MODELO DE LA ESTRUCTURA

Agrupación de las Historias de Usuario por Características o Épicas

Épica 1: Gestión de Usuarios

HU1 Registro de Usuario: Permitir que los clientes creen cuentas para facilitar pedidos recurrentes y seguimiento personalizado.

Épica 2: Gestión de Productos

HU2 Búsqueda de productos: Facilitar la búsqueda rápida y eficiente de productos por nombre o categoría.

Épica 3: Pagos y Transacciones

HU3 Pago en línea: Ofrecer a los usuarios métodos seguros y rápidos para realizar pagos digitales.

Épica 4: Administración del Inventario

HU4 Gestión de inventario: Permite al administrador actualizar y controlar fácilmente el inventario en tiempo real.

Épica 5: Comunicación con Clientes

HU5 Notificaciones de envío: Informar automáticamente al cliente sobre el estado de su pedido mediante notificaciones.

Identificación y Selección de Abstracciones

1- Para definir las clases del sistema, aplicamos los siguientes criterios de selección de abstracciones:
2- **Relevancia en el Dominio:** Se identificaron los elementos clave del sistema: Usuario, Producto, Inventario, Pedido, MetodoDePago, Notificación y Administrador.
3- **Encapsulación:** Se agruparon atributos y métodos en clases que representan entidades del negocio con sus responsabilidades bien definidas.
4- **Baja Acoplamiento y Alta Cohesión:** Se definieron clases con funcionalidades específicas, minimizando la dependencia entre ellas.
5- **Reutilización y Extensibilidad:** Se consideró la posibilidad de reutilizar clases en futuras expansiones del sistema.

Aplicamos los criterios vistos en clase (relevancia, encapsulación, cohesión, reutilización):

Abstracción Inicial	Criterio de Selección	Decisión
Usuario	Alta Relevancia	Mantener
Producto	Alta Relevancia	Mantener
Pedido	Alta Cohesión	Mantener
Método de Pago	Reutilización	Mantener
Inventario	Baja dependencia	Mantener
Notificación	Comunicación clave	Mantener
Administrador	Baja cohesión	Fusionar con Usuario

Conclusión de depuración:
La abstracción "Administrador" se fusiona en "Usuario" como rol específico.

Ejemplo del Proceso de Selección:

6- Se identificó inicialmente la necesidad de gestionar claramente los procesos relacionados con **pedidos, pagos, inventario, usuarios y comunicación con el cliente**.

7- A partir de esto, se definieron inicialmente las abstracciones **Pedido**, **MétodoDePago**, **Usuario**, **Producto**, **Inventario** y **Notificación**.

8- Posteriormente, aplicando los criterios vistos en clase (relevancia en el dominio, encapsulación, cohesión, reutilización y extensibilidad), se validó que las abstracciones seleccionadas tenían las siguientes relaciones necesarias:
   - **Pedido** requiere interacción directa con **Usuario**, **Producto**, **MétodoDePago** y **Notificación**.

    - **Producto** requiere interacción directa con **Inventario** para gestionar su disponibilidad
9- La abstracción **Administrador** fue fusionada con la clase **Usuario**, definiendo roles específicos en la misma.
10- Finalmente, se establecieron relaciones precisas entre las clases seleccionadas, asegurando alta cohesión y bajo acoplamiento, reflejando claramente la operación real del negocio de la pizzería.

Fichas CRC

Clase: Usuario
Responsabilidades: Registrar usuarios, autenticar usuarios, actualizar datos personales
Colaboradores: Pedido, Notificación, Método de Pago
Atributos: idUsuario, nombre, email, contraseña, rol
Métodos: registrarUsuario(), autenticar(), validarDatos()

Clase: Pedido
Responsabilidades:
Crear pedido, actualizar estado del pedido, cancelar pedido
Colaboradores: Usuario, Producto, Método de Pago, Notificación
Atributos: idPedido, fecha, estado, total

Métodos: crearPedido(), actualizarEstado(), cancelarPedido()

Clase: Producto
Responsabilidades: Mostrar detalles del producto, facilitar búsqueda
Colaboradores: Inventario
Atributos: idProducto, nombre, descripción, precio, categoría
Métodos: buscarProducto(), mostrarDetalles()

Clase: Inventario
Responsabilidades: Controlar existencias, actualizar stock
Colaboradores: Producto
Atributos: idInventario, cantidadDisponible
Métodos: actualizarStock(), verificarDisponibilidad()

Clase: MetodoDePago
Responsabilidades:

Procesar pagos, verificar transacciones
Colaboradores: Pedido
Atributos: idMetodoPago, tipo, estado
Métodos: procesarPago(), verificarPago()

Clase: Notificación
Responsabilidades: Enviar notificaciones, comunicar estado del pedido
Colaboradores: Pedido, Usuario
Atributos: idNotificacion, mensaje, fechaEnvio
Métodos: enviarNotificacion(), obtenerHistorial()

Diseño de las clases

class Usuario{

- idUsuario:int
- nombre:String

class Pedido{

- idPedido:int
- fecha:Date

- email:String
- contraseña:String
- rol:String
+ registrarUsuario():void
+ autenticar():boolean
+ validarDatos():boolean
}

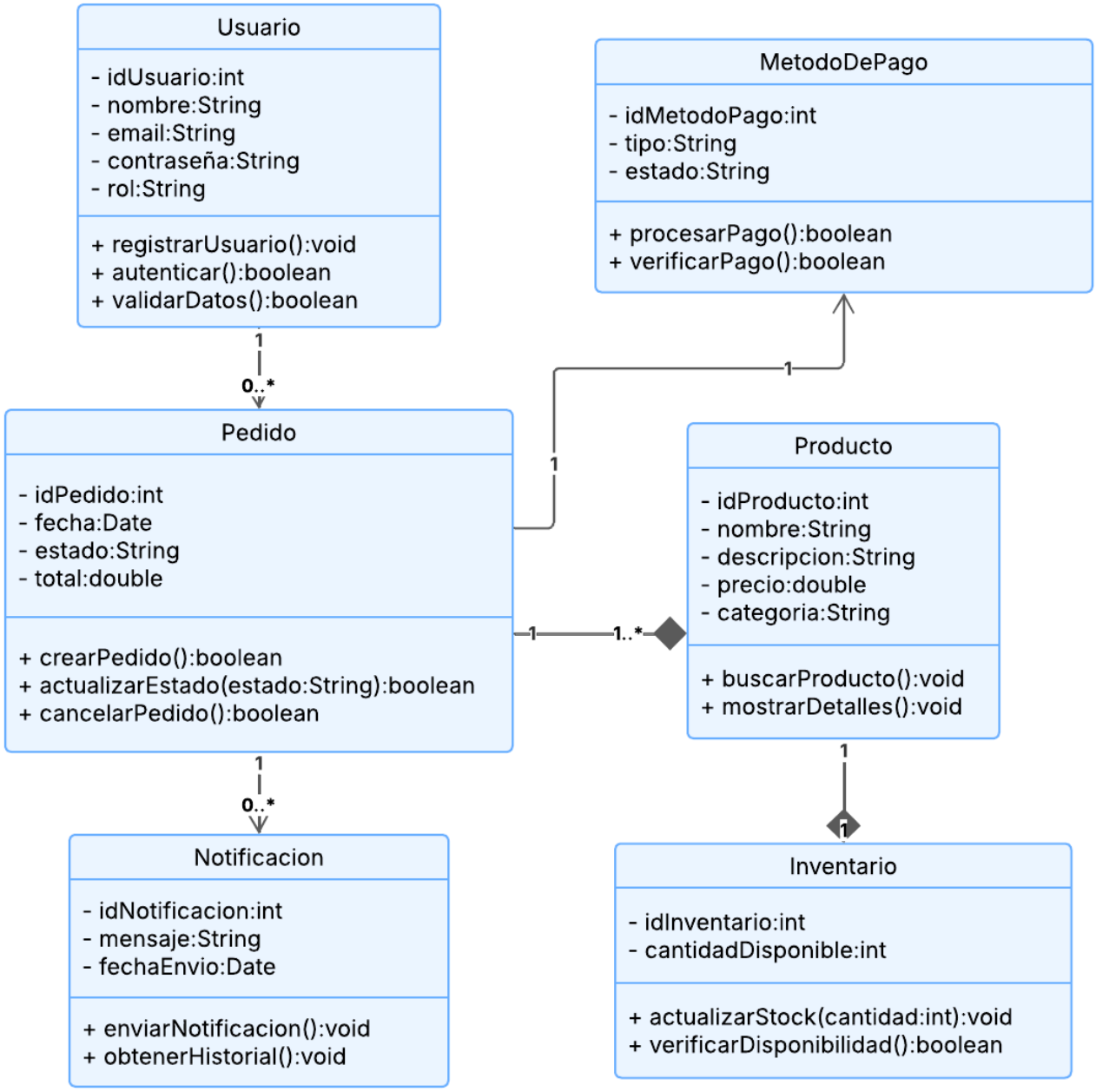
class MetodoDePago {
- idMetodoPago:int
- tipo:String
- estado:String
+ procesarPago():boolean
+ verificarPago():boolean
}

class Inventario {
- idInventario:int
- cantidadDisponible:int
+ actualizarStock(cantidad:int):void
+ verificarDisponibilidad():boolean
}

- estado:String
- total:double
+ crearPedido():boolean
+ actualizarEstado(estado:String):boolean
+ cancelarPedido():boolean
}

class Producto {
- idProducto:int
- nombre:String
- descripcion:String
- precio:double
- categoria:String
+ buscarProducto():void
+ mostrarDetalles():void
}

class Notificación {
- idNotificacion:int
- mensaje:String
- fechaEnvio:Date
+ enviarNotificacion():void
+ obtenerHistorial(): void
}


Usuario
- idUsuario:int
- nombre:String
- email:String
- contraseña:String
- rol:String
+ registrarUsuario():void
+ autenticar():boolean
+ validarDatos():boolean
MetodoDePago
- idMetodoPago:int
- tipo:String
- estado:String
+ procesarPago():boolean
+ verificarPago():boolean
Pedido
- idPedido:int
- fecha:Date
- estado:String
- total:double
+ crearPedido():boolean
+ actualizarEstado(estado:String):boolean
+ cancelarPedido():boolean
Producto
- idProducto:int
- nombre:String
- descripcion:String
- precio:double
- categoria:String
+ buscarProducto():void
+ mostrarDetalles():void
Notificacion
- idNotificacion:int
- mensaje:String
- fechaEnvio:Date
+ enviarNotificacion():void
+ obtenerHistorial():void
Inventario
- idInventario:int
- cantidadDisponible:int
+ actualizarStock(cantidad:int):void
+ verificarDisponibilidad():boolean
1
0..*
1
1
1
1..*
1
0..*
1
1