Institución:	
Carrera:	Licenciatura en Ing. De Software
Ciclo:	2025-I
Grupo:	1101
Asignatura:	Diseño de software
Título de actividad:	Estándar IEEE 1016 – Lógico y Dependencias
Integrantes:	Ramírez García Oswaldo Rios Carrera Jesús Vicente Vargas Angeles Uriel
Matriculas:	19-011-1318 19-011-0599 21-011-1167
Profesor:	Máximo Eduardo Sánchez Gutiérrez

Índice

5.4 Lógica del sistema.

El punto de vista lógico del sistema representa la organización funcional de los principales módulos que lo componen. En este caso, se documenta un Sistema de Comanda Digital para restaurantes que contempla distintos roles y funciones esenciales para su operación. Este diseño busca establecer una arquitectura clara y modular que facilite el flujo de pedidos, mantenga actualizado el inventario y genere informes útiles para la toma de decisiones.

El sistema se divide en módulos lógicos alineados con las responsabilidades de los usuarios del restaurante. El módulo de usuario incluye a clientes, cocineros y administradores. Permite a los clientes registrarse, consultar el menú y realizar pedidos, mientras que a los administradores les brinda herramientas para gestionar las comandas y supervisar la operación general del sistema.

El módulo de pedidos facilita el proceso mediante el cual los clientes solicitan sus platillos. Cada pedido cambia de estado según su avance, desde "En espera" hasta "Listo". Los administradores pueden modificar pedidos si es necesario y los cocineros acceden únicamente a la visualización de los mismos desde su estación.

El módulo de inventario controla los ingredientes y productos disponibles, actualizándose automáticamente cuando se confirma un pedido. Los administradores pueden modificar existencias o añadir nuevos productos según se requiera.

El módulo de informes recopila datos sobre ventas, pedidos e inventario, proporcionando información clave para evaluar el rendimiento del restaurante y respaldar decisiones estratégicas.

El módulo de ticket genera un comprobante para cada pedido. Aunque el sistema no incluye pagos, este módulo proporciona al cliente un desglose detallado como constancia de su solicitud.

La interacción entre módulos sigue una lógica de colaboración que asegura la coherencia del sistema y una experiencia fluida. Al crear un pedido, este se refleja en el módulo de pedidos y actualiza automáticamente el inventario. Los

administradores pueden gestionarlo y los cocineros solo lo visualizan. Una vez confirmado, se genera un ticket para el cliente. A su vez, los datos de pedidos e inventario alimentan el módulo de informes para generar reportes útiles. Esta organización lógica mejora la eficiencia, reduce errores y centraliza el control de los recursos.

5.4.1 Puntos clave a considerar del sistema.

Este apartado identifica las principales preocupaciones del diseño que deben abordarse para lograr un sistema funcional, eficiente y fácil de mantener.

Entre las preocupaciones clave se encuentra la modularidad y separación de responsabilidades. Cada módulo debe tener una función bien definida para evitar dependencias innecesarias. Por ejemplo, el módulo de pedidos no debe realizar tareas propias del inventario o de los informes.

También se debe cuidar el acceso y los permisos. Cada tipo de usuario debe tener acceso solo a las funciones que le corresponden. Los cocineros pueden visualizar pedidos, pero no modificarlos. Los clientes no deben alterar un pedido una vez enviado y los administradores cuentan con permisos avanzados para supervisar el sistema.

Otra preocupación importante es la actualización en tiempo real. Los pedidos deben reflejarse de inmediato en la pantalla del cocinero y el inventario debe ajustarse automáticamente al registrar una nueva comanda, asegurando que los datos estén siempre sincronizados.

La arquitectura debe permitir escalabilidad y mantenimiento. Esto implica poder añadir nuevas funciones sin afectar lo existente y modificar elementos como el menú sin alterar la lógica principal del sistema.

Por último, es esencial la generación de tickets e informes. Cada pedido debe contar con su respectivo comprobante y los informes deben ofrecer datos claros y relevantes para apoyar la gestión del restaurante.

Desde el punto de vista técnico, se utilizará PHP para la lógica del servidor, una base de datos SQL para gestionar pedidos, usuarios e inventario, y se procurará que la interfaz sea simple y accesible para todos los perfiles de usuario.

5.4.2 Módulos del sistema.

El sistema de gestión de comandas se estructura bajo una arquitectura modular que organiza sus principales funciones en componentes interconectados. Esta división permite una operación eficiente y facilita tanto el mantenimiento como la escalabilidad del sistema. A continuación, se describen los elementos centrales del diseño.

Módulo de Usuario.

Este módulo permite gestionar a los diferentes tipos de usuarios del sistema: clientes, cocineros y administradores. Los clientes pueden consultar el menú y realizar pedidos; los cocineros acceden a la visualización de comandas, y los administradores tienen acceso completo para gestionar pedidos, inventario e informes. Se incorpora un sistema de control de acceso que garantiza que cada perfil solo pueda realizar las acciones que le corresponden.

Módulo de Pedidos y Comandas.

Facilita a los clientes la creación de pedidos, que avanzan por distintos estados a medida que se procesan. Los administradores pueden modificar los pedidos antes de su entrega, y los cocineros tienen acceso a la visualización en tiempo real. Este módulo también interactúa con el inventario y genera el ticket correspondiente al finalizar el proceso.

Módulo de Inventario.

Contiene el registro de los ingredientes y productos disponibles. El sistema actualiza automáticamente las existencias al confirmarse un pedido. Solo los administradores pueden modificar el contenido del inventario, lo que asegura un control centralizado y preciso de los insumos.

Reúne datos relacionados con los pedidos, el uso del inventario y las ventas realizadas. Estos informes permiten evaluar el desempeño del restaurante y sirven como base para tomar decisiones estratégicas a nivel administrativo.

Módulo de Ticket.

Emite comprobantes detallados por cada pedido, mostrando los platillos seleccionados y el total a pagar. Aunque el sistema no contempla pagos en línea, el ticket proporciona una constancia clara para el cliente y puede incluir códigos o referencias para su seguimiento.

Estados de los Pedidos y su Gestión.

Los pedidos siguen una secuencia lógica de estados que permite controlar su ciclo de vida de forma eficiente:

- Pendiente: el cliente ha creado el pedido, pero aún no ha sido enviado a cocina.
- En preparación: el pedido ha sido aceptado y está en proceso de elaboración.
- Listo para entrega: el platillo ha sido terminado y está listo para servir.
- Entregado: el cliente ha recibido el pedido.

El sistema contempla reglas específicas para el manejo de estos estados. Por ejemplo, los clientes pueden cancelar un pedido mientras no se haya generado el ticket; los cocineros solo visualizan los pedidos en preparación; los administradores pueden modificarlos en cualquier fase, excepto cuando ya han sido entregados. La generación del ticket se habilita una vez que el pedido ha sido registrado y confirmado para entrega.

Relaciones entre los Módulos.

Cada módulo interactúa con otros para garantizar un flujo de trabajo eficiente.

Módulo	Interacción con	Descripción

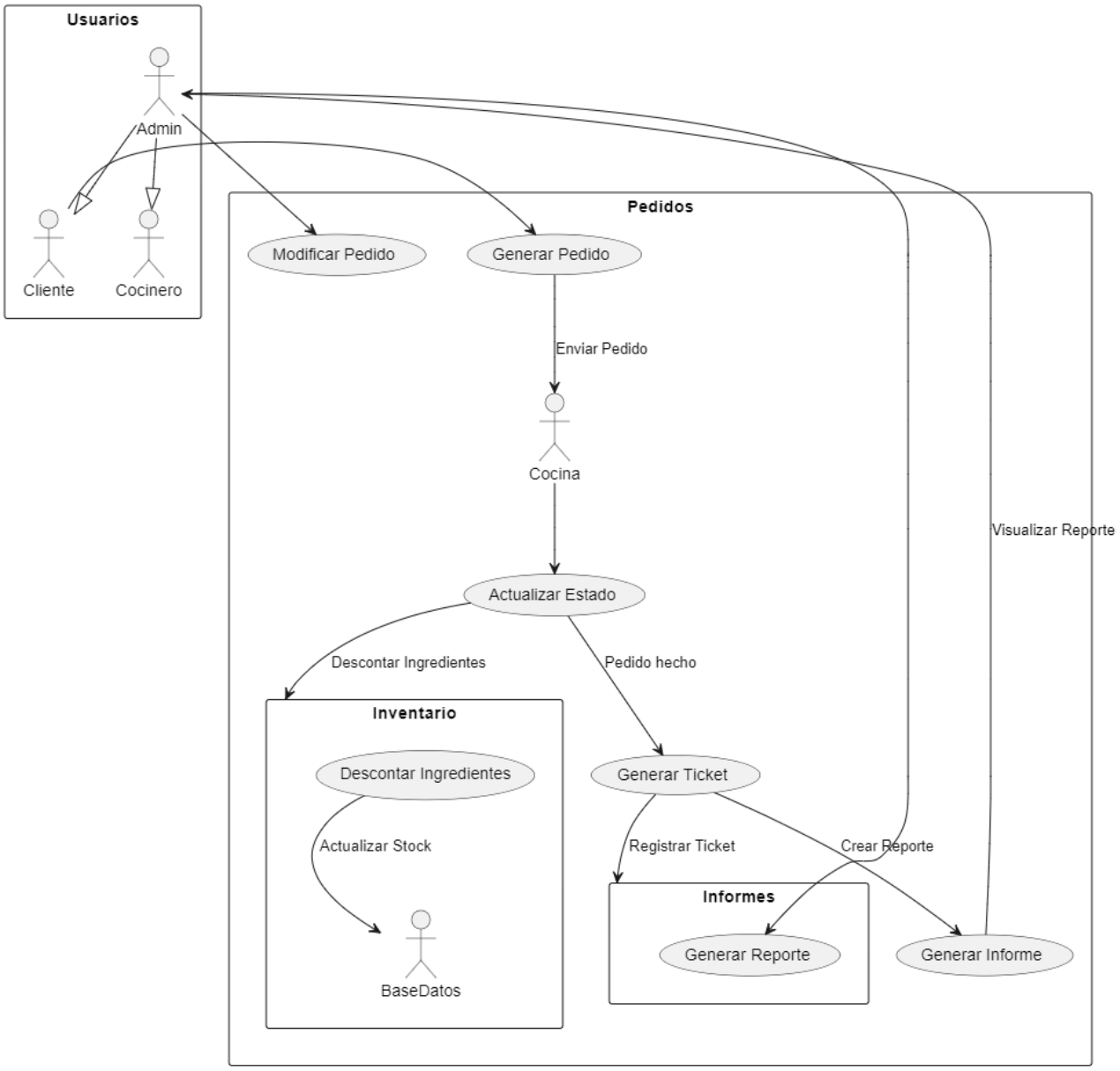
Usuarios	Pedidos, Inventario, Informes, Carrito.	Los clientes pueden ver el menú y realizar pedidos dentro del carrito. Los administradores gestionan pedidos, inventario e informes.
Pedidos	Inventario, Ticket, Informes.	Los pedidos afectan el inventario y generan tickets. Se registran en los informes.
Inventario	Pedidos, Informes.	Se actualiza con cada pedido y se refleja en los informes.
Informes	Pedidos, Inventario.	Se generan reportes basados en los pedidos realizados y el inventario.
Ticket	Pedidos	Se genera un ticket para cada pedido confirmado.

Datos Compartidos y Comunicación.

A continuación, se detallan los principales datos que se comparten entre módulos y cómo se comunican.

Dato Compartido	Fuente	Módulos que lo Usan
Lista de usuarios	Usuarios	Pedidos, Informes, Carrito.
Pedidos registrados	Pedidos	Ticket, Inventario, Informes.
Estado del pedido	Pedidos	Cocinero, Inventario, Ticket.
Inventario disponible	Inventario	Pedidos, Informes.
Reportes de ventas	Informes	Administrador.

El siguiente diagrama representa la interacción entre los módulos:



Este diseño modular garantiza que cada componente tenga una responsabilidad bien definida, mejorando la mantenibilidad y escalabilidad del sistema.

1.4.3 Organización interna del sistema.

Para representar esta vista se eligió UML, específicamente un diagrama de clases. Esta notación permite visualizar de manera clara la organización interna del sistema de pedidos, las clases que lo componen y cómo se relacionan entre sí.

En el sistema hay una clase principal llamada Usuario, que sirve como base para los tres tipos de usuarios que interactúan con la aplicación: el Cliente, el

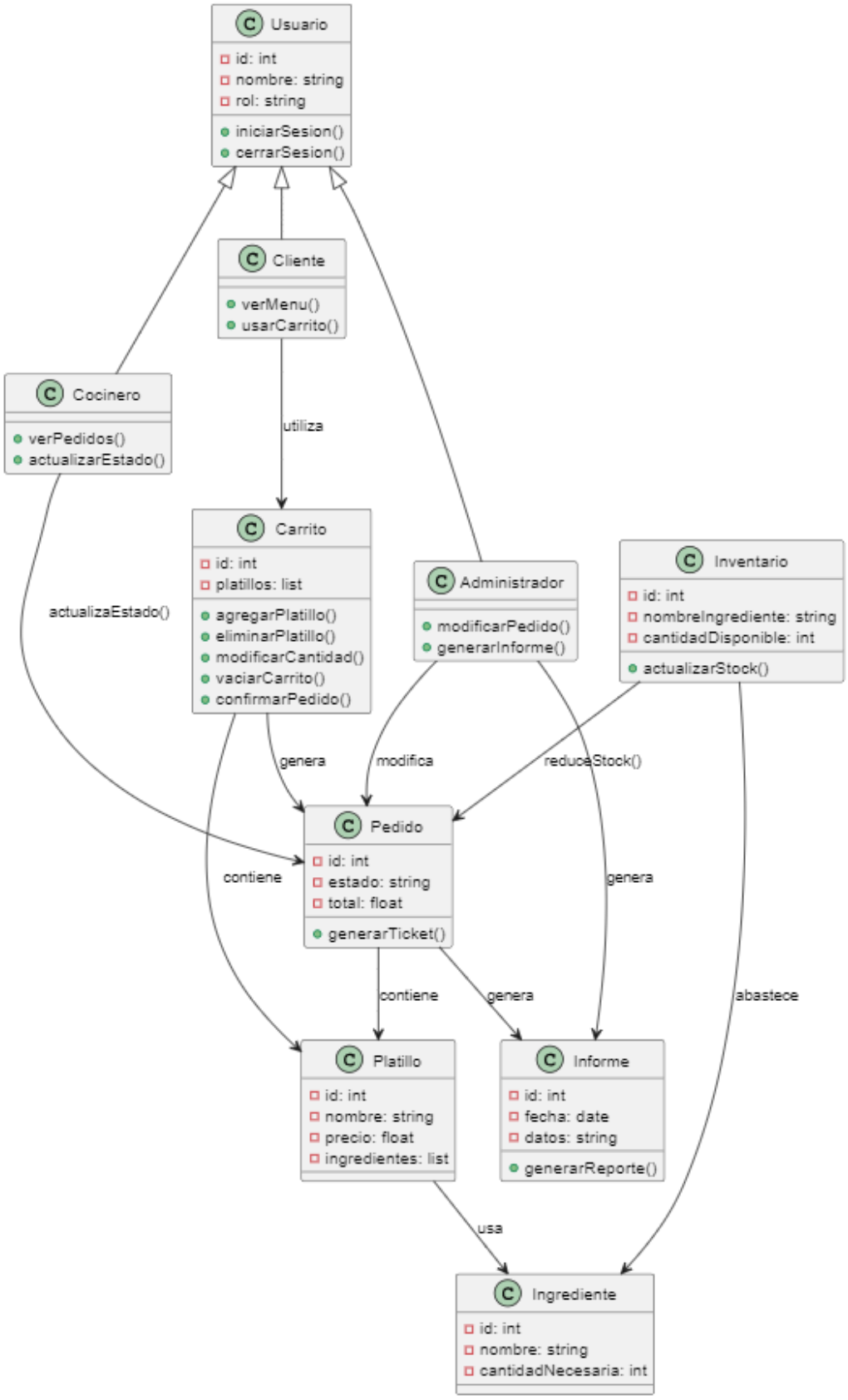
Administrador y el Cocinero. Cada uno tiene responsabilidades distintas, pero todos comparten la capacidad de iniciar y cerrar sesión.

El cliente tiene una funcionalidad especial, que es el carrito de compras. El carrito le permite seleccionar platillos, modificarlos o eliminarlos antes de confirmar el pedido final. Esta parte es importante porque representa cómo el cliente personaliza su orden antes de enviarla. Una vez confirmado, el contenido del carrito genera un Pedido.

El pedido contiene los platillos seleccionados, tiene un estado (como "pendiente" o "entregado") y está relacionado con otras partes del sistema. Por ejemplo, cuando se genera un pedido, se actualiza el inventario, ya que los ingredientes de los platillos deben descontarse automáticamente. A su vez, el cocinero puede ver estos pedidos y cambiar su estado conforme se van preparando y entregando.

El administrador tiene acceso a los pedidos, los puede modificar si es necesario y también puede generar informes que ayudan a revisar la actividad del restaurante, como las ventas o los productos más pedidos. El informe está representado como otra clase dentro del diagrama.

Este modelo lógico refleja de forma organizada cómo funciona el sistema internamente, destacando la relación entre los usuarios, los pedidos, el carrito, el inventario y los informes. A través del lenguaje UML, es posible comunicar de forma clara y comprensible tanto la estructura del sistema como las responsabilidades de cada parte.

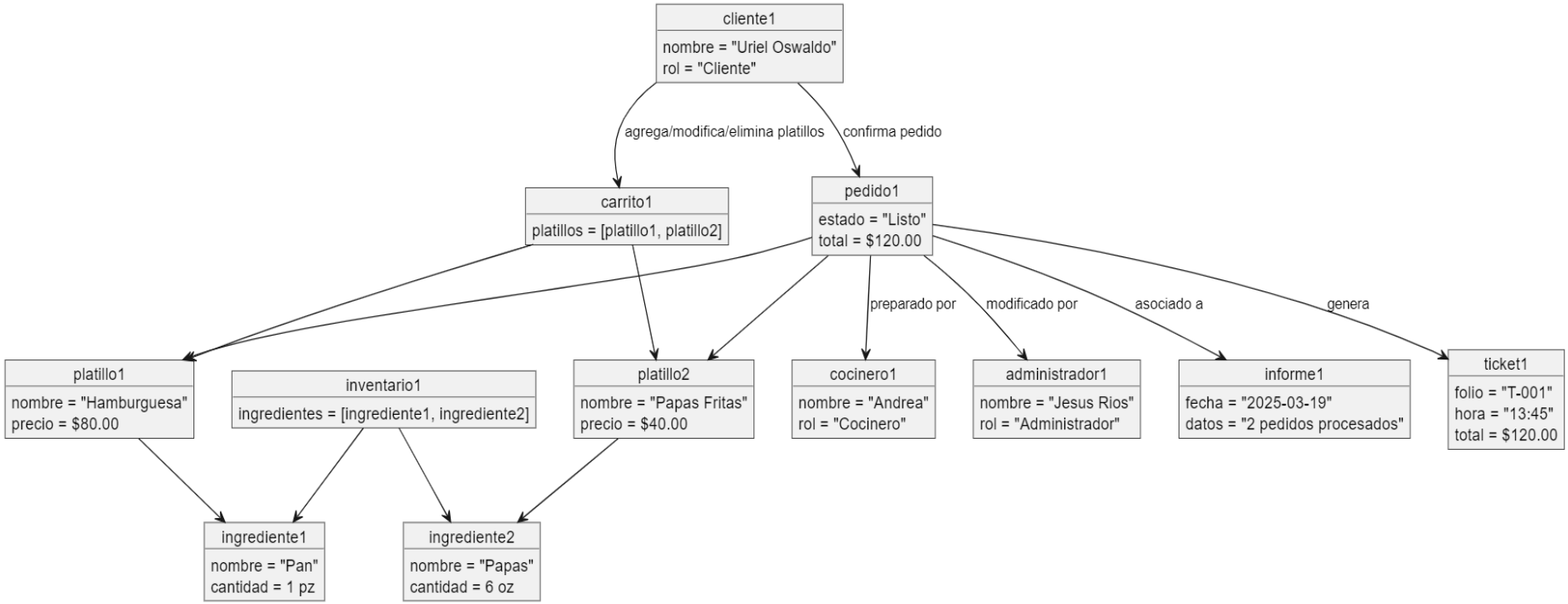

Usuario
id: int
nombre: string
rol: string
iniciarSesion()
cerrarSesion()
Cliente
verMenu()
usarCarrito()
Cocinero
verPedidos()
actualizarEstado()
utiliza
Carrito
id: int
platillos: list
agregarPlatillo()
eliminarPlatillo()
modificarCantidad()
vaciarCarrito()
confirmarPedido()
Administrador
modificarPedido()
generarInforme()
Inventario
id: int
nombreIngrediente: string
cantidadDisponible: int
actualizarStock()
actualizaEstado()
genera
modifica
reduceStock()
Pedido
id: int
estado: string
total: float
generarTicket()
contiene
genera
contiene
genera
abastece
Platillo
id: int
nombre: string
precio: float
ingredientes: list
Informe
id: int
fecha: date
datos: string
generarReporte()
usa
Ingrediente
id: int
nombre: string
cantidadNecesaria: int

El siguiente diagrama de objetos representa una instancia particular del sistema de pedidos de platillos en línea, donde se visualizan los objetos creados durante la operación típica de un cliente que realiza un pedido. Este ejemplo se enfoca en mostrar cómo los objetos interactúan entre sí en un momento dado, reflejando una estructura estática concreta que ilustra las relaciones de uso y colaboración entre instancias.

Se puede observar un objeto cliente1: Cliente que ha iniciado y ha comenzado a generar un pedido (pedido1: Pedido). Como parte del proceso, se ha creado un carrito (carrito1: Carrito) donde el cliente ha agregado dos platillos: platillo1: Platillo (Hamburguesa) y platillo2: Platillo (Papas fritas). Estos objetos representan los elementos seleccionados por el cliente antes de confirmar el pedido.

Cada platillo tiene asociados ingredientes provenientes del inventario (inventario1: Inventario y inventario2: Inventario), mostrando la relación entre el inventario y los elementos del menú. Una vez confirmado el pedido, este genera un ticket (ticket1: Ticket) asociado a la transacción. Además, un cocinero (cocinero1: Cocinero) tiene acceso al objeto pedido1 para visualizarlo y actualizar su estado según el flujo de trabajo en la cocina.

Este modelo ejemplifica una situación concreta en la que las clases del sistema cobran vida como objetos, permitiendo una mejor comprensión de las relaciones estáticas y su rol dentro de la ejecución del sistema. Además, sirve como puente entre el diseño conceptual del diagrama de clases y el comportamiento dinámico que será abordado en los diagramas de secuencia o colaboración.


cliente1
nombre = "Uriel Oswaldo"
rol = "Cliente"
agrega/modifica/elimina platillos
confirma pedido
carrito1
platillos = [platillo1, platillo2]
pedido1
estado = "Listo"
total = $120.00
preparado por
modificado por
asociado a
genera
platillo1
nombre = "Hamburguesa"
precio = $80.00
inventario1
ingredientes = [ingrediente1, ingrediente2]
platillo2
nombre = "Papas Fritas"
precio = $40.00
cocinero1
nombre = "Andrea"
rol = "Cocinero"
administrador1
nombre = "Jesus Rios"
rol = "Administrador"
informe1
fecha = "2025-03-19"
datos = "2 pedidos procesados"
ticket1
folio = "T-001"
hora = "13:45"
total = $120.00
ingrediente1
nombre = "Pan"
cantidad = 1 pz
ingrediente2
nombre = "Papas"
cantidad = 6 oz

1.5 Punto de vista de dependencias.

Este punto de vista describe las relaciones de dependencia entre los elementos estructurales del sistema. Su propósito es mostrar cómo los cambios o fallas en un componente pueden afectar a otros, permitiendo anticipar el impacto de modificaciones, identificar posibles problemas de mantenimiento y mejorar la planificación de pruebas o despliegue. Las dependencias definen rutas de comunicación, herencia, uso o asociación, tanto entre clases como entre componentes.

En el sistema de comanda digital, el punto de vista de dependencias se centra en el flujo de información y control que va desde la interacción del cliente con el sistema hasta la gestión interna de pedidos y recursos. Este enfoque permite comprender el acoplamiento entre las partes del sistema, asegurando una estructura manejable y mantenible.

1.5.1 Preocupaciones de diseño.

En el Sistema de comanda digital, las dependencias entre los distintos módulos y componentes juegan un papel clave para mantener una arquitectura clara y funcional. Las principales preocupaciones de diseño giran en torno a cómo se conectan e interactúan los elementos del sistema sin generar acoplamientos innecesarios que dificulten su mantenimiento, escalabilidad o reutilización.

Una de las preocupaciones más relevantes es lograr un bajo acoplamiento entre las clases. Por ejemplo, el módulo de generación de pedidos no debería depender directamente de cómo se actualiza el inventario, sino comunicarse con él a través de una interfaz bien definida. Esto permite que, si el componente de inventario se modifica en el futuro, no afecte directamente al resto del sistema.

Otra preocupación es asegurar que los cambios en los elementos administrativos, como la generación de informes, no afecten a la experiencia del cliente. Este tipo de separación busca mantener cada funcionalidad en su contexto sin interferencias externas, fomentando una estructura modular.

También se busca evitar dependencias circulares, especialmente entre clases de usuario y procesos internos como los pedidos o la gestión de platillos. Este tipo de relaciones puede complicar la comprensión y evolución del sistema.

Por último, se considera importante identificar claramente qué componentes son reutilizables, como el carrito de compras o el ticket de pedido, para diseñar sus dependencias de forma que puedan integrarse en otros proyectos similares sin dificultades.

1.5.2 Elementos de diseño.

Dentro del sistema, los elementos de diseño que se relacionan a través de dependencias son principalmente las clases, componentes lógicos y las interfaces que permiten la interacción entre distintos módulos. Cada uno de estos elementos ha sido pensado para cumplir una función específica y mantenerse lo más independiente posible.

Uno de los elementos clave es el **CarritoDeCompras**, que mantiene una relación directa con la clase **Platillo** y la clase **Pedido**. Esta dependencia está justificada, ya que el carrito requiere acceder a la información de los platillos para calcular totales, y finalmente genera un pedido cuando el cliente confirma su orden.

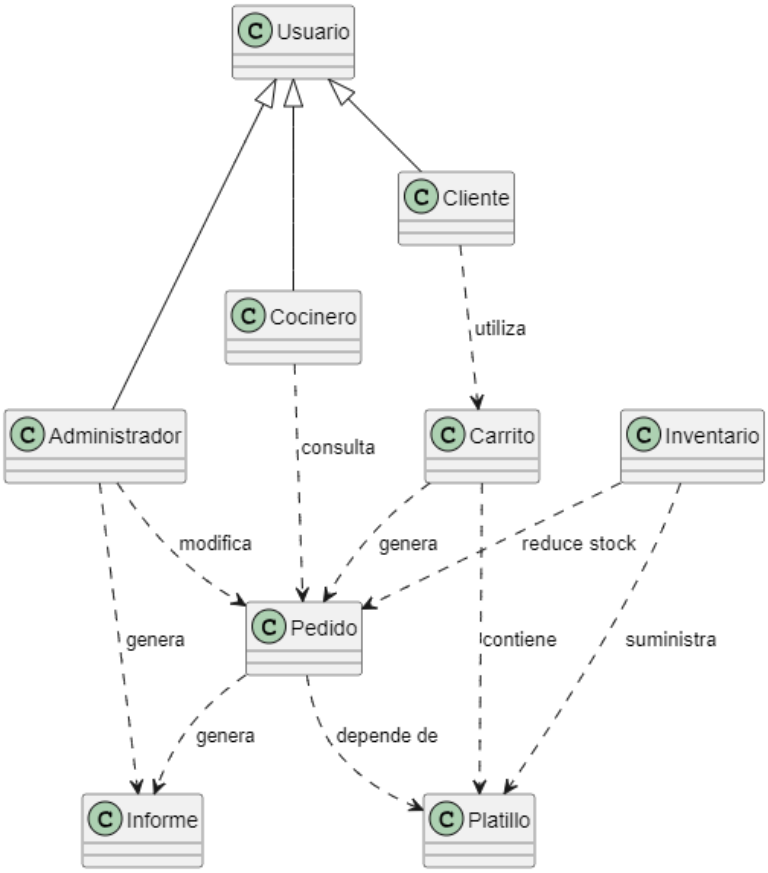
El componente **Pedido**, por su parte, depende del módulo **Inventario** para verificar la disponibilidad de los ingredientes al momento de confirmar la orden. Esta relación se da mediante una función de consulta y reducción de stock, controlada por métodos definidos en la clase **Inventario**, evitando acoplamientos excesivos.

La clase **Informe** depende de **Pedido** para obtener los datos históricos necesarios para generar reportes administrativos. Esta dependencia es de solo lectura y no interfiere con el comportamiento operativo del módulo de pedidos.

Otro elemento importante es la dependencia entre **Cocinero** y **Pedido**, ya que este actor necesita consultar y actualizar el estado de los pedidos. Esta relación está encapsulada dentro de un método específico que permite modificar únicamente el estado del pedido, sin exponer sus otros atributos internos.

En términos de diseño, se ha buscado que las dependencias fluyan en una sola dirección, siguiendo una estructura jerárquica desde las interfaces de usuario hacia los servicios internos, y de estos hacia los recursos del sistema, como el inventario o los informes.

A continuación, se presenta un diagrama de dependencias que representa gráficamente las relaciones descritas:



Este diagrama muestra las principales relaciones de dependencia entre los elementos clave del sistema. Se destacan los módulos de usuario, pedido, inventario, informe y platillo, así como sus interacciones. Las dependencias están orientadas de forma jerárquica desde los usuarios y sus interfaces hasta los servicios internos, lo que contribuye a mantener un bajo acoplamiento y alta cohesión en el diseño general del sistema.

1.5.2.1 Atributos de dependencias.

Las dependencias identificadas en el diseño del sistema presentan diversos atributos que determinan su impacto en la arquitectura general. Estos atributos permiten entender mejor la naturaleza de cada relación y facilitan la toma de decisiones para futuras modificaciones o ampliaciones del sistema.

Uno de los principales atributos es la “direccionalidad”, ya que todas las dependencias en el sistema son unidireccionales. Esto significa que un componente A puede depender de B, pero B no conoce ni utiliza A. Esta estrategia ayuda a mantener un bajo acoplamiento, lo que mejora la mantenibilidad.

Otro atributo importante es el “tiempo de enlace”. La mayoría de las dependencias están definidas en tiempo de compilación, lo que significa que las clases y métodos se conocen al momento de construir el sistema. Esto brinda mayor claridad en el flujo de ejecución, aunque limita la flexibilidad. Algunas dependencias, como las que involucran generación de informes o acceso al inventario, podrían adaptarse a un enlace dinámico en futuras versiones si se desea mayor modularidad.

En cuanto a la naturaleza de uso, se identifican dependencias de consulta, modificación o generación. Por ejemplo, el **CarritoDeCompras** consulta y modifica instancias de **Platillo**, y a su vez genera un **Pedido**. Por otro lado, la clase **Informe** depende de **Pedido**, pero únicamente en modo de lectura.

El “grado de criticidad” también se considera. Dependencias como las de **Inventario** con **Pedido** son críticas, ya que una falla en el inventario podría impedir procesar pedidos. Por eso, se contempla el manejo de errores y validaciones que permitan que el sistema responda adecuadamente en caso de inconsistencias.

Finalmente, el atributo de **frecuencia de uso** destaca que algunas dependencias, como las que existen entre el cliente y su carrito o entre cocinero y pedidos, son de uso constante. Estas relaciones están optimizadas para ofrecer una respuesta rápida y confiable, mientras que otras, como la generación de informes, son más esporádicas y permiten un procesamiento más diferido.

1.5.3 Ejemplos de idiomas.

Para ilustrar las dependencias presentes en el sistema, se ha utilizado PlantUML como lenguaje gráfico, permitiendo modelar visualmente las relaciones entre clases, componentes y paquetes. Esta herramienta facilita la comprensión del acoplamiento entre módulos de una manera clara y detallada, lo que resulta útil para el análisis estructural del software y posibles mejoras en el diseño.

El diagrama de dependencias generado muestra cómo interactúan entidades clave como Usuario, Cliente, Pedido, Platillo, Inventario, entre otras. Por ejemplo, la clase Cliente depende de Pedido para generar órdenes, mientras que Pedido se apoya en Platillo para conocer qué se ha solicitado. A su vez, Platillo tiene una relación directa con Inventario, el cual gestiona los ingredientes disponibles. Administrador, por otro lado, accede a Pedido para modificar información o generar informes.

Estas representaciones visuales permiten detectar puntos de acoplamiento fuerte, facilitando decisiones de refactorización cuando sea necesario. Además, la elección de PlantUML se justifica por su facilidad de integración con herramientas de documentación y su capacidad para expresar relaciones de dependencia de forma efectiva.

Complementando esta visualización, se han utilizado dos diagramas UML: el diagrama de componentes y el diagrama de paquetes.

Diagrama de componentes.

Este diagrama muestra cómo se relacionan los principales bloques funcionales del sistema. Se enfoca en los componentes lógicos que forman la arquitectura general del software, como la interfaz del cliente, el carrito de compras, el pedido, el inventario, entre otros. Las flechas entre componentes representan dependencias, es decir, cuándo un componente necesita acceder o utilizar otro para cumplir su función.

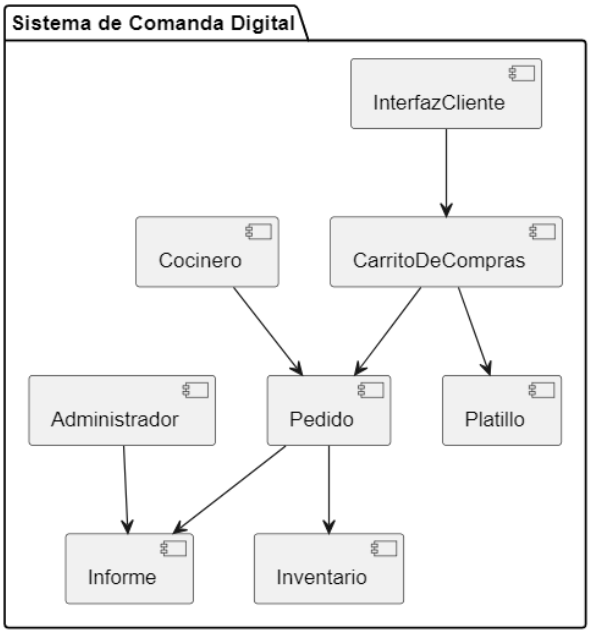


Diagrama de paquetes.

El diagrama de paquetes ofrece una visión de alto nivel sobre la organización del sistema en términos de agrupaciones lógicas. Cada paquete representa un módulo o conjunto de clases con responsabilidades relacionadas. Este diagrama facilita la comprensión del diseño modular del sistema y las dependencias entre paquetes, promoviendo una estructura ordenada y de bajo acoplamiento.