Prompt para Backend OpenBlind

Tecnologías Requeridas

- **Node.js** con **Express.js**
- **PostgreSQL** para datos críticos
- **MongoDB** para datos secundarios
- **Arquitectura MVC**
- **Sistema de logging en un solo archivo**
- **Encriptación** para datos sensibles
- **Sistema de roles simple**

Estructura de Base de Datos Híbrida

PostgreSQL (Datos Críticos)

Tabla: users

- id (PRIMARY KEY, UUID)

- email (UNIQUE, ENCRYPTED)

- password_hash (ENCRYPTED)

- role (admin/user - ENUM)

- telefono (ENCRYPTED)

- nombres (ENCRYPTED)

- apellidos (ENCRYPTED)

- fecha_nacimiento (ENCRYPTED)

- created_at

- updated_at

- is_active

Tabla: routes

- id (PRIMARY KEY, UUID)

- name (NOT NULL)

- location (NOT NULL)

- transport_name (NOT NULL)

- status (active/inactive)

- created_by (FK to users)

- created_at

- updated_at

Tabla: personalized_messages

- id (PRIMARY KEY, UUID)

- message (NOT NULL, ENCRYPTED)

- status (active/inactive)

- route_id (FK to routes)

- created_by (FK to users)

- created_at

- updated_at

Tabla: tourist_registrations

- id (PRIMARY KEY, UUID)

- destination_place (NOT NULL)

- name (NOT NULL)

- description (TEXT)

- latitude (DECIMAL)

- longitude (DECIMAL)

- status (active/inactive)

- created_by (FK to users)

- created_at

- updated_at

MongoDB (Datos Secundarios)

Collection: user_profiles

```
{
  _id: ObjectId,
  user_id: String, // Referencia a PostgreSQL
```

profile_image: String,

preferences: Object,

last_login: Date,

login_history: Array,

created_at: Date,

updated_at: Date

}

Collection: voice_guides

{

_id: ObjectId,

route_id: String, // Referencia a PostgreSQL

audio_file_url: String,

created_at: Date,

updated_at: Date

}

Collection: system_logs

{

_id: ObjectId,

timestamp: Date,

level: String, // info, warn, error

method: String,

url: String,

status_code: Number,

user_id: String,

ip_address: String,

message: String,

error_details: Object

}

Estructura de Carpetas Requerida

```
backend/
├── src/
│   ├── config/
│   │   ├── database.orm.js     // Configuración MongoDB
│   │   ├── database.sql.js     // Configuración PostgreSQL
│   │   ├── encryption.js       // Configuración de encriptación
│   │   └── config.js           // Configuraciones generales
│   ├── models/
│   │   ├── sql/                // Modelos PostgreSQL
│   │   │   ├── User.js
│   │   │   ├── Route.js
│   │   │   ├── PersonalizedMessage.js
│   │   │   └── TouristRegistration.js
│   │   └── nosql/              // Modelos MongoDB
│   │       ├── UserProfile.js
│   │       ├── VoiceGuide.js
│   │       └── SystemLog.js
│   ├── controllers/
│   │   ├── authController.js
│   │   ├── userController.js
│   │   ├── routeController.js
│   │   ├── messageController.js
│   │   ├── touristController.js
│   │   └── voiceGuideController.js
│   ├── middleware/
│   │   ├── auth.js
```

```
│   │   ├── roleCheck.js       // Middleware simple de roles
│   │   ├── validation.js
│   │   ├── encryption.js
│   │   ├── logging.js
│   │   └── errorHandler.js
│   ├── routes/
│   │   ├── auth.js
│   │   ├── users.js
│   │   ├── routes.js
│   │   ├── messages.js
│   │   ├── tourist.js
│   │   └── voiceGuides.js
│   ├── services/
│   │   ├── authService.js
│   │   ├── userService.js
│   │   ├── routeService.js
│   │   ├── encryptionService.js
│   │   └── logService.js
│   ├── utils/
│   │   ├── constants.js
│   │   ├── helpers.js
│   │   └── validators.js
│   └── app.js
├── logs/
│   └── application.log       // Único archivo de logs
├── package.json
├── .env
```

├── .gitignore

└── server.js

Funcionalidades Específicas Requeridas

Sistema de Autenticación Simple

- JWT para tokens
- Bcrypt para hash de contraseñas
- Middleware de autenticación
- **Dos roles simples**: admin y user

Middleware de Roles Simple

```
// Middleware que verifica si el usuario es admin
const requireAdmin = (req, res, next) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ message: 'Access denied' });
  }
  next();
};


// Middleware que verifica si el usuario está autenticado
const requireAuth = (req, res, next) => {
  // Verificar JWT token
};
```

Encriptación de Datos Sensibles

- AES-256 para encriptar campos sensibles
- Variables de entorno para claves de encriptación
- Funciones de encriptar/desencriptar en helpers

Sistema de Logging Simple

- Winston para logging
- Todas las peticiones y respuestas en un solo archivo: application.log

- Logs también guardados en MongoDB para consultas

Endpoints Principales Necesarios

// Autenticación

POST /api/auth/login

POST /api/auth/register

GET /api/auth/profile

// Usuarios (solo admin puede ver/editar otros usuarios)

GET /api/users // Solo admin

POST /api/users // Solo admin

GET /api/users/:id // Solo admin o propio usuario

PUT /api/users/:id // Solo admin o propio usuario

DELETE /api/users/:id // Solo admin

// Rutas

GET /api/routes // Cualquier usuario autenticado

POST /api/routes // Solo admin

PUT /api/routes/:id // Solo admin

DELETE /api/routes/:id // Solo admin

// Mensajes Personalizados

GET /api/messages // Cualquier usuario autenticado

POST /api/messages // Solo admin

PUT /api/messages/:id // Solo admin

DELETE /api/messages/:id // Solo admin

// Registro Turístico

GET /api/tourist-registrations // Cualquier usuario autenticado

POST /api/tourist-registrations // Solo admin

PUT /api/tourist-registrations/:id // Solo admin

DELETE /api/tourist-registrations/:id // Solo admin

// Guías de Voz

GET /api/voice-guides // Cualquier usuario autenticado

POST /api/voice-guides // Solo admin

DELETE /api/voice-guides/:id // Solo admin

Validaciones Básicas

- Validación de email único
- Validación de campos requeridos
- Sanitización de datos de entrada
- Validación de coordenadas geográficas

Variables de Entorno (.env)

Database

POSTGRES_HOST=localhost

POSTGRES_PORT=5432

POSTGRES_DB=openblind_db

POSTGRES_USER=postgres

POSTGRES_PASSWORD=12345

MONGODB_URI=mongodb://localhost:27017/openblind_nosql

Security

JWT_SECRET=your-jwt-secret

ENCRYPTION_KEY=your-encryption-key

BCRYPT_ROUNDS=12

Server

PORT=3000

NODE_ENV=development

Logging

LOG_LEVEL=info

Consideraciones Simples

1. **Dos roles únicamente**: admin (puede todo) y user (solo lectura)
2. **Logging centralizado**: Todo en un archivo + MongoDB
3. **Encriptación simple**: Solo para datos sensibles de usuarios
4. **API RESTful básica**: CRUD estándar para cada recurso
5. **Validaciones esenciales**: Solo lo necesario para seguridad básica
6. **CORS habilitado**: Para el frontend Angular

Archivos de Inicialización

- Script de migración simple para PostgreSQL
- Script de inicialización para MongoDB
- Datos de prueba básicos (un admin y un usuario normal)

Por favor, genera toda la estructura del backend siguiendo estas especificaciones simplificadas, incluyendo todos los archivos mencionados con código funcional pero sin complejidad innecesaria.