MANUAL TÉCNICO

Aplicación Móvil SIGE

10 DE DICIEMBRE DE 2025
INSTITUTO DE BIENESTAR FAMILIAR
Av. Tenerife #1e-40, Neiva, Huila

Contenido

1. Descripción General del Sistema 3
2. Arquitectura del Sistema 3
2.1 Enfoque Arquitectónico 4
2.2 Modelo de Capas (Vista Lógica) 4
Capa 1 – Cliente Móvil 4
Capa 2 – Servidor / API Backend 4
Capa 3 – Base de Datos 4
Capa 4 – Servicios Externos 5
3. Frontend Móvil – Flutter 5
3.1 Tecnologías Implementadas 5
3.2 Diseño de UI y UX 5
3.3 Arquitectura de Carpetas 6
3.4 Comunicación con Backend 6
4. Backend – Node.js + Express 6
4.1 Arquitectura del Backend 6
4.2 Objetivos del backend 7
4.3 Integración Firebase Storage 8
5. Base de Datos – MongoDB Atlas 9
5.1 Estructura de Colecciones 9
5.1.1 users 9
5.1.2 policies 9
5.1.3 notifications 9
6. Seguridad del Sistema 10
6.1 Autenticación 10
6.2 Encriptación 10
6.3 Políticas de Seguridad 10
7. Despliegue e Infraestructura 11
7.1 Integración Continua (CI/CD) 11
7.2 Configuración de Variables de Entorno (.env) 11

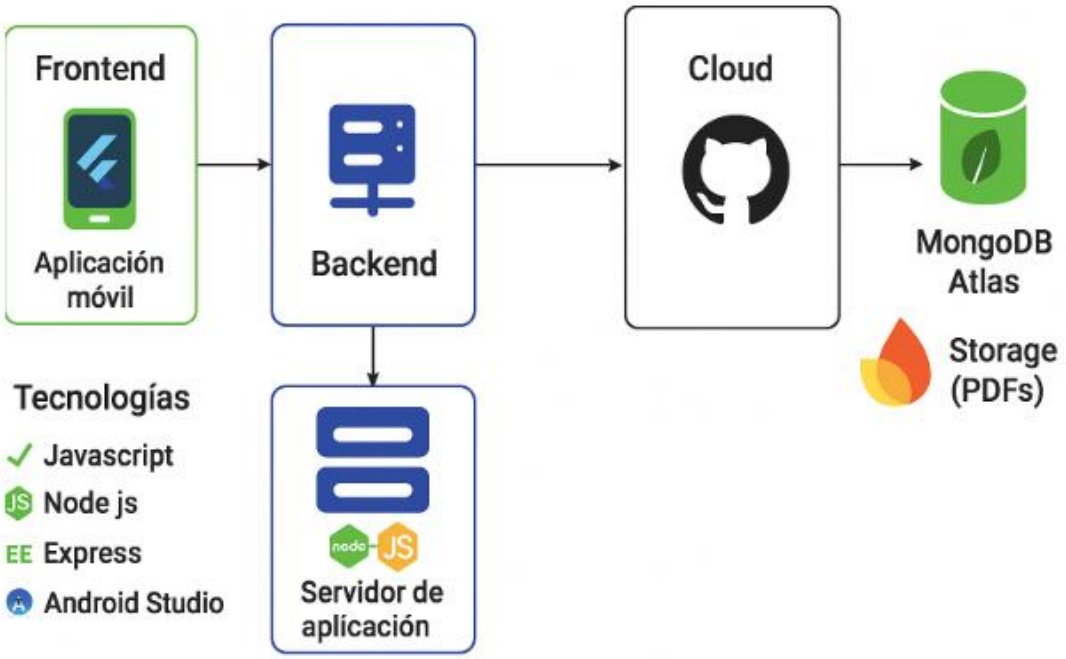
7.3 Flujo de despliegue 11
7.5 Elección de Render como plataforma de despliegue 12
7.6 Arquitectura de despliegue 12
7.7 Configuración del Servicio en Render 13
Paso 1 — Conexión del Repositorio 13
Paso 2 — Configuración de Entorno 13
8. Mantenibilidad y Escalabilidad 15
9. Procedimientos Técnicos Esenciales 15
9.1 Actualización del Backend 15
9.2 Actualización del Frontend 15
Generar APK o AAB: 15
10. Solución de Problemas 16
11. Conclusiones Técnicas 16

1. Descripción General del Sistema

El Sistema Integrado de Gestión Educativa (SIGE) es una aplicación móvil desarrollada para optimizar la consulta, difusión y administración de políticas institucionales, documentos oficiales, módulos de inducción y contenido estratégico del ICBF. Su arquitectura distribuida permite una interacción eficiente entre el cliente móvil, los servicios backend y los recursos almacenados en la nube, garantizando escalabilidad, disponibilidad y seguridad.

Desde una perspectiva técnica, SIGE incorpora principios de arquitectura limpia, modularidad, desacoplamiento, servicios RESTful, seguridad basada en tokens JWT y servicios cloud nativos, asegurando un sistema robusto y administrable a largo plazo.

2. Arquitectura del Sistema



2.1 Enfoque Arquitectónico

- SIGE adopta una arquitectura Cliente–Servidor reforzada con componentes en la nube y servicios de integración externa. Su diseño se estructura de la siguiente manera:
- Capa de Presentación (Frontend móvil) desarrollada en Flutter
- Capa de Servicios (Backend – API REST) implementada en Node.js con Express
- Capa de Datos basada en MongoDB Atlas
- Capa de Recursos Digitales en Firebase Storage
- Capa de Seguridad mediante JWT e integración con reglas de política de acceso

2.2 Modelo de Capas (Vista Lógica)

Capa 1 – Cliente Móvil

- Interface de usuario (UI)
- Lógica de presentación
- Consumo de API REST
- Manejo de estados simples basados en Stateful Widgets

Capa 2 – Servidor / API Backend

- Servicios de negocio (auth, documentos, políticas, usuarios)
- Validación de datos
- Gestión de roles
- Generación de token JWT
- Procesamiento de archivos PDF e imágenes

Capa 3 – Base de Datos

- Almacenamiento de documentos NoSQL
- Acceso mediante Mongoose (ODM)
- Estructuras jerárquicas y colecciones normalizadas

Capa 4 – Servicios Externos

- Firebase Storage para archivos
- Render para despliegue y orquestación
- GitHub como repositorio y sistema de control de versiones

3. Frontend Móvil – Flutter

3.1 Tecnologías Implementadas

- Flutter SDK 3.x (arquitectura declarativa)
- Dart 3.x

Paquetes principales:

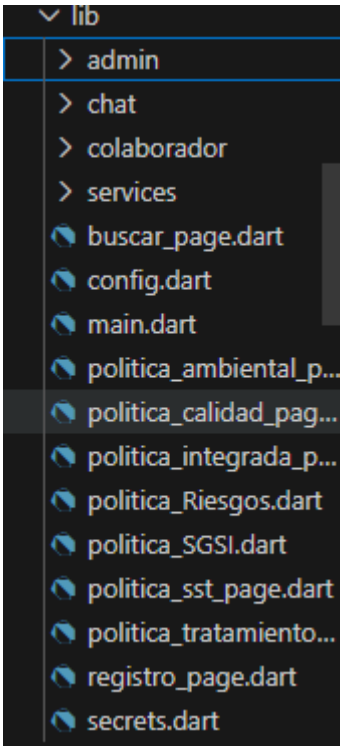
- http (consumo de servicios)
- image_picker (captura y selección de imágenes)
- flutter_dotenv (gestión de entornos)
- path_provider
- url_launcher

3.2 Diseño de UI y UX

El diseño de SIGE prioriza:

- Navegación fluida basada en bottom navigation bar
- Tarjetas informativas (cards) que agrupan pilares estratégicos
- Interfaz limpia con colores institucionales
- Componentes reutilizables para mejorar mantenibilidad

3.3 Arquitectura de Carpetas



3.4 Comunicación con Backend

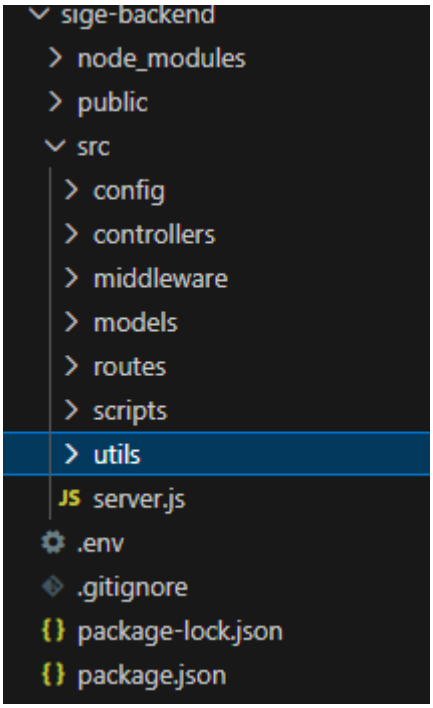
La app consume servicios REST usando métodos como:

- GET /api/policies
- POST /api/auth/login
- PUT /api/users/update
- POST /api/notifications/create

4. Backend – Node.js + Express

4.1 Arquitectura del Backend

El backend utiliza una arquitectura modularizada:



4.2 Objetivos del backend

- Gestionar autenticación segura (JWT)
- Manejar CRUD de políticas, roles, documentos y notificaciones
- Filtrar y validar información
- Subir archivos al servidor de Firebase Storage
- Emitir eventos de notificaciones para usuarios

4.2 Controladores Principales

Autenticación:

- Registro
- Inicio de sesión
- Renovación de perfil
- Validación de credenciales

Gestión de Políticas:

- Carga de documentos PDF
- Eliminación lógica
- Asignación por categorías

Gestión de Roles:

- Asignación de rol admin
- Roles de colaborador y coordinador
- Control de permisos por endpoint

Notificaciones

- Creación automática tras eventos críticos
- Consulta histórica por usuario

4.3 Integración Firebase Storage

El backend recibe imágenes en Base64 provenientes de Flutter, las procesa y ejecuta:

1. Conversión de Base64 a buffer
2. Subida al bucket Firebase
3. Obtención del enlace público
4. Almacenamiento en MongoDB

5. Base de Datos – MongoDB Atlas

5.1 Estructura de Colecciones

5.1.1 users

```
{
 name: String,
 email: String,
 phone: String,
 role: String,
 password: String,
 photoUrl: String,
 createdAt: Date
}
```

5.1.2 policies

```
{
 title: String,
 category: String,
 pdfUrl: String,
 createdBy: ObjectId,
 createdAt: Date
}
```

5.1.3 notifications

```
{
 type: String,
 message: String,
 userId: ObjectId,
```

createdAt: Date

}

5.1.4 change_logs

{

policyId: ObjectId,

description: String,

createdAt: Date

}

6. Seguridad del Sistema

6.1 Autenticación

- Se implementa mediante:
- JWT con tiempos de expiración definidos
- Middleware para validar tokens
- Rutas protegidas bajo autorización por rol

6.2 Encriptación

Las contraseñas se almacenan aplicando:

bcryptjs hash(salt=10)

6.3 Políticas de Seguridad

- Helmet (protección HTTP)
- CORS configurado para permitir solo orígenes autorizados
- Validación estricta para subida de archivos
- Restricciones para roles administrativos

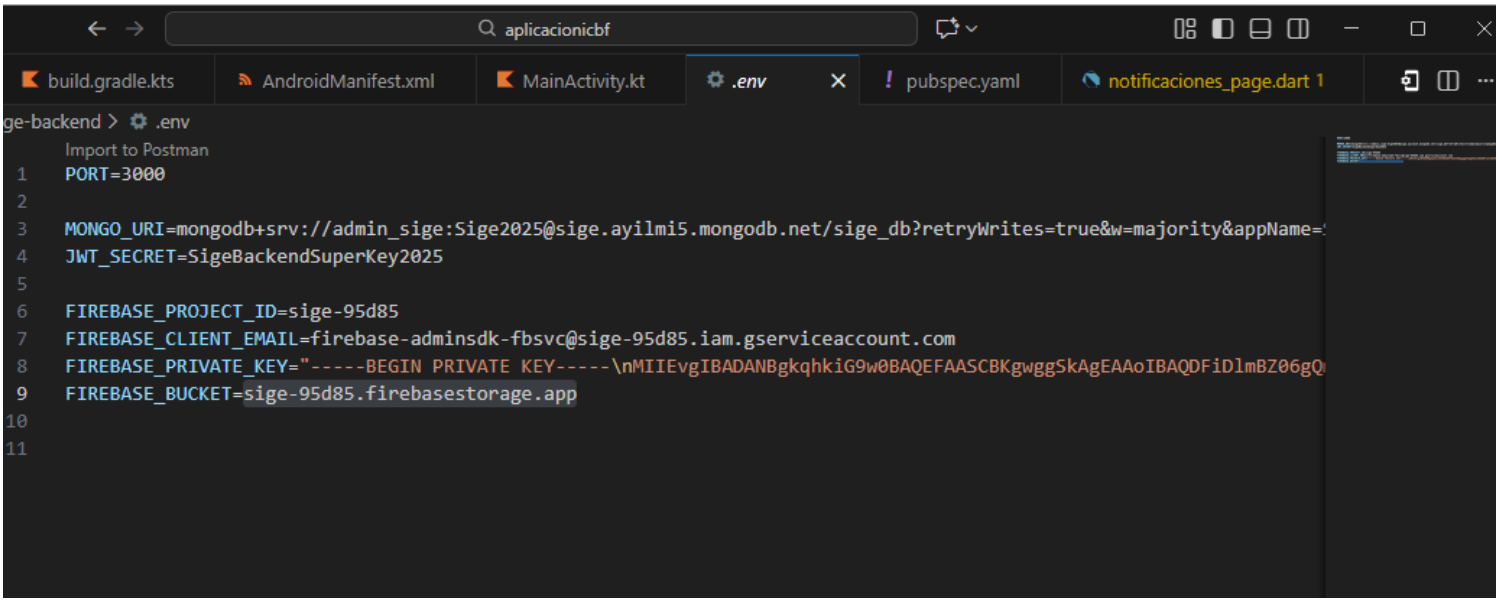
7. Despliegue e Infraestructura

7.1 Integración Continua (CI/CD)

- Render ejecuta automáticamente:
- Instalación de dependencias
- Construcción
- Ejecución del servidor

7.2 Configuración de Variables de Entorno (.env)

Las más relevantes son:



7.3 Flujo de despliegue

1. Código se envía a GitHub
2. Render detecta cambios
3. Ejecuta pipeline
4. Reinicia servicio en producción

7.4 Despliegue e Infraestructura en Render

El entorno de despliegue del sistema SIGE se implementó en la plataforma Render.com, un servicio de hosting en la nube orientado a aplicaciones web modernas. Render permite ejecutar el backend de SIGE de manera automática y escalable, integrándose con GitHub para habilitar flujos de CI/CD sin configuraciones complejas.

A continuación, se describe la implementación técnica del backend en Render.

7.5 Elección de Render como plataforma de despliegue
Para alojar el servidor de SIGE se seleccionó Render debido a:

Despliegue automatizado mediante integración con GitHub.

- Entorno de ejecución compatible con Node.js 25.
- Soporte para variables de entorno protegidas.
- HTTPS habilitado por defecto.
- Escalabilidad automática para el tráfico esperado.
- Logs centralizados para monitoreo.
- Reinicio automático del servicio ante fallos.
- Render proporciona un entorno robusto y administrado, adecuado para aplicaciones institucionales basadas en arquitectura cliente–servidor.

7.6 Arquitectura de despliegue

El backend en Render se ejecuta como un Web Service, el cual expone los endpoints REST utilizados por la aplicación móvil. La arquitectura se compone de:

- Web Service Render
- Ejecuta el servidor Node.js
- Escucha en el puerto estándar asignado por Render (process.env.PORT)
- MongoDB Atlas como base de datos externa
- Firebase Storage como almacenamiento de archivos
- Aplicación Flutter como cliente externo

La comunicación se establece mediante HTTPS, cumpliendo con requerimientos de seguridad institucional.

7.7 Configuración del Servicio en Render

Paso 1 — Conexión del Repositorio

El repositorio del backend se aloja en GitHub.

En Render:

1. Crear un nuevo Web Service.
2. Seleccionar "Build from Git repository".
3. Conectar la cuenta de GitHub.
4. Seleccionar el repositorio SIGE-Backend.

Paso 2 — Configuración de Entorno

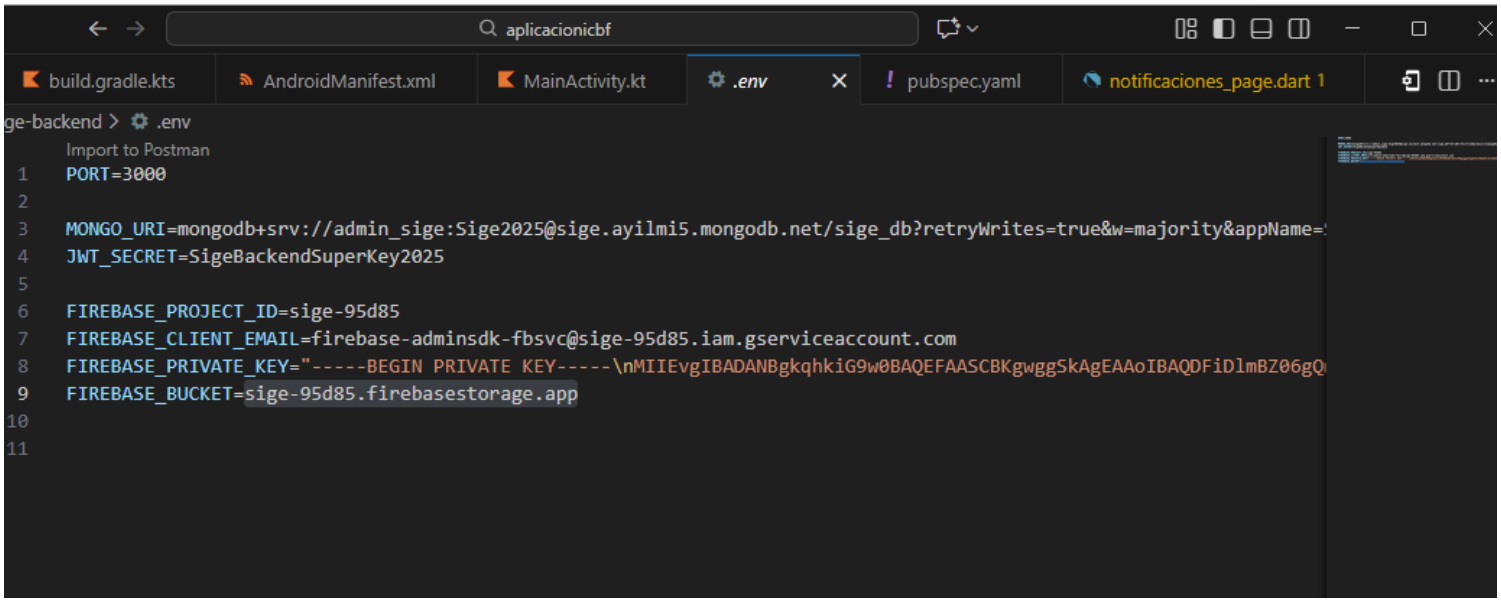
En Render se definieron los siguientes parámetros:

1. Runtime
2. Lenguaje: Node.js
3. Versión recomendada: 25.x

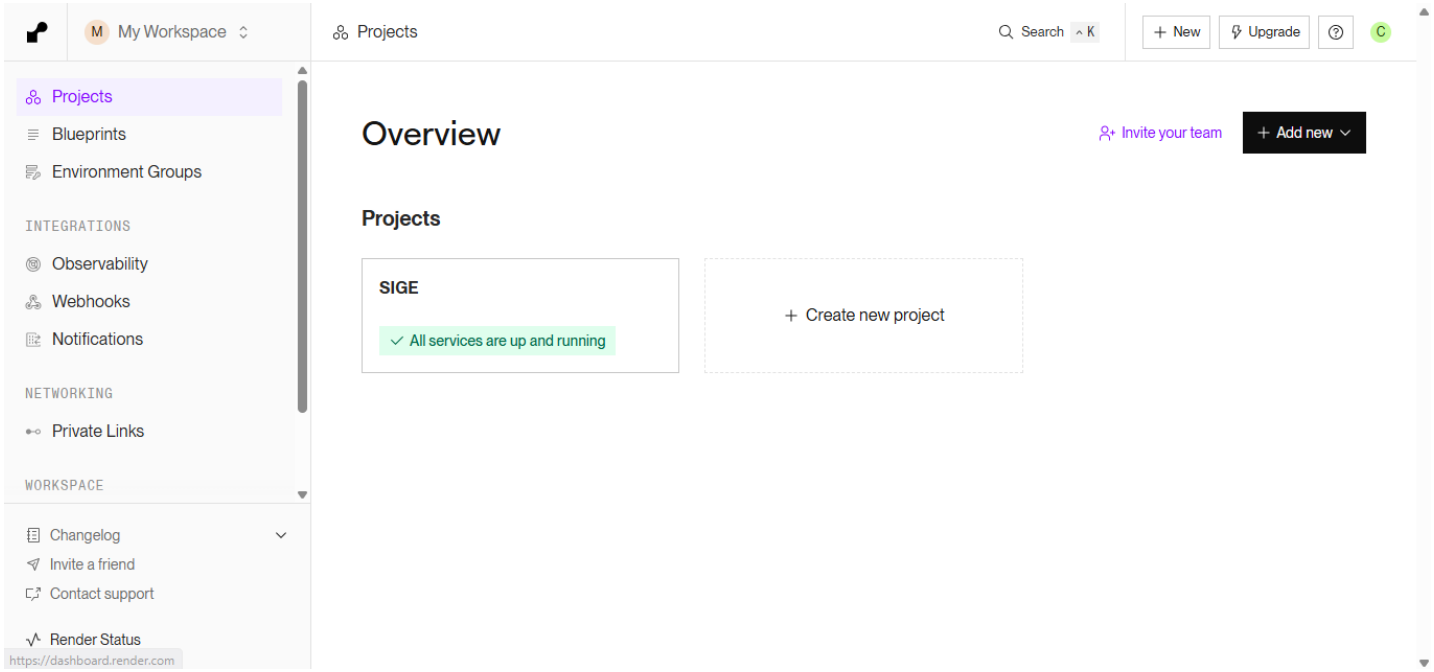
Comandos:

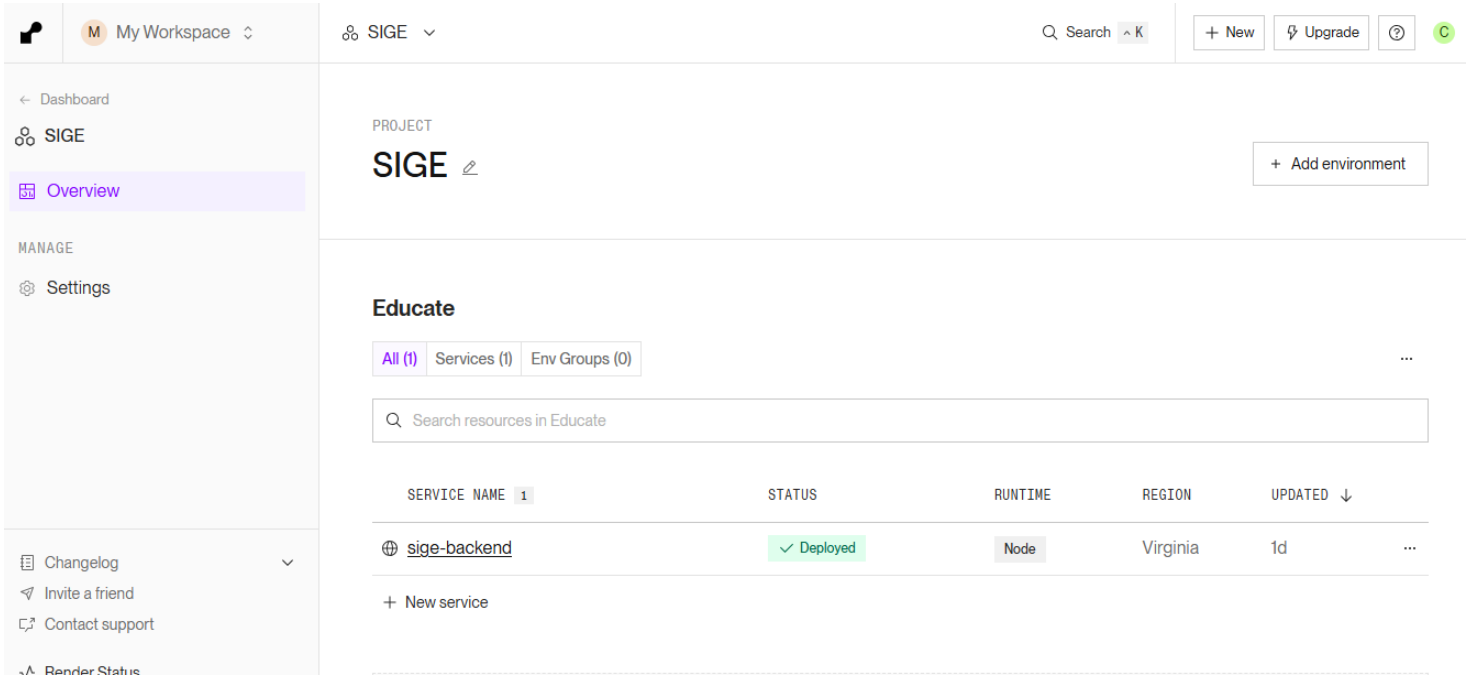
- npm install
- npm start
- node src/server.js

Variables de entorno:



Final de Render:





8. Mantenibilidad y Escalabilidad

SIGE está diseñado para crecer en funcionalidades mediante:

✓ Separación de responsabilidades

✓ Componentización

✓ Modelo NoSQL expandible

✓ Servicios REST desacoplados

✓ Uso de almacenamiento en la nube

9. Procedimientos Técnicos Esenciales

9.1 Actualización del Backend

- Crear nueva rama
- Modificar controladores según requerimiento
- Probar localmente con Postman
- Hacer merge hacia main
- Render aplicará despliegue automático

9.2 Actualización del Frontend

- Ejecutar flutter pub get
- Probar en emulador Android

Generar APK o AAB:

- flutter build apk --release
- flutter build appbundle –release

10. Solución de Problemas

Problema	Causa Técnica	Solución
401 Unauthorized	Token vencido	Generar nueva sesión
No carga documentos	API incorrecta	Verificar config.dart
Imagen no sube	Error Firebase Key	Revisar .env
Backend caído	Suspendido por inactividad	Reactivar Render
Flutter falla	Permisos insuficientes	Revisar AndroidManifest

11. Conclusiones Técnicas

SIGE constituye una solución móvil robusta, escalable y altamente integrada con servicios cloud que permite la gestión institucional del ICBF de forma centralizada. Su arquitectura híbrida entre Flutter, Node.js, MongoDB y Firebase ofrece un ecosistema tecnológico moderno, seguro y adaptable a la evolución futura del sistema.