Ingeniería del Software

Práctica 2

Diseño del sistema

2º Grado en Ingeniería Informática
Universidad de Córdoba
Curso 2024/25

1. Organización de la práctica

2. Diagrama de clases

3. Matrices de trazabilidad

4. Diagrama de secuencia

5. Pasos a seguir

Sesiones

- Sesión 1. Diagrama de clases.
  - Creación diagrama de clases UML.
  - Refinamiento de requisitos.
- Sesión 2. Diagramas de secuencia.
  - Creación diagramas de secuencia UML.
  - Elaboración, como mínimo, de tantos diagramas de secuencia como miembros del grupo.

Entregables

- Documento formal ampliado con:
  - Diagrama de clases
  - Matrices de trazabilidad
    - Requisitos *vs* casos de uso (después de los casos de uso).
    - Casos de uso *vs* diagrama de clases (después del diagrama de clases).
  - Diagramas de secuencia
- YouTrack/Trello:
  - Planificación y reuniones

Grupo	GM2	GM4	GM3	GM1
Fecha entrega	18/11	19/11	19/11	20/11

UML

- **U**nified **M**odeling **L**anguage

- Motivación: combinar y estandarizar una notación para describir sistemas orientados a objetos a partir de los lenguajes de modelado más conocidos:

  Booch – OOD Rumbaugh – OMT Jacobson – OOSE y Objectory

- UML es un **lenguaje gráfico** para visualizar, especificar, construir y documentar un sistema software desde distintos puntos de vista.

- UML permite describir los aspectos **estáticos** (estructura) y **dinámicos** (comportamiento) de un sistema

1. Organización de la práctica

2. Diagrama de clases

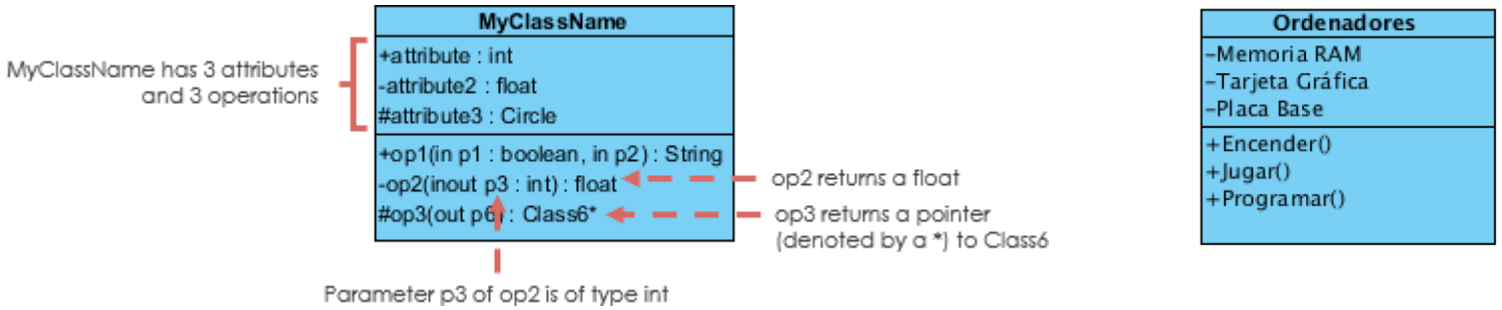
3. Matrices de trazabilidad

4. Diagrama de secuencia

5. Pasos a seguir

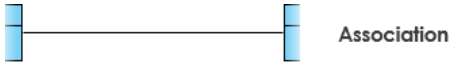
Clases

- Notación gráfica para representar estructuras de información estática.
- Una clase es una categoría o grupo de cosas que tienen unos atributos y realizan unas acciones.
- El diagrama muestra las clases, sistema, atributos, métodos, y relaciones entre ellos.

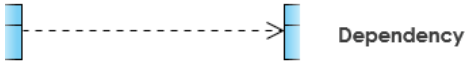


Relaciones entre clases

- **Asociación**. Conexión funcional (y de comunicación) entre clases. Suele indicar la cardinalidad.
  - Las clases “Biblioteca” y “Libro” tienen una relación de asociación *1..n*

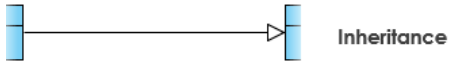


- **Dependencia**. Tipo de asociación donde una clase utiliza otra, y la definición de una clase se ve influenciada por cambios en la otra.
  - Las clases “Conductor” y “Coche”.

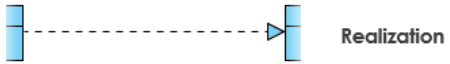


Relaciones entre clases

- **Herencia**. Representan una relación entre una clase “padre” y una o varias clases “hijas”.
  - Las clases “Estudiante” y “Profesor” heredan de una clase “Persona” más general.

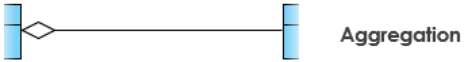


- **Realización**. Indica cómo se debe realizar una implementación de una interfaz.
  - La clase abstracta “Pago” y la clase “Pago_Efectivo”.

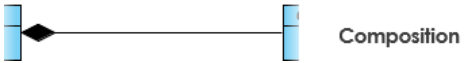


Relaciones entre clases

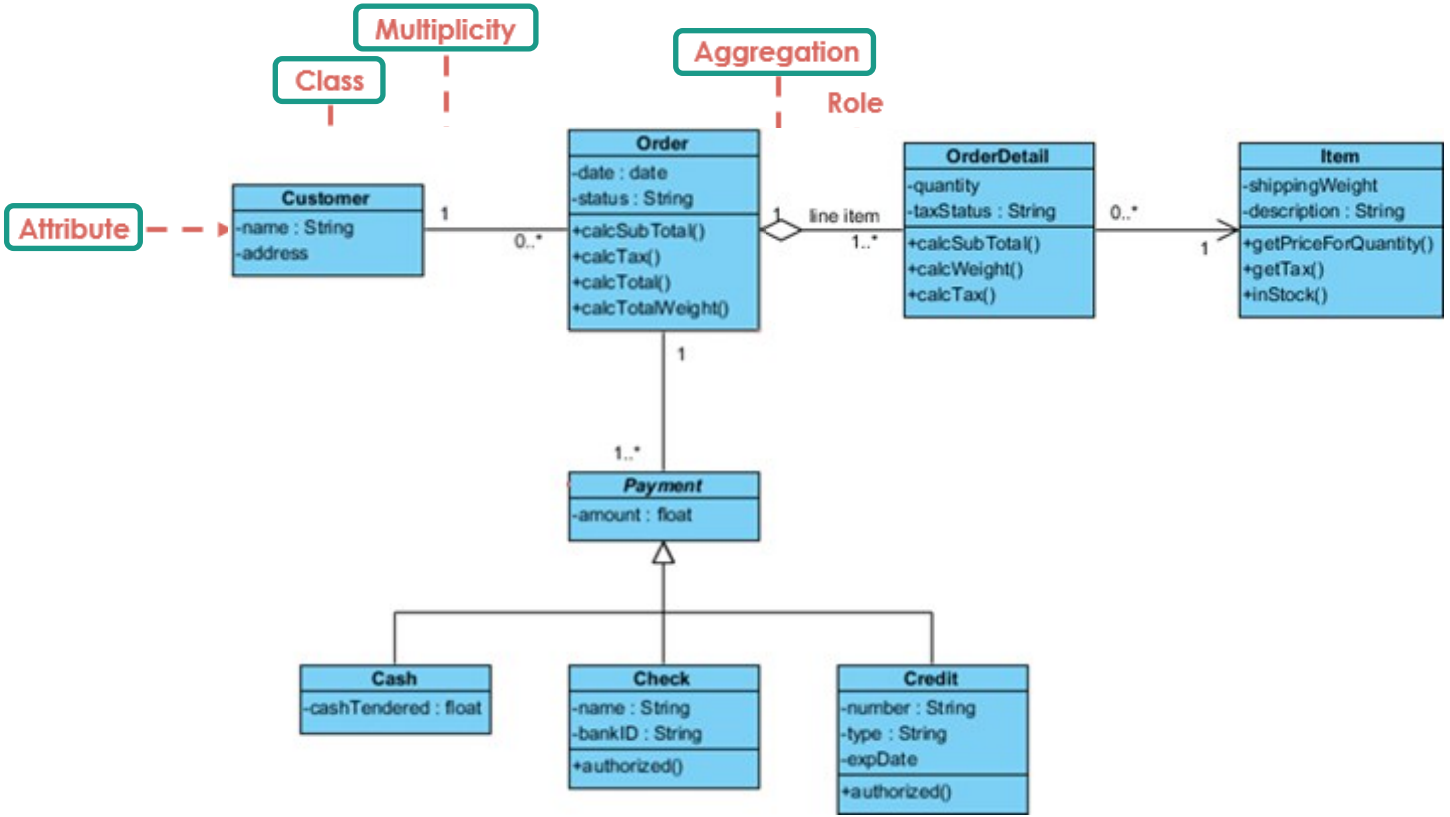
- **Agregación**. Indica que una clase es parte de otra, pero tienen diferentes líneas de vida.
  - La clase “Universidad” y la clase “Estudiante”.



- **Composición**. Agregación donde la clase que forma parte de la otra no puede existir por sí misma.
  - La clase “Serie” y la clase “Capítulo”.



Relaciones entre clases



1. Organización de la práctica

2. Diagrama de clases

3. Matrices de trazabilidad

4. Diagrama de secuencia

5. Pasos a seguir

Introducción

- Técnica de validación.
- Permite comprobar que los elementos diseñados dan respuesta a las necesidades expresadas en los requisitos.
- Ayuda a confirmar que todos los requisitos están cubiertos en el diseño.

	CU1	CU2	CU3	CU4	CU5
RF1	✓		✓		
RF2		✓			
RF3				✓	
RF4		✓		✓	
RF5	✓				
RF6					✓
RF7					✓
RF8		✓			

Tipos de matrices

- Requisitos funcionales frente a casos de uso.
  - Cada requisito debe estar cubierto, como mínimo, por un caso de uso.
  - Todo caso de uso debe dar respuesta a uno o más requisitos.
  - Recomendación: realizarla al comienzo de la práctica, para así poder refinar los casos de uso en caso de ser necesario.
- Casos de uso frente a clases.
  - Cada clase debe tener correspondencia con uno o varios casos de uso.
  - Todo caso de uso debe tener al menos una clase asociada.
  - Recomendación: realizarla tras terminar el diagrama de clases.

1. Organización de la práctica

2. Diagrama de clases

3. Matrices de trazabilidad

4. Diagrama de secuencia

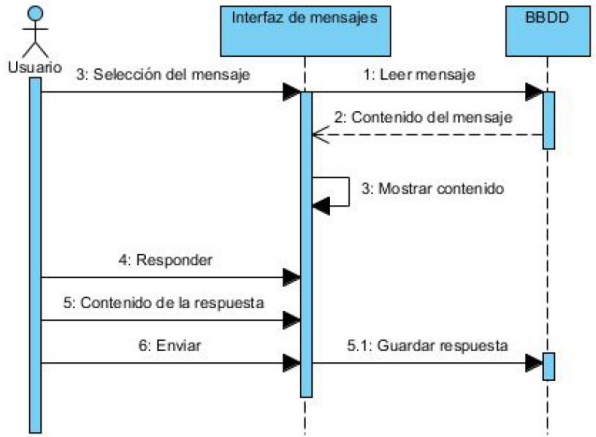
5. Pasos a seguir

Introducción

- Diagrama de interacción que nos permite modelar el comportamiento del sistema.
- Especifica colaboración entre objetos, o entre usuario y sistema.
- Operación a un nivel de abstracción elevado.

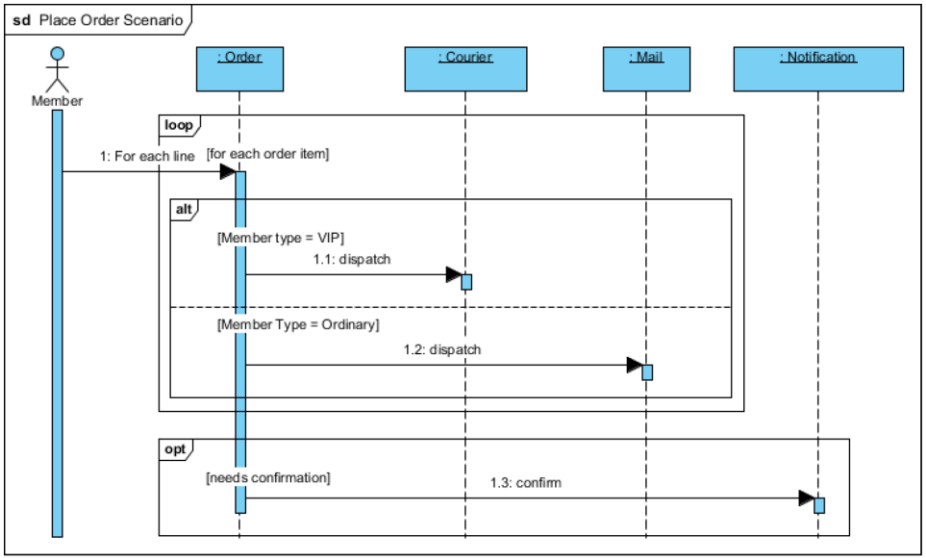
Elementos

- **Actor**. Agente externo que participa en la interacción.
- **Línea de vida**. Indica la presencia (o no) del objeto. El periodo durante el cual el objeto está activo se denomina “activación”.
- **Mensaje**. Comunicación entre dos líneas de vida. Deben ir numerados.

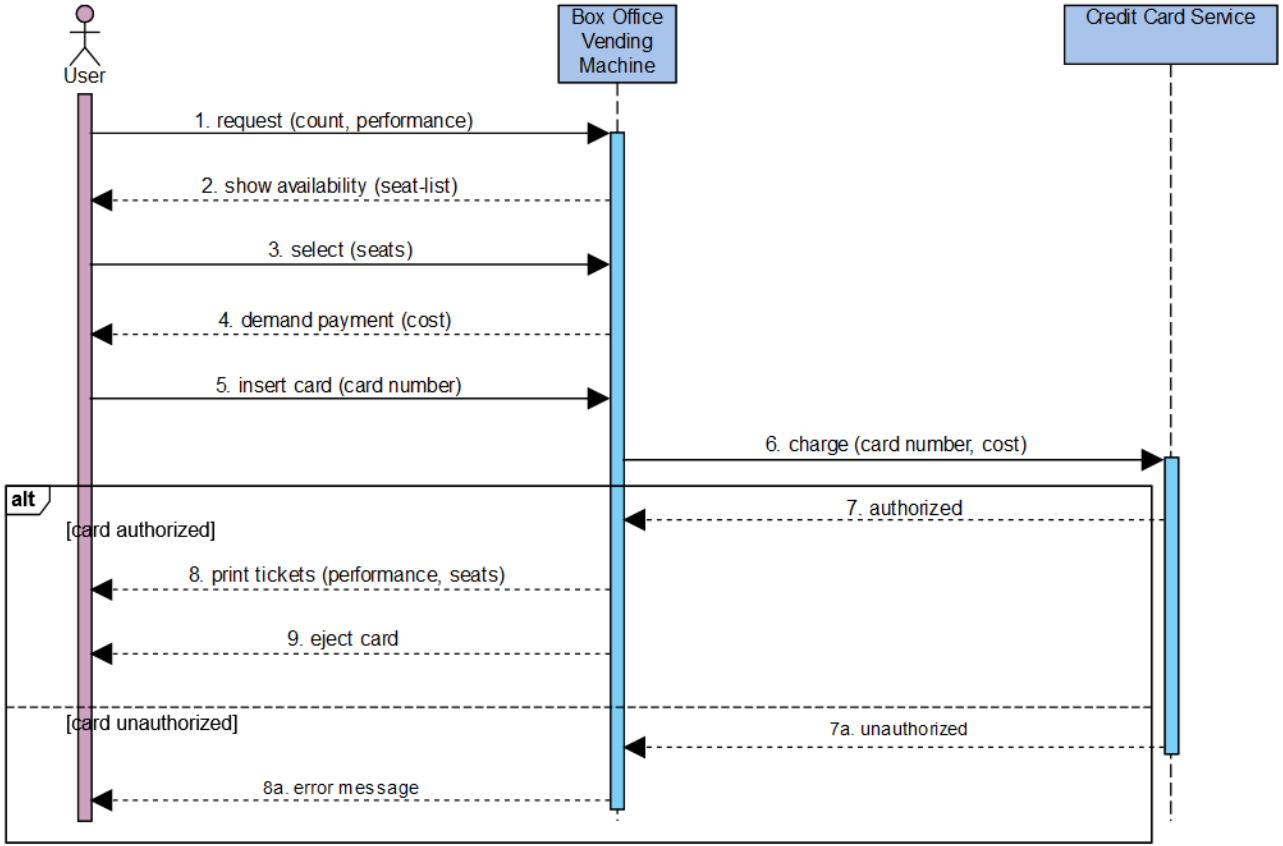


Elementos – bloques

- **Loop**. Fragmento que se ejecuta varias veces.
- **Alt**. Secuencias alternativas, se ejecuta la que cumple la condición.
- **Opt**. Fragmento opcional que se ejecuta solo si se cumple la condición.



Ejemplo


User
Box Office Vending Machine
Credit Card Service
1. request (count, performance)
2. show availability (seat-list)
3. select (seats)
4. demand payment (cost)
5. insert card (card number)
6. charge (card number, cost)
alt
[card authorized]
7. authorized
8. print tickets (performance, seats)
9. eject card
[card unauthorized]
7a. unauthorized
8a. error message

1. Organización de la práctica

2. Diagrama de clases

3. Matrices de trazabilidad

4. Diagrama de secuencia

5. Pasos a seguir

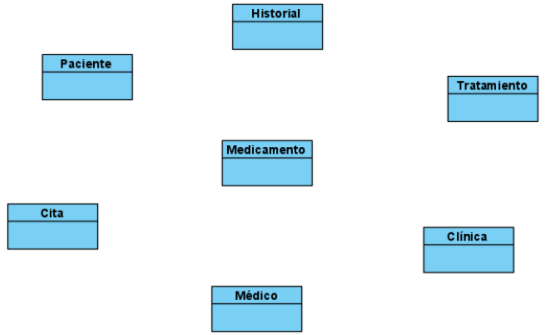
Validación (I)

1. Validación de casos de uso frente a requisitos funcionales

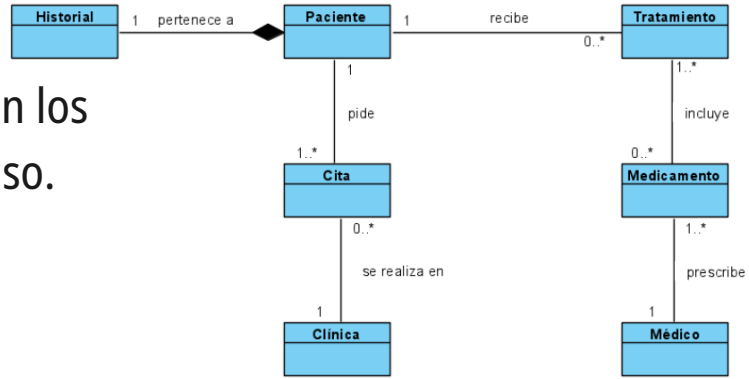
	CU1 Pedir cita	**CU2** Cambiar cita	**CU3** Buscar fecha	**CU4** Cambiar médico	**…**
RF1	X				
RF2	X	X	X		
RF3		X		X	
…					

Diseño de clases

1. Identificar los conceptos principales
   - Por lo general, cada uno será modelado como una clase.

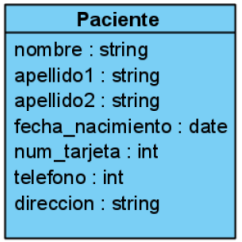


2. Identificar relaciones entre clases
   - Deben ser consistentes con lo establecido en los requisitos y los escenarios de los casos de uso.

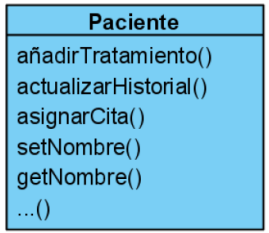


Diseño de clases

3. Definir atributos para cada clase
   - Nacen principalmente de los requisitos de información.
   - Pueden aparecer como parte de la descripción de la funcionalidad.
   - En una fase temprana de diseño, puede que solo tengamos claro el tipo, y no otros aspectos. Se puede refinar más adelante.



4. Definir operaciones de cada clase
   - Inspiradas en los escenarios de los casos de uso.
   - Qué clase se responsabiliza de las acciones que realiza el sistema.
   - Al avanzar, se irá completando la signatura de los métodos.



Diseño de clases

5. Completar la especificación de las clases
   - En forma de tabla.

Clase	Paciente		
Esta clase almacena la información personal del paciente y da acceso a su historial médico, además de permitir comprobar sus citas médicas y tratamientos activos.			
Atributos			
-	nombre	String	Nombre del paciente
-	num_tarjeta	int	Número identificativo de tarjeta sanitaria
...	...	...	...
Operaciones			
+	añadirTratamiento	void	Añade un tratamiento al historial del paciente
	fecha_inicio	Date	Fecha de inicio del tratamiento (DD/MM/AAAA)
...	...	...	...

Validación (II)

1. Validación de clases frente a casos de uso

	Paciente	Cita	Clínica	Médico	...
CU1: Pedir cita	X	X	X		
CU2: Pedir cita	X	X			
CU3: Pedir cita		X	X		
CU4: Cambiar médico		X		X	
...					

Diagrama de secuencia

1. Identificar quiénes colaboran
   - Normalmente el actor inicia la secuencia.
   - A continuación, conjunto de clases responsables de realizar las acciones que le corresponden al sistema.

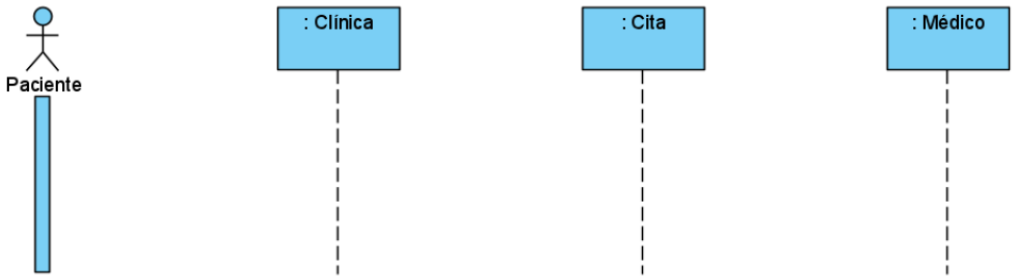
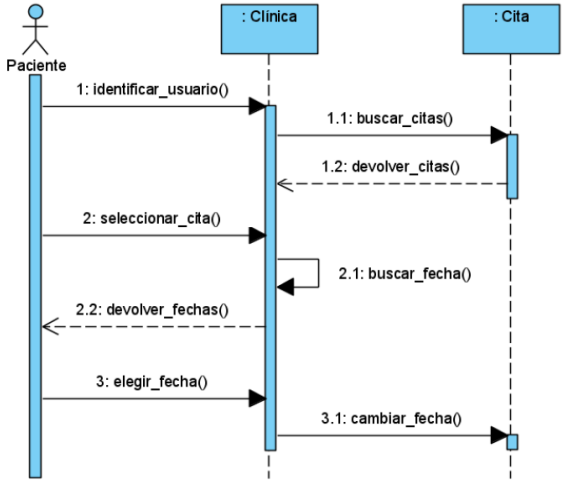


Diagrama de secuencia

2. Extraer los mensajes del escenario principal
   - Establecer quién inicia el mensaje, quién lo recibe, y en qué orden sucede.
   - El escenario principal define el orden temporal en que suceden las acciones principales.



   - Identificar las primeras secuencias de control.

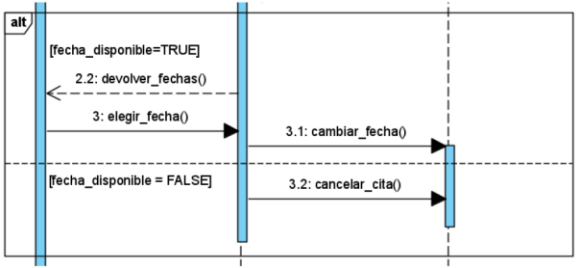


Diagrama de secuencia

3. Estudiar los escenarios alternativos
   - Pueden requerir nuevos bloques de alternativa.
   - Pueden implicar mensajes de retorno al actor para concluir.

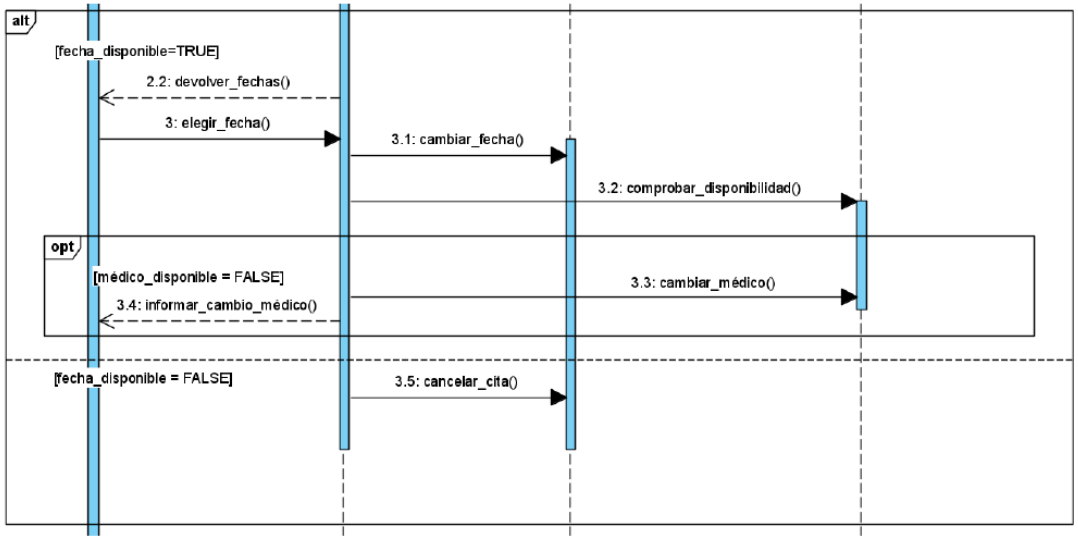
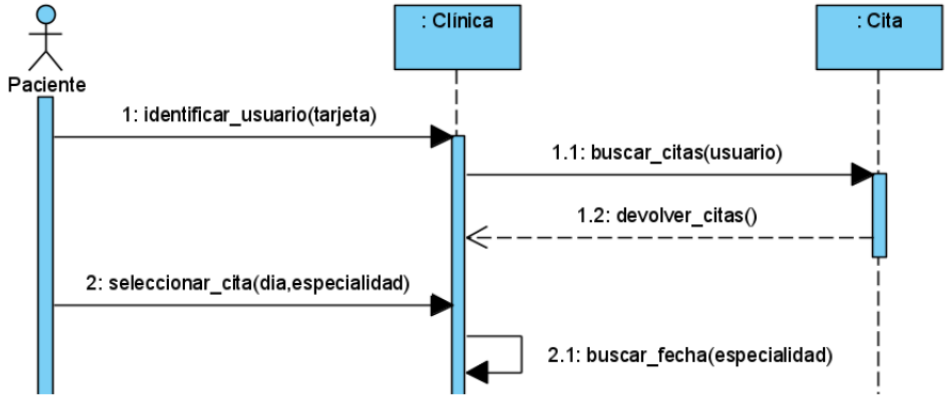


Diagrama de secuencia

4. Añadir los parámetros
   - Especificar los principales parámetros que se necesitan intercambiar.
   - Pueden ser entradas que aporta el actor, o creadas por las clases.
   - No se describen operaciones del código. Algunos parámetros podrían no conocerse aún.



1. Organización de la práctica
2. Diagrama de clases
3. Matrices de trazabilidad
4. Diagrama de secuencia
5. Pasos a seguir

Entregables

- Documento formal ampliado con:
  - Diagrama de clases
  - Matrices de trazabilidad
    - ❖ Requisitos *vs* casos de uso (después de los casos de uso).
    - ❖ Casos de uso *vs* diagrama de clases (después del diagrama de clases).
  - Diagramas de secuencia
- YouTrack/Trello:
  - Planificación y reuniones

Grupoº	GM2	GM4	GM3	GM1
Fecha entrega	18/11	19/11	19/11	20/11

Ingeniería del Software

Práctica 2

Diseño del sistema

2º Grado en Ingeniería Informática
Universidad de Córdoba
Curso 2024/25