PROYECTO – Entrega 3

OBJETIVOS, REQUISITOS Y ACTORES

- El **objetivo general** es la **meta principal** que se busca alcanzar con la realización del trabajo. No se centra en detalles, sino en la visión amplia de lo que se quiere lograr al finalizar.
- Características:
  - **Meta** → Representa el fin último al que apunta el proyecto.
  - **Amplios** → No se enfoca en cosas pequeñas o tareas concretas, sino en lo global.
  - **Largo plazo** → Su cumplimiento se ve al final del proyecto, no de inmediato.
  - **Visión** → Refleja hacia dónde quieres llegar, como una guía.
  - **Inspiración** → Motiva y da sentido al proyecto.
  - **No detallado** → No entra en pasos específicos, esos se desarrollan en los objetivos específicos.

Ejemplo: “Desarrollar una aplicación web que facilite la gestión de reservas en un restaurante"

OBJETIVOS, REQUISITOS Y ACTORES

- Para escribir un objetivo general, debe empezar con verbos como *desarrollar, crear, implementar, diseñar, analizar*.
  Ejemplo: *"Crear una plataforma web para…"*.
- **Claridad y concisión** → Debe ser directo y entendible, sin rodeos.
- **Viabilidad** → El objetivo debe ser posible:
  - **Realista**: No algo imposible o demasiado ambicioso.
  - **Con recursos disponibles**: Que se pueda hacer con los conocimientos, tiempo y herramientas que tienes en el ciclo.
  - **Con el proyecto propuesto**: Que realmente responda a lo que plantea tu proyecto.

OBJETIVOS SMART

- Los objetivos SMART son una forma de redactar y planificar metas de un proyecto de manera clara y alcanzable.

OBJETIVOS SMART

La palabra **SMART** es un acrónimo en inglés que significa:

- **S – Specific (Específico)**
  - El objetivo debe estar bien definido, sin ambigüedades.
  - Responde a *¿qué quiero lograr exactamente?*
  - Ejemplo en DAW: *"Desarrollar un módulo de autenticación con login y registro de usuarios"*.
- **M – Measurable (Medible)**
  - Debe poder evaluarse con indicadores o resultados concretos.
  - Responde a *¿cómo sé si lo logré?*
  - Ejemplo: *"El sistema debe permitir que al menos 100 usuarios se registren y accedan correctamente"*.
- **A – Achievable (Alcanzable)**
  - Debe ser posible de realizar con los conocimientos y recursos que tienes.
  - Responde a *¿puedo lograrlo con las herramientas y el tiempo disponibles?*
  - Ejemplo: *"Implementar pasarela de pago usando una API externa (Stripe o PayPal)"*.
- **R – Realistic (Realista)**
  - El objetivo debe tener sentido y ser útil para el proyecto.
  - No debe ser ni demasiado básico ni excesivamente ambicioso.
  - Ejemplo: *"Incorporar un sistema de comentarios en los productos de la tienda online"*.
- **T – Time-bound (Tiempo limitado)**
  - Todo objetivo debe tener un plazo definido.
  - Responde a *¿cuándo debo tenerlo terminado?*
  - Ejemplo: *"Completar el módulo de gestión de usuarios antes de la entrega de la primera evaluación"*.

OBJETIVOS SMART

- EJEMPLO: En nuestro proyecto de DAW, primero definimos un **Objetivo General** (ejemplo: *"Desarrollar una aplicación web de reservas para un restaurante"*) y después lo dividimos en **objetivos SMART**, como:
  - *"Diseñar la base de datos para la gestión de mesas y clientes antes de la segunda semana del proyecto"*.
  - *"Implementar un sistema de notificaciones por correo electrónico para confirmar reservas antes del final del segundo trimestre"*.

OBJETIVOS ESPECÍFICOS

- Son la “traducción” del **objetivo general** en metas más concretas, detalladas y medibles. Estos objetivos sirven para marcar los pasos necesarios que harán posible cumplir el objetivo general de tu aplicación web.

  - **Objetivo general** → es amplio, global, marca la meta final.
  - **Objetivos específicos** → son más concretos, desglosan el objetivo general en tareas claras y medibles.

¿Cómo escribir un objetivo específico?

- **Verbo en infinitivo**
  - Igual que el objetivo general, debe comenzar con verbos como: *desarrollar, diseñar, implementar, probar, documentar*.
  - Ejemplo: *"Diseñar la base de datos relacional para gestionar usuarios y reservas"*.
- **Detallado: ¿Qué, cómo, cuándo, dónde?**
  - Debe concretar el *qué* vas a hacer, *cómo* lo harás, *cuándo* (plazos), y *dónde* (en qué parte del sistema).
  - Ejemplo: *"Implementar un sistema de autenticación con login y registro de usuarios antes de la entrega de la primera evaluación"*.
- **Indicadores de éxito: criterios**
  - Deben permitir medir si el objetivo se ha cumplido.
  - Ejemplo: *"El sistema debe permitir al menos 100 registros exitosos y accesos correctos"*.

Ejemplo de objetivo específico

- **Objetivo general**: *"Desarrollar una aplicación web para la gestión de reservas en un restaurante"*.
- **Objetivos específicos**:
  - *Diseñar la base de datos relacional que permita gestionar usuarios, mesas y reservas antes de la tercera semana del proyecto.*
  - *Implementar un sistema de login y registro seguro con validación de correo electrónico antes de la entrega de la primera evaluación.*
  - *Integrar un módulo de notificaciones automáticas por correo para confirmar reservas antes del segundo trimestre.*

REQUISITOS FUNCIONALES

- Son **descripciones concretas de lo que el sistema debe ser capaz de hacer**. Es decir, explican las funciones que el software tendrá y cómo se comportará ante determinadas acciones del usuario.

  - Serían cosas como: *"El sistema debe permitir al usuario registrarse con nombre, correo y contraseña"* o *"El sistema debe enviar un correo de confirmación al realizar una reserva"*.

REQUISITOS FUNCIONALES - CARACTERÍSTICAS

- **Especificidad** → Cada requisito describe con detalle una acción concreta del sistema, no algo vago.
  Ejemplo: *"El sistema debe permitir que un usuario admin edite los datos de un cliente"*.
- **Medibles** → Debe poder verificarse si el requisito se cumple o no.
  Ejemplo: *"El login debe permitir al menos 100 inicios de sesión concurrentes sin errores"*.
- **Basados en la acción** → Siempre describen lo que el sistema hace (acciones).
  Ejemplo: *"El sistema debe generar un PDF con la factura de cada compra"*.

REQUISITOS FUNCIONALES

- **Descripción clara y detallada** → Cada requisito debe explicar bien qué hace el sistema.
  Ejemplo: *"El sistema debe permitir la recuperación de contraseña mediante correo electrónico"*.
- **Enumeración y organización** → Deben estar numerados y ordenados para poder revisarlos y no olvidar ninguno.
  Ejemplo:
  - RF1: El sistema debe permitir registrar nuevos usuarios.
  - RF2: El sistema debe autenticar usuarios mediante login.
  - RF3: El sistema debe permitir la edición del perfil.
- **Evitar ambigüedades** → Usar un lenguaje directo, sin términos confusos.
  Mal ejemplo: *"El sistema debe ser rápido"*.
  Buen ejemplo: *"El sistema debe responder a las consultas en menos de 2 segundos"*.

Los **requisitos funcionales** son la lista de funciones que tu aplicación web **debe cumplir obligatoriamente** para que se considere terminada.

REQUISITOS NO FUNCIONALES

- A diferencia de los **funcionales** (que dicen *qué hace* el sistema), los no funcionales indican **cómo debe comportarse el sistema** o bajo qué condiciones debe funcionar.
  - Funcional → *"El sistema debe permitir a los usuarios registrarse"*.
  - No funcional → *"El sistema debe responder a las solicitudes de registro en menos de 2 segundos"*.

REQUISITOS NO FUNCIONALES - CARACTERÍSTICAS

- **Calidad y restricciones** → Se centran en aspectos como rendimiento, seguridad, usabilidad, compatibilidad o disponibilidad.
  Ejemplo: *"El sistema debe estar disponible el 99% del tiempo"*.
- **Transversales** → No afectan a una sola función, sino al sistema completo.
  Ejemplo: *"La interfaz debe ser accesible desde cualquier navegador moderno"*.
- **Difíciles de medir directamente** → Suelen medirse con métricas técnicas o con la satisfacción del usuario.
  Ejemplo: *"La aplicación debe poder manejar 500 usuarios conectados simultáneamente sin caídas"*.

REQUISITOS NO FUNCIONALES - Cómo se Escriben

Especificar estándares y métricas

- Se deben incluir valores concretos y verificables.
  - Ejemplo: *"El sistema debe cargar la página principal en menos de 3 segundos con 100 usuarios concurrentes"*.

Priorizar según el impacto

- No todos los requisitos no funcionales tienen la misma importancia.
  - Ejemplo: En una aplicación bancaria, la **seguridad** es prioritaria sobre la estética.

Revisión continua

- Estos requisitos se ajustan durante el proyecto, porque a veces solo se entienden mejor conforme se prueba el sistema.

REQUISITOS NO FUNCIONALES - EJEMPLOS

- **Rendimiento**: El sistema debe soportar al menos 200 usuarios concurrentes.
- **Seguridad**: Todas las contraseñas deben almacenarse encriptadas.
- **Usabilidad**: La aplicación debe ser responsive y usable en móviles y ordenadores.
- **Disponibilidad**: El sistema debe estar en funcionamiento las 24 horas del día.

REQUISITOS NO FUNCIONALES - PERFILES DE USUARIO

- Son **individuos o sistemas externos que interactúan con tu aplicación web**.

En otras palabras, representan los diferentes **roles** que utilizarán o se relacionarán con el sistema.

- Importante: **no se refieren a una persona concreta**, sino al **rol** que desempeña en el sistema.

REQUISITOS NO FUNCIONALES - TIPOS DE ACTORES

- **Usuarios finales**
  - Son los que usan directamente la aplicación para cumplir una necesidad.
  - Ejemplo: en una tienda online → los clientes que compran productos.
- **Administradores**
  - Tienen más permisos y responsabilidades que los usuarios comunes.
  - Ejemplo: en la misma tienda online → el administrador que gestiona productos, inventario y usuarios.
- **Sistemas externos**
  - Otros sistemas con los que tu aplicación se comunica o integra.
  - Ejemplo: una API de pago como PayPal o Stripe, o un servicio de envío de correos.

REQUISITOS NO FUNCIONALES - TIPOS DE ACTORES

- Los **perfiles de usuario o actores** nos ayudan a definir:
  - **Quién interactúa con el sistema.**
  - **Qué rol cumple cada uno.**
  - **Qué necesidades tiene dentro del sistema.**

Esto es útil para luego elaborar los **casos de uso** o diagramas UML, donde se detalla qué puede hacer cada actor con tu aplicación.

REQUISITOS NO FUNCIONALES - TIPOS DE ACTORES

Es importante describir **correctamente** para un proyecto de software

- **Identidad del Actor**
  - Define *quién es* dentro del sistema (cliente, administrador, sistema externo, etc.).
  - Ejemplo: *"Cliente registrado"*.
- **Objetivos y Necesidades**
  - Qué busca conseguir ese actor dentro de la aplicación.
  - Ejemplo: *"Un cliente quiere reservar una mesa en el restaurante de forma rápida"*.
- **Comportamientos**
  - Cómo interactúa con el sistema.
  - Ejemplo: *"El cliente navega por el menú, selecciona una hora y realiza la reserva"*.
- **Restricciones y Limitaciones**
  - Límites que tiene ese actor.
  - Ejemplo: *"El cliente no puede acceder al panel de administración"*.

REQUISITOS NO FUNCIONALES - TIPOS DE ACTORES

Identificar a todos los actores

- Hacer una lista de todos los perfiles que van a interactuar con el sistema.

Definir objetivos y necesidades para cada actor

- Cada actor debe tener un propósito claro dentro de la aplicación.

Describir el comportamiento esperado

- Explicar qué acciones realiza y cómo se relaciona con el sistema.

Restricciones y limitaciones

- Indicar lo que *no puede hacer* para marcar la diferencia entre roles.

REQUISITOS NO FUNCIONALES - TIPOS DE ACTORES

- **Actor 1: Cliente**
  - Objetivo: Reservar una mesa de forma sencilla.
  - Comportamiento: Se registra, inicia sesión, consulta disponibilidad y realiza una reserva.
  - Restricciones: No puede acceder a la base de datos ni ver reservas de otros usuarios.
- **Actor 2: Administrador**
  - Objetivo: Gestionar reservas y usuarios.
  - Comportamiento: Puede añadir, modificar o eliminar reservas; gestionar usuarios.
  - Restricciones: Debe identificarse con credenciales de administrador.
- **Actor 3: Sistema externo (Pasarela de pago)**
  - Objetivo: Validar pagos online de las reservas.
  - Comportamiento: Procesa transacciones seguras.
  - Restricciones: No accede a datos personales fuera de la transacción.

REQUISITOS NO FUNCIONALES - ACTORES PRIMARIOS Y SECUNDARIOS

- Actor Primario
  - **Hace uso del sistema** → Es quien interactúa directamente con la aplicación.
  - **Tiene un objetivo** → Busca cumplir una necesidad concreta usando el sistema.
  - **Suele iniciar un caso de uso** → Es el que "dispara" la interacción con el software.
    - Ejemplos en una aplicación web de reservas:
      - Un **cliente** que quiere reservar una mesa.
      - Un **administrador** que quiere gestionar las reservas.
- Actor Secundario
  - **Ofrece un servicio al sistema** → No usa directamente la aplicación como un usuario humano, sino que la apoya o complementa.
  - **No necesariamente tiene objetivos** → No busca un beneficio propio, solo ayuda al sistema a funcionar.
  - **Suele tener un rol reactivo** → Responde a lo que pide el sistema.

Ejemplos en la misma app de reservas:

- Una **pasarela de pago (Stripe, PayPal)** que procesa los cobros.
- Un **servidor de correo** que envía notificaciones de confirmación.

REQUISITOS NO FUNCIONALES - ACTORES PRIMARIOS Y SECUNDARIOS

- **Actores primarios** → usuarios principales del sistema (con un objetivo claro).
- **Actores secundarios** → sistemas o entidades de apoyo que permiten que el sistema funcione, pero no son el foco principal.