Repositorios Carpeta Ciudadana (Soy Yo)

Herramienta centralizada para almacenar y gestionar la información personal y documentos relevantes de los ciudadanos, con el objetivo de mejorar la eficiencia y la calidad de los servicios gubernamentales.

- Análisis Técnico del Repositorio: Carpeta Ciudadana Mobile
- Análisis Técnico del Repositorio: Carpeta Ciudadana Backend
- Análisis Técnico del Repositorio: Carpeta Ciudadana Landing
- Análisis Técnico del Repositorio: totp-cc
- Análisis Técnico del Repositorio: cc-institutions-mock-api

Análisis Técnico del Repositorio: Carpeta Ciudadana Mobile

Información General

Nombre del Proyecto: Carpeta Ciudadana Mobile

Repositorio: github.com/ogticrd/carpeta-ciudadana-mobile

Una **aplicación móvil** destinada a los **ciudadanos** para interactuar con servicios digitales del gobierno.

Elemento	Observaciones
`.github/`	Contiene workflows de GitHub Actions (CI/CD). Se menciona una tarea de *cleanup cache*, lo que sugiere optimización de builds.
`src/`	Carpeta principal de código fuente. Cambios recientes sugieren mantenimiento activo.
`assets/`, `accests/`	Dos carpetas similares; podría haber una confusión de nombres o duplicación. Revisar si ambas son necesarias.
`.env.example`, `.env.local`	Uso de variables de entorno. Correcta separación entre variables públicas y privadas. Incluye integración con Sentry.
`App.tsx`	Proyecto basado en React Native (o React + Expo). Componente raíz de la app.
`Dockerfile`	Soporte para contenerización. Permite pruebas y despliegue estandarizado.
`GoogleService-Info.plist` / `google-services.json`	Indica integración con Firebase tanto para iOS como Android. Muy común en apps móviles.
`babel.config.js`, `metro.config.js`	Configuración típica de React Native.
`eas.json`	Archivo de configuración de EAS (Expo Application Services). Se usa para builds y actualizaciones OTA.
`package.json`, `tsconfig.json`	Proyecto en TypeScript, bien estructurado.

Stack Tecnológico

Elemento	Descripción
Framework base	React Native
Entorno móvil	Expo con EAS (Expo Application Services)
Lenguaje	TypeScript
Gestión de estado	Redux Toolkit + React Query
Navegación	React Navigation
Firebase	Integrado (push notifications, auth, etc.)
Feature flags	Unleash + GrowthBook
Autenticación	Expo Auth Session, Firebase, LocalAuth
Observabilidad	Sentry
Fuentes	Google Fonts (Poppins)

DevOps / Automatización

Área	Herramienta/Archivo	Comentarios
CI/CD	`.github/workflows` (presente)	GitHub Actions configurado (ej. limpieza de caché, tests)
Contenedores	`Dockerfile`	Contenedor de pruebas configurado
Reset de entorno	`"reset-project"`	Script personalizado (`./scripts/reset-project.js`) para reinicio de entorno. Declarado pero no se encontró el archivo – revisar si existe.
EAS (Expo Application Services)	`eas.json`	Integración con EAS: build y deployment OTA para móviles
Seguridad	`dependabot.yml`	Monitoreo automático de versiones y alertas

Análisis de Workflows CI/CD (`.github/workflows/`)

Archivo de Workflow	Propósito	Herramientas clave	Estado
`ci-check-linters.yml`	Ejecuta linters para mantener calidad de código	`eslint`, posiblemente `expo lint`	Activo y útil para calidad

Archivo de Workflow	Propósito	Herramientas clave	Estado
`cleanup-cache-branch.yml`	Limpia la caché de branches antiguas o cerradas	`actions/github-script`, `cache`	Mantenimiento automatizado
`preview-deployment.yml`	Despliega versiones de preview (probablemente PRs)	`Expo`, `EAS`, o similar	Muy útil para validaciones previas
`preview-deployment-slack-pr.yml`	Notifica en Slack sobre despliegues de preview	`slackapi/slack-github-action`	Integración con Slack para visibilidad

Calidad del Código

Herramienta	Presencia	Comentarios
ESLint	Detectado	Usa `@typescript-eslint`, `eslint-config-expo`
Prettier	No explícitamente declarado	
Análisis estático	No SonarQube u otra herramienta detectada	
Depuración de dependencias	Detectado	Usa `depcheck` y `unimported` para limpiar dependencias no usadas

Seguridad

Elemento	Presencia	Comentarios
`.env.example`	Si	Correcto uso para variables de entorno públicas
`.env.local`	Si	Presente en el repo (riesgo si contiene claves)
Sentry	Si	Integración con `@sentry/react-native`
Control de calidad estático (SAST)	NO	No detectado. Recomendado para escaneo de vulnerabilidades

Dependencias y Escalabilidad

Tipo	Ejemplos	Observaciones

UI / UX	`@gorhom/portal`, `react-native-tab-view`, `poppins`, `expo-checkbox`	Buen uso de librerías modernas
Estado	`redux-toolkit`, `react-query`	Excelente combinación (escalabilidad + optimización)
Autenticación / Seguridad	`expo-auth-session`, `expo-secure-store`, `expo-local-authentication`	Cubren login biométrico, tokens seguros
Observabilidad / Flags	`sentry`, `unleash`, `growthbook`	Nivel empresarial

Documentación

Archivo	Estado
`README.md`	No visible en estructura
`app.config.js`	Configuración específica de Expo
Otros	No se detecta documentación para desarrolladores o entorno de producción/despliegue

Análisis Técnico del Repositorio: Carpeta Ciudadana Backend

Información General

Nombre del Proyecto: Carpeta Ciudadana Backend

Repositorio: github.com/ogticrd/carpeta-ciudadana-backend

El backend centraliza la lógica de notificaciones y cuenta con un componente llamado "wrapper" que agrupa a las instituciones que funcionan de forma similar, como los ayuntamientos, las empresas distribuidoras de electricidad (EDES), y las compañías de agua. Esto permite manejar todas estas entidades de manera unificada, evitando realizar múltiples peticiones separadas para cada una y optimizando así la comunicación y el rendimiento del sistema.

Elemento	Descripción
Arquitectura	Microservicios
Framework principal	NestJS
Lenguaje	TypeScript
Comunicación entre servicios	gRPC
Contenedores	Docker, `docker-compose`, múltiples Dockerfiles
Gestión de paquetes	npm/yarn
Infraestructura como código	`docker-compose.yml`, múltiples archivos de entorno (`.env.example`)

Análisis de Microservicios

Microservicio	Funcionalidad
gateway	Punto de entrada, autenticación, health check, orquestación general

Microservicio	Funcionalidad
metadata	Gestión de usuarios, sesiones, OTPs, suscripciones de notificación
notification	Envío y manejo de notificaciones, workflows
wrapper-institutions	Integración con instituciones (ej. JCE, Edesur, pasaportes)

DevOps / Automatización

Elemento	Descripción
CI/CD	Workflows por microservicio (`gateway-deployment.yml`, etc.)
Limpieza de caché	`cleanup-cache-branch.yml` para mantener el entorno limpio
Dependabot	Actualización automática de dependencias (`dependabot.yml`)
Entornos definidos	`.env.example`, `development.Dockerfile`, múltiples `Dockerfile`
Linting y Formateo	`.eslintrc.js`, `.prettierrc`, uso de Biome
Debugging	Archivos `.vscode/launch.json` en cada microservicio

Análisis de Workflows CI/CD (`.github/workflows/`)

Archivo	Descripción
cleanup-cache-branch.yml	**Limpia las cachés asociadas a una Pull Request (PR)** cuando esta es cerrada.
gateway-deployment.yml	Este archivo parece estar relacionado con el **despliegue del Gateway**.
metadata-deployment.yml	Este archivo probablemente gestiona el **despliegue del microservicio Metadata**. La última actualización sugiere que se corrigió alguna cuestión relacionada con el entorno de producción.
notification-deployment.yml	Relacionado con el **despliegue del microservicio Notification**. La última actualización menciona la integración o actualización con **Novu** (probablemente una plataforma de notificaciones).
wrapper-institutions-deployment.yml	Este archivo gestiona el despliegue de **Wrapper Institutions**, que probablemente interactúa con servicios externos o APIs de instituciones gubernamentales.

Control de Calidad

Elemento	Detalles
Framework de testing	`jest`
Tipo de pruebas	`e2e` (end-to-end), ejemplos: `app.e2e-spec.ts`
Cobertura	Se encuentran `*.spec.ts` para pruebas unitarias (ej: `*.service.spec.ts`)

Seguridad

Elemento	Descripción
API Key Middleware	`auth/api-key.middleware.ts` (wrapper-institutions)
Manejo de tokens OTP	Varios microservicios trabajan con OTPs y suscripciones seguras
Entornos seguros	`.env.example` usado en todos los servicios
Dependabot	Protege contra vulnerabilidades de paquetes
Sin SonarQube	No se encontró integración con SonarQube

Herramientas y Librerías Detectadas

Tipo	Herramientas / Librerías
Backend Framework	NestJS
Contenedores	Docker, Docker Compose
Proto/gRPC	Archivos `.proto` para cada servicio
Lint/Formato	ESLint, Prettier, Biome
Tests	Jest
Gestión de paquetes	Yarn o npm
VSCode Dev Env	`.vscode` en todos los servicios

Documentación

Sección	Contenido Detectado

Introducción	Describe brevemente el propósito del backend: “simplificar la gestión de documentos y trámites ciudadanos”.
Arquitectura	Enumera los microservicios (gateway, metadata, notification, wrapper-institutions) y explica sus responsabilidades.
Tecnologías	Menciona NestJS, TypeScript y gRPC como stack principal.

Análisis Técnico del Repositorio: Carpeta Ciudadana Landing

Información General

Nombre del Proyecto: Carpeta Ciudadana Landing

Repositorio: github.com/ogticrd/carpeta-ciudadana-landing

Este repositorio parece estar orientado a crear una aplicación de **landing page** con funcionalidad para múltiples secciones. Está basado principalmente en **Vue.js** y utiliza **Quasar Framework** para facilitar el desarrollo y el diseño.

Carpeta/Archivo	Descripción
public/	Archivos estáticos como iconos, imágenes y recursos multimedia.
src/	Código fuente de la aplicación.
assets/	Recursos estáticos utilizados en la app, como imágenes.
components/	Componentes reutilizables de la interfaz de usuario.
layouts/	Plantillas (layouts) utilizadas en las páginas.
pages/	Páginas del sitio web (e.g., `FAQsPage.vue`, `IndexPage.vue`).
router/	Configuración de rutas de la aplicación.
stores/	Gestión del estado global usando **Pinia**.

Tecnología

Elemento	Descripción
Framework	Quasar (Vue.js 3) para el desarrollo de la aplicación.
SSR (Server-Side Rendering)	Configurado para SSR con Quasar, mejorando rendimiento y SEO.

Elemento	Descripción
Vite	Utiliza Vite como bundler para optimizar el desarrollo y la construcción.
Gestión de Estado	Usa **Pinia** para gestionar el estado global de la aplicación.
Ruteo	Usando **Vue Router** para gestionar la navegación.

Dependencias

Dependencia	Descripción
Dependencias principales	
`vue`, `vue-router`	Framework y librería de enrutamiento para Vue.js.
`quasar`	Framework para crear aplicaciones con diseño consistente y responsivo.
`pinia`	Gestión de estado para Vue 3 (alternativa a Vuex).
`axios`	Para realizar solicitudes HTTP.
Dependencias de desarrollo	
`eslint`, `prettier`	Herramientas de linting y formateo de código.
`@intlify/unplugin-vue-i18n`	Plugin para la internacionalización en Vue.js.
`vite-plugin-checker`	Plugin para verificaciones de tipos y linting.

Internacionalización (i18n)

Elemento	Descripción
i18n	Configurado para soportar múltiples idiomas.
Archivos de idioma	Se encuentra la configuración de idioma para **es-DO**.
vue-i18n	Utiliza el plugin **vue-i18n** para internacionalización.

Scripts en `package.json`

Script	Descripción

`dev`	Inicia la aplicación en modo de desarrollo con SSR.
`build`	Construye la aplicación para producción con un dominio específico.
`host`	Despliega la aplicación en **Firebase Hosting** después de construirla.
`lint`	Ejecuta ESLint en los archivos `.js` y `.vue` del proyecto.
`format`	Ejecuta Prettier para formatear el código según las configuraciones.

Control de Calidad

Elemento	Descripción
Pruebas	No se configuran pruebas unitarias ni de integración.

DevOps / Automatización

Elemento	Descripción
Integración Continua	No se mencionan explícitamente flujos de integración continua (CI). Sin embargo, la presencia de configuraciones de despliegue y scripts como `build` y `host` sugieren un enfoque manual para integración y despliegue.
Despliegue Continuo	El despliegue se realiza utilizando el script `host`, que ejecuta el despliegue a **Firebase Hosting**. Sin embargo, no hay un flujo automático o un archivo específico para gestionar el despliegue continuo (CD).
Faltan archivos de CI/CD	No hay archivos de configuración específicos para plataformas de CI/CD como GitHub Actions.
Automatización de Despliegue	El script `host` despliega manualmente la aplicación al entorno de Firebase.

Análisis Técnico del Repositorio: totp-cc

Información General

Nombre del Proyecto: totp-cc

Repositorio: github.com/ogticrd/totp-cc

Este repositorio es una aplicación escrita en **Go (Golang)**, que parece estar diseñada para generar y validar **TOTP (Time-based One-Time Passwords)** para ciudadanos como parte de un sistema de autenticación o verificación de identidad.

Carpeta/Archivo	Descripción
`config/`	Contiene lógica de configuración general (`config.go`) y sus pruebas. Posiblemente gestiona valores de `.env` y configuración de entornos.
`database/`	Lógica de conexión a base de datos, específicamente MongoDB. Incluye pruebas (`mongodb_test.go`).
`docs/`	Archivos Swagger (`swagger.json`, `swagger.yaml`) para la documentación de API, y código Go relacionado (`docs.go`).
`internal/handlers/`	Manejadores HTTP para generación y validación de TOTP (`generate.go`, `validate.go`).
`internal/routes/`	Define las rutas HTTP para los endpoints de generación y validación.
`internal/models/`	Estructuras de datos y lógica del dominio como `citizentotp.go`, validación y estructuras de respuesta.
`pkg/cipher/`	Lógica criptográfica de cifrado AES-128 y pruebas unitarias.
`pkg/totp/`	Implementación central del algoritmo TOTP, con pruebas.
`pkg/utils/`	Funciones auxiliares o utilitarias generales.
`router/`	Define el router principal de la aplicación (`router.go`), usando probablemente un framework como Gin o Echo.
`.env.example`	Archivo de ejemplo de variables de entorno.

Carpeta/Archivo	Descripción
`Dockerfile`	Para construir la imagen Docker de la aplicación.
`docker-compose.dev.yaml`	Compose para entorno de desarrollo. Posiblemente incluye MongoDB.
`Makefile`	Comandos automáticos para tareas comunes como build, test o lint.
`main.go`	Punto de entrada de la aplicación. Suele iniciar servidor, cargar configuración, rutas, etc.

Control de Calidad

Tipo de prueba	Implementado	Observaciones
Unitarias (`*_test.go`)	Sí	Presentes en `pkg/`, `config/`, `database/`.
Integración	No	No se observan pruebas de flujo completo.
Cobertura con herramientas	No	No se detectó integración con coverage tools.

Seguridad

Elemento	Estado / Recomendación
Manejo de secretos	Usa `.env.example`. Se recomienda revisar claves críticas.
Cifrado	AES-128 implementado correctamente.
Variables en entorno	Bien gestionadas, pero falta validación de existencia obligatoria.

Resumen de Arquitectura

Componente	Implementación
Tipo de Aplicación	Microservicio en Go para generación/validación de TOTP.
Almacenamiento	MongoDB.
API	RESTful con documentación Swagger.
Autenticación / Seguridad	Cifrado + TOTP (MFA o validación por usuario).

Componente	Implementación
Despliegue Local	Docker Compose (`.dev.yaml`).
Entrada Principal	`main.go` arranca router + dependencias.

DevOps / Automatización

Elemento	Estado / Observación
GitHub Actions / CI/CD	No se encontró `.github/workflows/` ni ningún pipeline CI/CD configurado.
Docker	Dockerfile para una app en go.
Docker Compose	Existe un archivo para entorno de desarrollo.
Despliegue automatizado	No se identificó ningún script de despliegue automático.

Tecnología

Categoría	Tecnología / Herramienta
Lenguaje	Go (Golang)
Framework web	Fiber
Contenedor	Docker (multi-stage)
Base de imagen	`golang:1.22-alpine`, `scratch`
Formato de build	Go Modules (`go.mod`)

Análisis Técnico del Repositorio: cc-institutions-mock-api

Información General

Nombre del Proyecto: cc-institutions-mock-api

Repositorio: github.com/ogticrd/cc-institutions-mock-api

API Mockup para proveer data fake de las Instituciones para la App Carpeta Ciudadana.

Carpeta / Archivo	Descripción breve
`.github/workflows/`	Contiene workflows de GitHub Actions para CI/CD (clean, development, staging).
`cmd/seeders`	Probablemente incluye scripts para sembrar datos en la base de datos.
`config/config.go`	Archivo de configuración global de la app (env vars, puertos, etc).
`database/mongodb.go`	Inicialización y conexión a MongoDB.
`docs/`	Contiene documentación Swagger (`.json`, `.yaml`) y archivo auxiliar.
`internals/`	Lógica principal dividida en submódulos según entidad o institución.
`models/`	Definiciones de modelos de datos usados por las APIs y la base de datos.
`router/router.go`	Inicializa las rutas HTTP del servidor.
`utils/utils.go`	Funciones auxiliares reutilizables (helpers).
`.dockerignore`	Archivos que se deben excluir del contexto de Docker.
`.env.example`	Variables de entorno de ejemplo para configurar localmente.
`Dockerfile`	Imagen Docker para compilar y ejecutar el backend.
`Makefile`	Tareas automatizadas (build, lint, run, etc).
`README.md`	Documentación inicial del proyecto.

Carpeta / Archivo	Descripción breve
`go.mod / go.sum`	Manejadores de dependencias de Go.
`main.go`	Punto de entrada principal de la aplicación Go.

Módulos por dominio (internals/handlers y routes)

Institución / Módulo	Handler (`handlers/`)	Rutas (`routes/`)	Modelo (`models/`)	Funcionalidad esperada
Educación	`educacion.go`	`educacion.go`	`educacion.go`	Consultas al Ministerio de Educación
Intrant	`intrant.go`	`intrant.go`	`intrant.go`	Licencias, tránsito y vehículos
JCE	`jce.go`	`jce.go`	`jce.go`	Datos de cédula y registro civil
MESCyT	`mescyt.go`	`mescyt.go`	`educacion.go`	Becas y títulos universitarios
Migración	`migracion.go`	`migration.go`	`migracion.go`	Entrada/salida del país
SIPEN	`sipen.go`	`sipen.go`	`pension.go`	Pensiones, AFP
SISALRIL	`sisalril.go`	`sisalril.go`	`sisalril.go`	Seguro familiar de salud
SNS	`sns.go`	`sns.go`	`sns.go`	Centros de salud, afiliación
Superseguros	`superseguros.go`	`superseguros.go`	`citizen.go` (?)	Información de seguros

Documentación y APIs

Archivo	Propósito
`docs/swagger.yaml`	Definición OpenAPI de las rutas y modelos
`docs.go`	Generación o integración de Swagger posiblemente con comentarios Go
`README.md`	Introducción al proyecto y guía de uso

Tecnología

Elemento	Descripción
Tecnología Backend	Go (Golang)
Framework	Ninguno especificado; se utiliza Go nativo para implementar la lógica del backend.
Base de Datos	MongoDB
Archivos de Configuración	Archivos como `config.go` y `database.go` sugieren la configuración de la base de datos y otros parámetros del sistema.
Entorno de Desarrollo	El repositorio contiene un `Makefile`, `Dockerfile`, y archivos de configuración como `.env.example` que permiten configurar y ejecutar el entorno de desarrollo.
Contenedores	El repositorio contiene un `Dockerfile` que permite construir una imagen Docker para el backend.
CI/CD	El repositorio incluye workflows de CI/CD en `.github/workflows/`, como `clean.yml`, `development.yml`, `staging.yml`, que gestionan despliegues automáticos.
Dependencias	`go.mod` y `go.sum` indican que el proyecto usa Go Modules para gestionar las dependencias del proyecto.
Servicios	El backend parece estar dividido en varios servicios, como `citizenapi`, con distintos manejadores en `handlers` que se encargan de diferentes aspectos del servicio.
Documentación	Se incluyen archivos de documentación como `swagger.json` y `swagger.yaml`, que pueden describir las APIs del servicio.

DevOps / Automatización

Área	Herramienta/Archivo	Comentarios
CI/CD	`.github/workflows` (presente)	GitHub Actions configurado (ej. limpieza de caché, tests)
Contenedores	`Dockerfile`	Contenedor de pruebas configurado
Seguridad	`dependabot.yml`	Monitoreo automático de versiones y alertas

Análisis de Workflows CI/CD (`.github/workflows/`)

Archivo	Descripción
development.yml	**Desplegar entorno de desarrollo en Cloud Run**
staging.yml	Desplegar a entorno de preproducción (staging)
clean.yml	Eliminar recursos temporales de PR cerrados