

República Bolivariana de Venezuela
Ministerio del Poder Popular para la Educación Universitaria
Universidad Nacional Experimental de las Telecomunicaciones e Informática (UNETI)
Unidad Curricular: Proyecto Socio-Tecnológico II
Sesión Didáctica I



Evaluación I:
Construcción de la documentación de un proyecto

Ingeniera en formación:

- Andrés Mora – V-32.297.424
- Alex Suárez – V-12.342.934
- Paula Unda – V-32.139.35

Sección: 8B

Prof. Guía: Yuly Delgado

Construcción de la documentación de un proyecto

Aporte Alex Suarez, Andrés Mora y Paula Unda

Gestión del Proyecto

1. Plan de Desarrollo de Software.
2. Planes de Iteraciones (uno por cada iteración definida).
3. Lista de Riesgos y estrategia de mitigación.
4. Glosario del proyecto

Requisitos

1. Documento de Especificaciones del Sistema (DES)
2. Lista de requisitos priorizados del usuario (Product Backlog).
3. Lista de tareas por iteración o sprint (Sprint Backlog).
4. Documento de gestión de requisitos por iteración o sprint.
5. Especificación de Requisitos de Software (ERS) / Modelo de Casos de Uso.

Análisis y Diseño

1. Vista de Casos de Uso a implementar
2. Descripción detallada de arquitectura
3. Modelo de Análisis.
4. Modelo de Diseño.

Entregables de gestión y requisitos para SIEDUCRES v1.0

Este paquete consolida planificación, riesgos y especificaciones de requisitos del proyecto “Plataforma Web como Apoyo Académico ante Interrupciones por Lluvias” (SIEDUCRES v1.0), alineado con las políticas de gestión del PST II, el diseño ya desarrollado y la documentación de requisitos existente.

Gestión del proyecto

1-Plan de desarrollo de software

- **Visión y propósito:** Garantizar continuidad pedagógica en escuelas rurales durante interrupciones por lluvias, con acceso a contenidos, actividades, calificaciones y comunicación de bajo consumo de datos.
- **Metodología:** Iterativa e incremental (SCRUM + UP). Sprints de 1–3 semanas; tareas entre 4–16 horas (máx. 20 horas), con revisión de calidad al cierre de cada sprint.
- **Roles:**
 - **Product Owner:** Dirección del plantel (prioriza valor pedagógico).
 - **Arquitecto/Dev lead:** arquitectura de tres capas, seguridad, manejo de sesiones y administración de roles.
 - **Equipo de desarrollo:** Frontend, backend, base de datos, QA.
 - **Docente coordinadora:** Operación y transferencia tecnológica.

- **Hitos y fases:**
 - **Fase 0 (Inicio, 1 semana):** Alcance, backlog inicial, ambiente, repositorios.
 - **Fase 1 (MVP pedagógico, 3–4 sprints):** Autenticación, gestión de usuarios, carga/consulta de contenidos, asignación/entrega de actividades, calificación básica.
 - **Fase 2 (Participación y seguimiento, 2–3 sprints):** Foros, notificaciones, progreso de visualización, historial, reportes.
 - **Fase 3 (Calidad y escalabilidad, 2 sprints):** Accesibilidad (WCAG AA), pruebas de rendimiento orientadas a asegurar tiempos de respuesta adecuados en condiciones reales de conectividad limitada, alineadas con los Requisitos No Funcionales establecidos para el sistema.
- **Criterios de aceptación por fase:**
 - **F1:** RF01–RF13 operativos con datos persistentes; flujos completos docente–estudiante.
 - **F2:** RF14–RF18–RF20 activos; métricas y exportaciones.
 - **F3:** RNF-01/02/03/04/06 verificables; planes de respaldo y restauración ejecutados.

Cronograma tentativo (16 semanas)

| Fase | Semanas | Entregables clave |
|-------------|----------------|--------------------------|
| | | |

| | | |
|-----------------------------|-------|--|
| Inicio | 1 | Plan, repositorio, CI básico |
| MVP (Sprints 1–4) | 4–7 | RF01–RF13 |
| Participación (Sprints 5–7) | 8–12 | RF14–RF18–RF20 |
| Calidad (Sprints 8–9) | 13–16 | RNF-01/02/03/04/06, manuales, capacitación |

Fuentes: Diseño de arquitectura/UI/BD ya elaborado y metas de rendimiento, accesibilidad y respaldo.

2-Planes de iteraciones

Sprint 1: Autenticación y gestión de usuarios

- **Objetivo:** RF01–RF06 operativos.
- **Historias (PO):**
 - **Registro de usuario (Admin):** Alta con rol y datos mínimos.
 - **Login / logout por rol:** Redirección a panel por rol.
 - **Edición y eliminación de usuarios.**
- **Hecho (DoD):** Validaciones, hash de contraseñas, sesiones seguras, pruebas de integración.

Sprint 2: Contenidos y consumo

- **Objetivo:** RF07–RF09 (carga, consulta, progreso).
- **Historias:**
 - **Carga por docente (PDF/DOCX/PPT/enlaces).**

- **Consulta/descarga por estudiante/representante.**
- **Registro de progreso de visualización.**
- **DoD:** Metadatos, restricciones por rol, métricas de progreso visibles.

Sprint 3: Actividades y entrega

- **Objetivo:** RF10–RF11–RF12–RF13.
- **Historias:**
 - **Asignar actividad con criterios y fecha.**
 - **Entrega con texto/archivo; confirmación.**
 - **Calificar y retroalimentar; notificar.**
 - **Consulta de calificaciones y retroalimentación.**
- **DoD:** Estados Pendiente/Entregado/Calificado, notificaciones de evaluación.

Sprint 4: Foros y notificaciones

- **Objetivo:** RF14–RF15–RF16.
- **Historias:**
 - **Crear temas en foro.**
 - **Responder público/privado con destinatarios.**
 - **Notificaciones automáticas por actividad/calificación/contenido.**
- **DoD:** Historial de notificaciones, filtros, privacidad efectiva.

Sprint 5: Historial y reportes

- **Objetivo:** RF17–RF18–RF19–RF20.
- **Historias:**
 - **Historial por lapsos (promedios, asistencia).**
 - **Reportes de participación (export PDF/Excel).**
 - **Encuestas y reporte consolidado por periodo/grupo.**
- **DoD:** Exportaciones, filtros, consistencia con datos de calificaciones y actividades.

Sprint 6–7: Accesibilidad y rendimiento

- **Objetivo:** RNF-02/03/01.
- **Historias:**
 - Accesibilidad WCAG AA (contraste, tamaño de texto, navegación simple)
 - Pruebas de carga básicas para validar el rendimiento del sistema en escenarios reales de conectividad limitada.
- **DoD:** Informe de pruebas, defectos corregidos, métricas cumplidas.

Sprint 8–9: Disponibilidad, respaldo y seguridad

- **Objetivo:** RNF-04/06 + sostenibilidad.
- **Historias:**
 - **Backups diarios y pruebas de restauración.**

- **TLS/SSL, revisión de inyección SQL y sesiones.**
- **Manual de usuarios y plan de capacitación (3 talleres).**
- **DoD:** Procedimientos documentados, restauración validada, capacitación impartida.

3-Lista de riesgos y mitigación

- **Lluvias extremas impiden reuniones:** Pasar a capacitación remota; materiales asincrónicos.
Mitigación: Talleres por chat y video; cronograma flexible.
- **Punto único de falla:** Limitaciones de infraestructura en las escuelas rurales (pocos equipos disponibles).
- **Mitigación:** Respaldos periódicos y uso de repositorios externos.
- **Sobrecarga docente en evaluaciones:** Ajuste de cargas y ventanas de publicación; soporte del equipo.
Mitigación: Calendario compartido; buffers de tiempo.
- **Presupuesto y equipos limitados:** Priorizar móviles Android; formatos ultraligeros; módulos críticos primero.
Mitigación: Diseño offline y sincronización; CDN futura.
- **Adopción tecnológica inicial baja:** Tres talleres, tutoría administrativa capacitada, manual paso a paso.
Mitigación: Soporte guiado, checklists, sesiones prácticas.

- **Riesgos de seguridad (datos personales):** Validaciones de entrada, manejo seguro de sesiones y protección ante inyección SQL.

Mitigación: Revisión de seguridad por sprint, política de privacidad.

4-Glosario del Proyecto.

Enlistado en orden de aparacion en el documento:

SIEDUCRES v1.0: Sistema Educativo Resiliente, plataforma web-móvil para continuidad pedagógica ante interrupciones por lluvias.

Iteración: Ciclo de trabajo dentro de una metodología ágil, con duración definida (1–3 semanas), en el que se desarrollan y entregan funcionalidades incrementales del sistema.

Iteración definida: Iteración previamente planificada con objetivos, tareas y criterios de aceptación establecidos, alineada al backlog y dependencias del proyecto.

Mitigación: Acciones preventivas o correctivas diseñadas para reducir el impacto de un riesgo identificado en el proyecto.

Product Backlog: Lista priorizada de requisitos e historias de usuario que representan el trabajo pendiente del proyecto.

Sprint: Iteración corta (1–3 semanas) en la que se seleccionan y desarrollan historias del Product Backlog.

Sprint (Sprint Backlog): Conjunto de tareas comprometidas por el equipo para cumplir los objetivos de un sprint específico.

Metodología SCRUM + UP: Combinación de **SCRUM** (gestión ágil de proyectos) y **Unified Process (UP)**, que integra iteraciones cortas con fases estructuradas de análisis, diseño, implementación y pruebas.

Product Owner (PO): Rol responsable de maximizar el valor del producto, priorizando requisitos y validando entregables según las necesidades del usuario.

Arquitecto/Dev lead: Responsable de definir la arquitectura del sistema, guiar decisiones técnicas y asegurar la calidad del desarrollo.

Equipo de desarrollo: Grupo multidisciplinario encargado de implementar funcionalidades: frontend, backend, base de datos y pruebas.

Frontend: Parte del sistema que interactúa directamente con el usuario, desarrollada en interfaces gráficas y componentes visuales.

Backend: Lógica del servidor y servicios que procesan datos, gestionan reglas de negocio y comunican la base de datos con el frontend.

Base de datos: Sistema de almacenamiento estructurado que guarda y organiza la información del proyecto.

QA (Quality Assurance): Proceso de aseguramiento de calidad mediante pruebas unitarias, de integración y validación de requisitos.

Hitos y fases: Momentos clave del proyecto que marcan avances significativos, organizados en fases (inicio, MVP, participación, calidad).

Backlog inicial: Primer conjunto de requisitos e historias de usuario definidos al inicio del proyecto.

MVP (Producto Mínimo Viable): Versión inicial del sistema con funcionalidades esenciales que permiten validar la propuesta.

MVP pedagógico: MVP orientado a garantizar continuidad pedagógica, incluyendo autenticación, gestión de usuarios y contenidos básicos.

WCAG AA: Accesibilidad básica (contraste, tamaño de texto, navegación simple) acorde a los lineamientos funcionales definidos..

Backup: Copia de seguridad periódica de datos para garantizar disponibilidad y recuperación ante fallos.

Diseño de arquitectura/UI/BD: Esquema técnico que integra arquitectura de software, interfaz de usuario (**UI**) y base de datos (**BD**).

PO: Abreviatura de **Product Owner**.

Historias (PO): Requisitos narrados desde la perspectiva del usuario, priorizados por el Product Owner.

DoD (Definition of Done): Criterios que determinan cuándo una tarea o historia se considera completada.

Hecho (DoD): Estado alcanzado cuando se cumplen los criterios de aceptación y pruebas definidos en el DoD.

UX (User Experience): Experiencia del usuario al interactuar con el sistema, medida en términos de facilidad, satisfacción y eficiencia.

Pruebas de usabilidad y mejoras UX: Evaluaciones prácticas para identificar problemas de interacción y optimizar la experiencia del usuario.

TLS/SSL: Protocolos de cifrado que garantizan la seguridad de las comunicaciones en la web.

Revisión de inyección SQL y sesiones: Prácticas de seguridad para prevenir ataques de inyección SQL y asegurar la gestión correcta de sesiones de usuario.

Buffers: Espacios de tiempo o recursos adicionales previstos para absorber retrasos o sobrecargas.

Buffers de tiempo: Margen temporal añadido al cronograma para mitigar riesgos de retraso.

CDN (Content Delivery Network): Red de servidores distribuidos que optimiza la entrega de contenidos digitales.

CDN future: Optimización futura del rendimiento mediante buenas prácticas de carga ligera y organización eficiente de recursos.

Hashing Argon2id: Algoritmo criptográfico de hashing seguro para proteger contraseñas y datos sensibles.

≥99.5% disponibilidad: Indicador de confiabilidad que asegura que el sistema estará operativo al menos el 99.5% del tiempo.

UI y BD: **UI** User Interface = Interfaz de usuario y **BD** Base de Datos, componentes esenciales de la arquitectura del sistema.

Arquitectura UI y BD: Diseño integrado que conecta la interfaz de usuario con la base de datos mediante capas de negocio.

3FN (Tercera Forma Normal): Regla de normalización en diseño de bases de datos que elimina redundancias y asegura integridad.

FK (Foreign Key): Clave foránea en una base de datos que establece relaciones entre tablas.

Endpoints REST para carga/listado/descarga: Interfaces de programación que permiten operaciones de carga, consulta y descarga de contenidos mediante servicios REST.

Guards y middleware: Mecanismos de seguridad y control en aplicaciones web que validan accesos y gestionan flujos de datos.

RF/RNF: Requisitos funcionales (RF) y no funcionales (RNF) que definen el comportamiento y calidad del sistema.

ERS (Especificación de Requisitos de Software): Documento que describe en detalle los requisitos funcionales y no funcionales del sistema, incluyendo actores, casos de uso, escenarios y criterios de aceptación.

CU (Casos de Uso): Modelos que describen interacciones entre actores y el sistema para cumplir requisitos específicos.

Requisitos

1-Documento de especificaciones del sistema (DES)

Requisitos Funcionales (resumen ejecutivo)

- **RF01–RF04:** Gestión de usuarios (registrar, editar, eliminar, configurar parámetros).
- **RF05–RF06:** Autenticación y cierre de sesión por rol.
- **RF07–RF09:** Carga/consulta de contenidos; progreso de visualización.
- **RF10–RF13:** Asignar, entregar, calificar actividades; consultar calificaciones.
- **RF14:** Notificaciones automáticas por eventos (actividad, calificación, contenido).
- **RF15–RF16:** Foros (crear tema; responder público/privado).
- **RF17–RF18:** Historial académico; reportes de participación (export).
- **RF19–RF20:** Encuestas de satisfacción; reporte consolidado.

Requisitos No Funcionales (síntesis)

- **RNF-01 (Rendimiento):** Soportar 500 usuarios concurrentes; carga de recursos ≤ 800 ms en 3G.
- **RNF-02 (Accesibilidad):** Compatible con Android gama baja/media; funciones básicas de accesibilidad.
- **RNF-03 (Usabilidad):** Flujo simple (2–3 clics), estados explícitos, mensajes claros.
- **RNF-04 (Disponibilidad/Respaldo):** Backups diarios y protocolo de actualizaciones; $\geq 99.5\%$ disponibilidad.
- **RNF-05 (Sostenibilidad):** Manual paso a paso; autonomía operativa docente.

- **RNF-06 (Seguridad):** Privacidad y cifrado de comunicaciones; sesiones seguras; política de datos.

La arquitectura basada en PHP 8.2, PostgreSQL 17, Bootstrap 5 y el modelo de tres capas soporta los RF/RNF definidos y permite su implementación progresiva según las iteraciones planificadas.

2-Lista de requisitos priorizados del usuario (Product Backlog)

| RF | Dependencias | Casos de Uso | Prioridad |
|------|------------------|---|-----------|
| RF01 | - | CU01.1 – Registrar usuario | Alta |
| RF02 | RF01 | CU01.2 – Editar usuario | Alta |
| RF03 | RF01, RF02 | CU01.3 – Eliminar usuario | Alta |
| RF04 | - | CU01.4 – Configurar parámetros del sistema | Media |
| RF05 | RF01, RF02, RF03 | CU02.1 – Iniciar sesión | Alta |
| RF06 | RF05 | CU02.2 – Cerrar sesión | Alta |
| RF07 | RF04, RF05 | CU03 – Cargar contenido académico | Alta |
| RF08 | RF07 | CU04 – Consultar contenido académico | Alta |
| RF09 | RF08 | CU04.1 – Consultar progreso de visualización | Media |
| RF10 | RF04, RF05 | CU05 – Asignar actividad | Alta |
| RF11 | RF10 | CU06 – Resolver actividad | Alta |
| RF12 | RF11 | CU07 – Revisar y calificar actividad | Alta |
| RF13 | RF12 | CU08 – Consultar calificación | Media |
| RF14 | RF07, RF10, RF12 | CU09 – Enviar notificación automática | Media |
| RF15 | RF04, RF05 | CU10 – Crear tema en foro | Media |
| RF16 | RF15 | CU11 – Responder mensajes en foros | Media |
| RF17 | RF13 | CU12 – Consultar historial académico | Media |
| RF18 | RF07, RF09, RF15 | CU13 – Generar reportes de participación | Media |
| RF19 | - | CU14 – Completar encuesta de satisfacción | Media |
| RF20 | RF19 | CU15 – Generar reporte de encuestas (<> CU14) | Media |

3-Lista de tareas por sprint (Sprint Backlog)

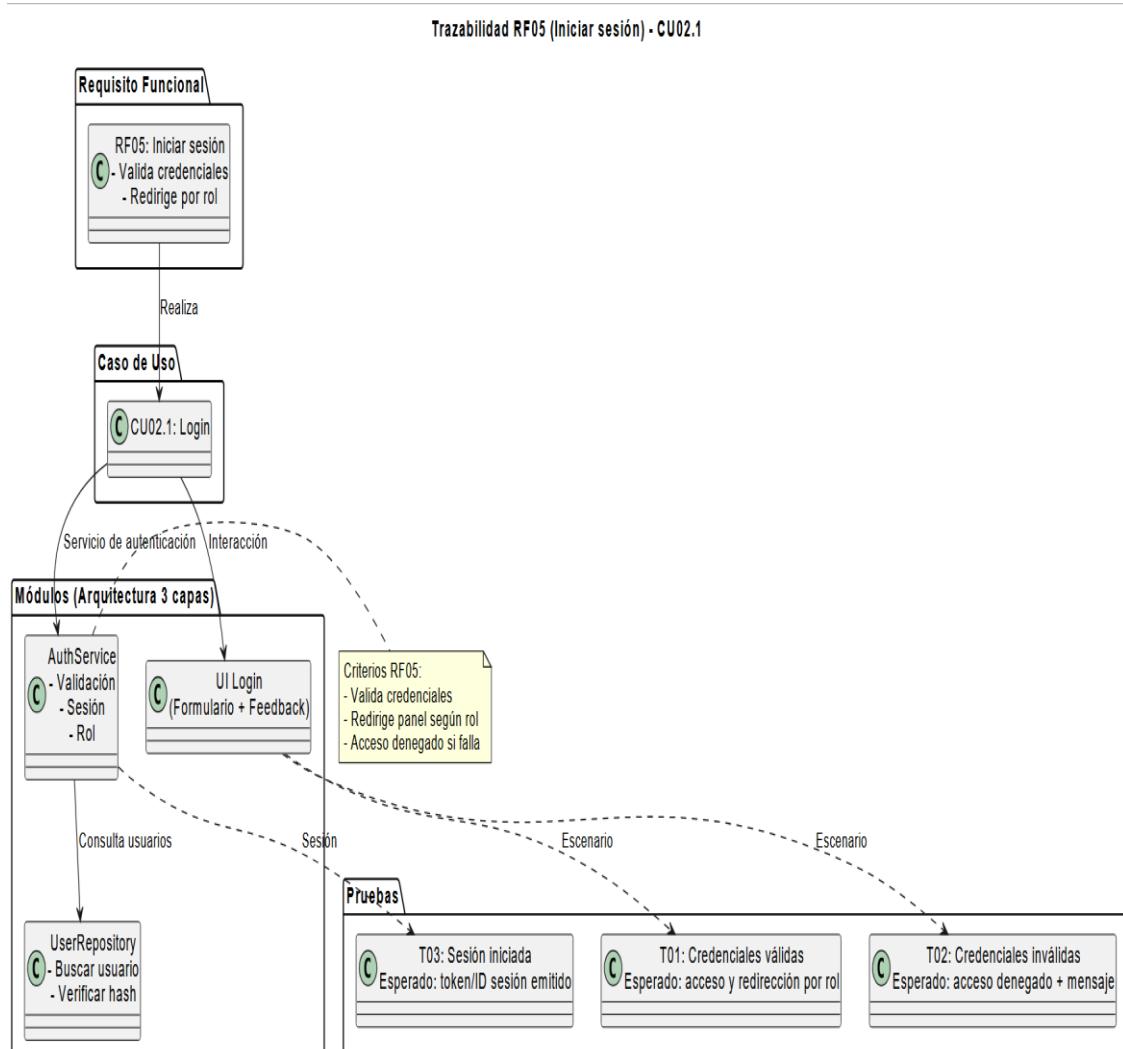
Ejemplo detallado para Sprint 2 (Contenidos):

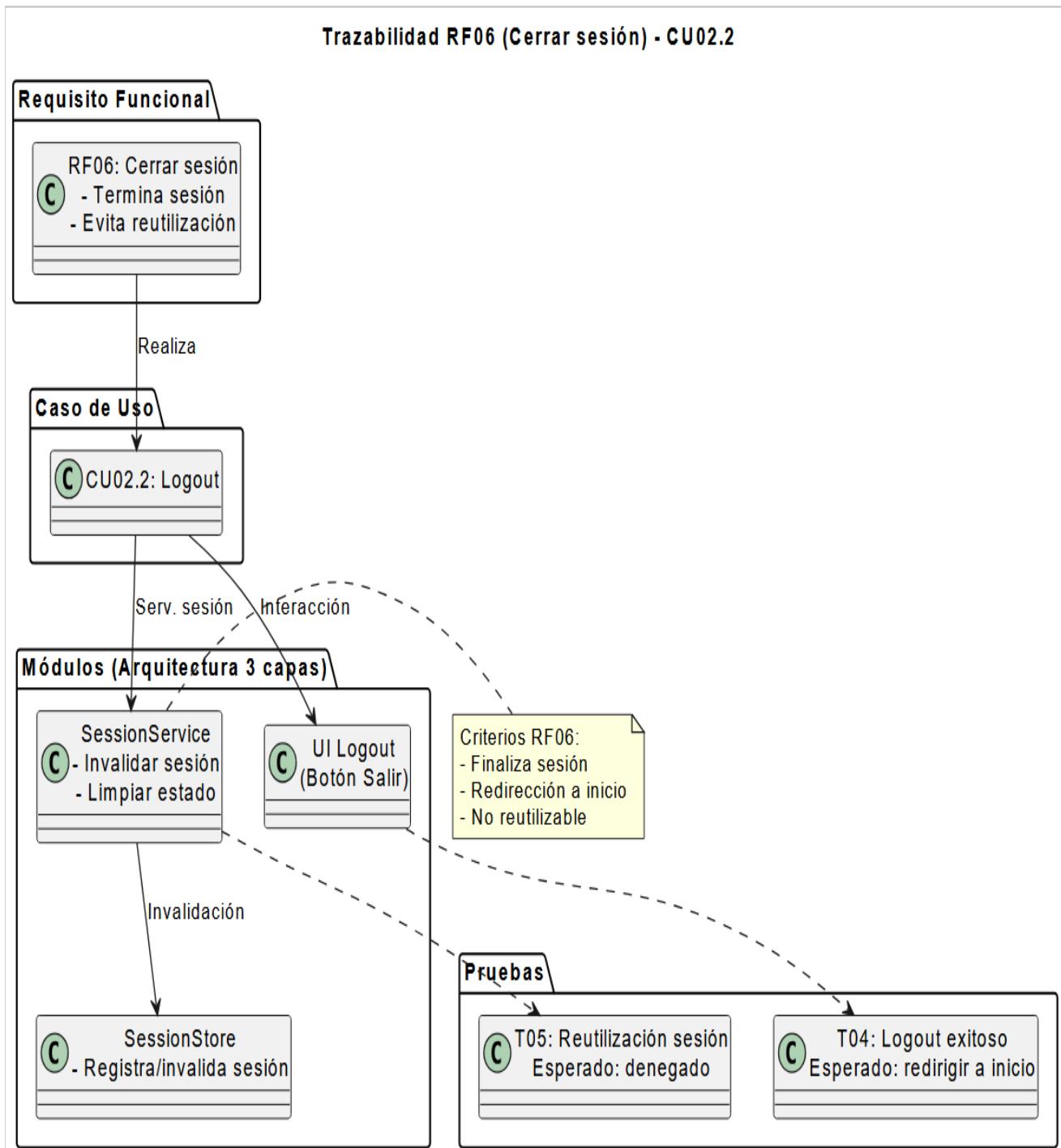
- **Arquitectura/BD:**
 - **Definir tablas:** Contenidos Académicos, Progreso Visualización (índices, FK).
 - **Servicios:** Endpoints REST para carga/listado/descarga.
- **Frontend:**
 - **UI carga:** Formulario con validaciones (formatos, tamaño).
 - **UI consulta:** Tarjetas con ícono, resumen, estado visto/no visto.
 - **Progreso:** Barra/porcentaje bajo título del contenido.
- **Seguridad:**
 - **Autorización por rol/matería:** Guards y middleware.
- **QA:**
 - **Pruebas integración:** Subida/consulta/descarga con roles.
 - **Pruebas unitarias:** Validaciones y servicios.
- **Documentación:**
 - **Guía docente:** Pasos para cargar contenidos.
 - **Registro de decisiones:** Campos obligatorios y límites de tamaño.

Cada sprint incluirá tareas análogas en arquitectura, frontend, backend, seguridad, QA y documentación, ajustadas a sus RF/RNF.

4-Documento de gestión de requisitos por iteración

- **Entrada:** Backlog priorizado, capacidad del equipo, dependencias tecnológicas/pedagógicas.
- **Selección:** Historias de mayor valor y menor riesgo, respetando dependencias (ej. RF10 antes de RF11).
- **Alineación RNF:** Incluir siempre historias RNF (accesibilidad/usabilidad/rendimiento) como “fondo” del sprint.
- **Trazabilidad:** Matriz RF→Casos de uso→Módulos→Pruebas→Entregables.





- **Revisión de cambios:** Ajustar criterios de aceptación y pruebas según retroalimentación del PO/docentes.
- **Cierre del sprint:** Demostración, informe de calidad, deuda técnica identificada y planificada.

Plantilla de reporte por sprint (Ejemplo Sprint 2)

Sprint: 2 (Contenidos y consumo)

Periodo: 2025-2 / Semana 4-5

Objetivo: RF07-RF09 activos (Carga, Consulta, Progreso)

Backlog comprometido

--
RF07: Cargar contenidos (Docente)

RF08: Consultar contenidos (Est/Rep)

RF09: Progreso de visualización (Docente/Est)

RNF-02: Accesibilidad mínima en vistas

Hecho (DoD) / Evidencias

--
- Subida/descarga funcional con roles

- Metadatos y estados visto/no visto

- Barra de progreso visible por contenido

- Pruebas integración (OK), unitarias (OK)

Métricas de calidad

--
Pruebas integración: 15/15 OK

Pruebas unitarias: 28/30 OK (2 pendientes)

Incidencias críticas: 0

Tiempo medio respuesta (3G): 720 ms

Accesibilidad (contraste/foco): OK

Riesgos y mitigación (estado)

--
R1: Lluvias -> Capacitaciones remotas (EN CURSO)

R2: Único PC -> Backups diarios + repositorio externo (OK)

Observaciones / Deuda técnica

--
- Refactor validación tamaños adjuntos (pendiente)

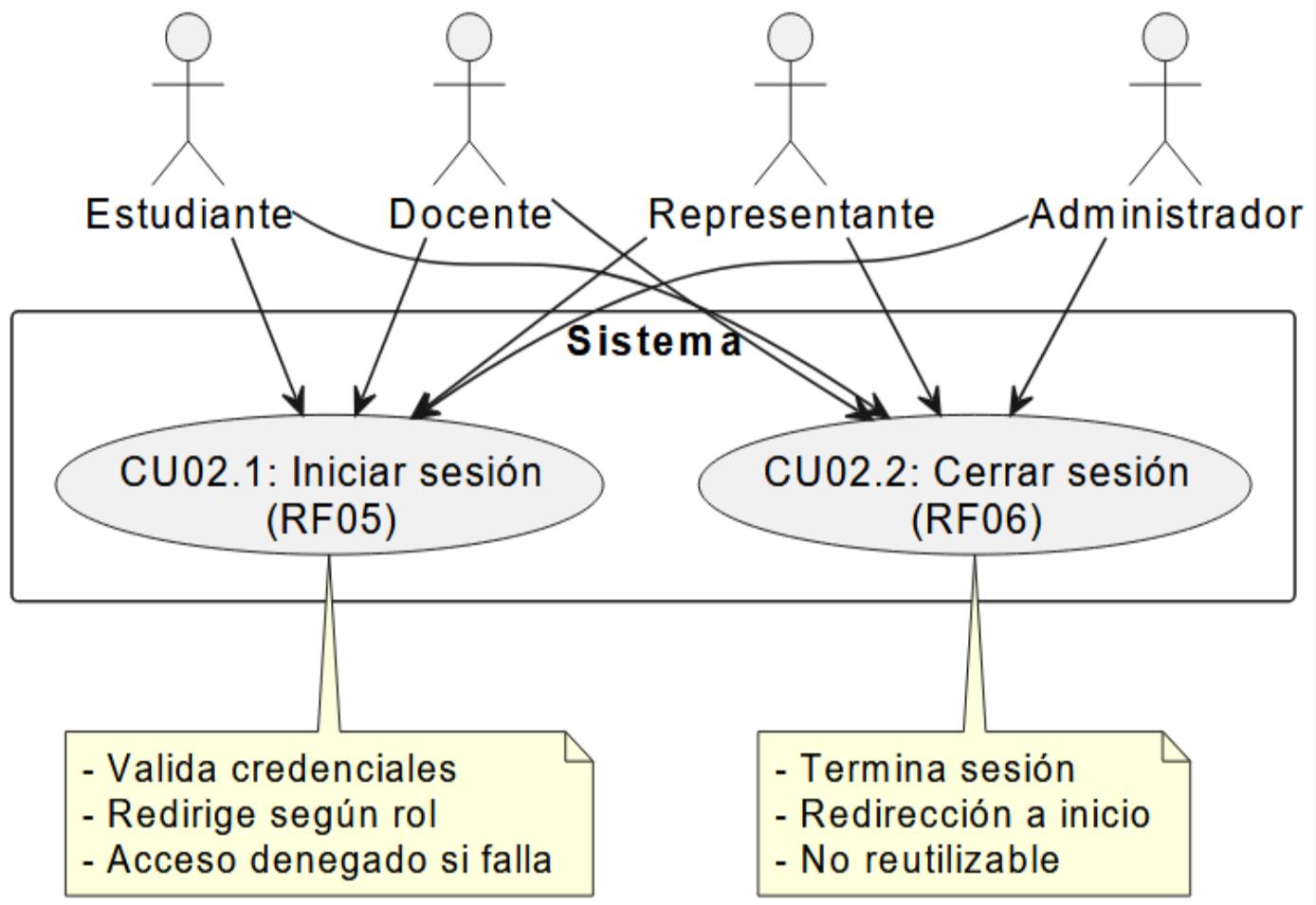
- Agregar textos alternativos en 2 iconos (accesibilidad)

5-Especificación de requisitos de software (ERS) / Modelo de casos de uso

- **Actores:** Administrador, Docente, Estudiante, Representante, Sistema automático.
- **Casos de uso clave:**
 - **CU01 (Gestionar usuarios y parámetros):** Incluye RF01–RF04.
 - **CU02 (Autenticarse):** RF05–RF06.

Propósito: Visualizar actores y límites del sistema para CU02.1 y CU02.2.

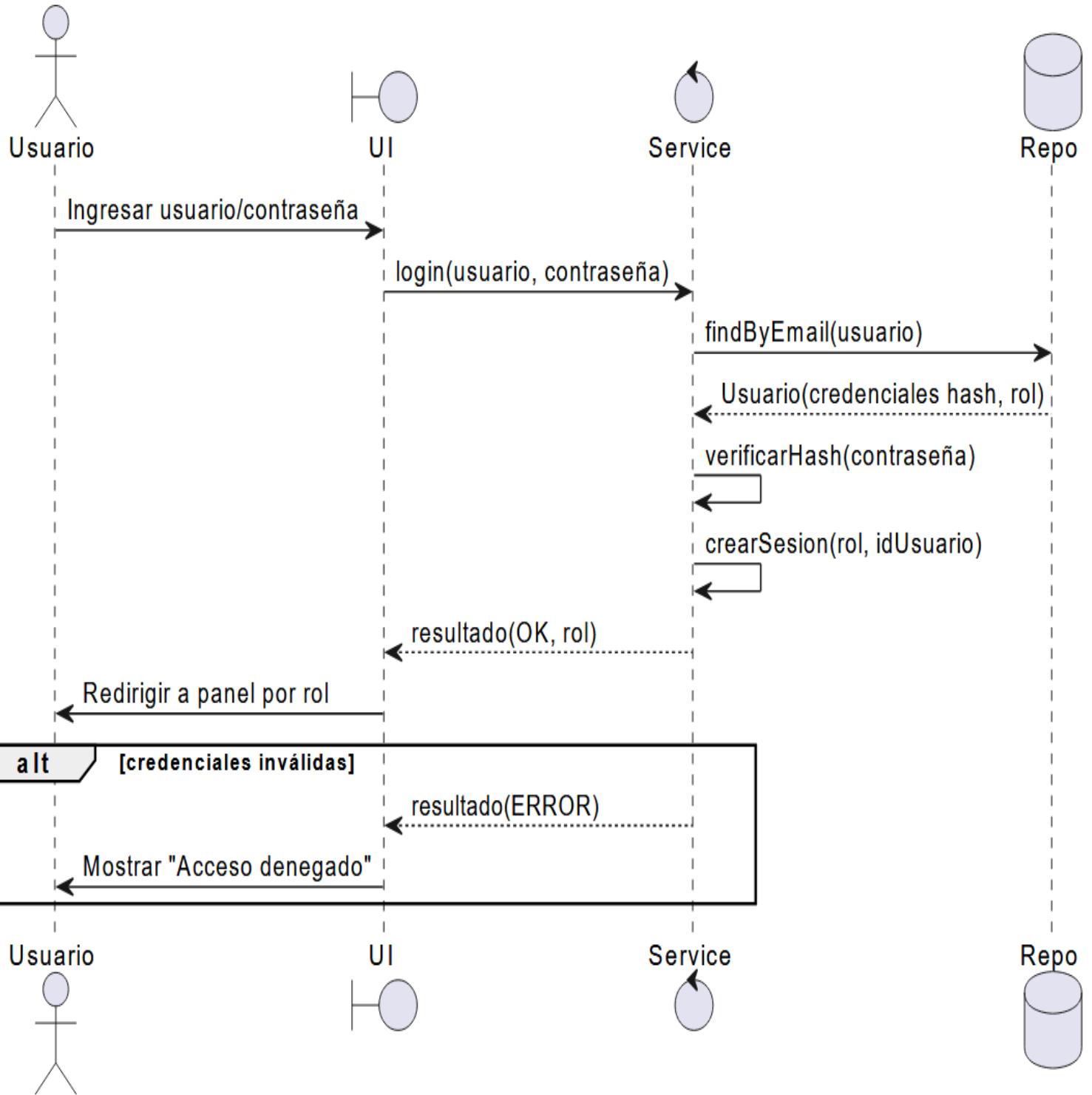
Diagrama de casos de uso: Autenticarse (Login/Logout)



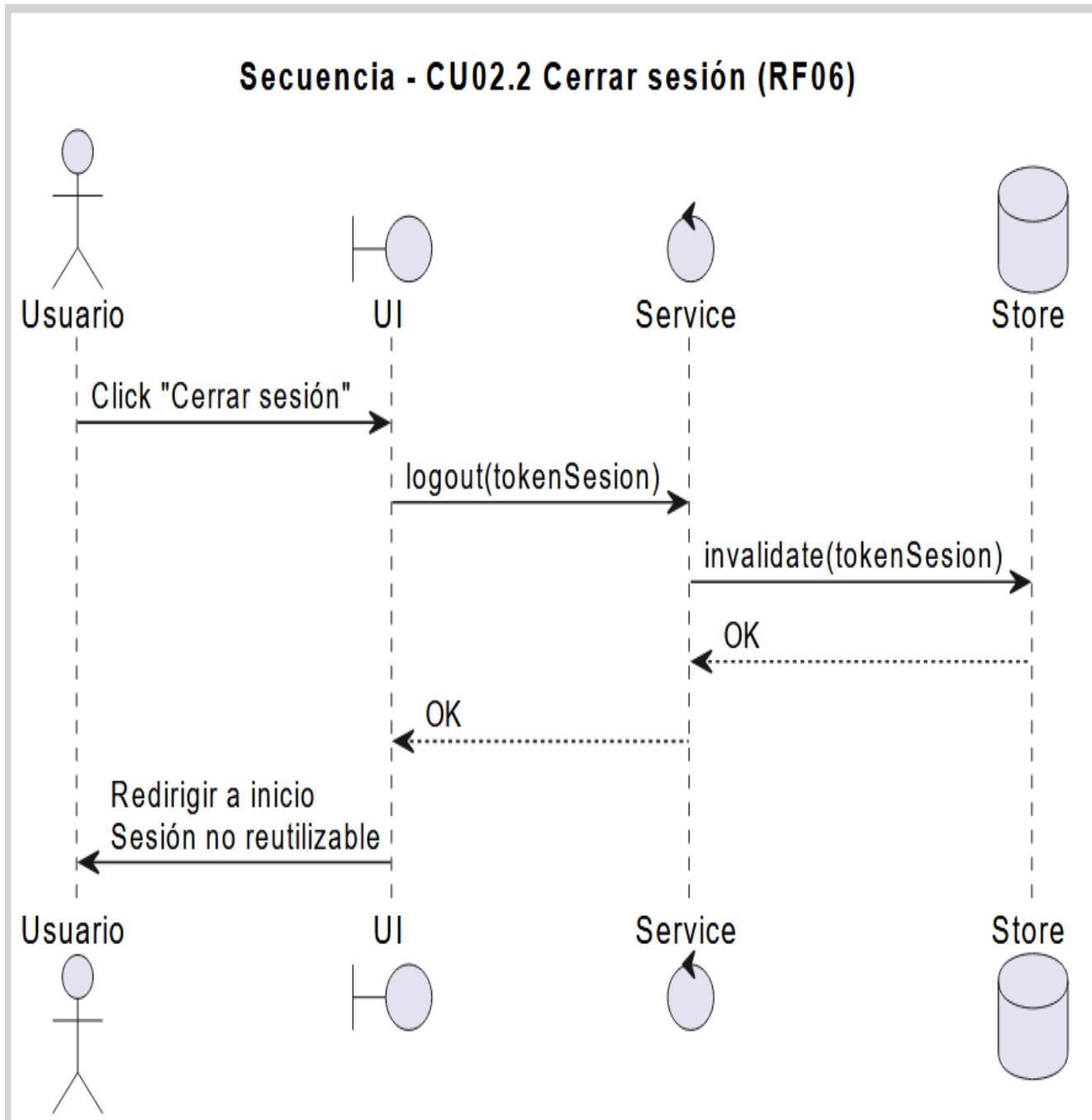
Propósito: Describir el intercambio entre capas según la arquitectura de tres capas ya definida en el diseño.

Secuencia CU02.1: Iniciar sesión (RF05)

Secuencia - CU02.1 Iniciar sesión (RF05)



Secuencia CU02.2: Cerrar sesión (RF06)



- **CU03–CU04 (Contenidos):** RF07–RF09.
- **CU05–CU08 (Actividades y calificaciones):** RF10–RF13.

- **CU09 (Notificaciones):** RF14.
 - **CU10–CU11 (Foro):** RF15–RF16.
 - **CU12–CU13 (Historial y reportes):** RF17–RF18.
 - **CU14–CU15 (Encuestas):** RF19–RF20.
- **Escenarios:**
 - **Registrar usuario:** Flujo normal, validaciones (correo único), errores y confirmaciones.
 - **Asignar y entregar actividad:** Validación de fecha, adjuntos opcionales, estados y notificaciones.
 - **Calificar y consultar calificación:** Registro, aviso y visualización con retroalimentación.
 - **Consultar historial:** Filtros por periodo/materia, exportaciones.
 - **Criterios de aceptación:** Consistencia entre calificaciones, actividades y reportes; accesos por rol; estados visibles.

El diagrama de casos de uso y los escenarios detallados complementan esta ERS y están coherentes con la arquitectura en tres capas descrita en el diseño.

Cierre y siguientes pasos

- **Validar con PO/docentes:** Confirmar prioridades y criterios de aceptación del MVP.

- **Ajustar plan de iteraciones:** Según disponibilidad docente y condiciones climáticas.
- **Configurar pruebas:** Usabilidad y rendimiento desde Sprint 2 para detectar tempranamente brechas.
- **Plan de transferencia:** Agendar 3 talleres y preparar manuales visuales para docentes y representantes.

Análisis y Diseño

1. Propósito de la Vista de Casos de Uso

La presente sección tiene como finalidad proporcionar una visión estructurada y completa de los casos de uso que conforman la funcionalidad del sistema propuesto. La Vista de Casos de Uso permite identificar el comportamiento esperado del sistema desde la perspectiva de sus actores, así como las relaciones existentes entre las funcionalidades principales y sus subcasos asociados.

Esta vista sirve como base para el **Modelo de Análisis**, el **Modelo de Diseño** y la planificación de iteraciones, garantizando coherencia en la implementación progresiva del software.

1.2. Actores del Sistema

Los actores identificados para el sistema son:

- **Administrador:** Responsable de la gestión de usuarios, parámetros del sistema y generación de reportes administrativos.
- **Docente:** Encargado de cargar contenidos, asignar y calificar actividades, participar en foros y generar reportes académicos.
- **Estudiante:** Usuario principal que consulta contenidos, entrega actividades, revisa calificaciones, participa en foros y completa encuestas.
- **Representante:** Actor que consulta información académica del estudiante, revisa calificaciones, contenidos y completa encuestas.
- **Sistema Automático:** Componente interno encargado del envío de notificaciones y del registro automático del progreso académico.

1.3. Lista de Casos de Uso Identificados

La Tabla 1 presenta los casos de uso definidos, indicando su actor asociado, tipo y relaciones *include* y *extend* cuando corresponda.

Tabla 1. Casos de Uso del Sistema

| Código | Nombre del Caso de Uso | Actor(es) | Tipo | Relación |
|--------|---------------------------------|---------------|-----------|--|
| CU01 | Gestionar usuarios y parámetros | Administrador | Principal | incluye CU01.1, CU01.2, CU01.3, CU01.4 |
| CU01.1 | Registrar usuario | Administrador | Incluido | include |
| CU01.2 | Editar usuario | Administrador | Incluido | include |
| CU01.3 | Eliminar usuario | Administrador | Incluido | include |

| | | | | |
|--------|-------------------------------------|------------------------------------|-----------|------------------------|
| CU01.4 | Configurar parámetros del sistema | Administrador | Incluido | include |
| CU02 | Autenticarse en el sistema | Todos | Principal | incluye CU02.1, CU02.2 |
| CU02.1 | Iniciar sesión | Todos | Incluido | include |
| CU02.2 | Cerrar sesión | Todos | Incluido | include |
| CU03 | Cargar contenido académico | Docente | Principal | — |
| CU04 | Consultar contenido académico | Estudiante, Representante | Principal | extiende CU04.1 |
| CU04.1 | Consultar progreso de visualización | Docente, Sistema Automático | Extendido | extend |
| CU05 | Asignar actividad | Docente | Principal | — |
| CU06 | Resolver actividad | Estudiante | Principal | — |
| CU07 | Revisar y calificar actividad | Docente | Principal | — |
| CU08 | Consultar calificación | Estudiante, Representante | Principal | — |
| CU09 | Enviar notificación automática | Sistema Automático | Principal | — |
| CU10 | Crear tema en foro | Docente, Estudiante | Principal | — |
| CU11 | Responder mensajes en foro | Docente, Estudiante, Representante | Principal | — |
| CU12 | Consultar historial académico | Estudiante, Representante | Principal | — |
| CU13 | Generar reportes de participación | Docente, Administrador | Principal | — |
| CU14 | Completar encuesta de satisfacción | Estudiante, Representante | Principal | incluye CU15 |

| | | | | |
|------|--|---------------|----------|---------|
| CU15 | Generar reporte de encuestas de satisfacción | Administrador | Incluido | include |
|------|--|---------------|----------|---------|

1.4. Agrupación de Casos de Uso por Módulos Funcionales

Con el propósito de facilitar el análisis y el diseño del sistema, los casos de uso se organizan en los siguientes módulos funcionales:

4.1 Módulo de Gestión de Usuarios y Seguridad

- CU01 – Gestionar usuarios y parámetros
- CU01.1 Registrar usuario
- CU01.2 Editar usuario
- CU01.3 Eliminar usuario
- CU01.4 Configurar parámetros
- CU02 – Autenticarse en el sistema
- CU02.1 Iniciar sesión
- CU02.2 Cerrar sesión

4.2 Módulo de Contenido Académico

- CU03 – Cargar contenido académico
- CU04 – Consultar contenido académico
- CU04.1 Consultar progreso (extend)

4.3 Módulo de Actividades Académicas

- CU05 – Asignar actividad

- CU06 – Resolver actividad
- CU07 – Revisar y calificar actividad
- CU08–Consultar calificación

4.4 Módulo de Historial y Reportes

- CU12 – Consultar historial académico
- CU13 – Generar reportes de participación

4.5 Módulo de Foros y Participación

- CU10 – Crear tema en foro
- CU11 – Responder mensajes en foro
- CU14 – Completar encuesta de satisfacción
- CU15 – Generar reportes de encuestas (*include*)

4.6 Módulo de Automatización

- CU09 – Enviar notificación automática

1.5. Casos de Uso Prioritarios para Diseño Detallado

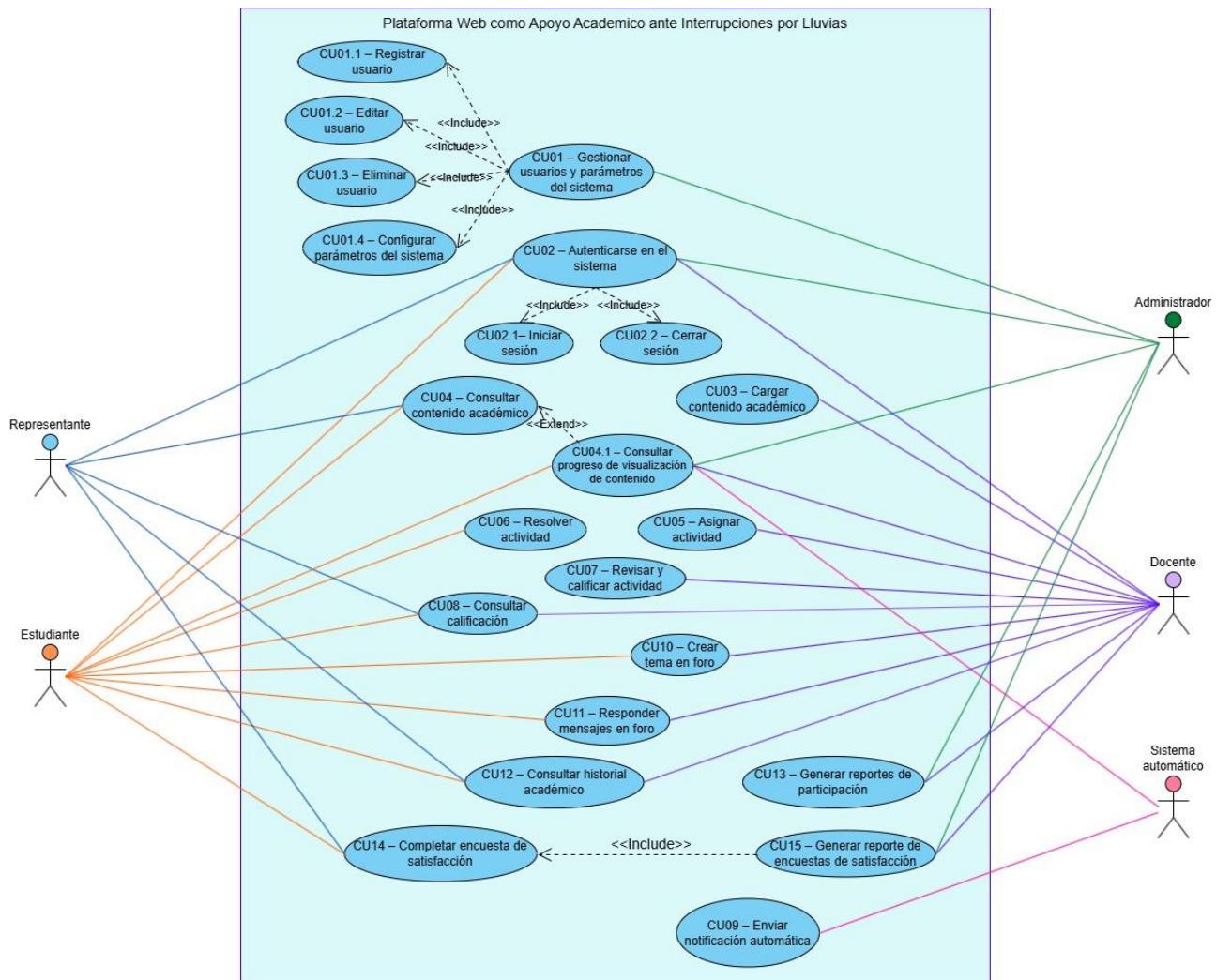
Se identifican como prioritarios para el análisis profundo y diseño formal los siguientes ocho casos de uso:

1. CU02 – Autenticarse en el sistema
2. CU01 – Gestionar usuarios y parámetros
3. CU03 – Cargar contenido académico

4. CU04 – Consultar contenido académico
5. CU05 – Asignar actividad
6. CU06 – Resolver actividad
7. CU07 – Revisar y calificar actividad
8. CU12 – Consultar historial académico

Estos casos representan el núcleo operacional del sistema y definen la funcionalidad mínima viable (MVP) necesaria para garantizar la continuidad académica ante interrupciones.

1.6 Diagrama de casos de uso



2. Descripción detallada de arquitectura

2.1. Introducción a la Arquitectura del Sistema

La plataforma web se desarrolla bajo una arquitectura cliente–servidor de tres capas, diseñada para ser modular, escalable y mantenible. Este enfoque garantiza la continuidad académica ante interrupciones presenciales y permite que los usuarios accedan al contenido, actividades, calificaciones y módulos de comunicación.

El sistema se implementará utilizando las siguientes tecnologías:

- **Backend:** PHP 8.2 ejecutado dentro de contenedores Docker.
- **Frontend:** HTML5, CSS3, Bootstrap 5 y el sistema de diseño **SieducresUI**.
- **Base de datos:** PostgreSQL 17 administrado mediante PgAdmin 4.
- **Infraestructura:** Docker y Docker Compose para crear un entorno portátil, reproducible y estandarizado entre los miembros del equipo.
- **Entorno de desarrollo:** Visual Studio Code.

Este conjunto tecnológico garantiza estabilidad, facilidad de despliegue, compatibilidad entre entornos y buena integración con servidores educativos.

2.2. Arquitectura General del Sistema

1. Capa de Presentación (Frontal)

Es la interfaz mediante la cual interactúan los usuarios: administrador, docente, estudiante y representante.

Características principales:

- Interfaz web responsive, accesible desde computadoras y dispositivos móviles.
- Formularios de inicio de sesión, registro, carga de contenido, asignación y resolución de actividades.
- Paneles diferenciados según el rol.
- Comunicación con el back-end mediante solicitudes HTTP y AJAX.

2. Capa de Lógica de Negocio (Back-end)

Implementado en **PHP 8.x** utilizando el **patrón Modelo–Vista–Controlador (MVC)** para asegurar la separación de responsabilidades.

Funciones principales:

- Gestión de usuarios, roles y permisos.
- Validación de credenciales y manejo de sesiones seguras.
- Gestión de contenidos, actividades, respuestas y calificaciones.
- Registro del progreso académico.
- Generación de informes y estadísticas.
- Ejecución de procesos automáticos (notificaciones y métricas).

El uso de **PDO** garantiza un acceso seguro y parametrizado a PostgreSQL.

3. Capa de Datos (Base de Datos Relacional)

Responsable del almacenamiento estructurado de toda la información del sistema.

Incluye:

- Datos de usuarios y roles.
- Contenidos académicos.
- Actividades, entregas y calificaciones.
- Registros de visualización y utilización de recursos.
- Historiador académico.
- Información de foros y encuestas.

Se utiliza **PostgreSQL** por su robustez, integridad referencial, transacciones seguras y rendimiento en entornos multiusuario.

2.3. Vista Lógica de la Arquitectura (Módulos del Sistema)

La lógica se organiza en módulos funcionales coherentes:

Módulo 1: Autenticación y Seguridad

- Inicio y cierre de sesión.
- Control de accesos por rol.
- Manejo de sesiones seguras.

Módulo 2: Gestión de Usuarios y Parámetros

- CRUD de usuarios.
- Configuración de roles y permisos.
- Ajustes administrativos de la plataforma.

Módulo 3: Contenido Académico

- Carga y clasificación de materiales.
- Consulta y descarga de contenidos.
- Registro automático del progreso de visualización.

Módulo 4: Actividades Académicas

- Creación de actividades.
- Envío y almacenamiento de respuestas.
- Calificación y retroalimentación.
- Consulta de calificaciones.

Módulo 5: Comunicación y Participación

- Creación y gestión de temas en foros.
- Participación mediante mensajes.
- Encuestas de satisfacción e informes.

Módulo 6: Automatización

- Notificaciones automáticas.
- Generación programada de informes.
- Registro automático de métricas del sistema.

2.4. Arquitectura Tecnológica (Tecnologías Utilizadas)

Frontend

- HTML5, CSS3, JavaScript.
- **Bootstrap 5 y el sistema de diseño SieducresUI** para componentes visuales unificados.
- Formularios validados tanto en cliente como en servidor.

Backend

- **PHP 8.2** operando bajo el patrón organizativo MVC simple (sin frameworks externos).
- Controladores, modelos y vistas organizados en carpetas.
- Conexión segura mediante PDO hacia PostgreSQL.
- Gestión de sesiones y sanitización de entradas.
- Registro de logs y manejo de errores.

Base de datos

- PostgreSQL 17.
- Administración mediante PgAdmin 4.
- Integridad referencial, claves foráneas, transacciones y funciones SQL según necesidad.

Infraestructura

- **Docker y Docker Compose** para levantar:
 - Contenedor PHP/Apache
 - Contenedor PostgreSQL
 - Contenedor PgAdmin

- Esto garantiza portabilidad del proyecto sin errores de configuración entre equipos.

Automatización

- Cron jobs dentro de contenedores Docker para notificaciones y generación de reportes.
- Integración opcional con APIs (correo institucional, mensajería).

2.5. Vista de Componentes del Sistema (Diagrama de Componentes)

Los componentes principales son:

Componente de Autenticación

- Manejo de sesiones seguras.
- Validación de credenciales.

Componente de Gestión de Usuarios y Roles

- Registro, edición y eliminación de usuarios.
- Asignación de permisos.

Componente de Contenido Académico

- Gestión de materiales.
- Registro de visualización.

Componente de ActividadeS

- Creación y asignación de actividades.
- Envío de respuestas.
- Evaluación docente.

Componente de Comunicación (Foros y Encuestas)

- Temas, respuestas y participación.
- Encuestas y análisis.

Componente de Reportes

- Calificaciones.
- Estadísticas de participación.
- Resultados de encuestas.

Componente de Automatización

- Notificaciones automáticas.
- Procesos programados.

Todos los componentes se comunican mediante controladores PHP en el back-end, los cuales procesan la lógica, interactúan con los modelos y envían las respuestas al front-end.

2.6. Vista de Despliegue

El sistema se despliega utilizando **Docker** y **Docker Compose**, lo que permite empaquetar y ejecutar cada parte de la plataforma en contenedores independientes.

La arquitectura de despliegue se organiza de la siguiente forma:

1. Contenedor Web (PHP + Apache)

- Ejecuta PHP 8.2.
- Procesa solicitudes HTTP/AJAX.
- Contiene los controladores, modelos, vistas y archivos del frontend.
- Se comunica internamente con el contenedor de base de datos.

2. Contenedor de Base de Datos (PostgreSQL 17)

- Almacena toda la información del sistema.
- Acceso controlado únicamente desde el contenedor web.
- Administración a través de PgAdmin (contenedor separado).

3. Contenedor PgAdmin 4

- Permite gestionar la base de datos de forma gráfica.
- Se conecta internamente al contenedor PostgreSQL por docker networks.

4. Cliente (Navegador Web)

- Accede al frontend servido por el contenedor PHP/Apache.
- Compatible con laptops, tablets y teléfonos.
- No requiere instalación adicional.

5. Contenedor de Automatización (Opcional)

- Ejecuta cron jobs para notificaciones y reportes.
- Puede integrarse con servicios externos (correo, Telegram, WhatsApp).

3. Modelo de análisis

El Modelo de Análisis representa los conceptos fundamentales del dominio de la plataforma web académica, identificando las entidades, atributos y relaciones necesarias para soportar los casos de uso.

Este modelo no incluye detalles técnicos, algoritmos, tipos de datos ni elementos propios de la implementación; en cambio, se enfoca en describir la estructura conceptual del sistema.

2. Clases del Dominio

A continuación se presentan las clases identificadas, depuradas y organizadas conceptualmente:

2.1. Clase Usuario

Representa a cualquier persona que interactúa con la plataforma.

Atributos conceptuales:

- nombre
- correo
- rol

Generalización:

- Administrador
- Docente
- Estudiante
- Representante

2.2. Clase Administrador (subclase de Usuario)

Administra la plataforma: usuarios, parámetros, reportes.

(No requiere atributos adicionales en análisis)

2.3. Clase Docente (subclase de Usuario)

Permite asignar actividades, cargar contenidos y calificar.

(No requiere atributos adicionales en análisis)

2.4. Clase Estudiante (subclase de Usuario)

Accede a contenidos, actividades y calificaciones.

Atributo conceptual adicional:

- sección o grado

2.5. Clase Representante (subclase de Usuario)

Consulta calificaciones e historial del estudiante.

(No requiere atributos adicionales en análisis)

2.6. Clase Contenido

Representa materiales académicos compartidos por el docente.

Atributos:

- titulo
- descripcion
- fechaPublicacion
- archivoAdjunto
- enlace
- asignatura

2.7. Clase Actividad

Tarea académica asignada a los estudiantes.

Atributos:

- titulo
- descripcion
- fechaPublicacion
- fechaEntrega
- criteriosEvaluacion
- archivoAdjunto
- asignatura

- estado

2.8. Clase Respuesta

Solución entregada por un estudiante para una actividad.

Atributos:

- texto
- fechaEnvio

2.9. Clase Calificación

Evaluación otorgada a una respuesta.

Atributos:

- puntaje
- comentario
- fechaEvaluacion

2.10. Clase Foro

Espacio para la discusión académica.

Atributos:

- tema
- fechaCreacion

2.11. Clase MensajeForo

Mensaje publicado dentro de un foro.

Atributos:

- contenido
- fecha
- esPrivado

2.12. Clase Encuesta

Encuesta de satisfacción aplicada a estudiantes y representantes.

Atributos:

- título
- descripción
- fechaPublicación

2.13. Clase RespuestaEncuesta

Respuesta dada a una encuesta.

Atributos:

- respuesta

2.14. Clase HistorialAcadémico

Registro del rendimiento académico del estudiante.

Atributos:

- periodo
- resumen

2.15. Clase Reporte

Documento generado a partir de métricas académicas.

Atributos:

- tipo
- fechaGeneración

2.16. Clase Notificación

Mensaje enviado automáticamente para informar novedades.

Atributos:

- mensaje
- fechaEnvio

2.17. Clase ProgresoVisualización

Registro del porcentaje de visualización de un contenido.

Atributos:

- porcentaje
- fechaUltimaVisualización

3. Relaciones del Modelo de Análisis

A continuación se describen las relaciones existentes entre las clases del dominio:

1. Usuario – Notificación

- Un usuario puede recibir múltiples notificaciones.
- Cardinalidad: **1 — 0..***

2. Usuario – MensajeForo

- Un usuario puede publicar varios mensajes en el foro.
- Cardinalidad: **1 — 0..***

3. Docente – Contenido

- El docente publica contenidos académicos.
- Cardinalidad: **1 — 0..***

4. Contenido – ProgresoVisualización

- Cada contenido puede registrar múltiples progresos de visualización.
- Cardinalidad: **1 — 0..***

5. Estudiante – ProgresoVisualización

- El estudiante posee progresos de visualización asociados.
- Cardinalidad: **1 — 0..***

6. Docente – Actividad

- El docente asigna actividades a los estudiantes.
- Cardinalidad: **1 — 0..***

7. Actividad – Respuesta

- Una actividad puede tener varias respuestas de los estudiantes.
- Cardinalidad: **1 — 0..***

8. Respuesta – Calificación

- Toda respuesta tiene una única calificación.
- Cardinalidad: **1 — 1**

9. Estudiante – Respuesta

- Un estudiante puede enviar múltiples respuestas.
- Cardinalidad: **1 — 0..***

10. Estudiante – HistorialAcadémico

- Cada estudiante tiene un único historial académico.
- Cardinalidad: **1 — 1**

11. HistorialAcadémico – Calificación

- El historial registra varias calificaciones del estudiante.
- Cardinalidad: **1 — 0..***

12. Foro – MensajeForo

- Un foro contiene uno a múltiples mensajes.
- Cardinalidad: **1 — 1..***

13. Encuesta – RespuestaEncuesta

- Cada encuesta puede obtener múltiples respuestas.
- Cardinalidad: **1 — 0..***

14. Usuario – RespuestaEncuesta

- Los usuarios (estudiantes o representantes) responden encuestas.

- Cardinalidad: 1 — 0..*

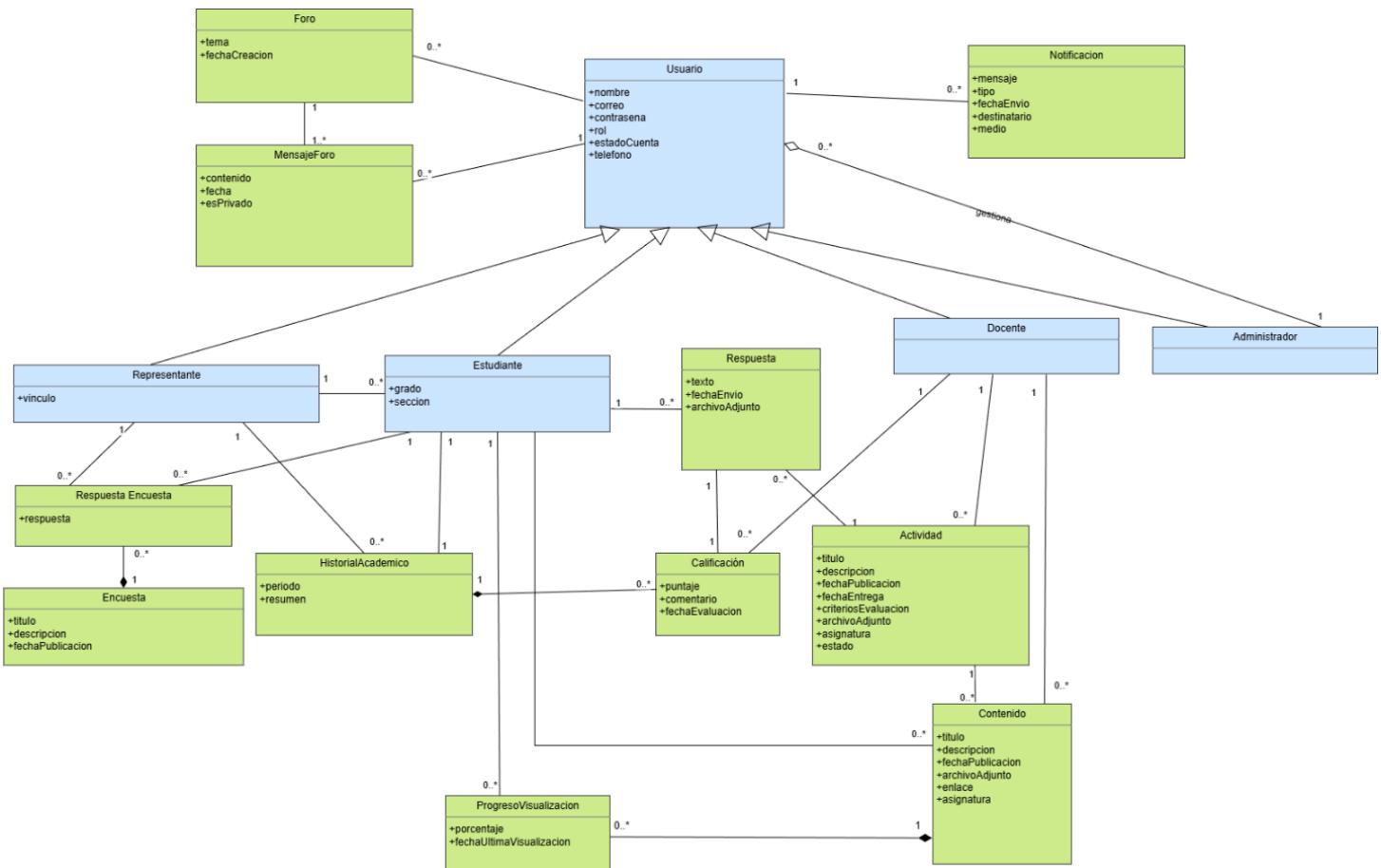
15. Representante – Estudiante

- Un representante puede estar asociado a uno o varios estudiantes.
- Cardinalidad: 1 — 0..*

Las relaciones del modelo se estructuran como sigue:

Los usuarios pueden recibir múltiples notificaciones y publicar mensajes en el foro. Los docentes publican contenidos y asignan actividades, mientras que los estudiantes generan respuestas que posteriormente son calificadas. Cada estudiante posee un historial académico que agrupa todas sus calificaciones. Los contenidos registran progresos de visualización asociados a los estudiantes.

Los foros contienen mensajes, las encuestas se relacionan con respuestas emitidas por usuarios, y los representantes pueden estar vinculados a varios estudiantes.



4. Modelo de diseño

El Modelo de Diseño tiene como propósito definir **cómo será implementado el sistema a nivel técnico**, tomando como base los elementos del Modelo de Análisis.

Mientras que el análisis describe los conceptos del dominio y sus relaciones, el diseño transforma esos conceptos en estructuras concretas que podrán programarse utilizando PHP, PostgreSQL y el patrón arquitectónico MVC.

Este modelo especifica:

- Las **clases técnicas** que implementarán la lógica del sistema (controladores, modelos, servicios).
- Las **interacciones entre componentes** a través de diagramas de secuencia.
- La **organización del código fuente** mediante paquetes o módulos.
- Las **operaciones, atributos y tipos de datos** necesarios para la implementación.
- La **traducción de clases del dominio** en clases de diseño alineadas con la arquitectura.
- La **correspondencia entre clases y tablas de la base de datos**.

El objetivo de este modelo es garantizar que la implementación sea coherente, modular, mantenable y alineada con los requisitos funcionales establecidos en el documento de especificación.