Arquitectura Anilisis

1. Carpeta raíz (backend)

Contiene los componentes centrales del sistema:

Elemento	Rol
server.js	Punto de entrada del backend: configura el servidor Express, los middlewares y las rutas.
package.json	Gestión de dependencias y comandos de ejecución (start, dev).

2. database

Representa la configuración de conexión con MongoDB:

Archivo	Función
.env	Contiene MONGO_URI y JWT_SECRET para conectar y firmar tokens.
mongoose.connect(...)	Se ejecuta desde server.js, estableciendo la conexión con la base de datos.

Relación:

- server.js →database: Establece la conexión mediante mongoose.

3. models (MongoDB con Mongoose)

Define las **colecciones** de MongoDB:

Modelo	Propósito
User.js	Representa a los usuarios (admin, student, parent, etc.). Incluye roles y opcionalmente hijos.
Grade.js	Representa calificaciones asociadas a usuarios tipo estudiante (studentId). Incluye descomposición y promedio.

Relación:

- controllers → models: Consultan y modifican los datos.
- models →database: Persisten en MongoDB.

4. classes (Lógica OO de calificaciones)

Implementa la arquitectura orientada a objetos para manejar notas individuales y agrupadas:

Clase	Función
NotaComponent.js	Clase abstracta para definir el contrato getValor().
NotaIndividual.js	Implementación simple, devuelve el valor directo.
GrupoNotas.js	Permite agrupar múltiples notas y calcular el promedio global.

Relación:

- controllers →classes: Utilizan esta lógica para estructurar las calificaciones.
- classes →models: Pueden trabajar con datos obtenidos del modelo Grade.

5. controllers (Lógica de negocio)

Archivo	Función principal
authController.js	Maneja login, registro y autenticación por roles.
gradesController.js	Crea y consulta calificaciones según el rol del usuario.
reportController.js	Genera un PDF de todas las calificaciones.

Relación:

- routes → controllers: Ejecutan los controladores desde cada endpoint.
- controllers →models: Persistencia y lectura.
- Controllers → classes: Lógica de notas compuestas.
- controllers → utils: Seguridad y validación.

---

6. routes (Enrutamiento Express)

Archivo	Finalidad
authRoutes.js	Login, registro y acceso como invitado.
gradesRoutes.js	CRUD de calificaciones.
reportRoutes.js	Endpoint para descargar el PDF generado.

Relación:

- server.js → routes: Los monta como middlewares.
- routes → controllers: Conectan lógica de negocio con Express.

7. utils

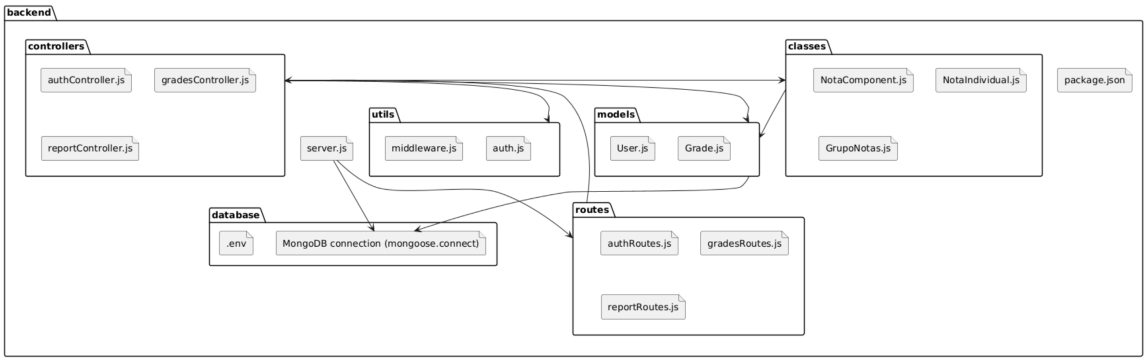
Contiene funciones auxiliares (middleware de autenticación, verificación de roles, etc.).
Aunque no especificaste el contenido exacto, pueden incluir:

- Protección de rutas (JWT, roles).
- Funciones compartidas para verificar permisos, validar input, etc.

Relaciones clave en el sistema

Fuente	Destino	Descripción
server.js	database	Configura conexión con MongoDB.
server.js	routes	Habilita endpoints en Express.
routes	controllers	Llama a funciones que procesan las peticiones.
controllers	models	Interactúan con la base de datos.
controllers	classes	Aplican lógica de notas en estructuras complejas.
models	database	Persisten los esquemas en MongoDB.

DIAGRAMA ARQUITECTURA:



TLFBJiGm3BpdAw9UWCFs4z2AWX07E2pY0xoDjuHDxIf9Yn3YtqdQi4eVz3XZpxZsPEyyMD-tHWXtrjI1XLQUs2Yqh_xBeBHOUw36e10JB8zGdv6K_1P2fbAfmCCH70PGXk-aGsTH8Ws512SVQCZWPwQ6x-zapKH1N3F9wpPYxB1CuCtODrPMbxAWyPDvjgpGk9dZcqDRVhWYWnujA4nOtbqRS0xdxI_ieUAsOqBo2vt8FP7IWrOzc9L9tt6iS5jEJDwoCMZdRj3xz-

h2BHIREBRxbxROiVKhEhUrt7jSEnv6UDDjavgSPbJkqdkzk46hbJBu2HONlj5z fZ2YuMB3PVu9waB0e8djJ8K8jLVoW0Rg487mG-4q5iDlqvdGYKn3xdQtgzWi-Map8Yqe8jc9H7wko4qnMy4fFojyN8X8sToQvYzucRROX_SKt-0I