Diseño de Arquitectura del Sistema

**Fecha:** 04/09/2025

**Autor:**

Danluis Enrique Romani Mora

Eduardo Chaparro Huaman

Fabrizcio Galileo Argandoña Montalvo

**Versión:** 1.0

# 1. Introducción

Este documento describe de manera detallada la arquitectura del sistema para la aplicación web que está diseñada para la Identificación de Inteligencias múltiples en los estudiantes y el apoyo pedagógico a los docentes. Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas.

# 2. Visión General de la Arquitectura

## 2.1. Estilo Arquitectónico

El sistema adopta una arquitectura en capas, basada en el patrón cliente-servidor, y se organiza en tres capas principales:

* Capa de Presentación (Frontend): Interfaz con la que interactúa el usuario.
* Capa de Lógica de Negocio (Backend): Contiene la lógica que gobierna las operaciones del sistema.
* Capa de Persistencia de Datos (Base de Datos): Encargada del almacenamiento estructurado y recuperación de información.

### Épica 1: Inicio de sesión

* **Presentación:**
  + Formulario de login (docente, alumno, admin).
  + Botón “Olvidé mi contraseña”.
  + Mensajes de error / confirmación.
* **Negocio:**
  + Servicio de Autenticación (validación de credenciales con hashing, encriptación).
  + Servicio de Roles y Permisos (docente/alumno/admin).
  + Servicio de Recuperación de Contraseña (token, envío email/validación seguridad).
  + Servicio de Sesiones (cierre manual, por inactividad, confirmación).
* **Datos:**
  + Tabla Usuarios (credenciales, rol, estado).
  + Tabla LogsAcceso (fecha/hora accesos admin).
  + Tabla Sesiones (control de sesión activa/inactiva).

### Épica 2: Test de inteligencias múltiples

* **Presentación:**
  + Interfaz del test (preguntas/respuestas).
  + Visualización de resultados y gráficos.
* **Negocio:**
  + Servicio de Test (gestión de preguntas, validación de respuestas).
  + Servicio de Cálculo de Resultados (puntajes, identificación de inteligencias).
  + Servicio de Guardado de Avances.
  + Servicio de Generación de Reportes y Gráficos.
* **Datos:**
  + Tabla PreguntasTest.
  + Tabla RespuestasAlumno.
  + Tabla ResultadosTest.
  + Tabla HistorialResultados.

### Épica 3: Chatbot con IA

* **Presentación:**
  + Caja de texto + historial de conversación.
  + Botón “Reintentar consulta fallida”.
  + Mensajes de error.
* **Negocio:**
  + Servicio de Chatbot (integración con API DeepSeek).
  + Servicio de Procesamiento en Tiempo Real.
  + Servicio de Personalización (usa perfil, resultados de test, historial).
  + Servicio de Manejo de Errores y Reintentos.
  + Servicio de Logs y Auditoría.
* **Datos:**
  + Tabla HistorialConversaciones.
  + Tabla LogsErroresAPI.
  + Tabla ConsultasDocenteAuditoria.

### Épica 4: Dashboard alumno

* **Presentación:**
  + Dashboard (resultados, gráficos dinámicos, indicadores, progreso académico).
  + Interactividad en gráficos.
* **Negocio:**
  + Servicio de Resultados (cargar inteligencias múltiples).
  + Servicio de Histórico (consultar pruebas pasadas).
  + Servicio de Progreso Académico.
* **Datos:**
  + Tabla ResultadosTest.
  + Tabla HistorialResultados.
  + Tabla IndicadoresProgreso.

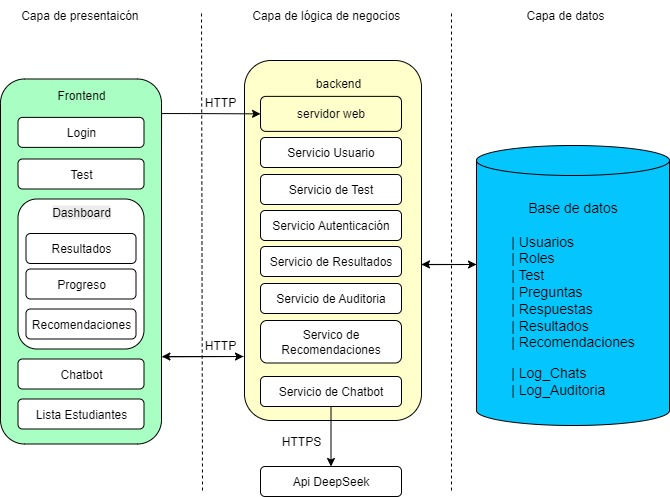
### Épica 5: Gestión de alumnos (CRUD)

* **Presentación:**
  + Listado de alumnos (docente/admin).
  + Formulario de registro/edición.
  + Confirmación de eliminación.
* **Negocio:**
  + Servicio de Gestión de Alumnos (CRUD).
  + Servicio de Búsqueda (por nombre, por código).
  + Servicio de Auditoría de Eliminaciones.
* **Datos:**
  + Tabla Alumnos (con código único).
  + Tabla HistorialAcciones (auditoría CRUD).

### Épica 6: Recomendaciones pedagógicas

* **Presentación:**
  + Vista de recomendaciones en panel del alumno.
  + Vista de recomendaciones en panel del docente.
* **Negocio:**
  + Servicio de Generación de Recomendaciones (basado en resultados).
  + Servicio de Actualización Automática (tras cada prueba).
  + Servicio de Consultas Históricas de Recomendaciones.
* **Datos:**
  + Tabla RecomendacionesAlumno.
  + Tabla HistorialRecomendaciones.

## 2.2. Diagrama General de Arquitectura:



# 3. Descripción de Componentes Principales

## 3.1. Frontend (Presentación)

* Framework: Angular
* Estilo:CSS
* Responsabilidades:
  + Renderizar interfaz gráfica amigable
  + Capturar entradas del usuario
  + Comunicarse con el backend vía api rest
  + Mostrar mensajes, errores y validaciones

## 3.2. Backend (Lógica de negocio)

* Lenguaje y Framework:
  + Java 17 con Spring Boot
  + Spring Data JPA para acceso a base de datos
  + Spring Security para autenticación y autorización
  + MySQL como motor de base de datos relacional
  + Maven/Gradle como gestor de dependencias
* Responsabilidades:

Exponer API RESTful

* Controladores (@RestController) para exponer los servicios.
* Endpoints organizados por recursos (e.g., /usuarios, /tareas, /chatbot, /reportes).

Gestión de la lógica de negocio

* + Servicios (@Service) que aplican reglas de negocio:
  + Validación de fechas de tareas (vencidas, próximas a vencer).
  + Reglas para asignación de tareas a alumnos.
  + Generación de recomendaciones pedagógicas en base a la inteligencia predominante.
  + Control de reportes individuales y grupales.

**Persistencia y manejo de datos**

* Repositorios (@Repository) con **Spring Data JPA** para acceso a MySQL.
* ORM con **Hibernate**.
* Relaciones: Usuarios ↔ Tareas ↔ Resultados de Test.

**Integración con Chatbot (DeepSeek API)**

* Cliente HTTP (con **Spring WebClient o RestTemplate**) para enviar prompts y recibir respuestas.
* Adaptador en el servicio de negocio para formatear respuestas del chatbot.
* Gestión de roles (ej: docente puede consultar progreso, alumno recibe orientación personalizada).

**Seguridad**

* **JWT (JSON Web Tokens)** para autenticación y autorización.
* Roles: ALUMNO, DOCENTE, ADMIN.
* Encriptación de contraseñas con **BCrypt**.

**Gestión de errores y validaciones**

* Manejo centralizado de excepciones con @ControllerAdvice.
* Validaciones con javax.validation (Bean Validation).
* Mensajes de error claros y consistentes.

### 3.2.1. Módulos del Backend

* authController.js: Manejo de login y registro

## 3.3. Base de Datos (Persistencia)

* Sistema Gestor: Mysql
* ORM: Sequelize o Prisma
* Esquema de Datos:
  + Tabla Alumnos
  + Tabla Inteligencias
  + Tabla Cursos
  + Tabla Competencias
  + Tabla Alumno Competencias

# 4. Integraciones Externas (Opcionales)

(No se requiere)

# 5. Seguridad

* Encriptación de contraseñas hash sal pimienta.
* Uso de HTTPS.

# 6. Escalabilidad y Despliegue

Posibles problemas que pueden surgir durante el desarrollo.

* **Frontend**: Desplegado dentro del .jar
* **Backend**: Desplegado como .jar
* **Base de Dato**s: MySQL en PHPmyAdmin

# 7. Conclusiones

La arquitectura que se está proponiendo tiene como objetivo el tener en consideración cada una de las funcionalidades que se definieron en el planteamiento del proyecto, considerando desde docentes, alumnos, notas, tipos de inteligencia hasta los reportes generados por la aplicación.

| **Nro.** | **Fecha** | **Autor(es)** | **Descripción** |
| --- | --- | --- | --- |
|  | 09/09/2025 | Danluis Enrique Romani Mora  Eduardo Chaparro Huaman  Fabrizcio Galileo Argandoña Montalvo | Se creó el documento y se sentaron las bases del proyecto que se va a desarrollar. |