“Año de la recuperación y consolidación de la economía peruana”

Universidad Continental

Taller de Proyectos 2

EAP Ing. Sistemas e Informática

DOCENTE: Daniel Gamarra Moreno

TEMA:

“Tutor Virtual De Lectura Crítica”

PRESENTADO POR:

APELLIDOS Y NOMBRES	CÓDIGO
Dueñas Guerra Jesús Korlant	72122566
Gutierrez Medina Jesús Manuel	75826567
Chavez Estrella José Jhovanni	75557750
Lopez Idone Jose Gianmarco	71698090
Reymundo Rodriguez Cristhian Jhon	75049277
Asistente Virtual	

HUANCAYO - PERÚ
2025

1. Declaración de la Visión del Proyecto

Desarrollamos una aplicación web de tutoría de lectura crítica que utiliza inteligencia artificial (IA) para generar preguntas, detectar falacias y ofrecer retroalimentación, complementada con automatización para personalizar el aprendizaje, todo bajo un modelo accesible y sostenible, buscando mejorar la comprensión lectora de estudiantes y docentes. (Reducir)

2. Project Charter

Nombre del Proyecto: Tutor Virtual de Lectura Crítica
Fecha de Inicio: 28/08/2025
Fecha de Finalización Prevista: 04/12/2025
Sponsor del Proyecto: Universidad Continental
Gerente del Proyecto: Dueñas Guerra Jesus Korlant
Equipo del Proyecto: Equipo de desarrollo full-stack MERN

1. **Propósito y descripción**
   Desarrollar una aplicación web interactiva que sirva como tutor virtual de lectura crítica, con apoyo de inteligencia artificial (IA de NLP) y automatización (n8n), orientada a estudiantes universitarios. El sistema busca mejorar la comprensión lectora y el pensamiento crítico, ofreciendo rutas personalizadas de estudio y retroalimentación inmediata.

2. **Objetivos del Proyecto:**

- Desarrollar una aplicación web con stack MERN (MongoDB, Express.js, React.js, Node.js).
- Entregar PMV funcional en 4 sprints
- Integrar al menos dos funcionalidades de IA: generación de preguntas y detección de sesgos.
- Implementar automatización con n8n para gestionar sesiones, notificaciones y seguimiento.
- Asegurar usabilidad, accesibilidad y diseño responsive.
- Cumplir con las prácticas de Git Flow, pruebas automatizadas y contenerización con Docker.

Objetivo	Fecha de Vencimiento
Desarrollar una aplicación web full-stack con el stack MERN (MongoDB, Express, React, Node.js) que permita evaluar y mejorar la comprensión lectora de estudiantes universitarios.	Semana 1 – 15
Implementar un módulo de generación automática de preguntas críticas y de opción múltiple mediante IA (OpenAI/Hugging Face).	Semana 6
Integrar un sistema de detección de sesgos y falacias en los textos cargados, con retroalimentación automática.	Semana 8
Configurar flujos de automatización con **n8n** para enviar recordatorios y registrar el progreso de los estudiantes.	Semana 10

Contenerizar la solución con **Docker** y establecer un pipeline CI/CD para garantizar portabilidad y despliegues reproducibles.	Semana 12
Desarrollar y ejecutar pruebas unitarias y E2E (Jest, Cypress) con al menos **70% de cobertura**.	Semana 14
Documentar el sistema con manuales de usuario, guía técnica y métricas de usabilidad y accesibilidad (WCAG 2.1 AA).	Semana 15

3. Alcance:

- Frontend en React.js con estado gestionado (Context API o Redux).
- Backend con Express.js y API REST.
- Base de datos MongoDB.
- Integración con modelos de NLP (Hugging Face/OpenAI) para IA.
- Automatización de flujos con n8n: asignación → lectura → actividad → evaluación → notificación.
- Pruebas unitarias y E2E con cobertura ≥70%.
- Documentación técnica y diagramas de arquitectura.

4. Requisitos de Alto Nivel:

a. Requisitos Funcionales

- El sistema debe permitir el registro y autenticación de usuarios (estudiantes y docentes).
- El sistema debe generar preguntas contextuales a partir de los textos cargados.
- El sistema debe detectar sesgos y falacias en los textos procesados.
- El sistema debe ofrecer rutas de estudio personalizadas.
- El sistema debe enviar notificaciones y recordatorios automáticos mediante flujos en n8n.
- El sistema debe registrar y mostrar el progreso de los estudiantes en la plataforma.
- El sistema debe permitir la evaluación de comprensión lectora en base a actividades programadas

b. Requisitos No Funcionales

- La aplicación debe desarrollarse con el stack MERN (MongoDB, Express.js, React.js, Node.js).
- El frontend debe estar desarrollado en React.js con manejo de estado mediante Context API o Redux.
- El backend debe implementarse con Express.js exponiendo servicios a través de una API REST.
- La base de datos debe ser MongoDB en la nube (ejemplo: MongoDB Atlas).

- El sistema debe integrar modelos de NLP (Hugging Face / OpenAI) a través de APIs externas.
- La aplicación debe ser responsive y cumplir con las normas de accesibilidad WCAG 2.1 AA.
- El sistema debe tener al menos 70% de cobertura en pruebas unitarias y E2E.
- Se debe entregar documentación técnica completa (README, diagramas de arquitectura, manual de usuario).
- El sistema debe desplegarse en contenedores Docker.

5. **Entregables Principales:**

- Código fuente en GitHub.
- Documentación (README, diagramas, informe técnico).
- Demo funcional con video explicativo (opcional para sobresaliente).
- Informe de impacto ambiental.

6. **Hitos y cronograma tentativo**

- Sprint 1 (28/08 – 11/09): Configuración inicial, base de datos, frontend básico.
- Sprint 2 (12/09 – 25/09): API REST, primeras integraciones de IA (preguntas contextuales).
- Sprint 3 (26/09 – 09/10): Detección de sesgos, pruebas iniciales, automatización con n8n.
- Sprint 4 (10/10 – 23/10): Mejoras de accesibilidad, responsive, pruebas E2E.
- Sprint 5 (24/10 – 07/11): Documentación, CI/CD, despliegue en Docker.
- Entrega final (04/12): Presentación y socialización de resultados.

7. **Riesgos identificados**

- Técnicos: problemas de integración con APIs externas de IA.
- Recursos: limitación de tiempo del equipo dentro del semestre académico.
- Usuarios: resistencia inicial al uso de herramientas digitales.
- Mitigación: pruebas tempranas, documentación clara, sesiones de retroalimentación con usuarios piloto.

Criterios de Aceptación:

- Aplicación funcional y desplegable con Docker.
- Cumplimiento de los requisitos técnicos y funcionales.
- Cobertura de pruebas ≥70%.
- Documentación completa y clara.

Suposiciones y Restricciones

Suposiciones:

Categoría	Descripción	Estado/Comentario

Tecnológica	Los modelos de IA de Hugging Face/OpenAI estarán disponibles vía API.	Validado, APIs accesibles en capa gratuita.
Recursos	El equipo cuenta con conocimientos técnicos en MERN e integración de IA.	En proceso de fortalecimiento con autoestudio y tutoriales.
Conectividad	Los usuarios tendrán acceso a internet estable para usar la plataforma.	Probablemente, se requiere piloto inicial en campus con WiFi.
Automatización	n8n permitirá la orquestación de flujos sin requerir licencia premium.	Confirmado, plan free disponible.
Académico	Se contará con retroalimentación del docente en cada sprint.	Pendiente de coordinación formal.

Restricciones:

Categoría	Descripción	Estado/Comentario
Tiempo	El desarrollo se limita al semestre (4 meses, 5 sprints).	Fijo, no ampliable.
Presupuesto	Presupuesto $0: solo recursos gratuitos (MongoDB Atlas, Hugging Face, etc.).	Permanente.
Tecnológica	Uso exclusivo de stack MERN y software open-source.	Aceptada por el equipo.
Calidad	Debe cumplir con ≥70 % cobertura en pruebas unitarias/E2E.	Restricción técnica.
Accesibilidad	La aplicación debe ser responsive y cumplir con WCAG 2.1 nivel AA.	Normativa interna.

Límites del Proyecto

Incluye:

- Desarrollo de la aplicación web con stack MERN (MongoDB, Express, React, Node.js).
- Implementación de módulos de IA para generación de preguntas y detección de sesgos/falacias.
- Automatización con n8n para recordatorios y notificaciones de avance.
- Panel de progreso para estudiantes y docentes.
- Pruebas unitarias y end-to-end con cobertura mínima del 70%.
- Documentación técnica y manual de usuario.

No incluye:

- Desarrollo de aplicación móvil nativa.
- Integración con sistemas de gestión académica institucional.
- Acceso a todas las bases de datos de pago (se usará solo contenido abierto).
- Soporte offline o aplicaciones de escritorio.

Resultado esperado	Indicador de éxito
Aplicación web funcional desplegada en la nube (Render/Vercel).	**Acceso a través de una URL pública y operativa.**
Generación automática de preguntas de comprensión lectora.	**Mínimo 5 preguntas por texto con nivel literal, inferencial y crítico.**
Módulo de detección de falacias y sesgos implementado.	**Identificación correcta en al menos el 80% de un set de prueba docente.**
Automatización de recordatorios y seguimiento en n8n.	**Notificaciones enviadas automáticamente a estudiantes según cronograma.**
Pruebas unitarias y E2E ejecutadas con éxito.	**Cobertura mínima del 70% en CI/CD.**
Documentación y manual de usuario finalizados.	**Documento entregado con guías de instalación y uso.**

Declaración del Equipo del Proyecto

1. **Visión del equipo**

Ser un equipo multidisciplinario comprometido con la creación de un tutor virtual de lectura crítica que aproveche la inteligencia artificial y la automatización para impactar en la formación académica, trabajando con colaboración, disciplina y responsabilidad compartida.

2. **Objetivo del equipo**

Lograr la entrega de un Producto Mínimo Viable (PMV) funcional, accesible y probado dentro del semestre académico, garantizando calidad técnica (≥70 % cobertura de pruebas) y valor educativo para los usuarios piloto.

3. **Roles y Responsabilidades:**

Rol	Responsabilidades
Líder de Proyecto Dueñas Guerra Jesús Korlant	Coordinación general, seguimiento de sprints, comunicación con stakeholders.
Frontend Developer Chavez Estrella José Jhovanni	Desarrollo de la interfaz en React.js, estado con Context API/Redux, pruebas.
Backend Developer Lopez Idone Jose Gianmarco	API REST con Express.js, integración con MongoDB, lógica de negocio.
Especialista en IA Gutierrez Medina Jesús Manuel	Integración de modelos de NLP, generación de preguntas, detección de sesgos.
DevOps/Automation Gutierrez Medina Jesús Manuel	Configuración de Docker, n8n, CI/CD, despliegue.
Tester Cristhian Jhon Reymundo Rodriguez	Pruebas unitarias, E2E, documentación de casos de prueba.

4. **Normas de trabajo**

- Puntualidad en entregables de sprint y asistencia a reuniones.
- Respeto en la comunicación (escucha activa, feedback constructivo).
- Uso obligatorio de control de versiones con Git Flow.
- Documentación mínima en cada entrega (README actualizado y commits claros).

5. Reuniones

- Daily (breve, 15 min): dos veces por semana vía meet para sincronizar avances.
- Sprint Planning y Review: al inicio y fin de cada sprint (presencial o virtual).
- Retrospectiva: al final de cada sprint para identificar mejoras de equipo.

6. Herramientas de comunicación y gestión

- **Gestión de tareas:** Trello (Kanban con backlog, en curso, pruebas, finalizado).
- **Comunicación sincrónica:** GoogleMeet (reuniones de voz/video).
- **Comunicación asincrónica:** WhatsApp (mensajes rápidos).
- **Repositorios:** GitHub (código, documentación).
- **Documentos compartidos:** Google Drive (manuales, informes).

7. Mecanismo de toma de decisiones

- Decisiones técnicas por consenso en reunión de equipo.
- En caso de empate, se prioriza el criterio del Líder de Proyecto, previa consulta al docente.

8. Valores del equipo

- **Compromiso:** cumplir con tiempos y entregables.
- **Transparencia:** reportar avances y obstáculos de forma clara.
- **Colaboración:** apoyo mutuo en tareas críticas.
- **Responsabilidad compartida:** el éxito o fracaso es del equipo, no individual.

Equipo:

- Dueñas Guerra Jesús Korlant – Líder de Proyecto / Full-Stack
- Chavez Estrella José Jhovanni – Frontend Specialist
- Lopez Idone Jose Gianmarco – Backend Specialist
- Gutierrez Medina Jesús Manuel – IA & Automation
- Cristhian Jhon Reymundo Rodriguez – Tester & Documentación

9. Presupuesto

Categoría	Detalle	Costo Estimado (S/.)
Infraestructura	**Servicios en la nube gratuitos (Render/Vercel, Railway, MongoDB Atlas). En caso de ampliación a planes básicos de pago.**	**800.00**

Licencias de IA / APIs	**OpenAI API, Hugging Face, o equivalentes (créditos iniciales para pruebas y despliegues).**	**1,200.00**
Herramientas de Automatización	**n8n (open-source, hosting en servidor propio gratuito).**	**0.00**
Contenerización y CI/CD	**Docker y GitHub Actions (gratuito en plan educativo).**	**0.00**
Gestión de Proyecto	**Herramientas de trabajo colaborativo (Trello, Jira, Google Drive).**	**0.00**
Recursos Humanos	**Dedicación estimada de 7 integrantes (trabajo académico no remunerado).**	**0.00**
Documentación y difusión	**Manual de usuario, informes técnicos, impresión de resúmenes finales.**	**300.00**
Reserva de Contingencia (15%)	**Para imprevistos técnicos o ampliación de uso de API.**	**375.00**
Reserva de Gestión (10%)	**Fondo adicional para cambios de alcance menores.**	**250.00**

- El presupuesto es académico y simulado, ya que la mayoría de servicios son gratuitos en sus planes iniciales.
- Se incluye un monto base para APIs .
- La reserva de contingencia cubre cualquier imprevisto técnico.
- La reserva de gestión permite flexibilidad ante cambios menores en el alcance.

BACKLOG INICIAL:

EPIC CENTRAL:
"Como estudiante universitario, quiero contar con una plataforma digital con inteligencia artificial y automatización, para mejorar mi comprensión lectora y pensamiento crítico, de manera que pueda reforzar mis habilidades académicas de forma autónoma."

Épica 01 – Evaluación de textos

Código	Historia de Usuario	Duración	Criterios de Aceptación	Prioridad
HU01	Como estudiante, quiero subir un texto (PDF/DOCX/TXT)	1 semana	- Subida ≤10 MB.- Texto guardado en BD.- Mensaje	Alta – base estructural

	para que el sistema lo procese.		de confirmación.	

Épica 02 – Funcionalidades con IA

Código	Historia de Usuario	Duración	Criterios de Aceptación	Prioridad
HU02	**Como estudiante, quiero que el sistema genere preguntas de comprensión lectora de diferentes niveles.**	**3 semanas**	**- ≥5 preguntas por texto.- Preguntas clasificadas (literal, inferencial, crítico).- Exportación en CSV.**	**Alta – valor académico**
HU03	**Como estudiante, quiero que el sistema detecte falacias y sesgos en el texto cargado.**	**3 semanas**	**- Reporte con tipo de falacia.- Destacar fragmento afectado.- Precisión ≥80% en pruebas docentes.**	**Alta – diferenciador**

Épica 03 – Seguimiento y Automatización

Código	Historia de Usuario	Duración	Criterios de Aceptación	Prioridad
HU04	Como estudiante, quiero ver un dashboard con mi progreso en las prácticas.	2 semanas	- Métricas de intentos completados.- Historial de notas por intento.- Visualización gráfica.	Media-Alta

HU05	Como estudiante, quiero recibir notificaciones automáticas para recordar mis actividades.	1 semana	- Integración con n8n.- Recordatorios vía correo o app.- Configuración personalizada.	Media
HU06	Como docente, quiero acceder a un panel para visualizar el rendimiento de mis estudiantes.	2 semanas	- Ver progreso agregado por curso.- Exportar métricas en PDF/Excel.- Roles diferenciados (docente/estudiante).	Media
HU07	Como usuario, quiero registrarme e iniciar sesión de forma segura.	1 semana	- Registro/login con credenciales cifradas.- Roles: estudiante/docente.- Sesión segura con JWT.	Alta – seguridad básica

Procesos de Equipo

Canales de comunicación

- Gestión de tareas: Trello (Kanban con columnas: Backlog, En curso, En prueba, Hecho).
- Comunicación sincrónica: Google Meet (reuniones semanales).
- Comunicación asincrónica: WhatsApp (mensajes rápidos).
- *(Opcional para mejora)* Discord o Slack para centralizar canales de voz/chat/documentos.

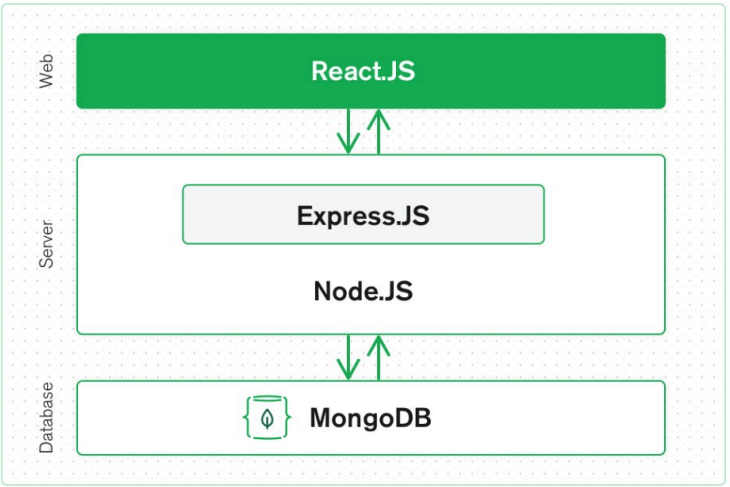
Acuerdos de trabajo

- **Horarios de reunión**:
    - *Daily* breve (15 min), dos veces por semana.
    - *Sprint Planning & Review*: al inicio y fin de cada sprint.

- *Retrospective*: al final de cada sprint.

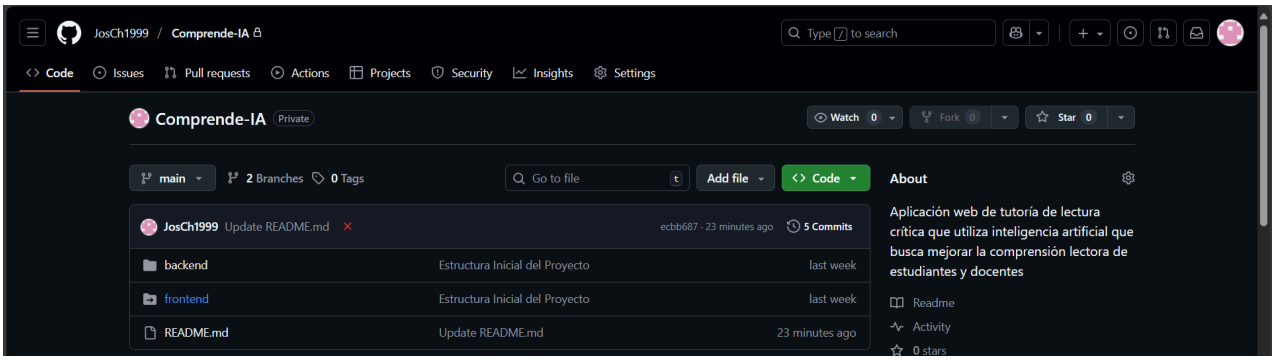
- **Definition of Done (DoD):**
  - Código subido a GitHub con Git Flow.
  - Pruebas unitarias implementadas (≥70% cobertura).
  - Documentación mínima en README.
  - Interfaz funcional y accesible en al menos 2 navegadores.
- **Roles iniciales**:
  - Líder de Proyecto: coordinación y comunicación con stakeholders.
  - Frontend Developer: interfaz React.js.
  - Backend Developer: API Express + MongoDB.
  - Especialista en IA: integración de NLP (Hugging Face / OpenAI).
  - DevOps/Automation: Docker, CI/CD, n8n.
  - Tester: pruebas unitarias y E2E.

Arquitectura mínima (Walking Skeleton)

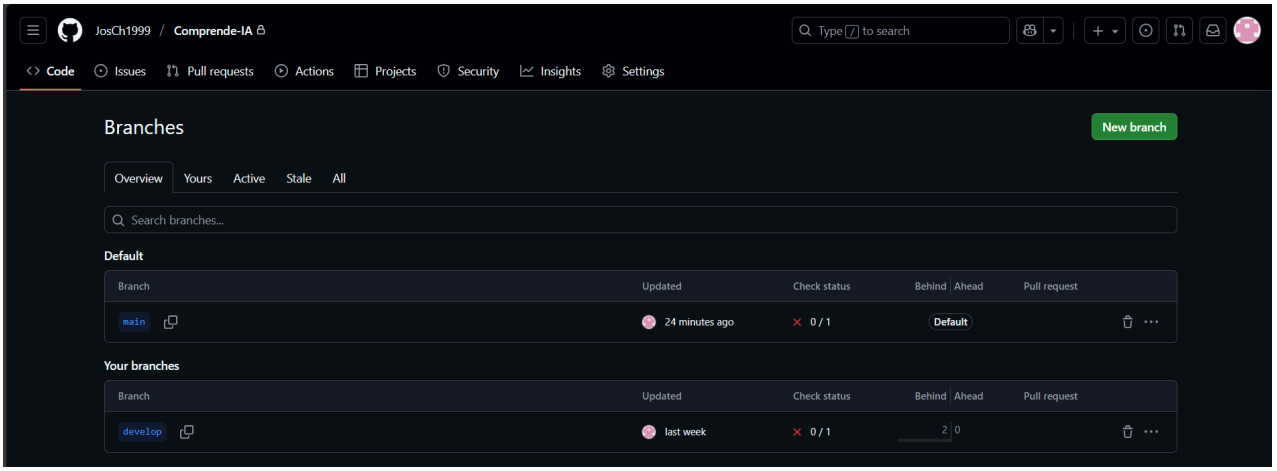


Nombre APP : AppComprende

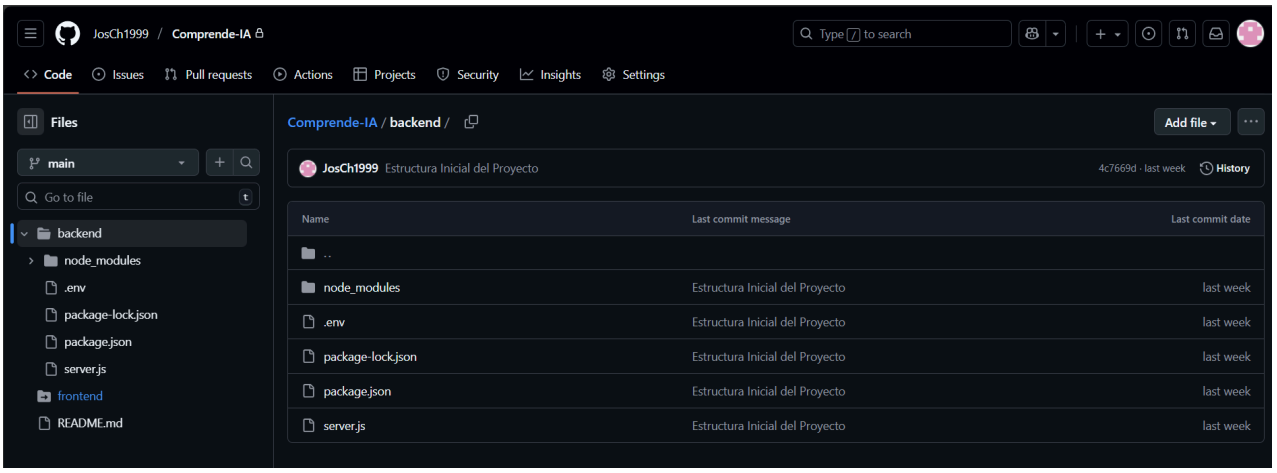
Creación de repositorio en GitHub (un integrante crea, el resto clona).



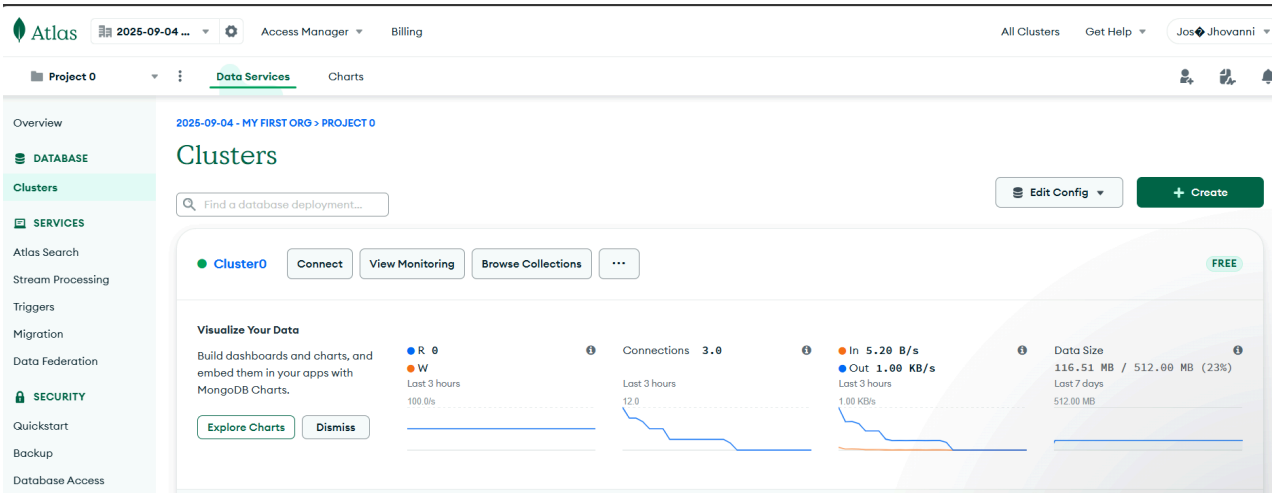
Configuración inicial de ramas (main, develop, ramas por funcionalidad).



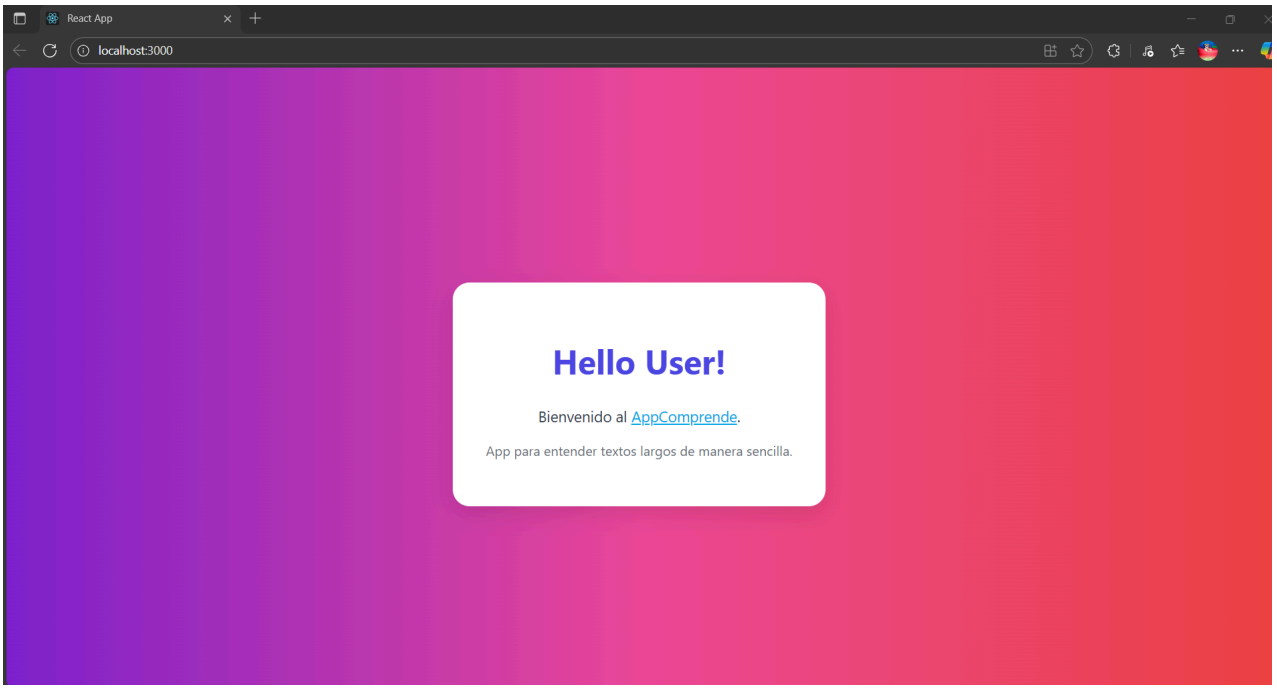
Estructura base del proyecto MERN (carpetas, dependencias mínimas).



Configuración básica: conexión Express–MongoDB, Hello World en React.



Evidencia del despliegue inicial (screenshot o URL)



Github: https://github.com/JosCh1999/Comprende-IA