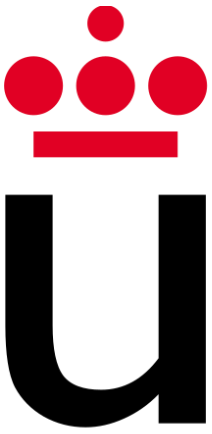


Universidad
Rey Juan Carlos

MODELADO DEL SOFTWARE
EJERCICIOS & PRÁCTICAS

GRADO EN MATEMÁTICAS
CURSO:2024/2025

Cristian Gómez Macías

TABLA DE CONTENIDO

EJERCICIOS TEORÍA

Tema 1: PREGUNTAS TEORÍA

1. ¿Qué es el software?
   - Conjunto de instrucciones, procedimientos definidos y documentación técnica necesaria para el desarrollo y ejecución de programas, que pueden ser diseñados tanto para un cliente específico como para un mercado general
   - Conjunto de componentes físicos y electrónicos que permiten la interacción entre el usuario y los sistemas de computación, como procesadores, memorias y periféricos
   - Colección de normas y especificaciones técnicas utilizadas exclusivamente para la fabricación y ensamblaje de dispositivos de hardware en sistemas de computación
   - Infraestructura de telecomunicaciones que permite la interconexión entre computadoras y otros dispositivos, facilitando la transmisión de datos
2. ¿Qué características presenta el software frente al hardware?
   - Tangible
   - Dependencia al hardware
   - Alta flexibilidad
   - Fallos por desgaste
3. ¿Qué es el mantenimiento perfectivo?
   - Ajuste del software a un nuevo entorno
   - Prevención de errores futuros
   - Corrección de fallos emergentes
   - Inclusión de nuevos requisitos y características
4. ¿La versión gold de un software es una versión limitada de este?
   - Verdadero
   - Falso

5. ¿Cuáles de las siguientes afirmaciones sobre qué es la Ingeniería del Software son correctas?

   - La Ingeniería del Software se refiere al conjunto de reglas y pensamiento estructurado que uno aplica en el entorno para producir sistemas o componentes de software
   - La Ingeniería del Software es la aplicación de los principios de la ciencia de la computación y las matemáticas para lograr soluciones efectivas para problemas presentados en el software
   - La Ingeniería del Software se centra en la creación de software de alta calidad desde un enfoque sistemático, eficiente y controlado
   - La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, funcionamiento y mantenimiento del software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software
   - La Ingeniería del Software es la disciplina de ingeniería encargada de todos los aspectos relacionados con la producción de software desde sus etapas más tempranas de la especificación del sistema hasta el mantenimiento del sistema tras su puesta en marcha

6. ¿Qué 3 aspectos son claves en el software?

   - Calidad
   - Seguridad
   - Comportamiento
   - Simulación
   - Persistencia
   - Proceso de desarrollo
   - Conectividad

7. ¿Qué tipos de mantenimiento software existen?

   - Preventivo
   - Adaptativo
   - Perfectivo
   - Emergencia
   - Paulativo

- Distribuido
- Conectivo
- Correctivo

8. ¿Qué paradigma hace uso de reglas lógicas para resolver un problema?
   - Orientado a objetos
   - Imperativo
   - Declarativo
9. ¿Cuál es el paradigma en el que cobra una especial relevancia el orden de secuencia de las instrucciones para el control del flujo de programa?
   - Orientado a objetos
   - Imperativo
   - Declarativo
10. En el paradigma declarativo lo principal es cómo resolver un problema, no qué se debe conseguir:
    - Verdadero
    - Falso
11. ¿Cuáles son algunos de los criterios para medir la calidad del software?
    - Precisión
    - Fiabilidad
    - Conectividad
    - Legalidad
    - Mantenibilidad
12. La usabilidad es la facilidad con la que los programadores pueden realizar correcciones
    - Verdadero
    - Falso

Tema 2: PREGUNTAS TEORÍA

1. ¿Las historias de usuario son?
   - Conjunto de tareas que se agrupan para realizar en la próxima iteracción siguiendo el modelo Scrum.
   - Conjunto de documentación técnica sobre el software desarrollado para el cliente.
   - Son descripciones de las funcionalidades a implementar del sistema.
   - Conjunto de estimaciones ofrecidas por los miembros del equipo sobre las tareas a realizar.
2. ¿El product owner en Scrum es el encargado de supervisar que el equipo sigue las reglas bajo este marco?
   - Verdadero
   - Falso
3. ¿Cuáles de las siguientes características no se corresponden con el de desarrollo software mediante el modelo incremental?
   - Documentación disponible en todo momento
   - Cada ciclo produce un incremento
   - Fácil planificación
   - Se adapta bien a entornos de alta incertidumbre
   - Proceso de desarrollo no lineal por etapas
4. ¿Las modelos de desarrollo software tradicional siguen un proceso lineal en el que no se pasa a la siguiente etapa sin acabar la actual?
   - Verdadero
   - Falso
5. ¿Cuál es el flujo del modelo de desarrollo en cascada?
   - Análisis, Evaluación, Implementación, Despliegue, Diseño
   - Diseño, Análisis, Implementación, Despliegue, Evaluación
   - Análisis, Implementación, Diseño, Despliegue, Evaluación

- Diseño, Implementación, Despliegue, Evaluación, Análisis
- Análisis, Diseño, Implementación, Evaluación, Despliegue
- Diseño, Análisis, Evaluación, Implementación, Despliegue
- Análisis, Implementación, Evaluación, Diseño, Despliegue

6. Algunas de las características de las metodologías ágiles frente a los modelos de desarrollo software tradicional son:
   - Requisitos variables
   - Equipo unidisciplinar
   - Cliente involucrado solo en el final del proceso
   - Entregas periódicas
   - Documentación exhaustiva
   - Flexible a cambios
7. La lista de tareas a realizar en la próxima iteracción en Scrum se denomina:
   - Sprint
   - Sprint backlog
   - Sprint meeting
   - Sprint review meeting
8. El planning póker se utiliza como medio de estimación de las historias de usuario
   - Verdadero
   - Falso
9. En las metodologías ágiles el proceso y su estado es conocido por todo el equipo de trabajo
   - Verdadero
   - Falso
10. Consecuencias del uso del método Lean:
    - Entregas justo en tiempo
    - Eliminación de desperdicios
    - Eficiencia y calidad

- Ninguna respuesta es correcta

11. Valores de la metodología XP:
    - Comunicación
    - Distribución
    - Simplicidad
    - Polaridad
    - Valentía
    - Documentación continua
    - Respeto
    - Feedback
12. ¿Qué función tiene un tracker en la metodología XP?
    - Rastrea las estimaciones hechas por el equipo y da retroalimentación sobre su precisión para mejorar futuras estimaciones
    - Actúa como guía/asistente de los demás miembros del equipo en seguir el proceso XP
    - Guiar al equipo en la resolución de problemas específicos
    - Ayudan al cliente con las pruebas funcionales y la ejecución de estas
13. Si se parte del modelo basado en prototipos a la hora de construir software, se requiere que:
    - Todos los prototipos sean funcionales
    - El cliente participa en todo el proceso
    - No pretender cambios una vez que se presente el prototipo
    - Integrar todas las funcionalidades al principio
    - Ninguna respuesta es correcta
14. ¿Scrum hace uso de un tablero para organizar proyectos y flujos de trabajo?
    - Verdadero
    - Falso

15. ¿Pasa la documentación a un segundo plano si se hace uso de medios ágiles para el desarrollo de software?
    - Verdadero
    - Falso
16. ¿Existe un límite de tareas a realizar al mismo tiempo en Kanban?
    - Verdadero
    - Falso

Tema 3: PREGUNTAS TEORÍA

1. ¿Cuál es la fase más larga y compleja del Proceso Unificado de Desarrollo de Software?
    - Inicio
    - Elaboración
    - Construcción
    - Transición
2. ¿Cuáles de las siguientes actividades se llevan a cabo en la fase de transición del Proceso Unificado?
    - Migración de datos desde sistemas antiguos
    - Preparación del caso de negocio
    - Formación de los usuarios
    - Definición del diseño y arquitectura
3. La fase de inicio del Proceso Unificado es la más larga y compleja, ya que se concentra en la implementación del sistema.
    - Verdadero
    - Falso
4. ¿Qué aspecto es fundamental establecer durante la fase de elaboración?
    - La arquitectura y el diseño del sistema
    - La migración de datos
    - La entrega del sistema a los usuarios

- La formación de los usuarios finales

5. En el Proceso Unificado, las fases del ciclo de vida se repiten hasta que el sistema esté listo para ser lanzado a los usuarios.
    - Verdadero
    - Falso
6. ¿Cuáles de las siguientes actividades se realizan en la fase de elaboración del Proceso Unificado?
    - Definir los requisitos del sistema
    - Establecer la arquitectura del sistema
    - Implementar el código fuente
    - Validar los casos de uso
7. La fase de transición en el Proceso Unificado se enfoca únicamente en la implementación de código y no incluye actividades relacionadas con los usuarios.
    - Verdadero
    - Falso
8. ¿Qué se entrega al final de la fase de construcción en el Proceso Unificado?
    - Un sistema completo listo para la transición
    - El plan de proyecto y los costos estimados
    - Diagramas de casos de uso
    - El plan de migración de datos
9. En el Proceso Unificado, cada iteración se generan versiones incrementales del sistema. ¿Cuáles de los siguientes aspectos deben ser revisados o mejorados en cada iteración para garantizar el éxito de las fases posteriores?
    - Refinamiento de los requisitos del sistema utilizando diagramas de casos de uso
    - Finalización y perfeccionamiento del diseño del sistema en base a la arquitectura definida
    - Definición de los costos y estimaciones para la fase de transición
    - Migración de datos y formación de los usuarios

- Identificación y mitigación de riesgos a lo largo del proyecto

Tema 4: PREGUNTAS TEORÍA

1. ¿Qué es UML?
   - Es una notación para crear un estándar con el objetivo de representar un sistema.
   - Es un lenguaje de programación para especificar el comportamiento de un sistema y el paso de mensajes entre los diferentes componentes de este.
   - Es un tipo de diagrama con el objetivo de representar la estructura interna de un sistema.
2. ¿Cómo se dividen los diagramas según lo que pretenden modelar de un sistema?
   - Estática
   - Conexión
   - Comportamiento
   - Simulación
   - Persistencia
3. ¿Los casos de usos sirven para especificar los componentes internos de un sistema?
   - Verdadero
   - Falso
4. La línea de vida de un diagrama de secuencia simboliza:
   - Al usuario/objeto que envía el mensaje
   - El mensaje que se envía
   - El tiempo para llevar a cabo dicha acción
5. Un fragmento **alt** en un diagrama de secuencia permite especificar:
   - Una secuencia que se repite un número determinado de veces
   - Una secuencia para especificar llamadas recursivas
   - Una secuencia para especificar paralelismo entre flujos de mensajes

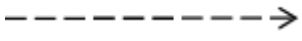
- Una secuencia if-else en el que en base al cumplimiento de una condición se ejecuta una secuencia u otra

6. La relación de herencia entre clases tiene asociada una multiplicidad:
   - Verdadero
   - Falso
7. Esta especificación de un nombre **Modelado del Software:Asignatura** es propia de:
   - Diagramas de objetos
   - Diagramas de componentes
   - Diagramas de estados
   - Diagramas de clases
   - Diagramas de paquetes
8. Las relaciones **extends** entre casos de usos indican que uno de ellos es requerido por el otro para llevarse a cabo:
   - Verdadero
   - Falso
9. Un modelo es:
   - Un conjunto de diagramas que representan el sistema en su totalidad.
   - Una representación abstracta de un sistema o proceso que omite detalles irrelevantes.
   - Un diagrama que muestra la relación entre clases en un sistema.
   - Un programa ejecutable que imita el comportamiento del sistema.
10. Los componentes se pueden agrupar dentro de un diagrama de paquetes:
    - Verdadero
    - Falso
11. El modelo de arquitectura propuesto por Krutchen consta de las vistas:
    - Lógica
    - Desarrollo

- Proceso
- Física
- Ninguna respuesta es correcta

12. En un diagrama de clases, la relación  indica:
    - Herencia
    - Composición
    - Agregación débil
    - Implementación
13. Un modelo tiene como finalidad:
    - Reducir la complejidad de un sistema
    - Servir como puente tecnológico entre diferentes profesionales
    - Como medio de documentación
    - Ninguna respuesta es correcta
14. Un elemento **Join** en un diagrama de actividad indica:
    - Paralelismo
    - Unión de ramas paralelas
    - Bifurcación
    - Envío de objeto
15. Una **interfaz** es una abstracción de una clase de forma que se puede heredar de esta atributos y métodos/funciones:
    - Verdadero
    - Falso
16. Algunas de las características de los modelos son:
    - Son inflexibles, no se pueden adaptar a las necesidades del usuario si no que este se tiene que adaptar a estos.
    - Son exhaustivos (recogen todas las características del sistema)
    - Son dependientes del tamaño del sistema a modelar

- Sólo son aptos para profesionales dedicados al desarrollo software.
- Los modelos no pueden ser modificados una vez creados.
- Sirven como medio para facilitar el análisis y comprensión del sistema.

17. Si se quisiera modelar el **comportamiento** de un sistema en función de los eventos que ocurren en él, se debería utilizar un:
    - Diagrama de clases
    - Diagrama de objetos
    - Diagrama de componentes
    - Diagrama de secuencia
    - Diagrama de estados
18. La multiplicidad **0 ..*** en una relación entre clases indica:
    - 0 o un número determinado
    - 0 o 1
    - 0 o un número indeterminado
    - Ninguna respuesta es correcta
19. En un diagrama de secuencia, si la información que se envía/recibe entre los elementos del sistema no espera que la otra parte envíe una respuesta, el mensaje es:
    - Síncrono
    - Asíncrono
    - De respuesta
    - Ninguna respuesta es correcta
20. ¿Es UML un estándar?
    - Verdadero
    - Falso
21. ¿Cuál es la etiqueta asociada a la siguiente relación  entre una clase y una interfaz?
    - extend

- include
- extends
- implements
- access

22. Los diagramas de componentes son una abstracción a más bajo nivel de un sistema que los diagramas de clases o de objetos
    - Verdadero
    - Falso
23. Los eventos, son propios de los diagramas de:
    - Diagramas de secuencia
    - Diagramas de actividad
    - Diagramas de estados
    - Diagramas de objetos
    - Diagramas de clases
24. En un diagrama de actividad, las actividades pueden comunicarse entre sí gracias al envío de objetos:
    - Verdadero
    - Falso
25. ¿Qué diagrama permite recoger los requisitos funcionales de un sistema?
    - Diagrama de actividad
    - Diagrama de eventos
    - Diagrama de secuencia
    - Diagrama de casos de uso
    - Ninguna respuesta es correcta

Temas 5 y 6: PREGUNTAS TEORÍA

1. ¿Qué es un requisito en ingeniería de software?

- Una función específica del sistema
- Una restricción legal
- Una descripción de servicios y restricciones operativas del sistema
- Un aspecto visual del sistema

2. La ingeniería de requisitos solo se lleva a cabo al inicio del desarrollo del software.

- Verdadero
- Falso

3. ¿Cuál de los siguientes NO es un tipo de requisito funcional?

- Permitir reservas de salas
- Generar informes de ocupación
- Bloquear horarios cuando se alcanza la capacidad máxima
- Normas de portabilidad del sistema

4. Los requisitos no funcionales incluyen aspectos como la portabilidad y la fiabilidad del sistema.

- Verdadero
- Falso

5. ¿Cuál es el objetivo principal de la ingeniería de requisitos?

- Crear el diseño visual del sistema
- Obtener una descripción completa del problema y requisitos del sistema
- Establecer los objetivos de marketing del sistema
- Redactar el manual del usuario

6. Los requisitos de dominio son específicos del entorno o aplicación en que se desarrollará el sistema.

- Verdadero

- Falso

7. ¿Qué técnica NO se utiliza comúnmente para la recogida de requisitos?

- Entrevistas
- Prototipos
- Publicidad en redes sociales
- Casos de uso

8. Los requisitos funcionales describen:

- Lo que el sistema debe hacer al entregarse
- Las restricciones legales del sistema
- Los estándares de seguridad
- La disponibilidad del hardware

9. La ingeniería de requisitos no se ve afectada por la carga de trabajo de las partes interesadas.

- Verdadero
- Falso

10. ¿Qué NO forma parte de los NFR (requisitos no funcionales)?

- Portabilidad
- Fiabilidad
- Gestión de inventario
- Seguridad

11. En un sistema de reservas, un requisito funcional sería:

- La portabilidad del sistema
- La capacidad de modificar una reserva
- La normativa de privacidad
- La facilidad de uso

12. Los requisitos no funcionales incluyen solo aspectos técnicos del sistema.

- Verdadero

- Falso

13. ¿Cuál es un desafío común en la ingeniería de requisitos?

- La estabilidad absoluta de requisitos durante el desarrollo
- La existencia de conflictos entre las partes interesadas
- La disponibilidad ilimitada de recursos
- La eliminación de toda restricción de hardware

14. ¿Qué aspecto NO es parte de la definición de requisitos funcionales?

- Servicios que debe ofrecer el sistema
- Comportamiento ante entradas específicas
- Restricciones organizacionales
- Tareas que el usuario debe realizar

15. La elicitación de requisitos se refiere a:

- La documentación de especificaciones
- El proceso de descubrir y definir los límites del sistema
- El diseño del hardware
- La verificación de errores

16. El enfoque ágil de desarrollo continúa la ingeniería de requisitos durante todo el ciclo de vida del sistema.

- Verdadero
- Falso

17. ¿Cuál de las siguientes NO es una técnica de elicitación?

- Diagramas de secuencia
- Reuniones
- Redacción de código
- Prototipos

18. Los requisitos externos en un sistema de software incluyen:

- La capacidad de recuperación ante fallos

- Restricciones legales o de interoperabilidad
- La interfaz de usuario
- Los criterios de usabilidad

19. La estabilidad de los requisitos es uno de los mayores desafíos en la ingeniería de requisitos.

- Verdadero
- Falso

20. En el contexto de requisitos, el término "NFR" significa:

- No Facturable y Recurrente
- Necesario para la Funcionalidad del Recurso
- Requisito No Funcional
- Requisito Natural Formal

21. Los requisitos funcionales suelen ser expresados en:

- Lenguaje natural
- Diagramas de casos de uso
- Diagramas de arquitectura
- Lenguaje de programación

22. ¿Cuál de las siguientes opciones describe mejor un requisito de fiabilidad?

- La capacidad de respuesta del sistema
- La consistencia del sistema en entornos inestables
- El tiempo que toma una tarea
- La facilidad de uso del sistema

23. En la elicitación de requisitos, se utilizan técnicas como:

- Entrevistas
- Prototipos
- Análisis de algoritmos
- Casos de uso

24. Un ejemplo de requisito no funcional sería:

- Responder en menos de 2 segundos
- Permitir modificar reservas
- Generar informes mensuales
- Ser compatible con sistemas externos

25. Un requisito de portabilidad define que el sistema debe poder:

- Ejecutarse en varios sistemas operativos
- Mantener su rendimiento en diferentes entornos
- Evitar conflictos con otros sistemas
- Ser fácil de modificar

26. El documento de requisitos es utilizado por:

- Desarrolladores
- Clientes que solicitan la construcción del sistema.
- Administradores de bases de datos
- Testers

27. La modificabilidad de un sistema es un tipo de requisito:

- Funcional
- De fiabilidad
- No funcional
- De integración

28. ¿Qué factores son esenciales en la fase de análisis de requisitos?

- Identificación de servicios
- Documentación de especificaciones
- Redacción de código
- Desarrollo de pruebas

29. Los requisitos no funcionales son también conocidos como:

- Requisitos secundarios

- Requisitos de calidad
- Requisitos de integración
- Requisitos contextuales

30. La disponibilidad de las partes interesadas es crucial para el proceso de:
    - Elicitación de requisitos
    - Codificación
    - Diseño de interfaz
    - Pruebas finales
31. Un sistema con alta usabilidad permite a los usuarios:
    - Comprender rápidamente las funciones del sistema
    - Personalizar completamente la interfaz
    - Reducir el tiempo de aprendizaje
    - Ejecutar tareas en varios dispositivos simultáneamente
32. ¿Cuál de los siguientes NO es un desafío común en la ingeniería de requisitos?
    - Evolución del sistema
    - Falta de disponibilidad de las partes interesadas
    - Completa estabilidad de los requisitos
    - Visión compartida
33. Un requisito de rendimiento en un sistema podría ser:
    - Aumentar la velocidad de respuesta a menos de X segundos
    - Facilitar la integración con sistemas externos
    - Mejorar el diseño visual
    - Asegurar la integridad de los datos
34. El framework NFR es utilizado para:
    - Modelar las propiedades de los requisitos no funcionales
    - Implementar algoritmos
    - Realizar pruebas del sistema

- Establecer la estructura de la base de datos

35. Los requisitos no funcionales que describen políticas y procedimientos de una organización son conocidos como:

- Requisitos de dominio
- Requisitos externos
- Requisitos organizacionales
- Requisitos de portabilidad

36. Los requisitos no funcionales que afectan la interacción del usuario se refieren a:

- Eficiencia
- Usabilidad
- Modificabilidad
- Seguridad

37. La estabilidad de los requisitos se ve afectada por cambios en:

- Necesidades del mercado
- Estructura del equipo de desarrollo
- Disponibilidad de recursos financieros
- Descubrimientos técnicos

38. La comprensión entre las partes interesadas puede mejorarse mediante:

- Lenguaje técnico avanzado
- Reuniones regulares
- Uso de un vocabulario común
- Exclusión de términos especializados

39. El tiempo de respuesta es un ejemplo de:

- Requisito de eficiencia
- Requisito funcional
- Requisito organizacional
- Requisito de mantenimiento

Tema 7: PREGUNTAS TEORÍA

1. En MDA, ¿Qué modelo recoge las necesidades del negocio sin centrarse en su implementación?
   - UML
   - PSM
   - PIM
   - CIM
2. ¿Qué elementos conforman un DSL en DSM?
   - Sintaxis abstracta
   - Sintaxis concreta
   - Semántica
   - Código ejecutable
3. En DSM, la semántica traduccional permite ejecutar un modelo directamente sin traducción previa.
   - Verdadero
   - Falso
4. ¿Qué es un modelo en el contexto de la Ingeniería Dirigida por Modelos?
   - Un diagrama visual de un sistema
   - Un conjunto de códigos fuente
   - Una base de datos de requisitos
   - Una abstracción de una realidad, eliminando elementos no esenciales
5. ¿Qué beneficios ofrece MDA?
   - Portabilidad
   - Coste cero de implementación
   - Productividad
   - Interoperabilidad
6. En MDA, ¿Cuáles son los modelos básicos especificados por MDA?

- BPEL
- CIM
- DSM
- PSM
- PIM
- BPMN

7. Los modelos pueden representarse en forma de árbol, visual o textualmente.
   - Verdadero
   - Falso
8. Un modelo recoge completamente todos los detalles del sistema real.
   - Verdadero
   - Falso
9. ¿Cuáles son los beneficios de utilizar modelos?
   - Menos coste de mantenimiento
   - Documentación redudante
   - Lenguaje común entre diferentes perfiles profesionales
   - Automatización del desarrollo mediante transformaciones
10. El Desarrollo Dirigido por Modelos (MDD) tiene como objetivo adaptar los principios de MDE cuando se desarrolla software.
    - Verdadero
    - Falso
11. En MDA, el modelo PSM se centra en las necesidades del negocio, sin detalles técnicos.
    - Verdadero
    - Falso

Tema 8: PREGUNTAS TEORÍA

1. ¿Qué es un metamodelo?
   - Un modelo de datos
   - Un modelo de modelos
   - Un diagrama de clases
   - Un lenguaje de programación
2. ¿Cuáles de las siguientes afirmaciones sobre ECORE son ciertas?
   - Es una versión simplificada de MOF
   - Es utilizado dentro de EMF
   - No permite la transformación a código
   - Se usa para definir metamodelos en Eclipse
3. ¿Cuáles de las siguientes características deben cumplir los metamodelos?
   - Operar a un nivel de abstracción mayor que el modelo
   - Permitir la creación de modelos basados en sus reglas
   - Definirse conforme a un metamodelo
   - No estar definido en MOF
4. ¿Cuál de las siguientes tecnologías está basada en el metamodelado?
   - SQL
   - UML
   - HTML
   - CSS
5. CMOF es una versión más simplificada que EMOF.
   - Verdadero
   - Falso
6. ¿Qué significa la sigla MOF?
   - Meta Object Facility
   - Model Oriented Framework
   - Model Object Factory
   - Metadata Object File

7. ¿Qué herramienta permite la definición y construcción de aplicaciones basadas en modelos en Eclipse?
   - UML
   - EMF
   - XMI
   - JSON
8. EMF permite definir metamodelos y generar código basado en ellos
   - Verdadero
   - Falso
9. EEnum en ECORE representa un tipo de dato enumerado.
   - Verdadero
   - Falso
10. ¿Qué elementos forman parte del metamodelo ECORE?
    - EClass
    - EReference
    - EAttribute
    - EQuery
11. La Ingeniería Dirigida por Modelos (MDE) busca facilitar la creación de software basado en modelos.
    - Verdadero
    - Falso
12. ¿Cuáles son las funciones del XMI?
    - Intercambio de modelos entre herramientas
    - Generación automática de código en Python
    - Almacenamiento y transformación de modelos
    - Creación de interfaces gráficas
13. ¿Qué elementos forman parte de EMF?
    - EMF Core
    - EMF.Edit
    - EMF.Codegen

- EMF.Compiler

14. Un modelo siempre debe cumplir con las reglas establecidas por su metamodelo.
    - Verdadero
    - Falso
15. Un metamodelo define los elementos y relaciones que pueden usarse en un modelo.
    - Verdadero
    - Falso
16. MOF es el lenguaje utilizado para definir los metamodelos en Eclipse.
    - Verdadero
    - Falso
17. XMI es un estándar basado en JSON para facilitar la interoperabilidad entre herramientas.
    - Verdadero
    - Falso
18. ECORE es un subconjunto de MOF utilizado en Eclipse.
    - Verdadero
    - Falso
19. EMF solo se puede usar en entornos Eclipse y no es compatible con otros IDEs.
    - Verdadero
    - Falso

Tema 9: PREGUNTAS TEORÍA

1. El analizador léxico se encarga de construir árboles de sintaxis abstracta a partir de tokens.
   - Verdadero
   - Falso
2. Un analizador sintáctico verifica que la estructura de los tokens siga las reglas de la gramática.
   - Verdadero
   - Falso
3. En gramáticas BNF y EBNF, ¿cuáles de las siguientes reglas son correctas?
   - A ::= C '+' B es una regla válida en BNF
   - D ::= {'0'|'1'|'2'}+ en EBNF significa que D puede tener uno o más valores de {0,1,2}
   - En BNF, los terminales se representan con letras mayúsculas
   - EBNF no admite expresiones regulares
4. ¿Qué es un DSL textual?
   - Un lenguaje de programación de propósito general
   - Un lenguaje de dominio específico basado en texto
   - Un lenguaje gráfico basado en diagramas
   - Un compilador para interpretar código
5. ¿Qué tipos de analizadores se utilizan en el procesamiento de lenguajes?
   - Analizador léxico
   - Analizador semántico
   - Analizador sintáctico
   - Analizador gráfico
6. Un DSL textual se caracteriza por utilizar representaciones gráficas en vez de símbolos textuales.
   - Verdadero
   - Falso

7. En Xtext, ¿cuál de las siguientes opciones NO es un componente clave del framework?
   - Analizador léxico
   - Generador de código
   - Motor de bases de datos SQL
   - Serializador de modelos
8. ¿Cuáles de las siguientes afirmaciones son verdaderas sobre los DSL textuales?
   - Permiten expresar de manera clara y concisa modelos específicos de un dominio
   - Se representan mediante una sintaxis textual basada en reglas
   - Requieren siempre de una interfaz gráfica para su edición
   - Son más adecuados para dominios donde la representación textual es clave, como SQL o LaTeX
9. Una gramática recursiva por la izquierda puede generar un bucle infinito si no se maneja correctamente.
   - Verdadero
   - Falso
10. ¿Cuáles de los siguientes operadores son utilizados en gramáticas EBNF?
    - ? para indicar opcionalidad
    - [] para indicar una lista de elementos opcionales
    - * para indicar una repetición obligatoria
    - {} para indicar repetición de 1 a n veces
11. ¿Cuáles de las siguientes afirmaciones sobre ANTLR son correctas?
    - Es un framework que permite generar analizadores sintácticos
    - Usa gramáticas basadas en EBNF pero con notación propia
    - Es exclusivamente compatible con Java
    - Permite manejar errores de sintaxis de manera eficiente
12. El analizador léxico se encarga de construir árboles de sintaxis abstracta a partir de tokens.
    - Verdadero

- Falso

13. Un analizador sintáctico verifica que la estructura de los tokens siga las reglas de la gramática.

    - Verdadero
    - Falso

Tema 10: PREGUNTAS TEORÍA

1. ¿Qué es un DSL visual?

    - Una representación textual para modelar un determinado sistema
    - Un modelo UML
    - Un asistente para la ejecución de transformaciones entre modelos
    - Una especificación basada en elementos gráficos para abstraer la complejidad de un sistema

2. Blockly permite generar código a partir de los bloques visuales.

    - Verdadero
    - Falso

3. ¿Qué es Sirius?

    - Un proyecto Eclipse para generar editores gráficos para DSL
    - Un framework de desarrollo web
    - Un proyecto de Eclipse centrado únicamente en la elaboración de gramáticas para DSL textuales
    - Un proyecto de Eclipse centrado únicamente en la elaboración de transformaciones entre modelos

4. ¿Cuáles son diferencias entre un DSL visual y un DSL textual?

    - El DSL visual usa elementos gráficos
    - No se puede generar código a partir de un DSL textual
    - El DSL visual usa elementos textuales
    - El DSL visual facilita más la comprensión de elementos y relaciones que un DSL textual

5. En Sirius, el archivo .odesign contiene la especificación visual de los elementos del metamodelo que hemos definido.

   - Verdadero
   - Falso

6. GMF no requiere un modelo EMF sobre el que personalizar la representación de los elementos.

   - Verdadero
   - Falso

7. Características de Sirius:

   - Solo permite trabajar haciendo uso de diagramas UML
   - La paleta de herramientas del editor es necesario construirla con otra herramienta aparte
   - Proporciona múltiples formas de representación visual para la especificación de los elementos del metamodelo
   - Permite empaquetar los DSL generados mediante plugins
   - No permite la integración con Xtext

Tema 11: PREGUNTAS TEORÍA

1. ¿Qué lenguajes forman parte de QVT?
   - QVTr
   - QVTc
   - QVTo
   - QVN
2. En QVTo, la directiva “mapping” define las vinculaciones entre modelos de entrada y salida.
   - Verdadero
   - Falso
3. ¿Qué tipo de transformación convierte un modelo en código fuente?
   - M2M
   - T2M
   - M2T
   - Horizontal
4. Una transformación in-place genera un nuevo modelo a partir de otro.
   - Verdadero
   - Falso
5. Características de ATL:
   - Compatible con el entorno de desarrollo integrado Eclipse
   - Solo admite transformaciones in place
   - Soporta transformación múltiple de modelos
   - Basado en OCL
6. ¿Qué tipo de transformación representa QVTo?
   - Visual
   - Declarativa
   - Imperativa basa en mapeos
   - Ninguna respuesta es correcta

7. JET permite automatizar la generación de código en proyectos EMF
   - Verdadero
   - Falso
8. Características de Xtend
   - Basado en Pascal
   - Métodos extendidos
   - Lambdas
   - Plantillas
   - Declarativo
9. ¿Qué estándar define QVTo para la realización de transformaciones entre modelos?
   - Visual
   - Declarativa
   - Imperativa basa en mapeos
   - Ninguna respuesta es correcta
10. ¿Qué es una transformación en el contexto del modelado del software?
    - Medio visual para escribir código
    - Proceso que convierte un modelo en otro modelo del mismo sistema
    - Medio textual para escribir código
    - Representación textual de un modelo
11. El primer paso para definir una transformación con QVTr es:
    - Definir relaciones entre los elementos del modelo
    - Importar contenido externo
    - Ejecutar la transformación del modelo
    - Definir las condiciones where/when
12. Una transformación vertical se da entre modelos del mismo nivel de abstracción.
    - Verdadero
    - Falso

Tema 1: SOLUCIONES PREGUNTAS TEORÍA

2. ¿Qué es el software?

   - Conjunto de instrucciones, procedimientos definidos y documentación técnica necesaria para el desarrollo y ejecución de programas, que pueden ser diseñados tanto para un cliente específico como para un mercado general
   - Conjunto de componentes físicos y electrónicos que permiten la interacción entre el usuario y los sistemas de computación, como procesadores, memorias y periféricos
   - Colección de normas y especificaciones técnicas utilizadas exclusivamente para la fabricación y ensamblaje de dispositivos de hardware en sistemas de computación
   - Infraestructura de telecomunicaciones que permite la interconexión entre computadoras y otros dispositivos, facilitando la transmisión de datos

3. ¿Qué características presenta el software frente al hardware?

   - Tangible
   - Dependencia al hardware
   - Alta flexibilidad
   - Fallos por desgaste

4. ¿Qué es el mantenimiento perfectivo?

   - Ajuste del software a un nuevo entorno
   - Prevención de errores futuros
   - Corrección de fallos emergentes
   - Inclusión de nuevos requisitos y características

5. ¿La versión gold de un software es una versión limitada de este?

   - Verdadero
   - Falso

6. ¿Cuáles de las siguientes afirmaciones sobre qué es la Ingeniería del Software son correctas?

   - La Ingeniería del Software se refiere al conjunto de reglas y pensamiento estructurado que uno aplica en el entorno para producir sistemas o componentes de software
   - La Ingeniería del Software es la aplicación de los principios de la ciencia de la computación y las matemáticas para lograr soluciones efectivas para problemas presentados en el software
   - La Ingeniería del Software se centra en la creación de software de alta calidad desde un enfoque sistemático, eficiente y controlado
   - La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, funcionamiento y mantenimiento del software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software
   - La Ingeniería del Software es la disciplina de ingeniería encargada de todos los aspectos relacionados con la producción de software desde sus etapas más tempranas de la especificación del sistema hasta el mantenimiento del sistema tras su puesta en marcha

7. ¿Qué 3 aspectos son claves en el software?

   - Calidad
   - Seguridad
   - Comportamiento
   - Simulación
   - Persistencia
   - Proceso de desarrollo
   - Conectividad

8. ¿Qué tipos de mantenimiento software existen?

   - Preventivo
   - Adaptativo
   - Perfectivo
   - Emergencia
   - Paulativo

- Distribuido
- Conectivo
- Correctivo

9. ¿Qué paradigma hace uso de reglas lógicas para resolver un problema?
   - Orientado a objetos
   - Imperativo
   - Declarativo
10. ¿Cuál es el paradigma en el que cobra una especial relevancia el orden de secuencia de las instrucciones para el control del flujo de programa?
    - Orientado a objetos
    - Imperativo
    - Declarativo
11. En el paradigma declarativo lo principal es cómo resolver un problema, no qué se debe conseguir:
    - Verdadero
    - Falso
12. ¿Cuáles son algunos de los criterios para medir la calidad del software?
    - Precisión
    - Fiabilidad
    - Conectividad
    - Legalidad
    - Mantenibilidad
13. La usabilidad es la facilidad con la que los programadores pueden realizar correcciones
    - Verdadero
    - Falso

Tema 2: SOLUCIONES PREGUNTAS TEORÍA

2. ¿Las historias de usuario son?
   - Conjunto de tareas que se agrupan para realizar en la próxima iteracción siguiendo el modelo Scrum.
   - Conjunto de documentación técnica sobre el software desarrollado para el cliente.
   - Son descripciones de las funcionalidades a implementar del sistema.
   - Conjunto de estimaciones ofrecidas por los miembros del equipo sobre las tareas a realizar.
3. ¿El product owner en Scrum es el encargado de supervisar que el equipo sigue las reglas bajo este marco?
   - Verdadero
   - Falso
4. ¿Cuáles de las siguientes características no se corresponden con el de desarrollo software mediante el modelo incremental?
   - Documentación disponible en todo momento
   - Cada ciclo produce un incremento
   - Fácil planificación
   - Se adapta bien a entornos de alta incertidumbre
   - Proceso de desarrollo no lineal por etapas
5. ¿Las modelos de desarrollo software tradicional siguen un proceso lineal en el que no se pasa a la siguiente etapa sin acabar la actual?
   - Verdadero
   - Falso
6. ¿Cuál es el flujo del modelo de desarrollo en cascada?
   - Análisis, Evaluación, Implementación, Despliegue, Diseño
   - Diseño, Análisis, Implementación, Despliegue, Evaluación

- Análisis, Implementación, Diseño, Despliegue, Evaluación
- Diseño, Implementación, Despliegue, Evaluación, Análisis
- Análisis, Diseño, Implementación, Evaluación, Despliegue
- Diseño, Análisis, Evaluación, Implementación, Despliegue
- Análisis, Implementación, Evaluación, Diseño, Despliegue

7. Algunas de las características de las metodologías ágiles frente a los modelos de desarrollo software tradicional son:
   - Requisitos variables
   - Equipo unidisciplinar
   - Cliente involucrado solo en el final del proceso
   - Entregas periódicas
   - Documentación exhaustiva
   - Flexible a cambios
8. La lista de tareas a realizar en la próxima iteracción en Scrum se denomina:
   - Sprint
   - Sprint backlog
   - Sprint meeting
   - Sprint review meeting
9. El planning póker se utiliza como medio de estimación de las historias de usuario
   - Verdadero
   - Falso
10. En las metodologías ágiles el proceso y su estado es conocido por todo el equipo de trabajo
    - Verdadero
    - Falso
11. Consecuencias del uso del método Lean:
    - Entregas justo en tiempo
    - Eliminación de desperdicios

- Eficiencia y calidad
- Ninguna respuesta es correcta

12. Valores de la metodología XP:
    - Comunicación
    - Distribución
    - Simplicidad
    - Polaridad
    - Valentía
    - Documentación continua
    - Respeto
    - Feedback
13. ¿Qué función tiene un tracker en la metodología XP?
    - Rastrea las estimaciones hechas por el equipo y da retroalimentación sobre su precisión para mejorar futuras estimaciones
    - Actúa como guía/asistente de los demás miembros del equipo en seguir el proceso XP
    - Guiar al equipo en la resolución de problemas específicos
    - Ayudan al cliente con las pruebas funcionales y la ejecución de estas
14. Si se parte del modelo basado en prototipos a la hora de construir software, se requiere que:
    - Todos los prototipos sean funcionales
    - El cliente participa en todo el proceso
    - No pretender cambios una vez que se presente el prototipo
    - Integrar todas las funcionalidades al principio
    - Ninguna respuesta es correcta
15. ¿Scrum hace uso de un tablero para organizar proyectos y flujos de trabajo?
    - Verdadero
    - Falso

16. ¿Pasa la documentación a un segundo plano si se hace uso de medios ágiles para el desarrollo de software?
    - Verdadero
    - Falso
17. ¿Existe un límite de tareas a realizar al mismo tiempo en Kanban?
    - Verdadero
    - Falso

Tema 3: SOLUCIONES PREGUNTAS TEORÍA

1. ¿Cuál es la fase más larga y compleja del Proceso Unificado de Desarrollo de Software?
    - Inicio
    - Elaboración
    - Construcción
    - Transición
2. ¿Cuáles de las siguientes actividades se llevan a cabo en la fase de transición del Proceso Unificado?
    - Migración de datos desde sistemas antiguos
    - Preparación del caso de negocio
    - Formación de los usuarios
    - Definición del diseño y arquitectura
3. La fase de inicio del Proceso Unificado es la más larga y compleja, ya que se concentra en la implementación del sistema.
    - Verdadero
    - Falso
4. ¿Qué aspecto es fundamental establecer durante la fase de elaboración?
    - La arquitectura y el diseño del sistema
    - La migración de datos

- La entrega del sistema a los usuarios
- La formación de los usuarios finales

5. En el Proceso Unificado, las fases del ciclo de vida se repiten hasta que el sistema esté listo para ser lanzado a los usuarios.
   - Verdadero
   - Falso
6. ¿Cuáles de las siguientes actividades se realizan en la fase de elaboración del Proceso Unificado?
   - Definir los requisitos del sistema
   - Establecer la arquitectura del sistema
   - Implementar el código fuente
   - Validar los casos de uso
7. La fase de transición en el Proceso Unificado se enfoca únicamente en la implementación de código y no incluye actividades relacionadas con los usuarios.
   - Verdadero
   - Falso
8. ¿Qué se entrega al final de la fase de construcción en el Proceso Unificado?
   - Un sistema completo listo para la transición
   - El plan de proyecto y los costos estimados
   - Diagramas de casos de uso
   - El plan de migración de datos
9. En el Proceso Unificado, cada iteración se generan versiones incrementales del sistema. ¿Cuáles de los siguientes aspectos deben ser revisados o mejorados en cada iteración para garantizar el éxito de las fases posteriores?
   - Refinamiento de los requisitos del sistema utilizando diagramas de casos de uso
   - Finalización y perfeccionamiento del diseño del sistema en base a la arquitectura definida
   - Definición de los costos y estimaciones para la fase de transición

- Migración de datos y formación de los usuarios
- Identificación y mitigación de riesgos a lo largo del proyecto

Tema 4: SOLUCIONES PREGUNTAS TEORÍA

1. ¿Qué es UML?
   - Es una notación para crear un estándar con el objetivo de representar un sistema.
   - Es un lenguaje de programación para especificar el comportamiento de un sistema y el paso de mensajes entre los diferentes componentes de este.
   - Es un tipo de diagrama con el objetivo de representar la estructura interna de un sistema.
2. ¿Cómo se dividen los diagramas según lo que pretenden modelar de un sistema?
   - Estática
   - Conexión
   - Comportamiento
   - Simulación
   - Persistencia
3. ¿Los casos de usos sirven para especificar los componentes internos de un sistema?
   - Verdadero
   - Falso
4. La línea de vida de un diagrama de secuencia simboliza:
   - Al usuario/objeto que envía el mensaje
   - El mensaje que se envía
   - El tiempo para llevar a cabo dicha acción
5. Un fragmento **alt** en un diagrama de secuencia permite especificar:
   - Una secuencia que se repite un número determinado de veces

- Una secuencia para especificar llamadas recursivas
- Una secuencia para especificar paralelismo entre flujos de mensajes
- Una secuencia if-else en el que en base al cumplimiento de una condición se ejecuta una secuencia u otra

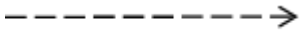
6. La relación de herencia entre clases tiene asociada una multiplicidad:
   - Verdadero
   - Falso
7. Esta especificación de un nombre **Modelado del Software:Asignatura** es propia de:
   - Diagramas de objetos
   - Diagramas de componentes
   - Diagramas de estados
   - Diagramas de clases
   - Diagramas de paquetes
8. Las relaciones **extends** entre casos de usos indican que uno de ellos es requerido por el otro para llevarse a cabo:
   - Verdadero
   - Falso
9. Un modelo es:
   - Un conjunto de diagramas que representan el sistema en su totalidad.
   - Una representación abstracta de un sistema o proceso que omite detalles irrelevantes.
   - Un diagrama que muestra la relación entre clases en un sistema.
   - Un programa ejecutable que imita el comportamiento del sistema.
10. Los componentes se pueden agrupar dentro de un diagrama de paquetes:
    - Verdadero
    - Falso
11. El modelo de arquitectura propuesto por Krutchen consta de las vistas:

- Lógica
- Desarrollo
- Proceso
- Física
- Ninguna respuesta es correcta

12. En un diagrama de clases, la relación  indica:
    - Herencia
    - Composición
    - Agregación débil
    - Implementación
13. Un modelo tiene como finalidad:
    - Reducir la complejidad de un sistema
    - Servir como puente tecnológico entre diferentes profesionales
    - Como medio de documentación
    - Ninguna respuesta es correcta
14. Un elemento **Join** en un diagrama de actividad indica:
    - Paralelismo
    - Unión de ramas paralelas
    - Bifurcación
    - Envío de objeto
15. Una **interfaz** es una abstracción de una clase de forma que se puede heredar de esta atributos y métodos/funciones:
    - Verdadero
    - Falso
16. Algunas de las características de los modelos son:
    - Son inflexibles, no se pueden adaptar a las necesidades del usuario si no que este se tiene que adaptar a estos.

- Son exhaustivos (recogen todas las características del sistema)
- Son dependientes del tamaño del sistema a modelar
- Sólo son aptos para profesionales dedicados al desarrollo software.
- Los modelos no pueden ser modificados una vez creados.
- Sirven como medio para facilitar el análisis y comprensión del sistema.

17. Si se quisiera modelar el **comportamiento** de un sistema en función de los eventos que ocurren en él, se debería utilizar un:
    - Diagrama de clases
    - Diagrama de objetos
    - Diagrama de componentes
    - Diagrama de secuencia
    - Diagrama de estados
18. La multiplicidad **0 ..*** en una relación entre clases indica:
    - 0 o un número determinado
    - 0 o 1
    - 0 o un número indeterminado
    - Ninguna respuesta es correcta
19. En un diagrama de secuencia, si la información que se envía/recibe entre los elementos del sistema no espera que la otra parte envíe una respuesta, el mensaje es:
    - Síncrono
    - Asíncrono
    - De respuesta
    - Ninguna respuesta es correcta
20. ¿Es UML un estándar?
    - Verdadero
    - Falso

21. ¿Cuál es la etiqueta asociada a la siguiente relación  entre una clase y una interfaz?

- extend
- include
- extends
- implements
- access

22. Los diagramas de componentes son una abstracción a más bajo nivel de un sistema que los diagramas de clases o de objetos

- Verdadero
- Falso

23. Los eventos, son propios de los diagramas de:

- Diagramas de secuencia
- Diagramas de actividad
- Diagramas de estados
- Diagramas de objetos
- Diagramas de clases

24. En un diagrama de actividad, las actividades pueden comunicarse entre sí gracias al envío de objetos:

- Verdadero
- Falso

25. ¿Qué diagrama permite recoger los requisitos funcionales de un sistema?

- Diagrama de actividad
- Diagrama de eventos
- Diagrama de secuencia
- Diagrama de casos de uso
- Ninguna respuesta es correcta

Temas 5 y 6: SOLUCIONES PREGUNTAS TEORÍA

1. ¿Qué es un requisito en ingeniería de software?

- Una función específica del sistema
- Una restricción legal
- Una descripción de servicios y restricciones operativas del sistema
- Un aspecto visual del sistema

2. La ingeniería de requisitos solo se lleva a cabo al inicio del desarrollo del software.

- Verdadero
- Falso

3. ¿Cuál de los siguientes NO es un tipo de requisito funcional?

- Permitir reservas de salas
- Generar informes de ocupación
- Bloquear horarios cuando se alcanza la capacidad máxima
- Normas de portabilidad del sistema

4. Los requisitos no funcionales incluyen aspectos como la portabilidad y la fiabilidad del sistema.

- Verdadero
- Falso

5. ¿Cuál es el objetivo principal de la ingeniería de requisitos?

- Crear el diseño visual del sistema
- Obtener una descripción completa del problema y requisitos del sistema
- Establecer los objetivos de marketing del sistema
- Redactar el manual del usuario

6. Los requisitos de dominio son específicos del entorno o aplicación en que se desarrollará el sistema.

- Verdadero
- Falso

7. ¿Qué técnica NO se utiliza comúnmente para la recogida de requisitos?

- Entrevistas
- Prototipos
- Publicidad en redes sociales
- Casos de uso

8. Los requisitos funcionales describen:

- Lo que el sistema debe hacer al entregarse
- Las restricciones legales del sistema
- Los estándares de seguridad
- La disponibilidad del hardware

9. La ingeniería de requisitos no se ve afectada por la carga de trabajo de las partes interesadas.

- Verdadero
- Falso

10. ¿Qué NO forma parte de los NFR (requisitos no funcionales)?

- Portabilidad
- Fiabilidad
- Gestión de inventario
- Seguridad

11. En un sistema de reservas, un requisito funcional sería:

- La portabilidad del sistema
- La capacidad de modificar una reserva
- La normativa de privacidad
- La facilidad de uso

12. Los requisitos no funcionales incluyen solo aspectos técnicos del sistema.

- Verdadero
- Falso

13. ¿Cuál es un desafío común en la ingeniería de requisitos?

- La estabilidad absoluta de requisitos durante el desarrollo
- La existencia de conflictos entre las partes interesadas
- La disponibilidad ilimitada de recursos
- La eliminación de toda restricción de hardware

14. ¿Qué aspecto NO es parte de la definición de requisitos funcionales?

- Servicios que debe ofrecer el sistema
- Comportamiento ante entradas específicas
- Restricciones organizacionales
- Tareas que el usuario debe realizar

15. La elicitación de requisitos se refiere a:

- La documentación de especificaciones
- El proceso de descubrir y definir los límites del sistema
- El diseño del hardware
- La verificación de errores

16. El enfoque ágil de desarrollo continúa la ingeniería de requisitos durante todo el ciclo de vida del sistema.

- Verdadero
- Falso

17. ¿Cuál de las siguientes NO es una técnica de elicitación?

- Diagramas de secuencia
- Reuniones
- Redacción de código
- Prototipos

18. Los requisitos externos en un sistema de software incluyen:

- La capacidad de recuperación ante fallos
- Restricciones legales o de interoperabilidad
- La interfaz de usuario
- Los criterios de usabilidad

19. La estabilidad de los requisitos es uno de los mayores desafíos en la ingeniería de requisitos.

- Verdadero
- Falso

20. En el contexto de requisitos, el término "NFR" significa:

- No Facturable y Recurrente
- Necesario para la Funcionalidad del Recurso
- Requisito No Funcional
- Requisito Natural Formal

21. Los requisitos funcionales suelen ser expresados en:

- Lenguaje natural
- Diagramas de casos de uso
- Diagramas de arquitectura
- Lenguaje de programación

22. ¿Cuál de las siguientes opciones describe mejor un requisito de fiabilidad?

- La capacidad de respuesta del sistema
- La consistencia del sistema en entornos inestables
- El tiempo que toma una tarea
- La facilidad de uso del sistema

23. En la elicitación de requisitos, se utilizan técnicas como:

- Entrevistas
- Prototipos
- Análisis de algoritmos

- Casos de uso

24. Un ejemplo de requisito no funcional sería:

- Responder en menos de 2 segundos
- Permitir modificar reservas
- Generar informes mensuales
- Ser compatible con sistemas externos

25. Un requisito de portabilidad define que el sistema debe poder:

- Ejecutarse en varios sistemas operativos
- Mantener su rendimiento en diferentes entornos
- Evitar conflictos con otros sistemas
- Ser fácil de modificar

26. El documento de requisitos es utilizado por:

- Desarrolladores
- Clientes que solicitan la construcción del sistema.
- Administradores de bases de datos
- Testers

27. La modificabilidad de un sistema es un tipo de requisito:

- Funcional
- De fiabilidad
- No funcional
- De integración

28. ¿Qué factores son esenciales en la fase de análisis de requisitos?

- Identificación de servicios
- Documentación de especificaciones
- Redacción de código
- Desarrollo de pruebas

29. Los requisitos no funcionales son también conocidos como:

- Requisitos secundarios
- Requisitos de calidad
- Requisitos de integración
- Requisitos contextuales

30. La disponibilidad de las partes interesadas es crucial para el proceso de:

- Elicitación de requisitos
- Codificación
- Diseño de interfaz
- Pruebas finales

31. Un sistema con alta usabilidad permite a los usuarios:

- Comprender rápidamente las funciones del sistema
- Personalizar completamente la interfaz
- Reducir el tiempo de aprendizaje
- Ejecutar tareas en varios dispositivos simultáneamente

32. ¿Cuál de los siguientes NO es un desafío común en la ingeniería de requisitos?

- Evolución del sistema
- Falta de disponibilidad de las partes interesadas
- Completa estabilidad de los requisitos
- Visión compartida

33. Un requisito de rendimiento en un sistema podría ser:

- Aumentar la velocidad de respuesta a menos de X segundos
- Facilitar la integración con sistemas externos
- Mejorar el diseño visual
- Asegurar la integridad de los datos

34. El framework NFR es utilizado para:

- Modelar las propiedades de los requisitos no funcionales
- Implementar algoritmos

- Realizar pruebas del sistema
- Establecer la estructura de la base de datos

35. Los requisitos no funcionales que describen políticas y procedimientos de una organización son conocidos como:

- Requisitos de dominio
- Requisitos externos
- Requisitos organizacionales
- Requisitos de portabilidad

36. Los requisitos no funcionales que afectan la interacción del usuario se refieren a:

- Eficiencia
- Usabilidad
- Modificabilidad
- Seguridad

37. La estabilidad de los requisitos se ve afectada por cambios en:

- Necesidades del mercado
- Estructura del equipo de desarrollo
- Disponibilidad de recursos financieros
- Descubrimientos técnicos

38. La comprensión entre las partes interesadas puede mejorarse mediante:

- Lenguaje técnico avanzado
- Reuniones regulares
- Uso de un vocabulario común
- Exclusión de términos especializados

39. El tiempo de respuesta es un ejemplo de:

- Requisito de eficiencia
- Requisito funcional
- Requisito organizacional

- Requisito de mantenimiento

Tema 7: SOLUCIONES PREGUNTAS TEORÍA

1. En MDA, ¿Qué modelo recoge las necesidades del negocio sin centrarse en su implementación?
   - UML
   - PSM
   - PIM
   - CIM
2. ¿Qué elementos conforman un DSL en DSM?
   - Sintaxis abstracta
   - Sintaxis concreta
   - Semántica
   - Código ejecutable
3. En DSM, la semántica traduccional permite ejecutar un modelo directamente sin traducción previa.
   - Verdadero
   - Falso
4. ¿Qué es un modelo en el contexto de la Ingeniería Dirigida por Modelos?
   - Un diagrama visual de un sistema
   - Un conjunto de códigos fuente
   - Una base de datos de requisitos
   - Una abstracción de una realidad, eliminando elementos no esenciales
5. ¿Qué beneficios ofrece MDA?
   - Portabilidad
   - Coste cero de implementación
   - Productividad

- Interoperabilidad

6. En MDA, ¿Cuáles son los modelos básicos especificados por MDA?
   - BPEL
   - CIM
   - DSM
   - PSM
   - PIM
   - BPMN
7. Los modelos pueden representarse en forma de árbol, visual o textualmente.
   - Verdadero
   - Falso
8. Un modelo recoge completamente todos los detalles del sistema real.
   - Verdadero
   - Falso
9. ¿Cuáles son los beneficios de utilizar modelos?
   - Menos coste de mantenimiento
   - Documentación redudante
   - Lenguaje común entre diferentes perfiles profesionales
   - Automatización del desarrollo mediante transformaciones
10. El Desarrollo Dirigido por Modelos (MDD) tiene como objetivo adaptar los principios de MDE cuando se desarrolla software.
    - Verdadero
    - Falso
11. En MDA, el modelo PSM se centra en las necesidades del negocio, sin detalles técnicos.
    - Verdadero
    - Falso

Tema 8: SOLUCIONES PREGUNTAS TEORÍA

1. ¿Qué es un metamodelo?
   - Un modelo de datos
   - Un modelo de modelos
   - Un diagrama de clases
   - Un lenguaje de programación
2. ¿Cuáles de las siguientes afirmaciones sobre ECORE son ciertas?
   - Es una versión simplificada de MOF
   - Es utilizado dentro de EMF
   - No permite la transformación a código
   - Se usa para definir metamodelos en Eclipse
3. ¿Cuáles de las siguientes características deben cumplir los metamodelos?
   - Operar a un nivel de abstracción mayor que el modelo
   - Permitir la creación de modelos basados en sus reglas
   - Definirse conforme a un metamodelo
   - No estar definido en MOF
4. ¿Cuál de las siguientes tecnologías está basada en el metamodelado?
   - SQL
   - UML
   - HTML
   - CSS
5. CMOF es una versión más simplificada que EMOF.
   - Verdadero
   - Falso
6. ¿Qué significa la sigla MOF?
   - Meta Object Facility
   - Model Oriented Framework
   - Model Object Factory

- Metadata Object File

7. ¿Qué herramienta permite la definición y construcción de aplicaciones basadas en modelos en Eclipse?

   - UML
   - EMF
   - XMI
   - JSON

8. EMF permite definir metamodelos y generar código basado en ellos

   - Verdadero
   - Falso

9. EEnum en ECORE representa un tipo de dato enumerado.

   - Verdadero
   - Falso

10. ¿Qué elementos forman parte del metamodelo ECORE?

    - EClass
    - EReference
    - EAttribute
    - EQuery

11. La Ingeniería Dirigida por Modelos (MDE) busca facilitar la creación de software basado en modelos.

    - Verdadero
    - Falso

12. ¿Cuáles son las funciones del XMI?

    - Intercambio de modelos entre herramientas
    - Generación automática de código en Python
    - Almacenamiento y transformación de modelos
    - Creación de interfaces gráficas

13. ¿Qué elementos forman parte de EMF?

    - EMF Core
    - EMF.Edit

- EMF.Codegen
- EMF.Compiler

14. Un modelo siempre debe cumplir con las reglas establecidas por su metamodelo.
    - Verdadero
    - Falso
15. Un metamodelo define los elementos y relaciones que pueden usarse en un modelo.
    - Verdadero
    - Falso
16. MOF es el lenguaje utilizado para definir los metamodelos en Eclipse.
    - Verdadero
    - Falso
17. XMI es un estándar basado en JSON para facilitar la interoperabilidad entre herramientas.
    - Verdadero
    - Falso
18. ECORE es un subconjunto de MOF utilizado en Eclipse.
    - Verdadero
    - Falso
19. EMF solo se puede usar en entornos Eclipse y no es compatible con otros IDEs.
    - Verdadero
    - Falso

Tema 9: SOLUCIONES PREGUNTAS TEORÍA

1. El analizador léxico se encarga de construir árboles de sintaxis abstracta a partir de tokens.

- Verdadero
- Falso

2. Un analizador sintáctico verifica que la estructura de los tokens siga las reglas de la gramática.
   - Verdadero
   - Falso
3. En gramáticas BNF y EBNF, ¿cuáles de las siguientes reglas son correctas?
   - A ::= C '+' B es una regla válida en BNF
   - D ::= {'0'|'1'|'2'}+ en EBNF significa que D puede tener uno o más valores de {0,1,2}
   - En BNF, los terminales se representan con letras mayúsculas
   - EBNF no admite expresiones regulares
4. ¿Qué es un DSL textual?
   - Un lenguaje de programación de propósito general
   - Un lenguaje de dominio específico basado en texto
   - Un lenguaje gráfico basado en diagramas
   - Un compilador para interpretar código
5. ¿Qué tipos de analizadores se utilizan en el procesamiento de lenguajes?
   - Analizador léxico
   - Analizador semántico
   - Analizador sintáctico
   - Analizador gráfico
6. Un DSL textual se caracteriza por utilizar representaciones gráficas en vez de símbolos textuales.
   - Verdadero
   - Falso
7. En Xtext, ¿cuál de las siguientes opciones NO es un componente clave del framework?
   - Analizador léxico
   - Generador de código

- Motor de bases de datos SQL
- Serializador de modelos

8. ¿Cuáles de las siguientes afirmaciones son verdaderas sobre los DSL textuales?
   - Permiten expresar de manera clara y concisa modelos específicos de un dominio
   - Se representan mediante una sintaxis textual basada en reglas
   - Requieren siempre de una interfaz gráfica para su edición
   - Son más adecuados para dominios donde la representación textual es clave, como SQL o LaTeX
9. Una gramática recursiva por la izquierda puede generar un bucle infinito si no se maneja correctamente.
   - Verdadero
   - Falso
10. ¿Cuáles de los siguientes operadores son utilizados en gramáticas EBNF?
    - ? para indicar opcionalidad
    - [] para indicar una lista de elementos opcionales
    - * para indicar una repetición obligatoria
    - {} para indicar repetición de 1 a n veces
11. ¿Cuáles de las siguientes afirmaciones sobre ANTLR son correctas?
    - Es un framework que permite generar analizadores sintácticos
    - Usa gramáticas basadas en EBNF pero con notación propia
    - Es exclusivamente compatible con Java
    - Permite manejar errores de sintaxis de manera eficiente
12. El analizador léxico se encarga de construir árboles de sintaxis abstracta a partir de tokens.
    - Verdadero
    - Falso
13. Un analizador sintáctico verifica que la estructura de los tokens siga las reglas de la gramática.
    - Verdadero

- Falso

Tema 10: SOLUCIONES PREGUNTAS TEORÍA

1. ¿Qué es un DSL visual?
   - Una representación textual para modelar un determinado sistema
   - Un modelo UML
   - Un asistente para la ejecución de transformaciones entre modelos
   - Una especificación basada en elementos gráficos para abstraer la complejidad de un sistema

2. Blockly permite generar código a partir de los bloques visuales.
   - Verdadero
   - Falso

3. ¿Qué es Sirius?
   - Un proyecto Eclipse para generar editores gráficos para DSL
   - Un framework de desarrollo web
   - Un proyecto de Eclipse centrado únicamente en la elaboración de gramáticas para DSL textuales
   - Un proyecto de Eclipse centrado únicamente en la elaboración de transformaciones entre modelos

4. ¿Cuáles son diferencias entre un DSL visual y un DSL textual?
   - El DSL visual usa elementos gráficos
   - No se puede generar código a partir de un DSL textual
   - El DSL visual usa elementos textuales
   - El DSL visual facilita más la comprensión de elementos y relaciones que un DSL textual

8. En Sirius, el archivo .odesign contiene la especificación visual de los elementos del metamodelo que hemos definido.
   - Verdadero

- Falso

9. GMF no requiere un modelo EMF sobre el que personalizar la representación de los elementos.

   - Verdadero
   - Falso

10. Características de Sirius:

    - Solo permite trabajar haciendo uso de diagramas UML
    - La paleta de herramientas del editor es necesario construirla con otra herramienta aparte
    - Proporciona múltiples formas de representación visual para la especificación de los elementos del metamodelo
    - Permite empaquetar los DSL generados mediante plugins
    - No permite la integración con Xtext

Tema 11: SOLUCIONES PREGUNTAS TEORÍA

1. ¿Qué lenguajes forman parte de QVT?

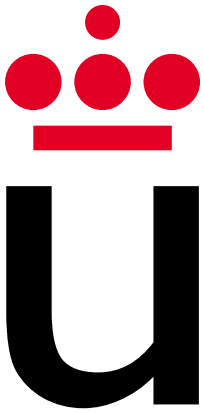
   - QVTr
   - QVTc
   - QVTo

- QVN

2. En QVTo, la directiva “mapping” define las vinculaciones entre modelos de entrada y salida.
   - Verdadero
   - Falso
3. ¿Qué tipo de transformación convierte un modelo en código fuente?
   - M2M
   - T2M
   - M2T
   - Horizontal
4. Una transformación in-place genera un nuevo modelo a partir de otro.
   - Verdadero
   - Falso
5. Características de ATL:
   - Compatible con el entorno de desarrollo integrado Eclipse
   - Solo admite transformaciones in place
   - Soporta transformación múltiple de modelos
   - Basado en OCL
6. ¿Qué tipo de transformación representa QVTo?
   - Visual
   - Declarativa
   - Imperativa basa en mapeos
   - Ninguna respuesta es correcta
7. JET permite automatizar la generación de código en proyectos EMF
   - Verdadero
   - Falso
8. Características de Xtend
   - Basado en Pascal
   - Métodos extendidos

- Lambdas
- Plantillas
- Declarativo

9. ¿Qué estándar define QVTo para la realización de transformaciones entre modelos?
   - Visual
   - Declarativa
   - Imperativa basa en mapeos
   - Ninguna respuesta es correcta
10. ¿Qué es una transformación en el contexto del modelado del software?
    - Medio visual para escribir código
    - Proceso que convierte un modelo en otro modelo del mismo sistema
    - Medio textual para escribir código
    - Representación textual de un modelo
11. El primer paso para definir una transformación con QVTr es:
    - Definir relaciones entre los elementos del modelo
    - Importar contenido externo
    - Ejecutar la transformación del modelo
    - Definir las condiciones where/when
13. Una transformación vertical se da entre modelos del mismo nivel de abstracción.
    - Verdadero
    - Falso



Universidad
Rey Juan Carlos

EJERCICIOS PRÁCTICOS

GRADO EN MATEMÁTICAS
CURSO:2024/2025

Cristian Gómez Macías

EJERCICIOS TEMA 4: LENGUAJE UNIFICADO DE MODELADO

EJERCICIOS DIAGRAMA DE ACTIVIDAD UML

EJERCICIO 1

En este ejercicio, se solicita que diseñar un diagrama de actividad en UML que represente el flujo completo flujo de actividades en una tienda en línea, donde intervienen tres actores principales:

1. Cliente: Es el usuario que realiza la compra.
2. Sistema de la Tienda: Se encarga de gestionar la información de productos y pedidos.
3. Banco: Procesa los pagos y confirma la transacción.

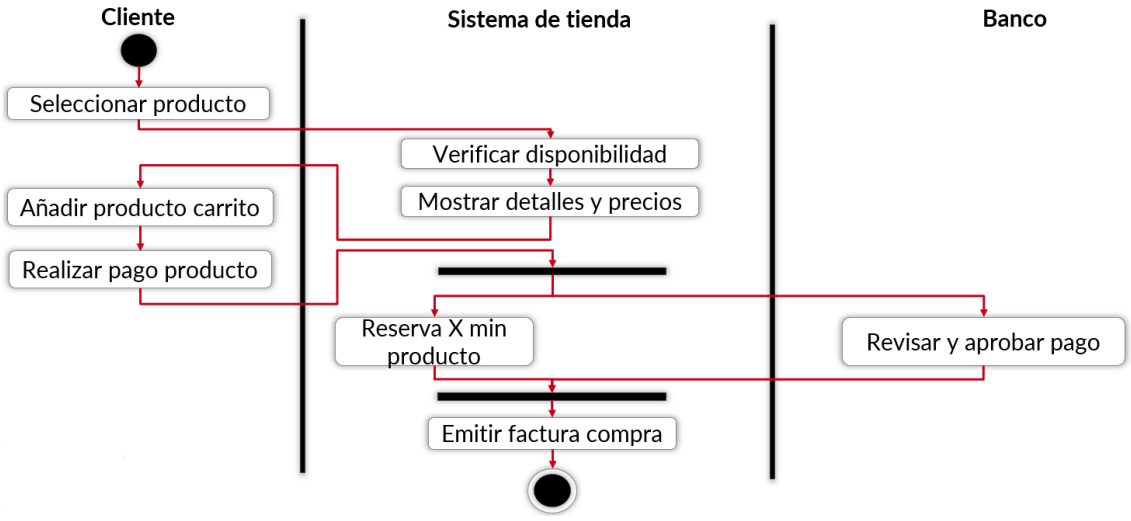
El flujo comienza cuando el cliente selecciona un producto en la tienda en línea. Luego, el sistema verifica la disponibilidad y muestra al cliente información detallada sobre precios y opciones.

Después, el cliente añade el producto al carrito y procede a pagar. En este punto, el sistema realiza dos tareas en paralelo:

- Solicita el pago al banco, que verifica los fondos y confirma la transacción.
- Reserva el producto en el inventario para evitar sobreventas durante unos minutos.

Una vez completadas ambas acciones, el sistema emite una confirmación de compra y genera el pedido. Finalmente, se emite la factura y el proceso concluye.

SOLUCIÓN EJERCICIO 1 DIAGRAMA DE ACTIVIDAD



EJERCICIO 2

En este ejercicio, se solicita que diseñar un diagrama de actividad en UML que represente el flujo completo de creación y generación de un DSL (Lenguaje Específico de Dominio) textual utilizando la herramienta Xtext.

Se debe considerar todas las etapas del proceso, desde la definición inicial de la gramática hasta la generación final de los componentes necesarios, como el metamodelo, pruebas, editor y clases Java. Hay que asegurar de modelar correctamente las decisiones que impliquen la corrección de errores y la posibilidad de ejecutar actividades en paralelo utilizando forks y joins.

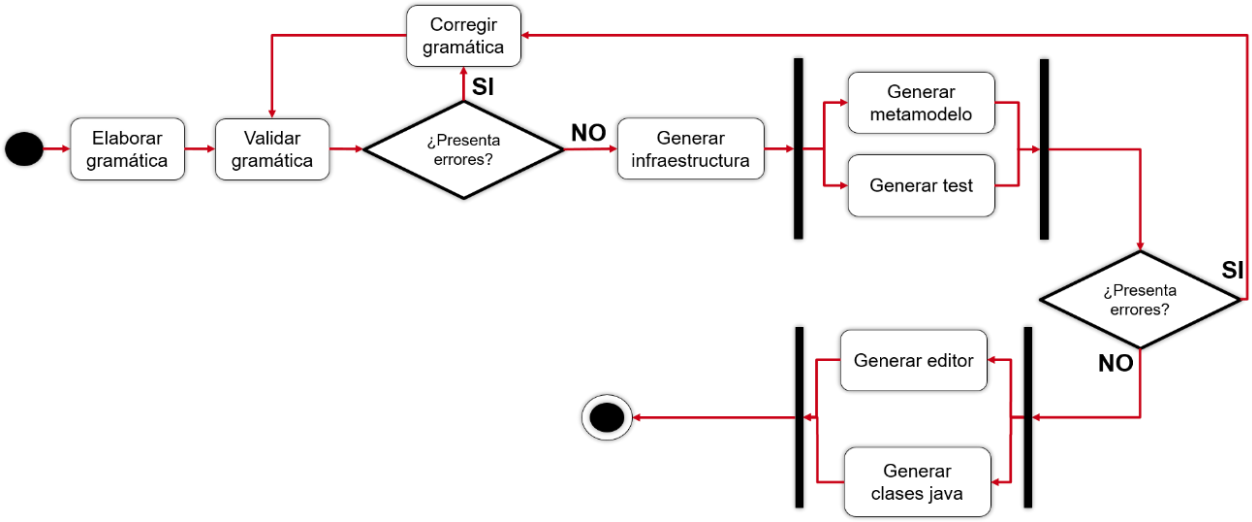
Requisitos del diagrama de actividad:

1. **Etapas clave del proceso** definidas en orden:

    1. Elaboración de la gramática del DSL.
    2. Validación de la gramática para verificar que no presenta errores.
    3. Generación de la infraestructura del lenguaje (parser, metamodelo, etc.).
    4. Ejecución de pruebas automáticas para verificar el funcionamiento de la infraestructura.
    5. Generación de componentes específicos como el editor y las clases Java (si el DSL requiere generar código).

2. **Decisiones**:
    - Incorporar decisiones que reflejen los momentos donde el proceso puede fallar (por ejemplo, validación de gramática o errores en la infraestructura).
    - Hay que asegurar que, en caso de errores, el flujo regrese a los pasos previos para realizar las correcciones necesarias.
3. **Ejecución paralela**:
    - Incluir actividades que pueden ejecutarse en paralelo, como la generación del **metamodelo** y las **pruebas** de la gramática, utilizando **forks y joins**.
4. **Corrección de errores**:
    - Definir adecuadamente el flujo de corrección de errores, de modo que el proceso vuelva a la corrección de la gramática o cualquier otro paso si se encuentran fallos.

SOLUCIÓN EJERCICIO 2 DIAGRAMA DE ACTIVIDAD

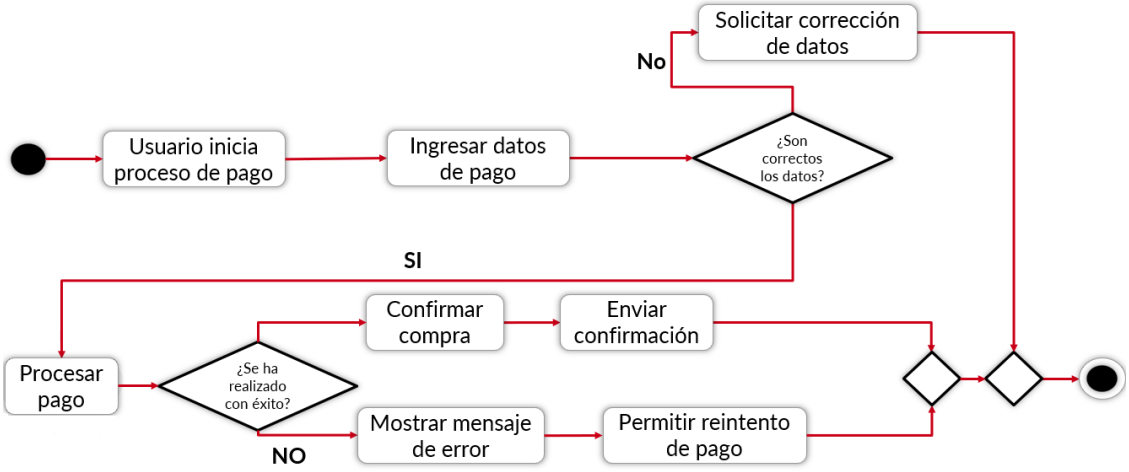


EJERCICIO 3

En este ejercicio, se solicita que diseñar un diagrama de actividad en UML que represente el flujo cuando un usuario realiza una compra en una tienda en línea. El sistema sigue el siguiente flujo de actividades:

- El usuario inicia el proceso de pago.
- El sistema verifica si los datos de pago son válidos.
    - Si los datos son incorrectos, se solicita corregirlos.
    - Si los datos son correctos, se procesa el pago.
        - Si el pago es exitoso, se confirma la compra y se envía un correo al usuario.
        - Si el pago falla, se muestra un mensaje de error y se da la opción de intentar nuevamente.

SOLUCIÓN EJERCICIO 3 DIAGRAMA DE ACTIVIDAD



EJERCICIOS DIAGRAMA DE ESTADOS UML

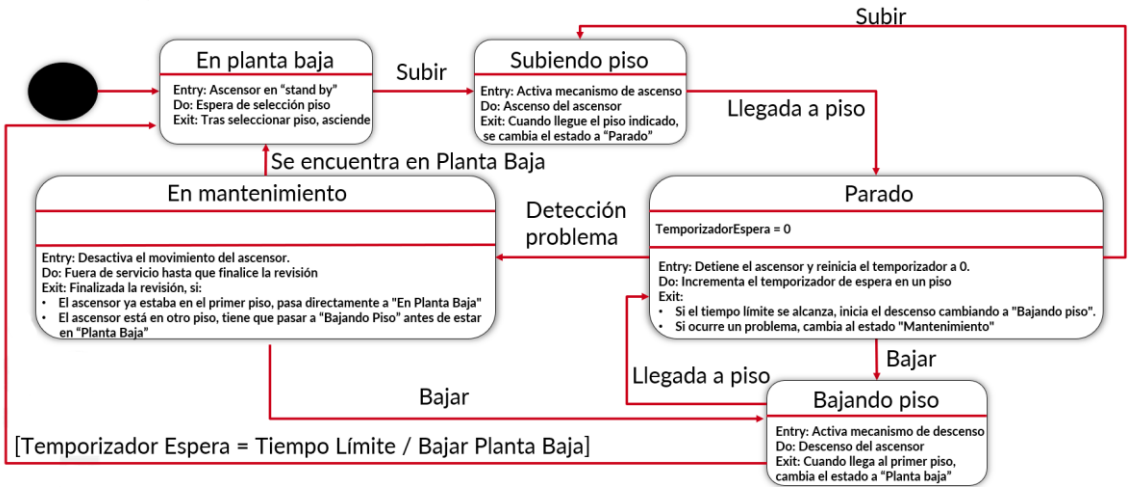
EJERCICIO 1

En este ejercicio, se solicita que diseñar un diagrama de estados en UML que represente el comportamiento de un ascensor que opera en un edificio con múltiples pisos. El sistema inicia en la planta baja y permite el movimiento hacia arriba o hacia abajo, deteniéndose en los pisos cuando sea necesario. Cuando el ascensor llega a un piso, se mantiene detenido por un tiempo determinado antes de continuar con su operación.

Si el ascensor se encuentra inactivo en un piso y el tiempo de espera excede un límite, este debe retornar automáticamente a la planta baja. Además, el sistema debe contemplar un estado en el cual el ascensor quede fuera de servicio debido a mantenimiento. En este estado, el ascensor no podrá moverse hasta que la revisión sea completada. Si el ascensor estaba en la planta baja, vuelve directamente al estado que representa dicho estado. Si el ascensor estaba en otro piso, primero tiene que bajar por otros pisos antes de regresar a la planta baja.

El modelo debe representar la transición entre estos estados considerando condiciones como el movimiento del ascensor, la llegada a un piso, la activación del temporizador y la detección de fallas.

SOLUCIÓN EJERCICIO 1 DIAGRAMA DE ESTADOS



EJERCICIO 2

Se solicita diseñar un diagrama de estados en UML que represente el comportamiento de un reproductor de música en un sistema de software.

El sistema inicia en un estado de espera (Stand by), donde el usuario puede seleccionar un contenido para reproducir. Una vez seleccionado, el sistema transita a un estado de carga, en el cual se verifica la disponibilidad del archivo y se configuran los parámetros necesarios para la reproducción.

Cuando el contenido ha sido cargado correctamente, el sistema pasa al estado de reproducción (Reproduciendo contenido musical), en el cual avanza el tiempo de reproducción de forma continua hasta que el usuario intervenga o la pista termine. Durante la reproducción, el usuario puede pausar, adelantar o rebobinar la pista, afectando el tiempo de reproducción en consecuencia.

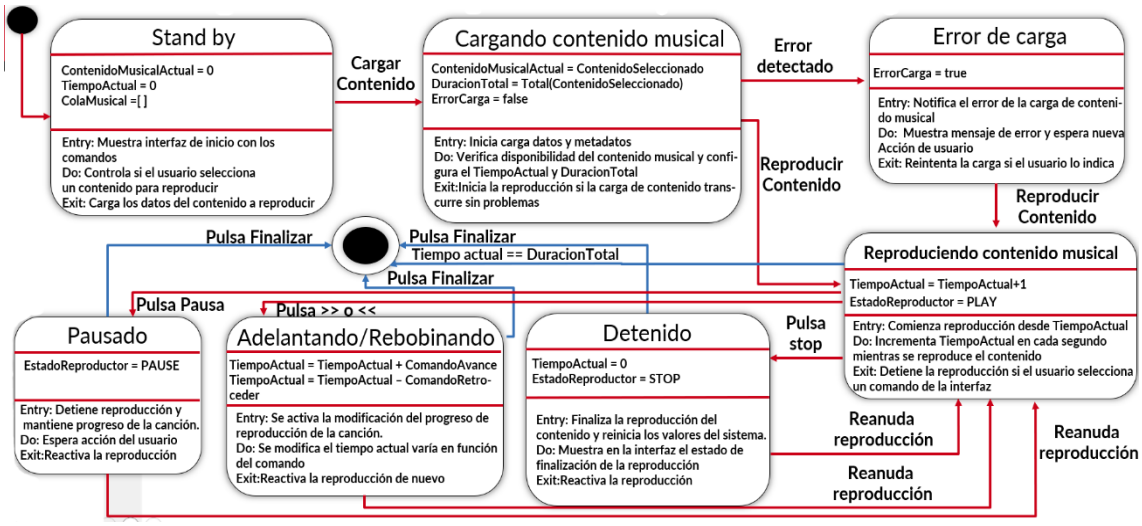
Si el usuario decide pausar la música, el sistema mantiene el progreso de la pista hasta que se reanude. En el caso de que el usuario decida adelantar o rebobinar, el tiempo de reproducción se ajusta dinámicamente en función del comando.

Si la pista finaliza porque TiempoActual == DuraciónTotal, el sistema transita directamente al estado final, sin pasar por el estado de detenido. Sin embargo, si el usuario detiene manualmente la reproducción, el sistema transita al estado de detenido, donde se reinician los valores del sistema y se espera una nueva acción del usuario.

En caso de un error de carga, el sistema debe notificar al usuario y permitirle intentar nuevamente la reproducción del contenido.

El modelo debe representar de manera clara las transiciones entre estos estados, asegurando que todas las interacciones del usuario y los eventos internos sean correctamente gestionadas.

SOLUCIÓN EJERCICIO 2 DIAGRAMA DE ESTADOS



El diagrama de estados anterior representa el funcionamiento del reproductor de música, detallando los estados en los que puede encontrarse y cómo cambia entre ellos según las acciones del usuario y las condiciones del sistema. El reproductor comienza en un estado de espera y avanza a otros estados a medida que se interactúa con él, hasta llegar a un estado final cuando la pista termina o el usuario decide detener la reproducción.

El estado inicial es el de Stand By, en el cual el sistema permanece inactivo hasta que el usuario seleccione una canción para reproducir. En este estado, el reproductor muestra su interfaz y revisa si se ha elegido una pista. Si el usuario selecciona un contenido, el sistema cambia al estado de Cargando Contenido Musical. En esta etapa, se verifican los datos de la canción, su duración y la disponibilidad del archivo. Si la carga se realiza correctamente, el sistema avanza automáticamente al estado de Reproduciendo Contenido Musical; en caso de error, pasa al estado Error de Carga, donde notifica al usuario y espera que decida si quiere reintentar la carga o regresar a la espera.

Cuando la pista se encuentra en el estado de Reproducción, el tiempo de reproducción se incrementa de manera constante mientras la canción sigue sonando. Desde este estado, el usuario puede interactuar con la música de distintas maneras. Si presiona pausa, el sistema cambia al estado Pausado, donde la reproducción se detiene temporalmente, pero se mantiene el progreso actual de la canción. Si el usuario adelanta o rebobina, el sistema pasa al estado Adelantando/Rebobinando, modificando el tiempo de reproducción según el comando recibido. Una vez que el usuario deja de adelantar o retroceder, el reproductor vuelve al estado de reproducción normal.

Si el usuario decide detener manualmente la música, el sistema pasa al estado Detenido, donde se reinician los valores del sistema y se muestra en la interfaz que la reproducción ha finalizado. Sin embargo, si la canción llega al final de forma natural porque TiempoActual == DuraciónTotal, el sistema no pasa por el estado de detenido, sino que va directamente al Estado Final, indicando que la sesión de reproducción ha terminado sin necesidad de intervención del usuario.

Las variables del sistema juegan un papel clave en el funcionamiento del reproductor. ContenidoMusicalActual almacena la canción que está en reproducción, TiempoActual

lleva el control del tiempo de la pista, DuraciónTotal indica cuánto dura la canción completa, y EstadoReproductor refleja si el sistema está en reproducción, pausa o detenido. Además, existen variables de control como ErrorCarga, que determina si hubo un problema al cargar la canción, y ColaMusical, que representa la lista de reproducción en caso de que haya varias canciones en espera.

En conclusión, este diagrama muestra de manera clara cómo un reproductor de música gestiona la reproducción de una canción, permitiendo pausarla, adelantarla, rebobinarla y detenerla. También maneja posibles errores y define una condición especial cuando la pista finaliza por sí sola, diferenciándola del caso en que el usuario detiene la música manualmente. Esto permite una administración eficiente del sistema y una experiencia fluida para el usuario.

EJERCICIOS DIAGRAMA DE SECUENCIA UML

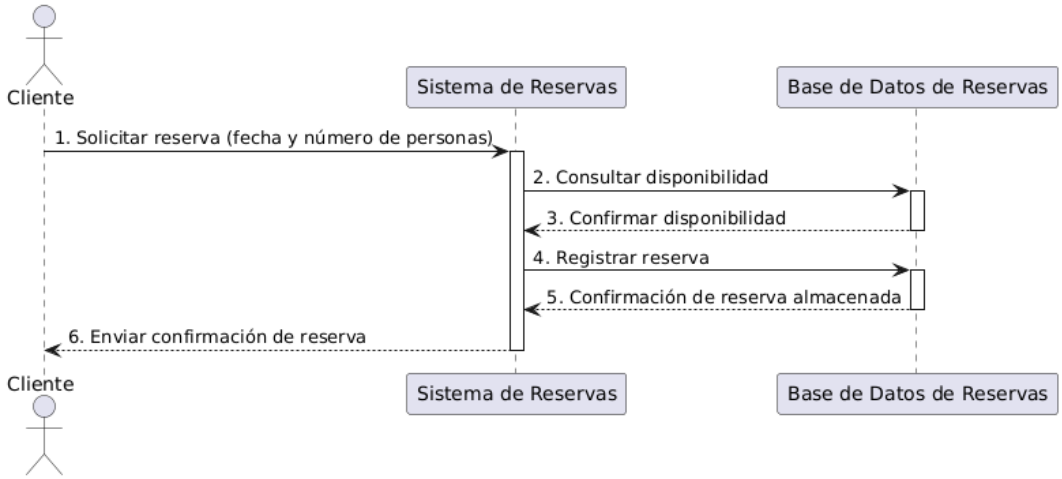
EJERCICIO 1

En este ejercicio se solicita elaborar un diagrama de secuencia UML para modelar el proceso de reserva de una mesa en su sistema.

Descripción del Flujo Principal

- El Cliente solicita una reserva indicando la fecha y el número de personas.
- El Sistema de Reservas consulta la disponibilidad con la Base de Datos de Reservas.
- Si hay disponibilidad, el Sistema confirma la reserva y la almacena en la base de datos.
- Si no hay disponibilidad, el Sistema informa al Cliente de que no hay mesas disponibles.
- Si la reserva se realiza correctamente, el sistema envía una confirmación al Cliente.

SOLUCIÓN EJERCICIO 1 DIAGRAMA DE SECUENCIA UML



EJERCICIO 2

En este ejercicio se solicita elaborar un diagrama de secuencia UML que modele el flujo de interacción entre el usuario, la interfaz del formulario (Boundary), el controlador del sistema (Control) y la base de datos (Entity).

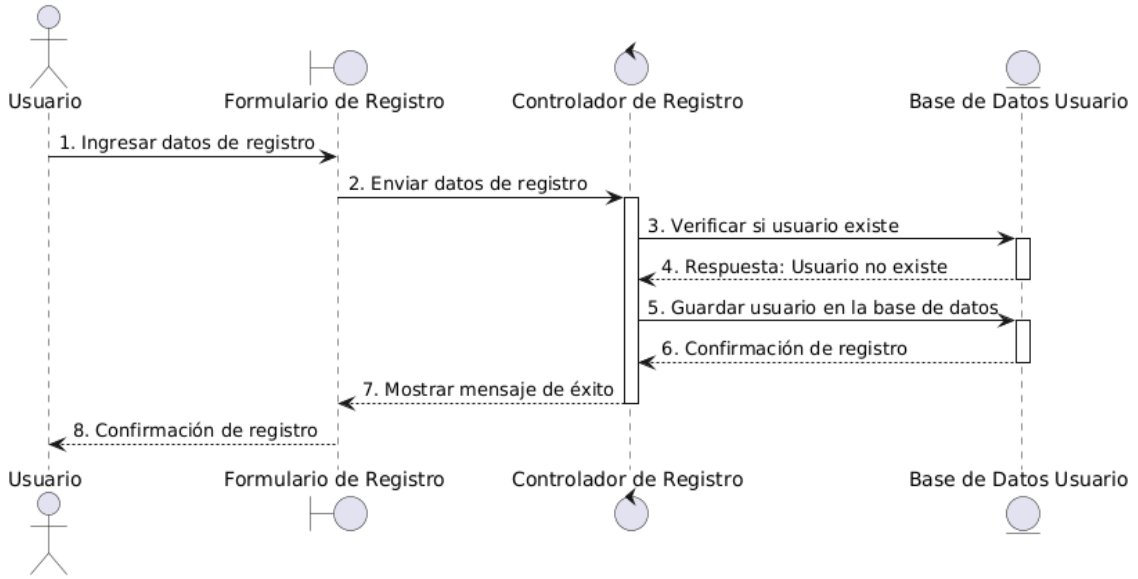
Una empresa desea implementar un sistema de gestión de usuarios donde los nuevos usuarios puedan registrarse en línea. El proceso de registro sigue los siguientes pasos:

- El usuario ingresa sus datos de registro (nombre, correo y contraseña) en un formulario web.
- El formulario envía la solicitud al servidor, donde un controlador verifica si el usuario ya existe en la base de datos.
- Si el usuario no existe, el controlador guarda la información en la base de datos.
- Si el usuario ya existe, se muestra un mensaje indicando que el correo ya está registrado.
- Una vez registrado correctamente, se muestra un mensaje de confirmación y se envía un correo de bienvenida.

Requisitos del diagrama de actividad:

- **Actor:** Usuario
- **Boundary:** Formulario de Registro
- **Control:** Controlador de Registro
- **Entity:** Base de Datos Usuario
- **Mensajes:** Interacciones entre estos elementos en orden cronológico.

SOLUCIÓN EJERCICIO 2 DIAGRAMA DE SECUENCIA UML



EJERCICIO 3

Un banco quiere modelar el proceso de autenticación y verificación de identidad de un usuario que intenta ingresar a su cuenta en línea. Para mejorar la seguridad, el sistema incluye intentos de autenticación limitados y un código de verificación enviado al usuario.

En este ejercicio se solicita dibujar un diagrama de secuencia UML que modele el flujo de compra e incluya flujos alternativos, mostrando las interacciones entre:

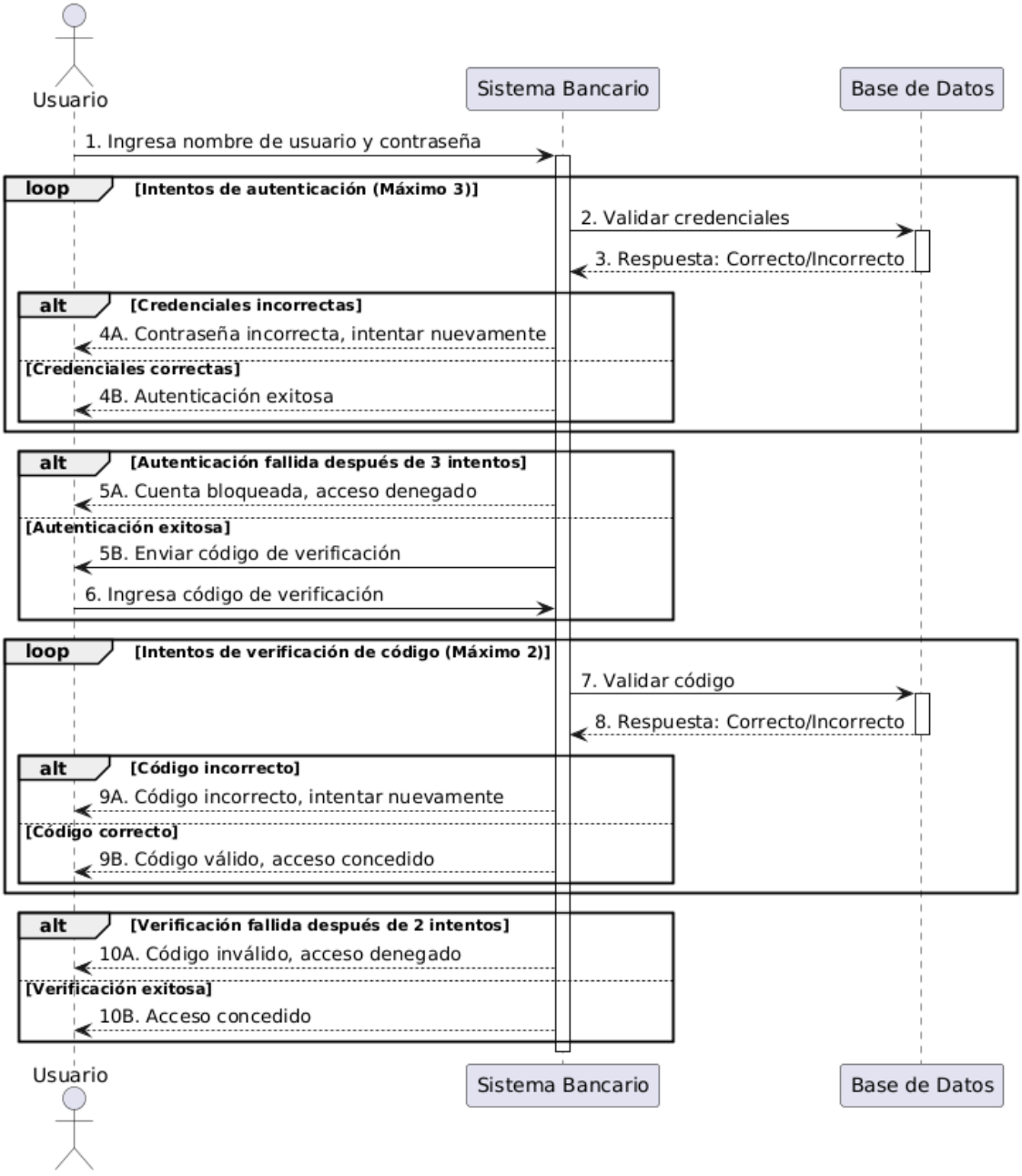
Descripción del Flujo Principal

- El usuario ingresa su nombre de usuario y contraseña.
- El sistema valida las credenciales.
    - Si son incorrectas, permite hasta 3 intentos antes de bloquear la cuenta (primer bucle).
    - Si son correctas, se envía un código de verificación al usuario
- El usuario ingresa el código recibido.
    - Si el código es incorrecto, permite hasta 2 intentos antes de cancelar el acceso (segundo bucle).
    - Si el código es correcto, el acceso es concedido.
- Si en cualquiera de los bucles el usuario agota sus intentos, el sistema ejecuta un break y bloquea el acceso.

Flujos Alternativos

- F1: Contraseña incorrecta → Permitir hasta 3 intentos antes de bloquear la cuenta (primer bucle).
- F2: Código incorrecto → Permitir hasta 2 intentos antes de cancelar la autenticación (segundo bucle).
- F3: Usuario supera los intentos en cualquier fase → Se ejecuta un break y el acceso es bloqueado.

SOLUCIÓN EJERCICIO 3 DIAGRAMA DE SECUENCIA UML



EJERCICIO 4

Una empresa de comercio electrónico quiere modelar el proceso de compra de productos en su tienda en línea, asegurándose de incluir validaciones, pagos y actualizaciones en el inventario.

En este ejercicio se solicita dibujar un diagrama de secuencia UML que modele el flujo de compra e incluya flujos alternativos, mostrando las interacciones entre:

- **Usuario** (Actor)
- **Boundary**: Formulario de Compra

- **Control**: Controlador de Pedidos y Pasarela de Pago
- **Entity**: Base de Datos de Inventario y Base de Datos de Pedidos

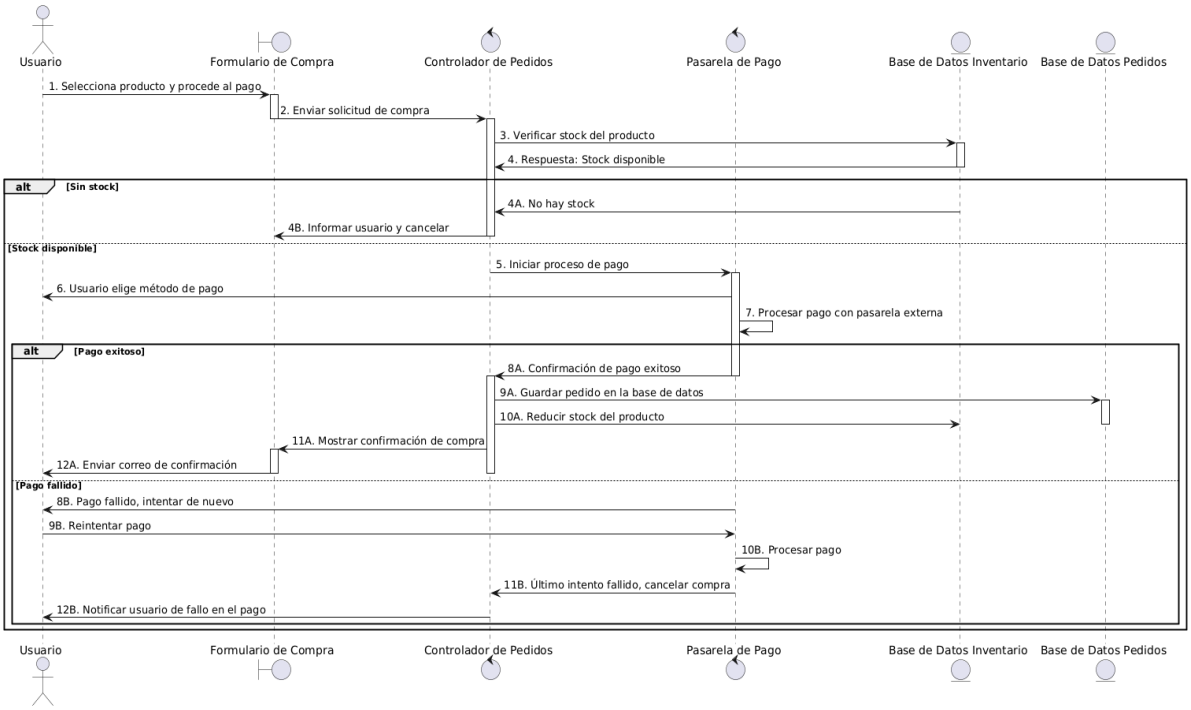
Descripción del Flujo Principal

- El usuario selecciona un producto y procede al pago.
- El formulario de compra recoge los datos e inicia el proceso con el Controlador de Pedidos.
- El controlador verifica la disponibilidad del producto en el inventario.
    - Si hay stock, se continúa con el pago.
    - Si no hay stock, se informa al usuario y se cancela la compra.
    - Si hay stock, el usuario selecciona el método de pago (tarjeta o PayPal).
    - El sistema procesa el pago mediante una pasarela externa.
- Si el pago es exitoso, se confirma la compra y se reduce el stock del producto.

Flujos Alternativos

- F1: Producto sin stock → Si el inventario no tiene suficientes unidades, el sistema cancela la compra y notifica al usuario.
- F2: Pago fallido → Si el pago es rechazado, el usuario puede intentar nuevamente o elegir otro método de pago.

SOLUCIÓN EJERCICIO 4 DIAGRAMA DE SECUENCIA UML



EJERCICIO 5

Una empresa de logística necesita modelar el flujo de gestión de pedidos, donde se debe verificar la disponibilidad, revisar el pedido y aprobarlo antes de su envío.

En este ejercicio se solicita dibujar un diagrama de secuencia UML que modele este flujo, asegurando:

- Uso de actores diferentes (Cliente, Supervisor, Gerente, Sistema de Gestión de Pedidos).
- Bucles y condiciones (loop, alt, opt) para modelar intentos de corrección y fallos.

Descripción del Flujo Principal

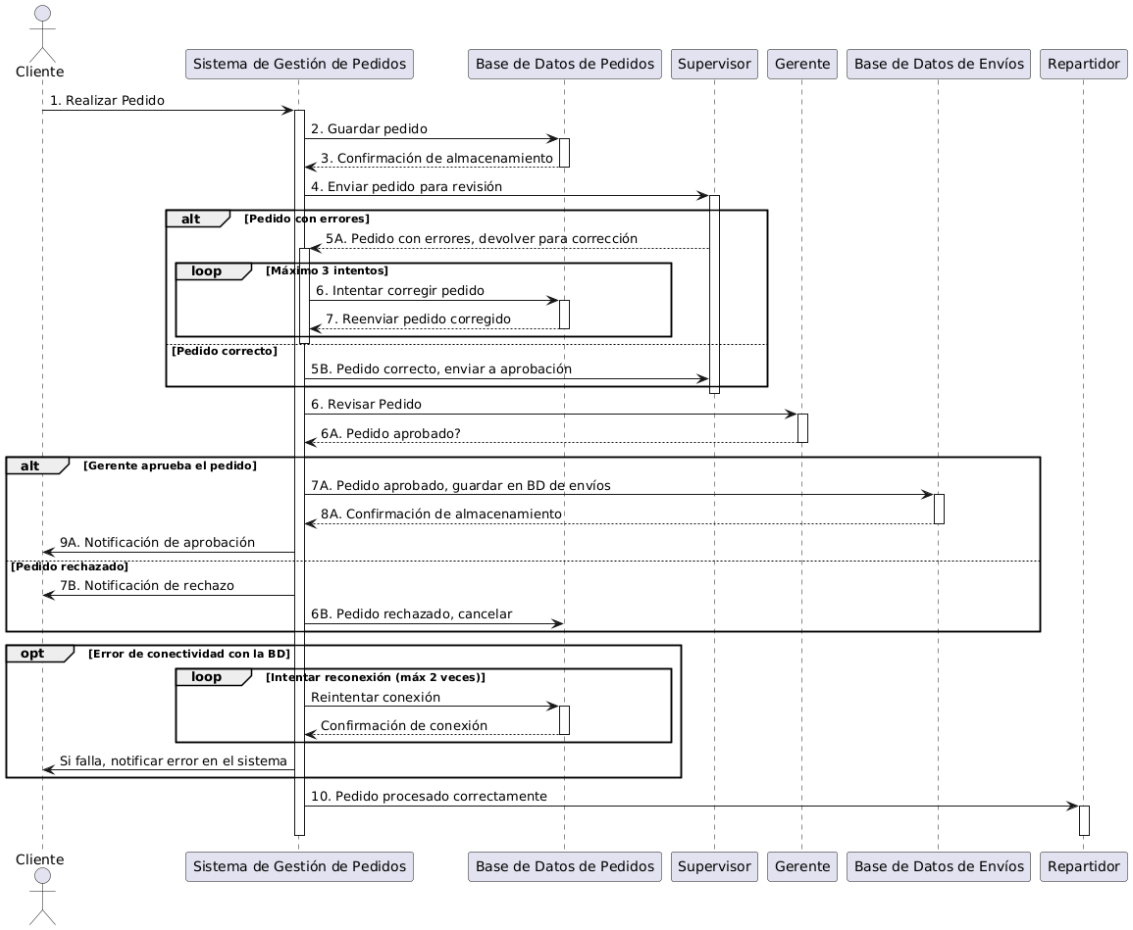
Un Cliente realiza un pedido y lo envía al Sistema de Gestión de Pedidos.
El sistema añade el pedido a la Base de Datos de Pedidos.
Se inicia un proceso de revisión que es realizado por un Supervisor:

- Si el pedido tiene errores, se devuelve al sistema para su corrección.
- Si está correcto, se envía para su aprobación final. Un Gerente revisa el pedido y decide si lo aprueba o rechaza:
  - Si lo aprueba, se envía a la Base de Datos de Envíos.
  - Si lo rechaza, se notifica al sistema y se marca el pedido como cancelado.
- Si el pedido fue aprobado, el sistema genera una orden de envío y la asigna a un repartidor.

Flujos Alternativos

- F1: Pedido con Errores → Si el supervisor detecta errores, se devuelve al sistema y este puede corregirlo antes de reenviarlo.
- F2: Pedido Rechazado → Si el gerente no aprueba el pedido, el sistema lo marca como cancelado y notifica al cliente.
- F3: Intento de Corrección → Si un pedido se devuelve para corrección, el sistema intentará tres veces corregirlo antes de rechazarlo automáticamente. *(Aquí se usa un bucle loop)*
- F4: Error de Conectividad → Si el sistema no puede comunicarse con la base de datos, reintenta la operación hasta 2 veces antes de notificar un fallo. *(Uso de alt y loop en secuencia)*

SOLUCIÓN EJERCICIO 5 DIAGRAMA DE SECUENCIA UML



EJERCICIOS DIAGRAMA DE CLASE UML

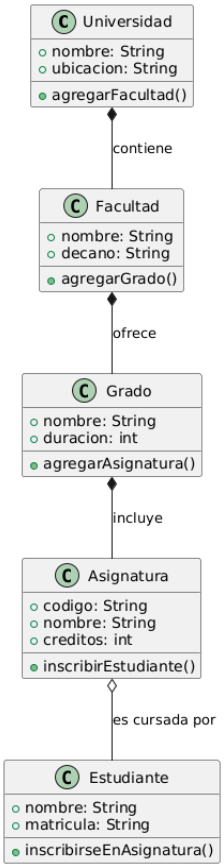
EJERCICIO 1

La universidad necesita modelar su sistema académico con un diagrama de clases que refleje la estructura organizativa de los estudios y las relaciones entre sus elementos.

El diagrama de clases que se modele debe cumplir las siguientes reglas:

- La Universidad puede tener varias Facultades, pero una Facultad solo pertenece a una Universidad.
- Cada Facultad tiene un conjunto de Grados, pero un Grado solo puede pertenecer a una Facultad.
- Cada Grado tiene varias Asignaturas, y cada Asignatura pertenece a un solo Grado.
- Una Asignatura puede ser cursada por múltiples Estudiantes, pero los Estudiantes pueden inscribirse en varias Asignaturas sin estar limitados a un solo Grado.

SOLUCIÓN EJERCICIO 1 DIAGRAMA DE CLASE UML



EJERCICIO 2

En este ejercicio se solicita elaborar un diagrama de clases UML para una empresa que está definiendo un sistema de control para dispositivos inteligentes en el hogar, que permite gestionar diferentes tipos de dispositivos electrónicos desde una aplicación centralizada.

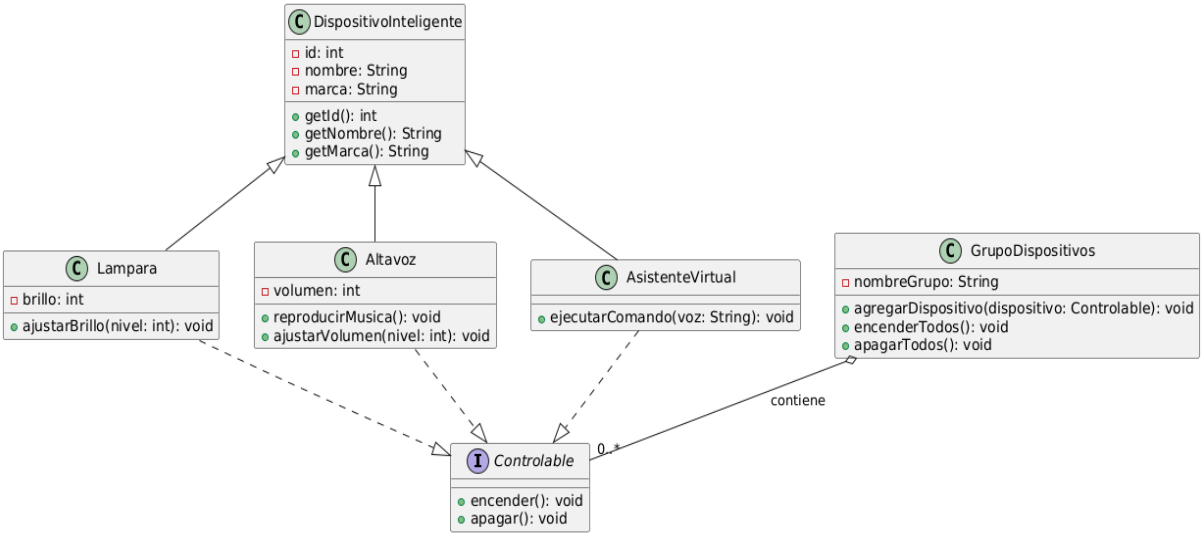
Los dispositivos pueden encenderse y apagarse, así como ajustar configuraciones específicas dependiendo de su tipo. Por ejemplo, una lámpara inteligente puede ajustar su brillo e intensidad, mientras que un altavoz inteligente puede reproducir música y ajustar el volumen.

Además, ciertos dispositivos permiten el control por voz, lo que les permite ejecutar comandos mediante asistentes virtuales. Sin embargo, no todos los dispositivos soportan esta funcionalidad.

El sistema debe permitir la gestión de dispositivos de diferentes fabricantes, asegurando que se puedan integrar en la plataforma sin importar la marca o el modelo. También debe permitir la creación de grupos de dispositivos, para que los usuarios puedan controlar varios dispositivos a la vez con una sola acción.

El diseño del software debe garantizar que los dispositivos compartan comportamientos comunes, pero que también puedan tener funcionalidades específicas dependiendo de su tipo. Además, la implementación debe permitir la integración de nuevos tipos de dispositivos en el futuro sin afectar la estructura del sistema

SOLUCIÓN EJERCICIO 2 DIAGRAMA DE CLASE UML



Justificación Interfaz Controlable:

La implementación de la interfaz Controlable por parte de los dispositivos se debe a que no todos ellos poseen las mismas funcionalidades, pero todos deben poder ser controlados de alguna manera.

No se utiliza una clase abstracta porque los dispositivos no comparten atributos o métodos concretos, sino únicamente una funcionalidad común (encender(), apagar()), aunque pueden tener diferentes capacidades específicas.

Cada tipo de dispositivo puede implementar Controlable de manera distinta, lo que permite flexibilidad en futuras expansiones sin necesidad de modificar una clase base.

EJERCICIO 3

En este ejercicio se solicita elaborar un diagrama de clases UML para una empresa de transporte inteligente que se encuentra elaborando un software para gestionar su flota de vehículos autónomos. El sistema debe permitir el control, supervisión y operación de diferentes tipos de vehículos sin intervención humana directa.

Cada vehículo dentro del sistema tiene características generales como un identificador único, información sobre su estado operativo, nivel de batería y ubicación en tiempo real. Es fundamental que los vehículos puedan moverse, detenerse y reportar su estado a una central de monitoreo.

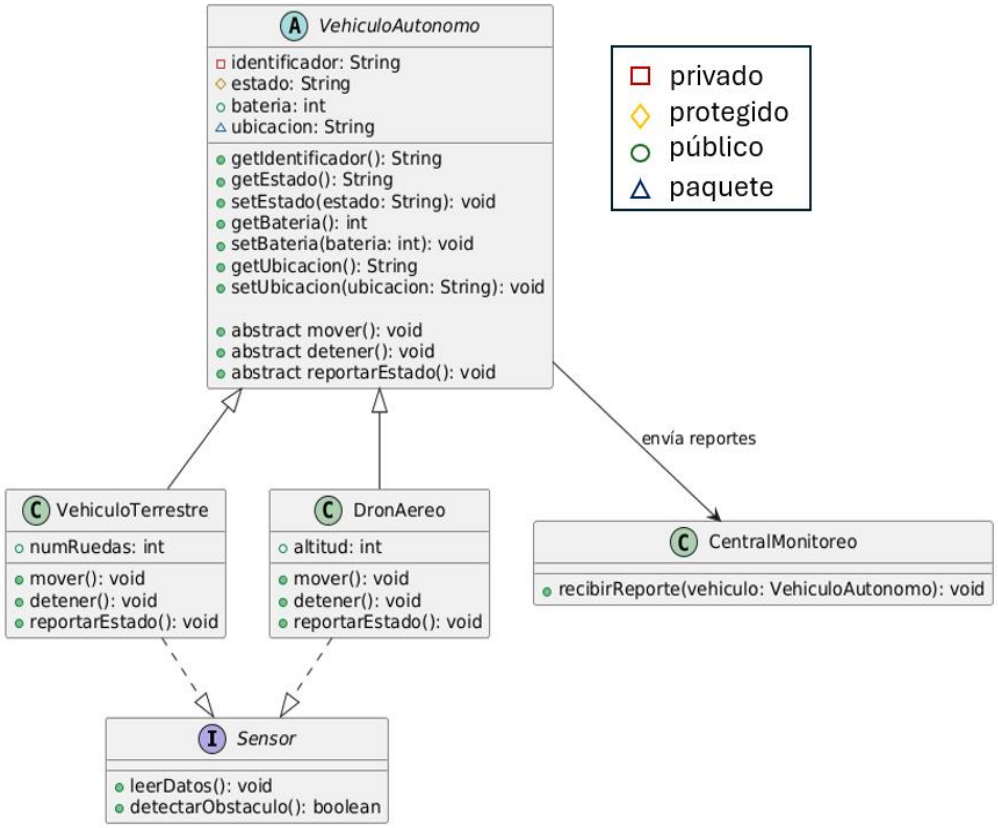
Dentro de la flota, se gestionan diferentes tipos de vehículos, cada uno con su propia forma de operación y navegación. Existen vehículos terrestres, que utilizan ruedas para desplazarse y operan en calles y carreteras, y drones aéreos, que se desplazan en el aire y requieren un control preciso de altitud y estabilidad.

Para asegurar la correcta navegación y evitar colisiones, los vehículos cuentan con sensores que les permiten leer datos del entorno y detectar obstáculos en su trayectoria. Es necesario que todos los vehículos autónomos tengan esta capacidad, independientemente de su tipo.

Además, el sistema debe incluir una central de monitoreo que reciba informes periódicos de los vehículos en operación. Desde esta central, se pueden analizar los reportes enviados, verificando la ubicación y el estado de cada unidad para tomar decisiones sobre su operación.

El software debe estar diseñado de manera estructurada, asegurando que cada componente del sistema cumpla con su función específica y se mantenga la escalabilidad del modelo para futuras expansiones de la flota.

SOLUCIÓN EJERCICIO 3 DIAGRAMA DE CLASE UML



EJERCICIO 4

En este ejercicio se solicita elaborar un diagrama de clases UML para una empresa tecnológica que se encuentra desarrollando un sistema integral de gestión para una red de hospitales. El sistema debe organizar y administrar pacientes, médicos, tratamientos, citas médicas y hospitales, garantizando la eficiencia en la atención y el acceso a la información.

Cada hospital cuenta con su propio equipo de médicos y personal administrativo, así como con un conjunto de especialidades médicas disponibles. Los médicos pueden estar asociados a múltiples hospitales, dependiendo de su disponibilidad y especialización.

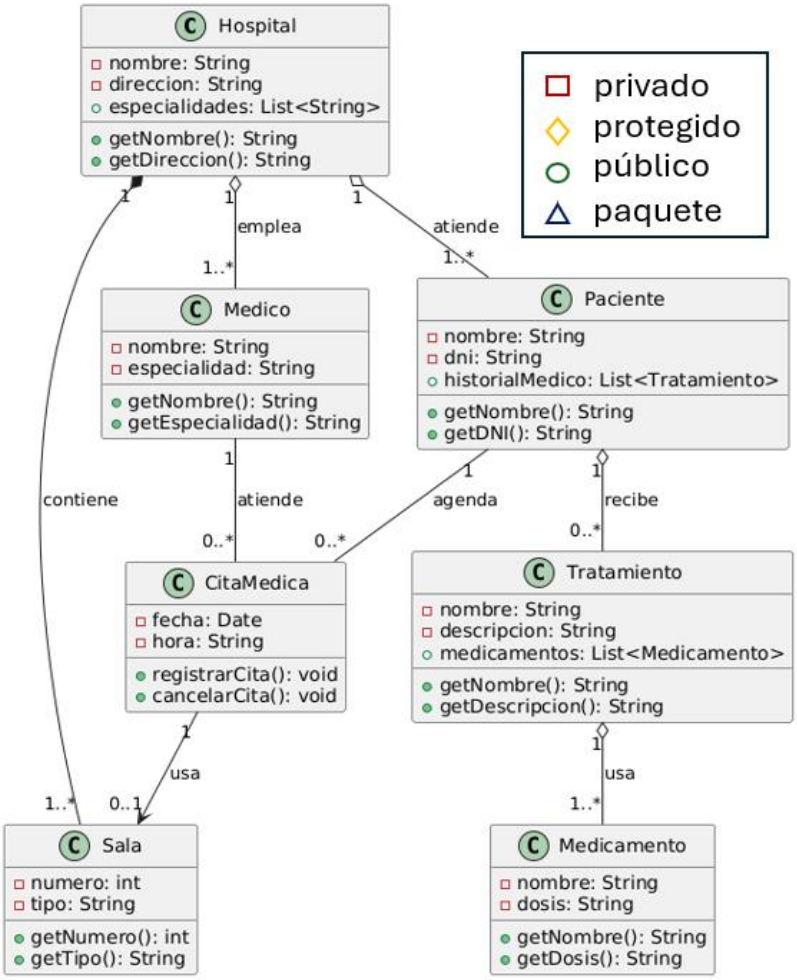
Los hospitales ofrecen diferentes tipos de tratamientos y procedimientos médicos, los cuales pueden incluir medicamentos específicos. Un tratamiento puede requerir varios medicamentos, y un medicamento puede ser utilizado en distintos tratamientos.

El sistema también debe gestionar a los pacientes registrados, los cuales pueden programar citas con médicos en cualquiera de los hospitales disponibles. Una cita médica está asociada a un solo paciente, pero puede involucrar a varios médicos en una consulta conjunta si es necesario.

Cada paciente puede recibir uno o más tratamientos, y los tratamientos pueden involucrar diferentes médicos según la especialidad requerida. Además, los hospitales pueden disponer de salas especializadas, que deben ser reservadas para procedimientos médicos específicos dentro de un hospital.

El sistema debe garantizar la organización estructurada de la información, permitiendo la correcta gestión de los recursos médicos y hospitalarios, así como el seguimiento detallado del historial clínico de los pacientes. Además, debe permitir la expansión a nuevos hospitales dentro de la red sin afectar la integridad de los datos actuales.

SOLUCIÓN EJERCICIO 4 DIAGRAMA DE CLASE UML



EJERCICIO 5

En este ejercicio se solicita elaborar un diagrama de clases UML para una empresa tecnológica está desarrollando un sistema integral de comercio electrónico con el fin de gestionar el catálogo de productos, las compras de los clientes, los pagos y los envíos de mercancía. La plataforma debe permitir la interacción entre vendedores, clientes y transportistas, garantizando la seguridad y eficiencia en todas las transacciones realizadas.

Los vendedores pueden registrar diferentes tipos de productos en la plataforma, indicando detalles como nombre, precio y cantidad disponible en stock. Estos productos pueden clasificarse en una o varias categorías, lo que facilita la organización del catálogo. Los clientes pueden realizar pedidos, seleccionando múltiples productos dentro de un mismo pedido y eligiendo su método de pago preferido.

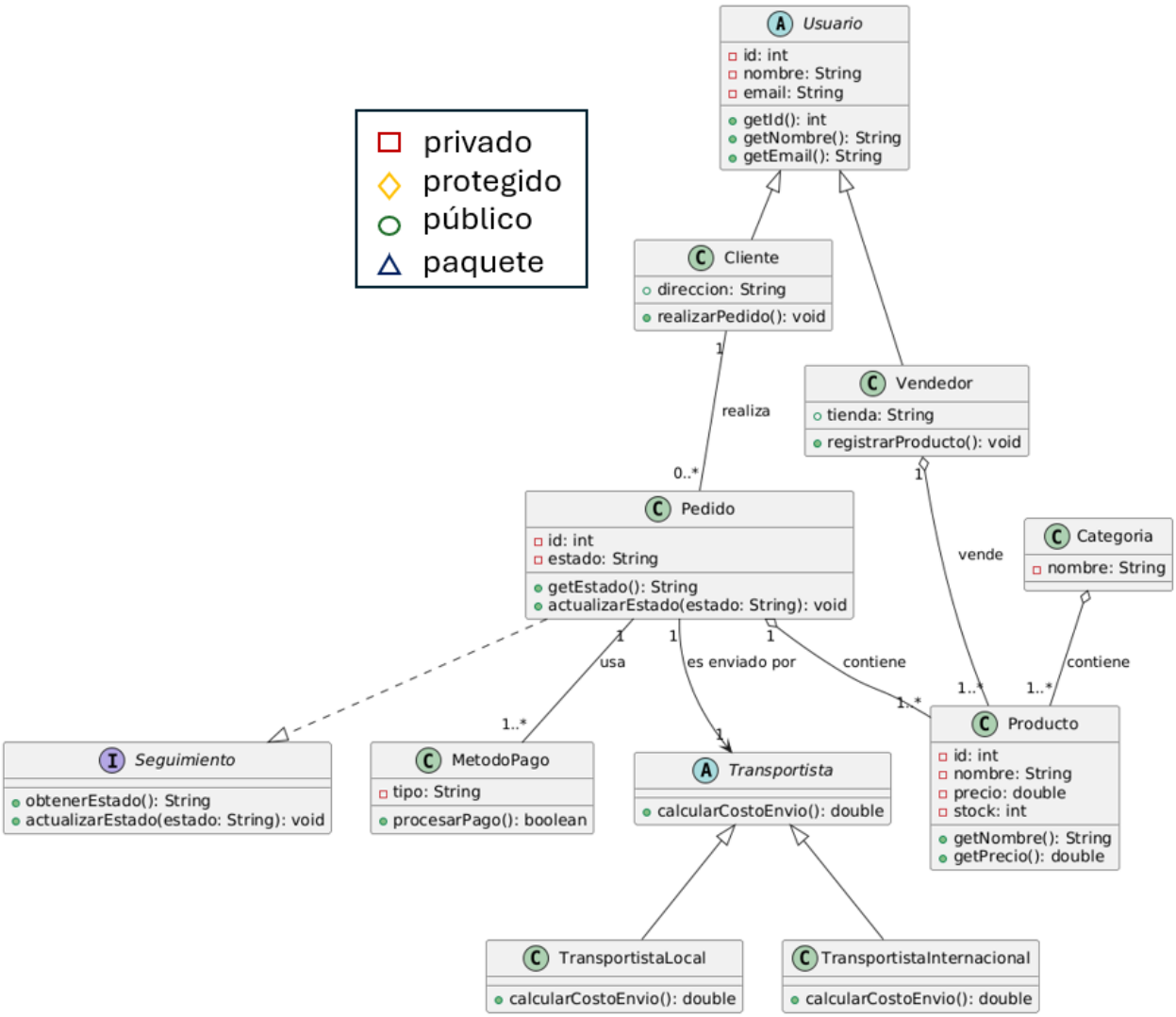
Para procesar los pagos, el sistema debe permitir la utilización de diferentes métodos de pago, asegurando compatibilidad con múltiples opciones disponibles en el mercado. Cada pedido tiene un estado específico, el cual debe actualizarse en función del avance en el proceso de compra y entrega.

Los pedidos requieren un sistema de envíos que permita la distribución de los productos adquiridos. La plataforma debe ofrecer soporte para diferentes tipos de transportistas, cada uno con su propia estrategia de cálculo de costos de envío.

Además, el sistema debe permitir el seguimiento de los pedidos, asegurando que tanto clientes como vendedores puedan conocer el estado de cada transacción en tiempo real. Para esto, los pedidos deben poder actualizar su estado conforme avanza el proceso de compra, pago y entrega.

El diseño del software debe garantizar modularidad, escalabilidad y claridad en las relaciones entre los diferentes elementos del sistema, permitiendo futuras mejoras y expansiones sin afectar la integridad de la plataforma.

SOLUCIÓN EJERCICIO 5 DIAGRAMA DE CLASE UML



EJERCICIO 6

Una fábrica inteligente necesita un sistema de gestión de producción para organizar y supervisar sus procesos de ensamblaje.

La fábrica cuenta con múltiples líneas de ensamblaje, y cada una de ellas está compuesta por varias estaciones de trabajo. Cada estación de trabajo realiza una tarea específica, como ensamblar, pintar o empaquetar.

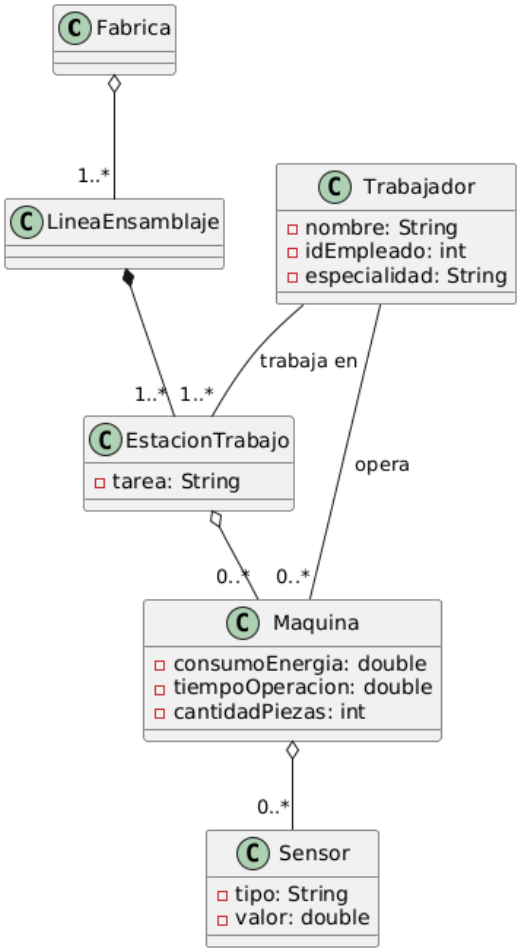
Cada línea de ensamblaje pertenece a una fábrica, pero una línea de ensamblaje puede ser trasladada a otra fábrica en caso de reestructuración. Por otro lado, las estaciones de trabajo son parte integral de una línea de ensamblaje y no pueden existir fuera de ella.

Dentro de una estación de trabajo, se encuentran máquinas que ejecutan las tareas de producción. Cada máquina tiene información sobre su consumo de energía, tiempo de operación y cantidad de piezas producidas. Las máquinas pueden estar equipadas con sensores que monitorean métricas clave como temperatura, presión o velocidad.

Los trabajadores de la fábrica operan estas estaciones de trabajo. Cada trabajador tiene un nombre, un número de identificación y una especialidad (por ejemplo, operador de ensamblaje, supervisor de línea, técnico de mantenimiento).

Los trabajadores pueden estar asignados a múltiples estaciones de trabajo a lo largo del día y pueden operar diferentes máquinas en el transcurso de sus turnos.

SOLUCIÓN EJERCICIO 6 DIAGRAMA DE CLASE UML



EJERCICIOS DIAGRAMA DE COMPONENTES-PAQUETES UML

EJERCICIO 1

En este ejercicio se solicita elaborar un diagrama de componentes UML para una biblioteca universitaria está desarrollando un sistema de biblioteca digital para gestionar el acceso a libros electrónicos, préstamos de documentos y administración de usuarios.

El sistema debe permitir a los usuarios registrados acceder a la plataforma mediante una interfaz web o una aplicación móvil. Desde ahí, pueden buscar libros, visualizar detalles, realizar préstamos y administrar su perfil.

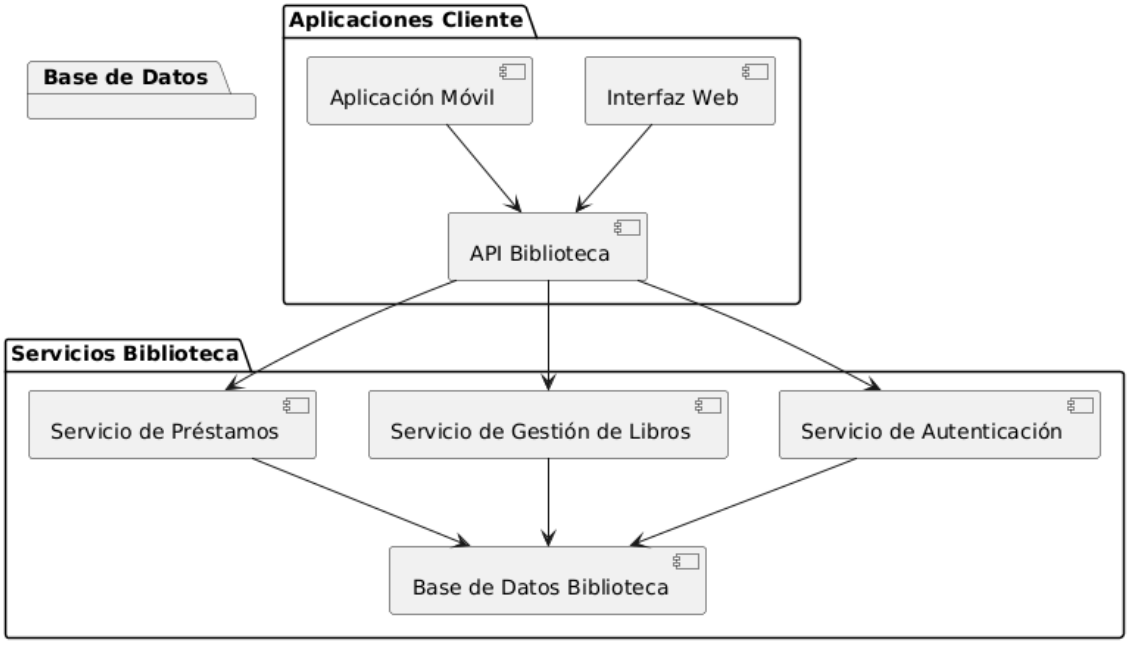
El sistema de la biblioteca está compuesto por varios servicios, entre ellos:

- Un servicio de autenticación, encargado de validar las credenciales de los usuarios.
- Un servicio de gestión de libros, que permite agregar, modificar y eliminar libros del catálogo.
- Un servicio de préstamos, que administra la disponibilidad de los libros y el estado de los préstamos.

Los servicios se comunican con una base de datos central, donde se almacenan la información de los usuarios, los libros disponibles y los registros de préstamo.

La arquitectura del sistema debe permitir la escalabilidad, asegurando que se puedan agregar nuevos servicios en el futuro sin afectar el funcionamiento de los existentes.

SOLUCIÓN EJERCICIO 1 DIAGRAMA DE COMPONENTES-PAQUETES UML



En la capa de presentación, los usuarios pueden acceder a la plataforma a través de una "Interfaz Web" o una "Aplicación Móvil", lo que permite el uso del sistema desde diferentes dispositivos. Ambos clientes dependen de un único componente intermedio, la API Biblioteca, que actúa como puente entre los clientes y los servicios internos del sistema. Todos estos componentes se han recogido en el paquete "Aplicaciones Cliente".

El componente API Biblioteca es el encargado de recibir las solicitudes de los usuarios y redirigirlas a los servicios adecuados. Esta estructura permite que cualquier modificación en la lógica de negocio no afecte directamente a los clientes, manteniendo la compatibilidad con futuras actualizaciones.

El sistema está compuesto por tres servicios principales dentro de la capa de negocio. El Servicio de Autenticación se encarga de validar las credenciales de los usuarios antes de que puedan acceder a la plataforma. El Servicio de Gestión de Libros permite a los

administradores de la biblioteca agregar, modificar y eliminar libros del catálogo, así como actualizar su disponibilidad. El Servicio de Préstamos gestiona la asignación de libros a los usuarios, verificando la disponibilidad de los ejemplares y actualizando el estado de los préstamos en función de las devoluciones.

Todos los servicios interactúan con una Base de Datos Biblioteca, en la que se almacenan los registros de usuarios, el catálogo de libros y los datos de los préstamos. La centralización de la información permite garantizar la integridad de los datos y facilita la administración de la biblioteca digital.

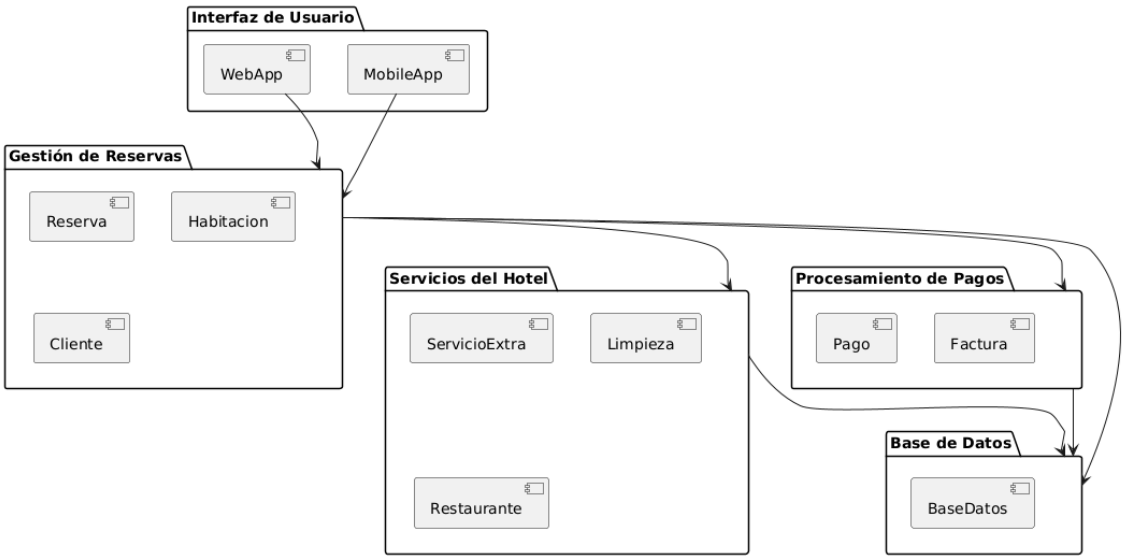
El diseño modular del sistema permite su escalabilidad, ya que nuevos servicios pueden ser agregados en el futuro sin afectar el funcionamiento de los componentes existentes. Además, la separación entre la API y los servicios internos garantiza que cualquier cambio en la lógica de negocio pueda realizarse sin modificar la interacción con los clientes. Esta estructura permite que el sistema evolucione sin comprometer la estabilidad ni la seguridad de la plataforma

EJERCICIO 2

Un hotel está desarrollando un sistema de gestión de reservas para automatizar la administración de habitaciones, clientes y pagos. El sistema debe permitir que los clientes reserven habitaciones, gestionen sus datos personales y realicen pagos en línea, junto con la emisión de su correspondiente factura. Además, el personal del hotel debe poder administrar las reservas, verificar la disponibilidad de habitaciones y gestionar los servicios adicionales (limpieza, restaurante, otros extras, etc.) que los huéspedes pueden solicitar.

El sistema está compuesto por diferentes módulos que interactúan entre sí para garantizar un funcionamiento eficiente y seguro. La estructura debe permitir la separación de responsabilidades, asegurando que cada funcionalidad del sistema se encapsule en su propio paquete para facilitar la escalabilidad y el mantenimiento del software.

SOLUCIÓN EJERCICIO 2 DIAGRAMA DE COMPONENTES-PAQUETES UML



En la capa de presentación, se han definido dos paquetes principales: Interfaz de Usuario, que contiene la WebApp y la MobileApp. Estos componentes representan las interfaces a través de las cuales los clientes y el personal del hotel interactúan con el sistema.

El paquete Gestión de Reservas se encarga de administrar las habitaciones, clientes y reservas dentro del hotel. Es el núcleo del sistema, ya que gestiona la disponibilidad de habitaciones, la información de los clientes y el control de las reservas activas. Este paquete se comunica con la Base de Datos para almacenar y recuperar información.

El paquete Procesamiento de Pagos contiene los componentes necesarios para gestionar los pagos de los clientes y la generación de facturas. Este paquete está separado de la gestión de reservas para garantizar que las operaciones financieras sean independientes y puedan manejarse con seguridad y trazabilidad.

El paquete Servicios del Hotel gestiona los servicios adicionales que los huéspedes pueden solicitar, como servicio de limpieza, acceso a restaurantes y otros servicios extra. Este paquete está vinculado al sistema de reservas para garantizar que los clientes puedan añadir servicios a sus reservas y que el hotel pueda administrarlos de manera eficiente.

Finalmente, el paquete Base de Datos centraliza toda la información del sistema, garantizando que los datos sean accesibles y gestionados de manera eficiente por los diferentes módulos. Tanto el procesamiento de pagos como la gestión de reservas y los servicios del hotel dependen de este paquete para almacenar y recuperar información de manera confiable.

EJERCICIOS TEMA 6: ELICITACIÓN DE REQUISITOS. CASOS DE USO

EJERCICIO 1

Un hotel desea implementar un sistema que permita a los clientes buscar habitaciones disponibles según la fecha y tipo de habitación, realizar reservas proporcionando sus datos personales y de pago, así como modificar o cancelar sus reservas si es necesario.

El sistema también deberá permitir el procesamiento de pagos de manera segura y eficiente.

Por otro lado, los administradores del hotel podrán gestionar el inventario de habitaciones, actualizar la disponibilidad y tarifas, y generar informes de ocupación para evaluar el rendimiento del hotel.

El sistema mostrará confirmaciones y notificaciones al cliente vía correo electrónico tras realizar o modificar una reserva.

Se pide:

A. Identificar los requisitos funcionales y no funcionales del sistema. En el caso de estos últimos, especificar en base a opinión sobre el sistema.
B. Dibujar el diagrama de casos de uso.

SOLUCIÓN EJERCICIO 1 – APARTADO A

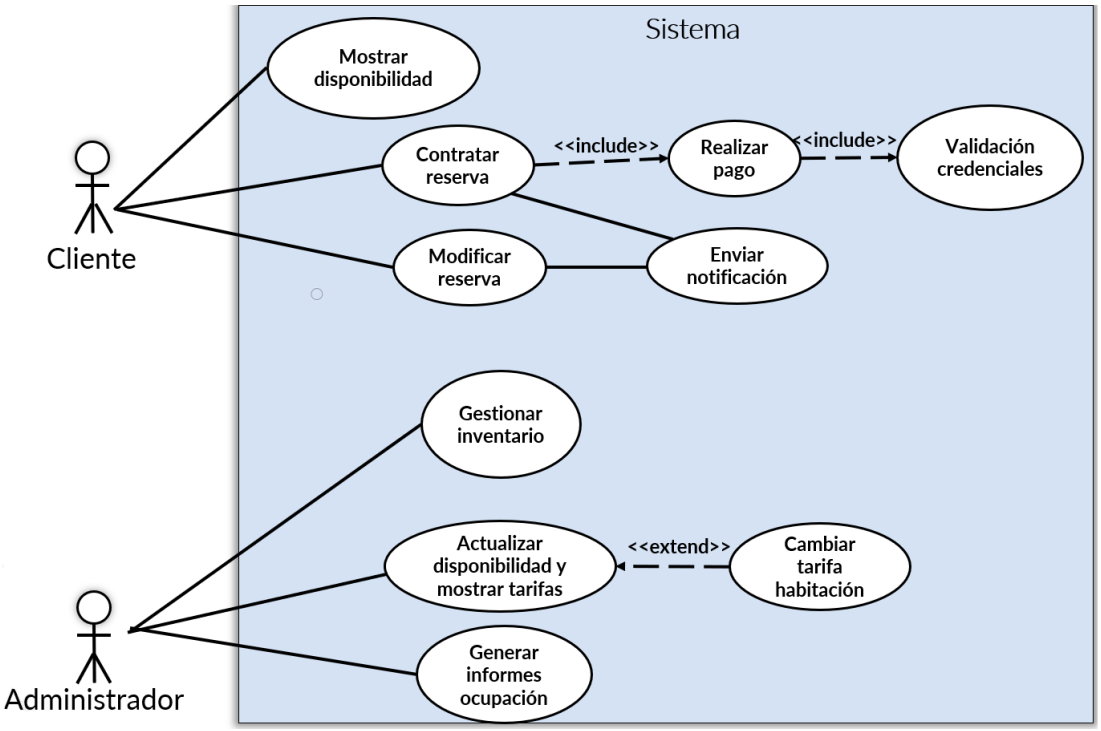
Requerimientos Funcionales:

- RF-001: Mostrar disponibilidad de habitaciones al cliente.
- RF-002: Permitir al cliente realizar una reserva proporcionando sus datos.
- RF-003: Modificar o cancelar una reserva ya realizada.
- RF-004: Procesar pagos de las reservas mediante tarjeta de crédito o débito.
- RF-005: Permitir al administrador actualizar la disponibilidad de las habitaciones.
- RF-006: Generar informes de ocupación para el administrador.

Requerimientos No Funcionales:

- RNF-001: El sistema debe responder en menos de 5 segundos para las búsquedas de disponibilidad.
- RNF-002: Interfaz amigable y fácil de usar para clientes y administradores.
- RNF-003: Seguridad de los datos personales y de pago mediante cifrado.
- RNF-004: Disponibilidad del sistema las 24h del día los 7 días de la semana, con un tiempo de inactividad menor al 1% mensual.
- RNF-005: Adaptabilidad a diferentes dispositivos, como móviles y tablets.

SOLUCIÓN EJERCICIO 1 – APARTADO B



EJERCICIO 2

Un centro deportivo desea implementar un sistema de gestión que facilite a socios, entrenadores y personal administrativo realizar, coordinar y supervisar procesos relacionados con las actividades y el uso de las instalaciones. El sistema debe permitir a los socios reservar clases, consultar horarios de actividades, realizar pagos de la matrícula por acceder al centro y solicitar servicios adicionales. Los entrenadores deben poder gestionar las inscripciones a sus clases, registrar la asistencia y planificar sesiones de entrenamiento personal. El sistema también debe ofrecer funcionalidades de generación de reportes para el personal administrativo, así como alertas automáticas para pagos pendientes o bajas asistencias.

Se pide:

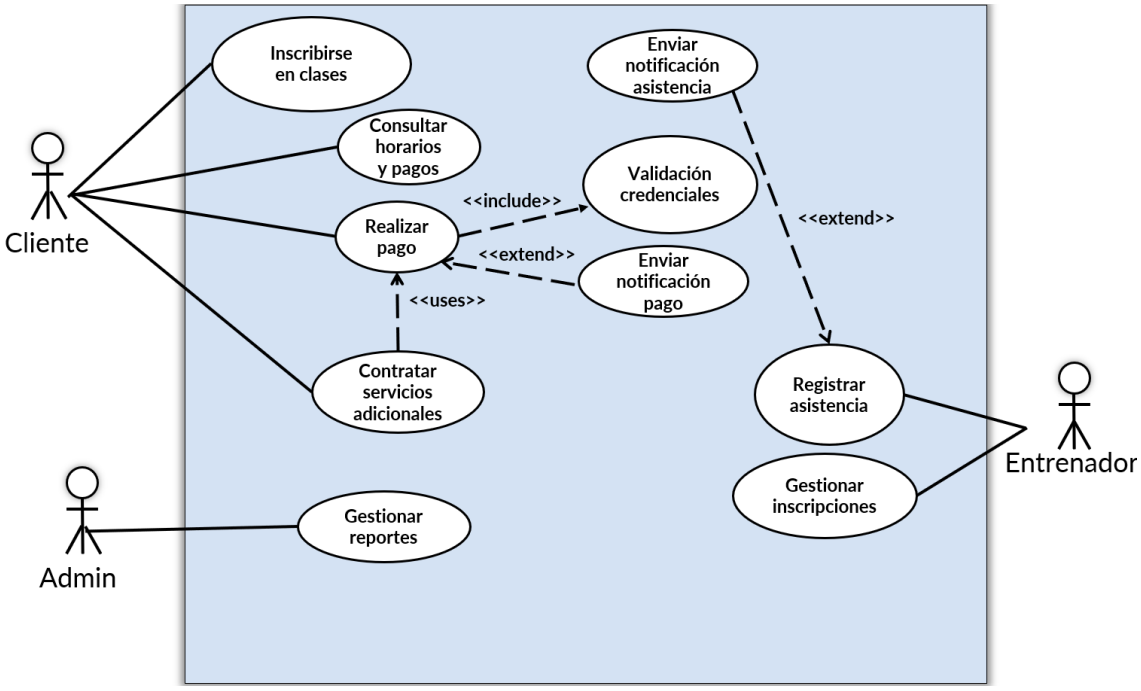
A. Matriz de requisitos

B. Diagrama de casos de uso

SOLUCIÓN EJERCICIO 2 – APARTADO A

ID	REQUISITO	TIPO	PRIORIDAD	ACTOR
R01	Los socios pueden inscribirse en clases y actividades.	Funcional	Alta	Socios
R02	Los socios pueden consultar sus horarios, estado de pagos y asistencia.	Funcional	Alta	Socios
R03	Los socios pueden realizar pagos de matrícula en línea.	Funcional	Alta	Socios
R04	Los socios pueden solicitar servicios adicionales.	Funcional	Media	Socios
R05	Los entrenadores pueden gestionar inscripciones a sus clases.	Funcional	Alta	Entrenadores
R06	Los entrenadores pueden registrar la asistencia de los socios.	Funcional	Alta	Entrenadores
R07	El sistema permite generar reportes de asistencia y pagos.	Funcional	Alta	Administradores
R08	El sistema envía alertas de pagos pendientes a los socios.	Funcional	Alta	Sistema
R09	El sistema envía alertas de bajas asistencias a los entrenadores.	Funcional	Media	Sistema
R10	Los socios deben tener pagos al día para acceder a ciertos servicios.	Regla de negocio	Alta	Sistema

SOLUCIÓN EJERCICIO 2 – APARTADO B



EJERCICIOS TEMA 9: DESARROLLO DE DSL TEXTUALES

EJERCICIO 1

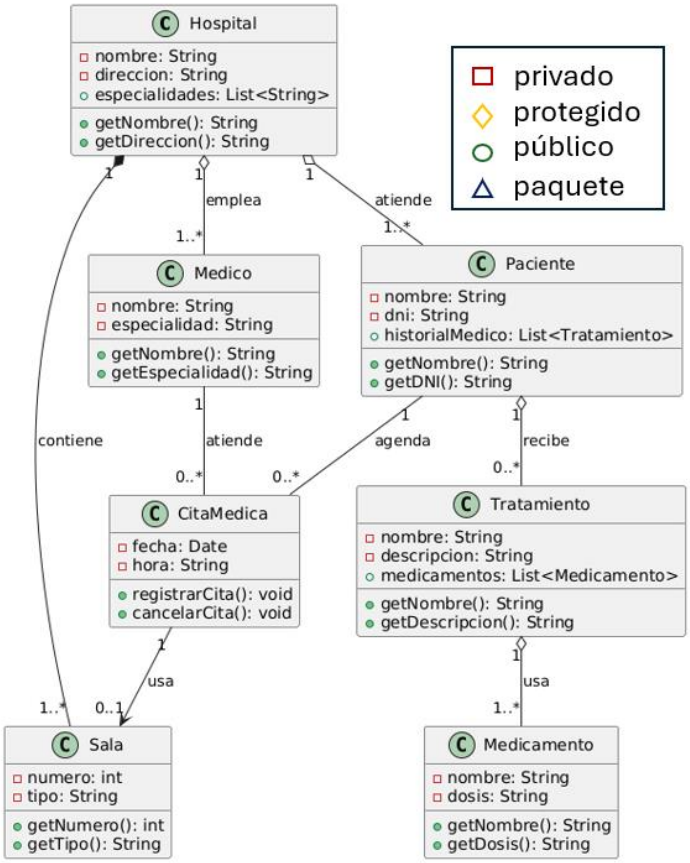
En este ejercicio se solicita diseñar una gramática en Xtext para representar un sistema hospitalario, incluyendo la gestión de hospitales, médicos, pacientes, citas médicas, salas, tratamientos y medicamentos.

El sistema debe permitir la definición de hospitales, los cuales pueden emplear múltiples médicos y atender a varios pacientes. Cada médico debe tener una especialidad y poder atender a pacientes mediante citas médicas, que estarán asociadas a una fecha, hora y a una sala específica en caso de que se requiera.

Cada paciente debe contar con un identificador único (DNI) y un historial médico que almacene los tratamientos que ha recibido. Los tratamientos pueden incluir una lista de medicamentos, especificando la dosis requerida.

El modelo debe reflejar correctamente las relaciones entre estas entidades, permitiendo representar de manera estructurada la gestión de un hospital y sus operaciones básicas.

Vista del diagrama de clases del sistema hospitalario:



SOLUCIÓN EJERCICIO 1

```
grammar org.xtext.SistemaHospital with
org.eclipse.xtext.common.Terminals

generate SistemaHospital "http://www.xtext.org/SistemaHospital"

Model:
    hospitales+=Hospital*
    medicos+=Medico*
    pacientes+=Paciente*
    citas+=CitaMedica*
    tratamientos+=Tratamiento*
    salas+=Sala*
    medicamentos+=Medicamento*;

Hospital:
    'Hospital' '{'
        'nombre:' nombre=STRING
        'direccion:' direccion=STRING
        'especialidades:' '[' especialidades+=STRING (','
especialidades+=STRING) * ']'
        'medicos:' '[' medicos+=[Medico] (',' medicos+=[Medico])*
']'
        'pacientes:' '[' pacientes+=[Paciente] (','
pacientes+=[Paciente]) * ']'
    '}';

Medico:
    'Medico' '{'
        'nombre:' nombre=STRING
        'especialidad:' especialidad=STRING
        'citas:' '[' citas+=[CitaMedica]* ']'
    '}';

Paciente:
    'Paciente' '{'
        'nombre:' nombre=STRING
        'dni:' dni=STRING
        'historialMedico:' '{' historialMedico+=Tratamiento* '}'
        'citas:' '[' citas+=[CitaMedica]* ']'
    '}';

CitaMedica:
    'CitaMedica' '{'
        'fecha:' fecha=STRING
        'hora:' hora=STRING
        'medico:' medico=[Medico]
        'paciente:' paciente=[Paciente]
        ('sala:' sala=[Sala])?
    '}';

Sala:
    'Sala' '{'
```

```
        'numero:' numero=INT
        'tipo:' tipo=STRING
    '}';

Tratamiento:
    'Tratamiento' '{'
        'nombre:' nombre=STRING
        'descripcion:' descripcion=STRING
        'medicamentos:' '{' medicamentos+=[Medicamento]* '}'
    '}';

Medicamento:
    'Medicamento' '{'
        'nombre:' nombre=STRING
        'dosis:' dosis=STRING
    '}';
```

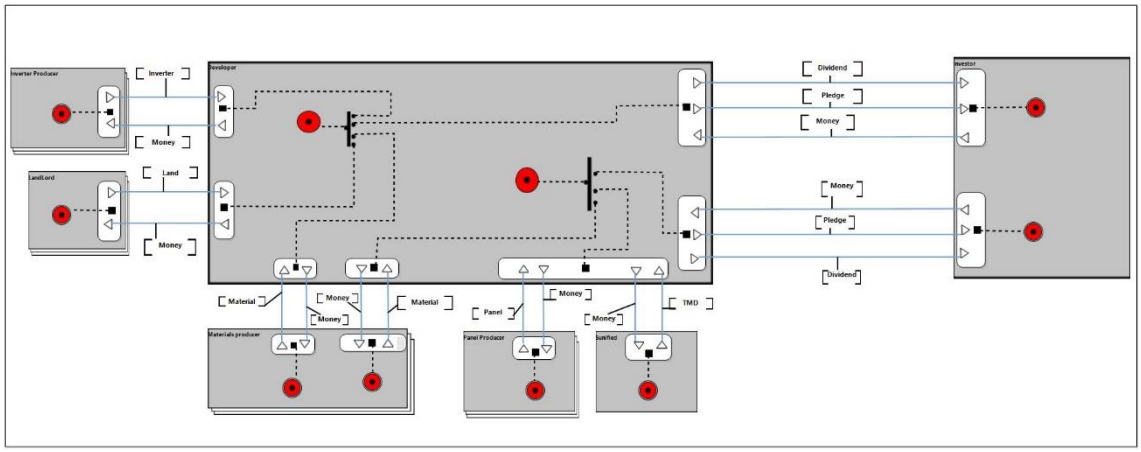
EJERCICIO 2

En este ejercicio se solicita diseñar una gramática en Xtext para representar un modelo e³value bajo una notación textual.

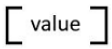
La notación e³value, constituye una herramienta para el modelado empresarial y de valor que permite la representación gráfica de ideas de negocio en las que intervienen diferentes actores o grupos de ellos. El foco de dicha notación es reflejar las actividades de valor, entendidas como aquellas acciones que uno o varios actores realizan con el fin de obtener beneficios, así como los intercambios de valor que se generan entre ellos, en los que cada actor intercambia diferentes objetos de valor que, dependiendo de su naturaleza, pueden ser bienes, dinero o incluso servicios.

El modelo e³value se centra en los diferentes actores (segmentos de mercado), las acciones o actividades que realizan para generar valor, y los intercambios que se producen entre ellos

Vista de un modelo e³value:



Leyenda de los elementos de la notación e^{3}value:

Elemento	*Representación*	*Descripción*
Actor		Un actor es una entidad individual que participa en el modelo de negocio. Por ejemplo: una compañía, un organismo público, etc.
Market Segment		El segmento de mercado (Market Segment) representa a un grupo de actores. Por ejemplo: clientes, repartidores, restaurantes.
Value Activity		Representan actividades que generan beneficios por parte de los actores. Se representan dentro de estos.
Start Stimulus		Refleja la necesidad de usuario que da inicio al intercambio de valor.
End Stimulus		Delimita la representación no mostrando más detalles sobre cómo se lleva a cabo la prestación del servicio.
Value Interface		Refleja el punto en el que se producen intercambios de valor entre actores o entre actividades de valor de un mismo actor.
Value Port		Refleja la dirección de un objeto de valor en un intercambio de valor.
Value Exchange		Representa los canales a través de los que se lleva a cabo un intercambio de objetos de valor.
Value Object		Representa los bienes, productos o servicios intercambiados por los actores.
Dependency path		Permiten enlazar los diferentes elementos contenidos en los actores.
AND Dependency		Permite representar el operador booleano AND cuando se combinan varias actividades en un actor.
OR Dependency		Permite representar el operador booleano OR en la realización de varias actividades de un actor.

SOLUCIÓN EJERCICIO 2

```
grammar org.xtext.E3ValueNotation with
org.eclipse.xtext.common.Terminals

generate e3Value "http://www.xtext.org/E3ValueNotation"

Model:
    elementsNotation+=Element*;

Element:
    Actor | MarketSegment | ValueActivity | StartStimulus |
EndStimulus | ValueInterface | ValuePort | ValueExchange |
ValueObject | Dependency;

Actor:
```

```
    'Actor' name=ID;

MarketSegment:
    'MarketSegment' name=ID;

ValueActivity:
    'ValueActivity' name=ID;

StartStimulus:
    'StartStimulus' name=ID;

EndStimulus:
    'EndStimulus' name=ID;

ValueInterface:
    'ValueInterface' name=ID;

ValuePort:
    'ValuePort' name=ID;

ValueExchange:
    'ValueExchange' name=ID 'from' source=[ValuePort] 'to'
target=[ValuePort];

ValueObject:
    'ValueObject' name=ID;

Dependency:
    AndDependency | OrDependency | DependencyPath;

AndDependency:
    'AND' '(' dependencies+=[ValueActivity|ValueInterface|ValuePort]
(',' dependencies+=[ValueActivity|ValueInterface|ValuePort]) * ')';

OrDependency:
    'OR' '(' dependencies+=[ValueActivity|ValueInterface|ValuePort]
(',' dependencies+=[ValueActivity|ValueInterface|ValuePort]) * ')';

DependencyPath:
    'DependencyPath' '(' elements+=[Element] (','
elements+=[Element]) * ')';
```

EJERCICIO 3

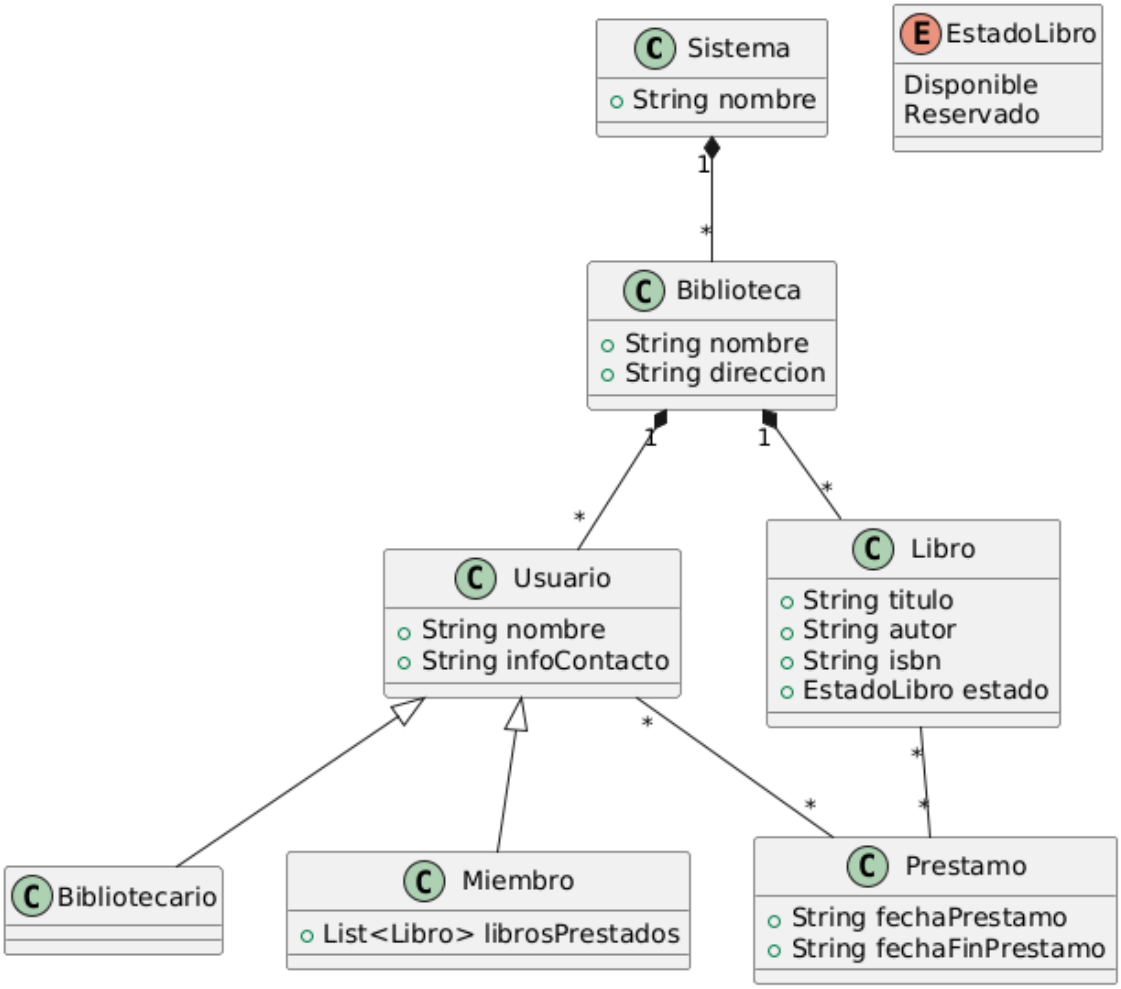
En este ejercicio se solicita diseñar el diagrama de clases derivado de una gramática en Xtext para representar un sistema de gestión de bibliotecas públicas. El diagrama de clases tiene que representar las clases, atributos y las relaciones entre ellas (asociaciones, multiplicidades, etc.) que se pueden derivar de la gramática.

La gramática de la que se tiene que partir para representar el diagrama de clases es:

```
grammar org.xtext.Biblioteca with org.eclipse.xtext.common.Terminals

generate biblioteca "http://www.xtext.org/Biblioteca"

Model:
    'Sistema' name=ID '{'
        bibliotecas+=Biblioteca*
    '}';

Biblioteca:
    'Biblioteca' name=ID '{'
        direccion=STRING
        libros+=Libro*
        usuarios+=Usuario*
    '}';

Usuario:
    Bibliotecario | Miembro;

Bibliotecario:
    'Bibliotecario' name=ID '{'
        infoContacto=STRING
    '}';

Miembro:
    'Miembro' name=ID '{'
        infoContacto=STRING
        librosPrestados+=Libro*
    '}';

Libro:
    'Libro' title=STRING '{'
        autor=STRING
        isbn=STRING
        estado=EstadoLibro
    '}';

EstadoLibro:
    'Disponible' | 'Reservado';

Prestamo:
    'Prestamo' '{'
        usuario=[Usuario]
        libro=[Libro]
        fechaPrestamo=STRING
        fechaFinPrestamo=STRING
    '}';
```

SOLUCIÓN EJERCICIO 3

El diagrama de clases derivado de la gramática Xtext anterior es el siguiente:



Universidad
Rey Juan Carlos

PRÁCTICAS

GRADO EN MATEMÁTICAS
CURSO:2024/2025

Cristian Gómez Macías

PRÁCTICA 1-A DESARROLLO DE DSL TEXTUALES

Modelado de páginas web con CSS y JavaScript usando Xtext

1. DESCRIPCIÓN

El objetivo de esta práctica es crear una gramática en **Xtext** utilizando el entorno de desarrollo **Sirius** para modelar páginas web de manera estructurada. El lenguaje desarrollado permitirá definir, en formato textual, la estructura de una página web, incluyendo HTML, CSS y JavaScript. Además, el sistema ofrecerá herramientas para validar y editar el código, asegurando su correcta configuración y funcionamiento.

2. OBJETIVOS DE LA PRÁCTICA

Desarrollar un DSL textual mediante la definición de la gramática en Xtext para modelar páginas web en un lenguaje de dominio específico (DSL), incluyendo la estructura HTML, las hojas de estilo CSS y los scripts en JavaScript. Además de elaborar la gramática, el sistema debe proporcionar herramientas avanzadas de edición, como coloreado de sintaxis, autocompletado, validaciones y mecanismos de corrección rápida (quickfixes), de forma que ayuden a mejorar la productividad y la calidad del código generado.

3. REQUISITOS DEL PROYECTO

El proyecto debe cumplir los siguientes requisitos, a los que sigue su correspondiente ponderación máxima en el cómputo de la nota final de la práctica:

Elaboración de la gramática Xtext **(4,5 PUNTOS)**: Se debe modelar una página web que incluya:

- Estructura HTML con elementos básicos (div, p, h1-h6, a, img, ul, ol, li, table).
- Hojas de estilo CSS que permitan la definición de clases y reglas de estilo.
- Scripts en JavaScript para agregar interactividad y definir funciones que interactúen con los elementos HTML.
- La gramática debe permitir una jerarquía anidada de elementos.

Coloreado de la sintaxis **(0,5 PUNTOS)**: Se tiene que implementar un esquema de resaltado de sintaxis (coloreado de la fuente) para mejorar la legibilidad del código, de forma que:

- Etiquetas HTML en un color específico.
- Propiedades CSS en otro color.
- Funciones y variables JavaScript resaltadas para facilitar su identificación.

Descriptor de etiquetas del modelo **(0,5 PUNTOS)**: Implementar un mecanismo de documentación contextual, permitiendo visualizar información relevante sobre cada elemento cuando el usuario pase el cursor sobre una etiqueta.

Autocompletado **(0,5 PUNTOS)**: Incluir una funcionalidad de autocompletado para sugerir nombres de etiquetas HTML, clases CSS y funciones JavaScript mientras el usuario edita el código.

Mecanismos de QuickFix **(1 PUNTO)**: Implementar correcciones automáticas para errores comunes en la escritura del código como:

- Cierre automático de etiquetas HTML si faltan en la estructura.
- Corrección de errores en nombres de propiedades CSS.

Validaciones **(1 PUNTO)**: Implementar validaciones para:

- Errores en la jerarquía HTML (por ejemplo, que un <li> esté dentro de un <ul> o <ol>).
- Propiedades CSS incorrectas o valores no válidos.
- Estructura de funciones JavaScript para verificar que tengan un nombre y un cuerpo válido.

Memoria en formato PDF **(2 PUNTOS)** que contenga la explicación de la gramática definida. A su vez, la memoria tiene que contener capturas de pantalla con las pruebas realizadas.

4. FORMATO DE ENTREGA

Se deberá entregar un archivo comprimido por la página de la asignatura en Aula Virtual que contenga:

- Memoria en formato PDF.
- Código fuente del proyecto Xtext o plugin para instalar el DSL textual elaborado.
- Al menos dos modelos textuales elaborados con el DSL que representen páginas web incluyendo aspectos tanto de HTML, CSS y JS.

5. FECHA DE ENTREGA

Se deberá de entregar antes de las 23:30 del día 21 de abril de 2025.

6. MATERIALES RECOMENDADOS

Tema 8: Introducción al Metamodelado.

Tema 9: Desarrollo de DSL textuales.

Vídeos de elaboración de DSL textuales.

7. MODELO TEXTUAL DE PRUEBA

A continuación, se ofrece un modelo textual de prueba como punto de referencia a la hora de elaborar la gramática solicitada:

```
WebPage {
    title: "Mi Página Web"

    body: {
        Div {
            class: "contenedor"
            content: {
                H1 {
                    text: "Hola a todos"
                }
                P {
                    text: "Este es un ejemplo de página modelada con
Xtext."
                }
                A {
                    href: "https://www.ejemplo.com"
                    text: "Haz clic aquí"
                }
            }
        }
    }

    styles: {
        CSSRule {
            selector: ".contenedor"
            properties: {
                Property {name: "background-color" value:
"lightgray"}
                Property {name: "padding" value: "20px"}
            }
        }
    }

    scripts: {
        JSFunction {
            name: "mostrarMensaje"
            body: {
                Var mensaje = "¡Hola, mundo!";
                Call console.log(mensaje);
            }
        }
    }
}
```

PRÁCTICA 1-B DESARROLLO DE DSL TEXTUALES

Modelado de bases de datos relacionales usando Xtext

1. DESCRIPCIÓN

En esta práctica se solicita definir una gramática en Xtext para modelar bases de datos relacionales mediante un lenguaje de dominio específico textual (DSL). Este DSL permitirá definir las entidades, atributos, claves primarias, claves foráneas y relaciones de un modelo de base de datos. Además, se proporcionará soporte para la validación, edición y generación de código SQL a partir del modelo definido.

2. OBJETIVOS DE LA PRÁCTICA

El objetivo principal de la práctica es desarrollar un lenguaje de modelado en Xtext que permita representar diagramas de entidad-relación de bases de datos de manera estructurada. Además de la construcción de la gramática, el sistema debe incluir herramientas avanzadas de edición, como coloreado de sintaxis, autocompletado, validaciones y mecanismos de corrección rápida (quickfixes), de forma que el modelo facilite la generación de bases de datos bien estructuradas.

3. REQUISITOS DEL PROYECTO

El proyecto debe cumplir con los siguientes requisitos:

Elaboración de la gramática en Xtext **(4.5 PUNTOS)**: El modelo debe permitir la definición de bases de datos con:

- Entidades con atributos, tipos de datos y restricciones.
- Claves primarias y claves foráneas para establecer relaciones.
- Relaciones entre entidades, indicando cardinalidad (1-1, 1-N, N-M).

Se debe poder definir restricciones de integridad, como valores únicos, claves no nulas y valores por defecto.

Coloreado de la sintaxis **(0.5 PUNTOS)**: Implementar un esquema de coloreado para mejorar la legibilidad, diferenciando:

- Palabras clave del DSL.
- Nombres de entidades y atributos.
- Tipos de datos y restricciones.

Descriptor de etiquetas del modelo **(0.5 PUNTOS)**: Incluir un mecanismo que proporcione información detallada sobre cada entidad, relación o atributo al pasar el cursor sobre ellos.

Autocompletado **(0.5 PUNTOS)**: Implementar sugerencias de autocompletado para nombres de entidades, atributos y tipos de datos durante la edición del modelo.

Mecanismos de QuickFix **(1 PUNTO)**: Implementar correcciones automáticas para errores comunes, tales como:

- Falta de claves primarias en una entidad.
- Referencias incorrectas en claves foráneas.
- Errores en la definición de relaciones.

Validaciones **(1 PUNTO)**: Implementar validaciones para:

- Nombres de entidades únicos dentro del modelo.
- Tipos de datos correctos en los atributos.
- Consistencia en las claves foráneas, asegurando que hagan referencia a una entidad existente.

Memoria en formato PDF **(2 PUNTOS)** que contenga la explicación de la gramática definida. A su vez, la memoria tiene que contener capturas de pantalla con las pruebas realizadas.

4. FORMATO DE ENTREGA

Se deberá entregar un archivo comprimido por la página de la asignatura en Aula Virtual que contenga:

- Memoria en formato PDF.
- Código fuente del proyecto Xtext o plugin para instalar el DSL textual elaborado.
- Al menos dos modelos textuales elaborados con el DSL que representen elementos del diagrama Entidad Relación de forma correcta y coherente.

5. FECHA DE ENTREGA

Se deberá de entregar antes de las 23:30 del día 21 de abril de 2025.

6. MATERIALES RECOMENDADOS

Tema 8: Introducción al Metamodelado.

Tema 9: Desarrollo de DSL textuales.

Vídeos de elaboración de DSL textuales.

7. MODELO TEXTUAL DE PRUEBA

A continuación, se ofrece un modelo textual de prueba como punto de referencia a la hora de elaborar la gramática solicitada:

```
Database {
    name: "SistemaVentas"

    entities: {
```

```
        Entity Cliente {
            attributes: {
                Attribute id INT PRIMARY_KEY
                Attribute nombre STRING NOT_NULL
                Attribute email STRING UNIQUE
            }
        }

        Entity Producto {
            attributes: {
                Attribute id INT PRIMARY_KEY
                Attribute nombre STRING NOT_NULL
                Attribute precio FLOAT NOT_NULL
            }
        }

        Entity Pedido {
            attributes: {
                Attribute id INT PRIMARY_KEY
                Attribute fecha DATE NOT_NULL
                Attribute cliente_id INT FOREIGN_KEY Cliente.id
            }
        }
    }

    relationships: {
        Relationship Cliente_Pedido {
            entity1: Cliente
            entity2: Pedido
            type: "1-N"
        }
    }
}
```

PRÁCTICA 2 DESARROLLO DE DSL VISUALES Y TRANSFORMACIONES

Desarrollo de un DSL Visual con Sirius para Diagramas de Estados UML

1. DESCRIPCIÓN

En el Lenguaje Unificado de Modelado (UML) se recoge los diagramas de estados para modelar el comportamiento dinámico de un sistema. Estos diagramas representan los diferentes estados por los que pasa un sistema o una entidad, así como las transiciones entre ellos, permitiendo visualizar su flujo de ejecución.

En esta práctica, se utilizará **Eclipse Sirius** para desarrollar un DSL visual que permita la creación y edición de diagramas de estados UML, basado en un metamodelo Ecore.

Además, se requiere la generación automática de código a partir del modelo visual. Para ello, se utilizará **ATL** o **Xtend**, permitiendo transformar el modelo en un lenguaje de programación adecuado, como Java, Python o C++.

El objetivo de esta práctica es diseñar un editor gráfico que permita la creación visual de estados, transiciones, acciones internas (ENTRY, DO, EXIT) y variables, proporcionando herramientas para la validación y edición del modelo, así como la transformación del modelo a código funcional.

2. OBJETIVO DE LA PRÁCTICA

El objetivo principal de la práctica es desarrollar un DSL visual para la creación de diagramas de estados UML, utilizando Eclipse Sirius y EMF.

El sistema debe permitir la creación y edición visual de diagramas de estados, en los que los usuarios puedan definir estados, transiciones, acciones internas y variables asociadas a los estados.

Además, el editor debe incluir funcionalidades avanzadas como validaciones, autocompletado y correcciones automáticas (quickfixes) para mejorar la calidad y coherencia de los modelos generados.

Por último, se debe implementar una transformación automática de modelos a código fuente mediante ATL o Xtend, permitiendo la ejecución del diagrama de estados en un lenguaje de programación específico.

3. REQUISITOS DEL PROYECTO

El proyecto debe cumplir con los siguientes requisitos:

1. Definición del metamodelo Ecore **(2 PUNTOS)**: Crear un metamodelo Ecore en Eclipse EMF, que contenga:

    - Estado, con atributos como nombre, identificador y una lista de variables internas.
    - Transición, con condiciones para activarse y el estado de destino.
    - Acciones internas, representando ENTRY, DO y EXIT para definir el comportamiento de cada estado.
    - Variables, que pueden ser modificadas en cada transición o dentro de los estados.

    El metamodelo debe ser coherente con las reglas de los diagramas de estados UML.

2. Implementación del DSL visual en Sirius **(3.5 PUNTOS)**:

    - Desarrollar un editor gráfico en Eclipse Sirius que permita la manipulación visual de los elementos del diagrama.
    - Proporcionar herramientas para crear estados, transiciones y definir acciones internas de manera intuitiva.
    - Definir representaciones gráficas para cada elemento UML.

3. Autocompletado y asistencia en la edición **(1 PUNTO)**:

    - Sugerir nombres de estados y variables mientras el usuario edita el modelo.
    - Permitir la creación automática de transiciones entre estados conectados.

4. Validaciones en tiempo de diseño: Implementar reglas para evitar errores comunes, como **(1 PUNTO)**:

    - Estados sin transiciones de entrada o salida.
    - Transiciones sin una condición válida.
    - Estados sin acciones internas definidas.

5. Mecanismos de QuickFix para corrección de errores **(1 PUNTO)**:

    - Ofrecer sugerencias automáticas para corregir problemas detectados en el modelo.
    - Proporcionar acciones rápidas, como agregar transiciones obligatorias o completar estados sin acciones.

6. Exportación del modelo **(1.5 PUNTOS)**: Permitir la exportación del diagrama en formato XMI para su integración con otras herramientas UML o su conversión a código a Java, Python, etc.

4. FORMATO DE ENTREGA

Se deberá entregar un archivo comprimido por la página de la asignatura en Aula Virtual que contenga:

- Memoria en formato PDF.
- Código fuente del proyecto Xtext o plugin para instalar el DSL visual elaborado.
- Al menos dos modelos visuales para el diagrama de Estados elaborados con el DSL que simulen las diferentes situaciones por las que pasa un sistema de forma correcta y coherente.

5. FECHA DE ENTREGA

Se deberá de entregar antes de las 23:30 del día 22 de mayo de 2025.

6. MATERIALES RECOMENDADOS

Tema 8: Introducción al Metamodelado.

Tema 10: Desarrollo de DSL visuales.

Tema 11: Transformaciones entre modelos.

Apuntes de Sirius.