

Institución:



Carrera:

Licenciatura en Ing. De Software

Ciclo:

2025-I

Grupo:

1101

Asignatura:

Diseño de software

Título de actividad:

Estándar IEEE 1016 – Interfaz

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

1. Especificación del Diseño del Software.	4
1.1 Introducción.....	4
1.2 Visión General del Diseño.	4
1.2.1 Estructura del Sistema.	4
1.2.1.1 DCU-1. Gestión menú.	5
1.2.1.2 DCU-2. Realización de pedidos.	7
1.2.1.3 DCU-3 Gestión de Administradores.....	9
1.2.1.4 DCU-4. Inventario.	10
1.2.1.5 DCU-5. Informes.	12
1.2.1.6 DCU-6. Personalización de plato.	14
1.2.1.7 DCU7.- Recopila los datos.....	16
1.3 Visión General de la Composición.	18
1.3.1 DP1 – menú.....	19
1.3.2 DP2 - Realización de pedidos.	21
1.3.3 DC3 - Gestión de Administradores.....	23
1.3.4 DP4 – Inventarios.	26
1.3.5 DP5-> Informe.....	28
1.3.6 DP6-> Personalización de pedidos.....	30
1.3.7 DP7 - Recopilación de datos.	32
1.4 Punto de Vista Lógico.	34
1.4.1 Puntos clave a considerar del sistema.	35
1.4.2 Módulos del sistema.	36
1.4.3 Organización interna del sistema.	40
1.5 Punto de vista de dependencias.....	45
1.5.1 Preocupaciones de diseño.	45
1.5.2 Elementos de diseño.....	46
1.5.2.1 Atributos de dependencias.	48
1.5.3 Ejemplos de idiomas.	49
1.6 Los patrones utilizan el punto de vista	51
1.6.1 Aspectos clave del diseño	51
1.6.2 Elementos de diseño.....	52
1.6.2.1Patrones: Se aplican varios patrones:	54

1.6.3 Restricciones del punto de vista	57
1.7 Interface	61
1.7.1. Aplicabilidad a distintos tipos de interfaces.....	62
1.7.2. Elementos del diseño de interfaz	62
1.7.2.1. Atributo de interfaz	62
1.7.3. Diagramas.....	63
1.7.3.1. DC2 Realización de pedidos.....	64
1.7.3.2 Interacción del Usuario con el Realización de pedidos.....	65

1. Especificación del Diseño del Software.

Este documento describe el diseño del Sistema de Comanda Digital para un restaurante, conforme al estándar IEEE 1016. Se detallan las decisiones de diseño, su estructura, comportamiento y justificación, con el fin de proporcionar una guía clara para su implementación y mantenimiento.

1.1 Introducción.

La Comanda Digital es un sistema desarrollado para mejorar la gestión de pedidos en un restaurante, optimizando la comunicación entre cliente, meseros y cocina. Se basa en una arquitectura cliente-servidor, donde los meseros ingresan órdenes a través de una interfaz digital y estas son enviadas en tiempo real al área de cocina.

Este documento se enfoca en describir el diseño del software, considerando los diagramas de casos de uso, paquetes, clases y el código disponible. La metodología de diseño se basa en la orientación a objetos, asegurando modularidad y escalabilidad.

1.2 Visión General del Diseño.

El sistema sigue una arquitectura multicapa, dividiéndose en:

- Capa de Presentación: Interfaz gráfica para meseros y personal de cocina.
- Capa de Lógica de Negocio: Procesamiento de órdenes, gestión de menú y control de estados de pedidos.
- Capa de Persistencia: Base de datos para almacenar pedidos, usuarios y menú.

1.2.1 Estructura del Sistema.

El sistema está compuesto por los siguientes módulos:

1. Gestión de Usuarios: Manejo de credenciales y roles (mesero, cocinero, administrador).

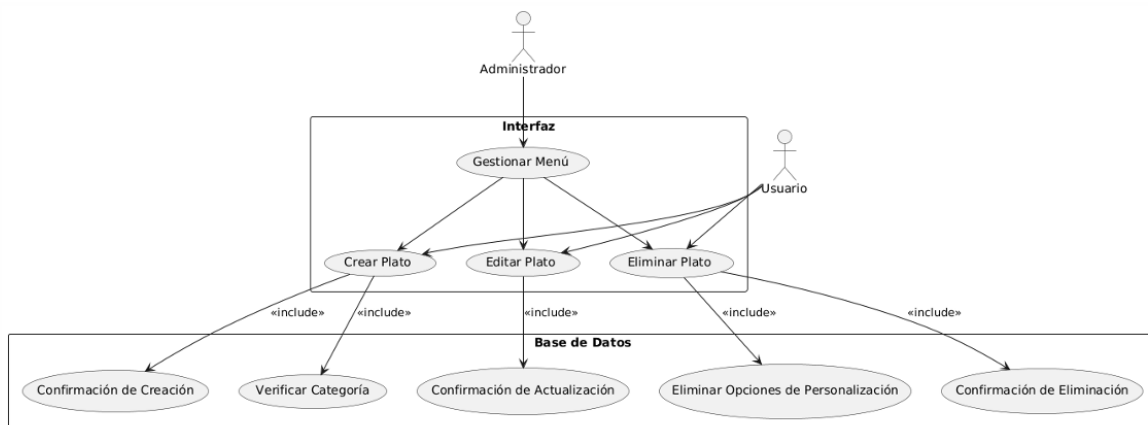
2. Gestión de Pedidos: Creación, actualización y seguimiento de órdenes en tiempo real.
3. Interfaz de Cocina: Visualización de pedidos pendientes y actualización de su estado.
4. Base de Datos: Almacenamiento de menús, pedidos y usuarios.

Cada módulo se comunica entre sí, asegurando independencia y escalabilidad. A continuación, se detallarán cada una de las funcionalidades haciendo uso de los diagramas de casos de uso para el sistema.

1.2.1.1 DCU-1. Gestión menú.

Diagramas de casos de uso específico

DCU-1. Gestión menú



1. Descripción del Caso de Uso.

El caso de uso "Gestión de Menú" representa las interacciones que los actores del sistema tienen con la funcionalidad de administración del menú en la aplicación. Este caso de uso permite a los administradores y usuarios realizar acciones sobre los platillos del menú, incluyendo su creación, edición y eliminación.

2. Actores Involucrados.

- **Administrador:** Responsable de gestionar el menú, creando, editando o eliminando platillos.
- **Usuario:** Puede visualizar el menú y sus opciones, pero no tiene permisos de modificación.

3. Flujo Principal de Eventos.

El administrador accede a la interfaz de "Gestión de Menú" utilizando sus credenciales. Dentro de esta interfaz, se le presentan tres opciones principales: crear un nuevo platillo, editar un platillo existente o eliminar un platillo. Si el administrador opta por crear un platillo, deberá ingresar la información correspondiente al nuevo platillo, posteriormente se verificará la categoría del mismo, y finalmente, se confirmará su creación y se almacenará en la base de datos. En caso de elegir editar un platillo, el administrador primero seleccionará el platillo que desea modificar, realizará los cambios necesarios y luego se confirmará la actualización en la base de datos. Por último, si la opción seleccionada es eliminar un platillo, el administrador deberá elegir el platillo a eliminar, tras lo cual se eliminarán las opciones de personalización que estén asociadas a dicho platillo, y se confirmará la eliminación en la base de datos.

4. Variantes o Excepciones.

Si el administrador intenta crear un platillo sin proporcionar la información completa requerida, el sistema mostrará un mensaje de error para alertar sobre la falta de datos. En caso de que intente editar un platillo que no se encuentra registrado en el sistema, se le notificará la ausencia de dicho registro. Finalmente, si la acción es eliminar un platillo, el sistema solicitará una confirmación por parte del administrador antes de llevar a cabo la eliminación para evitar acciones involuntarias.

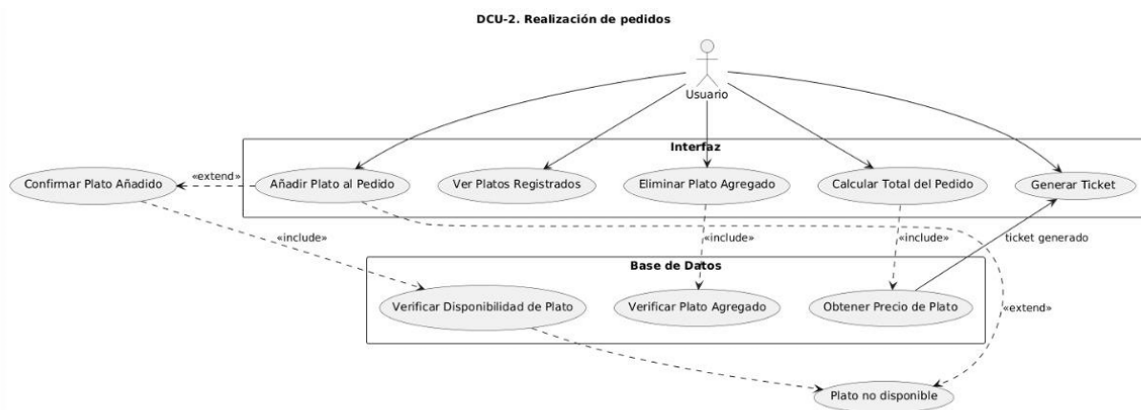
5. Relaciones entre Casos de Uso.

Las acciones de Crear Platillo, Editar Platillo y Eliminar Platillo se encuentran directamente relacionadas con el caso de uso general denominado "Gestión de

Menú". Cada una de estas acciones principales implica la ejecución de subprocesos específicos. Por ejemplo, la creación de un platillo incluye subprocesos como la "Confirmación de Creación" y la "Verificación de Categoría". De manera similar, la edición de un platillo conlleva la "Confirmación de Actualización", mientras que la eliminación de un platillo involucra la "Eliminación de Opciones de Personalización" y la "Confirmación de Eliminación".

1.2.1.2 DCU-2. Realización de pedidos.

DCU-2. Realización de pedidos



1. Descripción del Caso de Uso.

El caso de uso "Realización de Pedidos" representa las interacciones del usuario con el sistema para gestionar su pedido. Este caso de uso permite a los usuarios agregar, eliminar y visualizar platillos en su pedido, calcular el total del pedido y generar un ticket de compra.

2. Actores Involucrados.

- **Usuario:** Responsable de gestionar su pedido, agregando o eliminando platillos y finalizando la compra.

3. Flujo Principal de Eventos.

El usuario accede a la interfaz de "Realización de Pedidos", donde se le presentan diversas opciones: añadir un platillo al pedido, ver los platillos que ya están registrados, eliminar un platillo previamente agregado, calcular el total del pedido o generar el ticket de compra. Si el usuario selecciona añadir un platillo, el sistema primero verifica su disponibilidad; si está disponible, se confirma su adición y se almacena en la base de datos como parte del pedido. En caso de que el usuario elija eliminar un platillo agregado, deberá seleccionar el platillo en cuestión, el sistema verificará si efectivamente se encuentra en el pedido y, de ser así, lo eliminará de la base de datos. Si la opción elegida es calcular el total del pedido, el sistema obtendrá el precio de cada platillo añadido, sumará estos precios y mostrará el total al usuario. Finalmente, si el usuario opta por generar el ticket de compra, se confirmará la información del pedido y se generará un ticket que contendrá el resumen detallado del pedido junto con su costo total.

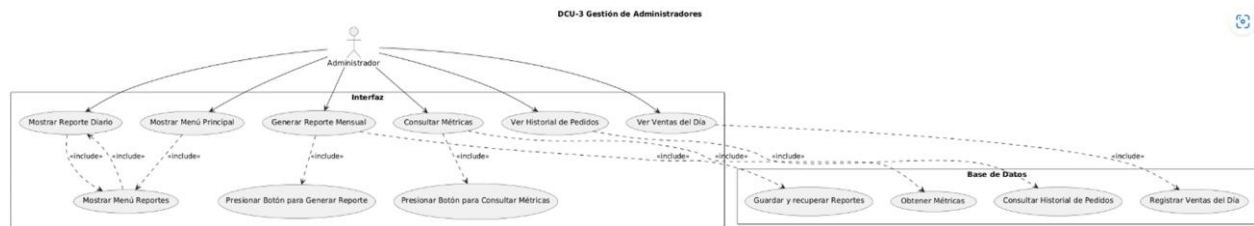
4. Variantes o Excepciones.

Si durante el proceso de añadir un platillo al pedido, el usuario intenta seleccionar uno que no se encuentra disponible, el sistema le mostrará un mensaje indicando "Plato no disponible". De manera similar, si el usuario intenta eliminar un platillo que no ha sido agregado previamente al pedido, se le notificará esta situación. Finalmente, en caso de que surjan errores durante la generación del ticket de compra, el sistema solicitará al usuario una nueva confirmación de la información del pedido para intentar generar el ticket nuevamente.

5. Relaciones entre Casos de Uso.

Las acciones de Añadir Platillo al Pedido, Eliminar Platillo Agregado y Calcular Total del Pedido forman parte del caso de uso general denominado "Realización de Pedidos". Cada una de estas acciones principales se compone de subprocesos específicos, tales como "Verificar Disponibilidad de Plato", "Verificar Plato Agregado", "Obtener Precio de Plato" y "Confirmar Plato Añadido". Por otro lado, la acción de "Generar Ticket" tiene una dependencia directa del cálculo previo del total del pedido y, además, está vinculada a la información almacenada en la base de datos.

1.2.1.3 DCU-3 Gestión de Administradores.



1. Descripción del Caso de Uso.

El caso de uso "Gestión de Administradores" representa las interacciones que los administradores del sistema tienen con la funcionalidad de administración y consulta de reportes, métricas e historial de pedidos. Permite realizar tareas clave como la generación de reportes, consulta de ventas y revisión del historial.

2. Actores Involucrados.

- **Administrador:** Responsable de generar reportes, consultar métricas, ver historial de pedidos y revisar ventas del día.

3. Flujo Principal de Eventos.

El administrador ingresa a la interfaz de "Gestión de Administradores" utilizando su licencia. Una vez dentro, se le presentan varias opciones: Mostrar Reporte Diario, Mostrar Menú Principal, Generar Reporte Mensual, Consultar Métricas, Ver Historial de Pedidos y Ver Ventas del Día. Si el administrador selecciona Mostrar Reporte Diario, se incluye la opción adicional "Mostrar Menú Reportes" y se recuperan y presentan los datos correspondientes al reporte diario. Al elegir Mostrar Menú Principal, se visualiza el menú principal con todas las opciones disponibles para el administrador. Si opta por Generar Reporte Mensual, al presionar el botón respectivo, el sistema accede a la base de datos para guardar y recuperar los reportes mensuales. Al seleccionar Consultar Métricas, al presionar el botón correspondiente, se obtiene la información de las métricas directamente desde la base de datos. Si el administrador elige Ver Historial de Pedidos, se realiza una consulta al historial de pedidos almacenado en la base de datos. Finalmente, al

seleccionar Ver Ventas del Día, se registran y muestran las ventas correspondientes al día actual, obteniendo esta información de la base de datos

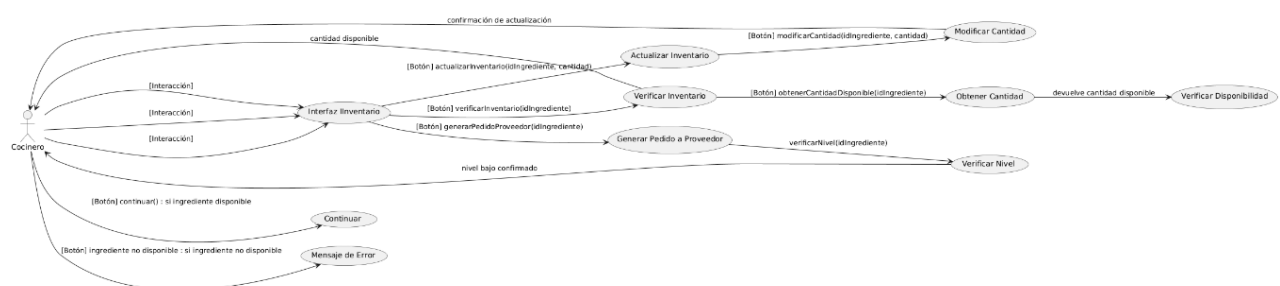
4. Variantes o Excepciones.

Si el administrador intenta generar un reporte pero no existen datos suficientes para su creación, el sistema le mostrará un mensaje de error indicando la falta de información necesaria. En caso de que se solicite una consulta de métricas y los datos requeridos no se encuentren disponibles en el sistema, se mostrará una notificación al administrador informando sobre la ausencia de la información. Finalmente, si el administrador intenta acceder al historial de pedidos sin una conexión activa a la base de datos, el sistema le advertirá sobre este problema de conectividad que impide la visualización del historial.

5. Relaciones entre Casos de Uso.

Las acciones de Mostrar Reporte Diario, Mostrar Menú Principal y Generar Reporte Mensual comparten el subproceso común de "Mostrar Menú Reportes". Específicamente, la acción de Generar Reporte Mensual abarca los subprocesos de "Presionar Botón para Generar Reporte" y "Guardar y Recuperar Reportes". Por su parte, Consultar Métricas implica los subprocesos de "Presionar Botón para Consultar Métricas" y "Obtener Métricas". La acción de Ver Historial de Pedidos se reduce al subproceso de "Consultar Historial de Pedidos", mientras que Ver Ventas del Día se centra en el subproceso de "Registrar Ventas del Día".

1.2.1.4 DCU-4. Inventario.



1. Descripción del Caso de Uso.

El caso de uso "Gestión de Inventario" describe las interacciones del cocinero con el sistema para administrar los ingredientes y su disponibilidad. Permite verificar cantidades, actualizar inventario y generar pedidos a proveedores cuando sea necesario.

2. Actores Involucrados.

- **Cocinero:** Responsable de gestionar el inventario, verificar disponibilidad de ingredientes y generar pedidos a proveedores.

3. Flujo Principal de Eventos.

El cocinero ingresa a la interfaz de "Gestión de Inventario", donde puede elegir entre tres opciones principales: Verificar Inventario, Actualizar Inventario o Generar Pedido a Proveedor. Si selecciona Verificar Inventario, al presionar el botón correspondiente, el sistema le solicitará un ingrediente específico, obtendrá la cantidad disponible en el inventario y mostrará el resultado. En caso de elegir Actualizar Inventario, al presionar el botón respectivo, podrá modificar la cantidad de un ingrediente, y el sistema confirmará la actualización una vez realizada. Si la opción es Generar Pedido a Proveedor, el sistema verificará el nivel del ingrediente seleccionado; si el nivel es bajo, se generará automáticamente un pedido al proveedor y se mostrará una confirmación al cocinero. Adicionalmente, si al intentar verificar o actualizar un ingrediente, este no se encuentra disponible en el sistema, se mostrará un mensaje de error y se ofrecerá la opción de generar un pedido para dicho ingrediente.

4. Variantes o Excepciones.

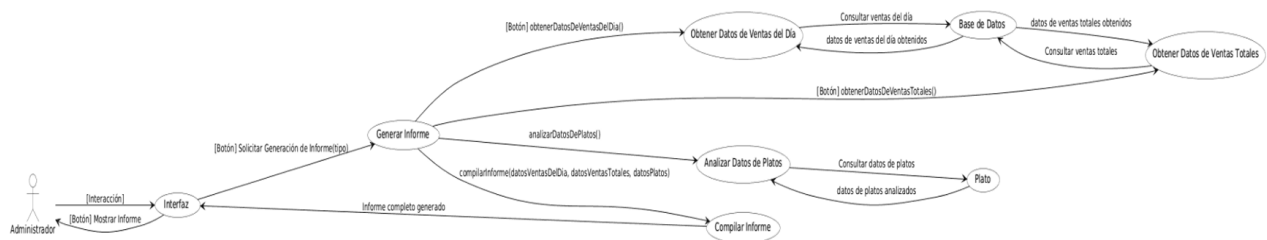
Si por alguna razón la cantidad disponible de un ingrediente no puede ser recuperada del sistema, se mostrará un mensaje de error al cocinero indicando el problema. Adicionalmente, si el sistema detecta que el nivel de algún ingrediente en el inventario es bajo, se generará automáticamente una notificación para alertar sobre la necesidad de realizar un nuevo pedido de dicho ingrediente.

5. Relaciones entre Casos de Uso.

La acción de Verificar Inventario comprende los subprocesos de "Obtener Cantidad" del ingrediente consultado y "Verificar Disponibilidad" del mismo. Por otro lado, la acción de Actualizar Inventario se centra en el subproceso de "Modificar Cantidad" del ingrediente seleccionado. Finalmente, la acción de Generar Pedido a Proveedor abarca los subprocesos de "Verificar Nivel" del ingrediente para determinar si es necesario pedir más y "Confirmación de Pedido" para asegurar que la solicitud se ha realizado correctamente.

1.2.1.5 DCU-5. Informes.

DCU-5. Informes



1. Descripción del Caso de Uso.

El caso de uso "Informes" permite a los administradores generar reportes basados en datos de ventas y platillos. Involucra la consulta, análisis y compilación de información para la generación de informes detallados.

2. Actores Involucrados.

- **Administrador:** Usuario responsable de solicitar la generación de informes.
- **Base de Datos:** Fuente de los datos de ventas y platillos.

3. Flujo Principal de Eventos.

El administrador ingresa a la interfaz del sistema y elige la opción "Mostrar Informe". A continuación, se le presenta la posibilidad de solicitar la generación del informe.

Si el administrador confirma esta solicitud, el sistema inicia una serie de procesos que incluyen la obtención de los datos de ventas correspondientes al día, la recuperación de los datos de ventas totales acumuladas, el análisis detallado de la información relacionada con los diferentes platillos ofrecidos y la compilación de toda la información obtenida en los pasos anteriores. Una vez que todos estos procesos se han completado, se genera un informe completo que queda a disposición del administrador para su revisión.

4. Variantes o Excepciones.

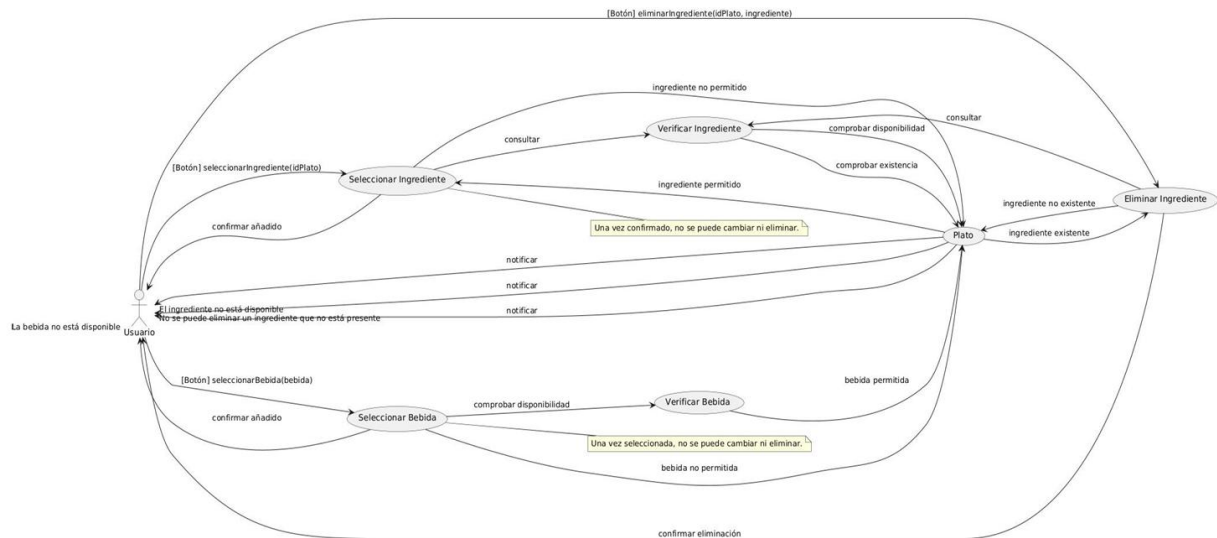
En caso de que la base de datos no responda a la solicitud de información, el sistema mostrará un mensaje de error para alertar al administrador sobre la imposibilidad de acceder a los datos. De igual manera, si al intentar generar el informe no se encuentran suficientes datos de ventas disponibles para realizar un análisis significativo, el sistema notificará esta situación al administrador

5. Relaciones entre Casos de Uso.

La acción de Generar Informe abarca los siguientes subprocesos: "Obtener Datos de Ventas del Día", que implica una consulta específica a la base de datos para recuperar la información de las ventas diarias; "Obtener Datos de Ventas Totales", que también requiere una consulta a la base de datos para acceder al registro de las ventas totales acumuladas; "Analizar Datos de Platos", que consiste en una consulta a la base de datos de platillos para obtener información relevante sobre el rendimiento y las ventas de cada platillo; y finalmente, "Compilar Informe", que es el proceso de unificación de toda la información recopilada en los pasos anteriores para presentarla de manera organizada y comprensible al administrador.

1.2.1.6 DCU-6. Personalización de plato.

DCU-6. Personalización de plato



1. Descripción del Caso de Uso.

El caso de uso "Personalización de Plato" permite a los usuarios modificar los ingredientes de su platillo y seleccionar bebidas, asegurando que solo se elijan opciones permitidas y disponibles.

2. Actores Involucrados.

- **Usuario:** Persona que personaliza su platillo.
- **Sistema:** Encargado de verificar la disponibilidad y validez de los ingredientes y bebidas.

3. Flujo Principal de Eventos.

El usuario que desea personalizar su plato accede a la opción correspondiente dentro de la interfaz. Allí, tiene la posibilidad de seleccionar ingredientes adicionales para añadir a su platillo o eliminar aquellos que ya están incluidos. El sistema lleva a cabo una verificación exhaustiva para asegurar tanto la disponibilidad de los ingredientes seleccionados como su validez dentro de las opciones permitidas. Si

un ingrediente cumple con estos criterios (es permitido y está disponible), el sistema confirma la selección realizada por el usuario. En el caso de que el usuario intente eliminar un ingrediente que no forma parte del platillo, el sistema le notificará esta situación. Es importante destacar que, una vez que la selección de ingredientes para el plato ha sido confirmada, el usuario ya no podrá realizar modificaciones adicionales. Adicionalmente, el usuario tiene la opción de seleccionar una bebida para acompañar su pedido. Al igual que con los ingredientes, el sistema verifica la disponibilidad y validez de la bebida seleccionada. Si la bebida cumple con los criterios establecidos (es permitida y está disponible), se confirma la selección del usuario. Una vez que la selección de la bebida ha sido confirmada, tampoco se podrán realizar cambios en esta elección.

4. Variantes o Excepciones.

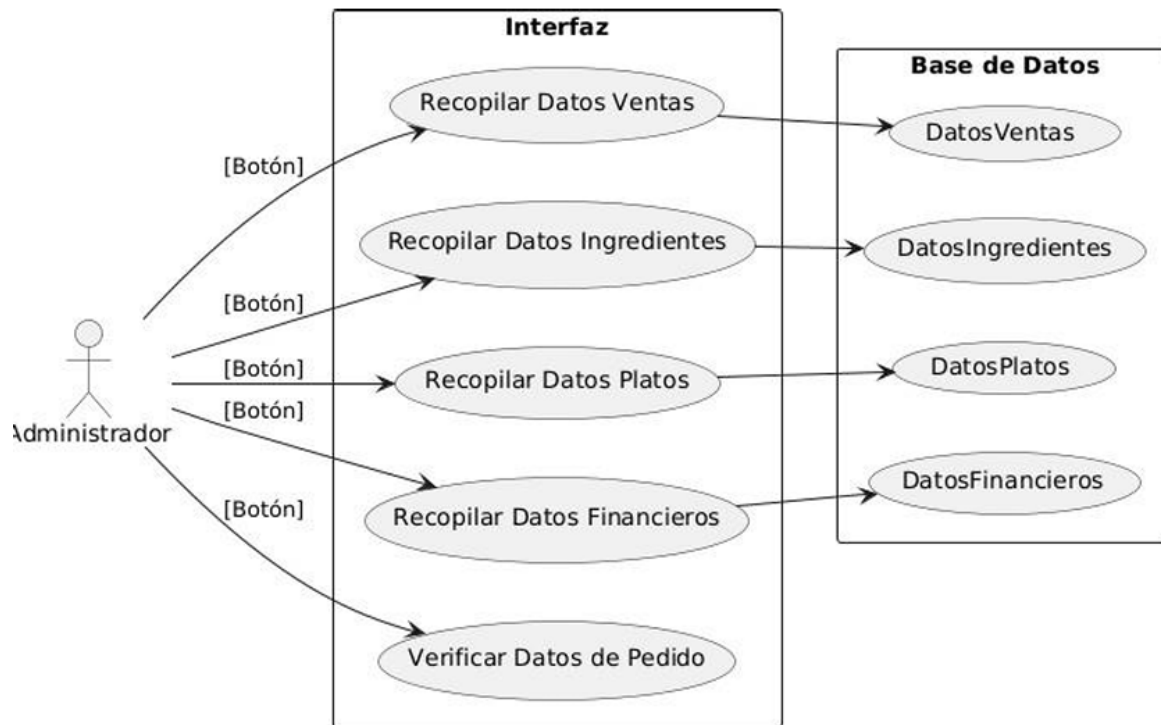
Si durante el proceso de personalización de su plato, el usuario selecciona un ingrediente que no se encuentra disponible en el inventario, el sistema le notificará inmediatamente sobre esta situación. De manera similar, si el usuario intenta eliminar un ingrediente que no ha sido previamente añadido al plato base, el sistema mostrará un mensaje de error indicando que dicho ingrediente no forma parte de la selección actual. En el caso de la selección de bebidas, si la bebida elegida por el usuario no está disponible, el sistema también le informará de esta falta de disponibilidad.

5. Relaciones entre Casos de Uso.

La acción de Seleccionar Ingrediente se compone de los subprocesos de "Verificar Ingrediente", para asegurar su validez y disponibilidad, y "Confirmar Añadido", para registrar la elección del usuario. Por otro lado, la acción de Eliminar Ingrediente abarca los subprocesos de "Verificar Existencia", para confirmar que el ingrediente a eliminar realmente forma parte del plato, y "Confirmar Eliminación", para registrar la modificación. Finalmente, la acción de Seleccionar Bebida incluye los subprocesos de "Verificar Bebida", para asegurar su disponibilidad, y "Confirmar Añadido", para registrar la selección de la bebida por parte del usuario.

1.2.1.7 DCU7.- Recopila los datos.

DCU7.- Recopila los datos



Descripción de caso de uso:

1. Actor (Administrador):

- Es el usuario principal del sistema que tiene el rol de recopilar y verificar datos.

2. Interfaz:

La interfaz del sistema está diseñada con diversas opciones, representadas usualmente como botones, que facilitan al administrador la tarea de recopilar datos específicos según sus necesidades.

3. Opciones en la Interfaz:

La funcionalidad de Recopilar Datos Ventas implica la recuperación de información detallada sobre las transacciones de venta directamente desde la base de datos, almacenando posteriormente estos datos en un repositorio denominado DatosVentas. De manera similar, la función de Recopilar Datos Ingredientes se encarga de extraer información relevante acerca de todos los ingredientes utilizados, guardándola en un espacio de almacenamiento llamado DatosIngredientes. Por otro lado, Recopilar Datos Platos tiene como objetivo obtener los datos correspondientes a todos los platillos disponibles en el sistema, almacenándolos en DatosPlatos. Asimismo, la función de Recopilar Datos Financieros se dedica a recuperar la información económica y financiera del sistema, depositándola en DatosFinancieros. Finalmente, aunque en el diagrama no se establece una conexión directa con la base de datos, la función de Verificar Datos de Pedido sugiere la existencia de un proceso de revisión y análisis de datos relacionados con los pedidos realizados.

4. Base de Datos:

Contiene tablas donde se almacenan los diferentes tipos de datos extraídos desde la interfaz.

Flujo del proceso:

Cuando el administrador elige una de las opciones disponibles en la interfaz, el sistema ejecuta la acción asociada a esa opción, lo que conlleva la recopilación de datos específicos. Una vez recuperada la información, estos datos se almacenan de manera organizada en la tabla correspondiente dentro de la base de datos. Posteriormente, el administrador tiene la flexibilidad de verificar y consultar estos datos según sus requerimientos.

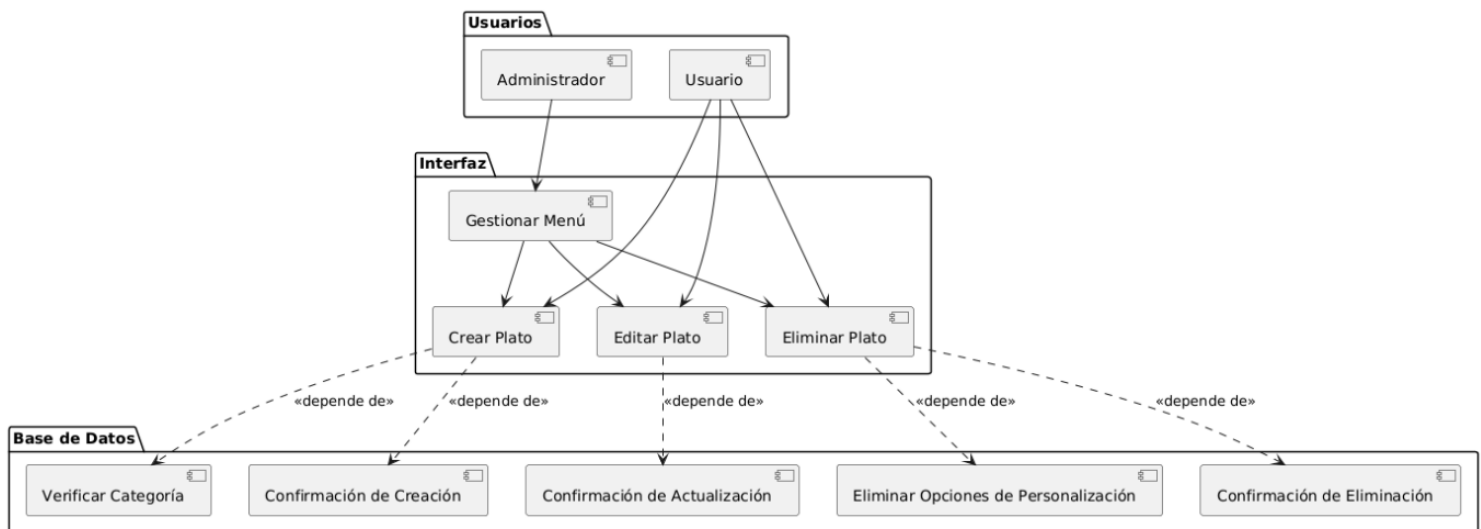
1.3 Visión General de la Composición.

Esta sección detalla el diseño interno de cada uno de los módulos del sistema de Comanda Digital, definiendo sus responsabilidades, funciones clave, entradas y salidas, así como las tecnologías empleadas.

En este apartado se utilizan diagramas de paquetes UML, el cual es una representación visual que agrupa los diferentes elementos de un sistema en módulos o paquetes lógicos. Su objetivo principal es organizar el diseño del sistema, mostrando cómo están relacionados los distintos componentes y cómo interactúan entre sí.

En este caso, el diagrama organiza los casos de uso del "Sistema de comanda" en paquetes que representan diferentes aspectos del sistema, permitiendo ver cómo los usuarios interactúan con las funciones de la interfaz y cómo estas dependen de la base de datos. Este tipo de diagrama ayuda a visualizar la relación entre los componentes del software, promoviendo el modularidad y el mantenimiento del código.

En los siguientes puntos, se ilustra la organización del Sistema de Comanda Digital, destacando la separación en capas y la interacción entre los módulos.



El Diagrama de Paquetes organiza los elementos del sistema en módulos lógicos, dividiendo los casos de uso en componentes más manejables. Se estructura en tres paquetes principales: Usuarios, Interfaz y Base de Datos.

El paquete "Usuarios" agrupa a los actores que interactúan con el sistema. Contiene dos roles: el *Administrador*, quien posee permisos avanzados para gestionar el menú del sistema, y el *Usuario*, quien puede interactuar con la gestión de platos. En cuanto a las relaciones, el Administrador tiene acceso a la funcionalidad de *Gestionar Menú*, mientras que tanto el Administrador como el Usuario pueden *Crear*, *Editar* y *Eliminar Platos*.

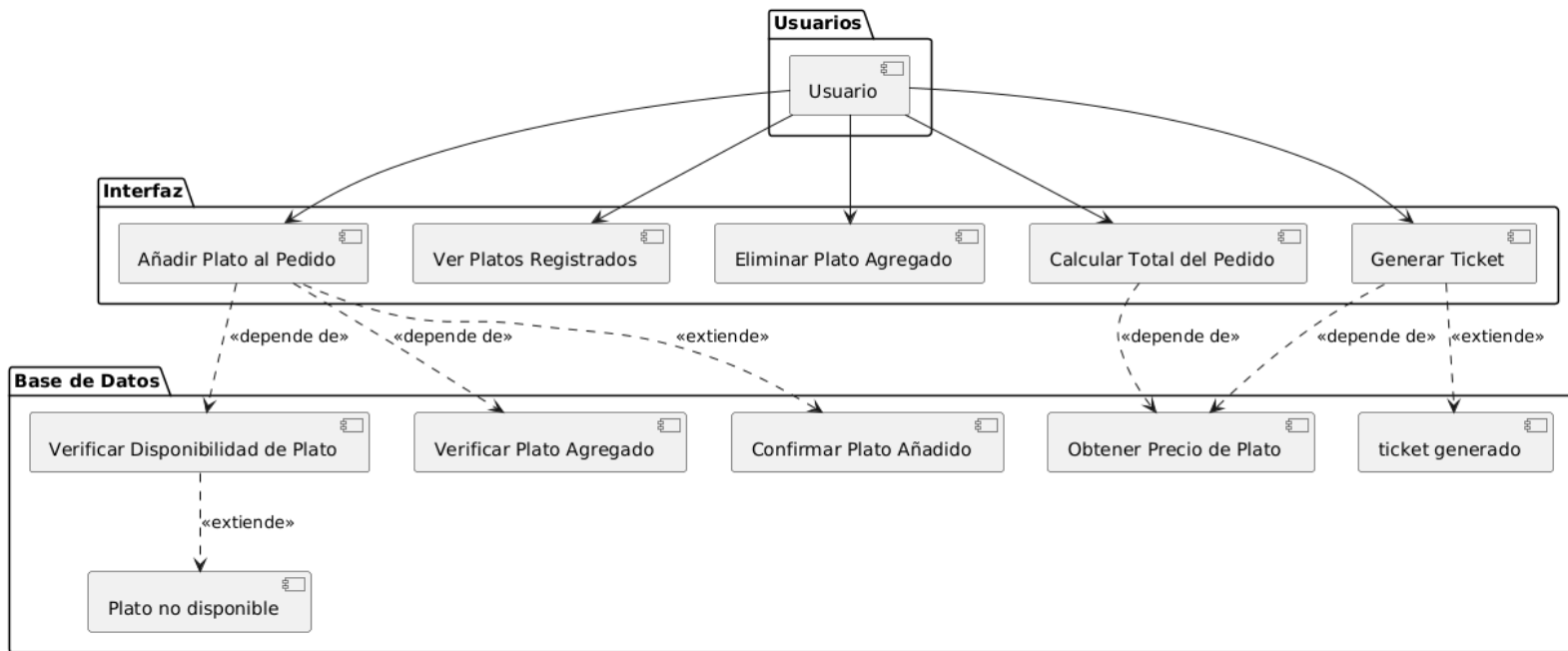
El paquete "Interfaz" representa las funcionalidades accesibles desde la interfaz del sistema. Incluye las acciones de *Gestionar Menú*, que permite acceder a la creación, edición y eliminación de platos; *Crear Plato*, que registra un nuevo plato; *Editar Plato*, que modifica los datos de un plato existente; y *Eliminar Plato*, que permite eliminar un plato del sistema. Las relaciones indican que la función *Gestionar Menú* se conecta con *Crear Plato*, *Editar Plato* y *Eliminar Plato*. Tanto el Administrador como el Usuario pueden utilizar estas funcionalidades.

El paquete "Base de Datos" agrupa los procesos encargados de gestionar la persistencia de datos. Contiene los siguientes elementos: *Confirmación de Creación*, que verifica que un plato ha sido registrado correctamente; *Verificar Categoría*, que comprueba la existencia de la categoría del plato antes de crearlo;

Confirmación de Actualización, que asegura que los cambios realizados a un plato se han guardado; *Eliminar Opciones de Personalización*, que elimina las opciones relacionadas antes de eliminar un plato; y *Confirmación de Eliminación*, que verifica que el plato ha sido eliminado exitosamente. Las relaciones muestran que *Crear Plato* depende de *Confirmación de Creación* y *Verificar Categoría*; *Editar Plato* depende de *Confirmación de Actualización*; y *Eliminar Plato* depende de *Eliminar Opciones de Personalización* y *Confirmación de Eliminación*.

Este diagrama permite visualizar la estructura y organización del Sistema de Comanda, así como las relaciones entre sus distintos módulos, facilitando el desarrollo, comprensión y mantenimiento del proyecto.

1.3.2 DP2 - Realización de pedidos.



El Diagrama de Paquetes de la funcionalidad de realización de pedidos en el *Sistema de Comanda* organiza los elementos en tres paquetes principales: Usuarios, Interfaz y Base de Datos.

El paquete "Usuarios" agrupa a los actores que interactúan con el sistema. Dentro de este paquete se encuentra el *Usuario*, quien es el actor principal encargado de realizar pedidos y gestionar los platos dentro de ellos. En cuanto a las relaciones, el Usuario puede interactuar con todas las funciones de la interfaz, incluyendo añadir platos al pedido, ver los platos registrados, eliminar platos agregados, calcular el total del pedido y generar un ticket.

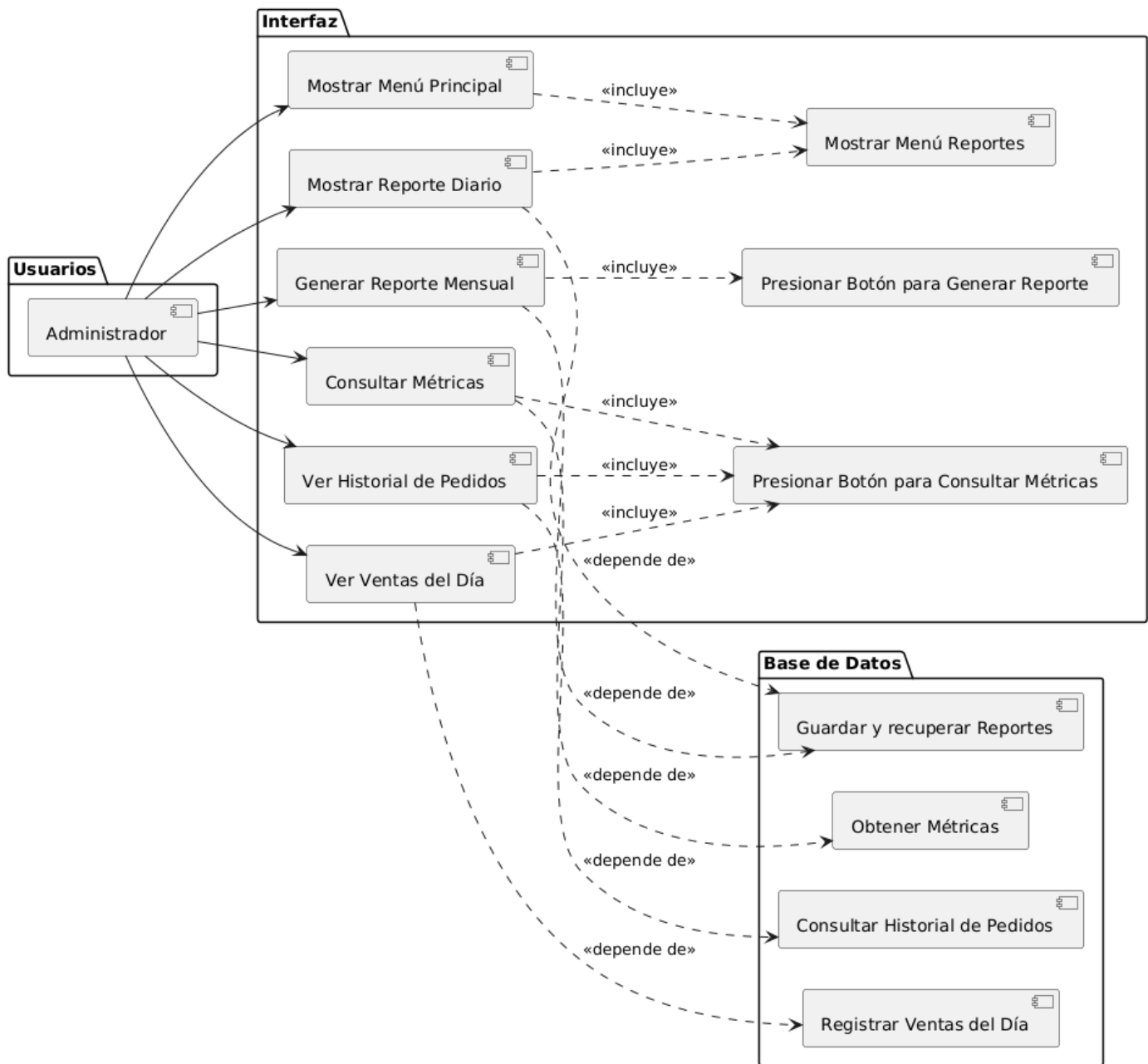
El paquete "Interfaz" representa las funciones disponibles para el Usuario en la interfaz del sistema. Este paquete incluye las siguientes funcionalidades: *Añadir Plato al Pedido*, que permite agregar un plato a un pedido en curso; *Ver Platos Registrados*, que muestra todos los platos disponibles en el sistema; *Eliminar Plato Agregado*, que permite eliminar un plato ya añadido; *Calcular Total del Pedido*, que calcula el monto total en función de los platos seleccionados; y *Generar Ticket*, que crea un ticket con el resumen del pedido. En cuanto a las relaciones, el Usuario interactúa directamente con estas funciones para gestionar su pedido. Además, la función *Añadir Plato al Pedido* depende de *Verificar Disponibilidad de Plato* y

Verificar Plato Agregado, con el fin de asegurar que el plato esté disponible y no haya sido agregado previamente. Por su parte, *Calcular Total del Pedido* y *Generar Ticket* dependen de *Obtener Precio de Plato* para calcular correctamente el monto final.

El paquete "Base de Datos" agrupa los procesos responsables de gestionar los datos del sistema. Este paquete contiene los siguientes elementos: *Verificar Disponibilidad de Plato*, que comprueba si un plato está disponible antes de ser agregado al pedido; *Verificar Plato Agregado*, que confirma si un plato ya fue añadido; *Obtener Precio de Plato*, que recupera el precio desde la base de datos; *Plato no disponible*, que se activa cuando un plato no se encuentra en el sistema; *Confirmar Plato Añadido*, que se ejecuta al agregar un plato exitosamente al pedido; y *Ticket generado*, que se activa cuando el sistema genera correctamente un ticket. Las relaciones muestran que *Añadir Plato al Pedido* depende tanto de *Verificar Disponibilidad de Plato* como de *Verificar Plato Agregado*, mientras que *Calcular Total del Pedido* y *Generar Ticket* dependen de *Obtener Precio de Plato* para determinar el costo total. Asimismo, si *Verificar Disponibilidad de Plato* determina que un plato no está en inventario, se activa *Plato no disponible*. Si un plato es añadido correctamente, se activa *Confirmar Plato Añadido*. Finalmente, cuando se genera un ticket exitosamente, se activa *Ticket generado*.

Este diagrama permite visualizar la organización interna del proceso de realización de pedidos dentro del *Sistema de Comanda*, mostrando de forma clara cómo las funcionalidades dependen unas de otras y cómo fluye la información entre la interfaz de usuario y la base de datos, facilitando así su análisis, desarrollo y mantenimiento.

1.3.3 DC3 - Gestión de Administradores



Este Diagrama de Paquetes ilustra cómo el *Administrador* interactúa con la interfaz del sistema y cómo esta se comunica a su vez con la base de datos. La estructura del diagrama facilita la comprensión del flujo de información en los procesos relacionados con la gestión de reportes y métricas dentro del *Sistema de Comanda*.

El paquete "Usuarios" agrupa a los actores que tienen acceso al sistema. Dentro de este paquete se encuentra el *Administrador*, quien es el usuario con acceso a funciones avanzadas como la visualización de reportes, métricas y datos de ventas. En cuanto a las relaciones, el Administrador tiene la capacidad de acceder a todas las funcionalidades de la interfaz del sistema.

El paquete "Interfaz" contiene las funciones disponibles para el administrador. Incluye las siguientes funcionalidades: *Mostrar Reporte Diario*, que permite visualizar el informe de ventas diarias; *Mostrar Menú Principal*, que da acceso a las distintas opciones administrativas; *Generar Reporte Mensual*, que permite obtener un resumen de ventas mensuales; *Consultar Métricas*, para revisar indicadores generales del sistema; *Ver Historial de Pedidos*, que muestra los registros de pedidos realizados; *Ver Ventas del Día*, que permite consultar las ventas de una fecha específica; y *Mostrar Menú Reportes*, que se activa cuando el administrador accede a la sección de reportes. También se incluyen eventos como *Presionar Botón para Generar Reporte*, que se activa al generar el reporte mensual, y *Presionar Botón para Consultar Métricas*, que se activa al consultar métricas, historial de pedidos o ventas del día.

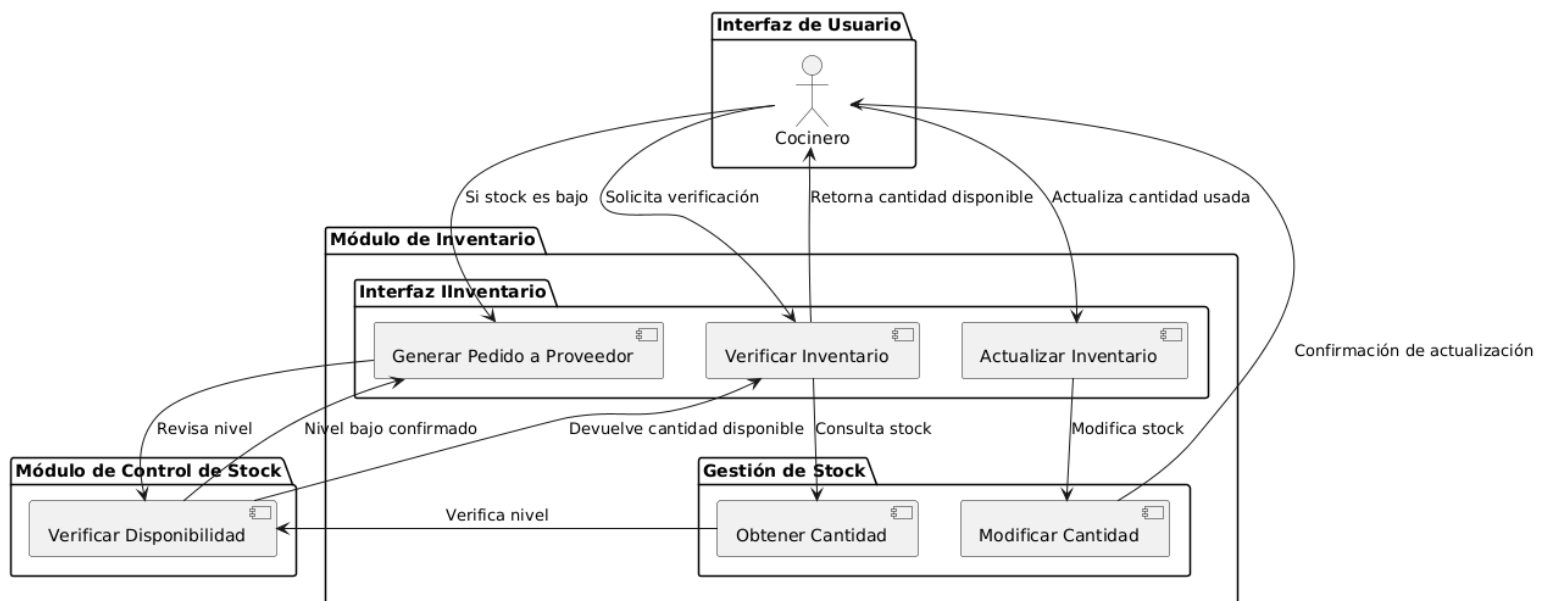
Las relaciones entre estos elementos reflejan cómo se conectan las funcionalidades. *Mostrar Reporte Diario* y *Mostrar Menú Principal* incluyen a *Mostrar Menú Reportes*, ya que ambas opciones permiten acceder a la sección de reportes. *Generar Reporte Mensual* incluye *Presionar Botón para Generar Reporte*, ya que es necesaria la acción del usuario para iniciar el proceso. Asimismo, *Consultar Métricas*, *Ver Historial de Pedidos* y *Ver Ventas del Día* incluyen *Presionar Botón para Consultar Métricas*, ya que todas requieren la interacción del usuario para mostrar los datos correspondientes.

El paquete "Base de Datos" contiene los procesos encargados de gestionar la persistencia de los datos relacionados con los reportes y métricas. Este paquete incluye los siguientes componentes: *Guardar y recuperar Reportes*, que permite almacenar y acceder a reportes previamente generados; *Obtener Métricas*, que recupera información clave sobre el funcionamiento del sistema; *Consultar Historial*

de Pedidos, que obtiene registros de pedidos realizados; y *Registrar Ventas del Día*, que guarda y consulta los datos sobre las ventas diarias.

Las relaciones en este paquete muestran las dependencias entre la interfaz y los procesos de base de datos. *Mostrar Reporte Diario* y *Generar Reporte Mensual* dependen de *Guardar y recuperar Reportes* para acceder a los datos correspondientes. *Consultar Métricas* depende de *Obtener Métricas*, mientras que *Ver Historial de Pedidos* se apoya en *Consultar Historial de Pedidos* para obtener los registros. Finalmente, *Ver Ventas del Día* depende de *Registrar Ventas del Día* para mostrar la información correspondiente a esa fecha.

Este diagrama proporciona una visión clara de la gestión administrativa del *Sistema de Comanda*, evidenciando cómo están organizadas las funcionalidades y cómo fluye la información entre los distintos módulos del sistema, lo que facilita su mantenimiento, comprensión y ampliación futura.



Este diagrama de paquetes representa la estructura modular del sistema, prescindiendo del uso de clases específicas y enfocándose exclusivamente en los paquetes y las interacciones que existen entre ellos. Su propósito es mostrar de forma clara cómo los diferentes módulos colaboran entre sí para garantizar una correcta gestión de ingredientes en el contexto del Sistema de Comanda, particularmente desde la perspectiva del cocinero como actor principal.

En la parte superior del diagrama se encuentra el paquete correspondiente a la Interfaz de Usuario, que incluye al cocinero como actor principal. Este usuario interactúa con el sistema a través de una interfaz gráfica, ejecutando acciones mediante botones o comandos que activan distintos procesos relacionados con el inventario de ingredientes. Aunque el cocinero no interactúa directamente con la base de datos o con los procesos internos, es quien inicia y controla las solicitudes de verificación y actualización de insumos.

La lógica del sistema se organiza a través de dos módulos funcionales: el Módulo de Inventario y el Módulo de Control de Stock. El Módulo de Inventario es el núcleo de la gestión operativa, ya que gestiona la consulta y actualización de la disponibilidad de ingredientes. Dentro de este módulo se encuentra la Interfaz de

Inventario, que expone las funcionalidades principales, como verificar la disponibilidad de un ingrediente, actualizar la cantidad utilizada y generar pedidos de reposición cuando se detectan niveles bajos de stock. Esta interfaz se comunica con un subcomponente denominado Gestión de Stock, el cual tiene como responsabilidad directa obtener y modificar la cantidad registrada de cada ingrediente. Este subcomponente permite consultar la cantidad actual a través de procesos internos y, cuando es necesario, actualiza estos valores en función del consumo de ingredientes.

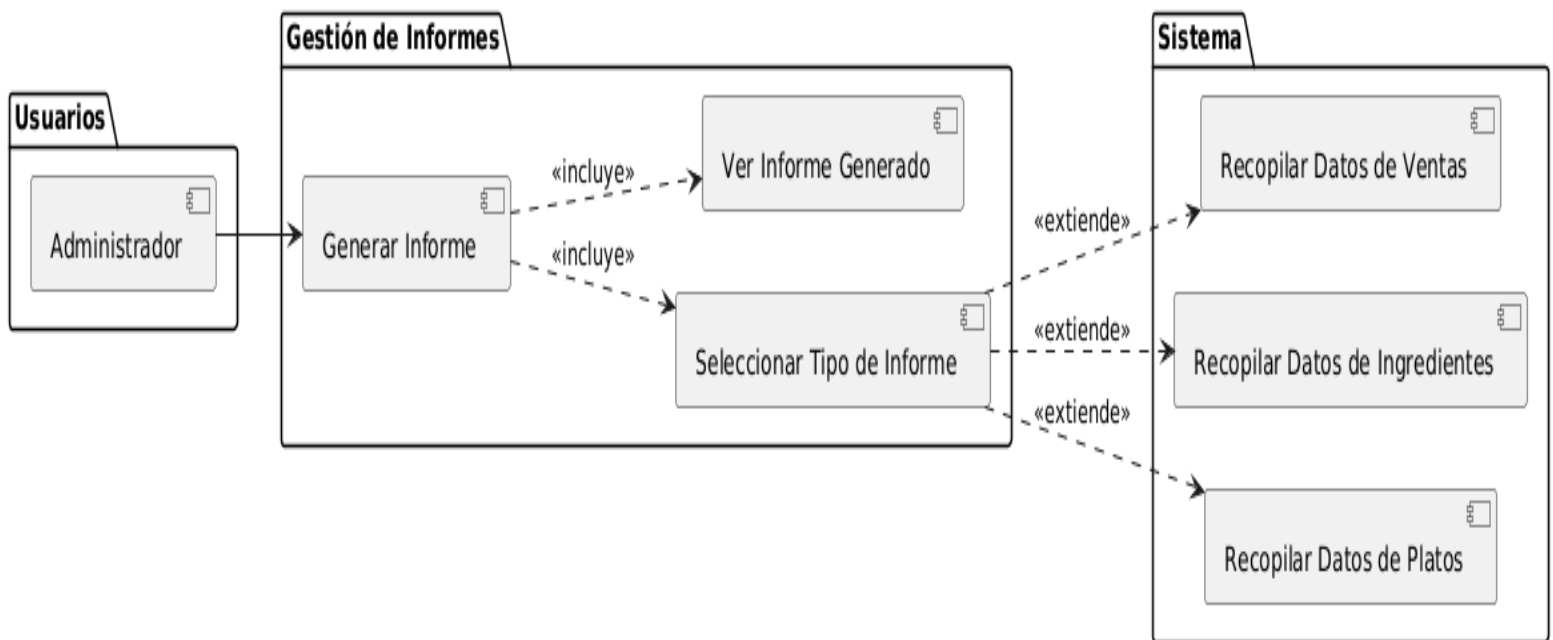
El Módulo de Control de Stock trabaja en conjunto con el Módulo de Inventario, actuando como una capa de verificación adicional que determina si la cantidad disponible de un ingrediente es suficiente para continuar con la preparación de un plato. Este módulo también participa en la generación de pedidos de reposición, ya que evalúa continuamente si los niveles registrados se encuentran por debajo del umbral mínimo requerido.

El flujo de interacción inicia cuando el cocinero solicita la verificación de un ingrediente desde la interfaz del sistema. Esta acción es procesada por la función correspondiente en la Interfaz de Inventario, la cual consulta la cantidad disponible mediante el submódulo de Gestión de Stock. Para ello, la función de obtención de cantidad recurre al Módulo de Control de Stock, que verifica si existe disponibilidad suficiente. Una vez verificada, la información es devuelta al cocinero. Si el ingrediente se encuentra disponible, el cocinero puede continuar con la operación; en caso contrario, el sistema notifica al usuario que no hay existencias.

Cuando se procede a utilizar un ingrediente, el sistema actualiza automáticamente el inventario. Este proceso es gestionado nuevamente por la Interfaz de Inventario, que modifica la cantidad registrada a través de la función de actualización de stock. Si como resultado de esta operación el nivel del ingrediente se reduce por debajo del mínimo permitido, el sistema activa la función de generación de pedido a proveedor. Para ello, consulta nuevamente el Módulo de Control de Stock, el cual confirma la necesidad de reposición, y si se cumple la condición, se genera un pedido para el proveedor correspondiente.

Este enfoque modular facilita la comprensión del sistema y promueve una arquitectura limpia, clara y escalable. La separación entre los roles de la interfaz de usuario, la gestión operativa del inventario y el control de stock permite distribuir de forma eficiente las responsabilidades de cada componente. Además, al no depender de clases específicas, este diseño es flexible y fácilmente adaptable a distintos entornos tecnológicos. En conjunto, el diagrama ofrece una representación eficaz de cómo se gestiona el flujo de información entre los actores y módulos del sistema durante el control de ingredientes, asegurando la disponibilidad y correcta administración de insumos en el contexto operativo del restaurante.

1.3.5 DP5-> Informe.



Este diagrama de paquetes muestra cómo el Administrador interactúa con la Interfaz de Gestión de Informes y cómo esta, a su vez, se comunica con el Sistema para recopilar los datos necesarios. La estructura del diagrama permite comprender de forma clara cómo se organizan y procesan los datos durante la generación de informes dentro del Sistema de Comanda, lo que resulta fundamental para la toma de decisiones administrativas.

En primer lugar, el Administrador actúa como el usuario principal de este módulo, siendo capaz de ejecutar distintas acciones como generar informes, seleccionar el tipo de informe deseado y visualizar los informes previamente generados. Todas estas acciones se canalizan a través de la Interfaz de Gestión de Informes, la cual actúa como intermediaria entre el usuario y los procesos internos del sistema. Esta interfaz se encarga de interpretar las solicitudes del administrador y de coordinar la interacción con los módulos que manejan el acceso a los datos.

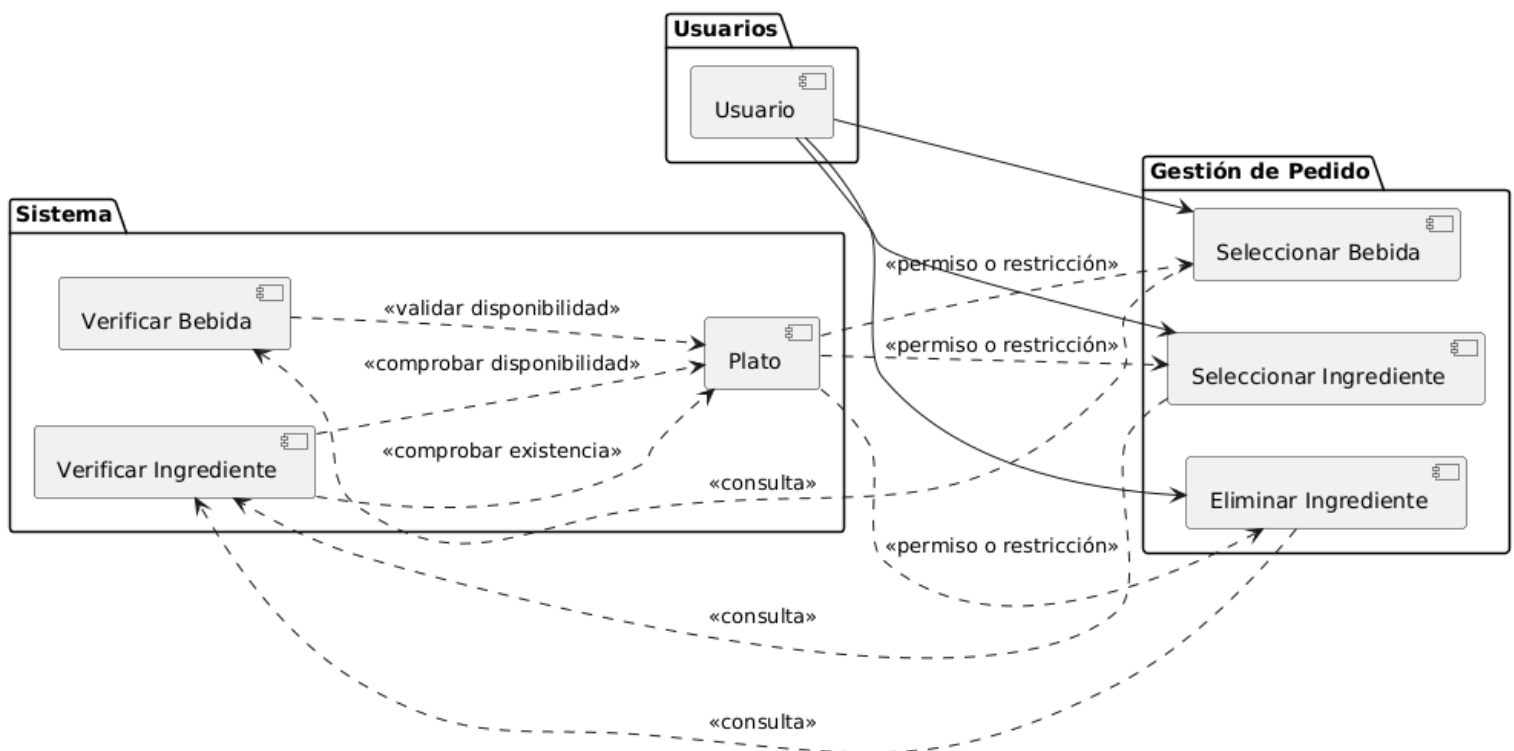
El funcionamiento general de la Interfaz de Gestión de Informes comienza con la acción del administrador, quien solicita la generación de un informe. Ante esta solicitud, la interfaz presenta al usuario la opción de seleccionar el tipo de informe requerido. En función de esta selección, el sistema accede a las fuentes de datos correspondientes en la base de datos para recopilar la información necesaria. Una vez que los datos han sido recuperados y procesados, la interfaz presenta el informe generado, mostrando los resultados de manera comprensible y estructurada. Por ejemplo, si el administrador selecciona la opción de generar un informe, la interfaz activa automáticamente la funcionalidad de selección del tipo de informe. Si el tipo de informe elegido corresponde a ventas, se activa el proceso de recopilación de datos de ventas, el cual accede al sistema para recuperar la información almacenada sobre transacciones. Una vez obtenidos los datos, se procesan internamente y se genera un informe, el cual es mostrado al administrador en la misma interfaz.

La comunicación entre la interfaz y el sistema es esencial para la obtención de datos precisos y actualizados. Para ello, el sistema dispone de funciones específicas que permiten acceder a las diferentes categorías de datos. Por ejemplo, los informes de ventas se generan a partir de la función de recopilación de datos de ventas. Del mismo modo, los informes relacionados con ingredientes acceden a la función correspondiente que recupera datos sobre disponibilidad, consumo o reposición de insumos. En el caso de informes sobre platos, el sistema consulta la información vinculada a los registros de los distintos platillos del menú.

En cuanto a las dependencias entre componentes, la acción de generar un informe incluye tanto la selección del tipo de informe como la visualización del informe generado. La funcionalidad de seleccionar el tipo de informe extiende su comportamiento en función del tipo de informe elegido, ya que determina cuál de los procesos de recopilación de datos será activado. Asimismo, la Interfaz de Gestión de Informes mantiene una dependencia directa del sistema, ya que requiere acceso a los datos para poder presentar resultados completos y coherentes.

En conjunto, este diagrama permite visualizar de forma clara y estructurada cómo se organizan los módulos implicados en la generación de informes. La interfaz cumple un rol central al actuar como puente entre el administrador y el sistema, garantizando un flujo de información organizado, seguro y eficiente para la consulta de reportes dentro del Sistema de Comanda. Esta separación lógica entre interfaz, procesos de selección, recopilación de datos y generación de informes favorece la mantenibilidad y escalabilidad del sistema.

1.3.6 DP6-> Personalización de pedidos.



Este diagrama de paquetes organiza la estructura del sistema en tres niveles claramente diferenciados, lo que permite visualizar de forma efectiva cómo se distribuyen las responsabilidades entre los distintos módulos y cómo fluye la información entre ellos. En el primer nivel se encuentra el paquete correspondiente a los Usuarios. Este nivel representa al usuario final, quien interactúa directamente con el sistema para personalizar su pedido. A través de esta interfaz, el usuario puede seleccionar ingredientes y bebidas que desee añadir a su plato, o bien eliminar ingredientes previamente seleccionados. Todas estas acciones constituyen solicitudes que son gestionadas por los niveles inferiores del sistema.

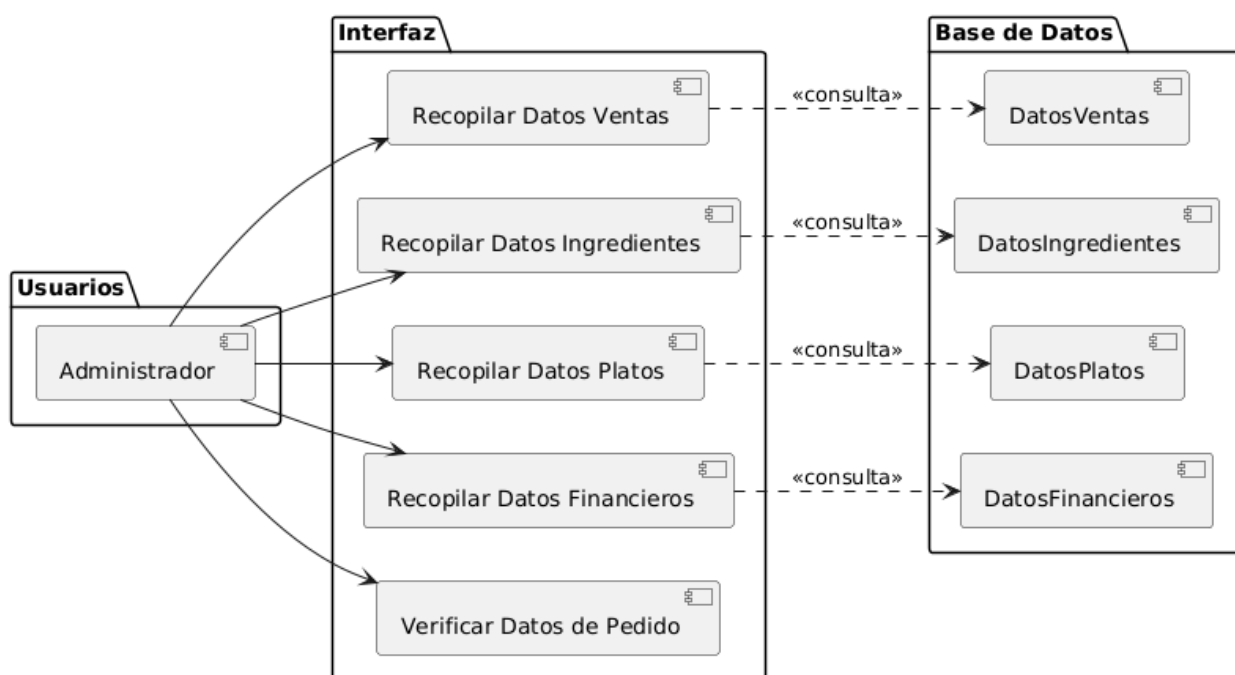
El segundo nivel corresponde al paquete de Gestión de Pedido. Este módulo actúa como intermediario entre el usuario y el sistema. Se encarga de manejar las opciones seleccionadas por el usuario, como la adición o eliminación de ingredientes y la elección de una bebida. Cada vez que el usuario realiza una acción, esta se traduce en una solicitud que se envía al tercer nivel del sistema para verificar la validez y disponibilidad de los elementos seleccionados. Este paquete no toma decisiones por sí mismo, sino que canaliza las peticiones y recibe respuestas desde el sistema para completar o rechazar las acciones.

El tercer nivel corresponde al núcleo del Sistema. Aquí se encuentra la lógica de negocio encargada de validar las solicitudes recibidas. Este módulo se responsabiliza de verificar la existencia y disponibilidad de los ingredientes y bebidas en base a las reglas definidas para cada tipo de plato. Dentro de este nivel, el módulo denominado Plato desempeña un rol clave al determinar si un ingrediente o una bebida seleccionados son válidos dentro del contexto del pedido. Si la selección realizada por el usuario cumple con los criterios establecidos y el elemento está disponible, el sistema confirma la acción. En caso contrario, se devuelve un mensaje de error que notifica al usuario que el ingrediente o bebida no se encuentra disponible o no puede ser agregado al plato.

El flujo de interacción entre los niveles sigue una secuencia estructurada. Cuando el usuario selecciona un ingrediente, el sistema verifica si dicho ingrediente está permitido para el plato en cuestión. Si la verificación es positiva, se confirma la

selección. Si no, el sistema informa al usuario que el ingrediente no está disponible o no es válido. En el caso de que el usuario desee eliminar un ingrediente, el sistema comprueba si ese ingrediente forma parte del plato actual. Si está presente, se permite la eliminación; de lo contrario, se notifica al usuario que no es posible eliminar un ingrediente inexistente. Del mismo modo, cuando el usuario selecciona una bebida, el sistema verifica su disponibilidad. Si la bebida está registrada y disponible, se confirma la selección. Si no está en inventario o no es válida, el sistema muestra un mensaje de error correspondiente. En conjunto, este diagrama permite entender con claridad cómo se organizan y vinculan las acciones del usuario con la lógica del sistema, facilitando un flujo de información ordenado y eficiente. La estructura en niveles asegura una separación de responsabilidades que mejora la mantenibilidad del sistema y permite escalar o modificar componentes de manera independiente sin afectar el funcionamiento general. Esta organización modular resulta esencial para el correcto funcionamiento del Sistema de Comanda en contextos donde la personalización del pedido es un elemento central en la experiencia del usuario.

1.3.7 DP7 - Recopilación de datos.



Este diagrama de paquetes representa la estructura de interacción entre el Administrador, la interfaz del sistema y la base de datos dentro del contexto del Sistema de Comanda. Su organización permite comprender de forma precisa cómo fluye la información entre los distintos niveles del sistema y cómo se gestiona el acceso a los datos administrativos.

En el primer nivel se encuentra el paquete correspondiente a los usuarios, donde el Administrador es el único actor con privilegios para acceder a las funciones relacionadas con la gestión y análisis de datos. A través de sus acciones, el Administrador puede solicitar información clave del sistema, incluyendo datos de ventas, registros de ingredientes, detalles sobre los platos disponibles, así como información financiera. Además, tiene la capacidad de verificar información específica de los pedidos realizados en el sistema.

El segundo nivel lo ocupa el paquete de la Interfaz. Esta actúa como intermediaria directa entre el Administrador y la base de datos. Cada vez que el Administrador realiza una solicitud, la interfaz se encarga de traducir dicha solicitud en una consulta dirigida a la base de datos. Posteriormente, recupera los datos requeridos y los presenta de manera accesible y organizada al usuario. Esta capa es fundamental para garantizar una experiencia fluida, ya que no solo gestiona la comunicación, sino que también se ocupa de estructurar y mostrar la información de forma clara, permitiendo al Administrador tomar decisiones con base en datos actualizados.

El tercer nivel del sistema está compuesto por la Base de Datos. Este paquete contiene toda la información almacenada relacionada con ventas, ingredientes, platos, pedidos y finanzas. La base de datos es responsable de responder de manera eficiente y confiable a las solicitudes de la interfaz, asegurando que los datos entregados estén actualizados y sean consistentes con la actividad registrada en el sistema. Además de almacenar la información, también garantiza su integridad y disponibilidad continua. El flujo de interacción entre estos componentes sigue una secuencia lógica. Cuando el Administrador solicita, por ejemplo, los datos de ventas, la interfaz transmite esta solicitud a la base de datos, la cual recupera la información necesaria y la devuelve a la interfaz, que finalmente

presenta los resultados al Administrador. Este mismo proceso se replica para la recopilación de datos de ingredientes, platos y finanzas. En cada caso, la interfaz traduce la solicitud, consulta la base de datos, y muestra los resultados. Asimismo, cuando el Administrador desea verificar los datos de un pedido específico, el proceso sigue el mismo patrón: la interfaz localiza los datos en la base de datos y los presenta al usuario. En cuanto a las dependencias entre los componentes, la interfaz depende directamente de la base de datos para acceder a la información del sistema. A su vez, el Administrador depende de la interfaz para consultar y visualizar estos datos de manera eficiente. Finalmente, la base de datos actúa como el núcleo que almacena y organiza toda la información utilizada en los distintos procesos de gestión y control administrativo.

En resumen, este diagrama de paquetes permite visualizar con claridad cómo se estructura el sistema en torno a tres niveles funcionales bien definidos: el usuario administrador, la interfaz de interacción, y la base de datos como fuente central de información. Esta arquitectura garantiza un flujo de datos eficaz, una gestión ordenada y una respuesta oportuna a las necesidades de información del Administrador, asegurando así la precisión, trazabilidad y accesibilidad de los datos dentro del Sistema de Comanda.

1.4 Punto de Vista Lógico.

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.

1.4.1 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.

1.4.2 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.

Módulo de Informes.

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.

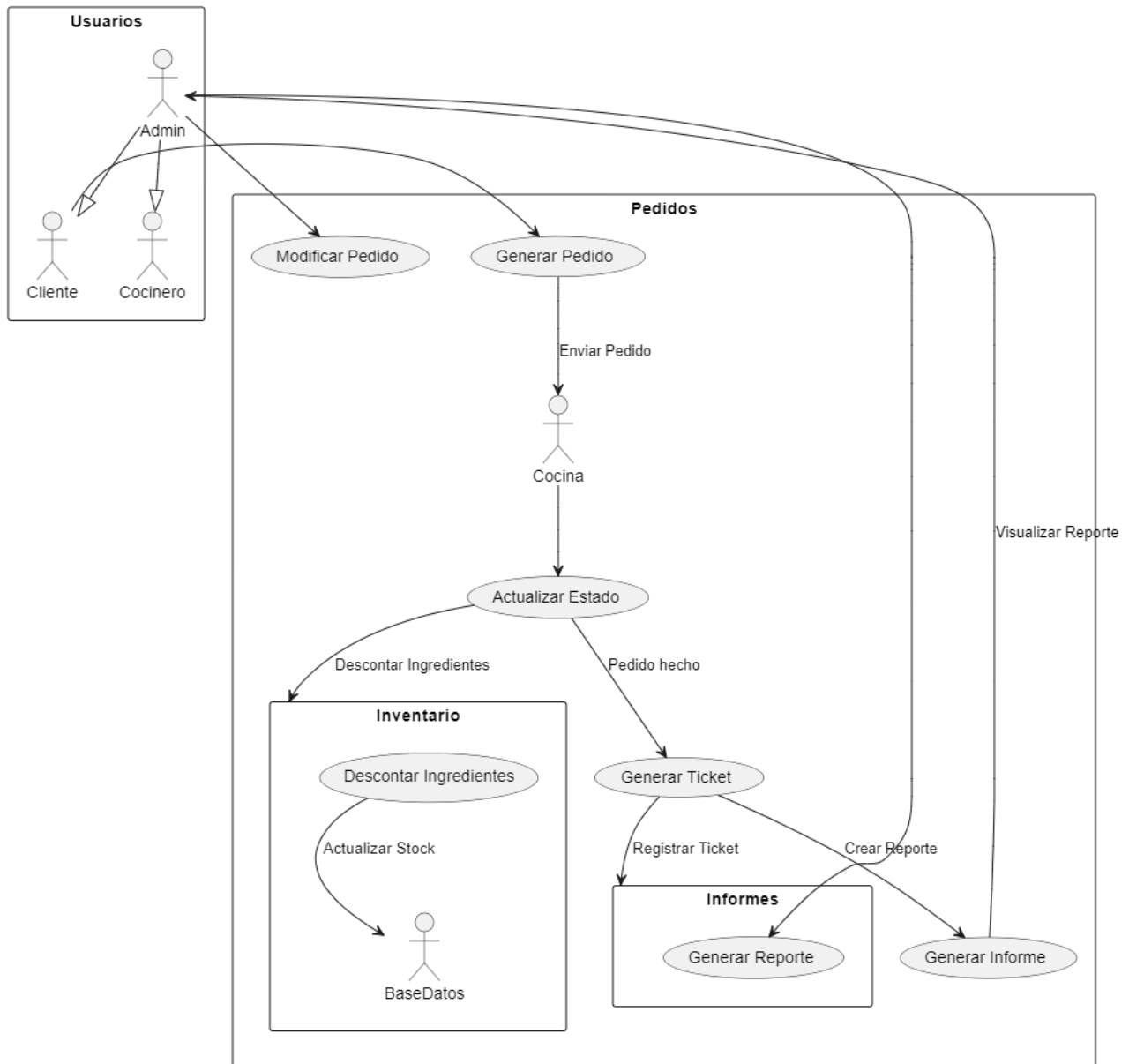
Módulo	Interacción con	Descripción
Usuarios	Pedidos, Inventario, Informes, Carrito.	Los clientes pueden ver el menú y realizar pedidos dentro del carrito. Los administradores gestionan pedidos, inventario e informes.
Pedidos	Inventario, Ticket, Informes.	Los pedidos afectan el inventario y generan tickets. 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

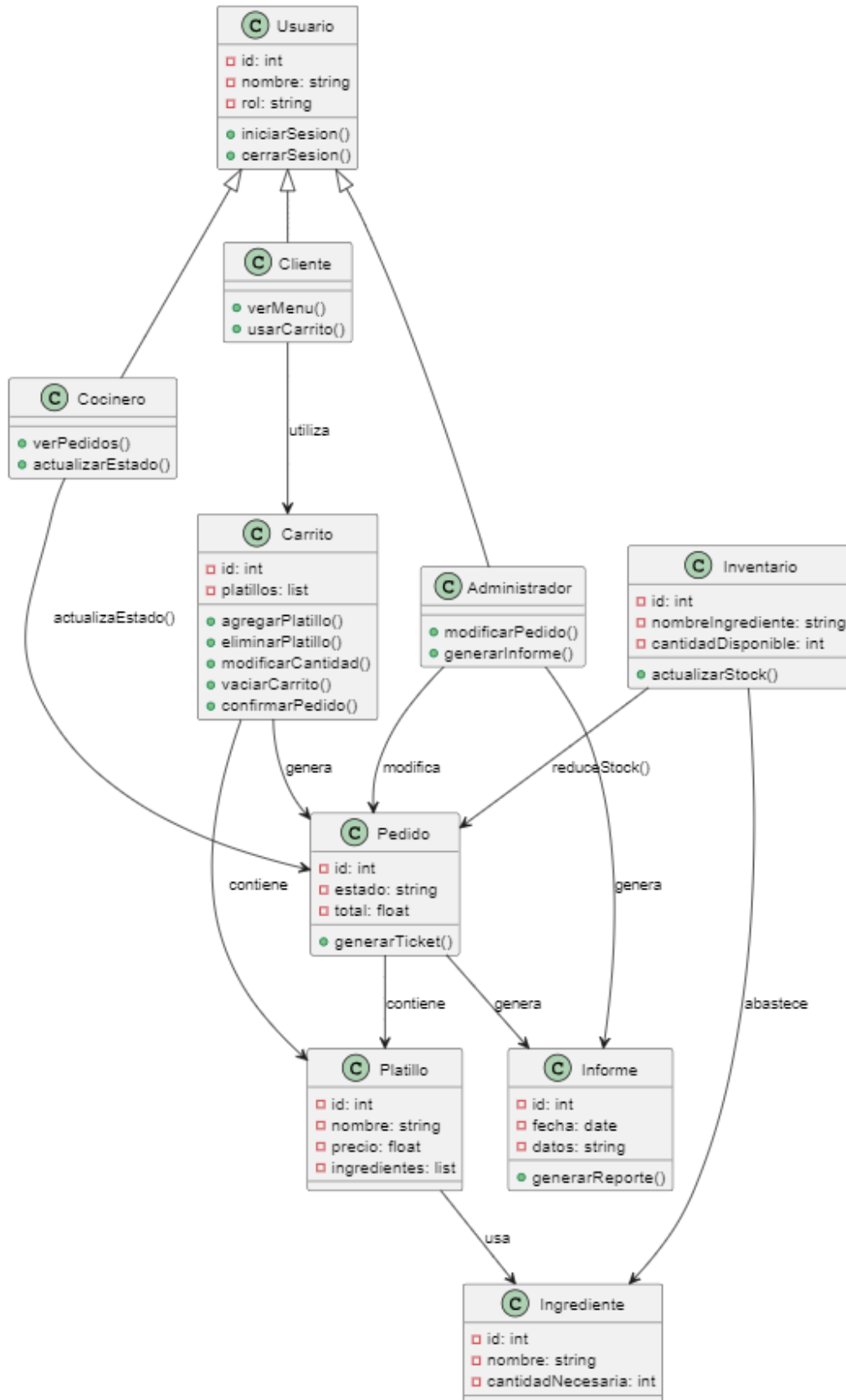
Administrador y el Cocinero. Cada uno tiene responsabilidades distintas, pero todos comparten la capacidad de iniciar y cerrar sesión.

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.

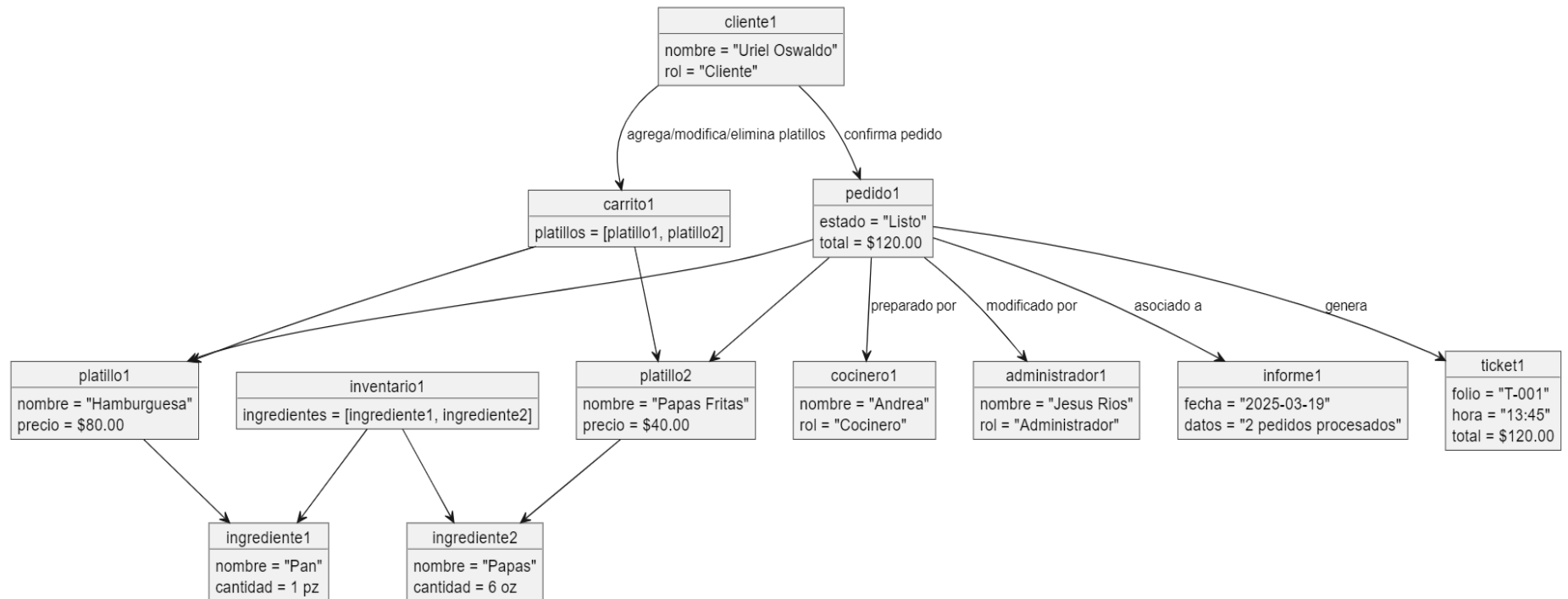


El siguiente diagrama de objetos representa una instancia particular del sistema de pedidos de platillos en línea, donde se visualizan los objetos creados durante la operación típica de un cliente que realiza un pedido. Este ejemplo se enfoca en mostrar cómo los objetos interactúan entre sí en un momento 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.



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.

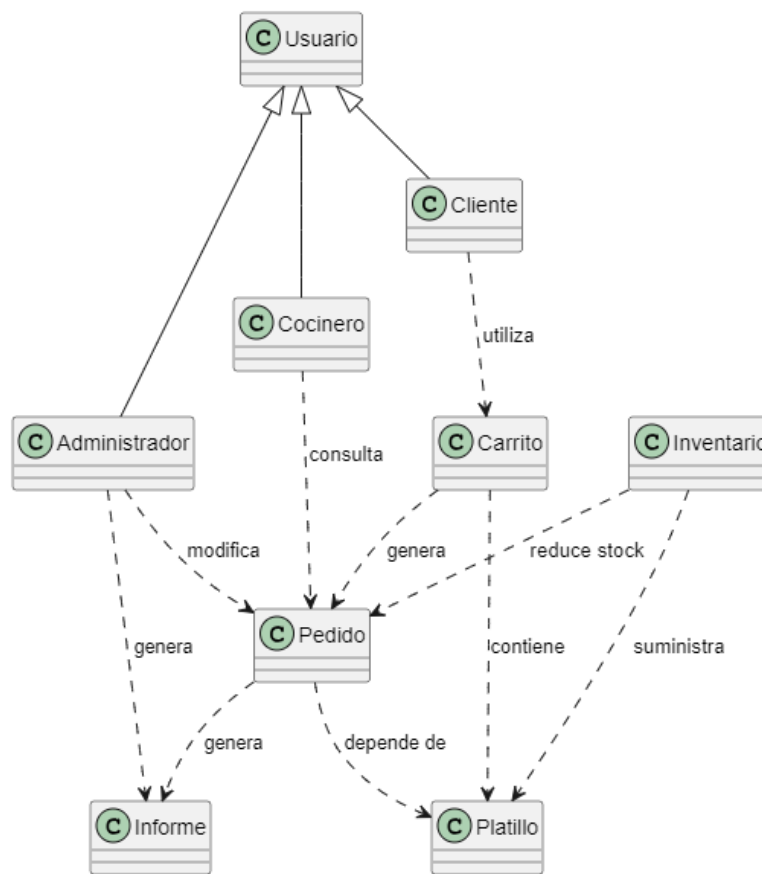
El componente **Pedido**, por su parte, depende del módulo **Inventario** para verificar la disponibilidad de los ingredientes al momento de confirmar la orden. Esta relación se da mediante una función de consulta y reducción de stock, controlada por métodos definidos en la clase **Inventario**, evitando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

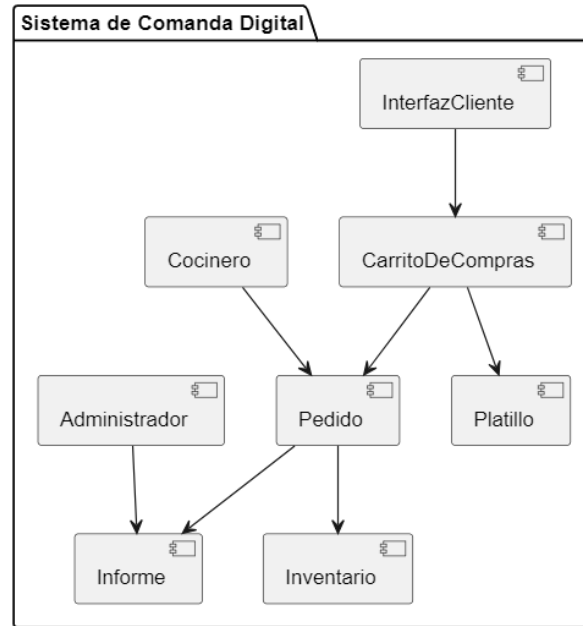
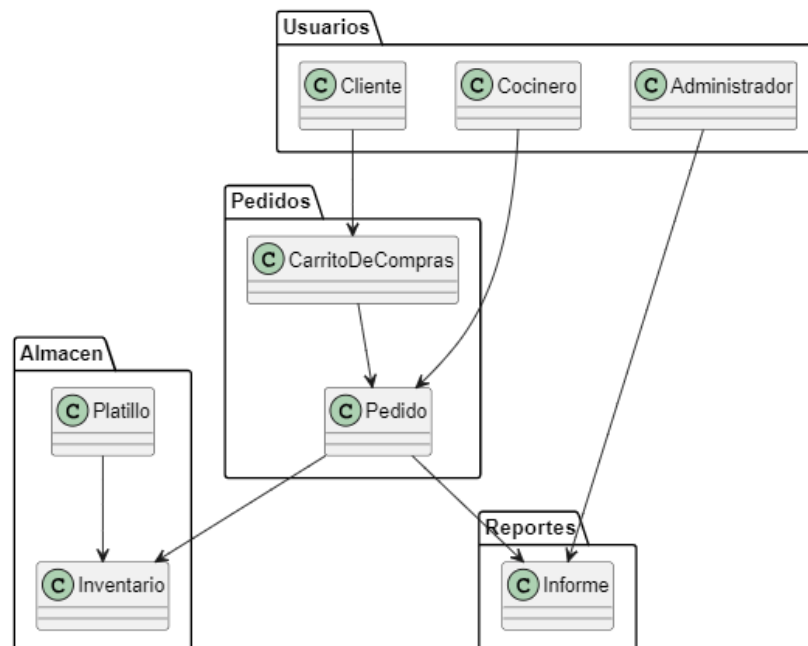


Diagrama de paquetes.

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.



1.6 Los patrones utilizan el punto de vista

1.6.1 Aspectos clave del diseño

El diseño del sistema se basa en una arquitectura modular y flexible, fundamentada en la aplicación de patrones de diseño ampliamente aceptados en la ingeniería de software. Esto permite no solo una implementación coherente y bien estructurada, sino también una solución escalable y mantenible a largo plazo.

Uno de los aspectos clave del diseño es la adopción del patrón **Modelo-Vista-Controlador (MVC)**. Este patrón se refleja directamente en la organización del sistema: por ejemplo, las clases como cliente1, platillo1, carrito1, pedido1 y inventario1 representan el **modelo**, ya que encapsulan la lógica de negocio y el estado de los datos. La **vista**, en un entorno gráfico o web, sería responsable de mostrar la información almacenada en estas clases al usuario (como los platillos disponibles o los pedidos en curso). El **controlador**, que estaría representado por clases como PedidoController o CarritoController (en una implementación orientada), se encargaría de procesar las acciones del usuario (agregar platillos, confirmar pedidos, etc.) y coordinar los modelos y las vistas.

Para reforzar la comunicación entre componentes, se emplea el patrón **Observer**. Este patrón es esencial para actualizar automáticamente a los usuarios del sistema sobre eventos importantes. Por ejemplo, cuando un pedido (pedido1) cambia de estado a "Listo", tanto el cocinero (cocinero1) como el administrador (administrador1) son notificados. Esta dinámica es clave para mantener el flujo de trabajo ágil y actualizado sin necesidad de comprobaciones constantes.

Además, el patrón **Singleton** se aplica en clases que deben tener una única instancia global, como informe1, encargada de generar reportes del sistema. Solo debe existir una instancia de esta clase para evitar duplicidad en la generación de informes o inconsistencias en la lectura de datos administrativos, garantizando así una fuente única de verdad.

Otro patrón presente es la **Inyección de Dependencias**, especialmente útil al vincular componentes como el inventario (inventario1) con los platillos (platillo1, platillo2). En lugar de que las clases se instancien entre sí de forma directa, las

dependencias se inyectan desde el exterior, permitiendo mayor flexibilidad, facilidad de pruebas y bajo acoplamiento. Por ejemplo, un servicio `InventarioService` podría inyectar dinámicamente los ingredientes (`ingrediente1`, `ingrediente2`) necesarios a los distintos platillos según la disponibilidad.

La arquitectura del sistema permite seguir la trazabilidad completa del pedido desde que el cliente (`cliente1`, como “Uriel Oswaldo”) agrega productos al `carrito1`, hasta que se confirma como `pedido1`, es preparado por `cocinero1`, revisado por `administrador1`, asociado a `informe1` y finalmente se genera el `ticket1` correspondiente. Esta secuencia lógica está respaldada por relaciones claras entre clases y el uso estratégico de patrones.

Por último, los principios SOLID están presentes en el diseño: por ejemplo, cada clase como `cliente1`, `platillo1` o `ticket1` cumple con el **principio de responsabilidad única**, ya que cada una tiene un propósito bien definido. Asimismo, el sistema está diseñado para ser extensible sin modificar las clases existentes, lo que refleja el **principio abierto/cerrado**. Todo esto contribuye a una estructura sólida, preparada para adaptarse a nuevas funcionalidades como métodos de pago, historial de pedidos, o reportes gráficos

1.6.2 Elementos de diseño

Los elementos de diseño del sistema reflejan una estructura arquitectónica centrada en patrones de colaboración, clases especializadas, conectores abstractos y roles funcionales claramente definidos. A continuación, se describen los distintos componentes conforme a la norma.

En cuanto a las entidades de diseño, el sistema exhibe múltiples colaboraciones entre entidades clave. Por ejemplo, la colaboración entre las clases `cliente1` y `carrito1` permite capturar la lógica de selección de productos; a su vez, la clase `pedido1` colabora con `platillo1` e `inventario1` para generar la solicitud de cocina.

Las clases principales en el sistema incluyen `cliente1`, `carrito1`, `pedido1`, `platillo1`, `ingrediente1`, `cocinero1`, `administrador1`, `informe1` y `ticket1`. Cada una representa

una unidad funcional con responsabilidades específicas, alineadas al principio de responsabilidad única (SRP).

Existen conectores lógicos que permiten la interacción entre entidades. Por ejemplo, el conector entre carrito1 y pedido1 representa la transición de elementos seleccionados hacia un pedido formal. Otro conector importante es el que existe entre pedido1 e informe1, ya que establece la transferencia de estado para reportes administrativos.

Cada entidad cumple un rol claro dentro de una colaboración. Por ejemplo, cliente1 actúa como iniciador de la interacción, cocinero1 como ejecutor del pedido, e informe1 como agregador de datos. Los roles definen comportamientos, no estructuras, lo que permite extender funcionalidades sin alterar la arquitectura base.

Se utiliza el marco arquitectónico MVC. En este modelo, platillo1, carrito1 y pedido1 fungen como el modelo; las vistas estarían representadas por formularios y pantallas de gestión; y el controlador sería una capa lógica que orquesta las operaciones, posiblemente con clases como **PedidoController** o **InventarioService**. También se planea utilizar un **framework** de desarrollo como Spring **Boot** o Laravel para facilitar la implementación modular, con soporte para patrones como la inyección de dependencias y el manejo de eventos.

Respecto a los patrones utilizados, se aplica el patrón MVC para separar presentación, lógica y datos. Además, se utiliza el patrón **Observer** para notificar estados de pedidos a cocineros y administradores, y el patrón **Singleton** en la clase informe1 para generar reportes únicos. También se aplica la inyección de dependencias entre servicios e inventarios.

En cuanto a las relaciones de diseño, cliente1 está asociado a carrito1 mediante una relación directa uno-a-uno. pedido1 está asociado a uno o varios platillos (como platillo1 o platillo2), y también a un ticket1. Además, cocinero1 y administrador1 usan los datos de pedido1 para sus funciones. carrito1 colabora con cliente1 para construir una orden válida, mientras que informe1 colabora con pedido1 y ticket1 para consolidar estadísticas del sistema.

Los conectores actúan como vínculos lógicos entre las entidades. Por ejemplo, el cambio de estado en pedido1 (de “en preparación” a “listo”) activa un conector de evento que comunica esta información a cocinero1 y administrador1. Este conector puede implementarse utilizando el patrón **Observer**.

Cada entidad está correctamente identificada en el modelo: cliente1, carrito1, platillo1, pedido1, entre otras. Esto facilita el rastreo de interacciones y mantiene el sistema coherente.

Finalmente, existen algunas restricciones de diseño importantes. Por ejemplo, un cliente1 solo puede tener un carrito1 activo, y un pedido1 no puede generarse si el carrito1 está vacío. Además, informe1 solo puede generarse a partir de pedidos completados, y un cocinero1 no puede modificar el pedido una vez que se genera el ticket1. Estas restricciones se implementarán mediante validaciones internas en los métodos del sistema y a través de reglas de negocio en la lógica de los controladores.

1.6.2.1 Patrones: Se aplican varios patrones:

Modelo-Vista-Controlador:

(MVC)

El sistema sigue el patrón MVC para organizar la lógica. Las clases cliente1, carrito1, platillo1 y pedido1 representan el Modelo. Los formularios de entrada y salida (en interfaz web o móvil) representan las Vistas. La lógica de control se centraliza en controladores como PedidoController.

Ejemplo (controlador procesando un pedido):

```
php PedidoController.php x +
php PedidoController.php
1 class PedidoController {
2     protected $pedidoService;
3
4     public function __construct(PedidoService $pedidoService) {
5         $this->pedidoService = $pedidoService;
6     }
7
8     public function confirmar(Request $request) {
9         $this->pedidoService->procesar($request->input('carrito_id'));
10    }
11 }
12 |
```

Este fragmento demuestra el uso de MVC e Inyección de Dependencias en el constructor

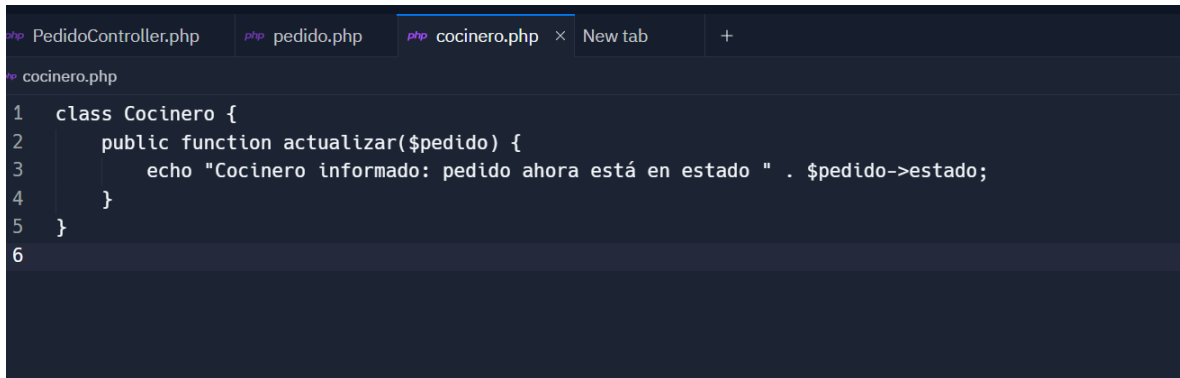
Observer (Observador)

El patrón Observer permite que cocinero1 y administrador1 se mantengan informados sobre cambios en pedido1.

Ejemplo (pedido notifica a observadores):

```
php PedidoController.php php pedido.php x php cocinero.php x New tab +
pedido.php
1 class Pedido {
2     private $estado;
3     private $observadores = [];
4
5     public function agregarObservador($observador) {
6         $this->observadores[] = $observador;
7     }
8
9     public function cambiarEstado($nuevoEstado) {
10        $this->estado = $nuevoEstado;
11        $this->notificar();
12    }
13
14    private function notificar() {
15        foreach ($this->observadores as $obs) {
16            $obs->actualizar($this);
17        }
18    }
19 }
20
```

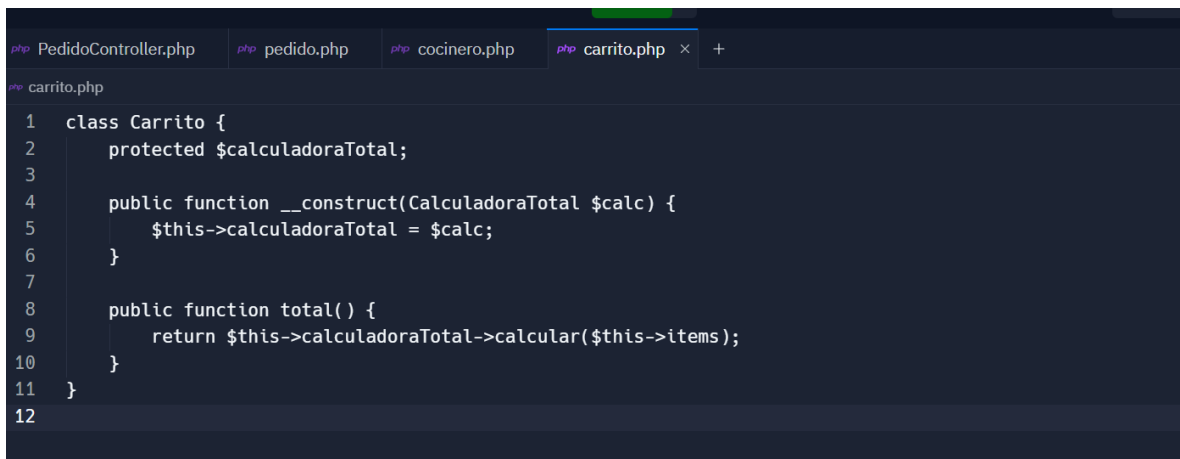
Y un ejemplo de observador:



```
php PedidoController.php  php pedido.php  php cocinero.php x New tab +
php cocinero.php
1  class Cocinero {
2      public function actualizar($pedido) {
3          echo "Cocinero informado: pedido ahora está en estado " . $pedido->estado;
4      }
5  }
6
```

Inyección de Dependencias

Ejemplo: carrito1 requiere un servicio de cálculo de totales:

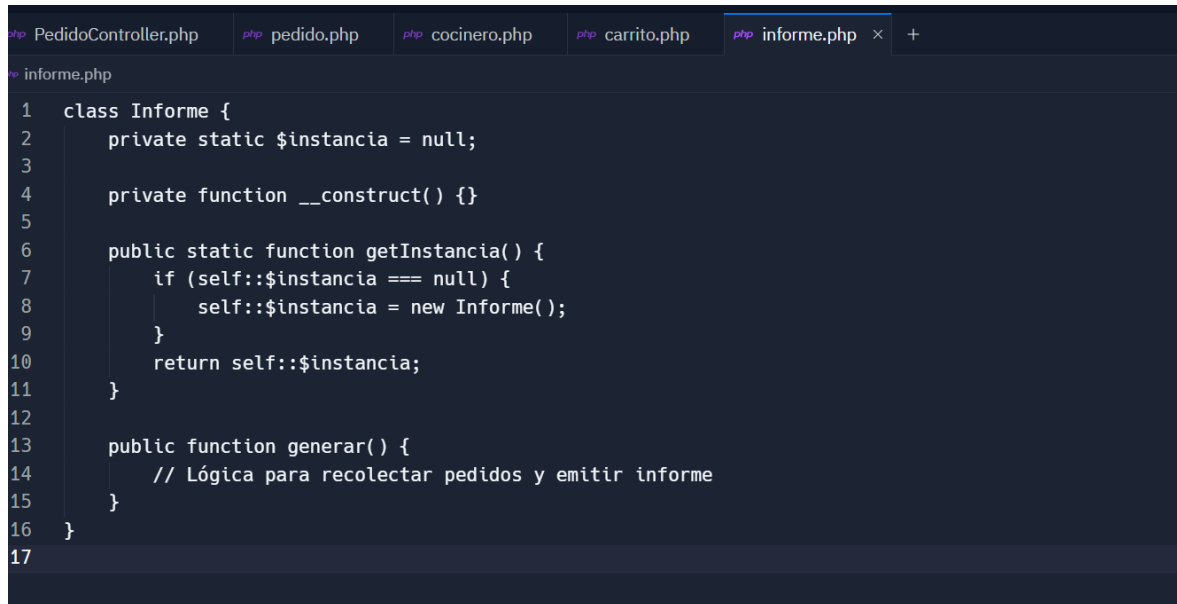


```
php PedidoController.php  php pedido.php  php cocinero.php  php carrito.php x +
php carrito.php
1  class Carrito {
2      protected $calculadoraTotal;
3
4      public function __construct(CalculadoraTotal $calc) {
5          $this->calculadoraTotal = $calc;
6      }
7
8      public function total() {
9          return $this->calculadoraTotal->calcular($this->items);
10     }
11 }
12
```

Esto permite cambiar la implementación de cálculo sin modificar la clase Carrito.

Singleton (ej. informe1)

La clase informe1 es instanciada una única vez para garantizar consistencia en los reportes generados.



```
1 class Informe {
2     private static $instancia = null;
3
4     private function __construct() {}
5
6     public static function getInstance() {
7         if (self::$instancia === null) {
8             self::$instancia = new Informe();
9         }
10        return self::$instancia;
11    }
12
13    public function generar() {
14        // Lógica para recolectar pedidos y emitir informe
15    }
16 }
17
```

Esto asegura que cualquier controlador o módulo que necesite generar un reporte acceda a la misma instancia compartida

1.6.3 Restricciones del punto de vista

Esta sección define las reglas, notaciones y estructuras necesarias para garantizar que el diseño basado en patrones se mantenga coherente, correcto y aplicable durante la implementación. Para ello, se incluye el uso de lenguajes estructurados, restricciones arquitectónicas y el diagrama de estructura compuesta como lenguaje de representación.

Para describir las restricciones y relaciones entre entidades, se utilizaron dos lenguajes complementarios. Por un lado, se empleó el Lenguaje Unificado de Modelado (UML), especialmente el diagrama de estructura compuesta, el cual permite representar visualmente los roles, conectores y colaboraciones entre clases como cliente1, carrito1, pedido1, cocinero1, informe1, entre otras. Por otro lado, también se usó un lenguaje estructurado, mediante fragmentos de código en PHP

con estilo Laravel, para mostrar cómo se aplican e implementan los patrones arquitectónicos como **MVC**, **Observer**, **Singleton** e Inyección de Dependencias dentro de las clases del sistema.

El diagrama de estructura compuesta muestra cómo interactúan internamente las clases dentro del sistema, haciendo visibles los patrones aplicados. Por ejemplo, cliente1 se conecta con carrito1, el cual a su vez se comunica con platillo1. Luego, carrito1 se transforma en pedido1, el cual notifica a cocinero1 y a administrador1 utilizando el patrón **Observer**. A continuación, pedido1 se asocia con ticket1 e informe1, donde este último utiliza el patrón **Singleton**. Además, se visualiza cómo servicios como **InventarioService** y **CalculadoraTotal** son inyectados respectivamente en pedido1 y carrito1, siguiendo el principio de inyección de dependencias.

Respecto a las restricciones arquitectónicas, se establecen reglas obligatorias que deben cumplirse en el diseño. Una de ellas es la separación de responsabilidades bajo el modelo MVC. Según esta regla, las clases como cliente1, carrito1 y pedido1 no deben contener lógica de presentación. Las vistas no deben encargarse de la lógica de negocio ni de manipular directamente los datos, y los controladores, como **PedidoController**, solo deben actuar como coordinadores de flujo, sin acceder directamente a la base de datos.

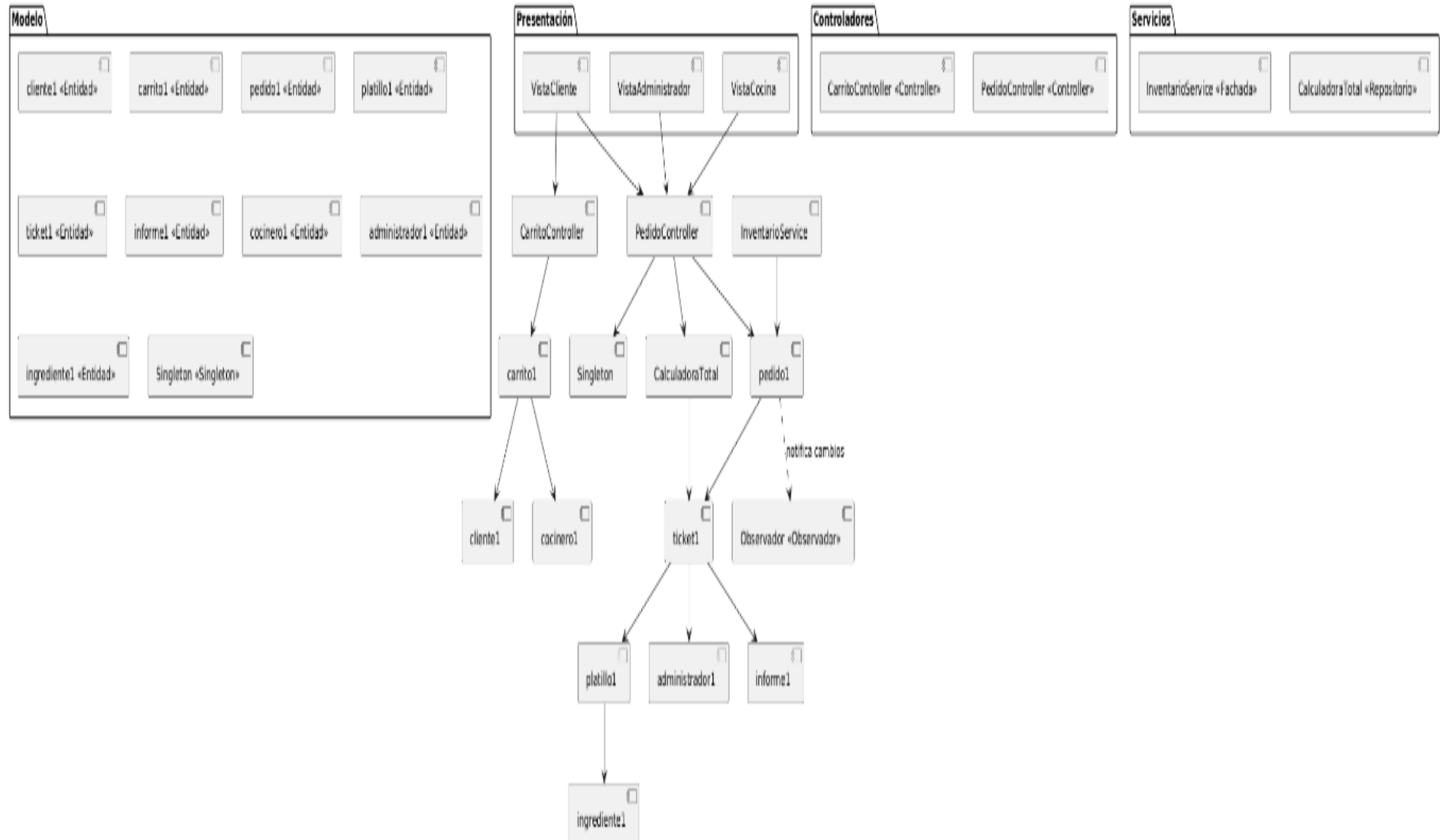
En cuanto al control de instancias, la clase informe1 debe implementarse como un **Singleton**, lo que significa que solo puede existir una instancia activa en el sistema. Por su parte, aunque ticket1 se crea por cada pedido, no debe tener múltiples asociaciones inversas hacia pedido1.

Sobre la notificación de cambios, pedido1 debe permitir que se registren observadores como cocinero1 y administrador1 para que estos puedan recibir actualizaciones de estado. Estos observadores deben implementar una interfaz común, como podría ser **ObservadorEstadoPedido**, y deben registrarse correctamente en el sistema.

En lo que respecta a la inyección de dependencias, clases como carrito1, pedido1 y los servicios auxiliares no deben crear internamente sus propias dependencias. En su lugar, estas deben ser recibidas desde el exterior, ya sea mediante el constructor o a través de un contenedor de inyección. Por ejemplo, **PedidoService** no debe crear una instancia de **InventarioService**, sino que debe recibirla ya construida.

También existen restricciones de colaboración entre entidades. cliente1 solo puede tener un carrito1 activo a la vez. Para que un carrito1 pueda convertirse en un pedido1, debe contener al menos un platillo1. Además, pedido1 no puede generar un ticket1 si antes no ha pasado por los estados de “confirmado” y “preparado”. informe1, por su parte, solo se puede generar a partir de pedidos que ya han sido entregados.

Por último, se definen algunas reglas complementarias de diseño. Los conectores del sistema deben procurar una baja dependencia entre módulos, las clases deben seguir el principio de responsabilidad única, y se deben evitar referencias cíclicas entre entidades, como por ejemplo una cadena tipo carrito1 → pedido1 → carrito1, que no debe existir en la arquitectura.



Resumen de patrones utilizados e implementación

Patrón aplicado	Clases involucradas	Rol dentro del sistema	Tipo de colaboración o relación
MVC (Modelo-Vista-Controlador)	cliente1, carrito1, pedido1, platillo1, PedidoController	Separar presentación, lógica de negocio y estructura de datos	cliente1 usa carrito1; carrito1 genera pedido1
Observer	pedido1, cocinero1, administrador1	Notificar cambios de estado automáticamente	pedido1 notifica a observadores cuando cambia
Inyección de Dependencias	pedido1, carrito1, InventarioService, CalculadoraTotal	Desacoplar clases mediante servicios inyectados	Servicios inyectados a pedido1 y carrito1
Singleton	informe1	Garantizar instancia única del informe	informe1 se accede globalmente desde distintos módulos
Asociación entre entidades	cliente1 → carrito1 → pedido1 → ticket1	Flujo de información y operación secuencial	cliente1 agrega platillos → carrito1 → pedido1

1.7 Interface

Su propósito principal es establecer claramente cómo esa entidad se comunica con el exterior, especificando tanto los servicios que ofrece como los que requiere. Este punto de vista es fundamental cuando múltiples componentes deben interactuar correctamente y de forma controlada, ya sea durante el desarrollo o al integrar subsistemas. Este apartado aborda diversas preocupaciones de diseño. Una de ellas es garantizar que las interfaces estén bien especificadas antes de ser implementadas, lo cual permite detectar errores de integración con anticipación. Además, cumple una función documental: los detalles que se registran aquí pueden ser usados por diseñadores, desarrolladores, testers e incluso como parte de manuales para usuarios técnicos.

En cuanto al contenido, esta sección identifica qué entidades están involucradas en la interacción. Describe cuáles son los servicios proporcionados por el componente, y cuáles necesita de otros. Luego detalla los mecanismos de interacción, que pueden incluir llamadas directas a funciones, intercambio de mensajes, acceso a bases de datos o el uso de eventos. Un aspecto clave del Interface Viewpoint es la descripción precisa de atributos de la interfaz. También se debe definir el orden esperado de las interacciones si hay secuencias específicas que deben seguirse.

1.7.1. Aplicabilidad a distintos tipos de interfaces

El punto de vista de interfaz no se limita únicamente a interfaces de software tradicionales como APIs o servicios web. También se aplica a interfaces de usuario, interfaces de hardware, o cualquier otro medio por el cual una entidad del sistema interactúe con otra. Esto incluye, por ejemplo, la forma en que un módulo se comunica con la base de datos, cómo se expone una funcionalidad a través de una interfaz gráfica, o cómo se envían datos a otro sistema mediante archivos o mensajes. Esta flexibilidad permite que el Interface Viewpoint se use en una amplia variedad de contextos, adaptándose a sistemas distribuidos, aplicaciones móviles, embebidas, o incluso entornos híbridos donde conviven múltiples tecnologías.

1.7.2. Elementos del diseño de interfaz

Se centra en los elementos que deben estar presentes en una descripción adecuada de la interfaz de una entidad de diseño. Esta descripción no se limita a señalar que existe una conexión entre dos partes del sistema, sino que se adentra en cómo se produce esa interacción, qué se espera de cada lado, y en qué condiciones ocurre esa comunicación. Para comenzar, se requiere identificar con claridad cuál es la entidad de diseño a la que pertenece la interfaz. Esto implica especificar el módulo, componente o subsistema dentro del cual se encuentra.

Posteriormente, se debe detallar cómo esta entidad se comunica con otras, lo cual incluye tanto los servicios que ofrece como los servicios que espera recibir de otras partes del sistema. Es decir, no basta con decir que una interfaz "permite el registro de pedidos", sino que se debe definir qué funciones específicas están disponibles, qué entradas esperan, qué salidas generan, y cómo se maneja cualquier posible error durante esa interacción. Un aspecto clave en esta sección es que la forma de interacción debe quedar bien especificada. Esto significa indicar si la comunicación entre componentes se realiza mediante llamadas a procedimientos, intercambio de mensajes, acceso a una base de datos compartida o alguna otra técnica.

1.7.2.1. Atributo de interfaz

El atributo de interfaz incluye tanto los mecanismos de interacción como las reglas que controlan esas interacciones. Entre los mecanismos se consideran formas de invocar, interrumpir o comunicarse con la entidad, ya sea por paso de parámetros, áreas de datos comunes, mensajes, o acceso directo a datos internos.

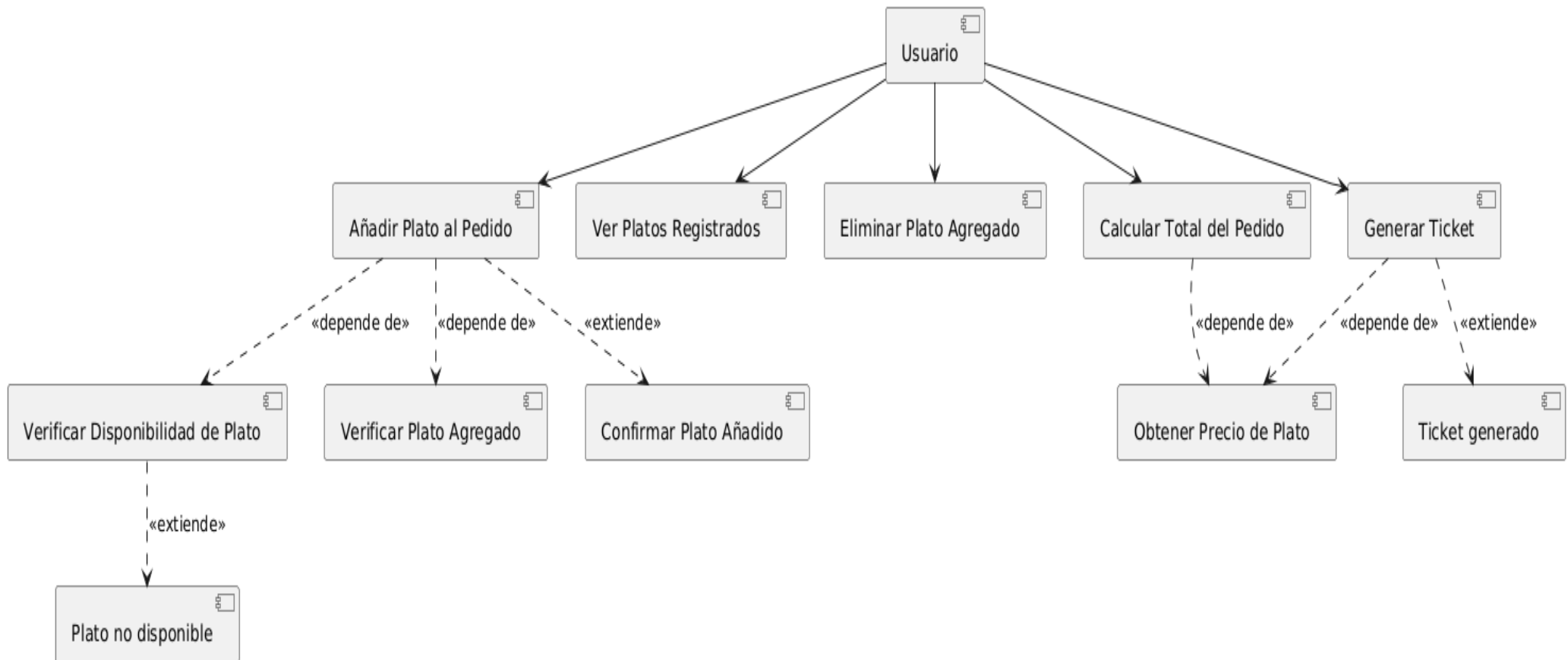
Las reglas de interacción abarcan el protocolo de comunicación, el formato de los datos, los valores permitidos y el significado de esos valores. También se describe el rango de entradas válidas, el formato y tipo de cada entrada o salida, y los posibles códigos de error. En sistemas de información, este atributo debe incluir también los formatos de pantalla, los tipos de entrada, y si aplica, el lenguaje interactivo usado en la interfaz.

Este atributo es especialmente útil en proyectos sin interfaces externas, ya que permite documentar con precisión las interacciones entre componentes internos del sistema, como por ejemplo entre módulos de gestión de pedidos, usuarios y base de datos

1.7.3. Diagramas

La sección 5.8.3 del estándar IEEE 1016 se enfoca en los lenguajes y representaciones utilizadas para describir las interfaces del sistema. En este punto se busca mostrar de forma clara cómo interactúan los usuarios con el software, detallando tanto las entradas permitidas como los resultados esperados a través de la interfaz. Para lograrlo, se hace uso de diagramas y representaciones gráficas que ilustran el funcionamiento visual y estructural de dichas interfaces. En el contexto del sistema de comanda digital, se incluyen diagramas que permiten visualizar la distribución de las pantallas, las opciones disponibles para cada tipo de usuario (como meseros, cocineros y administradores), así como los flujos de interacción que se generan durante el uso del sistema. Estas representaciones facilitan la comprensión de la interfaz y permiten verificar que su diseño cumple con los requisitos funcionales establecidos.

1.7.3.1. DC2 Realización de pedidos



1.7.3.2 Interacción del Usuario con el Realización de pedidos

Usuario-> añadir plato al pedido

El usuario comienza explorando los platos disponibles a través de la opción Ver Platos Registrados, donde el sistema le presenta un menú completo con nombres, descripciones y precios de cada opción. Una vez que el usuario selecciona un plato para agregar a su pedido, el sistema verifica su disponibilidad. Si el plato no está disponible, muestra un mensaje de error claro: "El plato no está disponible". En caso contrario, confirma la acción con un mensaje de éxito: "Plato agregado correctamente".

Usuario -> Ver Platos Registrados (VerPlatos)

La interacción del Usuario con el componente "Ver Platos Registrados" es directa. Cuando el Usuario solicita ver los platos que el sistema tiene registrados, este componente accede a la información almacenada de los platos y la presenta al Usuario. En el diagrama actual, no se muestran dependencias directas con otros componentes para esta funcionalidad, lo que sugiere que "Ver Platos Registrados" se encarga de obtener y mostrar la información por sí mismo. La función principal de este componente es simplemente mostrar la lista de platos disponibles al Usuario para su consulta

Eliminar Plato Agregado (EliminarPlato)

Cuando el Usuario decide eliminar un plato que ya había añadido a su pedido, interactúa con el componente "Eliminar Plato Agregado". El Usuario debe indicar qué plato desea remover de su pedido, y este componente se encarga de realizar la modificación correspondiente en la información del pedido del Usuario. Aunque no se muestran dependencias explícitas en el diagrama, se podría inferir que este componente interactúa con los datos del pedido actual del Usuario para llevar a cabo la eliminación. La acción principal de este componente es actualizar el pedido del Usuario, removiendo el plato seleccionado.

Usuario-> Calcular Total del Pedido

Cuando el usuario solicita *Calcular Total del Pedido*, el sistema procesa los precios individuales de todos los platos agregados y muestra el monto total a pagar. Finalmente, al optar por *Generar Ticket*, el sistema produce un resumen detallado que incluye la lista completa de platos seleccionados, sus precios individuales, el total de la compra y la fecha y hora de emisión, ya sea en formato digital o impreso.

Usuario -> Calcular Total del Pedido (CalcularTotal)

La interacción del Usuario con el componente "Calcular Total del Pedido" ocurre cuando el Usuario desea conocer el costo total de los platos que ha agregado a su pedido hasta ese momento. El Usuario podría accionar un botón o una función específica dentro de la interfaz del sistema para solicitar este cálculo.

Usuario -> Generar ticket

La interacción del Usuario con el componente "Generar Ticket" se da cuando el Usuario ha finalizado su pedido y desea obtener un comprobante. Al igual que con el cálculo del total, el Usuario probablemente activará una opción específica en la interfaz para iniciar la generación del ticket. El diagrama indica que el componente "Generar Ticket" también *depende de* "Obtener Precio de Plato". Esto sugiere que, al igual que para calcular el total, el componente necesita acceder a los precios individuales de los platos en el pedido para poder incluirlos detalladamente en el ticket

Añadir Plato al Pedido-> Verificar Disponibilidad de Plato

Cuando el componente Añadir Plato al Pedido recibe una solicitud interna del sistema (originada por la acción del Usuario de querer agregar un plato), su primer paso crucial es determinar si el plato en cuestión está actualmente disponible. Para lograr esto, el componente Añadir Plato al Pedido establece una comunicación con el componente Verificar Disponibilidad de Plato. Esta comunicación se realiza mediante una llamada o invocación de una función o servicio expuesto por el componente Verificar Disponibilidad de Plato.

Añadir Plato al Pedido-> Verificar Plato Agregado

Una vez que el componente Añadir Plato al Pedido ha determinado que el plato está disponible, o incluso en paralelo a esta verificación, necesita considerar si el plato ya ha sido agregado previamente al pedido del Usuario. Para esto, interactúa con el componente Verificar Plato Agregado. El componente Añadir Plato al Pedido comunica al componente Verificar Plato Agregado la información del plato que se intenta añadir y el identificador del pedido del Usuario.

Interacción entre "Añadir Plato al Pedido" y "Confirmar Plato Añadido"

La interacción entre el componente Añadir Plato al Pedido y Confirmar Plato Añadido ocurre una vez que el plato ha sido exitosamente añadido (o su cantidad actualizada) en el pedido del Usuario. Esta interacción representa una extensión de la funcionalidad principal de "Añadir Plato al Pedido". Cuando el componente Añadir Plato al Pedido completa la lógica para agregar el plato al pedido, activa o utiliza el componente Confirmar Plato Añadido. Esto podría implicar pasarle información sobre el plato que se añadió (nombre, cantidad, precio unitario, etc.) y el pedido al que se añadió.

Verificar Disponibilidad de plato-> plato no disponible

El componente Verificar Disponibilidad de Plato tiene como responsabilidad principal determinar si un plato específico solicitado por el Usuario (a través del componente "Añadir Plato al Pedido") puede ser añadido a su pedido en ese momento. Para llevar a cabo esta verificación, este componente consulta diversas fuentes de información, como el inventario actual de ingredientes, la capacidad de la cocina, o cualquier otra regla de negocio que pueda afectar la disponibilidad de un plato

Calcular total del pedido -> obtener precio de plato

Cuando el componente Calcular Total del Pedido necesita llevar a cabo su función, que es determinar el costo total de los artículos que el Usuario ha añadido a su pedido, requiere acceder a la información de los precios individuales de cada uno de esos artículos. Para obtener estos precios, el componente Calcular Total del Pedido establece una relación de dependencia con el componente Obtener Precio de Plato.

(Generar ticket) -> obtener precio de plato

Cuando el Usuario finaliza su pedido y solicita la generación de un ticket o comprobante (a través de una acción en la interfaz que activa el componente Generar Ticket), este componente necesita acceder a diversos detalles del pedido para poder construir el ticket, y uno de los detalles cruciales es el precio de cada uno de los platos incluidos en el pedido. Para obtener esta información de precios, el componente Generar Ticket establece una relación de dependencia con el componente Obtener Precio de Plato.

Generar ticket -> ticket generado

El componente Generar Ticket tiene como responsabilidad principal tomar la información del pedido final del Usuario, incluyendo los detalles de los platos, sus

cantidades y los precios (obtenidos del componente Obtener Precio de Plato), y formatearla en un documento que sirva como comprobante o factura, es decir, el ticket. Una vez que el componente Generar Ticket ha completado el proceso de recopilación de la información necesaria, ha realizado los cálculos pertinentes (como subtotales, impuestos y el total final) y ha formateado esta información en la estructura del ticket, se produce la interacción con el componente Ticket generado.