This document provides an analysis of the current system and generates a similar documentation structure based on the provided example.

**📘 Documentación del Sistema Educativo - v1**

**Fecha de emisión:** 2024-06-06

1. Introducción y Fuente de Verdad

Este documento establece la arquitectura oficial, tareas asignables, estándares y diseños del Sistema Educativo.

Su propósito es alinear el trabajo entre múltiples desarrolladores humanos y asistencias de IA, bajo un único conjunto de reglas.

⚠️ **Nota oficial:** Esta documentación es la única fuente oficial para decisiones de diseño, codificación y arquitectura.

Cualquier tarea que se realice fuera de este estándar debe ser previamente autorizada por el responsable del sistema.

2. Contenido del Documento

1. Introducción y Fuente de Verdad
2. Arquitectura del Sistema
3. Modelos de Datos (Inferidos)
4. Endpoints API REST (Inferidos)
5. Detalle de Tareas por Módulo (Inferidas)
6. Diseño de Interfaces (Wireframes - Adaptados)
7. Contratos de Datos y Convenciones de Respuesta (Inferidos)
8. Flujo Integrado de Usuario (Inferido)
9. Matriz de Conversación entre Tareas (Inferida)
10. Flujos Adicionales de Usuario (Inferidos)
11. Conjuntos de Datos Mock para Desarrollo Paralelo (Sugerido)

3. Arquitectura del Sistema

El sistema parece estar basado en una arquitectura cliente-servidor clásica con separación clara entre frontend (React.js - inferido de la estructura de carpetas) y backend (Node.js con Express). La estructura de carpetas en el backend sugiere una organización por módulos o funcionalidades más que un estricto patrón MVC, aunque elementos de controlador, rutas y posibles modelos existen.

La comunicación se realiza mediante una API RESTful. La presencia de un middleware de autenticación (backend/middlewares/auth.js) y rutas de autenticación (backend/routes/auth/index.js) sugiere que la seguridad se basa en autenticación, probablemente con tokens (JWT es común en Node.js).

La base de datos utilizada no se especifica explícitamente en server.js, pero la presencia de backend/db.sqlite3 y archivos como backend/config/db.js y backend/models/... indica que se utiliza una base de datos, y backend/config/db.js probablemente contiene la configuración de conexión. La mención de backend/db.sqlite3 sugiere SQLite, al menos para desarrollo o pruebas iniciales.

**3.1 Estructura de carpetas - Backend**

backend/

├── config/ # Configuraciones (DB, posiblemente autenticación)

├── controllers/ # Lógica de negocio y manejo de solicitudes

├── middlewares/ # Funciones de validación y autenticación

├── models/ # Posibles definiciones de estructuras de datos o interacciones con la DB

├── routes/ # Archivos de rutas por módulo

├── uploads/ # Directorio para archivos subidos (justificativos, observaciones)

├── server.js # Punto de entrada del backend

└── ... otros archivos ...

**text (auto)**

**3.2 Estructura de carpetas - Frontend**

frontend/

├── public/ # Archivos estáticos

├── src/

│ ├── assets/ # Posiblemente recursos como CSS, imágenes

│ ├── components/ # Componentes reutilizables

│ ├── images/ # Imágenes específicas

│ ├── pages/ # Vistas/páginas principales

│ ├── services/ # Lógica para interactuar con la API

│ ├── styles/ # Archivos de estilos

│ └── App.js # Enrutamiento principal y lógica raíz

└── ... otros archivos ...

**text (auto)**

4. Modelos de Datos (Inferidos)

Basado en los nombres de archivos en las carpetas backend/models/, backend/controllers/, backend/routes/, se infieren los siguientes modelos de datos principales. La estructura y los campos específicos deberían ser confirmados leyendo los archivos de modelo y controlador correspondientes.

* **Person:** Probablemente representa a personas en el sistema (estudiantes, profesores, etc.). (Visto en backend/models/persona/Person.js, backend/controllers/persona/personController.js, frontend/src/components/PersonForm.js)
* **Course:** Representa cursos o asignaturas. (Visto en backend/models/curso/Course.js, backend/controllers/curso/CursoController.js, frontend/src/components/curso/CursoFormulario.js)
* **Assessment:** Representa evaluaciones o notas. (Visto en backend/models/notas/Assessment.js, backend/controllers/notas/notasController.js, frontend/src/components/AssessmentForm.js)
* **Colegio:** Representa la información del colegio. (Visto en backend/controllers/ColegioController.js, backend/routes/ColegioRoutes.js, frontend/src/pages/DatosColegio.js)
* **User:** Probablemente para autenticación y gestión de usuarios del sistema. (Visto en backend/controllers/usuarios/usuarioController.js, backend/routes/usuarios/usuarioRoutes.js)
* **Role:** Probablemente para gestionar roles de usuario y permisos. (Visto en frontend/src/services/RoleService.js, backend/middlewares/requiereRol.js)
* **Asignatura:** Representa asignaturas. (Visto en backend/controllers/asignatura/AsignaturaController.js, backend/routes/asignatura/AsignaturaRoutes.js, frontend/src/components/AsignarAsignaturas.js)
* **Jerarquia:** Posiblemente relacionado con la estructura organizacional o académica. (Visto en backend/controllers/jerarquiaController.js, backend/routes/jerarquiaRoutes.js)
* **Matricula:** Representa el proceso de matrícula de estudiantes. (Visto en backend/controllers/matriculaController.js, backend/routes/matriculaRoutes.js, frontend/src/components/FormularioMatricula.js)
* **Reference:** Parece ser utilizado para listas desplegables o datos de referencia. (Visto en backend/controllers/referenceController.js, backend/routes/referenceRoutes.js, frontend/src/components/PersonForm.js)
* **Bloques:** Posiblemente relacionado con bloques horarios o académicos. (Visto en backend/controllers/bloques/BloquesController.js, backend/routes/bloques/BloquesRoutes.js, frontend/src/components/bloques/Bloques.js)
* **Horarios:** Representa horarios de clases o actividades. (Visto en backend/controllers/horarios/HorariosController.js, backend/routes/horarios/horarioRoutes.js, frontend/src/components/horarios/GestionHorariosIntegrado.js)
* **CalendarioHorarios:** Posiblemente la integración de calendarios y horarios. (Visto en backend/controllers/calendariohorarios/calendariohorariosController.js, backend/routes/calendariohorarios/calendariohorariosRoutes.js, frontend/src/components/calendario/CalendarioHorarios.js)
* **Electivo:** Representa asignaturas o cursos electivos. (Visto en backend/controllers/electivo/ElectivoController.js, backend/routes/electivo/electivoRoutes.js, frontend/src/components/electivo/Electivo.js)

*Nota: La relación entre estos modelos (tablas en la base de datos) debe ser confirmada leyendo el código de backend, especialmente los archivos de modelos y controladores.*

5. Endpoints API REST (Inferidos)

Basado en el uso de rutas en backend/server.js y los nombres de archivos en backend/routes/ y backend/controllers/, se infieren los siguientes endpoints API. Los métodos HTTP específicos (GET, POST, PUT, DELETE) y la estructura exacta de las URLs deben confirmarse leyendo los archivos de rutas.

* /api/matricula: Rutas relacionadas con el proceso de matrícula.
* /api/persons: Rutas relacionadas con la gestión de personas (estudiantes, etc.).
* /api/profesores: Rutas relacionadas con la gestión de profesores.
* /api/colegios: Rutas relacionadas con la información del colegio.
* /api/cursos: Rutas relacionadas con la gestión de cursos.
* /api/electivos: Rutas relacionadas con la gestión de electivos.
* /api/references: Rutas para obtener datos de referencia/listas desplegables.
* /api/crear (POST): Posiblemente para crear un colegio inicialmente (basado en app.post('/api/crear', colegioController.createColegio);).
* /api/asignaturas: Rutas relacionadas con la gestión de asignaturas.
* /api/bloques: Rutas relacionadas con la gestión de bloques.
* /api/horarios: Rutas relacionadas con la gestión de horarios.
* /api/auth: Rutas relacionadas con la autenticación de usuarios (login, etc.).
* /api/usuarios: Rutas relacionadas con la gestión de usuarios del sistema.
* /api/calendariohorarios: Rutas relacionadas con calendarios y horarios.
* /api/notas: Rutas relacionadas con la gestión de notas y evaluaciones.
* /uploads: Para servir archivos subidos.

6. Detalle de Tareas por Módulo (Inferidas)

Basado en los nombres de archivos y la estructura del proyecto, se pueden inferir tareas lógicas agrupadas por funcionalidad. Se requiere un análisis más profundo del código para definir tareas específicas con inputs, outputs, dependencias y estados precisos como en el ejemplo. Sin embargo, aquí se presenta una estructura inferida:

**Módulo: Autenticación y Usuarios**

* Implementación de registro de usuarios.
* Implementación de inicio de sesión (login) con generación de tokens (probablemente JWT).
* Implementación de middleware de autenticación para proteger rutas.
* Gestión de usuarios (CRUD).
* Gestión de roles y permisos.
* Funcionalidad para cambiar clave.

**Módulo: Gestión de Colegio**

* Creación y gestión de información del colegio.

**Módulo: Gestión de Personas (Estudiantes, Profesores, etc.)**

* Registro y gestión de datos personales.

**Módulo: Gestión Académica (Cursos, Asignaturas, Electivos)**

* Creación y gestión de cursos.
* Creación y gestión de asignaturas.
* Asignación de asignaturas.
* Gestión de electivos.

**Módulo: Matrícula**

* Registro y gestión del proceso de matrícula.

**Módulo: Horarios y Calendario**

* Gestión de bloques horarios.
* Gestión de horarios específicos.
* Integración de calendario y horarios.

**Módulo: Notas y Evaluaciones**

* Registro y gestión de notas y evaluaciones.

**Módulo: Datos de Referencia**

* Gestión y provisión de datos para listas desplegables.

**Módulo: Archivos Subidos**

* Gestión de subida y acceso a archivos (justificativos, observaciones).

7. Diseño de Interfaces (Wireframes - Adaptados)

Adaptando los wireframes del ejemplo al contexto de un sistema educativo, se podrían tener pantallas similares para las siguientes funcionalidades:

**WF1 – Pantalla de Login**

+-----------------------------------+

| INICIAR SESIÓN |

+-----------------------------------+

| Usuario/Email: [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]|

| Clave: [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]|

| |

| [ Ingresar ] ¿Olvidó su clave?|

+-----------------------------------+

**text (auto)**

**WF2 – Panel Principal (Adaptado al Menú Existente)**

+--------------------+-------------------------------+

| Menú | Área de trabajo |

|--------------------|-------------------------------|

| • Matrícula | [Vista seleccionada] |

| • Personas | |

| • Profesores | |

| • Colegio | |

| • Cursos | |

| • Electivos | |

| • Asignaturas | |

| • Bloques | |

| • Horarios | |

| • Calendario | |

| • Usuarios | |

| • Notas | |

+--------------------+-------------------------------+

**text (auto)**

**WF3 – Gestión de Cursos**

+--------------------------------------------------------+

| Cursos |

+--------------------------------------------------------+

| [ + Nuevo Curso ] [ Buscar: \_\_\_\_\_\_\_\_ ] [🔍] |

+--------------------------------------------------------+

| Nombre del Curso | Nivel | Jornada | Acciones |

|------------------|-------|---------|------------------|

| 1° Básico A | Básico| Mañana | [✏️][🗑️] |

| 8° Básico B | Básico| Tarde | [✏️][🗑️] |

+--------------------------------------------------------+

**text (auto)**

**WF4 – Gestión de Notas**

+-------------------------------------------------------------+

| Registro de Notas |

+-------------------------------------------------------------+

| Curso: [ Seleccionar Curso ▼ ] |

| Asignatura: [ Seleccionar Asignatura ▼ ] |

| Evaluación: [ Nombre/Tipo Evaluación ] |

| Fecha Evaluación: [\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_] |

| |

| Estudiante | Nota |

|-------------------|-------|

| Juan Pérez | [\_\_\_\_\_]|

| Ana Gómez | [\_\_\_\_\_]|

+-------------------------------------------------------------+

| [ Guardar Notas ] |

+-------------------------------------------------------------+

**text (auto)**

8. Contratos de Datos y Convenciones de Respuesta (Inferidos)

Basado en la estructura de las rutas y la probable interacción cliente-servidor, se infiere que el sistema intercambia datos en formato JSON. Las convenciones de respuesta HTTP probablemente siguen estándares RESTful (200 OK, 201 Created, 400 Bad Request, etc.), aunque esto debe ser confirmado por la implementación del backend.

Ejemplos de formatos de datos (a ser confirmados por el código):

**Formato de Usuario (User)**

{

"userId": 12,

"username": "jperez",

"email": "juan.perez@colegio.cl",

"role": {

"roleId": 2,

"name": "profesor"

}

}

**json**

**Formato de Curso (Course)**

{

"courseId": 5,

"name": "8 Básico A",

"level": "Básico",

"jornada": "Mañana"

}

**json**

**Formato de Nota (Assessment)**

{

"assessmentId": 101,

"studentId": 45,

"asignaturaId": 8,

"courseId": 5,

"evaluationName": "Prueba Unidad 1",

"score": 6.5,

"date": "YYYY-MM-DDTHH:mm:ssZ"

}

**json**

9. Flujo Integrado de Usuario (Inferido)

Ejemplo de un flujo posible:

1. Usuario (Profesor) accede a POST /api/auth/login con credenciales válidas → recibe token.
2. Token se usa para autenticar llamadas posteriores.
3. Profesor accede a la sección de notas (ruta /api/notas).
4. Selecciona un curso y una asignatura.
5. Sistema carga la lista de estudiantes matriculados en ese curso/asignatura.
6. Profesor ingresa las notas para cada estudiante.
7. Envía los datos con POST /api/notas (o similar) para guardar las notas.

10. Matriz de Conversación entre Tareas (Inferida)

| **Tarea Origen (Módulo)** | **Provee a Tarea (Módulo)** | **Formato Compartido (Inferido)** | **Comentario** |
| --- | --- | --- | --- |
| Autenticación/Usuarios | Todos los módulos protegidos | Token JWT | Permite acceso a funcionalidades |
| Gestión de Personas (Estudiante) | Matrícula, Notas, Horarios | { personId, name, ... } | Datos de estudiantes y profesores |
| Gestión de Cursos | Matrícula, Horarios, Notas | { courseId, name, ... } | Permite asociar estudiantes y notas a cursos |
| Gestión de Asignaturas | Horarios, Notas, Profesores | { asignaturaId, name, ... } | Permite asociar asignaturas a cursos/prof. |
| Matrícula | Personas, Cursos | Relación estudiante-curso | Define la población para cursos/asignaturas |
| Horarios | Cursos, Asignaturas, Profesores | { horarioId, day, time, ... } | Define la estructura de clases |
| Notas | Estudiantes, Asignaturas | { assessmentId, score, ... } | Registra el rendimiento académico |
| Datos de Referencia | Formularios varios | Listas de opciones ({ id, name }) | Pobla selectores y campos con datos maestros |

11. Conjuntos de Datos Mock para Desarrollo Paralelo (Sugerido)

Se recomienda crear una carpeta /mock/ en la raíz del proyecto (o dentro de frontend/mock/) para almacenar datos de ejemplo que permitan a los desarrolladores trabajar en el frontend y backend de forma independiente.

Ejemplos de archivos mock:

* /mock/mockUser.json: Datos de un usuario con roles.
* /mock/mockStudent.json: Datos de un estudiante.
* /mock/mockProfessor.json: Datos de un profesor.
* /mock/mockCourse.json: Datos de un curso.
* /mock/mockAsignatura.json: Datos de una asignatura.
* /mock/mockMatricula.json: Datos de una matrícula.
* /mock/mockNotas.json: Datos de notas para un estudiante/asignatura.
* /mock/mockHorario.json: Datos de un horario.
* /mock/mockReferenceData.json: Datos para listas desplegables (ej: lista de comunas, tipos de documento).

*Nota: La estructura y los campos específicos de estos mocks deben coincidir con los formatos de datos definidos (o inferidos) en la sección 8.*