer lugar se importa la fecha en la que se está realizando el pedido y se calcula cuántos días hacen parte del último mes, una vez hecho esto, se recorre todo el historial de pedidos buscando si alguno del usuario que está creando el nuevo pedido es del último mes, si no, no pasa nada, mientras que si se encuentra alguno se resta un 10% del valor total al pedido.

esHoraPico(): Tras esto se verifica se la hora en la que se realiza el pedido es hora pico, entre las 12 del día y las 2 de la tarde o entre las 7 y las 9 de la noche; de ser así se aumenta en un 20% el valor del pedido.

Interacción 3:

Tras la creación del pedido su estado será por defecto RECIBIDO, y la última interacción será la modificación del pedido dependiendo del transcurso del envío, las opciones que se dan son modificar el estado del pedido a “en preparación”, “en camino”, “entregado o cancelado”, también cambiar el repartidor, pedir más

productos, cancelar el pedido y marcar incidencias, como retraso, producto en la estado o cliente ausente al momento de la entrega.

5. Realizar reserva:

   Interacción 1:

   menuReserva(dataManager): Lo primero que se realiza es mostrar las opciones de gestión de reservas, en este caso se elige crearReserva, activando el método tomarDatosResreva()

   TomarDatosReserva(dataManager): Lo que hace es método es pedir los datos de la reservación que se va a realiza, que serían la fecha, la hora y la cantidad de personas que asistirán, si la cantidad de personas es mayor a 8 se pedirá que se vuelva a ingresar el número, pues esa es la máxima cantidad de asientos que tiene una mesa.

   Interacción 2:

   CrearReserva(String fechaHora, int cantidadPersonas, dataManager):Se enseñan las sucursales y se pide que se elija en cuál realizará la reservación, tras elegirse se empieza a recorrer el arreglo de mesas de la sucursal verificando que no esté reservada y que tenga la suficiente cantidad de asientos para todos los participantes.

   Interacción 3:
   Se crea la reservación y se añade a la lista de reservaciones de la sucursal, dando la opción de en un futuro poder ser aplazada o cancelada.

manual de usuario

funcionamiento de la aplicación

1) al iniciar la aplicación debe de seleccionar entre todas las posibles acciones que se le ofrecen la que quiera quiera para esto deberás de seleccionar el número mostrado antes del nombre de la acción por ejemplo vas a ver esto:
   1)ver sucursales
   2)contratación
   3)pedir orden fisica
   4)pedir domicilio
   5)Realizar reservación
   en esta interfaz deberás de elegir el número de la opción que quiera realizar por ejemplo, si lo que desea hacer es abrir una ver sucursal deberá ingresar el número 1 únicamente, sin letras paréntesis ni nada que lo acompañe y darle al enter se le iniciará la acción de ver sucursales.

   Una vez ingresado hayas ingresado en ver sucursales se le pedirá un número de documento para ingresar, una vez ingresado, se verificará que el documento de identidad suministrado concuerdo con el de un administrador

(cédulas por defecto de administrador, contraseñas 4488123(la contraseña es la misma para todos))

1. 12345
2. 10453
3. 42012

)

autorizado si es así se continuará con normalidad y de no coincidir se te notificará con un mensaje en la pantalla y se te regresará al menú.

en caso de coincidir se te dará la bienvenida con el nombre guardado en la cuenta de admin y se te mostrar el menú de finanzas, en este debes proseguir como en el menu principal ingresando únicamente el número de la opción sin acompañamientos

1) ver finanzas generales (en esta se te mostrará la siguiente información)
    a) Deudas crediticias
    b) renta
    c) gasto en salarios
    d) gasto en recursos
    e) capital bruto
    f) capital neto
    g) solvencia
2) ver sucursales (en este se te mostrarán todas las sucursales guardadas en la base de datos)
3) abrir sucursal(en esta se pedirá un préstamo en el cual se analiza la solvencia y la deuda total de la empresa, sin ningún préstamo es aprobado se mostrará en pantalla el mensaje No se ha concretado ningún préstamo. Si el préstamo es aprobado entonces en automático se habilitaran las acciones comprar terreno, se compraran las mesas y el iniciara una función de conseguir empleados)
4) cerrar sucursal(le muestra la lista de sucursales existente este tambien funciona como los menus anteriores y si selecciona un numero que no haga parte del menu no se cerrara ninguna sucursal y solo queda una sucursal no se permitirá borrarla atrás de esto se cobrara una liquidacion por las mesas existentes en la sucursal y los salarios de los empleados y por ultimo se eliminara la sucursal del arreglo de sucursales)
5) pagar deudas(al seleccionar esta opcion se analizara si es posible retirar financiamiento de las sucursales existentes buscando un minimo de 40 millones como presupuesto por sucursal esto lo hace pero si no es posible retirar presupuesto y que esta sucursal quede con uno de 40 millones como minimo no se le retirara presupuesto y lo que se logre descontar se le restara a las deudas de la empresa)
6) Salir(sale al menu principal)

Contrataciones

esta función al igual que la anterior pide documento y contraseña de administrador al avanzar abre el menú administrativo donde puedes regresar o entrar a los recursos humanos.

al entrar a recursos humanos muestra un menu con 5 opciones.

1)ver meseros(muestra a los meseros contratados en la empresa)
2)buscar mesero(esta opcion pide que ingrese el id de algun mesero(los id de los meseros se puede conocer con la primera opcion), si se encuentra el mesero que coincida con esa id te mostrará toda su información guardada en la base de datos, de no encontrarse se mostrar un mensaje en pantalla que dice no se encontró mesero compatible)
3)contrar mesero (pide un id y se asegura de que no sea uno repetido, luego se pide que se ingrese el nombre del mesero, su direccion, se ingrese su edad, muestra la lista de sucursales con cupos para nuevos meseros y se pide que elija uno de la misma forma que en los menus anteriores , pide los años de experiencia del mesero y su fecha de contratacion)
4)despedir mesero (pide id y despide al mesero de esa id)
5)salir al menu

Crear orden fisica.

en esta función se cargará al usuario como cliente pidiendo su id en caso de no encontrar uno con esta id se va a crear uno nuevo con esta id y se perdieran los siguiente datos

- nombre
- telefono

después de cargarlo como cliente se le pedirá que ingrese una sucursal de la misma manera que los anteriores menus, si elije un numero no existente se le mostrara esa opcion como no esta disponible.

al elegir una sucursal se pediera que ingrese la cantidad de personas que hay presentes incluyendo al cliente y hay que ingresar un numero entero.

tras esto se le asignara una mesa que cumpla con la cantidad de asientos minima para la cantidad antes ingresada.

pide la cantidad de platos a ordenar

enseña el menu y pide que se elija un plato del menu y este proceso se repite el numero de veces que se ingreso anteriormente.

luego se enseña la factura y se pide al usuario que califique el servicio que se le dio tras eso se guardan los datos automáticamente y se cierra el programa.

Pedir domicilios

este abre un menu con dos opciones

- Realizar nuevo pedido (se pide el id del cliente y si no coincide con la de un cliente ya existente se crea de la misma manera que en la función anterior luego confirma

que este en horario de atencion, después muestra una lista de los productos disponibles en dolares luego se pide que se ingrese el id del pedido en caso de ingresarse 0 se finaliza la elección de productos luego pide que se ingrese el barrio al cual sera enviado con el precio del envio a cada barrio se muestra el tiempo estimado de entrega)

- Ir al menu de gestion(se abre otra menu con 3 opciones nuevas)
    - ver pedidos
    - modificar un pedido
    - cancelar pedido

realizar reservación

En esta se abe otro menu con tres opciones

- realizar reserva(primero pide la fecha y la hora, luego la cantidad de personas a asistir luego las sucursales y pide que se elija una y hace lo mismo que en la tercera funcion de encontrar una mesa)
- aplazar reserva
- cancelar reserva