TP Integrador: Aplicación de delivery.

El objetivo del presente proyecto es construir el MVP de una aplicación que permite gestionar delivery de comida de restaurantes y rotiserías.

La aplicación tiene dos tipos de usuarios, vendedores (los restaurantes o rotiserías) y compradores.

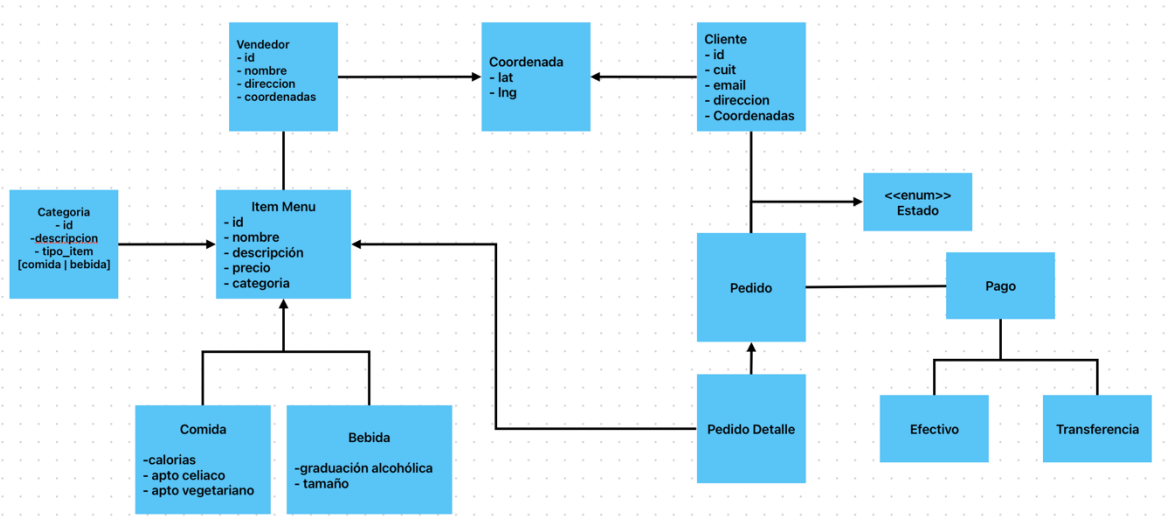
Los vendedores pueden ingresar al sistema para

- Gestionar sus datos.
- Gestionar menú de platos y bebidas que ofrecen
- Revisar la lista de pedidos recibidos por parte de los clientes.
- Actualizar el estado de un pedido, desde que es recibido, aceptado, preparado, hasta enviado.

Los compradores pueden ingresar al sistema para

- Gestionar sus datos
- Crear un pedido indicando para un restaurante en particular que seleccione, cuales platos solicitará y la forma de pago. El precio de un pedido se calculará diferente según la forma de pago.
- Ver su historial de pedidos.

Modelo de datos



Etapa 1: Modulo de gestión del Vendedor y Gestion Cliente

- Objetivo: Permitir que un vendedor o un cliente gestione sus datos.
- Temas aplicados: Introduccion a java, estructura de una clase, modificadores de acceso, clases principales (String, System), asociación entre clases.

Descripción:

- Crear las clases Cliente, Vendedor y Coordenada.
- Crear un método main que cree 3 instancias de vendedores y agregarlos a un arreglo.
- Iterar sobre el arreglo para buscar vendedores por nombre o por id.
- Eliminar un vendedor del arreglo.
- Crear un método main que cree 3 instancias de Clientes y agregarlos a un arreglo.
- Iterar sobre el arreglo para buscar Clientes por nombre o por id.
- Eliminar un cliente del arreglo.
- Agregar a vendedor un método "distancia" que reciba un Cliente como argumento y calcule la distancia en kilómetros entre ellos (usar una implementación de la fórmula de Haversine).

Etapa 2: Modulo de gestión del Menú

- Objetivo: Permitir que un vendedor gestione el menú de comidas.
- Temas aplicados: Herencia, sobrescritura, sobrecarga.

Descripción

- Crear la clase abstracta ItemMenu y las subclases concretas "plato" y "bebida".
- La clase ItemMenu debe tener el método abstracto "peso" y los métodos abstractos "esComida", "esBebida" y "aptoVegano".
- El método "peso" es utilizado para calcular el peso de un pedido. En los alimentos se guarda directamente como un atributo y se le suma un 10% más por los envases. En las bebidas se calcula como el producto del volumen en ml por 0.99 si es una bebida con alcohol y 1.04 si es una bebida gaseosa. Además, se le suma un 20% de peso por envases. Los otros métodos indican si el alimento es una comida o una bebida y en caso de ser comida si es aptoVegano.
- Modificar la clase Vendedor y agregarle una lista de ItemsMenu donde cada vendedor tendrá la lista de los platos que ofrece.
- Agregar a la clase vendedor métodos getItem que retornen:
    - Lista Bebidas
    - Lista Comidas
    - Lista Comidas Veganas
    - Lista Bebidas sin Alcohol

Etapa 3: Modulo de gestión de Búsqueda

- Objetivo: Realizar búsquedas sobre los ítems de menu.
- Temas aplicados: Interfaces, streams y lambda, excepciones.

Descripción

- Crear la interface "ItemsPedidoDao" que tendrá diversos métodos para realizar búsquedas.
- Crear una clase ItemsPedidoMemory que implemente ItemsPedidoDao y que realizará la búsqueda de pedidos por diversos criterios usando el API de Streams
    - Filtrado
    - Ordenar por criterios
    - Búsqueda por rango de precios.
    - Buscar por restaurante
- Cuando no se encuentre un dato lanzará un "ItemNoEncontradoException".

Etapa 4: Modulo de gestión de Creación de pedido

- Objetivo: Crear un pedido como cliente.
- Temas aplicados: Patron strategy y gestión de excepciones

Descripción

- Un cliente busca ítems de un vendedor y genera un pedido.
- El pedido solo puede contener ítems de un vendedor
- El cliente debe determinar la forma de pago
    - Si es mediante transferencia debe informar cual es su cuit y su cbu
    - Si es mediante MercadoPago debe informar su alias
- El precio del pedido es el precio de los productos y se aplica un recargo por la forma de pago
    - MercadoPago recarga el 4%.
    - Transferencia recarga el 2%.
- El pedido se queda con el estado RECIBIDO.

Etapa 5: Modulo de Actualización de estados.

- Objetivo: Buscar un pedido y actualizar el estado
- Temas aplicados: Patron observer

Descripción

- Un vendedor puede buscar todos los pedidos que le llegaron por estado y puede actualizar el estado.
- Implemente el patron observer de forma tal que el cliente se subscriba a observar el estado de los pedidos y cuando un vendedor cambie el estado reciba esta información.
- Cuando el cliente recibe un mensaje que el pedido ha sido aceptado y esta en estado EN_ENVIO genera un pago. (crea un objeto con los datos del pago, fecha, monto, etc).

Etapa 6: Crear UI Swing.

- Objetivo: Crear interfaces en SWING

Descripción

- Crear una pantalla para buscar, editar, crear o eliminar Vendedores
- Crear una pantalla para buscar, editar, crear o eliminar Clientes
- Crear una pantalla para buscar, editar, crear o eliminar ItemsMenu
- Crear una pantalla para buscar, editar, crear o eliminar Pedidos

Etapa 7: Conexión a base de datos y Unit Testing

- Objetivo: Persistir información en la base de datos
- Temas aplicados: Conexión a base de datos. Patron Singleton. Patrón Factory/Builder.

Descripción

- Crear la interface DAO para cada uno de los datos del dominio del problema que se persistirán en la base de datos y la implementación a la base de datos.
- Guardar utilizando las interfaces confeccionadas en SWING:
    - Vendedores
    - Clientes
    - ItemsPedido
    - Pedidos
    - Pago

Etapa 8: SpringBoot

- Objetivo: implementar el tp en SpringBoot
- Temas aplicados: Sprinboot

Descripción

- Implementar TP en Springboot.
- Realizar las interfaces de consulta a través de html.