GOBIERNO DE LA PROVINCIA DE CÓRDOBA

CFI CONSEJO FEDERAL DE INVERSIONES

PROVINCIA DE CÓRDOBA - CFI

INFORME FINAL DE PROYECTO

ANÁLISIS E INSTALACIÓN DE AMBIENTES NO PRODUCTIVOS EN ARQUITECTURA CLOUD PARA APLICATIVO REDI

MARZO 2025

RESUMEN EJECUTIVO

El presente informe documenta el estado final del proyecto "Análisis e Instalación de Ambientes No Productivos en Arquitectura Cloud para el Aplicativo REDI", cuyo propósito fue evaluar la viabilidad de una migración del sistema REDI desde su infraestructura actual on-premises hacia un entorno basado en la nube, garantizando su escalabilidad, seguridad y eficiencia operativa.

A lo largo del proyecto, se realizó un análisis exhaustivo de la arquitectura actual del sistema, identificando sus principales limitaciones en términos de capacidad de crecimiento, gestión de recursos, monitoreo y seguridad. Se evaluó la compatibilidad del sistema con un esquema basado en servicios provistos por la nube de Amazon Web Services (AWS), definiendo una nueva arquitectura que incorpora Virtual Private Clouds (VPCs), balanceo de carga (ALB), contenedores en Kubernetes (EKS con Fargate), RDS Proxy para la gestión de bases de datos y Web Application Firewall (WAF) para la protección contra amenazas.

Para validar la factibilidad de esta arquitectura, se llevó a cabo la instalación y configuración de ambientes de Desarrollo, Testing y Pre-Producción (ambientes no productivos o menores), replicando la funcionalidad ofrecida por el ambiente de Producción pero sin afectar la operatividad del sistema actual. Se definieron estrategias para el manejo de redes, segmentación de subredes, reglas de seguridad y administración de acceso, asegurando un entorno seguro y controlado para la ejecución de pruebas.

Además, se estableció un esquema de integración y despliegue continuo (CI/CD) en Azure DevOps, con el objetivo de evaluar el impacto de la automatización en la entrega de versiones del sistema. Se configuraron pipelines de compilación, prueba y despliegue, validando su correcto funcionamiento y su capacidad para reducir errores en futuras implementaciones.

Los resultados obtenidos a lo largo de este proceso han permitido concluir que la arquitectura en la nube propuesta ofrece beneficios sustanciales en comparación con el esquema actual, mejorando la escalabilidad, la resiliencia y la eficiencia operativa del sistema REDI. Sin embargo, el análisis también ha identificado desafíos técnicos, particularmente en la integración con servicios gubernamentales externos y en la gestión de conexiones con la base de datos Oracle aún alojada on-premise.

Este estudio establece una base técnica y operativa para futuras decisiones estratégicas sobre la evolución del sistema REDI. Si bien la implementación en producción no formó parte del alcance de este proyecto, la validación realizada sobre los entornos no productivos demuestra que una

migración gradual hacia la nube es viable y que la infraestructura propuesta puede soportar la carga operativa del sistema con los ajustes adecuados.

El proyecto concluye con un conjunto de recomendaciones para una posible implementación en producción, así como con la documentación detallada de las configuraciones realizadas, los análisis técnicos y los hallazgos obtenidos en cada una de las fases del proceso. Esta información servirá como insumo fundamental para la toma de decisiones sobre la evolución futura del sistema REDI.

Contenido

INTRODUCCIÓN

Propósito

El informe final documenta el análisis y la culminación del proyecto "Análisis e Instalación de Ambientes No Productivos en Arquitectura de Cloud para el Aplicativo REDI", llevado a cabo en el marco de la modernización tecnológica del sistema REDI. A lo largo del proyecto, se evaluó la viabilidad y se implementaron entornos de desarrollo, testing y preproducción en AWS, con el propósito de garantizar un modelo escalable, seguro y eficiente para futuras migraciones.

Transcurridos seis meses desde el inicio del proyecto, el objetivo principal de este informe es detallar las actividades realizadas, los hitos alcanzados y los resultados obtenidos en la implementación de los entornos en la nube, así como identificar los próximos pasos para la evolución del sistema.

La infraestructura cloud diseñada busca mejorar la estabilidad, disponibilidad y seguridad del sistema REDI, reduciendo la dependencia de tecnologías on-premise y optimizando la gestión operativa a través de herramientas de automatización como Azure DevOps, Kubernetes (EKS con Fargate), RDS Proxy y Web Application Firewall (WAF).

Este análisis no solo evalúa la ejecución técnica del proyecto, sino que también proporciona una visión integral sobre los beneficios, desafíos y oportunidades detectadas en la migración de REDI hacia la nube. Además, se establecen recomendaciones estratégicas para la continuidad del proceso de modernización, asegurando que el sistema pueda operar de manera eficiente dentro de la nueva infraestructura.

Descripción General del Proyecto

El proyecto "Análisis e Instalación de Ambientes No Productivos en Arquitectura de Cloud para el Aplicativo REDI" tiene como objetivo evaluar la viabilidad y los beneficios de la implementación de infraestructura en la nube para soportar los entornos de desarrollo, testing y preproducción del sistema REDI. Este análisis busca identificar las mejoras potenciales en escalabilidad, flexibilidad y seguridad en comparación con la infraestructura on-premise actual.

A lo largo del proyecto, se ha realizado una evaluación detallada de la arquitectura existente, detectando sus limitaciones en cuanto a escalabilidad, mantenimiento y eficiencia operativa. Sobre esta base, se ha diseñado una

infraestructura teórica en la nube utilizando servicios nativos de AWS, tales como Amazon Virtual Private Cloud (VPC), Elastic Kubernetes Service (EKS), RDS Proxy, Web Application Firewall (WAF) y Load Balancers (ALB/NLB), con el propósito de establecer un modelo de referencia para futuros despliegues.

El análisis incluyó la configuración y validación de entornos no productivos en AWS, replicando condiciones similares a las de producción para evaluar el desempeño de las aplicaciones y la interacción con los sistemas actuales. Además, se han definido estrategias de integración y automatización mediante Azure DevOps, permitiendo la gestión estructurada de los despliegues y asegurando una trazabilidad completa de las implementaciones.

Este estudio proporciona las bases para una posible adopción de infraestructura en la nube, estableciendo un marco comparativo con el entorno actual y permitiendo una toma de decisiones fundamentada en términos de costos, seguridad y operatividad del sistema REDI.

Fecha de Firma del Contrato y Plazos

El contrato para la ejecución del proyecto de “Análisis e Instalación de Ambientes No Productivos en Arquitectura de Cloud para el Aplicativo REDI” fue firmado el 11 de septiembre de 2024, estableciendo un plazo de seis meses para completar todas las fases del proyecto.

Este informe final cubre el período completo del proyecto, enfocándose en los hitos alcanzados, las actividades realizadas y las estrategias utilizadas para mitigar riesgos.

RESUMEN DEL AVANCE DEL PROYECTO

Actividades Técnicas Realizadas

Análisis de la arquitectura actual y evaluación del estado inicial

El sistema REDI está diseñado sobre una arquitectura tradicional monolítica. La infraestructura on-premises, donde los componentes críticos están alojados en entornos locales, presenta diversos desafíos en términos de escalabilidad, seguridad y eficiencia operativa. Todo esto dificulta su capacidad para adaptarse a nuevas demandas, mejorar su desempeño y emplear nuevas tecnologías y servicios.

Dentro de la infraestructura original, se identificaron varios componentes clave que conforman el sistema. Las dos aplicaciones principales del ecosistema son URGP y URGP_WEB, gestionadas a través del motor de procesos Oracle BPM. La base de datos utilizada es parte también de la solución basada en Oracle que el Gobierno de la Provincia de Córdoba licenció hace años. Esta base de datos está instalada on-premises en el Super Centro de Gobierno, con una conectividad limitada y restricciones en la gestión eficiente de recursos.

Adicionalmente, el sistema mantiene integraciones con plataformas gubernamentales externas, tales como el Tribunal Superior de Justicia, CDD y Ciudadano Digital (CiDi), lo que genera complejidades extra en la comunicación y monitoreo de procesos.

La evaluación inicial de la arquitectura evidenció varias limitaciones críticas que afectan el rendimiento y la operatividad del sistema. En primer lugar, la infraestructura presenta una escalabilidad limitada, ya que el sistema no puede gestionar eficientemente aumentos en la demanda debido a su estructura rígida y su dependencia de recursos físicos locales, en gran medida compartidos con otros aplicativos alojados en el Super Centro. El acceso a la base de datos Oracle representa otro desafío significativo, dado que la arquitectura original carecía de optimización en la gestión de conexiones y recursos, lo cual afecta la eficiencia en las consultas y procesamiento de datos.

Desde el punto de vista de la seguridad, la arquitectura carece de mecanismos robustos para la protección contra amenazas externas, como ataques distribuidos de denegación de servicio (DDoS) o inyección SQL, lo que representa un riesgo potencial para la integridad del sistema. A su vez, el monitoreo del sistema resulta insuficiente e ineficiente, ya que no cuenta con herramientas adecuadas para la supervisión en tiempo real, lo que dificulta la

detección temprana de problemas operativos y la trazabilidad de los eventos dentro del sistema.

En respuesta a estas limitaciones, el presente proyecto propuso una migración hacia una arquitectura basada en la nube, específicamente en AWS, con el objetivo de mejorar la escalabilidad, seguridad y monitoreo del sistema.

La nueva arquitectura definida incorpora servicios avanzados como Kubernetes (EKS con Fargate) para la gestión dinámica de contenedores, RDS Proxy para optimizar el acceso a la base de datos Oracle, Application Load Balancer (ALB) para distribuir de manera eficiente el tráfico y Web Application Firewall (WAF) para fortalecer la protección del sistema contra ataques externos.

Esta migración permitirá que el sistema REDI evolucione hacia una infraestructura más flexible y escalable, asegurando un desempeño óptimo, mayor resiliencia y una gestión más eficiente de los recursos, facilitando su adaptación a nuevas necesidades y demandas operativas.

Definición de la infraestructura en la nube

La infraestructura propuesta para el aplicativo REDI fue diseñada sobre Amazon Web Services (AWS) con el objetivo de garantizar escalabilidad, eficiencia operativa, seguridad y monitoreo continuo. La arquitectura definida contempla la segmentación de servicios en distintos componentes estratégicos, asegurando una administración optimizada de los recursos en la nube.

Se definió la creación de una VPC (Virtual Private Cloud), una red privada aislada dentro de AWS. Esta red permite organizar los recursos del sistema y segmentar el tráfico en subredes públicas y privadas, garantizando la seguridad del entorno al evitar la exposición directa de servicios internos a internet. La segmentación en subredes permite separar los componentes críticos del sistema de los accesibles públicamente, reduciendo la superficie de ataque y mejorando la administración del tráfico interno.

Para la gestión de aplicaciones en contenedores, se definió la utilización de Amazon Elastic Kubernetes Service (EKS), un servicio gestionado de Kubernetes que permite el despliegue, escalado y operación de aplicaciones en contenedores. Se estableció que el modo de ejecución será AWS Fargate, lo que elimina la necesidad de administrar servidores físicos o virtuales, ya que este servicio asigna automáticamente los recursos de CPU y memoria en función de la demanda del sistema. Con esta arquitectura, se garantiza la escalabilidad dinámica, permitiendo que los servicios se ajusten a los cambios en la carga de trabajo sin intervención manual.

Kubernetes es un sistema de orquestación de contenedores que permite gestionar múltiples aplicaciones distribuidas dentro de un entorno unificado. Gracias a EKS, se logra una administración eficiente del ciclo de vida de los contenedores, asegurando alta disponibilidad y facilidad en la implementación de nuevas versiones del software.

Para distribuir las solicitudes de los usuarios de manera equitativa entre los servicios desplegados, se definió la configuración de Application Load Balancers (ALB) en dos niveles. El Internal Load Balancer gestionará la distribución de tráfico dentro del clúster EKS, asegurando que la carga de procesamiento se mantenga equilibrada entre las instancias activas. Por otro lado, el External Load Balancer administrará el tráfico entrante desde aplicaciones externas, controlando y limitando el acceso público a los servicios internos.

Además, se determinó la implementación de Oracle HTTP Server (OHS) como punto de entrada para la gestión del tráfico HTTP/HTTPS hacia las aplicaciones URGP y URGP_WEB. Estos servidores facilitarán la distribución eficiente de las solicitudes mediante mecanismos internos de balanceo de carga, asegurando la disponibilidad y estabilidad del sistema.

Dado que la base de datos principal del aplicativo REDI seguirá operando en infraestructura on-premise, se definió la utilización de AWS RDS Proxy para gestionar las conexiones de los servicios en la nube con la base de datos Oracle. RDS Proxy optimiza las conexiones concurrentes, reduciendo la carga sobre la base de datos y mejorando la estabilidad del acceso a los datos. Esta solución permite evitar problemas de saturación en la base de datos y mejorar el tiempo de respuesta de las aplicaciones.

Se estableció la configuración de AWS Web Application Firewall (WAF) para la protección de los servicios expuestos al público. El WAF es un sistema de filtrado de tráfico que mitiga amenazas como ataques DDoS, inyecciones SQL y Cross-Site Scripting (XSS), garantizando que sólo las solicitudes legítimas lleguen a las aplicaciones. Se definirán reglas de seguridad estrictas que permitan bloquear accesos no autorizados y prevenir vulnerabilidades comunes en aplicaciones web.

Además, se determinó el uso de AWS Certificate Manager (ACM) para la administración de certificados SSL/TLS, lo que permitirá establecer comunicaciones seguras mediante HTTPS. Este servicio gestiona la renovación automática de los certificados, evitando fallos de seguridad por vencimientos o configuraciones incorrectas.

Para la gestión de credenciales y claves API, se estableció la utilización de AWS Secrets Manager, lo que permitirá almacenar información sensible de

manera cifrada y con acceso restringido únicamente a los servicios que la requieran. Secrets Manager evita que credenciales críticas sean expuestas en archivos de configuración o código fuente, reduciendo riesgos de seguridad.

En términos de control de accesos, se definió que cada componente de la infraestructura contará con su propio rol IAM (Identity and Access Management), lo que permitirá aplicar restricciones granulares y asegurar que cada servicio tenga únicamente los permisos necesarios para operar.

Para garantizar el monitoreo continuo del sistema, se estableció la integración de herramientas especializadas en la recolección de métricas y análisis de rendimiento. Se definió la utilización de Prometheus para la captura de métricas en tiempo real sobre el uso de CPU, memoria y red de los contenedores y servicios. Estos datos serán visualizados a través de Grafana, lo que facilitará la interpretación y análisis del estado del sistema.

Además, se determinó el uso de AWS CloudWatch para la recolección de logs, monitoreo de eventos y generación de alertas en caso de detección de anomalías operativas. Con esta configuración, se asegura una supervisión proactiva del sistema, permitiendo la identificación temprana de posibles fallos o degradaciones en el rendimiento.

Para garantizar la disponibilidad de la información y la continuidad del servicio ante incidentes, se definió la implementación de un sistema de backup automático mediante AWS Backup. Este servicio realizará copias de seguridad programadas de las instancias EC2 y bases de datos alojadas en RDS, asegurando que, en caso de contingencia, la información pueda ser recuperada de manera eficiente.

Asimismo, se establecerán snapshots periódicos de las bases de datos, lo que permitirá la restauración de versiones anteriores en caso de pérdida de datos o fallos críticos. Esta estrategia de respaldo minimizará el impacto de cualquier incidente, garantizando la continuidad operativa del sistema.

Resumen de la Arquitectura Definida

La infraestructura diseñada para el aplicativo REDI se basará en AWS, con una arquitectura flexible y segura que integra:

- Una VPC segmentada en subredes públicas y privadas, asegurando el aislamiento de recursos sensibles.
- Un clúster Kubernetes administrado con EKS y Fargate, eliminando la gestión manual de servidores y permitiendo escalabilidad dinámica.

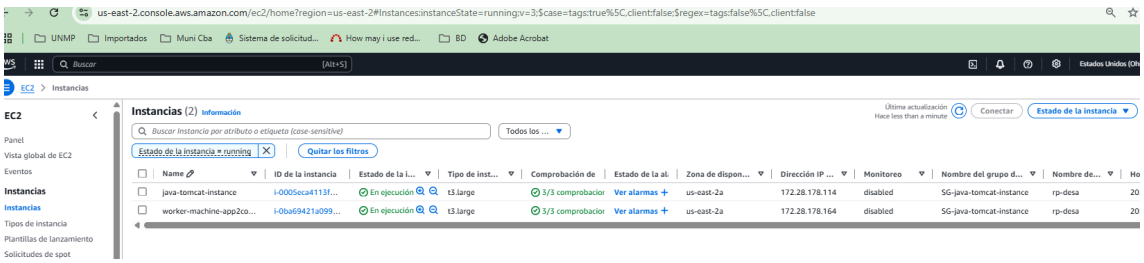
- Balanceadores de carga internos y externos, asegurando distribución equitativa del tráfico y disponibilidad de los servicios.
- RDS Proxy para optimizar las conexiones con la base de datos Oracle on-premise.
- Sistemas de seguridad avanzados como WAF y AWS Certificate Manager, garantizando protección contra amenazas y comunicaciones cifradas.
- Gestión segura de credenciales y permisos a través de AWS Secrets Manager e IAM Roles.
- Monitoreo continuo con Prometheus, Grafana y CloudWatch, proporcionando métricas detalladas y alertas en tiempo real.
- Un sistema de backup y recuperación automática, minimizando riesgos de pérdida de información y asegurando la disponibilidad del sistema.

Con esta arquitectura, se establecen las bases para un sistema altamente seguro, escalable y resiliente, preparado para adaptarse a futuras necesidades y optimizar la operatividad del aplicativo en la nube.

Para más detalle, referirse al documento adjunto “RP04-Arquitectura-sistema-Infraestructura_REDI.pdf”.

Instalación y Configuración del Ambiente de Desarrollo

La instalación y configuración del ambiente de desarrollo en AWS se llevó a cabo exitosamente, asegurando una infraestructura segmentada, protegida y monitoreada. Este entorno permite a los desarrolladores realizar pruebas y ajustes antes de avanzar a las siguientes etapas del proyecto, garantizando así una transición fluida hacia los entornos de prueba y producción.



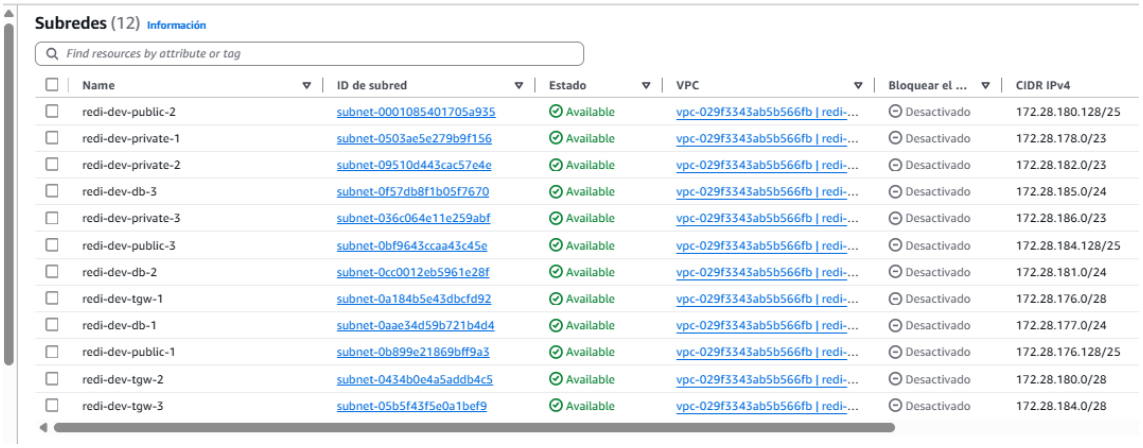
Ver imagen adjunta “Infraestructura Dev – AWS.jpg”

Se creó la VPC "redi-dev-vpc", la cual opera dentro del esquema de direcciones IP 172.28.176.0/20. Esta configuración permite que todos los recursos internos del ambiente de desarrollo utilicen direcciones privadas, evitando exposiciones innecesarias a internet y fortaleciendo la seguridad de la infraestructura.

Ver imagen adjunta “VPC Dev – AWS.jpg”

Dentro de esta VPC, se definieron subredes con el objetivo de segmentar el tráfico y organizar los recursos de manera eficiente. Se establecieron 12 subredes, distribuidas en tres zonas de disponibilidad de AWS, lo que permite garantizar redundancia y tolerancia a fallos.

Las subredes públicas están destinadas a los recursos que necesitan acceso a internet, tales como servidores que reciben peticiones externas. Estas incluyen redi-dev-public-1, redi-dev-public-2 y redi-dev-public-3, cada una ubicada en una zona de disponibilidad diferente.



Ver imagen adjunta “Subredes Dev – AWS.jpg”

Las subredes privadas alojan servidores internos y servicios que no deben ser accesibles desde el exterior, asegurando mayor control y protección. Se configuraron redi-dev-private-1, redi-dev-private-2 y redi-dev-private-3, en diferentes zonas de disponibilidad.

Para la gestión de bases de datos, se crearon subredes dedicadas con el objetivo de proporcionar aislamiento y seguridad. Las subredes redi-dev-db-1, redi-dev-db-2 y redi-dev-db-3 permiten alojar los servidores de bases de datos en un entorno protegido.

Además, se implementaron subredes específicas para Transit Gateway (TGW), encargadas de facilitar la conexión de la VPC de desarrollo con otras redes internas del gobierno o sistemas externos. Las subredes redi-dev-tgw-1, redi-dev-tgw-2 y redi-dev-tgw-3 cumplen esta función.

Se establecieron cinco tablas de enrutamiento, fundamentales para organizar el tráfico dentro de la infraestructura. La tabla redi-dev-rt-private gestiona el tráfico entre las subredes privadas, mientras que la tabla redi-dev-rt-public administra el tráfico de las subredes públicas, permitiendo el acceso a internet cuando sea necesario. Para la conexión con redes externas mediante AWS Transit Gateway, se configuró la tabla redi-dev-rt-tgw. Finalmente, la tabla redi-dev-rt-database asegura que solo los servicios autorizados puedan comunicarse con las bases de datos, protegiendo así los datos sensibles.

Ver imagen adjunta “Enrutamiento Dev – AWS.jpg”

Para garantizar la conectividad de los diferentes servicios dentro de AWS y con sistemas externos, se establecieron dos mecanismos de conexión esenciales. Se configuró redi-dev-igw, un Internet Gateway (IGW) que permite a los recursos con acceso público comunicarse con internet de manera segura. Además, se implementó redi-dev-s3-endpoint, un endpoint de Amazon S3, que permite que los servicios dentro de la VPC accedan directamente al almacenamiento en la nube sin necesidad de salir a internet, mejorando la seguridad y reduciendo costos de tráfico de datos.

Para la ejecución de los servicios del ambiente de desarrollo, se configuraron dos instancias EC2 (Elastic Compute Cloud), asegurando la capacidad de procesamiento necesaria para ejecutar aplicaciones y servicios en AWS.

La primera instancia, java-tomcat-instance, está destinada a alojar el servidor de aplicaciones Tomcat, utilizado para la ejecución del sistema REDI en

su fase de desarrollo. Esta instancia cuenta con 2 vCPUs y 8GB de RAM, opera en la zona de disponibilidad us-east-2a y tiene asignada la dirección IP interna 172.28.178.114.

La segunda instancia, worker-machine-app2container, está configurada para ejecutar tareas de conversión y pruebas de aplicaciones en contenedores, como parte del proceso de modernización del sistema hacia una arquitectura basada en microservicios. Sus características son similares a las de la instancia anterior, con 2 vCPUs y 8GB de RAM, en la zona de disponibilidad us-east-2a, con dirección IP interna 172.28.178.164.

Ambas instancias fueron validadas con comprobaciones de estado exitosas, confirmando que se encuentran en pleno funcionamiento.

Para garantizar la seguridad del ambiente de desarrollo, se implementaron diversas medidas de protección. Se asignaron roles y políticas IAM (Identity and Access Management) para limitar el acceso a los recursos únicamente a los servicios y usuarios autorizados. Se configuró AWS Secrets Manager para almacenar credenciales y claves API de forma segura, evitando su exposición en los repositorios de código.

Asimismo, se establecieron grupos de seguridad (Security Groups) con reglas estrictas de control de tráfico, definiendo qué tipo de conexiones están permitidas en cada instancia y restringiendo accesos no autorizados.

Para garantizar la disponibilidad de la información y la continuidad del servicio ante fallos, se definieron estrategias de respaldo y recuperación. Se habilitó AWS Backup para realizar copias de seguridad automáticas de las instancias EC2 y bases de datos. Además, se configuraron snapshots de RDS, que generan copias diarias de las bases de datos, permitiendo la recuperación en caso de incidentes.

Con esta infraestructura, el ambiente de desarrollo se encuentra completamente configurado, proporcionando un espacio seguro y eficiente para que los desarrolladores realicen pruebas y ajustes antes de la transición a los siguientes entornos del proyecto.

Instalación de Infraestructura y Servicios de DevOps

Se crearon y configuraron los repositorios de código en Azure Repos, estableciendo una estructura de control de versiones organizada y eficiente. Cada repositorio fue configurado con archivos esenciales como .gitignore, README.md y pom.xml en los casos de proyectos basados en Maven, además

de definir reglas de acceso para garantizar la seguridad y trazabilidad de los cambios realizados en el código.

Dentro del proyecto, se configuraron los siguientes repositorios principales:

- portal-backend: código fuente del backend principal del portal.
- rgp-servicios: servicios internos del sistema.
- rgp-web-backend: código fuente del backend de la aplicación web.
- rgp-pagos-online: procesamiento de pagos en línea.
- rgp-tasas-backend: gestión de tasas e impuestos.
- rgp-integracion: código de integración con sistemas externos.
- rgp-common: librerías compartidas entre distintos módulos del sistema.
- solicitudes-frontend: código fuente de la interfaz de usuario del módulo de solicitudes.
- solicitudes-backend: código fuente del backend del módulo de solicitudes.

Cada repositorio sigue una estrategia de ramas de Git, estructurada para facilitar el desarrollo y la estabilidad del código. Se definió la rama develop-2 como la principal para el desarrollo activo, mientras que las feature branches se utilizan para la implementación de nuevas funcionalidades y las release branches están destinadas a la estabilización y entrega de versiones finales. Esta estructura permite un control detallado sobre la evolución del código y facilita la integración de cambios de manera ordenada.

Para la gestión de la integración y despliegue continuo (CI/CD), se configuraron pipelines en Azure DevOps, permitiendo la automatización de la integración y el despliegue del código en los entornos configurados dentro de AWS. Los pipelines implementados abarcan los mismos módulos definidos en los repositorios, asegurando la automatización de pruebas y despliegues en cada iteración.

Cada pipeline sigue un proceso estructurado en fases, asegurando que cada cambio en el código pase por un conjunto de validaciones antes de ser desplegado en el ambiente de desarrollo.

El flujo del pipeline de CI/CD incluye varias etapas clave. Primero, se realiza la extracción del código fuente, donde se obtiene la última versión desde Azure Repos. Luego, en la fase de compilación y análisis de código, se utiliza Maven para proyectos en Java, y se ejecuta un análisis de calidad con SonarQube, permitiendo detectar vulnerabilidades y errores en el código.

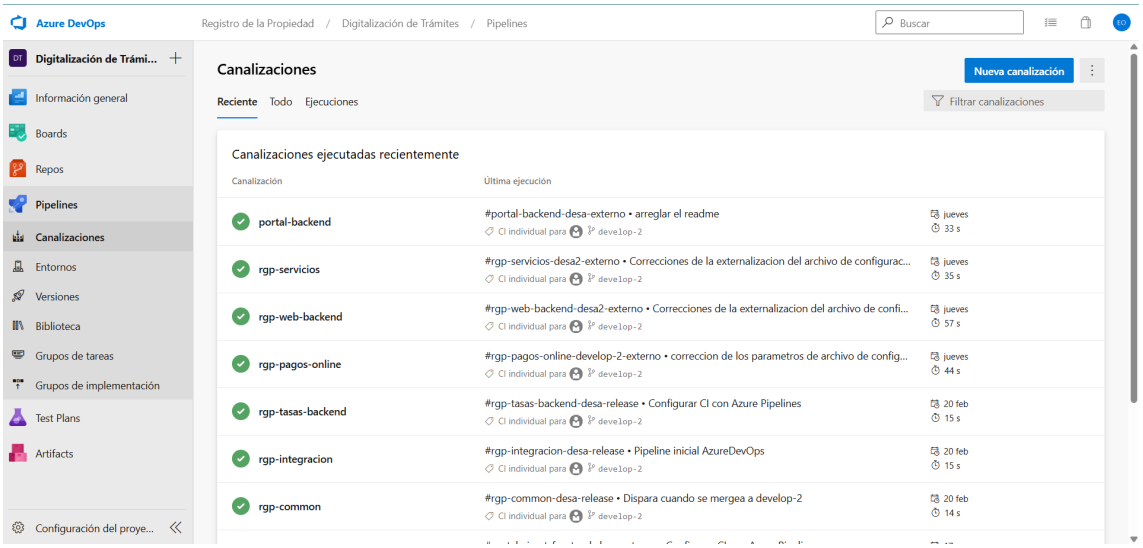
Posteriormente, se lleva a cabo la ejecución de pruebas automatizadas, donde se realizan pruebas unitarias con JUnit y pruebas de integración con Postman/Newman, validando así la correcta comunicación entre las API del sistema.

El despliegue en AWS se realiza mediante Azure Pipelines, utilizando servidores EC2 y el clúster Kubernetes (EKS) para la implementación de servicios. Se gestionan los despliegues con AWS CodeDeploy y Helm Charts, asegurando que cada versión del software se implemente de manera controlada y segura. Adicionalmente, se configuró AWS RDS Proxy para optimizar las conexiones con la base de datos Oracle, mejorando la eficiencia en el acceso a los datos.

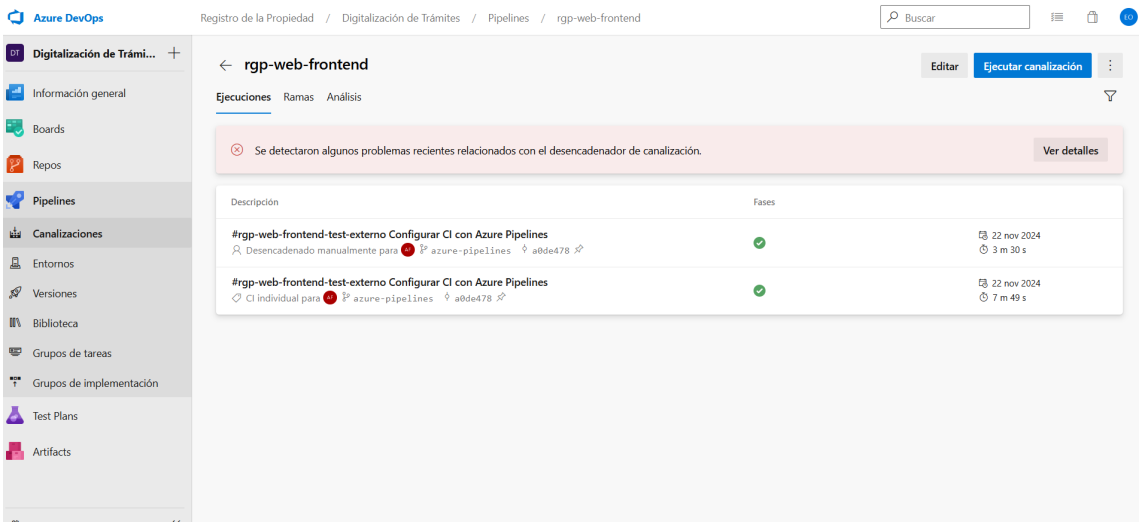
Tras la configuración de los pipelines, se realizaron diversas pruebas para verificar su correcto funcionamiento. Se validó la ejecución exitosa de cada canalización en Azure DevOps, confirmando que los despliegues se realizan sin errores y que cada módulo del sistema es implementado de manera correcta.

En particular, se destacó la validación del pipeline del portal-backend, donde se llevó a cabo una corrección en la externalización de archivos de configuración (properties). Este ajuste aseguró que la información de conexión a la base de datos se gestione de manera flexible y segura, evitando configuraciones rígidas dentro del código fuente.

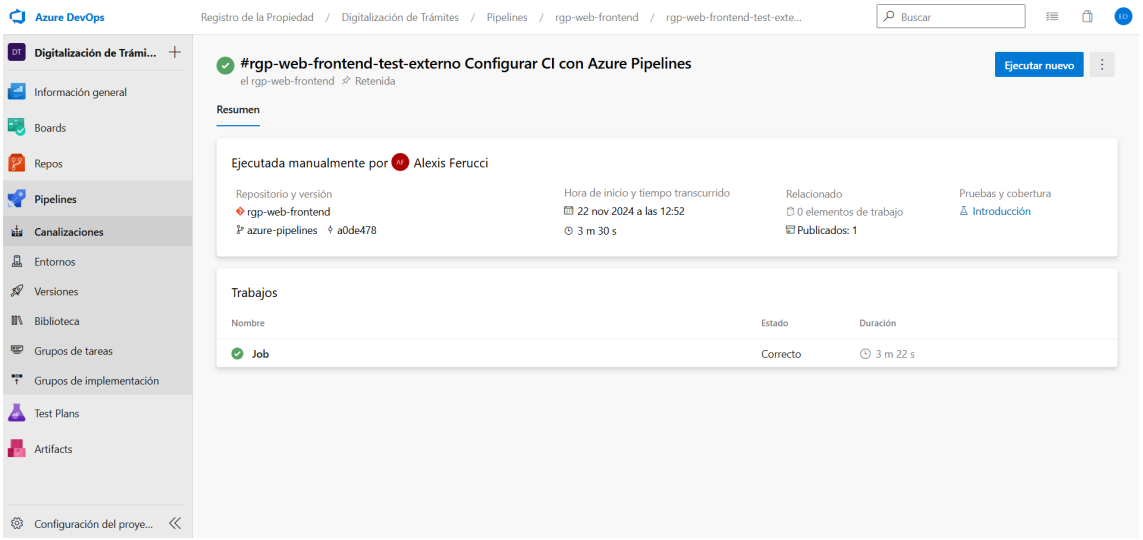
Los logs de ejecución en Azure DevOps reflejan que las canalizaciones fueron ejecutadas sin fallas, con tiempos de ejecución óptimos y sin errores en las fases de pruebas y despliegue. Esto garantiza que el proceso de automatización de CI/CD cumple con los requisitos del proyecto, facilitando la implementación ágil y segura de nuevas versiones dentro del ecosistema AWS.



Ver imagen adjunta “Pipelines AZURE – AWS.jpg”



Ver imagen adjunta “Pipelines AZURE – AWS 2.jpg”



Ver imagen adjunta “Pipelines AZURE – AWS 3.jpg”

Instalación, Configuración y Ajuste del Ambiente de Testing

Tras la validación completa del ambiente de desarrollo, se llevó a cabo la implementación del ambiente de Testing en AWS, con el objetivo de replicar fielmente la infraestructura de desarrollo pero incorporando configuraciones específicas para la ejecución de pruebas funcionales, de integración y de rendimiento antes de la transición a preproducción.

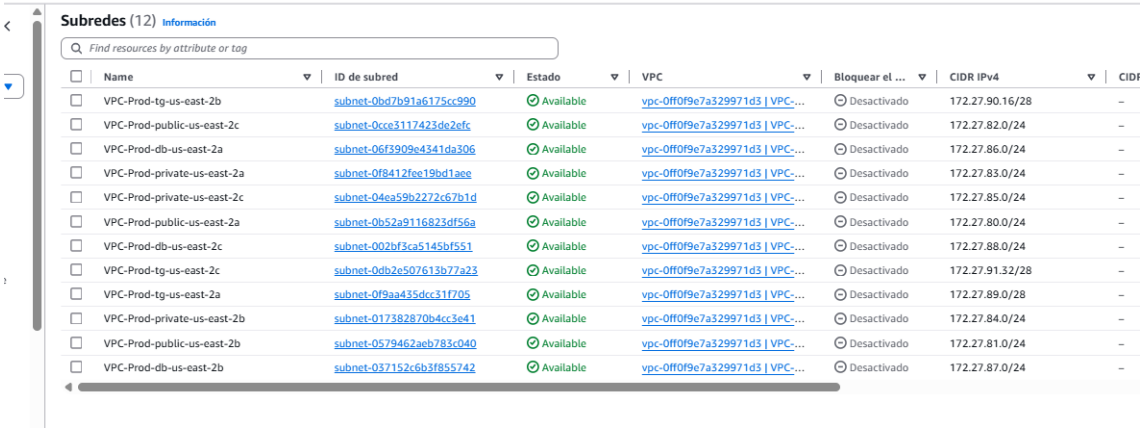
Para lograr esto, se desplegó una nueva VPC de Testing con una arquitectura similar a la de desarrollo, pero completamente aislada, asegurando

así un entorno controlado para la ejecución de pruebas sin interferencias externas. La VPC fue configurada con el identificador vpc-0ff0f9e7a329971d3, con un rango de red CIDR 172.27.80.0/20, y su estado se encuentra disponible y operativo.

Ver imagen adjunta “VPC Test – AWS”

Como parte de la infraestructura, se implementaron nueve subredes distribuidas en tres zonas de disponibilidad en AWS, lo que proporciona redundancia y tolerancia a fallos. Estas subredes fueron organizadas en diferentes categorías según su función dentro del sistema. Las subredes públicas se destinaron a la exposición de servicios externos para pruebas específicas. Las subredes privadas fueron configuradas para alojar servidores internos, protegiendo así los recursos sensibles del entorno de pruebas. Se crearon subredes dedicadas a bases de datos, asegurando el aislamiento y la seguridad de los datos utilizados en Testing. Adicionalmente, se incorporaron subredes para Transit Gateway (TGW), facilitando la conexión con otras redes dentro de AWS o sistemas locales. Para garantizar la seguridad del tráfico, se configuraron reglas de enrutamiento con accesos restringidos, evitando interferencias con otros entornos y permitiendo un flujo de datos controlado entre los componentes del sistema.

Ver imagen adjunta “Enrutamiento Test – AWS.jpg”



Ver imagen adjunta “Subredes Test – AWS.jpg”

Dentro del ambiente de Testing, se desplegaron instancias EC2 con las mismas especificaciones utilizadas en desarrollo, lo que asegura consistencia en las pruebas. Actualmente, se encuentran operativas dos instancias clave. La primera, worker-machine-app2container, es un servidor utilizado para la ejecución de tareas de conversión y pruebas con contenedores. Su identificador es i-0592a8e097b3..., corresponde a un tipo t3.large, y está alojado en la zona de disponibilidad us-east-2a, con dirección IP interna 172.27.83.187. La segunda, java-tomcat-instance, es el servidor encargado de alojar las aplicaciones del sistema para pruebas de integración y funcionalidad. Su identificador es i-0d433f98fd1d..., también es un t3.large, se encuentra en la misma zona de disponibilidad y tiene dirección IP interna 172.27.83.107. Ambas instancias fueron validadas mediante comprobaciones de estado exitosas (3/3), asegurando su correcto funcionamiento y disponibilidad dentro del ambiente de pruebas.

Además de la infraestructura de servidores, se configuraron servicios esenciales para el óptimo desempeño de las pruebas. Se implementaron balanceadores de carga (ALB y NLB) para distribuir el tráfico entre los servidores, asegurando estabilidad y evitando sobrecargas en el sistema. Se habilitó RDS Proxy, optimizando las conexiones con la base de datos Oracle, mejorando la eficiencia en el acceso a los datos y reduciendo la latencia en las consultas. Finalmente, se configuró AWS Secrets Manager como un mecanismo seguro de almacenamiento de credenciales y configuraciones de prueba, garantizando el cumplimiento de estándares de seguridad en la gestión de datos sensibles dentro del entorno de Testing.

Con la instalación y configuración de este ambiente, el sistema cuenta con un entorno dedicado para la validación y certificación de nuevas versiones antes de su despliegue en producción. Esto permite ejecutar pruebas de manera

estructurada y confiable, reduciendo los riesgos de errores en los entornos productivos y asegurando un proceso de desarrollo seguro y eficiente dentro del sistema.

Incorporación del Ambiente de Testing a la Integración Continua

Como parte del proceso de automatización y validación del entorno de pruebas, se integró el ambiente de Testing en el flujo de CI/CD (Integración y Despliegue Continuo) dentro de Azure DevOps. Esta integración permite que cada nueva versión del sistema sea validada en un entorno seguro y controlado antes de ser promovida a producción, asegurando calidad y estabilidad en cada iteración del software.

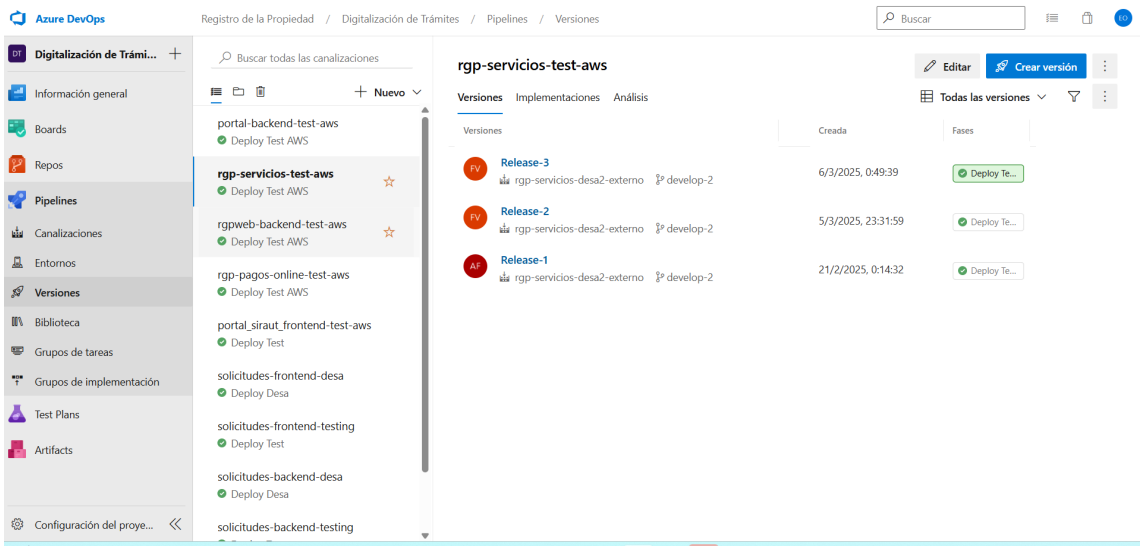
Para garantizar que el entorno de Testing funcione como un filtro de calidad previo a la implementación en preproducción, se llevaron a cabo una serie de configuraciones dentro de Azure DevOps. Se establecieron pipelines específicos para el ambiente de Testing, lo que permite la ejecución de validaciones previas al despliegue de nuevas versiones. Además, se implementó la automatización del despliegue en AWS, asegurando que cada versión liberada pase por una serie de pruebas en un entorno controlado y estructurado.

Uno de los aspectos clave en este proceso fue la implementación de un sistema de versionado, permitiendo la trazabilidad de cada release y su impacto en las pruebas. Esto facilita la identificación de problemas y la validación de versiones antes de ser promovidas a los siguientes entornos. Asimismo, se definieron criterios de validación previos a la promoción del código a producción, asegurando que solo aquellas versiones estables sean desplegadas en entornos superiores.

La infraestructura del ambiente de Testing en AWS se integró en Azure DevOps bajo el pipeline denominado Deploy Test AWS, lo que permite la ejecución de pruebas de manera automatizada.

En este pipeline, se estableció un sistema de versionado para gestionar las distintas etapas del ciclo de vida del software.

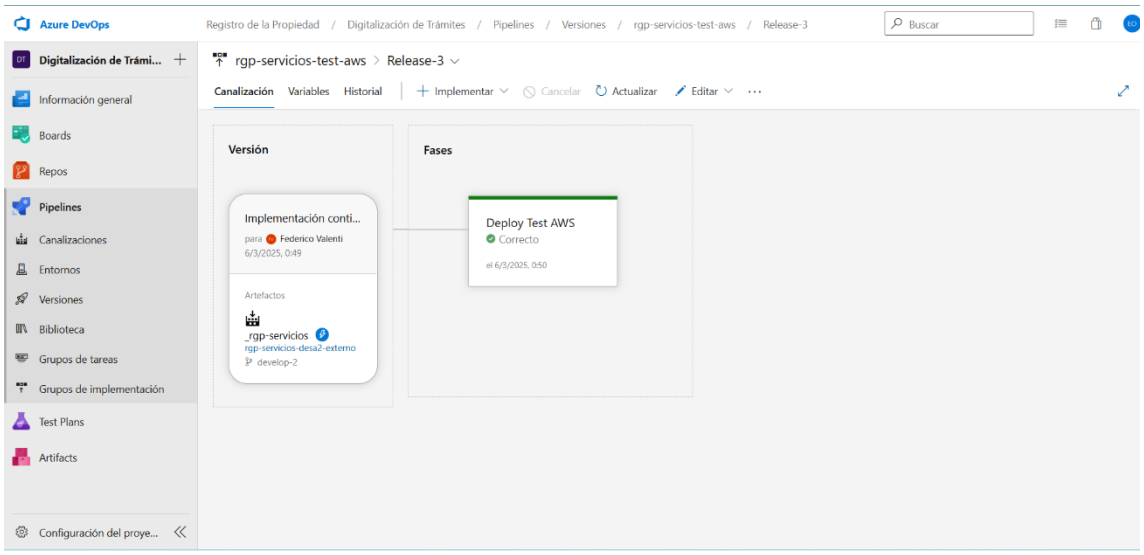
Durante el proceso de despliegue, se validaron (a modo de ejemplo) las versiones Release-1 (21/02/2025), Release-2 (05/03/2025) y Release-3 (06/03/2025), asegurando que cada modificación realizada en el código pase por un proceso riguroso de pruebas antes de su promoción a preproducción.



Ver imagen adjunta “validación de ejecución AZURE - AWS.jpg”

Con el objetivo de verificar la correcta ejecución de los despliegues en el entorno de Testing, se validó la ejecución de los pipelines de CI/CD, asegurando que cada versión fuera desplegada y probada automáticamente en el ambiente de pruebas. La última versión validada, Release-3, fue desplegada exitosamente en el ambiente de Testing el 06/03/2025 a las 00:50, dentro del pipeline rgp-servicios-test-aws.

El flujo del pipeline se estructuró en distintas fases para garantizar la calidad del software en cada etapa. Primero, se realizó la compilación del código fuente en Azure DevOps, asegurando la integridad y compatibilidad del código. Posteriormente, se ejecutaron pruebas automatizadas utilizando herramientas como JUnit y Postman, verificando el correcto funcionamiento de las funcionalidades implementadas. Una vez superadas estas pruebas, se procedió con el despliegue en AWS, utilizando CodeDeploy y Helm Charts en Kubernetes (EKS), asegurando que las aplicaciones fueran correctamente desplegadas en la infraestructura de Testing.Con este proceso, cada nueva versión del sistema es evaluada en un entorno que replica la infraestructura de producción, permitiendo minimizar riesgos y asegurar la estabilidad del sistema antes de su implementación final. La integración del ambiente de Testing en la cadena de CI/CD ha optimizado la calidad del software, mejorando la detección de errores y facilitando una transición más segura hacia la preproducción y producción.



Ver imagen adjunta “validación de ejecución AZURE - AWS.jpg”

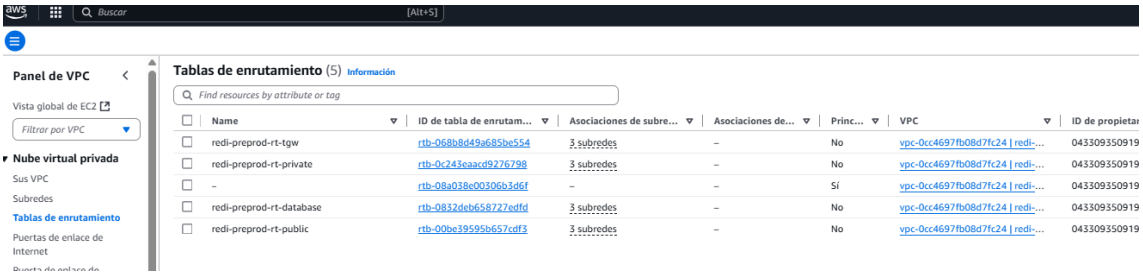
Configuración y Ajuste del Ambiente de Preproducción

El ambiente de preproducción fue configurado con el objetivo de replicar con la mayor fidelidad posible el entorno productivo, permitiendo así validar el comportamiento del sistema antes de su despliegue final. Para ello, se implementó la VPC "redi-preprod-vpc", con un rango de red CIDR 172.28.192.0/20, asegurando una segmentación de red mediante subredes públicas y privadas. Esta configuración garantiza la seguridad, el aislamiento de los recursos y el control del tráfico de red.

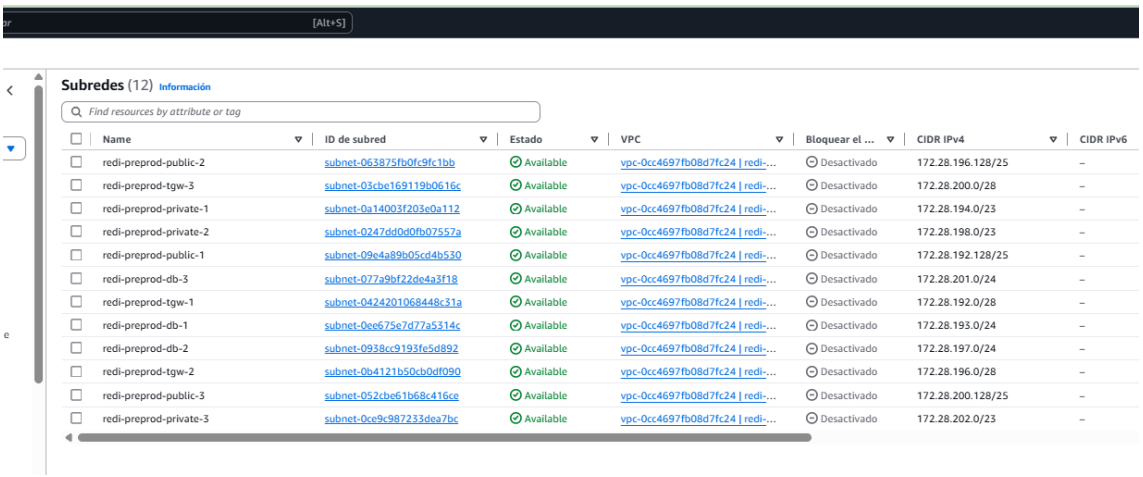
Ver imagen adjunta “VPC preproducción - AWS.jpg”

Para lograr una posible infraestructura de producción, se configuraron diversos elementos dentro de AWS. La VPC "redi-preprod-vpc" fue establecida con el ID vpc-0cc4697fb08d7fc24 y el rango de red previamente definido. Su estado se encuentra disponible y cuenta con la resolución de DNS habilitada, lo que facilita la gestión de nombres de dominio dentro del entorno de preproducción. Además, se asignó como ACL de red principal el identificador acl-05c5132fd1703f878, proporcionando una capa adicional de seguridad en el tráfico de red.

En cuanto a la configuración de acceso y conectividad, se deshabilitó el acceso público para evitar conexiones no autorizadas, asegurando así un control más estricto sobre los recursos internos. Se definieron grupos de reglas en el firewall de Route 53 Resolver, restringiendo accesos conforme a los requerimientos de seguridad del sistema. También se establecieron opciones avanzadas de DHCP (Dynamic Host Configuration Protocol) bajo la configuración dopt-0a5e14d6219eb594c, lo que permite la asignación dinámica y controlada de direcciones IP a los servicios dentro de la VPC. Por último, se configuró la tabla de enrutamiento principal rtb-08a030c068c36d86f, asegurando una correcta distribución del tráfico de red y permitiendo la conectividad entre los distintos componentes del sistema.



Ver imagen adjunta “Enrutamiento preproducción - AWS”



Ver imagen adjunta “Subredes preproducción - AWS.jpg”

Esta infraestructura permite un control detallado sobre el tráfico de red, asegurando que únicamente las instancias y servicios internos tengan acceso a los recursos de la VPC de preproducción. Con esta configuración, el sistema se encuentra preparado para realizar pruebas previas al despliegue en

producción, garantizando estabilidad, seguridad y un rendimiento óptimo en cada fase del proceso.

Integración del Ambiente de Preproducción a la Integración Continua

La integración del entorno de preproducción en la cadena de Integración Continua (CI) se llevó a cabo mediante la configuración de pipelines avanzados en Azure DevOps, con el objetivo de automatizar y optimizar los procesos de despliegue, prueba y validación del sistema. Estos pipelines fueron diseñados para gestionar la implementación de imágenes Docker, la ejecución de pruebas de regresión y la validación de seguridad, asegurando que cada nueva versión del sistema cumpliera con los estándares requeridos antes de su despliegue en producción.

El pipeline CI/CD implementado incluyó procesos automatizados para la validación del código fuente, pruebas unitarias y de integración en un entorno que replicaba fielmente la infraestructura productiva. De esta manera, cada iteración del software se sometió a un conjunto de verificaciones previas al despliegue, lo que permitió identificar errores en etapas tempranas del desarrollo. Además, se establecieron pruebas de regresión automatizadas para garantizar la estabilidad del sistema frente a cambios, evitando que nuevas funcionalidades afectaran negativamente a las existentes.

Los proyectos rgp-web-backend, portal-backend, rgp-pagos-online y portal-siraut-frontend fueron integrados dentro de la estructura de CI/CD, asegurando que cada componente del sistema pudiera desplegarse en el ambiente de preproducción y someterse a pruebas antes de su migración a producción. Estos despliegues permitieron validar la conectividad entre servicios internos, la interoperabilidad con sistemas externos y la correcta funcionalidad de las aplicaciones en un entorno similar al productivo.

En términos de seguridad, se implementaron políticas de acceso restringido mediante el servicio IAM (Identity and Access Management) de AWS. Estas configuraciones aseguraron que únicamente los pipelines autorizados y usuarios con permisos específicos pudieran acceder a los recursos del entorno de preproducción, protegiendo la integridad y confidencialidad de los datos utilizados en las pruebas.

La incorporación del entorno de preproducción en la cadena CI/CD ha optimizado significativamente la calidad del sistema, permitiendo la detección y corrección de errores de manera proactiva. Como resultado, se ha logrado una

transición más eficiente y segura hacia el entorno productivo, asegurando estabilidad operativa y reduciendo riesgos en cada despliegue.

Reingeniería del Proyecto Productivo del Registro de la Propiedad

Como parte del proceso de migración e integración del sistema en AWS, se llevó a cabo una reingeniería integral del entorno productivo del Registro de la Propiedad. Esta tarea, no contemplada inicialmente dentro del alcance del proyecto, resultó crítica para garantizar el correcto funcionamiento del portal web y sus servicios asociados en la nueva infraestructura.

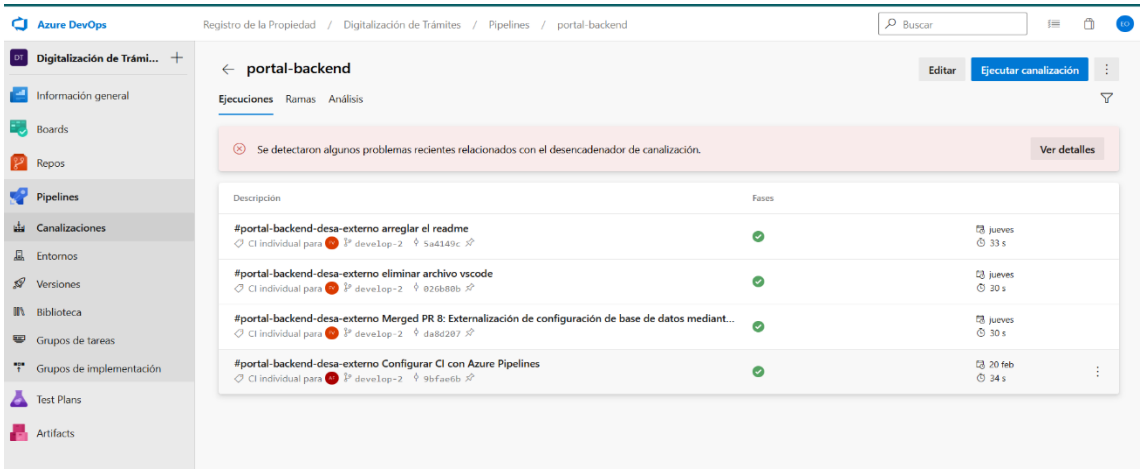
El objetivo principal fue eliminar la dependencia de Oracle WebLogic, normalizando la arquitectura para operar de forma nativa en AWS y optimizando el rendimiento, escalabilidad e interoperabilidad de los componentes. Para lograrlo, se trabajó en la adecuación y migración de los principales módulos del sistema, incluyendo rgp-web-backend, portal-backend, rgp-pagos-online y portal-siraut-frontend.

La refactorización del código permitió su adaptación a un entorno cloud-native, eliminando dependencias con la infraestructura on-premise. Asimismo, se implementó la “containerización” con Docker para asegurar la modularidad y portabilidad del sistema, y se desplegó en Kubernetes (EKS) en AWS, lo que garantiza alta disponibilidad y escalabilidad automática. Además, se optimizó la interoperabilidad con sistemas externos, asegurando la compatibilidad y continuidad operativa del portal web.

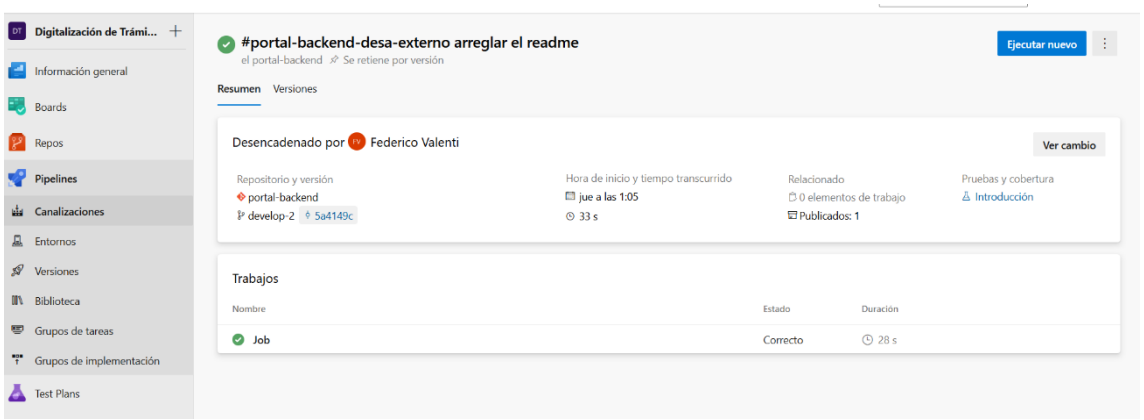
Para gestionar el ciclo de vida de desarrollo y despliegue, se implementaron pipelines de CI/CD en Azure DevOps. Estos procesos permitieron la normalización del código en la rama develop-2, consolidando las actualizaciones de cada módulo y automatizando el flujo de integración y despliegue, lo que redujo significativamente los tiempos de entrega. A su vez, se configuró la ejecución de pruebas automatizadas en cada actualización y se establecieron despliegues estructurados en ambientes de Testing y preproducción, asegurando la estabilidad del sistema antes de su paso a producción.

El flujo de los pipelines fue diseñado para garantizar la calidad y eficiencia del software, iniciando con la extracción del código fuente desde Azure Repos, seguido por la compilación y análisis de calidad con SonarQube. Posteriormente, se ejecutaron pruebas unitarias y de integración utilizando herramientas como JUnit y Postman, para luego realizar el despliegue automatizado en AWS EKS. Finalmente, se habilitó el monitoreo del estado del despliegue con AWS

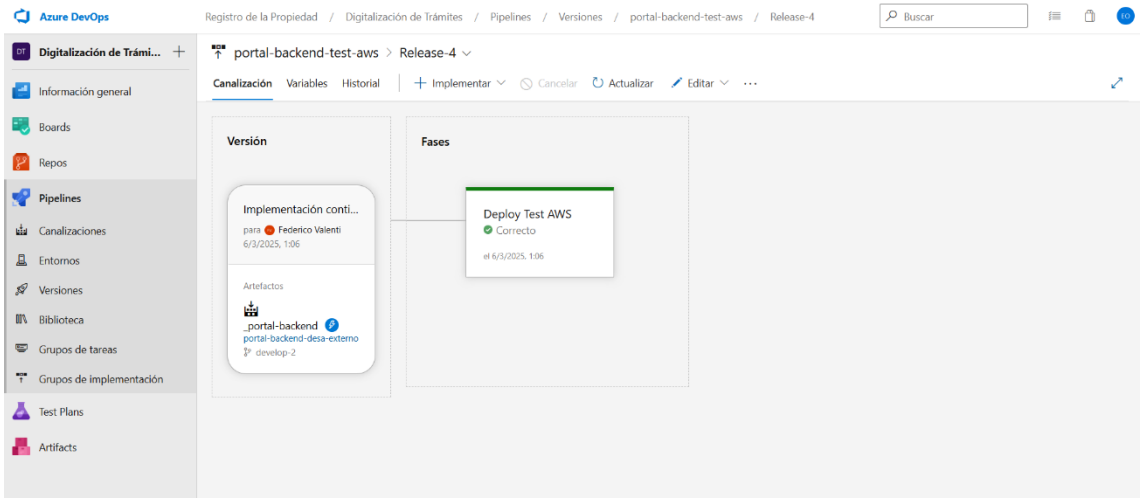
CloudWatch y Azure Monitor, permitiendo la supervisión continua del rendimiento del sistema.



Ver imagen adjunta “Pipelines AZURE - AWS 4.jpg”



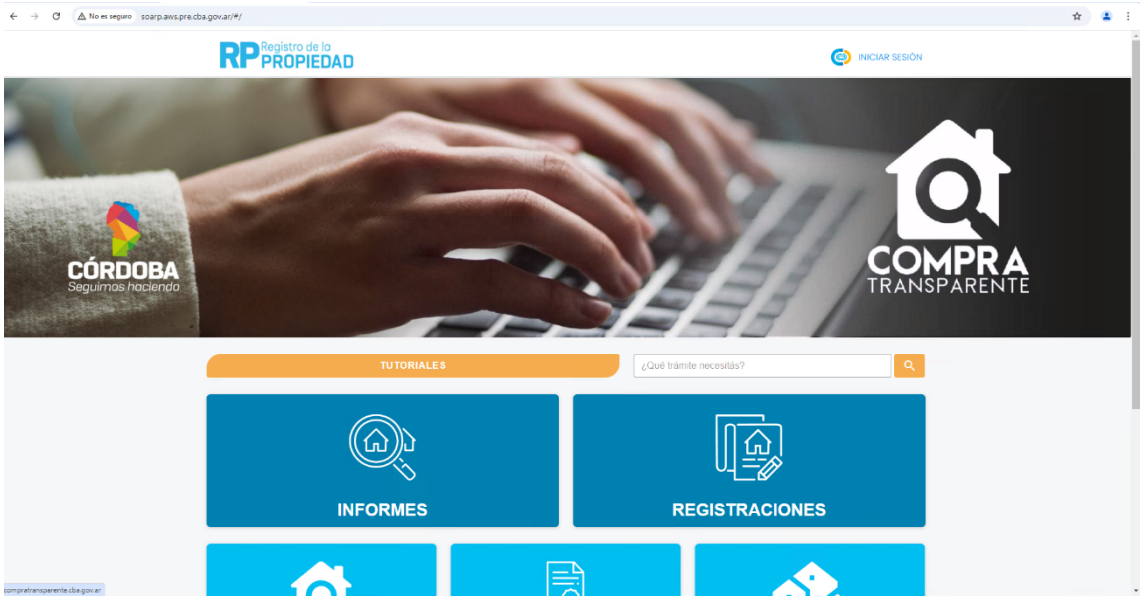
Ver imagen adjunta “Pipelines AZURE - AWS 5.jpg”



Ver imagen adjunta “Pipelines AZURE - AWS 6.jpg”

Una vez desplegados los proyectos en los ambientes de testing y preproducción, se realizaron pruebas de performance y latencia para verificar la conectividad entre los servicios migrados y on-premise. Además, se evaluó el tiempo de respuesta de las APIs bajo carga operativa real, se probaron las capacidades de escalabilidad en Kubernetes y se midió la estabilidad del sistema ante un alto volumen de transacciones. Estas pruebas demostraron mejoras significativas en la eficiencia del sistema, reduciendo la latencia y optimizando la velocidad de procesamiento de solicitudes.

Ver imagen adjunta “Portal web - test - AWS.jpg”



Ver imagen adjunta “Portal web - preproducción - AWS.jpg”

Como resultado de la reingeniería, el sistema ahora es completamente compatible con AWS, eliminando la dependencia de tecnologías legadas. Se logró optimizar el tiempo de respuesta y procesamiento de datos, automatizar el despliegue y pruebas con pipelines CI/CD en Azure DevOps, y establecer una infraestructura preparada para futuras ampliaciones y mayor demanda operativa. Con esta transformación, el Registro de la Propiedad cuenta con un sistema modernizado, seguro, escalable y eficiente, listo para evolucionar y adaptarse a nuevas necesidades tecnológicas.

CUMPLIMIENTO DE HITOS

El proyecto "Análisis e Instalación de Ambientes No Productivos en Arquitectura de Cloud para el Aplicativo REDI" ha avanzado conforme al plan establecido, logrando el cumplimiento de los hitos clave en el proceso de evaluación y diseño de infraestructura en la nube. A lo largo de esta etapa, se han completado tareas fundamentales relacionadas con la definición, configuración y validación de entornos no productivos en AWS, garantizando una visión integral sobre la viabilidad y beneficios de la nube en el ecosistema REDI.

Los hitos alcanzados incluyen:

1. Análisis de la Arquitectura Actual y Evaluación del Estado Inicial:
   Se realizó un estudio detallado de la infraestructura on-premise del sistema REDI, identificando limitaciones en escalabilidad, mantenimiento y eficiencia operativa. Este diagnóstico sirvió de base para el diseño de una arquitectura en la nube más flexible y segura.

2. Definición de la Infraestructura en la Nube:
   Se diseñó un modelo de infraestructura en AWS que incorpora VPC segmentadas, Kubernetes (EKS con Fargate), RDS Proxy, balanceadores de carga (ALB/NLB), WAF y AWS Secrets Manager, con el objetivo de optimizar la seguridad y disponibilidad del sistema.

3. Instalación y Configuración del Ambiente de Desarrollo:
   Se configuró un entorno de desarrollo en AWS, incluyendo la implementación de instancias EC2, segmentación de red en subredes privadas y públicas, reglas de seguridad en IAM y monitoreo mediante AWS CloudWatch. Este entorno permitió la validación de configuraciones y pruebas iniciales sobre el sistema.

4. Instalación, Configuración y Ajuste del Ambiente de Testing:
   Se estableció el ambiente de pruebas en AWS, replicando la infraestructura de desarrollo pero con configuraciones específicas para la ejecución de pruebas funcionales, de integración y de rendimiento. Se habilitó la conectividad con los sistemas on-premise y se realizaron pruebas de interoperabilidad.

5. Configuración y Ajuste del Ambiente de Preproducción:
   Se creó un entorno que simula las condiciones del ambiente productivo, garantizando la evaluación del rendimiento y estabilidad del sistema antes

de cualquier posible migración. Se integraron servicios de seguridad avanzados, como Web Application Firewall (WAF) y RDS Proxy, asegurando la protección del tráfico y optimización de las conexiones con bases de datos.

6. Integración con la Infraestructura de DevOps:
   Se configuraron pipelines en Azure DevOps para la automatización de compilación, pruebas y despliegues, permitiendo una validación estructurada de cada versión del software. Se implementaron procesos de CI/CD que facilitaron la detección temprana de errores y mejoraron la calidad del sistema.

7. Validación y Pruebas en Entornos No Productivos:
   Se realizaron pruebas en los entornos de desarrollo, Testing y preproducción, evaluando el impacto de la infraestructura en la nube y su capacidad de soportar el sistema REDI. Se verificó la conectividad con sistemas externos y se analizaron métricas de rendimiento.

El cumplimiento de estos hitos ha permitido consolidar un análisis detallado sobre la viabilidad de la nube en la gestión de entornos no productivos del sistema REDI, proporcionando información clave para la toma de decisiones sobre futuras implementaciones en producción.

DESVIACIONES Y AJUSTES

Durante la ejecución del proyecto "Análisis e Instalación de Ambientes No Productivos en Arquitectura de Cloud para el Aplicativo REDI", se identificaron ciertas desviaciones con respecto al plan original. Estas variaciones estuvieron principalmente relacionadas con la disponibilidad de infraestructura, la configuración de servicios críticos y la integración con sistemas on-premise. Para mitigar estos inconvenientes, se implementaron ajustes estratégicos que permitieron continuar con el desarrollo del análisis dentro del cronograma establecido.

Inicialmente, la configuración de los entornos de desarrollo, Testing y preproducción debía ejecutarse en un tiempo determinado. Sin embargo, se presentaron retrasos debido a la disponibilidad limitada de recursos en AWS, lo que afectó la provisión de instancias y servicios esenciales. Para contrarrestar este problema, se adoptó una estrategia de optimización de recursos, ajustando la cantidad y tipo de instancias EC2 utilizadas, así como la gestión de cuotas y permisos en AWS para acelerar los despliegues.

Durante la implementación de los entornos, se detectaron diferencias en la configuración de Web Application Firewall (WAF) y en las políticas de acceso definidas en IAM (Identity and Access Management), lo que requirió ajustes en las reglas de seguridad y en la segmentación de redes privadas y públicas. Se establecieron políticas más restrictivas para evitar accesos no autorizados y se optimizaron los grupos de seguridad para permitir únicamente el tráfico necesario entre los servicios de la plataforma.

Uno de los desafíos más relevantes fue la integración del entorno en la nube con los sistemas gubernamentales existentes. La conectividad entre AWS y las bases de datos Oracle alojadas on-premise presentó problemas de latencia y estabilidad. Para abordar esta situación, se implementó AWS RDS Proxy, optimizando la gestión de conexiones y asegurando un rendimiento adecuado en la comunicación entre los sistemas locales y los nuevos entornos en la nube.

A pesar de las desviaciones detectadas, los ajustes aplicados permitieron que el análisis se desarrollara dentro del marco de tiempo planificado, sin comprometer la calidad del estudio ni la estabilidad de los entornos desplegados. Estas correcciones fortalecieron la arquitectura propuesta, asegurando un diseño más robusto, seguro y alineado con los objetivos de escalabilidad y automatización del sistema REDI.

CONCLUSION

El proyecto "Análisis e Instalación de Ambientes No Productivos en Arquitectura de Cloud para el Aplicativo REDI" ha permitido evaluar en profundidad la viabilidad de trasladar los entornos de desarrollo, Testing y preproducción a una infraestructura basada en la nube, específicamente en Amazon Web Services (AWS). Durante el desarrollo del proyecto, se analizaron las ventajas y desafíos de esta arquitectura, considerando factores clave como escalabilidad, seguridad, automatización y eficiencia operativa.

A través de la implementación y validación de entornos no productivos, se logró establecer una infraestructura replicable que facilita la ejecución de pruebas exhaustivas sin comprometer la estabilidad del sistema en producción. Se diseñó una arquitectura basada en VPC segmentadas, Kubernetes con EKS y Fargate, RDS Proxy, balanceadores de carga y Web Application Firewall (WAF), garantizando una mejor gestión del tráfico, mayor seguridad y un desempeño optimizado de las aplicaciones.

Además, la automatización de procesos mediante Azure DevOps permitió estructurar un flujo de CI/CD que facilita la integración y el despliegue de nuevas versiones, asegurando la detección temprana de errores y la validación continua del sistema en entornos controlados. La infraestructura desplegada permitió ejecutar pruebas funcionales, de integración y de rendimiento, confirmando la estabilidad de los componentes clave del aplicativo REDI en la nube.

El análisis realizado confirma que la arquitectura en la nube es una alternativa viable para la gestión de los entornos no productivos del sistema REDI, proporcionando un modelo más eficiente y seguro en comparación con la infraestructura tradicional. Sin embargo, cualquier futura transición a un ambiente productivo deberá considerar aspectos adicionales como costos operativos, gobernanza del entorno cloud y estrategias de contingencia para garantizar una migración efectiva y sin interrupciones del servicio.

Desde South Hive, es nuestro deseo poder seguir contribuyendo a proyectos con un fuerte impacto social, promoviendo la vinculación entre actores públicos y privados para generar, en última instancia, una mejor experiencia para los ciudadanos.