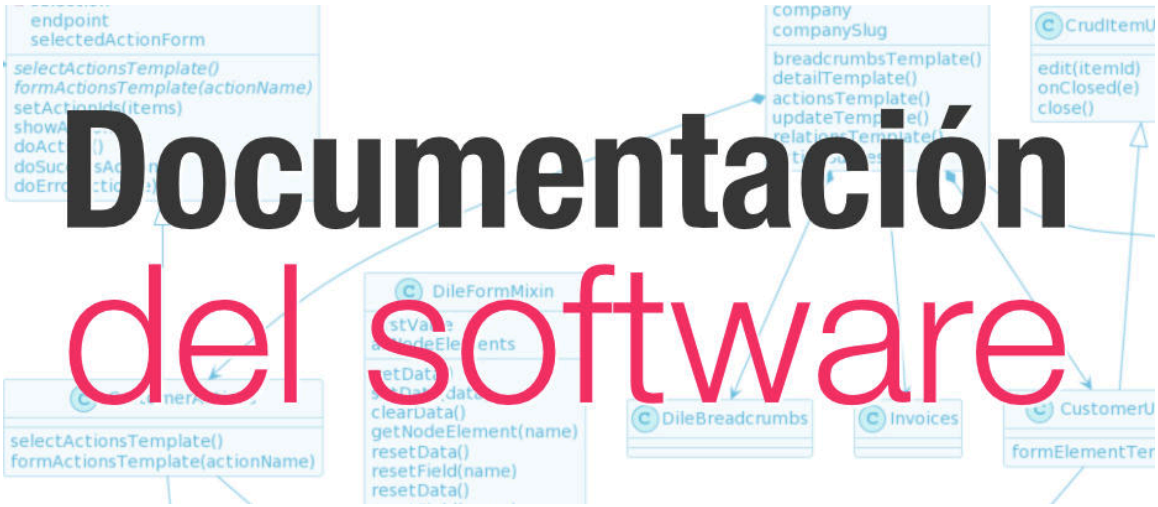
Gestión Desarrollo de Software – 2025

Prof. Politi Raul

Clase 20 (Unidad 7)

Documentación



Introducción

La **documentación técnica** en el desarrollo de software es el conjunto de textos, diagramas, especificaciones y manuales que describen el funcionamiento interno y externo de un sistema informático. Su objetivo principal es **transmitir información clara y estructurada** sobre cómo fue diseñado, desarrollado, implementado y cómo debe mantenerse el software a lo largo del tiempo.

En otras palabras, la documentación técnica **actúa como la memoria del proyecto**, permitiendo que cualquier persona (desarrollador, tester, analista o administrador) pueda comprender

su estructura, propósito y modo de operación sin depender exclusivamente del conocimiento de sus creadores.

Importancia de la Documentación tecnica

La documentación técnica es un **pilar fundamental en la ingeniería de software**, ya que:

- Asegura la **continuidad del proyecto** cuando cambian los integrantes del equipo.
- Facilita el **mantenimiento y la evolución** del sistema.
- Reduce el riesgo de errores por **falta de información** o **mala interpretación**.
- Favorece la **comunicación entre áreas técnicas y no técnicas**, como desarrollo, QA, soporte o clientes.
- Sirve como **base para auditorías y revisiones de calidad**.

En proyectos grandes o distribuidos, la ausencia de documentación puede derivar en **pérdida de tiempo, duplicación de trabajo y fallas graves en la implementación**.

Tipos de documentación técnica

La documentación técnica puede clasificarse según su propósito y destinatario. A continuación, se describen los tipos más comunes:

1. Documentación del sistema

Incluye la descripción completa de la **arquitectura del software**, los **módulos**, las **dependencias** y la **interacción entre componentes**.
Suele contener diagramas UML, diagramas de clases, de secuencia, de casos de uso y de despliegue.

Ejemplo: un diagrama que muestra cómo se comunican el servidor, la base de datos y el cliente en una aplicación web.

---

2. Documentación del código

Es aquella que explica **cómo está implementado el sistema a nivel de código fuente**.
Incluye comentarios en el código, convenciones de nombres, estructura de carpetas, y explicaciones sobre funciones o clases.
Herramientas como **JSDoc, Doxygen, Sphinx o Swagger** permiten generar documentación automática a partir de los comentarios en el código.

---

3. Manual de instalación y configuración

Describe los pasos necesarios para **instalar, configurar y ejecutar el software** en distintos entornos (desarrollo, pruebas, producción).
Incluye requisitos de hardware y software, dependencias, comandos de instalación, configuración de bases de datos, variables de entorno y detalles de despliegue.

---

4. Manual técnico de mantenimiento

Está destinado al **equipo de soporte o mantenimiento**. Explica cómo realizar actualizaciones, resolver errores, restaurar copias de seguridad, monitorear el sistema y aplicar parches de seguridad.
Suele incluir procedimientos ante fallos, logs del sistema y recomendaciones de optimización.

---

5. Documentación de APIs

En sistemas modernos que exponen servicios REST o GraphQL, se crea documentación técnica específica para describir **endpoints, métodos, parámetros, códigos de respuesta, ejemplos de peticiones y respuestas**.
Herramientas populares para esto son **Swagger (OpenAPI)** y **Postman Documentation**.

---

6. Manual de usuario técnico

Orienta a usuarios avanzados o técnicos que utilizan el software, explicando comandos, scripts o configuraciones especiales.
Difiere del manual de usuario final (más orientado al cliente o consumidor del sistema).

Contenidos mínimos de la documentación técnica

Una documentación técnica bien estructurada debe incluir:

- **Resumen del sistema:** objetivo, alcance y características principales.
- **Arquitectura del sistema:** capas, componentes y flujos de datos.
- **Diseño de base de datos:** modelo entidad-relación, estructuras de tablas, relaciones y restricciones.
- **Interfaces:** descripción de las pantallas, formularios o endpoints.
- **Dependencias:** frameworks, librerías, SDKs o entornos necesarios.

- **Control de versiones:** información del repositorio y convención de ramas.
- **Plan de mantenimiento:** políticas de actualización y respaldo.
- **Historial de versiones:** cambios relevantes a lo largo del tiempo.

Buenas prácticas para documentar proyectos de software

1. **Documentar desde el inicio:** No esperar al final del desarrollo para hacerlo.
2. **Mantener la documentación sincronizada con el código:** Cada cambio importante debe reflejarse en la documentación.
3. **Usar un lenguaje claro y técnico:** Evitar ambigüedades o expresiones informales.
4. **Incluir ejemplos prácticos:** Ayudan a entender la aplicación real del sistema.
5. **Apoyarse en herramientas de automatización:** Como Markdown, Wiki, ReadTheDocs, o herramientas integradas a GitHub/GitLab.
6. **Establecer una estructura estándar:** Para que todos los proyectos del equipo sigan el mismo formato.
7. **Revisar periódicamente:** Verificar que los diagramas, manuales y dependencias sigan vigentes.

Herramientas y formatos más usados

- **Markdown (.md):** Muy utilizado en repositorios GitHub o GitLab.
- **PDF / DOCX:** Para documentación formal o académica.
- **Wikis colaborativas:** Confluence, Notion o GitHub Wiki.
- **Swagger / Postman:** Para documentación de APIs.
- **Plantillas UML:** Generadas con Visual Paradigm, Lucidchart o Draw.io.
- **Generadores automáticos:** JSDoc (JavaScript), JavaDoc (Java), Doxygen (C++), etc.

Ventajas y desventajas de la documentación técnica

Ventajas

- Facilita la **transferencia de conocimiento** entre equipos.
- Permite una **mayor trazabilidad y control de cambios**.
- Mejora la **calidad del software y la comunicación** interna.
- Reduce los **tiempos de mantenimiento** y la dependencia de individuos clave.
- Sirve como **referencia para auditorías o certificaciones de calidad** (por ejemplo, ISO 9001 o CMMI).

Desventajas

- Requiere **tiempo y esfuerzo adicional**.
- Puede **quedar obsoleta rápidamente** si no se actualiza.

- En proyectos pequeños, algunos equipos la consideran una **tarea burocrática innecesaria**.
- Una mala documentación puede **confundir más que ayudar**, si está mal redactada o incompleta.

Aspecto	Documentación Técnica	Documentación Funcional
Orientación	Interna (para desarrolladores y técnicos)	Externa (para usuarios o clientes)
Contenido	Código, arquitectura, base de datos, APIs	Requisitos, casos de uso, objetivos del sistema
Nivel de detalle	Alto, con lenguaje técnico	Medio o bajo, con lenguaje natural
Propósito	Implementar, mantener y escalar el sistema	Entender qué hace el sistema y cómo lo usa el usuario

Conclusión

La documentación técnica no es un accesorio opcional del desarrollo, sino un **componente esencial de la calidad del software**.
En un entorno donde los equipos son cada vez más grandes, distribuidos y con alta rotación, mantener una documentación actualizada es sinónimo de **eficiencia, transparencia y profesionalismo**.

Un proyecto sin documentación puede funcionar a corto plazo, pero carece de **sostenibilidad y escalabilidad** en el tiempo. Por ello, todo desarrollador o líder técnico debe considerar la documentación como una **inversión estratégica**, no como una carga.