Ingeniería del Software II

TP 2 - Diagramas de Secuencia

Docentes a cargo:
Victor Valotto, Cielo Godoy

Alumnas:
Inderkumer, Priscila
Murzi, Ana Sol

29 - SEP - 2025

1. Objetivo

A partir de los casos de uso que cada grupo definió en Ingeniería de Software I, y considerando las decisiones de arquitectura de alto nivel vistas hasta ahora, generar diagramas de secuencia que representen la colaboración entre objetos del sistema. El objetivo es ejercitar cómo se distribuyen las responsabilidades entre clases y componentes, y cómo se traduce el comportamiento dinámico del sistema.

2. Actividades

A- Repaso teórico

- **¿Qué información brinda un diagrama de secuencia?**
  Un diagrama de secuencia representa el comportamiento de los objetos dentro de un segmento específico de ejecución del software
  Es útil para visualizar y entender la secuencia de llamadas entre diferentes objetos y cómo colaboran entre sí para llevar a cabo una funcionalidad específica
- **¿Qué elementos lo componen y qué significa cada uno de ellos?**
  Consta de objetos que se representan del modo usual, rectángulos con nombre, mensajes representados por líneas continuas con una punta de flecha y el tiempo representado como una progresión vertical
  A medida que se avanza hacia abajo en el diagrama, se representa el flujo temporal de los eventos, mostrando el orden en que ocurren las interacciones
  Cada elemento en el diagrama de secuencia está conectado mediante líneas verticales (líneas de vida) que indican la existencia de un objeto a lo largo del tiempo
    - Objetos
      Se colocan cerca de la parte superior del diagrama de izquierda a derecha y se acomodan de manera que simplifiquen al diagrama
    - Mensaje
      Puede ser:
        - Simple: es la transferencia del control de un objeto a otro
        - Sincrónico: si un objeto envía un mensaje sincrónico, esperará la respuesta a tal mensaje antes de continuar con su trabajo
        - Asincrónico: si un objeto envía un mensaje asincrónico, no esperará una respuesta antes de continuar
    - Tiempo
      Se representa en dirección vertical
      Se inicia en la parte superior y avanza hacia la parte inferior
      Un mensaje que esté más cerca de la parte superior ocurrirá antes que uno que esté cerca de la parte inferior
    - Líneas de vida
      Indican la existencia de un objeto a lo largo del tiempo
      Las flechas entre las líneas de vida representan los mensajes que se envían entre los objetos

- **¿Qué relación existe entre los diagramas de secuencia y los casos de uso?**
  Está estrechamente relacionado a los casos de uso, proporcionando una visión detallada de la implementación de estos

- **¿Se debe realizar un diagrama de secuencia por cada caso de uso?**
  No, se recomienda elaborarlo cuando un caso de uso es crítico, complejo o involucra muchos objetos. Para casos de uso simples es suficiente con documentarlos en un diagrama de casos de uso.

- **¿Qué diferencia hay entre un diagrama de secuencia de instancia y uno genérico?**

Diagrama de secuencia de instancia	Diagrama de secuencia genérico
• Muestra un escenario concreto con instancias reales de objetos en un momento determinado • Es útil para ilustrar cómo interactúan ciertos objetos en una ejecución particular	• Representa la lógica general de interacción entre clases o tipos de objetos, sin centrarse en casos concretos • Sirve como plantilla para múltiples ejecuciones posibles, ya que es más abstracto y reusable

- **¿De qué manera se relacionan los diagramas de secuencia y la arquitectura?**
  Los diagramas de secuencia se relacionan con la arquitectura porque:
  - Muestra cómo las capas de arquitectura interactúan dinámicamente para cumplir con un caso de uso
  - Validan decisiones de diseño al comprobar que las interacciones cumplen con los requisitos definidos en la arquitectura
  - Sirven de puente entre la arquitectura estática y el comportamiento dinámico del sistema
  - Facilitan identificar responsabilidades y dependencias entre los módulos de la arquitectura, asegurando que el diseño sea coherente con la estructura

B. Selección de casos de uso

- Elegir dos casos de uso relevantes de su proyecto (los más críticos o los que involucren mayor interacción entre componentes).

  UC22- Generar alerta de mascota perdida
  UC23 - Consultar ubicación a sistema de geolocalización

C. Diagramación

- Generar el diagrama de secuencia para cada escenario principal de los dos casos de uso seleccionados.

  DiagramaSecuencia-UC22
  DiagramaSecuencia-UC23

- Identificar qué objetos (clases instanciadas) participan y cómo colaboran.

  Objetos participantes (instancias de clases) de UC22:

  - **Dueño →** instancia de la clase `Dueño` (actor externo, usuario del sistema).
  - **UI (InterfazWeb) →** instancia de `InterfazWeb`, objeto *boundary* que recibe la interacción del usuario.
  - **MC (MascotaController) →** instancia de `MascotaController`, *controller* que coordina la lógica del caso de uso.

  - **MS (MascotaService) →** instancia de `MascotaService`, gestiona operaciones de negocio sobre mascotas.
  - **AS (AlertaService) →** instancia de `AlertaService`, gestiona la creación y persistencia de alertas.
  - **NS (NotificacionService) →** instancia de `NotificacionService`, administra el envío de notificaciones a destinatarios.
  - **MR (MascotaRepository) →** instancia de `MascotaRepository`, acceso a datos de mascotas en la base.
  - **AR (AlertaRepository) →** instancia de `AlertaRepository`, acceso a datos de alertas en la base.
  - **DB (BaseDatos) →** instancia de `BaseDatos`, entidad persistente.
  - **SN (ServicioNotificacion) →** instancia de un servicio externo para enviar notificaciones (SMS, email, push, etc.).

  Colaboración entre objetos de UC22:

  1. **Dueño** reporta la mascota como perdida a través de la **UI**.
  2. La **UI** comunica la petición al **MascotaController (MC)**.
  3. **MC** delega a **MascotaService (MS)** que:
     - Consulta a **MascotaRepository (MR)** para obtener los datos de la mascota desde la **BaseDatos (DB)**.
     - Actualiza el estado de la mascota a `"PERDIDA"` en la base.
  4. Una vez actualizada la mascota, **MS** solicita a **AlertaService (AS)** crear la alerta.
  5. **AS** construye los datos de la alerta y los persiste mediante **AlertaRepository (AR)** en la **BaseDatos (DB)**.
  6. **AS** solicita a **NotificacionService (NS)** que envíe notificaciones.
     - **NS** determina los destinatarios de la alerta.
     - Por cada destinatario, usa el **ServicioNotificacion (SN)** externo para enviar el mensaje.
     - Se gestiona la respuesta: si falla, el sistema reintenta.
  7. **NS** confirma a **AS** las notificaciones enviadas.
  8. **AS** responde al **MC** con la alerta creada.
  9. **MC** devuelve a la **UI** el resultado (alerta creada o error).
  10. Finalmente, la **UI** muestra al **Dueño** un mensaje de confirmación o de error.

Objetos participantes (instancias de clases) de UC23:

- **Dueño ➜** instancia de la clase `Dueño` (actor externo, usuario del sistema).
- **UI (InterfazWeb) ➜** instancia de la clase `InterfazWeb`, capa de presentación (boundary).

- **MC (MascotaController) ➜** instancia de la clase `MascotaController`, controlador de la lógica de aplicación.
- **MS (MascotaService) ➜** instancia de la clase `MascotaService`, lógica de negocio relacionada con mascotas.
- **GS (GeolocationService) ➜** instancia de la clase `GeolocationService`, lógica de negocio para servicios de geolocalización.
- **MR (MascotaRepository) ➜** instancia de la clase `MascotaRepository`, maneja la persistencia de la información de mascotas.
- **DB (BaseDatos) ➜** entidad de almacenamiento, instancia de la clase `BaseDatos`.
- **API (APIGeolocalización) ➜** instancia de la clase que representa el servicio externo de geolocalización.
- **Disp (DispositivoGPS) ➜** instancia de la clase `DispositivoGPS`, componente externo que provee datos de ubicación.

Colaboración entre objetos de UC23:

1. **Dueño** interactúa con la **UI** para reportar una mascota como perdida.
2. La **UI** delega en el **MascotaController** (MC) la solicitud `reportarPerdida(mascotaId)`.
3. **MC** usa el **MascotaService** (MS) para procesar el reporte.
4. **MS** consulta al **MascotaRepository** (MR) para recuperar los datos de la mascota desde la **BaseDatos** (DB).
5. Si la mascota tiene geolocalización habilitada, **MS** invoca al **GeolocationService** (GS).
6. **GS** se comunica con la **APIGeolocalización** (API), que a su vez solicita la ubicación al **DispositivoGPS** (Disp).
    - Si el dispositivo responde con coordenadas ➜ estas se validan en **GS** y se devuelven a **MS**, que las guarda vía **MR** en la **BaseDatos**.
    - Si el dispositivo no tiene señal ➜ se propaga un error hacia arriba hasta la **UI**.
7. Finalmente, la **UI** muestra al **Dueño** ya sea el mapa con la ubicación de la mascota o un mensaje de error.

- Reflejar en los diagramas las decisiones de diseño tomadas en la arquitectura (ejemplo: capas, servicios, repositorios, controladores).

D. Análisis final

- Explicar en un breve texto cómo los diagramas de secuencia obtenidos ayudan a entender la distribución de responsabilidades y cómo se relacionan con la arquitectura de su proyecto.

Los diagramas de secuencia elaborados resultan fundamentales para comprender la **distribución de responsabilidades** en el sistema, ya que muestran de forma explícita cómo se reparten las tareas entre controladores, servicios, repositorios, entidades y servicios externos.
En cada interacción de puede observar qué capa o componente es responsable de orquestar el flujo (controller), aplicar la lógica de negocio (service), acceder a la persistencia (repository/base de datos) o integrarse con terceros (APIs externas).

Además, estos diagramas se relacionan directamente con la **arquitectura del proyecto** documentada mediante el modelo C4 del TP1-B:

- En el nivel de contenedores, permiten ver cómo el frontend, el backend y los servicios externos se comunican.
- En el nivel de componentes, refuerzan la separación en capas típica de la arquitectura en capas/MVC adoptada: *boundary* ➜ *controller* ➜ *service* ➜ *repository* ➜ *entity*.
- A su vez, permiten validar atributos de calidad priorizados como confiabilidad y recuperación, al mostrar cómo se manejan fallos (ej. API sin respuesta, dispositivo sin señal) sin comprometer la lógica principal.

3. Entregables

Un documento PDF que contenga:

- Respuestas al repaso teórico (punto 1).
- Los dos diagramas de secuencia elaborados (punto 3).
- Un análisis final reflexivo (punto 4).