ARQUITECTURA DEL SOFTWARE

Sistema: Plataforma de Gestión de Calificaciones
Proyecto: Unidad Educativa Mahanaym
Versión Analizada: 1.1
Autores: Isaac Escobar, Eduardo Mortensen, Diego Ponce

1. Arquitectura General

El sistema sigue una arquitectura **MVC (Modelo – Vista – Controlador)**, típica de aplicaciones web modernas.
Se compone de:

- **Frontend Web:**
  Interfaz gráfica en HTML/CSS/JavaScript que permite la interacción de los usuarios (administradores, docentes, estudiantes y padres).
- **Backend API (Node.js + Express):**
  Servidor que gestiona la lógica del sistema a través de controladores, modelos y middleware.
- **Base de Datos:**
  Almacena la información académica como usuarios, calificaciones, y registros de auditoría.

2. Estructura del Proyecto (MVC)

Componente	Función
Modelos (M)	Representan entidades como `Usuario`, `Nota`, `Estudiante`. Interactúan con la base de datos.
Controladores (C)	Procesan las solicitudes del frontend, ejecutan la lógica de negocio, y devuelven respuestas. Ej: `authController`, `gradesController`.
Vistas (V)	El frontend que ve el usuario, desarrollado con tecnologías web (no incluido como código aquí, pero asumido desde la arquitectura).
Middleware	Módulos que validan tokens, roles, y aseguran la seguridad antes de ejecutar controladores.

3. Patrón de Diseño Aplicado: Composite

El sistema implementa explícitamente el patrón **Composite**, útil para representar **jerarquías de notas**.

- `NotaComponent.js` (abstracto)
- `NotaIndividual.js` (nota simple)
- `GrupoNotas.js` (colección de notas)

Esto permite tratar una nota simple y un conjunto de notas de la misma forma, mejorando la flexibilidad del sistema al calcular promedios o generar informes.

4. Fortalezas Identificadas

- Uso correcto de la **separación de responsabilidades (MVC)**.
- Aplicación de un **patrón de diseño estructural (Composite)**.
- Estructura modular y escalable.
- Soporte para **roles y autenticación** (middleware incluido).
- Preparado para aplicar los lineamientos del Ministerio de Educación.

5. script startUML

```
@startuml

' Configuración de estilo visual

skinparam style strictuml

skinparam packageStyle rectangle

skinparam backgroundColor #FDFDFD

skinparam shadowing false


title Arquitectura del Sistema de Gestión de Calificaciones\n(MVC + Patrón Composite)


' === ACTOR PRINCIPAL ===

actor "Usuario (Admin / Docente / Estudiante / Padre)" as User

' El usuario interactúa con el sistema desde el navegador


' === FRONTEND WEB ===

package "Frontend Web\n(HTML/CSS/JS)" as FE {

  [Interfaz Web\nFormulario Login, Panel de Notas,\nConfiguración de Porcentajes]
```

```
  ' Representa la interfaz del navegador: login, paneles, formularios
}

' === BACKEND COMPLETO ===
package "Backend API\n(Node.js + Express)" as BE {

  ' --- CAPA DE CONTROLADORES (C) ---
  package "Rutas y Controladores" {
    [authController.js] as AuthCtrl         ' Maneja el login y registro
    [gradesController.js] as GradesCtrl     ' Lógica para registrar y consultar notas
    [reportController.js] as ReportCtrl     ' Generación de informes académicos
    [student_controller.js] as StudentCtrl  ' Administración de estudiantes
  }

  ' --- CAPA MIDDLEWARE (Autenticación / Seguridad) ---
  package "Middleware" {
    [authMiddleware.js] as AuthMW           ' Verifica rol, permisos
    [verifyToken.js] as TokenMW             ' Verifica token JWT
  }

  ' --- CAPA DE MODELOS (M) ---
  package "Modelos (M)" {
    [User.js]           ' Modelo de usuario (login, roles)
    [Grade.js]          ' Modelo de notas/calificaciones
    [userModel.js]      ' Modelo extendido u opcional de usuarios
  }

  ' --- ARCHIVOS DE CONFIGURACIÓN ---
  package "Configuración" {
    [db.js]                  ' Configura conexión a base de datos
    [ministerioPorcentajes.js] ' Define reglas y porcentajes oficiales del Ministerio
```

```
  }

  ' --- PATRÓN COMPOSITE ---
  package "Patrón Composite\n(Composite.js)" {
    [NotaComponent.js] <<abstract>>    ' Componente base abstracto
    [NotaIndividual.js]                ' Representa una sola nota
    [GrupoNotas.js]                    ' Agrupa varias notas en conjunto
  }
}

' === BASE DE DATOS ===
database "Base de Datos\nMongoDB / SQL" as DB
' Base de datos relacional o NoSQL según implementación

' === FLUJOS DE COMUNICACIÓN ===

' Usuario interactúa con el frontend
User --> FE : Interacción vía navegador

' El frontend envía peticiones al backend por rutas
FE --> AuthCtrl : Solicitud Login
FE --> GradesCtrl : Registrar/Consultar notas
FE --> ReportCtrl : Ver reportes
FE --> StudentCtrl : Datos de estudiantes

' Middleware se aplica antes de ejecutar controladores protegidos
AuthCtrl --> AuthMW
GradesCtrl --> TokenMW
ReportCtrl --> AuthMW

' Los controladores acceden a modelos para operaciones en la DB
```

AuthCtrl --> [User.js]

GradesCtrl --> [Grade.js]

StudentCtrl --> [userModel.js]

' Los modelos consultan o actualizan la base de datos

[User.js] --> DB

[Grade.js] --> DB

[userModel.js] --> DB

' El controlador de calificaciones utiliza el patrón Composite

GradesCtrl --> [GrupoNotas.js]

GradesCtrl --> [NotaIndividual.js]

[GrupoNotas.js] --> [NotaComponent.js]

[NotaIndividual.js] --> [NotaComponent.js]

' Esto permite tratar notas individuales y grupos como objetos equivalentes

@enduml

6. Diagrama de la Arquitectura