**UNIVERSIDAD CONTINENTAL**

**FACULTAD DE INGENIERÍA**

**ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**



**PROYECTO**

**TUTOR VIRTUAL DE LECTURA CRÍTICA**

# PORTADA

**PRESENTADO POR:**

|  |  |
| --- | --- |
| **APELLIDOS Y NOMBRES** | **CÓDIGO** |
| **Calderon Espejo Eduardo Martin** | **75345913** |
| **Campos Aquino Alexandra Karim** | **75256374** |
| **Huaman Capcha Estefani Abigail** |  |

**ASESOR:**

**HUANCAYO – PERÚ**

**2023**

# LISTA DE CONTENIDO

[PORTADA 1](#_heading=h.z7dqtecv8zvl)

[LISTA DE CONTENIDO 2](#_heading=h.k5xtzqhrc3uw)

[LISTA DE TABLAS 7](#_heading=h.ic8qiquwj7nj)

[LISTA DE FIGURAS 8](#_heading=h.bpzzs3wvmrah)

[CAPÍTULO 1 9](#_heading=h.b1dcz076inps)

[PLANTEAMIENTO DEL ESTUDIO 9](#_heading=h.tnav31e8y7cx)

[1.1. Aspectos Generales de la Empresa 9](#_heading=h.49s6e4o6e0on)

[1.1.1. Organigrama 9](#_heading=h.tdxlpe5b1k8r)

[1.1.2. Misión y visión 9](#_heading=h.m83bsmv3xvaj)

[1.2. Diagnóstico del Problema 9](#_heading=h.q91hcss5ul8p)

[1.3. Procesos de la Empresa 9](#_heading=h.2wz7vdgytf38)

[1.4. Oportunidad Encontrada 9](#_heading=h.z95k70lb8q6s)

[1.5. Detalles del Proyecto 9](#_heading=h.fn4hlhxpwc8l)

[CAPÍTULO 2 10](#_heading=h.iya66nlfegc8)

[ESTUDIO DE FACTIBILIDAD 10](#_heading=h.vk5ykbzbppz8)

[2.1. Alternativas de Solución 10](#_heading=h.2zkvh7tbbz9h)

[2.2. Factibilidad Técnica 10](#_heading=h.qpv2aodpfs5q)

[2.2.1. Hardware: Servidor 10](#_heading=h.i9zebjjnk7j2)

[2.3. Factibilidad Económica 12](#_heading=h.5r334hmqf9j0)

[2.3.1. Gastos generales 12](#_heading=h.kqcai5uj1my5)

[2.4. Factibilidad Operacional 13](#_heading=h.vc5h7q9bnlso)

[2.4.1. Actividad del Sistema Tutor Virtual 13](#_heading=h.8jlrqpab984r)

[2.4.1.1 Fase de Análisis 13](#_heading=)

[2.4.1.2 Fase de Diseño y Desarrollo 13](#_heading=)

[2.4.1.3 Fase de Transición 14](#_heading=)

[2.4.1.4 Fase de Implementación 15](#_heading=)

[CAPÍTULO 3 16](#_heading=h.4bd4a7rqib39)

[ANÁLISIS DE REQUERIMIENTOS 16](#_heading=h.x2cwfllp5z1r)

[3.1. Metas del Sistema de Información 16](#_heading=h.aymy0dqw2hdh)

[3.2. Requisitos del Sistema 16](#_heading=h.h7da4afoabyw)

[3.2.1. Requerimientos funcionales 16](#_heading=h.fiitkdv90jyk)

[3.2.2. Requerimientos no funcionales 16](#_heading=h.mjpr90beincz)

[3.3. Identificación de Actores del Sistema 16](#_heading=h.rnk4vcshvkhi)

[3.3.1. … 16](#_heading=h.5awyhko7c4dv)

[CAPÍTULO 4 17](#_heading=h.atus9krb9bvb)

[PLANIFICACIÓN DEL PROYECTO 17](#_heading=h.wwdy18ylijqx)

[4.1. Definición de Roles de Trabajo 17](#_heading=h.ikn2l7tsd3h6)

[4.1.1. Product owner 17](#_heading=h.hirwfxodkry9)

[4.1.2. Scrum master 17](#_heading=h.4p3swh9i9mpy)

[4.1.3. Team member 17](#_heading=h.n5lkydpnndmn)

[4.1.4. Tester 17](#_heading=h.3s8uzxart3ca)

[4.2. Product Backlog 17](#_heading=h.d9ya1popefj9)

[4.3. Sprint Backlog 17](#_heading=h.jpxst9cd4qlt)

[4.3.1. Sprint 1 17](#_heading=h.fcgnvggnugjy)

[4.3.2. Sprint 2 17](#_heading=h.z6zlcq78dzyj)

[4.3.3. Sprint 3 17](#_heading=h.nta2buw3xrmh)

[4.3.4. Sprint 4 17](#_heading=h.fh3c30toizvq)

[4.3.5. Sprint 5 18](#_heading=h.93jbem7evau9)

[4.4. Planificación de Sprints 18](#_heading=h.4trdrqpdbr5i)

[4.4.1. Historias de usuario 18](#_heading=h.qc85zxz51n0)

[4.4.2. Priorización de historias de usuario 18](#_heading=h.ikjww2aq5lh3)

[4.5. Cronograma de Actividades 18](#_heading=h.9k8iwkbsxtq8)

[4.6. Gestión de Riesgos 18](#_heading=h.g8knakt1xv3f)

[CAPÍTULO 5 19](#_heading=h.s44lxif439vl)

[DISEÑO DEL SISTEMA DE INFORMACIÓN 19](#_heading=h.pp5tbcoboi61)

[5.1. Diseño de Diagramas UML 19](#_heading=h.qqv04mm42g6k)

[5.1.1. Diagramas de casos de uso 19](#_heading=h.7cj5hrnjxkc5)

[5.1.2. Diagramas de secuencia 19](#_heading=h.9x2txm52dg7o)

[5.1.3. Diagramas de colaboración 19](#_heading=h.63dr0epmo6fz)

[5.1.4. Diagramas de clases 19](#_heading=h.jrb0c7xgx3ve)

[5.2. Diseño de Base de Datos 19](#_heading=h.l16g74290x6h)

[5.2.1. Diseño conceptual (E/R) 19](#_heading=h.qhekyy16o995)

[5.2.2. Diseño lógico 19](#_heading=h.j3lecwq6hu0e)

[5.2.3. Diseño físico 19](#_heading=h.gy324l8m966z)

[5.2.4. Modelado de base de datos 19](#_heading=h.73le9k1b691v)

[5.3. Diseño de Interfaces Básicas 19](#_heading=h.oikeomlo4iq8)

[5.3.1. Acceso login 20](#_heading=h.9ooj3j5ozlzi)

[5.3.2. Interfaz … 20](#_heading=h.ciw1rbpeg04y)

[CAPÍTULO 6 21](#_heading=h.1huy16kcidb9)

[CODIFICACIÓN DEL SOFTWARE 21](#_heading=h.daohw65kyddd)

[6.1. Desarrollo del Sprint 1 21](#_heading=h.pnocbnmporh3)

[6.1.1. Sprint planning 21](#_heading=h.uhsuz86tzvf7)

[6.1.2. Sprint backlog 21](#_heading=h.ab895r5nnxop)

[6.1.3. Historias de usuarios 21](#_heading=h.o6krbwz7tq1f)

[6.1.4. Taskboard 21](#_heading=h.azuee8r9hk9r)

[6.1.5. Daily scrum 21](#_heading=h.9q7zuqd6oaii)

[6.1.6. Sprint review 21](#_heading=h.hny01ozr4ad)

[6.1.7. Criterios de aceptación 21](#_heading=h.ej2jck5g7mmy)

[6.1.8. Resultados del sprint 21](#_heading=h.7ucvf6wuc5y6)

[6.1.8.1. Evidencias. 21](#_heading=h.pblfri7dw4bi)

[6.1.8.2. Prueba de desarrollo. 21](#_heading=h.hcx3fky72yo0)

[6.1.8.3. …. 22](#_heading=h.mhzi5zqro3z7)

[6.1.9. Sprint retrospective 22](#_heading=h.adbiys9290r8)

[6.2. Desarrollo del Sprint 2 22](#_heading=h.kmgvi62p267z)

[6.2.1. Sprint planning 22](#_heading=h.vsb71f975q3q)

[6.2.2. Sprint backlog 22](#_heading=h.77lahxcuj9w)

[6.2.3. Historias de usuarios 22](#_heading=h.15kxp61nsyav)

[6.2.4. Taskboard 22](#_heading=h.8zwuvln0mb6c)

[6.2.5. Daily scrum 22](#_heading=h.gjw042oora8f)

[6.2.6. Sprint review 22](#_heading=h.4j5rfbilbipn)

[6.2.7. Criterios de aceptación 22](#_heading=h.yttu53whnqe3)

[6.2.8. Resultados del sprint 22](#_heading=h.8i937eoxheo5)

[6.2.8.1. Evidencias. 22](#_heading=h.ejmno4xel3oe)

[6.2.8.2. Prueba de desarrollo. 23](#_heading=h.ddfdm4w7y6l1)

[6.2.8.3. …. 23](#_heading=h.thrfylhqrwa9)

[6.2.9. Sprint retrospective 23](#_heading=h.bjtw3g44u0jz)

[6.3. Desarrollo del Sprint 3 23](#_heading=h.5tigjfonjdec)

[6.3.1. Sprint planning 23](#_heading=h.sppyhmwq1mmc)

[6.3.2. Sprint backlog 23](#_heading=h.shat03cwibv)

[6.3.3. Historias de usuarios 23](#_heading=h.g7po3xfqnav6)

[6.3.4. Taskboard 23](#_heading=h.soz7c5vym4bp)

[6.3.5. Daily scrum 23](#_heading=h.r199sba3gc1)

[6.3.6. Sprint review 23](#_heading=h.q81728yz7e28)

[6.3.7. Criterios de aceptación 23](#_heading=h.cxo2mqco7y28)

[6.3.8. Resultados del sprint 23](#_heading=h.oib7kibbybk5)

[6.3.8.1. Evidencias. 24](#_heading=h.3ojbeb3uwygz)

[6.3.8.2. Prueba de desarrollo. 24](#_heading=h.hjz8l2jc9veg)

[6.3.8.3. …. 24](#_heading=h.6j5q82cjnc4q)

[6.3.9. Sprint retrospective 24](#_heading=h.l1z2io91d3cf)

[6.4. Desarrollo del Sprint 4 24](#_heading=h.1w28xmb4kimd)

[6.4.1. Sprint planning 24](#_heading=h.rafoora92yk4)

[6.4.2. Sprint backlog 24](#_heading=h.kgiakcv9xjr4)

[6.4.3. Historias de usuarios 24](#_heading=h.zi16jy4gvxf1)

[6.4.4. Taskboard 24](#_heading=h.a6jon9rhw14z)

[6.4.5. Daily scrum 24](#_heading=h.csdrt8alv3be)

[6.4.6. Sprint review 24](#_heading=h.3ihk741y9ab7)

[6.4.7. Criterios de aceptación 24](#_heading=h.9xyutq39p89m)

[6.4.8. Resultados del sprint 25](#_heading=h.4hjp9asym3cq)

[6.4.8.1. Evidencias. 25](#_heading=h.evbarceemktl)

[6.4.8.2. Prueba de desarrollo. 25](#_heading=h.ttfq98k63sjv)

[6.4.8.3. …. 25](#_heading=h.ujnqs5vzom3w)

[6.4.9. Sprint retrospective 25](#_heading=h.dtct6ep0ju74)

[CAPÍTULO 7 26](#_heading=h.bvageqv0aj6d)

[PRUEBAS DE SOFTWARE 26](#_heading=h.jqb0gfr1ciee)

[7.1. Plan de Pruebas 26](#_heading=h.onahvsxs7kdv)

[CONCLUSIONES 27](#_heading=h.g812lti91hxs)

[RECOMENDACIONES 28](#_heading=h.x2neqnarob8h)

[ANEXOS 29](#_heading=h.lf223sx260iw)

[Anexo 01. Manual Técnico 30](#_heading=h.gazvp07egcp2)

[Anexo 02. Manual de Usuario 31](#_heading=h.nd19ifhuxtok)

# LISTA DE TABLAS

[Tabla 1 Modelo de CPU - Servidor 12](#_Toc214823627)

[Tabla 2 4.2 Product Backlog 27](#_Toc214823628)

[Tabla 3 sprit 1 29](#_Toc214823629)

[Tabla 4 sprint 2 30](#_Toc214823630)

[Tabla 5 sprint 3 30](#_Toc214823631)

[Tabla 6 Planificación de Sprints 31](#_Toc214823632)

[Tabla 7 Sprint backlog 43](#_Toc214823633)

[Tabla 8 Sprint backlog 47](#_Toc214823634)

[Tabla 9 2 Sprint backlog-2 51](#_Toc214823635)

# LISTA DE FIGURAS

[Ilustración 1 Gastos del proyecto 13](#_Toc214823610)

[Ilustración 2 EPC01 - Plataforma Web y Experiencia de Usuario 33](#_Toc214823611)

[Ilustración 3 EPC02 - Inteligencia Artificial para Lectura Crítica 33](#_Toc214823612)

[Ilustración 4 EPC03 - Automatización y Reportes con n8n 34](#_Toc214823613)

[Ilustración 5 Registro e inicio de sesión 34](#_Toc214823614)

[Ilustración 6 Análisis de lectura con IA 35](#_Toc214823615)

[Ilustración 7 Automatización con n8n 35](#_Toc214823616)

[Ilustración 8 Proceso de Login 36](#_Toc214823617)

[Ilustración 9 Análisis de Lectura con IA 36](#_Toc214823618)

[Ilustración 10 Automatización con n8n 37](#_Toc214823619)

[Ilustración 11 Diagramas de clases 37](#_Toc214823620)

[Ilustración 12 Diseño conceptual (E/R) 38](#_Toc214823621)

[Ilustración 13 Diseño lógico 39](#_Toc214823622)

[Ilustración 14 Diseño fisico 40](#_Toc214823623)

[Ilustración 15 . Componentes del sprint 1 en Jira 46](file:///C:\Users\KARIM-ALEXANDRA\Downloads\Plantilla%20de%20Informe%20(1).docx#_Toc214823624)

[Ilustración 16 Componentes del sprint 2 en Jira 50](file:///C:\Users\KARIM-ALEXANDRA\Downloads\Plantilla%20de%20Informe%20(1).docx#_Toc214823625)

[Ilustración 17 Componentes del sprint 3 en Jira 57](#_Toc214823626)

# CAPÍTULO 1

# PLANTEAMIENTO DEL ESTUDIO

## Aspectos Generales de la Empresa

El proyecto Tutor Virtual de Lectura Crítica se desarrolla en el marco de la Escuela Académico Profesional de Ingeniería de Sistemas de la Universidad Continental, con el propósito de aplicar metodologías ágiles y tecnologías innovadoras en el ámbito educativo.  
Este proyecto busca dar respuesta a la necesidad de mejorar la comprensión lectora y el pensamiento crítico en estudiantes universitarios, mediante una plataforma digital que combine la Inteligencia Artificial (IA), el análisis de textos y la automatización de procesos educativos.

El equipo está conformado por estudiantes del área de sistemas que desempeñan roles específicos dentro de un entorno Scrum, lo que permite un desarrollo colaborativo, iterativo y centrado en el valor para el usuario final.

### Organigrama

### Misión y visión

**Misión:**

Desarrollar una plataforma educativa digital que promueva el pensamiento crítico y la comprensión lectora mediante el uso de tecnologías emergentes, brindando a estudiantes y docentes una herramienta eficiente, accesible y adaptativa que mejore sus habilidades de análisis textual y razonamiento lógico.

**Visión:**

Para estudiantes y docentes quienes buscan potenciar su capacidad de análisis crítico y mejorar la comprensión lectora mediante una metodología innovadora, el Tutor Virtual de Lectura Crítica es una plataforma educativa avanzada que utiliza Inteligencia Artificial (IA) para generar preguntas de análisis profundo, identificar sesgos y falacias en los textos, y evaluar automáticamente la comprensión de los estudiantes.

A diferencia de los sistemas tradicionales de enseñanza que solo entregan contenido estático, nuestro producto ofrecerá una automatización completa con N8N, que gestionará la asignación de actividades, enviará recordatorios personalizados y proporcionará reportes detallados del progreso de los estudiantes, ayudando a desarrollar hábitos de lectura crítica de manera eficiente y accesible.

## Diagnóstico del Problema

En la actualidad, se evidencia una deficiencia en la comprensión lectora y el pensamiento crítico entre los estudiantes universitarios. Muchos de ellos presentan dificultades para identificar ideas principales, argumentos, falacias y sesgos en textos académicos o de opinión.  
A esto se suma la falta de herramientas interactivas que permitan una práctica constante y personalizada, lo que limita el desarrollo de competencias analíticas necesarias para el ámbito académico y profesional.

Los métodos tradicionales de enseñanza de lectura crítica se basan en materiales estáticos y evaluaciones manuales, lo que genera poca motivación y escaso seguimiento del progreso individual.  
Por tanto, existe la necesidad de una solución tecnológica innovadora que automatice y personalice el aprendizaje, utilizando IA para generar ejercicios dinámicos, retroalimentaciones automáticas y métricas de desempeño en tiempo real.

## Procesos de la Empresa

El proceso de desarrollo del proyecto se estructura bajo la metodología Scrum, lo que permite un flujo de trabajo iterativo e incremental.  
Los principales procesos que se llevan a cabo son:

1. Planificación del Sprint: definición de historias de usuario y priorización de tareas.
2. Desarrollo Frontend: diseño e implementación de las interfaces interactivas con base en Figma.
3. Desarrollo Backend: creación de la lógica del sistema, gestión de la base de datos y APIs.
4. Pruebas de Calidad: verificación funcional y corrección de errores por el área de QA.
5. Revisión y Retrospectiva: análisis del desempeño y mejoras continuas del equipo.
6. Documentación: registro técnico y funcional del avance del proyecto en GitHub y Jira.

## Oportunidad Encontrada

La principal oportunidad identificada radica en mejorar la educación lectora con el apoyo de la tecnología, aplicando IA y automatización para personalizar la enseñanza y evaluación.  
El mercado educativo digital en Perú aún tiene pocas herramientas enfocadas en lectura crítica, lo cual representa una ventaja competitiva para desarrollar una plataforma pionera en este campo.

Además, el uso de herramientas open source y APIs inteligentes permite reducir costos de desarrollo y acelerar la implementación, generando una solución escalable para instituciones educativas que buscan fortalecer las competencias comunicativas de sus estudiantes.

## Detalles del Proyecto

**Nombre del Proyecto:** Tutor Virtual de Lectura Crítica

**Tipo de Proyecto:** Plataforma web educativa con Inteligencia Artificial

**Metodología de Desarrollo:** Scrum

**Tecnologías Utilizadas:**

* Frontend: HTML, CSS, JavaScript, React o Vue.js
* Backend: Node.js, Express
* Base de Datos: MongoDB
* Automatización: N8N
* Diseño: Figma
* Gestión: Jira y GitHub

**Objetivo General:**Desarrollar una plataforma educativa virtual que fortalezca la comprensión lectora y el pensamiento crítico en estudiantes universitarios mediante el uso de IA y herramientas interactivas.

**Duración Estimada:** 3 sprints de desarrollo iterativo.

**Equipo Responsable:** Integrantes de la Escuela Profesional de Ingeniería de Sistemas.

# CAPÍTULO 2

# ESTUDIO DE FACTIBILIDAD

## Alternativas de Solución

El software educativo en desarrollo será construido bajo el nombre “Tutor Virtual de Lectura Crítica”, el cual se implementará como una plataforma web inteligente destinada a fortalecer la comprensión lectora y el pensamiento crítico de los estudiantes.

El sistema será desarrollado utilizando JavaScript como lenguaje principal y Node.js como entorno de ejecución, empleando MongoDB como sistema gestor de base de datos por su flexibilidad y capacidad para manejar grandes volúmenes de información no estructurada.

Asimismo, se integrará n8n como herramienta de automatización para la gestión de tareas y notificaciones automáticas, y se incluirán módulos de Inteligencia Artificial (IA) para la generación de preguntas analíticas y detección de sesgos en los textos.

El uso de estas tecnologías garantiza un desarrollo rápido, seguro y de bajo costo, al utilizar plataformas open-source y servicios gratuitos en la nube como Vercel o Netlify para el despliegue de la aplicación.  
 De esta manera, el proyecto ofrece una solución moderna, adaptable y sostenible dentro del entorno educativo.

## Factibilidad Técnica

La factibilidad técnica se evalúa mediante los enfoques de **hardware y software**, garantizando que el sistema pueda ser implementado correctamente con los recursos disponibles y el conocimiento técnico del equipo de desarrollo.

### Hardware: Servidor

El CPU que servirá como servidor también presenta características de **Intel Core i9-12900K** de 16 núcleos y 24 hilos de ejecución; **64MB de caché**, frecuencia base de **3.2GHz** y máxima con turbo de **5.2GHz**, capacidad de memoria **DDR5** de alto rendimiento, **28 LANES PCI Express**, un **TDP de 125W**, y link speed de **16GT/s** en los PCI Express.

Este equipo de gama alta permitirá ejecutar procesos de Inteligencia Artificial, automatización y almacenamiento de datos de manera eficiente y estable.

Tabla Modelo de CPU - Servidor

|  |  |
| --- | --- |
|  | **Valores** |
| **1** | **Número de núcleos:** 16 |
| **2** | **Memoria caché:** 64 MB |
| **3** | **Frecuencia mínima:** 2.4 GHz |
| **4** | **Frecuencia base / máxima turbo:** 3.2 / 5.2 GHz |
| **5** | **RAM compatible:** DDR5 |
| **6** | **Memoria ECC compatible:** No |
| **7** | **Número de PCI:** 1x16 + 1x8 + 1x4 |
| **8** | **Máximo PCI Express:** 16 GT/s |
| **9** | **TDP:** 125 W |
| **10** | **Disco Duro:** SSD NVMe 1 TB |
| **11** | **Tarjeta de red:** 2500 Mbps – Wi-Fi 6 |
| **12** | **Tarjeta de video:** NVIDIA RTX 4070 Ti – 12 GB GDDR6X |

### 

## Factibilidad Económica

Para realizar el estudio de factibilidad económica se analizaron los gastos generales asociados al desarrollo del sistema de información y los participantes involucrados.  
El proyecto emplea recursos gratuitos y de código abierto, por lo que los costos de desarrollo se mantienen en un nivel accesible.

### Gastos generales

*Ilustración 1 Gastos del proyecto*

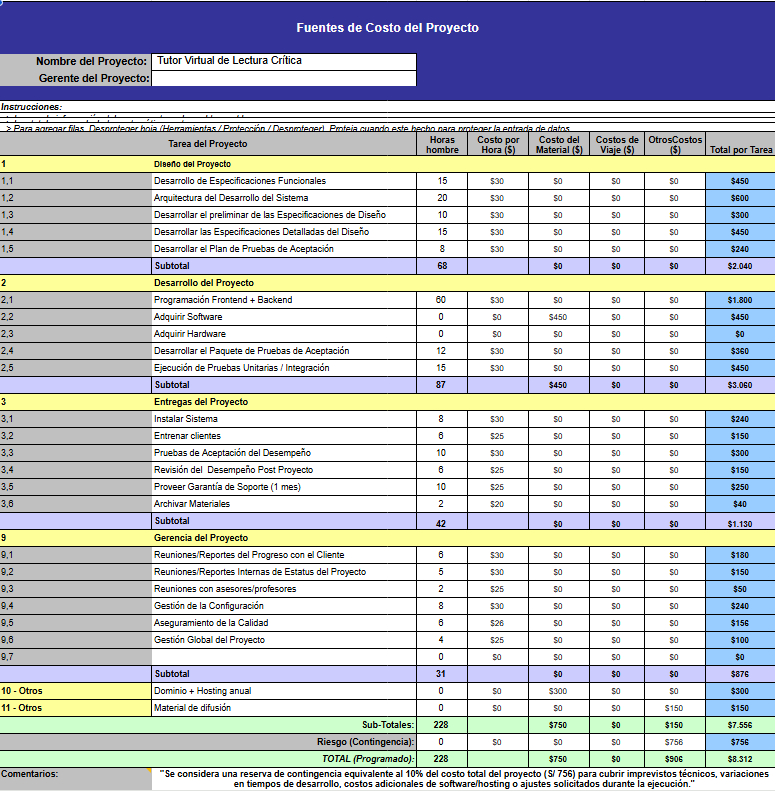


Ilustración Gastos del proyecto

*Fuente. Elaboración propia*

## Factibilidad Operacional

La factibilidad operacional permite determinar la capacidad de implementación y funcionamiento del sistema en el entorno académico.  
El Tutor Virtual de Lectura Crítica ofrecerá una interfaz accesible para estudiantes y docentes, facilitando la práctica de lectura crítica y la generación automática de reportes de desempeño.

### Actividad del Sistema Tutor Virtual

### 2.4.1.1 Fase de Análisis

* Estudio previo y entrevistas con docentes y estudiantes.
* Identificación de requisitos de usuario (panel docente y panel estudiante).
* Identificación de requisitos de rendimiento, hardware y software.
* Identificación de requisitos de interfaz y accesibilidad.
* Especificación de requerimientos del producto.
* Análisis de los flujos de trabajo (lectura, preguntas, retroalimentación y reportes).

### 2.4.1.2 Fase de Diseño y Desarrollo

Durante esta fase se realizará la planificación, codificación y maquetación del sistema a través de tres sprints principales:

**Sprint 1**

* Registro e inicio de sesión de usuarios.
* Creación del perfil docente y configuración de datos persistentes.
* Diseño de la interfaz intuitiva del sistema.
* Implementación del panel de lecturas asignadas (frontend y pruebas básicas).

**Sprint 2**

* Desarrollo del módulo de generación automática de preguntas mediante IA.
* Implementación del sistema de detección de sesgos y falacias en los textos.
* Desarrollo del módulo de retroalimentación automática para los estudiantes.
* Creación del panel docente para el seguimiento del desempeño y visualización de métricas.

**Sprint 3**

* Configuración de flujos automatizados con n8n para recordatorios y tareas.
* Desarrollo del módulo de reportes detallados para docentes.
* Implementación de notificaciones personalizadas y sistema de alertas.
* Pruebas funcionales e integración completa del sistema.

### 2.4.1.3 Fase de Transición

* Carga de lecturas iniciales y configuración de usuarios de prueba.
* Validación del rendimiento del motor IA en distintos navegadores.
* Pruebas internas de funcionamiento (QA y validación de interfaz).
* Pruebas piloto con un grupo de 10 a 15 estudiantes.
* Corrección de errores y ajustes finales.

### 2.4.1.4 Fase de Implementación

* Despliegue del sistema en la nube (Vercel y MongoDB Atlas).
* Entrega final del aplicativo funcional.
* Elaboración del manual técnico y del manual del usuario.
* Capacitación a los docentes sobre el uso de la plataforma y sus reportes.

El recurso de tiempo queda plasmado en un cronograma de desarrollo estructurado en sprints iterativos, en los cuales se determinan los plazos, prioridades, duración y responsables para cada una de las actividades descritas con anterioridad.  
De esta manera, se garantiza una implementación progresiva y controlada del sistema, asegurando su funcionamiento óptimo y su fácil adopción por parte de los usuarios finales.

# CAPÍTULO 3

# ANÁLISIS DE REQUERIMIENTOS

## Metas del Sistema de Información

El sistema Tutor Virtual de Lectura Crítica tiene como meta principal desarrollar una plataforma educativa que mejore la comprensión lectora y el pensamiento crítico mediante el uso de Inteligencia Artificial, automatización de procesos y un entorno web intuitivo orientado a estudiantes y docentes.

Estas metas se organizan en **tres EPICS**, que representan los entregables principales del sistema:

### EPC01: Plataforma Web y Experiencia de Usuario

Permite la interacción inicial del usuario, acceso al sistema, configuración del perfil y gestión de lecturas.

### EPC02: Inteligencia Artificial para Lectura Crítica

Incluye los módulos inteligentes que permiten:

* Analizar textos
* Generar preguntas
* Detectar falacias
* Retroalimentar automáticamente al estudiante

### EPC03: Automatización y Reportes con N8N

Automatiza procesos internos del sistema como:

* Envío de recordatorios
* Notificaciones personalizadas
* Registro automático de métricas
* Generación de reportes

## Requisitos del Sistema

### Requerimientos funcionales

Basados en las Historias de Usuario (HU) definidas en Jira para los Sprints 1, 2 y 3.

Los **requerimientos funcionales** describen las funciones que el sistema debe realizar para cumplir con los objetivos del proyecto.

### EPIC: EPC01 Plataforma Web y UX (HU1–HU4)

**RF01.** El sistema debe permitir el registro e inicio de sesión (HU1).  
**RF02.** El sistema debe permitir configurar el perfil del docente (HU2).  
**RF03.** El sistema debe ejecutar pruebas básicas para validar funcionamiento (HU3).  
**RF04.** El docente debe visualizar y gestionar lecturas asignadas (HU4).

### EPIC: EPC02 Inteligencia Artificial para Lectura Crítica (HU5–HU8)

**RF05.** El sistema debe generar preguntas de análisis profundo mediante IA (HU5).  
**RF06.** El sistema debe detectar falacias y sesgos en textos académicos (HU6).  
**RF07.** El sistema debe ofrecer retroalimentación automática e inmediata al estudiante (HU7).  
**RF08.** El sistema debe mostrar en el panel docente el progreso de los estudiantes (HU8).

### EPIC: EPC03 Automatización y Reportes con N8N (HU9–HU13)

**RF09.** El sistema debe enviar recordatorios automáticos mediante n8n (HU9).  
**RF10.** El sistema debe emitir notificaciones personalizadas según rendimiento (HU10).  
**RF11.** El sistema debe generar reportes detallados y métricas gráficas (HU11).  
**RF12.** El sistema debe registrar automáticamente métricas de uso y desempeño (HU12).  
**RF13.** El sistema debe garantizar la seguridad de datos mediante protocolos actuales (HU13).

### Requerimientos no funcionales

Los requerimientos no funcionales definen las características de calidad que debe cumplir el sistema Tutor Virtual de Lectura Crítica para garantizar su adecuado funcionamiento, seguridad y experiencia de uso. Estos requerimientos complementan a los funcionales y permiten asegurar la eficiencia, confiabilidad y accesibilidad de la plataforma.

### RNF01. Seguridad

El sistema debe garantizar la protección de la información personal y académica de los usuarios, aplicando medidas como:

* Cifrado de contraseñas mediante algoritmos seguros (bcrypt u otro equivalente).
* Uso obligatorio del protocolo **HTTPS** para todas las comunicaciones.
* Implementación de **tokens JWT** para el manejo seguro de sesiones.
* Control de acceso basado en roles (Administrador, Docente, Estudiante).
* Registro de eventos críticos para auditoría (intentos fallidos, accesos no autorizados).

### RNF02. Rendimiento

* Las páginas deben cargar en un tiempo máximo de **3 segundos** bajo condiciones normales de uso.
* El tiempo de generación de preguntas mediante IA no debe exceder los **6 segundos**.
* El sistema debe soportar al menos **50 usuarios concurrentes** sin degradación perceptible.

### RNF03. Usabilidad y UI/UX

* La interfaz debe ser intuitiva, consistente y de fácil navegación para estudiantes y docentes.
* El diseño visual debe seguir lineamientos modernos de **UX/UI**, alineados al prototipo realizado en Figma.
* Debe proporcionar feedback claro al usuario en cada acción (mensajes, alertas, validaciones).

### RNF04. Accesibilidad

* El sistema debe cumplir con los estándares **WCAG 2.1 nivel AA**, garantizando accesibilidad a personas con discapacidad visual, motriz o cognitiva.
* El contenido debe ser totalmente funcional mediante teclado y compatible con lectores de pantalla.

### RNF05. Compatibilidad

La plataforma debe funcionar correctamente en los principales navegadores y dispositivos:

* Google Chrome
* Mozilla Firefox
* Microsoft Edge
* Safari
* Compatibilidad con dispositivos móviles, tablets y computadoras (Responsive Design).

### RNF06. Escalabilidad

* La arquitectura basada en Node.js y MongoDB debe permitir el crecimiento del sistema sin necesidad de reestructurarlo.
* Debe ser posible integrar nuevos módulos basados en IA o flujos de automatización sin afectar el desempeño general.

### RNF07. Mantenibilidad

* El código debe seguir estándares de calidad (Clean Code, SOLID).
* El proyecto debe estar documentado en GitHub siguiendo buenas prácticas de versionado.
* Se debe mantener una estructura modular para facilitar actualizaciones y correcciones.

### RNF08. Disponibilidad

* El sistema debe permanecer disponible al menos **el 95% del tiempo**.
* Los servicios en la nube (Vercel, MongoDB Atlas y n8n) deben contar con mecanismos de respaldo y recuperación.

## Identificación de Actores del Sistema

Los actores representan a los usuarios o sistemas que interactúan con el *Tutor Virtual de Lectura Crítica*. Estos se han clasificado según los **EPICs definidos en Jira** para asegurar una relación clara entre funcionalidades y responsables.

## Actores del EPIC: EPC01 – Plataforma Web y Experiencia de Usuario

### 1. Estudiante

### Usuario principal que interactúa con las lecturas, preguntas y retroalimentación generada por la IA. Responsabilidades:

### Registrarse e iniciar sesión en el sistema (HU1).

### Visualizar y resolver las lecturas asignadas (HU4).

### Revisar su progreso y resultados.

### 2. Docente

### Encargado de gestionar contenidos y monitorear el avance académico. Responsabilidades:

### Iniciar sesión y gestionar su perfil (HU1, HU2).

### Asignar lecturas y actividades (HU4).

### Administrar el avance de los estudiantes.

### 3. Administrador

### Responsable del mantenimiento global del sistema. Responsabilidades:

### Crear, editar y eliminar usuarios (HU16–HU18).

### Supervisar el correcto funcionamiento del sistema.

### 4. Sistema de Autenticación (Auth Service)

### Actor externo encargado de validar credenciales y generar tokens. Funciones:

### Validar usuario y contraseña (HU1).

### Gestionar sesiones seguras (JWT)

## Actores del EPIC: EPC02 – Inteligencia Artificial para Lectura Crítica

### 5. Motor de Inteligencia Artificial (IA)

### Componente externo encargado del procesamiento automatizado del texto. Funciones:

### Generación de preguntas de análisis crítico (HU5).

### Detección de sesgos y falacias (HU6).

### Generación de retroalimentación automática (HU7).

### 6. Backend / Orquestador de Análisis

### Módulo interno que coordina la comunicación entre IA, base de datos y frontend. Funciones:

### Enviar textos a la IA para su análisis.

### Almacenar preguntas y respuestas en la base de datos.

### Proveer información al panel docente (HU8).

## Actores del EPIC: EPC03 – Automatización y Reportes con n8n

### 7. Plataforma de Automatización (n8n)

### Herramienta externa usada para automatizar procesos del sistema. Funciones:

### Enviar recordatorios automáticos (HU9).

### Emitir notificaciones personalizadas según desempeño (HU10).

### Gestionar flujos automatizados.

### 8. Base de Datos (MongoDB Atlas)

### Almacén de datos en la nube. Funciones:

### Guardar usuarios, lecturas, respuestas, métricas y reportes (HU12).

### Mantener integridad, seguridad y disponibilidad de datos.

### 9. Módulo de Reportes

### Componente interno que procesa y visualiza la información. Funciones:

### Generar reportes detallados con gráficos y métricas (HU11).

### Facilitar la toma de decisiones por parte del docente.

### 

# CAPÍTULO 4

# PLANIFICACIÓN DEL PROYECTO

## 

## Definición de Roles de Trabajo

### 4.1.1. Product Owner

### Responsable: *Rodríguez Santiago Luis Gerardo* Define las funcionalidades, prioriza el Product Backlog y asegura la alineación del proyecto con los objetivos académicos.

### 4.1.2. Scrum Master

### Responsable: *Marlon Andrés Salinas Gamión* Facilita ceremonias Scrum, elimina impedimentos y asegura el cumplimiento del marco ágil.

### 4.1.3. Team Member (Desarrolladores)

### Backend: Calderón Espejo Eduardo

### Frontend: Huamán Capcha Estefani Abigail Desarrollan la lógica, interfaces y funcionalidad integral del sistema.

### 4.1.4. Tester / QA

### Responsable: *Jhosep Steven Huali Cuyotupa* Ejecuta pruebas y validar requisitos funcionales y no funcionales.

### 4.1.5. Analista Funcional

### Responsable: *Alexandra Karim Campos Aquino* Define HU, especificaciones y modelado de procesos.

### 4.1.6. Diseñador UX/UI

### Responsable: *Jhoanes Lenin Espejo Mayhua* Produce prototipos, diseño visual y experiencia de usuario.

### 4.1.7. Documentador

### Responsable: *Kenyi Mauri Yaranga Ruiz* Administra documentación, control de versiones y manuales.

## Product Backlog

Tabla 4.2 Product Backlog

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **EPIC** | **Historia de Usuario** | **Descripción breve** | **Prioridad** |
| **HU1** | EPC01 | Registro e inicio de sesión | Permite crear cuentas e iniciar sesión mediante credenciales seguras. | Alta |
| **HU2** | EPC01 | Perfil docente | Permite editar y visualizar los datos del docente. | Alta |
| **HU3** | EPC01 | Pruebas básicas del sistema | Verificar funcionamiento inicial de rutas, vistas y servicios base. | Alta |
| **HU4** | EPC01 | Panel de lecturas asignadas | Permite gestionar y visualizar lecturas. | Alta |
| **HU5** | EPC02 | Generación de preguntas (IA) | Crear preguntas analíticas a partir de un texto. | Alta |
| **HU6** | EPC02 | Detección de sesgos y falacias | Analizar el texto e identificar falacias argumentativas. | Alta |
| **HU7** | EPC02 | Retroalimentación automática | Calificar y retroalimentar las respuestas del estudiante. | Alta |
| **HU8** | EPC02 | Panel de desempeño | Mostrar métricas y rendimiento de los estudiantes. | Alta |
| **HU9** | EPC03 | Recordatorios automáticos (n8n) | Notificar actividades pendientes mediante flujos automatizados. | Media |
| **HU10** | EPC03 | Notificaciones personalizadas | Alertas inteligentes según desempeño. | Media |
| **HU11** | EPC03 | Reportes detallados | Generar informes gráficos de progreso. | Alta |
| **HU12** | EPC03 | Registro automático de métricas | Guardar datos de uso y desempeño. | Alta |
| **HU13** | EPC03 | Seguridad de datos | Implementación de protocolos de seguridad y privacidad. | Alta |

## Sprint Backlog

### 4.3.1. Sprint 1

|  |  |  |
| --- | --- | --- |
| Duración: 15 días | Inicio: 08/09/2025 | Final: 22/09/2025 |

### 

### Objetivo:

### Implementar las funcionalidades esenciales de acceso al sistema, pruebas iniciales del entorno y gestión básica de lecturas asignadas, garantizando que los usuarios puedan registrarse, iniciar sesión y acceder al contenido mínimo del tutor virtual.

### Actividades clave:

### Configuración del entorno de desarrollo.

### Desarrollo de la funcionalidad de registro e inicio de sesión.

### Implementación del perfil docente.

### Creación del panel de lecturas asignadas.

### Ejecución de pruebas básicas del sistema.

### Tabla 1. Sprint 1

Tabla sprit 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| REQ | ID | TIPO | DESCRIPCIÓN | ÉPICA |
| R01 | HU1 | HU | Registro e inicio de sesión | EPC01 Plataforma Web |
| R02 | HU2 | HU | Perfil docente | EPC01 Plataforma Web |
| R03 | HU3 | HU | Pruebas básicas del sistema | EPC01 Plataforma Web |
| R04 | HU4 | HU | Panel de lecturas asignadas | EPC01 Plataforma Web |
| R05 | HT01 | HT | Configuración del entorno MERN | EPC01 Plataforma Web |

### 

### *Tabla 1: Sprint 1. Elaboración propia.*

### 4.3.2. Sprint 2

|  |  |  |
| --- | --- | --- |
| Duración: 14 días | Inicio: 03/10/2025 | Final: 17/10/2025 |

### Objetivo:

### Desarrollar las funcionalidades centrales basadas en inteligencia artificial para análisis de lectura crítica, incluyendo la generación de preguntas, detección de falacias y retroalimentación automática.

### Actividades clave:

### Integración del motor de IA con backend.

### Implementación del módulo de generación de preguntas.

### Desarrollo de detección de sesgos y falacias.

### Integración del módulo de retroalimentación automática.

### Desarrollo del panel docente de desempeño.

### Tabla 2. Sprint 2

Tabla sprint 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| REQ | ID | TIPO | DESCRIPCIÓN | ÉPICA |
| R06 | HU5 | HU | Generación de preguntas de análisis | EPC02 IA Lectura Crítica |
| R07 | HU6 | HU | Detección de sesgos y falacias | EPC02 IA Lectura Crítica |
| R08 | HU7 | HU | Retroalimentación automática | EPC02 IA Lectura Crítica |
| R09 | HU8 | HU | Panel docente de desempeño | EPC02 IA Lectura Crítica |
| R10 | HT02 | HT | Integración del motor de IA al backend | EPC02 IA Lectura Crítica |

### 

### *Tabla 2: Sprint 2. Elaboración propia.*

### 4.3.3. Sprint 3

|  |  |  |
| --- | --- | --- |
| Duración: 14 días | Inicio: 24/10/2025 | Final: 07/11/2025 |

### Objetivo:

### Implementar los módulos de automatización con n8n y completar los reportes detallados, métricas automáticas y mejoras de seguridad del sistema.

### Actividades clave:

### Implementación de flujos en n8n.

### Envío de recordatorios automáticos.

### Notificaciones personalizadas.

### Registro automático de métricas.

### Desarrollo de reportes detallados.

### Implementación de seguridad de datos (roles, permisos, validaciones).

### Tabla 3. Sprint 3

Tabla sprint 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| REQ | ID | TIPO | DESCRIPCIÓN | ÉPICA |
| R11 | HU9 | HU | Recordatorios automáticos (n8n) | EPC03 Automatización |
| R12 | HU10 | HU | Notificaciones personalizadas | EPC03 Automatización |
| R13 | HU11 | HU | Reportes detallados | EPC03 Automatización |
| R14 | HU12 | HU | Registro automático de métricas | EPC03 Automatización |
| R15 | HU13 | HU | Seguridad de datos | EPC03 Automatización |
| R16 | HT03 | HT | Integración de n8n con backend | EPC03 Automatización |

### 

### *Tabla 3: Sprint 3. Elaboración propia.*

### 

## Planificación de Sprints

Tabla Planificación de Sprints

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Duración** | **EPIC** | **Objetivo Principal** |
| **Sprint 1** | 2 semanas | EPC01 | Base del sistema y pantallas principales |
| **Sprint 2** | 2 semanas | EPC02 | Módulos IA y panel docente |
| **Sprint 3** | 2 semanas | EPC03 | Automatización y reportes |
| **Sprint 4** | 1 semana | — | Correcciones y mejoras |
| **Sprint 5** | 1 semana | — | Documentación y despliegue |

### Historias de usuario

### Priorización de historias de usuario

## Cronograma de Actividades

## Gestión de Riesgos

# CAPÍTULO 5

# DISEÑO DEL SISTEMA DE INFORMACIÓN

## Diseño de Diagramas UML

### Diagramas de casos de uso

Ilustración EPC01 - Plataforma Web y Experiencia de Usuario

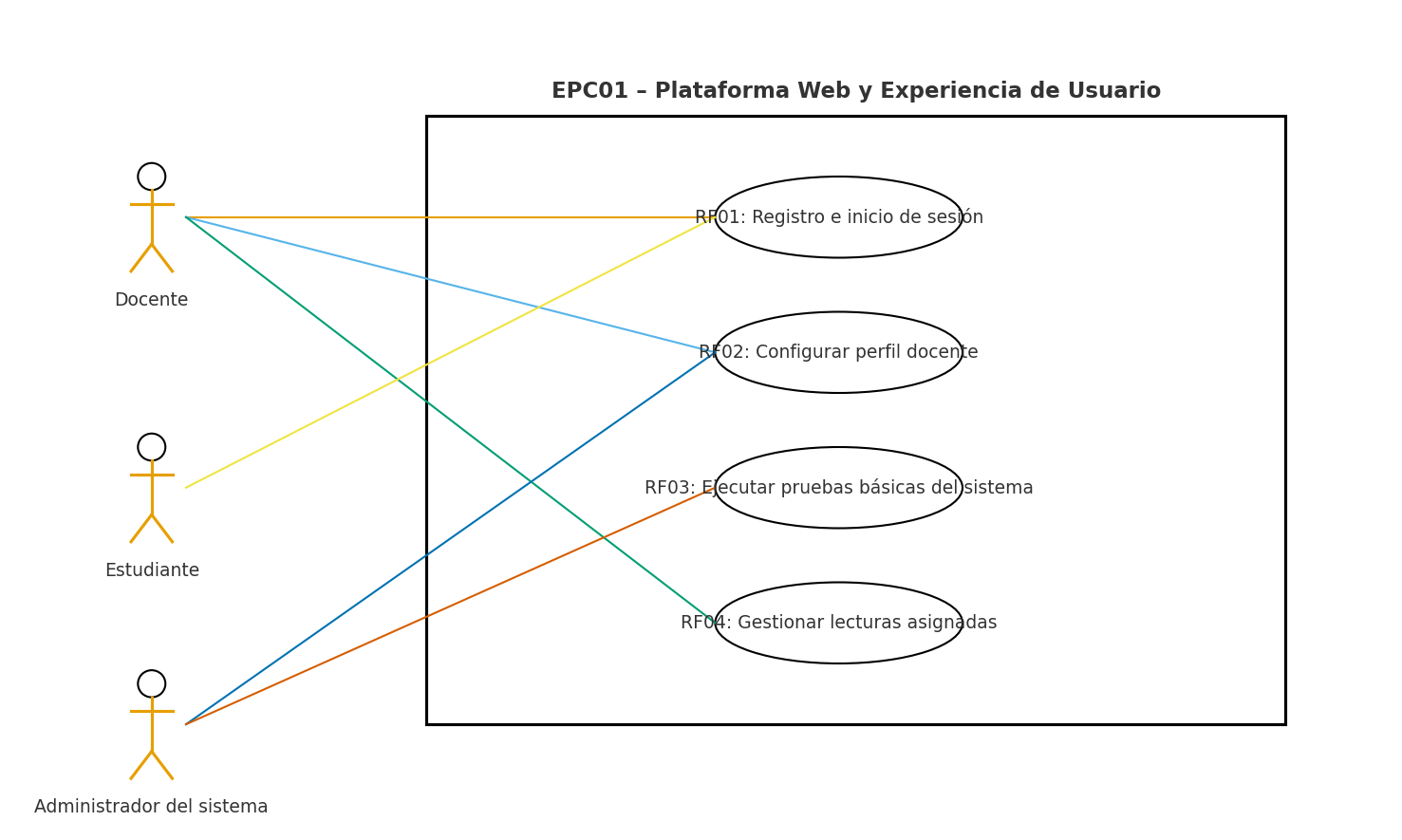


Ilustración EPC02 - Inteligencia Artificial para Lectura Crítica

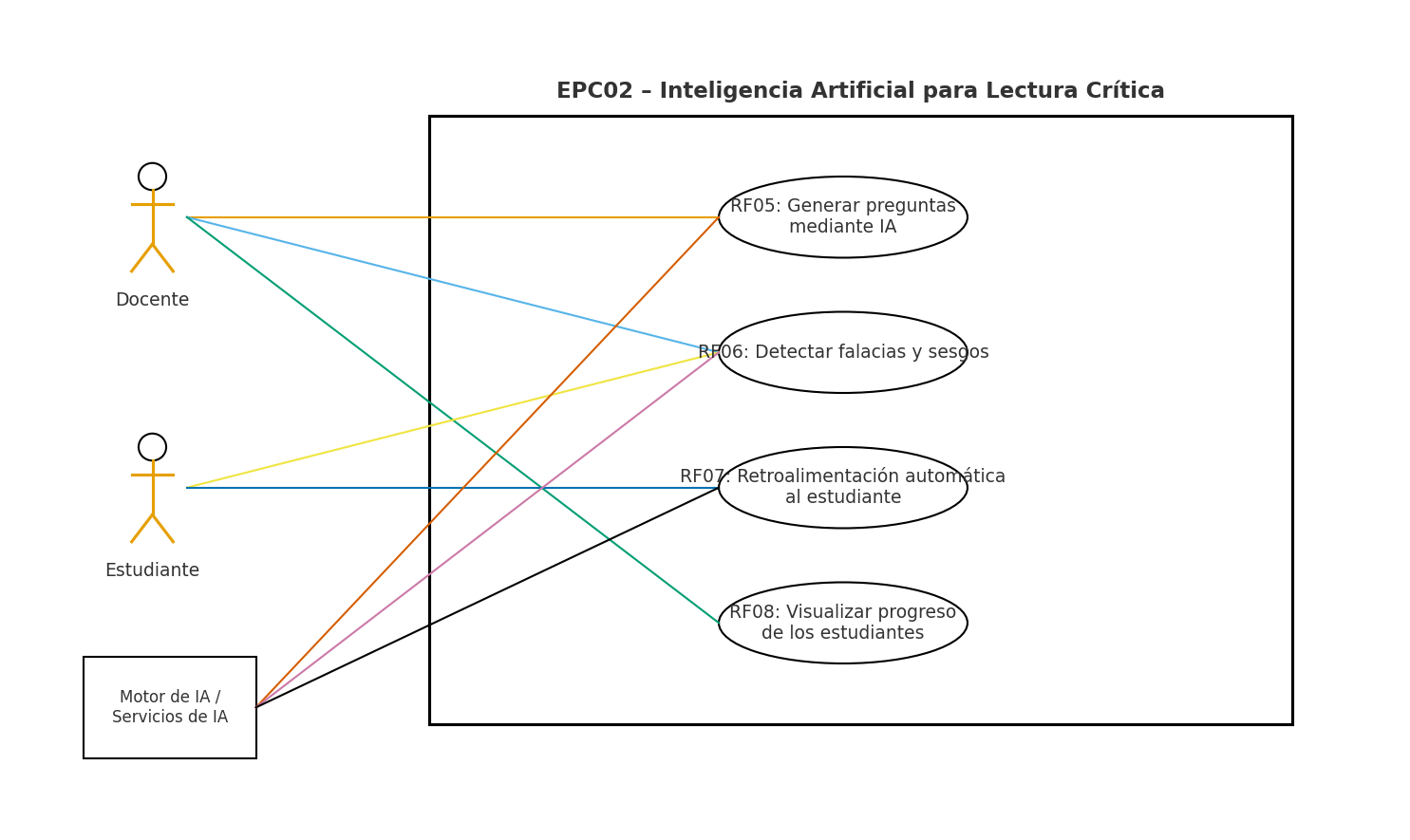
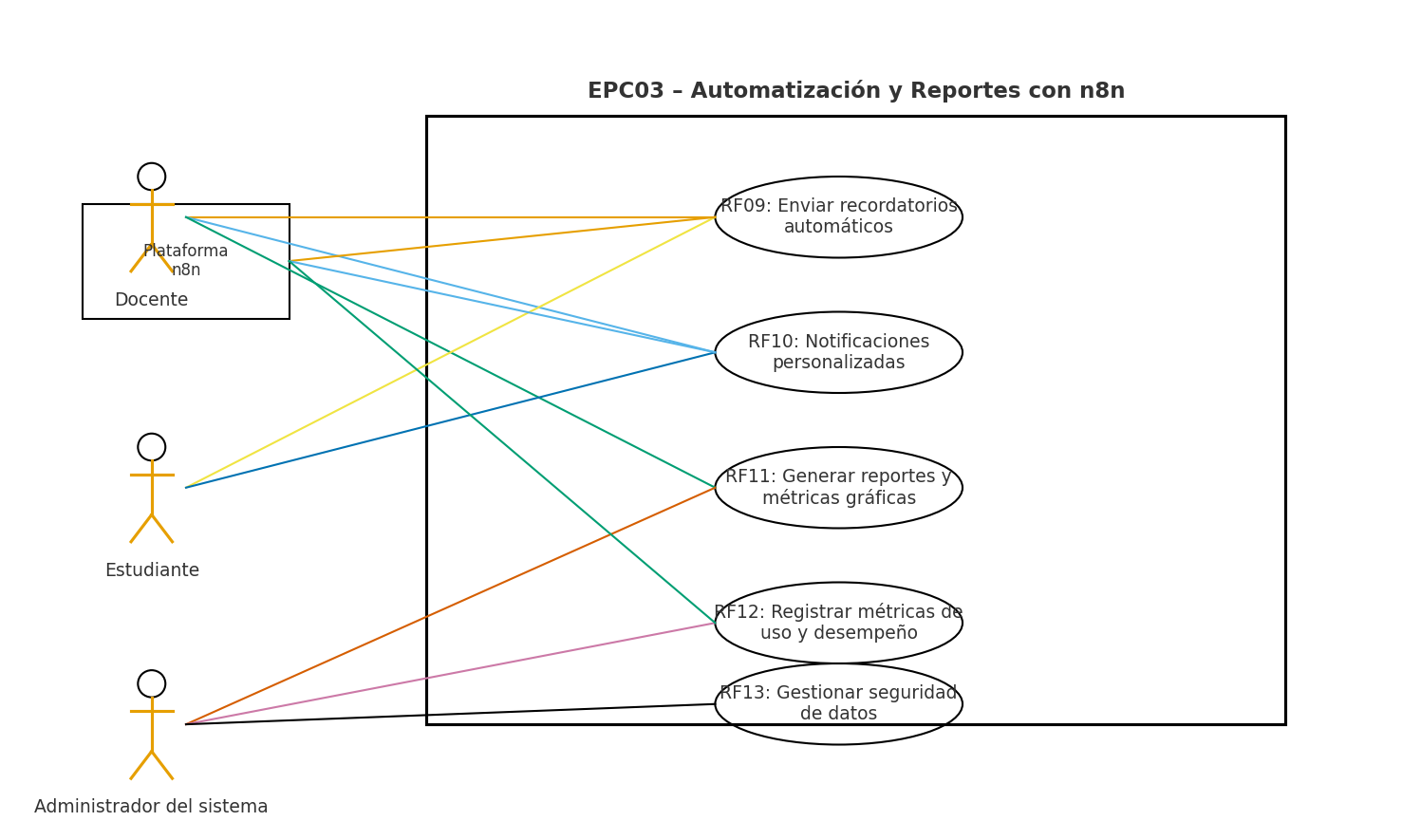


Ilustración EPC03 - Automatización y Reportes con n8n



### Diagramas de secuencia

Ilustración Registro e inicio de sesión

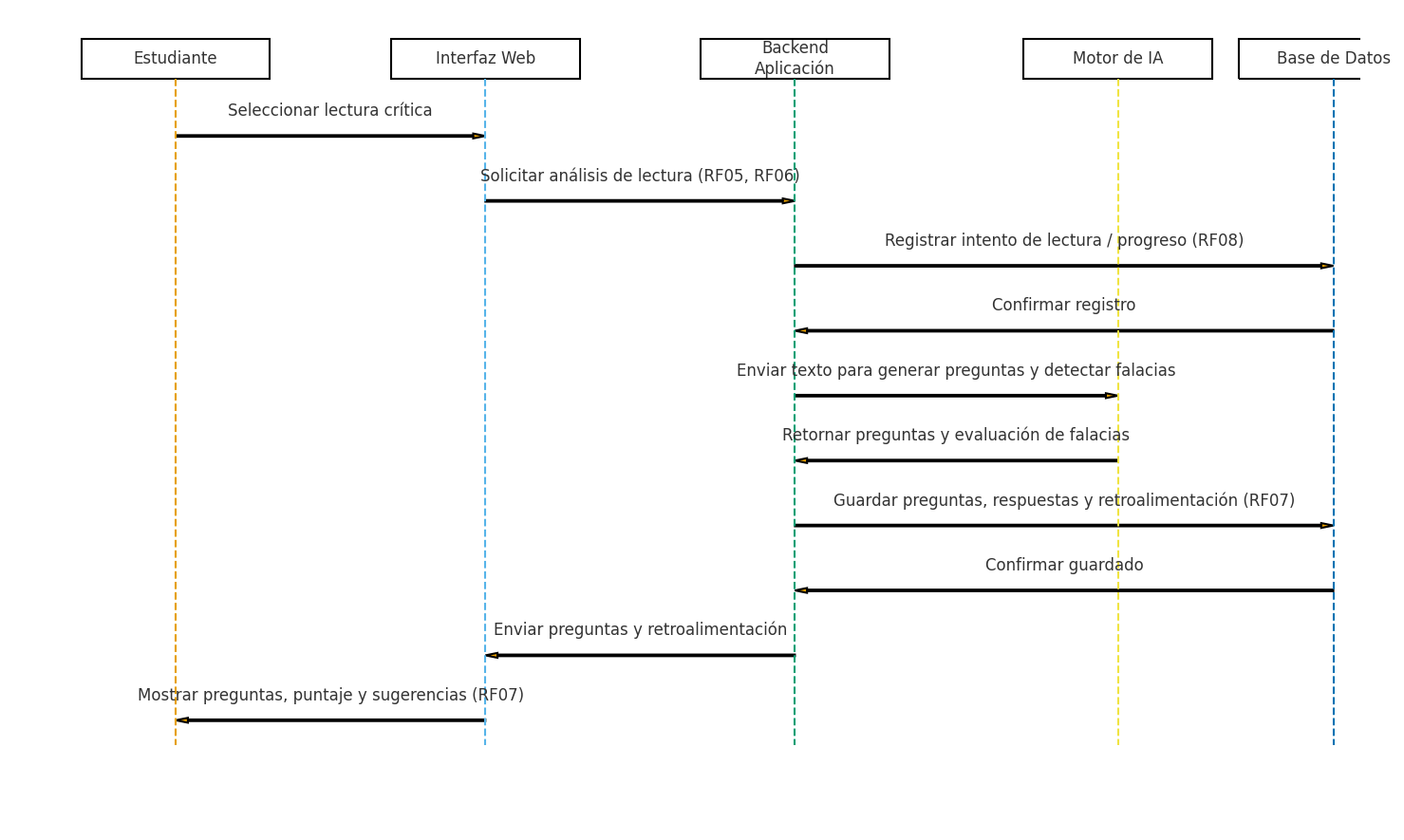


Ilustración Análisis de lectura con IA

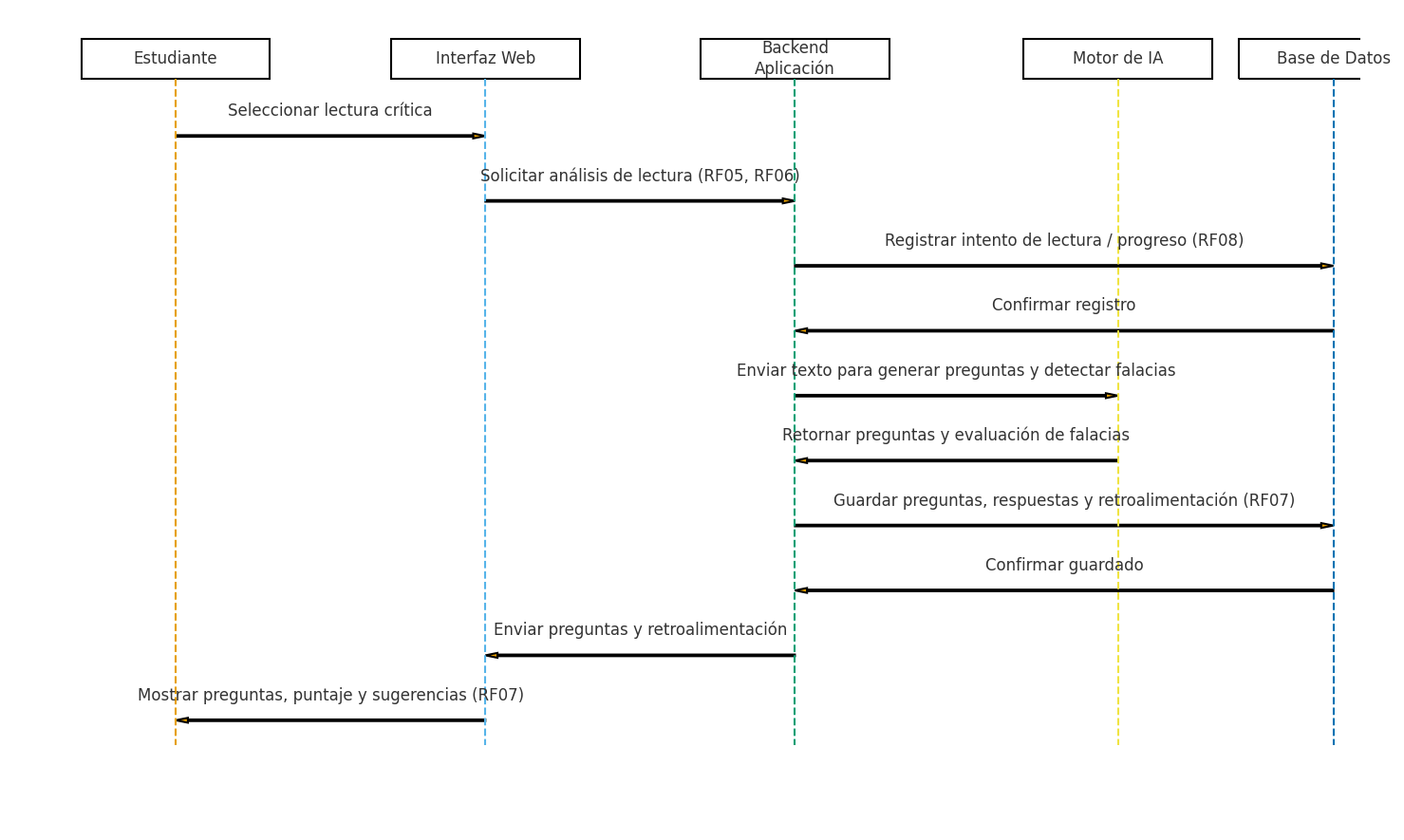
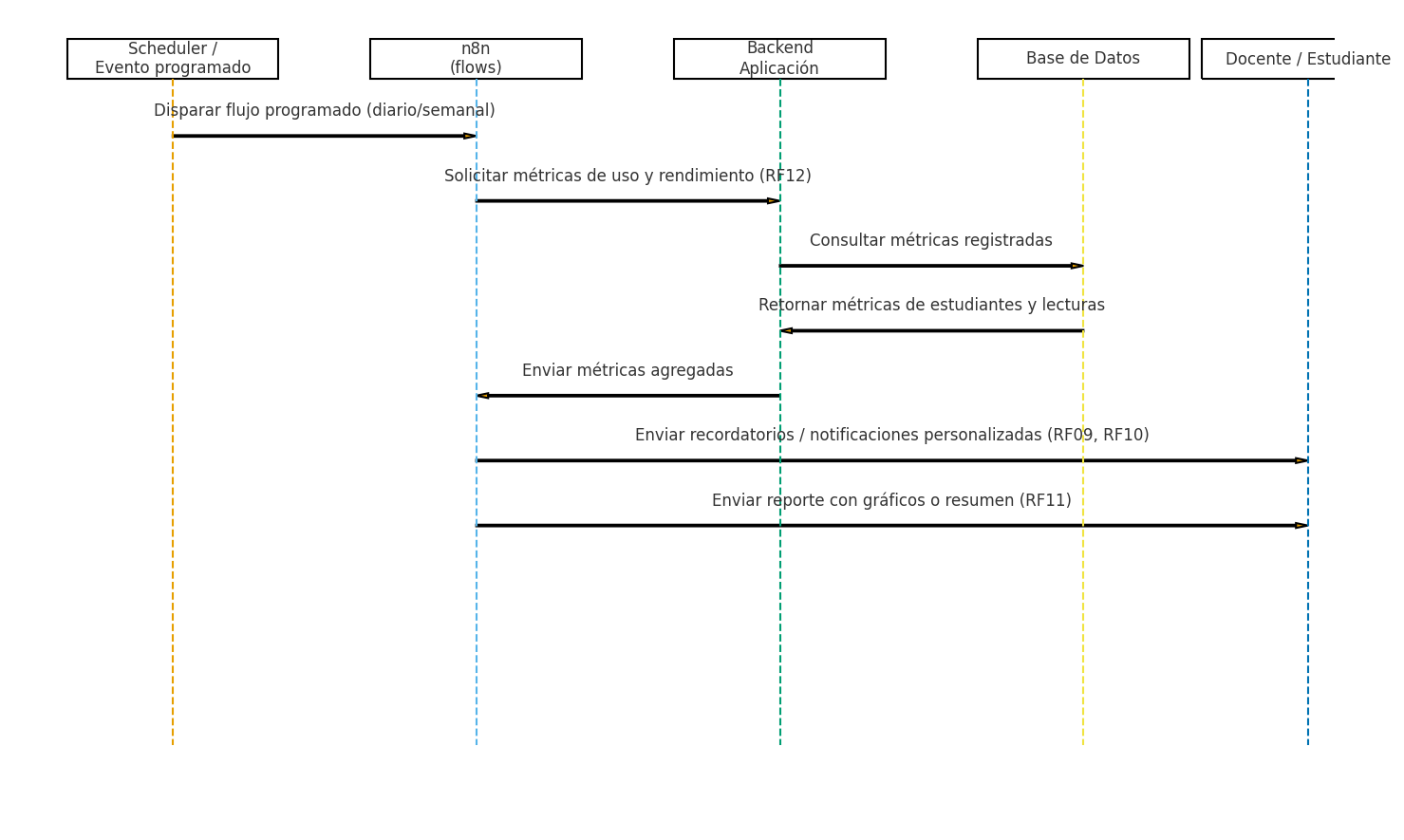


Ilustración Automatización con n8n



### Diagramas de colaboración

Ilustración Proceso de Login

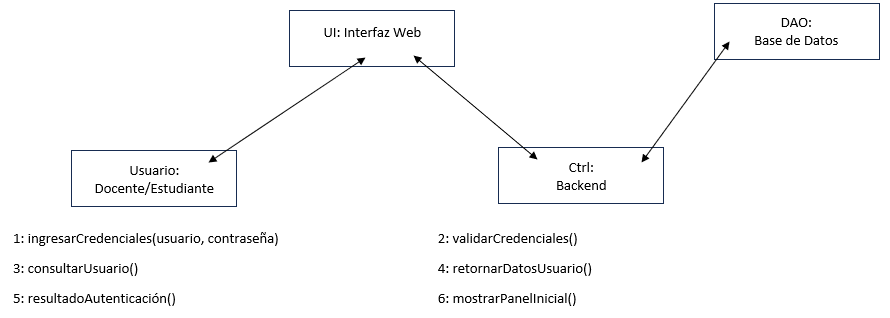


Ilustración Análisis de Lectura con IA

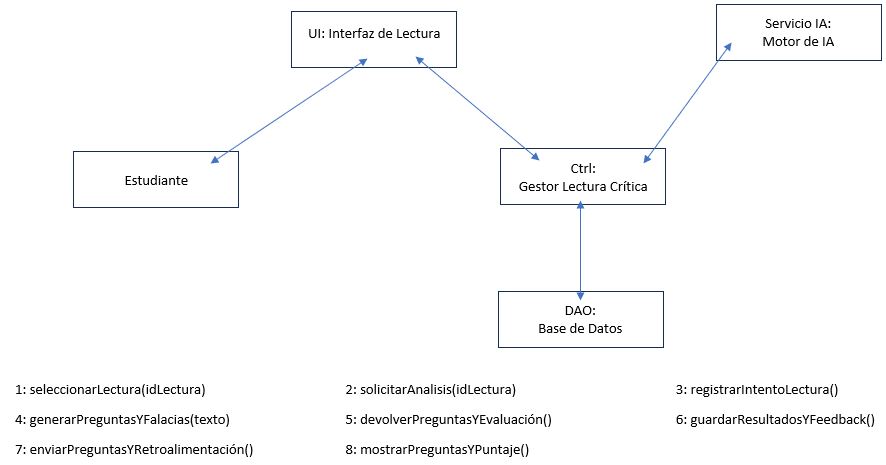
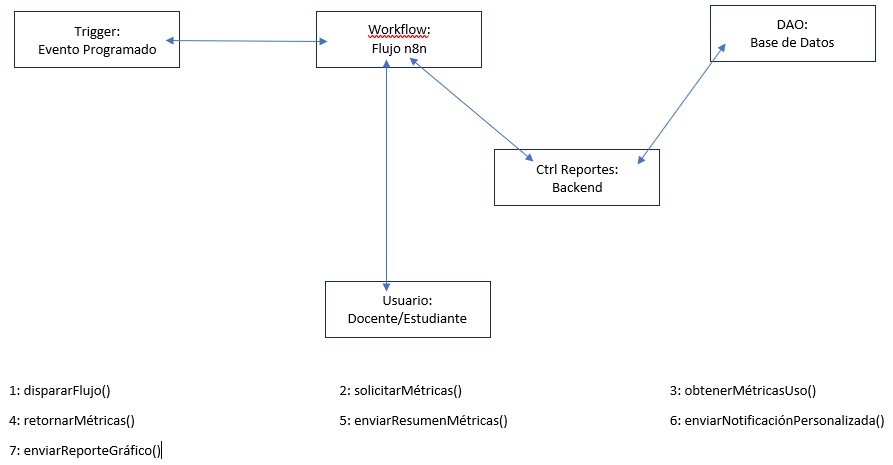
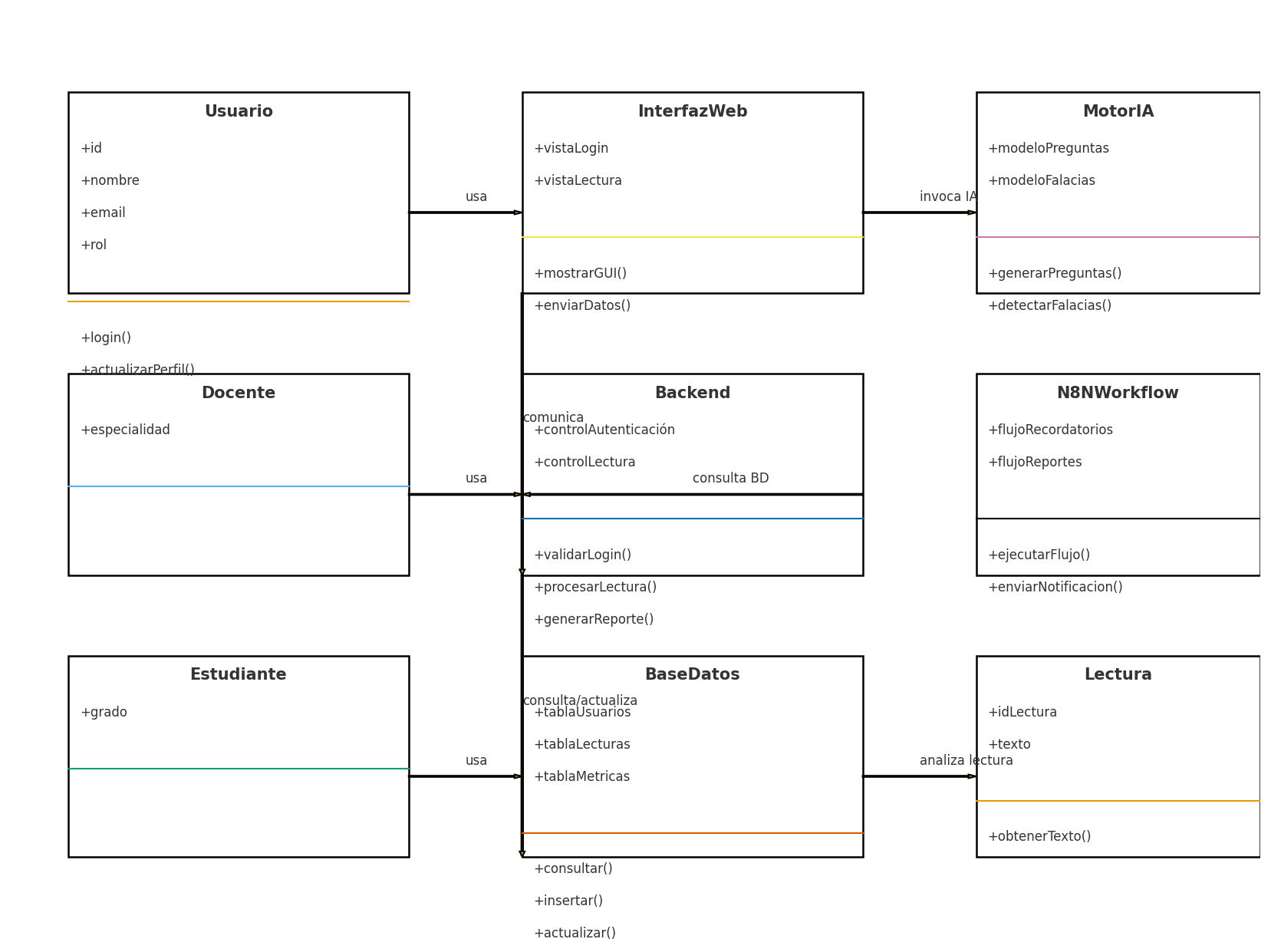


Ilustración Automatización con n8n



### Diagramas de clases

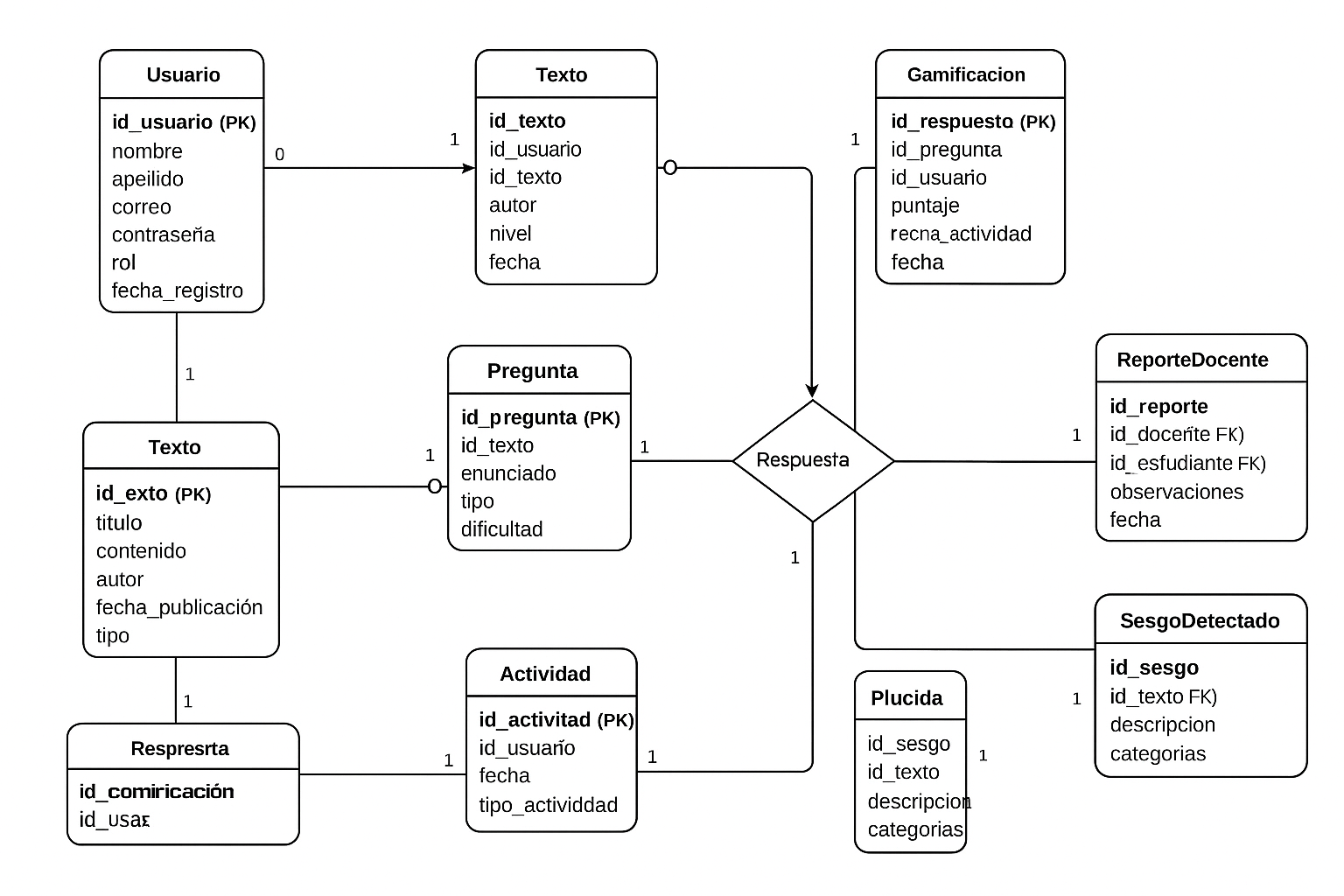
Ilustración Diagramas de clases



## Diseño de Base de Datos

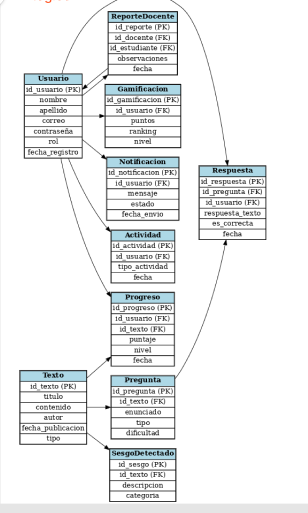
### Diseño conceptual (E/R)

Ilustración Diseño conceptual (E/R)



### Diseño lógico

Ilustración Diseño lógico



### Diseño físico

Ilustración Diseño fisico



### Modelado de base de datos

Este modelo corresponde a un **sistema educativo inteligente** que combina:

### Gestión de contenido

* Textos, preguntas, respuestas
* Detección de sesgos en textos

### Seguimiento del aprendizaje

* Progreso por texto
* Actividades por usuario
* Respuestas correctas e incorrectas

### Motivación y gamificación

* Puntos, nivel, ranking

### Comunicación

* Notificaciones
* Reportes generados por docentes hacia estudiantes

### Roles definidos en el sistema

* Docente (genera reportes)
* Estudiante (realiza actividades, responde preguntas)
* Administrador (puede existir según el rol)

Con este modelo se garantiza:

Trazabilidad completa del aprendizaje  
Identificación de sesgos en textos  
 Gestión eficiente entre estudiantes y docentes  
 Seguimiento gamificado del rendimiento

## Diseño de Interfaces Básicas

### Acceso login



### Interfaz principal



# CAPÍTULO 6

# CODIFICACIÓN DEL SOFTWARE

## 6.1 Desarrollo del Sprint 1

## 6.1.1 Sprint planning

En el Sprint 1 se organizaron las funciones esenciales que conforman el primer MVP del Tutor Virtual de Lectura Crítica. Para esta iteración se seleccionaron cuatro historias de usuario, orientadas a habilitar el ingreso al sistema, la edición del perfil docente y la visualización preliminar de las lecturas asignadas.

Durante este periodo se construyó la base del proyecto, habilitando el registro, el inicio de sesión y las vistas iniciales tanto del panel docente como del panel de lecturas. Paralelamente se realizaron pruebas de funcionamiento inicial, asegurando que la estructura general del sistema estuviera lista para incorporar módulos más complejos.

El sistema fue desarrollado con una arquitectura basada en Node.js, Express, React y MongoDB, empleando GitHub para el control de versiones y Jira para la gestión del proyecto.

### 6.1.2 Sprint backlog

Tabla Sprint backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Nº** | **Historia de Usuario** | **Tarea Técnica** | **Responsable** |
| **HU1** | **Registro e inicio de sesión** | **Implementación del sistema de autenticación (backend y frontend)** | **—** |
| **HU3** | **Pruebas básicas del sistema** | **Ejecución de pruebas iniciales de funcionamiento** | **—** |
| **HU2** | **Perfil docente** | **Diseño y desarrollo de la interfaz del perfil docente** | **—** |
| **HU4** | **Panel de lecturas asignadas** | **Construcción del panel principal y carga de datos** | **—** |

**6.1.3. Historias de usuario**

### HU1 – Registro e inicio de sesión

**Criterios de aceptación:**

* El usuario puede registrarse con correo válido y contraseña segura.
* El sistema valida datos obligatorios (no permite campos vacíos).
* El login funciona y redirige al dashboard correcto según rol (estudiante/docente).

### HU3 – Pruebas básicas del sistema

**Criterios de aceptación:**

* Deben implementarse pruebas unitarias para login y registro.
* El 80% de los casos de prueba deben pasar exitosamente.
* Los errores deben registrarse en un log para revisión

### HU2 – Perfil docente

**Criterios de aceptación:**

* La interfaz principal se carga en menos de 3 segundos.
* Menús y botones tienen nombres claros y consistentes.

### HU4 – Panel de lecturas

**Criterios de aceptación:**

* El estudiante visualiza una lista con al menos 1 lectura asignada.
* Las lecturas se muestran con título, fecha y estado (pendiente/completada).
* El panel se actualiza automáticamente tras completar una lectura.

### 6.1.4. Taskboard

El seguimiento del avance se realizó a través de Jira, donde se asignó cada historia de usuario a un responsable y se monitorea su progreso de manera constante. Durante la ejecución del sprint, todas las actividades registradas avanzaron adecuadamente y alcanzaron el estado de completadas dentro del periodo planificado.

### 6.1.5. Daily scrum

La coordinación diaria del equipo se realizó utilizando:

* Reuniones breves presenciales.
* Comunicación vía WhatsApp..
* Revisión constante de commits en GitHub.

Esto permitió mantener sincronía entre las tareas del backend, frontend y las pruebas.

### 6.1.6 Sprint review

Al cierre del sprint, el equipo presentó:

* Un sistema de registro e inicio de sesión operativo.
* La vista funcional del perfil docente.
* El panel preliminar de lecturas está cargando datos correctamente.
* Navegación estable y sin errores visibles.

No se modificaron los alcances definidos.

### 6.1.7. Resultados del sprint

#### 6.1.7.1. Evidencias.

En Jira se visualizan las historias en estado **Completado**, con una duración aproximada de 8 días laborables.  
 En la aplicación web se muestran:

\* Registro funcional.  
\* Inicio de sesión con redirección al dashboard.  
\* Perfil docente editable.  
\* Lecturas disponibles desde la interfaz.



Ilustración 15 . Componentes del sprint 1 en Jira

Resultados del sprint 1 en la aplicación web

En la aplicación web se puede visualizar el desarrollo del sprint 1, donde hay actividades de prueba, con sus respectivos estados, se puede observar las personas responsables, el informante y la prioridad que tuvo.

#### 6.1.7.2. Prueba de desarrollo.

Las pruebas aplicadas fueron principalmente manuales bajo el enfoque de caja negra. Esto permitió verificar la funcionalidad de los formularios, validar el uso correcto de credenciales durante el inicio de sesión y confirmar que los datos visualizados en la interfaz se recuperaban y mostraban correctamente. No se detectaron fallos críticos durante estas pruebas, lo que evidenció una correcta implementación de las funcionalidades básicas.

No se detectaron fallos importantes.

### 6.1.8. Sprint retrospective

Durante la retrospectiva del sprint, el equipo identificó como fortaleza la comunicación constante y la correcta construcción de la base del sistema. También se reconoció que la fase de configuración tomó más tiempo del previsto, lo que generó la necesidad de planificar mejor la modularización en futuros sprints. Entre las mejoras propuestas se destacó la importancia de estandarizar el uso de Git para evitar conflictos en versiones posteriores del código.

## 6.2 Desarrollo del Sprint 2

### 6.2.1 Sprint planning

El segundo sprint se enfocó en el desarrollo del MVP 2, orientado completamente a la integración de herramientas de Inteligencia Artificial dentro de la plataforma. Para este ciclo se seleccionaron cuatro historias de usuario centradas en la generación automática de preguntas, la detección de sesgos y falacias, la creación de retroalimentación automática y la habilitación del panel de desempeño docente. La planificación contempló la configuración del motor de IA y la adecuación de la lógica necesaria para procesar textos de manera eficiente dentro del sistema.

### 6.2.2 Sprint backlog

Tabla Sprint backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Nº** | **Historia de Usuario** | **Tarea Técnica** | **Responsable** |
| **HU5** | Generación de preguntas de análisis | Implementar módulo de IA que genere mínimo 5 preguntas por texto; validar relevancia del 70%; mostrar preguntas en formato de opción múltiple. | — |
| **HU6** | Detección de sesgos y falacias | Crear modelo de análisis que identifique mínimo 3 sesgos/falacias; mostrar resultados en listado; asegurar funcionamiento en 80% de textos. | — |
| **HU7** | Retroalimentación automática | Desarrollar sistema de puntaje inmediato; generar explicaciones cortas (<30 palabras); optimizar respuesta para mostrarse en menos de 2 segundos. | — |
| **HU8** | Panel docente de desempeño | Construir panel con gráficos de puntajes; incluir promedio y resultados individuales; agregar filtros por fecha o lectura. | — |

### 6.2.3 Historias de usuarios

**HU5: Generación de preguntas de análisis**

**Criterios de aceptación:**

* La IA genera mínimo 5 preguntas por cada texto.
* El 70% de las preguntas es relevante al contenido del texto.
* Las preguntas se muestran en el formato de opción múltiple.

**HU6: Detección de sesgos y falacias**

**Criterios de aceptación:**

* El sistema identifica al menos 3 tipos de sesgos/falacias en un texto de prueba.
* Los resultados se muestran en un listado claro para el docente.
* La detección funciona en al menos 80% de los textos cargados.

**HU7: Retroalimentación automática**

**Criterios de aceptación:**

* El estudiante recibe un puntaje inmediato tras enviar sus respuestas.
* Cada respuesta incorrecta muestra una breve explicación (<30 palabras).
* La retroalimentación aparece en menos de 2 segundos después de enviar.

**HU8: Panel docente de desempeño**

**Criterios de aceptación:**

* El docente visualiza un gráfico con los puntajes de cada estudiante.
* El panel muestra promedio de la clase y resultados individuales.
* Se puede filtrar el rendimiento por fecha o por lectura.

### 6.2.4 Taskboard

En Jira se observó el avance progresivo de las historias de usuario desde la fase "En curso" hasta su culminación. A lo largo del sprint, cada tarea avanzó de acuerdo con la planificación y llegó a completarse satisfactoriamente antes del cierre de esta iteración.

### 6.2.5 Daily scrum

Durante las reuniones diarias se trataron temas como el ajuste de prompts de la IA, la validación de la pertinencia de las preguntas generadas y la confirmación de la precisión en la detección de sesgos y falacias. También se trabajó en el diseño de las gráficas del panel docente y se definieron ajustes necesarios para mejorar la coherencia entre los módulos del sistema.

### 6.2.6 Sprint review

En la revisión del sprint, el equipo presentó un conjunto completo de funcionalidades, entre ellas la generación de preguntas basadas en lecturas cargadas, la detección automática de sesgos y falacias, la creación de retroalimentación inmediata para estudiantes y un panel docente con métricas funcionales. Todas las historias de usuario fueron entregadas conforme a la planificación inicial.

### 6.2.7 Resultados del sprint

Entre las evidencias del sprint se encontraron preguntas generadas correctamente mediante IA, reportes de sesgos funcionando de manera estable y un panel docente operativo con diversas métricas. El sistema mostró tiempos de respuesta adecuados y se constató que los módulos se integraron correctamente.

#### 6.2.7.1 Evidencias.

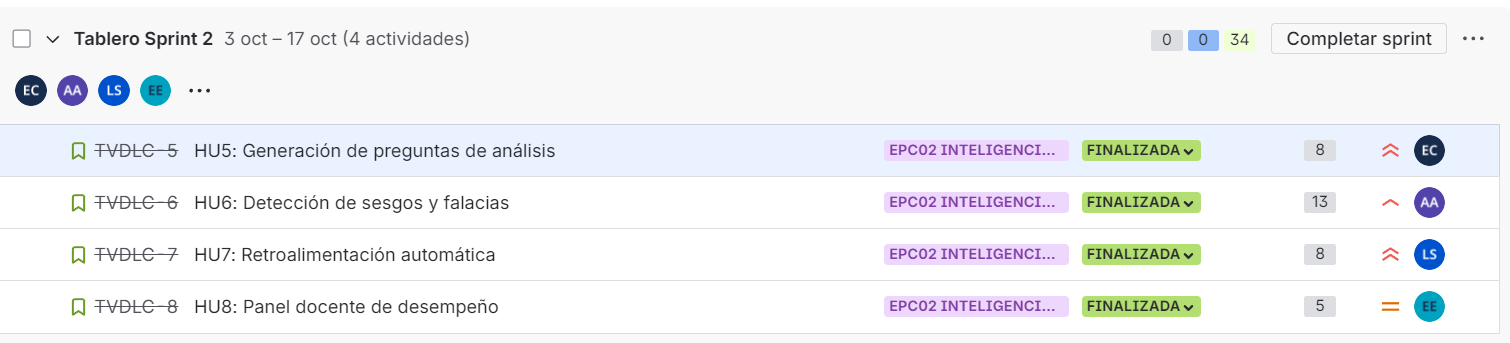


Ilustración 16 Componentes del sprint 2 en Jira

Como se puede observar se obtiene el desarrollo finalizado del sprint 2, que abarca la generación de preguntas, detección de sesgos y falacias, retroalimentación y el panel docente de desempeño, al lado derecho se puede observar los puntos y la preferencia del desarrollo junto a los asignados

#### 6.2.7.2 Prueba de desarrollo.

Se realizaron pruebas de funcionamiento sobre el flujo completo de lectura, análisis y retroalimentación, complementadas con pruebas unitarias simples. La aplicación mantuvo tiempos de respuesta apropiados, logrando un rendimiento estable incluso considerando los procesos automáticos de IA.

### 6.2.8 Sprint retrospective

En la retrospectiva se destacó como fortaleza la correcta integración con el motor de IA, mientras que como dificultad principal surgió la necesidad de ajustar los prompts de forma constante. El equipo propuso documentar los patrones más efectivos para evitar problemas en futuras iteraciones.

## 6.3 Desarrollo del Sprint 3

### 6.3.1 Sprint planning

El Sprint 3 se enfocó en consolidar la automatización de procesos internos mediante n8n, el fortalecimiento de la comunicación personalizada con los usuarios y la ampliación de las capacidades administrativas del sistema. Para esta iteración se trabajó con un conjunto más amplio de historias de usuario, orientadas tanto a la inteligencia operativa de la plataforma como a la seguridad, el rendimiento y la gestión integral de usuarios. Este sprint representó un paso importante porque permitió cerrar el ciclo funcional del sistema, integrando aspectos críticos como recordatorios automáticos, métricas de actividad, accesibilidad y manejo de cuentas, asegurando que la aplicación pueda sostener un funcionamiento estable y seguro conforme comienza a escalar.

Durante la planificación se definieron las dependencias entre módulos, especialmente aquellas vinculadas a la automatización de tareas con n8n, la comunicación interna y la administración de usuarios. También se establecieron ajustes necesarios en la arquitectura para soportar procesos más intensivos, lo que incluyó la revisión del rendimiento general y la implementación de criterios de seguridad más robustos.

### 6.3.2 Sprint backlog

Tabla 9 2 Sprint backlog-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Nº** | **Historia de Usuario** | **Tarea Técnica** | **Responsable** |
| **HU9** | Recordatorios automáticos (n8n) | Integrar n8n para automatizar recordatorios (por ejemplo, correos o mensajes) | — |
| **HU10** | Notificaciones personalizadas | Desarrollar sistema de notificaciones para usuarios con opciones personalizadas | — |
| **HU11** | Reportes detallados | Generar reportes con datos de actividad, rendimiento y métricas | — |
| **HU12** | Registro automático de métricas | Implementar tracking automático de métricas (uso, tiempo, respuestas, etc.) | — |
| **HU13** | Seguridad de datos | Aplicar medidas de cifrado, control de acceso, y protección de datos | — |
| **HU14** | Rendimiento del sistema | Optimizar tiempos de respuesta, manejar carga y escalabilidad | — |
| **HU15** | Usabilidad accesible | Implementar mejoras de accesibilidad (WCAG, navegación, contraste, etc.) | — |
| **HU16** | Creación de usuarios | Añadir funcionalidad para que el admin cree nuevos usuarios | — |
| **HU17** | Modificación de usuarios | Permitir que el admin edite datos de otros usuarios (nombre, rol, etc.) | — |
| **HU18** | Eliminación de usuarios | Habilitar eliminación segura de cuentas con confirmación y respaldo | — |

### 6.3.3 Historias de usuarios

* **HU9: Recordatorios automáticos (n8n)**

El sistema debe enviar recordatorios basados en fechas y plazos configurados previamente.  
Los flujos en n8n deben ejecutarse de manera automática sin intervención manual.

Los recordatorios deben llegar al usuario dentro del tiempo programado.

Debe registrarse internamente cada recordatorio enviado para fines de auditoría.

* **HU10: Notificaciones personalizadas**

Las notificaciones deben activarse únicamente cuando ocurra un evento relevante definido por el sistema.

El contenido del mensaje debe adaptarse a la actividad o condición del usuario.

La notificación debe mostrarse de manera clara en la interfaz correspondiente.

El usuario debe recibir las notificaciones sin retrasos perceptibles.

* **HU11: Reportes detallados**

El sistema debe generar reportes completos con datos actualizados.

Los reportes deben incluir tablas, métricas y visualizaciones gráficas.

Los reportes deben permitir exportación o descarga.

Toda la información del reporte debe coincidir con los registros almacenados en la base de datos.

* **HU12: Registro automático de métricas**

El sistema debe almacenar métricas sin necesidad de acciones del usuario.

Las métricas deben registrarse en tiempo real o en intervalos definidos.

Cada métrica almacenada debe asociarse al usuario correspondiente.

Debe existir consistencia entre los datos mostrados en la interfaz y los que están en la base de datos.

* **HU13: Seguridad de datos**

Todos los datos sensibles deben manejarse con cifrado o protección equivalente.

El sistema debe validar permisos antes de permitir el acceso a información restringida.

Las operaciones críticas deben registrarse para seguimiento y auditoría.  
No deben existir filtraciones de datos ni accesos no autorizados durante las pruebas.

* **HU14: Rendimiento del sistema**

El sistema debe cargar de manera fluida, manteniendo tiempos de respuesta óptimos.

Las consultas a la base de datos deben estar optimizadas para reducir demoras.

Las vistas con mayor carga deben renderizarse sin bloqueos ni interrupciones.

La plataforma debe soportar múltiples usuarios sin afectar la estabilidad.

* **HU15: Usabilidad Accesible**

La interfaz debe cumplir criterios básicos de accesibilidad como contraste, tamaño de letra y navegación fluida.

El sistema debe permitir navegación mediante teclado.

Los elementos importantes deben ser compatibles con lectores de pantalla.  
La experiencia del usuario no debe verse afectada por limitaciones visuales o motrices.

* **HU16: Creación de usuarios**

La plataforma debe permitir registrar nuevos usuarios con todos los campos obligatorios.

Deben aplicarse validaciones para evitar datos incompletos o inválidos.

El sistema debe mostrar un mensaje de confirmación cuando el registro sea exitoso.  
El usuario creado debe aparecer inmediatamente en la lista de usuarios administrables.

* **HU17: Modificación de usuarios**

Debe ser posible editar los datos de un usuario ya registrado sin afectar su cuenta.

Los cambios deben guardarse correctamente en la base de datos.

El sistema debe mostrar un mensaje claro después de la actualización.  
La información editada debe visualizarse de manera inmediata en la interfaz.

* **HU18: Eliminación de usuarios**

El sistema debe permitir eliminar usuarios de manera segura y definitiva.

Antes de eliminar, debe solicitarse una confirmación para evitar errores.

El usuario eliminado ya no debe aparecer en los listados ni acceder al sistema.  
La operación debe registrarse para control interno o auditoría.

### 6.3.4 Taskboard

El avance del sprint fue monitoreado desde Jira, donde cada historia de usuario se movió progresivamente desde las columnas “Por hacer”, “En progreso” y “Finalizado”. Este tablero permitió visualizar en tiempo real la carga de trabajo, los responsables asignados y los avances logrados. En general, el flujo del sprint se mantuvo estable, aunque algunos módulos relacionados con seguridad y rendimiento requirieron más revisiones de lo previsto.

### 6.3.5 Daily scrum

Durante las reuniones diarias se revisaron los flujos creados en n8n, validando que los recordatorios se ejecutaran en los tiempos programados. También se supervisó la activación correcta de las notificaciones personalizadas y se evaluó la precisión y completitud de los reportes generados. Se monitoreó el registro automático de métricas para confirmar que la base de datos almacenara toda la información sin fallos. Además, se dedicó un espacio importante a revisar controles de seguridad, optimizaciones de rendimiento y ajustes de accesibilidad, especialmente en contraste visual, compatibilidad con lector de pantalla y navegación por teclado. Finalmente, se revisó el funcionamiento del módulo de gestión de usuarios, asegurando que las funciones de crear, editar y eliminar perfiles se ejecutarán correctamente.

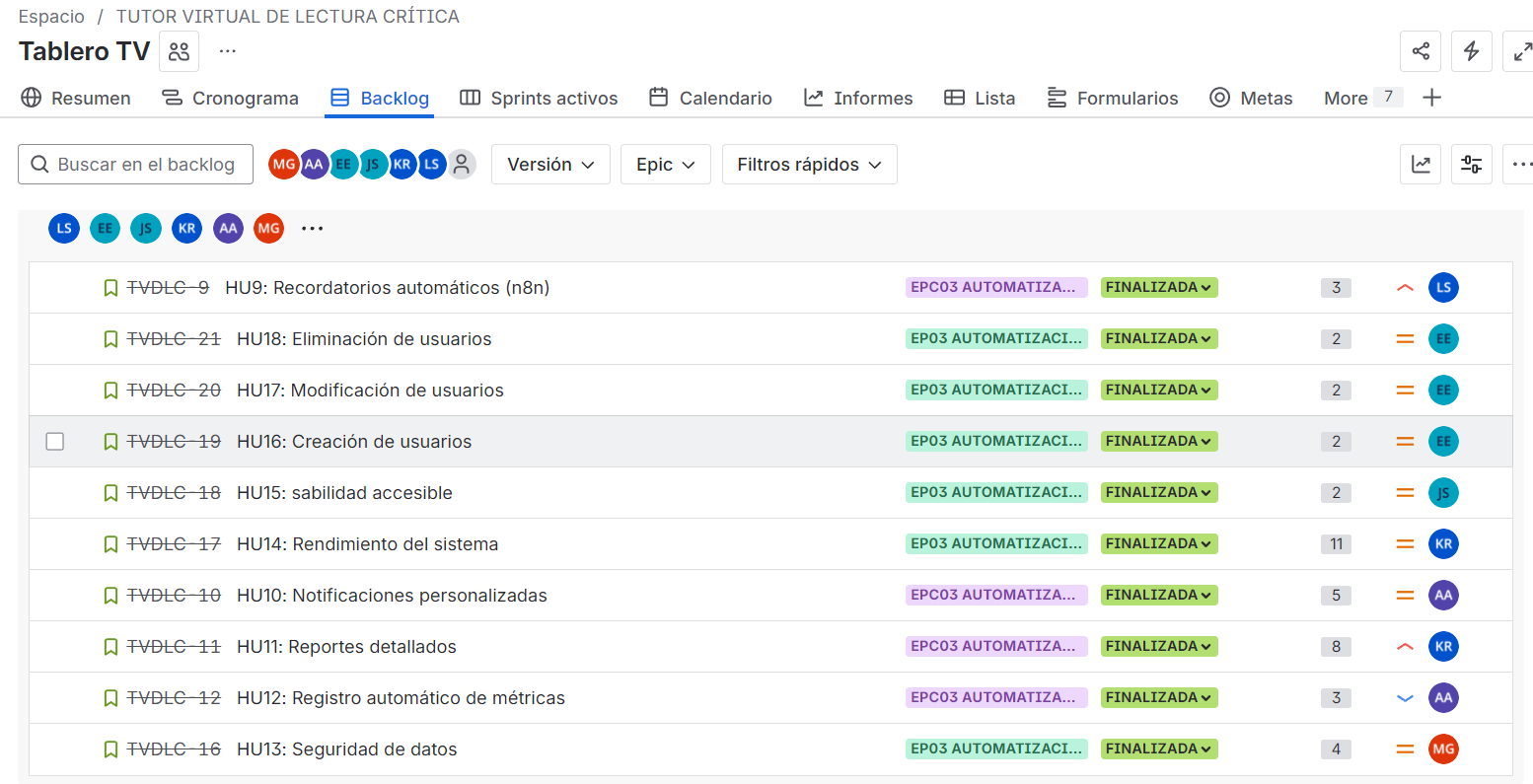
### 6.3.6 Sprint review

En la revisión del sprint, el equipo presentó una versión operativa del sistema que incluía recordatorios automáticos, notificaciones generadas según el comportamiento del usuario, reportes completos y descargables, y un registro continuo de métricas internas. También se mostraron las mejoras implementadas en cuanto a seguridad, rendimiento y accesibilidad, evidenciando una experiencia más fluida y segura. El módulo de gestión de usuarios también fue demostrado, confirmando que ya era posible crear nuevas cuentas, actualizarlas y eliminarlas sin afectar la estabilidad del sistema.

### 6.3.7 Resultados del sprint

#### 6.3.7.1 Evidencias.

Ilustración Componentes del sprint 3 en Jira



Como se puede observar en la imagen, las historias de usuario del sprint 3 están finalizadas, con los responsables asignados y preferencia de cada HU, además mostró resultados favorables, demostrando flujos automatizados estables, notificaciones visibles, reportes completos y un registro constante de métricas. También se evidenciaron mejoras significativas en seguridad, rendimiento y accesibilidad, elementos clave para la etapa de despliegue. El módulo de administración de usuarios quedó completamente funcional, lo que permitirá un control adecuado de los perfiles dentro de la plataforma.

**6.3.7.2 Prueba de desarrollo.**

Las pruebas realizadas incluyeron validaciones unitarias sobre cada módulo, pruebas de integración entre n8n, backend y frontend, test de seguridad, simulaciones de carga para evaluar rendimiento y revisiones manuales orientadas a garantizar que la interfaz fuera accesible para diferentes tipos de usuarios. Las pruebas relacionadas con la gestión de usuarios siguieron un enfoque de caja negra para verificar la coherencia de cada operación.

#### 

### 6.3.8 Sprint retrospective

Durante la retrospectiva se reconoció que el sprint alcanzó resultados sólidos, especialmente en automatizaciones y seguridad. Sin embargo, también se identificó que la integración con n8n demandó más tiempo del estimado debido a la complejidad de sus flujos. Se sugirió reforzar la documentación interna y establecer guías de procedimiento para futuras automatizaciones. Además, se concluyó que las mejoras en rendimiento y accesibilidad aportaron una experiencia más fluida, por lo que deben mantenerse como prioridades en los próximos sprints.

# CAPÍTULO 7

# PRUEBAS DE SOFTWARE

## Plan de Pruebas

Para garantizar que las funcionalidades del **Tutor Virtual de Lectura Crítica** se mantengan estables a medida que se incorporan nuevas características, se implementó un flujo de **integración continua (CI)** utilizando GitHub Actions. Este flujo se ejecuta automáticamente cada vez que se realiza un *push* o se abre un *pull request* hacia el repositorio del proyecto, y tiene como objetivo principal ejecutar las pruebas automatizadas del frontend y del backend antes de que los cambios se integren en la rama principal.

El pipeline define dos trabajos principales: uno para el **frontend** (aplicación React) y otro para el **backend** (API en Node.js/Express). En ambos casos se sigue una secuencia similar:

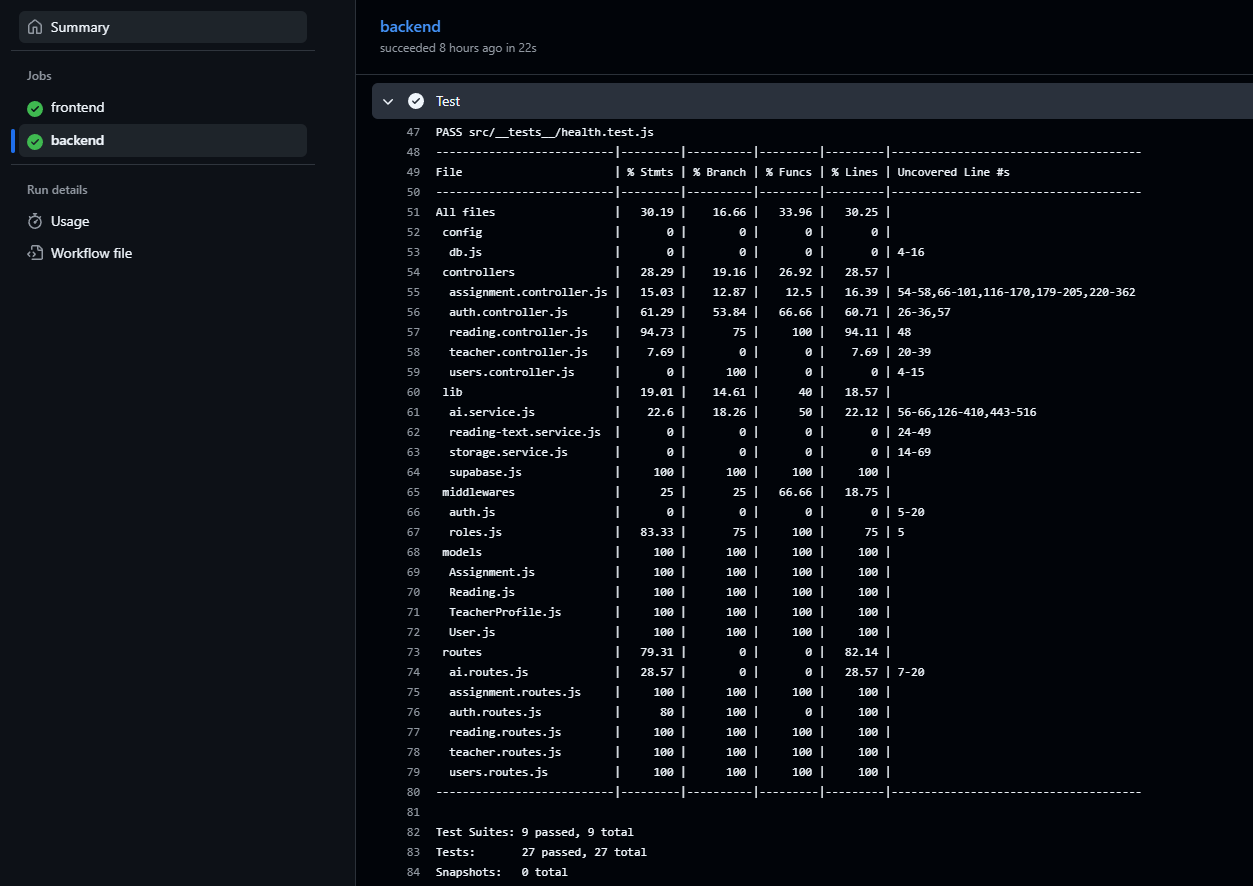
1. **Descarga del código fuente** desde el repositorio.
2. **Configuración del entorno de Node.js** con la versión requerida por el proyecto.
3. **Instalación de dependencias** mediante npm install.
4. **Ejecución de las pruebas automatizadas** con el comando npm test (incluyendo el reporte de cobertura).
5. Publicación del resultado de la ejecución (éxito o fallo) directamente en la interfaz del repositorio.

De esta forma, cada cambio en el código es validado automáticamente. Si alguna prueba falla, el pipeline marca el build como incorrecto y evita que los cambios se mezclen sin revisión, lo que reduce la probabilidad de introducir errores en producción y facilita la detección temprana de regresiones.

A continuación se presenta el archivo de configuración del workflow utilizado para la ejecución automática de pruebas. Este archivo se encuentra ubicado en la ruta .github/workflows/ del repositorio y define el comportamiento del pipeline descrito:

De esta forma, cada cambio en el código es validado automáticamente. Si alguna prueba falla, el pipeline marca el *build* como incorrecto y evita que los cambios se mezclen sin revisión, lo que reduce la probabilidad de introducir errores en producción y facilita la detección temprana de regresiones.

A continuación se presenta el archivo de configuración del workflow utilizado para la ejecución automática de pruebas. Este archivo se encuentra ubicado en la ruta .github/workflows/ del repositorio y define el comportamiento del pipeline descrito:



# CONCLUSIONES

1. El proyecto *Tutor Virtual de Lectura Crítica* responde a una problemática real y vigente: la baja comprensión lectora y el limitado pensamiento crítico en estudiantes universitarios. La propuesta tecnológica se alinea a necesidades educativas actuales, facilitando un aprendizaje más activo, personalizado y automatizado.
2. El uso de tecnologías modernas como Node.js, React/Vue, MongoDB, IA y N8N demuestra que el sistema posee una factibilidad técnica sólida, ya que el equipo cuenta con los conocimientos necesarios y los recursos tecnológicos son accesibles y escalables.
3. La metodología Scrum aplicada en el desarrollo permite un trabajo organizado, iterativo y orientado al valor, lo cual asegura que las funcionalidades se construyan de forma progresiva y controlada a través de sprints bien definidos.
4. La plataforma presenta una arquitectura eficiente y sostenible, apoyada en servicios cloud gratuitos, herramientas open source y automatización de procesos; lo cual reduce significativamente los costos y facilita la implementación en entornos educativos.

# RECOMENDACIONES

1. Fortalecer la fase de pruebas (QA) con casos de prueba más amplios y pruebas de estrés para asegurar que los módulos de IA y la automatización con N8N mantengan su rendimiento bajo diferentes cargas de usuarios.
2. Implementar herramientas adicionales de monitoreo como Grafana, LogRocket o Google Analytics para evaluar el comportamiento de los usuarios y mejorar la toma de decisiones en futuras iteraciones del proyecto.
3. Ampliar la base de lecturas y los niveles de dificultad de las preguntas generadas por IA, permitiendo una experiencia más completa para estudiantes de distintos niveles académicos y áreas disciplinares.
4. Capacitar periódicamente a docentes y administradores sobre el uso de la plataforma, especialmente en la interpretación de métricas y reportes automatizados, para asegurar una correcta adopción del sistema.
5. Considerar en futuros sprints la integración con aplicaciones educativas existentes (como Google Classroom o Moodle) para facilitar la interoperabilidad y permitir que más instituciones incorporen la plataforma sin fricciones técnicas.

# ANEXOS

## Anexo 01. Manual Técnico

**Manual Técnico**

**Sistema Web:** Tutor Virtual de Lectura Crítica  
**Repositorio GitHub:** kenyiYR/TUTOR-VIRTUAL-DE-LECTURA-CR-TICA ([GitHub](https://github.com/kenyiYR/TUTOR-VIRTUAL-DE-LECTURA-CR-TICA))  
**Fecha:** 26 de noviembre de 2025  
**Versión:** 1.0  
**Autor:** Alexandra Campos   
**Contacto:** 75256374@continental.edu.pe

**Índice**

1. Introducción
2. Requisitos del Sistema
3. Arquitectura del Sistema
4. Tecnologías Utilizadas
5. Estructura del Código
6. Instalación y Configuración
7. Base de Datos
8. Seguridad
9. Pruebas
10. Mantenimiento y Actualizaciones
11. Solución de Problemas

**1. Introducción**

**1.1 Descripción General**

El **Tutor Virtual de Lectura Crítica** es una aplicación web que usa la pila MERN para facilitar lecturas, preguntas y ejercicios de pensamiento crítico. Permite que los estudiantes respondan preguntas, reciban retroalimentación y almacena su progreso. El sistema está diseñado para ser desplegado en un entorno en la nube.

**1.2 Alcance del Manual**

Este manual está destinado a desarrolladores y al equipo técnico encargado del mantenimiento y despliegue. Cubre cómo instalar, configurar, desplegar y resolver problemas del sistema.

**2. Requisitos del Sistema**

**Hardware recomendado:**

* Servidor: mínimo 2 núcleos, 4 GB de RAM, 10 GB libre
* Cliente: dispositivo con navegador moderno

**Software:**

* Node.js (versión compatible con el repo)
* MongoDB (puede ser Atlas)
* Git para clonar el repositorio

**Dependencias del proyecto:**  
React, Vite, Express, Mongoose.

**3. Arquitectura del Sistema**

La arquitectura es **MERN**

[Usuario (navegador)]

⇄ fetch / axios

[Frontend: React + Vite]

⇄ API REST

[Backend: Node.js + Express]

⇄ Mongoose

[MongoDB]

Es una estructura clásica cliente-servidor, con la base de datos gestionada por Mongoose en el backend.

**4. Tecnologías Utilizadas**

* **Frontend:** React + Vite
* **Backend:** Node.js + Express
* **Base de Datos:** MongoDB (Mongoose)
* **Comunicación HTTP:** REST con fetch o axios
* **Contenerización:** Uso de docker-compose.yml
* **Lint / Calidad:** .eslint.config.js

**5. Estructura del Código**

TUTOR-VIRTUAL-DE-LECTURA-CR-TICA/

├─ .github/ → configuraciones de workflows GitHub

├─ config/ → archivos de configuración (posiblemente DB, entorno)

├─ docs/ → documentación del proyecto

├─ documentos/ → (archivos de lecturas, recursos)

├─ mock/ → datos de simulación o pruebas

├─ public/ → recursos públicos (posiblemente para frontend)

├─ src/ → código fuente

│

├─ README.md → descripción principal del proyecto :

├─ docker-compose.yml → para ejecutar contenedores Docker :

├─ eslint.config.js

├─ index.html → página raíz (puede ser del frontend)

├─ package.json → dependencias, scripts, meta del proyecto

├─ package-lock.json → lock de dependencias :contentReference[oaicite:11]{index=11}

└─ vite.config.js → configuración de Vite para el proyecto frontend

**Scripts importantes**

* docker compose down
* docker compose up –build
* npm install
* npm run dev -- --host 0.0.0.0 --port 5173

**6. Instalación y Configuración**

Aquí los pasos para instalar y poner en marcha el proyecto

**Clonar el repositorio**

1. git clone https://github.com/kenyiYR/TUTOR-VIRTUAL-DE-LECTURA-CR-TICA.git
2. cd TUTOR-VIRTUAL-DE-LECTURA-CR-TICA
3. **Docker ( docker-compose.yml)**
   * Levantar los servicios:
   * docker compose up --build
   * Esto debe arrancar tu backend y la base de datos.
4. **modo local tradicional:**
   * Instalar dependencias:
   * npm install
   * Variables de entorno: crea un archivo .env en la raíz del proyecto si es necesario (o en la carpeta config/) con algo como:
   * MONGODB\_URI=tu\_uri\_de\_mongo
   * PORT=3000
   * JWT\_SECRET=tu\_secreto\_jwt
   * Ejecutar el proyecto:
   * npm run dev # o el script que definas en package.json

**7. Base de Datos**

**conexión:**

// backend/src/config/db.js

import mongoose from 'mongoose';

const uri = process.env.MONGODB\_URI;

mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('MongoDB conectada'))

.catch(err => console.error('Error MongoDB', err));

**Índices recomendados:** índice en User.email, Reading.titulo, y Question.readingId para acelerar consultas.

**8. Seguridad**

* Autenticación con **JWT**
* Es recomendable validar todas las entradas en las rutas (body, params, query) para evitar inyección.
* Usar HTTPS en producción.

**9. Pruebas**

**Estrategia recomendada**

* Unitarias (Jest) para controllers y utilidades.
* Integración API (supertest + Jest) para rutas.
* E2E (Cypress) para flujos: login → realizar ejercicio → recibir feedback.
* Postman: colección con endpoints para pruebas manuales.

**Comandos ejemplo**

# Backend

cd backend

npm test

# Frontend

cd frontend

npm test

**Flujo Git sugerido**

* ramas: main, develop, feature/\*, hotfix/\*
* Pull Requests para revisión
* CI: ejecutar tests antes de merge (GitHub Actions u otra)

**Otras recomendaciones**

* Revisar dependencias (npm audit)
* Monitoreo: logs (winston), métricas (Prometheus/Datadog opcional)
* Backups regulares de MongoDB (Atlas snapshots)

**11. Solución de Problemas Comunes**

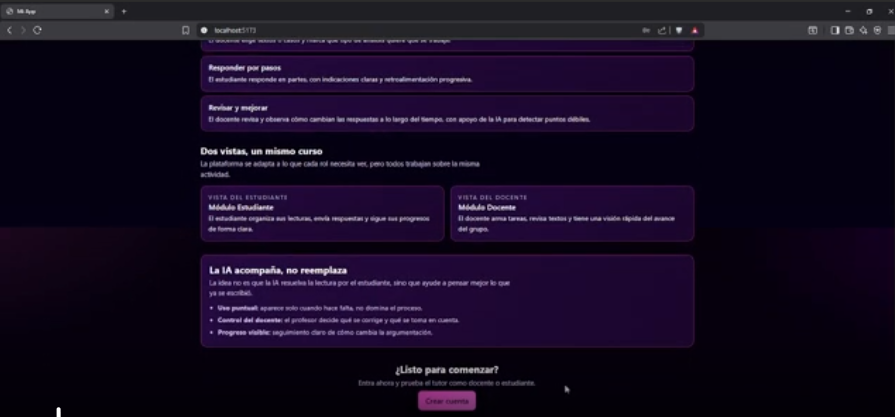
* **No conecta frontend → backend**: verificar VITE\_API\_URL y CORS en backend.
* **401 Unauthorized**: token ausente/expirado; revisar Authorization header Bearer <token>.
* **Error de conexión a DB**: comprobar MONGODB\_URI, credenciales y reglas de red en Atlas.
* **Errores al npm install**: versión de Node incompatible → usar nvm para cambiar versión.

## Anexo 02. Manual de Usuario

**GUIA DE USUARIO DE LA PLATAFORMA LECTURA CRITICA**

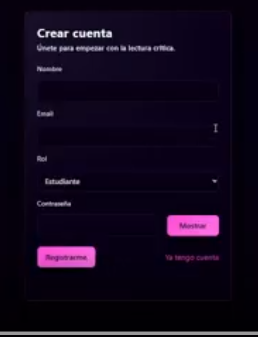
**1. Creación de la Cuenta**

En esta pantalla el usuario debe ingresar sus datos personales para poder crear una cuenta en el sistema. Para iniciar el proceso, debe presionar sobre el botón **“CREAR CUENTA”**, tal como se muestra en la figura siguiente. Al hacer clic, el sistema desplegará una ventana donde se solicitará información básica para la creación de la cuenta.



Dentro del formulario de registro, el usuario debe completar los siguientes campos:

* **Nombre completo**
* **Correo electrónico**
* **Rol**
* **Contraseña**

Los datos ingresados serán verificados por el sistema. Una vez completados todos los campos, el usuario debe presionar el botón **“REGISTRARME”**.

**Creación de la Cuenta con Rol Docente**

En esta pantalla, el usuario que desea registrarse como **Docente** debe presionar el botón **“Registrarse”**, tal como se muestra en la figura siguiente. Al seleccionar esta opción, el sistema abrirá una ventana donde podrá realizar el registro de su información personal.



**Perfil Docente**

Una vez creada la cuenta con rol Docente, el sistema mostrará la pantalla de **Perfil Docente**, donde el usuario debe completar su información profesional para habilitar los módulos de enseñanza. En esta ventana, como se observa en la figura, se presentan diversos campos que el docente debe llenar o actualizar.

En el formulario se deberá registrar la siguiente información:

* **Especialidad:**  
  El docente debe escribir el área profesional o disciplina en la que se desempeña.
* **Biografía:**  
  Espacio destinado para una breve descripción del perfil profesional, experiencia y habilidades del docente.
* **Cursos (separados por coma):**  
  Aquí se deben ingresar los cursos que el docente está apto para dictar.
* **Foto (URL):**  
  Campo para colocar el enlace de una fotografía profesional del docente.
* **LinkedIn:**  
  Enlace al perfil profesional del docente en esta red.
* **GitHub:**  
  En caso de tener proyectos o repositorios, el docente podrá colocar su enlace de GitHub.
* **Días disponibles:**  
  Sección donde el docente selecciona los días de la semana en los que puede dictar clases. Los botones incluyen:  
  **Lun, Mar, Mié, Jue, Vie, Sáb, Dom.**
* **Horario:**  
  Espacio para indicar el rango de horas en que el docente está disponible, por ejemplo **09:00–13:00**.

Una vez completados todos los campos, el usuario debe presionar el botón **“Guardar cambios”** para registrar su información en el sistema.



**Pantalla Principal del Docente**

Una vez que el docente completa su perfil, el sistema lo redirige automáticamente a la **Pantalla Principal**, donde podrá gestionar todas las actividades académicas. En esta ventana, como se muestra en la figura, se encuentra el módulo de **Asignación de lecturas**, que permite subir textos y organizarlos para los estudiantes de manera clara y centralizada.

En esta pantalla, el docente cuenta con las siguientes secciones y opciones:

Asignación de lecturas

En la parte superior se muestra el encabezado **“Asignación de lecturas”**, donde el sistema indica que aquí se pueden subir textos y asignarlos a los estudiantes. A la derecha se encuentra el botón **“Ver tablero”**, el cual permite visualizar las lecturas ya asignadas o en proceso.

Subir nueva lectura

El recuadro central muestra el formulario para subir un nuevo documento. Incluye los siguientes campos:

* **Título de la lectura:**  
  El docente debe ingresar el nombre del texto o material que desea asignar.
* **Seleccionar archivo:**  
  Permite elegir un documento en formato **PDF, DOCX o TXT** desde el dispositivo del docente.
* **Descripción (opcional):**  
  Espacio para agregar detalles como el curso, tema o tipo de análisis que realizarán los estudiantes.

Una vez completados los campos, el docente debe presionar el botón **“Subir lectura”** para registrar el material en el sistema.

Lecturas creadas:

En la parte inferior se muestra la lista de lecturas que el docente ha subido.  
Si aún no existe ningún archivo, aparece el mensaje:

**“No tienes lecturas registradas todavía.**  
**Sube tu primer texto arriba para empezar a trabajar con el tutor.”**

Esta sección se actualizará automáticamente cuando el docente agregue nuevas lecturas.



**Gestión de Lecturas Creadas**

Después de subir una lectura, el sistema muestra automáticamente la sección **“Lecturas creadas”**, donde el docente puede visualizar el material cargado y asignarlo a un grupo de estudiantes. En esta pantalla, como se observa en la figura, se presenta la información principal del texto y las opciones de administración.

Visualización del material subido

Cada lectura aparece con:

* **Título del texto** (por ejemplo, *Paco Yunque*).
* **Vista previa de la descripción**, donde se muestra un resumen o introducción del contenido.
* El botón **“Ver más”**, que permite ampliar la descripción completa del texto.
* El botón **“Ver archivo”**, que abre el documento cargado para su revisión.
* El botón **“Crear asignación”**, ubicado en la parte derecha, el cual permite iniciar el proceso de envío del material a los estudiantes.

Asignar lectura a los estudiantes

Debajo de la tarjeta del material, el sistema muestra un formulario sencillo para seleccionar a los estudiantes que recibirán la lectura. Este formulario incluye:

* **Campo para ingresar código de estudiante**   
  El docente puede escribir los correos de los estudiantes separados por comas como una versión rápida para pruebas.
* **Campo para la fecha de entrega**  
  Se debe ingresar en el formato **dd/mm/aaaa**, indicando la fecha límite en la que los estudiantes deberán entregar la actividad.

Una vez completados los datos, el docente debe presionar el botón **“Confirmar asignación”** para enviar la lectura al grupo seleccionado.



Luego de asignar lectura saldrá un mensaje asignación creada



**Tablero de Lecturas (Vista del Docente)**

El **Tablero de lecturas** es la sección donde el docente puede revisar el avance de sus estudiantes, verificar las lecturas entregadas y enviar retroalimentación directamente desde esta pantalla. Esta vista permite gestionar el desempeño sin necesidad de cambiar a otras secciones del sistema.

Encabezado de la pantalla

En la parte superior se muestran dos botones:

* **Actualizar**  
  Permite recargar la información del tablero para visualizar nuevas entregas o cambios recientes realizados por los estudiantes.
* **Volver a lecturas**  
  Regresa a la pantalla principal del módulo de lecturas del docente, donde se administran las lecturas creadas o asignadas.

Filtro de lecturas

Debajo del encabezado se encuentra el selector:

**“Todas mis lecturas”**

Este filtro permite:

* Seleccionar una lectura específica para revisar únicamente las entregas relacionadas a ella.
* Mantener la opción “Todas mis lecturas” para visualizar el listado completo de entregas realizadas por los estudiantes en todas las actividades asignadas.

Información del estudiante

Cada entrega aparece listada con los siguientes datos:

* **Nombre del estudiante:** Identifica a quién pertenece la entrega.
* **Correo del estudiante:** Se muestra para referencia del docente.
* **Estado de la entrega:** Indicado mediante una etiqueta (por ejemplo: “Pendiente”).
* **Campo de puntaje:**  
  El docente puede ingresar la nota correspondiente (por ejemplo, del 0 al 20 o según el criterio definido).

Retroalimentación al estudiante

El sistema cuenta con un espacio para ingresar:

* **Comentario:**  
  Campo donde el docente escribe observaciones, recomendaciones o retroalimentación sobre la lectura realizada por el estudiante.
* **Botón “Enviar feedback”:**  
  Registra tanto el puntaje como el comentario, y envía esta información al estudiante para que pueda consultarla desde su perfil.



**Creación de la Cuenta con Rol Alumno**

Para que un estudiante pueda acceder al sistema y visualizar las lecturas asignadas, es necesario que primero cree una cuenta con el **Rol Alumno**. En esta pantalla, el usuario debe presionar el botón **“Registrarse”**, tal como se muestra en la interfaz inicial del sistema.

Al seleccionar esta opción, el sistema abrirá una ventana donde el estudiante deberá ingresar sus datos personales para completar el registro.

1.1. Completar datos personales

El estudiante debe llenar los siguientes campos del formulario:

* **Nombre completo**
* **Correo electrónico**
* **Rol**
* **Contraseña**
* **Confirmación de contraseña**

Una vez ingresados los datos, el estudiante debe seleccionar el **Rol “Alumno”** en la lista desplegable correspondiente.

1.2. Finalizar registro

Después de completar toda la información solicitada, el usuario debe presionar el botón **“Registrarme”**.

El sistema revisará los datos ingresados y enviará automáticamente un mensaje de confirmación al correo electrónico registrado.



**Perfil de Estudiante**

Después de crear su cuenta y acceder al sistema, el alumno es dirigido automáticamente al **Perfil de estudiante**, donde deberá completar información básica que permitirá personalizar sus actividades y el seguimiento de su aprendizaje. Esta pantalla, como se muestra en la figura, contiene diversos campos que el estudiante debe registrar antes de continuar.

Datos académicos

En la parte superior se encuentran los campos relacionados al área de estudio:

* **Programa / carrera:**  
  Aquí el estudiante debe seleccionar o escribir su carrera profesional (por ejemplo: *Ing. de Sistemas*).
* **Ciclo o semestre:**  
  El alumno debe indicar el ciclo académico en el que se encuentra actualmente.

Objetivo del curso

* **Objetivo principal con este curso:**  
  El estudiante debe escribir qué espera lograr, por ejemplo: *Aprender a leer críticamente*.  
  Esta información ayuda a personalizar las recomendaciones del sistema.

Intereses de lectura

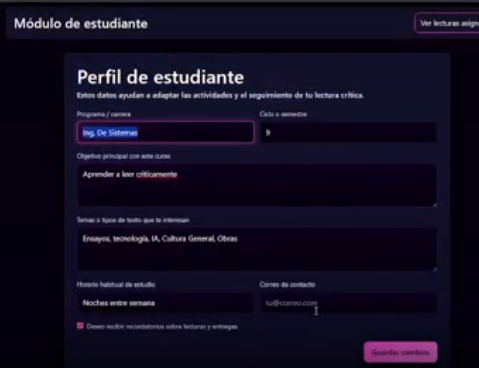
* **Temas o tipos de textos que te interesan:**  
  Se puede registrar uno o varios intereses, tales como *Ensayos, tecnología, IA, cultura general, obras literarias*, entre otros.

Datos de contacto y disponibilidad

* **Horario habitual de estudio:**  
  El alumno debe indicar cuándo suele estudiar (por ejemplo: *Noches entre semana*).
* **Correo de contacto:**  
  Campo donde se ingresa el correo principal para recibir notificaciones y recordatorios.
* **Casilla de notificaciones:**  
  El estudiante puede activar la opción **“Deseo recibir recordatorios sobre lecturas y entregas”** para mantenerse al tanto de sus actividades.

Guardar información

Una vez completados todos los campos, el alumno debe presionar el botón **“Guardar cambios”** para registrar su perfil en el sistema.



**Pantalla de Lecturas Asignadas**

Esta pantalla muestra al estudiante las lecturas enviadas por el docente. Desde aquí puede revisar el contenido, marcar su progreso, subir archivos y responder preguntas generadas automáticamente por el sistema.

Información de la lectura asignada

En la parte superior se muestra:

* **Título de la lectura** (en este caso: *Paco Yunque*).
* **Descripción breve del texto**, que sirve como introducción o resumen proporcionado por el docente.
* **Estado de la lectura**, mostrado mediante una etiqueta (como “Pendiente”).

Junto a esto aparece el botón **“Abrir lectura”**, que permite acceder al texto completo.

Marcar progreso

El estudiante cuenta con el botón **“Marcar como leído”**, el cual sirve para indicar que ha revisado la lectura.  
Al activarlo, el sistema registra el avance y notifica al docente.

Subida de archivo (Entrega del estudiante)

La plataforma permite que el estudiante adjunte un archivo relacionado a la lectura o actividad asignada.  
Para ello dispone de:

* **Campo “Subir archivo”** donde puede seleccionar un archivo en formato PDF, DOCX o TXT.
* **Botón “Enviar tarea”** que registra el archivo como entrega oficial.

Este archivo queda visible para el docente en su Tablero de Lecturas.

Notas para el docente

El estudiante puede añadir un comentario o mensaje breve dirigido al docente.  
Este campo es utilizado para:

* Aclarar dudas.
* Brindar contexto sobre la entrega.
* Enviar observaciones sobre la lectura.

El comentario se envía junto con la tarea.

Preguntas generadas por IA

En la parte inferior se encuentra el módulo **“Preguntas generadas por IA”**, donde se presentan preguntas de comprensión automática relacionadas con el texto.

El sistema indica:

* **Cantidad total de preguntas**.
* **Cuántas ya se respondieron**.
* **Cuántas están pendientes**.

Cada pregunta está organizada por niveles (literal, inferencial, crítico), y el estudiante puede abrirlas haciendo clic en **“Ver preguntas”**.

Una vez respondidas, las preguntas quedan registradas como parte del progreso del estudiante y pueden ser revisadas por el docente.

