1. Ciclo de vida del desarrollo de software (SDLC)

El **SDLC** (Software Development Life Cycle) describe las etapas que se siguen para planificar, construir, probar y mantener un software.
Las fases más comunes son:

1. **Planificación**
    - Se define el objetivo, alcance y recursos del proyecto.
    - Identificación de requisitos iniciales.
    - Ejemplo: decidir si la app será web o móvil, estimar tiempos y presupuesto.
2. **Análisis de requisitos**
    - Se recopilan necesidades del cliente y usuario final.
    - Se documenta qué debe hacer el software (funcional y no funcional).
    - Ejemplo: “El sistema debe permitir registrar usuarios con email y contraseña”.
3. **Diseño**
    - Arquitectura del sistema (estructura de bases de datos, APIs, interfaces).
    - Diseño de UI/UX y diagramas de flujo.
    - Ejemplo: diseñar en Figma cómo se verá el login y cómo interactúa con el backend.
4. **Desarrollo / Implementación**
    - Los programadores escriben el código según el diseño aprobado.
    - Se integran módulos y se aplican buenas prácticas.
    - Ejemplo: programar el backend en Django y el frontend en React.
5. **Pruebas**
    - Validar que el software funcione según lo esperado.
    - Pruebas unitarias, de integración, de usuario, de rendimiento.
    - Ejemplo: probar que el formulario de registro no acepte emails inválidos.
6. **Despliegue**
    - Publicar el software en el entorno de producción para que los usuarios lo usen.
    - Configuración de servidores, hosting, seguridad y monitoreo.
7. **Mantenimiento**

- Corregir errores, aplicar actualizaciones y agregar nuevas funciones.
- Ejemplo: lanzar una versión 2.0 con mejoras de seguridad y nuevas páginas.

📌 Importante: No siempre es lineal. En metodologías ágiles como **Scrum**, estas fases son cíclicas y se repiten en iteraciones cortas.

2️⃣ ¿Qué es DevOps? (Desde el punto de vista del desarrollo)

DevOps es una cultura y conjunto de prácticas que **integran el desarrollo de software (Dev)** y las operaciones de TI (Ops)**, para entregar software más rápido, seguro y con mejor calidad.

🔷 **Desde el punto de vista del desarrollo**:

- Un desarrollador en un equipo DevOps no solo escribe código, también piensa en **cómo se va a desplegar, probar, monitorear y escalar**.
- DevOps rompe la barrera tradicional donde “los devs programan” y “los ops despliegan”.
- Implica trabajar con **automatización, integración continua (CI)** y **entrega continua (CD)**.

🔷 **Objetivos principales**:

- Reducir el tiempo entre escribir el código y que esté disponible para el usuario.
- Mejorar la calidad mediante pruebas automatizadas.
- Automatizar tareas repetitivas (despliegues, configuración de servidores).

🔷 **Herramientas típicas**:

- **Git/GitHub/GitLab** → control de versiones.
- **Jenkins, GitHub Actions, GitLab CI** → integración y entrega continua (CI/CD).
- **Docker, Kubernetes** → contenedores y orquestación.
- **Terraform, Ansible** → infraestructura como código.
- **Prometheus, Grafana** → monitoreo.

🔷 **Ejemplo en práctica**:

1. El desarrollador hace un commit en GitHub.
2. Un pipeline CI ejecuta pruebas automatizadas.

3. Si todo pasa, el CD despliega automáticamente en el servidor (Heroku, AWS, etc.).
4. El equipo de operaciones monitorea métricas y logs, y los desarrolladores reciben feedback rápido.

En resumen: **DevOps no es un rol fijo, es una filosofía y conjunto de prácticas** para que el desarrollo y la operación sean un flujo continuo y eficiente.