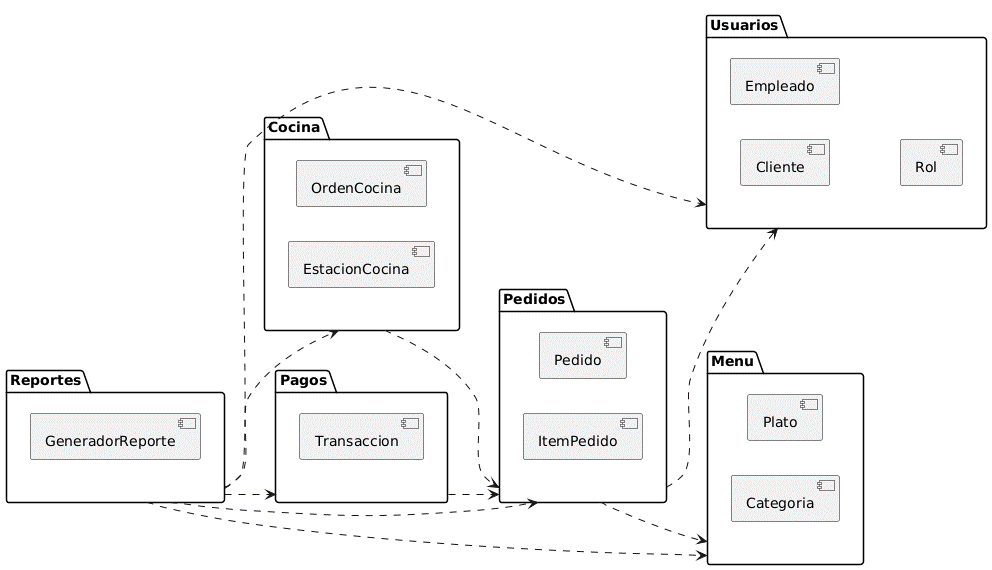
**Diagrama de Paquetes: Organizando las Funcionalidades**

El Diagrama de Paquetes te ayuda a organizar tu sistema en **grupos lógicos de funcionalidades**. Piensa en ellos como las carpetas principales de tu proyecto de software. Esto es crucial para la **modularidad**, haciendo que el sistema sea más fácil de entender y manejar.

**¿Cómo hacerlo para el sistema de pedidos?**

Imagina las grandes áreas de tu sistema:

1. **Gestión de Usuarios:** Todo lo relacionado con clientes, personal, roles.
2. **Gestión de Menú:** Platos, categorías, precios.
3. **Procesamiento de Pedidos:** Crear, modificar, rastrear pedidos.
4. **Gestión de Cocina:** Preparación de platos.
5. **Pagos:** Procesamiento de transacciones.
6. **Reportes:** Generación de informes.



**Cómo interpretarlo:**

* **Paquetes:** Los rectángulos con una pestaña (como Usuarios, Pedidos, Menu) representan los módulos lógicos.
* **Dependencias:** Las flechas punteadas indican que un paquete "usa" o "depende de" otro. Por ejemplo, el paquete Pedidos depende de Usuarios (un pedido lo hace un Cliente) y de Menu (un pedido contiene Platos). El paquete Pagos probablemente depende de Pedidos para saber qué procesar. Cocina depende de Pedidos para recibir las órdenes, y Reportes dependería de casi todos los demás para consolidar la información.

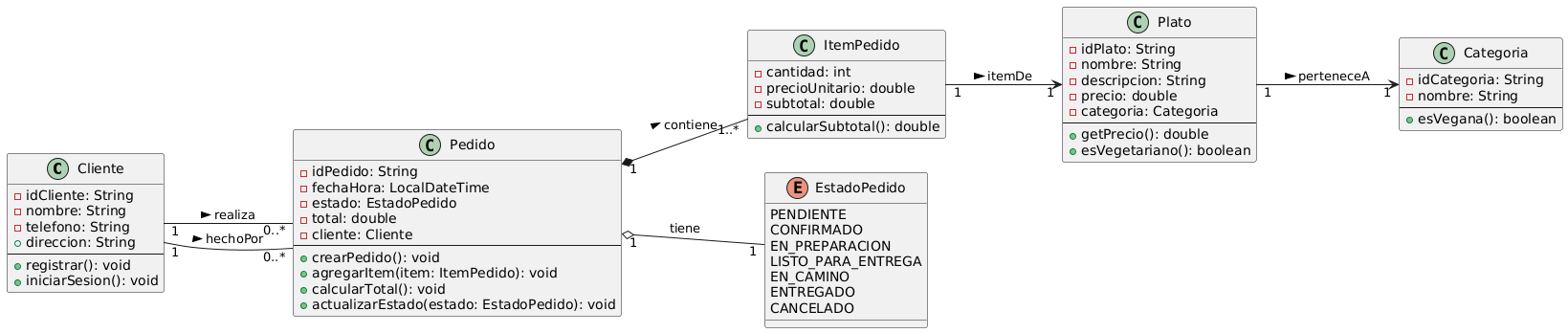
**Diagramas de Clases de la Solución: El Blueprint del Código**

El Diagrama de Clases de la Solución es el **diseño detallado de las clases que vas a codificar**. Aquí definimos los atributos (datos) y métodos (comportamientos) de cada clase, y las relaciones precisas entre ellas, incluyendo visibilidad y multiplicidad. Este es tu mapa directo para la implementación.

**¿Cómo hacerlo para el sistema de pedidos?**

Nos centraremos en el paquete Pedidos y Menu para este ejemplo.

**Ejemplo de Diagrama de Clases (parcial):**



**Cómo interpretarlo:**

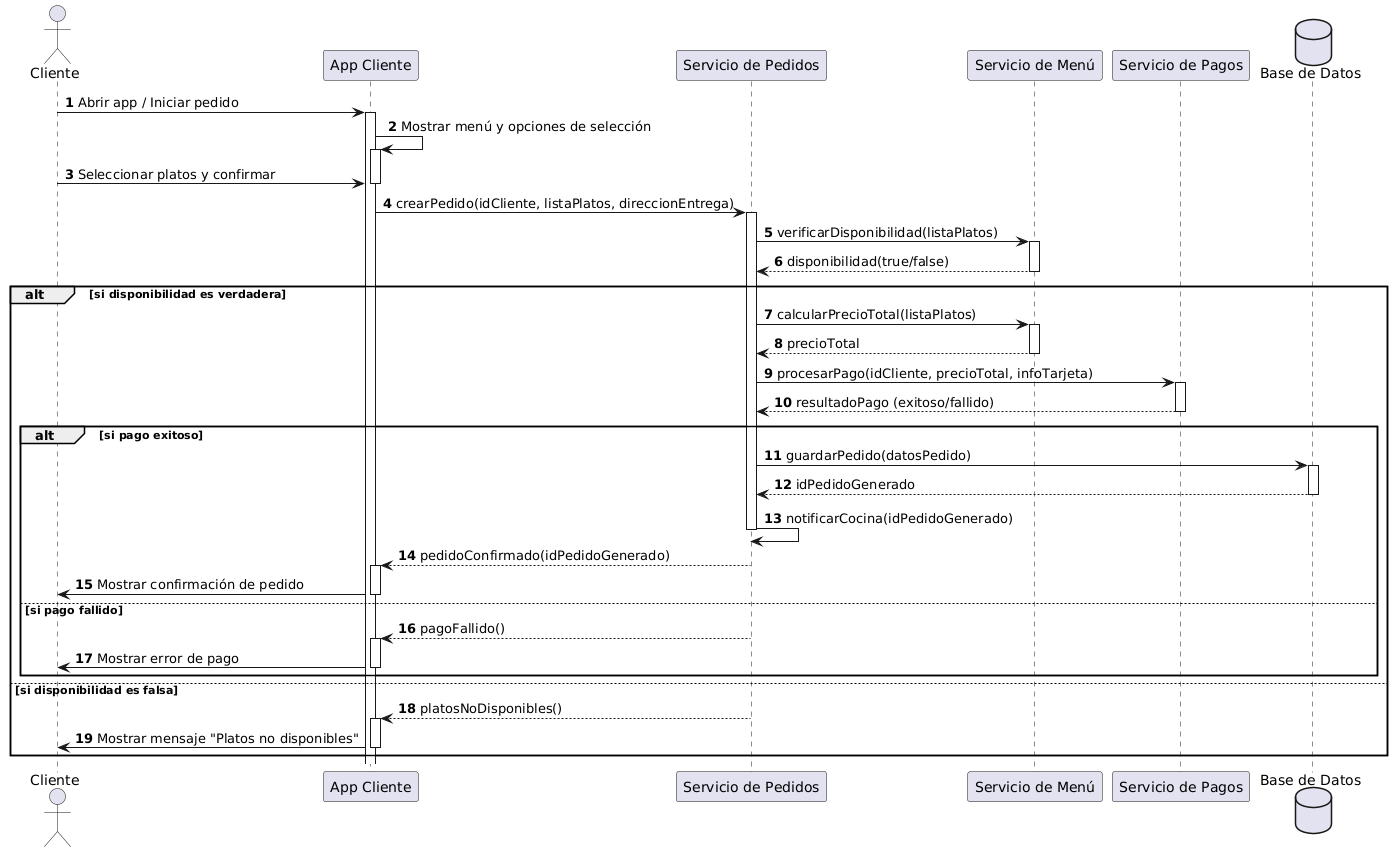
* **Clases (Cliente, Plato, Pedido, etc.):** Cada caja es una clase.
* **Atributos (- idCliente: String):** Las variables que tiene la clase. El + o - indica si es público o privado.
* **Métodos (+ registrar(): void):** Las funciones que la clase puede ejecutar.
* **Relaciones:**
  + **Asociación:** Líneas simples. Un Cliente **realiza** un Pedido. La multiplicidad (1 a \*) significa que "un cliente puede tener **muchos** pedidos".
  + **Composición (Pedido <> ItemPedido):** El rombo relleno en Pedido indica que Pedido **compone** ItemPedido. Un ItemPedido no existe sin un Pedido. Si borras el pedido, se borran sus ítems.
  + **Herencia:** Si tuvieras Usuario genérico y Cliente e Empleado fueran subclases, verías flechas con triángulos vacíos apuntando de Cliente y Empleado hacia Usuario.

**Diagramas de Secuencia: El Flujo Paso a Paso**

Los Diagramas de Secuencia son como un **guion de película detallado** para una interacción específica (por ejemplo, "Realizar un Nuevo Pedido"). Muestran el **orden exacto de los mensajes** entre los objetos a lo largo del tiempo.

**¿Cómo hacerlo para el sistema de pedidos (proceso "Realizar Pedido")?**

**Ejemplo de Diagrama de Secuencia (simplificado):**



**Cómo interpretarlo:**

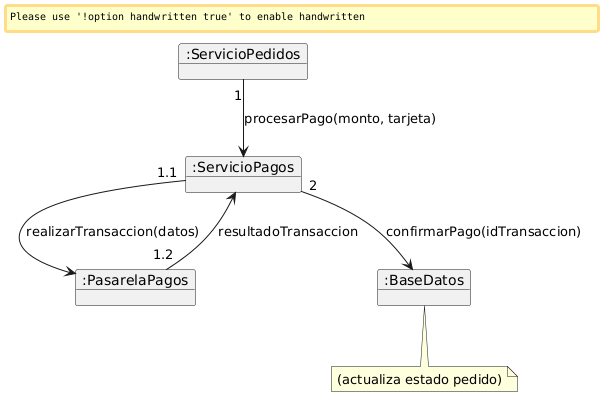
* **Rectángulos Arriba (:Cliente, :AppCliente, etc.):** Son los actores y objetos que interactúan.
* **Líneas punteadas verticales:** Las "líneas de vida" de cada participante a lo largo del tiempo.
* **Flechas horizontales:** Los mensajes. Se leen de arriba a abajo.
  + solicitarPedido: Un mensaje del Cliente a la AppCliente.
  + verificarDisponibilidad: Una llamada de método del ServicioPedidos al ServicioMenu.
  + --> (flecha de retorno): disponibilidad (true/false) es la respuesta del ServicioMenu al ServicioPedidos.
* **Barras de Activación:** Los rectángulos verticales en las líneas de vida indican cuando un objeto está activo procesando un mensaje.
* **if/else (fragmentos alt):** Muestran decisiones en el flujo.

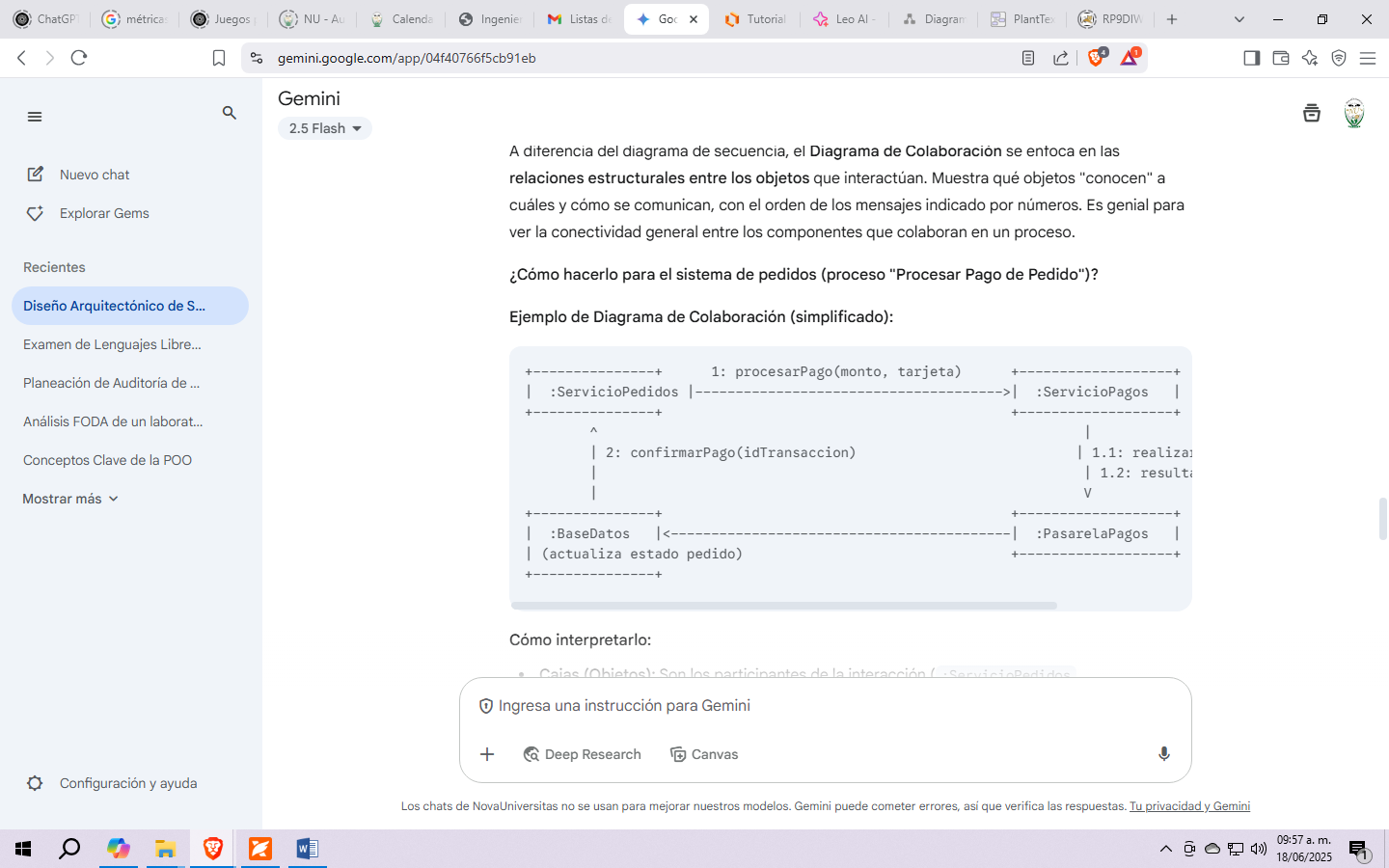
**Diagramas de Colaboración (o de Comunicación): Quién Habla Con Quién**

A diferencia del diagrama de secuencia, el **Diagrama de Colaboración** se enfoca en las **relaciones estructurales entre los objetos** que interactúan. Muestra qué objetos "conocen" a cuáles y cómo se comunican, con el orden de los mensajes indicado por números. Es genial para ver la conectividad general entre los componentes que colaboran en un proceso.

**¿Cómo hacerlo para el sistema de pedidos (proceso "Procesar Pago de Pedido")?**

**Ejemplo de Diagrama de Colaboración (simplificado):**





**Cómo interpretarlo:**

* **Cajas (Objetos):** Son los participantes de la interacción (:ServicioPedidos, :ServicioPagos, :PasarelaPagos, :BaseDatos).
* **Líneas (Enlaces):** Indican que los objetos están conectados y pueden comunicarse.
* **Mensajes numerados:** Las flechas con números indican el orden de los mensajes.
  + 1: procesarPago(): El ServicioPedidos inicia el proceso con ServicioPagos.
  + 1.1: realizarTransaccion(): El ServicioPagos llama a la PasarelaPagos. El 1.1 indica que es una sub-llamada dentro del mensaje 1.
  + 1.2: resultadoTransaccion: La PasarelaPagos responde a ServicioPagos.
  + 2: confirmarPago(): El ServicioPagos actualiza la BaseDatos.
* **Énfasis en conexiones:** Puedes ver rápidamente que ServicioPedidos se conecta con ServicioPagos, que a su vez se conecta con PasarelaPagos y BaseDatos.

**Diagramas de Actividades: El Flujo del Proceso de Negocio**

Los Diagramas de Actividades son excelentes para **modelar los flujos de trabajo o procesos de negocio** completos, mostrando las actividades, decisiones, y flujos paralelos. Son como un flujograma avanzado para el software o incluso para procesos manuales que el software apoya.

**¿Cómo hacerlo para el sistema de pedidos (proceso "Preparar y Entregar Pedido")?**

**Ejemplo de Diagrama de Actividades (con carriles):**

**Cómo interpretarlo:**

* **Carriles (Swimlanes):** Las columnas (Cliente, Cocina, Repartidor, Sistema) muestran quién es responsable de cada actividad.
* **Círculo de inicio:** O marca el comienzo del proceso.
* **Actividades (Preparar Platos, Empaquetar Pedido):** Son los pasos del proceso.
* **Bifurcación (Fork, la barra gruesa):** Después de "Empaquetar Pedido", el flujo se divide. La Cocina puede "Notificar Pedido Listo" (quizás al Sistema) **al mismo tiempo** que el Sistema "Notifica al Repartidor". Estos flujos ocurren en paralelo.
* **Unión (Join, la barra gruesa):** Después de las notificaciones, los flujos se unen antes de que el repartidor recoja.
* **Rombo (Decisión):** Si hubiera una decisión como "El cliente está en casa?", el rombo tendría dos salidas (Sí/No).
* **Círculo de fin:** O con borde marca el final del proceso.