Cuenta Única Ciudadana: segunda versión

Consolidación de información para la segunda versión del proyecto de Cuenta Única Ciudadana

- Temas de debate
- Seguridad y cumplimiento
- Roadmap del proyecto
- Propuesta de valor

Temas de debate

Propósito de este documento

En todo proyecto ambicioso hay decisiones que se toman con confianza y otras que se asumen sin discutir lo suficiente. Este documento recoge los temas que necesitan ser debatidos abiertamente por el equipo antes de avanzar más. No son críticas ni señalamientos: son puntos donde la decisión correcta no es obvia y donde diferentes perspectivas pueden llevar a mejores resultados.

Cada tema se presenta con el contexto necesario, las opciones disponibles, las implicaciones de cada una y las preguntas que el equipo debe responder. La idea es que este documento sirva como agenda para sesiones de discusión técnica, no como un dictamen.

1. Infraestructura multi-cloud: ¿pragmatismo o complejidad innecesaria?

Actualmente el portal ciudadano está en Google Cloud Platform (Cloud Run) y la infraestructura de eventos y el backoffice están en AWS (EventBridge, Lambda, DynamoDB). Esta decisión tiene sentido técnico: Cloud Run es excelente para aplicaciones web con tráfico variable y EventBridge es un servicio sin equivalente directo en GCP para este caso de uso.

Sin embargo, operar en dos proveedores cloud tiene implicaciones que vale la pena discutir. El equipo necesita mantener expertise en ambas plataformas. Los costos se dividen en dos facturas diferentes, lo que dificulta el seguimiento presupuestario. La comunicación entre los componentes (por ejemplo, el portal enviando eventos a EventBridge) cruza proveedores, lo que agrega latencia y puntos de fallo. Y si algún día se necesita un ambiente de disaster recovery, hay que replicarlo en dos clouds.

La alternativa sería consolidar todo en un solo proveedor. AWS tendría la ventaja de que ya aloja la parte más compleja (eventos, Lambda, DynamoDB) y el backoffice. El portal podría desplegarse en AWS Amplify Hosting, ECS Fargate, o incluso App Runner, que es el equivalente de AWS a Cloud Run. La desventaja es que Cloud Run ya funciona y migrar tiene un costo de tiempo.

Preguntas para el equipo: ¿Hemos medido la latencia adicional que introduce la comunicación cross-cloud? ¿Cuál es el costo operativo real de mantener dos proveedores (no solo en dinero, sino en tiempo del equipo)? ¿Hay un punto en el futuro donde la complejidad multi-cloud nos va a morder, por ejemplo cuando necesitemos un VPC privado para comunicar servicios? ¿O es que la separación actual es lo suficientemente limpia como para no ser un problema en la práctica?

2. El agente de IA: ¿qué problema resuelve exactamente?

Se ha mencionado que CUC v2 tendrá un agente de inteligencia artificial para ayudar a los ciudadanos. La tecnología está en investigación, lo cual es correcto, pero antes de elegir la tecnología, el equipo necesita alinear lo que espera de este agente.

Hay una diferencia grande entre un chatbot con respuestas predefinidas (que puede resolver el 80% de las preguntas frecuentes con muy poco riesgo), un agente conversacional basado en un modelo de lenguaje grande (que puede manejar preguntas abiertas pero puede generar respuestas incorrectas o confusas), y un sistema híbrido que usa IA para entender la intención del ciudadano y luego ejecuta acciones predefinidas (más seguro que un LLM puro pero más flexible que un chatbot).

En un contexto gubernamental, la precisión de las respuestas no es negociable. Si un ciudadano pregunta "¿puedo usar mi cuenta para acceder al portal de becas?" y el agente dice "sí" cuando la respuesta real es "depende de si el portal de becas ya está integrado", eso genera desconfianza. Los LLMs son propensos a este tipo de errores (conocidos como "alucinaciones" en la literatura técnica), y en un contexto donde la información gubernamental debe ser exacta, esto es un riesgo real.

También está el tema de la escalación a un agente humano. ¿Cómo se determina que el agente de IA no puede resolver el caso? ¿Después de cuántos intentos? ¿Según el tipo de pregunta? ¿El ciudadano puede pedir hablar con un humano directamente? ¿Qué pasa fuera del horario laboral cuando no hay agentes humanos disponibles?

Preguntas para el equipo: ¿Cuáles son los 20 casos de uso más comunes que el agente debe resolver? ¿Podemos empezar con algo más simple (un FAQ interactivo bien hecho) y evolucionar hacia IA cuando tengamos datos de qué preguntan realmente los ciudadanos? ¿Cuál es el presupuesto operativo para un servicio de IA (los modelos de lenguaje se cobran por uso y el costo puede escalar rápido)? ¿Quién se responsabiliza si el agente da información incorrecta a un ciudadano?

3. El buzón de notificaciones: ¿canal de comunicación o bandeja de spam gubernamental?

El buzón ciudadano es una funcionalidad con mucho potencial pero también con riesgo de mal uso. Si cada institución gubernamental puede enviar notificaciones a los ciudadanos a través de CUC, existe el riesgo de que el buzón se llene de mensajes irrelevantes y los ciudadanos dejen de prestarle atención, lo que anularía completamente su utilidad.

Hay varias preguntas de diseño que necesitan respuesta. ¿Quién puede enviar notificaciones? ¿Solo los agentes de CUC o cualquier institución integrada? ¿Hay un proceso de aprobación para las notificaciones masivas? ¿El ciudadano puede configurar qué tipo de notificaciones quiere recibir? ¿Hay límites de frecuencia?

También está el tema técnico. ¿Las notificaciones son en tiempo real (WebSockets, Server-Sent Events) o se consultan cada vez que el ciudadano abre su cuenta (polling)? La primera opción es mejor para la experiencia pero más compleja de implementar y mantener. La segunda es más simple pero menos inmediata.

Y un tema que nadie ha mencionado aún: ¿las notificaciones se envían también por correo electrónico o solo se muestran dentro de la plataforma? Si solo están dentro de la plataforma, el ciudadano no se entera hasta que inicia sesión. Si se envían por correo, ¿eso no duplica un canal que ya existe?

Preguntas para el equipo: ¿Cuál es el modelo de gobernanza de las notificaciones? ¿Hemos hablado con las instituciones que integran CUC para entender qué tipo de comunicaciones querrían enviar? ¿El buzón es un MVP que empieza solo con notificaciones del sistema (seguridad, actividad de cuenta) y después se abre a otras instituciones? ¿O lanzamos con la capacidad completa desde el inicio?

4. Open source: ¿estamos listos?

La decisión de hacer CUC v2 open source es excelente desde el punto de vista de transparencia y potencial de colaboración. Sin embargo, open source no es solo publicar el código en GitHub. Es un compromiso continuo que tiene implicaciones operativas.

Un proyecto open source gubernamental necesita, como mínimo: documentación clara de cómo contribuir (CONTRIBUTING.md), un código de conducta (CODE_OF_CONDUCT.md), un proceso de revisión de pull requests que sea oportuno (si alguien contribuye y no recibe respuesta en

semanas, no va a volver), un proceso de reporte y manejo de vulnerabilidades de seguridad (SECURITY.md), y una licencia de software explícita que defina qué pueden hacer otros con el código.

También hay que pensar en qué partes del código son open source y cuáles no. El portal ciudadano puede ser completamente abierto. Pero el backoffice, que gestiona datos de ciudadanos y tiene acceso administrativo a las cuentas, ¿también debería ser público? ¿No facilita eso que alguien estudie el sistema para encontrar vectores de ataque?

Hay precedentes internacionales que vale la pena revisar. El servicio login.gov del gobierno de Estados Unidos es open source (github.com/18F/identity-idp) y ha navegado estos temas con éxito. El Gobierno del Reino Unido publica código a través de su Government Digital Service (GDS) en github.com/alphagov. Estonia, referente mundial en gobierno digital, también tiene componentes open source de su infraestructura X-Road. Estos casos demuestran que es posible, pero también que requiere un compromiso institucional sostenido.

Preguntas para el equipo: ¿Tenemos el ancho de banda para atender contribuciones externas, o el código va a estar público pero en la práctica nadie va a revisar los pull requests? ¿Hemos definido la licencia? ¿MIT, Apache 2.0, algo más restrictivo? ¿El backoffice será open source o solo el portal ciudadano? ¿Hay una política de OGTIC sobre publicación de código fuente que debamos seguir?

5. Verificación facial: accesibilidad y casos límite

La verificación facial con AWS Rekognition es un componente central del registro, pero tiene limitaciones que debemos reconocer y planificar.

El primer tema es la accesibilidad. Un ciudadano que tiene una discapacidad visual severa puede tener dificultades con la prueba de liveness, que requiere seguir instrucciones visuales en pantalla. ¿Cuál es la alternativa para estos ciudadanos? ¿Un proceso de verificación presencial? ¿Una llamada de video con un agente? Si no hay alternativa, estamos excluyendo a un segmento de la población del acceso a servicios digitales.

El segundo tema son los dispositivos de gama baja. Los umbrales actuales son 90% para liveness y 80% para similitud facial. ¿Estos umbrales funcionan bien con cámaras de baja resolución? ¿Se han probado con los dispositivos más comunes en República Dominicana? Un umbral muy alto puede rechazar a ciudadanos legítimos; un umbral muy bajo puede aceptar intentos fraudulentos. Encontrar el balance correcto requiere datos que probablemente solo obtengamos en producción.

El tercer tema es la dependencia de la foto de la JCE. La comparación facial depende de la foto almacenada en la base de datos de la Junta Central Electoral. ¿Qué tan recientes son estas fotos?

Si un ciudadano renovó su cédula hace 10 años y su apariencia cambió significativamente, la comparación puede fallar legítimamente. ¿Cuál es el proceso para estos casos?

Y el cuarto tema, más técnico: la dependencia de AWS. Si AWS Rekognition tiene una interrupción del servicio (ha pasado), el registro de CUC se detiene completamente. ¿Tenemos un plan de contingencia? ¿Se puede permitir un registro temporal sin verificación facial que se complete después?

Preguntas para el equipo: ¿Hemos hecho pruebas de usabilidad de la verificación facial con ciudadanos reales, especialmente adultos mayores y personas con dispositivos básicos? ¿Cuál es el plan para ciudadanos que no pueden pasar la verificación facial por razones legítimas? ¿Tenemos métricas de la v1 sobre la tasa de éxito/fracaso de la verificación facial? ¿Deberíamos tener un fallback manual para cuando Rekognition no esté disponible?

6. Monitoreo y observabilidad: ¿Sentry, CloudWatch o algo más?

La versión 1 usaba Sentry para captura de errores. La versión 2 tiene una infraestructura de eventos en AWS que cubre parte del monitoreo. Pero no se ha tomado una decisión clara sobre la estrategia completa de observabilidad.

Un sistema de identidad gubernamental necesita, como mínimo: monitoreo de errores de aplicación (el equivalente a lo que hacía Sentry), monitoreo de rendimiento (tiempos de respuesta de cada endpoint, tiempos de carga de cada página), monitoreo de infraestructura (uso de CPU, memoria, conexiones de red en Cloud Run, ejecuciones y errores en Lambda), alertas automáticas (cuando algo falle, el equipo debe enterarse antes que los ciudadanos), y dashboards operativos (una vista en tiempo real del estado del sistema).

Las opciones no son mutuamente excluyentes. Se podría usar Sentry para errores de aplicación (es excelente en esto y tiene contexto que CloudWatch no tiene, como el stack trace completo con variables locales), CloudWatch para la infraestructura de AWS, Cloud Monitoring para Cloud Run, y algo como Grafana para unificar dashboards. Pero eso son cuatro herramientas diferentes, lo que fragmenta la experiencia del equipo de operaciones.

Una alternativa sería usar una plataforma unificada como Datadog o New Relic que cubra todas las necesidades. Pero estas tienen un costo significativo que hay que evaluar.

Preguntas para el equipo: ¿Cuál fue la experiencia con Sentry en la v1? ¿Realmente se usaba o se instaló y nadie miraba las alertas? ¿Tenemos presupuesto para una herramienta de observabilidad comercial? ¿Quién va a ser responsable de mirar los dashboards y responder a las alertas? Porque la mejor herramienta del mundo no sirve si nadie la mira.

7. Protección contra bots y abuso: ¿qué reemplaza a reCAPTCHA?

La versión 1 usaba Google reCAPTCHA v3 para prevenir el registro automatizado por bots y Cloudflare para protección DDoS. En el código de la versión 2 no se ve ninguna implementación de CAPTCHA ni de protección similar.

ORY Network tiene protección contra fuerza bruta (rate limiting por IP y por cuenta), pero eso solo protege el login. El registro, que involucra llamadas a la API del Registro Civil y a AWS Rekognition (ambas con costo), necesita protección propia contra abuso.

Las opciones incluyen: mantener reCAPTCHA (funciona, pero Google recolecta datos del usuario, lo cual puede ser un problema de privacidad para un servicio gubernamental), usar Cloudflare Turnstile (alternativa a reCAPTCHA que es más respetuosa con la privacidad), implementar rate limiting propio en las API routes (más simple pero menos sofisticado), o confiar en la protección a nivel de Cloudflare/WAF si se mantiene Cloudflare como proxy.

Preguntas para el equipo: ¿Vamos a seguir usando Cloudflare como proxy/WAF? ¿Hay alguna política de OGTIC sobre el uso de servicios que recolectan datos de usuarios (como reCAPTCHA)? ¿Cuál es el costo estimado si un bot hace miles de llamadas a la API del Registro Civil o a Rekognition antes de que lo detectemos?

8. Pruebas de carga y límites del sistema

Antes de lanzar la v2, necesitamos entender los límites del sistema. ¿Cuántos registros simultáneos puede manejar? ¿Cuántos logins por segundo? ¿Qué pasa si hay un pico de tráfico (por ejemplo, porque el gobierno anuncia un nuevo servicio que requiere CUC)?

Cloud Run escala automáticamente, pero tiene límites configurables (máximo de instancias, máximo de conexiones por instancia). ORY Network tiene sus propios límites según el plan contratado. AWS Rekognition tiene límites de API por región. Lambda tiene límites de concurrencia. DynamoDB tiene límites de capacidad de lectura/escritura.

Cada uno de estos límites puede convertirse en un cuello de botella si no se dimensiona correctamente. Y en un servicio gubernamental, la caída del sistema durante un pico de demanda tiene implicaciones políticas además de técnicas.

Preguntas para el equipo: ¿Hemos hecho pruebas de carga? ¿Conocemos los límites del plan de ORY Network que estamos usando? ¿Cuál es el pico de tráfico más alto que ha tenido CUC v1? ¿Tenemos un plan de capacidad que defina cuántos usuarios simultáneos debemos soportar?

9. Migración de usuarios de v1 a v2

Un tema que no se ha discutido explícitamente: ¿cómo se migran los usuarios existentes de CUC v1 a CUC v2? Si ambas versiones usan ORY Network, las identidades deberían ser las mismas. Pero si hay cambios en los traits de la identidad (campos nuevos, estructura diferente), se necesita una estrategia de migración.

¿Se van a migrar todos los usuarios de golpe? ¿Se va a hacer una migración gradual donde algunos usuarios usan v1 y otros v2? ¿Los usuarios necesitan hacer algo (como iniciar sesión en el nuevo portal) para completar la migración, o es transparente?

Y si el dominio cambia (de cuentaunica.gob.do a algo diferente), ¿cómo se manejan las integraciones existentes que apuntan al dominio actual? ¿Hay un período de transición con ambos dominios activos?

Preguntas para el equipo: ¿Los traits de la identidad en ORY cambian entre v1 y v2? ¿Las integraciones OAuth existentes (gob.do, SoyYo RD, becas) necesitan actualizar algo de su lado? ¿Cuál es el plan de transición para que el lanzamiento no interrumpa los servicios que dependen de CUC?

Próximos pasos

Se recomienda que el equipo tome este documento como agenda para una o varias sesiones de trabajo donde se discutan estos temas. Para cada tema, el resultado debería ser una decisión documentada con su justificación, o un plan de investigación con fecha de cierre si la información actual no es suficiente para decidir.

Los temas no tienen todos la misma urgencia. Los que bloquean el progreso del desarrollo (como la decisión de monitoreo y la protección contra bots) deberían discutirse primero. Los que afectan el lanzamiento pero no el desarrollo (como la estrategia open source y la migración de usuarios) pueden esperar un poco más, pero no indefinidamente.

Este documento debe actualizarse después de cada sesión de debate, registrando las decisiones tomadas y eliminando los temas resueltos.

Seguridad y cumplimiento

Roadmap del proyecto

1. Portal Ciudadano

Esta es la aplicación principal que usan los ciudadanos. Es un proyecto en Next.js 16 con App Router, integrado con ORY Network para la gestión de identidades y AWS Rekognition para la verificación biométrica.

Lo que ya se hizo

El esqueleto de la aplicación está construido y funcional en ambiente de desarrollo. Se tomaron decisiones fundamentales de arquitectura que definen el rumbo del proyecto:

Se eligió Next.js 16 como framework principal, aprovechando el App Router y los React Server Components para tener una separación clara entre lo que se ejecuta en el servidor y lo que llega al navegador del ciudadano. Esta decisión se tomó porque Next.js permite renderizar las páginas en el servidor, lo que mejora tanto el rendimiento como el SEO, algo importante para un portal gubernamental que debe ser accesible desde cualquier dispositivo y conexión.

La integración con ORY Network está funcionando. Se implementaron los flujos de inicio de sesión, recuperación de contraseña, verificación de email y gestión de configuraciones del usuario, todos delegados a los componentes oficiales de ORY Elements React. Esto significa que no estamos reinventando la rueda en temas de seguridad: ORY maneja el almacenamiento de credenciales, la emisión de tokens OAuth 2.0 y los flujos de OpenID Connect.

El flujo de registro de tres pasos ya está implementado en su forma base. El primer paso valida la cédula del ciudadano contra la API del Registro Civil dominicano. El segundo paso recoge el correo electrónico y la contraseña. El tercer paso realiza la verificación facial usando AWS Rekognition Face Liveness, que compara el rostro en vivo del ciudadano con la foto almacenada en la base de datos de la Junta Central Electoral. Este cambio de orden respecto a la versión anterior (donde la verificación facial era el segundo paso) fue intencional: permite que el ciudadano complete la información de su cuenta antes de pasar por el proceso biométrico, reduciendo la frustración en caso de que necesite corregir datos.

Se configuró la internacionalización con next-intl, soportando español e inglés. Todas las cadenas de texto están externalizadas en archivos de traducción, lo que facilita agregar nuevos idiomas en el futuro.

El sistema de componentes UI está basado en shadcn/ui con Radix UI y Tailwind CSS. Esto nos da componentes accesibles por defecto (cumplen con ARIA) y un sistema de diseño consistente. Se implementó soporte para modo oscuro usando next-themes.

La validación de formularios usa Zod para esquemas de validación y React Hook Form para la gestión del estado de los formularios. Esto asegura que los datos se validen tanto en el cliente como en el servidor.

Se crearon las rutas protegidas del dashboard ciudadano: página principal, perfil, configuraciones, historial, soporte y acerca de. Aunque varias de estas páginas aún son esqueletos, la estructura de navegación y el sistema de protección de rutas (que verifica la sesión de ORY antes de permitir el acceso) están completos.

El pipeline de CI/CD está configurado con GitHub Actions para desplegar automáticamente a Google Cloud Run cuando se hace push a la rama staging. El Dockerfile usa una imagen base de Bun para builds rápidos y produce una imagen optimizada con el output standalone de Next.js.

Las API routes del servidor están implementadas para todo el flujo de registro: búsqueda de ciudadano por cédula, creación de cuenta, creación de sesión de liveness, verificación de resultado de liveness, y reset de sesión de registro. También están las rutas para gestión de sesiones de ORY (obtener sesión actual, revocar sesiones, logout).

Lo que está en progreso

El diseño visual gubernamental está siendo refinado. Aunque la estructura está lista, se necesita pulir la experiencia visual para que se sienta institucional pero moderna. Esto incluye el landing page, las transiciones entre pasos del registro y la consistencia visual en todas las pantallas.

Se está trabajando en la gestión de errores. El código tiene manejo básico de errores, pero falta una estrategia unificada que cubra todos los escenarios: errores de red, timeouts de ORY, fallos de Rekognition, problemas con la cámara del dispositivo, etc. Cada error debe mostrar un mensaje claro al ciudadano y, al mismo tiempo, registrarse internamente para que el equipo de soporte pueda diagnosticarlo.

Lo que falta por hacer

Funcionalidades del Dashboard Ciudadano. Las páginas del dashboard están creadas como esqueletos pero necesitan contenido real. La página de perfil debe mostrar la información del ciudadano extraída de los traits de ORY (nombre, cédula, fecha de nacimiento, género). La página de historial debe mostrar las sesiones activas y el historial de actividad. La página de configuraciones necesita permitir el cambio de contraseña, la activación de segundo factor de autenticación (2FA) y la gestión de preferencias de notificación.

Sistema de Notificaciones (Buzón Ciudadano). Esta es una funcionalidad completamente nueva que no existía en CUC v1. La idea es que los ciudadanos puedan recibir notificaciones dentro de su cuenta: avisos de seguridad, comunicaciones gubernamentales, alertas de actividad sospechosa, recordatorios de trámites pendientes, entre otros. Se necesita definir la arquitectura del sistema de notificaciones (almacenamiento, entrega en tiempo real vs polling, tipos de notificación), diseñar la interfaz del buzón, implementar la API de notificaciones y conectarla con el backoffice para que los agentes puedan enviar notificaciones.

Agente de Inteligencia Artificial. Se planea incorporar un agente conversacional que ayude a los ciudadanos a resolver dudas sobre su cuenta, guiarlos en procesos y, cuando no pueda resolver un problema, escalar a un agente humano de Atención Ciudadana. La tecnología aún está en evaluación. Se debe definir qué modelo de lenguaje usar, cómo integrarlo en la interfaz, qué conocimiento base necesita, cómo manejar la escalación a humanos y cómo garantizar que las respuestas sean precisas y no generen confusión en temas gubernamentales.

Mejoras en la Verificación Facial. El flujo básico funciona, pero hay escenarios que necesitan mejor manejo: qué pasa cuando el ciudadano no tiene cámara, cuando la iluminación es mala, cuando el navegador no soporta la API de MediaDevices, cuando el rostro no coincide pero el ciudadano insiste en que es él. Se necesita una estrategia de fallback y un proceso claro de escalación para estos casos.

Gestión de Dispositivos y Sesiones. ORY maneja las sesiones, pero el ciudadano necesita poder ver en cuántos dispositivos tiene sesión activa y poder cerrar sesiones remotamente. Esto ya se ve en el código como un endpoint de revocación de sesión, pero la interfaz para que el ciudadano lo use aún no está construida.

Accesibilidad. Aunque shadcn/ui y Radix UI proporcionan componentes accesibles por defecto, se necesita una auditoría completa de accesibilidad (WCAG 2.1 AA como mínimo) en todos los flujos. Un portal gubernamental debe ser usable por personas con discapacidades visuales, motoras o cognitivas.

Pruebas Automatizadas. No hay tests en el proyecto actualmente. Se necesita implementar pruebas unitarias para la lógica de negocio (validaciones, servicios), pruebas de integración para las API routes, y pruebas end-to-end para los flujos críticos (registro, login, recuperación de contraseña). Dado que es un sistema de identidad, las pruebas son fundamentales para evitar regresiones que puedan afectar a miles de ciudadanos.

Documentación del Proyecto Open Source. Si el proyecto será open source, necesita un README completo, guías de contribución (CONTRIBUTING.md), código de conducta (CODE_OF_CONDUCT.md), documentación de la API interna, y guías para que desarrolladores externos puedan levantar el proyecto localmente.

Seguridad: Rate Limiting y Protección contra Abuso. La versión anterior usaba reCAPTCHA y Cloudflare. En la nueva versión, se necesita definir qué mecanismos de protección se van a implementar: rate limiting en las API routes, protección contra fuerza bruta en el login (ORY tiene esto incorporado, pero hay que configurarlo correctamente), y protección contra bots en el

registro.

Monitoreo y Observabilidad. La versión anterior usaba Sentry. Se necesita decidir si se continúa con Sentry o se migra a otra solución. Además, se deben implementar métricas de rendimiento (tiempos de respuesta, tasas de error), métricas de negocio (registros completados vs abandonados, pasos donde se pierden usuarios), y alertas automáticas cuando algo falle.

2. Backoffice

El backoffice es una aplicación separada, también en Next.js 16, que le da herramientas al equipo de Atención Ciudadana y a los administradores para gestionar las cuentas de los ciudadanos y darles soporte.

Lo que ya se hizo

Se definió la arquitectura base en AWS: EventBridge recibe los eventos (webhooks) que emite ORY Network cada vez que ocurre algo relevante (un ciudadano se registra, inicia sesión, cambia su contraseña, falla un intento de login, etc.). Estos eventos son procesados por funciones Lambda que los transforman y almacenan en DynamoDB. Esta arquitectura permite escalar horizontalmente sin preocuparse por la infraestructura, ya que cada componente es serverless.

Lo que está en progreso

Se está trabajando en el diseño y la implementación de la aplicación web del backoffice. Las funcionalidades principales que se están construyendo incluyen la búsqueda y visualización de cuentas ciudadanas, la capacidad de realizar acciones sobre las cuentas (bloquear, desbloquear, restaurar, eliminar), y la visualización de métricas en tiempo real.

Lo que falta por hacer

Dashboard de Métricas en Tiempo Real. Los administradores necesitan poder ver: cuántos ciudadanos se han registrado hoy, esta semana, este mes. Cuáles son las integraciones OAuth más populares (qué servicios gubernamentales reciben más tráfico desde CUC). En qué paso del registro se quedan los ciudadanos que no completan el proceso. Cuáles son los errores más frecuentes. Cuántas sesiones activas hay en este momento. Todo esto debe alimentarse de los eventos que llegan a EventBridge y se procesan con Lambda.

Gestión de Cuentas Ciudadanas. Los agentes de Atención Ciudadana necesitan poder buscar un ciudadano por cédula, correo o nombre. Ver toda la información de su cuenta. Ver el historial

completo de actividad (cada login, cada cambio de contraseña, cada error). Realizar acciones administrativas como resetear la contraseña, desbloquear la cuenta, verificar manualmente una identidad cuando el proceso biométrico falle.

Trazabilidad Completa del Journey del Ciudadano. Esta es una de las funcionalidades más ambiciosas. La idea es que cuando un ciudadano llame a soporte diciendo "intenté registrarme y no pude", el agente pueda ver exactamente qué pasó: el ciudadano ingresó su cédula a las 10:15am, la validación fue exitosa, pasó al paso 2, ingresó su correo, pasó al paso 3, la verificación facial falló con un error de iluminación insuficiente a las 10:18am, el ciudadano lo intentó de nuevo y falló por timeout a las 10:19am, y luego abandonó el proceso. Para lograr esto, se necesita que el portal ciudadano envíe eventos detallados de cada acción del usuario, no solo los eventos que ORY emite naturalmente.

Sistema de Tickets y Escalación. Cuando el agente de IA no pueda resolver un problema del ciudadano, debe crear un ticket que un agente humano pueda gestionar desde el backoffice. Se necesita diseñar el flujo completo: cómo se crea el ticket, qué información incluye, cómo se asigna a un agente, cómo se le da seguimiento, y cómo se notifica al ciudadano cuando su caso se resuelve.

Envío de Notificaciones. Los agentes deben poder enviar notificaciones a ciudadanos individuales o grupos de ciudadanos desde el backoffice. Estas notificaciones llegan al buzón ciudadano del portal. Se necesita definir los tipos de notificación, las plantillas, y quién tiene permisos para enviar qué tipo de notificación.

Control de Acceso Basado en Roles (RBAC). El backoffice tendrá diferentes tipos de usuarios: administradores con acceso total, agentes de Atención Ciudadana con acceso limitado a gestión de cuentas y soporte, y visualizadores que solo pueden ver métricas. Se necesita implementar este sistema de permisos.

3. Infraestructura y DevOps

Lo que ya se hizo

El CI/CD del portal ciudadano está configurado con GitHub Actions desplegando a Google Cloud Run. El Dockerfile está optimizado con multi-stage builds usando Bun como runtime. Las variables de entorno están separadas correctamente entre las de build time (NEXT_PUBLIC_*) y las de runtime.

Lo que falta por hacer

Ambientes de Desarrollo, Staging y Producción. Actualmente solo hay configuración para staging. Se necesita definir y configurar los tres ambientes con sus respectivas variables de entorno, URLs de ORY, credenciales de AWS, y dominios.

Infraestructura como Código. Los recursos de AWS (EventBridge, Lambda, DynamoDB) y de GCP (Cloud Run, Artifact Registry) deben estar definidos en código, idealmente usando Terraform o AWS CDK para los recursos de AWS y Terraform para los de GCP. Esto permite replicar la infraestructura de forma determinística y mantener un historial de cambios.

Monitoreo de Infraestructura. Se necesitan dashboards de CloudWatch para los recursos de AWS y de Cloud Monitoring para Cloud Run. Alertas automáticas cuando hay errores, latencia alta, o recursos saturados.

Dominio y SSL. Definir cuál será el dominio de CUC v2 (si se mantiene cuentaunica.gob.do o se migra a algo bajo digital.gob.do), configurar los certificados SSL y la integración con Cloudflare o el servicio de protección DDoS que se elija.

Backups y Recuperación ante Desastres. Aunque ORY Network maneja las identidades en su infraestructura, se necesita una estrategia de respaldo para los datos en DynamoDB, los logs, y las configuraciones. También se necesita un plan de recuperación ante desastres que defina los tiempos máximos aceptables de inactividad (RTO) y de pérdida de datos (RPO).

4. Seguridad

Lo que ya se hizo

ORY Network proporciona cifrado en tránsito (HTTPS/TLS 1.2+), cifrado en reposo (AES-256), hash seguro de contraseñas (bcrypt con salt), y tokens de sesión con expiración configurable. AWS Rekognition maneja los datos biométricos bajo los estándares de cumplimiento de AWS (SOC 2, ISO 27001). La validación de cédulas incluye verificación de formato y checksum.

Lo que falta por hacer

Auditoría de Seguridad. Antes del lanzamiento, se necesita una auditoría de seguridad completa: revisión del código, pruebas de penetración, y verificación de cumplimiento con las normativas dominicanas de protección de datos.

Política de Contraseñas. Definir y documentar la política de contraseñas: longitud mínima, complejidad requerida, verificación contra bases de datos de contraseñas filtradas (ORY soporta esto con HaveIBeenPwned), y política de expiración.

Segundo Factor de Autenticación (2FA). ORY soporta TOTP (Google Authenticator, Authy) y WebAuthn (llaves de seguridad, biometría del dispositivo). Se necesita implementar la interfaz para que los ciudadanos puedan activar y gestionar su 2FA.

Manejo de Datos Sensibles. Documentar qué datos personales se almacenan, dónde se almacenan, quién tiene acceso, por cuánto tiempo se retienen, y cómo se eliminan cuando el ciudadano lo solicita. Esto es especialmente importante para un proyecto open source donde el código es público.

5. Experiencia de Usuario e Investigación

Lo que falta por hacer

Pruebas de Usabilidad. Antes del lanzamiento, se deben realizar pruebas de usabilidad con ciudadanos reales: personas de diferentes edades, niveles de educación, y familiaridad con tecnología. El registro con verificación facial es un proceso que puede intimidar a muchas personas, y necesitamos asegurarnos de que las instrucciones sean claras y el proceso sea lo más sencillo posible.

Optimización del Flujo de Registro. Medir la tasa de abandono en cada paso del registro e identificar oportunidades de mejora. Si muchos ciudadanos abandonan en la verificación facial, tal vez necesitamos mejorar las instrucciones o el manejo de errores en ese paso.

Soporte para Dispositivos de Gama Baja. Muchos ciudadanos dominicanos usan dispositivos Android de gama baja con cámaras de baja resolución. Se necesita probar el flujo de verificación facial en estos dispositivos y ajustar los umbrales de confianza si es necesario.

Página de Estado del Servicio. Crear una página pública donde los ciudadanos puedan verificar si CUC está funcionando correctamente. Esto reduce la carga de soporte cuando hay interrupciones planificadas o incidentes.

Resumen del estado actual

Para tener una vista rápida del progreso general:

El portal ciudadano tiene su estructura base completa, con los flujos de autenticación y registro funcionando en desarrollo. Las decisiones de arquitectura están tomadas y validadas. Sin embargo,

falta completar las funcionalidades del dashboard, el sistema de notificaciones, el agente de IA, las pruebas automatizadas, y la auditoría de seguridad y accesibilidad.

El backoffice tiene su arquitectura AWS definida pero la aplicación web está en etapas tempranas de desarrollo. Las funcionalidades más ambiciosas como la trazabilidad completa del journey del ciudadano y el sistema de tickets requieren trabajo coordinado entre el equipo del portal y el del backoffice.

La infraestructura necesita madurar en cuanto a ambientes, infraestructura como código, monitoreo, y planes de recuperación ante desastres.

Este es un proyecto ambicioso, y reconocemos que hay mucho por hacer. Pero la base es sólida: las decisiones tecnológicas están tomadas, la integración con ORY funciona, la verificación biométrica está probada, y el equipo tiene claridad sobre hacia dónde vamos.

Propuesta de valor