Grupo #5

MANUAL TÉCNICO - SISTEMA HRM (Backend)

Este manual documenta la arquitectura, configuración, modelo de datos y especificación de la API del sistema HRM, desarrollado con Node.js y TypeScript.

1. ARQUITECTURA DEL SISTEMA

Stack Tecnológico

Capa	Tecnología	Versión	Propósito
Frontend	React + TypeScript	18.x	Interfaz de usuario dinámica (SPA)
Backend	Node.js + Express + TypeScript	20.x	API RESTful y lógica de negocio
Base de Datos	MySQL + Prisma ORM	8.x	Persistencia de datos relacional
Autenticación	JWT + bcrypt	-	Seguridad y gestión de sesiones sin estado
Validación	Zod	3.x	Validación de esquemas en tiempo de ejecución
Documentación	OpenAPI 3.0	-	Especificación formal y tipado de la API

Diagrama de Arquitectura de Alto Nivel

El sistema sigue un patrón modular en el backend y una arquitectura de cliente-servidor para la comunicación:

```
graph TD
    A[Cliente Web/Móvil] -->|Peticiones HTTP| B(React SPA con TypeScript);
    B -->|Llamadas API REST| C(Express API con TypeScript);
```

```
C -->|Consultas ORM| D[MySQL Database];

subgraph Backend
    C
    E[Prisma ORM & Migrations]
    F[Middleware JWT & Seguridad]
    C --> E
    C --> F
end



subgraph Frontend
    B
    G[Validación Zod]
    B --> G
end

D
```

2. CONFIGURACIÓN TÉCNICA

Estructura de Proyecto (Backend)

El proyecto utiliza una arquitectura basada en módulos (dominios de negocio):

```
hrm-system/
├── src/
│   ├── modules/        # Lógica de negocio por dominios (auth, employees, etc.)
│   ├── core/           # Componentes transversales (config, middlewares)
│   ├── types/          # Tipos TypeScript (incluyendo los generados de OpenAPI)
│   └── server.ts       # Punto de entrada de la aplicación
```

Grupo #5

```
├── prisma/          # Esquema de la DB, migraciones y seeds
├── scripts/         # Scripts de utilidad (seed, test, validate-openapi)
└── docs/            # Archivo de especificación OpenAPI (openapi.yaml)
```

Variables de Entorno (.env)

Las siguientes variables son obligatorias para el arranque del servicio.

Variable	Propósito	Ejemplo de Valor
DATABASE_URL	Cadena de conexión para Prisma/MySQL.	mysql://user:pass@localhost:3306/hrm_db
JWT_SECRET	Clave secreta para firmar el Token de Acceso.	tu_jwt_secret_super_seguro
JWT_REFRESH_SECRET	Clave secreta para firmar el Token de Refresco.	tu_refresh_secret
JWT_EXPIRES_IN	Tiempo de expiración del Token de Acceso (segundos).	900 (15 minutos)

JWT_REFRESH_EXPIRES_IN	Tiempo de expiración del Token de Refresco (segundos).	604800 (7 días)
NODE_ENV	Entorno de ejecución (development, production).	development
PORT	Puerto donde se ejecuta el servidor Express.	4000
UPLOAD_MAX_FILE_SIZE	Tamaño máximo de archivos permitidos (bytes).	10485760 (10 MB)

3. MODELO DE DATOS (Prisma Schema)

El modelo de datos relacional se gestiona completamente a través de Prisma.

Entidades Principales

```
// Entidades principales
model User { ... } // @unique en email y employeeId
model Employee {
  // Relacionado con Department y User. Contiene datos personales y laborales.
  ...
  department Department @relation(...)
  user       User?
  ...
```

```
}
model Department { ... } // Contiene la lista de departamentos de la empresa.
model Document {
  // Almacena metadatos de archivos subidos. La clave real del archivo (storageKey)
  // debe usarse para acceder al sistema de almacenamiento (ej. S3/local storage).
  ...
  employee Employee @relation(...)
  uploader User     @relation(...)
}
model Payroll { ... } // Cabecera de la nómina, relacionada con un periodo y departamento.
model PayrollItem { ... } // Detalle de nómina por empleado.
```

Enumeraciones Clave

Enum	Valores	Propósito
UserRole	ADMIN, RRHH, EMPLEADO	Define los niveles de acceso.
EmployeeStatus	ACTIVE, INACTIVE, SUSPENDED, VACATION	Estado laboral del empleado.
PayrollStatus	DRAFT, FINALIZED, PAID	Estado del procesamiento de la nómina.

4. SISTEMA DE AUTENTICACIÓN Y AUTORIZACIÓN

Flujo de Seguridad

1. **Login:** El cliente envía credenciales a POST /auth/login. El servidor responde con un accessToken (corto) y un refreshToken (largo).
2. **Acceso a Recursos:** Todas las rutas protegidas requieren el accessToken en el encabezado: Authorization: Bearer <accessToken>.
3. **Middleware verifyJWT:** Decodifica el token, autentica al usuario y adjunta el objeto User a la solicitud (req.user).
4. **Middleware authorize:** Tras la autenticación, verifica que el req.user.role esté dentro de la lista de roles permitidos para ese *endpoint*.

Roles y Permisos

Rol	Alcance y Permisos
ADMIN	Superusuario. Acceso total (lectura/escritura) a todos los módulos y capacidad para gestionar otros usuarios.
RRHH	Acceso a gestión de Empleados, Documentos de terceros y procesamiento de Nóminas. No puede gestionar usuarios de tipo ADMIN.
EMPLEADO	Acceso restringido. Solo puede leer su propia información de perfil, ver y descargar sus Documentos y consultar sus Nóminas personales.

5. ESPECIFICACIÓN DE LA API (OpenAPI)

La especificación completa está en *docs/openapi.yaml.*

Formato de Respuesta Estándar (JSON)

Tipo	Estructura	Códigos HTTP Comunes
Éxito	{"success": true, "data": {...}, "message": "..."}	200 (OK), 201 (Creado), 204 (Sin contenido)
Error	{"success": false, "error": {"code": "...", "message": "...", "details": {...}}}	400, 401, 403, 404, 422, 500

Grupo #5

Endpoints Principales por Módulo

Módulo	Método	Endpoint	Descripción	Roles Requeridos
Auth	POST	/auth/login	Iniciar sesión y emitir tokens.	Público
Auth	GET	/auth/me	Obtener perfil del usuario autenticado.	Todos
Empleados	GET	/employees	Listar todos los empleados con filtros.	ADMIN, RRHH
Empleados	POST	/employees	Crear nuevo empleado (y usuario asociado).	ADMIN, RRHH
Empleados	GET	/employees/:id	Obtener detalles de un empleado.	Todos (sujeto a su propio registro)
Documentos	POST	/employees/:id/documents	Subir un documento (multipart/form-data).	ADMIN, RRHH

Documentos	**GET**	**/employees/:id/documents/:docId/download**	**Descargar el documento.**	**ADMIN, RRHH, EMPLEADO***
Nóminas	**POST**	**/payroll**	**Crear una nueva nómina (borrador).**	**ADMIN, RRHH**
Nóminas	**POST**	**/payroll/:id/finalize**	**Calcular y marcar una nómina como finalizada.**	**ADMIN, RRHH**
Usuarios	**PUT**	**/users/:id**	**Actualizar rol o estado de un usuario.**	**ADMIN**

6. CONFIGURACIÓN Y DESPLIEGUE

Proceso de Instalación (Desarrollo)

 # 1. Clonar repositorio

git clone <repository-url>

cd hrm-backend

 # 2. Instalar dependencias

npm install

Grupo #5

 # 3. Configurar variables de entorno (Crear y editar .env)

cp .env.example .env

 # 4. Configurar base de datos (Generar el cliente Prisma y aplicar el esquema)

npx prisma generate

npx prisma db push

 # 5. Poblar datos iniciales (Crea usuarios base, ej. ADMIN)

npm run db:seed

 # 6. Ejecutar en desarrollo

npm run dev

Scripts Disponibles (NPM)

Script	Descripción	Comando Real
dev	Inicia el servidor en modo *watch* (desarrollo).	tsx watch src/server.ts
start	Inicia la aplicación compilada (producción).	node dist/server.js
build	Compila el código TypeScript a JavaScript (carpeta dist).	tsc
db:push	Sincroniza el esquema de Prisma con la DB (desarrollo rápido).	npx prisma db push
db:seed	Ejecuta el script de siembra de datos.	tsx scripts/seed.ts
db:reset	Resetea la DB, aplica push y ejecuta seeds.	npm run db:push && npm run db:seed
test:api	Ejecuta las pruebas de integración.	tsx scripts/test-api.ts

7. CALIDAD Y MANTENIMIENTO

Medidas de Calidad y Seguridad

- **Tipado Estricto**: Uso de TypeScript en modo estricto para type-safety.
- **Validación Estricta**: Uso de Zod para validar *payloads* de entrada en todos los *controllers*.
- **Seguridad Web**: Uso de helmet para proteger encabezados HTTP y bcrypt para hashing de contraseñas (salt 12).
- **Manejo de Errores**: Respuestas HTTP semánticas (401, 403, 422, 409) y formato estándar de error.

Consideraciones de Mantenimiento

- **Métricas**: Endpoint de salud (/health) para monitoreo de estado.
- **Actualizaciones**: Recomendación de mantener las dependencias y el motor de Node.js actualizados.
- **Prisma**: Utilizar el flujo de migraciones de Prisma para cambios en el esquema de la base de datos de manera controlada.