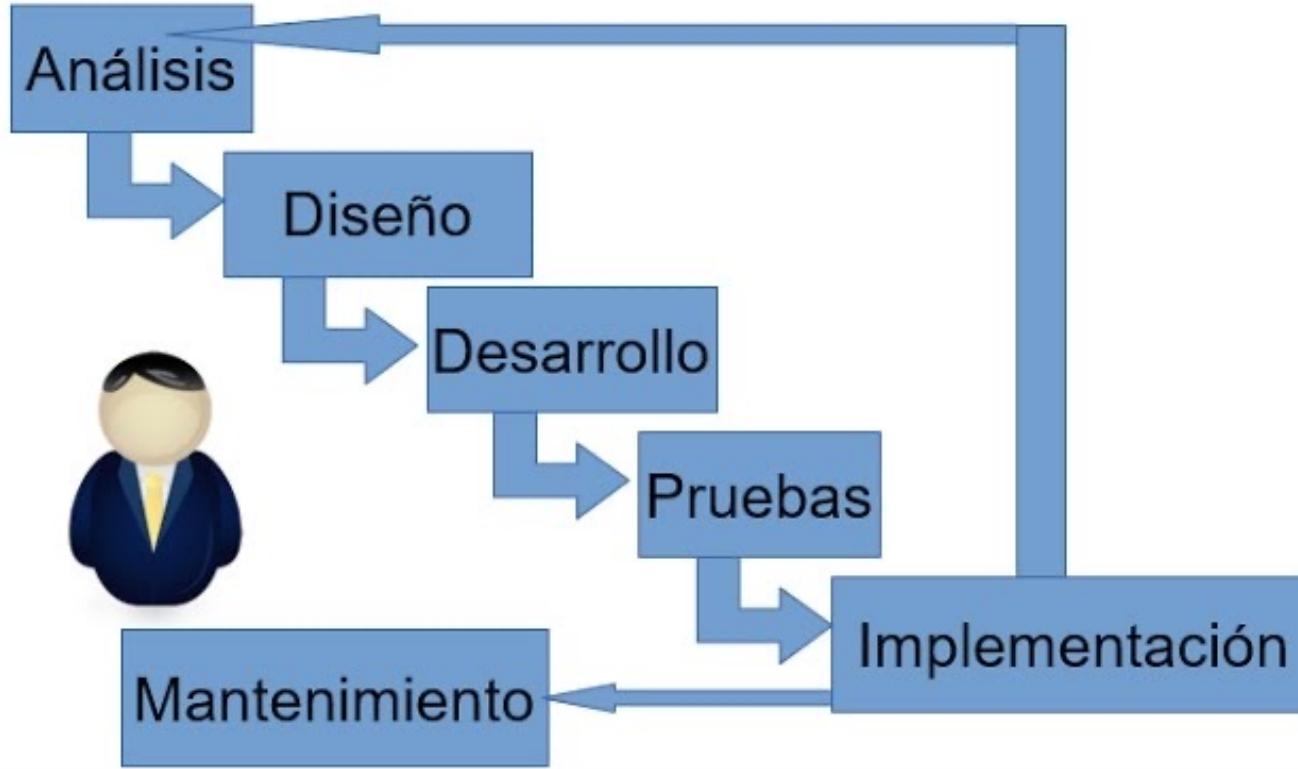


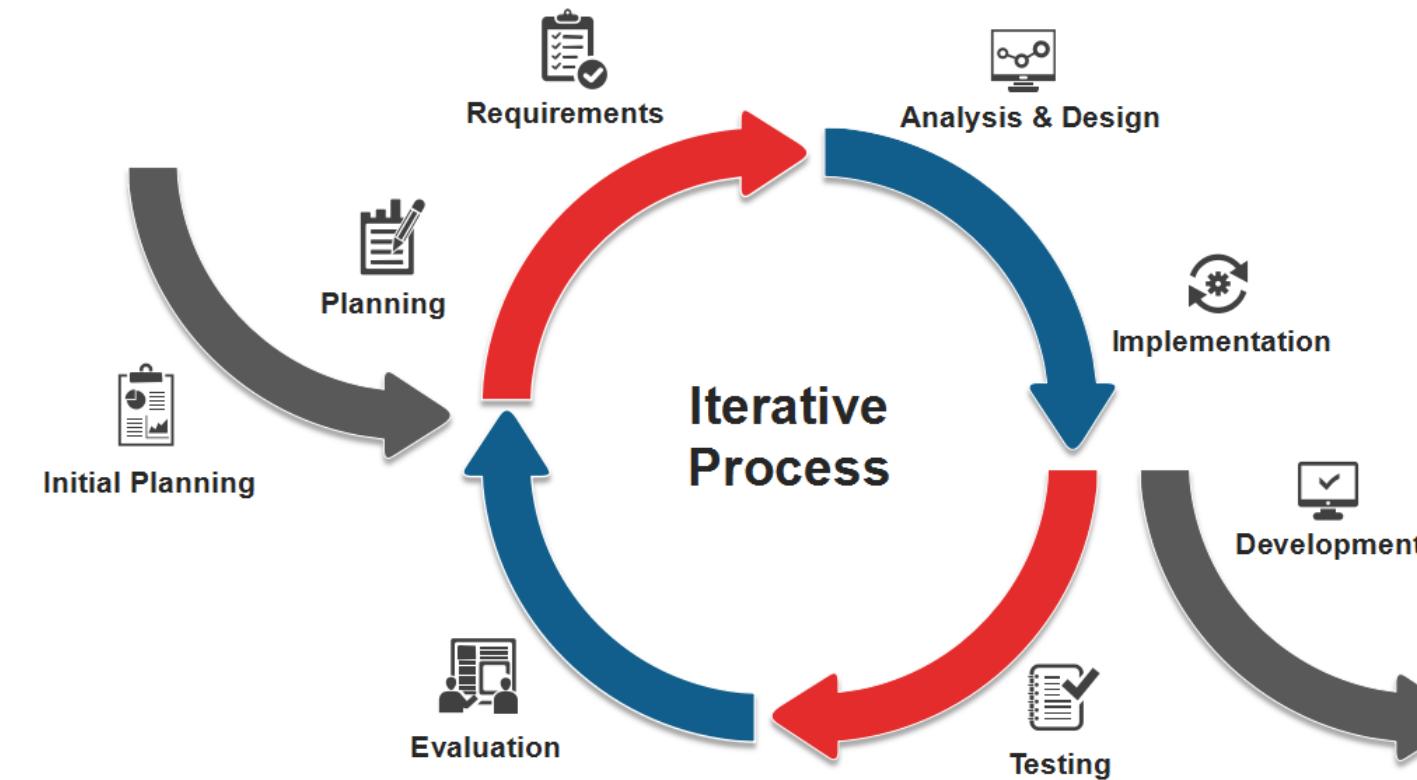
Código del curso: **SOFT-09**
Introducción a la Ingeniería
del Software.



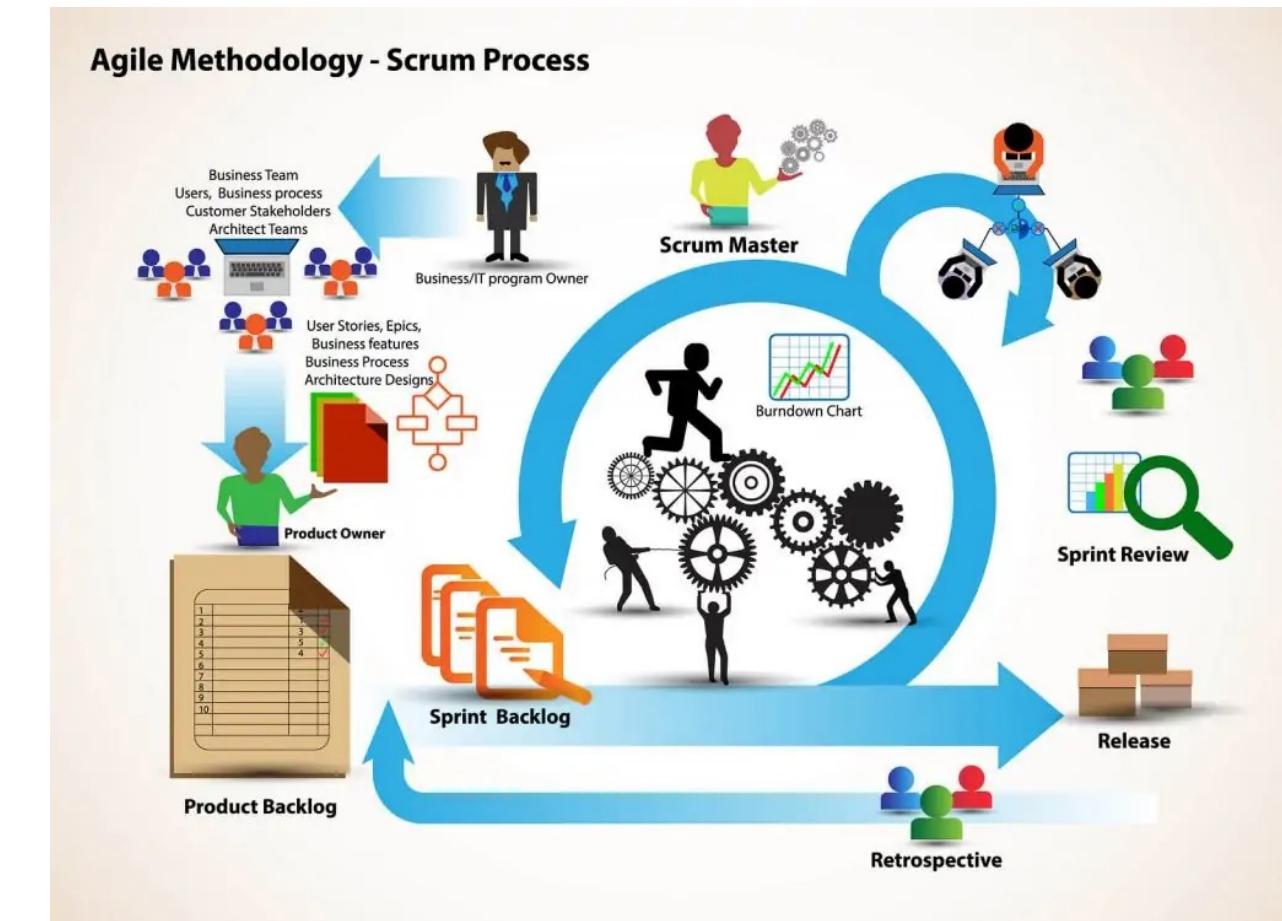
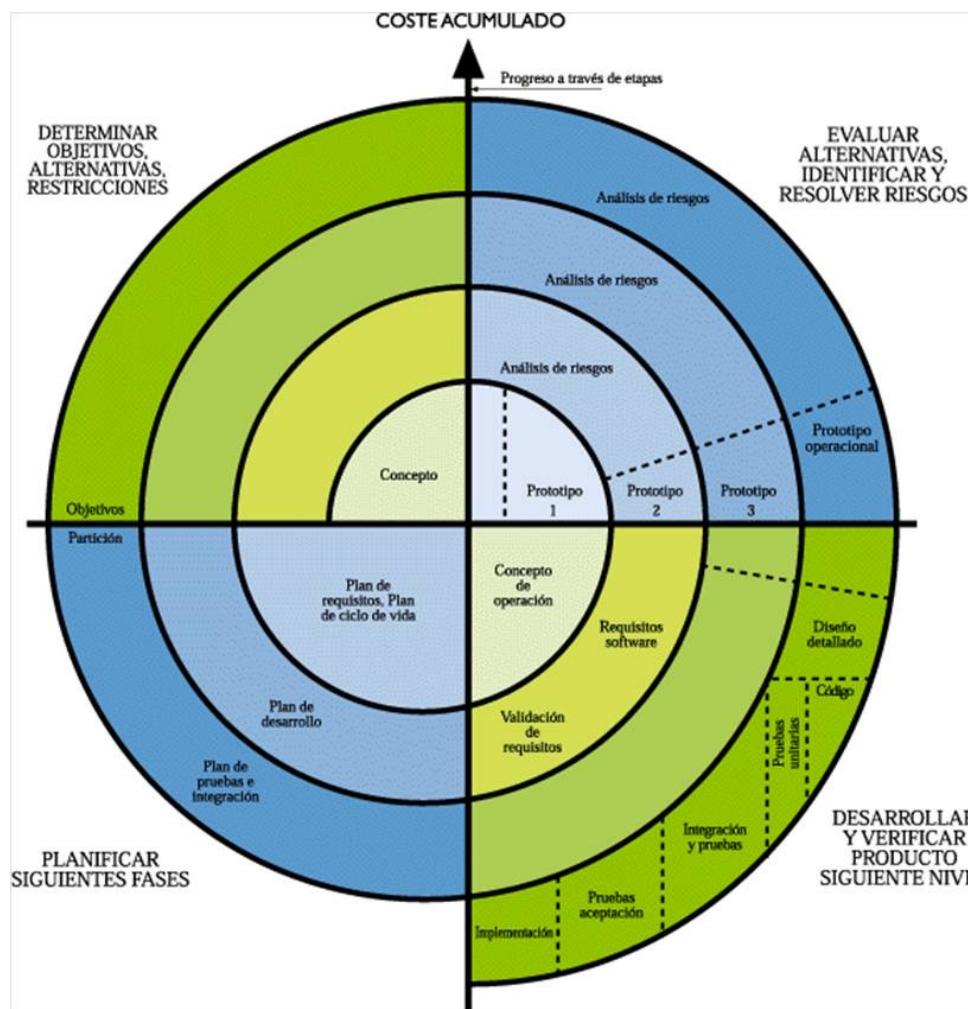
Proceso en cascada o clásico



Modelo de proceso iterativo



This slide is 100% editable. Adapt it to your needs and capture your audience's attention.



AGENDA

TEMA 3: Etapas del Desarrollo del software

Ingeniería de Requerimientos

Diseño del Software

Construcción del Software

Depuración y pruebas de software

Despliegue e implementación del Software

Mantenimiento del Software

ETAPAS DEL DESARROLLO DEL SOFTWARE



1. Planificación:

• **Objetivo:** Crea una aplicación para que los clientes de la cafetería "Cenfo Café" puedan ver el menú, hacer pedidos y pagar desde su celular.

• **Acciones:** Se definen los objetivos claros, se establece un presupuesto, se estima el tiempo de desarrollo y se asignan los recursos necesarios.

2. Análisis de Requisitos:

• **Objetivo:** Entender a fondo qué necesita el cliente (la cafetería y sus usuarios).

• **Acciones:** Se hacen entrevistas con los dueños y empleados para saber qué funciones son importantes (ver menú, pedir, pagar, recibir notificaciones) y cómo debe interactuar el usuario.

3. Diseño:

• **Objetivo:** Definir la arquitectura y la interfaz de la aplicación. Toda la documentación de diagramas también va acá.

• Acciones:

• **Diseño de interfaz o prototipo:** Se crea cómo se verá la aplicación (los botones, los colores, la disposición de los elementos).

• **Diseño de la base de datos:** Se estructura cómo se guardarán los datos de los menús, pedidos y usuarios.

4. Desarrollo (Codificación):

- **Objetivo:** Escribir el código fuente de la aplicación.

- **Acciones:** Los programadores escriben las instrucciones para que la aplicación funcione, usando lenguajes de programación y herramientas específicas.

5. Pruebas:

- Objetivo: Asegúrese de que la aplicación funcione correctamente y no tenga errores.

- **Acciones:** Se realizan diferentes tipos de pruebas:

- **Pruebas unitarias:** Se verifica cada pequeña parte del código por separado.

- **Pruebas de integración:** Se comprueba que las diferentes partes del código funcionan bien juntas.

- **Pruebas de usuario:** Se da la aplicación a los dueños y empleados para que la usen y encuentren problemas.

6. Implementación (Despliegue):

- Objetivo: Poner la aplicación a disposición de los usuarios finales.
- **Acciones:** La aplicación se publica en las tiendas de aplicaciones (como Google Play Store o Apple App Store) para que los clientes de "CenfoCafé" la puedan descargar e instalar.

7. Mantenimiento:

- Objetivo: Mantenga la aplicación funcionando y actualizada.
- **Acciones:**
 - **Corrección de errores:** Se solucionan los problemas que surjan después de su lanzamiento.
 - **Actualizaciones:** Se agregan nuevas funciones, se mejora el rendimiento o se adapta la aplicación a nuevas versiones de sistemas operativos.

Definición de la ingeniería de requerimientos.

Es el proceso de recopilar requerimientos, descubrir, analizar, documentar y verificar las necesidades del usuario, es la pieza fundamental en un proyecto de desarrollo de software.



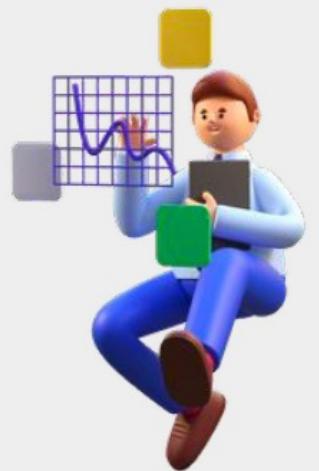
PARTES INTERESADAS (stakeholders)



Personas Usuarias



Equipo de Soporte



Equipo de ventas y
mercadeo



Desarrolladores



Patrocinador

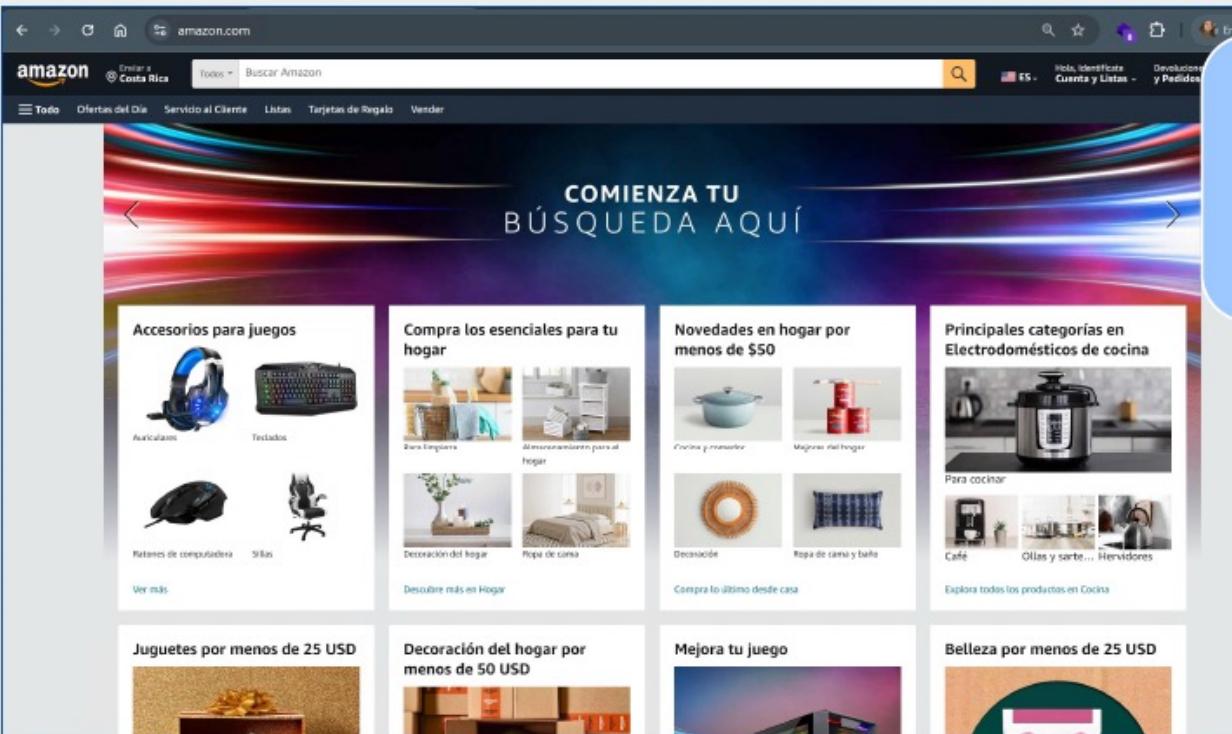
¿Por qué es importante realizar la Ingeniería de Requerimientos?



¿Cómo le
parece que
quedó el Sw?
¿Bonito
Verdad?

Se ve bien, pero yo
creo que ese no es
el de la empresa,
¿no lo han traído
equivocado?

Tipos de Requerimientos



Los requerimientos funcionales

son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.



Los requerimientos NO funcionales

tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, auditabilidad y otros.

Características de los requerimientos

Un requerimiento
debe ser

Especificado por escrito

Posible de probar o verificar

Conciso

Completo

Consistente

No ambiguo

REQUERIMIENTOS FUNCIONALES

• Registro de usuario:

- ✓ El sistema debe permitir a los usuarios registrarse proporcionando su nombre, correo electrónico y contraseña.
- ✓ El sistema debe asignar un identificador único a cada nuevo usuario registrado.

• Inicio de sesión:

- ✓ El sistema debe permitir a los usuarios iniciar sesión utilizando su nombre de usuario y contraseña.
- ✓ El sistema debe validar las credenciales del usuario durante el inicio de sesión.

• Gestión de productos:

- ✓ El sistema debe permitir a los administradores agregar nuevos productos al catálogo, especificando nombre, precio y descripción.
- ✓ El sistema debe descontar el producto del inventario cuando se realiza una venta.

• Proceso de compra:

- ✓ El sistema debe permitir a los usuarios agregar productos a un carrito de compras.
- ✓ El sistema debe enviar un correo electrónico de confirmación al usuario después de que se complete una compra.

• Búsqueda de productos:

- ✓ El sistema debe permitir a los usuarios buscar productos por nombre o categoría.
- ✓ El sistema debe mostrar una lista de productos que coincidan con los criterios de búsqueda del usuario.



REQUERIMIENTOS NO FUNCIONALES

•Rendimiento:

- ✓ El sistema debe procesar solicitudes de usuarios en un promedio de 2 segundos, incluso con mucho tráfico.
- ✓ Las consultas de búsqueda deben completarse en menos de 1 segundo.

•Seguridad:

- ✓ El sistema debe utilizar encriptación de 256 bits para el almacenamiento de datos.
- ✓ El sistema debe implementar un sistema robusto para la recuperación de contraseñas por correo electrónico.

•Usabilidad:

- ✓ La interfaz de usuario debe ser intuitiva y fácil de aprender para usuarios y administradores.
- ✓ El sistema debe contar con un módulo de ayuda en línea y manuales de usuario bien estructurados.

•Disponibilidad:

- ✓ El sistema debe mantener un tiempo de actividad del 99,9% para garantizar el acceso constante del usuario.

•Escalabilidad:

- ✓ El sistema debe ser capaz de manejar un crecimiento en el número de usuarios y registros sin una degradación del rendimiento.

•Mantenibilidad:

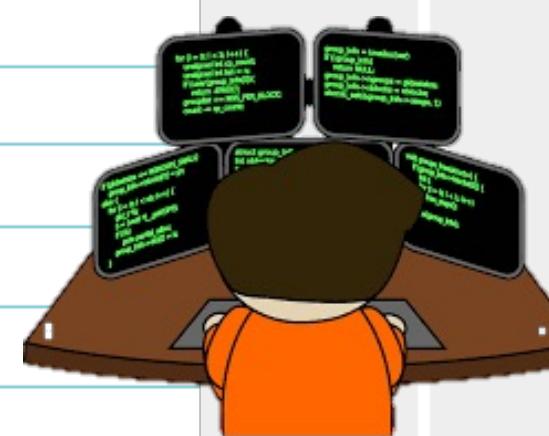
- ✓ El sistema debe diseñarse de forma modular para facilitar su mantenimiento y actualización.

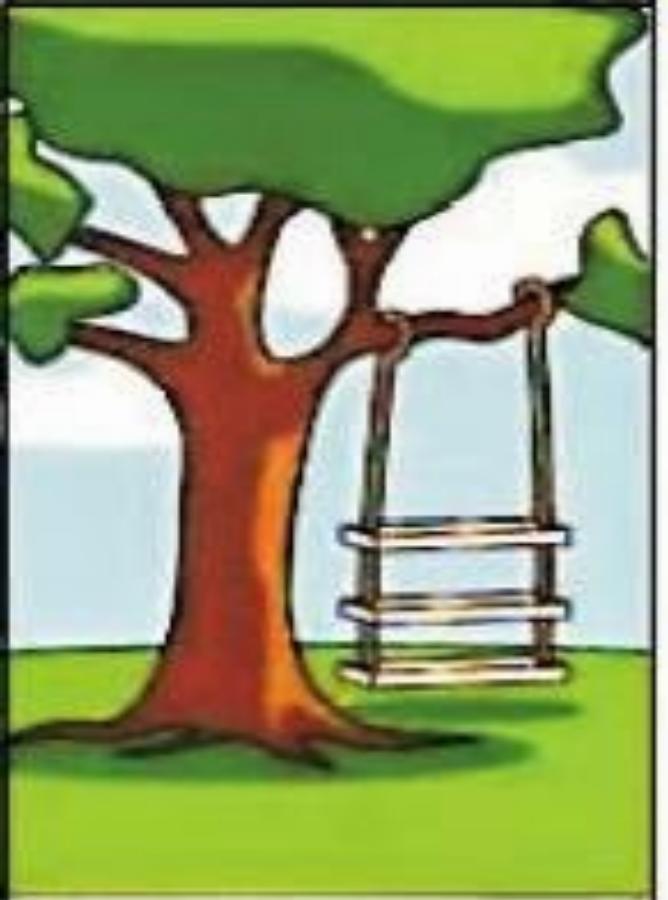
•Portabilidad:

- ✓ El software debe poder funcionar en diferentes sistemas operativos o plataformas de hardware.

•Confiabilidad:

- ✓ En caso de fallo, el sistema debe ser capaz de recuperar y restaurar los datos desde la última copia de seguridad.





La solicitud del usuario



Lo que entendió el líder del proyecto



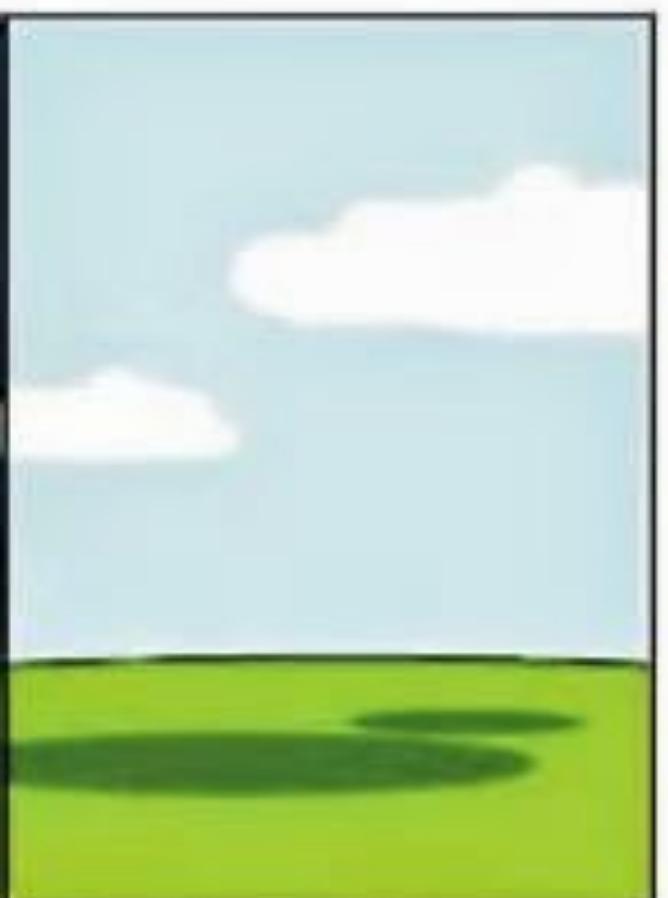
El diseño del analista de sistemas



El enfoque del programador



La recomendación del consultor externo



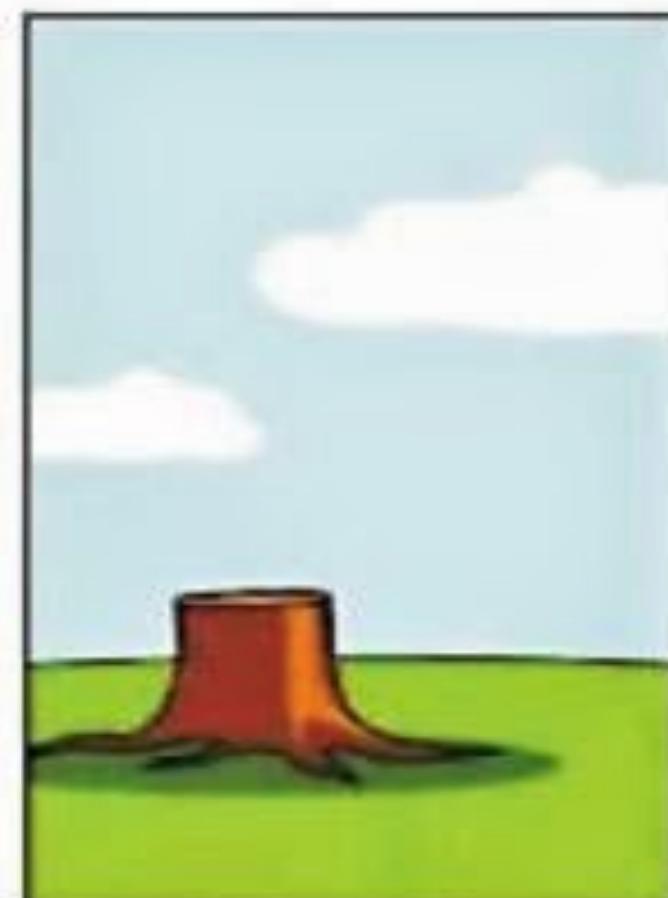
La documentación del proyecto



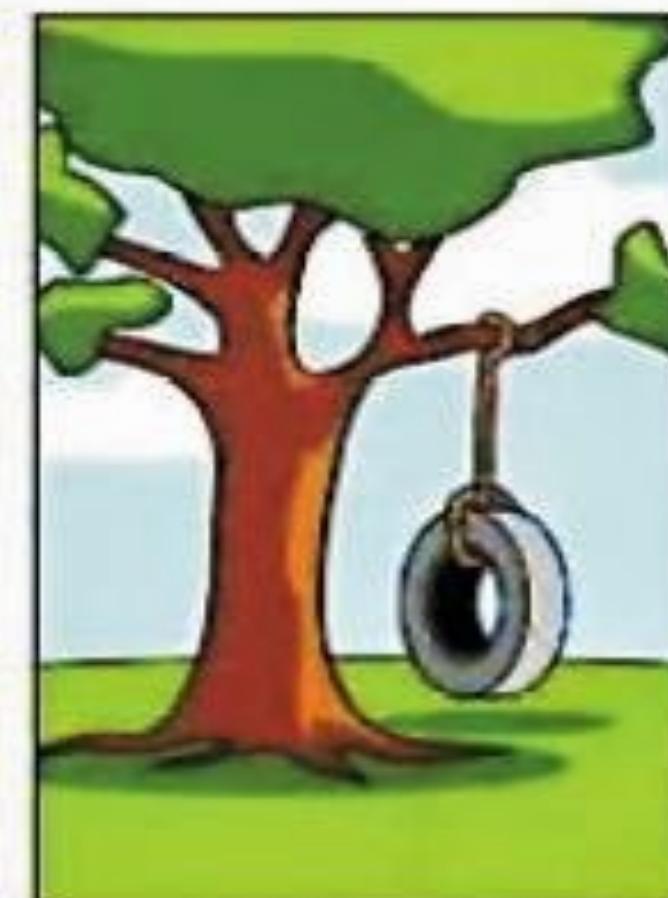
La implantación en producción



El presupuesto del proyecto



El soporte operativo



Lo que el usuario realmente necesitaba

DISEÑO DEL SOFTWARE

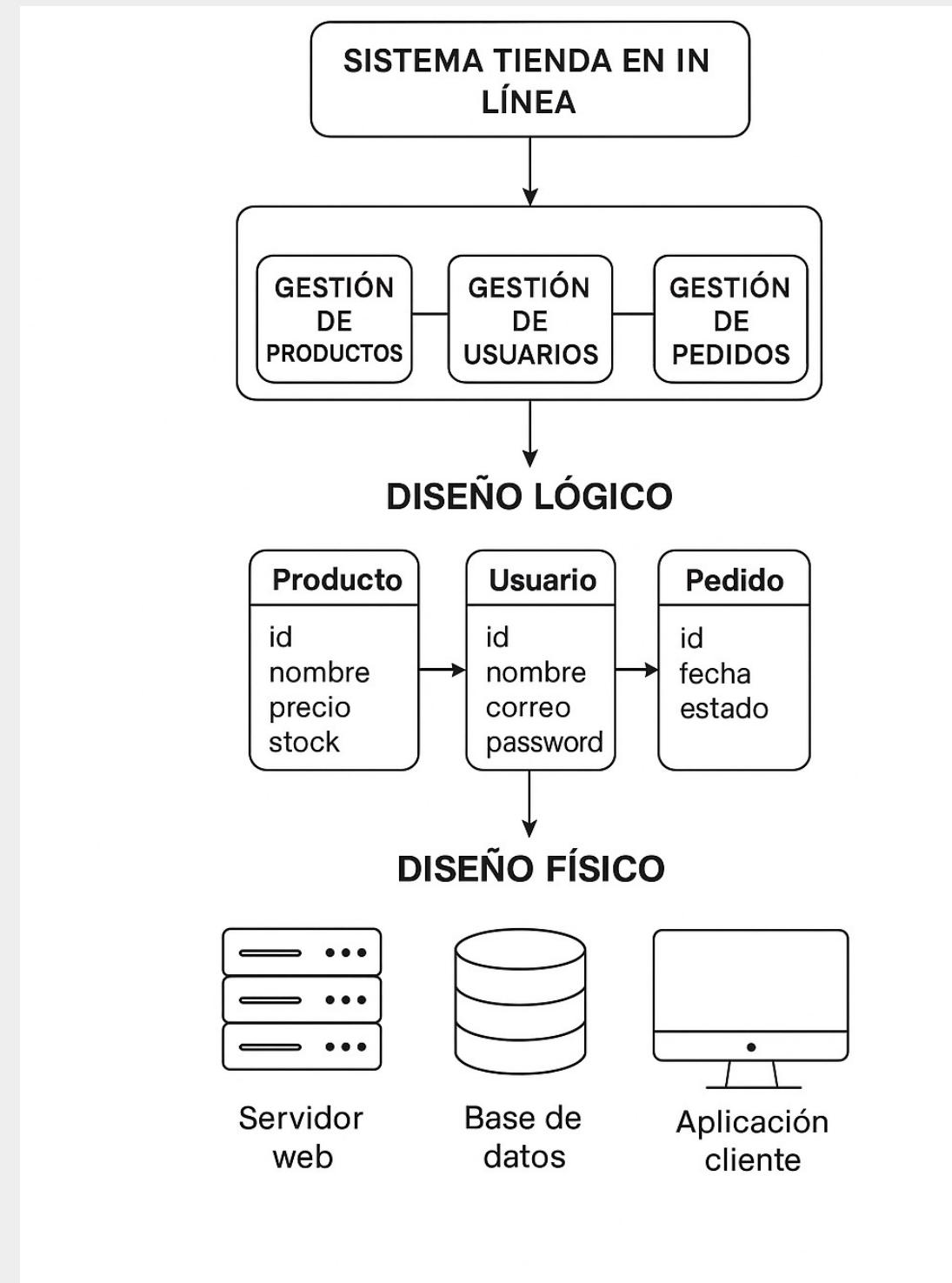
El diseño del software ayuda a modificar elementos que se utilizan para armar el formato que tendrá el programa, la importancia del diseño del software se puede definir en una sola palabra: **calidad**; ya que dentro del diseño es donde se establece la calidad del proyecto.



El diseño de software es una actividad creativa

El diseño es la única manera de materializar con precisión los requerimientos del cliente.

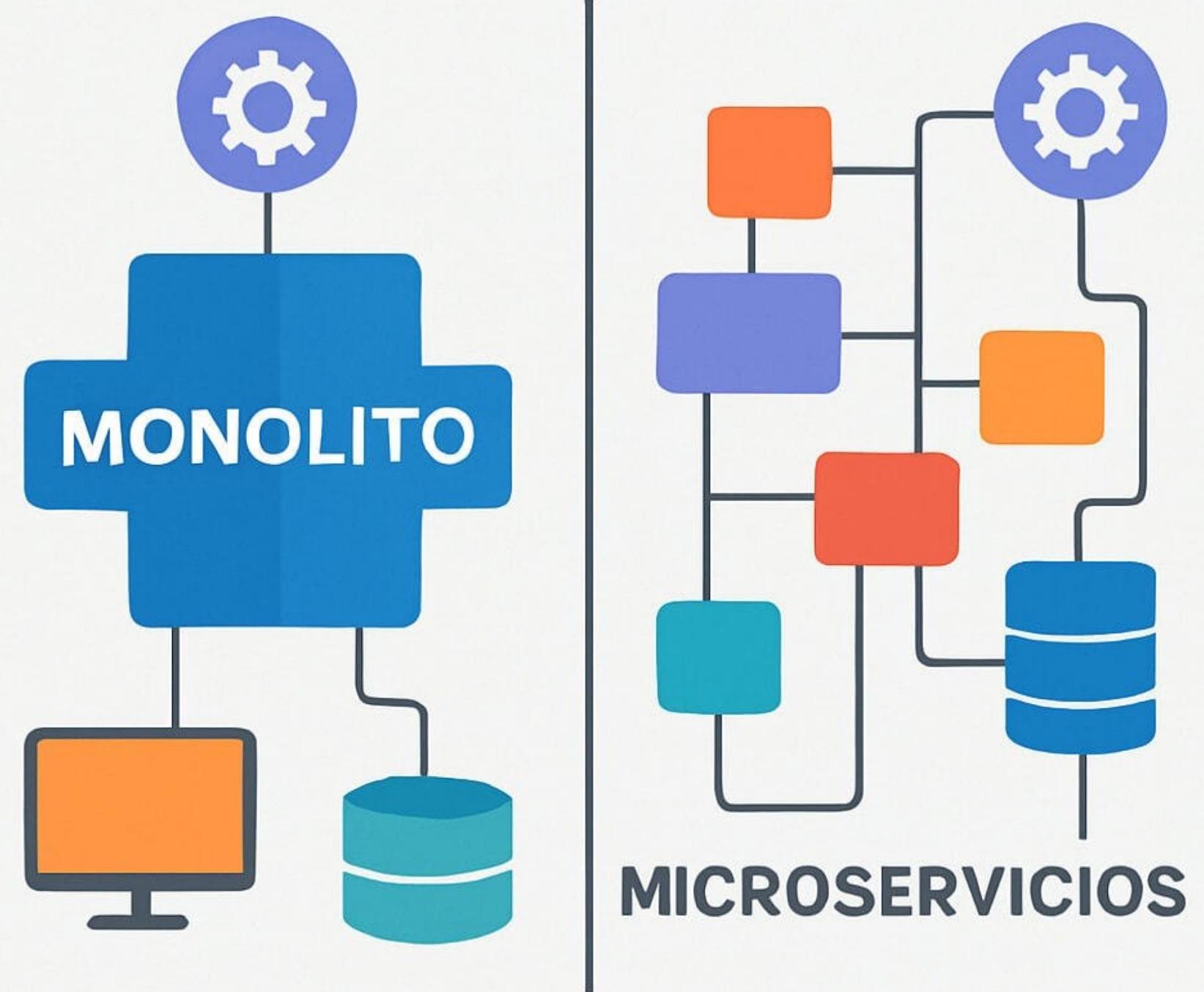
DISEÑO DEL SOFTWARE



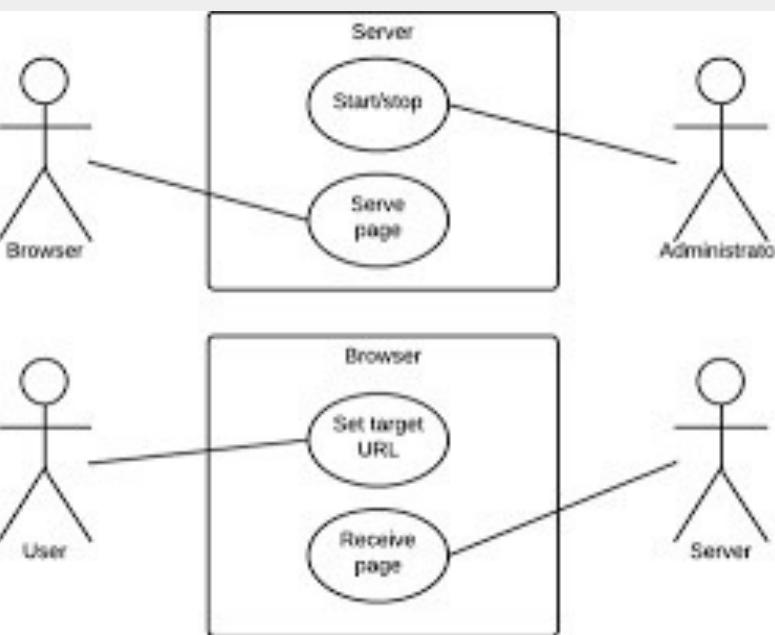
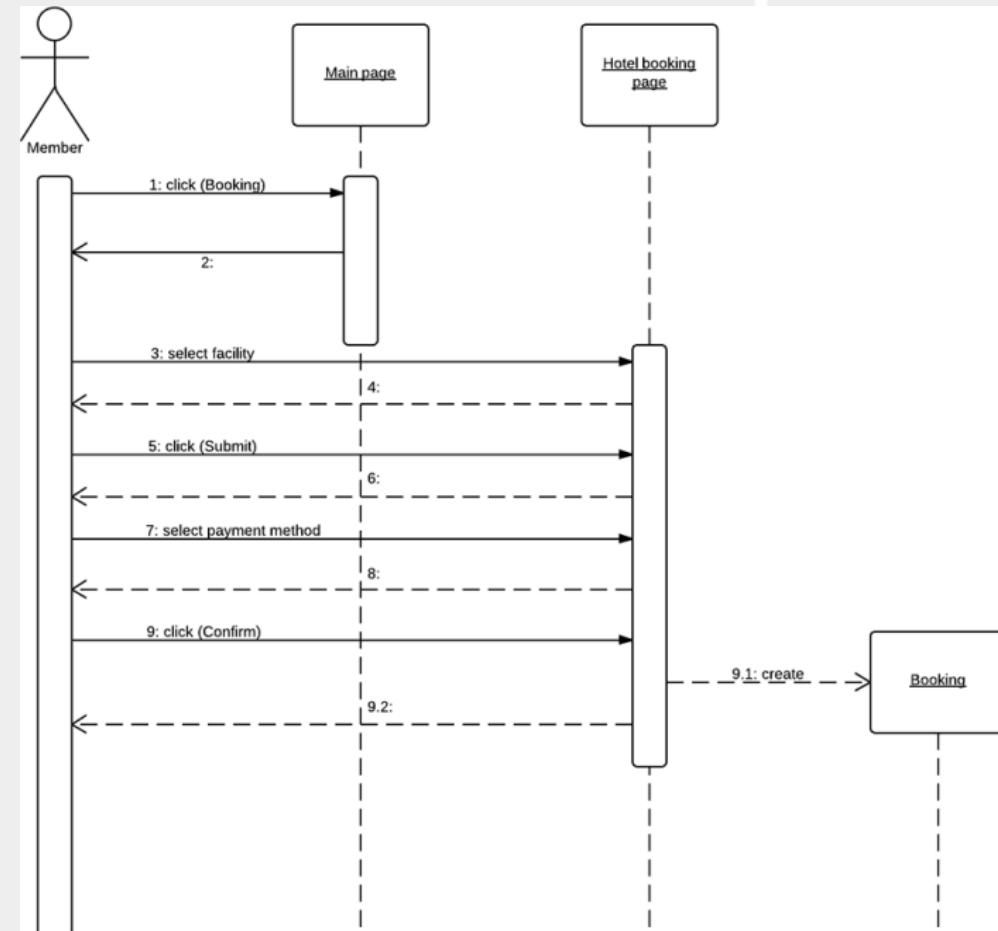
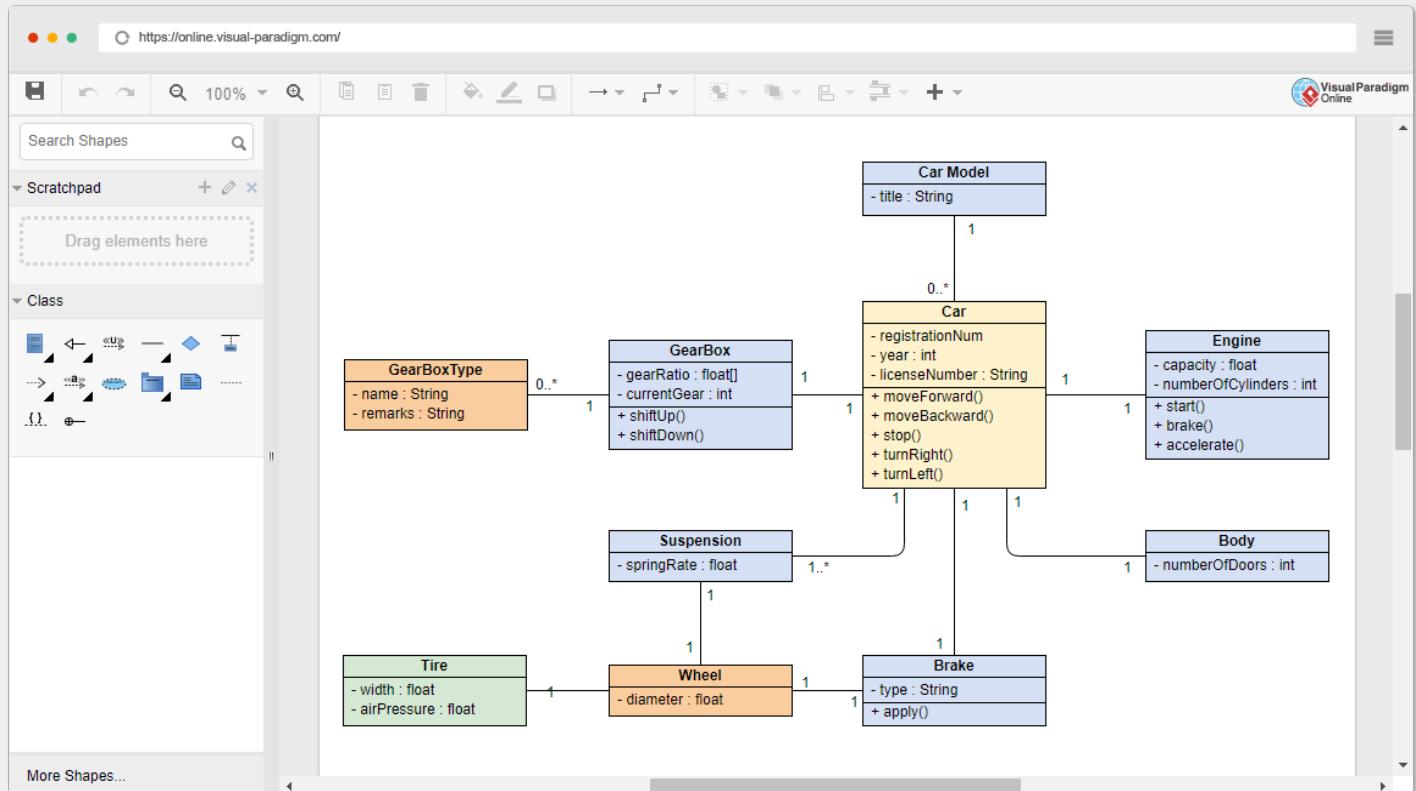
Los 4 elementos del diseño son:

1. **El diseño arquitectónico (Monolito vs micro servicios o capas)**
2. **El diseño de los datos.**
3. **El diseño de la interfaz (wireframes, flujos de UX)**
4. **El diseño a nivel de componente o de procedimientos (clases, módulos).**

¿QUÉ ARQUITECTURA CONVIENE A TU EMPRESA?



EL DISEÑO DEL SOFTWARES ES TAMBIÉN LA DOCUMENTACIÓN



DIAGRAMAS DE CASO DE USO, DIAGRAMAS DE SECUENCIA, DIAGRAMAS DE CLASES, DIAGRAMAS DE ESTADOS.

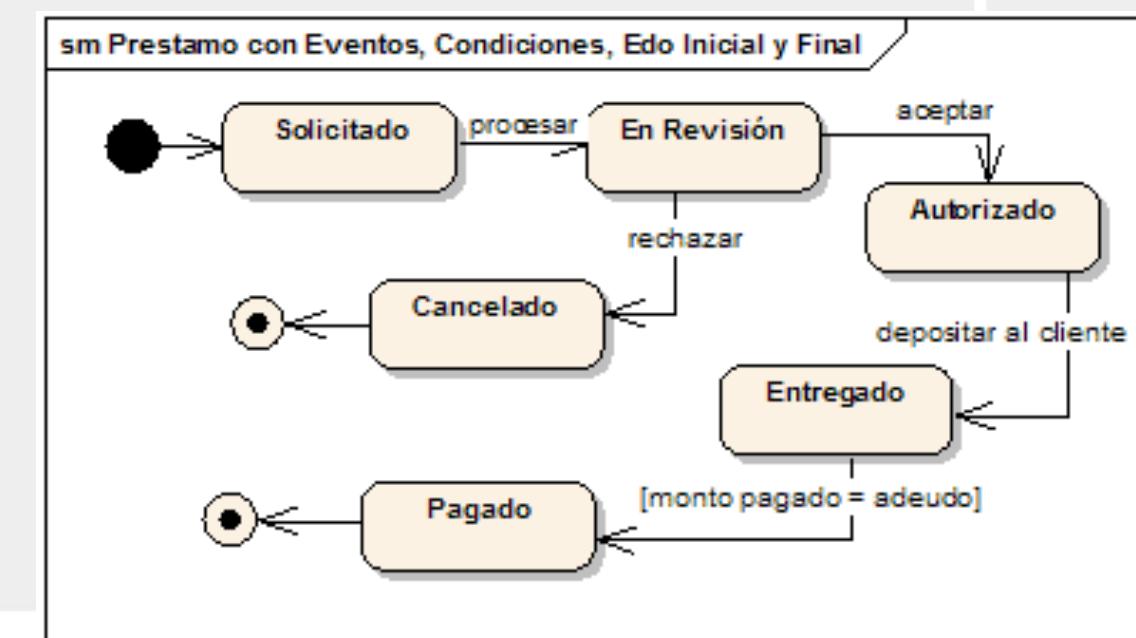
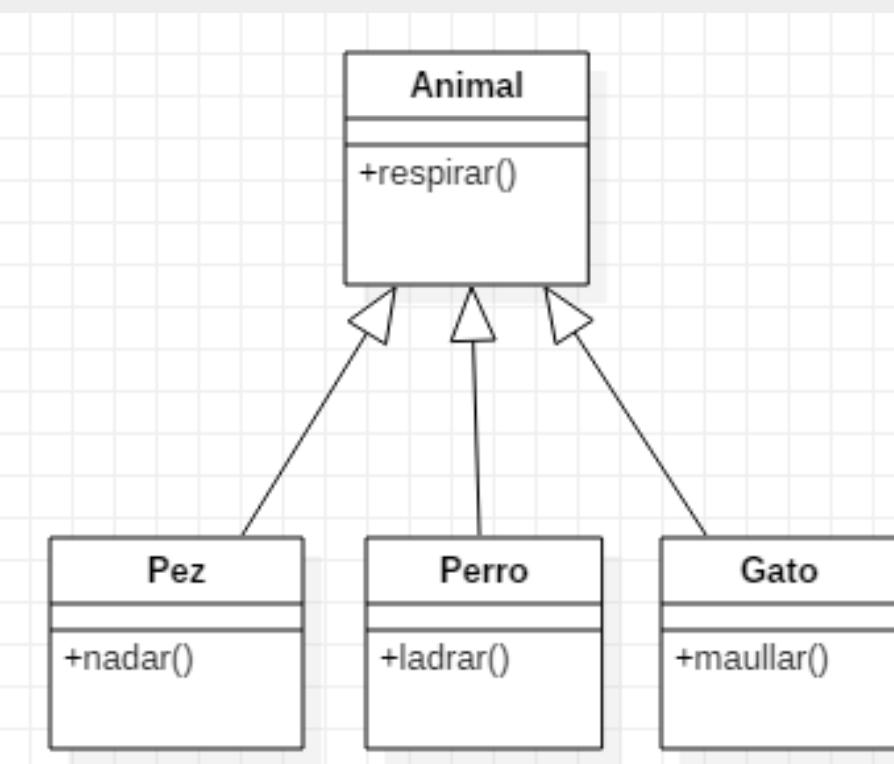
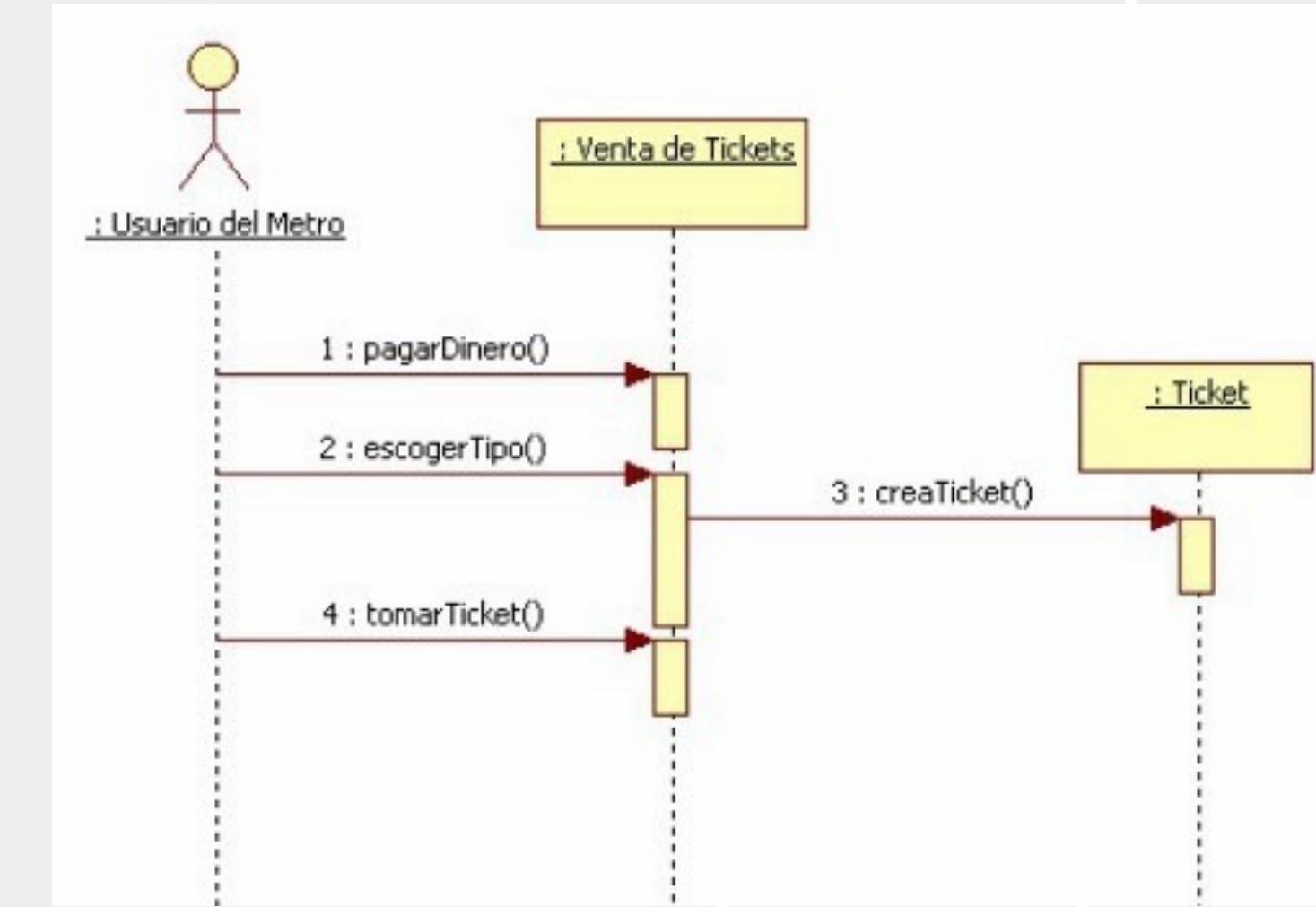
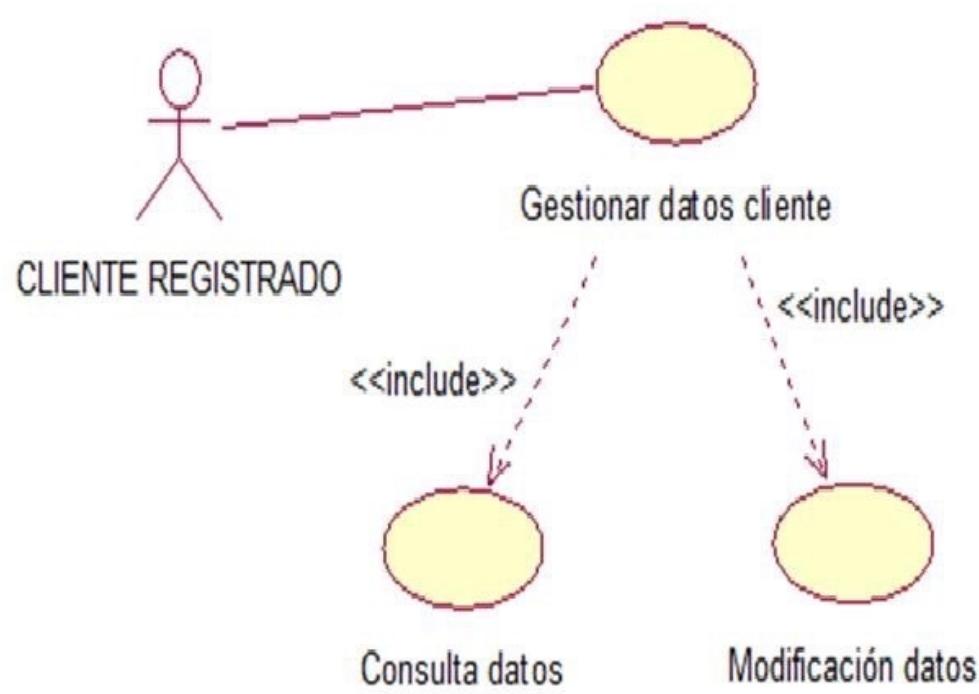
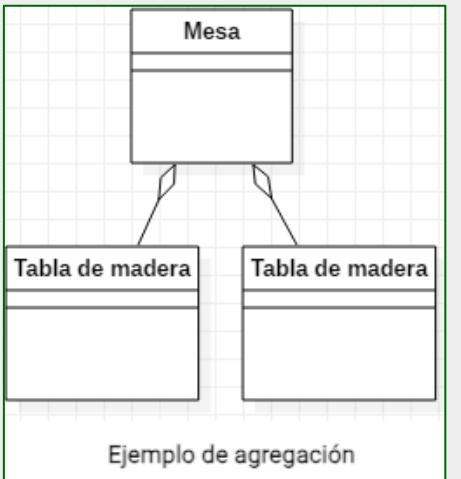
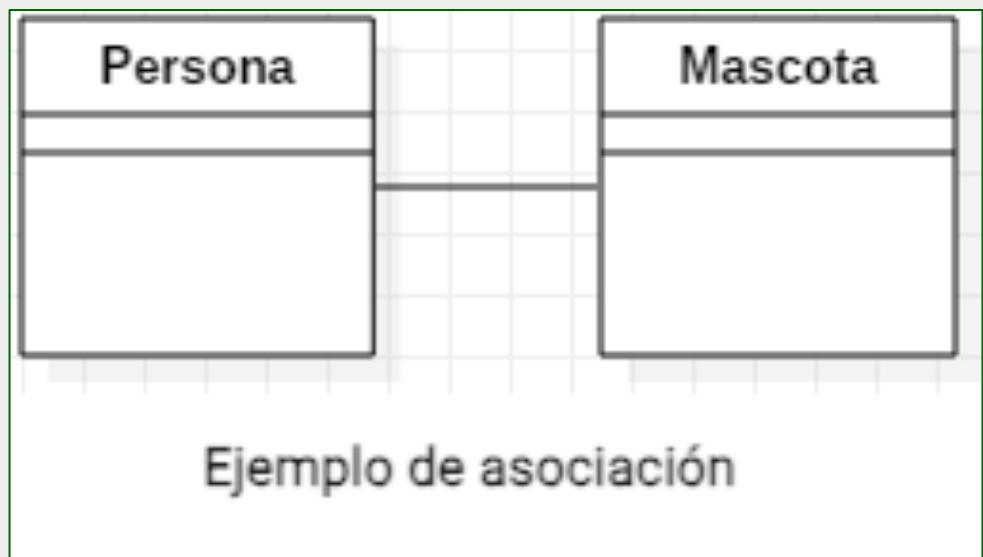
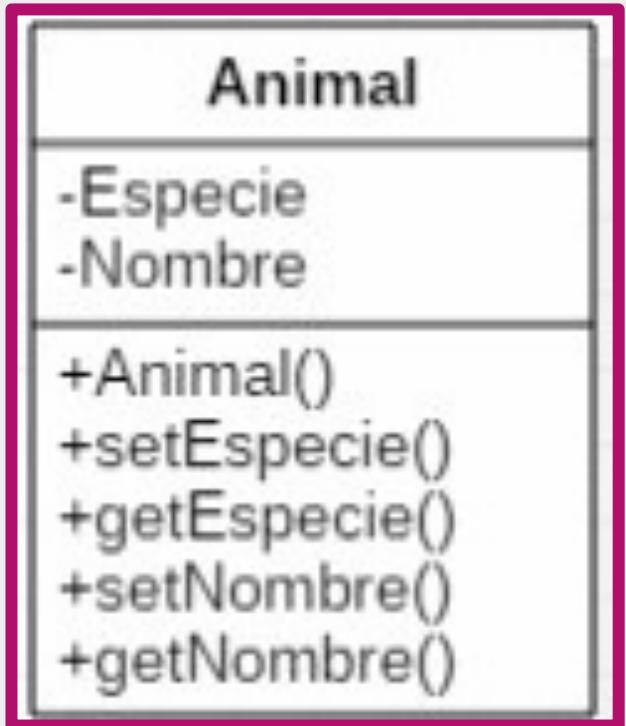
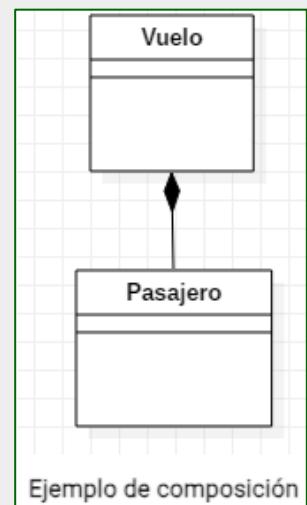


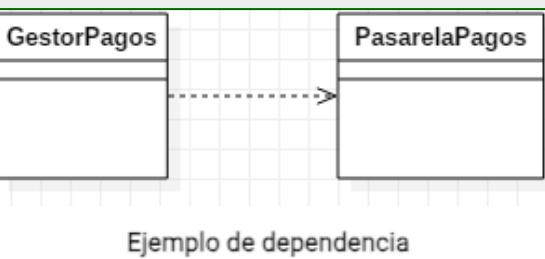
DIAGRAMA DE CLASES



«Las mesas están formadas por tablas de madera y tornillos o, dicho de otra manera, las tablas forman parte de una mesa». Como ves, el tornillo podría formar parte de más objetos, por lo que interesa especialmente su abstracción en otra clase.



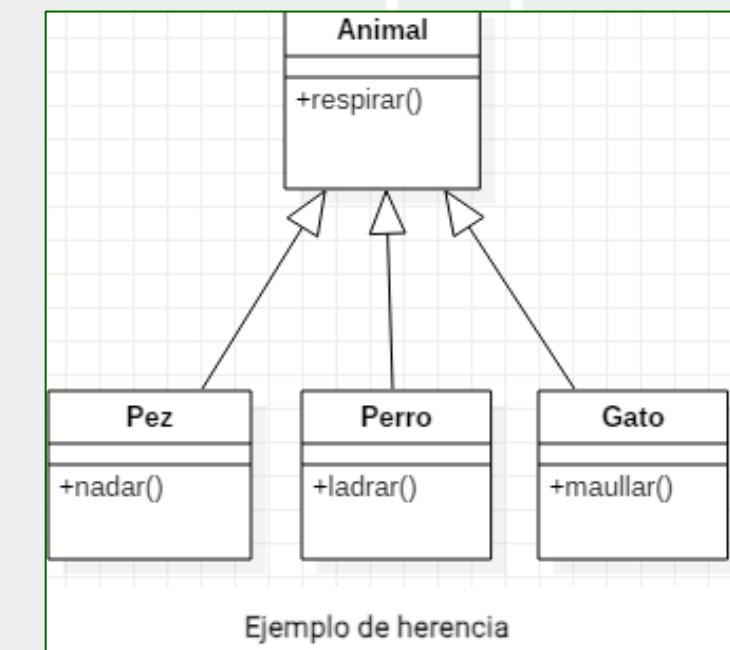
«Un vuelo de una compañía aérea está compuesto por pasajeros, que es lo mismo que decir que un pasajero está asignado a un vuelo»



Representa que una clase requiere de otra para ofrecer sus funcionalidades

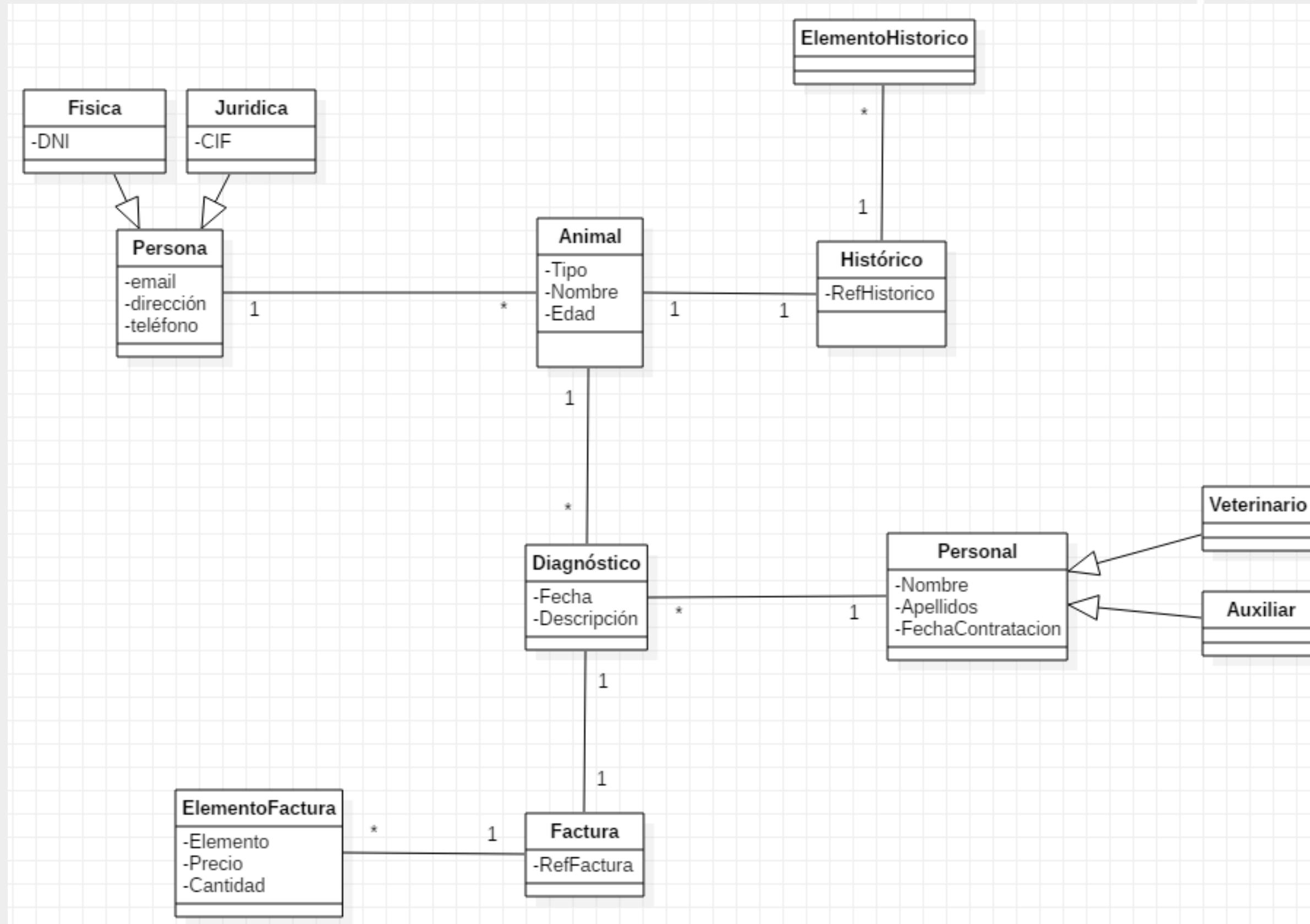
Tipos de relaciones

- ❖ Asociación.
- ❖ Agregación.(rombo vacío)
- ❖ Composición.(rombo relleno)
- ❖ Dependencia.(flecha discontinua)
- ❖ Herencia.



Un pez, un perro y un gato son animales.

EJEMPLO DE UN DIAGRAMA DE CLASES



EJEMPLO DE CASOS DE USO

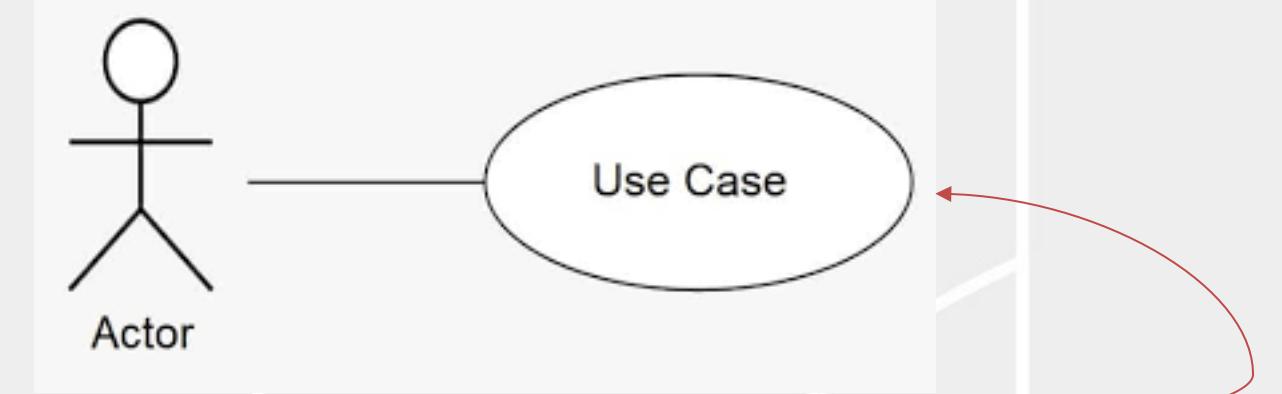
Sistema de un Restaurante

- **Actores:**

Cliente, Camarero, Cocinero, Cajero.

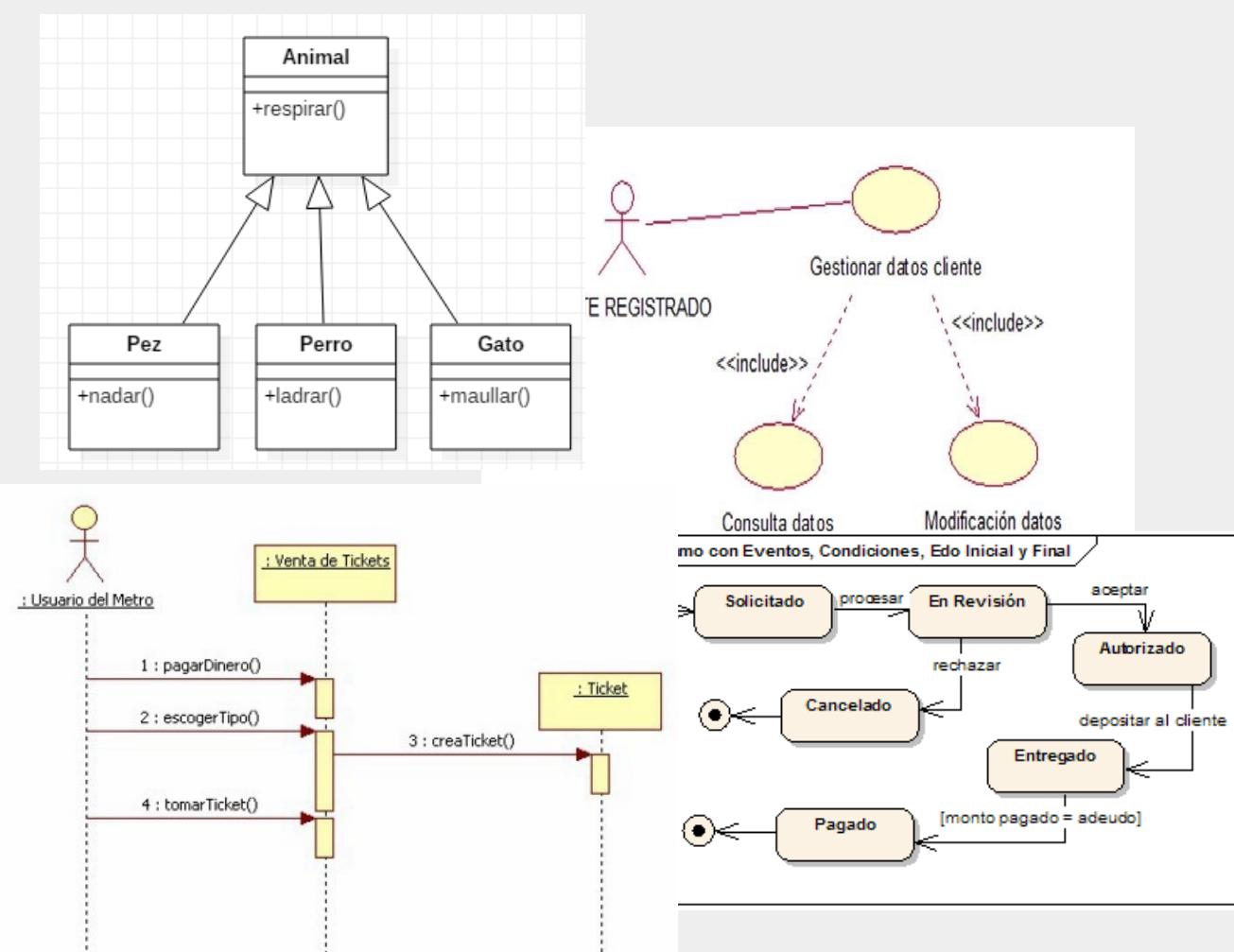
- **Casos de Uso:**

Pedir Comida, Servir Comida, Preparar Comida, Pagar Comida.



RECORDEMOS QUE....

El diseño no es únicamente en papel ni en diagramas, sino también el código fuente.



Microsoft Visual Studio Screenshot:

The screenshot shows a Microsoft Visual Studio interface with the following components:

- Top Bar:** FILE, EDIT, VIEW, TELERIK, PROJECT, BUILD, DEBUG, TEAM, FORMAT, TABLE, TOOLS, TEST, ANALYZE, WINDOW, HELP.
- Toolbar:** Standard icons for file operations.
- Status Bar:** Firefox - Debug - (New Inline Style...), Heading 1 - 32px, B, I, U, A, etc.
- Code Editor:** The code editor displays the file `Incidents.aspx.cs` with C# code for an ASPX page. The code includes declarations for `ContentPlaceHolder`s, a `GridView`, and various `BoundField`s.
- Solution Explorer:** Shows a solution named "WebClient" containing five projects: Data, Presentation, Business, Services, and Common. The "Presentation" project is expanded, showing "Client" and "Shell".
- Toolbars:** Solution Explorer, Team Explorer, Class View.
- Properties Window:** Shows properties for selected items, including an `H1` element with a `(id)` attribute.
- Design View:** Below the code editor, a preview of the "Incidents List" page is shown, featuring a header "Incidents List" and a table with columns: product, account, agent, number, shortDescription, description, createdOn, source, contactInformation, priority, status, severity, and initialDiagnos.

CONSTRUCCIÓN DEL SOFTWARE

Creación de software funcional o productivo.

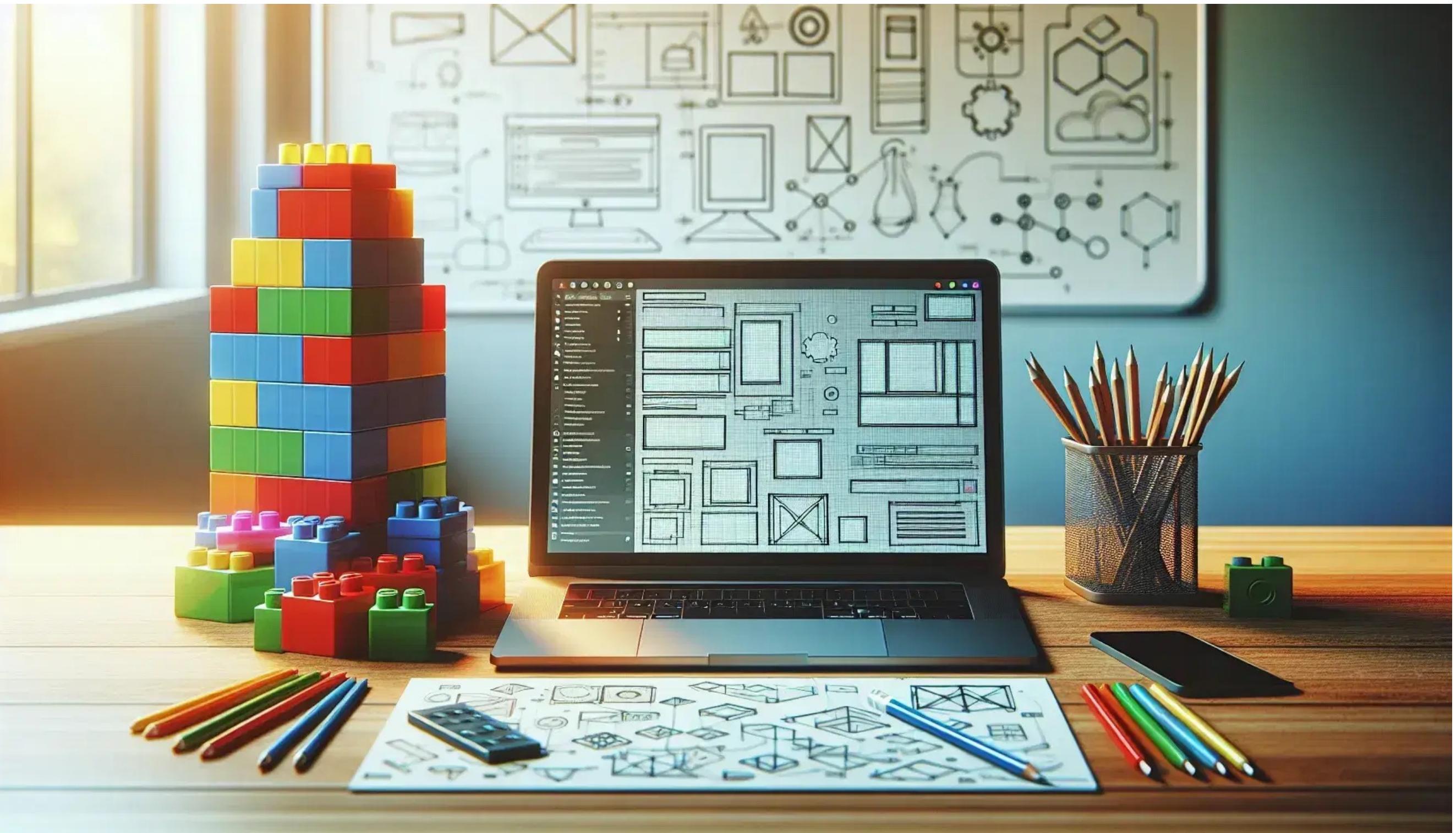
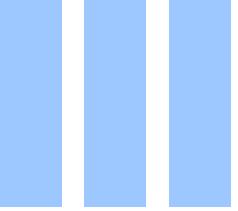
La calidad de la construcción afecta de manera importante la calidad del software.



FUNDAMENTOS DE LA CONSTRUCCIÓN DEL SOFTWARE

- Minimizar la complejidad.
- Anticipar a los cambios.
- Construcción de verificación
- Reutilización.
- Estándares en la construcción.

CONSTRUCCIÓN DEL DISEÑO DEL SOFTWARE



Desarrollo de un sistema de gestión de biblioteca.

1. Diseño Conceptual

- Se identifican los **módulos principales**: gestión de usuarios, préstamos, catálogo de libros y reportes.

Definimos las **entidades y relaciones**:

Usuario → Préstamo → Libro.

Establecemos los **requerimientos funcionales**

(qué debe hacer el sistema) en esta etapa no entramos en detalles técnicos.

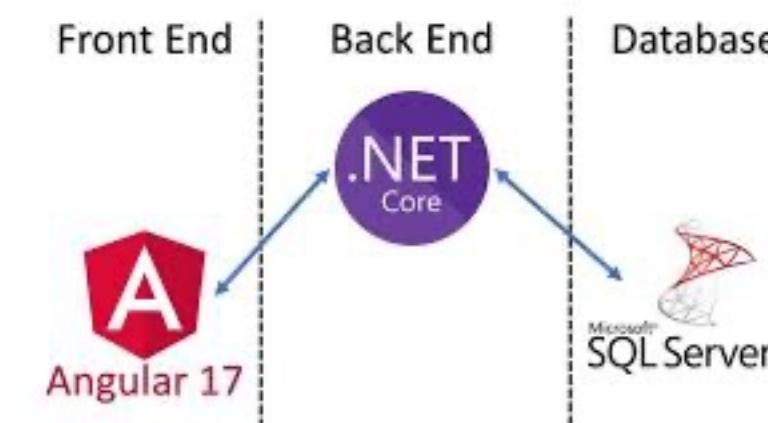
Herramienta que podemos usar: Diagramas de flujo o diagramas de casos de uso UML.



2. Diseño Lógico

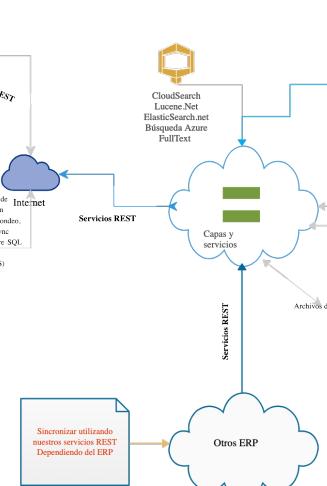
En el sistema de biblioteca anterior, se pasa a estructurar cómo se comunicará internamente:

- Se crean **diagramas de clases UML**, donde se definen atributos y métodos.
 - Se diseña la **base de datos lógica**:
 - Tabla Usuarios(id, nombre, correo, tipoUsuario)
 - Tabla Libros(id, título, autor, estado)
 - Tabla Prestamos(id, idUsuario, idLibro, fechaInicio, fechaFin)
 - Se definen **interfaces** entre módulos, como funciones o servicios que intercambian datos.
- Herramienta usada:** Modelado UML, modelo entidad-relación (MER).



3. Diseño Físico

- Se eligen las tecnologías:
 - Backend en .NET Core
 - Base de datos en SQL Server
 - Frontend en React.js
- Se definen los componentes físicos del sistema:
 - Servidor web
 - Servidor de base de datos
 - Aplicación cliente
- Se crean diagramas de despliegue (Deployment Diagrams) mostrando cómo se distribuyen los componentes en la infraestructura.



4. Diseño de la Interfaz de Usuario (UI/UX)

Para el módulo de préstamos:

- Se diseñan prototipos o wireframes de pantallas:
 - Pantalla de login
 - Panel principal
 - Formulario de préstamo

- Se define la navegación entre pantallas y los mensajes de error o validaciones.

Herramienta usada: Figma, Adobe XD o Balsamiq.

The screenshot shows the BiblioSys library management application interface. The top navigation bar includes the logo, the title "BiblioSys Sistema de Gestión", and a "Nuevo Libro" button. The main content area is divided into sections:

- Usuarios:** A sidebar with icons for "Catálogo", "Préstamos", and "Reportes".
- Catálogo:** A search bar with placeholder text "Buscar por título, autor o ISBN...". Below it is a table listing books:

Título	Autor	ISBN	Categoría	Año	Disponibles	Estado
Cien años de soledad	Gabriel García Márquez	978-0307474728	Ficción	1967	3 / 5	D
Don Quijote de la Mancha	Miguel de Cervantes	978-8424936464	Clásicos	1605	1 / 4	L
La sombra del viento	Carlos Ruiz Zafón	978-8408163374	Misterio	2001	0 / 6	N
1984	George Orwell	978-0451524935	Distopía	1949	2 / 3	D
El principito	Antoine de Saint-Exupéry	978-0156012195	Infantil	1943	5 / 8	D

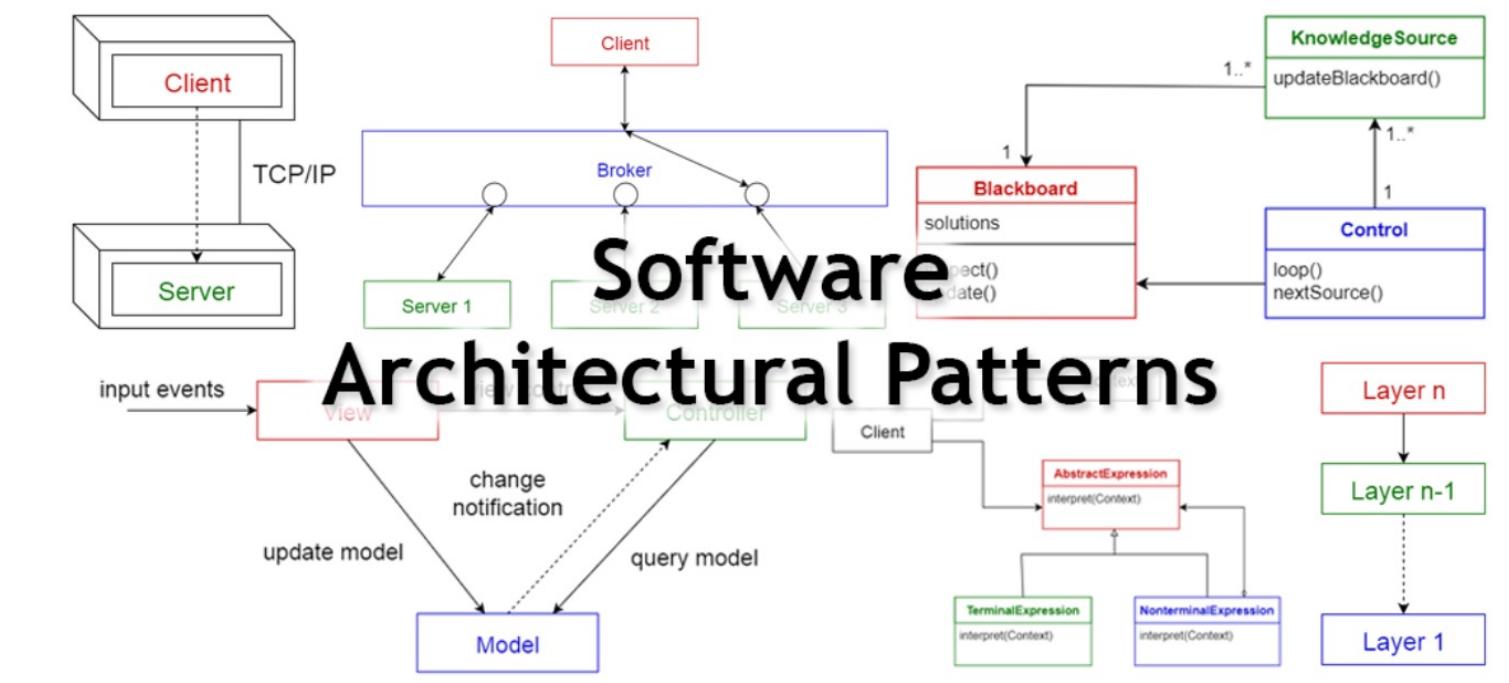
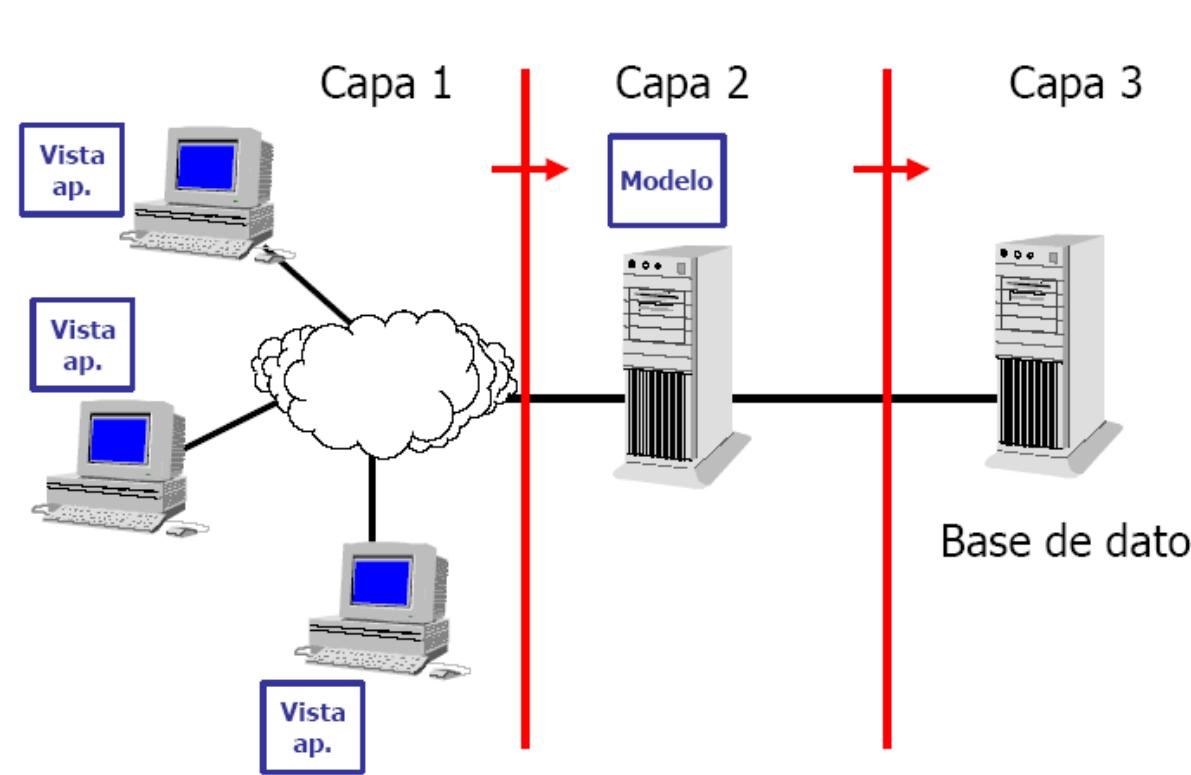
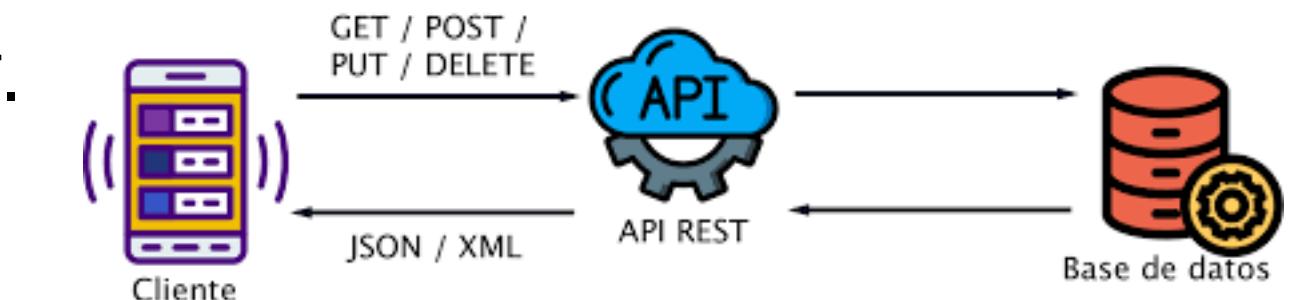
5. Diseño Arquitectónico

Arquitectura de microservicios, donde cada módulo (libros, clientes, pagos, préstamos libros)

funciona de manera independiente y se comunica mediante APIs REST.

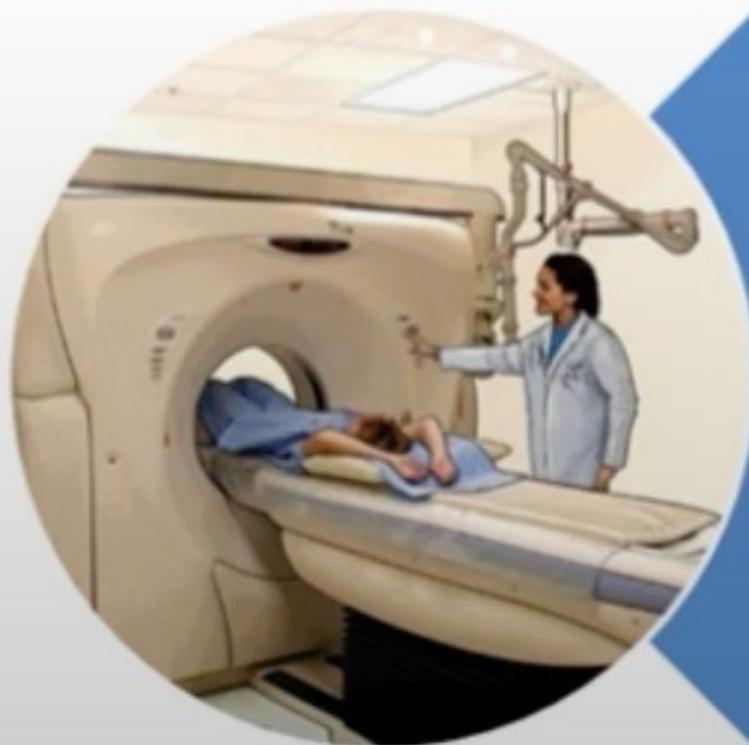
Se define la arquitectura de software:

- Capa de presentación
- Capa de negocio
- Capa de datos
- **Herramienta usada: Diagramas de arquitectura C4 o UML.**

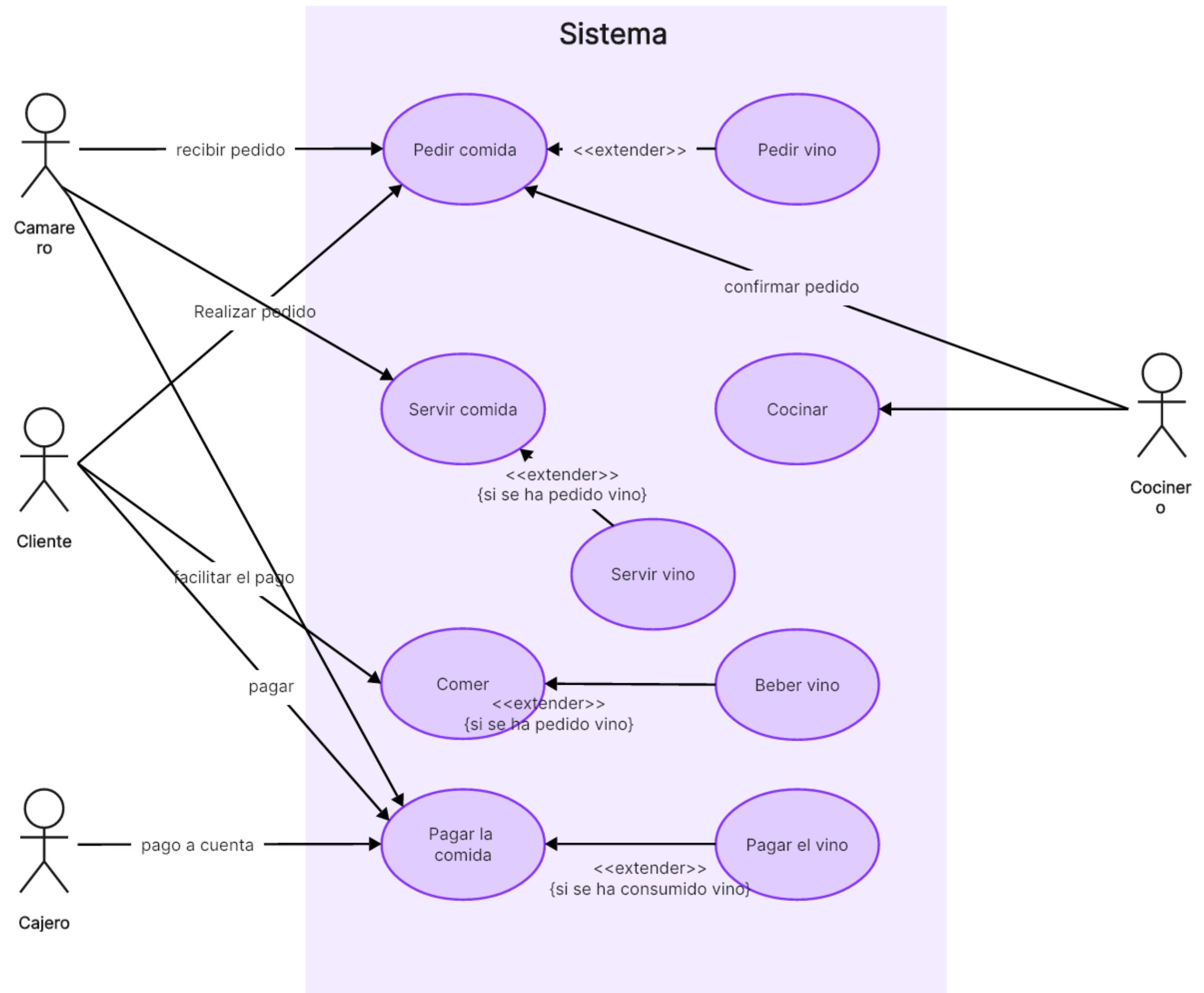
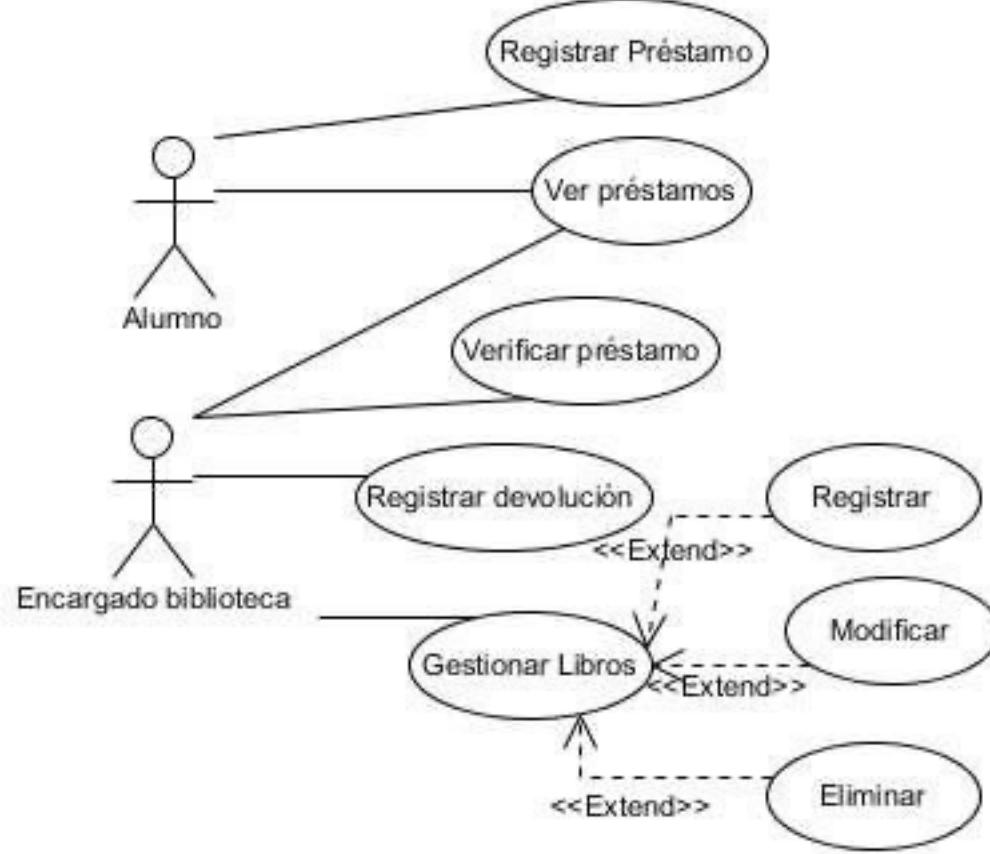




¿Viajaría Ud. en un avión
cuyo software ha sido
diseñado por Ud.?



¿Por qué el software es
importante para la sociedad
actual?



DEPURACIÓN Y PRUEBAS DEL SOFTWARE

The screenshot shows a software development environment with a debugger interface. The toolbar at the top has a 'Continuar' (Continue) button highlighted with a yellow box. Below the toolbar, the status bar shows 'Proceso: [20996] BubbleSort.exe'. The main area displays a C# code editor for 'Program.cs' with a breakpoint at line 8. A tooltip for a System.IndexOutOfRangeException is shown, indicating an index was outside the bounds of the array. The tooltip includes the error message: 'Excepción no controlada System.IndexOutOfRangeException: 'Index was outside the bounds of the array.''. The code editor shows a bubble sort algorithm.

```
using System;
namespace BubbleSort
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(string.Join(", ", BubbleSort(new int[] { 5, 4, 7, 1, 2, 0 })));
        }

        private static int[] BubbleSort(int[] arreglo)
        {
            for (int i = 0; i < arreglo.Length; i++)
            {
                for (int j = 0; j < arreglo.Length - i; j++)
                {
                    if (arreglo[j] > arreglo[j + 1])
                    {
                        var temp = arreglo[j + 1];
                        arreglo[j + 1] = arreglo[j];
                        arreglo[j] = temp;
                    }
                }
            }
            return arreglo;
        }
    }
}
```

Mejoramos la calidad del Software

DESPLIEGUE E IMPLEMENTACIÓN DEL SOFTWARE



MANTENIMIENTO DEL SOFTWARE

Mantenimiento predictivo: Evalúa el flujo de ejecución del programa para predecir con certeza el momento en el que se producirá la falla, y así determinar cuándo es adecuado realizar los ajustes correspondientes.

Mantenimiento correctivo: Corrige los defectos encontrados en el *software*, y que originan un comportamiento distinto al deseado. Estas fallas pueden ser de procesamiento, rendimiento (por ejemplo, uso ineficiente de los recursos de *hardware*), programación (inconsistencias en la ejecución), seguridad o estabilidad, entre otras.

Mantenimiento adaptativo: Si se requiere cambiar el entorno de uso de la aplicación (que incluye al sistema operativo, a la plataforma de *hardware* o, en el caso de las aplicaciones web, al navegador), puede ser indispensable modificarla para mantener su plena funcionalidad en estas nuevas condiciones.



Mantenimiento evolutivo: Es un caso especial donde la adaptación resulta prácticamente obligatoria, ya que de lo contrario el programa quedaría obsoleto con el paso del tiempo. Por ejemplo, el cambio de versión en un navegador (muchas veces impuesto sin el consentimiento del usuario) suele obligar a realizar ajustes en plugins y aplicaciones web.

Mantenimiento perfectivo: Por distintas razones, el usuario puede solicitar el agregado de nuevas funcionalidades o características no contempladas al momento de la implementación del software. El mantenimiento perfectivo adapta la aplicación a este requerimiento.

PRÁCTICA CASOS DE USO



Universidad CENFOTEC

Escuela de Ingeniería del Software

Código del curso: SOFT-09

Nombre del curso: Introducción a la ingeniería del software

Docente facilitador: Rosibel Enríquez Vargas

PRÁCTICA EN CLASE

1. Objetivo del ejercicio:

Evaluar la capacidad del estudiante para analizar los requerimientos del sistema y representarlos gráficamente mediante un diagrama de casos de uso UML, identificando correctamente las interacciones entre usuarios y el sistema.

2. Instrucciones generales

Lea cuidadosamente el enunciado del sistema hospitalario y los requerimientos funcionales descritos.

1. Identifique los actores principales del sistema.

2. Determine los casos de uso que cada actor puede realizar, según las funcionalidades descritas.

3. Dibuje el diagrama de casos de uso UML, representando:

- Los actores con sus respectivos roles.
- Los casos de uso.
- Las relaciones entre actores y casos de uso.
- Relaciones de inclusión o extensión si las considera necesarias.

El diagrama debe elaborarse de forma clara, ordenada y legible, utilizando una herramienta de modelado (Lucidchart, Draw.io, Visual Paradigm, StarUML, etc.)



Contexto:

Se le solicita desarrollar un Sistema Hospitalario que permita registrar, consultar y administrar la información relacionada con los pacientes, los doctores, las citas médicas y las distintas sedes del hospital en Costa Rica, mejorando la eficiencia en la atención y el acceso a los datos.

"Sistema de Gestión Hospitalaria "Cenfomédica"

El Hospital "Cenfomédica" desea implementar un sistema informático que le permita administrar de forma eficiente la información de pacientes, doctores, citas médicas y sedes.

Actualmente, gran parte del proceso se realiza en hojas de cálculo de Excel y registros manuales (libros de actas, cuadernos, etc.) lo que genera que haya pérdida de información, citas duplicadas y dificultad para dar seguimiento a los pacientes. El nuevo sistema debe permitir el control centralizado de toda la información hospitalaria desde cualquier sede de Costa Rica.

A continuación, encontrará los **actores** que participarán en este sistema.

El administrador del sistema: gestiona la información general (sedes, doctores, pacientes).

El doctor: consulta sus citas y el historial médico de sus pacientes.

El paciente: puede registrarse, solicitar citas y consultar sus citas programadas.

El recepcionista: agenda, modifica o cancela citas según disponibilidad.

Requerimientos funcionales

RF1: El sistema debe permitir registrar nuevos pacientes con datos personales (nombre, cédula, edad, dirección, teléfono, correo electrónico).

RF2: El sistema debe permitir consultar, modificar y eliminar la información de los pacientes.

RF3: El sistema debe permitir asociar a cada paciente su historial de citas médicas.

RF4: El sistema debe mostrar la lista de doctores disponibles por especialidad y sede.

RF5: El sistema debe permitir consultar el horario de atención de cada doctor.

RF6: El sistema debe permitir crear, modificar o cancelar citas médicas.



Universidad
CENFOTEC

SOMOS LO QUE SABEMOS

ii Feliz noche !!

