

**Universidad
Tecnológica
del Perú**

PROYECTO FINAL DEL CURSO

Análisis, diseño e implementación del sistema:

Plataforma de Gestión de Aprendizaje Adaptativo "AdaptaTest"

CURSO

CURSO INTEGRADOR I: SISTEMAS - SOFTWARE

CICLO

VI

DOCENTE

Mg. Ana Meliza Garayar Ttito

INTEGRANTES

Doloriert Tasayco Joseph Aldair

Trigoso Rojas Rodrigo Alexander

ICA - PERÚ

2025

Contenido

INTRODUCCIÓN	5
RESUMEN.....	6
ANTECEDENTES	6
Antecedentes Internacionales	6
Antecedentes Nacionales	6
Antecedentes Regionales.....	7
Planteamiento del Problema.....	7
Objetivos	7
Objetivo General.....	7
Objetivos Específicos.....	7
Justificación	8
Marco Referencial Y Teórico.....	9
Inicio del proyecto	13
Descripción de la Organización	13
Alcance y objetivos	13
Análisis de Riesgos	14
Limitaciones del Proyecto	15
Supuestos Fundamentales.....	16
Requerimientos Del Proyecto.....	17
Requerimientos Funcionales	17
Requerimientos No Funcionales	21
CASOS DE USO.....	24
ACTORES DEL SISTEMA.....	24
CASOS DE USO POR ACTOR.....	25
DESCRIPCIÓN DETALLADA DE CASOS DE USO	29
Lean Canvas	43
PMBOK.....	43
Project Charter:	43
WBS:.....	44

Cronograma de Actividades GANT:	44
Prototipado:	45
Descripción de procesos:	51
Diagrama de proceso Bizagi:	51
Registro de usuario	51
Creación de Cursos y Secciones	52
Gestión de Contenido	53
Proceso de Evaluación	54
Análisis de resultados	55
Diagrama de clases:	56
Diagrama entidad Relación:	57
CREACIÓN DE BASE DE DATOS:	57
DISEÑO LÓGICO DE BASE DE DATOS - ADAPTATEST	57
1. INTRODUCCIÓN	57
2. MODELO ENTIDAD-RELACIÓN	57
3. DIAGRAMA DE CARDINALIDADES	69
4. REGLAS DE NEGOCIO	70
5. NORMALIZACIÓN	71
6. CONCLUSIONES	71
DISEÑO FÍSICO DE BASE DE DATOS - ADAPTATEST	71
1. INTRODUCCIÓN	71
2. SISTEMA DE GESTIÓN DE BASE DE DATOS	72
3. ESQUEMAS DE COLECCIONES	72
4. TIPOS DE DATOS DETALLADOS	79
5. ESTRATEGIA DE ÍNDICES	80
6. RESTRICCIONES DE INTEGRIDAD	81
7. SEGURIDAD	82
8. OPTIMIZACIONES DE RENDIMIENTO	84
9. BACKUP Y RECUPERACIÓN	85
10. MONITOREO	86
DICCCIONARIO DE DATOS	86
ENLACE A GITHUB Y CONTROL DE VERSIONES	111

Enlace al Repositorio	111
Control de Versiones	113
CAPTURAS DE LA APLICACIÓN	113
Arquitectura y Tecnologías (MVC / DAO).....	113
Capturas de las Interfaces	128

INTRODUCCIÓN

El presente proyecto documenta el análisis, diseño, desarrollo e implementación de "AdaptaTest", una Plataforma de Gestión de Aprendizaje Adaptativo (LMS) de nueva generación, concebida para instituciones educativas de nivel secundario y superior. El sistema aborda una de las problemáticas más críticas en la educación moderna: la falta de personalización en los procesos de enseñanza y evaluación. "AdaptaTest" integra un robusto sistema de gestión académica con un innovador motor de evaluación que ajusta dinámicamente la dificultad de las pruebas al nivel de competencia de cada estudiante en tiempo real, ofreciendo una experiencia de aprendizaje verdaderamente individualizada.

En un contexto donde la digitalización de la educación se ha vuelto fundamental, la implementación de un sistema de software como "AdaptaTest" es de vital importancia para superar las limitaciones del modelo tradicional de "talla única". Un sistema de software permite no solo automatizar tareas administrativas complejas como la matrícula y la calificación, sino también escalar la personalización del aprendizaje a un nivel inalcanzable de forma manual. Al proporcionar analíticas detalladas sobre el rendimiento, la plataforma empodera a los docentes con información precisa para adaptar su pedagogía, convirtiendo los datos en una herramienta para cerrar brechas de conocimiento y mejorar la retención académica.

Este trabajo se ha organizado siguiendo las fases del ciclo de vida del desarrollo de software, comenzando con el análisis de requerimientos y la planificación estratégica. Posteriormente, se detalla el diseño de la arquitectura del sistema, el desarrollo de un backend escalable con tecnología MERN, y la construcción de un frontend interactivo con React y Redux. El propósito final del proyecto es la entrega de un Producto completamente funcional, que no solo valida la solución técnica propuesta, sino que también demuestra su potencial como un modelo de negocio viable para optimizar los procesos pedagógicos en las instituciones educativas del Perú.

RESUMEN

Este proyecto detalla la creación de "AdaptaTest", una plataforma web orientada a instituciones educativas que enfrentan el problema de la baja retención de conocimiento y la falta de personalización en sus modelos de enseñanza a distancia. El objetivo general es desarrollar un sistema que mejore la eficacia pedagógica a través de un motor de evaluación que adapta la dificultad de las pruebas en tiempo real al rendimiento del estudiante. La solución propuesta es una plataforma SaaS (Software como Servicio) con una arquitectura multi-rol (Administrador, Coordinador, Profesor, Estudiante y Padre de Familia) que permite una gestión académica integral. Los beneficios clave incluyen la optimización del tiempo de estudio del alumno, la provisión de analíticas detalladas para los educadores y la automatización de los procesos administrativos y curriculares de la institución.

ANTECEDENTES

El desarrollo de sistemas de gestión de aprendizaje (LMS) y plataformas de aprendizaje adaptativo es un campo de investigación y desarrollo activo a nivel global, con una creciente adopción en el sistema educativo peruano.

Antecedentes Internacionales

A nivel mundial, el aprendizaje adaptativo se considera una estrategia clave para personalizar la educación a gran escala. Plataformas como Knewton y ALEKS han sido pioneras en el uso de algoritmos para crear rutas de aprendizaje individualizadas, demostrando mejoras en la participación y el rendimiento estudiantil. Investigaciones recientes se centran en la integración de Inteligencia Artificial para potenciar estos sistemas, permitiendo no solo adaptar el contenido, sino también ofrecer retroalimentación en tiempo real y analíticas predictivas sobre el éxito del estudiante. Casos de estudio en Latinoamérica, como la implementación de asistentes virtuales y chatbots en Brasil, muestran una tendencia hacia la automatización y el soporte personalizado del estudiante.

Antecedentes Nacionales

En Perú, la adopción de plataformas LMS se aceleró significativamente a raíz de la pandemia, con instituciones de educación superior implementando soluciones como Moodle y Canvas. Universidades como la UTP y la Continental ya utilizan Canvas para gestionar sus cursos, destacando su flexibilidad para el acceso 24/7 a los materiales. Sin embargo, la mayoría de estas implementaciones se centran en la gestión de contenido y la evaluación tradicional. Aunque el interés en la IA

para personalizar la educación está creciendo en el ámbito universitario peruano, la implementación de verdaderos sistemas de aprendizaje adaptativo aún es un área emergente y con un gran potencial de desarrollo.

Antecedentes Regionales

En la región de Ica y zonas aledañas, la digitalización de la educación sigue un patrón similar al nacional, con un enfoque en la adopción de aulas virtuales para garantizar la continuidad del servicio educativo. La problemática local se centra en asegurar el acceso y la calidad de la educación a distancia. Un sistema como AdaptaTest, que no solo gestiona, sino que optimiza el aprendizaje, representa una oportunidad de innovación significativa para las instituciones de la región, permitiéndoles ofrecer una ventaja competitiva a través de una metodología pedagógica más eficaz y moderna.

Planteamiento del Problema

Las instituciones educativas, desde colegios hasta academias pre-universitarias, enfrentan una ineficiencia sistemática en sus modelos de enseñanza, tanto presenciales como virtuales. El problema central es la **falta de personalización del proceso de evaluación y refuerzo académico a escala**. Los métodos tradicionales (exámenes estandarizados, tareas uniformes) no se adaptan al ritmo de aprendizaje individual, generando que estudiantes avanzados se desmotiven y que aquellos con dificultades no reciban el refuerzo específico que necesitan, lo que resulta en brechas de conocimiento y bajas tasas de retención. A esto se suma una gestión académica fragmentada, con procesos manuales para la administración de matrículas, contenidos y seguimiento del rendimiento, consumiendo valioso tiempo de docentes y administrativos.

Objetivos

Objetivo General

Desarrollar e implementar un prototipo funcional (MVP) de la plataforma de aprendizaje adaptativo "AdaptaTest", que permita a las instituciones educativas gestionar su currículo académico y ofrecer a sus estudiantes una experiencia de evaluación personalizada que optimice su proceso de aprendizaje a través de la medición de maestría por competencias.

Objetivos Específicos

- Crear un módulo que permita a las instituciones cargar y gestionar al menos 5 materias con sus competencias y subcompetencias, garantizando que el

100% de las funcionalidades CRUD (crear, leer, actualizar, eliminar) operen correctamente para la administración eficiente del currículo académico.

- Desarrollar un algoritmo que ajuste automáticamente la dificultad de las preguntas según el desempeño del estudiante, incorporando mínimo 3 niveles de dificultad por competencia y alcanzando 95% de precisión en la adaptación después de 5 respuestas consecutivas.
 - Construir un banco de al menos 200 preguntas distribuidas en las competencias piloto, con mínimo 40 preguntas por competencia (MODULOS) clasificadas por nivel de dificultad y validadas por expertos para garantizar calidad pedagógica.
 - Implementar un sistema que calcule y visualice el nivel de maestría por competencia en escala 0-100%, generando reportes individualizados para el 100% de estudiantes que completen al menos una evaluación.
 - Lograr un uptime mínimo del 99% durante el período de prueba piloto, minimizando interrupciones del servicio y estableciendo la base técnica confiable para el funcionamiento continuo de la plataforma.
-

Justificación

La realización de este proyecto es fundamental ya que aborda directamente una necesidad crítica en el sector educativo actual: la personalización del aprendizaje.

- **Beneficio Tecnológico:** El proyecto implica la integración de tecnologías modernas (stack MERN, Redux) y el diseño de un algoritmo de adaptación, lo que representa un desafío técnico relevante y una aplicación práctica de los conocimientos de la carrera de Ingeniería de Sistemas.
- **Beneficio Académico:** Para los estudiantes, la plataforma ofrece un método de estudio más eficiente, enfocando su tiempo en sus áreas débiles. Para la institución, representa una innovación pedagógica que puede mejorar sus resultados académicos y tasas de retención.
- **Beneficio Económico:** Como modelo de negocio SaaS, AdaptaTest presenta una solución de bajo costo para instituciones que no pueden permitirse sistemas LMS de alta gama, automatizando tareas administrativas y reduciendo la necesidad de tutorías personalizadas masivas.
- **Beneficio Social:** Al ofrecer una herramienta que se adapta a diferentes ritmos de aprendizaje, el proyecto promueve la equidad educativa,

brindando a cada estudiante el apoyo que necesita para alcanzar su máximo potencial.

Marco Referencial Y Teórico

Marco Referencial

El marco referencial contextualiza el desarrollo de AdaptaTest dentro del panorama actual de investigaciones, proyectos y desarrollos tecnológicos en sistemas de aprendizaje adaptativo e inteligencia artificial aplicada a la educación.

Investigaciones y Proyectos Internacionales:

- **Sistemas de Tutoría Inteligente (ITS):** A nivel internacional, se han desarrollado múltiples proyectos de investigación centrados en sistemas de tutoría inteligente que proporcionan retroalimentación y apoyo personalizado a los estudiantes. Investigaciones recientes demuestran que la IA puede evaluar el progreso y los patrones de aprendizaje de los estudiantes y proporcionar recomendaciones personalizadas para ayudar a los educadores a crear planes de enseñanza adaptados a las necesidades específicas de cada estudiante.
- **Plataformas Comerciales Exitosas:** Proyectos como Knewton, ALEKS y Smart Sparrow han establecido precedentes importantes en el campo del aprendizaje adaptativo. Estas plataformas han demostrado la viabilidad técnica y comercial de sistemas que utilizan algoritmos de machine learning para personalizar rutas de aprendizaje, validando el modelo de negocio y la efectividad pedagógica de este enfoque.
- **Investigaciones en IA o algoritmos en la Educación:** Estudios recientes han confirmado que los sistemas adaptativos basados en IA o algoritmos inteligentes permiten la automatización de determinados procedimientos y optimización de métodos, alcanzando resultados cada vez más específicos amoldados a la situación de cada alumno.

Contexto Nacional Peruano:

- **Proyectos Universitarios:** En el contexto peruano, instituciones como la Universidad Continental han iniciado proyectos exploratorios en aprendizaje adaptativo basado en IA, posicionándose como pioneras en la investigación de sistemas que se basan en la teoría del aprendizaje personalizado y utilizan técnicas avanzadas de IA o algoritmos de planificación para ajustar el contenido educativo en tiempo real. Esto representa un cambio significativo del modelo tradicional de enseñanza

homogénea, permitiendo a los estudiantes avanzar a su propio ritmo y recibir retroalimentación en tiempo real (Aparicio-Gómez & Aparicio-Gómez, 2024. p.1).

- **Implementaciones LMS Existentes:** Las universidades peruanas han desarrollado experiencia significativa en la implementación de Learning Management Systems tradicionales como Canvas, Blackboard y Moodle. Estas implementaciones proporcionan un contexto de referencia importante para AdaptaTest, ya que demuestran la receptividad del mercado educativo peruano hacia soluciones tecnológicas y establecen expectativas de funcionalidad y usabilidad.
- **Brechas Identificadas:** Las investigaciones nacionales han identificado que, aunque existe adopción de plataformas LMS, la mayoría se centra en gestión de contenido sin aprovechar completamente las capacidades adaptativas. Esta brecha representa la oportunidad que AdaptaTest busca cubrir mediante la integración de algoritmos adaptativos con funcionalidades tradicionales de gestión académica.

Contexto Regional:

- **Proyectos Piloto Locales:** En la región de Ica, diversas instituciones educativas han implementado proyectos piloto utilizando plataformas gratuitas como Chamilo LMS y Google Classroom o como la Universidad Tecnológica del Perú, implementando su propio sistema UTP+ Class. Estos proyectos han generado experiencias valiosas sobre los desafíos de implementación tecnológica en contextos con limitaciones de infraestructura y han demostrado la necesidad de soluciones más especializadas.

MARCO TEÓRICO

El marco teórico de AdaptaTest se sustenta en la convergencia de múltiples corrientes teóricas y metodológicas que proporcionan la base científica para el diseño e implementación del sistema de aprendizaje adaptativo.

Teorías Pedagógicas Fundamentales

1. **Constructivismo Educativo:** El constructivismo sostiene que el aprendizaje es un proceso activo donde los estudiantes construyen su propio conocimiento a partir de sus experiencias y saberes previos. Esta teoría, desarrollada por pioneros como Jean Piaget, Lev Vygotsky, Jerome Bruner y David Ausubel, establece que el conocimiento se construye activamente a partir de la experiencia y la interacción con el entorno.

Aplicación en AdaptaTest: AdaptaTest implementa principios constructivistas al permitir que cada estudiante construya su ruta de aprendizaje basada en su conocimiento previo y respuestas. El sistema crea un espacio favorable al aprendizaje donde cada alumno reconstruye su aprendizaje de manera personalizada, priorizando el proceso de aprendizaje sobre objetivos curriculares rígidos.

2. Teoría del Aprendizaje Personalizado: La teoría del aprendizaje personalizado sostiene que la educación debe adaptarse a las necesidades, intereses, ritmos y estilos de aprendizaje individuales de cada estudiante. Esta teoría reconoce que los estudiantes poseen diferentes fortalezas, preferencias y velocidades de procesamiento de información.

Implementación Técnica: Los sistemas adaptativos basados en esta teoría utilizan técnicas avanzadas de IA o algoritmos para ajustar el contenido educativo en tiempo real, analizando continuamente el rendimiento del estudiante y modificando la dificultad, secuencia y tipo de contenido presentado.

Teorías Tecnológicas y de Sistemas

3. Sistemas Adaptativos Inteligentes: Los sistemas adaptativos inteligentes se basan en la teoría de que los sistemas computacionales pueden modificar automáticamente su comportamiento en respuesta a cambios en el entorno o en los datos de entrada. Estos sistemas utilizan algoritmos de aprendizaje automático para mejorar continuamente su rendimiento sin intervención humana explícita.

Componentes Teóricos

- **Modelo del Estudiante:** Representación computacional del conocimiento, habilidades y preferencias del estudiante
- **Modelo del Dominio:** Estructuración del conocimiento del área temática
- **Motor Adaptativo:** Algoritmos que determinan la secuencia y dificultad del contenido
- **Interfaz Adaptativa:** Presentación personalizada de la información

Teorías de Medición y Evaluación

4. Teoría de Respuesta al Ítem (TRI): La TRI proporciona el marco teórico para la evaluación adaptativa, estableciendo relaciones matemáticas entre la habilidad del estudiante, la dificultad de las preguntas y la probabilidad de respuesta correcta. Esta teoría permite estimar con precisión el nivel de competencia del estudiante con un menor número de preguntas.

Aplicación en AdaptaTest: El sistema utiliza principios de TRI para:

- Seleccionar preguntas de dificultad óptima
- Estimar el nivel de maestría en tiempo real
- Minimizar el tiempo de evaluación manteniendo precisión
- Proporcionar mediciones comparables entre estudiantes

5. Teoría de la Competencia: La teoría de competencias establece que el aprendizaje debe organizarse en torno a capacidades específicas que integran conocimientos, habilidades y actitudes. Una competencia representa la capacidad de movilizar y aplicar correctamente recursos en situaciones específicas.

Implementación en AdaptaTest: El sistema estructura el contenido académico en competencias y subcompetencias, permitiendo:

- Evaluación granular del conocimiento
- Identificación precisa de brechas de aprendizaje
- Seguimiento del progreso por áreas específicas
- Personalización basada en fortalezas y debilidades identificadas

Teorías de Arquitectura de Software

6. Arquitectura Basada en Servicios: La arquitectura de software de AdaptaTest se fundamenta en principios de modularidad, escalabilidad y mantenibilidad. La separación de responsabilidades permite que diferentes componentes del sistema (autenticación, evaluación, adaptación, reportes) operen independientemente mientras mantienen comunicación coherente.

Implementación MERN

La elección del stack MERN (MongoDB, Express.js, React, Node.js) se basa en:

- **Escalabilidad horizontal:** Capacidad de manejar crecimiento de usuarios
- **Flexibilidad de datos:** Adaptación a estructuras de datos complejas
- **Tiempo real:** Procesamiento inmediato de respuestas y adaptación
- **Mantenibilidad:** Separación clara entre frontend y backend

7. Teoría de Experiencia de Usuario (UX): El diseño de AdaptaTest se fundamenta en principios de usabilidad y experiencia de usuario que priorizan la facilidad de uso, la accesibilidad y la satisfacción del usuario. La teoría UX establece que la tecnología debe ser intuitiva y reducir la carga cognitiva del usuario.

Implementación Práctica

- **Diseño centrado en el usuario:** Interfaces adaptadas a cada rol
- **Retroalimentación inmediata:** Respuestas del sistema en tiempo real
- **Navegación intuitiva:** Flujos de trabajo claros y predecibles
- **Accesibilidad universal:** Compatibilidad con diversos dispositivos y capacidades

Inicio del proyecto

Descripción de la Organización

El proyecto "AdaptaTest" es desarrollado por un equipo multidisciplinario de dos estudiantes de la carrera de Ingeniería de Sistemas, quienes han constituido una iniciativa emprendedora orientada a la innovación tecnológica en el sector educativo. El equipo combina competencias técnicas especializadas en desarrollo de software full-stack, arquitectura de sistemas y algoritmos adaptativos, con una visión estratégica enfocada en la transformación digital de la educación peruana.

Estructura del Equipo:

- **Doloriert Tasayco Joseph Aldair:** Especialización en desarrollo backend, arquitectura de sistemas y algoritmos de aprendizaje adaptativo
- **Trigoso Rojas Rodrigo Alexander:** Especialización en desarrollo frontend, experiencia de usuario (UX/UI) y gestión de proyectos ágiles

Filosofía Organizacional: La organización se fundamenta en la aplicación de metodologías ágiles (Scrum, Kanban) y herramientas tecnológicas de vanguardia para la construcción de soluciones educativas escalables. El enfoque organizacional prioriza la iteración rápida, la validación temprana con usuarios reales y la mejora continua basada en retroalimentación del mercado.

Visión Estratégica: Posicionarse como pioneros en el desarrollo de sistemas de aprendizaje adaptativo en el contexto peruano, creando soluciones tecnológicas que democratizan el acceso a educación personalizada de calidad y generen impacto social medible en los indicadores educativos nacionales.

Alcance y objetivos

El proyecto contempla el desarrollo, ejecución y validación de un prototipo funcional (MVP) de la plataforma de aprendizaje adaptativo "AdaptaTest", con capacidad de expansión posterior.

Objetivos Específicos Cuantificables

1. Implementación del Sistema Multi-Rol: Desarrollar y validar 5 roles diferenciados: Administrador, Coordinador, Profesor, Estudiante y Padre de Familia. Lograr 100% de funcionalidad operativa para la gestión de permisos y accesos por rol y alcanzar 95% de satisfacción en usabilidad específica para cada rol

2. Desarrollo del Módulo de Biblioteca de Contenido: Crear una interfaz intuitiva que permita al 90% de los profesores crear y gestionar contenido de manera autónoma. Lograr que el 85% de los docentes capacitados puedan utilizar todas las funcionalidades sin asistencia técnica e implementar un sistema de reutilización de contenido que reduzca en 60% el tiempo de preparación de evaluaciones.

3. Motor de Evaluación Adaptativa: Procesar exitosamente un mínimo de 200 evaluaciones adaptativas durante el piloto. Alcanzar 95% de precisión en la estimación del nivel de competencia del estudiante. Reducir en 40% el tiempo promedio de evaluación comparado con métodos tradicionales y generar reportes de progreso individualizados para el 100% de estudiantes participantes.

4. Validación de Ciclo Completo: Ejecutar 5 ciclos completos de prueba: desde creación de curso hasta evaluación estudiantil. Lograr 100% de éxito en la trazabilidad de datos a través de todo el flujo del sistema y validar la integridad y consistencia de datos en todas las transacciones del sistema.

Análisis de Riesgos

Riesgos Técnicos (Probabilidad: Media | Impacto: Alto)

- **RT-01:** Complejidad algorítmica del motor adaptativo puede generar errores de precisión
 - *Mitigación:* Implementar pruebas A/B con algoritmos alternativos y validación estadística continua
- **RT-02:** Escalabilidad del sistema bajo carga concurrente
 - *Mitigación:* Pruebas de estrés progresivas y implementación de arquitectura de microservicios

Riesgos de Cronograma (Probabilidad: Alta | Impacto: Medio)

- **RC-01:** Curva de aprendizaje de tecnologías especializadas (IA, algoritmos adaptativos)

- *Mitigación:* Capacitación anticipada del equipo y consultoría especializada externa
- **RC-02:** Retrasos en validación con usuarios reales
 - *Mitigación:* Establecimiento de acuerdos tempranos con instituciones piloto

Riesgos de Adopción (Probabilidad: Media | Impacto: Alto)

- **RA-01:** Resistencia cultural al cambio tecnológico en instituciones tradicionales
 - *Mitigación:* Programa de acompañamiento y capacitación integral para docentes
- **RA-02:** Limitaciones de infraestructura tecnológica en instituciones objetivo
 - *Mitigación:* Desarrollo de versión optimizada para conexiones de baja velocidad

Riesgos Financieros (Probabilidad: Baja | Impacto: Medio)

- **RF-01:** Limitaciones presupuestarias para pruebas piloto extensas
 - *Mitigación:* Búsqueda de financiamiento a través de concursos de innovación y alianzas estratégicas

Limitaciones del Proyecto

Limitaciones Tecnológicas

- **L-01:** MVP disponible exclusivamente como aplicación web responsive, sin desarrollo de aplicaciones móviles nativas iOS/Android
- **L-02:** Motor de evaluación adaptativa operará inicialmente con un banco limitado de 500-1000 preguntas por materia piloto
- **L-03:** Capacidad máxima del sistema: 1000 usuarios registrados y 100 usuarios concurrentes

Limitaciones Funcionales

- **L-04:** Ausencia de integración con sistemas de pago y plataformas LMS existentes en la primera versión

- **L-05:** Reportes analíticos limitados a métricas básicas de rendimiento y progreso
- **L-06:** Interfaz inicial priorizará funcionalidad sobre elementos de diseño visual avanzado

Limitaciones Geográficas y de Mercado

- **L-07:** Pruebas piloto restringidas a instituciones educativas de la región de Ica
- **L-08:** Validación limitada a educación secundaria y superior, excluyendo educación primaria
- **L-09:** Soporte técnico disponible únicamente en horario laboral durante fase piloto

Supuestos Fundamentales

Supuestos Tecnológicos

- **S-01:** Las instituciones piloto disponen de infraestructura básica de conectividad (mínimo 2 Mbps) y equipos con navegadores web modernos
- **S-02:** Disponibilidad continua de servicios en la nube (AWS, Google Cloud) para hosting y almacenamiento de datos
- **S-03:** Estabilidad de las tecnologías del stack MERN durante el período de desarrollo (18 meses)

Supuestos de Participación

- **S-04:** Compromiso confirmado de al menos 3 instituciones educativas para participar como pilotos, con mínimo 150 estudiantes y 20 docentes
- **S-05:** Disponibilidad de docentes para dedicar 4 horas semanales durante 8 semanas para capacitación y pruebas
- **S-06:** Participación de coordinadores académicos en el proceso de validación y retroalimentación

Supuestos de Mercado

- **S-07:** Receptividad del mercado educativo peruano hacia soluciones SaaS de costo medio (entre plataformas gratuitas y sistemas empresariales)

- **S-08:** Alineación de los flujos académicos modelados (matrícula, asignación, evaluación) con los procesos estándar de instituciones educativas peruanas
- **S-09:** Creciente demanda de soluciones de personalización educativa post-pandemia mantendrá tendencia durante el desarrollo del proyecto

Supuestos Regulatorios

- **S-10:** Cumplimiento de normativas de protección de datos personales (Ley 29733) sin requerir certificaciones adicionales para el MVP
- **S-11:** No aparición de regulaciones restrictivas sobre el uso de IA en educación durante el período de desarrollo

Requerimientos Del Proyecto

Requerimientos Funcionales

Los requerimientos funcionales definen las funcionalidades específicas que debe realizar el sistema AdaptaTest para satisfacer las necesidades de los usuarios y cumplir con los objetivos del proyecto.

RF-01: Gestión de Usuarios y Autenticación

RF-01.1: Registro y Autenticación de Usuarios

- El sistema debe permitir el registro de usuarios con validación de email
- Debe implementar autenticación segura con encriptación de contraseñas
- Debe soportar recuperación de contraseñas mediante email
- Debe mantener sesiones de usuario con tokens de seguridad
- Debe permitir logout seguro y manejo de sesiones múltiples

RF-01.2: Gestión de Roles y Permisos

- El sistema debe implementar 5 roles diferenciados: Administrador, Coordinador, Profesor, Estudiante y Padre de Familia
- Cada rol debe tener permisos específicos y acceso restringido a funcionalidades
- Debe permitir la asignación y modificación de roles por administradores
- Debe validar permisos en tiempo real para todas las operaciones

RF-02: Módulo de Administración Institucional

RF-02.1: Gestión de Institución

- El sistema debe permitir crear y configurar perfiles de instituciones educativas
- Debe gestionar información institucional (nombre, dirección, logo, configuraciones)
- Debe permitir la configuración de períodos académicos y calendarios

RF-02.2: Gestión de Carreras y Cursos

- El sistema debe permitir crear, editar y eliminar carreras académicas
- Debe gestionar cursos por carrera con información detallada
- Debe permitir la organización jerárquica de contenidos académicos
- Debe gestionar prerequisitos y correlativas entre cursos

RF-03: Gestión Académica

RF-03.1: Gestión de Secciones

- El sistema debe permitir crear secciones por curso con cupos limitados
- Debe asignar profesores a secciones específicas
- Debe permitir la apertura y cierre de secciones según disponibilidad

RF-03.2: Sistema de Matrículas

- El sistema debe permitir la matrícula de estudiantes en secciones disponibles
- Debe validar prerequisitos y cupos disponibles
- Debe generar comprobantes de matrícula automáticamente
- Debe permitir la modificación y cancelación de matrículas dentro del plazo establecido

RF-04: Biblioteca de Contenido Educativo

RF-04.1: Gestión de Material Pedagógico

- El sistema debe permitir a los profesores subir y organizar material educativo
- Debe soportar múltiples formatos de archivo (PDF, videos, imágenes, documentos)
- Debe implementar un sistema de categorización y etiquetado de contenido

- Debe permitir la reutilización de material entre diferentes secciones

RF-04.2: Creación de Módulos y Lecciones

- El sistema debe permitir crear módulos temáticos con lecciones estructuradas
- Debe organizar el contenido en secuencias lógicas de aprendizaje
- Debe permitir la asociación de objetivos de aprendizaje por módulo
- Debe implementar un sistema de prerequisitos entre lecciones

RF-04.3: Banco de Preguntas

- El sistema debe permitir crear y gestionar bancos de preguntas por materia
- Debe categorizar preguntas por competencia, dificultad y tipo
- Debe permitir la importación masiva de preguntas desde archivos externos
- Debe implementar sistema de validación y revisión de preguntas por expertos

RF-05: Motor de Evaluación Adaptativa

RF-05.1: Algoritmo de Adaptación

- El sistema debe implementar un algoritmo que ajuste la dificultad de preguntas en tiempo real
- Debe calcular el nivel de maestría del estudiante basado en respuestas previas
- Debe seleccionar preguntas óptimas según la Teoría de Respuesta al Ítem (TRI)
- Debe mantener un balance entre exploración y explotación en la selección de preguntas

RF-05.2: Evaluaciones Dinámicas

- El sistema debe generar evaluaciones personalizadas para cada estudiante
- Debe adaptar la secuencia y dificultad basándose en el rendimiento en tiempo real
- Debe determinar automáticamente cuándo finalizar una evaluación basado en precisión alcanzada
- Debe proporcionar retroalimentación inmediata después de cada respuesta

RF-05.3: Medición de Competencias

- El sistema debe evaluar el nivel de maestría por competencias específicas
- Debe generar perfiles de conocimiento individualizados por estudiante
- Debe identificar fortalezas y áreas de oportunidad de manera automática
- Debe recalcular niveles de competencia después de cada evaluación

RF-06: Sistema de Reportes y Analíticas

RF-06.1: Reportes para Estudiantes

- El sistema debe generar reportes individuales de progreso académico
- Debe mostrar nivel de maestría por competencia con visualizaciones gráficas
- Debe proporcionar recomendaciones personalizadas de estudio
- Debe incluir comparaciones con el rendimiento promedio del grupo

RF-06.2: Reportes para Profesores

- El sistema debe generar reportes de rendimiento grupal por sección
- Debe identificar estudiantes con dificultades específicas por competencia
- Debe proporcionar analíticas sobre la efectividad de material pedagógico
- Debe incluir estadísticas de participación y tiempo dedicado por estudiante

RF-06.3: Reportes Institucionales

- El sistema debe generar reportes ejecutivos para coordinadores y administradores
- Debe incluir métricas de adopción y uso de la plataforma
- Debe proporcionar análisis comparativo entre carreras y cursos
- Debe generar indicadores de calidad educativa y satisfacción estudiantil

RF-07: Sistema de Notificaciones

RF-07.1: Notificaciones en Tiempo Real

- El sistema debe enviar notificaciones push sobre evaluaciones programadas
- Debe notificar cambios importantes en cursos y cronogramas

- Debe alertar sobre deadlines y fechas importantes
- Debe informar sobre nuevos contenidos y materiales disponibles

RF-07.2: Comunicación Multi-Canal

- El sistema debe enviar notificaciones por email y plataforma web
- Debe permitir configuración de preferencias de notificación por usuario
- Debe implementar notificaciones para padres sobre el progreso de estudiantes
- Debe incluir recordatorios automáticos para evaluaciones pendientes

RF-08: Interfaz de Usuario Diferenciada por Rol

RF-08.1: Dashboard del Estudiante

- El sistema debe mostrar un dashboard personalizado con progreso académico
- Debe listar evaluaciones pendientes y resultados recientes
- Debe mostrar recomendaciones de estudio personalizadas
- Debe incluir acceso directo a materiales de estudio y recursos

RF-08.2: Dashboard del Profesor

- El sistema debe mostrar todas las secciones asignadas al profesor
- Debe permitir acceso rápido a herramientas de creación de contenido
- Debe mostrar analíticas en tiempo real del rendimiento estudiantil
- Debe incluir herramientas de comunicación con estudiantes

RF-08.3: Dashboard Administrativo

- El sistema debe proporcionar vista general del estado institucional
- Debe mostrar métricas clave de uso y rendimiento de la plataforma
- Debe incluir herramientas de configuración y gestión de usuarios
- Debe permitir monitoreo en tiempo real de la actividad del sistema

Requerimientos No Funcionales

Los requerimientos no funcionales definen las características de calidad que debe cumplir el sistema en términos de rendimiento, seguridad, usabilidad y otros atributos técnicos.

RNF-01: Rendimiento y Escalabilidad

RNF-01.1: Tiempo de Respuesta

- El sistema debe responder a consultas simples en menos de 2 segundos
- Las evaluaciones adaptativas deben cargarse en menos de 3 segundos
- Los reportes complejos deben generarse en menos de 10 segundos
- La navegación entre páginas debe ser fluida con tiempos menores a 1 segundo

RNF-01.2: Capacidad de Usuarios Concurrentes

- El sistema debe soportar mínimo 500 usuarios concurrentes sin degradación
- Debe escalar horizontalmente para manejar picos de tráfico
- El rendimiento debe mantenerse estable con base de datos de hasta 100,000 registros
- Debe optimizar consultas para mantener tiempos de respuesta constantes

RNF-01.3: Disponibilidad del Servicio

- El sistema debe mantener un uptime mínimo del 99.5%
- Debe implementar mecanismos de recuperación automática ante fallos
- El tiempo de inactividad planificado no debe exceder 2 horas mensuales
- Debe incluir sistemas de backup y recuperación de desastres

RNF-02: Seguridad

RNF-02.1: Autenticación y Autorización

- El sistema debe implementar autenticación multifactor para roles administrativos
- Debe usar tokens JWT con expiración automática para sesiones
- Las contraseñas deben cumplir políticas de complejidad (mínimo 8 caracteres, mayúsculas, números, símbolos)
- Debe implementar protección contra ataques de fuerza bruta con bloqueo temporal

RNF-02.2: Protección de Datos

- Todos los datos sensibles deben estar encriptados en tránsito (TLS 1.3)

- Los datos en reposo deben usar encriptación AES-256
- Debe cumplir con la Ley de Protección de Datos Personales (Ley 29733) del Perú
- Debe implementar anonimización de datos para reportes estadísticos

RNF-02.3: Auditoría y Trazabilidad

- El sistema debe registrar todas las acciones críticas con timestamps
- Debe mantener logs de acceso y modificaciones por usuario
- Los registros de auditoría deben ser inmutables y respaldados
- Debe permitir trazabilidad completa de evaluaciones y calificaciones

RNF-03: Usabilidad

RNF-03.1: Experiencia de Usuario

- El sistema debe alcanzar una puntuación mínima de 70 en la escala SUS
- La curva de aprendizaje para nuevos usuarios no debe exceder 2 horas
- Debe ser intuitivo para usuarios con conocimientos básicos de computación
- El 90% de tareas comunes debe realizarse en máximo 3 clics

RNF-03.2: Accesibilidad

- El sistema debe cumplir con estándares WCAG 2.1 nivel AA
- Debe ser compatible con lectores de pantalla para usuarios con discapacidades
- Debe soportar navegación por teclado para todas las funcionalidades
- Los contrastes de color deben cumplir estándares de accesibilidad

RNF-03.3: Compatibilidad de Dispositivos

- El sistema debe ser completamente responsive para dispositivos móviles
- Debe ser compatible con navegadores Chrome, Firefox, Safari y Edge (últimas 2 versiones)
- Debe funcionar correctamente en resoluciones desde 320px hasta 4K
- Debe optimizarse para tablets y dispositivos táctiles

RNF-04: Mantenibilidad

RNF-04.1: Arquitectura Modular

- El código debe seguir principios SOLID y patrones de diseño establecidos
- Debe implementar arquitectura de microservicios para facilitar mantenimiento
- El sistema debe tener cobertura de pruebas automatizadas mínima del 80%
- Debe incluir documentación técnica actualizada y comprensible

RNF-04.2: Monitoreo y Diagnóstico

- El sistema debe incluir herramientas de monitoreo en tiempo real
- Debe generar alertas automáticas ante anomalías o errores críticos
- Debe incluir métricas de rendimiento y uso para optimización continua
- Debe facilitar diagnóstico rápido de problemas técnicos

CASOS DE USO

ACTORES DEL SISTEMA

Actor 1: ESTUDIANTE (Student)

Descripción: Usuario que se matricula en cursos, consume contenido educativo, realiza evaluaciones y realiza entregas de tareas.

Características:

- Puede pertenecer a universidad o colegio
- Visualiza su progreso académico
- Interactúa con contenido adaptativo

Actor 2: PROFESOR (Professor)

Descripción: Usuario que crea contenido educativo, gestiona módulos, evalúa a estudiantes y monitorea el progreso de sus secciones.

Características:

- Crea y gestiona su biblioteca personal de módulos
- Publica contenido en secciones asignadas
- Califica tareas y retroalimenta estudiantes

Actor 3: COORDINADOR (Coordinator)

Descripción: Usuario que gestiona carreras académicas y mallas curriculares (solo universidades).

Características:

- Administra la malla curricular de su carrera
 - Crea y configura cursos
 - Supervisa la estructura académica
-

Actor 4: ADMINISTRADOR (Admin)

Descripción: Usuario con privilegios completos dentro de su institución.

Características:

- Gestiona usuarios de la institución
 - Crea carreras, cursos y ciclos académicos
 - Asigna coordinadores e instructores
 - Procesa calificaciones finales
-

Actor 5: PADRE DE FAMILIA (Parent)

Descripción: Usuario que supervisa el progreso de sus hijos (solo colegios).

Características:

- Visualiza calificaciones de sus hijos
 - Recibe notificaciones de rendimiento
 - Consulta entregas de tareas
-

CASOS DE USO POR ACTOR

ESTUDIANTE

Módulo de Autenticación

- **CU-001:** Iniciar Sesión

- **CU-002:** Cerrar Sesión
- **CU-003:** Ver Perfil

Módulo de Matrícula

- **CU-004:** Inscribirse en Carrera (solo universidades)
- **CU-005:** Ver Progreso Académico
- **CU-006:** Ver Cursos Elegibles
- **CU-007:** Matricularse en Sección
- **CU-008:** Matricularse en Lote (Carrito)
- **CU-009:** Ver Mis Matrículas
- **CU-010:** Ver Historial Académico

Módulo de Contenido Educativo

- **CU-011:** Ver Módulos de una Sección
- **CU-012:** Ver Lecciones de un Módulo
- **CU-013:** Marcar Lección como Completada
- **CU-014:** Descargar Material de Apoyo

Módulo de Evaluaciones Adaptativas

- **CU-015:** Iniciar Evaluación de Módulo
- **CU-016:** Responder Pregunta en Evaluación
- **CU-017:** Ver Resultado de Evaluación
- **CU-018:** Ver Mi Maestría por Módulo

Módulo de Tareas

- **CU-019:** Ver Tareas de una Sección
- **CU-020:** Entregar Tarea
- **CU-021:** Ver Mi Entrega
- **CU-022:** Ver Calificación y Retroalimentación

PROFESOR

Módulo de Biblioteca Personal

- **CU-023:** Crear Módulo en Biblioteca
- **CU-024:** Ver Mis Módulos
- **CU-025:** Editar Módulo
- **CU-026:** Eliminar Módulo
- **CU-027:** Crear Lección en Módulo
- **CU-028:** Editar Lección
- **CU-029:** Eliminar Lección
- **CU-030:** Crear Pregunta en Módulo
- **CU-031:** Editar Pregunta
- **CU-032:** Eliminar Pregunta
- **CU-033:** Ver Banco de Preguntas

Módulo de Gestión de Secciones

- **CU-034:** Ver Mis Secciones
- **CU-035:** Publicar Módulo en Sección
- **CU-036:** Ver Módulos Publicados
- **CU-037:** Configurar Criterios de Aprobación
- **CU-038:** Ver Estudiantes Matriculados

Módulo de Tareas

- **CU-039:** Crear Tarea en Sección
- **CU-040:** Ver Tareas de una Sección
- **CU-041:** Ver Entregas de una Tarea
- **CU-042:** Calificar Entrega
- **CU-043:** Dar Retroalimentación

Módulo de Analytics

- **CU-044:** Ver Analíticas de Sección
- **CU-045:** Ver Estudiantes con Dificultades
- **CU-046:** Ver Preguntas Más Difíciles
- **CU-047:** Ver Maestría Promedio por Módulo

- **CU-048:** Ver Vista Previa de Calificaciones
-

COORDINADOR

Módulo de Gestión de Cursos

- **CU-049:** Crear Curso
- **CU-050:** Ver Cursos de la Institución
- **CU-051:** Ver Detalles de Curso
- **CU-052:** Subir Sílabo de Curso

Módulo de Gestión de Carreras

- **CU-053:** Ver Mi Carrera Asignada
- **CU-054:** Añadir Curso a Malla Curricular
- **CU-055:** Ver Malla Curricular Completa

Módulo de Gestión de Secciones

- **CU-056:** Crear Sección para Curso
 - **CU-057:** Ver Secciones de un Curso
-

ADMINISTRADOR

Módulo de Gestión de Usuarios

- **CU-058:** Registrar Usuario
- **CU-059:** Ver Usuarios de la Institución
- **CU-060:** Filtrar Usuarios por Rol

Módulo de Gestión de Carreras

- **CU-061:** Crear Carrera
- **CU-062:** Ver Todas las Carreras
- **CU-063:** Asignar Coordinador a Carrera
- **CU-064:** Ver Detalles de Carrera

Módulo de Ciclos Académicos

- **CU-065:** Crear Ciclo Académico

- **CU-066:** Ver Ciclos Académicos
- **CU-067:** Activar Ciclo Académico

Módulo de Calificaciones

- **CU-068:** Ver Vista Previa de Calificaciones de Sección
 - **CU-069:** Procesar Calificaciones Finales de Sección
-

PADRE DE FAMILIA

Módulo de Supervisión

- **CU-070:** Ver Hijos Asignados
 - **CU-071:** Ver Calificaciones de Hijo
 - **CU-072:** Ver Progreso Académico de Hijo
 - **CU-073:** Ver Tareas Pendientes de Hijo
-

DESCRIPCIÓN DETALLADA DE CASOS DE USO

CU-001: Iniciar Sesión

Actor Principal: Todos los actores

Precondiciones:

- El usuario debe tener una cuenta registrada en el sistema
- La institución debe estar activa

Flujo Principal:

1. El usuario accede a la página de inicio de sesión
2. El sistema solicita: email, contraseña e institución
3. El usuario ingresa sus credenciales
4. El sistema valida las credenciales
5. El sistema verifica que la institución esté activa
6. El sistema genera un token JWT
7. El sistema redirige al usuario al dashboard según su rol

Flujos Alternativos:

- **FA-001:** Si las credenciales son incorrectas, el sistema muestra un mensaje de error
- **FA-002:** Si la institución no está activa, el sistema muestra mensaje de institución inactiva

Postcondiciones:

- El usuario está autenticado en el sistema
 - Se genera un token de sesión válido por 30 días
-

CU-007: Matricularse en Sección

Actor Principal: Estudiante

Precondiciones:

- El usuario debe estar autenticado como estudiante
- Debe existir un ciclo académico activo
- Debe haber secciones disponibles

Flujo Principal:

1. El estudiante accede al módulo de matrícula
2. El sistema muestra las secciones disponibles del ciclo activo
3. El estudiante selecciona una sección
4. El sistema valida que el estudiante no esté ya matriculado
5. El sistema verifica que hay capacidad disponible
6. El sistema valida los prerequisitos (solo universidades)
7. El sistema crea la matrícula con estado "enrolled"
8. El sistema muestra confirmación de matrícula exitosa

Flujos Alternativos:

- **FA-001:** Si ya está matriculado, muestra mensaje de error
- **FA-002:** Si no hay capacidad, muestra mensaje "Sin vacantes"
- **FA-003:** Si no cumple prerequisitos, muestra cursos faltantes
- **FA-004:** Si la validación falla, cancela la operación

Postcondiciones:

- Se crea un registro en ENROLLMENTS con status "enrolled"
- El contador de matrículas de la sección aumenta

Reglas de Negocio:

- RN-001: Un estudiante no puede matricularse dos veces en la misma sección
 - RN-002: La sección debe tener capacidad disponible
 - RN-003: El estudiante debe haber aprobado los prerequisitos (universidades)
-

CU-008: Matricularse en Lote (Carrito)

Actor Principal: Estudiante

Precondiciones:

- El usuario debe estar autenticado como estudiante
- Debe existir un ciclo académico activo

Flujo Principal:

1. El estudiante accede al módulo de matrícula por lote
2. El sistema muestra cursos elegibles según su progreso
3. El estudiante selecciona múltiples secciones (carrito)
4. El estudiante confirma la matrícula
5. El sistema ejecuta validaciones para TODAS las secciones:
 - Capacidad disponible
 - Prerrequisitos cumplidos
 - No duplicados
6. Si todas las validaciones pasan, el sistema crea todas las matrículas
7. El sistema muestra confirmación de matrícula exitosa

Flujos Alternativos:

- **FA-001:** Si alguna validación falla, NO se crea ninguna matrícula
- **FA-002:** El sistema muestra lista de errores específicos por sección

- **FA-003:** El estudiante puede modificar su selección y reintentar

Postcondiciones:

- Se crean múltiples registros en ENROLLMENTS (todo o nada)
- Las capacidades de las secciones se actualizan

Reglas de Negocio:

- RN-001: Todas las validaciones deben pasar antes de crear matrículas
 - RN-002: Es una operación atómica (todo o nada)
-

CU-015: Iniciar Evaluación de Módulo

Actor Principal: Estudiante

Precondiciones:

- El usuario debe estar autenticado como estudiante
- El módulo debe estar publicado en una sección donde esté matriculado
- Debe haber preguntas disponibles en el módulo

Flujo Principal:

1. El estudiante accede a un módulo
2. El estudiante hace clic en "Iniciar Evaluación"
3. El sistema crea una sesión de evaluación con:
 - student: ID del estudiante
 - module: ID del módulo
 - currentMastery: 25 (inicial)
 - status: "in_progress"
4. El sistema selecciona la primera pregunta según maestría (dificultad 2)
5. El sistema muestra la pregunta al estudiante

Flujos Alternativos:

- **FA-001:** Si no hay preguntas disponibles, muestra mensaje de error

Postcondiciones:

- Se crea un registro en EVALUATION_SESSIONS
 - El estudiante puede comenzar a responder preguntas
-

CU-016: Responder Pregunta en Evaluación**Actor Principal:** Estudiante**Precondiciones:**

- Debe existir una sesión de evaluación activa
- La sesión debe estar en estado "in_progress"

Flujo Principal:

1. El estudiante selecciona una opción de respuesta
2. El estudiante confirma su respuesta
3. El sistema valida si la respuesta es correcta
4. Si es correcta:
 - currentMastery += difficulty * 2
 - score.correct += 1
5. Si es incorrecta:
 - currentMastery -= (6 - difficulty)
 - score.incorrect += 1
6. El sistema limita currentMastery entre 0 y 100
7. El sistema registra el intento en PERFORMANCE_LOGS
8. El sistema agrega la pregunta a questionsAnswered
9. El sistema verifica condiciones de finalización:
 - Si questionsAnswered.length >= 10, finaliza
 - Si currentMastery >= 95, finaliza
10. Si no finaliza, selecciona la siguiente pregunta según nueva maestría
11. El sistema muestra la siguiente pregunta

Flujos Alternativos:

- **FA-001:** Si la sesión finaliza:
 - Cambiar status a "completed"
 - Actualizar o crear registro en MASTERY con el puntaje más alto
 - Mostrar resumen de resultados

Postcondiciones:

- Se actualiza la sesión de evaluación
- Se crea un registro en PERFORMANCE_LOGS
- Puede actualizarse el registro en MASTERY

Algoritmo de Selección de Pregunta:

targetDifficulty = Math.ceil(currentMastery / 20)

targetDifficulty = Math.min(Math.max(targetDifficulty, 1), 5)

Buscar pregunta con difficulty = targetDifficulty

Si no existe, buscar difficulty \pm 1

CU-023: Crear Módulo en Biblioteca

Actor Principal: Profesor

Precondiciones:

- El usuario debe estar autenticado como profesor

Flujo Principal:

1. El profesor accede a "Mi Biblioteca"
2. El profesor hace clic en "Crear Módulo"
3. El sistema muestra formulario con campos:
 - Título (requerido)
 - Descripción (opcional)
4. El profesor completa el formulario
5. El profesor confirma la creación
6. El sistema crea el módulo con:

- owner: ID del profesor
- institution: ID de la institución del profesor
- publishedIn: [] (vacío inicialmente)
- order: 1

7. El sistema muestra confirmación de creación exitosa

Postcondiciones:

- Se crea un registro en MODULES
 - El módulo aparece en la biblioteca del profesor
-

CU-027: Crear Lección en Módulo

Actor Principal: Profesor

Precondiciones:

- El usuario debe estar autenticado como profesor
- El módulo debe existir y pertenecer al profesor

Flujo Principal:

1. El profesor accede a un módulo de su biblioteca
2. El profesor hace clic en "Añadir Lección"
3. El sistema muestra formulario con:
 - Título (requerido)
 - Tipo de contenido (text/video_url/document_url/slides_url)
 - Contenido o URL (según tipo)
 - Archivo adjunto (opcional)
4. El profesor completa el formulario
5. El profesor confirma la creación
6. El sistema calcula el orden (número de lecciones + 1)
7. El sistema crea la lección
8. El sistema muestra confirmación

Flujos Alternativos:

- **FA-001:** Si el tipo es "text", valida que haya contenido Markdown
- **FA-002:** Si es otro tipo, valida que haya URL válida

Postcondiciones:

- Se crea un registro en LESSONS
 - La lección aparece en el módulo en el orden correcto
-

CU-030: Crear Pregunta en Módulo

Actor Principal: Profesor

Precondiciones:

- El usuario debe estar autenticado como profesor
- El módulo debe existir y pertenecer al profesor

Flujo Principal:

1. El profesor accede al banco de preguntas de un módulo
2. El profesor hace clic en "Crear Pregunta"
3. El sistema muestra formulario con:
 - Texto de la pregunta (requerido)
 - Tipo (multiple_choice/true_false)
 - Dificultad (1-5)
 - Opciones (mínimo 2):
 - Texto de la opción
 - ¿Es correcta? (checkbox)
4. El profesor completa el formulario
5. El profesor marca al menos una opción como correcta
6. El profesor confirma la creación
7. El sistema valida que haya al menos una respuesta correcta
8. El sistema crea la pregunta
9. El sistema muestra confirmación

Flujos Alternativos:

- **FA-001:** Si no hay respuesta correcta, muestra error de validación
- **FA-002:** Si hay menos de 2 opciones, muestra error

Postcondiciones:

- Se crea un registro en QUESTIONS
 - La pregunta está disponible para evaluaciones adaptativas
-

CU-035: Publicar Módulo en Sección

Actor Principal: Profesor

Precondiciones:

- El usuario debe estar autenticado como profesor
- El módulo debe existir y pertenecer al profesor
- El profesor debe ser instructor de la sección

Flujo Principal:

1. El profesor accede a una de sus secciones
2. El profesor hace clic en "Añadir Módulo"
3. El sistema muestra la biblioteca del profesor
4. El profesor selecciona un módulo
5. El profesor confirma la publicación
6. El sistema valida que el profesor es el instructor
7. El sistema valida que el módulo le pertenece
8. El sistema añade la sección al array publishedIn del módulo
9. El sistema muestra confirmación

Postcondiciones:

- El módulo se agrega a publishedIn del módulo
 - Los estudiantes de la sección pueden ver el módulo
-

CU-039: Crear Tarea en Sección

Actor Principal: Profesor

Precondiciones:

- El usuario debe estar autenticado como profesor
- Debe ser instructor de la sección

Flujo Principal:

1. El profesor accede a una de sus secciones
2. El profesor hace clic en "Crear Tarea"
3. El sistema muestra formulario con:
 - Título (requerido)
 - Instrucciones (opcional)
 - Fecha de vencimiento (opcional)
 - Puntaje posible (default: 100)
4. El profesor completa el formulario
5. El profesor confirma la creación
6. El sistema valida que sea el instructor
7. El sistema crea la tarea
8. El sistema muestra confirmación

Postcondiciones:

- Se crea un registro en ASSIGNMENTS
 - Los estudiantes pueden ver y entregar la tarea
-

CU-042: Calificar Entrega

Actor Principal: Profesor

Precondiciones:

- El usuario debe estar autenticado como profesor
- Debe existir una entrega de un estudiante
- Debe ser instructor de la sección

Flujo Principal:

1. El profesor accede a las entregas de una tarea
2. El profesor selecciona una entrega

3. El sistema muestra el contenido de la entrega
4. El profesor ingresa:
 - o Calificación (0-100)
 - o Retroalimentación (texto)
5. El profesor confirma la calificación
6. El sistema actualiza la entrega
7. El sistema muestra confirmación

Postcondiciones:

- Se actualiza el registro en SUBMISSIONS con grade y feedback
 - El estudiante puede ver su calificación
-

CU-054: Añadir Curso a Malla Curricular

Actor Principal: Coordinador

Precondiciones:

- El usuario debe estar autenticado como coordinador
- Debe tener una carrera asignada
- El curso debe existir

Flujo Principal:

1. El coordinador accede a "Mi Carrera"
2. El coordinador hace clic en "Añadir Curso a Malla"
3. El sistema muestra formulario con:
 - o Lista de cursos disponibles
 - o Número de ciclo (1-20)
4. El coordinador selecciona un curso y ciclo
5. El coordinador confirma
6. El sistema valida que el curso no esté ya en la malla
7. El sistema valida que el ciclo sea mayor que el de los prerequisitos
8. El sistema añade el curso al ciclo correspondiente

9. El sistema muestra confirmación

Flujos Alternativos:

- **FA-001:** Si el curso ya está en la malla, muestra error
- **FA-002:** Si el ciclo es inválido (\leq al de prerequisitos), muestra error detallado

Postcondiciones:

- Se actualiza el array curriculum de la carrera
 - El curso aparece en el ciclo correspondiente de la malla
-

CU-068: Ver Vista Previa de Calificaciones de Sección

Actor Principal: Administrador, Profesor

Precondiciones:

- El usuario debe estar autenticado
- La sección debe tener criterios de aprobación configurados

Flujo Principal:

1. El usuario accede a una sección
2. El usuario hace clic en "Vista Previa de Calificaciones"
3. El sistema obtiene todos los estudiantes matriculados
4. Para cada estudiante, el sistema verifica:
 - **Maestría:** Si superan el % mínimo en todos los módulos
 - **Tareas:** Si entregaron todas las tareas requeridas
 - **Lecciones:** Si completaron el % requerido
5. El sistema muestra una tabla con:
 - Nombre del estudiante
 - Estado de cada criterio (\checkmark o X)
 - Estado final proyectado
6. El sistema calcula estadísticas generales

Postcondiciones:

- Se muestra un reporte sin modificar datos

- Los administradores pueden decidir si procesar
-

CU-069: Procesar Calificaciones Finales de Sección

Actor Principal: Administrador, Profesor

Precondiciones:

- El usuario debe estar autenticado
- La sección debe tener criterios configurados
- Debe haber estudiantes matriculados

Flujo Principal:

1. El usuario accede a una sección
2. El usuario hace clic en "Procesar Calificaciones Finales"
3. El sistema muestra advertencia de confirmación
4. El usuario confirma el procesamiento
5. Para cada estudiante matriculado (status: "enrolled"):
 - El sistema verifica todos los criterios de aprobación
 - Si cumple todos: actualiza status a "passed"
 - Si no cumple alguno: actualiza status a "failed"
6. El sistema guarda todos los cambios
7. El sistema genera reporte de resultados
8. El sistema muestra resumen con:
 - Total procesados
 - Aprobados
 - Desaprobados
 - Detalle por estudiante

Flujos Alternativos:

- **FA-001:** Si el usuario cancela, no se procesa nada

Postcondiciones:

- Se actualizan los status de ENROLLMENTS

- Los estudiantes ven sus resultados finales en su historial

Reglas de Negocio:

- RN-001: Un estudiante aprueba solo si cumple TODOS los criterios configurados
 - RN-002: La operación es irreversible (cambio de status)
-

MATRIZ DE TRAZABILIDAD

Caso de Uso	Colecciones Afectadas	Validaciones Principales
CU-001	USERS, INSTITUTIONS	Email único por institución, institución activa
CU-007	ENROLLMENTS, SECTIONS	Capacidad, prerequisitos, no duplicados
CU-008	ENROLLMENTS, SECTIONS	Validación atómica múltiple
CU-015	EVALUATION_SESSIONS, MODULES	Módulo publicado, preguntas disponibles
CU-016	EVALUATION_SESSIONS, PERFORMANCE_LOGS, MASTERY	Algoritmo adaptativo
CU-023	MODULES	Ownership del profesor
CU-027	LESSONS, MODULES	Ownership del módulo
CU-030	QUESTIONS, MODULES	Respuesta correcta obligatoria
CU-035	MODULES (publishedIn)	Instructor de sección, owner de módulo
CU-039	ASSIGNMENTS, SECTIONS	Instructor de sección
CU-042	SUBMISSIONS	Instructor, entrega existente
CU-054	CAREERS (curriculum), COURSES	Validación de prerequisitos en malla

CU-069	ENROLLMENTS	Criterios de aprobación configurados
--------	-------------	--------------------------------------

Lean Canvas

Lean Canvas



PMBOK

Project Charter:

Project Charter							
Título	Plataforma de Gestión de Aprendizaje Adaptativo	Jefe de Proyecto	Aldair Doloriert				
Fecha inicio	14/08/2025	Fecha fin	2/12/2025	Sponsor(Patrocinadora)			
Necesidad del negocio							
Las instituciones educativas, desde colegios hasta academias pre-universitarias, enfrentan una ineficiencia sistemática en sus modelos de enseñanza, tanto presenciales como virtuales. El problema central es la falta de personalización del proceso de evaluación y refuerzo académico a escala.							
Alcance		Entregables					
Sistema web con tecnologías HTML, CSS y JavaScript.		Interfaz responsive para web y móvil.					
Plataforma de aprendizaje adaptativo.		Modulos para subir lecciones y evaluaciones.					
Modulos de gestión académica para roles administrativos.		Modulo de administrador para crear secciones.					
Modulos de gestión de contenido para maestros		Base de datos en MongoDB para guardar el					
Riesgos y problemas		Suposiciones y dependencias					
Posible falta de experiencia técnica en algunas fases del desarrollo.		Disponibilidad continua de servicios en la nube (AWS, Google Cloud) para hosting y almacenamiento de datos. Participación de					
Baja adopción por parte de los clientes debido a su posible complejidad.							
Costos							
El costo incluirá el desarrollo del sistema web, la configuración de la base de datos y el alojamiento de los servidores web, asimismo este se estimará en base al mercado local y el tiempo de desarrollo.							
Cronograma							
Hitos y Actividades		Responsable	Fecha Inicio	Fecha Fin			
1- Planificación del Proyecto (Misión, Visión, Etc		Aldair	12/08/2025	18/08/2025			
2- Reunión inicial con el cliente		Ambos	19/08/2025	20/08/2025			
3- Requerimientos funcionales y no funcionales.		Ambos	21/08/2025	27/08/2025			
4- Diseño de la arquitectura de software.		Ambos	28/08/2025	17/09/2025			
5- Pruebas y depuración.		Ambos	18/09/2025	1/11/2025			
Equipo		Comité de Aprobación					
Jefe de Proyecto	Joseph Aldair Doloriert Tasayco	Sponsor(Patrón)	UTP				

WBS:

1. Planificación del Proyecto

- 1.1 Recolección de requerimientos
 - 1.2 Definición de objetivos y alcance
 - 1.3 Cronograma inicial

2. Diseño de la Plataforma

- 2.1 Diseño de arquitectura del sistema
 - 2.2 Diseño de base de datos
 - 2.3 Diseño UI/UX

3. Desarrollo

- 3.1 Backend (API, lógica de negocio, evaluaciones adaptativas)
 - 3.2 Frontend (interfaz web/móvil)
 - 3.3 Módulo de gestión de contenidos
 - 3.4 Módulo de evaluaciones adaptativas
 - 3.5 Módulo administrativo

4. Pruebas

- 4.1 Pruebas unitarias
 - 4.2 Pruebas de integración
 - 4.3 Pruebas de aceptación de usuarios

5. Implementación

- 5.1 Configuración del servidor
 - 5.2 Despliegue de la plataforma
 - 5.3 Capacitación a docentes y administradores

6. Mantenimiento y Soporte

- ## 6.1 Actualizaciones

6.2 Soporte técnico

Cronograma de Actividades GANT:

Prototipado:

AdaptaTest

Login

Adapta Test

ADAPTANDO LA FORMA DE APRENDER

Nuestros Socios

Conoce a los desarrolladores

Las personas que hicieron posible adaptar tu aprendizaje

Aldair Doloriert
Backend

Rodrigo Trigoso
Frontend

#	PRODUCT	COMPANY	RESOURCES
© 2025 Adapta Test. All rights reserved.	Features	About Us	Blog
	Pricing	Privacy Policy	Changelog
	Testimonials	Terms of Services	Brand
	Integration		Help

Social Links

Iniciar Sesión

Correo Electrónico

Contraseña

#

© 2025 Adapta Test. All rights reserved.

PRODUCT

[Features](#)

[Pricing](#)

[Testimonials](#)

[Integration](#)

COMPANY

[About Us](#)

[Privacy Policy](#)

[Terms of Services](#)

RESOURCES

[Blog](#)

[Changelog](#)

[Brand](#)

[Help](#)

Social Links



Crear una cuenta

Nombre Completo

Fecha de Nacimiento

Carrera

Correo Electrónico

Contraseña

Registrarse



© 2025 Adapta Test. All rights reserved.

PRODUCT

[Features](#)

[Pricing](#)

[Testimonials](#)

[Integration](#)

COMPANY

[About Us](#)

[Privacy Policy](#)

[Terms of Services](#)

RESOURCES

[Blog](#)

[Changelog](#)

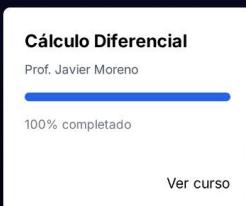
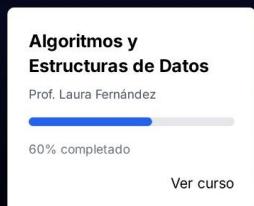
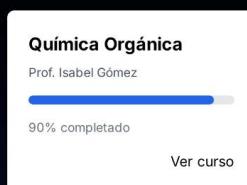
[Brand](#)

[Help](#)

Social Links



Ciclos



#

© 2025 Adapta Test. All rights reserved.

PRODUCT

[Features](#)[Pricing](#)[Testimonials](#)[Integration](#)

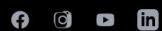
COMPANY

[About Us](#)[Privacy Policy](#)[Terms of Services](#)

RESOURCES

[Blog](#)[Changelog](#)[Brand](#)[Help](#)

Social Links



Matemáticas I

Prof. Ana López

75% completado

Módulos

Módulo 1: Introducción a los Conjuntos



Módulo 2: Operaciones con Conjuntos



Módulo 3: Lógica Proposicional



Módulo 4: Tablas de Verdad



Examen Parcial 1

Este examen cubre los módulos del 1 al 4. Asegúrate de haber completado los módulos antes de empezar.

[Iniciar Examen](#)

Módulo 5: Funciones y Relaciones



Módulo 6: Tipos de Funciones



Módulo 7: Álgebra de Funciones



Módulo 8: Límites y Continuidad



Examen Parcial 2

Este examen cubre los módulos del 5 al 8. Asegúrate de haber completado los módulos antes de empezar.

[Iniciar Examen \(Bloqueado\)](#)

AdaptaTest

© 2025 Adapta Test. All rights reserved.

PRODUCT

Features

COMPANY

About Us

RESOURCES

Blog

Pricing

Privacy Policy

Changelog

Testimonials

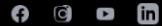
Terms of Services

Brand

Integration

Help

Social Links



Examen Parcial 1

Pregunta 1 de 10

¿Cuál de las siguientes operaciones con conjuntos resulta en el conjunto de todos los elementos que están en A, o en B, o en ambos?

- Intersección ($A \cap B$)
- Unión ($A \cup B$)
- Diferencia ($A - B$)
- Complemento (A')

[Siguiente Pregunta →](#)**AdaptaTest**

© 2025 Adapta Test. All rights reserved.

PRODUCT

- [Features](#)
- [Pricing](#)
- [Testimonials](#)
- [Integration](#)

COMPANY

- [About Us](#)
- [Privacy Policy](#)
- [Terms of Services](#)

RESOURCES

- [Blog](#)
- [Changelog](#)
- [Brand](#)
- [Help](#)

Social Links



Descripción de procesos:

Diagrama de proceso Bizagi:

Registro de usuario

Objetivo: Registrar usuarios y habilitarlos para acceder al sistema.

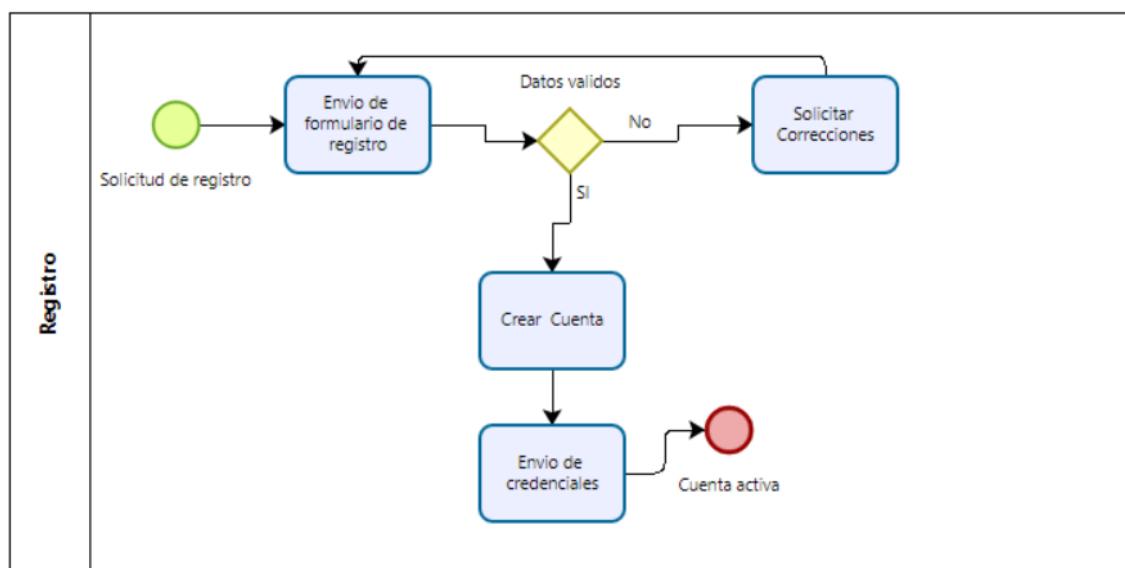
Actores: Estudiante / Padre, Administrador (o Secretaría), Soporte TI

Alcance: Desde la solicitud de acceso hasta la entrega de credenciales.

Paso a paso (AS-IS):

1. El usuario (estudiante/profesor/padre) solicita acceso mediante un formulario físico o en línea. (Inicio)
2. La Secretaría/Admin recibe la solicitud y revisa los datos manualmente.
3. ¿Los datos están completos y válidos? (Decisión)
 - Sí sí: el administrador crea la cuenta en el sistema
 - Si no: se solicita al usuario correcciones o envío de documentos complementarios.
4. El administrador envía por correo las credenciales / instrucción de primer acceso.
5. Fin: Usuario activo.

Problemas observados (AS-IS): revisión manual, tiempos de respuesta largos.



Creación de Cursos y Secciones

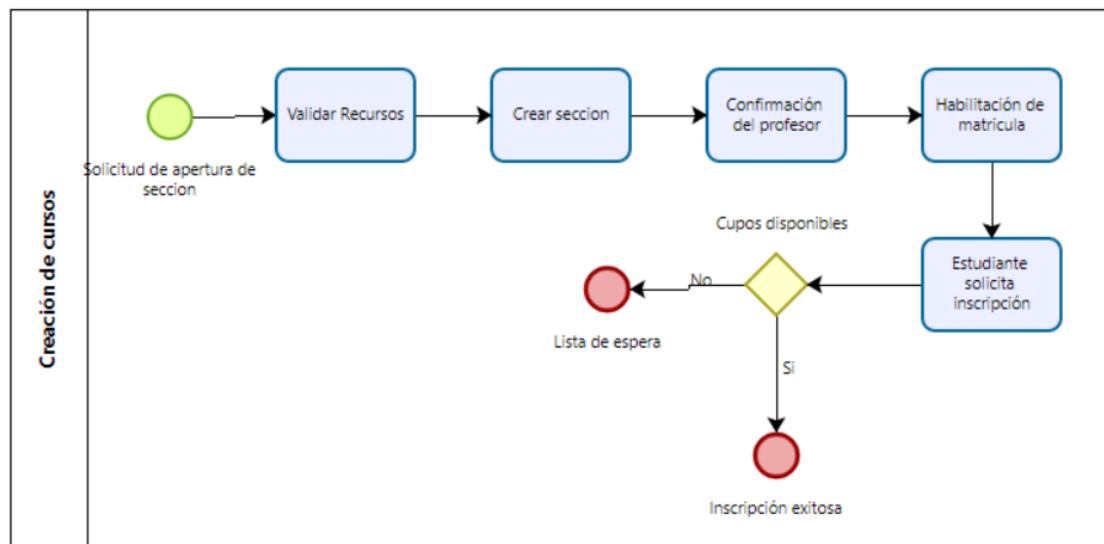
Objetivo: Crear cursos y secciones; matricular estudiantes.

Actores: Coordinador Académico, Profesor, Secretaría/Admisiones, Estudiante

Puntos críticos: revisión manual de prerequisitos, propenso a errores humanos.

Paso a paso (AS-IS):

1. El Coordinador solicita apertura de curso/sección para un periodo (vía correo o formulario interno).
2. Secretaría valida la disponibilidad de recursos (aula, profesor) y crea la sección en el sistema o en planilla.
3. Profesor asignado confirma carga horaria y material inicial.
4. Abre período de matrícula: los estudiantes solicitan inscripción (presencial o por portal web simple).
5. Secretaría procesa matrículas verificando prerequisitos y cupos.
6. Si hay cupo disponible: se confirma matrícula y se emite comprobante (manual o automático si el sistema lo soporta).
7. Si no: se coloca en lista de espera; se notifica al estudiante.
8. Fin: Sección abierta con lista final de inscritos.



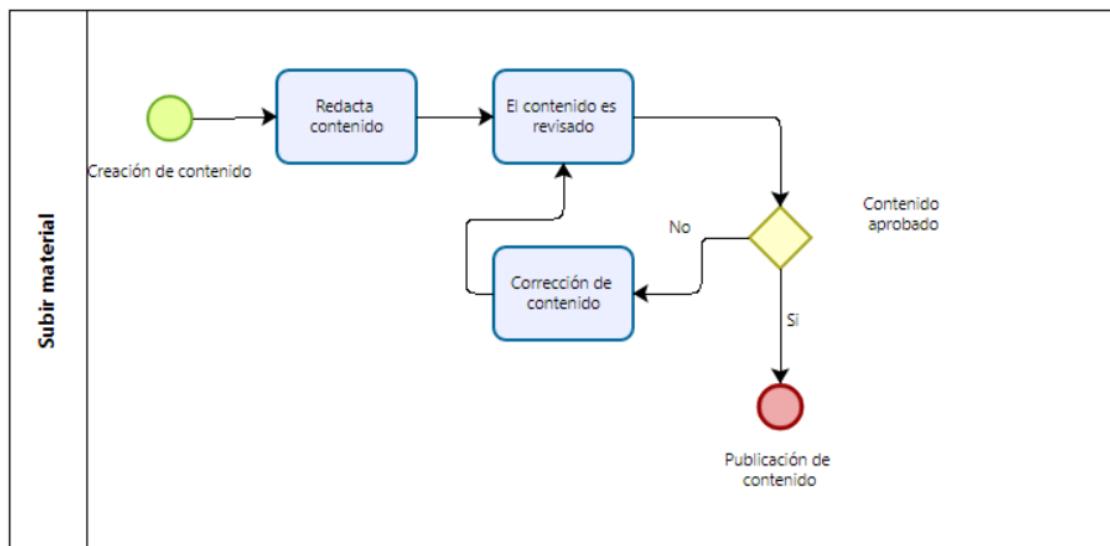
Gestión de Contenido

Objetivo: Crear materiales, preguntas y organizar el banco de preguntas.

Actores: Profesor, Revisor/Comité Académico, Coordinador

Paso a paso (AS-IS):

1. El profesor redacta material y preguntas (documento local o planilla).
2. El Profesor envía material al Coordinador o Comité para revisión (correo/impreso).
3. El Revisor valida la pertinencia, corrige errores y aprueba o devuelve con observaciones.
4. Si se aprueba, se ingresa manualmente en la Biblioteca/Banco (upload a LMS o registro en hoja de cálculo).
5. El banco se etiqueta manualmente por competencia y dificultad (si existe proceso formal) o queda sin clasificación estructurada.
6. Fin: Banco de preguntas disponible (pero su calidad y clasificación puede variar).



Proceso de Evaluación

Objetivo: Tomar exámenes/evaluaciones y registrar calificaciones.

Actores: Profesor, Estudiantes, Coordinador, Secretaría

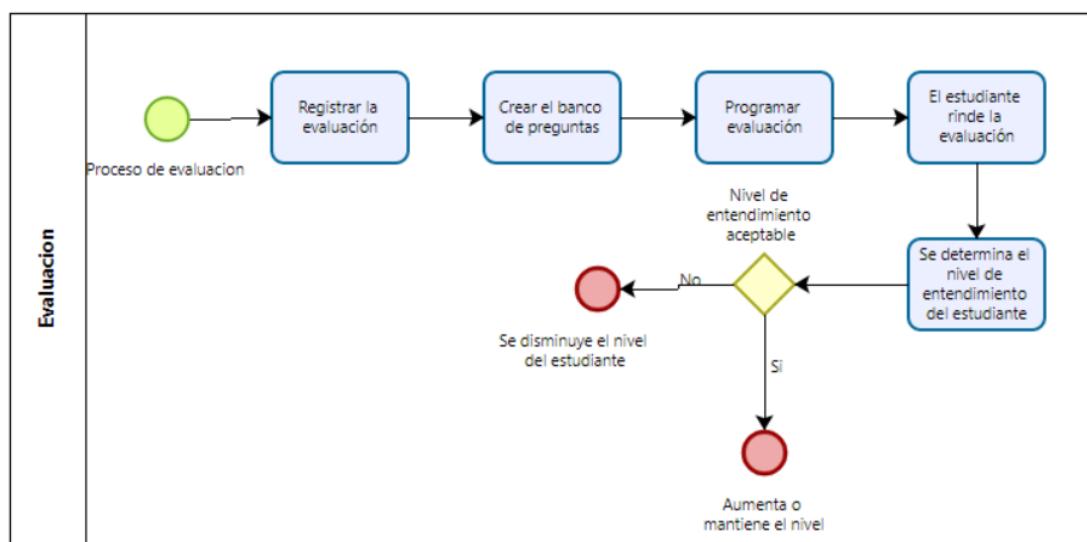
Entradas: Examen (papel o digital), banco de preguntas (si se usa), lista de estudiantes

Salidas: Calificaciones registradas, reportes impresos o en hojas.

Paso a paso (AS-IS):

1. Profesor diseña examen (manualmente con preguntas) y lo valida con comité si aplica.
2. Se programa fecha y aula; se notifica a los estudiantes.
3. Día de la evaluación: estudiantes rinden examen.
4. Profesor corrige manualmente las respuestas automáticas.
5. Profesor ingresa calificaciones en el sistema o en planilla; coordina con Secretaría si hay apelaciones.
6. Secretaría publica/entrega reportes y constancias.
7. Fin: registro de calificaciones.

Cuellos de botella: corrección manual consume tiempo; retraso en publicación de resultados; baja granularidad de reportes; no hay adaptación automática; trazabilidad limitada.



Análisis de resultados

Objetivo: Generar reportes de rendimiento para estudiantes y profesores.

Actores: Profesor, Coordinador, Administrador

Entradas: Calificaciones, registros de participación, logs (cuando existen)

Salidas: Reportes impresos o en Excel/PDF.

Paso a paso (AS-IS):

1. El Profesor solicita reportes (o los genera manualmente) reuniendo datos de calificaciones y asistencia.
2. Los datos se consolidan en hojas de cálculo o módulos básicos del LMS.
3. Profesor/Coordinador mapea métricas y genera gráficos manuales.
4. Reportes se envían o se discuten en reuniones.
5. Fin.

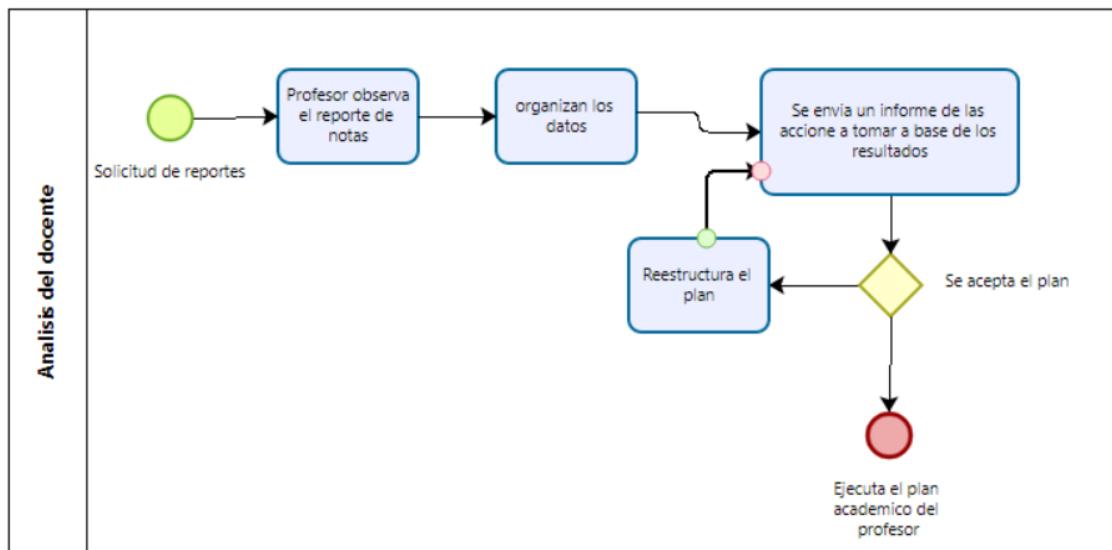


Diagrama de clases:

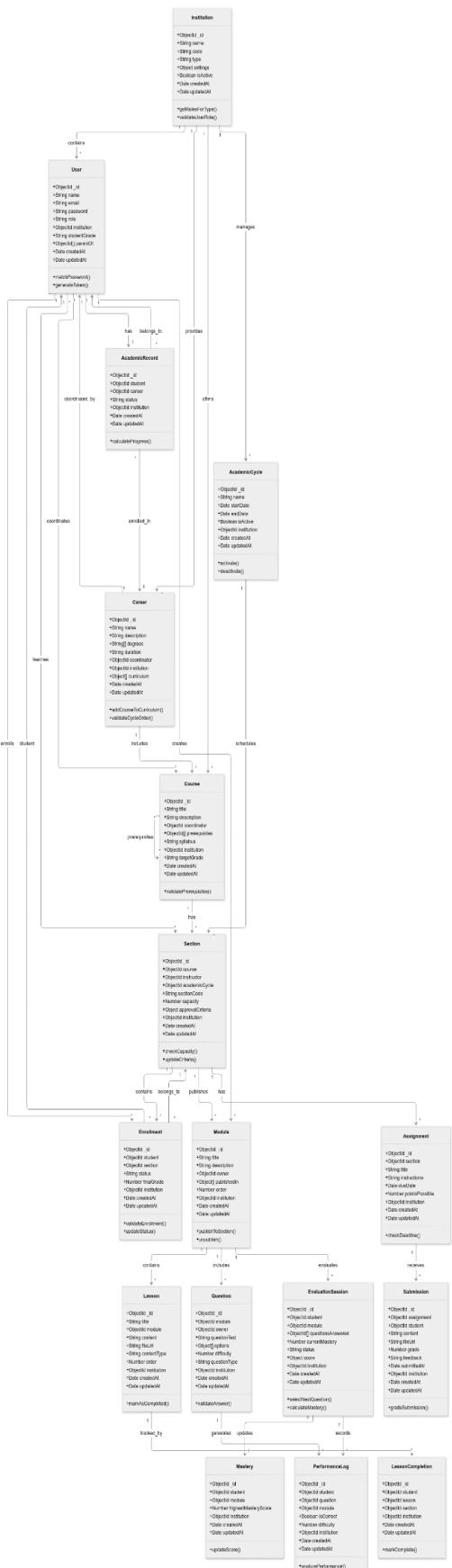
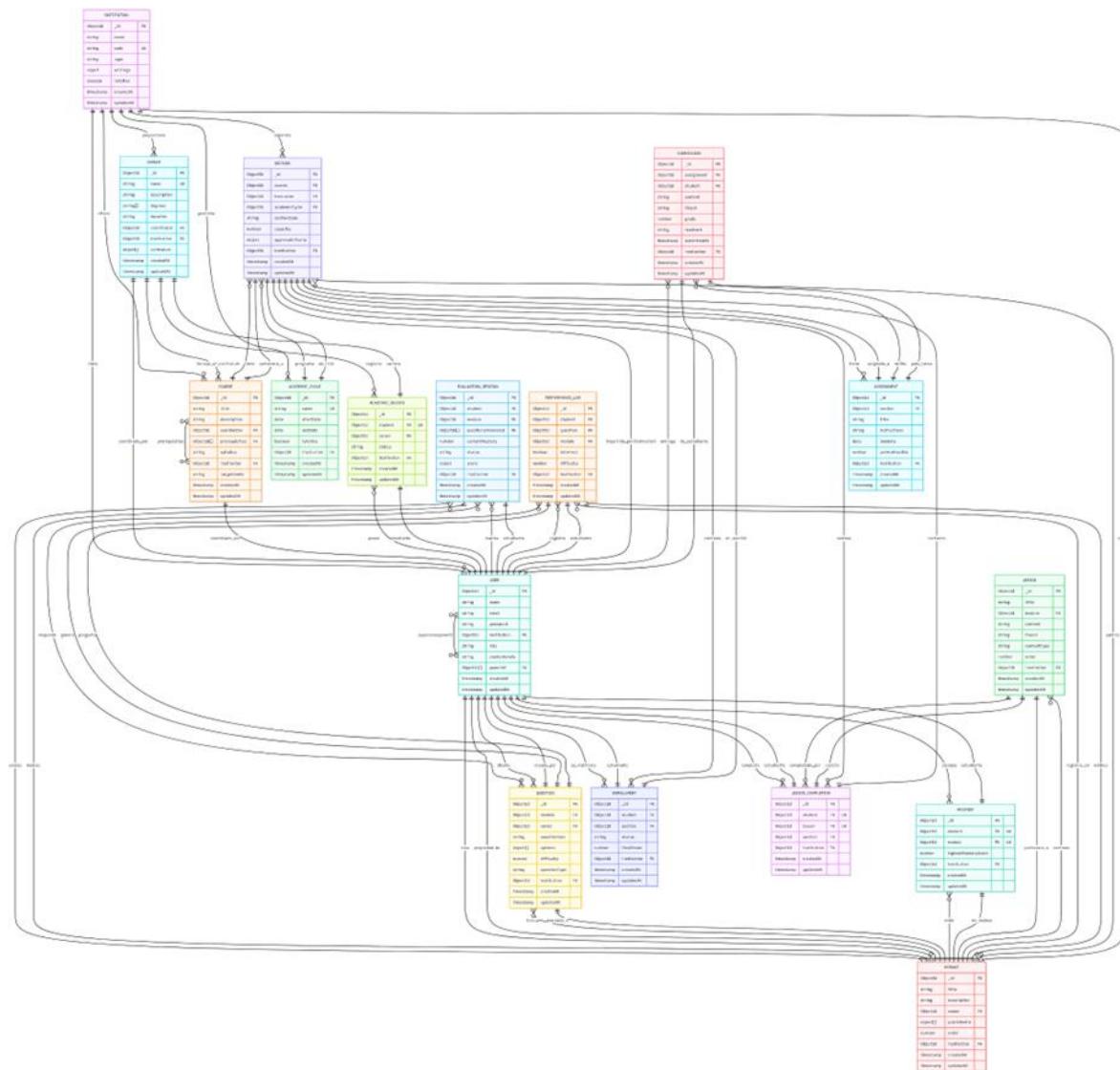


Diagrama entidad Relación:



CREACIÓN DE BASE DE DATOS:

DISEÑO LÓGICO DE BASE DE DATOS - ADAPTEST

1. INTRODUCCIÓN

AdaptaTest es un sistema de gestión de aprendizaje (LMS) adaptativo multi-tenant que soporta instituciones educativas de tipo universidad y colegio. La base de datos está diseñada para manejar el aislamiento de datos entre instituciones, evaluaciones adaptativas, gestión de cursos y seguimiento del progreso académico.

2. MODELO ENTIDAD-RELACIÓN

2.1 Entidades Principales

INSTITUTION (Institución)

Entidad central del sistema que implementa el modelo multi-tenant.

Atributos:

- `_id`: Identificador único
- `name`: Nombre de la institución
- `code`: Código único institucional
- `type`: Tipo (university/high_school)
- `settings`: Configuraciones personalizadas (JSON)
- `isActive`: Estado de activación
- `createdAt`: Fecha de creación
- `updatedAt`: Fecha de actualización

Relaciones:

- Una institución tiene muchos usuarios (1:N)
 - Una institución tiene muchos cursos (1:N)
 - Una institución tiene muchos ciclos académicos (1:N)
-

USER (Usuario)

Representa a todos los actores del sistema.

Atributos:

- `_id`: Identificador único
- `name`: Nombre completo
- `email`: Correo electrónico
- `password`: Contraseña encriptada (bcrypt)
- `institution`: Referencia a Institution
- `role`: Rol (student/professor/coordinator/admin/parent)
- `studentGrade`: Grado del estudiante (solo high_school)
- `parentOf`: Array de referencias a estudiantes (solo parent)
- `createdAt`: Fecha de creación
- `updatedAt`: Fecha de actualización

Restricciones:

- Email único por institución (índice compuesto)
- Password debe ser encriptado antes de almacenar

Relaciones:

- Un usuario pertenece a una institución (N:1)
 - Un estudiante puede tener muchas matrículas (1:N)
 - Un profesor puede ser instructor de muchas secciones (1:N)
 - Un coordinador puede gestionar una carrera (1:1)
-

COURSE (Curso)

Representa las asignaturas o materias.

Atributos:

- `_id`: Identificador único
- `title`: Título del curso
- `description`: Descripción detallada
- `institution`: Referencia a Institution
- `coordinator`: Referencia a User (opcional)
- `prerequisites`: Array de referencias a otros Courses
- `syllabus`: Ruta del archivo de sílabo
- `targetGrade`: Grado objetivo (solo high_school)
- `createdAt`: Fecha de creación
- `updatedAt`: Fecha de actualización

Relaciones:

- Un curso pertenece a una institución (N:1)
 - Un curso puede tener muchos prerequisitos (N:N autorreferencial)
 - Un curso tiene muchas secciones (1:N)
 - Un curso puede estar en muchas mallas curriculares (N:N)
-

ACADEMIC_CYCLE (Ciclo Académico)

Representa períodos académicos (semestres, trimestres, etc.).

Atributos:

- `_id`: Identificador único
- `name`: Nombre del ciclo (ej: "2025-I")
- `startDate`: Fecha de inicio
- `endDate`: Fecha de fin
- `isActive`: Indicador de ciclo activo (boolean)
- `institution`: Referencia a Institution
- `createdAt`: Fecha de creación
- `updatedAt`: Fecha de actualización

Restricciones:

- Solo un ciclo puede estar activo por institución

Relaciones:

- Un ciclo pertenece a una institución (N:1)
- Un ciclo tiene muchas secciones (1:N)

SECTION (Sección)

Representa una instancia específica de un curso en un ciclo.

Atributos:

- `_id`: Identificador único
- `course`: Referencia a Course
- `instructor`: Referencia a User
- `academicCycle`: Referencia a AcademicCycle
- `sectionCode`: Código de sección
- `capacity`: Capacidad máxima de estudiantes
- `approvalCriteria`: Objeto con criterios de aprobación
 - `mastery`: Configuración de maestría

- summativeAssessments: Array de evaluaciones sumativas
- completion: Configuración de completitud
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Restricciones:

- sectionCode único por ciclo académico

Relaciones:

- Una sección pertenece a un curso (N:1)
 - Una sección pertenece a un ciclo académico (N:1)
 - Una sección es impartida por un instructor (N:1)
 - Una sección tiene muchas matrículas (1:N)
 - Una sección tiene muchos módulos publicados (1:N)
-

ENROLLMENT (Matrícula)

Representa la inscripción de un estudiante en una sección.

Atributos:

- _id: Identificador único
- student: Referencia a User
- section: Referencia a Section
- status: Estado (enrolled/passed/failed/withdrawn)
- finalGrade: Calificación final (numérica)
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Restricciones:

- Un estudiante solo puede matricularse una vez por sección (índice único compuesto)

Relaciones:

- Una matrícula pertenece a un estudiante (N:1)
 - Una matrícula pertenece a una sección (N:1)
-

CAREER (Carrera)

Representa programas académicos (solo universidades).

Atributos:

- _id: Identificador único
- name: Nombre de la carrera
- description: Descripción
- degrees: Array de títulos otorgados
- duration: Duración del programa
- coordinator: Referencia a User
- institution: Referencia a Institution
- curriculum: Array de objetos con estructura:
 - cycleNumber: Número del ciclo
 - courses: Array de referencias a Course
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Relaciones:

- Una carrera pertenece a una institución (N:1)
 - Una carrera tiene un coordinador (N:1)
 - Una carrera incluye muchos cursos en su malla (N:N)
-

ACADEMIC_RECORD (Expediente Académico)

Registro único del estudiante en su carrera.

Atributos:

- _id: Identificador único

- student: Referencia a User (único)
- career: Referencia a Career
- status: Estado (active/graduated/withdrawn)
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Restricciones:

- Un estudiante solo puede tener un expediente académico

Relaciones:

- Un expediente pertenece a un estudiante (1:1)
 - Un expediente está asociado a una carrera (N:1)
-

MODULE (Módulo)

Unidad de contenido didáctico creada por profesores.

Atributos:

- _id: Identificador único
- title: Título del módulo
- description: Descripción
- owner: Referencia a User (profesor)
- publishedIn: Array de objetos:
 - section: Referencia a Section
- order: Orden de presentación
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Relaciones:

- Un módulo pertenece a un profesor (N:1)
- Un módulo puede estar publicado en muchas secciones (N:N)

- Un módulo tiene muchas lecciones (1:N)
 - Un módulo tiene muchas preguntas (1:N)
-

LESSON (Lección)

Contenido educativo dentro de un módulo.

Atributos:

- _id: Identificador único
- title: Título de la lección
- module: Referencia a Module
- content: Contenido principal (Markdown)
- contentType: Tipo (text/video_url/document_url/slides_url)
- fileUrl: URL del archivo adjunto
- order: Orden dentro del módulo
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Relaciones:

- Una lección pertenece a un módulo (N:1)
-

LESSON_COMPLETION (Completitud de Lección)

Registra el progreso del estudiante en lecciones.

Atributos:

- _id: Identificador único
- student: Referencia a User
- lesson: Referencia a Lesson
- section: Referencia a Section
- institution: Referencia a Institution
- createdAt: Fecha de completado

- updatedAt: Fecha de actualización

Restricciones:

- Índice único compuesto (student, lesson)

Relaciones:

- Una completitud pertenece a un estudiante (N:1)
 - Una completitud está asociada a una lección (N:1)
-

QUESTION (Pregunta)

Preguntas para evaluaciones adaptativas.

Atributos:

- _id: Identificador único
- module: Referencia a Module
- owner: Referencia a User (profesor)
- questionText: Texto de la pregunta
- options: Array de objetos:
 - text: Texto de la opción
 - isCorrect: Indicador de respuesta correcta
- difficulty: Nivel de dificultad (1-5)
- questionType: Tipo (multiple_choice/true_false)
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Relaciones:

- Una pregunta pertenece a un módulo (N:1)
 - Una pregunta tiene un creador (N:1)
-

EVALUATION_SESSION (Sesión de Evaluación)

Sesión activa de evaluación adaptativa.

Atributos:

- `_id`: Identificador único
- `student`: Referencia a User
- `module`: Referencia a Module
- `questionsAnswered`: Array de referencias a Question
- `currentMastery`: Puntaje actual de maestría (0-100)
- `status`: Estado (`in_progress/completed`)
- `score`: Objeto con puntajes:
 - `correct`: Respuestas correctas
 - `incorrect`: Respuestas incorrectas
- `institution`: Referencia a Institution
- `createdAt`: Fecha de inicio
- `updatedAt`: Fecha de actualización

Relaciones:

- Una sesión pertenece a un estudiante (N:1)
 - Una sesión evalúa un módulo (N:1)
-

MASTERY (Maestría)

Registro del nivel más alto de maestría alcanzado.

Atributos:

- `_id`: Identificador único
- `student`: Referencia a User
- `module`: Referencia a Module
- `highestMasteryScore`: Puntaje más alto (0-100)
- `institution`: Referencia a Institution
- `createdAt`: Fecha de creación
- `updatedAt`: Fecha de actualización

Restricciones:

- Índice único compuesto (student, module)

Relaciones:

- Un registro de maestría pertenece a un estudiante (N:1)
 - Un registro de maestría está asociado a un módulo (N:1)
-

PERFORMANCE_LOG (Registro de Desempeño)

Log detallado de cada respuesta del estudiante.

Atributos:

- _id: Identificador único
- student: Referencia a User
- question: Referencia a Question
- module: Referencia a Module
- isCorrect: Indicador de respuesta correcta
- difficulty: Dificultad de la pregunta
- institution: Referencia a Institution
- createdAt: Fecha del intento
- updatedAt: Fecha de actualización

Relaciones:

- Un log pertenece a un estudiante (N:1)
 - Un log está asociado a una pregunta (N:1)
-

ASSIGNMENT (Tarea)

Tareas asignadas en una sección.

Atributos:

- _id: Identificador único
- section: Referencia a Section
- title: Título de la tarea
- instructions: Instrucciones detalladas

- dueDate: Fecha de vencimiento
- pointsPossible: Puntaje máximo
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Relaciones:

- Una tarea pertenece a una sección (N:1)
 - Una tarea puede tener muchas entregas (1:N)
-

SUBMISSION (Entrega)

Entregas de tareas por estudiantes.

Atributos:

- _id: Identificador único
- assignment: Referencia a Assignment
- student: Referencia a User
- content: Contenido de la entrega
- fileUrl: URL del archivo adjunto
- grade: Calificación asignada
- feedback: Retroalimentación del profesor
- submittedAt: Fecha de entrega
- institution: Referencia a Institution
- createdAt: Fecha de creación
- updatedAt: Fecha de actualización

Restricciones:

- Índice único compuesto (assignment, student)

Relaciones:

- Una entrega pertenece a un estudiante (N:1)
- Una entrega está asociada a una tarea (N:1)

3. DIAGRAMA DE CARDINALIDADES

INSTITUTION (1) ———< (N) USER

INSTITUTION (1) ———< (N) COURSE

INSTITUTION (1) ———< (N) ACADEMIC_CYCLE

INSTITUTION (1) ———< (N) SECTION

USER (1) ———< (N) ENROLLMENT [como student]

USER (1) ———< (N) SECTION [como instructor]

USER (1) ———< (N) MODULE [como owner]

USER (1) ——— (1) CAREER [como coordinator]

USER (1) ——— (1) ACADEMIC_RECORD [como student]

COURSE (1) ———< (N) SECTION

COURSE (N) ———< (N) COURSE [prerrequisitos]

COURSE (N) ———< (N) CAREER [en curriculum]

ACADEMIC_CYCLE (1) ———< (N) SECTION

SECTION (1) ———< (N) ENROLLMENT

SECTION (1) ———< (N) ASSIGNMENT

SECTION (N) ———< (N) MODULE [publishedIn]

MODULE (1) ———< (N) LESSON

MODULE (1) ———< (N) QUESTION

MODULE (1) ———< (N) EVALUATION_SESSION

MODULE (1) ———< (N) MASTERY

LESSON (1) ———< (N) LESSON_COMPLETION

QUESTION (1) ———< (N) PERFORMANCE_LOG

ASSIGNMENT (1) ———< (N) SUBMISSION

CAREER (1) ———< (N) ACADEMIC_RECORD

4. REGLAS DE NEGOCIO

4.1 Multi-tenancy

- Todos los datos están aislados por institución
- Las consultas siempre deben filtrar por institution
- Los usuarios solo pueden acceder a datos de su institución

4.2 Ciclos Académicos

- Solo puede haber un ciclo activo por institución
- Las secciones solo se crean para el ciclo activo

4.3 Matrículas

- Un estudiante no puede matricularse dos veces en la misma sección
- Se valida la capacidad de la sección antes de matricular
- Se validan prerequisitos antes de matricular (universidades)

4.4 Evaluaciones Adaptativas

- El algoritmo ajusta la dificultad según la maestría actual
- Solo se guarda el puntaje más alto de maestría
- Las sesiones finalizan después de 10 preguntas o al alcanzar 95% de maestría

4.5 Mallas Curriculares

- Un curso no puede estar en un ciclo menor o igual al de sus prerequisitos
- Un curso solo puede aparecer una vez en la malla curricular

4.6 Entregas de Tareas

- Un estudiante solo puede entregar una vez por tarea
 - Solo estudiantes matriculados pueden entregar tareas
-

5. NORMALIZACIÓN

El diseño cumple con la **Tercera Forma Normal (3NF)**:

- **1NF:** Todos los atributos contienen valores atómicos
- **2NF:** No existen dependencias parciales
- **3NF:** No existen dependencias transitivas

Ejemplo de normalización aplicada:

Originalmente, se podría haber almacenado información del curso directamente en Enrollment, pero se separó en entidades relacionadas:

- Enrollment → Section → Course

Esto elimina redundancia y asegura integridad referencial.

6. CONCLUSIONES

El diseño lógico de AdaptaTest está optimizado para:

- Escalabilidad multi-tenant
- Integridad referencial
- Evaluaciones adaptativas en tiempo real
- Seguimiento detallado del progreso académico
- Flexibilidad para diferentes tipos de instituciones educativas

DISEÑO FÍSICO DE BASE DE DATOS - ADAPTATEST

1. INTRODUCCIÓN

Este sección describe la implementación física de la base de datos AdaptaTest en **MongoDB**, detallando los esquemas de colecciones, tipos de datos, índices, y estrategias de optimización.

2. SISTEMA DE GESTIÓN DE BASE DE DATOS

SGBD Seleccionado: MongoDB 6.0+

ODM (Object Document Mapper): Mongoose 7.0+

Justificación:

- Flexibilidad de esquema para diferentes tipos de instituciones
 - Soporte nativo para documentos embebidos y arrays
 - Escalabilidad horizontal
 - Excelente rendimiento en operaciones de lectura
 - Integración perfecta con Node.js/Express
-

3. ESQUEMAS DE COLECCIONES

3.1 INSTITUTIONS

```
{
  _id: ObjectId,
  name: String (required, max: 200),
  code: String (required, unique, uppercase, max: 20),
  type: String (enum: ['university', 'high_school'], required),
  settings: {
    maxStudentsPerSection: Number (default: 30, min: 1, max: 200),
    allowParentAccess: Boolean (default: false),
    gradingScale: String (default: '0-20'),
    requiresPrerequisites: Boolean (default: true)
  },
  isActive: Boolean (default: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ code: 1 } // Único
{ isActive: 1 }
{ type: 1 }
```

3.2 USERS

```
{
  _id: ObjectId,
  name: String (required, trim, min: 3, max: 150),
  email: String (required, lowercase, trim, match: regex),
  password: String (required, min: 60, max: 60), // Bcrypt hash
  institution: ObjectId (ref: 'Institution', required, index: true),
  role: String (enum: ['student', 'professor', 'coordinator', 'admin',
  'parent'], required),
  studentGrade: String (enum: ['1ro', '2do', '3ro', '4to', '5to'],
  optional),
  parentOf: [ObjectId] (ref: 'User', optional),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ institution: 1, email: 1 } // Único compuesto
{ role: 1 }
{ institution: 1, role: 1 }
```

Middleware Pre-Save (Bcrypt):

```
// Encriptación automática de contraseñas
if (password.isModified) {
  salt = bcrypt.genSalt(10)
  password = bcrypt.hash(password, salt)
}
```

3.3 COURSES

```
{
  _id: ObjectId,
  title: String (required, trim, max: 200),
  description: String (required, max: 2000),
  institution: ObjectId (ref: 'Institution', required, index: true),
  coordinator: ObjectId (ref: 'User', optional, index: true),
  prerequisites: [ObjectId] (ref: 'Course', default: []),
  syllabus: String (max: 500), // Ruta del archivo
  targetGrade: String (enum: ['1ro', '2do', '3ro', '4to', '5to']),
  optional,
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ institution: 1 }
{ institution: 1, title: 1 }
{ coordinator: 1 }
{ prerequisites: 1 }
```

3.4 ACADEMIC_CYCLES

```
{
  _id: ObjectId,
  name: String (required, unique, max: 50),
  startDate: ISODate (required),
  endDate: ISODate (required),
  isActive: Boolean (default: false),
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Validación:

```
// Validar que endDate > startDate
validate: function() {
  return this.endDate > this.startDate;
}
```

Índices:

```
{ institution: 1, isActive: 1 }
{ startDate: 1 }
{ name: 1 } // Único
```

3.5 SECTIONS

```
{
  _id: ObjectId,
  course: ObjectId (ref: 'Course', required, index: true),
  instructor: ObjectId (ref: 'User', required, index: true),
  academicCycle: ObjectId (ref: 'AcademicCycle', required, index: true),
  sectionCode: String (required, uppercase, max: 20),
  capacity: Number (required, min: 1, max: 200),
  approvalCriteria: {
    mastery: {
      required: Boolean (default: false),
      minPercentage: Number (min: 1, max: 100)
    },
    summativeAssessments: [
      assignment: ObjectId (ref: 'Assignment'),
      minGrade: Number (min: 0, max: 100)
    ],
    completion: {
      requiredLessonsPercentage: Number (min: 1, max: 100),
      allAssignmentsRequired: Boolean (default: false)
    }
  },
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ sectionCode: 1, academicCycle: 1 } // Único compuesto
{ course: 1 }
{ instructor: 1 }
{ academicCycle: 1 }
{ institution: 1 }
```

3.6 ENROLLMENTS

```
{
  _id: ObjectId,
  student: ObjectId (ref: 'User', required, index: true),
  section: ObjectId (ref: 'Section', required, index: true),
  status: String (enum: ['enrolled', 'passed', 'failed', 'withdrawn']),
  default: 'enrolled',
  finalGrade: Number (min: 0, max: 100, optional),
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ student: 1, section: 1 } // Único compuesto
{ section: 1, status: 1 }
{ student: 1, status: 1 }
{ institution: 1 }
```

3.7 CAREERS

```
{
  _id: ObjectId,
  name: String (required, unique, trim, max: 200),
  description: String (required, max: 2000),
  degrees: [String] (required, min: 1),
  duration: String (required, max: 50),
  coordinator: ObjectId (ref: 'User', optional, index: true),
  institution: ObjectId (ref: 'Institution', required, index: true),
  curriculum: [
    cycleNumber: Number (required, min: 1, max: 20),
    courses: [ObjectId] (ref: 'Course')
  ],
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ institution: 1 }
{ coordinator: 1 }
{ name: 1 } // Único
{ 'curriculum.cycleNumber': 1 } // Índice en subdocumento
```

3.8 ACADEMIC_RECORDS

```
{
  _id: ObjectId,
  student: ObjectId (ref: 'User', required, unique, index: true),
  career: ObjectId (ref: 'Career', required, index: true),
  status: String (enum: ['active', 'graduated', 'withdrawn'], default: 'active'),
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ student: 1 } // Único
{ career: 1 }
{ institution: 1, status: 1 }
```

3.9 MODULES

```
{
  _id: ObjectId,
  title: String (required, trim, max: 200),
  description: String (max: 2000),
  owner: ObjectId (ref: 'User', required, index: true),
  publishedIn: [
    section: ObjectId (ref: 'Section')
  ],
}
```

```
    order: Number (default: 1, min: 1),
    institution: ObjectId (ref: 'Institution', required, index: true),
    createdAt: ISODate,
    updatedAt: ISODate
}
```

Índices:

```
{ owner: 1 }
{ 'publishedIn.section': 1 }
{ institution: 1 }
{ owner: 1, institution: 1 }
```

3.10 LESSONS

```
{
  _id: ObjectId,
  title: String (required, trim, max: 200),
  module: ObjectId (ref: 'Module', required, index: true),
  content: String (max: 50000), // Contenido Markdown
  contentType: String (enum: ['text', 'video_url', 'document_url',
'slides_url'], default: 'text'),
  fileUrl: String (max: 500),
  order: Number (required, min: 1),
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ module: 1, order: 1 }
{ institution: 1 }
```

3.11 LESSON_COMPLETIONS

```
{
  _id: ObjectId,
  student: ObjectId (ref: 'User', required, index: true),
  lesson: ObjectId (ref: 'Lesson', required, index: true),
  section: ObjectId (ref: 'Section', required, index: true),
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}
```

Índices:

```
{ student: 1, lesson: 1 } // Único compuesto
{ section: 1, student: 1 }
{ institution: 1 }
```

3.12 QUESTIONS

```
{
  _id: ObjectId,
  module: ObjectId (ref: 'Module', required, index: true),
  owner: ObjectId (ref: 'User', required, index: true),
```

```

questionText: String (required, max: 2000),
options: [
  _id: ObjectId (auto-generated),
  text: String (required, max: 500),
  isCorrect: Boolean (required, default: false)
],
difficulty: Number (required, min: 1, max: 5),
questionType: String (enum: ['multiple_choice', 'true_false']),
default: 'multiple_choice',
institution: ObjectId (ref: 'Institution', required, index: true),
createdAt: ISODate,
updatedAt: ISODate
}

```

Validación:

```

// Al menos debe haber una opción correcta
validate: function() {
  return this.options.some(opt => opt.isCorrect === true);
}

```

Índices:

```

{ module: 1, difficulty: 1 }
{ owner: 1 }
{ institution: 1 }

```

3.13 EVALUATION_SESSIONS

```

{
  _id: ObjectId,
  student: ObjectId (ref: 'User', required, index: true),
  module: ObjectId (ref: 'Module', required, index: true),
  questionsAnswered: [ObjectId] (ref: 'Question', default: []),
  currentMastery: Number (default: 25, min: 0, max: 100),
  status: String (enum: ['in_progress', 'completed'], default:
  'in_progress'),
  score: {
    correct: Number (default: 0, min: 0),
    incorrect: Number (default: 0, min: 0)
  },
  institution: ObjectId (ref: 'Institution', required, index: true),
  createdAt: ISODate,
  updatedAt: ISODate
}

```

Índices:

```

{ student: 1, module: 1 }
{ status: 1 }
{ institution: 1 }

```

3.14 MASTERY

```

{
  _id: ObjectId,
  student: ObjectId (ref: 'User', required, index: true),
  module: ObjectId (ref: 'Module', required, index: true),

```

```
        highestMasteryScore: Number (default: 0, min: 0, max: 100),
        institution: ObjectId (ref: 'Institution', required, index: true),
        createdAt: ISODate,
        updatedAt: ISODate
    }
```

Índices:

```
{ student: 1, module: 1 } // Único compuesto
{ module: 1, highestMasteryScore: -1 } // Para rankings
{ institution: 1 }
```

3.15 PERFORMANCE_LOGS

```
{
    _id: ObjectId,
    student: ObjectId (ref: 'User', required, index: true),
    question: ObjectId (ref: 'Question', required, index: true),
    module: ObjectId (ref: 'Module', required, index: true),
    isCorrect: Boolean (required),
    difficulty: Number (required, min: 1, max: 5),
    institution: ObjectId (ref: 'Institution', required, index: true),
    createdAt: ISODate,
    updatedAt: ISODate
}
```

Índices:

```
{ student: 1, module: 1 }
{ question: 1 }
{ module: 1, isCorrect: 1 }
{ institution: 1 }
{ createdAt: -1 } // Para análisis temporal
```

3.16 ASSIGNMENTS

```
{
    _id: ObjectId,
    section: ObjectId (ref: 'Section', required, index: true),
    title: String (required, trim, max: 200),
    instructions: String (max: 5000),
    dueDate: ISODate,
    pointsPossible: Number (default: 100, min: 0, max: 1000),
    institution: ObjectId (ref: 'Institution', required, index: true),
    createdAt: ISODate,
    updatedAt: ISODate
}
```

Índices:

```
{ section: 1 }
{ section: 1, dueDate: 1 }
{ institution: 1 }
```

3.17 SUBMISSIONS

```
{
    _id: ObjectId,
```

```

        assignment: ObjectId (ref: 'Assignment', required, index: true),
        student: ObjectId (ref: 'User', required, index: true),
        content: String (max: 10000),
        fileUrl: String (max: 500),
        grade: Number (min: 0, max: 100),
        feedback: String (max: 2000),
        submittedAt: ISODate (default: Date.now),
        institution: ObjectId (ref: 'Institution', required, index: true),
        createdAt: ISODate,
        updatedAt: ISODate
    }
}

```

Índices:

```

{ assignment: 1, student: 1 } // Único compuesto
{ student: 1 }
{ assignment: 1, submittedAt: 1 }
{ institution: 1 }

```

4. TIPOS DE DATOS DETALLADOS

4.1 Tipos Primitivos MongoDB

Tipo MongoDB	Mongoose Type	Uso en AdaptaTest
ObjectId	Schema.Types.ObjectId	IDs y referencias
String	String	Textos, nombres, descripciones
Number	Number	Calificaciones, puntajes, capacidades
Boolean	Boolean	Estados activos, flags
Date	Date	Timestamps, fechas de eventos
Array	[]	Listas de requisitos, opciones
Mixed	Schema.Types.Mixed	settings, approvalCriteria

4.2 Tipos Personalizados

Enums:

```

// Roles de usuario
['student', 'professor', 'coordinator', 'admin', 'parent']

// Tipos de institución
['university', 'high_school']

// Estados de matrícula
['enrolled', 'passed', 'failed', 'withdrawn']

// Estados de expediente
['active', 'graduated', 'withdrawn']

// Tipos de contenido
['text', 'video_url', 'document_url', 'slides_url']

// Tipos de pregunta
['multiple_choice', 'true_false']

// Estados de sesión
['in_progress', 'completed']

```

5. ESTRATEGIA DE ÍNDICES

5.1 Índices Únicos

Garantizan la integridad de datos críticos:

```
// Institution
{ code: 1 } // Código único por institución

// User
{ institution: 1, email: 1 } // Email único por institución

// AcademicCycle
{ name: 1 } // Nombre único de ciclo

// Section
{ sectionCode: 1, academicCycle: 1 } // Sección única por ciclo

// Enrollment
{ student: 1, section: 1 } // Matrícula única

// AcademicRecord
{ student: 1 } // Un expediente por estudiante

// LessonCompletion
{ student: 1, lesson: 1 } // Completitud única

// Mastery
{ student: 1, module: 1 } // Maestría única

// Submission
{ assignment: 1, student: 1 } // Entrega única
```

5.2 Índices Compuestos

Optimizan queries comunes:

```
// Búsqueda de usuarios por institución y rol
{ institution: 1, role: 1 }

// Búsqueda de secciones activas
{ academicCycle: 1, course: 1 }

// Búsqueda de matrículas por estado
{ section: 1, status: 1 }

// Análisis de desempeño
{ student: 1, module: 1, createdAt: -1 }

// Lecciones ordenadas
{ module: 1, order: 1 }
```

5.3 Índices de Texto

Para búsquedas de texto completo:

```

// Búsqueda de cursos
db.courses.createIndex({
  title: "text",
  description: "text"
}, {
  weights: { title: 10, description: 5 },
  name: "course_search_index"
})

// Búsqueda de preguntas
db.questions.createIndex({
  questionText: "text"
}, {
  name: "question_search_index"
})

```

5.4 Índices TTL (Time To Live)

Para limpieza automática de sesiones:

```

// Eliminar sesiones incompletas después de 7 días
db.evaluation_sessions.createIndex(
  { createdAt: 1 },
  {
    expireAfterSeconds: 604800,
    partialFilterExpression: { status: "in_progress" }
  }
)

```

6. RESTRICCIONES DE INTEGRIDAD

6.1 Integridad Referencial

Mongoose Virtual Populate:

```

// En el modelo Section
sectionSchema.virtual('enrollments', {
  ref: 'Enrollment',
  localField: '_id',
  foreignField: 'section'
})

```

6.2 Validaciones a Nivel de Schema

```

// Validación de email
email: {
  type: String,
  required: true,
  match: [/^\\w+([\\.-]?\\w+)*\\w+([\\.-]?\\w+)*(\\.\\w{2,3})+$/, 
          'Por favor, introduce un email válido.']
}

// Validación de rango
capacity: {
  type: Number,
  required: true,
  min: [1, 'La capacidad debe ser al menos 1'],
}

```

```

        max: [200, 'La capacidad no puede exceder 200']
    }

// Validación personalizada
validate: {
    validator: function(v) {
        return this.endDate > this.startDate;
    },
    message: 'La fecha de fin debe ser posterior a la fecha de inicio'
}

```

6.3 Middleware de Validación

```

// Pre-save para AcademicCycle
academicCycleSchema.pre('save', async function(next) {
    if (this.isActive) {
        // Desactivar otros ciclos activos de la misma institución
        await this.constructor.updateMany(
            {
                institution: this.institution,
                _id: { $ne: this._id }
            },
            { isActive: false }
        )
    }
    next()
})

```

7. SEGURIDAD

7.1 Cifrado de Contraseñas

Implementación con Bcrypt:

```

// Pre-save hook en User model
userSchema.pre('save', async function(next) {
    if (!this.isModified('password')) {
        return next()
    }

    const salt = await bcrypt.genSalt(10)
    this.password = await bcrypt.hash(this.password, salt)
    next()
})

// Método de comparación
userSchema.methods.matchPassword = async function(enteredPassword) {
    return await bcrypt.compare(enteredPassword, this.password)
}

```

Configuración:

- Salt rounds: 10
- Longitud hash: 60 caracteres
- Algoritmo: bcrypt (Blowfish)

7.2 Tokens JWT

```
// Generación de token
const generateToken = (userId, institutionId) => {
  return jwt.sign(
    {
      id: userId,
      institution: institutionId
    },
    process.env.JWT_SECRET,
    {
      expiresIn: '30d',
      algorithm: 'HS256'
    }
  )
}
```

7.3 Principio de Menor Privilegio

Roles y Permisos en MongoDB:

```
// Crear usuario de aplicación con permisos limitados
use adaptatest_db
db.createUser({
  user: "adaptatest_app",
  pwd: "secure_password_here",
  roles: [
    {
      role: "readWrite",
      db: "adaptatest_db"
    }
  ]
})

// Usuario de solo lectura para reportes
db.createUser({
  user: "adaptatest_READONLY",
  pwd: "readonly_password",
  roles: [
    {
      role: "read",
      db: "adaptatest_db"
    }
  ]
})
```

7.4 Prevención de Inyección NoSQL

Sanitización de inputs:

```
const mongoSanitize = require('express-mongo-sanitize')

// En server.js
app.use(mongoSanitize({
  replaceWith: '_',
  onSanitize: ({ req, key }) => {
    console.warn(`Attempted NoSQL injection on key: ${key}`)
  }
})
```

```
}))
```

7.5 Protección de Datos Sensibles

```
// Excluir password de queries por defecto
userSchema.methods.toJSON = function() {
  const obj = this.toObject()
  delete obj.password
  return obj
}

// Select explícito para password
User.findById(id).select('+password')
```

8. OPTIMIZACIONES DE RENDIMIENTO

8.1 Conexión a MongoDB

```
// Configuración optimizada
mongoose.connect(process.env.MONGO_URI, {
  maxPoolSize: 10, // Pool de conexiones
  minPoolSize: 2,
  socketTimeoutMS: 45000,
  serverSelectionTimeoutMS: 5000,
  family: 4 // Usar IPv4
})
```

8.2 Lean Queries

Para operaciones de solo lectura:

```
// Query normal (retorna documento Mongoose completo)
const users = await User.find({ institution: id })

// Lean query (retorna objeto plano JavaScript - más rápido)
const users = await User.find({ institution: id }).lean()
```

8.3 Proyecciones

Seleccionar solo campos necesarios:

```
// Malo - trae todo el documento
const courses = await Course.find({ institution: id })

// Bueno - solo campos necesarios
const courses = await Course.find(
  { institution: id },
  'title description coordinator'
)
```

8.4 Pagination

```
const page = parseInt(req.query.page) || 1
const limit = parseInt(req.query.limit) || 20
const skip = (page - 1) * limit
```

```

const results = await Model.find(query)
  .limit(limit)
  .skip(skip)
  .sort({ createdAt: -1 })

```

8.5 Agregación para Analytics

```

// Ejemplo: Estudiantes con dificultades por sección
const analytics = await PerformanceLog.aggregate([
  { $match: { module: { $in: moduleIds } } },
  {
    $group: {
      _id: '$student',
      totalQuestions: { $sum: 1 },
      correctAnswers: {
        $sum: { $cond: ['$isCorrect', 1, 0] }
      }
    }
  },
  {
    $project: {
      student: '_id',
      successRate: {
        $multiply: [
          { $divide: ['$correctAnswers', '$totalQuestions'] },
          100
        ]
      }
    }
  },
  { $match: { successRate: { $lt: 40 } } }
])

```

9. BACKUP Y RECUPERACIÓN

9.1 Estrategia de Backup

```

# Backup completo diario
mongodump --uri="mongodb://user:pass@host:27017/adaptatest_db" \
--out=/backups/${(date +%Y%m%d)}

# Backup incremental (oplog)
mongodump --uri="mongodb://user:pass@host:27017/adaptatest_db" \
--oplog \
--out=/backups/incremental

```

9.2 Restauración

```

# Restaurar backup completo
mongorestore --uri="mongodb://user:pass@host:27017" \
/backups/20250930/adaptatest_db

# Restaurar con oplog
mongorestore --uri="mongodb://user:pass@host:27017" \
--oplogReplay \
/backups/incremental

```

10. MONITOREO

10.1 Métricas Clave

```
// Estadísticas de colección
db.enrollments.stats()

// Operaciones lentas
db.setProfilingLevel(1, { slowms: 100 })
db.system.profile.find().sort({ ts: -1 }).limit(5)

// Uso de índices
db.enrollments.explain("executionStats").find({
  student: ObjectId("..."))
})
```

10.2 Alertas Recomendadas

- Conexiones activas > 80% del pool
 - Queries lentas > 100ms
 - Espacio en disco < 20%
 - Replica lag > 10 segundos
 - Número de índices no utilizados
-

DICCIONARIO DE DATOS

Información del Documento

Proyecto: AdaptaTest - Sistema de Gestión de Aprendizaje Adaptativo

Versión: 1.0

Fecha: Octubre 2025

SGBD: MongoDB 6.0+

ODM: Mongoose 7.0+

Autor: Equipo de Desarrollo AdaptaTest

Convenciones de Nomenclatura

- **Colecciones:** Nombres en minúsculas, plural (ej: users, courses)
 - **Campos:** camelCase (ej: firstName, createdAt)
 - **Referencias:** ObjectId de MongoDB
 - **Timestamps:** Formato ISODate de MongoDB
 - **Enums:** snake_case en minúsculas
-

1. INSTITUTIONS

Propósito: Almacena información de las instituciones educativas. Es la entidad central del modelo multi-tenant.

Colección: institutions

Estructura de Campos

Campo	Tipo	Tamaño	Nulo	Default	Descripción
_id	ObjectID	-	NO	Auto	Identificador único
name	String	200	NO	-	Nombre oficial completo
code	String	20	NO	-	Código único (ej: "UTP", "SANMARTIN")
type	String	-	NO	-	Tipo: "university" o "high_school"
settings	Object	-	SÍ	{}	Configuraciones personalizadas
settings.maxStudentsPerSection	Number	-	SÍ	30	Capacidad máxima por sección
settings.allowParentAccess	Boolean	-	SÍ	false	Permite acceso a padres
settings.gradingScale	String	20	SÍ	"0-20"	Escala de calificación
settings.requiresPrerequisites	Boolean	-	SÍ	true	Requiere requisitos
isActive	Boolean	-	NO	true	Estado de activación
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de última modificación

Restricciones

- **Primary Key:** _id
- **Unique Key:** code
- **Check:** type IN ('university', 'high_school')
- **Check:** maxStudentsPerSection >= 1 AND maxStudentsPerSection <= 200

Índices

```
{
  { code: 1 } // Único
  { isActive: 1 }
  { type: 1 }
```

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439011"),
  "name": "Universidad Tecnológica del Perú",
  "code": "UTP",
  "type": "university",
  "settings": {
    "maxStudentsPerSection": 35,
    "allowParentAccess": false,
    "gradingScale": "0-20",
    "requiresPrerequisites": true
  },
  "isActive": true,
  "createdAt": ISODate("2025-01-15T10:00:00Z"),
  "updatedAt": ISODate("2025-01-15T10:00:00Z")
}
```

2. USERS

Propósito: Almacena todos los usuarios del sistema con diferentes roles.

Colección: users

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
name	String	150	NO	-	Nombre completo
email	String	255	NO	-	Correo electrónico
password	String	60	NO	-	Hash bcrypt (60 caracteres)
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
role	String	-	NO	-	Rol del usuario
studentGrade	String	-	SÍ	null	Grado (solo high_school)
parentOf	Array[ObjectId]	-	SÍ	[]	Hijos (solo parent)
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Valores Permitidos

role:

- student - Estudiante
- professor - Profesor/Docente
- coordinator - Coordinador de carrera

- `admin` - Administrador institucional
- `parent` - Padre de familia (solo `high_school`)

studentGrade:

- 1ro, 2do, 3ro, 4to, 5to

Restricciones

- **Primary Key:** `_id`
- **Unique Key:** `(institution, email)` - Índice compuesto
- **Foreign Key:** `institution → INSTITUTIONS(_id)`
- **Foreign Key:** `parentOf[] → USERS(_id)`
- **Check:** Email debe cumplir formato válido
- **Check:** Password debe ser hash bcrypt de 60 caracteres

Índices

```
{ institution: 1, email: 1 } // Único compuesto
{ role: 1 }
{ institution: 1, role: 1 }
{ email: 1 }
```

Reglas de Negocio

1. El email es único por institución (multi-tenancy)
2. La contraseña se encripta automáticamente con bcrypt (10 salt rounds)
3. `studentGrade` solo aplica para instituciones tipo `high_school`
4. `parentOf` solo aplica para usuarios con rol `parent`

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439012"),
  "name": "María López García",
  "email": "mlopez@utp.edu.pe",
  "password": "$2a$10$N9qo8uLOickgx2ZMRZoMyeIjZAgcf17p921dGxad68LJZdL17lhWy",
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "role": "student",
  "studentGrade": null,
  "parentOf": [],
  "createdAt": ISODate("2025-01-15T10:30:00Z"),
  "updatedAt": ISODate("2025-01-15T10:30:00Z")
}
```

3. COURSES

Propósito: Representa los cursos o asignaturas ofrecidas.

Colección: courses

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
title	String	200	NO	-	Título del curso
description	String	2000	NO	-	Descripción detallada
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
coordinator	ObjectId	-	SÍ	null	FK → USERS (profesor)
prerequisites	Array[ObjectId]	-	SÍ	[]	FK → COURSES
syllabus	String	500	SÍ	null	Ruta del archivo PDF
targetGrade	String	-	SÍ	null	Grado objetivo (high school)
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** _id
- **Foreign Key:** institution → INSTITUTIONS(_id)
- **Foreign Key:** coordinator → USERS(_id)
- **Foreign Key:** prerequisites[] → COURSES(_id)
- **Check:** targetGrade IN ('1ro', '2do', '3ro', '4to', '5to') OR NULL

Índices

```
{ institution: 1 }
{ institution: 1, title: 1 }
{ coordinator: 1 }
{ prerequisites: 1 }
{ title: "text", description: "text" } // Texto completo
```

Reglas de Negocio

1. Los requisitos solo aplican en instituciones tipo university
2. Un curso no puede ser requisito de sí mismo
3. targetGrade solo para instituciones tipo high_school

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439013"),
  "title": "Cálculo Diferencial",
  "description": "Introducción al cálculo diferencial y sus
  aplicaciones",
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "coordinator": ObjectId("507f1f77bcf86cd799439014"),
  "prerequisites": [ObjectId("507f1f77bcf86cd799439015")],
```

```

    "syllabus": "/uploads/syllabus-calculo-diferencial.pdf",
    "targetGrade": null,
    "createdAt": ISODate("2025-01-20T14:00:00Z"),
    "updatedAt": ISODate("2025-01-20T14:00:00Z")
}

```

4. ACADEMIC_CYCLES

Propósito: Representa períodos académicos (semestres, trimestres).

Colección: academiccycles

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
name	String	50	NO	-	Nombre (ej: "2025-I")
startDate	Date	-	NO	-	Fecha de inicio
endDate	Date	-	NO	-	Fecha de fin
isActive	Boolean	-	NO	false	Ciclo activo actual
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** _id
- **Unique Key:** name
- **Foreign Key:** institution → INSTITUTIONS(_id)
- **Check:** endDate > startDate

Índices

```

{ name: 1 } // Único
{ institution: 1, isActive: 1 }
{ startDate: 1 }

```

Reglas de Negocio

1. Solo un ciclo puede estar activo (isActive: true) por institución
2. Al activar un ciclo, los demás se desactivan automáticamente
3. La fecha de fin debe ser posterior a la de inicio

Ejemplo

```

{
  "_id": ObjectId("507f1f77bcf86cd799439016"),
  "name": "2025-I",
  "startDate": ISODate("2025-03-01T00:00:00Z"),
  "endDate": ISODate("2025-07-31T23:59:59Z"),
  "isActive": true,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
}

```

```

    "createdAt": ISODate("2025-01-10T09:00:00Z"),
    "updatedAt": ISODate("2025-03-01T00:00:00Z")
}

```

5. SECTIONS

Propósito: Instancia específica de un curso en un ciclo académico.

Colección: sections

Estructura de Campos

Campo	Tipo	Tam año	N ull	Def ault	Descripción
_id	ObjectID	-	NO	Aut o	Identificador único
course	ObjectID	-	NO	-	FK → COURSES
instructor	ObjectID	-	NO	-	FK → USERS (profesor)
academicCycle	ObjectID	-	NO	-	FK → ACADEMIC_CYCLES
sectionCode	String	20	NO	-	Código (ej: "A1", "B2")
capacity	Number	-	NO	-	Capacidad máxima
approvalCriteria	Object	-	SÍ	{}	Criterios de aprobación
approvalCriteria.mastery.required	Boolean	-	SÍ	false	Requiere maestría
approvalCriteria.mastery.minPercentage	Number	-	SÍ	null	% mínimo (1-100)
approvalCriteria.summativeAssessments	Array	-	SÍ	[]	Evaluaciones sumativas
approvalCriteria.completion.requiredLessonsPercentage	Number	-	SÍ	null	% lecciones
approvalCriteria.completion.allAssignmentsRequired	Boolean	-	SÍ	false	Todas las tareas
institution	ObjectID	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Aut o	Fecha de creación
updatedAt	Date	-	NO	Aut o	Fecha de modificación

Restricciones

- Primary Key: _id

- **Unique Key:** (sectionCode, academicCycle)
- **Foreign Keys:** course, instructor, academicCycle, institution
- **Check:** capacity >= 1 AND capacity <= 200

Índices

```
{ sectionCode: 1, academicCycle: 1 } // Único
{ course: 1 }
{ instructor: 1 }
{ academicCycle: 1 }
```

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439017"),
  "course": ObjectId("507f1f77bcf86cd799439013"),
  "instructor": ObjectId("507f1f77bcf86cd799439014"),
  "academicCycle": ObjectId("507f1f77bcf86cd799439016"),
  "sectionCode": "A1",
  "capacity": 35,
  "approvalCriteria": {
    "mastery": {
      "required": true,
      "minPercentage": 75
    },
    "summativeAssessments": [],
    "completion": {
      "requiredLessonsPercentage": 80,
      "allAssignmentsRequired": true
    }
  },
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-02-15T10:00:00Z"),
  "updatedAt": ISODate("2025-02-15T10:00:00Z")
}
```

6. ENROLLMENTS

Propósito: Registra la matrícula de estudiantes en secciones.

Colección: enrollments

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
student	ObjectId	-	NO	-	FK → USERS
section	ObjectId	-	NO	-	FK → SECTIONS
status	String	-	NO	enrolled	Estado de matrícula
finalGrade	Number	-	SÍ	null	Calificación final (0-100)
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de matrícula
updatedAt	Date	-	NO	Auto	Fecha de modificación

Valores Permitidos

status:

- enrolled - Matriculado (activo)
- passed - Aprobado
- failed - Desaprobado
- withdrawn - Retirado

Restricciones

- **Primary Key:** _id
- **Unique Key:** (student, section)
- **Foreign Keys:** student, section, institution
- **Check:** finalGrade >= 0 AND finalGrade <= 100 OR NULL
- **Check:** status IN ('enrolled','passed','failed','withdrawn')

Índices

```
{ student: 1, section: 1 } // Único
{ section: 1, status: 1 }
{ student: 1, status: 1 }
```

Reglas de Negocio

1. Un estudiante no puede matricularse dos veces en la misma sección
2. Se valida capacidad disponible antes de matricular
3. Se validan prerequisitos (solo universities)

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439018"),
  "student": ObjectId("507f1f77bcf86cd799439012"),
  "section": ObjectId("507f1f77bcf86cd799439017"),
  "status": "enrolled",
  "finalGrade": null,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-03-01T08:30:00Z"),
  "updatedAt": ISODate("2025-03-01T08:30:00Z")
}
```

7. CAREERS

Propósito: Programas académicos (solo universidades).

Colección: careers

Estructura de Campos

Campo	Tipo	Tamaño	Nul	Defau	Descripción
_id	ObjectId	-	NO	Auto	Identificador único

name	String	200	NO	-	Nombre de la carrera
description	String	2000	NO	-	Descripción
degrees	Array[String]	-	NO	-	Títulos otorgados
duration	String	50	NO	-	Duración (ej: "10 Ciclos")
coordinator	ObjectId	-	SÍ	null	FK → USERS (coordinator)
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
curriculum	Array[Object]	-	SÍ	[]	Malla curricular
curriculum[].cycleNumber	Number	-	NO	-	Número de ciclo (1-20)
curriculum[].courses	Array[ObjectID]	-	NO	-	FK → COURSES
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** _id
- **Unique Key:** name
- **Foreign Keys:** coordinator, institution, curriculum[].courses []
- **Check:** degrees.length >= 1
- **Check:** cycleNumber >= 1 AND cycleNumber <= 20

Índices

```
{ name: 1 } // Único
{ institution: 1 }
{ coordinator: 1 }
{ "curriculum.cycleNumber": 1 }
```

Reglas de Negocio

1. Solo para instituciones tipo university
2. Un curso no puede estar en ciclo ≤ al de sus prerequisitos
3. Un curso solo aparece una vez en la malla

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439019"),
  "name": "Ingeniería de Sistemas",
  "description": "Programa de formación en ingeniería de software",
  "degrees": ["Bachiller en Ingeniería", "Ingeniero de Sistemas"],
```

```

    "duration": "10 Ciclos",
    "coordinator": ObjectId("507f1f77bcf86cd79943901a"),
    "institution": ObjectId("507f1f77bcf86cd799439011"),
    "curriculum": [
      {
        "cycleNumber": 1,
        "courses": [
          ObjectId("507f1f77bcf86cd799439013"),
          ObjectId("507f1f77bcf86cd799439015")
        ]
      }
    ],
    "createdAt": ISODate("2025-01-05T12:00:00Z"),
    "updatedAt": ISODate("2025-02-10T15:30:00Z")
  }

```

8. ACADEMIC_RECORDS

Propósito: Expediente académico del estudiante (solo universities).

Colección: academicrecords

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
student	ObjectId	-	NO	-	FK → USERS (único)
career	ObjectId	-	NO	-	FK → CAREERS
status	String	-	NO	active	Estado del expediente
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de inscripción
updatedAt	Date	-	NO	Auto	Fecha de modificación

Valores Permitidos

status:

- active - Activo
- graduated - Graduado
- withdrawn - Retirado

Restricciones

- **Primary Key:** _id
- **Unique Key:** student
- **Foreign Keys:** student, career, institution

Índices

```

{ student: 1 } // Único
{ career: 1 }
{ institution: 1, status: 1 }

```

Reglas de Negocio

1. Un estudiante solo puede tener un expediente académico
2. Solo para instituciones tipo university

Ejemplo

```
{  
  "_id": ObjectId("507f1f77bcf86cd79943901b"),  
  "student": ObjectId("507f1f77bcf86cd799439012"),  
  "career": ObjectId("507f1f77bcf86cd799439019"),  
  "status": "active",  
  "institution": ObjectId("507f1f77bcf86cd799439011"),  
  "createdAt": ISODate("2025-03-01T09:00:00Z"),  
  "updatedAt": ISODate("2025-03-01T09:00:00Z")  
}
```

9. MODULES

Propósito: Unidades de contenido didáctico en biblioteca personal del profesor.

Colección: modules

Estructura de Campos

Campo	Tipo	Tamaño	Nul l	Defaul t	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
title	String	200	NO	-	Título del módulo
description	String	2000	SÍ	null	Descripción
owner	ObjectId	-	NO	-	FK → USERS (profesor)
publishedIn	Array[Object]	-	SÍ	[]	Secciones donde se publicó
publishedIn[].secti on	ObjectId	-	NO	-	FK → SECTIONS
order	Number	-	NO	1	Orden de presentación
institution	ObjectId	-	NO	-	FK → INSTITUTION S
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** `_id`
- **Foreign Keys:** `owner, publishedIn[].section, institution`
- **Check:** `order >= 1`

Índices

```
{ owner: 1 }
{ "publishedIn.section": 1 }
{ institution: 1 }
{ owner: 1, institution: 1 }
```

Reglas de Negocio

1. El propietario debe tener rol `professor`
2. Un módulo puede publicarse en múltiples secciones
3. El orden determina la secuencia dentro de la sección

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd79943901c"),
  "title": "Límites y Continuidad",
  "description": "Conceptos fundamentales de límites",
  "owner": ObjectId("507f1f77bcf86cd799439014"),
  "publishedIn": [
    { "section": ObjectId("507f1f77bcf86cd799439017") }
  ],
  "order": 1,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-02-20T10:00:00Z"),
  "updatedAt": ISODate("2025-03-01T14:00:00Z")
}
```

10. LESSONS

Propósito: Contenido educativo individual dentro de un módulo.

Colección: `lessons`

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
<code>_id</code>	<code>ObjectId</code>	-	NO	Auto	Identificador único
<code>title</code>	<code>String</code>	200	NO	-	Título de la lección
<code>module</code>	<code>ObjectId</code>	-	NO	-	FK → MODULES
<code>content</code>	<code>String</code>	50000	SÍ	null	Contenido Markdown
<code>contentType</code>	<code>String</code>	-	NO	text	Tipo de contenido
<code>fileUrl</code>	<code>String</code>	500	SÍ	null	URL de archivo
<code>order</code>	<code>Number</code>	-	NO	-	Orden en el módulo
<code>institution</code>	<code>ObjectId</code>	-	NO	-	FK → INSTITUTIONS
<code>createdAt</code>	<code>Date</code>	-	NO	Auto	Fecha de creación
<code>updatedAt</code>	<code>Date</code>	-	NO	Auto	Fecha de modificación

Valores Permitidos

contentType:

- `text` - Texto Markdown
- `video_url` - URL de video (YouTube, Vimeo)
- `document_url` - URL de documento (PDF, Word)
- `slides_url` - URL de presentación

Restricciones

- **Primary Key:** `_id`
- **Foreign Keys:** `module`, `institution`
- **Check:** `order >= 1`
- **Check:** `contentType IN ('text', 'video_url', 'document_url', 'slides_url')`

Índices

```
{ module: 1, order: 1 }
{ institution: 1 }
{ module: 1 }
```

Reglas de Negocio

1. Si `contentType` es `text`, se usa `content`
2. Para otros tipos, se usa `fileUrl` o `content` como URL
3. El orden determina la secuencia de presentación

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd79943901d"),
  "title": "Introducción a los Límites",
  "module": ObjectId("507f1f77bcf86cd79943901c"),
  "content": "# Límites\n\nUn límite es el valor al que se aproxima...",  

  "contentType": "text",
  "fileUrl": null,
  "order": 1,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-02-20T11:00:00Z"),
  "updatedAt": ISODate("2025-02-20T11:00:00Z")
}
```

11. LESSON_COMPLETIONS

Propósito: Rastrea el progreso de estudiantes en lecciones.

Colección: lessoncompletions

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
student	ObjectId	-	NO	-	FK → USERS
lesson	ObjectId	-	NO	-	FK → LESSONS
section	ObjectId	-	NO	-	FK → SECTIONS (contexto)
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de completado
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** _id
- **Unique Key:** (student, lesson)
- **Foreign Keys:** student, lesson, section, institution

Índices

```
{ student: 1, lesson: 1 } // Único
{ section: 1, student: 1 }
{ institution: 1 }
```

Reglas de Negocio

1. Una lección solo se marca como completada una vez
2. Se usa para calcular % de completitud en criterios de aprobación

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd79943901e"),
  "student": ObjectId("507f1f77bcf86cd799439012"),
  "lesson": ObjectId("507f1f77bcf86cd79943901d"),
  "section": ObjectId("507f1f77bcf86cd799439017"),
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-03-05T16:30:00Z"),
  "updatedAt": ISODate("2025-03-05T16:30:00Z")
}
```

12. QUESTIONS

Propósito: Banco de preguntas para evaluaciones adaptativas.

Colección: questions

Estructura de Campos

Campo	Tipo	Tamaño	Nul l	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
module	ObjectId	-	NO	-	FK → MODULES

owner	ObjectId	-	NO	-	FK → USERS (profesor)
questionText	String	2000	NO	-	Texto de la pregunta
options	Array[Object]	-	NO	-	Opciones de respuesta
options[]._id	ObjectId	-	NO	Auto	ID de la opción
options[] .text	String	500	NO	-	Texto de la opción
options[] .isCorrect	Boolean	-	NO	-	¿Es correcta?
difficulty	Number	-	NO	-	Dificultad (1-5)
questionType	String	-	NO	multiple_choice	Tipo de pregunta
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Valores Permitidos

questionType:

- multiple_choice - Opción múltiple
- true_false - Verdadero/Falso

difficulty:

- 1 - Muy fácil
- 2 - Fácil
- 3 - Medio
- 4 - Difícil
- 5 - Muy difícil

Restricciones

- **Primary Key:** _id
- **Foreign Keys:** module, owner, institution
- **Check:** difficulty >= 1 AND difficulty <= 5
- **Check:** options.length >= 2
- **Check:** Al menos una opción debe tener isCorrect: true

Índices

```
{ module: 1, difficulty: 1 }
```

```

{ owner: 1 }
{ institution: 1 }
{ module: 1 }
{ questionText: "text" } // Búsqueda de texto completo

```

Reglas de Negocio

1. Debe tener mínimo 2 opciones
2. Al menos una opción debe ser correcta
3. La dificultad es clave para el algoritmo adaptativo
4. Para `true_false`, solo 2 opciones

Ejemplo

```

{
  "_id": ObjectId("507f1f77bcf86cd79943901f"),
  "module": ObjectId("507f1f77bcf86cd79943901c"),
  "owner": ObjectId("507f1f77bcf86cd799439014"),
  "questionText": "¿Cuál es el límite de  $f(x)=1/x$  cuando  $x \rightarrow \infty$ ?",
  "options": [
    {
      "_id": ObjectId("507f1f77bcf86cd799439020"),
      "text": "0",
      "isCorrect": true
    },
    {
      "_id": ObjectId("507f1f77bcf86cd799439021"),
      "text": "1",
      "isCorrect": false
    },
    {
      "_id": ObjectId("507f1f77bcf86cd799439022"),
      "text": "Infinito",
      "isCorrect": false
    }
  ],
  "difficulty": 2,
  "questionType": "multiple_choice",
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-02-22T14:00:00Z"),
  "updatedAt": ISODate("2025-02-22T14:00:00Z")
}

```

13. EVALUATION_SESSIONS

Propósito: Sesión activa de evaluación adaptativa.

Colección: evaluationsessions

Estructura de Campos

Campo	Tipo	Tamaño	Nul l	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
student	ObjectId	-	NO	-	FK → USERS

module	ObjectId	-	NO	-	FK → MODULES
questionsAnswered	Array[ObjectId]	-	SÍ	[]	FK → QUESTIONS
currentMastery	Number	-	NO	25	Maestría actual (0-100)
status	String	-	NO	in_progress	Estado de la sesión
score	Object	-	NO	{}	Puntajes
score.correct	Number	-	NO	0	Respuestas correctas
score.incorrect	Number	-	NO	0	Respuestas incorrectas
institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de inicio
updatedAt	Date	-	NO	Auto	Fecha de modificación

Valores Permitidos

status:

- in_progress - En progreso
- completed - Completada

Restricciones

- **Primary Key:** _id
- **Foreign Keys:** student, module, questionsAnswered[], institution
- **Check:** currentMastery >= 0 AND currentMastery <= 100
- **Check:** status IN ('in_progress', 'completed')

Índices

```
{
  student: 1, module: 1
}
{
  status: 1
}
{
  institution: 1
}
{
  student: 1
}
{
  createdAt: 1 } // TTL index para limpieza automática
```

Algoritmo Adaptativo

1. **Inicio:** currentMastery = 25
2. **Respuesta correcta:** currentMastery += difficulty * 2
3. **Respuesta incorrecta:** currentMastery -= (6 - difficulty)
4. **Límites:** 0 ≤ currentMastery ≤ 100
5. **Finalización:** 10 preguntas o maestría ≥ 95%

Reglas de Negocio

1. Una sesión finaliza después de 10 preguntas o al alcanzar 95% de maestría
2. Las sesiones `in_progress` se eliminan automáticamente después de 7 días (índice TTL)
3. La próxima pregunta se selecciona según maestría actual

Ejemplo

```
{  
  "_id": ObjectId("507f1f77bcf86cd799439023"),  
  "student": ObjectId("507f1f77bcf86cd799439012"),  
  "module": ObjectId("507f1f77bcf86cd79943901c"),  
  "questionsAnswered": [  
    ObjectId("507f1f77bcf86cd79943901f"),  
    ObjectId("507f1f77bcf86cd799439024")  
  ],  
  "currentMastery": 35,  
  "status": "in_progress",  
  "score": {  
    "correct": 1,  
    "incorrect": 1  
  },  
  "institution": ObjectId("507f1f77bcf86cd799439011"),  
  "createdAt": ISODate("2025-03-10T14:00:00Z"),  
  "updatedAt": ISODate("2025-03-10T14:05:00Z")  
}
```

14. MASTERY

Propósito: Registro del nivel más alto de maestría alcanzado por estudiante en cada módulo.

Colección: masteries

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
<code>_id</code>	ObjectId	-	NO	Auto	Identificador único
<code>student</code>	ObjectId	-	NO	-	FK → USERS
<code>module</code>	ObjectId	-	NO	-	FK → MODULES
<code>highestMasteryScore</code>	Number	-	NO	0	Puntaje más alto (0-100)
<code>institution</code>	ObjectId	-	NO	-	FK → INSTITUTIONS
<code>createdAt</code>	Date	-	NO	Auto	Fecha de primer registro
<code>updatedAt</code>	Date	-	NO	Auto	Fecha de última mejora

Restricciones

- **Primary Key:** `_id`
- **Unique Key:** `(student, module)`
- **Foreign Keys:** `student, module, institution`
- **Check:** `highestMasteryScore >= 0 AND highestMasteryScore <= 100`

Índices

```
{ student: 1, module: 1 } // Único
{ module: 1, highestMasteryScore: -1 } // Rankings
{ institution: 1 }
```

Reglas de Negocio

1. Solo se guarda el puntaje más alto (`$max operator`)
2. Se actualiza automáticamente al finalizar una sesión de evaluación
3. Se usa para validar criterios de aprobación de secciones

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439025"),
  "student": ObjectId("507f1f77bcf86cd799439012"),
  "module": ObjectId("507f1f77bcf86cd79943901c"),
  "highestMasteryScore": 78,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-03-10T14:10:00Z"),
  "updatedAt": ISODate("2025-03-15T16:20:00Z")
}
```

15. PERFORMANCE_LOGS

Propósito: Log detallado de cada respuesta del estudiante para analytics.

Colección: `performancelogs`

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
<code>_id</code>	<code>ObjectId</code>	-	NO	Auto	Identificador único
<code>student</code>	<code>ObjectId</code>	-	NO	-	FK → USERS
<code>question</code>	<code>ObjectId</code>	-	NO	-	FK → QUESTIONS
<code>module</code>	<code>ObjectId</code>	-	NO	-	FK → MODULES
<code>isCorrect</code>	<code>Boolean</code>	-	NO	-	¿Respuesta correcta?
<code>difficulty</code>	<code>Number</code>	-	NO	-	Dificultad (1-5)
<code>institution</code>	<code>ObjectId</code>	-	NO	-	FK → INSTITUTIONS
<code>createdAt</code>	<code>Date</code>	-	NO	Auto	Fecha del intento
<code>updatedAt</code>	<code>Date</code>	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** `_id`
- **Foreign Keys:** `student, question, module, institution`

- **Check:** difficulty >= 1 AND difficulty <= 5

Índices

```
{
  student: 1, module: 1
}
{
  question: 1
}
{
  module: 1, isCorrect: 1
}
{
  institution: 1
}
{
  createdAt: -1
}
{
  student: 1, module: 1, createdAt: -1
}
```

Uso en Analytics

1. **Preguntas más difíciles:** Agregar por question donde isCorrect: false
2. **Estudiantes con dificultades:** Filtrar por tasa de éxito < 40%
3. **Progreso temporal:** Analizar por createdAt

Reglas de Negocio

1. Se crea un log por cada respuesta en una sesión de evaluación
2. No se puede modificar una vez creado (inmutable)
3. Se usa para generar reportes y analytics

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439026"),
  "student": ObjectId("507f1f77bcf86cd799439012"),
  "question": ObjectId("507f1f77bcf86cd79943901f"),
  "module": ObjectId("507f1f77bcf86cd79943901c"),
  "isCorrect": true,
  "difficulty": 2,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-03-10T14:02:00Z"),
  "updatedAt": ISODate("2025-03-10T14:02:00Z")
}
```

16. ASSIGNMENTS

Propósito: Tareas asignadas a estudiantes en una sección.

Colección: assignments

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
_id	ObjectId	-	NO	Auto	Identificador único
section	ObjectId	-	NO	-	FK → SECTIONS
title	String	200	NO	-	Título de la tarea
instructions	String	5000	SÍ	null	Instrucciones detalladas
dueDate	Date	-	SÍ	null	Fecha de vencimiento
pointsPossible	Number	-	NO	100	Puntaje máximo

institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** `_id`
- **Foreign Keys:** `section, institution`
- **Check:** `pointsPossible >= 0 AND pointsPossible <= 1000`

Índices

```
{ section: 1 }
{ section: 1, dueDate: 1 }
{ institution: 1 }
```

Reglas de Negocio

1. Solo el instructor de la sección puede crear tareas
2. Las tareas pueden ser parte de criterios de aprobación

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439027"),
  "section": ObjectId("507f1f77bcf86cd799439017"),
  "title": "Tarea 1: Ejercicios de Límites",
  "instructions": "Resolver los ejercicios del 1 al 20 del libro...",
  "dueDate": ISODate("2025-03-20T23:59:59Z"),
  "pointsPossible": 100,
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-03-08T10:00:00Z"),
  "updatedAt": ISODate("2025-03-08T10:00:00Z")
}
```

17. SUBMISSIONS

Propósito: Entregas de tareas realizadas por estudiantes.

Colección: submissions

Estructura de Campos

Campo	Tipo	Tamaño	Null	Default	Descripción
<code>_id</code>	ObjectId	-	NO	Auto	Identificador único
<code>assignment</code>	ObjectId	-	NO	-	FK → ASSIGNMENTS
<code>student</code>	ObjectId	-	NO	-	FK → USERS
<code>content</code>	String	10000	SÍ	null	Contenido de la entrega
<code>fileUrl</code>	String	500	SÍ	null	URL del archivo adjunto
<code>grade</code>	Number	-	SÍ	null	Calificación (0-100)
<code>feedback</code>	String	2000	SÍ	null	Retroalimentación
<code>submittedAt</code>	Date	-	NO	Auto	Fecha de entrega

institution	ObjectId	-	NO	-	FK → INSTITUTIONS
createdAt	Date	-	NO	Auto	Fecha de creación
updatedAt	Date	-	NO	Auto	Fecha de modificación

Restricciones

- **Primary Key:** `_id`
- **Unique Key:** `(assignment, student)`
- **Foreign Keys:** `assignment, student, institution`
- **Check:** `grade >= 0 AND grade <= 100 OR NULL`

Índices

```
{ assignment: 1, student: 1 } // Único
{ student: 1 }
{ assignment: 1, submittedAt: 1 }
{ institution: 1 }
```

Reglas de Negocio

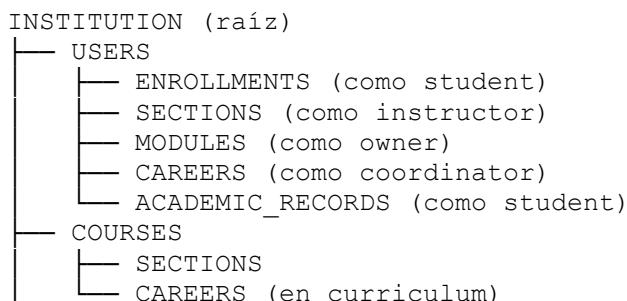
1. Un estudiante solo puede entregar una vez por tarea
2. Solo estudiantes matriculados pueden entregar
3. El profesor califica después de la entrega
4. Se puede entregar después de la fecha límite (tardía)

Ejemplo

```
{
  "_id": ObjectId("507f1f77bcf86cd799439028"),
  "assignment": ObjectId("507f1f77bcf86cd799439027"),
  "student": ObjectId("507f1f77bcf86cd799439012"),
  "content": "Adjunto archivo con las soluciones",
  "fileUrl": "/uploads/submissions/tareal-mlopez.pdf",
  "grade": 85,
  "feedback": "Buen trabajo. Revisar ejercicio 15.",
  "submittedAt": ISODate("2025-03-19T18:30:00Z"),
  "institution": ObjectId("507f1f77bcf86cd799439011"),
  "createdAt": ISODate("2025-03-19T18:30:00Z"),
  "updatedAt": ISODate("2025-03-22T10:15:00Z")
}
```

RESUMEN DE RELACIONES

Diagrama de Dependencias



```
└── ACADEMIC_CYCLES
    └── SECTIONS
    └── CAREERS
        └── ACADEMIC_RECORDS

MODULES (biblioteca del profesor)
└── LESSONS
└── QUESTIONS
└── EVALUATION_SESSIONS
└── MASTERY
└── PERFORMANCE_LOGS

SECTIONS (instancia de curso)
└── ENROLLMENTS
└── ASSIGNMENTS
    └── SUBMISSIONS
└── MODULES (publicados en)

LESSONS
└── LESSON_COMPLETIONS
```

ÍNDICES Y PERFORMANCE

Índices Críticos para Performance

1. **Multi-tenancy:**
 - o Todas las colecciones: { institution: 1 }
2. **Búsquedas frecuentes:**
 - o Users: { institution: 1, email: 1 } (único)
 - o Enrollments: { student: 1, section: 1 } (único)
 - o Sections: { academicCycle: 1, course: 1 }
3. **Evaluaciones adaptativas:**
 - o Questions: { module: 1, difficulty: 1 }
 - o Mastery: { student: 1, module: 1 } (único)
4. **Analytics:**
 - o Performance Logs: { student: 1, module: 1, createdAt: -1 }

CONSIDERACIONES DE SEGURIDAD

Cifrado y Protección

1. **Contraseñas:**
 - o Algoritmo: bcrypt
 - o Salt rounds: 10
 - o Longitud hash: 60 caracteres
 - o Nunca se devuelve en queries (.select('-password'))
2. **Tokens JWT:**
 - o Incluyen: userId, institutionId
 - o Expiración: 30 días
 - o Algoritmo: HS256
3. **Validación de acceso:**

- Todas las queries filtran por institution
- Middleware protect valida token
- Middleware authorize valida roles

Principio de Menor Privilegio

```
// Usuario de aplicación
{
  user: "adaptatest_app",
  roles: [{ role: "readWrite", db: "adaptatest_db" }]
}

// Usuario de solo lectura
{
  user: "adaptatest_READONLY",
  roles: [{ role: "read", db: "adaptatest_db" }]
}
```

MANTENIMIENTO Y LIMPIEZA

Tareas Automáticas

- 1. TTL Index en EVALUATION_SESSIONS:**
 - Elimina sesiones in_progress > 7 días
- 2. Validaciones de Integridad:**
 - Schema validation en MongoDB
 - Validaciones en Mongoose schemas
 - Middleware pre/post hooks

Tareas Manuales Recomendadas

- 1. Backups diarios:**
2. mongodump --db adaptatest_db --out /backups/\$(date +%Y%m%d)
- 3. Análisis de índices no utilizados:**
4. db.collection.aggregate([{\$indexStats: {}}])
- 5. Compactación de colecciones:**
6. db.runCommand({ compact: 'performancelogs' })

GLOSARIO DE TÉRMINOS

- **Multi-tenant:** Arquitectura donde múltiples instituciones comparten la misma base de datos
- **Maestría (Mastery):** Nivel de dominio del estudiante en un módulo (0-100%)
- **Evaluación Adaptativa:** Sistema que ajusta la dificultad según el desempeño
- **Malla Curricular:** Estructura de cursos por ciclo en una carrera
- **Ciclo Académico:** Período de tiempo (semestre, trimestre)
- **ODM:** Object Document Mapper (Mongoose para MongoDB)
- **TTL:** Time To Live (eliminación automática)

ENLACE A GITHUB Y CONTROL DE VERSIONES

Enlace al Repositorio

El código fuente completo del proyecto "AdaptaTest" está alojado en un repositorio de GitHub, permitiendo el control de versiones, la colaboración y la revisión continua del código.

Enlace al repositorio: <https://github.com/ezzadjg/adapta-test>

Historial de Commits:

Primera Versión:

Commits on Aug 20, 2025		
upp	ezzADIG committed on Aug 20	2bbb5e7 ⌂ <>
up	ezzADIG committed on Aug 20	63ef9f7 ⌂ <>
update	ezzADIG committed on Aug 20	598c58f ⌂ <>
update	ezzADIG committed on Aug 20	d08a37d ⌂ <>
update	ezzADIG committed on Aug 20	110836f ⌂ <>
Commits on Aug 18, 2025		
update	ezzADIG committed on Aug 18	36f2c6d ⌂ <>
Update	ezzADIG committed on Aug 18	880a5e5 ⌂ <>
first commit	ezzADIG committed on Aug 18	5496ad6 ⌂ <>
Commits on Aug 22, 2025		
update	ezzADIG committed on Aug 22	ad8de2c ⌂ <>
update	ezzADIG committed on Aug 22	fb6ae90 ⌂ <>
Commits on Aug 21, 2025		
update	ezzADIG committed on Aug 21	6be4a91 ⌂ <>
update	ezzADIG committed on Aug 21	81ca003 ⌂ <>
update	ezzADIG committed on Aug 21	a1f223f ⌂ <>
up	ezzADIG committed on Aug 21	7d6ac50 ⌂ <>
update	ezzADIG committed on Aug 21	665c97d ⌂ <>

Commits on Aug 25, 2025

- update**
ezzADIG committed on Aug 25
c123424

Commits on Aug 24, 2025

- update**
ezzADIG committed on Aug 24
d31b518
- update**
ezzADIG committed on Aug 24
e89b23b

Commits on Aug 23, 2025

- update**
ezzADIG committed on Aug 23
a8f8c16
- update**
ezzADIG committed on Aug 23
59b3ed5
- update**
ezzADIG committed on Aug 23
cccd9913

Commit principal que inició la segunda versión del proyecto:

update multi tenant

main

1 parent: c123424 commit a9db096

models

- academicCycleModel.js
- academicRecordModel.js
- assignmentModel.js
- careerModel.js
- courseModel.js
- enrollmentModel.js
- evaluationSessionModel...
- institutionModel.js
- lessonCompletionMode...
- lessonModel.js
- masteryModel.js
- moduleModel.js
- performanceLogModel.js
- questionModel.js
- sectionModel.js
- submissionModel.js

20 files changed +455 -283 lines changed

adapta-test-backend/src/middleware/authMiddleware.js

```

1  @@ -1,44 +1,53 @@
2  - const jwt = require('jsonwebtoken');
3  - const User = require('../models/userModel');
4  +
5  + const jwt = require('jsonwebtoken');
6  + const User = require('../models/userModel');
7  +
8  + const protect = async (req, res, next) => {
9  +     let token;
10 +
11     if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {
12         try {
13             // 1. Obtener el token del header (Bearer TOKEN)
14             token = req.headers.authorization.split(' ')[1];
15
16             // 2. Verificar el token
17             const decoded = jwt.verify(token, process.env.JWT_SECRET);
18
19             // 3. Obtener los datos del usuario desde la BD (sin la contraseña)
20             // + adjuntarlos al objeto de la respuesta (req.user)
21         } catch (err) {
22             return res.status(401).json({ message: 'No se pudo validar el token' });
23         }
24     }
25
26     next();
27 }

```

Ajustes a las lógica y solución de errores tras el cambio, dando inicio a la versión actual.

Commits on Oct 15, 2025

- up**
ezzADIG committed 2 weeks ago
41a7000

Commits on Oct 13, 2025

- up**
ezzADIG committed 2 weeks ago
ddae20

Commits on Oct 12, 2025

- up**
ezzADIG committed 3 weeks ago
4ed9b1a
- up**
ezzADIG committed 3 weeks ago
58d4b07

Commits on Oct 11, 2025

- up**
ezzADIG committed 3 weeks ago
bdd3295
- up**
ezzADIG committed 3 weeks ago
893a305
- up**
ezzADIG committed 3 weeks ago
6c63be2
- up**
ezzADIG committed 3 weeks ago
924228f

Control de Versiones

El desarrollo de "AdaptaTest" ha seguido un proceso iterativo, evolucionando a través de tres versiones principales. Cada versión ha incrementado la complejidad y funcionalidad del sistema. A la fecha, el proyecto se encuentra en el desarrollo de la **Versión 3.0**.

- **v 1.0: Backend Funcional y Pruebas Iniciales** La primera versión del proyecto se centró en la validación del backend. El objetivo principal era entregar un servidor funcional, pero diseñado para operar en una sola institución (modelo *single-tenant*). El frontend desarrollado en esta etapa era mínimo y su propósito principal era servir como herramienta de prueba para validar la funcionalidad del backend.
- **v 2.0: Transición a Plataforma Multi-Institucional** Esta versión representó el cambio arquitectónico más significativo del proyecto. El sistema migró de un modelo de página institucional única a una plataforma de gestión de aprendizaje (LMS) completa, similar en concepto a Canvas. Este cambio introdujo la arquitectura *multi-tenant*, lo cual se reflejó en la implementación de una página de inicio y la adición de un selector de instituciones en la pantalla de "Iniciar Sesión".
- **v 3.0: Versión Actual y Refinamiento de UI/UX** Esta es la versión que se encuentra actualmente en desarrollo. El enfoque principal de esta etapa es el refinamiento de la experiencia de usuario (UX) y la interfaz de usuario (UI), aplicando un sistema de diseño consistente. Paralelamente, se están corrigiendo y estabilizando los problemas de lógica de negocio derivados de la compleja transición a la arquitectura de plataforma de la v 2.0.

CAPTURAS DE LA APLICACIÓN

Arquitectura y Tecnologías (MVC / DAO)

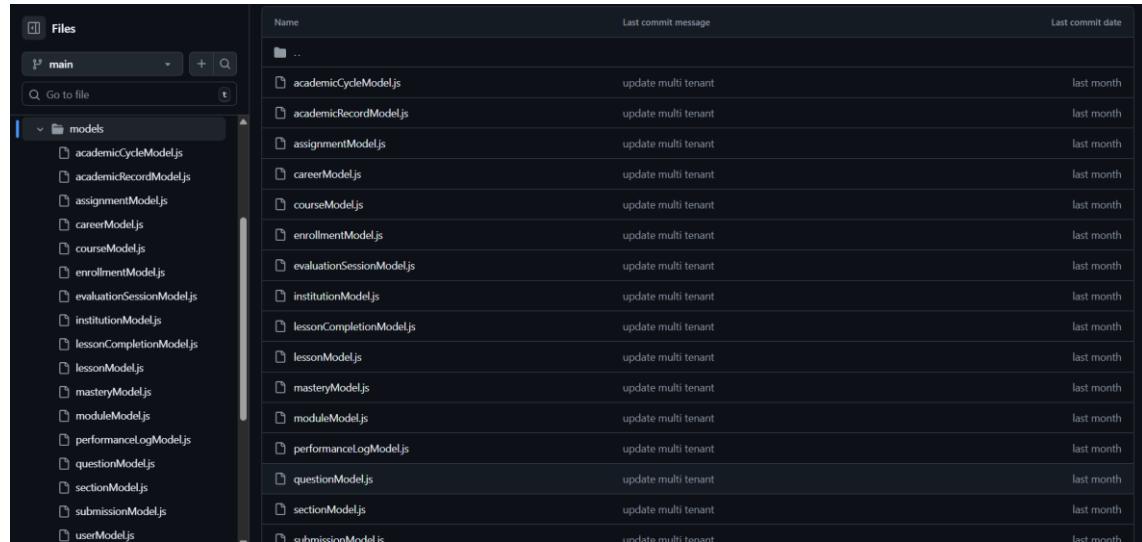
El sistema está desarrollado con una arquitectura moderna de aplicación web, separando claramente las responsabilidades del **Frontend** (cliente) y el **Backend** (servidor).

Arquitectura del Backend (adapta-test-backend)

El backend sigue una arquitectura de API RESTful basada en el patrón **Modelo-Vista-Controlador (MVC)**, complementada con una capa de **Servicios** para la lógica de negocio compleja.

Modelos (M): Definen los esquemas de la base de datos y la lógica de acceso a datos (DAO) utilizando Mongoose.

[Ubicación]: adapta-test-backend/src/models/



The screenshot shows a file explorer interface with a sidebar and a main content area. The sidebar on the left has a 'Files' tab, a dropdown menu set to 'main', a '+' button, a search bar, and a 'Go to file' input field. Below these are sections for 'Recent' and 'Recent locations'. The main content area shows a 'models' folder expanded, containing numerous files named like 'academicCycleModel.js', 'academicRecordModel.js', etc. To the right of the file list is a table titled 'Last commit message' with columns for 'Name', 'Last commit message', and 'Last commit date'. The table lists 21 entries, each corresponding to a file in the 'models' folder, all with the message 'update multi tenant' and a timestamp of 'last month'.

Name	Last commit message	Last commit date
..		
academicCycleModel.js	update multi tenant	last month
academicRecordModel.js	update multi tenant	last month
assignmentModel.js	update multi tenant	last month
careerModel.js	update multi tenant	last month
courseModel.js	update multi tenant	last month
enrollmentModel.js	update multi tenant	last month
evaluationSessionModel.js	update multi tenant	last month
institutionModel.js	update multi tenant	last month
lessonCompletionModel.js	update multi tenant	last month
lessonModel.js	update multi tenant	last month
masteryModel.js	update multi tenant	last month
moduleModel.js	update multi tenant	last month
performanceLogModel.js	update multi tenant	last month
questionModel.js	update multi tenant	last month
sectionModel.js	update multi tenant	last month
submissionModel.js	update multi tenant	last month
userManager.js	update multi tenant	last month

Archivos de ejemplo: userModel.js, courseModel.js, sectionModel.js, questionModel.js.

```
userModel.js

const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");

const userSchema = mongoose.Schema(
{
  name: {
    type: String,
    required: [true, "Por favor, añade un nombre."],
  },
  email: {
    type: String,
    required: [true, "Por favor, añade un email."],
    match: [/^[\w\.-]+@[^\w\.-]+\.[^\w\.-]{2,}/, "Por favor, introduce un email válido."],
  },
  password: {
    type: String,
    required: [true, "Por favor, añade una contraseña."],
  },
  // CAMPO AÑADIDO: Referencia a la institución a la que pertenece el usuario
  role: {
    type: String,
    enum: [
      "student",
      "professor",
      "coordinator",
      "admin",
      "parent",
      "superadmin",
    ], // ← AÑADIR 'superadmin'
    required: true,
  },
  institution: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Institution",
    // Ahora es requerido solo si el rol NO es superadmin
    required: function () {
      return this.role !== "superadmin";
    },
  },
  // CAMPO AÑADIDO: Para estudiantes de colegio
  studentGrade: {
    type: String, // "1ro", "2do", "3ro", "4to", "5to"
  },
  // CAMPO AÑADIDO: Para padres, los IDs de los estudiantes que supervisa
  parentOf: [
    {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
    },
  ],
  {
    timestamps: true,
  }
);

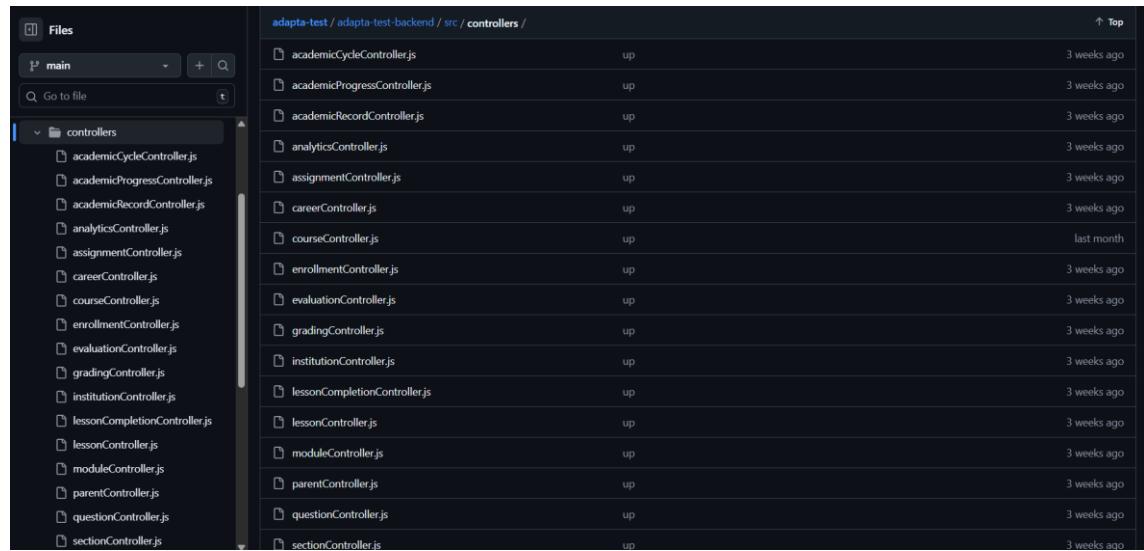
// ÍNDICE AÑADIDO: Asegura que un email sea único por institución
userSchema.index({ institution: 1, email: 1 }, { unique: true });

// Middleware para encriptar la contraseña (se mantiene igual)
userSchema.pre("save", async function (next) {
  if (!this.isModified("password")) {
    next();
  }
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});
```

Vistas (V): En esta arquitectura de API, la "vista" es la respuesta en formato JSON que se envía al cliente (el frontend de React).

Controladores (C): Contienen la lógica principal de la aplicación. Reciben las peticiones HTTP, interactúan con los modelos y servicios, y envían la respuesta.

[Ubicación]: adapta-test-backend/src/controllers/



The screenshot shows a file explorer interface with the following details:

- Left Panel (Files):** Shows a tree view of files under the "controllers" folder. The files listed are: academicCycleController.js, academicProgressController.js, academicRecordController.js, analyticsController.js, assignmentController.js, careerController.js, courseController.js, enrollmentController.js, evaluationController.js, gradingController.js, institutionController.js, lessonCompletionController.js, lessonController.js, moduleController.js, parentController.js, questionController.js, and sectionController.js.
- Right Panel (List View):** Shows a list of files from the "controllers" folder, ordered by modification date. All files were last modified 3 weeks ago, except for courseController.js which was last modified last month.

File	Last Modified
academicCycleController.js	3 weeks ago
academicProgressController.js	3 weeks ago
academicRecordController.js	3 weeks ago
analyticsController.js	3 weeks ago
assignmentController.js	3 weeks ago
careerController.js	3 weeks ago
courseController.js	last month
enrollmentController.js	3 weeks ago
evaluationController.js	3 weeks ago
gradingController.js	3 weeks ago
institutionController.js	3 weeks ago
lessonCompletionController.js	3 weeks ago
lessonController.js	3 weeks ago
moduleController.js	3 weeks ago
parentController.js	3 weeks ago
questionController.js	3 weeks ago
sectionController.js	3 weeks ago

Archivos de ejemplo: userController.js, courseController.js, evaluationController.js.

```
lessonController.js

const Lesson = require("../models/lessonModel");
const Module = require("../models/moduleModel");

// @desc   Crear una nueva lección en un módulo de la biblioteca del profesor
// @route  POST /api/modules/:moduleId/lessons
// @access Private/Professor (solo el dueño del módulo)
const createLessonInModule = async (req, res) => {
  const { title, content, contentType, fileUrl } = req.body;
  const { moduleId } = req.params;

  // 1. Verificar que el módulo padre existe DENTRO de la institución
  const parentModule = await Module.findOne({
    _id: moduleId,
    institution: req.institution._id,
  });

  if (!parentModule) {
    return res
      .status(404)
      .json({ message: "Módulo no encontrado en esta institución." });
  }

  // 2. Verificar que el usuario es el dueño del módulo
  if (parentModule.owner.toString() !== req.user._id.toString()) {
    return res
      .status(403)
      .json({ message: "No autorizado para añadir lecciones a este módulo." });
  }

  // 3. Crear la lección, asignando la institución del módulo padre
  const lesson = await Lesson.create({
    title,
    content,
    contentType,
    fileUrl,
    module: moduleId,
    order: (await Lesson.countDocuments({ module: moduleId })) + 1,
    institution: req.institution._id, // Asignar institución
  });

  res.status(201).json(lesson);
};

// @desc   Obtener todas las lecciones de un módulo de la biblioteca
// @route  GET /api/modules/:moduleId/lessons
// @access Private
const getLessonsFromModule = async (req, res) => {
  const { moduleId } = req.params;

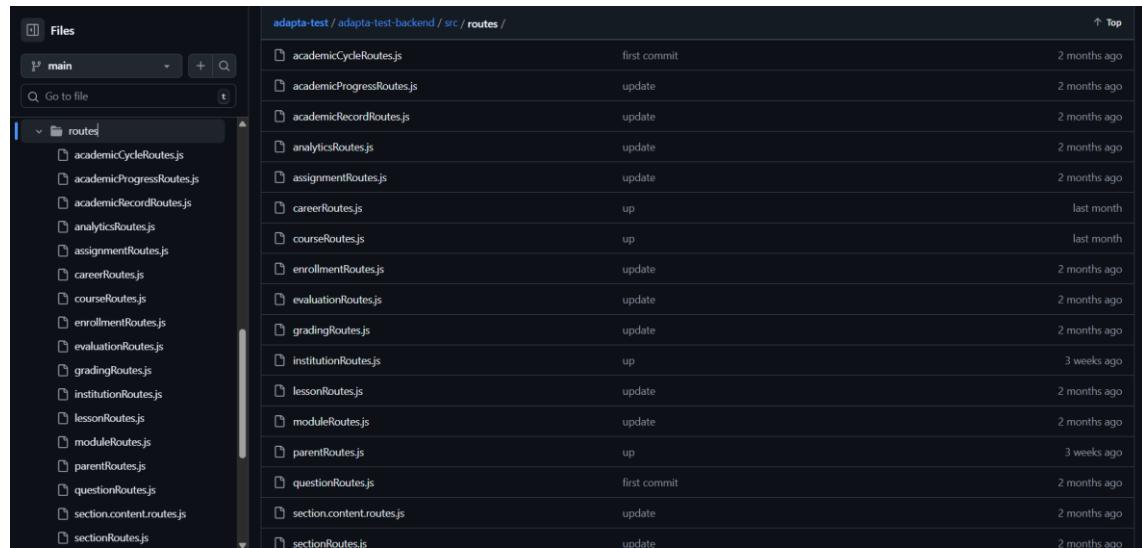
  // 1. Verificar que el módulo existe en la institución del usuario
  const parentModule = await Module.findOne({
    _id: moduleId,
    institution: req.institution._id,
  });
  if (!parentModule) {
    return res
      .status(404)
      .json({ message: "Módulo no encontrado en esta institución." });
  }

  // 2. Obtener las lecciones de ese módulo (ya están implícitamente seguras)
  const lessons = await Lesson.find({ module: moduleId }).sort("order");
  res.json(lessons);
};

module.exports = {
  createLessonInModule,
  getLessonsFromModule,
};
```

Rutas: Definen los *endpoints* de la API (ej. /api/users/login) y los asocian a las funciones del controlador correspondientes.

[Ubicación]: adapta-test-backend/src/routes/



The screenshot shows a GitHub repository interface for the 'routes' directory. On the left, there's a file tree view with the 'main' branch selected. The 'routes' folder contains numerous files, each ending in '.js'. On the right, a detailed commit history is displayed for each file, showing the author, commit message, and date of the last update. Most commits are dated '2 months ago', except for one 'first commit' and a few from '3 weeks ago'.

File	Author	Date
academicCycleRoutes.js	first commit	2 months ago
academicProgressRoutes.js	update	2 months ago
academicRecordRoutes.js	update	2 months ago
analyticsRoutes.js	update	2 months ago
assignmentRoutes.js	update	2 months ago
careerRoutes.js	up	last month
courseRoutes.js	up	last month
enrollmentRoutes.js	update	2 months ago
evaluationRoutes.js	update	2 months ago
gradingRoutes.js	update	2 months ago
institutionRoutes.js	up	3 weeks ago
lessonRoutes.js	update	2 months ago
moduleRoutes.js	update	2 months ago
parentRoutes.js	up	3 weeks ago
questionRoutes.js	first commit	2 months ago
section.content.routes.js	update	2 months ago
sectionRoutes.js	update	2 months ago

Archivos de ejemplo: userRoutes.js, courseRoutes.js, sectionRoutes.js.

```

● ● ● userRoutes.js

const express = require("express");
const router = express.Router();
const { protect, authorize } = require("../middleware/authMiddleware");
const {
  registerUser,
  loginUser,
  getUserProfile,
  getUsers,
  getAvailableInstitutions, // ← Importamos la nueva función
} = require("../controllers/userController");

// @route   GET /api/users/institutions
// @desc    Ruta PÚBLICA para obtener la lista de instituciones para el login
// @access  Public
router.get("/institutions", getAvailableInstitutions);

// @route   POST /api/users/login
// @desc    Login de usuario (ahora requiere institución)
// @access  Public
router.post("/login", loginUser);

// @route   GET /api/users/profile
// @desc    Obtener perfil del usuario logueado
// @access  Private
router.get("/profile", protect, getUserProfile);

// @route   GET /api/users
// @desc    Obtener todos los usuarios de la institución del admin
// @access  Private (Solo Admin de institución)
router
  .route("/")
  .get(protect, authorize("admin"), getUsers)
  // @route  POST /api/users
  // @desc   Registrar un nuevo usuario en la institución del admin
  // @access Private (Solo Admin de institución)
  .post(protect, authorize("admin"), registerUser);

module.exports = router;

```

Servicios: Encapsulan reglas de negocio reutilizables (ej. validaciones de prerequisitos, reglas por tipo de institución).

[Ubicación]: adapta-test-backend/src/services/

adapta-test / adapta-test-backend / src / services /			Add file	...
ezzADJG up		de069f2 · last month	History	
Name	Last commit message	Last commit date		
..				
institutionRulesService.js	up		last month	

Archivo de ejemplo: institutionRulesService.js.

```
○○○ institutionRulesService.js

class InstitutionRulesService {
    static getRulesForInstitution(institution) {
        const rules = {
            university: {
                allowedRoles: ['student', 'professor', 'coordinator', 'admin'],
                hasPrerequisites: true,
                hasCareerPrograms: true,
                enrollmentType: 'selective', // Por curso individual
                maxCoursesPerStudent: 8,
                gradingSystem: {
                    scale: '0-20',
                    passingGrade: 11,
                    allowsRetakes: true,
                    maxRetakes: 2
                },
                evaluationRules: {
                    adaptiveEnabled: true,
                    maxQuestions: 15,
                    difficultyLevels: 5,
                    masteryThreshold: 75
                }
            },
            high_school: {
                allowedRoles: ['student', 'professor', 'admin', 'parent'],
                hasPrerequisites: false,
                hasCareerPrograms: false,
                enrollmentType: 'automatic', // Por grado completo
                maxCoursesPerStudent: 12,
                gradingSystem: {
                    scale: '0-20',
                    passingGrade: 11,
                    allowsRetakes: true,
                    maxRetakes: 3,
                    requiresParentNotification: true
                },
                evaluationRules: {
                    adaptiveEnabled: true,
                    maxQuestions: 10,
                    difficultyLevels: 3,
                    masteryThreshold: 65
                }
            }
        };
        if (!rules[institution.type]) {
            throw new Error(`Reglas no definidas para el tipo de institución: ${institution.type}`);
        }

        // Combina las reglas base con las configuraciones personalizadas de la institución
        // Nota: Esto requiere que 'institution.settings' tenga la misma estructura.
        const institutionSettings = institution.settings || {};
        const finalRules = { ...rules[institution.type] };

        // Fusionamos las configuraciones anidadas
        if (institutionSettings.gradingSystem) {
            finalRules.gradingSystem = { ...finalRules.gradingSystem, ...institutionSettings.gradingSystem };
        }
        if (institutionSettings.evaluationRules) {
            finalRules.evaluationRules = { ...finalRules.evaluationRules, ...institutionSettings.evaluationRules };
        }

        return {
            ...finalRules,
            ...institutionSettings // Sobrescribimos las de primer nivel
        };
    }

    static validateUserRole(role, institution) {
        const rules = this.getRulesForInstitution(institution);
        return rules.allowedRoles.includes(role);
    }

    static async canEnrollInCourse(student, course, institution) {
        const rules = this.getRulesForInstitution(institution);

        if (rules.enrollmentType === 'automatic') {
            // Colegios: El curso debe ser para el grado del estudiante.
            // Asumimos que el modelo 'Course' tiene un campo 'targetGrade'.
            return course.targetGrade === student.studentGrade;
        }

        // Universidades: validar prerequisitos si es necesario
        if (rules.hasPrerequisites) {
            return this.hasMetPrerequisites(student, course);
        }

        return true; // Si no hay prerequisitos, se puede matricular.
    }

    // Placeholder para la lógica de prerequisitos (la implementaremos en detalle después)
    static async hasMetPrerequisites(student, course) {
        // TODO: Implementar la lógica para verificar si el estudiante ha aprobado los cursos
        // que son prerequisitos para 'course'.
        console.log(`Verificando prerequisitos para ${student.name} en el curso ${course.title}...`);
        return true; // Por ahora, siempre retorna true para no bloquear el desarrollo.
    }
}

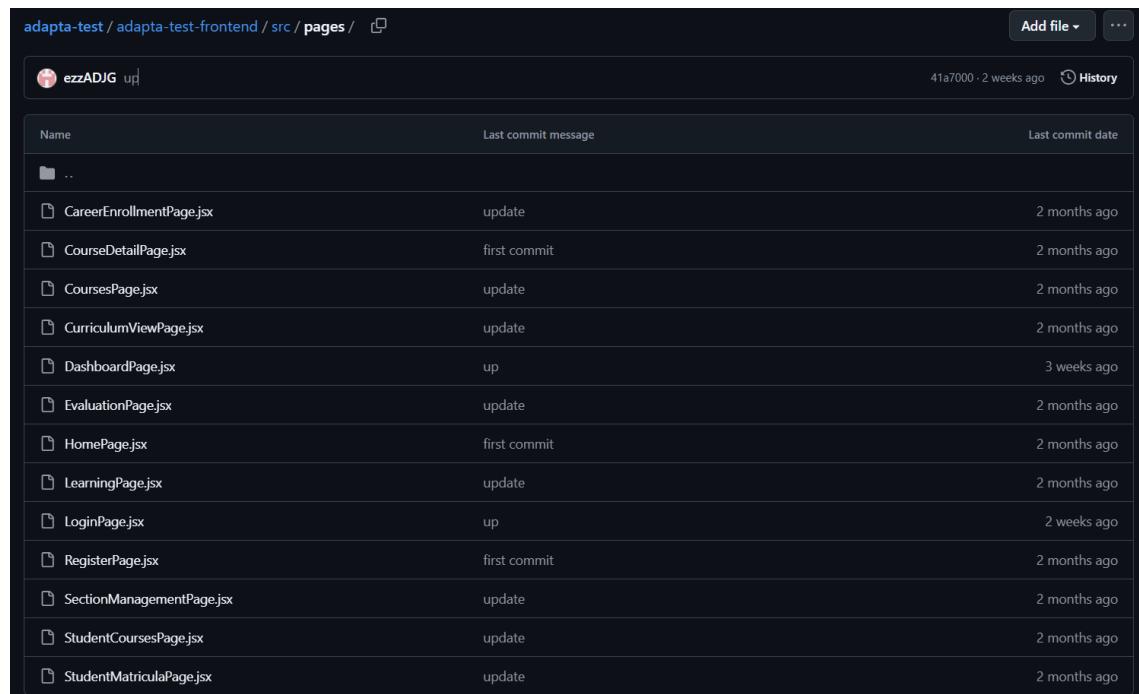
module.exports = InstitutionRulesService;
```

Arquitectura del Frontend (adapta-test-frontend)

El frontend está construido con **React** y sigue una arquitectura de **Componentes** con un manejo de estado global centralizado mediante **Redux**, siguiendo un patrón de diseño *Feature-Sliced*.

Pages (Páginas): Componentes de alto nivel que definen cada vista principal de la aplicación.

[Ubicación]: adapta-test-frontend/src/pages/



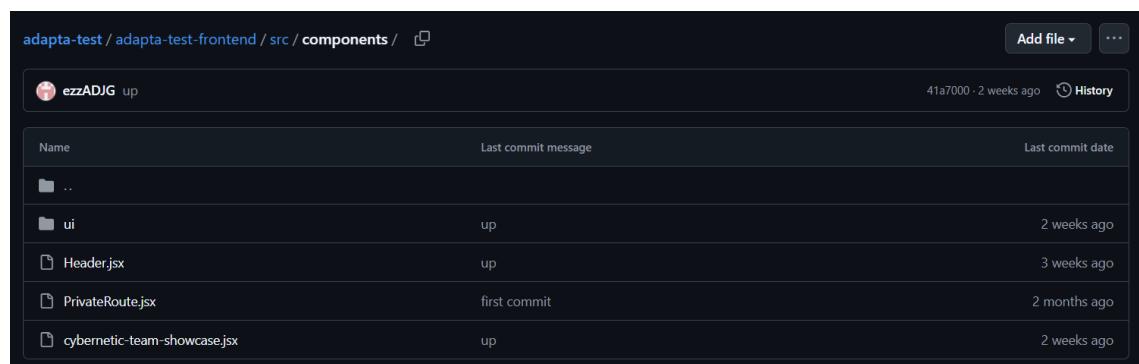
A screenshot of a GitHub repository interface showing the contents of the 'pages' directory. The directory contains several files: .., CareerEnrollmentPage.jsx, CourseDetailPage.jsx, CoursesPage.jsx, CurriculumViewPage.jsx, DashboardPage.jsx, EvaluationPage.jsx, HomePage.jsx, LearningPage.jsx, LoginPage.jsx, RegisterPage.jsx, SectionManagementPage.jsx, StudentCoursesPage.jsx, and StudentMatriculaPage.jsx. Each file has a timestamp indicating its last commit message and date.

Name	Last commit message	Last commit date
..		
CareerEnrollmentPage.jsx	update	2 months ago
CourseDetailPage.jsx	first commit	2 months ago
CoursesPage.jsx	update	2 months ago
CurriculumViewPage.jsx	update	2 months ago
DashboardPage.jsx	up	3 weeks ago
EvaluationPage.jsx	update	2 months ago
HomePage.jsx	first commit	2 months ago
LearningPage.jsx	update	2 months ago
LoginPage.jsx	up	2 weeks ago
RegisterPage.jsx	first commit	2 months ago
SectionManagementPage.jsx	update	2 months ago
StudentCoursesPage.jsx	update	2 months ago
StudentMatriculaPage.jsx	update	2 months ago

Archivos de ejemplo: LoginPage.jsx, DashboardPage.jsx, LearningPage.jsx.

Components (Componentes): Componentes de UI reutilizables (ej. Header.jsx) y componentes de lógica (ej. PrivateRoute.jsx).

[Ubicación]: adapta-test-frontend/src/components/



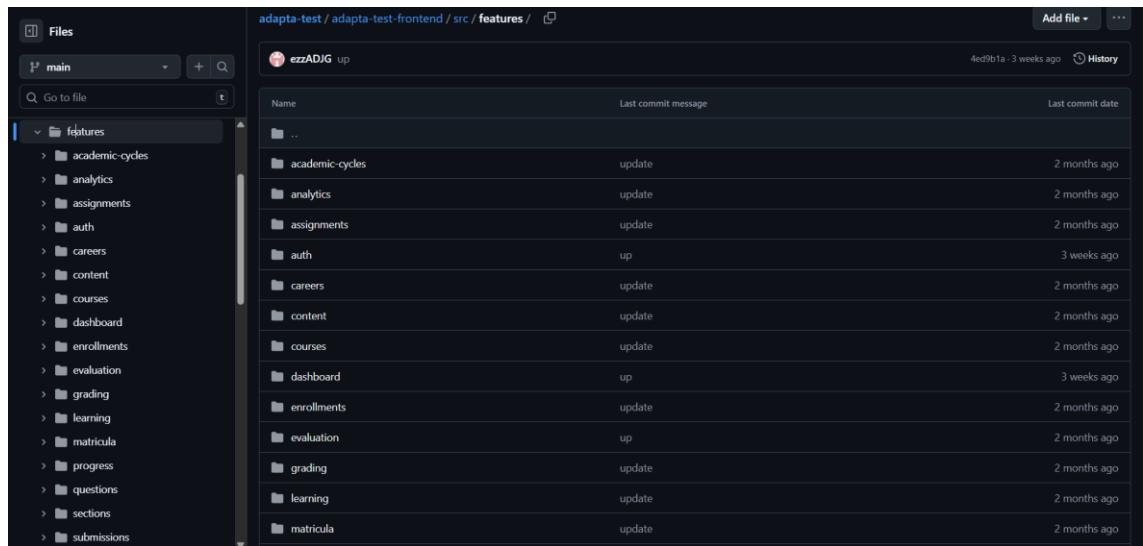
A screenshot of a GitHub repository interface showing the contents of the 'components' directory. The directory contains several files: .., ui, Header.jsx, PrivateRoute.jsx, and cybernetic-team-showcase.jsx. Each file has a timestamp indicating its last commit message and date.

Name	Last commit message	Last commit date
..		
ui	up	2 weeks ago
Header.jsx	up	3 weeks ago
PrivateRoute.jsx	first commit	2 months ago
cybernetic-team-showcase.jsx	up	2 weeks ago

Archivos de ejemplo: Header.jsx, PrivateRoute.jsx.

Features (Lógica de Estado): Es el núcleo de la lógica del frontend. Cada carpeta dentro de src/features/ agrupa la lógica de estado (*Slices* de Redux) y la lógica de comunicación con la API (**Servicios**) para una entidad específica.

[Ubicación]: adapta-test-frontend/src/features/



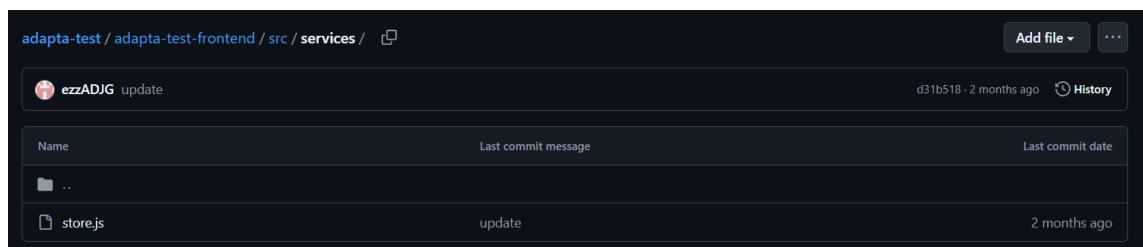
The screenshot shows a GitHub repository interface for the 'adaptas-test-frontend' project. The left sidebar shows a tree view of the 'src/features' directory, which contains subfolders like 'academic-cycles', 'analytics', 'assignments', 'auth', 'careers', 'content', 'courses', 'dashboard', 'enrollments', 'evaluation', 'grading', 'learning', 'matricula', 'progress', 'questions', 'sections', and 'submissions'. The right panel displays a list of commits from the 'ezzADJG' user. The commits are as follows:

Name	Last commit message	Last commit date
..		
academic-cycles	update	2 months ago
analytics	update	2 months ago
assignments	update	2 months ago
auth	up	3 weeks ago
careers	update	2 months ago
content	update	2 months ago
courses	update	2 months ago
dashboard	up	3 weeks ago
enrollments	update	2 months ago
evaluation	up	2 months ago
grading	update	2 months ago
learning	update	2 months ago
matricula	update	2 months ago
progress		
questions		
sections		
submissions		

Archivos de ejemplo: features/auth/authSlice.js, features/auth/authService.js, features/courses/courseSlice.js.

Store (Almacén Central): Archivo que configura el almacén central de Redux, combinando todos los *reducers* de las diferentes funcionalidades en un único estado global.

[Ubicación]: adapta-test-frontend/src/services/



The screenshot shows a GitHub repository interface for the 'adaptas-test-frontend' project. The left sidebar shows a tree view of the 'src/services' directory, which contains a single file named 'store.js'. The right panel displays a list of commits from the 'ezzADJG' user. The commits are as follows:

Name	Last commit message	Last commit date
..		
store.js	update	2 months ago

Archivo de ejemplo: store.js.

```
store.js

import { configureStore } from '@reduxjs/toolkit';
import authReducer from '../features/auth/authSlice';
import courseReducer from '../features/courses/courseSlice';
import enrollmentReducer from '../features/enrollments/enrollmentSlice';
import sectionReducer from '../features/sections/sectionSlice';
import contentReducer from '../features/content/contentSlice';
import assignmentReducer from '../features/assignments/assignmentSlice';
import learningReducer from '../features/learning/learningSlice';
import questionReducer from '../features/questions/questionSlice';
import evaluationReducer from '../features/evaluation/evaluationSlice';
import careerReducer from '../features/careers/careerSlice';
import usersReducer from '../features/users/usersSlice';
import academicCycleReducer from '../features/academic-cycles/academicCycleSlice';
import progressReducer from '../features/progress/progressSlice';
import matriculaReducer from '../features/matricula/matriculaSlice';
import submissionReducer from '../features/submissions/submissionSlice';
import gradingReducer from '../features/grading/gradingSlice';
import analyticsReducer from '../features/analytics/analyticsSlice';

export const store = configureStore({
  reducer: {
    auth: authReducer,
    courses: courseReducer,
    enrollments: enrollmentReducer,
    sections: sectionReducer,
    content: contentReducer,
    assignments: assignmentReducer,
    learning: learningReducer,
    questions: questionReducer,
    evaluation: evaluationReducer,
    careers: careerReducer,
    users: usersReducer,
    academicCycles: academicCycleReducer,
    progress: progressReducer,
    matricula: matriculaReducer,
    submissions: submissionReducer,
    grading: gradingReducer,
    analytics: analyticsReducer,
  },
});
```

Librerías y Dependencias Clave

La arquitectura del sistema se apoya en un conjunto de librerías clave que definen su funcionalidad.

Dependencias del Backend (adapta-test-backend)

- **express:** Framework principal de Node.js para construir el servidor y las rutas de la API.

- **mongoose**: Actúa como la capa de acceso a datos (DAO/ORM) para la base de datos MongoDB.
- **bcryptjs**: Utilizada para encriptar (hashear) las contraseñas de los usuarios antes de guardarlas.
- **jsonwebtoken**: Crea y verifica los JSON Web Tokens (JWT) que gestionan la sesión del usuario y protegen las rutas.
- **multer**: Middleware para manejar la subida de archivos, utilizado para cargar el sílabus del curso.
- **nodemon**: (Desarrollo) Reinicia automáticamente el servidor al detectar cambios en el código.



```

package.json

{
  "name": "adapta-test-backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node src/server.js",
    "dev": "nodemon src/server.js",
    "test": "echo \"Error: no test specified\" && exit 1",
    "data:import": "node seeder.js",
    "data:destroy": "node seeder.js -d"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "bcryptjs": "^3.0.2",
    "dotenv": "^17.2.1",
    "express": "^5.1.0",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.17.1",
    "multer": "^2.0.2"
  },
  "devDependencies": {
    "nodemon": "^3.1.10"
  }
}

```

Dependencias del Frontend (adapta-test-frontend)

- **react**: Librería fundamental para construir la interfaz de usuario basada en componentes.
- **react-router-dom**: Proporciona el enrutamiento del lado del cliente para la navegación entre páginas.
- **@reduxjs/toolkit y react-redux**: Sistema de gestión de estado global que centraliza la información del usuario, cursos, etc..
- **axios**: Cliente HTTP para realizar peticiones a la API del backend desde los "servicios" del frontend.
- **tailwindcss**: Framework de utilidades CSS que define el diseño y la apariencia de toda la aplicación.
- **@radix-ui/***: Librería de componentes de UI accesibles que sirven como base funcional para los elementos visuales (Select, Dialog, etc.).
- **framer-motion**: Librería de animación para crear transiciones fluidas y efectos visuales atractivos.
- **lucide-react**: Proporciona el conjunto de iconos SVG utilizados en la aplicación.
- **sonner**: Librería para mostrar notificaciones (*toasts*) al usuario (ej. "Inicio de sesión exitoso").
- **vite**: (Desarrollo) Empaquetador y servidor de desarrollo del frontend que proporciona recarga en caliente (HMR).

```
package.json

{
  "name": "adapta-test-frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@radix-ui/react-dialog": "^1.1.15",
    "@radix-ui/react-dropdown-menu": "^2.1.16",
    "@radix-ui/react-label": "^2.1.7",
    "@radix-ui/react-select": "^2.2.6",
    "@radix-ui/react-slot": "^1.2.3",
    "@reduxjs/toolkit": "^2.8.2",
    "@tailwindcss/vite": "^4.1.14",
    "axios": "^1.11.0",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "framer-motion": "^12.23.24",
    "lucide-react": "^0.545.0",
    "react": "^19.1.1",
    "react-dom": "^19.1.1",
    "react-redux": "^9.2.0",
    "react-router-dom": "^7.8.1",
    "sonner": "^2.0.7",
    "tailwind-merge": "^3.3.1",
    "tailwindcss-animate": "^1.0.7"
  },
  "devDependencies": {
    "@eslint/js": "^9.33.0",
    "@types/node": "^24.7.2",
    "@types/react": "^19.1.10",
    "@types/react-dom": "^19.1.7",
    "@vitejs/plugin-react": "^5.0.0",
    "autoprefixer": "^10.4.21",
    "eslint": "^9.33.0",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.20",
    "globals": "^16.3.0",
    "postcss": "^8.5.6",
    "tailwindcss": "^3.4.18",
    "vite": "^7.1.2"
  }
}
```

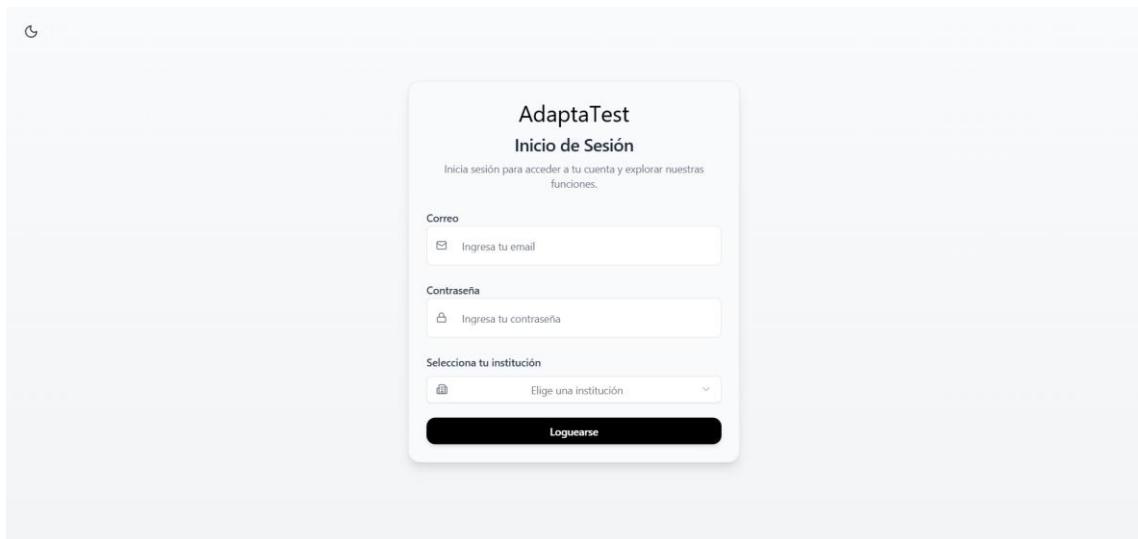
Capturas de las Interfaces

Las siguientes interfaces de usuario clave demuestran el flujo funcional de la aplicación. Las imágenes de referencia de estas interfaces se encuentran en la sección "Prototipado" de este documento.

Página de Login (Inicio de Sesión)

Descripción: Interfaz de inicio de sesión que carga dinámicamente la lista de instituciones disponibles desde la API. Utiliza componentes Card, Input y Select para una apariencia moderna y sonner para notificaciones.

[Ubicación]: adapta-test-frontend/src/pages/LoginPage.jsx



Dashboards (Paneles de Control)

Descripción: La aplicación presenta un panel de control (DashboardPage) que se adapta al rol del usuario (Estudiante, Profesor, Administrador).

[Ubicación]:

adapta-test-frontend/src/pages/DashboardPage.jsx (Lógica de enrutamiento)

adapta-test-frontend/src/features/dashboard/StudentDashboard.jsx (Vista Estudiante)

adapta-test-frontend/src/features/dashboard/ProfessorDashboard.jsx (Vista Profesor)

adapta-test-frontend/src/features/dashboard/AdminDashboard.jsx (Vista Admin)

The screenshot shows the 'AdaptaTest' administration dashboard. At the top, there's a header with the text 'AdaptaTest | Universidad Tecnológica Demo' on the left and 'Hola, Admin UTD (admin) Dashboard Cerrar Sesión' on the right. Below the header is a navigation bar with tabs: 'Dashboard de Administración', 'Gestionar Usuarios' (which is selected and highlighted in grey), 'Gestionar Carreras', 'Gestionar Cursos', and 'Gestión Académica'. The main content area is titled 'Usuarios de la Institución' and lists several users with their email and role: 'Admin UTD (admin@utd.com) - Rol: admin', 'Aldair Doloriert (aldair@utd.com) - Rol: student', 'Alumno UTD (alumno@utd.com) - Rol: student', 'Coordinador 2 UTD (coordinator2@utd.com) - Rol: coordinator', 'Coordinador 3 UTD (coordinator3@utd.com) - Rol: coordinator', 'Coordinador UTD (coordinator@utd.com) - Rol: coordinator', 'Profesor UTD 2 (profesor2@utd.com) - Rol: professor', and 'Profesor UTD (profesor@utd.com) - Rol: professor'. Each user entry is enclosed in a light blue box.

Página de Aprendizaje (Estudiante)

Descripción: Muestra los módulos, lecciones y tareas de una sección. Incluye un modal (AssignmentModal) para que los estudiantes puedan enviar sus tareas (createSubmission) y revisar sus calificaciones y feedback (getMySubmission).

[Ubicación]: adapta-test-frontend/src/pages/LearningPage.jsx

The screenshot shows the 'AdaptaTest' learning page for a student. At the top, there's a header with the text 'AdaptaTest | Universidad Tecnológica Demo' on the left and 'Hola, Alumno UTD (student) Dashboard Cerrar Sesión' on the right. Below the header is a section titled 'Cálculo I' with the sub-section 'Sección C1-001, impartida por Profesor UTD.' and 'Contenido del Curso'. This section contains two modules: 'MODULO 1 [+] Evaluuar mis conocimientos' and 'Modulo 2 [+] Evaluuar mis conocimientos'. To the right of these modules is a sidebar titled 'Tareas' which is currently empty. The overall layout is clean with a white background and light blue borders around the content boxes.

Gestión de Sección (Profesor)

Descripción: Panel con pestañas para que el profesor gestione su curso. Incluye creación de contenido (createAndPublishModuleToSection), creación de tareas (createAssignment), calificación (gradeSubmission) y visualización de analíticas (getSectionAnalytics).

[Ubicación]: adapta-test-frontend/src/pages/SectionManagementPage.jsx

Gestionando: Cálculo I

Sección: C1-001

Módulos y Contenido[Tareas](#) [Criterios de Aprobación](#) [Analíticas](#)

Módulos del Curso

MODULO 1 [+]

Modulo 2 [+]

Añadir Nuevo Módulo

Título del nuevo módulo

Crear y Publicar Módulo

Página de Evaluación Adaptativa (Estudiante)

Descripción: Interfaz donde el estudiante realiza la evaluación. Inicia la sesión (startEvaluation) y envía respuestas una por una (submitAnswer). El backend determina la siguiente pregunta basándose en el rendimiento.

[Ubicación]: adapta-test-frontend/src/pages/EvaluationPage.jsx

Evaluación en Progreso

¿Como se llama lo que da wifi?

Respuesta Correcta

Opción B

¡Evaluación Completada!

Resultado Final:

Respuestas Correctas: 0

Respuestas Incorrectas: 1

Nivel de Maestría Alcanzado: 20.00%

Volver al Curso