UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

FACULTAD DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN Y CIENCIAS DE LA COMPUTACIÓN.

Manual Técnico

Versión: 2.0

HRM System

GUATEMALA, OCTUBRE 2025

Grupo #5

INDICE

INTRODUCCIÓN

El presente documento constituye el **Manual Técnico del Sistema HRM (Human Resource Management System)**, una aplicación web desarrollada con el objetivo de ofrecer una solución integral para la **gestión del personal, las nóminas, documentos, reportes y usuarios de una empresa**.

Este manual tiene como propósito **documentar la arquitectura, la configuración técnica, la estructura de módulos, el modelo de datos, la seguridad, el despliegue y las prácticas de mantenimiento** del sistema.

El Sistema HRM está compuesto por dos partes principales:

- **Frontend:** Una SPA (Single Page Application) desarrollada con **React 18, TypeScript y Vite**, desplegada en **Vercel**.
- **Backend:** Una API RESTful desarrollada con **Node.js, Express y TypeScript**, con persistencia de datos en **MySQL** gestionada por **Prisma ORM**, y desplegada en **Render**.

La base de datos se aloja en **Railway**, lo que garantiza conectividad estable, acceso remoto seguro y soporte para entornos de producción.

El sistema implementa una arquitectura **cliente-servidor moderna**, con **autenticación basada en JWT**, comunicación bajo protocolo **HTTPS** y separación lógica entre presentación, negocio y datos.

MANUAL TÉCNICO – SISTEMA HRM

1. Objetivo Del Manual

Este documento tiene los siguientes objetivos principales:

1. **Describir** la arquitectura técnica, herramientas, componentes y flujos del backend del Sistema HRM.
2. **Guiar** en la instalación, configuración y despliegue del sistema en entornos locales o cloud.
3. **Documentar** los modelos de datos, la API RESTful y la seguridad implementada.
4. **Establecer** procedimientos de mantenimiento, buenas prácticas y lineamientos de calidad del software.
5. **Servir** como base de conocimiento para nuevos desarrolladores o administradores del sistema.

2. Descripción General Del Sistema

El **Sistema HRM** permite administrar de forma centralizada los recursos humanos de una organización, facilitando la creación, gestión y control de empleados, usuarios, nóminas, documentos y reportes.

2.1 Características Principales

- Interfaz moderna y adaptable (SPA).
- Gestión de empleados, usuarios, documentos y nóminas.
- Autenticación mediante tokens JWT (JSON Web Tokens).
- Validación de datos con **Zod** y tipado estricto con **TypeScript**.
- Arquitectura modular en backend y frontend.
- Despliegue 100% en la nube (Render, Vercel y Railway).
- Base de datos relacional normalizada en **MySQL**.

2.2 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

3 Configuración Y Despliegue

Proceso de Instalación (Desarrollo)

Clonar repositorios

1. git clone https://github.com/Javiii3er/hrm-backend.git
   cd hrm-backend

   git clone https://github.com/Javiii3er/hrm-frontend.git
   cd hrm-frontend

2. **Instalar dependencias**
   npm install (hasta aquí termina la configuración para el frontent)

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

   NODE_ENV=development
   PORT=4000
   DATABASE_URL="mysql://usuario:usuario@localhost:3306/hrm_db"

JWT_SECRET="supersecreto123_usa_una_mas_larga_en_produccion"
JWT_REFRESH_SECRET="mi_clave_refresh_token_muy_segura_y_bastante_larga_2025"

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

3.1 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

4 Roles de Usuario

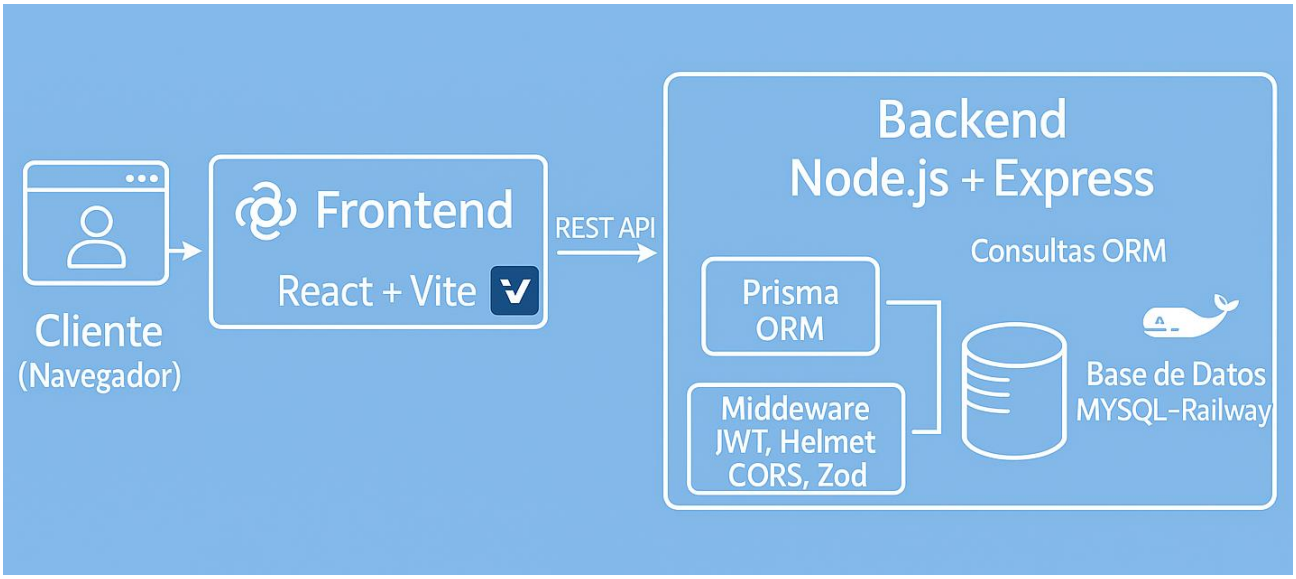
ROL	DESCRIPCIÓN
ADMIN	Control total del sistema. Puede crear, editar y eliminar usuarios, empleados, departamentos y nóminas.
RRHH	Gestiona empleados, documentos y nóminas, sin acceso a la administración general.
EMPLEADO	Acceso limitado a su información personal, documentos y nóminas propias.

5 Módulos Principales

- **Autenticación:** Login, refresh tokens y control de acceso.
- **Empleados:** CRUD completo con relación a usuarios y departamentos.
- **Departamentos:** Organización jerárquica por áreas laborales.
- **Usuarios:** Administración de roles y credenciales.
- **Documentos:** Carga, descarga y almacenamiento de archivos asociados.
- **Nóminas:** Cálculo, registro y control de pagos.
- **Reportes:** Exportación en PDF/CSV.

6 Arquitectura General Del Sistema

El Sistema HRM se basa en una arquitectura **cliente-servidor** y **multicapa**, dividiendo la lógica en presentación (frontend), negocio (backend) y persistencia (base de datos).

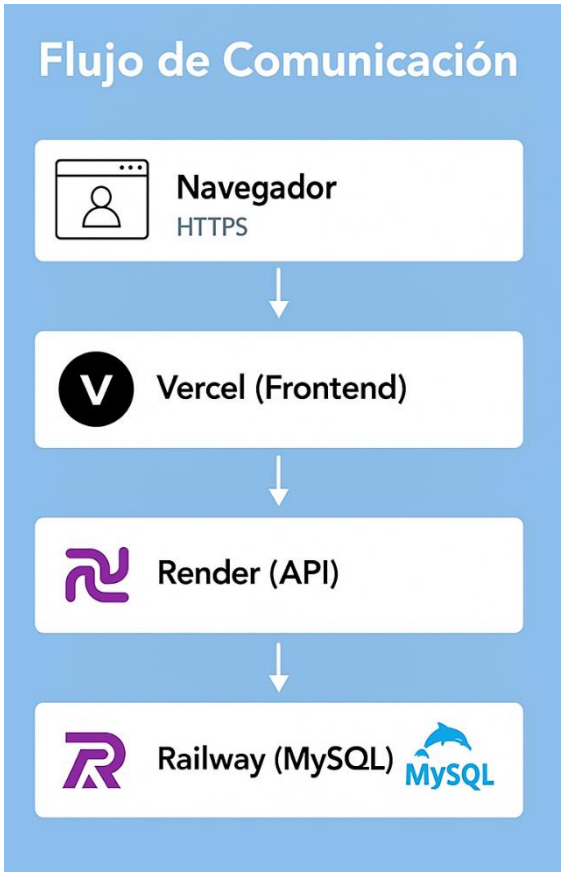


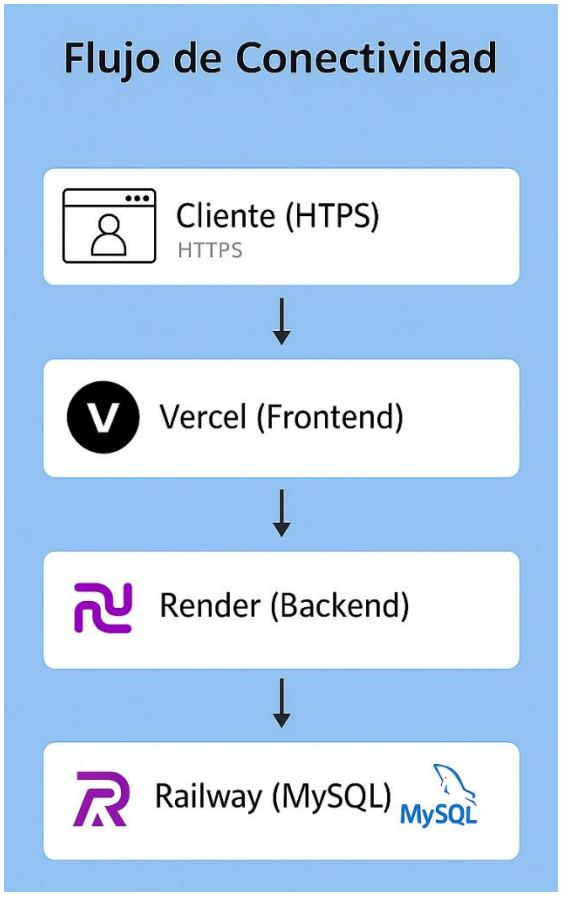
7 Componentes Cloud

Capa	Servicio	Tecnología	Descripción
Frontend	Vercel	React + Vite	Interfaz SPA servida por CDN global.
Backend	Render	Node.js + Express	API RESTful con endpoints de negocio.
Base de Datos	Railway	MySQL 8	Base relacional conectada por Prisma ORM.

8 Flujo de Comunicación y Conectividad

La comunicación entre Vercel y Render se realiza mediante HTTPS, y Render conecta a Railway vía TCP 3306, utilizando variables seguras de entorno.





- El cliente (SPA) en Vercel realiza peticiones HTTPS al backend alojado en Render.
- Render procesa la solicitud, accede a Railway mediante conexión TCP 3306 usando DATABASE_URL.
- Todas las comunicaciones se realizan bajo protocolo HTTPS, asegurando confidencialidad e integridad.

9 Configuración Técnica Del Backend y Frontend

9.1 Estructura del Proyecto (Backend)

```
hrm-backend/
├── prisma/              # Esquema y migraciones de base de datos
├── scripts/             # Scripts de seed, tests y mantenimiento
├── src/
│   ├── core/
│   │   ├── config/      # Configuración global y entorno
│   │   ├── middleware/  # Autenticación, logging, error handler
│   │   ├── utils/       # Funciones auxiliares y helpers
│   ├── modules/
│   │   ├── auth/        # Login, refresh y logout
│   │   ├── employees/   # Gestión de empleados
│   │   ├── departments/ # Gestión de departamentos
│   │   ├── users/       # Administración de usuarios
│   │   ├── documents/   # Archivos y carga de documentos
│   │   ├── payroll/     # Nóminas y pagos
│   │   ├── reports/     # Generación de reportes
│   │   └── profile/     # Perfil del usuario autenticado
│   ├── app.ts           # Configuración de Express
│   └── server.ts        # Punto de entrada principal
├── .env                 # Variables de entorno (API local)
├── .env.production      # Base de datos Railway (pública)
└── docs/
    └── openapi.yaml     # Documentación de la API
```

9.2 Estructura del Proyecto (Frontend)

```
hrm-frontend/
├── public/                    # Archivos estáticos públicos (favicon, manifest, imágenes)
│   └── index.html             # Plantilla HTML base
│
├── src/
│   ├── core/                  # Núcleo funcional del sistema
│   │   ├── api/
│   │   │   └── client.ts      # Cliente Axios centralizado con interceptores JWT
│   │   ├── contexts/          # Contextos globales (autenticación, toasts, etc.)
│   │   │   ├── AuthContext.tsx    # Contexto de autenticación del usuario
│   │   │   ├── ToastContext.tsx   # Contexto para notificaciones globales
│   │   │   └── useAuth.ts         # Hook personalizado para autenticación
│   │   ├── hooks/             # Hooks reutilizables para lógica global
│   │   │   ├── useFetch.ts        # Hook genérico de solicitudes asíncronas
│   │   │   ├── usePagination.ts   # Hook para manejo de paginación
│   │   │   └── useDebounce.ts     # Hook para optimizar búsquedas
│   │   ├── utils/             # Utilidades y funciones auxiliares
│   │   │   ├── formatters.ts      # Funciones para formatear fechas, monedas, etc.
│   │   │   ├── validators.ts      # Validaciones comunes
│   │   │   └── constants.ts       # Constantes globales (roles, estados, etc.)
│   │   └── types/             # Tipos globales y generados
│   │       ├── api.gen.ts         # Tipos generados desde OpenAPI
│   │       └── global.ts          # Tipos generales (ApiResponse, ApiError, etc.)
│
│   ├── components/            # Componentes reutilizables de UI
│   │   ├── Navbar.tsx         # Barra de navegación principal
```

```
│   │   ├── Sidebar.tsx          # Menú lateral (por rol)
│   │   ├── Button.tsx           # Botón genérico reutilizable
│   │   ├── Modal.tsx            # Componente modal reutilizable
│   │   ├── Toast.tsx            # Notificación visual
│   │   └── LoadingSpinner.tsx   # Indicador de carga
│
│   ├── layout/                  # Layouts generales de la aplicación
│   │   ├── AppLayout.tsx        # Layout principal (encabezado, sidebar, contenido)
│   │   ├── AuthLayout.tsx       # Layout de autenticación
│   │   └── ProtectedRoute.tsx   # Control de acceso a rutas privadas
│
│   ├── modules/                 # Módulos funcionales (por dominio)
│   │   ├── auth/                # Módulo de autenticación
│   │   │   ├── pages/LoginPage.tsx # Página de inicio de sesión
│   │   │   └── hooks/useLogin.ts   # Hook para manejar el login
│   │   ├── employees/           # Módulo de empleados
│   │   │   ├── components/EmployeeForm.tsx  # Formulario de creación/edición
│   │   │   ├── components/EmployeeList.tsx  # Listado principal
│   │   │   ├── hooks/useEmployees.ts        # Lógica de negocio con React Query
│   │   │   └── pages/EmployeesPage.tsx      # Página principal de empleados
│   │   ├── departments/         # Módulo de departamentos
│   │   │   ├── hooks/useDepartments.ts
│   │   │   └── pages/DepartmentsPage.tsx
│   │   ├── users/               # Módulo de usuarios
│   │   │   ├── components/UserForm.tsx
│   │   │   ├── hooks/useUsers.ts
```

```
│   │   │   └── pages/UsersPage.tsx
│   │   ├── documents/          # Módulo de documentos
│   │   │   ├── components/DocumentUploader.tsx
│   │   │   ├── hooks/useDocuments.ts
│   │   │   └── pages/DocumentsPage.tsx
│   │   ├── payroll/            # Módulo de nóminas
│   │   │   ├── components/PayrollTable.tsx
│   │   │   ├── hooks/usePayroll.ts
│   │   │   └── pages/PayrollPage.tsx
│   │   ├── reports/            # Módulo de reportes
│   │   │   ├── components/ReportGenerator.tsx
│   │   │   ├── hooks/useReports.ts
│   │   │   └── pages/ReportsPage.tsx
│   │   └── profile/            # Perfil del usuario autenticado
│   │       ├── hooks/useProfile.ts
│   │       └── pages/ProfilePage.tsx
│
│   ├── styles/                 # Archivos de estilos globales
│   │   ├── variables.css       # Variables de colores, fuentes, espaciado
│   │   ├── global.css          # Estilos generales del sistema
│   │   └── home.css            # Estilos de la página de inicio
│
│   ├── AppRoutes.tsx           # Definición de rutas principales (React Router 6)
│   ├── App.tsx                 # Componente raíz de la aplicación
│   └── main.tsx                # Punto de entrada del Frontend
├── .env                        # Variables de entorno (API local)
```

```
├── .env.production          # Variables para entorno productivo (Render/Vercel)
├── vite.config.ts           # Configuración de Vite (puerto, proxy, alias)
├── tsconfig.json            # Configuración de TypeScript
├── package.json             # Dependencias y scripts de npm
├── README.md                # Documentación del proyecto
└── index.html               # Plantilla HTML base
```

9.3 Estructura Lógica de Capas

CAPA	DIRECTORIO	FUNCIÓN PRINCIPAL
Presentación (UI)	/components, /layout, /modules/*/pages	Interfaz gráfica y navegación.
Lógica de Negocio	/modules/*/hooks, /core/hooks	Peticiones, validaciones y gestión de datos.
Gestión de Estado y Contextos	/core/contexts	Autenticación, notificaciones, sesión global.
Comunicación con la API	/core/api/client.ts	Cliente Axios central con interceptores JWT.
Configuración Global	/vite.config.ts, /tsconfig.json	Compilación, alias, y entorno de desarrollo.

9.4 Scripts Principales (package.json)

SCRIPT	DESCRIPCIÓN	COMANDO REAL
npm run dev	Ejecuta el servidor de desarrollo	vite
npm run build	Compila el proyecto para producción	vite build
npm run preview	Sirve el build localmente para pruebas	vite preview
npm run lint	Verifica la calidad del código	eslint .

9.5 Dependencias Clave

CATEGORÍA	LIBRERÍA	USO
Framework Base	React 18, React DOM	Construcción de la SPA
Lenguaje	TypeScript	Tipado estático
Enrutamiento	React Router 6	Navegación y rutas privadas
Estado Remoto	React Query	Manejo de peticiones y caché
Formularios	React Hook Form	Validación y control de formularios
Validación	Zod	Esquemas de validación
UI	Bootstrap 5, Bootstrap Icons	Diseño responsivo
HTTP Client	Axios	Comunicación con el backend
PDF / Exportación	jspdf, html2canvas	Generación de reportes y descargas
Alertas y Notificaciones	SweetAlert2, ToastContext	Feedback visual del sistema

Grupo #5

10 Variables de Entorno

Variable	Propósito	Ejemplo
DATABASE_URL	Conexión MySQL (Railway)	mysql://user:pass@host:3306/hrm_db
JWT_SECRET	Firma del token de acceso	clave_super_segura
JWT_REFRESH_SECRET	Firma del token de refresco	clave_refresco_segura
JWT_EXPIRES_IN	Tiempo de vida del token (s)	900
JWT_REFRESH_EXPIRES_IN	Tiempo de vida del refresh token	604800
NODE_ENV	Entorno (dev/prod)	production
PORT	Puerto del servidor	4000

ENTORNO	ARCHIVO	VARIABLE	DESCRIPCIÓN
Desarrollo	.env	VITE_API_URL=http://localhost:4000/api	Dirección de la API local
Producción	.env.production	VITE_API_BASE_URL=https: hrm-backend-o1nu.onrender.com/api	URL pública del backend (Render)

11 Modelo De Datos (Prisma Orm)

El modelo de datos del Sistema HRM se gestiona completamente a través de Prisma ORM, que define las entidades, relaciones y enumeraciones dentro de una base de datos relacional MySQL 8.

Cada modelo representa un dominio de negocio (usuario, empleado, nómina, documento, etc.) con sus claves primarias, relaciones y restricciones únicas.

11.1 Entidades Principales

ENTIDAD	DESCRIPCIÓN
User	Controla credenciales, roles y tokens. Posee campos únicos en email y employeeId.
Employee	Contiene los datos personales y laborales del empleado. Está relacionado con Department y con User.
Department	Representa las áreas o divisiones de la empresa. Cada empleado pertenece a un departamento.
Document	Registra los archivos subidos (contratos, constancias, etc.), incluyendo metadatos y la referencia al archivo real (storageKey) en el almacenamiento. Se relaciona con el empleado propietario y con el usuario que subió el documento.
Payroll	Cabecera de la nómina. Define un período y departamento. Contiene la información general de un proceso de pago.
PayrollItem	Detalle individual de la nómina, asociado a un empleado específico.
AuditLog	Registra acciones del sistema (creación, actualización, eliminación), asociadas a un usuario.

11.2 Relaciones Principales

Relación	Tipo	Descripción
User ↔ Employee	1:1	Un usuario puede estar vinculado a un empleado. (employeeId es único en User)
Employee ↔ Department	N:1	Varios empleados pueden pertenecer a un mismo departamento.
Employee ↔ Document	1:N	Un empleado puede tener varios documentos asociados.
Payroll ↔ PayrollItem	1:N	Cada nómina agrupa varios registros de pago (uno por empleado).
User ↔ AuditLog	1:N	Un usuario puede generar múltiples registros de auditoría (acciones realizadas).
Document ↔ User (uploader)	N:1	Cada documento conserva el usuario que lo subió.

11.3 Enumeraciones

ENUM	VALORES	PROPÓSITO
UserRole	ADMIN, RRHH, EMPLEADO	Define roles del sistema.
EmployeeStatus	ACTIVE, INACTIVE, SUSPENDED, VACATION	Estados laborales.
PayrollStatus	DRAFT, FINALIZED, PAID	Ciclos de nómina.

11.4 Representación en Prisma Schema (Estructura Técnica Simplificada)

```
model User {
  id           String     @id @default(uuid())
  email        String     @unique
  passwordHash String
  role         UserRole   @default(EMPLEADO)
  employeeId   String?    @unique
  employee     Employee?  @relation(fields: [employeeId], references: [id])
  auditLogs    AuditLog[]
  documents    Document[]
  createdAt    DateTime   @default(now())
  updatedAt    DateTime   @updatedAt
}

model Employee {
  id           String       @id @default(uuid())
  nationalId   String       @unique
  firstName    String
  lastName     String
  email        String
  phone        String?
  departmentId String
  department   Department   @relation(fields: [departmentId], references: [id])
  user         User?
  status       EmployeeStatus @default(ACTIVE)
  documents    Document[]
  createdAt    DateTime     @default(now())
  updatedAt    DateTime     @updatedAt
}

model Department {
  id        String     @id @default(uuid())
  name      String
  employees Employee[]
  payrolls  Payroll[]
}

model Document {
  id        String   @id @default(uuid())
  fileName  String
```

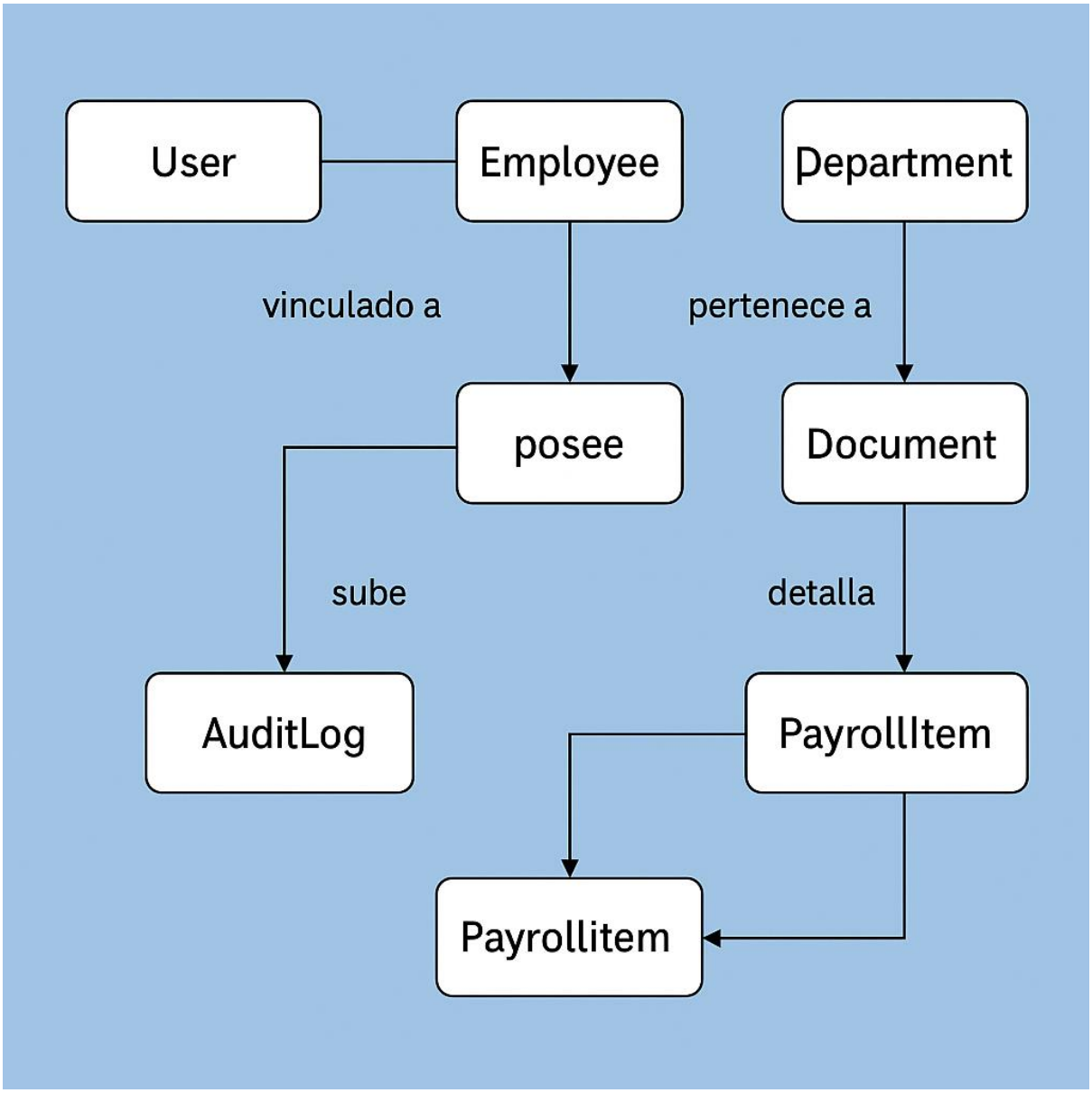
```
  storageKey  String
  mimeType    String
  employeeId  String
  uploaderId  String
  employee    Employee @relation(fields: [employeeId], references: [id])
  uploader    User     @relation(fields: [uploaderId], references: [id])
  createdAt   DateTime @default(now())
}

model Payroll {
  id          String       @id @default(uuid())
  periodStart  DateTime
  periodEnd    DateTime
  departmentId String
  department   Department   @relation(fields: [departmentId], references: [id])
  status       PayrollStatus @default(DRAFT)
  items        PayrollItem[]
  createdAt    DateTime     @default(now())
}

model PayrollItem {
  id        String    @id @default(uuid())
  payrollId  String
  employeeId String
  amount     Decimal   @db.Decimal(10,2)
  payroll    Payroll   @relation(fields: [payrollId], references: [id])
  employee   Employee  @relation(fields: [employeeId], references: [id])
}

model AuditLog {
  id       String   @id @default(uuid())
  userId    String
  action    String
  timestamp DateTime @default(now())
  user      User     @relation(fields: [userId], references: [id])
```

11.5 Diagrama Conceptual Simplificado



12 Autenticación Y Autorización

El sistema utiliza **JWT (JSON Web Tokens)** para autenticar usuarios y controlar accesos.

12.1 Flujo de Autenticación

1. El usuario envía sus credenciales a POST/auth/login.
2. El servidor valida la información y genera:
    - accessToken (vida corta)
    - refreshToken (vida larga)
3. El cliente guarda ambos tokens en localStorage.
4. Cada solicitud posterior incluye el token: Authorization: Bearer <accessToken>
5. Si expira, el cliente solicita /auth/refresh para renovarlo.

12.2 Middlewares de Seguridad

- **verifyJWT**: Verifica la validez del token y decodifica los datos del usuario.
- **authorize**: Valida permisos según el rol asignado.
- **helmet**: Añade cabeceras HTTP seguras.
- **cors**: Controla orígenes permitidos (localhost y dominio de Vercel).
- **compression**: Optimiza la respuesta HTTP.

13 ESPECIFICACIÓN DE LA API (OpenAPI)

La especificación completa está en docs/openapi.yaml.

13.1 Formato de Respuesta Estándar

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

13.2 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

14 Despliegue En La Nube

14.1 Componentes del Entorno Productivo

COMPONENTE	SERVICIO	URL
Frontend (UI)	Vercel	https://hrm-frontend-sigma-three.vercel.app
Backend (API)	Render	https://hrm-backend-o1nu.onrender.com
Base de Datos (MySQL)	Railway	Base conectada mediante DATABASE_URL

15 Seguridad, Calidad Y Mantenimiento

15.1 Medidas de Seguridad

- Autenticación JWT con renovación de token.
- Roles y permisos definidos a nivel de middleware.
- Encriptación de contraseñas con **bcrypt (salt=12)**.
- Validación de entradas con **Zod**.
- Restricción CORS para dominios de producción.
- Logs de auditoría en tabla AuditLog.
- Uso de helmet para headers HTTP seguros.

15.2 Prácticas de Calidad

- Tipado estricto con TypeScript (strict: true).
- Manejo centralizado de errores (errorHandler).
- Validación en controladores y servicios.

- Código modular y reutilizable.
- Versionado de esquema con **Prisma Migrations**.
- Documentación viva con **OpenAPI 3.0**.

15.3 Mantenimiento

- Monitoreo de estado con /health.
- Actualización periódica de dependencias (npm audit fix).
- Revisión semestral de roles y permisos.
- Backups automáticos de Railway.

16 Futuras Mejoras Y Escalabilidad

1. Migración completa a HTTPS con certificados SSL personalizados.
2. Integración de almacenamiento externo (AWS S3 / GCP Storage) para documentos.
3. Sistema de notificaciones por correo electrónico (Nodemailer / Queue).
4. Pruebas automatizadas (Jest / Supertest).
5. Implementación CI/CD con GitHub Actions.
6. Panel de monitoreo en tiempo real con logs centralizados.
7. Optimización de consultas Prisma mediante índices adicionales.

17 DESPLIEGUE EN LA NUBE

El **Sistema HRM** fue desplegado completamente en entornos cloud gratuitos, distribuyendo cada componente según su función:

COMPONENTE	SERVICIO	TECNOLOGÍA	PROPÓSITO
Backend (API)	Render	Node.js + Express + Prisma	Servidor API REST
Base de Datos	Railway	MySQL 8	Persistencia relacional
Frontend (SPA)	Vercel	React + TypeScript + Vite	Interfaz de usuario (cliente web)

17.1 Despliegue de la Base de Datos en Railway

Objetivo: Alojar la base de datos MySQL del Sistema HRM en un servicio remoto, accesible desde Render y Vercel.

Pasos realizados

1. Se accedió a https://railway.app con la cuenta de GitHub vinculada al proyecto.
2. Se creó un nuevo proyecto con el nombre **"HRM Database"**.
3. Se seleccionó el **DATABASE**, y luego en Deploy se seleccionó **MySQL** el cual generó automáticamente una instancia con usuario, contraseña y puerto asignados.
4. Luego se **Exportó** la base local e importó en Railway con la opción mysqldump + mysql y se ejecutó en CMD para evitar problemas de rutas.

   Se exporta la base local (`hrm_db`) a un archivo (para ello se tuvo que crear una carpeta y utilizar su ruta):

   ```
   mysqldump -u root -p hrm_db > "C:\backup\hrm_db_backup.sql"
   ```

5. Ahora al tener el archivo de respaldo en:

```
C:\backup\hrm_db_backup.sql
```

Puedes abrirlo en el Bloc de notas o VS Code para confirmar.

6. Siguiente paso: subirlo a tu base en Railway

Ahora que el respaldo local está listo, ejecuta este comando (desde **CMD**)

```
mysql -h shinkansen.proxy.rlwy.net -P 26174 -u root -p
railway < C:\backup\hrm_db_backup.sql
```

Cuando te pida la contraseña, escribe la correspondiente:

****

Esto:

- Conectará a tu base de datos **Railway**
- Leerá tu archivo hrm_db_backup.sql
- Y creará todas las tablas, relaciones y datos de tu base local ahí mismo.

7. Para confirmar que se importó correctamente

Cuando termine, conéctate otra vez a Railway con:

```
mysql -h shinkansen.proxy.rlwy.net -P 26174 -u root -p railway
```

Y dentro del prompt escribe:

```
SHOW TABLES;
```

La cual debe mostrar:

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_railway  |
+--------------------+
| _prisma_migrations |
```

```
| audit_logs          |
| departments         |
| documents           |
| employees           |
| payroll_items       |
| payrolls            |
| users               |
+---------------------+
8 rows in set (0.15 sec)
```

Esto confirma **al 100 %** que la base de datos fue importada correctamente a Railway:

8. Desde el panel de Railway se obtuvo la **cadena de conexión (DATABASE_URL)** con el formato:
   mysql://user:password@containers.railway.app:port/railway

9. Esta cadena se configuró en el archivo .env del backend:

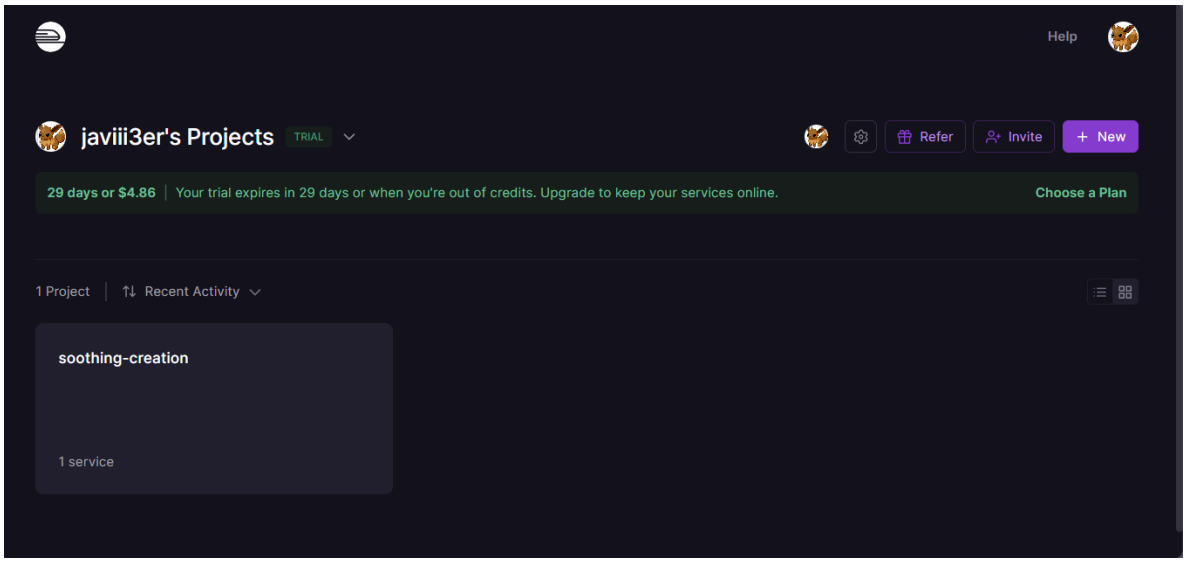
```
DATABASE_URL="mysql://user:password@containers.railway.app:port/
railway"
```

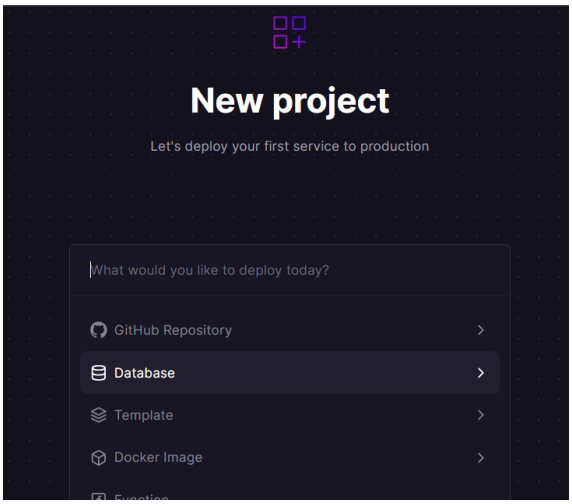
10. Se verificó la conexión ejecutando:

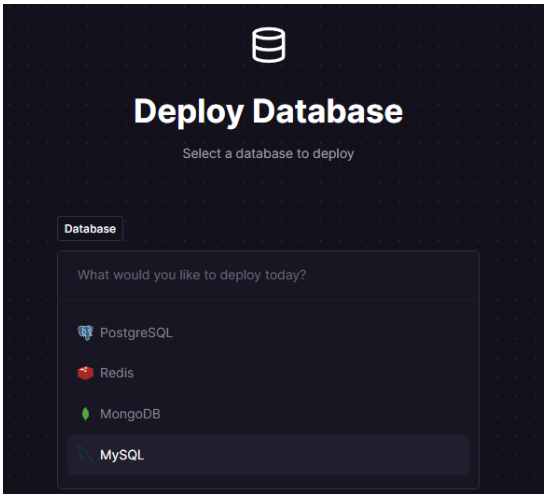
```
npx prisma generate
npx prisma db pull
```

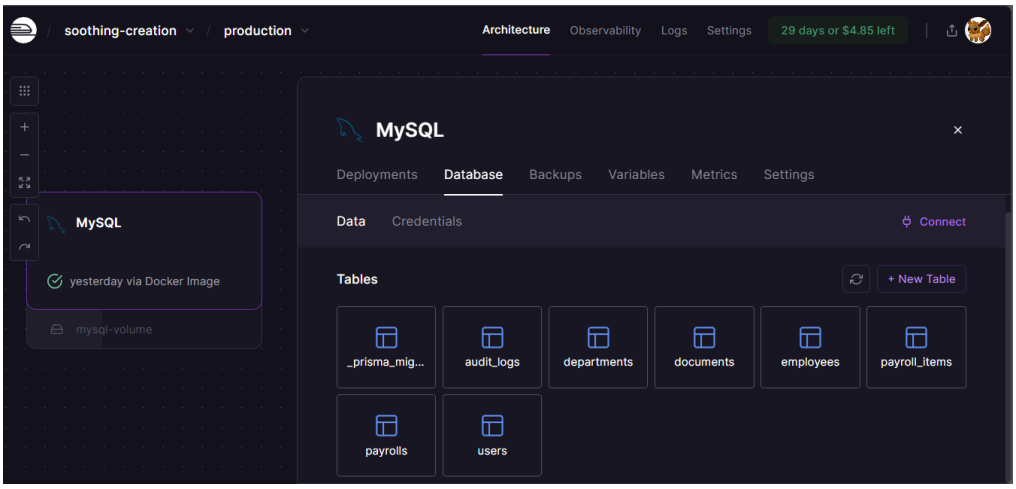
11. Resultado exitoso: “Datasource db: MySQL database hrm_db at railway.app”.

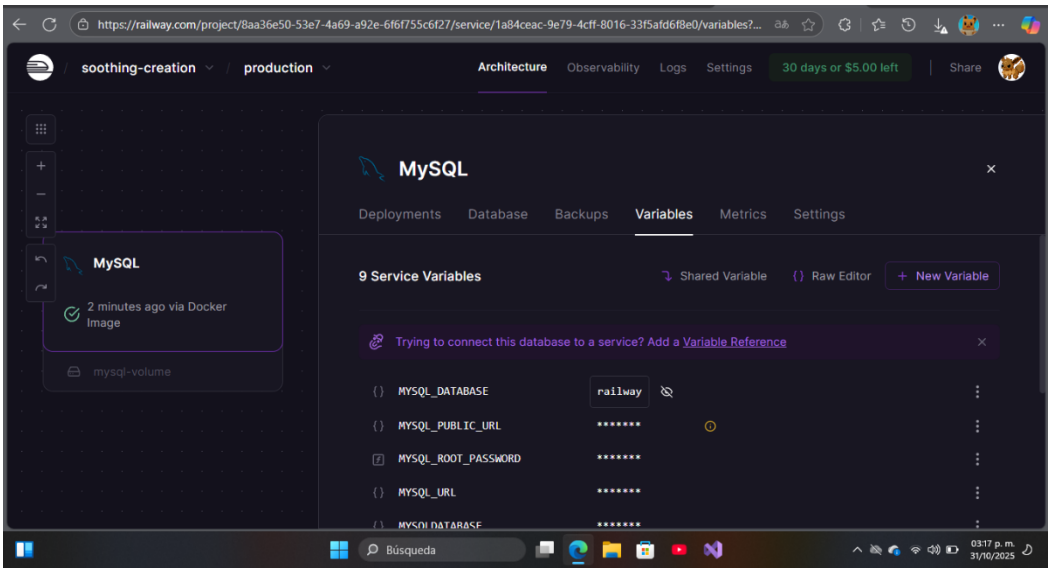
Railway confirmó la base de datos activa y accesible desde el servicio Render (backend).

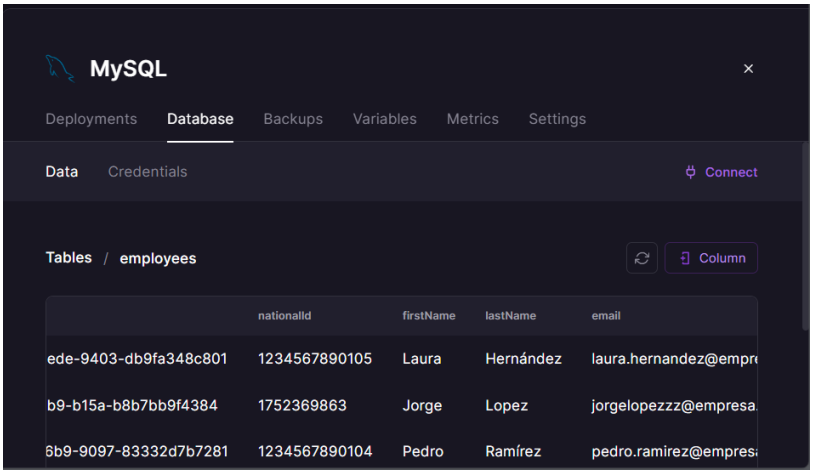

Help
javiii3er's Projects
TRIAL
Refer
Invite
+ New
29 days or $4.86 | Your trial expires in 29 days or when you're out of credits. Upgrade to keep your services online.
Choose a Plan
1 Project
Recent Activity
soothing-creation
1 service


New project
Let's deploy your first service to production
What would you like to deploy today?
GitHub Repository
Database
Template
Docker Image


Deploy Database
Select a database to deploy
Database
What would you like to deploy today?
PostgreSQL
Redis
MongoDB
MySQL







Resultado

La base de datos MySQL quedó desplegada y funcional en la nube, accesible desde cualquier servicio conectado mediante la URL protegida.

17.2 Despliegue del Backend en Render

Objetivo: Publicar la API RESTful del Sistema HRM (Node.js + TypeScript + Express) de forma segura y accesible.

Pasos realizados

1. Se accedió a https://render.com con la cuenta de GitHub.
2. Haz clic en el botón azul **"New +"** (arriba a la derecha).
3. Luego selecciona **"Web Service"**.

4. **Paso 2. Conectar el repositorio**
5. Render te pedirá que conectes tu cuenta de **GitHub**
6. Luego verás una lista con tus repositorios.
7. Busca y selecciona **hrm-backend**.
8. Haz clic en **"Connect"**.

9. En la configuración inicial:
    - **Environment:** Node
    - **Build Command:**
    - npm install && npm run build
    - **Start Command:**
    - node dist/server.js
    - **Branch:** main
    - **Auto Deploy:** Enabled (cada push a main genera nueva versión).

10. En la sección *Environment Variables*, se agregaron las siguientes claves:

    DATABASE_URL="mysql://user:password@containers.railway.app:port/railway"

    JWT_SECRET="clave_super_segura"

    JWT_REFRESH_SECRET="clave_refresco_segura"

NODE_ENV="production"

PORT=4000

Paso final: Crear el servicio

1. Verifica nuevamente que el plan seleccionado sea
   Free – 512 MB RAM / 0.1 CPU (sin costo).
   Este plan apaga el servidor cuando no hay actividad, pero para pruebas y despliegue académico es perfecto.

2. Revisa que los campos sean así:
   - **Build Command:** npm install && npm run build
   - **Start Command:** npm run start
   - **Root Directory:** (vacío)
   - **Region:** Ohio (US East)

3. Luego haz clic en **Deploy Web Service**.

Qué pasará después

- Render clonará tu repositorio desde GitHub.
- Instalará todas las dependencias.
- Compilará el código TypeScript.
- Iniciará el servidor con las variables que agregaste.

Mientras tanto, verás los logs en vivo

==> Uploading build...

==> Uploaded in 6.4s. Compression took 4.6s

==> Build successful 🎉

==> Deploying...

==> Running 'npm run start'

> hrm-backend@1.0.0 start

> node dist/server.js

Conectado a la base de datos MySQL

Servidor HRM ejecutándose...

Entorno: production

Puerto: 10000

Health check: http://localhost:10000/health

API Base: http://localhost:10000/api

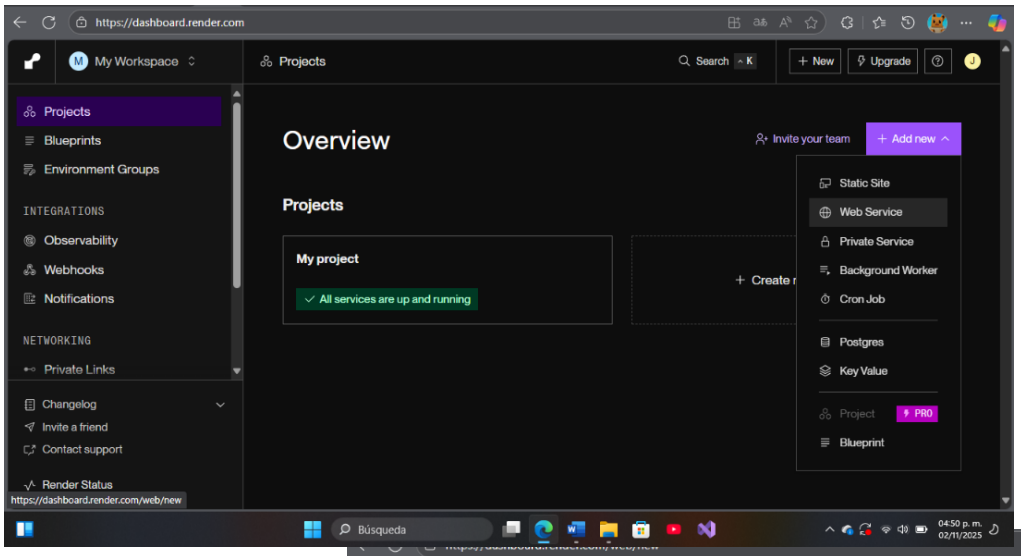
Descargas disponibles en: http://localhost:10000/downloads

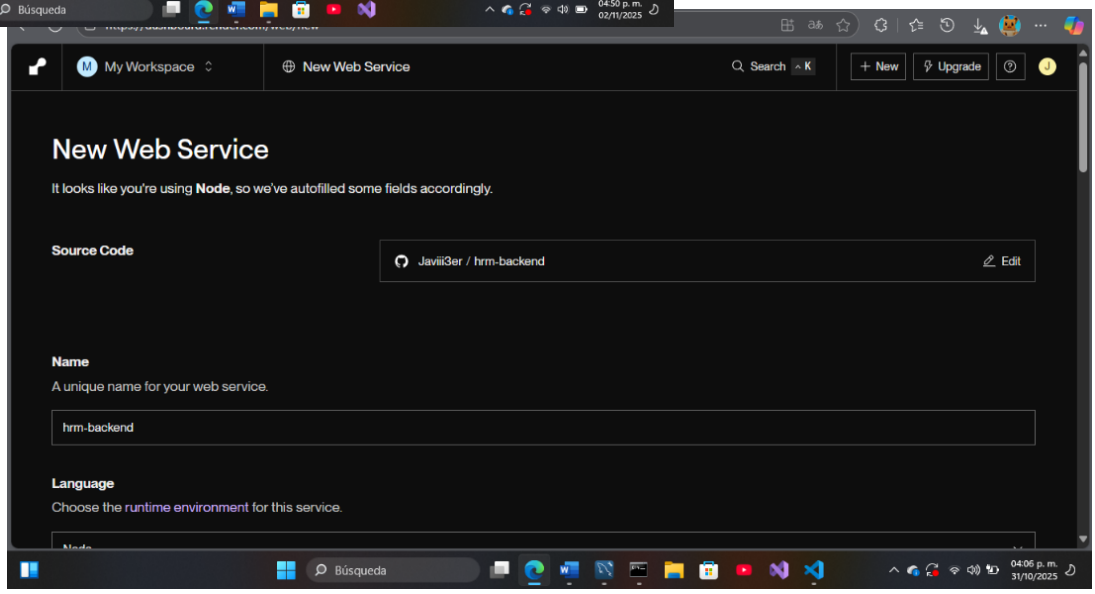
HEAD / 404 - 5ms

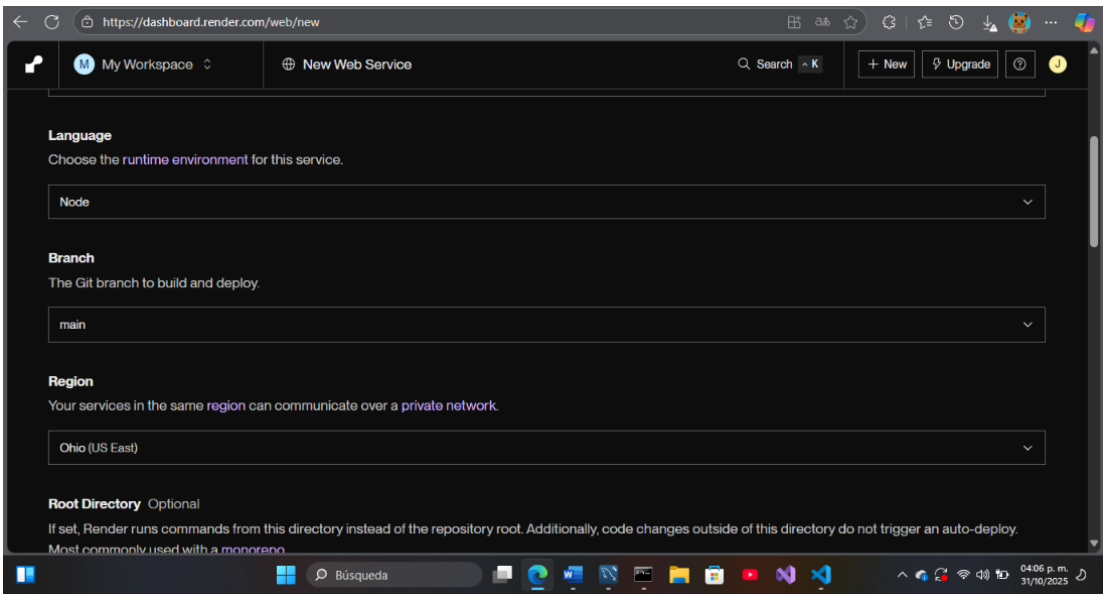
==> Your service is live 🎉

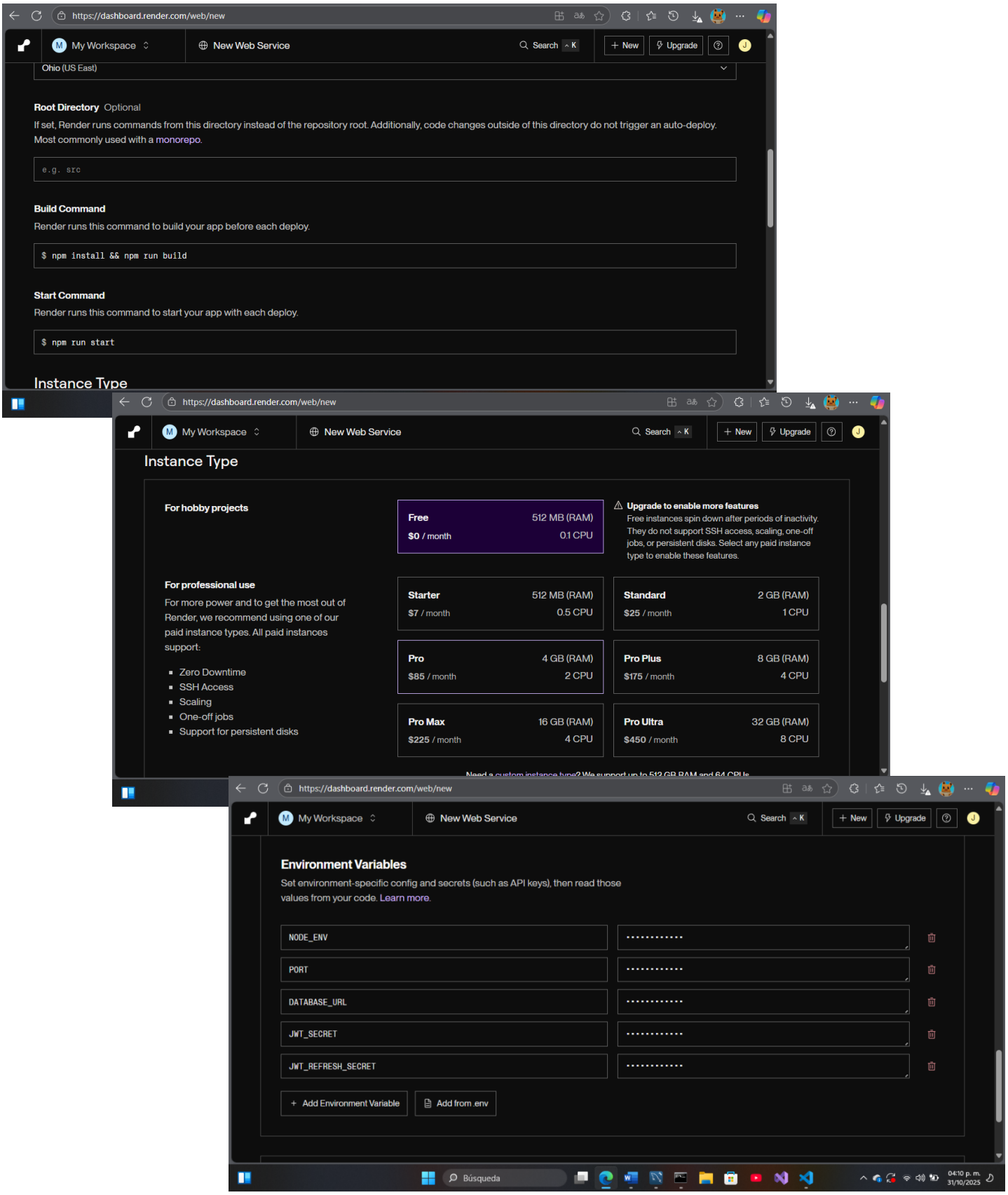
Resultado

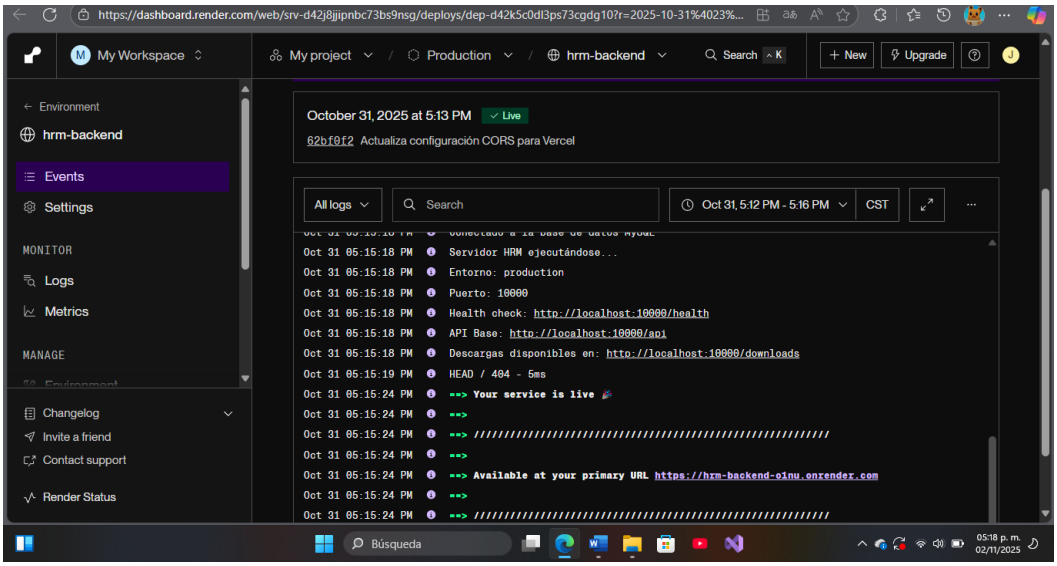
La API REST fue desplegada correctamente, con conexión estable hacia Railway y lista para recibir solicitudes del frontend hospedado en Vercel.


https://dashboard.render.com
My Workspace
Projects
Search
New
Upgrade
Projects
Blueprints
Environment Groups
INTEGRATIONS
Observability
Webhooks
Notifications
NETWORKING
Private Links
Changelog
Invite a friend
Contact support
Render Status
Overview
Invite your team
Add new
Projects
My project
All services are up and running
Static Site
Web Service
Private Service
Background Worker
Cron Job
Postgres
Key Value
Project
PRO
Blueprint
https://dashboard.render.com/web/new
Búsqueda
04:50 p. m.
02/11/2025


My Workspace
New Web Service
Search
New
Upgrade
New Web Service
It looks like you're using Node, so we've autofilled some fields accordingly.
Source Code
Javiii3er / hrm-backend
Edit
Name
A unique name for your web service.
hrm-backend
Language
Choose the runtime environment for this service.
Búsqueda
04:06 p. m.
31/10/2025


https://dashboard.render.com/web/new
My Workspace
New Web Service
Search
New
Upgrade
Language
Choose the runtime environment for this service.
Node
Branch
The Git branch to build and deploy.
main
Region
Your services in the same region can communicate over a private network.
Ohio (US East)
Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy.
Most commonly used with a monorepo
Búsqueda
04:06 p. m.
31/10/2025


https://dashboard.render.com/web/new
My Workspace
New Web Service
Search K
New
Upgrade
Ohio (US East)
Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.
e.g. src
Build Command
Render runs this command to build your app before each deploy.
$ npm install && npm run build
Start Command
Render runs this command to start your app with each deploy.
$ npm run start
Instance Type
https://dashboard.render.com/web/new
My Workspace
New Web Service
Search K
New
Upgrade
Instance Type
For hobby projects
Free
$0 / month
512 MB (RAM)
0.1 CPU
Upgrade to enable more features
Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.
For professional use
For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:
Zero Downtime
SSH Access
Scaling
One-off jobs
Support for persistent disks
Starter
$7 / month
512 MB (RAM)
0.5 CPU
Standard
$25 / month
2 GB (RAM)
1 CPU
Pro
$85 / month
4 GB (RAM)
2 CPU
Pro Plus
$175 / month
8 GB (RAM)
4 CPU
Pro Max
$225 / month
16 GB (RAM)
4 CPU
Pro Ultra
$450 / month
32 GB (RAM)
8 CPU
https://dashboard.render.com/web/new
My Workspace
New Web Service
Search K
New
Upgrade
Environment Variables
Set environment-specific config and secrets (such as API keys), then read those values from your code. Learn more.
NODE_ENV
PORT
DATABASE_URL
JWT_SECRET
JWT_REFRESH_SECRET
Add Environment Variable
Add from .env
Búsqueda
04:10 p. m.
31/10/2025


https://dashboard.render.com/web/srv-d42j8jijpnbc73bs9nsg/deploys/dep-d42k5c0dl3ps73cgdg10?r=2025-10-31%4023%...
My Workspace
My project
Production
hrm-backend
Search
New
Upgrade
Environment
hrm-backend
Events
Settings
MONITOR
Logs
Metrics
MANAGE
Changelog
Invite a friend
Contact support
Render Status
October 31, 2025 at 5:13 PM
Live
62bf0f2 Actualiza configuración CORS para Vercel
All logs
Search
Oct 31, 5:12 PM - 5:16 PM
CST
Oct 31 05:15:18 PM Servidor HRM ejecutándose...
Oct 31 05:15:18 PM Entorno: production
Oct 31 05:15:18 PM Puerto: 10000
Oct 31 05:15:18 PM Health check: http://localhost:10000/health
Oct 31 05:15:18 PM API Base: http://localhost:10000/api
Oct 31 05:15:18 PM Descargas disponibles en: http://localhost:10000/downloads
Oct 31 05:15:19 PM HEAD / 404 - 5ms
Oct 31 05:15:24 PM ==> Your service is live
Oct 31 05:15:24 PM ==>
Oct 31 05:15:24 PM ==> ///
Oct 31 05:15:24 PM ==>
Oct 31 05:15:24 PM ==> Available at your primary URL https://hrm-backend-o1nu.onrender.com
Oct 31 05:15:24 PM ==>
Oct 31 05:15:24 PM ==> ///
Búsqueda
05:18 p. m.
02/11/2025

17.3 Despliegue del Frontend en Vercel

Objetivo: Publicar la aplicación SPA del Sistema HRM desarrollada con React + Vite.

Pasos realizados

1. Se accedió a https://vercel.com con la cuenta de GitHub .
2. Se seleccionó **New Project → Import Git Repository → hrm-frontend**.
3. Durante la configuración inicial:
    - **Framework detectado:** Vite + React
    - **Build Command:** npm run build
    - **Output Directory:** dist
4. Se agregó en el panel de Vercel la variable de entorno:
5. VITE_API_BASE_URL="https://hrm-backend-o1nu.onrender.com/api"
6. Se hizo clic en **Deploy** y Vercel ejecutó automáticamente:

    npm install

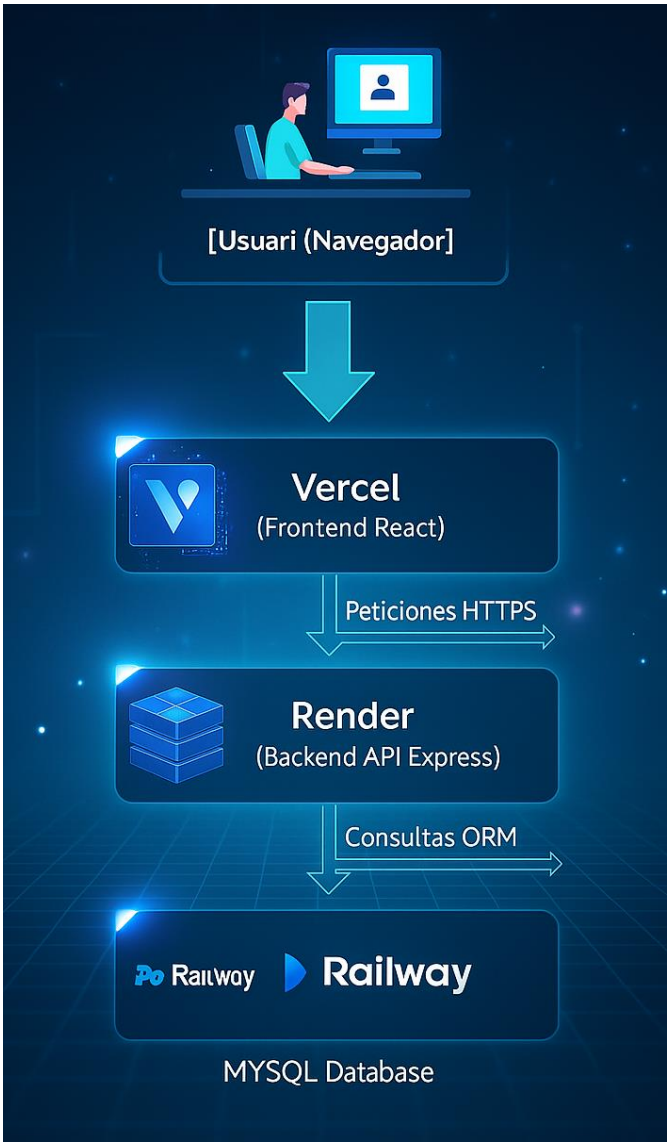
    npm run build

7. El despliegue finalizó con el mensaje:
8. Congratulations! You just deployed a new project to Usuauario projects.
9. La URL pública generada fue: https://hrm-frontend-sigma-three.vercel.app
10. Se verificó el funcionamiento completo accediendo al login del sistema.
    El frontend logró comunicarse correctamente con la API Render y la base de datos Railway, confirmando la integración final.

Resultado

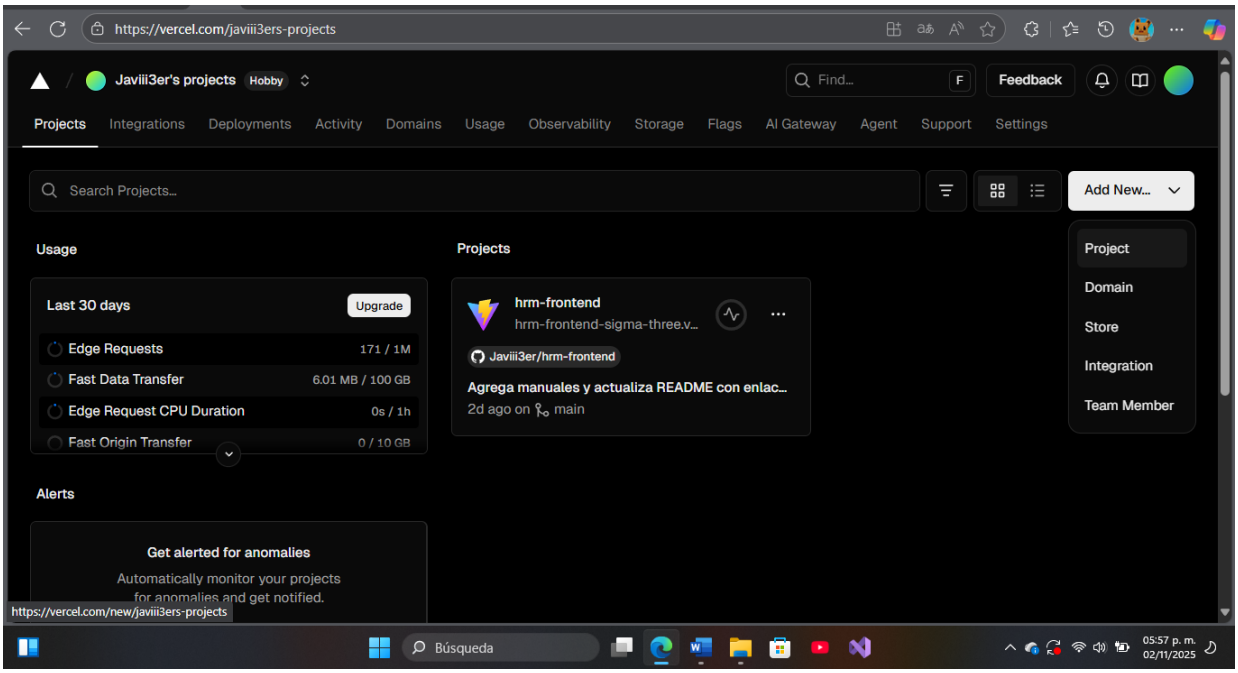
El frontend quedó accesible globalmente bajo HTTPS, compilado y optimizado, con conexión funcional al backend y base de datos remota.

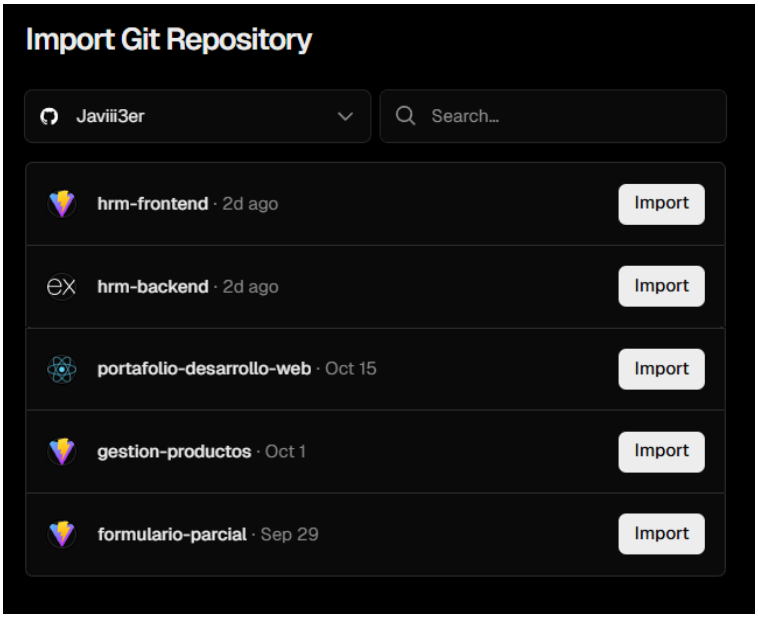
Flujo Completo del Sistema en Producción

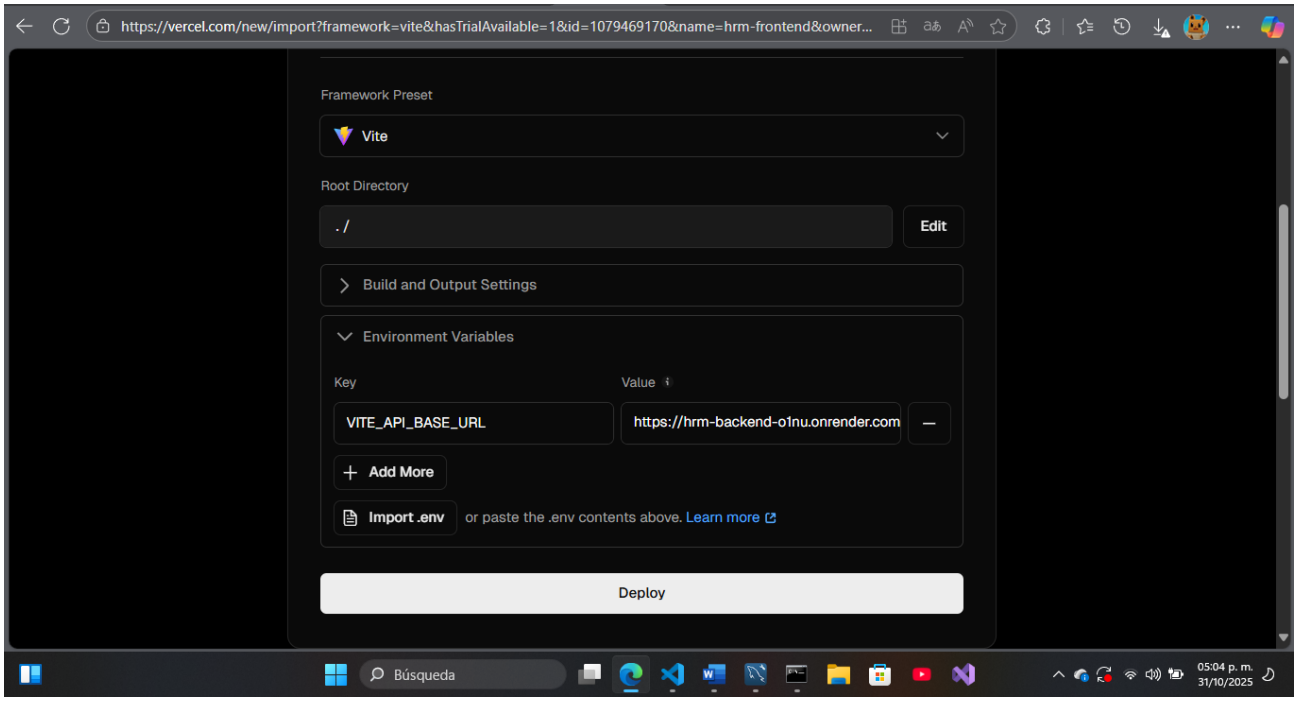


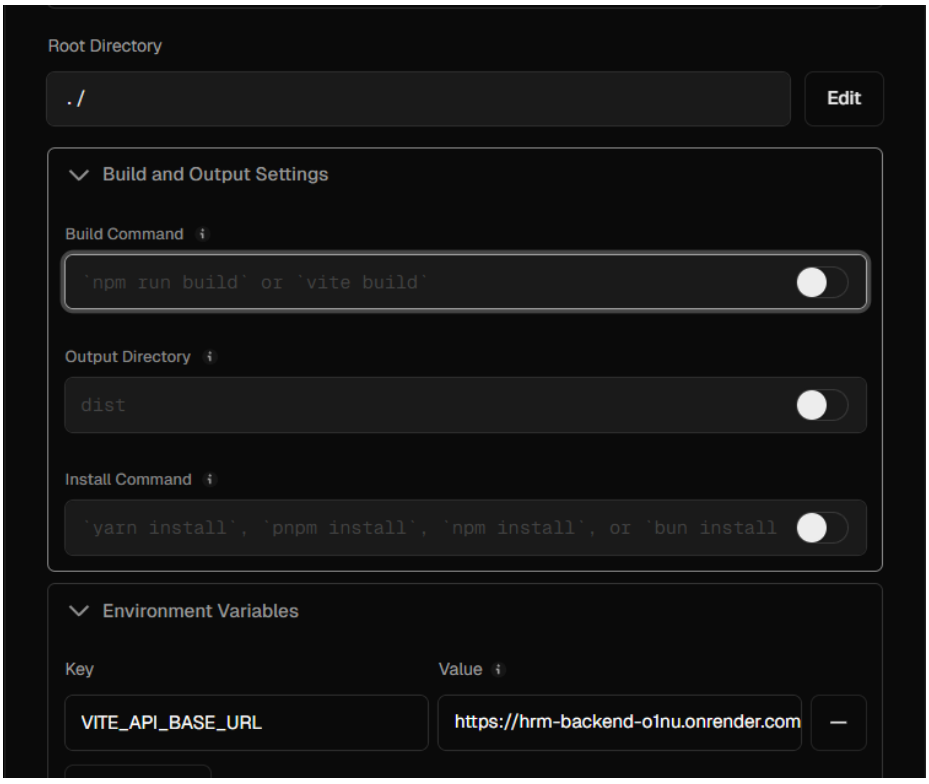
Endpoints de Producción:

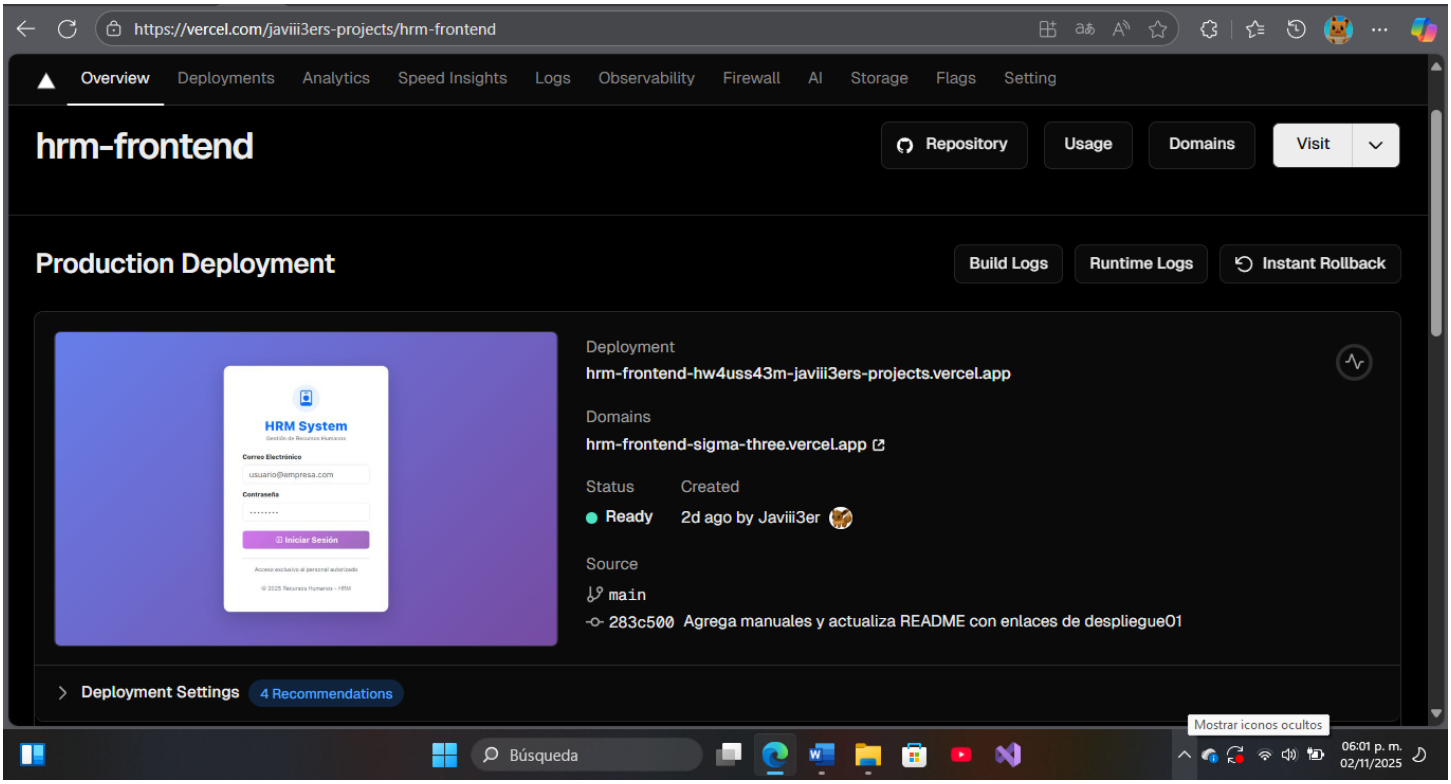
Componente	URL
Frontend	https://hrm-frontend-sigma-three.vercel.app
Backend	https://hrm-backend-o1nu.onrender.com
API Health Check	https://hrm-backend-o1nu.onrender.com/api/health






https://vercel.com/new/import?framework=vite&hasTrialAvailable=1&id=1079469170&name=hrm-frontend&owner...
Framework Preset
Vite
Root Directory
./
Edit
Build and Output Settings
Environment Variables
Key
Value
VITE_API_BASE_URL
https://hrm-backend-o1nu.onrender.com
Add More
Import .env
or paste the .env contents above. Learn more
Deploy
Búsqueda
05:04 p. m.
31/10/2025


Root Directory
./
Edit
Build and Output Settings
Build Command
`npm run build` or `vite build`
Output Directory
dist
Install Command
`yarn install`, `pnpm install`, `npm install`, or `bun install`
Environment Variables
Key
Value
VITE_API_BASE_URL
https://hrm-backend-o1nu.onrender.com


https://vercel.com/javiii3ers-projects/hrm-frontend
Overview
Deployments
Analytics
Speed Insights
Logs
Observability
Firewall
AI
Storage
Flags
Setting
hrm-frontend
Repository
Usage
Domains
Visit
Production Deployment
Build Logs
Runtime Logs
Instant Rollback
HRM System
Deployment
hrm-frontend-hw4uss43m-javiii3ers-projects.vercel.app
Domains
hrm-frontend-sigma-three.vercel.app
Status
Created
Ready
2d ago by Javiii3er
Source
main
283c500 Agrega manuales y actualiza README con enlaces de despliegue01
Deployment Settings
4 Recommendations
Búsqueda
Mostrar iconos ocultos
06:01 p. m.
02/11/2025

CONCLUSIÓN

El **Sistema HRM (Backend)** implementa una arquitectura moderna, escalable y segura, basada en principios REST y buenas prácticas de desarrollo profesional.
Su despliegue cloud en **Render (API)**, **Vercel (Frontend)** y **Railway (Base de Datos)** garantiza accesibilidad global, independencia de servicios y mantenimiento sencillo.

Este documento sirve como **referencia técnica completa** para cualquier desarrollador, docente o evaluador que necesite comprender la infraestructura, el flujo y las decisiones tecnológicas detrás del proyecto HRM.

La integración completa del ecosistema **Railway + Render + Vercel** permite que el sistema HRM opere de forma remota, segura y estable, sin necesidad de ejecutar servicios locales (como npm run dev o MySQL local).
Todos los módulos del sistema (autenticación, empleados, nóminas, documentos, reportes) son accesibles vía web desde cualquier dispositivo.