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.

Revisión #2
Creado 2026-03-25 17:11:42 UTC por Marluan Espiritusanto
Actualizado 2026-04-15 09:42:10 UTC por Marluan Espiritusanto