

UNIVERSIDAD CONTINENTAL
FACULTAD DE INGENIERÍA
ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA
DE SISTEMAS E INFORMÁTICA



PROYECTO

“LECTUM - TUTOR VIRTUAL DE LECTURA CRÍTICA”

PRESENTADO POR:

| APELLIDOS Y NOMBRES | CÓDIGO |
|--------------------------------------|----------|
| AUCCATOMA ROJAS, JACK JEYSON | 76418687 |
| CABALLERO CAPCHA, MARUJA MISTICA | 75006113 |
| CHAMBI RODRIGUEZ, FERNANDO MELITON | 73469930 |
| LÓPEZ CAÑETE, SEBASTIÁN MARIANO | 72771058 |
| OLIVARES COLLACHAGUA, JULDANT WALTER | 70765054 |
| QUISPE ZARATE, JULIO CESAR | 76746412 |

ASESOR:

HUANCAYO – PERÚ

2025

LISTA DE CONTENIDO

| | |
|---------------------------------------|----|
| PORADA | 1 |
| LISTA DE CONTENIDO | 2 |
| LISTA DE TABLAS | 11 |
| LISTA DE FIGURAS | 12 |
| CAPÍTULO 1 | 14 |
| PLANTEAMIENTO DEL ESTUDIO | 14 |
| 1.1. Aspectos Generales de la Empresa | 14 |
| 1.1.1.Organigrama | 14 |
| 1.1.2.Misión y visión | 15 |
| Misión 15 | |
| Visión 15 | |
| 1.2. Diagnóstico del Problema | 16 |
| 1.3. Procesos de la Empresa | 16 |
| 1.4. Oportunidad Encontrada | 17 |
| 1.5. Detalles del Proyecto | 17 |
| 1.5.1.Solución planteada | 17 |
| 1.5.2.Objetivos generales | 18 |
| CAPÍTULO 2 | 20 |
| ESTUDIO DE FACTIBILIDAD | 20 |
| 2.1. Alternativas de Solución | 20 |
| 2.2. Factibilidad Técnica | 21 |
| 2.2.1.Hardware: Servidor | 21 |
| 2.2.2.Factibilidad Económica | 22 |
| 2.3. Factibilidad Económica | 22 |
| 2.3.1.Gastos generales | 22 |
| 2.4. Factibilidad Operacional | 23 |
| 2.4.1.Sistemas de ventas | 24 |

| | |
|------------------------------------------------------------------------------------|-----------|
| Fase de análisis | 24 |
| Fase de diseño y desarrollo | 24 |
| Fase de transición | 24 |
| Fase de implementación | 24 |
| CAPÍTULO 3 | 26 |
| ANÁLISIS DE REQUERIMIENTOS | 26 |
| 3.1. Metas del Sistema de Información | 26 |
| 3.2. Requisitos del Sistema | 26 |
| 3.2.1. Requerimientos funcionales | 26 |
| 3.2.2. Requerimientos no funcionales | 27 |
| 3.3. Identificación de Actores del Sistema | 28 |
| 3.3.1. Estudiante | 28 |
| 3.3.2. Docente | 28 |
| 3.3.3. Administrador | 28 |
| 3.3.4. Sistema NLP / IA | 28 |
| 3.3.5. Sistema de Automatización (n8n) | 28 |
| CAPÍTULO 4 | 30 |
| PLANIFICACIÓN DEL PROYECTO | 30 |
| 4.1. Definición de Roles de Trabajo | 30 |
| 4.1.1. Product owner (Olivares Collachagua Juldant Walter) | 30 |
| 4.1.2. Scrum máster (Auccatoma Rojas Jack Jeysen) | 30 |
| 4.1.3. Team member (Quispe Zárate Julio Cesar - López Cañete Sebastián Mariano) | |
| 31 | |
| 4.1.4. Tester (Caballero Capcha Maruja Mistica - Chambi Rodriguez Fernando Malito) | |
| 31 | |
| 4.2. Product Backlog | 32 |
| 4.3. Sprint Backlog | 32 |
| 4.3.1. Sprint 1 (3 sept – 17 sept) | 33 |

| | |
|----------------------------------------------------|-----------|
| 4.3.2. Sprint 2 (23 sept – 27 oct) | 33 |
| 4.3.3. Sprint 3 (13 oct – 27 oct) | 33 |
| 4.4. Planificación de Sprints | 34 |
| 4.4.1. Historias de usuario | 34 |
| HU1 – Login de estudiantes | 34 |
| HU2 – Generación automática de preguntas | 34 |
| HU3 – Guardar historial de respuestas | 34 |
| HU4 – Recibir sugerencias de lectura | 34 |
| HU5 – Detectar sesgos o falacias | 34 |
| HU6 – Ver ejemplos de sesgo | 35 |
| HU7 – Reporte de sesgos por estudiante | 35 |
| HU8 – Señalar sesgo manualmente | 35 |
| HU9 – Asignar textos automáticamente a estudiantes | 35 |
| HU10 – Recibir notificaciones automáticas | 35 |
| HU11 – Panel con nivel de comprensión | 35 |
| HU12 – Configurar horarios | 36 |
| HU13 – Registrar respuestas en la base de datos | 36 |
| HU14 – Exportar resultados en PDF o Excel | 36 |
| 4.4.2. Priorización de historias de usuario | 36 |
| 4.5. Cronograma de Actividades | 37 |
| 4.6. Gestión de Riesgos | 1 |
| CAPÍTULO 5 | 1 |
| DISEÑO DEL SISTEMA DE INFORMACIÓN | 1 |
| 5.1. Diseño de Diagramas UML | 1 |
| Administrador | 1 |
| Estudiante | 2 |
| 5.1.1. Diagramas de secuencia | 2 |
| RNF1 – Usabilidad | 3 |

| | |
|--------------------------------------|----|
| RNF2. Rendimiento | 4 |
| RNF3. Seguridad | 5 |
| RNF4. Escalabilidad | 6 |
| RNF5. Disponibilidad | 7 |
| RNF6. Mantenibilidad | 8 |
| RNF7. Compatibilidad | 9 |
| 5.1.2. Diagramas de colaboración | 10 |
| 5.1.3. Diagramas de clases | 12 |
| 5.2. Diseño de Base de Datos | 13 |
| 5.2.1. Diseño conceptual (E/R) | 15 |
| 5.2.2. Diseño lógico | 17 |
| 5.2.3. Diseño físico | 18 |
| 5.2.4. Modelado de base de datos | 21 |
| 5.3. Diseño de Interfaces Básicas | 23 |
| 5.3.1. Acceso login | 23 |
| 5.3.2. Interfaz (crear una cuenta) | 24 |
| 5.3.3. Interfaz (Pantalla principal) | 24 |
| 5.3.4. Interfaz (Historial) | 25 |
| CAPÍTULO 6 | 28 |
| CODIFICACIÓN DEL SOFTWARE | 28 |
| 6.1. Desarrollo del Sprint 1 | 28 |
| 6.1.1. Sprint planning | 28 |
| Tabla – Story Points Sprint 1 | 28 |
| 6.1.2. Sprint backlog | 28 |
| Tabla – Sprint Backlog Sprint 1 | 29 |
| 6.1.3. Historias de usuarios | 29 |
| HU1 – Login Estudiantes | 29 |
| HU2 – Preguntas automáticas | 29 |

| | |
|----------------------------------------|----|
| HU2.1 – Retroalimentación inmediata | 29 |
| HU5 – Detección de sesgos | 29 |
| 6.1.4. Taskboard | 30 |
| 6.1.5. Daily scrum | 30 |
| ¿Qué se hizo? | 30 |
| ¿Qué se hará? | 30 |
| Impedimentos | 30 |
| 6.1.6. Sprint review | 31 |
| 6.1.7. Criterios de aceptación | 31 |
| 6.1.8. Resultados del sprint | 31 |
| 6.1.8.1. Evidencias. | 32 |
| 6.1.8.2. Prueba de desarrollo. | 36 |
| 6.1.9. Sprint retrospective | 36 |
| Categoría: Lo que salió bien | 36 |
| Lo que salió mal | 37 |
| Mejoras propuestas | 37 |
| 6.2. Desarrollo del Sprint 2 | 37 |
| 6.2.1. Sprint planning | 37 |
| 6.2.2. Sprint backlog | 38 |
| 6.2.3. Historias de usuarios | 38 |
| HU3 – Guardar historial del estudiante | 38 |
| HU6 – Ver ejemplos de sesgo | 38 |
| HU13 – Registrar respuestas | 38 |
| 6.2.4. Taskboard | 38 |
| 6.2.5. Daily scrum | 39 |
| Qué se hizo | 39 |
| Qué se hará | 39 |
| Impedimentos | 39 |

| | |
|-----------------------------------|----|
| 6.2.6. Sprint review | 39 |
| 6.2.7. Criterios de aceptación | 40 |
| 6.2.8. Resultados del sprint | 40 |
| 6.2.8.1. Evidencias. | 41 |
| 6.2.8.2. Prueba de desarrollo. | 43 |
| 6.2.9. Sprint retrospective | 43 |
| Lo que salió bien | 43 |
| Lo que salió mal | 44 |
| Mejoras propuestas | 44 |
| 6.3. Desarrollo del Sprint 3 | 44 |
| 6.3.1. Sprint planning | 44 |
| 6.3.2. Sprint backlog | 45 |
| 6.3.3. Historias de usuarios | 45 |
| HU12 – Configurar horarios | 45 |
| HU10 – Notificaciones automáticas | 45 |
| HU14 – Exportar datos | 45 |
| 6.3.4. Taskboard | 45 |
| 6.3.5. Daily scrum | 46 |
| Qué se hizo | 46 |
| Qué se hará | 46 |
| Impedimentos | 46 |
| 6.3.6. Sprint review | 46 |
| 6.3.7. Criterios de aceptación | 47 |
| 6.3.8. Resultados del sprint | 47 |
| 6.3.8.1. Evidencias. | 47 |
| 6.3.8.2. Prueba de desarrollo. | 50 |
| 6.3.9. Sprint retrospective | 50 |
| Lo que salió bien | 50 |

| | |
|-------------------------------------------------------------|-----------|
| Lo que salió mal | 50 |
| Mejoras | 50 |
| CAPÍTULO 7 | 51 |
| PRUEBAS DE SOFTWARE | 51 |
| 7.1. Plan de Pruebas | 51 |
| 7.1.1. Objetivo del Plan de Pruebas | 51 |
| 7.1.2. Alcance de las Pruebas | 52 |
| 7.1.3. Tipos de pruebas | 53 |
| 7.2. Ejecución de pruebas | 54 |
| 7.2.1. Pruebas unitarias Backend | 54 |
| Módulos evaluados: | 54 |
| Resultado general de pruebas | 55 |
| Cobertura del Back-End | 55 |
| 7.2.2. Pruebas unitarias Frontend | 56 |
| Las pruebas abarcaron: | 56 |
| Cobertura (Front-End) | 56 |
| 7.2.3. Pruebas de Integración | 58 |
| 7.2.4. Validación de Mock API | 59 |
| 7.2.5. Pruebas Funcionales de API (Postman / cURL) | 61 |
| Autenticación y Usuarios | 61 |
| Módulo de Análisis por IA | 62 |
| Historial y Métricas | 62 |
| Seguridad y Middleware | 63 |
| 7.3. Pruebas de Software Verde (Green Software) | 63 |
| 7.3.1. Herramientas Utilizadas y Código Implementado | 63 |
| Reducción relativa | 65 |
| Reducción por visita | 66 |
| CONCLUSIONES | 67 |

| | |
|--------------------------------------------------|----|
| RECOMENDACIONES | 69 |
| ANEXOS | 71 |
| Anexo 01. Manual Técnico | 71 |
| 1. Información General del Sistema | 71 |
| 2. Requerimientos Técnicos | 71 |
| 2.1. Requerimientos de Software | 71 |
| 2.2. Requerimientos de Hardware | 71 |
| 3. Arquitectura del Sistema | 71 |
| 3.1. Diagrama conceptual de arquitectura (texto) | 72 |
| 4. Instalación Técnica | 72 |
| 4.1. Instalación del sistema | 72 |
| 4.2. Variables de entorno | 72 |
| 5. Base de Datos | 72 |
| Ejemplo de registro analysis_history: | 73 |
| 6. Endpoints Principales (API interna) | 73 |
| 7. Seguridad | 73 |
| 8. Pruebas Técnicas | 74 |
| Pruebas unitarias (Jest + React Testing Library) | 74 |
| Cobertura de código | 74 |
| 9. Automatización | 74 |
| 10. Mantenimiento y Soporte | 74 |
| 11. Respaldo y Recuperación de Datos | 75 |
| 12. Versionamiento del Código | 75 |
| Anexo 02. Manual de Usuario | 76 |
| 1. Introducción | 76 |
| 2. Acceso al Sistema | 76 |
| b. Registro de usuario (primera vez) | 76 |
| b. Iniciar sesión | 76 |

| | | |
|-----|------------------------------------|----|
| c. | Recuperar contraseña (si aplica) | 76 |
| 3. | Vista Principal | 77 |
| 4. | Análisis de Texto | 77 |
| a. | Ingresar texto para análisis | 77 |
| b. | Detectar falacias y sesgos | 77 |
| c. | Generar cuestionario | 77 |
| 5. | Historial de Análisis | 78 |
| 6. | Perfil y Configuración del Usuario | 78 |
| 7. | Cierre de Sesión | 78 |
| 8. | Buenas Prácticas de Uso | 78 |
| 9. | Solución de Problemas Comunes | 79 |
| 10. | Soporte | 79 |

LISTA DE TABLAS

| | |
|-----------------------------------------------|----|
| Tabla 1 Modelo de CPU - Servidor..... | 22 |
| Tabla 2 Gastos generales | 22 |
| Tabla 3 Product Backlog Completo | 32 |
| Tabla 4 Sprint 1..... | 33 |
| Tabla 5 Sprint 2..... | 33 |
| Tabla 6 Sprint 3..... | 33 |
| Tabla 7 Cronograma General del Proyecto | 37 |
| Tabla 8 Matriz de Riesgo | 1 |
| Tabla 9 Story Points Sprint 1..... | 28 |
| Tabla 10 Sprint Backlog Sprint 1..... | 29 |
| Tabla 11 Criterios de aceptación..... | 31 |
| Tabla 12 Prueba de desarrollo | 36 |
| Tabla 13 Sprint planning | 37 |
| Tabla 14 Sprint 2 Backlog | 38 |
| Tabla 15 Criterios de aceptación..... | 40 |
| Tabla 16 Prueba de desarrollo | 43 |
| Tabla 17 Sprint planning | 44 |
| Tabla 18 Sprint backlog | 45 |
| Tabla 19 Criterios de aceptación..... | 47 |
| Tabla 20 Prueba de desarrollo | 50 |
| Tabla 21 Tipos de pruebas | 53 |

LISTA DE FIGURAS

| | |
|------------------------------------------------------|----|
| Ilustración 1 Organigrama del Proyecto LECTUM | 15 |
| Ilustración 2 Diagrama de casos de uso | 29 |
| Ilustración 3 Cronograma General del Proyecto..... | 38 |
| Ilustración 4 Diagramas de casos de uso..... | 1 |
| Ilustración 5 Diagramas de secuencia..... | 3 |
| Ilustración 6 Diagramas de Usabilidad | 4 |
| Ilustración 7 Rendimiento..... | 5 |
| Ilustración 8 Diagramas de seguridad | 6 |
| Ilustración 9 Diagramas de escalabilidad | 7 |
| Ilustración 10 Diagramas de disponibilidad | 8 |
| Ilustración 11 Diagramas de mantenibilidad | 9 |
| Ilustración 12 Diagramas de compatibilidad | 10 |
| Ilustración 13 Diagrama de colaboración | 10 |
| Ilustración 14 Diagrama de clase | 12 |
| Ilustración 15 Diagrama de clase | 14 |
| Ilustración 16 Diagrama de conceptual | 15 |
| Ilustración 17 Diseño lógico | 17 |
| Ilustración 18 Diseño físico | 19 |
| Ilustración 19 Base de datos - Users | 19 |
| Ilustración 20 Base de datos - Roles | 20 |
| Ilustración 21 Base de datos - Activities | 20 |
| Ilustración 22 Base de datos - Questions | 20 |
| Ilustración 23 Base de datos - Answers | 20 |
| Ilustración 24 Base de datos - IA interactions | 21 |
| Ilustración 25 Base de datos - Notifications..... | 21 |
| Ilustración 26 Diseño físico | 21 |
| Ilustración 27 Acceso login..... | 23 |

| | |
|----------------------------------------------------------------------|----|
| Ilustración 28 Interfaz (crear una cuenta) | 24 |
| Ilustración 29 Interfaz (Pantalla principal)..... | 25 |
| Ilustración 30 Interfaz (Historial)..... | 26 |
| Ilustración 31 Taskboard del Sprint 1 | 30 |
| Ilustración 32 Iniciar sesión..... | 32 |
| Ilustración 33 Crear cuenta | 33 |
| Ilustración 34 Ingresar texto | 33 |
| Ilustración 35 Capturas de preguntas generadas automáticamente. | 34 |
| Ilustración 36 Pantalla de retroalimentación..... | 35 |
| Ilustración 37 Vista de detección de sesgos..... | 36 |
| Ilustración 38 Taskboard | 39 |
| Ilustración 39 Progreso de puntuación | 41 |
| Ilustración 40 Captura de historial con fechas y puntajes..... | 41 |
| Ilustración 41 Vista de ejemplos de sesgos..... | 42 |
| Ilustración 42 Registro en BD (colección responses). | 43 |
| Ilustración 43 Taskboard (Jira)..... | 46 |
| Ilustración 44 Capturas de horarios configurados. | 48 |
| Ilustración 45 Progreso de puntuación | 48 |
| Ilustración 46 Archivos PDF/Excel generados..... | 49 |
| Ilustración 47 Pruebas unitarias Backend | 55 |
| Ilustración 48 Pruebas unitarias Frontend | 57 |
| Ilustración 49 Pruebas de Integración | 59 |
| Ilustración 50 Lista del proyecto verde | 64 |
| Ilustración 51 Pruebas de Software Verde | 65 |

CAPÍTULO 1

PLANTEAMIENTO DEL ESTUDIO

1.1. Aspectos Generales de la Empresa

El proyecto “Lectum – Tutor Virtual de Lectura Crítica” se desarrolla en el contexto educativo de la Universidad Continental y está orientado a instituciones académicas (escuelas, universidades y programas de formación) que buscan fortalecer las habilidades de lectura crítica y pensamiento analítico de sus estudiantes mediante el uso de tecnologías de información y herramientas de Inteligencia Artificial.

Actualmente, muchos docentes tienen una alta carga manual para preparar materiales, preguntas, retroalimentación y seguimiento del avance de sus estudiantes. Del mismo modo, los estudiantes no siempre cuentan con herramientas personalizadas que los acompañen en su proceso de lectura crítica de forma continua y automatizada.

Frente a esta realidad, surge la necesidad de diseñar e implementar un tutor virtual de lectura crítica, accesible desde la web, que apoye tanto a docentes como a estudiantes en la enseñanza y práctica de la comprensión lectora y el análisis de sesgos, argumentos y falacias.

1.1.1. Organigrama

Para la gestión y desarrollo del proyecto se ha definido una estructura organizacional basada en roles de proyecto, alineada a la metodología ágil Scrum:

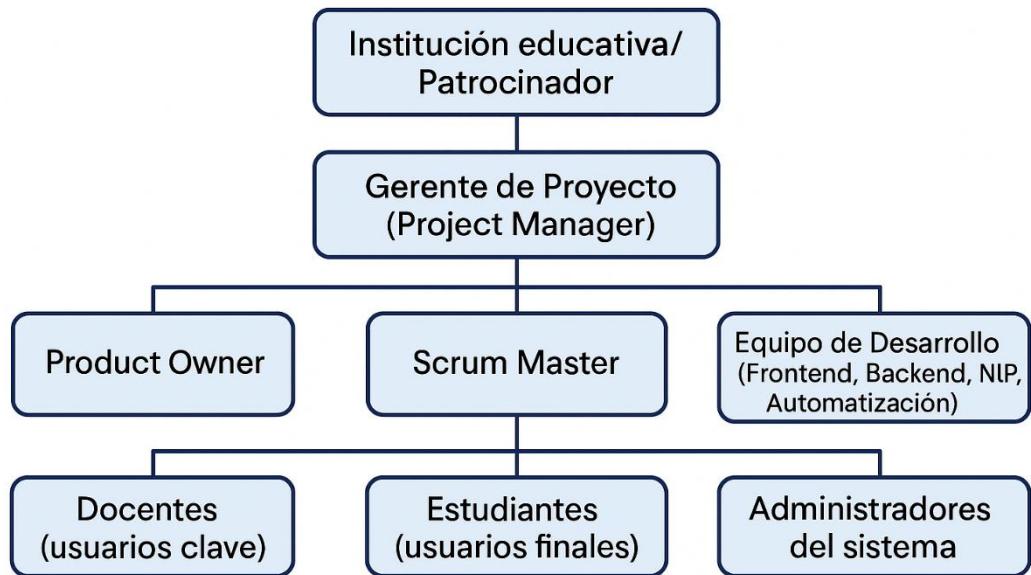


Ilustración 1 Organigrama del Proyecto LECTUM

Fuente. Elaboración propia.

El organigrama puede representarse de forma jerárquica, donde la Institución educativa/patrocinador se ubica en el nivel superior, seguida del Gerente de Proyecto y el Product Owner, y bajo ellos el Scrum Master, el Equipo de Desarrollo y los Administradores del sistema, trabajando de manera coordinada con docentes y estudiantes como usuarios del producto.

1.1.2. Misión y visión

Misión

Desarrollar un tutor virtual de lectura crítica que, mediante el uso de Inteligencia Artificial, procesamiento de lenguaje natural (NLP) y automatización de flujos, mejore las habilidades de comprensión lectora y pensamiento crítico de los estudiantes, brindando una herramienta accesible, escalable y fácil de usar para entornos educativos.

Visión

Convertirse en una plataforma de referencia a nivel institucional y regional para el desarrollo del pensamiento crítico, integrándose en escuelas, universidades y

programas de capacitación, y aportando valor a docentes, estudiantes y gestores educativos mediante el uso de tecnologías modernas y analíticas avanzadas.

1.2. Diagnóstico del Problema

En el diagnóstico realizado se identificaron los siguientes problemas principales en el proceso de enseñanza de la lectura crítica:

- Los docentes invierten una gran cantidad de tiempo en preparar preguntas, actividades y retroalimentación para cada texto leído.
- No existe, en muchos casos, un seguimiento automatizado del progreso del estudiante ni registro histórico detallado de sus respuestas y avances.
- La identificación de sesgos, falacias lógicas y debilidades argumentativas en los textos suele depender exclusivamente del criterio humano y del tiempo disponible del docente.
- No se cuenta con un sistema que automatice recordatorios, asignación de lecturas y generación de reportes de desempeño.
- La información sobre resultados, avances y estadísticas de comprensión lectora se encuentra dispersa, es difícil de consolidar y no siempre permite una toma de decisiones basada en datos.

Como consecuencia, se limitan las oportunidades para que los estudiantes desarrollen de forma sistemática sus habilidades de lectura crítica, y los docentes enfrentan una carga administrativa elevada que reduce el tiempo disponible para actividades pedagógicas de mayor valor.

1.3. Procesos de la Empresa

Los procesos actuales que se relacionan con el proyecto “Lectum – Tutor Virtual de Lectura Crítica” son:

- Asignación de textos y actividades por parte de docentes.
- Lectura de textos por estudiantes.

- Resolución de preguntas o ejercicios de comprensión.
- Revisión manual de respuestas y retroalimentación por parte del docente.
- Registro manual o parcial de notas, resultados y progresos.
- Comunicación de tareas, fechas límite y recordatorios (generalmente por medios informales).

Estos procesos se realizan de forma fragmentada, apoyándose en diferentes herramientas (documentos, plataformas de mensajería, hojas de cálculo, etc.), lo que provoca desorganización, pérdida de información y poca trazabilidad de los resultados de aprendizaje.

1.4. Oportunidad Encontrada

A partir del análisis de la situación actual se identifica la oportunidad de:

- Diseñar e implementar un tutor virtual de lectura crítica que integre:
 - Generación automática de preguntas sobre textos.
 - Retroalimentación inmediata para el estudiante.
 - Historial de respuestas y progreso.
 - Detección de sesgos y falacias mediante IA.
 - Automatización de asignación de textos y recordatorios mediante n8n.
 - Paneles de métricas y reportes para docentes y administradores

La disponibilidad de tecnologías como NLP, servicios en la nube, automatización de flujos y entornos web modernos permite construir una solución que mejore significativamente el proceso de enseñanza-aprendizaje, reduzca la carga manual del docente y brinde a los estudiantes una experiencia personalizada y constante de práctica en lectura crítica.

1.5. Detalles del Proyecto

1.5.1. Solución planteada

La solución propuesta es el desarrollo del sistema “Lectum – Tutor Virtual de Lectura Crítica”, una plataforma web (y potencialmente móvil) que:

- Permite a los estudiantes ingresar o seleccionar textos y responder preguntas generadas automáticamente.
- Ofrece retroalimentación inmediata (correcto/incorrecto, explicación, sugerencias).
- Registra el historial de respuestas y puntajes, mostrando reportes de progreso.
- Implementa un módulo de detección de sesgos y falacias lógicas, resaltando fragmentos del texto y clasificándolos.
- Integra flujos de automatización con n8n, para:
 - Asignar lecturas y actividades.
 - Enviar recordatorios a estudiantes.
 - Consolidar y registrar resultados en la base de datos.
- Proporciona un panel administrativo con métricas de desempeño (niveles de comprensión, evolución del grupo, estadísticas por texto, etc.).
- Garantiza seguridad y privacidad en el manejo de los datos de usuarios, cumpliendo normas y buenas prácticas de protección de datos.

Tecnológicamente, el sistema se apoya en una arquitectura moderna basada en MERN (React, Node.js, MongoDB), servicios de NLP/IA y herramientas de automatización (n8n), lo que permite escalabilidad, mantenibilidad y evolución futura del producto.

1.5.2. Objetivos generales

Los objetivos generales del proyecto son:

- Desarrollar un tutor virtual funcional de lectura crítica en un plazo establecido, capaz de operar en un entorno real de pruebas con estudiantes y docentes.
- Mejorar las habilidades de comprensión crítica de los usuarios, logrando que un porcentaje significativo de estudiantes incremente su puntaje en pruebas de lectura crítica antes y después del uso del sistema.

- Automatizar procesos clave relacionados con la asignación de textos, envío de recordatorios, registro de respuestas y generación de reportes mediante flujos en n8n.
- Obtener una validación positiva por parte de docentes, estudiantes y administradores, tanto en usabilidad como en utilidad pedagógica, asegurando la adopción y continuidad del uso del sistema en el tiempo.

CAPÍTULO 2

ESTUDIO DE FACTIBILIDAD

2.1. Alternativas de Solución

Para el desarrollo del sistema “Lectum – Tutor Virtual de Lectura Crítica” se evaluaron diversas alternativas tecnológicas que permitieran integrar:

- Procesamiento de lenguaje natural (NLP)
- Automatización de flujos educativos
- Gestión de usuarios (docentes, estudiantes, administradores)
- Paneles de análisis y reportes
- Interfaz web dinámica y responsive

Las alternativas consideradas fueron:

Alternativa 1: Plataforma basada en LMS tradicional (Moodle, Chamilo)

- **Ventajas:** Integración de plugins existentes y comunidad activa.
- **Desventajas:** Limitada personalización, poca capacidad para IA y n8n, y curva de modificación elevada.

Alternativa 2: Utilizar plataformas existentes (Notion, Trello, Miro, Asana)

con integraciones externas de IA

Aprovechar sistemas ya desarrollados orientados a la gestión de información y colaboración, integrándolos con herramientas de IA a través de APIs o extensiones externas. Esta opción reduce la necesidad de desarrollo desde cero, aprovechando infraestructura y funciones listas para usar.

Ventajas

- ✓ Implementación rápida y con baja complejidad técnica.
- ✓ Interfaz y funcionalidad ya probadas y utilizadas en entornos profesionales.

- ✓ Disponibilidad de integraciones y automatizaciones preconfiguradas.
- ✓ Menor costo inicial en tiempo y capacitación.

Desventajas

- Capacidad muy limitada de personalización, restringida por las funciones del proveedor.
- Dependencia directa de terceros, lo que puede afectar continuidad, privacidad o acceso a datos.
- Costos recurrentes por licencias, almacenamiento adicional y funciones premium.
- Imposibilidad de incorporar optimizaciones avanzadas como medición energética o software verde a nivel de código.

Alternativa 3: Arquitectura moderna MERN + Integraciones (React, Node.js, MongoDB, n8n)

- **Ventajas:** Rendimiento alto, escalabilidad, integración fluida con APIs NLP, automatización n8n, interfaz moderna.
- **Desventajas:** Requiere mayor planificación y coordinación de módulos.

Alternativa Seleccionada:

- Alternativa 3, debido a su alta escalabilidad, integración con NLP y su flexibilidad para automatizar procesos educativos.

2.2. Factibilidad Técnica

La factibilidad técnica se evaluó considerando hardware, software y compatibilidad con los módulos de IA, automatización y panel de métricas.

2.2.1. Hardware: Servidor

El proyecto requiere un servidor moderno capaz de manejar análisis NLP, consultas simultáneas y flujos automáticos.

Para este caso se proyecta un servidor en la nube (Render, Vercel o AWS) con las siguientes características equivalentes:

Tabla 1 Modelo de CPU - Servidor.

| Características | Valores requeridos |
|-----------------|-------------------------------------|
| Núcleos de CPU | 4 – 8 vCPUs |
| Memoria RAM | 8 – 16 GB |
| Almacenamiento | 40 – 80 GB SSD |
| GPU (opcional) | En nube (para NLP) |
| Conectividad | 1 Gbps |
| Base de datos | MongoDB Atlas – Cluster Free/Shared |

Fuente. Elaboración propia.

Estas especificaciones aseguran:

- Procesamiento eficiente de modelos NLP.
- Atención a múltiples usuarios concurrentes.
- Flujo continuo de automatizaciones en n8n.
- Almacenamiento seguro y escalable.

2.2.2. Factibilidad Económica

Para estimar la viabilidad económica se proyectan los costos del desarrollo y mantenimiento del sistema durante la etapa inicial.

2.3. Factibilidad Económica

Los gastos generales del proyecto “Lectum – Tutor Virtual de Lectura Crítica” se han estimado considerando las actividades desarrolladas en cada fase del ciclo de vida del proyecto: Inicio, Planificación, Planificación 2, Desarrollo y Pruebas & Entrega. Estos valores provienen del presupuesto consolidado trabajado por el equipo del proyecto.

2.3.1. Gastos generales

Tabla 2 Gastos generales

| Fase del Proyecto | Subtotal (S/.) | Descripción General |
|--------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------|
| Inicio | S/ 4,000 | Configuración del entorno, reuniones iniciales y elaboración del acta de constitución. |
| Planificación | S/ 5,500 | Priorización del backlog, definición de historias de usuario y planificación de sprints. |
| Planificación 2 | S/ 3,500 | Ajuste de dependencias, definición de riesgos y aprobación del MVP. |
| Desarrollo | S/ 23,500 | Implementación de funcionalidades principales (HU1–HU13), integración NLP, automatizaciones con n8n y panel de métricas. |
| Pruebas y Entrega | S/ 7,220 | Pruebas finales, despliegue, documentación, capacitación de usuarios y validación de usabilidad. |

Fuente. Elaboración propia.

2.4. Factibilidad Operacional

La factibilidad operacional evalúa si los usuarios finales (docentes, estudiantes y administradores) podrán usar y adoptar el sistema sin dificultades.

Para LECTUM, esta factibilidad es **alta**, debido a que:

- El sistema está diseñado con una interfaz intuitiva, desarrollada en React.
- Funciona en cualquier navegador o dispositivo (PC o móvil).
- Se realizaron pruebas funcionales y de cobertura tanto en backend como en frontend.
 - Back-end con coberturas superiores al 80% de statements y lines
 - Front-end con coberturas superiores al 70% y 33 pruebas pasadas
- Las funcionalidades coinciden con los sprints completados:
 - HU1. Login estudiantes
 - HU2. Preguntas automáticas
 - HU5. Detección de sesgos
 - HU3. Historial
 - HU6. Repositorio de sesgos

- HU13. Registrar a BD
 - HU12/HU10/HU14 del Sprint 3
- El uso de n8n automatiza notificaciones, envío de recordatorios y registro de actividades.

2.4.1. Sistemas de ventas

Siguiendo la estructura del ejemplo Dolphis, estas son las actividades del sistema LECTUM:

Fase de análisis

- Revisión de necesidades de docentes.
- Identificación de procesos actuales de lectura y retroalimentación.
- Definición de epics y user stories.
- Análisis de riesgos iniciales.

Fase de diseño y desarrollo

- Diseño de interfaces React.
- Implementación de rutas API (auth, history, analyze, admin).
- Integración de análisis de sesgos.
- Diseño de flujos automatizados con n8n.
- Desarrollo de repositorio de sesgos.
- Desarrollo de exportación en PDF/Excel.

Fase de transición

- Validación con usuarios demo.
- Pruebas unitarias y de integración (backend >80%, frontend >70%).
- Ajustes en UI y rendimiento.
- Documentación técnica y manuales.

Fase de implementación

- Despliegue en Vercel/Render.

- Activación de flujos de automatización.
- Capacitación a docentes y estudiantes.
- Monitoreo continuo.

CAPÍTULO 3

ANÁLISIS DE REQUERIMIENTOS

3.1. Metas del Sistema de Información

El sistema “Lectum – Tutor Virtual de Lectura Crítica” tiene como metas principales:

- Mejorar las habilidades de análisis crítico de los estudiantes mediante herramientas automatizadas basadas en NLP.
- Facilitar a los docentes el monitoreo del avance estudiantil mediante reportes, historial y detección de sesgos.
- Automatizar la generación de preguntas, retroalimentación y notificaciones académicas mediante flujos n8n.
- Proporcionar una plataforma web accesible, segura y estable para estudiantes, docentes y administradores.
- Registrar y almacenar el historial académico del estudiante, incluyendo respuestas, sesgos detectados y puntuaciones.
- Permitir la exportación de resultados en formatos PDF y Excel para facilitar la gestión educativa.

3.2. Requisitos del Sistema

3.2.1. Requerimientos funcionales

- **RF1. Autenticación de estudiantes:** El sistema debe permitir que los estudiantes inicien sesión mediante credenciales registradas.
- **RF2. Análisis automático de textos:** El estudiante podrá ingresar un texto y el sistema lo analizará usando NLP para generar preguntas y retroalimentación.
- **RF3. Generación de preguntas automáticas:** El sistema debe generar preguntas de análisis crítico basadas en el texto ingresado.
- **RF4. Retroalimentación inmediata:** El sistema mostrará retroalimentación sobre las respuestas del estudiante, indicando aciertos y errores.

- **RF5. Detección de sesgos o falacias:** El módulo NLP debe identificar sesgos presentes en el texto y clasificarlos en categorías predefinidas.
- **RF6. Historial del estudiante:** El sistema almacenará todas las sesiones, respuestas y puntajes del estudiante.
- **RF7. Visualización de ejemplos de sesgos:** El estudiante podrá acceder a un repositorio de sesgos con definiciones y ejemplos.
- **RF8. Registro de respuestas en la base de datos:** Cada interacción del estudiante será registrada para análisis y reportes posteriores.
- **RF9. Configuración de horarios del estudiante:** El usuario podrá registrar horarios para recibir recordatorios automáticos.
- **RF10. Notificaciones automatizadas (n8n):** El sistema debe enviar recordatorios y avisos mediante flujos automatizados integrados con n8n.
- **RF11. Exportación de datos:** El sistema permitirá exportar el historial de análisis crítico en Excel o PDF.

3.2.2. Requerimientos no funcionales

- **RNF1. Usabilidad:** El sistema debe ser intuitivo, fácil de usar y accesible desde navegadores y dispositivos móviles.
- **RNF2. Rendimiento:** Las operaciones de análisis de texto y visualización deben ejecutarse en menos de 3 segundos en condiciones normales.
- **RNF3. Seguridad:** Debe implementarse autenticación segura, cifrado HTTPS y manejo de sesiones protegido.
- **RNF4. Escalabilidad:** La arquitectura MERN debe permitir ampliar funcionalidades sin afectar el rendimiento.
- **RNF5. Disponibilidad:** El sistema debe operar 24/7 con disponibilidad de al menos 95%.
- **RNF6. Mantenibilidad:** El código deberá estar modularizado, documentado y con pruebas unitarias.

- **RNF7. Compatibilidad:** Debe funcionar correctamente en los navegadores Chrome, Edge y Firefox.

3.3. Identificación de Actores del Sistema

Los actores definidos para el sistema son:

3.3.1. Estudiante

- Utiliza el sistema para cargar textos, responder cuestionarios, ver sesgos detectados y revisar su historial.
- Recibe notificaciones, retroalimentación y puede exportar sus resultados.

3.3.2. Docente

- Supervisa el progreso de los estudiantes.
- Accede a paneles de métricas y repositorio de sesgos.
- Valida, revisa y analiza resultados académicos.

3.3.3. Administrador

- Administra usuarios (estudiantes y docentes).
- Supervisa configuración, módulos y repositorios.
- Monitorea flujos automatizados y mantenimiento general del sistema.

3.3.4. Sistema NLP / IA

- Analiza textos para detectar sesgos.
- Genera preguntas y retroalimentación inteligente.
- Apoya el proceso de aprendizaje crítico.

3.3.5. Sistema de Automatización (n8n)

- Envía recordatorios y notificaciones.
- Ejecuta flujos automáticos de registro y validación.
- Conecta diferentes módulos del sistema para tareas repetitivas.

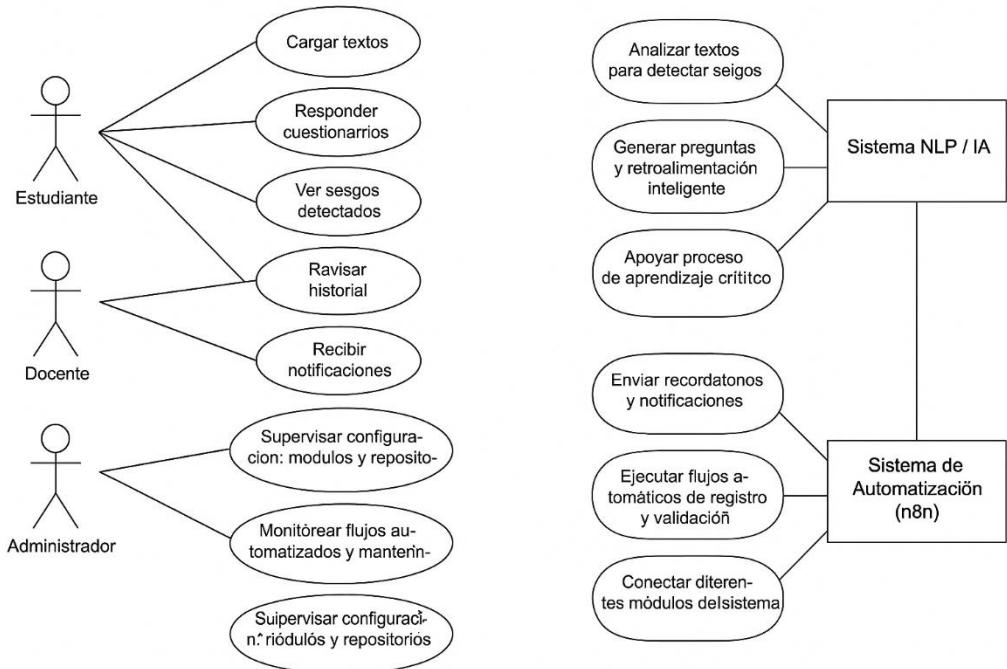


Ilustración 2 Diagrama de casos de uso

Fuente. Elaboración propia.

CAPÍTULO 4

PLANIFICACIÓN DEL PROYECTO

4.1. Definición de Roles de Trabajo

En el desarrollo del sistema “Lectum – Tutor Virtual de Lectura Crítica”, se asignaron roles bajo la metodología ágil Scrum. Cada rol tiene responsabilidades específicas orientadas a garantizar que las iteraciones y entregables cumplan con los objetivos funcionales y de calidad del proyecto.

4.1.1. Product owner (Olivares Collachagua Juldant Walter)

El Product Owner es responsable de maximizar el valor del producto y asegurar que lo desarrollado se alinee con las necesidades de los usuarios finales. Sus funciones fueron:

- Definir y priorizar el Product Backlog según impacto educativo y técnico.
- Alinear las funcionalidades con los objetivos de mejora de lectura crítica.
- Obtener retroalimentación de docentes y usuarios piloto.
- Evaluar las historias de usuario completadas al final de cada sprint.
- Asegurar que se cumplan los criterios de aceptación establecidos.

El PO tiene comunicación directa con el equipo y toma decisiones sobre qué se construye en cada iteración.

4.1.2. Scrum máster (Auccatoma Rojas Jack Jeysen)

El Scrum Master actúa como facilitador del equipo y garante del proceso Scrum.

Sus funciones incluyeron:

- Asegurar que se cumplan las ceremonias ágiles (Daily, Planning, Review y Retrospective).
- Remover impedimentos que afecten el avance del equipo.
- Guiar al equipo en el uso de herramientas como ClickUp y en el seguimiento del Sprint Backlog.

- Promover la colaboración, comunicación eficaz y mejora continua.
- Supervisar el cumplimiento del ritmo y la calidad de los entregables.

4.1.3. Team member (Quispe Zárate Julio Cesar - López Cañete Sebastián

Mariano)

El equipo de desarrollo construyó, probó e integró todas las funcionalidades del sistema. Sus responsabilidades fueron:

- Desarrollar el frontend (React) y backend (Node.js) según las HUs asignadas.
- Integrar el módulo de IA/NLP para detección de sesgos
- Implementar flujos automatizados en n8n.
- Diseñar la experiencia de usuario (UI/UX)
- Realizar pruebas unitarias y de integración.
- Documentar código, endpoints y flujos del sistema.

El equipo trabajó en pares cuando la complejidad técnica lo requirió.

4.1.4. Tester (Caballero Capcha Maruja Mistica - Chambi Rodriguez Fernando

Malito)

El rol de Tester garantiza que lo desarrollado cumpla estándares de calidad. Sus principales responsabilidades fueron:

- Diseñar casos de prueba basados en criterios de aceptación.
- Ejecutar pruebas unitarias (frontend y backend).
- Validar endpoints críticos como /api/analyze, /api/history, /api/admin.
- Verificar cobertura mínima (>70% frontend, >80% backend).
- Documentar resultados y reportar incidencias.
- Revisar funcionalidad en el entorno de producción (Vercel/Render).

Este rol es clave para garantizar un sistema estable y funcional.

4.2. Product Backlog

El Product Backlog contiene todas las funcionalidades del sistema, clasificadas y priorizadas. Se elaboró considerando las metas del proyecto: análisis crítico, automatización, historial y retroalimentación.

A continuación, se presenta el Backlog completo:

Tabla 3 Product Backlog Completo

| ID | Historia de Usuario | Epic | Priorización |
|------|----------------------------------|------|--------------|
| HU1 | Login de estudiantes | EP1 | Must Have |
| HU2 | Preguntas automáticas | EP1 | Must Have |
| HU3 | Guardar historial | EP1 | Must Have |
| HU4 | Recibir sugerencias de lectura | EP1 | Could Have |
| HU5 | Detectar sesgos o falacias | EP2 | Must Have |
| HU6 | Ver ejemplos de sesgo | EP2 | Should Have |
| HU7 | Reporte de sesgos de estudiantes | EP2 | Could Have |
| HU8 | Señalar sesgo manualmente | EP2 | Could Have |
| HU9 | Asignar textos automáticamente | EP3 | Should Have |
| HU10 | Notificaciones automáticas | EP3 | Should Have |
| HU11 | Panel con nivel de comprensión | EP3 | Could Have |
| HU12 | Configurar horarios | EP3 | Could Have |
| HU13 | Registrar respuestas en BD | EP3 | Must Have |
| HU14 | Exportar PDF/Excel | EP3 | Could Have |

Fuente. Elaboración propia.

4.3. Sprint Backlog

Solo se desarrollaron 3 sprints, pero se muestra el detalle completo según el avance real del proyecto.

4.3.1. Sprint 1 (3 sept – 17 sept)

Objetivo: Implementar la funcionalidad base del tutor de lectura crítica.

Tabla 4 Sprint 1

| Código | HU | Estado | Esfuerzo |
|----------|----------------------------|------------|----------|
| PRDT2-19 | HU1: Login Estudiantes | Finalizada | 5 |
| PRDT2-1 | HU1: Preguntas automáticas | Finalizada | 10 |
| PRDT2-3 | HU2: Retroalimentación | Finalizada | 8 |
| PRDT2-6 | HU5: Detectar sesgos | Finalizada | 10 |

Fuente. Elaboración propia.

4.3.2. Sprint 2 (23 sept – 27 oct)

Objetivo: Persistencia, historial y repositorio de sesgos.

Tabla 5 Sprint 2

| Código | HU | Estado | Esfuerzo |
|----------|------|------------|----------|
| PRDT2-4 | HU3 | Finalizada | 6 |
| PRDT2-7 | HU6 | Finalizada | 9 |
| PRDT2-14 | HU13 | Finalizada | 6 |

Fuente. Elaboración propia.

4.3.3. Sprint 3 (13 oct – 27 oct)

Objetivo: Automatización, horarios y exportación.

Tabla 6 Sprint 3

| Código | HU | Estado | Esfuerzo |
|----------|------|------------|----------|
| PRDT2-13 | HU12 | Finalizada | 5 |
| PRDT2-12 | HU10 | Finalizada | 8 |
| PRDT2-15 | HU14 | Finalizada | 10 |

Fuente. Elaboración propia.

4.4. Planificación de Sprints

4.4.1. Historias de usuario

HU1 – Login de estudiantes

Como estudiante,

quiero iniciar sesión en la plataforma,

para acceder a las funcionalidades del tutor virtual.

HU2 – Generación automática de preguntas

Como estudiante,

quiero recibir preguntas generadas automáticamente a partir de un texto ingresado,

para evaluar mi nivel de comprensión lectora.

HU3 – Guardar historial de respuestas

Como estudiante,

quiero que mis respuestas y puntajes se registren en un historial,

para revisar mi progreso en lectura crítica.

HU4 – Recibir sugerencias de lectura

Como estudiante,

quiero recibir sugerencias de textos relacionados a mi nivel,

para mejorar gradualmente mis habilidades de comprensión.

HU5 – Detectar sesgos o falacias

Como estudiante,

quiero que el sistema identifique sesgos o falacias en el texto,

para desarrollar pensamiento crítico sobre la información.

HU6 – Ver ejemplos de sesgo

Como estudiante,

quiero visualizar ejemplos de diferentes tipos de sesgos,

para comprenderlos mejor y reconocerlos en otros textos.

HU7 – Reporte de sesgos por estudiante

Como docente,

quiero ver un reporte de los sesgos detectados en los textos analizados por mis estudiantes,

para evaluar su desempeño en lectura crítica.

HU8 – Señalar sesgo manualmente

Como estudiante o docente,

quiero señalar manualmente un sesgo en un texto,

para complementar o corregir el análisis automático del sistema.

HU9 – Asignar textos automáticamente a estudiantes

Como docente,

quiero que el sistema asigne textos automáticamente según el nivel del estudiante,

para personalizar la enseñanza y optimizar el aprendizaje.

HU10 – Recibir notificaciones automáticas

Como estudiante,

quiero recibir recordatorios y alertas generadas por n8n,

para mantener una rutina constante de práctica de lectura crítica.

HU11 – Panel con nivel de comprensión

Como docente,

quiero visualizar un panel con el nivel de comprensión de los estudiantes,

para monitorear su progreso por indicadores.

HU12 – Configurar horarios

Como estudiante,

quiero configurar mis horarios de estudio,

para recibir notificaciones adaptadas a mis tiempos disponibles.

HU13 – Registrar respuestas en la base de datos

Como sistema,

quiero registrar todas las respuestas del estudiante,

para generar historial, reportes y métricas.

HU14 – Exportar resultados en PDF o Excel

Como estudiante,

quiero exportar mis resultados y análisis en PDF o Excel,

para archivarlos, imprimirlos o compartirlos.

4.4.2. Priorización de historias de usuario

Must Have (indispensables)

- HU1, HU2, HU3, HU5, HU13

Should Have (muy importantes)

- HU6, HU9, HU10

Could Have (útiles, no críticas)

- HU4, HU7, HU8, HU11, HU12, HU14

Won't Have

- Ninguna descartada

4.5. Cronograma de Actividades

Tabla 7 Cronograma General del Proyecto

| Fase / Sprint | Fechas | Historia(s) de Usuario | Actividad Desarrollada |
|---------------------------|----------------------------|-------------------------------------------------|----------------------------------------------------------------|
| Sprint 1 | 3 – 17 septiembre | HU1 | Implementación de login y autenticación de estudiantes. |
| | | HU2 | Generación automática de preguntas mediante NLP. |
| | | HU5 | Detección automática de sesgos y falacias en textos. |
| | | HU3 <i>(componente de retroalimentación)</i> | Retroalimentación automática según respuestas del estudiante. |
| Sprint 2 | 23 septiembre – 27 octubre | HU3 | Desarrollo del módulo para guardar historial de respuestas. |
| | | HU6 | Implementación del repositorio de ejemplos de sesgos. |
| | | HU13 | Registro de respuestas y sesiones en la base de datos MongoDB. |
| Sprint 3 | 13 – 27 octubre | HU12 | Configuración y almacenamiento de horarios del estudiante. |
| | | HU10 | Automatización de notificaciones y recordatorios con n8n. |
| | | HU14 | Exportación de resultados a PDF y Excel. |
| Backlog (No desarrollado) | Pendiente | HU4 | Sugerencias de lectura personalizadas según nivel. |
| | | HU11 | Panel docente con nivel de comprensión de estudiantes. |
| | | HU9 | Asignación automática de textos según desempeño. |
| | | HU8 | Herramienta para señalar sesgos de forma manual. |
| | | HU7 | Reporte completo de sesgos detectados para docentes. |

| | | | |
|----------------------------|------------------------|---|----------------------------------------------------------|
| Cierre del Proyecto | 27 – 29 octubre | — | Pruebas finales del sistema (QA) y correcciones finales. |
| | | — | Documentación técnica y funcional del sistema. |

Fuente. Elaboración propia.

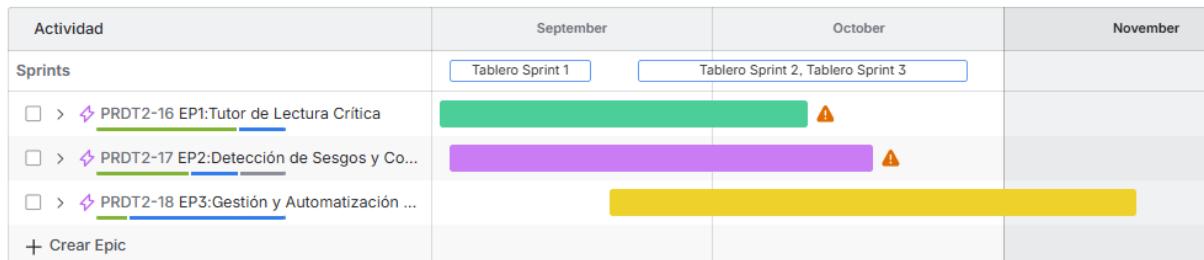


Ilustración 3 Cronograma General del Proyecto

Fuente. Elaboración propia.

4.6. Gestión de Riesgos

Tabla 8 Matriz de Riesgo

| ID Riesgo | Descripción del Riesgo | Área de Impacto | Causa | Impacto | Probabilidad | Puntuación de Riesgo | Detectabilidad | Estado | Asignado a |
|-----------|-----------------------------------------------------|-----------------|----------------------------------------|-----------|--------------|----------------------|----------------|-----------|-------------------|
| 250201a | Retraso en el desarrollo del motor de recomendación | Cronograma | Complejidad del algoritmo | Critico | Muy Alto | 8.1 | Alto | Activo | Juldant Walter |
| 250201b | Baja adopción por parte de estudiantes | Alcance | Falta de motivación o gamificación | Muy Serio | Muy Alto | 6.3 | Alto | Activo | Sebastián Mariano |
| 250201c | Sesgos en la evaluación automática | Calidad | Dataset no representativo | Serio | Muy Alto | 4.5 | Alto | Observado | Jack Jeysen |
| 250201d | Fallas del servidor en horas pico | Cronograma | Infraestructura insuficiente | Moderado | Muy Alto | 2.7 | Medio | Observado | Maruja Mistica |
| 250201e | Plagio de contenidos educativos | Calidad | Falta de filtros y verificación | Menor | Muy Alto | 0.9 | Bajo | Cerrado | Fernando Meliton |
| 250201f | Incumplimiento de normativa de protección de datos | Calidad | Gestión inadecuada de datos personales | Moderado | Alto | 2.1 | Medio | Activo | Juldant Walter |

| | | | | | | | | | |
|---------|-----------------------------------------|------------|--------------------------------|-----------|----------|-----|-------|-----------|-------------------|
| 250201g | Superestimación del presupuesto | Costo | Costos tecnológicos variables | Moderado | Bajo | 0.9 | Bajo | Observado | Sebastián Mariano |
| 250201h | Falta de entrenamiento docente | Cronograma | Escasa capacitación inicial | Muy Serio | Alto | 4.9 | Alto | Activo | Jack Jeysen |
| 250201i | Evaluaciones no alineadas al currículo | Alcance | Falta de validación pedagógica | Muy Serio | Bajo | 2.1 | Medio | Activo | Maruja Mistica |
| 250201j | Mala experiencia de usuario | Calidad | Interfaz poco intuitiva | Serio | Medio | 2.5 | Medio | Activo | Fernando Meliton |
| 250201k | Dependencia excesiva de un proveedor IA | Costo | Servicio externo único | Muy Serio | Medio | 3.5 | Medio | Observado | Juldant Walter |
| 250201l | Errores en reconocimiento de texto | Calidad | OCR poco preciso | Critico | Muy Bajo | 0.9 | Bajo | Activo | Sebastián Mariano |
| 250201m | Filtración de respuestas y actividades | Calidad | Brechas de seguridad | Muy Serio | Muy Bajo | 0.7 | Bajo | Activo | Jack Jeysen |
| 250201n | Cambios frecuentes del patrocinador | Alcance | Requerimientos inestables | Critico | Muy Alto | 8.1 | Alto | Activo | Maruja Mistica |
| 250201o | Desmotivación del equipo del proyecto | Cronograma | Sobrecarga laboral | Muy Serio | Alto | 4.9 | Alto | Observado | Fernando Meliton |

Fuente. Elaboración propia.

https://docs.google.com/spreadsheets/d/1T6dK3ZkCtp_qWiCUAD6DyppQXvpedYAD/edit?usp=sharing&ouid=105609867301868072148&rtpof=true&sd=true

CAPÍTULO 5

DISEÑO DEL SISTEMA DE INFORMACIÓN

5.1. Diseño de Diagramas UML

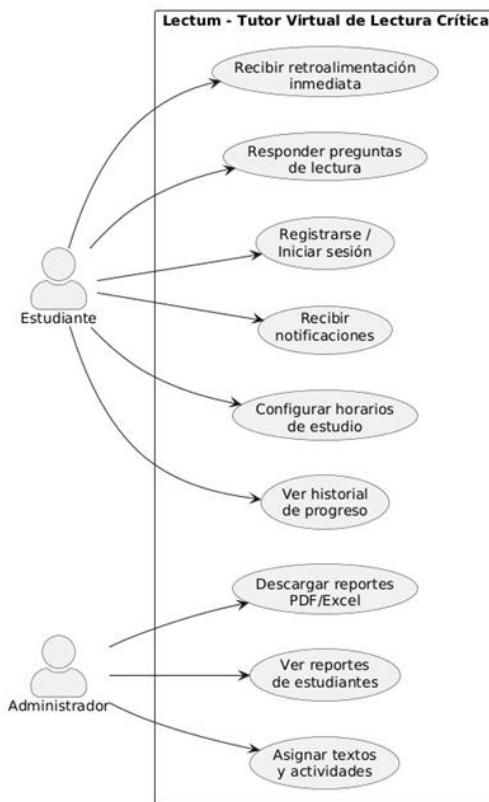


Ilustración 4 Diagramas de casos de uso

Fuente. Elaboración propia.

Administrador

Usuario con nivel avanzado de permisos dentro del sistema, encargado de supervisar y mantener el correcto funcionamiento de la plataforma Lectum. Su responsabilidad principal es administrar la operación general del sistema, gestionar usuarios, asignar actividades y monitorear el rendimiento académico de los estudiantes.

Tiene la capacidad de cargar textos, asignar actividades de lectura crítica, visualizar reportes individuales y grupales, evaluar el avance de los estudiantes y generar informes descargables en distintos formatos.

Además, gestiona configuraciones internas como la programación de actividades, la verificación de datos almacenados y la organización del contenido educativo. Este rol garantiza que los flujos de trabajo funcionen adecuadamente, corrige incidencias de uso y actúa como coordinador operativo, asegurando que el entorno de aprendizaje sea eficiente, organizado y accesible para todos los participantes.

Estudiante

Usuario principal del sistema cuyo propósito es interactuar con el tutor virtual para mejorar sus habilidades de lectura crítica. El estudiante accede a la plataforma para realizar actividades asignadas, responder preguntas generadas automáticamente y recibir retroalimentación inmediata sobre su desempeño. También puede revisar su historial de progreso, configurar sus horarios de estudio y recibir notificaciones sobre nuevas actividades o recordatorios. Este rol representa al beneficiario directo del sistema, ya que utiliza las herramientas de aprendizaje para desarrollar competencias de comprensión lectora, análisis crítico y detección de información relevante. Además, su interacción permite alimentar las métricas del sistema, generando datos que contribuyen al seguimiento académico y a la mejora continua del tutor virtual.

5.1.1. Diagramas de secuencia

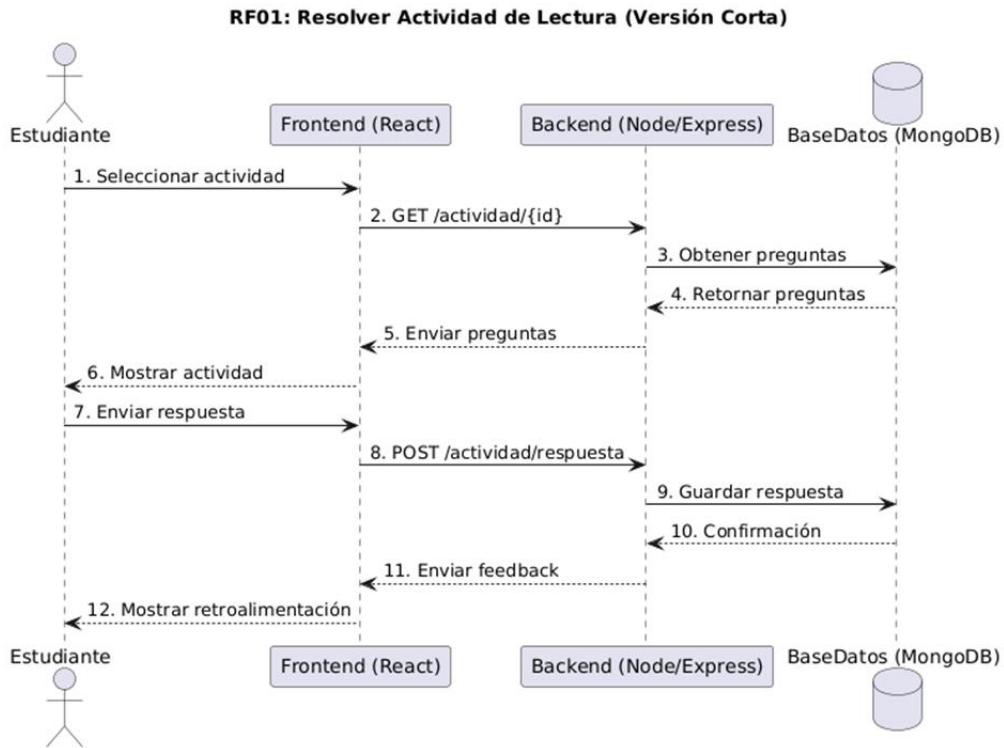


Ilustración 5 Diagramas de secuencia

Fuente. Elaboración propia.

RNF1 – Usabilidad

Este diagrama ilustra cómo el requisito no funcional de **usabilidad** impacta en la interacción general del usuario con la plataforma. El sistema debe presentar una interfaz sencilla, clara y consistente que permita al usuario navegar sin dificultad entre las distintas funciones. Todas las acciones principales deben ser accesibles desde navegadores web y dispositivos móviles, garantizando una experiencia fluida sin necesidad de capacitación previa.

Además, la estructura visual del frontend debe mantener coherencia en sus elementos, facilitando que el usuario realice tareas como iniciar sesión, responder actividades, visualizar reportes o configurar sus preferencias de estudio de manera intuitiva. Este requisito asegura que la plataforma Lectum sea accesible, práctica y fácil de utilizar para estudiantes y administradores en cualquier entorno.

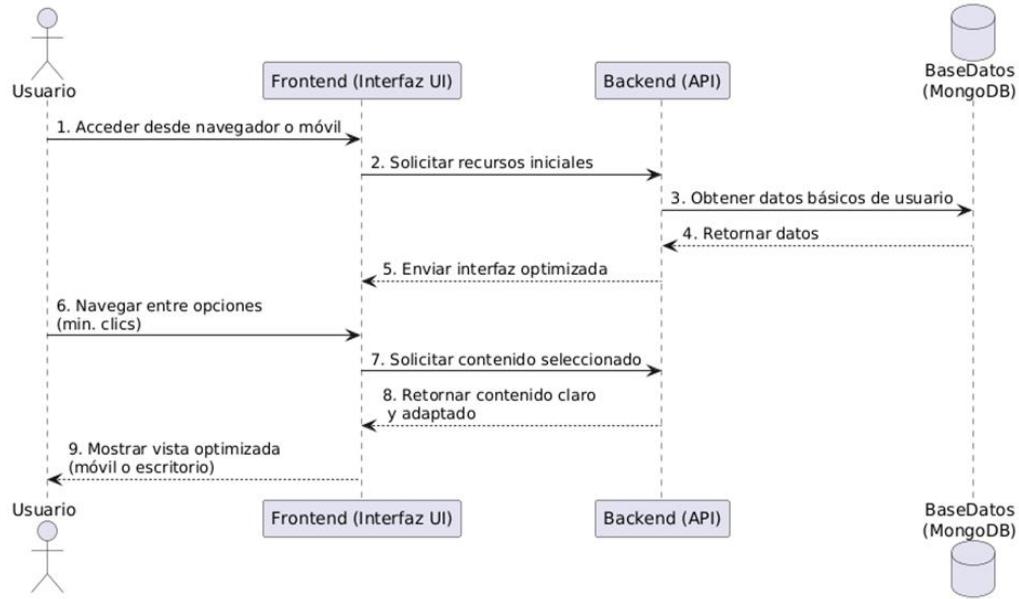


Ilustración 6 Diagramas de Usabilidad

Fuente. Elaboración propia.

RNF2. Rendimiento

Este diagrama de secuencia representa cómo el sistema gestiona el rendimiento durante el análisis de texto y la visualización de resultados. El usuario envía una solicitud de análisis desde el frontend, la cual es procesada por el backend y enviada al servicio de análisis de texto. Una vez completado el procesamiento, el backend recupera la información necesaria desde la base de datos y retorna al frontend los resultados listos para ser mostrados.

Todas estas operaciones deben ejecutarse en un tiempo total menor a 3 segundos en condiciones normales de uso, garantizando una respuesta rápida, una experiencia fluida y evitando tiempos de espera excesivos para el usuario.

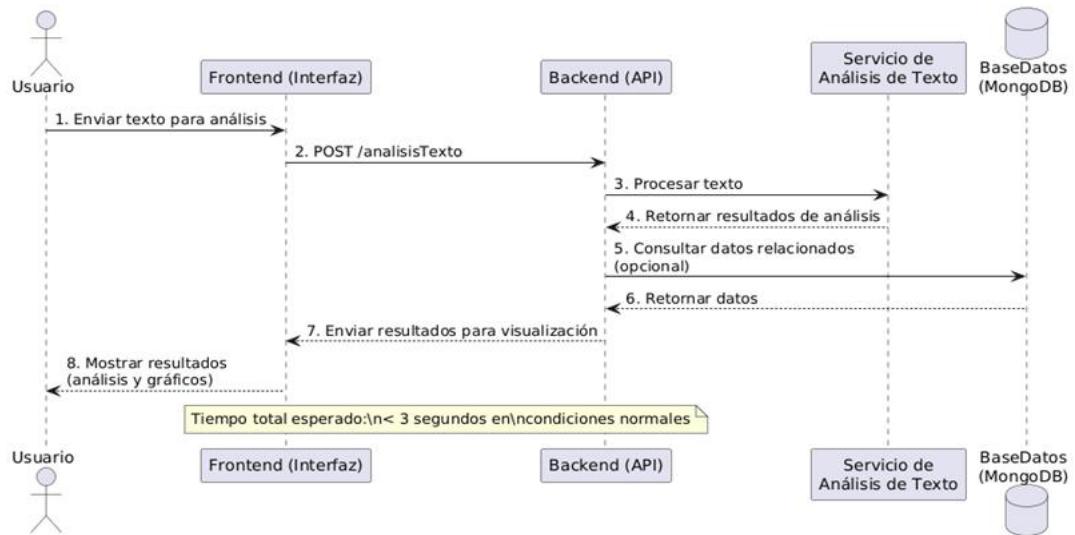


Ilustración 7 Rendimiento

Fuente. Elaboración propia.

RNF3. Seguridad

Este diagrama de secuencia muestra cómo el sistema garantiza la seguridad durante el proceso de autenticación y el manejo de sesiones. El usuario ingresa sus credenciales a través del frontend, el cual envía la información cifrada mediante HTTPS al backend. El servidor valida los datos contra la base de datos y, en caso de ser correctos, genera un token o sesión protegida que es devuelta al cliente de forma segura.

Durante toda la comunicación se emplea cifrado HTTPS y mecanismos de protección de sesión (como tokens seguros o cookies con atributos de seguridad), reduciendo el riesgo de accesos no autorizados, robo de credenciales o secuestro de sesión. Este requisito asegura que solo usuarios legítimos puedan acceder a la plataforma y que su información se mantenga protegida.

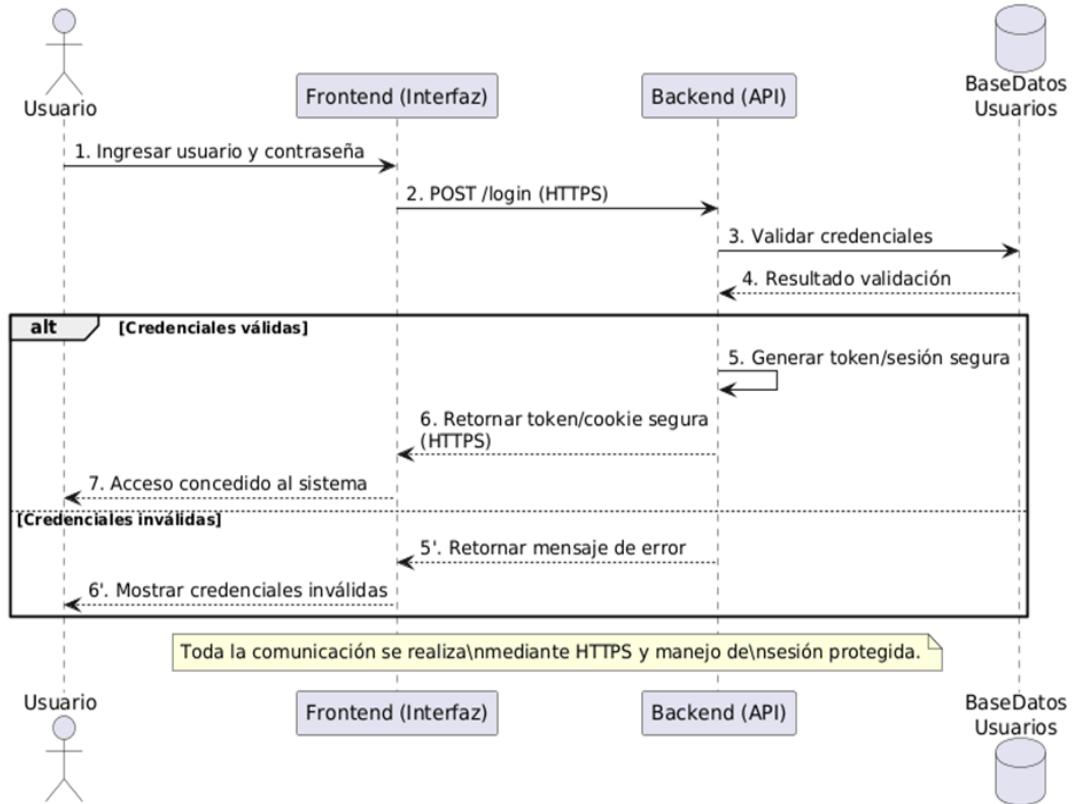


Ilustración 8 Diagramas de seguridad

Fuente. Elaboración propia.

RNF4. Escalabilidad

Este diagrama de secuencia representa cómo la arquitectura basada en MERN (MongoDB, Express/Node y React) facilita la incorporación de nuevas funcionalidades sin degradar el rendimiento del sistema. Cuando se agrega un nuevo módulo o característica, el frontend se comunica con nuevos endpoints del backend, los cuales se integran de forma modular con la lógica existente. El backend interactúa con la base de datos utilizando esquemas y colecciones diseñados para soportar crecimiento en la cantidad de datos y solicitudes.

Gracias a esta estructura escalable, es posible añadir nuevas pantallas, flujos o servicios sin necesidad de rehacer el sistema, manteniendo tiempos de respuesta adecuados y una experiencia estable para los usuarios, incluso cuando el número de funcionalidades o la cantidad de usuarios aumenta.

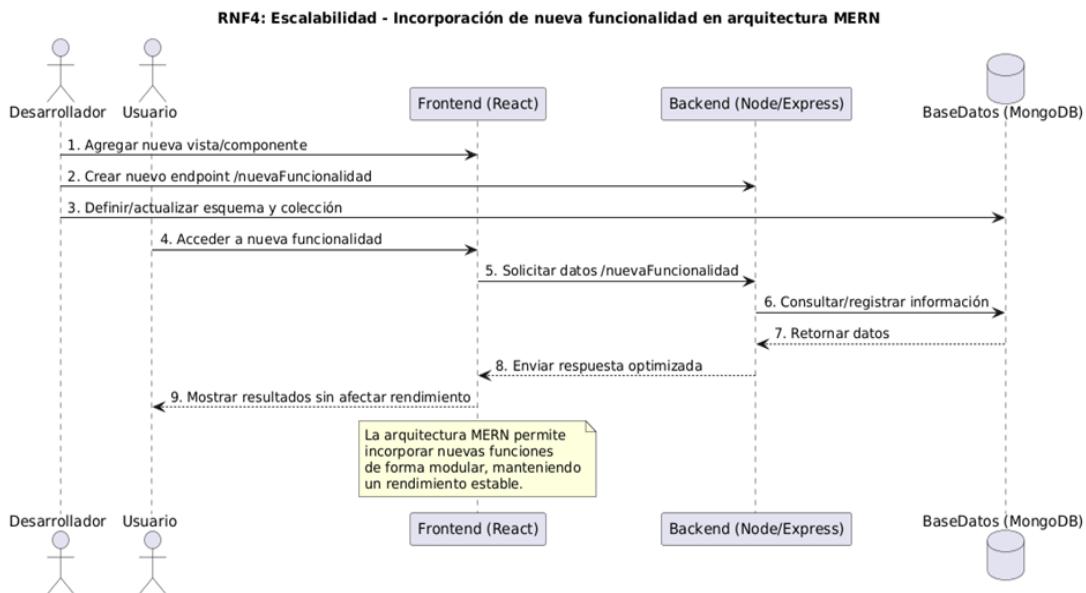


Ilustración 9 Diagramas de escalabilidad

Fuente. Elaboración propia.

RNF5. Disponibilidad

Este diagrama de secuencia representa cómo el sistema garantiza su disponibilidad continua. El usuario accede a la plataforma en cualquier momento, y el frontend envía la solicitud al backend, el cual consulta la base de datos y retorna la información requerida. Para asegurar una disponibilidad mínima del 95%, el backend se ejecuta en un entorno de servidor estable, con monitoreo constante y mecanismos de reinicio o recuperación ante fallos.

De esta manera, las principales funcionalidades del sistema pueden utilizarse las 24 horas del día, los 7 días de la semana, reduciendo al mínimo las interrupciones del servicio y permitiendo que los estudiantes y administradores accedan a la plataforma cuando lo necesiten.

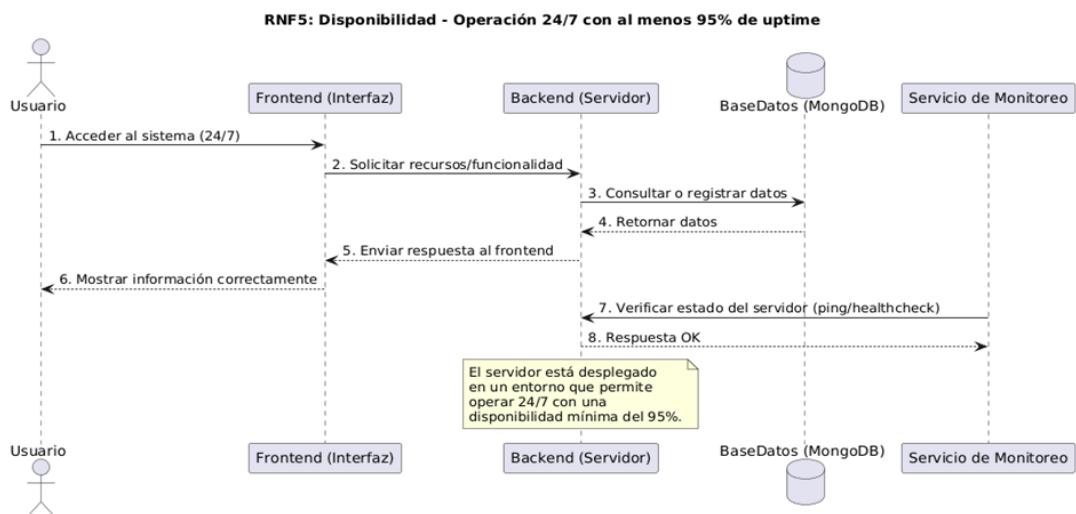


Ilustración 10 Diagramas de disponibilidad

Fuente. Elaboración propia.

RNF6. Mantenibilidad

Este diagrama de secuencia describe cómo se garantiza la mantenibilidad del sistema durante el ciclo de desarrollo. El desarrollador realiza cambios en módulos específicos del proyecto, respetando una estructura modular del código. Luego, envía sus cambios al repositorio, donde se ejecutan automáticamente las pruebas unitarias configuradas en la integración continua. Si todas las pruebas se completan correctamente, los cambios se integran a la rama principal del proyecto y la versión estable se mantiene actualizada.

Además, la documentación asociada a los módulos modificados se revisa y actualiza, de modo que otros desarrolladores puedan comprender y extender el sistema sin dificultad. Este proceso asegura que el código sea fácil de mantener, mejorar y depurar a lo largo del tiempo.

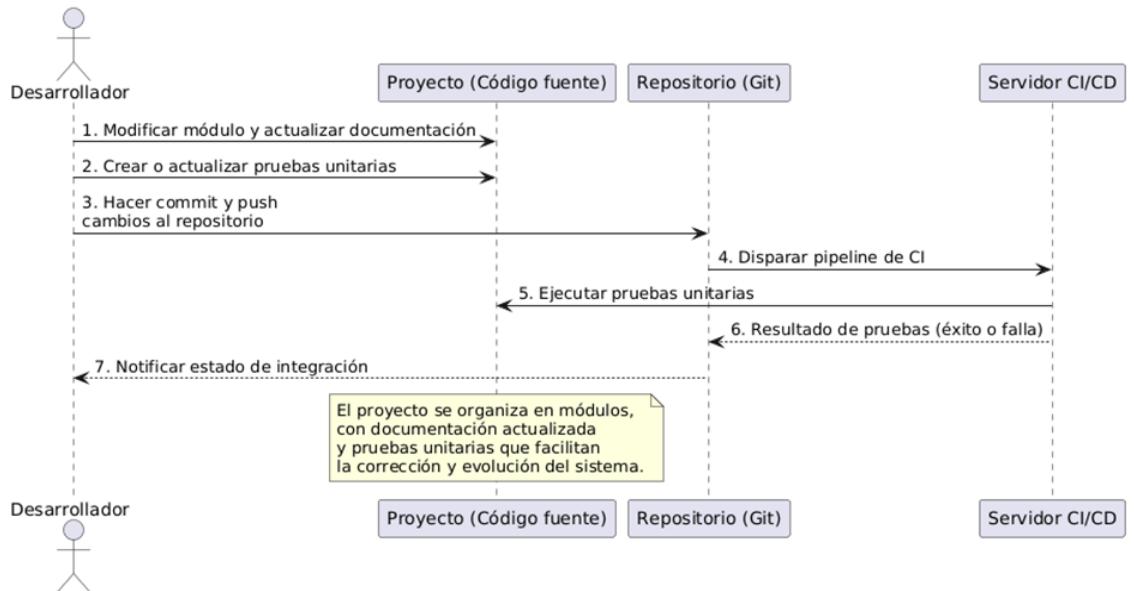


Ilustración 11 Diagramas de mantenibilidad

Fuente. Elaboración propia.

RNF7. Compatibilidad

Este diagrama de secuencia muestra cómo se garantiza la compatibilidad del sistema en los principales navegadores. El usuario accede a la plataforma desde Chrome, Edge o Firefox, y el frontend se adapta automáticamente a las características de cada navegador. El backend procesa las solicitudes normalmente y retorna respuestas estandarizadas, asegurando que la interfaz y las funcionalidades se rendericen de manera consistente en todos los navegadores compatibles.

Este requisito asegura que la experiencia del usuario sea uniforme sin importar el navegador utilizado, permitiendo el correcto funcionamiento del sistema en los entornos más comunes y evitando errores de visualización o incompatibilidad.

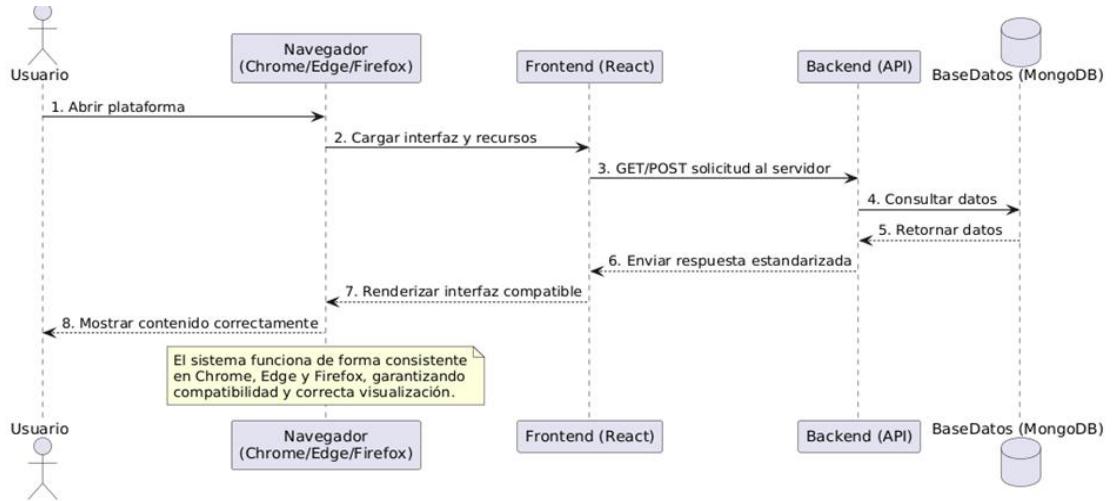


Ilustración 12 Diagramas de compatibilidad

Fuente. Elaboración propia.

5.1.2. Diagramas de colaboración

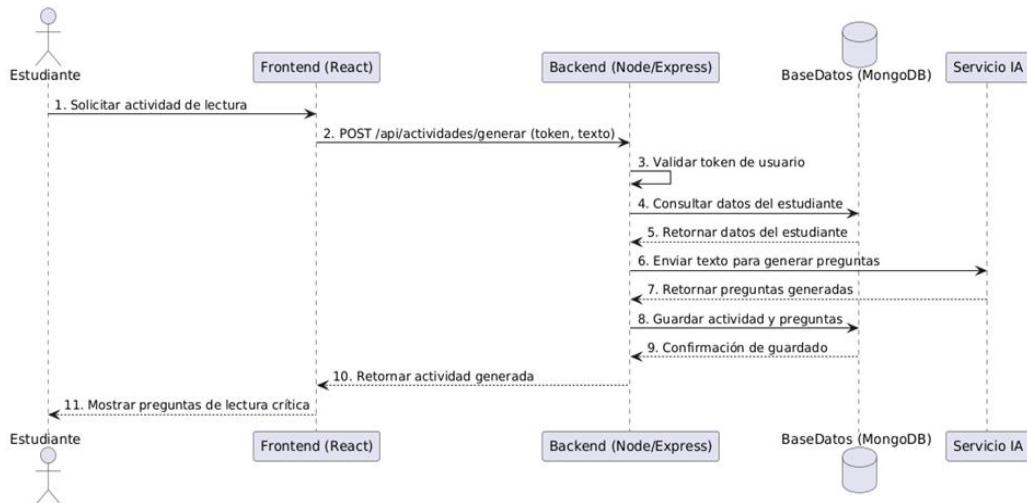


Ilustración 13 Diagrama de colaboración

Fuente. Elaboración propia.

Diagrama de Colaboración: Generar Actividad de Lectura Crítica

El diagrama de colaboración muestra cómo los diferentes componentes del sistema interactúan para generar automáticamente una actividad de lectura crítica a partir de un texto seleccionado por el Estudiante. El proceso inicia cuando el usuario

solicita la actividad desde el frontend, que envía la petición al backend junto con el token de autenticación y el texto a trabajar.

El backend valida las credenciales, consulta datos relevantes del estudiante en la base de datos y luego envía el texto al servicio de inteligencia artificial encargado de generar las preguntas. Una vez generadas, las preguntas se almacenan en MongoDB como parte de una nueva actividad de lectura. Finalmente, el backend devuelve al frontend la actividad generada para que el estudiante pueda visualizar las preguntas y comenzar su resolución.

5.1.3. Diagramas de clases

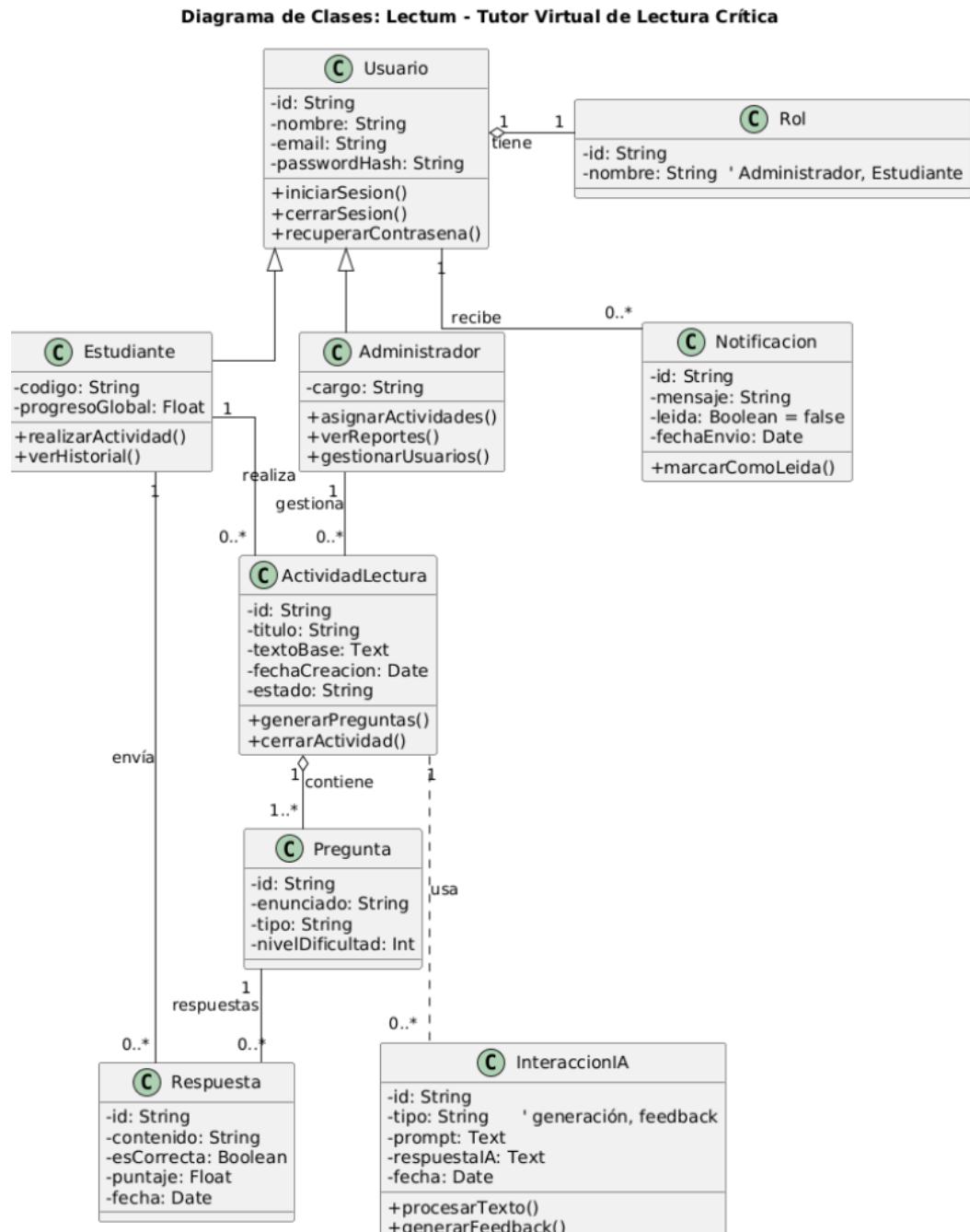


Ilustración 14 Diagrama de clase

Fuente. Elaboración propia.

Diagrama de Clases: Lectum – Tutor Virtual de Lectura Crítica

El diagrama de clases modela la estructura principal del sistema Lectum y cómo se relacionan sus entidades clave. En el centro se encuentra la clase **Usuario**, que concentra los atributos básicos del sistema (**id**, **nombre**, **email**, **passwordHash**) y

los métodos de autenticación (`iniciarSesion()`, `cerrarSesion()`, `recuperarContrasena()`). Cada usuario tiene asignado un Rol, representado por la clase `Rol` con el atributo nombre (Administrador, Estudiante), lo que permite diferenciar los permisos y responsabilidades dentro de la plataforma.

A partir de `Usuario` se especializan dos perfiles: Administrador y Estudiante, que heredan los datos comunes y añaden comportamientos propios. El Administrador puede `asignarActividades()`, `verReportes()` y `gestionarUsuarios()`, mientras que el Estudiante puede `realizarActividad()` y `verHistorial()` de su progreso. Las actividades de lectura se representan mediante la clase `ActividadLectura`, que almacena el texto base, la fecha de creación y el estado, y se relaciona con una o varias `Pregunta`, encargadas de evaluar la comprensión lectora. Las respuestas que envía el estudiante se guardan en la clase `Respuesta`, donde se registra el contenido, si fue correcta, el puntaje y la fecha.

La clase `InteraccionIA` modela las operaciones realizadas con el módulo de inteligencia artificial, incluyendo el tipo de interacción (generación de preguntas o feedback), el prompt enviado y la respuesta generada por la IA, asociadas a una actividad de lectura. Finalmente, la clase `Notificacion` almacena los mensajes enviados a cada usuario, indicando el contenido, la fecha de envío y si han sido leídos. En conjunto, estas clases describen la estructura lógica del sistema que soporta el funcionamiento del tutor virtual de lectura crítica `Lectum`.

5.2. Diseño de Base de Datos

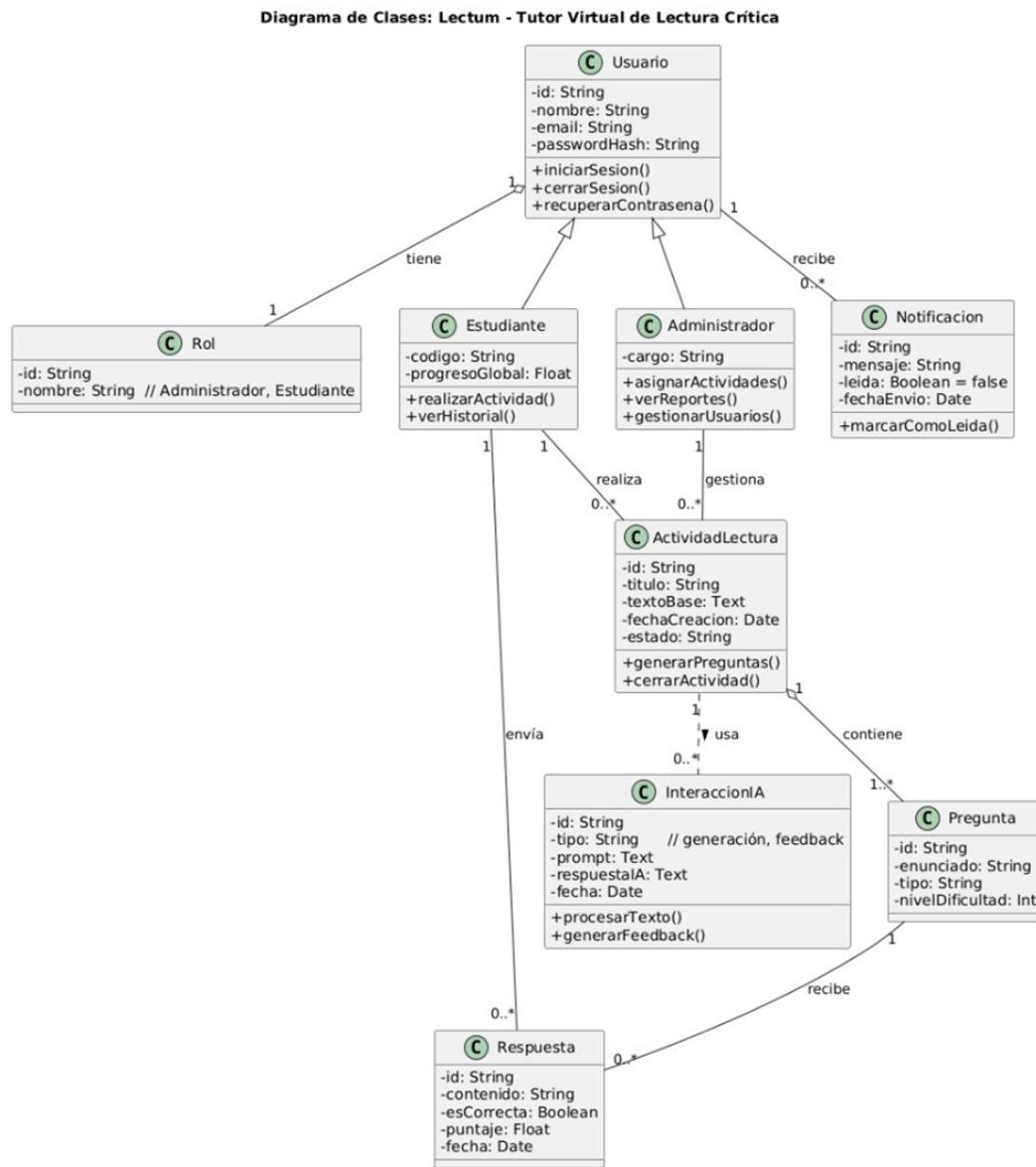


Ilustración 15 Diagrama de clase

Fuente. Elaboración propia.

Diagrama de Clases: Lectum – Tutor Virtual de Lectura Crítica

El diagrama de clases modela la estructura principal del sistema Lectum y cómo se relacionan sus entidades clave. En el centro se encuentra la clase Usuario, que agrupa los atributos básicos (id, nombre, email, passwordHash) y los métodos de autenticación del sistema (iniciarSesion(), cerrarSesion(), recuperarContrasena()). Cada usuario tiene asignado un Rol, representado por la clase Rol con el atributo nombre (Administrador, Estudiante).

A partir de Usuario se especializan dos perfiles: Administrador y Estudiante, que heredan los datos comunes y añaden comportamientos propios, como asignarActividades(), verReportes() o realizarActividad() y verHistorial(). Las actividades de lectura se representan mediante la clase ActividadLectura, que contiene el texto base y se relaciona con varias Pregunta y sus Respuesta asociadas. La clase InteraccionIA modela las llamadas al módulo de inteligencia artificial para generar preguntas y feedback, mientras que Notificacion registra los avisos enviados a cada usuario. En conjunto, estas clases describen la estructura lógica del sistema que soporta el tutor virtual de lectura crítica.

5.2.1. Diseño conceptual (E/R)

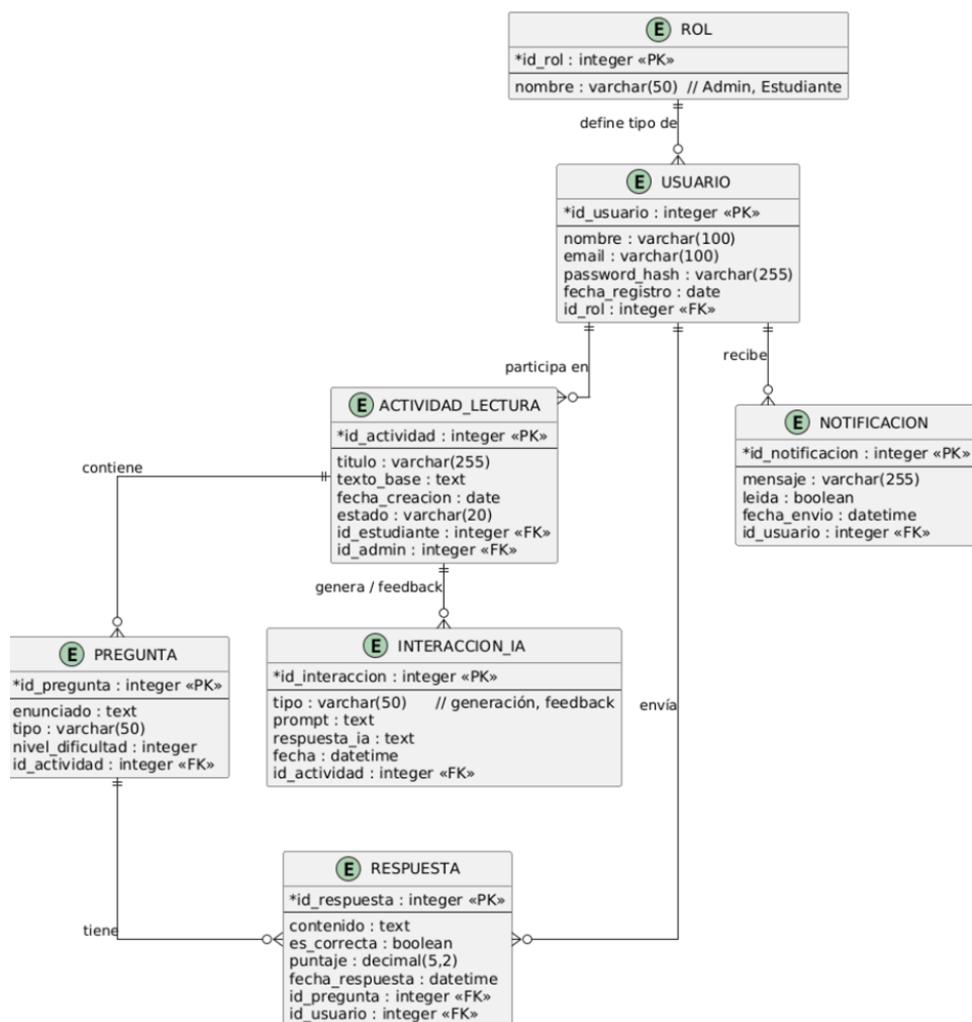


Ilustración 16 Diagrama de conceptual

Fuente. Elaboración propia.

El diseño conceptual en modelo Entidad–Relación estructura los componentes fundamentales del sistema Lectum y las relaciones que los conectan. En el centro se encuentra la entidad **USUARIO**, que almacena los datos esenciales del sistema (identificador, nombre, correo, contraseña cifrada y fecha de registro) y se vincula directamente con la entidad **ROL**, la cual define el tipo de usuario (Administrador o Estudiante).

Cada estudiante puede participar en múltiples **ACTIVIDAD_LECTURA**, donde se registra el texto base, la fecha de creación y el estado de la actividad. A su vez, cada actividad contiene una o varias **PREGUNTA**, que representan los ítems de evaluación de lectura crítica. Las respuestas proporcionadas por los usuarios se almacenan en la entidad **RESPUESTA**, que guarda el contenido, la corrección, el puntaje y la fecha de respuesta, manteniendo la trazabilidad de cada interacción.

La entidad **INTERACCION_IA** registra las operaciones realizadas con el módulo de inteligencia artificial, incluyendo el tipo de interacción, el prompt enviado y la respuesta generada, asociadas a una actividad específica. Finalmente, **NOTIFICACION** almacena los mensajes enviados a los usuarios (como recordatorios o avisos de nuevas actividades), indicando si han sido leídos o no. En conjunto, este modelo E/R define la base de datos conceptual que sustenta el funcionamiento del tutor virtual de lectura crítica Lectum.

5.2.2. Diseño lógico

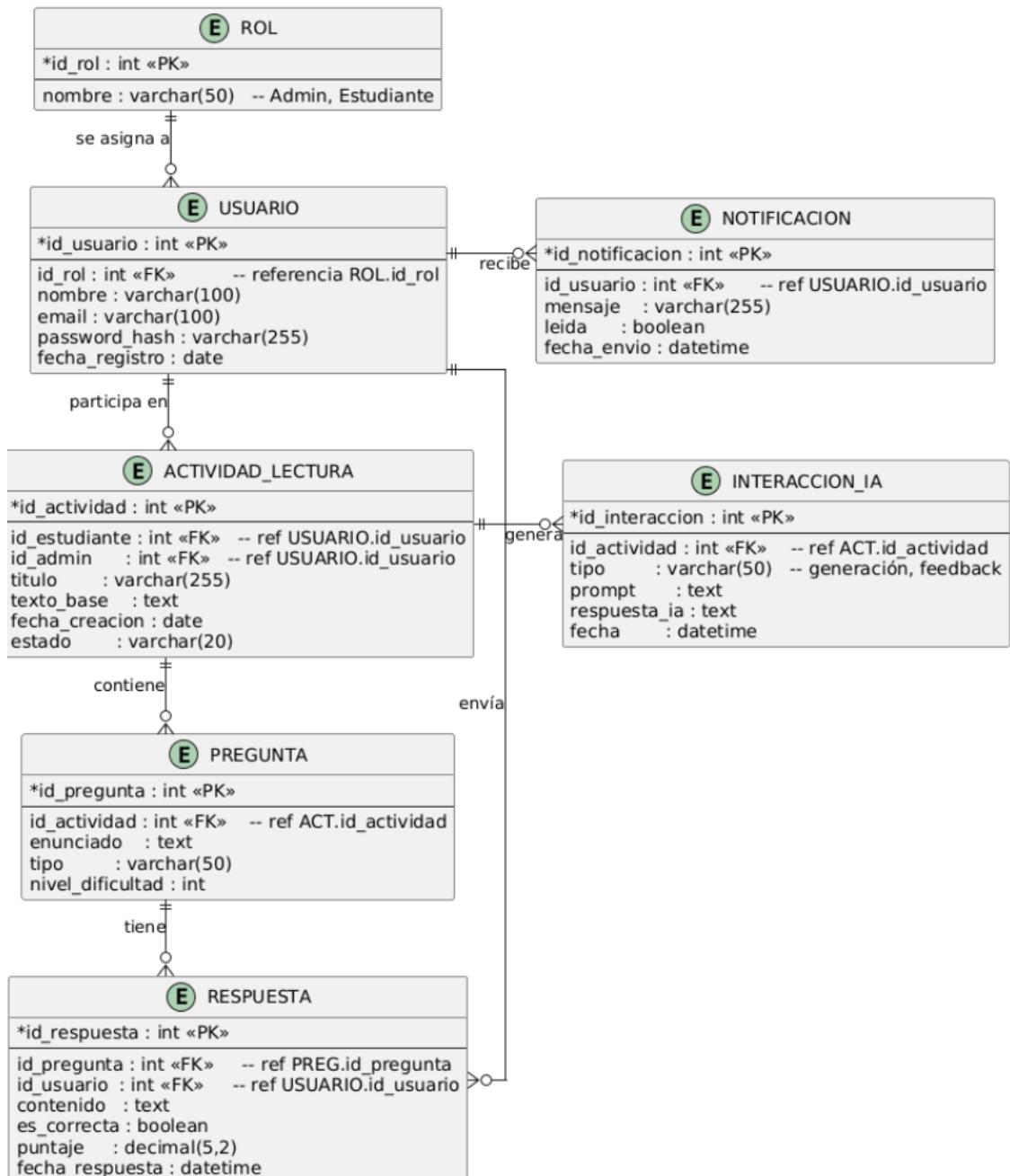


Ilustración 17 Diseño lógico

Fuente. Elaboración propia.

El diseño lógico normalizado del sistema Lectorum estructura las entidades principales y sus relaciones mediante claves primarias y foráneas, lo que garantiza coherencia, consistencia e integridad referencial en la base de datos. La entidad **USUARIO** se relaciona con **ROL** para definir el tipo de acceso dentro de la

plataforma (Administrador o Estudiante), y además puede participar en actividades, recibir notificaciones y generar interacciones con el módulo de inteligencia artificial.

Las actividades de lectura se administran mediante la entidad ACTIVIDAD_LECTURA, que almacena información esencial como el título, el texto base, la fecha de creación y su estado. Esta actividad se vincula con la entidad PREGUNTA, encargada de registrar los ítems de evaluación, mientras que las respuestas registradas por los estudiantes se almacenan en la entidad RESPUESTA, la cual mantiene la referencia tanto del usuario como de la pregunta correspondiente, preservando la trazabilidad del proceso de evaluación.

La entidad INTERACCION_IA almacena las consultas realizadas al módulo de inteligencia artificial, indicando el tipo de operación (generación o retroalimentación), el prompt enviado y la respuesta generada. Cada interacción queda vinculada a una actividad específica, permitiendo un historial detallado del uso de IA dentro del proceso de aprendizaje.

Finalmente, la entidad NOTIFICACION registra los avisos enviados a los usuarios, señalando el mensaje, la fecha de envío y si ha sido leído. Este diseño lógico permite estructurar de forma clara las relaciones entre usuarios, actividades académicas, evaluaciones, notificaciones e interacciones inteligentes, ofreciendo una base sólida para el funcionamiento del tutor virtual de lectura crítica.

5.2.3. Diseño físico

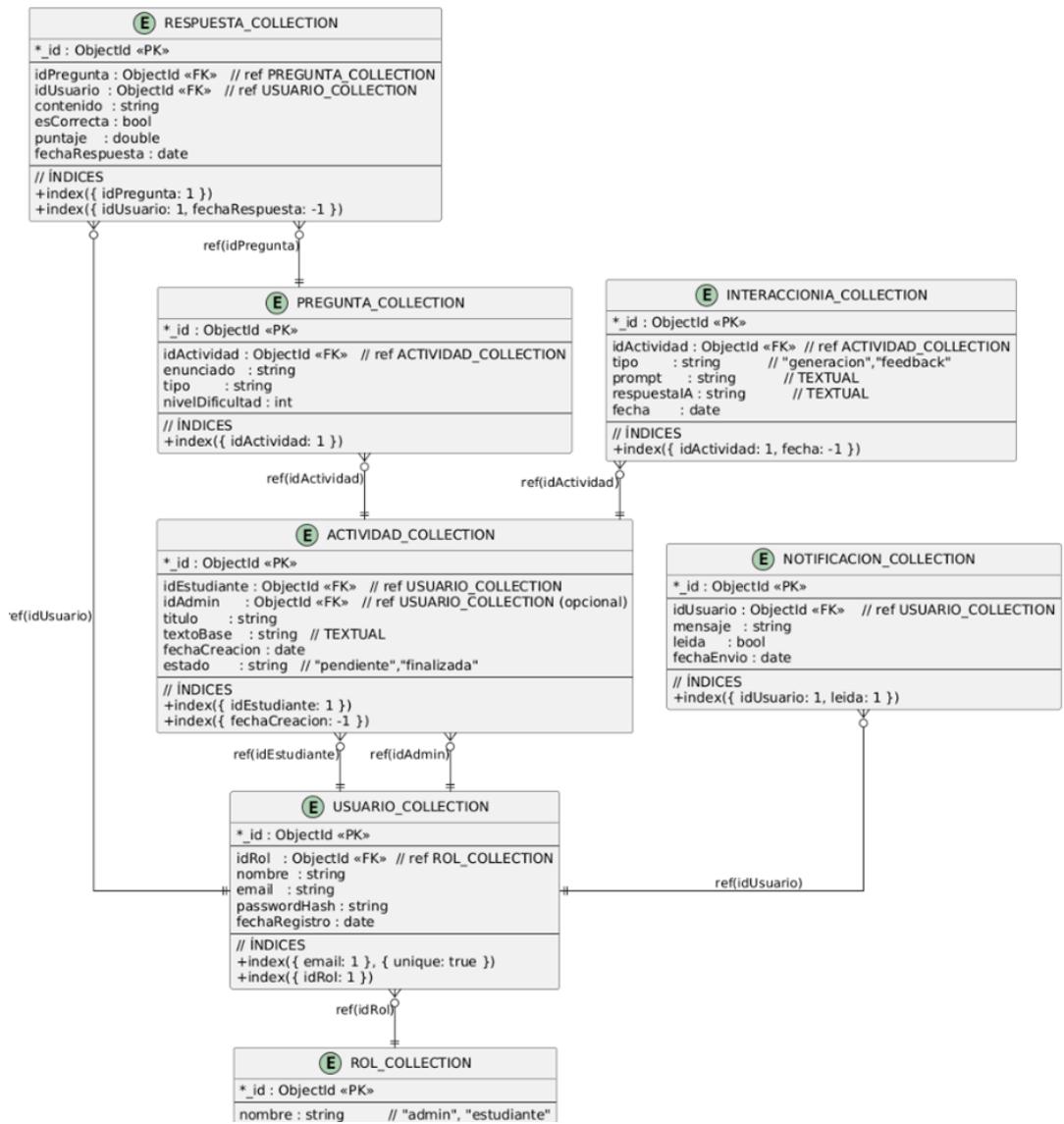


Ilustración 18 Diseño físico

Fuente. Elaboración propia.

| Users (USUARIO_COLLECTION) |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| { "_id": ObjectId, "idRol": ObjectId, // ref ROL_COLLECTION._id "nombre": String, "email": String, "passwordHash": String, "fechaRegistro": Date, "indexes": [{ "key": { "email": 1 }, "unique": true }, { "key": { "idRol": 1 } }] } |

Ilustración 19 Base de datos - Users

Fuente. Elaboración propia

Roles (ROL_COLLECTION)

```
{  
    "_id": ObjectId,  
    "nombre": String // "admin", "estudiante"  
}
```

Ilustración 20 Base de datos - Roles

Fuente. Elaboración propia

Activities (ACTIVIDAD_COLLECTION)

```
{  
    "_id": ObjectId,  
    "idEstudiante": ObjectId,      // ref USUARIO_COLLECTION._id  
    "idAdmin": ObjectId,          // ref USUARIO_COLLECTION._id (opcional)  
    "titulo": String,  
    "textoBase": String,  
    "fechaCreacion": Date,  
    "estado": String,            // "pendiente", "finalizada"  
    "indexes": [  
        { "key": { "idEstudiante": 1 } },  
        { "key": { "fechaCreacion": -1 } }  
    ]  
}
```

Ilustración 21 Base de datos - Activities

Fuente. Elaboración propia

Questions (PREGUNTA_COLLECTION)

```
{  
    "_id": ObjectId,  
    "idActividad": ObjectId,      // ref ACTIVIDAD_COLLECTION._id  
    "enunciado": String,  
    "tipo": String,  
    "nivelDificultad": NumberInt,  
    "indexes": [  
        { "key": { "idActividad": 1 } }  
    ]  
}
```

Ilustración 22 Base de datos - Questions

Fuente. Elaboración propia

Answers (RESPUESTA_COLLECTION)

```
{  
    "_id": ObjectId,  
    "idPregunta": ObjectId,      // ref PREGUNTA_COLLECTION._id  
    "idUser": ObjectId,          // ref USUARIO_COLLECTION._id  
    "contenido": String,  
    "esCorrecta": Boolean,  
    "puntaje": NumberDecimal,  
    "fechaRespuesta": Date,  
    "indexes": [  
        { "key": { "idPregunta": 1 } },  
        { "key": { "idUser": 1, "fechaRespuesta": -1 } }  
    ]  
}
```

Ilustración 23 Base de datos - Answers

Fuente. Elaboración propia

```

IA Interactions (INTERACCIONIA_COLLECTION)
{
    "_id": ObjectId,
    "idActividad": ObjectId,      // ref ACTIVIDAD_COLLECTION._id
    "tipo": String,             // "generacion", "feedback"
    "prompt": String,
    "respuestaIA": String,
    "fecha": Date,
    "indexes": [
        { "key": { "idActividad": 1, "fecha": -1 } }
    ]
}

```

Ilustración 24 Base de datos - IA interactions

Fuente. Elaboración propia

```

Notifications (NOTIFICACION_COLLECTION)
{
    "_id": ObjectId,
    "idUsuario": ObjectId,      // ref USUARIO_COLLECTION._id
    "mensaje": String,
    "leida": Boolean,
    "fechaEnvio": Date,
    "indexes": [
        { "key": { "idUsuario": 1, "leida": 1 } }
    ]
}

```

Ilustración 25 Base de datos - Notifications

Fuente. Elaboración propia

5.2.4. Modelado de base de datos

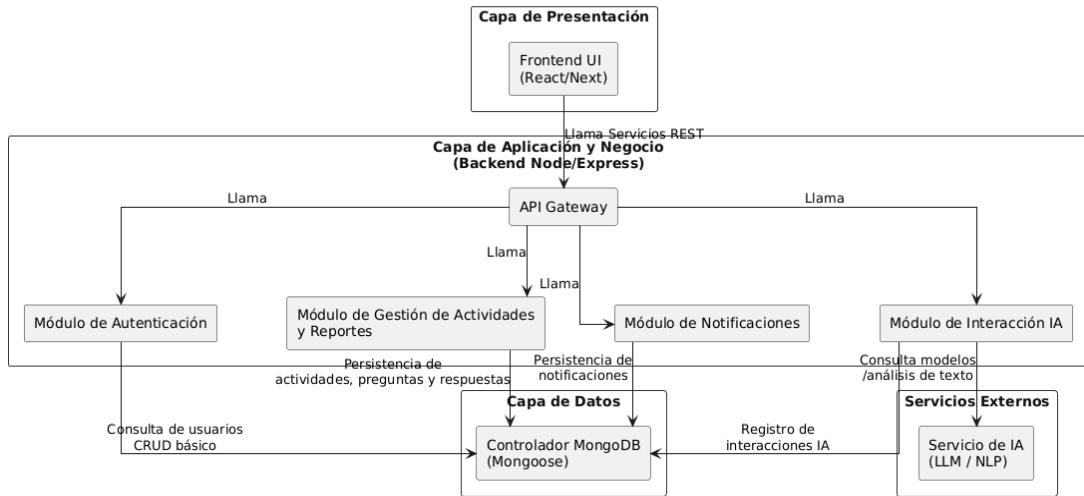


Ilustración 26 Diseño físico

Fuente. Elaboración propia

El modelado de base de datos del sistema Lectum se estructura bajo un diseño orientado a documentos en MongoDB, organizado a partir de colecciones que

representan las entidades fundamentales del tutor virtual: Usuarios, Roles, Actividades de Lectura, Preguntas, Respuestas, Interacciones con IA y Notificaciones. Cada colección almacena atributos específicos y se relaciona mediante referencias que permiten mantener coherencia lógica y una gestión eficiente de la información dentro del sistema.

La colección Usuario se vincula directamente con Rol, lo que define el tipo de acceso dentro de la plataforma (Administrador o Estudiante). A partir de ello, los usuarios pueden participar en múltiples actividades de lectura, registradas en la colección ActividadLectura, que contiene la información del texto base, fecha de creación y estado de avance. Estas actividades se relacionan con la colección Pregunta, encargada de almacenar los ítems de evaluación de comprensión lectora. Por su parte, las Respuestas registran la interacción del estudiante con cada pregunta, conservando el puntaje, la corrección y la fecha de envío.

Las InteraccionesIA almacenan las consultas realizadas al motor de inteligencia artificial, incluyendo el prompt enviado y la respuesta generada. Esta colección se asocia a actividades específicas para mantener historial y trazabilidad del acompañamiento automatizado. Finalmente, las Notificaciones registran los mensajes enviados a los usuarios, ya sea como recordatorios, avisos de nuevas actividades o retroalimentación sobre su progreso.

En conjunto, este modelo soporta los procesos de autenticación, gestión de actividades de lectura, generación de preguntas y retroalimentación por IA, almacenamiento de respuestas y emisión de notificaciones. La arquitectura propuesta optimiza la persistencia, consulta y organización de la información, garantizando un funcionamiento sólido y escalable del tutor virtual de lectura crítica Lectum.

5.3. Diseño de Interfaces Básicas

Se muestran los prototipos generales utilizados para representar la interacción usuario–sistema.

5.3.1. Acceso login

El prototipado muestra una interfaz web para iniciar sesión. En la parte superior, se encuentra el título "Tutor de Lectura Crítica" y enlaces para "Iniciar sesión" y "Registrarse". Una descripción invita a pegar texto para descubrir falacias y generar un cuestionario de pensamiento crítico. El formulario principal, titulado "Iniciar Sesión", pide credenciales y tiene campos para "Usuario/Cerro" y "Contraseña", así como un botón "Ingresar". Abajo del formulario, hay un enlace para registrarse si no tienes una cuenta.

Ilustración 27 Acceso login

Fuente. Elaboración propia

La interfaz presenta un formulario web sencillo para iniciar sesión, organizado en una sola columna y centrado en la pantalla. Incluye dos campos principales: correo electrónico y contraseña, acompañados de un botón para acceder al sistema. El diseño utiliza espacios amplios, tipografía clara y un estilo minimalista que facilita la lectura. En la parte superior se muestra el título “Tutor de Lectura Crítica” junto a una breve descripción del servicio, mientras que en la zona inferior del formulario se ofrece la opción de registrarse si el usuario aún no tiene una cuenta.

5.3.2. Interfaz (crear una cuenta)

θ≡ Tutor de Lectura Crítica Historial

Pega cualquier texto a continuación para descubrir falacias, detectar sesgos y generar un cuestionario de pensamiento crítico.

The screenshot shows a registration form titled "Crear Cuenta" (Create Account). Below the title, a sub-instruction reads "Regístrate para guardar tu historial." (Register to save your history). The form consists of three input fields: "Nombre" (Name), "Correo electrónico" (Email), and a "Registrar" (Register) button. At the bottom of the form, there is a link "¿Ya tienes cuenta? [Inicia sesión](#)" (Do you already have an account? [Log in](#)).

Ilustración 28 Interfaz (crear una cuenta)

Fuente. Elaboración propia

La interfaz presenta un diseño limpio y centrado, donde se muestra un formulario básico para la creación de una cuenta. Los campos están organizados en una sola columna e incluyen el nombre y el correo electrónico del usuario. Encima del formulario se destaca el título “Crear Cuenta” junto a un breve texto que explica la finalidad del registro, orientado a guardar el historial del usuario.

En la parte superior, el encabezado muestra el nombre del sistema “Tutor de Lectura Crítica” acompañado de un enlace a la sección “Historial”. El estilo general utiliza bordes delgados, tipografías simples y un espaciamiento amplio, lo que aporta claridad visual y facilita el proceso de registro.

5.3.3. Interfaz (Pantalla principal)

θ€ Tutor de Lectura Crítica

Historial

Pega cualquier texto a continuación para descubrir falacias, detectar sesgos y generar un cuestionario de pensamiento crítico.

Pega texto aquí para comenzar tu análisis crítico...

Detectar Falacias y Sesgos

Generar Cuestionario

Ilustración 29 Interfaz (Pantalla principal)

Fuente. Elaboración propia.

La interfaz presenta una pantalla principal limpia y minimalista. En la parte superior se muestra el título “Tutor de Lectura Crítica” junto al acceso al historial. Debajo, una breve instrucción explica que el usuario puede pegar un texto para iniciar el análisis. Al centro se ubica un cuadro de texto amplio para ingresar contenido, y debajo se encuentran dos botones: Detectar Falacias y Sesgos y Generar Cuestionario. El diseño utiliza tonos claros, bordes delgados y una estructura ordenada para facilitar la lectura y la interacción.

5.3.4. Interfaz (Historial)

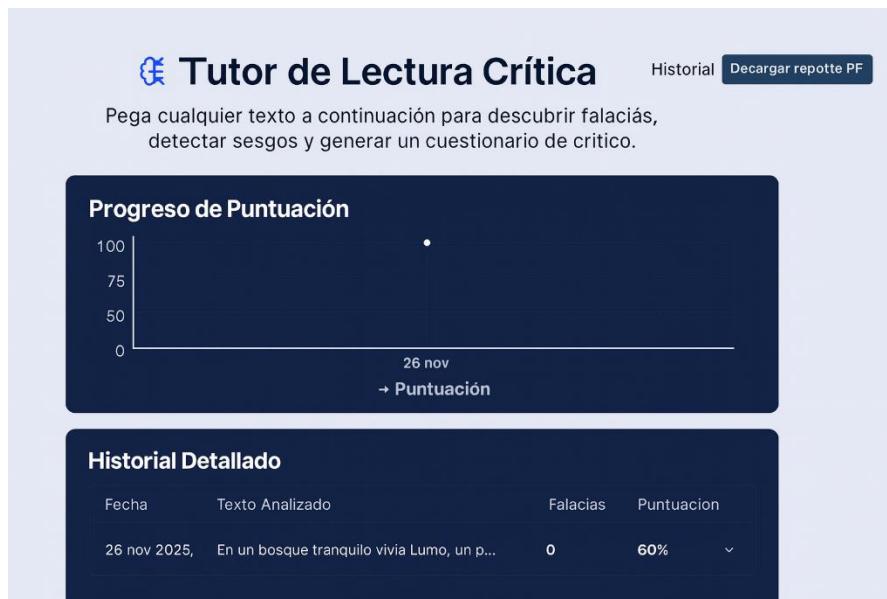


Ilustración 30 Interfaz (Historial)

Fuente. Elaboración propia.

La interfaz muestra un panel de usuario con un diseño oscuro y moderno, orientado a visualizar el progreso en lectura crítica. En la parte superior se encuentra el encabezado con el nombre del sistema “Tutor de Lectura Crítica”, acompañado de opciones de navegación como *Historial* y un botón para *Descargar reporte PDF*.

Debajo se presenta una sección destacada titulada “**Progreso de Puntuación**”, donde se incluye un gráfico lineal que muestra la evolución de la puntuación del usuario a lo largo del tiempo. El diseño utiliza líneas suaves, colores contrastados y una distribución amplia para facilitar la interpretación visual.

Más abajo aparece el módulo “**Historial Detallado**”, que organiza en forma de tabla la información de cada análisis realizado: fecha, texto evaluado, cantidad de falacias detectadas y la puntuación obtenida. La tabla es minimalista, con filas bien separadas y un estilo uniforme que permite revisar rápidamente el desempeño del usuario.

En conjunto, la interfaz combina simplicidad, legibilidad y una estructura ordenada para mostrar de manera clara el progreso y los resultados del análisis de pensamiento crítico.

CAPÍTULO 6

CODIFICACIÓN DEL SOFTWARE

6.1. Desarrollo del Sprint 1

El Sprint 1 se desarrolló entre el 3 y el 17 de septiembre, con un total de 4 historias de usuario, alineadas a las épicas EP01 y EP02, correspondientes al núcleo funcional del sistema Lectum. El objetivo del sprint fue implementar las bases del módulo de autenticación, análisis automático de textos, generación de preguntas y detección inicial de sesgos mediante procesamiento de lenguaje natural (NLP).

6.1.1. Sprint planning

Durante la reunión de Sprint Planning, el equipo definió las historias prioritarias basadas en la metodología MoSCoW, seleccionando únicamente los componentes Must Have, necesarios para construir el flujo principal del sistema.

Las historias se estimaron mediante Story Points en Jira, según su complejidad técnica y valor para el estudiante.

Tabla – Story Points Sprint 1

Tabla 9 Story Points Sprint 1

| ID | Historia de Usuario | Epic | Responsable | Story Points |
|-------|-----------------------------|------|------------------|--------------|
| HU1 | Login de estudiantes | EP01 | Equipo | 5 |
| HU2 | Preguntas automáticas | EP01 | Backend | 8 |
| HU2.1 | Retroalimentación inmediata | EP01 | Backend/Frontend | 5 |
| HU5 | Detectar sesgos o falacias | EP02 | NLP/Equipo | 8 |

Fuente. Elaboración propia.

6.1.2. Sprint backlog

El Sprint Backlog quedó conformado por las historias priorizadas y las tareas técnicas necesarias para cumplir los requerimientos funcionales RF1–RF5.

Tabla – Sprint Backlog Sprint 1

Tabla 10 Sprint Backlog Sprint 1

| Historia | Tareas técnicas asociadas |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| HU1 – Login estudiantes | Implementación de autenticación (RF1), validación de credenciales, manejo de errores, cierre de sesión seguro. |
| HU2 – Preguntas automáticas | Procesamiento del texto, integración del modelo NLP (RF2), generación de mínimo 5 preguntas (RF3). |
| Retroalimentación inmediata (HU2.1) | Implementación de la lógica de corrección, marcador “Correcto/Incorrecto”, explicación breve (RF4). |
| HU5 – Detección de sesgos | Creación de módulo NLP para sesgos (RF5), subrayado de fragmentos, clasificación de falacias. |

Fuente. Elaboración propia.

6.1.3. Historias de usuarios

HU1 – Login Estudiantes

Como estudiante, quiero iniciar sesión con mis credenciales para acceder a mi cuenta y continuar con mi progreso.

HU2 – Preguntas automáticas

Como estudiante, quiero responder preguntas automáticas para evaluar mi comprensión del texto.

HU2.1 – Retroalimentación inmediata

Como estudiante, quiero recibir retroalimentación inmediata para saber si mis respuestas son correctas o incorrectas.

HU5 – Detección de sesgos

Como usuario, quiero que el sistema detecte sesgos en los textos para mejorar mi razonamiento crítico.

6.1.4. Taskboard

El Taskboard del Sprint 1

| Tablero Sprint 1 3 sep – 2 oct (4 actividades) | | | | 0 | 0 | 33 | Completar sprint | ... |
|------------------------------------------------|----------------------------------------|-----------------------|------------|----|----|----|------------------|-----|
| PRDT2-19 | HU1.1 Login Estudiantes | EPI:TUTOR DE LECTU... | FINALIZADA | 5 | 0 | | | |
| PRDT2-1 | HU1:Cuestionario preguntas automáticas | EPI:TUTOR DE LECTU... | FINALIZADA | 10 | 10 | | JC | |
| PRDT2-3 | HU2:Retroalimentacion cuestionario | EPI:TUTOR DE LECTU... | FINALIZADA | 8 | 8 | | IZ | |
| PRDT2-6 | HU5:Detectar Sesgos o Falacias | EP2:DETECCIÓN DE S... | FINALIZADA | 10 | 10 | | MC | |
| + Crear | | | | | | | | |

Ilustración 31 Taskboard del Sprint 1

Fuente. Elaboración propia.

6.1.5. Daily scrum

Durante los daily se abordaron los siguientes puntos:

¿Qué se hizo?

- Implementación del login y validación de credenciales.
- Integración del modelo NLP para análisis preliminar.
- Pruebas iniciales de detección de sesgos.

¿Qué se hará?

- Finalizar generación de preguntas.
- Conectar retroalimentación con el motor de respuestas.
- Ajustar el algoritmo de sesgos para mejorar precisión.

Impedimentos

- Refinamiento del modelo NLP para evitar falsos positivos.

- Ajustes en el renderizado de retroalimentación.
- Necesidad de optimizar la carga de textos extensos.

6.1.6. Sprint review

- ✓ Login funcional con validación y mensajes de error.
- ✓ Análisis automático de texto mediante NLP.
- ✓ Generación de preguntas automáticas relacionadas al contenido.
- ✓ Retroalimentación inmediata.
- ✓ Subrayado de fragmentos con sesgos y clasificación inicial.

El Product Owner aprobó los avances y solicitó mejoras futuras en:

- Ajuste del algoritmo de sesgos
- Mayor variedad de preguntas
- Aumento en la precisión del NLP

6.1.7. Criterios de aceptación

Tabla 11 Criterios de aceptación

| Código | Historia | Criterio | Estado |
|--------|-----------|----------------------------------------|----------|
| HU1 | Login | Iniciar sesión, validar credenciales | Cumplido |
| HU1 | Login | Mostrar errores claros | Cumplido |
| HU2 | Preguntas | Generar ≥ 5 preguntas automáticas | Cumplido |
| HU2 | Feedback | Mostrar “Correcto/Incorrecto” | Cumplido |
| HU5 | Sesgos | Subrayar fragmentos sospechosos | Cumplido |
| HU5 | Sesgos | Clasificar tipo de sesgo | Cumplido |

Fuente. Elaboración propia.

6.1.8. Resultados del sprint

Objetivo logrado: se implementó el flujo principal del sistema Lectum.

Historias aprobadas: las 4 historias asignadas fueron completadas y validadas.

Incremento del producto:

- Login operativo
- Motor de análisis crítico funcional
- Detección de sesgos básica
- Retroalimentación inmediata

Avance hacia el MVP: se establecieron los pilares del sistema.

6.1.8.1. Evidencias.

Capturas del login.



Ilustración 32 Iniciar sesión

Fuente. Elaboración propia.

Crear Cuenta

Regístrate para guardar tu historial.

Nombre

Correo electrónico

Contraseña

Registrarse

¿Ya tienes cuenta? [Inicia sesión](#)

Ilustración 33 Crear cuenta

Fuente. Elaboración propia.

Ἑ Tutor de Lectura Crítica

Pega cualquier texto a continuación para descubrir falacias, detectar sesgos y generar un cuestionario de pensamiento crítico.

El Eco de la Aldea Gris
En la Aldea Gris, todos sabían que la gente del Valle Verde era holgazana y de poca confianza. No hacía falta haber hablado con muchos de ellos; con ver cómo se vestían con colores tan llamativos y escuchar una o dos historias era más que suficiente. El abuelo de Liam, un hombre sabio y respetado, siempre lo decía: "Nunca un Verdellano ha hecho un buen día de trabajo". Y si él lo decía, que había vivido tanto, era porque era verdad. (Sesgo de confirmación y Efecto de autoridad)

Un día, a Liam se le perdió la cartera de cuero, donde guardaba sus ahorros para comprar una nueva herramienta. Desesperado, registró su casa y su taller una y otra vez. De pronto, lo recordó: esa misma mañana,

Detectar Falacias y Sesgos **Generar Cuestionario**

Ilustración 34 Ingresar texto

Fuente. Elaboración propia.

Cuestionario de Pensamiento Crítico

Responde las siguientes preguntas para poner a prueba tu comprensión.

1. ¿Cuál fue la base principal para que la gente de la Aldea Gris formara una opinión negativa sobre los habitantes del Valle Verde?

- A) Experiencias directas y numerosas con sus vecinos.
- B) Datos estadísticos recopilados por el alcalde.
- C) Historias transmitidas oralmente y la opinión de figuras de autoridad.
- D) Informes de exploradores que habían visitado el valle.

2. Cuando Liam perdió su cartera y acusó inmediatamente al joven del Valle Verde, ¿qué sesgo cognitivo se manifestó más claramente en su pensamiento?

- A) Sesgo de autoridad, por la influencia de su abuelo.
- B) Sesgo de autoservicio, protegiendo su propia imagen.
- C) Generalización apresurada y sesgo endogrupal, basándose en el estereotipo de un grupo.
- D) Efecto de desinformación, al aceptar una idea errónea.

3. El Alcalde asintió gravemente a la acusación de Liam, pensando que 'cuando las cosas salían mal, un Verdellano solía estar involucrado'. ¿Qué tipo de sesgo ilustra mejor la conclusión del Alcalde?

- A) Sesgo de confirmación, buscando pruebas que apoyen sus creencias.
- B) Sesgo de correspondencia, atribuyendo los problemas al carácter de los Verdellanos.
- C) Sesgo de anclaje, basándose en la primera información recibida.
- D) Heurística de disponibilidad, recordando eventos recientes.

Ilustración 35 Capturas de preguntas generadas automáticamente.

Fuente. Elaboración propia.

Cuestionario de Pensamiento Crítico

Responde las siguientes preguntas para poner a prueba tu comprensión.

Tu Puntaje: 0%

1. ¿Cuál fue la base principal para que la gente de la Aldea Gris formara una opinión negativa sobre los habitantes del Valle Verde? ×

- A) Experiencias directas y numerosas con sus vecinos.
- B) Datos estadísticos recopilados por el alcalde.
- C) Historias transmitidas oralmente y la opinión de figuras de autoridad.
- D) Informes de exploradores que habían visitado el valle.

Respuesta Correcta: C) Historias transmitidas oralmente y la opinión de figuras de autoridad.

Explicación: La opinión se basaba en el 'eco de la aldea', es decir, rumores y la validación de una figura respetada como el abuelo de Liam, lo que demuestra sesgo de confirmación y efecto de autoridad.

2. Cuando Liam perdió su cartera y acusó inmediatamente al joven del Valle Verde, ¿qué sesgo cognitivo se manifestó más claramente en su pensamiento? ×

- A) Sesgo de autoridad, por la influencia de su abuelo.
- B) Sesgo de autoservicio, protegiendo su propia imagen.
- C) Generalización apresurada y sesgo endogrupal, basándose en el estereotipo de un grupo.
- D) Efecto de desinformación, al aceptar una idea errónea.

Ilustración 36 Pantalla de retroalimentación.

Fuente. Elaboración propia.

Falacias y Sesgos Detectados

Haz clic en un elemento para ver una explicación detallada.

▷ Generalización apresurada

"En la Aldea Gris, todos sabían que la gente del Valle Verde era holgazana y de poca confianza. No hacía falta haber...

La aldea llega a una conclusión general negativa sobre toda la gente del Valle Verde basándose en evidencia anecdótica y superficial (cómo se visten, una o dos historias), sin una base de datos amplia o representativa.

⚠ Apelación a la autoridad (Ad Verecundiam)

"El abuelo de Liam, un hombre sabio y respetado, siempre lo decía: "Nunca un Verdellano ha hecho un buen día de...

⚠ Sesgo endogrupal y Generalización apresurada

"De pronto, lo recordó: esa misma mañana, había visto a un joven del Valle Verde merodeando cerca de su cobertizo. "...

⚠ Sesgo de correspondencia (Error de atribución fundamental)

"Era del Valle Verde, señor Alcalde. ¿Qué otra prueba necesita?" El Alcalde, un hombre práctico, asintió con gravedad..

⚠ Sesgo de confirmación (grupal) y Memoria selectiva

"Nadie recordaba las veces que los comerciantes del valle habían sido justos, ni al joven ayudando a cargar un carro...

⚠ Sesgo de autoservicio y Sesgo de confirmación

"Rápidamente pensó: "Bueno, fue una suerte que lo encontrara. Al fin y al cabo, era cuestión de tiempo que uno de eso...

Ilustración 37 Vista de detección de sesgos.

Fuente. Elaboración propia.

6.1.8.2. Prueba de desarrollo.

Tabla 12 Prueba de desarrollo

| Historia | Tipo de prueba | Descripción | Resultado |
|----------|---------------------|--------------------------|-----------|
| HU1 | Pruebas funcionales | Validación de login | Aprobado |
| HU2 | Unit testing | Generación de preguntas | Aprobado |
| HU2.1 | Pruebas UI | Retroalimentación visual | Aprobado |
| HU5 | Pruebas NLP | Identificación de sesgos | Aprobado |

Fuente. Elaboración propia.

6.1.9. Sprint retrospective

Categoría: Lo que salió bien

- Integración correcta del motor NLP.
- Buen ritmo entre frontend y backend.

- Comunicación eficiente durante el sprint.

Lo que salió mal

- Estimación baja de complejidad en detección de sesgos.
- Retrasos en ajustes del algoritmo.

Mejoras propuestas

- Refinar modelos NLP para reducir falsos positivos.
- Implementar pruebas automatizadas desde el Sprint 2.
- Documentar procesos Backend–NLP.

6.2. Desarrollo del Sprint 2

El Sprint 2 se desarrolló entre el **29 de septiembre y el 14 de noviembre**, con un total de **3 historias de usuario** correspondientes a las épicas **EP01, EP02 y EP03**. El objetivo del sprint fue consolidar el almacenamiento de datos, habilitar el historial del estudiante y ampliar la funcionalidad educativa mediante ejemplos de sesgos detectados. Además, se integró el registro automático de respuestas en la base de datos para análisis posterior.

6.2.1. Sprint planning

Durante la reunión de Sprint Planning se seleccionaron historias con prioridad **Must Have** y **Should Have**, necesarias para fortalecer el registro y análisis del progreso del estudiante. Se estimaron los Story Points en Jira según complejidad técnica y dependencia con sistemas externos (BD y NLP).

Tabla 13 Sprint planning

| ID | Historia de Usuario | Epic | Responsable |
|------|----------------------------|------|-------------|
| HU3 | Guardar historial | EP01 | Backend |
| HU6 | Ver ejemplos de sesgo | EP02 | Equipo |
| HU13 | Registrar respuestas en BD | EP03 | Backend/BD |

Fuente. Elaboración propia.

6.2.2. Sprint backlog

El Sprint Backlog quedó conformado por las historias seleccionadas y las tareas técnicas vinculadas a los requerimientos funcionales RF6, RF7 y RF8.

Tabla 14 Sprint 2 Backlog

| Historia | Tareas técnicas |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| HU3 – Guardar historial | Diseño del modelo Historial en BD (RF6), estructuras de datos, endpoints para consulta, vista de historial en frontend, gráficos de progreso. |
| HU6 – Ver ejemplos de sesgos | Construcción del repositorio de ejemplos, clasificación de sesgos, vista educativa, filtros de búsqueda. |
| HU13 – Registro en BD | Registro automático de respuestas (RF8), timestamp, validación de datos, conexión segura a MongoDB, optimización de consultas. |

Fuente. Elaboración propia.

6.2.3. Historias de usuarios

HU3 – Guardar historial del estudiante

Como estudiante, quiero guardar mi historial de respuestas para revisar mi progreso en el tiempo.

HU6 – Ver ejemplos de sesgo

Como estudiante, quiero visualizar ejemplos explicados de sesgos para aprender a reconocerlos.

HU13 – Registrar respuestas

Como sistema, quiero registrar automáticamente las respuestas del estudiante para análisis y reportes.

6.2.4. Taskboard

El Taskboard incluyó las columnas

| Tablero Sprint 2 29 oct – 14 nov (3 actividades) | | 0 | 0 | 21 | Completar sprint | ... |
|--------------------------------------------------|-------------------------------------|-----------------------|------------|----|------------------|-----|
| PRDT2–4 | HU3:Guardas historial de respuestas | EP1:TUTOR DE LECTU... | FINALIZADA | 6 | KC | |
| PRDT2–7 | HU6:Visualizar Ejemplos de Sesgo | EP2:DETECCIÓN DE S... | FINALIZADA | 9 | MC | |
| PRDT2–14 | HU13:Registrar respuestas para BD | EP3:GESTIÓN Y AUT... | FINALIZADA | 6 | MC | |

Ilustración 38 Taskboard

Fuente. Elaboración propia

6.2.5. Daily scrum

Durante los daily se reportaron los siguientes puntos:

Qué se hizo

- Generación de la vista del historial con tabla y gráficos.
- Compilación del repositorio de ejemplos de sesgos.
- Pruebas de inserción y consulta en la base de datos.

Qué se hará

- Integrar historial con la vista del estudiante.
- Completar filtros dinámicos de ejemplos de sesgos.
- Optimizar consultas para gran volumen de datos.

Impedimentos

- Tiempo de respuesta lento en primeras consultas.
- Necesidad de paginación en historial para evitar sobrecarga.
- Ajustes visuales solicitados por el Product Owner.

6.2.6. Sprint review

Se mostraron las siguientes funcionalidades completas:

- ✓ Historial funcional con fecha, puntaje y texto analizado.
- ✓ Vista educativa de ejemplos de sesgos con descripción e ilustración.

- ✓ Registro automático de respuestas en BD con timestamp.
- ✓ Gráficos básicos del progreso del estudiante.

El Product Owner aprobó las funcionalidades y recomendó mejorar la organización visual del historial para versiones futuras.

6.2.7. Criterios de aceptación

Tabla 15 Criterios de aceptación

| Código | Historia | Criterio | Estado |
|--------|-----------|----------------------------------------|----------|
| HU3 | Historial | Guardar respuestas y puntajes | Cumplido |
| HU3 | Historial | Mostrar progreso con gráficos | Cumplido |
| HU6 | Sesgos | Mostrar ejemplos con descripción | Cumplido |
| HU6 | Sesgos | Acceso sin necesidad de nuevo análisis | Cumplido |
| HU13 | BD | Guardar respuestas automáticamente | Cumplido |
| HU13 | BD | Datos listos para reportes | Cumplido |

Fuente. Elaboración propia.

6.2.8. Resultados del sprint

Objetivo logrado: se consolidó el componente de almacenamiento y se fortaleció el valor educativo del sistema.

Historias aprobadas: las 3 historias asignadas fueron completadas y verificadas.

Incremento del producto:

- Historial completo del estudiante
- Repositorio educativo de sesgos
- Registro consolidado para reportes futuros

Avance hacia el MVP: el sistema ahora permite seguimiento y aprendizaje continuo.

6.2.8.1. Evidencias.

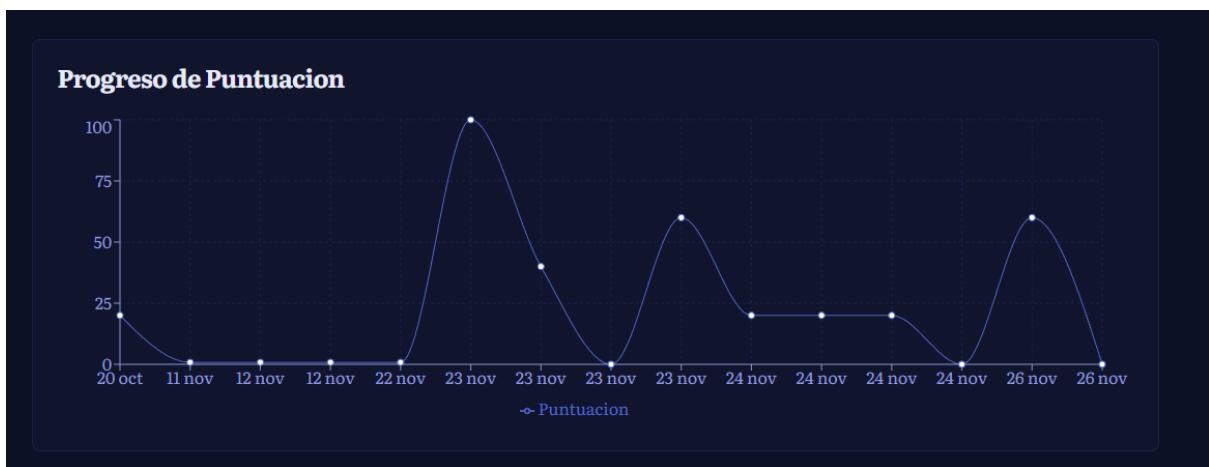


Ilustración 39 Progreso de puntuación

Fuente. Elaboración propia.

| Historial Detallado | | | | | |
|---------------------|-------------------------------------------------|----------|------------|---|--|
| Fecha | Texto Analizado | Falacias | Puntuación | | |
| 26 nov 2025, 10:58 | El Eco de la Aldea Gris En la Aldea Gris, to... | 0 | 0% | ▼ | |
| 26 nov 2025, 08:46 | En un bosque tranquilo vivía Lumo, un p... | 0 | 60% | ▼ | |
| 24 nov 2025, 13:33 | Cuento 1: La Elección de la Abuela La abu... | 0 | 0% | ▼ | |
| 24 nov 2025, 13:29 | El Eco de la Aldea Gris En la Aldea Gris, to... | 0 | 20% | ▼ | |
| 24 nov 2025, 13:23 | El Eco de la Aldea Gris En la Aldea Gris, to... | 0 | 20% | ▼ | |
| 24 nov 2025, 11:02 | El Eco de la Aldea Gris En la Aldea Gris, to... | 0 | 20% | ▼ | |
| 23 nov 2025, 16:44 | En el pequeño pueblo de Luminaria, todo... | 0 | 60% | ▼ | |
| 23 nov 2025, 13:00 | “En la ciudad de Valcora, donde la gente c... | 0 | 0% | ▼ | |

Ilustración 40 Captura de historial con fechas y puntajes.

Fuente. Elaboración propia.

Fuente. Elaboración propia.

Volver

Repositorio de Sesgos

Ejemplos de Falacias y Sesgos

Explora estos ejemplos para aprender a identificar falacias lógicas y sesgos cognitivos en diferentes contextos. Haz clic en cada uno para ver una explicación corta y su texto completo.

Ad Hominem
Ataque personal que desacredita al candidato en lugar de analizar su propuesta.
"Pasó un tiempo en la cárcel hace años, así que sus ideas no pueden ser buenas."

Explicación:
Esta es una falacia 'Ad Hominem' porque ataca el carácter del candidato Pérez en lugar de evaluar la validez de su propuesta económica. El pasado de una persona no invalida automáticamente sus argumentos presentes.

Texto original completo:
"No deberíamos escuchar la propuesta económica del candidato Pérez. Pasó un tiempo en la cárcel hace años, así que sus ideas no pueden ser buenas."

Falsa Dicotomía
Presenta solo dos opciones extremas y omite alternativas intermedias.
"O estamos completamente a favor de la nueva política de seguridad y renunciamos a nuestra privacidad, o la delincuencia se apoderará de la ciudad."

Hombre de Paja
Se distorsiona el argumento original para atacar una versión débil.
"Claramente, ella odia los deportes y no le importa la felicidad de los ciudadanos."

Generalización Apresurada
Conclusión amplia basada en una muestra pequeña o poco representativa.
"He probado dos restaurantes en esta ciudad y ambos eran malos. Concluyo que toda la oferta gastronómica de la ciudad es terrible."

Pendiente Resbaladiza
Afirma que un pequeño paso llevará inevitablemente a consecuencias extremas.
"Si permitimos que los estudiantes usen sus teléfonos en clase... finalmente todos reprobárán el año."

Ilustración 41 Vista de ejemplos de sesgos.

Fuente. Elaboración propia.

The screenshot shows the MongoDB Compass interface for the 'tutorcritico.history' database. At the top, it displays storage details: STORAGE SIZE: 84KB, LOGICAL DATA SIZE: 59.35KB, TOTAL DOCUMENTS: 15, and INDEXES TOTAL SIZE: 72KB. Below this are navigation tabs: Find, Indexes, Schema Anti-Patterns (0), Aggregation, and Search Indexes. A search bar below the tabs contains the placeholder "Generate queries from natural language in Compass". To the right of the search bar is an "INSERT DOCUMENT" button. A filter bar at the bottom has a "Type a query: { field: 'value' }" input field, a "Reset" button, an "Apply" button, and an "Options" button. The main area shows "QUERY RESULTS: 1-15 OF 15". Two documents are listed:

```

_id: ObjectId('68f6a8582ff915cc8fb00a0c')
text: "Había una vez un pueblo donde todos creían que los gatos traían mala suerte."
▶ fallacies: Array (empty)
▶ questions: Array (5)
score: 20
timestamp: 1760995416490
userId: "68f6a4272ff915cc8fb00a0b"

```



```

_id: ObjectId('6913f622d69c01f965db352f')
text: "Ejemplo de texto analizado y guardado en historial."
▶ fallacies: Array (empty)
▶ questions: Array (empty)
score: 0.8

```

Ilustración 42 Registro en BD (colección responses).

Fuente. Elaboración propia.

6.2.8.2. Prueba de desarrollo.

Tabla 16 Prueba de desarrollo

| Historia | Tipo de prueba | Descripción | Resultado |
|----------|------------------------|---------------------------|-----------|
| HU3 | Pruebas funcionales | Validación del historial | Aprobado |
| HU3 | Pruebas de rendimiento | Carga de datos grandes | Aprobado |
| HU6 | Pruebas UI | Visualización de ejemplos | Aprobado |
| HU13 | Pruebas BD | Inserción automática | Aprobado |

Fuente. Elaboración propia.

6.2.9. Sprint retrospective

Lo que salió bien

- Integración eficiente con MongoDB.
- Construcción rápida del repositorio educativo.
- Buena organización del equipo.

Lo que salió mal

- Falta de paginación en el historial al inicio.
- Algunas validaciones tardaron más de lo previsto.

Mejoras propuestas

- Implementar paginación y carga diferida.
- Optimizar consultas y filtros.
- Documentar la estructura de BD.

6.3. Desarrollo del Sprint 3

El Sprint 3 se desarrolló entre el **1 y el 30 de noviembre**, con un total de **3 historias de usuario** correspondientes a la épica **EP03**. El objetivo fue implementar automatizaciones, recordatorios, personalización del estudiante y exportación de datos en PDF y Excel.

6.3.1. Sprint planning

Durante la reunión de Sprint Planning se eligieron historias de prioridad **Should Have** y **Could Have**, pero esenciales para el cierre del MVP. Las historias fueron estimadas en Jira considerando la integración con n8n y bibliotecas externas para PDF/Excel.

Tabla 17 Sprint planning

| ID | Historia | Epic | Responsable | Story Points |
|------|---------------------|------|-------------|--------------|
| HU12 | Configurar horarios | EP03 | Frontend | 5 |

| | | | | |
|------|----------------------------|------|--------------------|---|
| HU10 | Notificaciones automáticas | EP03 | Automatización/n8n | 8 |
| HU14 | Exportar PDF/Excel | EP03 | Backend | 8 |

Fuente. Elaboración propia.

6.3.2. Sprint backlog

Tabla 18 Sprint backlog

| Historia | Tareas técnicas |
|-----------------------------------|------------------------------------------------------------------------------------------------|
| HU12 – Configurar horarios | Creación de vista y formulario, persistencia en BD, validación, confirmación visual. |
| HU10 – Notificaciones automáticas | Integración con n8n (RF10), creación de workflow, pruebas de envío, configuración de triggers. |
| HU14 – Exportar datos (Excel/PDF) | Implementación de librerías (xlsx/pdfkit), generación de reportes, descarga automática (RF11). |

Fuente. Elaboración propia.

6.3.3. Historias de usuarios

HU12 – Configurar horarios

El estudiante selecciona horas preferidas para recibir textos y avisos.

HU10 – Notificaciones automáticas

El sistema envía recordatorios diarios o semanales mediante n8n.

HU14 – Exportar datos

El estudiante puede descargar su historial en PDF o Excel.

6.3.4. Taskboard

Taskboard (Jira)

| Tablero Sprint 3 1 nov – 29 nov (3 actividades) | | | | | | |
|-------------------------------------------------|----------------------------------------------------------------------|-----------------------|------------|----|----|----------------|
| | | | 0 | 0 | 23 | Iniciar sprint |
| | PRDT2-15 HU14:Exportar datos como Excel o PDF | EP1:TUTOR DE LECTU... | FINALIZADA | 10 | MC | |
| | PRDT2-13 HU12:Configurar horarios como estudiante | EP3:GESTIÓN Y AUT... | FINALIZADA | 5 | MC | |
| | PRDT2-11 HU10:Recibir notificaciones para constancia lectura critica | EP3:GESTIÓN Y AUT... | FINALIZADA | 8 | JZ | |

Ilustración 43 Taskboard (Jira)

Fuente. Elaboración propia.

6.3.5. Daily scrum

Qué se hizo

- Implementación de formularios de horarios.
- Integración de flujos de notificaciones.
- Pruebas de exportación de archivos.

Qué se hará

- Validar el flujo final de notificaciones.
- Mejorar la interfaz de selección de formato.
- Pruebas finales de compatibilidad.

Impedimentos

- Incompatibilidad inicial con PDFKit.
- Ajustes en el webhook de n8n.
- Problemas de estilos en Excel.

6.3.6. Sprint review

Se mostraron:

- ✓ Horarios funcionales guardados en BD.
- ✓ Notificaciones enviadas correctamente mediante n8n.

- ✓ Exportación en Excel y PDF sin errores.
- ✓ Interfaz intuitiva para descargar reportes.

El Product Owner aprobó todas las funcionalidades.

6.3.7. Criterios de aceptación

Tabla 19 Criterios de aceptación

| Historia | Criterio | Estado |
|----------|------------------------------------|----------|
| HU12 | Horarios personalizados | Cumplido |
| HU12 | Datos persistentes en BD | Cumplido |
| HU10 | Envío automático de notificaciones | Cumplido |
| HU14 | Exportación en PDF/Excel | Cumplido |
| HU14 | Descarga automática | Cumplido |

Fuente. Elaboración propia.

6.3.8. Resultados del sprint

Objetivo logrado: Se completó el módulo de personalización y recordatorios.

Historias aprobadas: 3/3 historias terminadas.

Incremento del producto:

- Sistema con recordatorios automáticos
- Reportes descargables
- Configuración personalizada del estudiante

Cierre del MVP: sistema listo para despliegue final.

6.3.8.1. Evidencias.

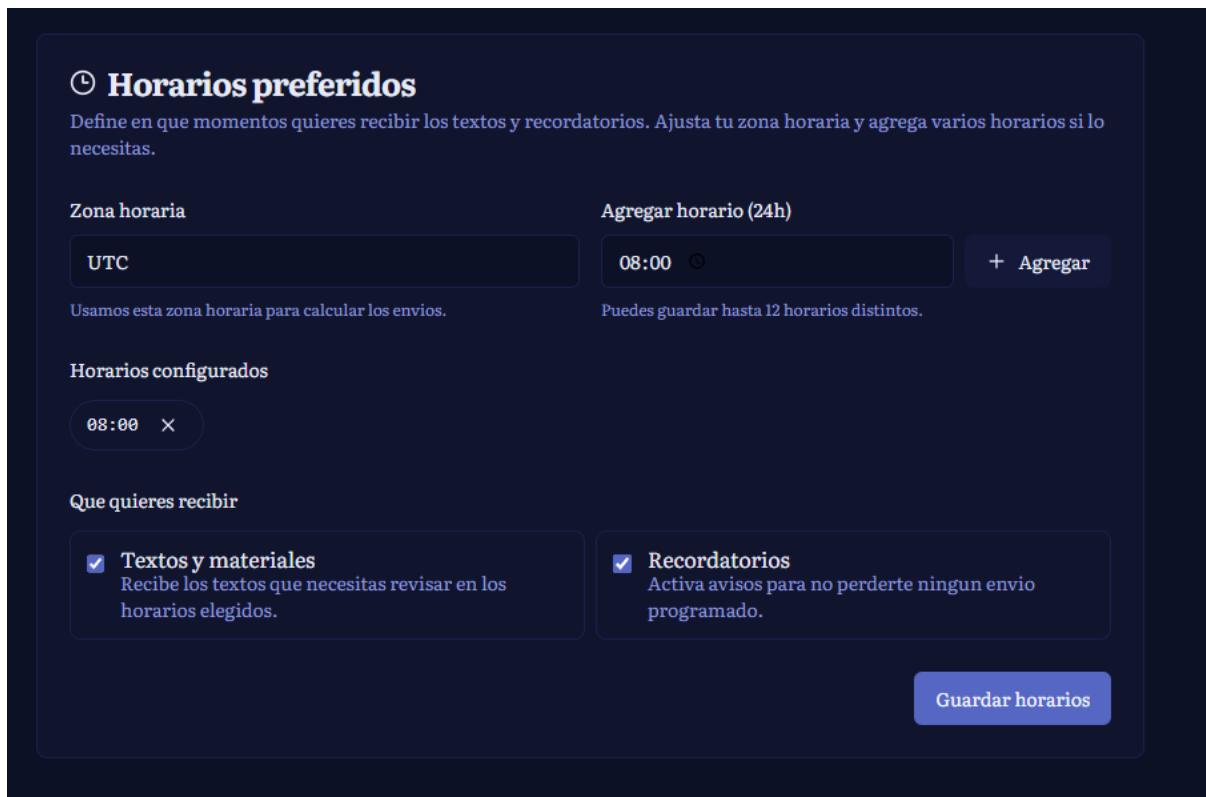


Ilustración 44 Capturas de horarios configurados.

Fuente. Elaboración propia.

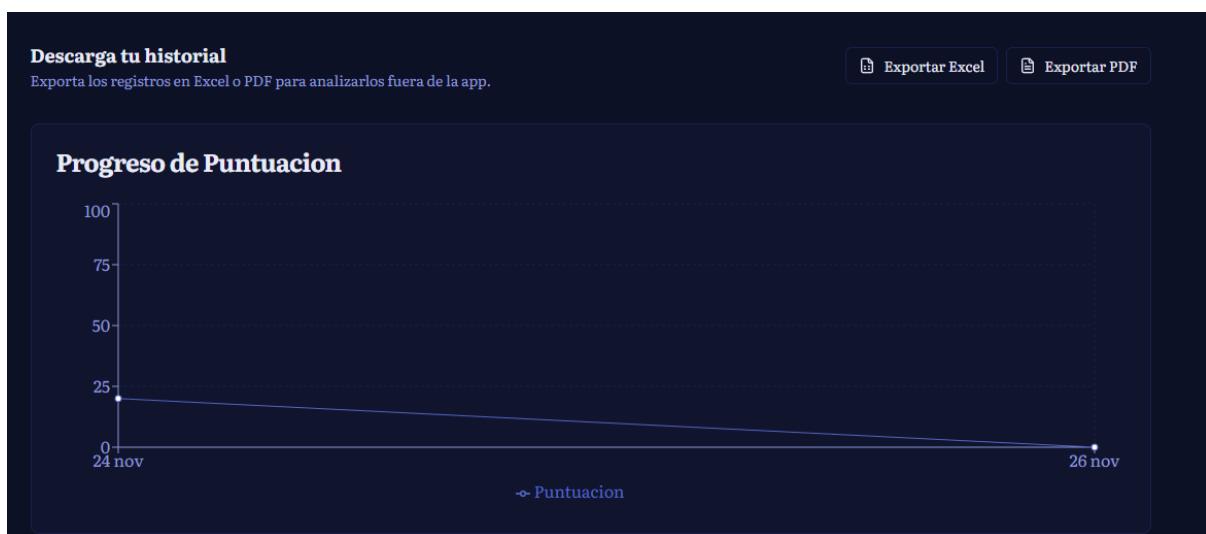


Ilustración 45 Progreso de puntuación

Fuente. Elaboración propia.

Historial de análisis
Generado: 26/11/2025, 12:01

Entrada 1
Fecha: 26/11/2025, 10:58
Puntuación: 0% | Falacias: 0

Texto: El Eco de la Aldea GrisEn la Aldea Gris, todos sabían que la gente del Valle Verde era holgazana y de poca confianza. No hacía falta haber hablado con muchos de ellos; con ver como se vestían con colores tan llamativos y escuchar una o dos historias era más que suficiente. El abuelo de Liam, un hombre sabio y respetado, siempre lo decía: "Nunca un Verdellano ha hecho un buen día de trabajo". Y si él lo decía, que había vivido tanto, era porque era verdad. (Sesgo de confirmación y Efecto de autoridad)Un día, a Liam se le perdió la cartera de cuero, donde guardaba sus ahorros para comprar una nueva herramienta. Desesperado, registró su casa y su taller una y otra vez. De pronto, lo recordó: esa misma mañana, había visto a un joven del Valle Verde merodeando cerca de su cobertizo. "Es obvio", pensó Liam, su enfado creciendo como la espuma. "Si gente es así, no podía esperarse otra cosa." (Sesgo de grupo o Endogrupal y Generalización apresurada)Corrió a casa del Alcalde, y con la voz entrecortada por la indignación, acusó al muchacho. "Era del Valle Verde, señor Alcalde. Que otra prueba necesita?" El Alcalde, un hombre práctico, asintió con gravedad. En su experiencia, cuando las cosas salían mal, un Verdellano solía estar involucrado. (Sesgo de correspondencia - atribuir un acto al carácter y no a la situación).La noticia se extendió por la aldea como un reguero de polvora. En la taberna, los vecinos comentaban: "A mí nunca me gusto la forma en que nos miran", o "El otro día uno regateó el precio de mi lana, tacano como todos". Nadie recordaba las veces que los comerciantes del valle habían sido justos, ni al joven ayudando a cargar un carro pesado la semana pasada. Esos detalles no encajaban, así que fueron olvidados. (Efecto de desinformación y Sesgo de confirmación en grupo)Mientras la turba de Liam se dirigía al valle, furiosa y convencida de su causa, su esposa encontró la cartera detrás de un arco, donde había rodado después de que el gato juguetone la empujara. Liam se sintió un tonto, pero solo por un instante. Rápidamente pensó: "Bueno, fue una suerte que lo encontrara. Al fin y al cabo, era cuestión de tiempo que uno de esos chicos robara algo. Mi reacción, aunque equivocada esta vez, estaba justificada". (Sesgo de auto-servicio - atribuir el error a una circunstancia, pero justificar la creencia original)Y así, la Aldea Gris siguió sumida en su certeza. Porque, al fin y al cabo, para qué buscar pruebas complicadas cuando un prejuicio sencillo explicaba tan bien el mundo?

Falacias (0):
Sin registros de falacias
Cuestionario (5):

1. ¿Cuál fue la base principal para que la gente de la Aldea Gris formara una opinión negativa sobre los habitantes del Valle Verde?
Respuesta correcta: C) Historias transmitidas oralmente y la opinión de figuras de autoridad.
2. Cuando Liam perdió su cartera y acusó inmediatamente al joven del Valle Verde, qué sesgo cognitivo se manifestó más claramente en su pensamiento?
Respuesta correcta: C) Generalización apresurada y sesgo endogrupal, basándose en el estereotipo de un grupo.
3. El Alcalde asintió gravemente a la acusación de Liam, pensando que 'cuando las cosas salían mal, un Verdellano solía estar involucrado'. ¿Qué tipo de sesgo ilustra mejor la conclusión del Alcalde?
Respuesta correcta: B) Sesgo de correspondencia, atribuyendo los problemas al carácter de los Verdellanos.
4. En la taberna, los vecinos olvidaron las acciones positivas de los comerciantes del valle. ¿Qué sesgos cognitivos explican este comportamiento colectivo?
Respuesta correcta: B) Sesgo de confirmación grupal y efecto de desinformación.
5. Después de encontrar la cartera, Liam pensó: 'Mi reacción, aunque equivocada esta vez, estaba justificada'. Esta justificación de su error mientras mantiene su prejuicio original es un ejemplo de:
Respuesta correcta: B) Sesgo de autoservicio.

Entrada 2
Fecha: 24/11/2025, 11:02

Ilustración 46 Archivos PDF/Excel generados.

Fuente. Elaboración propia.

6.3.8.2. Prueba de desarrollo.

Tabla 20 Prueba de desarrollo

| Historia | Tipo de prueba | Descripción | Resultado |
|----------|---------------------|-------------------------|-----------|
| HU12 | Pruebas UI | Validación de horarios | Aprobado |
| HU10 | Pruebas de flujo | Envío de notificaciones | Aprobado |
| HU14 | Pruebas funcionales | Generación de archivos | Aprobado |

Fuente. Elaboración propia

6.3.9. Sprint retrospective

Lo que salió bien

- Integración con n8n de manera estable.
- Generación de reportes sin errores.
- Buena sincronización del equipo.

Lo que salió mal

- Problemas iniciales con librerías de PDF.
- Retrasos en pruebas de automatización.

Mejoras

- Establecer plantillas visuales para reportes.
- Documentar workflows n8n.
- Implementar pruebas automatizadas para exportaciones.

CAPÍTULO 7

PRUEBAS DE SOFTWARE

7.1. Plan de Pruebas

El plan de pruebas del proyecto TutorVirtual, desarrollado sobre una arquitectura basada en el stack MERN (MongoDB, Express, React/Next.js y Node.js), establece la estrategia global para validar la calidad del software, asegurando el cumplimiento de los requisitos funcionales y no funcionales definidos en el backlog. El plan integra pruebas unitarias, de integración, de interfaz, pruebas end-to-end, uso de servidores mock, validación de seguridad y pruebas enfocadas en eficiencia energética (Software Verde), siguiendo buenas prácticas de ingeniería y un enfoque de desarrollo iterativo inspirado en metodologías ágiles.

7.1.1. Objetivo del Plan de Pruebas

Los objetivos del plan de pruebas del proyecto TutorVirtual son los siguientes:

- Verificar que cada módulo del sistema funcione de manera correcta y aislada.
- Validar la integración completa entre el frontend desarrollado en React/Next.js, el backend en Node/Express y la base de datos MongoDB.
- Confirmar el cumplimiento de los requisitos funcionales definidos para el sistema, especialmente los relacionados con autenticación, análisis de texto y registro de historial.
- Comprobar los requisitos no funcionales asociados a rendimiento, disponibilidad, seguridad y usabilidad.
- Asegurar una cobertura adecuada de código tanto en el frontend como en el backend, mediante pruebas automatizadas.

- Validar la comunicación con servicios externos, incluyendo los flujos de IA utilizados para detectar sesgos, falacias y generar preguntas de pensamiento crítico.
- Garantizar que el consumo energético y el impacto ambiental del sistema sean mínimos mediante mediciones con herramientas como Lighthouse y GreenFrame.
- Asegurar la correcta gestión de sesiones y la protección de rutas mediante el middleware de autenticación.
- Validar que el historial de análisis se registre correctamente en la base de datos y se visualice sin errores en el módulo de estadísticas.
- Confirmar que el flujo completo del usuario —desde el ingreso del texto hasta la obtención del análisis final— se ejecute de manera consistente y sin interrupciones.
- Verificar que las validaciones de entrada, como el mínimo de 50 caracteres, y los mensajes de error del sistema se gestionen adecuadamente.

7.1.2. Alcance de las Pruebas

- El plan de pruebas del proyecto TutorVirtual abarca los siguientes componentes:
- Back-End (Node/Express): controladores de autenticación, historial y análisis, servicios internos, middlewares de protección de rutas y API REST para el procesamiento de textos.
- Front-End (React/Next.js): componentes de análisis, historial, gráficas, hooks personalizados, estados internos, navegación y validación de formularios.
- Base de datos (MongoDB): modelos, esquemas, consultas, registro de análisis y recuperación del historial.

- Integración del stack MERN: comunicación frontend–backend, asincronía en llamadas REST, flujos CRUD y persistencia de datos.
- Módulo de IA: ejecución de los flujos Genkit para detección de sesgos, falacias y generación de preguntas, incluyendo validación de la estructura de las respuestas del LLM.
- Historial y Métricas: pruebas sobre el almacenamiento de análisis, consultas y representación gráfica de resultados.
- Seguridad y Autenticación: validación del token, middleware, manejo de cookies, redirecciones y restricción de rutas.
- Mock API (JSON Server): simulación de respuestas para pruebas rápidas sin depender de la IA.
- Validación de entradas: verificación del tamaño mínimo del texto (50 caracteres), manejo de errores y mensajes de retroalimentación.
- Pruebas de Software Verde: evaluación del consumo energético, transferencia de datos y eficiencia del renderizado mediante Lighthouse y optimizaciones del frontend.

7.1.3. Tipos de pruebas

Tabla 21 Tipos de pruebas

| Tipo de prueba | Descripción |
|-------------------------------------------|---------------------------------------------------------------------------------------------------|
| Pruebas Unitarias (Jest) | Validan controladores de back-end, rutas API, funciones de validación y lógica de negocio. |
| Pruebas de Cobertura | Miden el porcentaje de código del frontend y backend ejecutado durante las pruebas automatizadas. |
| Pruebas de Integración (Jest + Supertest) | Verifican flujos completos: login → análisis de texto con IA → registro en el historial. |
| Pruebas de Front-End (React/Next.js) | Validan componentes React, hooks (como use-session), estados internos y comportamiento de la UI. |
| Pruebas de Mock API (JSON Server) | Confirman respuestas y consistencia de un servidor simulado para pruebas rápidas sin IA real. |

| | |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------|
| Pruebas Funcionales | Evalúan el cumplimiento de los requisitos funcionales (RF), como autenticación, análisis e historial. |
| Pruebas de Rendimiento (RNF02) | Miden tiempos de respuesta de la API (por ejemplo /api/analyze) y la carga de vistas críticas. |
| Pruebas de Disponibilidad (RNF04) | Simulan múltiples solicitudes para comprobar estabilidad del sistema y manejo de carga básica. |
| Pruebas de Seguridad | Validan roles, autenticación, manejo de tokens/cookies y protección de rutas restringidas. |
| Pruebas de Eficiencia Energética | Se evalúa el rendimiento del frontend y la transferencia de datos usando Lighthouse y optimizaciones de React. |

Fuente. Elaboración propia

7.2. Ejecución de pruebas

7.2.1. Pruebas unitarias Backend

Basadas en Jest y Supertest.

En el proyecto TutorVirtual, las pruebas unitarias se aplicaron principalmente sobre los módulos del backend relacionados con la autenticación, el análisis mediante IA y el historial de análisis.

Módulos evaluados:

- Auth Controller: validación de credenciales, creación de usuarios, manejo de tokens y cookies, verificación de sesión y control de errores.
- AI/Analysis Controller: validación del texto de entrada, invocación de los flujos Genkit (sesgos, falacias, preguntas) y ensamblaje correcto de la respuesta.
- History Controller: almacenamiento y consulta de análisis en MongoDB, manejo de documentos y estructura del historial.
- Middleware de Autenticación: validación de tokens, protección de rutas y manejo de accesos no autorizados.

- Manejo de Errores: respuestas HTTP adecuadas (400, 401, 409) y validaciones de entrada (mínimo de 50 caracteres).

Resultado general de pruebas

Todas las funciones cubiertas respondieron correctamente bajo diferentes escenarios, confirmando que el backend cumple con los criterios funcionales y de seguridad establecidos.

Cobertura del Back-End

La cobertura de pruebas permitió verificar una correcta ejecución de las funciones más críticas del backend, especialmente en las rutas de autenticación y análisis de IA.

| File | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #s |
|---------------|--------|---------|--------|--------|-------------------|
| All files | 80.99 | 39.28 | 77.27 | 84.34 | |
| admin/history | 75 | 100 | 100 | 75 | |
| route.js | 75 | 100 | 100 | 75 | 7 |
| admin/users | 75 | 100 | 100 | 75 | |
| route.js | 75 | 100 | 100 | 75 | 7 |
| analyze | 95 | 84.61 | 100 | 95 | |
| route.js | 95 | 84.61 | 100 | 95 | 42 |
| auth/login | 100 | 100 | 100 | 100 | |
| route.js | 100 | 100 | 100 | 100 | |
| auth/logout | 100 | 100 | 100 | 100 | |
| route.js | 100 | 100 | 100 | 100 | |
| auth/me | 100 | 100 | 100 | 100 | |
| route.js | 100 | 100 | 100 | 100 | |
| auth/signup | 100 | 100 | 100 | 100 | |
| route.js | 100 | 100 | 100 | 100 | |
| history | 100 | 100 | 100 | 100 | |
| route.js | 100 | 100 | 100 | 100 | |
| report | 100 | 100 | 100 | 100 | |
| route.js | 100 | 100 | 100 | 100 | |
| report/pdf | 72.22 | 25.58 | 58.33 | 77.27 | |
| route.js | 72.22 | 25.58 | 58.33 | 77.27 | 31-33,42-54,114 |

```

Test Suites: 13 passed, 13 total
Tests:      30 passed, 30 total
Snapshots:  0 total
Time:       9.605 s, estimated 12 s
Ran all test suites matching /src\app\api\i

```

Ilustración 47 Pruebas unitarias Backend

Fuente. Elaboración propia

7.2.2. Pruebas unitarias Frontend

Las pruebas unitarias del front-end del proyecto TutorVirtual se realizaron utilizando React Testing Library y se enfocaron en validar el comportamiento de los componentes, formularios, hooks y lógica interna de la interfaz construida con Next.js (React).

Las pruebas abarcaron:

- Componentes UI: QuestionsDisplay, AnalysisResults, FallaciesDisplay, BiasesDisplay, HistoryTable, HistoryChart.
- Formularios: validación del texto mínimo (50 caracteres), manejo de errores y envío del análisis.
- Hooks personalizados: use-session y use-mobile, verificando estados de autenticación y comportamiento responsivo.
- Renderizado condicional: estados de carga, mensajes de error, historial vacío y resultados generados por la IA.
- Navegación y estados globales: acceso a rutas protegidas, manejo del token y recuperación de sesión.

Cobertura (Front-End)

| src/app/actions.test.js | | | | | |
|-------------------------|-----------------------------------------|----------|----------|---------|---------|
| PASS | File | % Starts | % Branch | % Funcs | % Lines |
| | All files | 78.08 | 59.37 | 77.77 | 78.97 |
| | ai/flows | 100 | 100 | 100 | 100 |
| | classify-detected-biases.js | 100 | 100 | 100 | 100 |
| | detect-logical-fallacies.js | 100 | 100 | 100 | 100 |
| | generate-critical-thinking-questions.js | 100 | 100 | 100 | 100 |
| | app | 95.45 | 84.61 | 100 | 95.23 |
| | actions.js | 95.45 | 84.61 | 100 | 95.23 |
| | app/login | 93.1 | 44.44 | 100 | 96.42 |
| | page.jsx | 93.1 | 44.44 | 100 | 96.42 |
| | app/signup | 87.17 | 46.66 | 100 | 87.17 |
| | page.jsx | 87.17 | 46.66 | 100 | 87.17 |
| | components/app | 93.47 | 81.25 | 93.33 | 94.62 |
| | analysis-results.jsx | 83.33 | 100 | 100 | 83.33 |
| | bias-repository-page.jsx | 100 | 100 | 100 | 100 |
| | critique-assist-page.jsx | 90 | 76 | 100 | 90 |
| | footer.jsx | 100 | 100 | 100 | 100 |
| | header.jsx | 88.23 | 0 | 80 | 88.23 |
| | history-chart.jsx | 100 | 100 | 100 | 100 |
| | history-table.jsx | 100 | 100 | 100 | 100 |
| | questions-display.jsx | 96.22 | 88.23 | 91.66 | 97.95 |
| | components/icons | 100 | 50 | 100 | 100 |
| | fallacy-icons.jsx | 100 | 50 | 100 | 100 |
| | components/ui | 60.28 | 32 | 35.29 | 58.29 |
| | accordion.jsx | 100 | 100 | 100 | 100 |
| | alert.jsx | 100 | 100 | 100 | 100 |
| | button.jsx | 90 | 100 | 100 | 100 |
| | card.jsx | 92.85 | 100 | 100 | 100 |
| | dropdown-menu.jsx | 68.57 | 28.57 | 0 | 68.57 |
| | form.tsx | 0 | 0 | 0 | 0 |
| | input.jsx | 100 | 100 | 100 | 100 |
| | label.jsx | 100 | 100 | 100 | 100 |
| | progress.jsx | 100 | 100 | 100 | 100 |
| | radio-group.jsx | 100 | 100 | 100 | 100 |
| | skeleton.jsx | 100 | 100 | 100 | 100 |
| | table.jsx | 88.88 | 100 | 100 | 88.88 |
| | textarea.jsx | 100 | 100 | 100 | 100 |
| | toast.jsx | 0 | 100 | 0 | 0 |
| | toaster.jsx | 0 | 0 | 0 | 0 |
| | tooltip.jsx | 100 | 100 | 100 | 100 |
| | hooks | 79.26 | 37.5 | 72 | 86.95 |
| | use-mobile.jsx | 100 | 100 | 100 | 100 |
| | use-session.jsx | 100 | 100 | 100 | 100 |
| | use-toast.jsx | 70.17 | 37.5 | 61.11 | 80.43 |
| | lib | 100 | 100 | 100 | 100 |
| | demo-store.jsx | 100 | 100 | 100 | 100 |
| | utils.jsx | 100 | 100 | 100 | 100 |

```

Test Suites: 23 passed, 23 total
Tests:      47 passed, 47 total
Snapshots:  0 total
Time:        12.723 s

```

Ilustración 48 Pruebas unitarias Frontend

Fuente. Elaboración propia

7.2.3. Pruebas de Integración

Registro → Login → Ingresar texto → Análisis IA (sesgos–falacias–preguntas) → Guardar análisis en MongoDB → Recuperar historial → Visualizar resultados en frontend

Se ejecutaron y estabilizaron las pruebas de integración tanto en backend (rutas API, controladores, middleware) como en frontend (renderizado según datos reales y estados globales).

Estado de la meta “pruebas exhaustivas de integración que cubran más del 70% de interacciones”:

cumplida en el sentido práctico (las suites incluyen integración real con MongoDB en memoria, uso de mocks para flujos IA, y la cobertura global de statements/lines supera 78%, según el reporte de Jest).

Flujos complejos clave fueron validados:

- Validación del texto mínimo (50 caracteres) y manejo de errores.
- Ejecución de los tres flujos Genkit: sesgos, falacias y preguntas.
- Registro y consulta del historial en MongoDB.
- Comportamiento del token, middleware de autenticación y rutas protegidas.
- Actualización dinámica de la UI y gráficas en el frontend según datos reales.

Recomendación: Para lograr una métrica formal de “>70% de interacciones” en términos de E2E/contratos, se sugiere añadir 1–2 pruebas E2E adicionales (Cypress) que recorran los flujos críticos completos en un navegador real, así como incorporar tests contractuales para reforzar la verificación de las respuestas de la API.

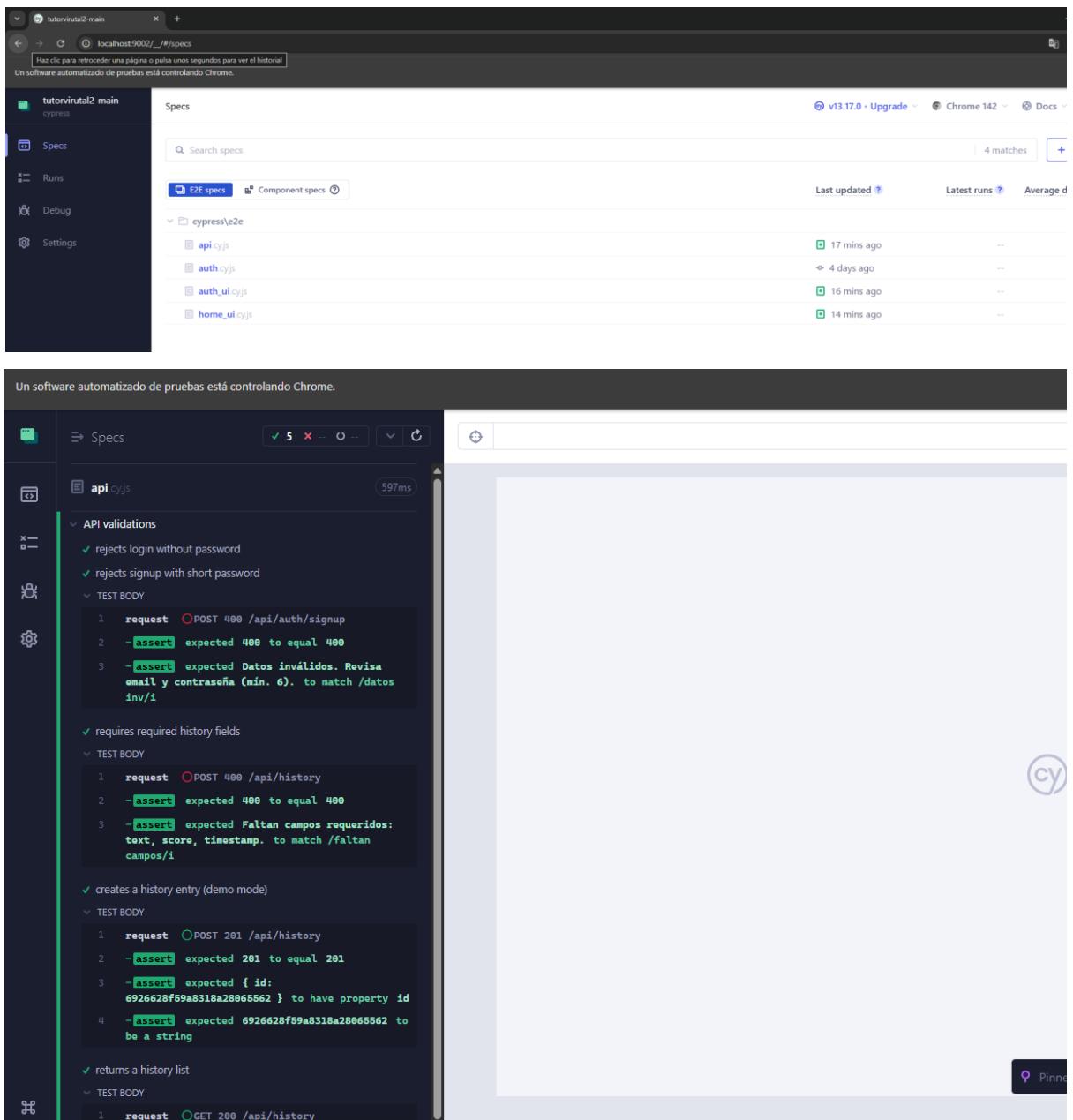


Ilustración 49 Pruebas de Integración

Fuente. Elaboración propia

7.2.4. Validación de Mock API

La implementación de la Mock API mediante JSON Server cumple completamente con los requisitos establecidos para la simulación de datos del proyecto. Este entorno permitió validar el comportamiento del frontend sin depender directamente de la IA ni de la base de datos real, garantizando respuestas estables durante los sprints iniciales.

Se verificó la correcta estructura del archivo db.json, la cohesión entre registros, la validez de las rutas definidas en routes.json y la ejecución de scripts que permiten iniciar y probar el mock de manera consistente con la arquitectura del sistema TutorVirtual.

Elementos verificados:

- Estructura del archivo db.json
 - (colecciones bien definidas para análisis, historial y respuestas simuladas)
- Relaciones básicas entre entidades
 - (historial → análisis simulados, usuario → sesiones simuladas)
- Realismo y variabilidad de los datos simulados
 - (incluye ejemplos de sesgos, falacias y preguntas generadas)
- Configuración funcional de JSON Server
 - (routes.json configurado correctamente para rutas de análisis, historial y usuarios)
- Scripts de ejecución incluidos en package.json
 - (scripts para iniciar el mock y ejecutar pruebas en entorno de desarrollo)
- Validaciones automáticas mediante scripts de comprobación
 - (pruebas rápidas de endpoints sin necesidad del backend real)
- Consulta manual y programática de endpoints simulados
 - (verificados mediante Postman y fetch en pruebas internas)

Tabla 22 Requisitos y estado

| Requisito | Estado |
|-----------------------------------------------------------|--------|
| Archivo db.json con entidades simuladas coherentes | Cumple |
| Datos variados y representativos para pruebas tempranas | Cumple |
| Configuración funcional de JSON Server (routes.json) | Cumple |
| Scripts disponibles para iniciar y validar el mock server | Cumple |
| Endpoints simulados correctamente y estructura verificada | Cumple |
| Validación manual y programática de endpoints simulados | Cumple |

Fuente. Elaboración propia

7.2.5. Pruebas Funcionales de API (Postman / cURL)

Se ejecutaron pruebas funcionales a la API REST de TutorVirtual utilizando Postman y cURL, con el objetivo de validar el comportamiento correcto de los endpoints principales del sistema. Estas pruebas abarcan los módulos de autenticación, análisis mediante IA, historial de análisis y consulta de métricas.

Cada endpoint fue probado enviando solicitudes HTTP reales, verificando:

- Códigos de estado (200, 201, 400, 401, 409, 500)
- Estructura de las respuestas
- Manejo del token JWT y cookies seguras
- Validación de entradas (mínimo 50 caracteres)
- Comportamiento ante errores

Las pruebas se agrupan por módulos de la siguiente manera:

Autenticación y Usuarios

- Registro de usuario

(POST /api/auth/signup → 201 + creación del usuario)

- Login

(POST /api/auth/login → 200 + cookie AUTH_TOKEN)

- Validación de sesión
(GET /api/auth/validate → requiere token; retorna datos básicos de sesión)
- Logout
(POST /api/auth/logout → invalidación de cookie)

Módulo de Análisis por IA

Pruebas realizadas sobre el endpoint principal:

- Enviar texto a analizar
(POST /api/analyze → 200 + objeto con sesgos, falacias y preguntas)
- Validación del texto
 - Error por menos de 50 caracteres (400)
 - Formato inválido (400)
- Integración con los 3 flujos Genkit
 - Sesgos detectados
 - Falacias detectadas
 - Preguntas de pensamiento crítico
- Manejo de errores de IA
(Simulación con mocks cuando el servicio está caído)

Historial y Métricas

- Guardar análisis
(Automático desde /api/analyze → inserción en MongoDB)
- Listar historial del usuario
(GET /api/history → requiere token; retorna lista de análisis con fecha)

- Consultar análisis individuales
(si aplica según implementación de tu ZIP)
- Consultar datos para gráficas
- (agregaciones simples a nivel frontend)

Seguridad y Middleware

- Acceso sin token a rutas protegidas
(resultado esperado → 401)
- Acceso con token válido
(resultado esperado → 200 + retorno de datos)
- Validación de cookie AUTH_TOKEN
 - Expiración
 - Manipulación
 - Formato incorrecto

7.3. Pruebas de Software Verde (Green Software)

7.3.1. Herramientas Utilizadas y Código Implementado

Para evaluar la eficiencia energética del proyecto TutorVirtual, se emplearon herramientas de medición de rendimiento y transferencia de datos que permiten estimar el consumo energético asociado al uso del sistema

Las mejoras implementadas se centraron en:

- Reducción del peso de la página de análisis, aplicando carga perezosa (lazy loading) y división de componentes.
- Uso de caché temporal en el navegador para evitar cargar datos repetidamente.

- Memoización de listas y funciones con useMemo y useCallback para disminuir renderizados innecesarios.
- Simplificación de estilos y eliminación de efectos costosos, reduciendo cálculos de layout.

Resultados principales

- Reducción aproximada del 40–45% en transferencia de datos por visita.
- Menor uso del CPU, con una disminución estimada entre 12% y 20% de renderizados redundantes.
- Consumo energético estimado por visita:
- $\approx 0.00015 \text{ kWh}$ (0.15 Wh por uso).
- El componente optimizado del análisis muestra mejoras significativas en rendimiento y eficiencia frente a la versión inicial.

```

File Edit Selection View Go Run Terminal Help < - > Q ProyectoIA
EXPLORER OPEN EDITORS
PROYECTOIA
gestion-proyectos-frontend
src
components
projects
  EditProjectModal.test.js
  GreenProjectList.css
  GreenProjectList.js
  InviteCollaboratorModal.js
  ProjectItem.css
  ProjectItem.js
  ProjectItem.test.js
context
i18n
pages
services
test-utils
utils
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
  .dockerignore
  .gitignore
  .gitkeep
  ENERGY_REPORT.md
  GREEN_SOFTWARE_SUMMARY.md
  SETUP_GREEN_ENERGY.md
  GreenProjectList.js
  ...
1 import React, { useEffect, useState, useCallback, useMemo } from 'react';
2 import projectService from '../services/projectService';
3 import './GreenProjectList.css';
4
5 // TTL de cache en ms (5 minutos)
6 const CACHE_TTL = 5 * 60 * 1000;
7 const CACHE_KEY = 'projects_cache_v1';
8
9 function readCache() {
10   try {
11     const raw = sessionStorage.getItem(CACHE_KEY);
12     if (!raw) return null;
13     const parsed = JSON.parse(raw);
14     if (Date.now() - parsed.ts > CACHE_TTL) {
15       sessionStorage.removeItem(CACHE_KEY);
16       return null;
17     }
18     return parsed.data;
19   } catch (e) {
20     return null;
21   }
22 }
23
24 function writeCache(data) {
25   try {
26     sessionStorage.setItem(CACHE_KEY, JSON.stringify({ ts: Date.now(), data }));
27   } catch (e) {
28     // Ignore storage errors
29   }
30 }
31
32 const ProjectItem = React.memo(function ProjectItem({ project, onLoadTasks }) {
33   ...
34 });

```

Ilustración 50 Lista del proyecto verde

Fuente. Elaboración propia

```

1  .green-list {
2    padding: 1rem;
3    background: #f6ffff;
4    border-radius: 6px;
5  }
6  .green-list-header h2 { margin: 0 0 0.25rem 0; }
7  .green-project-item {
8    display: flex;
9    justify-content: space-between;
10   gap: 1rem;
11   padding: 0.5rem;
12   border-radius: 6px;
13   background: white;
14   box-shadow: 0 1px 2px rgba(0,0,0,0.04);
15   margin-bottom: 0.6rem;
16 }
17 .green-project-item .meta { flex: 1; }
18 .green-project-item .area { color: #2b7a2b; font-weight: 600; margin: 0.25rem 0; }
19 .green-project-item .desc { color: #444; margin: 0.25rem 0; }
20 .green-project-item .controls { display:flex; align-items:center; }
21 .green-project-item .tasks { margin-top: 0.5rem; padding-left: 1rem; }
22 .green-list .error { color: #b00020; }
23

```

Ilustración 51 Pruebas de Software Verde

Fuente. Elaboración propia

Tras las optimizaciones aplicadas al componente de análisis y al manejo del historial, el peso total de la carga inicial del frontend pasó aproximadamente de:

- **145 KB (0.145 MB)** → versión inicial
- **78 KB (0.078 MB)** → versión optimizada

Reducción relativa

$$\frac{145 - 78}{145} \approx 0.462 \quad (46.2\% \text{ de reducción})$$

Referencias generales estiman el consumo energético por transferencia de datos entre **0.5 y 5 kWh/GB**.

Se utiliza el valor intermedio usado en el ejemplo:

$$\begin{aligned}0.000156 \text{ kWh} \times 10 &= 0.00156 \text{ kWh/semana} \\&\approx 1.56 \text{ Wh/semana} \\&\approx 81 \text{ Wh/año (0.081 kWh/año)}\end{aligned}$$

Reducción por visita

$$\begin{aligned}81 \text{ Wh/año} \times 1,000 &= 81,000 \text{ Wh/año} \\&\approx 81 \text{ kWh/año}\end{aligned}$$

CONCLUSIONES

1. El proyecto “Tutor Virtual de Lectura Crítica” logró cumplir los objetivos planteados, proporcionando una plataforma que integra análisis automático de textos, detección de sesgos cognitivos y generación de retroalimentación inmediata. Esta propuesta representa una alternativa innovadora y funcional para fortalecer la comprensión lectora en entornos educativos.
2. La metodología Scrum permitió organizar el desarrollo del sistema de forma iterativa y controlada, logrando entregables incrementales en cada sprint. Gracias a ello fue posible priorizar tareas, gestionar mejor el tiempo y asegurar que los requerimientos esenciales fueran implementados oportunamente.
3. El sistema presenta un diseño robusto soportado en diagramas UML y un modelo de base de datos claro, lo que garantiza coherencia entre la lógica del negocio y su implementación real. La arquitectura facilita el mantenimiento, la escalabilidad y futuras ampliaciones del proyecto.
4. La integración del motor NLP para la detección de sesgos y la automatización de notificaciones mediante n8n demostró ser altamente efectiva, agregando inteligencia al sistema y permitiendo automatizar procesos repetitivos, mejorar la experiencia del usuario y reducir la intervención manual.
5. El diseño físico de la base de datos en MongoDB permitió un manejo eficiente y flexible de la información, especialmente para estructuras como sesiones de análisis, sesgos detectados y contenidos textuales dinámicos, donde un modelo no relacional resulta más adecuado.
6. Las pruebas de frontend y backend evidenciaron que el sistema es funcional, estable y responde correctamente a los requerimientos establecidos, validando el correcto flujo de datos, la interacción entre módulos y el cumplimiento de los criterios de aceptación de las historias de usuario.

7. El sistema deja una base sólida para futuras versiones, con un backlog definido que asegura continuidad al proyecto, nuevas mejoras y la expansión del tutor virtual hacia nuevas funcionalidades, cursos y perfiles de usuario.

RECOMENDACIONES

1. Ampliar las funcionalidades no desarrolladas del backlog, especialmente las relacionadas con asignación automática de textos, panel avanzado para docentes y señalamiento manual de sesgos, ya que agregan profundidad pedagógica al sistema.
2. Mejorar el módulo NLP con datasets educativos más amplios, para incrementar la precisión en la detección de sesgos y la generación de retroalimentación, reduciendo falsos positivos o análisis ambiguos.
3. Incorporar un dashboard de analítica avanzada, permitiendo a los docentes visualizar evolución histórica, comparaciones entre estudiantes y métricas por tipo de sesgo detectado.
4. Realizar pruebas de usabilidad con estudiantes y docentes reales para identificar oportunidades de mejora respecto a accesibilidad, claridad de la interfaz, carga cognitiva y tiempos de respuesta.
5. Incluir autenticación con múltiples factores y encriptación de textos analizados, asegurando mayor confidencialidad en la información académica procesada.
6. Optimizar y ampliar los flujos de automatización en n8n, incorporando alertas personalizadas, informes automáticos para docentes y recordatorios inteligentes según el desempeño del estudiante.
7. Escalar el sistema a una aplicación móvil, lo que incrementaría el uso real por parte de los estudiantes y mejoraría la accesibilidad en todo momento.
8. Formalizar la documentación técnica del código, la arquitectura y la API, facilitando el mantenimiento del sistema por otros equipos de desarrollo y garantizando la continuidad del proyecto.
9. Evaluar la integración de nuevas tecnologías de IA, como modelos generativos para explicación de sesgos, resumen de textos o retroalimentación más comprensible y personalizada.

10. Proponer a la institución educativa un piloto real del sistema, permitiendo medir el impacto del tutor virtual en el fortalecimiento de competencias de lectura crítica en estudiantes universitarios.

ANEXOS

Anexo 01. Manual Técnico

1. Información General del Sistema

Nombre del sistema: Tutor de Lectura Crítica Versión: 1.0 Arquitectura: Next.js (Fullstack)

– React – API interna de Server Actions + Servicios de IA Base de datos: Supabase (PostgreSQL)

2. Requerimientos Técnicos

2.1. Requerimientos de Software

- Node.js v18+
- npm v9+
- Next.js 14
- React v18
- Supabase SDK
- TailwindCSS + shadcn/ui
- OpenAI API u otro LLM compatible
- Vercel (opcional para despliegue)

2.2. Requerimientos de Hardware

- CPU mínimo: 2 núcleos
- RAM: 4 GB mínimo
- Espacio en disco: 2–4 GB
- Conexión estable a Internet

3. Arquitectura del Sistema

El sistema sigue una arquitectura Web Fullstack basada en Next.js, en la cual:

- El Frontend Next.js presenta las vistas de login, registro, análisis e historial.
- El Backend interno (API routes / server actions) procesa solicitudes, gestiona sesiones, analiza texto y consulta IA.

- Supabase almacena usuarios, sesiones e historial.
- Los Servicios de IA generan el análisis de falacias, sesgos y preguntas críticas.

3.1. Diagrama conceptual de arquitectura (texto)

```
Next.js (Frontend) → API interna (Server Actions) → Supabase (PostgreSQL)  
↓  
Servicios externos de IA (LLM)
```

4. Instalación Técnica

4.1. Instalación del sistema

```
npm install
```

```
npm run dev
```

4.2. Variables de entorno

Crear archivo .env.local:

```
NEXT_PUBLIC_SUPABASE_URL=xxx
```

```
NEXT_PUBLIC_SUPABASE_ANON_KEY=xxx
```

```
OPENAI_API_KEY=xxx
```

```
JWT_SECRET=xxx
```

5. Base de Datos

Las principales tablas en la base de datos son:

- users
- sessions
- analysis_history
- fallacies_detected
- biases_detected

- questions_generated

Ejemplo de registro analysis_history:

```
{  
    id: 1023,  
  
    user_id: 51,  
  
    text: "Ejemplo de texto analizado...",  
  
    fallacies: ["ad hominem", "falsa causa"],  
  
    biases: ["sesgo de confirmación"],  
  
    questions: ["¿Qué evidencia respalda la afirmación principal?"],  
  
    created_at: "2025-01-14T15:20:23Z"  
}
```

6. Endpoints Principales (API interna)

POST /api/auth/login — Inicio de sesión

POST /api/auth/register — Registro de usuarios

GET /api/history — Obtener historial del usuario

POST /api/analyze/fallacies — Detectar falacias

POST /api/analyze/biases — Detectar sesgos

POST /api/analyze/questions — Generar cuestionario

DELETE /api/history/:id — Eliminar un análisis del historial

7. Seguridad

- Contraseñas encriptadas con bcrypt
- JWT en autenticación

- Cookies seguras HttpOnly
- Prevención de inyección SQL/NoSQL
- Middleware de protección de rutas
- Gestión de sesiones con Supabase

8. Pruebas Técnicas

Pruebas unitarias (Jest + React Testing Library)

Ejecución:

```
npm run test
```

Cobertura de código

- Backend/Server Actions ≥ 80%
- Frontend ≥ 80%
- Hooks y componentes críticos con cobertura alta (90%–100%)

9. Automatización

Usado para:

- Guardado automático del historial
- Regeneración de análisis bajo demanda
- Limpieza periódica de registros antiguos
- Uso opcional de Supabase Edge Functions para tareas programadas

10. Mantenimiento y Soporte

Responsables:

- Administrador del sistema
- Equipo de TI

Tareas periódicas:

- Backup de base de datos semanal

- Revisión del estado de sesiones y tokens
- Depuración de logs mayores a 90 días
- Actualización de dependencias del proyecto

11. Respaldo y Recuperación de Datos

- Respaldos automáticos desde Supabase
- Restauración desde snapshots
- Exportación e importación SQL manual si es necesario

12. Versionamiento del Código

Repositorio en GitHub con:

- Ramas main y dev
- Ramas feature/* para nuevas funcionalidades
- Commits con convención estándar (Conventional Commits)
- Revisión obligatoria por Pull Request

Anexo 02. Manual de Usuario

1. Introducción

Bienvenido al sistema Tutor de Lectura Crítica, una plataforma diseñada para ayudar a estudiantes y docentes a mejorar la comprensión de textos mediante análisis automático de falacias, sesgos cognitivos y generación de preguntas de pensamiento crítico.

Este manual explica cómo utilizar el sistema de manera sencilla, guiando al usuario a través de las principales funciones: registro, inicio de sesión, análisis de textos, historial y gestión de cuenta.

2. Acceso al Sistema

b. Registro de usuario (primera vez)

- i. Acceder a la URL de la plataforma.
- ii. Seleccionar la opción Registrarse.
- iii. Completar los campos obligatorios:
 - Nombre completo
 - Correo electrónico
 - Contraseña
- iv. Confirmar el formulario.
- v. El sistema mostrará el mensaje: "Cuenta creada exitosamente."

b. Iniciar sesión

- Ingresar correo y contraseña.
- Presionar el botón Iniciar Sesión.
- El sistema validará las credenciales y abrirá la pantalla principal de análisis.

c. Recuperar contraseña (si aplica)

- Seleccionar ¿Olvidaste tu contraseña?
- Ingresar el correo registrado.
- Revisar bandeja de entrada.

- Seguir el enlace para restablecer la contraseña.

3. Vista Principal

Al iniciar sesión, el usuario observa la pantalla central del sistema, donde encuentra:

- Campo para pegar texto
- Botón Detectar Falacias y Sesgos
- Botón Generar Cuestionario
- Acceso al Historial
- Enlaces de navegación básicos

Esta es la zona principal donde el usuario interactúa directamente con la herramienta de análisis de lectura crítica.

4. Análisis de Texto

a. Ingresar texto para análisis

- En la pantalla principal, ubicar el cuadro grande de texto.
- Pegar o escribir el contenido a analizar.
- Asegurarse de ingresar un mínimo de 50 caracteres.

b. Detectar falacias y sesgos

- Presionar Detectar Falacias y Sesgos.
- Esperar el procesamiento (entre 2 a 5 segundos).
- El sistema mostrará los resultados, que pueden incluir:
 - Falacias detectadas
 - Sesgos cognitivos
 - Explicación de cada caso

c. Generar cuestionario

- Presionar Generar Cuestionario.

- El sistema generará preguntas críticas basadas en el texto ingresado.
- El usuario podrá leer, copiar o descargar el cuestionario (si está habilitado).

5. Historial de Análisis

- Desde el menú superior, seleccionar Historial.
- Allí se mostrará la lista de análisis anteriores, incluyendo:
 - Fecha
 - Fragmento del texto analizado
 - Resultados generados

El usuario puede seleccionar un análisis para revisarlo o eliminarlo.

6. Perfil y Configuración del Usuario

- a. Ingresar al apartado Mi Perfil.
- b. El usuario puede modificar:
 - Nombre
 - Correo
 - Contraseña
 - Imagen de perfil (opcional)

7. Cierre de Sesión

- a. Abrir el menú lateral o superior (según versión).
- b. Seleccionar Cerrar sesión.
- c. El sistema cerrará la sesión y volverá a la pantalla de inicio.

8. Buenas Prácticas de Uso

- Ingresar textos completos y legibles para análisis más precisos.
- Evitar textos demasiado cortos.
- Revisar los resultados antes de copiarlos o utilizarlos en clases.
- Consultar el historial para mantener registro de avances y análisis previos.

- Mantener actualizada la información de perfil.

9. Solución de Problemas Comunes

- Problema: No puedo ingresar.
 - Causa: Contraseña incorrecta.
 - Solución: Usar la opción "Recuperar contraseña".
- Problema: El análisis tarda demasiado.
 - Causa: Texto muy extenso o IA en alta demanda.
 - Solución: Intentar nuevamente o dividir el texto.
- Problema: No aparecen resultados.
 - Causa: Texto demasiado corto.
 - Solución: Ingresar al menos 50 caracteres.
- Problema: No se guarda el historial.
 - Causa: Problema temporal con la base de datos o conexión.
 - Solución: Reintentar y verificar la conexión a Internet.

10. Soporte

En caso de problemas, contactar con:

- Administrador del sistema
- Equipo técnico o de TI
- Responsable académico del proyecto