



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

Migración del sistema web ePlanning de una arquitectura monolítica a una arquitectura de microservicios en la nube para una corporación de belleza

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Héctor Jesús NORIEGA YARLEQUÉ

ASESOR

Armando David ESPINOZA ROBLES

Lima, Perú

2022



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Noriega, H. (2022). *Migración del sistema web ePlanning de una arquitectura monolítica a una arquitectura de microservicios en la nube para una corporación de belleza*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Software]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Héctor Jesús Noriega Yarlequé
Tipo de documento de identidad	DNI
Número de documento de identidad	70569363
URL de ORCID	https://orcid.org/0000-0002-8872-1405
Datos de asesor	
Nombres y apellidos	Espinoza Robles Armando David
Tipo de documento de identidad	DNI
Número de documento de identidad	08633326
URL de ORCID	https://orcid.org/0000-0001-5671-5927
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Trujillo Trejo John Ledgard
Tipo de documento	DNI
Número de documento de identidad	06187585
Miembro del jurado 1	
Nombres y apellidos	Jorge Luis Chávez Soto
Tipo de documento	DNI
Número de documento de identidad	08675814
Miembro del jurado 2 (Asesor)	
Nombres y apellidos	Espinoza Robles Armando David
Tipo de documento	DNI
Número de documento de identidad	08633326
Datos de investigación	
Línea de investigación	No aplica

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2022
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Software

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 20:05 horas del día 26 de agosto del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Lic. Trujillo Trejo John Ledgard (Presidente), Lic. Chávez Soto Jorge Luis (Miembro) y el Lic. Espinoza Robles Armando David (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/nvd-sypk-tbu>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“MIGRACIÓN DEL SISTEMA WEB EPLANNING DE UNA ARQUITECTURA MONOLÍTICA A UNA ARQUITECTURA DE MICROSERVICIOS EN LA NUBE PARA UNA CORPORACIÓN DE BELLEZA”**, por el Bachiller Noriega Yarlequé Héctor Jesús; para obtener el Título Profesional de Ingeniero de Software.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

El Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, el Bachiller obtuvo la nota de **18 (DIECIOCHO)**.

A continuación el Presidente de Jurados el Lic. Trujillo Trejo John Ledgard, declara al Bachiller **Ingeniero de Software**.

Siendo las 20:55 horas, se levantó la sesión.

Presidente

Lic. Trujillo Trejo John Ledgard

Miembro

Lic. Chávez Soto Jorge Luis

Miembro Asesor

Lic. Espinoza Robles Armando David



Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería Software

INFORME DE EVALUACIÓN DE ORIGINALIDAD
Nº 022-EPISW-FISI-2022

1. Autoridad Académica que emite el Informe de Originalidad:	Directora de la Escuela Profesional de Ingeniería de Software
2. Apellidos y Nombres de la autoridad académica:	Dra. Nora Bertha La Serna Palomino
3. Operador del programa informático de similitudes:	Dra. Nora Bertha La Serna Palomino
3. Documento evaluado:	Tesis para Pregrado Título: “Migración del sistema web ePlanning de una arquitectura monolítica a una arquitectura de microservicios en la nube para una corporación de belleza”
5. Autores del documento:	Noriega Yarlequé, Héctor Jesús
6. Fecha de recepción de documento	Recepción: 10/09/2022
7. Fecha de aplicación del programa detector de similitudes:	Revisión: 10/09/2022
8. Software utilizado:	Turnitin
9. Configuración del programa detector de similitudes:	Excluye textos entrecomillados: Sí Excluye biografías: Sí Excluye cadenas menores a 40 palabras: Sí Otro criterio (especificar): No
10. Porcentaje de similitudes según programa detector de similitudes	Seis por ciento (6%)
11. Fuentes originales de las similitudes encontradas	Se adjuntan en 01 (una) foja al presente informe
12. Observaciones:	Ninguna
13. Calificación de originalidad i. Documento cumple criterios de originalidad, sin observaciones ii. Documento cumple criterios de originalidad, con observaciones iii. Documento no cumple criterios de originalidad.	Documento cumple criterio de originalidad, sin observación
14. Fecha del Informe:	14/11/2022

Dra. Nora Bertha La Serna Palomino
Directora (e) de la EPISW

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMATICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

Migración del sistema web ePlanning de una arquitectura monolítica a una arquitectura de microservicios en la nube para una corporación de belleza

Autor: Noriega Yarlequé, Héctor Jesús

Asesor: Espinoza Robles, Armando David

Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de Ingeniero de Software

Fecha: mayo de 2022

RESUMEN

El presente informe de experiencia profesional describe el desarrollo del proyecto para la migración del sistema web ePlanning del área de planeamiento de una corporación de belleza, de una arquitectura monolítica a una arquitectura basada en microservicios sin servidor. La problemática que acontece en la corporación es que, para maximizar sus ventas, se necesitaba elaborar ofertas más específicas para sus consultoras de acorde a sus perfiles de compra y debido a esto el tamaño y número de cargas que se realizaban en ePlanning aumentaron sustancialmente viéndose afectado tanto en rendimiento como en disponibilidad, entonces nació la necesidad de migrar a una arquitectura escalable, con alto rendimiento y disponibilidad. Bajo la metodología SCRUM se migró a una arquitectura en la nube basada en microservicios y sin servidor, gracias a esta implementación se mejoraron los tiempos en la planificación, se tuvo un sistema con alto rendimiento, escalabilidad y se redujeron los costos de operación.

Palabras clave: Planeamiento, consultora, microservicios, nube, sin servidor.

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS
FACULTY OF SYSTEMS AND INFORMATICS ENGINEERING
PROFESSIONAL SCHOOL OF SOFTWARE ENGINEERING

**Migration of ePlanning web system from a monolithic architecture to a
microservices-based architecture in the cloud for a makeup corporation.**

Author: Noriega Yarlequé Héctor Jesús

Advisor: Espinoza Robles Armando David

**Title: Professional Sufficiency Work to opt for the Professional Title of Software
Engineer**

Date: May de 2022

ABSTRACT

This professional experience report describes the development of the project for the migration of the ePlanning web system of the planning area of a beauty corporation, from a monolithic architecture to an architecture based on serverless microservices. The problem that occurs in the corporation is that to maximize sales, it was necessary to develop more specific offers for their consultants according to their buying profiles and because of this the size and number of loads that were made in ePlanning increased substantially being affected both in performance and availability, then was born the need to implement a scalable system, with high performance and availability that can support the new load, and be prepared for the increase of loads that was planned for the future. Under the SCRUM methodology, a solution was designed and implemented in the cloud focused on a microservices-based architecture and without server, thanks to this implementation, planning times were improved, a service with high performance, scalability and reduced operating costs.

Keywords: Planning, consulting firm, microservices, cloud, serverless.

TABLA DE CONTENIDO

RESUMEN	III
ABSTRACT	IV
INTRODUCCIÓN	1
CAPÍTULO I - TRAYECTORIA PROFESIONAL	3
CAPÍTULO II: CONTEXTO EN EL QUE SE DESARROLLÓ LA EXPERIENCIA	6
2.1 EMPRESA- ACTIVIDAD QUE REALIZA	6
2.2 VISIÓN.....	6
2.3 MISIÓN.....	6
2.4 ORGANIZACIÓN DE LA EMPRESA	7
2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS.....	7
2.6 EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN.....	8
CAPÍTULO III - ACTIVIDADES DESARROLLADAS	9
3.1. SITUACIÓN PROBLEMÁTICA.....	9
3.1.1. <i>DEFINICIÓN DEL PROBLEMA:</i>	9
3.2 SOLUCIÓN.....	10
3.2.1. <i>OBJETIVOS</i>	10
3.2.2 <i>ALCANCE</i>	10
3.2.3 <i>ETAPAS Y METODOLOGÍAS</i>	11
3.2.4 <i>FUNDAMENTOS UTILIZADOS</i>	12
3.2.5 <i>IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS</i>	24
3.3 EVALUACIÓN	55
3.3.1 <i>EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO - BENEFICIO</i>	55
3.3.2 <i>Interpretación del VAN y del TIR</i>	57
CAPÍTULO IV: REFLEXIÓN CRÍTICA DE LA EXPERIENCIA	59
CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES	60
5.1 CONCLUSIONES.....	60
5.2 RECOMENDACIONES	60
5.3 FUENTES DE INFORMACIÓN	60
5.4 GLOSARIO	61
ANEXOS.....	62

INDICE DE FIGURAS

Figura 1.....	7
Figura 2.....	28
Figura 3.....	30
Figura 4.....	32
Figura 5.....	32
Figura 6.....	33
Figura 7.....	34
Figura 8.....	35
Figura 9.....	35
Figura 10.....	39
Figura 11.....	41
Figura 12.....	42
Figura 13.....	46
Figura 14.....	47
Figura 15.....	47
Figura 16.....	48
Figura 17.....	49
Figura 18.....	49
Figura 19.....	50
Figura 20.....	50
Figura 21.....	51

Figura 22.....	51
Figura 23.....	52
Figura 24.....	52
Figura 25.....	52
Figura 26.....	53
Figura 27.....	53
Figura 28.....	54

INDICE DE TABLAS

Tabla 1.....	3
Tabla 2.....	3
Tabla 3.....	4
Tabla 4.....	13
Tabla 5.....	14
Tabla 6.....	14
Tabla 7.....	14
Tabla 8.....	25
Tabla 9.....	26
Tabla 10.....	27
Tabla 11.....	28
Tabla 12.....	30
Tabla 13.....	30
Tabla 14.....	31
Tabla 15.....	31
Tabla 16.....	32
Tabla 17.....	32
Tabla 18.....	33
Tabla 19.....	33
Tabla 20.....	34
Tabla 21.....	34

Tabla 22.....	35
Tabla 23.....	36
Tabla 24.....	37
Tabla 25.....	37
Tabla 26.....	38
Tabla 27.....	38
Tabla 28.....	39
Tabla 29.....	40
Tabla 30.....	42
Tabla 31.....	45
Tabla 32.....	45
Tabla 33.....	54
Tabla 34.....	55
Tabla 35.....	56
Tabla 36.....	56
Tabla 37.....	57
Tabla 38.....	57
Tabla 39.....	58
Tabla 40.....	59

INTRODUCCIÓN

El presente trabajo describe la migración del sistema web ePlanning de una arquitectura monolítica a una arquitectura basada en microservicios sin servidor en la nube.

El trabajo se realizó en una corporación multinacional de belleza con más de 50 años de experiencia en la venta directa, cuenta con tres marcas, y está ubicada en 15 países diferentes. El área de planeamiento digital es aquella que se encarga de planificar para cada país y campaña las diferentes ofertas que serán ofrecidas a las consultoras para su venta.

Desde el año 2015, ePlanning, un sistema web monolítico y desarrollado bajo el patrón de diseño MVC, era el sistema principal del área de planeamiento, desde este sistema se realizan las cargas de las planificaciones de todas las ofertas y productos a vender. El planner (usuarios del sistema encargados de realizar la planificación de ofertas) previamente habiendo recibido inputs de otros sistemas de historial de ofertas, fuerza de ventas, etc.; llenaban manualmente en una plantilla de Excel las ofertas por campaña y país, esta plantilla era cargada a ePlanning el cual era el encargado de realizar diferentes operaciones como cálculos de precios, cálculo de descripciones comerciales, cálculos de tipos de Oferta, así también como la comunicación con otros sistemas como eran SICC (Software para la obtención del CUV - Código Único de Venta) o CDI (Software de composición de Imágenes).

Para el año 2019 con el fin de maximizar el poder de venta, nació la necesidad de elaborar ofertas más específicas para cada consultora basándose en su historial de compra, aumentando en gran cantidad las ofertas planificadas, y debido a esta granularidad en la composición de ofertas las cargas eran cada vez más grandes, el cual se veía afectado en rendimiento y disponibilidad por la arquitectura monolítica, se pasó por varias iteraciones para optimizarlo, pero con la proyección para el aumento de carga en algún momento el sistema ya no aguantaría.

Es así como nace la necesidad de migrar a una arquitectura que permita soportar estas cargas con grandes cantidades de ofertas y no se vea afectado en rendimiento,

manteniendo así la continuidad del negocio y generando ahorros en los costos de operación. A continuación, se detalla la estructura del trabajo:

- En el CAPÍTULO I, se explica mi trayectoria en el ambiente profesional, así como aquellas funciones y actividades realizadas en los diferentes lugares donde he ejercido, los estudios y conocimientos que he adquirido a lo largo de esta y que me permitieron diseñar e implementar soluciones como la expuesta en este informe.
- En el CAPÍTULO II, se detalla datos de la corporación de belleza para la cual se implementó esta migración y se detalla su organigrama; así como su misión, visión , logros e impacto en la sociedad peruana.
- En el CAPÍTULO III, se explica detalladamente el análisis, diseño e implementación realizado para la migración de sistema ePlanning a una arquitectura basada en microservicios sin servidor.
- En el CAPÍTULO IV, se realiza un análisis del proyecto implementado y los pasos por las diferentes fases desde su concepción hasta la puesta en producción.
- En el CAPÍTULO V, se desarrolla las conclusiones finales del proyecto y recomendaciones para tomar en cuenta para una implementación similar.

CAPÍTULO I - TRAYECTORIA PROFESIONAL

PRESENTACION PROFESIONAL

Soy **BACHILLER** en **INGENIERÍA DE SOFTWARE** con más de 7 años de experiencia en desarrollo de software, gran habilidad para el análisis y diseño de soluciones, enfocado en las buenas prácticas de desarrollo de software, dentro de mi experiencia he trabajado en proyectos desarrollados en diferentes lenguajes, puestas a producción a gran escala en diferentes sectores empresariales como son transporte marítimo, seguros de vida, seguridad informática, telecomunicaciones, belleza y actualmente en infraestructura de pagos. Cuento con un alto nivel de responsabilidad, con capacidades altas de autoaprendizaje, proactividad y eficiencia para el trabajo en equipo.

FORMACIÓN ACADÉMICA

Tabla 1 - Formación académica profesional

FORMACIÓN ACADÉMICA PROFESIONAL

Formación	Institución	Periodo
Bachiller en Ingeniería de Software	Universidad Nacional Mayor de San Marcos - Facultad de Ingeniería de Sistemas e Informática - Escuela Académico Profesional de Ingeniería de Software.	2009 - 2016

Nota: Autoría propia.

Tabla 2 - Cursos

CURSOS REALIZADOS

Curso	Institución	Año
AWS Certified Solutions Architect - Associate (SAA-C02)	Udemy	2022
AWS Certified Solutions Architect - Associate (SAA-C02)	BSG Institute	2019
Visual Studio 2015 Web Developer	Cibertec	2016

Nota: Autoría propia.

EXPERIENCIA PROFESIONAL

Tabla 3 - Experiencia profesional

KUSHKI PAGOS Cargo: Senior Software Engineer Mayo 2021 - Actualidad	Desarrollo backend de microservicios para sistema Backoffice. Desarrollo frontend para sistema Backoffice. Implementación de ETLs para reportería de sistema transaccional. Mantenimiento y soporte en la atención de tickets por bugs detectados. Tecnologías utilizadas: Node.js con Typescript, React, Golang, Scala, AWS.
BELCORP - INETUM PERU Cargo: Senior Software Engineer Julio 2018 - Mayo 2021	Mantenimiento e implementación de features para sistema web ePlanning. Diseño e implementación para la migración de sistema legado ePlanning a una arquitectura de microservicios sin servidor. Diseño e implementación de solución para algoritmo de composición de imágenes, para sistema CDI. Tecnologías utilizadas: ASP.Net, C#, .Net Core 3.1, SQL Server, Oracle, Aurora, AWS.
Tata Consultancy Services – Claro Perú Cargo: Software Engineer Junio 2017 – Julio 2018.	Desarrollo y mantenimiento de sistemas de ventas SISACT, SISACT Prepago, SICAR, etc. Desarrollo de sistema de Levantamiento Secreto de Telecomunicaciones LST. Elaboración de documento de arquitectura para proyecto de consulta de huella a RENIEC. Tecnologías utilizadas: ASP.Net, C#, VB. Net, SQL Server, Oracle.
Teamsoft SAC – Claro Perú Cargo: Software Engineer	Desarrollo y mantenimiento de sistemas de ventas SISACT, SISACT Prepago, SICAR,

Enero 2017 – Junio 2017	etc. Tecnologías utilizadas: ASP.Net, C#, VB. Net, SQL Server, Oracle.
SAETA- Soluciones Aplicadas en Tecnología Avanzada Cargo: Analista Programador Agosto 2016 – Diciembre 2016	Diseño e implementación de software para control de accesos mediante la huella, Pass Identity. Implementación de portal de pólizas electrónicas, E-Poliza. Tecnologías utilizadas: ASP.Net, C#, VB. Net, SQL Server, PostgreSQL.
Materia Gris S.A.C – La Positiva Seguros Analista Programador Junior Mayo 2015 – Agosto 2016	Implementación de proyecto Minisites, servicios web SOAP para venta de pólizas por internet. Mantenimiento al sistema web Oficina Virtual. Tecnologías utilizadas: C#, Oracle, Informix, WCF.
APM Terminals Callao Practicante Pre Profesional IT – Aplicaciones Febrero 2014 – Diciembre 2014	Mantenimiento e implementación de sistemas internos: Intranet, Extranet, etc. Tecnologías utilizadas: ASP.Net, C#, VB. Net, SQL Server.
Nota: Autoría propia.	

CAPÍTULO II: CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1 EMPRESA- ACTIVIDAD QUE REALIZA

El presente trabajo de suficiencia profesional se implementó en una corporación de belleza con presencia en más de 13 países y que cuenta más de 50 años de experiencia en la venta directa mediante sus consultoras. Desde sus inicios, han enfocado su negocio en empoderar a millones de mujeres buscando la independencia económica a través de la venta de productos. Actualmente, esta corporación cuenta con varios reconocimientos a nivel mundial y Latinoamérica. En este contexto se maneja en 5 ejes transversales para hacer realidad su compromiso con la sostenibilidad, que son:

- Empoderamiento femenino, trabajando en la realización personal de las mujeres.
- Talento extraordinario, brindando a sus colaboradores oportunidades de crecimiento profesional.
- Cuidado del planeta, con el compromiso de proteger y conservar el medio ambiente.
- Integridad y transparencia, promoviendo un actuar ético y transparente.
- Productos responsables, productos libres de crueldad animal, centrados en innovación científica.

2.2 VISIÓN

Llegar a ser una corporación multinacional de venta directa de productos de belleza con el mejor prestigio y ser la más competitiva en el mercado, basándose en empoderar a la mujer y darles prosperidad a todos sus colaboradores.

2.3 MISIÓN

Empoderar a la mujer elevando su nivel de vida y de todos los miembros que son parte de esta corporación, brindándoles mejoras a través de oportunidades y puestos de trabajo en el aspecto económico, profesional y personal, y con productos de belleza de estándares y calidad global.

2.4 ORGANIZACIÓN DE LA EMPRESA

A continuación, se muestra el organigrama del área de TI de esta corporación.



Figura 1. Organigrama Corporación.

Fuente: Autoría propia.

2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

Durante el desarrollo del proyecto en mención, el autor se desempeñó como Senior Software Engineer en la sub gerencia de Aplicaciones dentro de la gerencia de Tecnología de Información de esta corporación.

La sub Gerencia de Aplicaciones, era la encargada de monitorear e implementar todas las aplicaciones y soluciones para una correcta continuidad del negocio, así como para los nuevos requerimientos que eran solicitados por parte de la corporación.

2.6 EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN

A continuación se listan las funciones y tareas desarrolladas en la corporación de belleza:

- Análisis, diseño e implementación de requerimientos para sistema web ePlanning, tanto en backend, como en frontend y stores procedures en base de datos.
- Soporte al usuario y resolución de incidencias para el sistema web ePlanning.
- Diseño e implementación de solución para algoritmo de posicionamiento de sistema CDI, composición de imágenes.
- Diseño e implementación de migración de sistema legado ePlanning a una arquitectura orientada a microservicios sin servidor.
- Realización de instructivos y manuales de los proyectos desarrollados.
- Capacitaciones al área de planeamiento sobre sistema ePlanning.

CAPÍTULO III - ACTIVIDADES DESARROLLADAS

3.1. SITUACIÓN PROBLEMÁTICA

La corporación de belleza por el año 2018 inició un proceso para maximizar las ventas, elaborando estrategias e invirtiendo en soluciones para dicho fin, el área de planeamiento no ajena a este crecimiento necesitaba generar ofertas que den mayor rentabilidad a cada una de sus consultoras y por ende a la empresa, entonces se debían estructurar ofertas más personalizadas a comparación del enfoque tradicional de las ofertas masivas en la cual una oferta era visibilizada para todas las consultoras, los sistemas debían estar preparados para el aumento en las cargas.

3.1.1. DEFINICIÓN DEL PROBLEMA:

El área de planeamiento de la corporación de belleza alrededor del año 2018, inició con el aumento en la carga de data para el sistema ePlanning y este comenzó a presentar lentitud y caídas, se hicieron varias modificaciones para mejorar el rendimiento, pero debido a la arquitectura monolítica del sistema, ya era necesario migrar la aplicación a una arquitectura escalable, con mayor rendimiento y que soporte altas cargas de trabajo.

El sistema ePlanning es un sistema web elaborado bajo el patrón MVC basado en una arquitectura monolítica, un servidor para la aplicación y otro para la base de datos; nace la necesidad de migrar a una arquitectura basada en microservicios para que pueda soportar el aumento de carga y esté preparada para futuras mejoras.

3.2 SOLUCIÓN

El presente trabajo busca diseñar e implementar la migración del sistema web monolítico ePlanning a una arquitectura basada en microservicios.

3.2.1. OBJETIVO

3.2.1.1 OBJETIVO GENERAL

Realizar la migración el sistema web ePlanning a una arquitectura basada en microservicios para soportar el aumento de carga del sistema.

3.2.1.2 OBJETIVOS ESPECÍFICOS

- Analizar y diseñar una arquitectura basada en microservicios y sin servidor, capaz de soportar altas cargas de trabajo.
- Implementar módulos y procesos de ePlanning en nueva arquitectura.
- Generar un ahorro en el coste de operación con la nueva arquitectura.

3.2.2 ALCANCE

El alcance del proyecto puede ser definido por las actividades listadas a continuación:

- Realizar análisis de procesos y funcionalidades actuales de ePlanning, así como integraciones con otros sistemas.
- Diseñar la arquitectura de la nueva solución de ePlanning.
- Implementar la nueva arquitectura en la nube.
- Implementar los procesos y funcionalidades en la nueva arquitectura.
- Monitorear el rendimiento de la solución.

Los siguientes aspectos no están considerados dentro del alcance del proyecto, y por lo tanto no serán abordados en el mismo:

- El proyecto guiará y acompañará al área de DevOps para la configuración de los ambientes y despliegues automáticos de la migración a desarrollar, el proceso de configuración como para el CI/CD, será responsabilidad del área de DevOps.
- El proyecto guiará y acompañará al área de seguridad para validar la correcta configuración de red y comunicación entre los componentes propios del sistema a desarrollar, así también como la comunicación con los sistemas externos, pero las configuraciones de redes como tal y su correcta comunicación, será responsabilidad del área de seguridad.
- El proyecto, implementará y realizará validaciones en el ambiente de pruebas en AWS de la corporación, es responsabilidad de la corporación proveer al proyecto la cuenta de AWS con los permisos necesarios para su implementación.

3.2.3 ETAPAS Y METODOLOGÍAS

Todo el proceso de migración, desde la concepción de la idea, análisis, diseño, implementación y puesta en producción fue trabajado bajo la metodología SCRUM, mediante las reuniones diarias se lograban establecer los objetivos, así como detectar amenazas en corto tiempo y cumplir con los objetivos propuestos, a continuación se describen las fases del proyecto:

- Inicio
- Planificación y estimación
- Implementación
- Revisión y Retrospectiva
- Liberación

3.2.4 FUNDAMENTOS UTILIZADOS

3.2.4.1 Metodología SCRUM

Scrum es un marco de trabajo que permite la colaboración entre equipos, los cuales aprenden a través de las experiencias a autoorganizarse mientras atacan un problema y realizan retrospectivas sobre los logros y fracasos obtenidos para una mejora continua.

Principalmente es utilizado por los equipos de desarrollo de software, pero sus principios y métodos se pueden aplicar a cualquier equipo de trabajo o área, es por ello que es tan popular. A pesar que suele considerarse constantemente como un marco de gestión de proyectos ágil, SCRUM incluye un conjunto de reuniones, herramientas y métodos que colaboran con los equipos a estructurar y coordinar mejor su trabajo. Las fases de SCRUM se definen como inicio, planificación y estimación, implementación, revisión del sprint, liberación.

Tabla 4 - Fases de SCRUM

Fases	Actividades a realizar
Inicio	- Se identifican los miembros del Equipo Scrum. - Crear una lista de productos priorizado
Planificación y estimación	- Se crean las historias de usuario. - Se define el esfuerzo por cada historia - Se crea el sprint Backlog
Implementación	- Se inicia con la construcción de la solución. - Se realizan reuniones diarias.
Revisión del sprint	- Se realiza la revisión del sprint. - Se realiza la retrospectiva.
Liberación	- Se realiza el pase a producción del producto total. - Se realiza monitoreo del correcto funcionamiento. - Se obtiene la aprobación por parte del product owner.

Nota: Autoría propia.

Así mismo dentro de los roles de SCRUM tenemos: Product Owner, Scrum Master, Scrum Team.

Tabla 5 - Roles SCRUM

Rol	Descripción
Product Owner	Es el representante del negocio con el equipo se encarga de comunicar los requerimientos y/o priorizarlos.
Scrum Master	Hace el rol de líder servicial, traslada la visión del proyecto al equipo y se encarga de llevar a cabo las reuniones o ceremonias de SCRUM.
Equipo Scrum	Grupo de profesionales con los conocimientos técnicos necesarios y que desarrollan el proyecto de manera conjunta llevando a cabo las historias a las que se comprometen al inicio de cada sprint.

Nota: Autoría propia.

Dentro de los artefactos o productos de trabajo esenciales para SCRUM tenemos: Product Backlog, Sprint Backlog y el tablero Kanban.

Tabla 6 - Artefactos SCRUM

Artefacto	Descripción
Product Backlog	Es el conjunto de todos los requerimientos que se va a implementar para sacar el producto, está correctamente priorizado.
Sprint Backlog	Es el conjunto de historias que se trabajaran para la iteración.
Tablero Kanban	Aquí se puede visualizar el avance de las historias o requerimientos, normalmente se maneja por 3 columnas: trabajo por hacer, trabajo en progreso y trabajo hecho.

Nota: Autoría propia.

Dentro de las ceremonias, SCRUM establece 5 eventos tenemos: Product Planning, el Sprint Planning, los Daily Meetings, la Sprint Review, y la Sprint Retrospective.

Tabla 7 - Product Backlog

Rol	Descripción
Product Planning	Es el planeamiento de los requerimientos totales del proyecto, aquí se analizan, definen y priorizan.
Sprint Planning	Es el planeamiento de los requerimientos a trabajar para la siguiente iteración del proyecto, aquí se analizan, definen y priorizan.
Daily Meetings	Es la reunión diaria, aquí es donde se comenta el trabajo realizado, el que se va a realizar y si hay problemas.
Sprint Review	Es la presentación del entregable del sprint, debe ser un producto tangible.
Sprint Retrospective	Se analizan fortalezas y debilidades y reconocen métricas para la mejora continua.

Nota: Autoría propia.

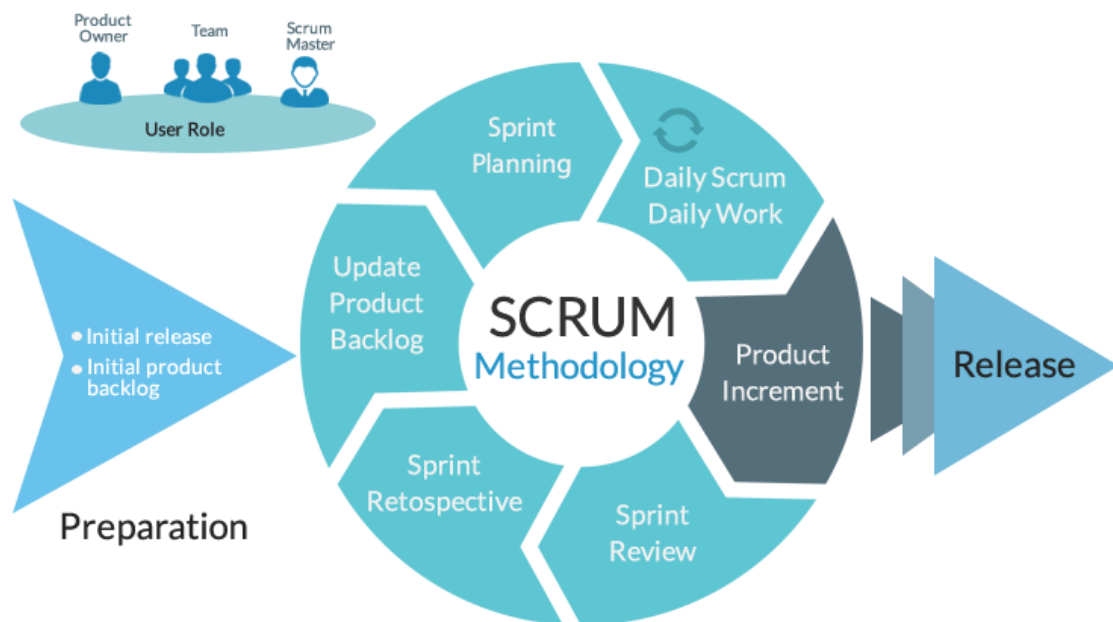


Figura 2: Ciclo de vida SRUM

Fuente: Ventajas y desventajas de la metodología Scrum (2019)

3.2.4.2 Arquitectura basada en microservicios

Ya hace mucho tiempo que la perspectiva para el desarrollo de aplicaciones era básicamente la arquitectura monolítica, esta perspectiva tiene ciertas desventajas, ya que a medida que la aplicación va creciendo es más difícil darle mantenimiento y agregar nuevas funcionalidades.

Los microservicios es un modelo de arquitectura y a su vez, un modelo para desarrollar software, en el cual las aplicaciones se subdividen en pequeñas partes, que no son dependientes de las otras, a comparación de la perspectiva tradicional en donde todo el proceso se encontraba en una sola pieza, cada microservicio cumple una tarea en específico. Adicionalmente muestro dos definiciones de dos de los más grandes proveedores de servicios en la nube:

La idea central de una arquitectura basada en microservicios es dividir las funcionalidades en unidades verticales cohesivas, no mediante capas tecnológicas, sino a través de la implementación de un dominio específico. La siguiente figura representa una arquitectura de referencia para una aplicación de microservicios típica en AWS. (AWS, 2021)

Una arquitectura de microservicios consiste en una colección de pequeños servicios autónomos. Cada servicio es autónomo y debe implementar una capacidad comercial única dentro de un contexto delimitado. Un contexto acotado es una división natural dentro de un negocio y proporciona un límite explícito dentro del cual existe un modelo de dominio. (Azure, 2020).

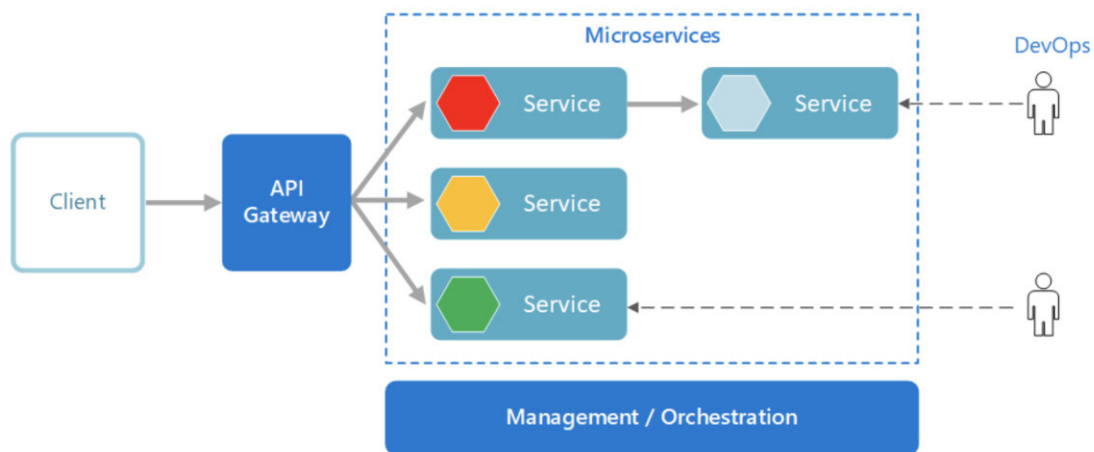


Figura 3: Arquitectura orientada a microservicios

Fuente: Microservices architecture design, Azure (2019)

3.2.4.3 Arquitectura sin servidor

Es un modelo de arquitectura desarrollo que se da en la nube y nos da la posibilidad de diseñar y ejecutar aplicaciones sin tener que aprovisionar servidores.

El agente de servicios en la nube debe encargarse de las labores de rutina y entretanto complejas como son el acondicionamiento, manutención y adecuación de la los servidores, como podrían ser instalación de actualizaciones, parches, etc. Normalmente se mantiene el código de la aplicación en repositorios o containers que son ejecutados al ser llamado alguna tarea.

Una vez que la aplicación es instalada, responden a las llamadas por ejecución e incrementan o reducen de manera automática en función de la demanda. Normalmente, el agente de servicios en la nube ofrece este modelo a un precio que se mide a través de ejecución por eventos, así es que te cobran por cada vez que ejecutas algo y no mientras está en reposo, adicionalmente a esto tenemos las ventajas del escalado automático y alta disponibilidad.

3.2.4.4 Amazon Web Services

Amazon Web Services es un proveedor de servicios en la nube y el que lidera actualmente el mercado, con un 33% de participación en el cuarto trimestre del 2021 superando a sus principales competidores como son Azure y Google Cloud, con 21% y 10% respectivamente según estimaciones de Synergy Research Group.

3.2.4.5 Lambda

Amazon Lambda es un servicio de la nube sin servidor y de ejecución por eventos, ejecuta tu código a demanda, sin invertir esfuerzo en suministrar o administrar servidores. Se puede llamar a Lambda desde casi todos los servicios de Amazon Web Services y actualmente soporta más de 6 lenguajes de programación. Solamente se le cobrará por el tiempo que la tarea se esté ejecutando.

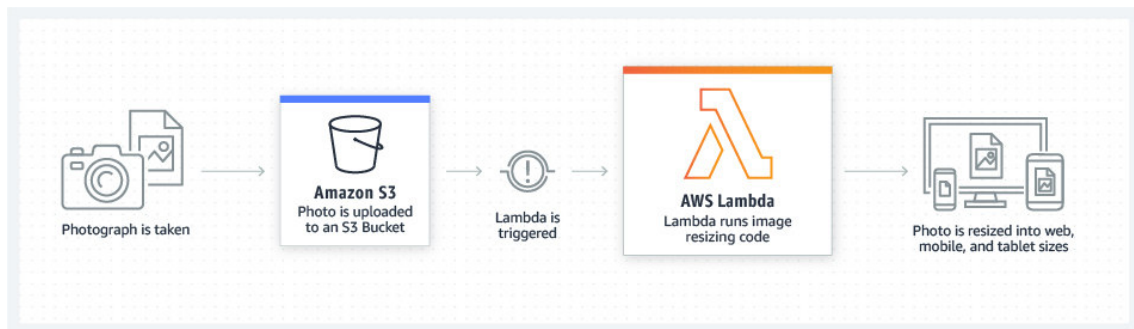


Figura 4: AWS Lambda service
Fuente: Amazon Web Services

3.2.4.6 API Gateway

Amazon API Gateway es un servicio en la nube totalmente administrado que permite a los desarrolladores de software crear, publicar, administrar, monitorear y proteger sus APIs a cualquier nivel. Las API son la "puerta de entrada" para que otras aplicaciones puedan acceder a datos, alguna lógica del negocio o funcionalidad de otros servicios. Este servicio te permite crear tanto APIs REST y WebSocket que permite la comunicación bidireccional entre aplicaciones en tiempo real y de forma asíncrona.

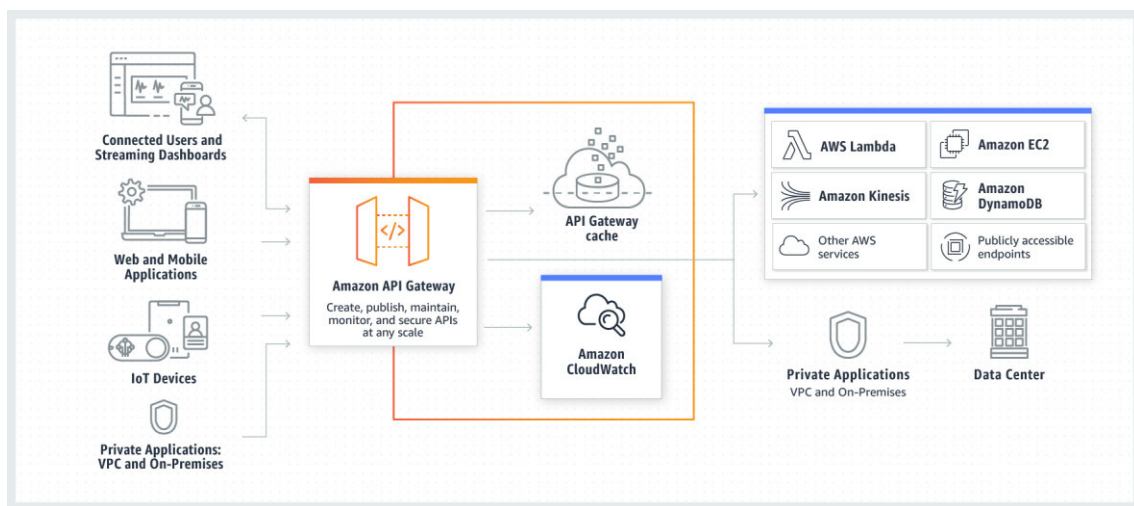


Figura 5: AWS API Gateway
Fuente: Amazon Web Services

3.2.4.7 Step Functions

Amazon Step Functions es un componente en la nube de enfocado a flujos de trabajo y con poco código, con esto los desarrolladores pueden crear aplicaciones distribuidas, automatizadas y crear flujos de datos mediante servicios de Amazon. Amazon Step Functions administra las fallas, los reintentos, la ejecución en paralelo, la comunicación con otros servicios y la visibilidad del proceso para que los desarrolladores se centren en la lógica del negocio que genere más valor.

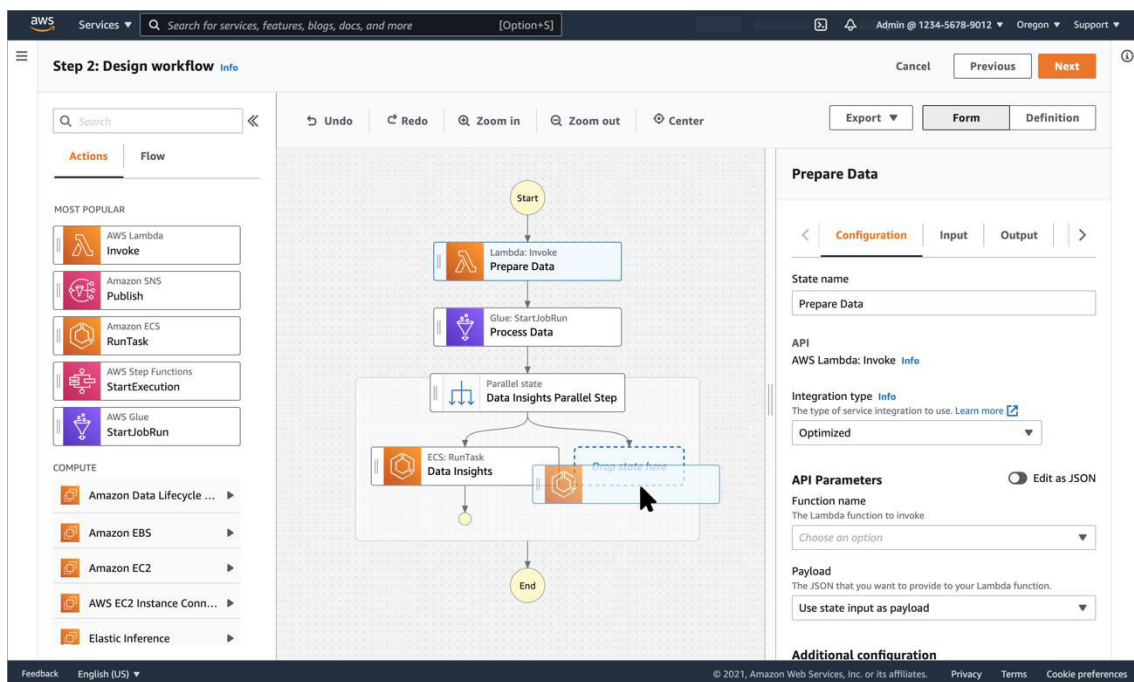


Figura 6: AWS Step Functions

Fuente: Amazon Web Services

3.2.4.8 Fargate

Amazon Fargate es un servicio en la nube sin servidor, basado en la ejecución por eventos que permite a los desarrolladores centrarse en el desarrollo de software sin tener que suministrar ni configurar servidores. El código se coloca en contenedores que son almacenados en Amazon Elastic Container Service (ECS) y ejecutados a demanda.

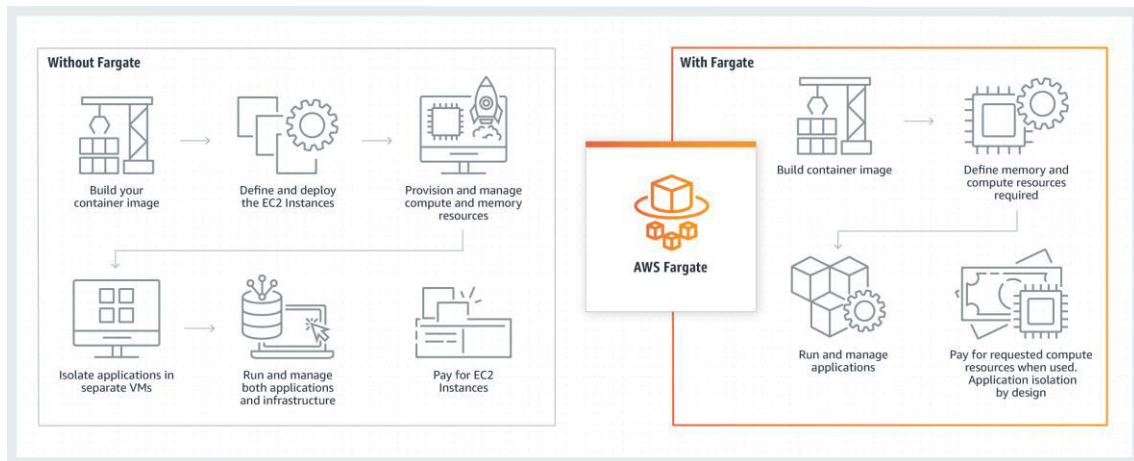


Figura 7: AWS Fargate service

Fuente: Amazon Web Services

3.2.4.9 SQS

Amazon SQS es un servicio de mensajería de colas totalmente administrado que permite desacoplar componentes y adaptar la escala de microservicios, sistemas distribuidos y aplicaciones sin servidor. SQS suprime la complicación y los gastos que generalmente se asocian al suministro y marcha de middleware enfocado a mensajes. Con Amazon SQS, puede enviar, recibir y almacenar mensajes entre cualquier componente en la nube.

Amazon SQS brinda dos figuras de colas de mensajes. Primero las colas estándar nos ofrecen un gran volumen de procesamiento, un mejor ordenamiento y nos garantiza la entrega de mensajes al menos una vez. Y también tenemos las colas FIFO que ofrecen garantizar que el mensaje se procese solamente una vez, y en el orden en el que fue enviado.

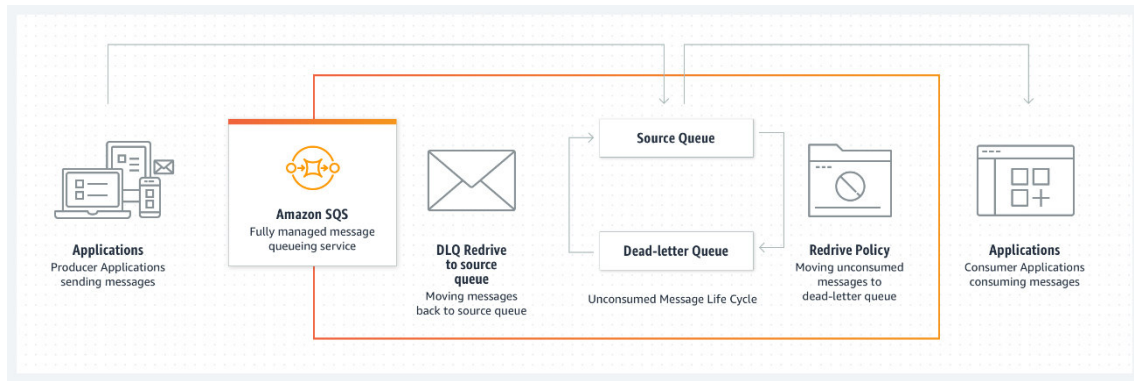


Figura 8: AWS SQS service

Fuente: Amazon Web Services

3.2.4.10 S3 - Simple Storage Service

Amazon S3 es un servicio en la nube para almacenamiento de objetos que brinda alta disponibilidad de los datos, escalabilidad, seguridad de sus datos y rendimiento. Se puede almacenar y proteger grandes cantidades de datos. Los tipos de almacenamiento de S3 ofrecen rentabilidad y características para administrar los repositorios fáciles de utilizar, puede reducir costos, administrar sus datos y configurar las políticas de acceso a sus datos.



Figura 9: AWS Lambda service

Fuente: Amazon Web Service

3.2.4.11 Lenguaje de programación: Microsoft .Net Core

.NET Core es el entorno para desarrollo de software de Microsoft más actual, de código libre, alto rendimiento que trabaja en cualquier plataforma y que permite el desarrollo de aplicaciones de todo tipo.

Su lanzamiento se dio en el año 2016, y es parte del resultado por volver a la tecnología .NET Framework multiplataforma, ya que siempre dependía del sistema operativo Windows para poder funcionar.

Microsoft .NET Core, a comparación del .NET Framework tradicional, no mantiene una dependencia con el sistema operativo Windows y utiliza el sistema de administrador de paquetes NuGet, por el que recibiremos las distintas actualizaciones de .NET Core.

3.2.4.12 “EH-UNMSM: E-Health Cloud para la mejora de procesos de la clínica universitaria UNMSM mediante una arquitectura de microservices” (Cayo, D. 2018)

En el trabajo de investigación de tesis de Cayo, D. (2018), para obtener el grado de ingeniero de sistemas plantea el problema que en un principio la Clínica Universitaria solo tenía pocas áreas de salud. Con el pasar del tiempo, la cantidad de pacientes aumentó así también como las especialidades, creciendo por ende el almacenamiento de información. Los tiempos de atención fueron en aumento y los trámites documentarios para la obtención de resultados se volvían cada vez más lentos..

La tesis propone el desarrollo de un aplicativo E-Health Cloud que describe el análisis y diseño de los procesos, la identificación de los requerimientos de software, la comunicación entre los distintos componentes, el armado de la infraestructura, y el despliegue en la nube. El aplicativo propuesto permite mejorar el proceso de diagnóstico, mejorar la comunicación entre las distintas áreas médicas usando una arquitectura de software basada en microservicios. (Cayo, D. 2018)

3.2.4.13 “Aplicación móvil de control académico utilizando la arquitectura de microservicios bajo la metodología de desarrollo scrum. Caso: Escuela profesional de Obstetricia de la Universidad Nacional Mayor de San Marcos” (Cuba, O. & Espinoza, M. 2020)

En nuestro país, la calidad de la educación en universidades e institutos muestra un bajo nivel si lo comparamos con otras universidades de latinoamérica. Los estándares destacables del SINEACE son la evaluación del cargo docente, la supervisión adecuada a estudiantes y un sistema de información que apoye a la administración de los estudiantes. La escuela de obstetricia de la UNMSM se proyecta a implementar un sistema de información que permita el cumplir con los estándares solicitados por el SINEACE, en estos tiempos el record de los alumnos, tales como asistencias a clases, libreta de notas y anuncios no se encuentran disponibles en tiempo real y se debe esperar mucho tiempo para visualizar estos datos, ya que se realiza de manera tradicional en documentos físicos, archivos excel u otros medios que dificultan el control de información académica entre profesores y alumnos. La tesis propone el diseño e implementación un sistema informático que mejore la administración de los datos académicos mediante una solución móvil. (Cuba, O. & Espinoza, M. ,2020)

3.2.5 IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS

De las fases definidas, se detalla a continuación para un mejor entendimiento, la implementación en este proyecto basados en la metodología SCRUM:

Tabla 8 - Fases SCRUM

Fases	Actividades realizadas
Inicio	Se define product owner y objetivos del proyecto. Se define Scrum Master y stakeholders. Se define equipo SCRUM. Se desarrollan las épicas. Se crea el Product Backlog.
Planificación y estimación	Se definen las historias de usuario. Se estiman las historias de usuario. Se crea el Sprint Backlog
Implementación	Se trabaja en los entregables. Se realizan los Daily Stand Up. Refinamiento del Product Backlog.
Revisión y Retrospectiva	Se realiza la revisión del sprint. Se realiza la retrospectiva.
Liberación	Se realiza el pase a producción del producto total. Se realiza monitoreo del correcto funcionamiento. Se obtiene la aprobación por parte del product owner.

Nota: Autoría propia

3.2.5.1 INICIO

Según metodología SCRUM, se definieron los siguiente roles:

Tabla 9 - Roles SCRUM

ROL	RESPONSABLE
Equipo SCRUM	Ingeniero de Software Senior
	Analista de Calidad
Scrum Master	Analista Funcional de proyecto.
Product Owner	Analista de planeamiento.
Stakeholders	Área de planeamiento.

Nota: Autoría propia

En este sentido se debe implementar una solución que soporte todos los procesos y cálculos con los que contaba el sistema ePlanning y sea enfocado a microservicios ya que serán consumidos por otro sistema, obteniendo los siguientes beneficios:

- Soporte a la gran cantidad de carga de ofertas que se venía teniendo por la segmentación en la composición.
- Reducir los costos de operación, debido a que se maneja en una operación sin servidor.
- Una arquitectura altamente escalable y resistente a cambios debido al desacoplamiento de los componentes.

Dentro del proceso de cargas de ofertas, se ha dividido en los siguiente sub procesos:

1. Cálculo de Tipo de Oferta: La solución debe calcular en base a la composición de las ofertas, palanca, categorías y el catálogo de tipos de oferta previamente cargado poder calcular el tipo de oferta, el cual se registrará en una nueva propiedad dentro de la oferta.

2. Cálculo de descripciones comerciales: Las ofertas cargadas pueden contar con uno o varios productos, cada uno de estos productos cuenta con una descripción comercial individual, pero en conjunto las descripciones finales de la oferta deben ser calculadas en base a los estándares que proporciona la corporación para el nombre de sus ofertas en base a su composición, se detalla a continuación un ejemplo para su entendimiento:

Tabla 10 - Descripciones Comerciales

Producto	Descripción Comercial
	Colonita Nitro 100 ml.
	Pack x 3 Colonias Nitro 100 ml.
	Colonia Nitro 100 ml. + Colonia Winner 100 ml.

Nota: Autoría propia.

3. Cálculo de Ranking Tono: Cuando se tiene una oferta con un producto del cual puedes elegir entre su variedad de tonos, se debe obtener el producto con el mejor Ranking, y el código de este será tomado en cuenta para la composición de la oferta y a futuro cuando se componga su imagen. Ejemplo: Se tiene el labial Colorfix con los tonos, fucsia, rosado, rojo, etc. Si el tono fucsia es el tono con el mejor Ranking, este será marcado y a futuro cuando se elabore la imagen de la oferta es el producto que será mostrado.



Figura 10: AWS Lambda service

Fuente: Amazon Web Services

4. Cálculo de Código Único de Venta y Código Único de Venta Padre por oferta:

Aquí el sistema ePlanning debe conectarse a un sistema legado llamado SICC, el cual genera los Código Único de Venta (CUV), así como el CUV Padre por oferta.

En el ejemplo se puede visualizar que, para cada producto en específico de la oferta, se generó un CUV pero a su vez se genera un CUV Padre que es el código que identificará a toda la oferta.

Tabla 11 - Product Backlog

Número de Oferta	CUC	SAP	Descripción	CUV	Cuv Padre
1	20201010	20256783	Colonia Nitro Intense	20805045	20805045
1	20201010	20343434	Colonia Nitro Clásica	25434234	20805045
1	20211083	204343443	Colonia Winner	3467823	20805045

Nota: Autoría propia.

5. Integración con portal de ventas de ventas: El portal de ventas, es el sistema final donde las consultoras pueden visualizar las diferentes ofertas que tiene por campaña, aquí ellas pueden seleccionar las ofertas, agregarlos a un carrito de compras y luego realizar sus pedidos.

6. Integración con sistema de Composición de imágenes (CDI): Una vez que se ha completado con los cálculos necesarios para cada oferta, el sistema ePlanning debe integrarse con CDI para poder diagramar la imagen de acuerdo a su composición y luego ser visualizada en el portal de ventas.

7. Elaboración de Web ePlanning para monitorización: Se debe crear una para realizar el monitoreo de los envíos a ePlanning, pestaña para proceso de carga de ofertas para palanca Arma tu Pack (que tiene un comportamiento totalmente diferente al de otras palancas), descarga de descripciones comerciales.

3.2.5.1.3 CONSIDERACIONES / OBSERVACIONES

- El sistema ePlanning no garantiza la disposición y correcto funcionamiento de los sistemas subyacentes, tales como SICC, Portal de Ventas, CDI, etc., las validaciones deberán ser realizadas por su respectivo equipo.
- El paso el ePlanning tradicional a la arquitectura de microservicios se hará paulatinamente por países, empezando por aquellos con menor carga a mayor carga, para en el interín ir evaluando el rendimiento y el comportamiento del sistema.
- Los catálogos de datos, tales como Descripciones comerciales, Tipos de Oferta, Ranking tonos, etc. son responsabilidad del planner y de su carga al sistema ePlanning, para el correcto funcionamiento de los cálculos

3.2.5.2 PLANIFICACIÓN Y ESTIMACIÓN

En conjunto del Product Owner y el Scrum Master, basadas en el análisis previamente realizado se definieron las siguientes historias de usuario y sus estimaciones:

Tabla 12 - User History 1

ID: HU - 01	RESPONSABLE: Software Engineer
TÍTULO: Diseño e implementación de nueva arquitectura ePlanning	
TIEMPO ESTIMADO: 40 horas.	
DESCRIPCIÓN: Se debe proponer el diseño de la arquitectura altamente escalable y desacoplada.	
TAREAS:	
- Diseñar arquitectura, en base a las observaciones mostradas abajo. - Elaborar documento cloudformation en formato YAML, con la infraestructura de la solución para configurar el despliegue automático.	
CRITERIOS DE ACEPTACIÓN:	
- Debe ser orientada a microservicios. - Debe ser una arquitectura sin servidor. - Debe ser altamente escalable.	

Nota: Autoría propia.

Tabla 13 - User History 2

ID: HU - 02	RESPONSABLE: Software Engineer
TÍTULO: Elaboración de documento de roles y permisos.	
TIEMPO ESTIMADO: 16 horas.	
DESCRIPCIÓN: Se debe elaborar un documento con los roles a generar para cada componente dentro de la infraestructura y los permisos de comunicación entre estos.	
TAREAS:	
- Elaborar documento de roles en formato JSON. - Se debe replicar los roles para los ambientes DEV, QA y PRD.	
CRITERIOS DE ACEPTACIÓN:	
- Ninguno.	

Nota: Autoría propia.

Tabla 14 - User History 3

ID: HU - 03	RESPONSABLE: Héctor Noriega
TÍTULO: Implementación de lambda belc-ep-parquet-reader	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: Se debe elaborar un lambda que pueda leer un archivo parquet y devolver un archivo en formato json.	
TAREAS:	
- Crear lambda belc-ep-parquet-reader - El lambda debe permitir leer un archivo parquet de S3. - Se debe transformar el archivo leído a archivo JSON. - Se debe guardar el archivo en S3.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe visualizar el archivo transformado a formato JSON.	

Nota: Autoría propia.

Tabla 15 - User History 4

ID: HU - 04	RESPONSABLE: Héctor Noriega
TÍTULO: Implementación de lambda belc-ep-call-step-function	
TIEMPO ESTIMADO: 12 horas.	
DESCRIPCIÓN: Crear lambda que sirve para conectar el endpoint API Gateway con el stepFunction.	
TAREAS:	
- Crear lambda belc-ep-call-step-function. - El lambda debe estar expuesta por Api Gateway en un método POST. - El lambda debe recibir el request y ejecutar el step function enviando el request como parámetro. - Si existe algún error en la ejecución, mostrar mensaje de error.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que el step function se haya iniciado.	

Nota: Autoría propia.

Tabla 16 - User History 5

ID: HU - 05	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-desc-commercial	
TIEMPO ESTIMADO: 40 horas.	
DESCRIPCIÓN: El microservicio debe calcular la descripción comercial final para cada oferta dentro del archivo.	
TAREAS:	
- Crear lambda belc-ep-desc-commercial - El lambda debe leer el archivo JSON generado. - El lambda debe leer el maestro de descripciones. - El lambda debe calcular la descripción comercial por oferta. - El lambda debe actualizar la descripción comercial en el archivo JSON para cada oferta. - Si existe algún error en la ejecución, mostrar mensaje de error.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que las descripciones comerciales de las ofertas se hayan calculado correctamente.	

Nota: Autoría propia.

Tabla 17 - User History 6

ID: HU - 06	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-ranking	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: El microservicio debe calcular el min ranking tono para cada oferta.	
TAREAS:	
- Crear lambda belc-ep-desc-ranking - El lambda debe leer el archivo JSON generado. - El lambda debe leer el maestro de rankings. - El lambda debe calcular el ranking tono por oferta actualizarlo el archivo JSON para cada oferta.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que el ranking tono de las ofertas se hayan calculado correctamente.	

Nota: Autoría propia.

Tabla 18 - User History 7

ID: HU - 07	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-offer-type	
TIEMPO ESTIMADO: 16 horas.	
DESCRIPCIÓN: El microservicio debe calcular el tipo de oferta para cada oferta.	
TAREAS:	
- Crear lambda belc-ep-offer-type. - La lambda debe leer el archivo JSON generado. - La lambda debe leer el maestro de TipoOferta. - Debe calcular el offertype por oferta en base al maestro y actualizar en archivo JSON.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que el tipo oferta de las ofertas se hayan calculado correctamente.	

Nota: Autoría propia

Tabla 19 - User History 8

ID: HU - 08	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-cuv	
TIEMPO ESTIMADO: 16 horas.	
DESCRIPCIÓN: El microservicio debe insertar un mensaje en la cola de procesos.	
TAREAS:	
- Crear lambda belc-ep-cuv - La lambda debe estar expuesto por API Gateway método POST - La lambda debe insertar un mensaje en la cola de procesos. - Si existe algún error en la ejecución, mostrar mensaje de error.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que el mensaje se haya insertado en la cola de procesos correctamente.	

Nota: Autoría propia.

Tabla 20 - User History 9

ID: HU - 09	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-reprocess-description	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: El microservicio debe permitir calcular la descripción comercial para cada oferta dentro de la carga.	
TAREAS:	
- Crear lambda belc-ep-reprocess-description - La lambda debe ser expuesta por Api Gateway, método POST - La lambda debe leer el maestro de descripciones. - La lambda debe calcular la descripción comercial por oferta. - Debe actualizar la descripción comercial en el archivo JSON para cada oferta. - Si existe algún error en la ejecución, mostrar mensaje de error.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que las descripciones comerciales de las ofertas se hayan calculado correctamente.	

Nota: Autoría propia.

Tabla 21 - User History 10

ID: HU - 10	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-call-fargate	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: El microservicio debe permitir ejecutar el task de fargate.	
TAREAS:	
- Crear lambda belc-ep-call-fargate - La lambda debe leer el mensaje de la cola de procesos y eliminar el mensaje. - Valida si existe un proceso en ejecución del mismo país en la cola de control. - Si no existe una ejecución en proceso para el país, la lambda debe ejecutar el task de Fargate configurado y pasarle el request por parámetro. - Si la ejecución del task se realiza correctamente, debe insertar el país en ejecución a la cola de control. - Si existe una ejecución en proceso para el país, no se debe ejecutar la tarea y debe reinsertar el mensaje a la cola de procesos, con un delay de 5 minutos. - Si existe algún error en la ejecución, mostrar mensaje de error.	
OBSERVACIONES:	

-
- Al ejecutar la lambda se debe validar que, si existe un proceso en ejecución para el país, no se ejecute el task de Fargate y que se debe reinsertar el mensaje a la cola de procesos con un delay de 5 min.
 - Al ejecutar la lambda se debe validar que, si **no** existe un proceso en ejecución para el país, se ejecute el task de Fargate y que se debe insertar el mensaje con el país a la cola de procesos.
-

Nota: Autoría propia.

Tabla 22 - User History 11

ID: HU - 11	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-portal-ventas	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: El microservicio debe permitir la integración con el sistema Portal de ventas.	
TAREAS:	
- Crear lambda belc-ep-portal-ventas - La lambda debe leer el archivo json. - La lambda debe transformar el archivo a formato CSV. - La lambda debe dejar el archivo CSV en el bucket del portal de ventas. - Si existe algún error en la ejecución, mostrar mensaje de error.	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que se genere el archivo CSV correctamente en la ruta del bucket del portal de ventas.	

Nota: Autoría propia.

Tabla 23 - User History 12

ID: HU - 12	RESPONSABLE: Software Engineer
TÍTULO: Implementación de lambda belc-ep-call-cdi	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: El microservicio debe permitir la integración el sistema CDI.	
TAREAS:	
- Crear lambda belc-ep-call-cdi - La lambda debe leer el archivo json y transformar a la estructura para CDI. - La lambda debe dejar el archivo JSON en el bucket de CDI. - La lambda debe ejecutar un request al endpoint de CDI	
CRITERIOS DE ACEPTACIÓN:	
- Al ejecutar la lambda se debe validar que se genere el archivo JSON correctamente en la ruta del bucket de CDI y que el proceso de CDI haya iniciado correctamente al ejecutar su request.	
Nota: Autoría propia.	

Tabla 24 - User History 13

ID: HU - 13	RESPONSABLE: Software Engineer
TÍTULO: Implementación de sistema dockerizado belc-eplanning-task	
TIEMPO ESTIMADO: 40 horas.	
DESCRIPCIÓN: La tarea debe permitir calcular los CUVS, CUVs Padre, así como las integraciones con los sistemas SICC, CDI y el portal de Ventas.	
TAREAS:	
- Se debe implementar un programa dockerizado belc-eplanning-task - El programa debe leer el archivo JSON y transformarlo a un insert masivo a la BD de SICC. - El programa debe ejecutar los stored procedures de SICC, para la generación de CUVS. - El programa debe leer el resultado de las tablas finales de SICC para obtener los CUVS de cada oferta. - El programa debe calcular el CUV Padre de cada oferta. - El programa de actualizar los valores de CUV y CUV Padre en el archivo JSON para cada oferta. - El programa debe guardar el archivo en S3. - El programa ejecutará la lambda de integración con El portal de Ventas	

-
- El programa ejecutará la lambda de integración con CDI
 - Una vez finalizado el proceso, la lambda debe eliminar el país de la cola de control, para que pueda ingresar el siguiente proceso.
-

CRITERIOS DE ACEPTACIÓN:

- Al ejecutar el task se debe validar que se hayan calculado correctamente los CUVS y CUVS Padres.
 - Se debe validar que se haya hecho la ejecución de la lambda de integración del portal de ventas
 - Se debe validar que se haya hecho la ejecución de la lambda de integración de CDI.
 - Al finalizar el proceso, ya sea correctamente o con error, se debe eliminar el mensaje de la cola de control.
-

Nota: Autoría propia.

Tabla 25 - User History 14

ID: HU - 14	RESPONSABLE: Héctor Noriega
TÍTULO: Implementación de sistema web ePlanning	
TIEMPO ESTIMADO: 40 horas.	
DESCRIPCIÓN: Implementación de portal web para monitorear cargas.	
TAREAS:	
- Se debe implementar una aplicación web serverless - La aplicación web debe permitir monitorear las cargas que han ingresado a las APIS de ePlanning. - La aplicación debe permitir la descarga de las ofertas cargadas en cada proceso.	
CRITERIOS DE ACEPTACIÓN:	
- El sistema debe permitir visualizar las cargas. - El sistema debe permitir la descarga de las ofertas cargas.	
Nota: Autoría propia.	

Tabla 26 - User History 15

ID: HU - 15	RESPONSABLE: Héctor Noriega
TÍTULO: Implementación módulo ArmatuPack en web ePlanning	
TIEMPO ESTIMADO: 32 horas.	
DESCRIPCIÓN: Implementar un módulo en la web ePlanning para la carga de ofertas de la palanca Arma tu Pack.	
TAREAS:	
- Se debe implementar una vista para permitir las cargas de la palanca ArmatuPack. - El sistema debe permitir la carga de un archivo Excel (.xlsx). - El sistema debe transformarlo el archivo excel a un formato json y guardarlo en S3. - El sistema debe llamar a la lambda de CUV.	
CRITERIOS DE ACEPTACIÓN:	
- El sistema debe permitir la carga de un archivo xlsx. - El sistema debe realizar los cálculos correctamente para la palanca ATP. - El sistema debe enviar la carga a la lambda de CUV para calcular	
Nota: Autoría propia.	

Tabla 27 - User History 16

ID: HU - 16	RESPONSABLE: Héctor Noriega
TÍTULO: Módulo descarga de descripciones comerciales en web ePlanning	
TIEMPO ESTIMADO: 24 horas.	
DESCRIPCIÓN: Implementar un módulo para descarga de descripciones comerciales.	
TAREAS:	
- Se debe implementar una vista para la descarga de descripciones Comerciales. - El sistema debe permitir elegir país y campaña, y con estos datos realizar la búsqueda de las cargas que se hicieron. - El sistema debe permitir la descarga de las descripciones comerciales al seleccionar una carga en formato xlsx.	
CRITERIOS DE ACEPTACIÓN:	
- El sistema debe realizar la búsqueda de las cargas para el país y campaña seleccionados. - Debe permitir la descarga de las descripciones comerciales en formato xlsx.	
Nota: Autoría propia.	

Así mismo se definen el product backlog:

Tabla 28 - Product Backlog

ID	User History	Prioridad	Tiempo
HU - 1	Diseño e implementación de arquitectura ePlanning	Alta	40 horas
HU - 2	Elaboración de documento de roles y permisos.	Alta	16 horas
HU - 3	Implementación de lambda belc-ep-parquet-reader	Alta	24 horas
HU - 4	Implementación de lambda belc-ep-call-step-function	Alta	12 horas
HU - 5	Implementación de lambda belc-ep-desc-commercial	Alta	40 horas
HU - 6	Implementación de lambda belc-ep-ranking	Alta	24 horas
HU - 7	Implementación de lambda belc-ep-offer-type	Alta	16 horas
HU - 8	Implementación de lambda belc-ep-cuv	Alta	16 horas
HU - 9	Implementación de lambda belc-ep-reprocess-description	Alta	24 horas
HU - 10	Implementación de lambda belc-ep-call-fargate	Alta	24 horas
HU - 11	Implementación de lambda belc-ep-portal-ventas	Alta	24 horas
HU - 12	Implementación de lambda belc-ep-call-cdi	Alta	24 horas
HU - 13	Implementación de sistema dockerizado belc-eplanning-task	Alta	40 horas
HU - 14	Implementación de sistema web ePlanning	Alta	40 horas
HU - 15	Implementación módulo ArmatuPack en web ePlanning	Alta	32 horas
HU - 16	Implementación módulo de descarga de descripciones comerciales en web ePlanning	Alta	24 horas

Nota: Autoría propia.

Realizaremos el sprint backlog teniendo en cuenta los tiempos definidos para cada historia de usuario, adicionalmente tendremos como referencia un horario laboral de 40 horas semanales divididos en 8 horas diarias por 5 días (lunes a viernes). El total de horas de implementación son 476 horas aproximadamente. Se ha tomado entonces 6 sprints, teniendo un tiempo de sprint de 2 semanas.

Tabla 29 - Sprint Backlog

SPRINT	ID Historia	Historia de Usuario
SPRINT 1	HU - 1	Diseño e implementación de arquitectura ePlanning
	HU - 2	Elaboración de documento de roles y permisos.
	HU - 3	Implementación de lambda belc-ep-parquet-reader
SPRINT 2	HU - 4	Implementación de lambda belc-ep-call-step-function
	HU - 5	Implementación de lambda belc-ep-desc-commercial
	HU - 6	Implementación de lambda belc-ep-ranking
SPRINT 3	HU - 7	Implementación de lambda belc-ep-offer-type
	HU - 8	Implementación de lambda belc-ep-cuv
	HU - 9	Implementación de lambda belc-ep-reprocess-description
	HU - 10	Implementación de lambda belc-ep-call-fargate
SPRINT 4	HU - 11	Implementación de belc-ep-portal-ventas
	HU - 12	Implementación de lambda belc-ep-call-cdi
	HU - 13	Implementación de sistema dockerizado belc-eplanning-task
SPRINT 5	HU - 14	Implementación de sistema web ePlanning
	HU - 15	Implementación módulo ArmatuPack en web ePlanning
SPRINT 6	HU - 16	Implementación módulo Descarga de descripciones Comerciales en web ePlanning

Nota: Autoría propia.

3.2.5.3 IMPLEMENTACIÓN

3.2.5.3.1 Sprint I

En este sprint se desarrollaron las siguientes historias de usuario:

3.2.5.3.1.1 Diseño e implementación de arquitectura ePlanning

Actualmente la arquitectura en la nube, nos da la posibilidad de implementar diferentes tipos de arquitectura para cualquier proyecto que necesitemos, asegurándonos: disponibilidad, rendimiento, escalabilidad, etc.

- La arquitectura debe ser enfocada a microservicios, con el más alto nivel de desacoplamiento, para garantizar la escalabilidad a futuro del sistema y ser tolerante a fallos.
- La arquitectura debe ser orientada a un enfoque sin servidor, ya que con esto se reduce el nivel de configuración, aprovisionamiento de servidores y se genera gran ahorro en costos, debido a que se cobra por lo que se ejecuta y el tiempo que se ejecuta.

Arquitectura Pre Migración sistema ePlanning:

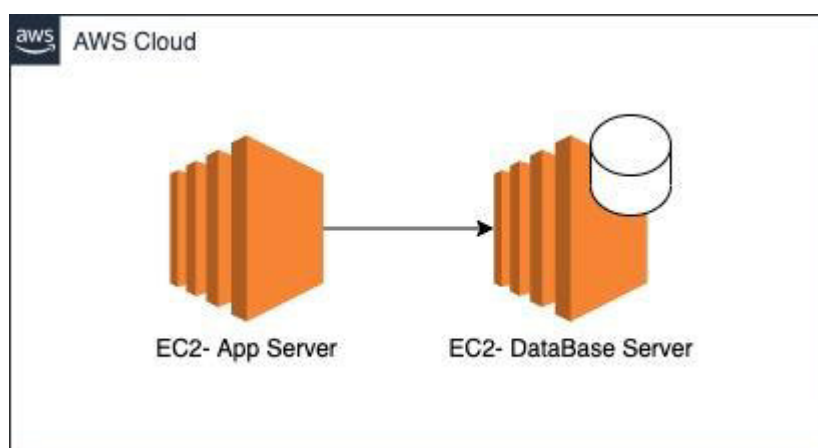


Figura 11: Arquitectura

Fuente: Autoría propia.

Tabla 30 - Componentes Arquitectura Pre Migración

Recurso	Características
AWS EC2 App Server	Sistema Operativo: Windows Server 2019 Tipo de instancia: m5.xlarge Almacenamiento: 200 Gb
AWS EC2 Database Server	Sistema Operativo: Windows Server 2019 Tipo de instancia: m5.2xlarge Almacenamiento: 200 Gb

Nota: Autoría propia.

Arquitectura Post Migración sistema ePlanning:

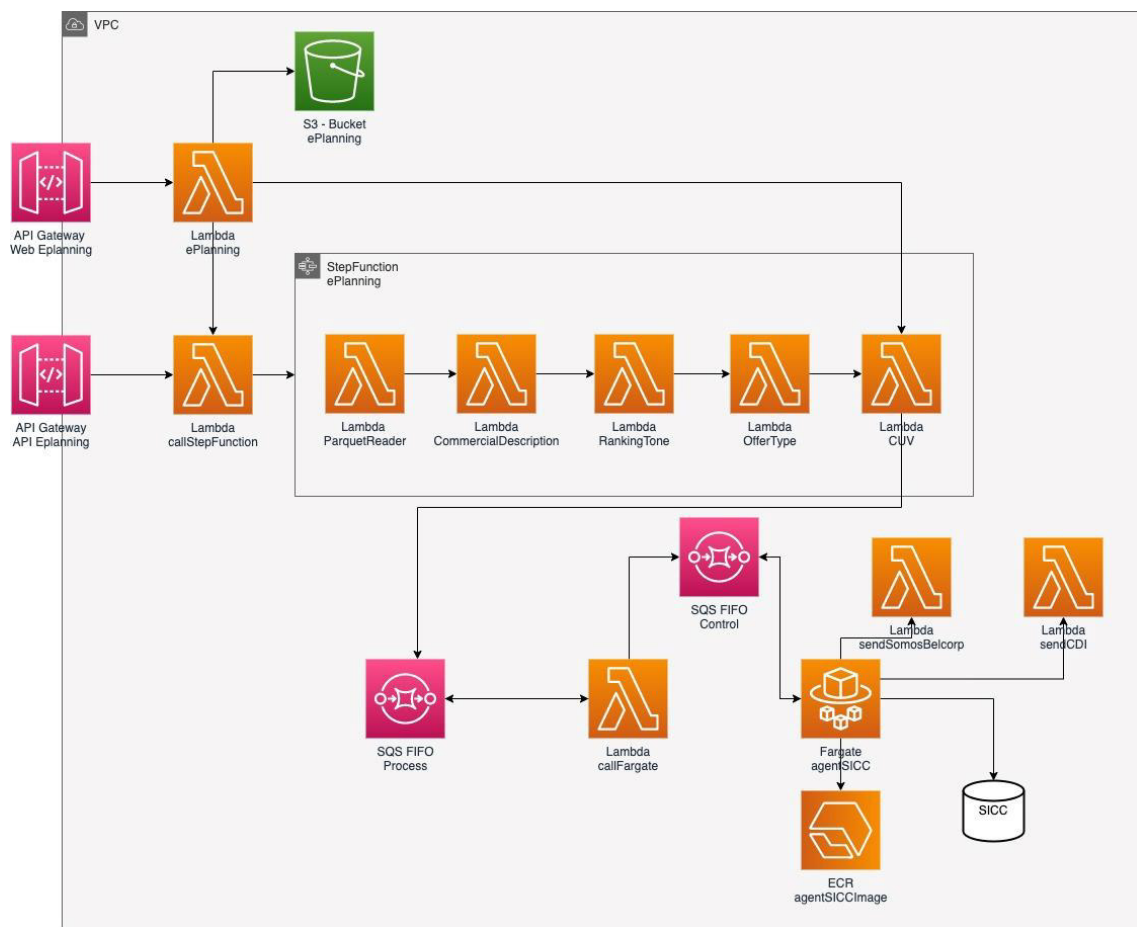


Figura 12: Arquitectura Post

Fuente: Amazon Web Services

Tabla 31 - Componentes Arquitectura Post Migración

Recurso	Componente	Características	Descripción
Función Lambda	ePlanning	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que maneja el front de la web ePlanning.
Función Lambda	callStepFunction	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda llamada por el API Gateway, y hace la ejecución del Step Function.
Función Lambda	parquetReader	Arquitectura: x86 Timeout: 3 minutos Memoria: 512mb Almacenamiento: 512mb	Es la lambda que se encarga de transformar el archivo contiene las ofertas a procesar, de formato parquet a JSON.
Función Lambda	commercialDescription	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que se encarga de calcular las descripciones comerciales de cada oferta.
Función Lambda	rankingTone	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que se encarga de calcular el mínimo ranking tono de cada oferta.
Función Lambda	offerType	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que se encarga de calcular el tipo de oferta de cada oferta.
Función Lambda	CUV	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que se encarga de insertar en la cola de proceso.
Función Lambda	callFargate	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que valida en la cola de control, si existe algún proceso en ejecución para ese país, si no existe, procede a llamar al fargate.
Función Lambda	sendPortalVentas	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que arma el archivo y conecta con el api de la web Portal de ventas.

Función Lambda	sendCDI	Arquitectura: x86 Timeout: 1 minuto Memoria: 512mb Almacenamiento: 512mb	Es la lambda que arma el archivo y conecta con el api del sistema CDI (Software de composición de imágenes).
API Gateway	Web ePlanning	Tipo: REST	Es el API que controla la web ePlanning.
API Gateway	API ePlanning	Tipo: REST	Es el API que controla los endPoints de ePlanning.
AWS SQS FIFO	Control	Retención de mensaje: 4 días. Tamaño máximo de mensaje: 256kb Timeout de visibilidad de mensajes: 0	Es la cola que contiene los países que se encuentran en ejecución.
AWS SQS FIFO	Process	Retención de mensaje: 4 días. Tamaño máximo de mensaje: 256kb Timeout de visibilidad de mensajes: 0	Es la cola que contiene todos los procesos.
AWS Fargate	agentSICC	Sistema Operativo: Linux Arquitectura CPU: x86 Monto de vCPU: 0.5 Memoria: 4Gb Almacenamiento: 20 Gb	Se encarga de ejecutar los Stored Procedures de SICC, para obtener los CUV.
AWS ECR	agentSICCImage	Almacenamiento: 100mb	Se encarga de almacenar la imagen dockerizada que será utilizada por fargate.
AWS S3	ePlanning	Tipo de Almacenamiento: Standard	Es el repositorio del sistema ePlanning
Step Function	ePlannnig	Tipo: Standard	Se encarga de manejar el flujo de trabajo entre las lambdas.

Nota: Autoría propia.

3.2.5.3.1.2 Elaboración de documento de roles y permisos

Los roles y permisos permiten la interacción entre componentes de AWS, basándonos en el Well-Architected Framework, en el área de seguridad, nos indica que solo debe darse los permisos necesarios a los componentes necesarios, es así que se generan los siguientes roles:

Tabla 32 – Roles y Permisos

Role	Componente	Políticas	Componentes
RoleStepFunction	Step Function	lambda:InvokeFunction	belc-ep-parquet-reader belc-ep-desc-commercial belc-ep-ranking belc-ep-offer-type belc-ep-cuv
RoleExecStep	Lambda	states:DescribeExecution, states:ListExecutions states:GetExecutionHistory states:StartExecution	SF-eplanning-machine
RoleS3	Lambda	s3:PutObject s3:GetObject s3:ListBucket s3:PutObjectAcl	Bucket eplanning-api
RoleInsertSQS	Lambda	sqs:GetQueueUrl sqs:SendMessage	SQS-ep-sicc-control.fifo SQS-ep-sicc-proceso.fifo
RoleFargate	Lambda	ecs:RunTask sqs:DeleteMessage	belc-eplanning-task
RoleECSTask	ECS- Task	iam:PassRole	ecsTaskExecutionRole RoleECSTask
		sqs:GetQueueUrl sqs:ReceiveMessage sqs:SendMessage sqs:GetQueueAttributes	SQS-ep-sicc-control.fifo SQS-ep-sicc-proceso.fifo
		sqs:DeleteMessage sqs:GetQueueUrl sqs:ReceiveMessage sqs:SendMessage	SQS-ep-sicc-control.fifo SQS-ep-sicc-proceso.fifo
		s3:PutObject s3:GetObject s3:ListBucket s3:PutObjectAcl	Bucket eplanning-api Bucket CDI Bucket Portal Ventas
		sts:AssumeRole	RoleCallFargate ecsTaskExecutionRole

Nota: Autoría propia.

3.2.5.3.1.2 Implementación de lambda belc-ep-parquet-reader

Se implementó lambda de acuerdo a las especificaciones en historia de usuario

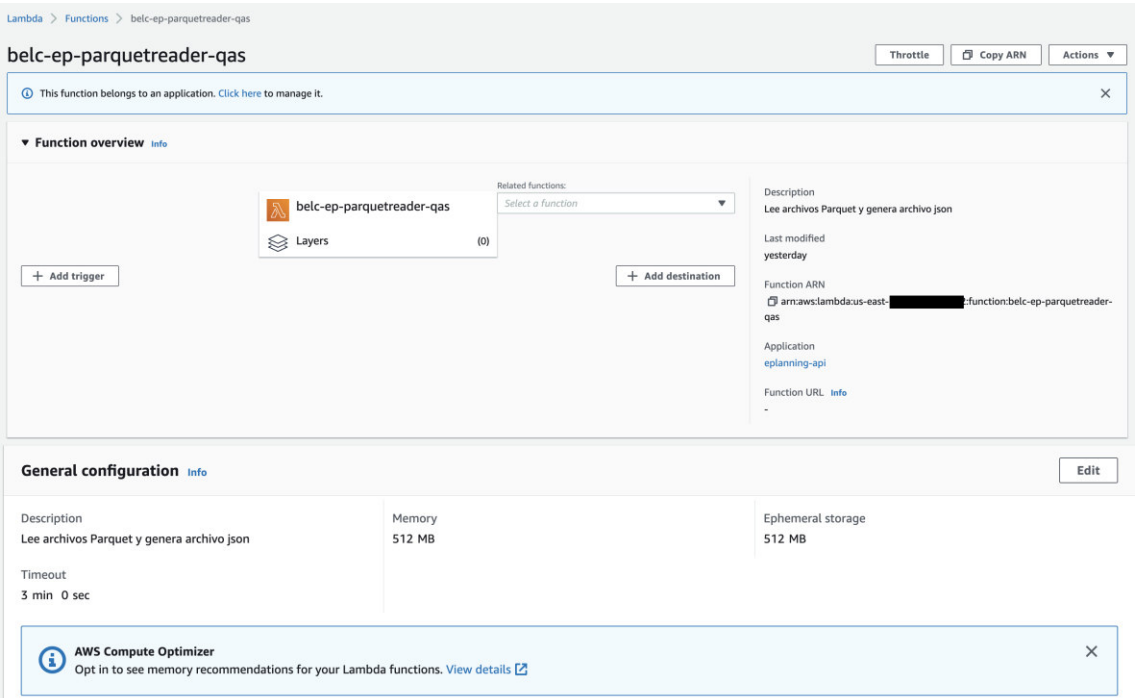
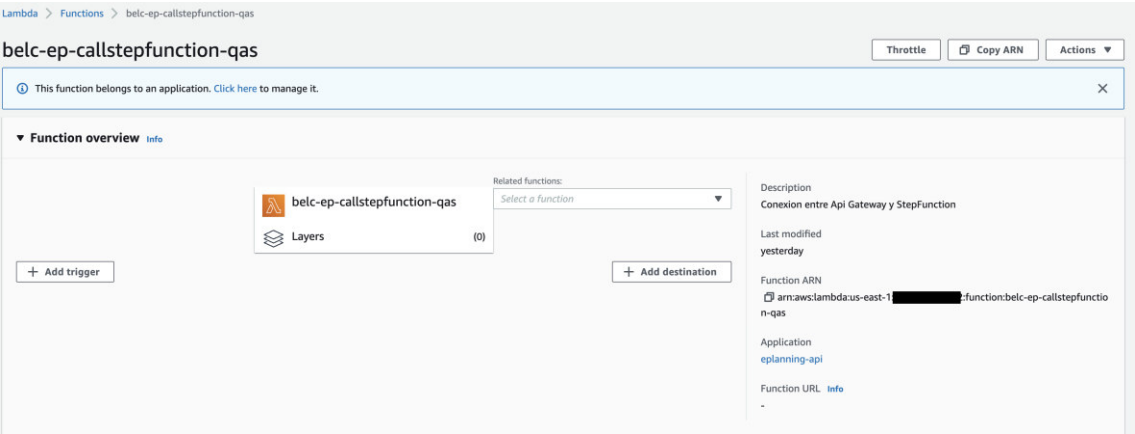


Figura 13. Organigrama Corporación.
Fuente: Autoría propia.

3.2.5.3.2 Sprint II

3.2.5.3.2.2 Implementación de lambda belc-ep-call-step-function

Se implementó lambda de acuerdo a las especificaciones en historia de usuario



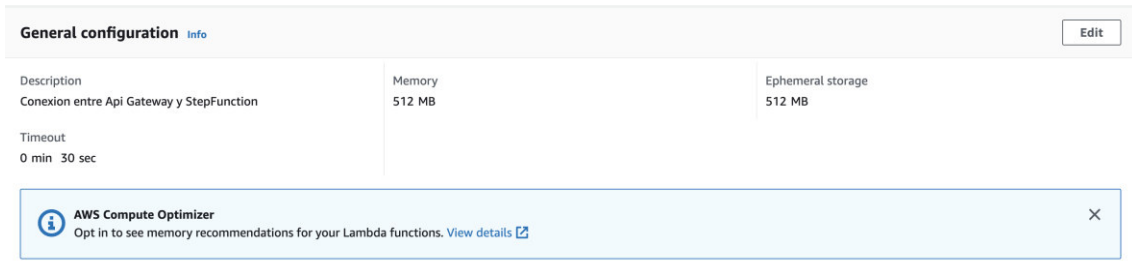


Figura 14. belc-ep-call-step-function.
Fuente: Autoría propia.

3.2.5.3.2 Implementación de lambda belc-ep-desc-commercial

Se implementó lambda de acuerdo a las especificaciones en historia de usuario

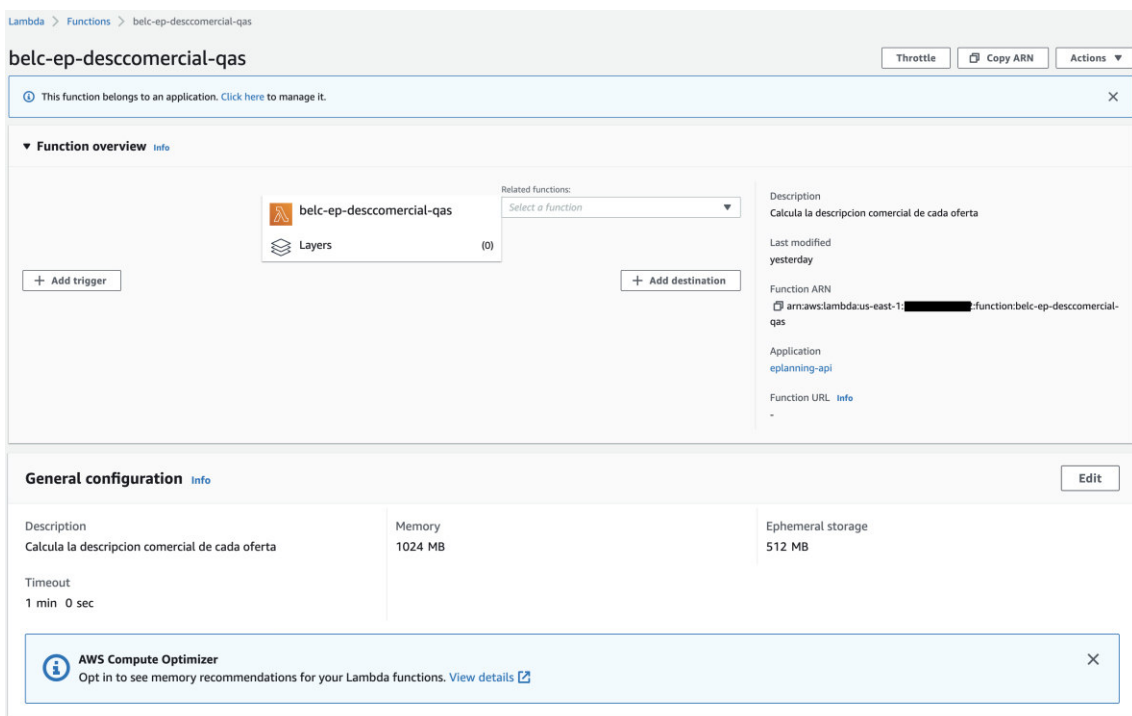


Figura 15. belc-ep-desc-commercial.
Fuente: Autoría propia.

3.2.5.3.2.3 Implementación de lambda belc-ep-rankingtono

Se implementó lambda de acuerdo a las especificaciones en historia de usuario

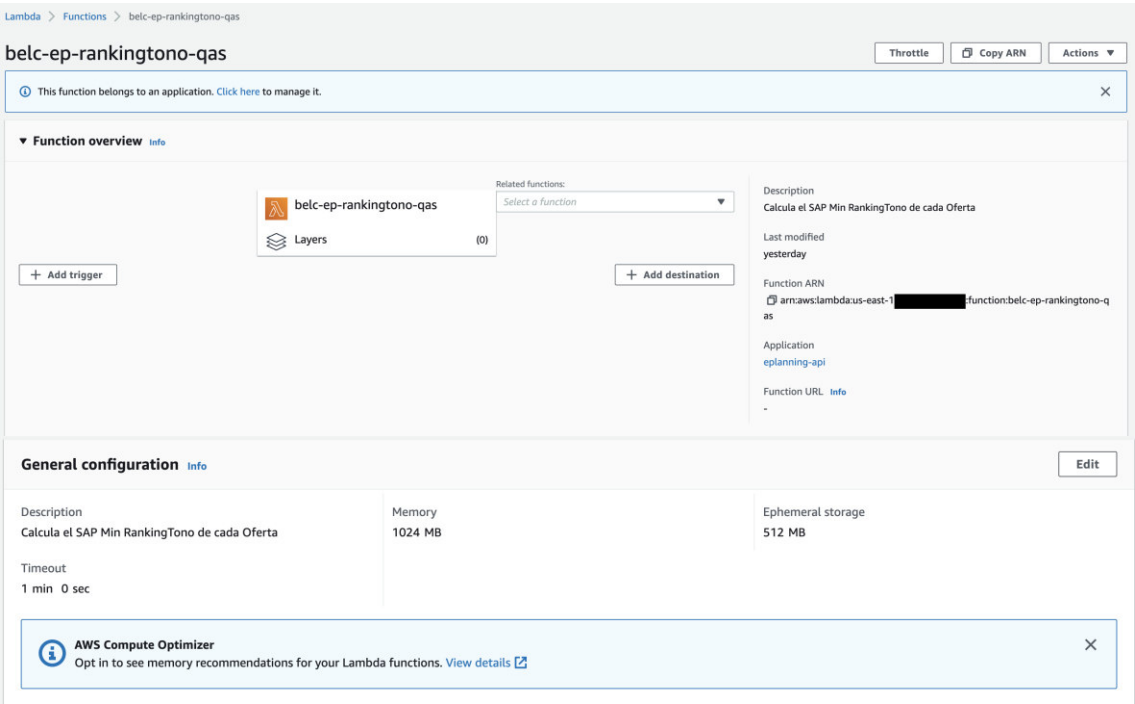


Figura 16. belc-ep-rankingtono.
Fuente: Autoría propia.

3.2.5.3.3 Sprint III

3.2.5.3.3.1 Implementación de lambda belc-ep-offer-type

Se implementó lambda de acuerdo a las especificaciones en historia de usuario

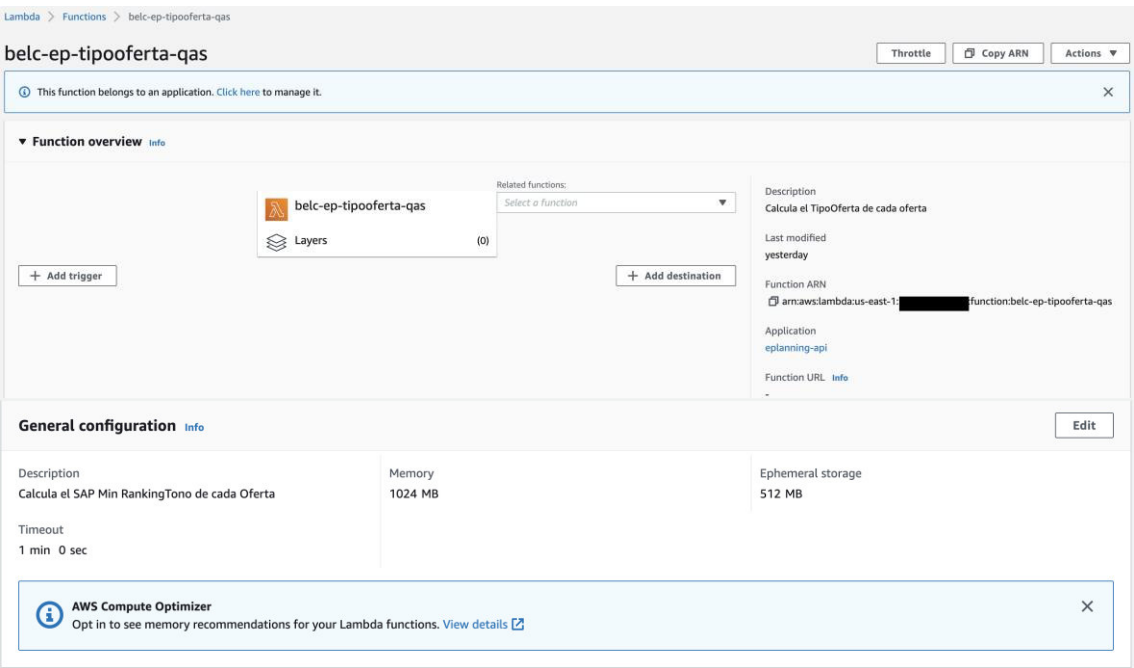


Figura 17. belc-ep-rankingtono.
Fuente: Autoría propia.

3.2.5.3.3.2 Implementación de lambda belc-ep-cuv

Se implementó lambda de acuerdo a las especificaciones en historia de usuario.

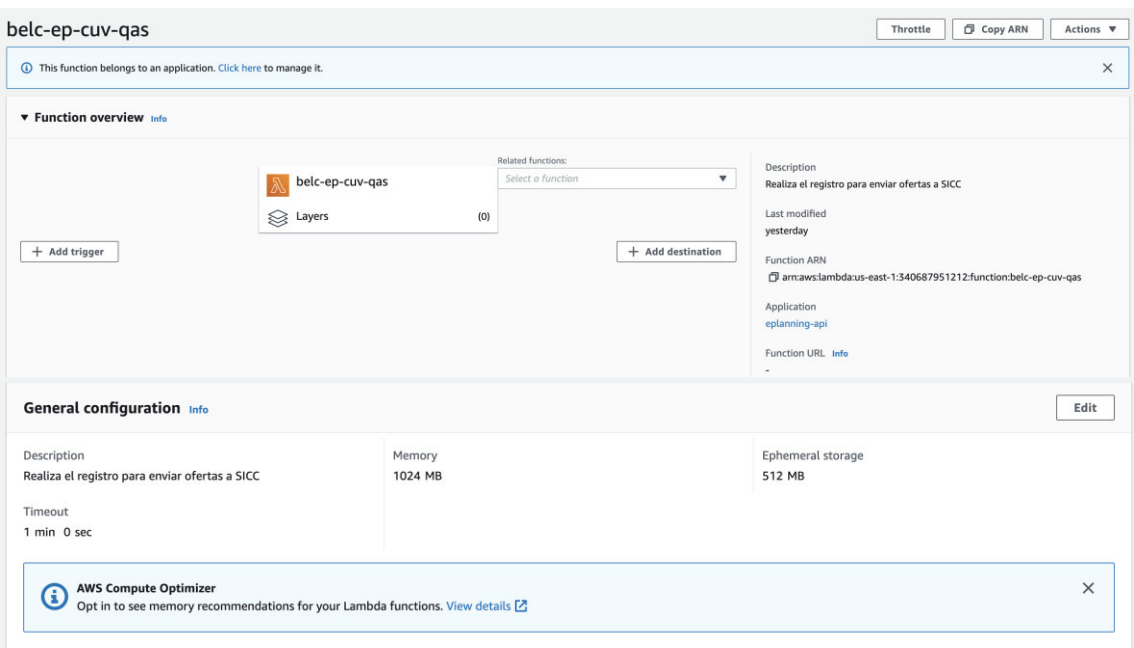


Figura 18. belc-ep-cuv.
Fuente: Autoría propia.

3.2.5.3.3 Implementación de lambda belc-ep-reprocess-description

Se implementó lambda de acuerdo a las especificaciones en historia de usuario.

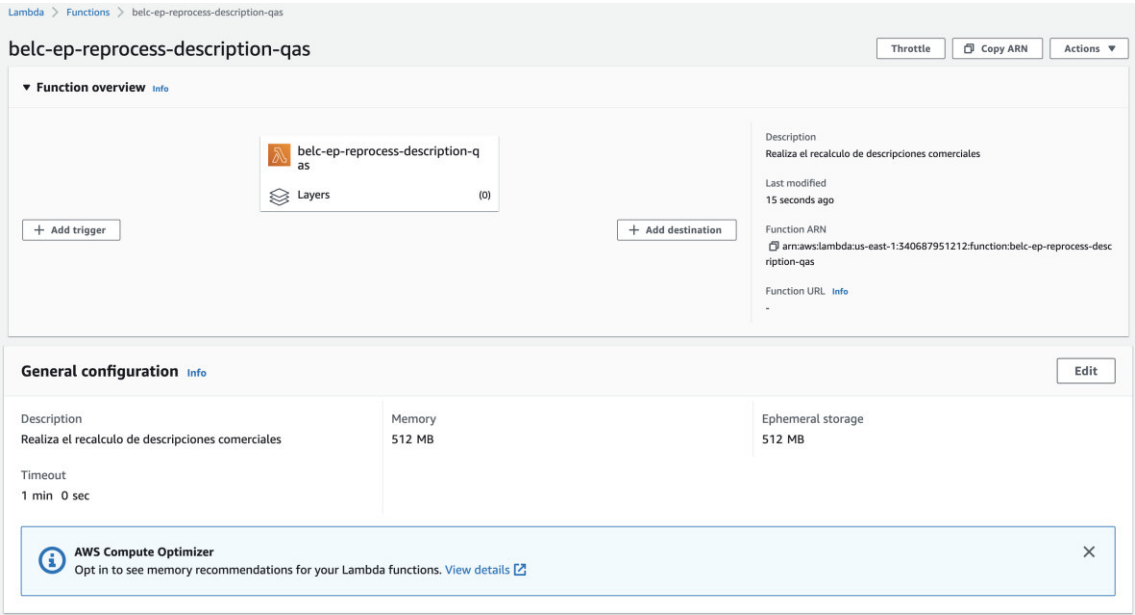


Figura 19. belc-ep-reprocess-description.
Fuente: Autoría propia.

3.2.5.3.3.3 Implementación de lambda belc-ep-call-fargate

Se implementó lambda de acuerdo a las especificaciones en historia de usuario.

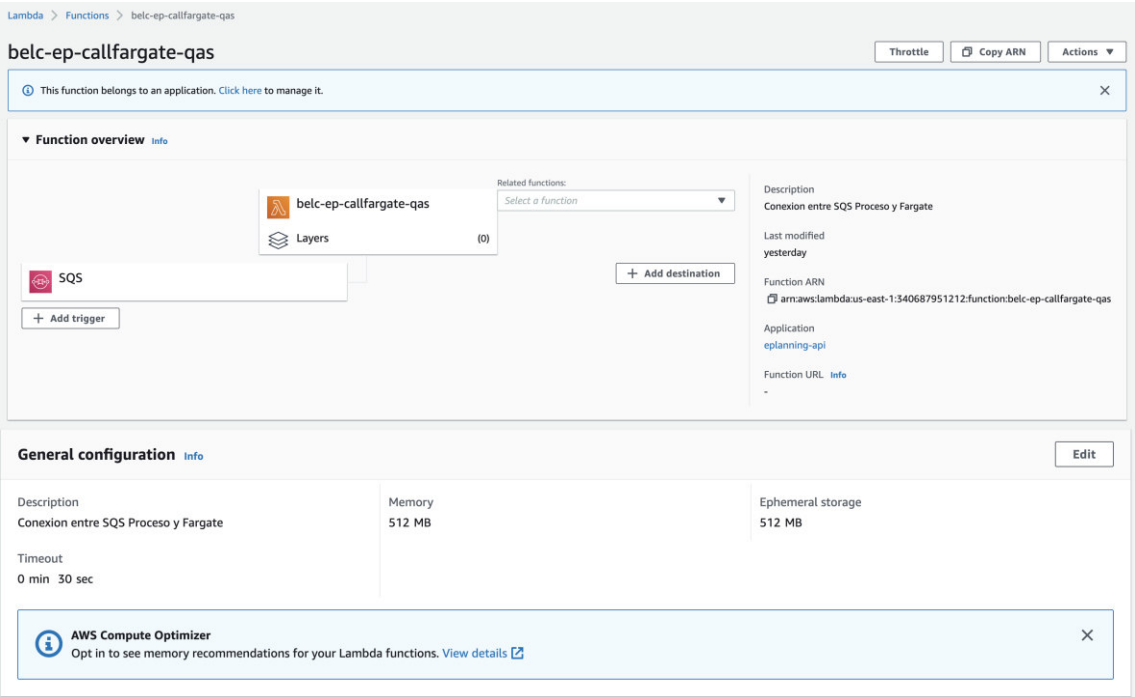


Figura 20. belc-ep-call-fargate.
Fuente: Autoría propia.

3.2.5.3.4 Sprint IV

3.2.5.3.4.1 Implementación de lambda belc-ep-portal-ventas

Se implementó lambda de acuerdo a las especificaciones en historia de usuario.

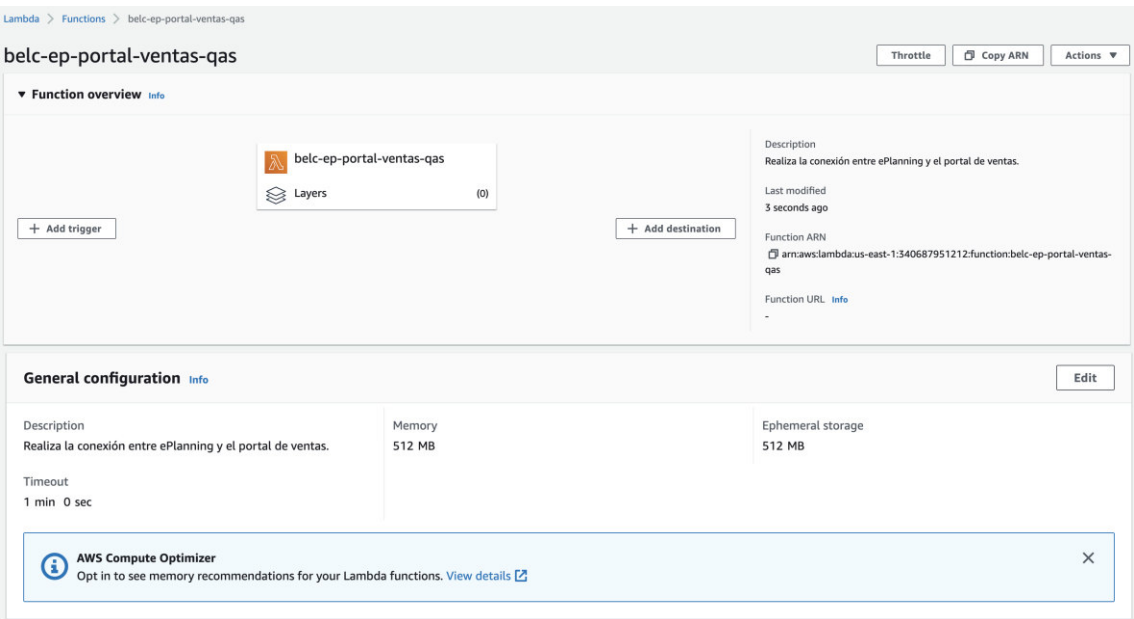


Figura 21. Lambda belc-ep-portal-ventas.

Fuente: Autoría propia.

3.2.5.3.4.2 Implementación de lambda belc-ep-call-cdi

Se implementó lambda de acuerdo a las especificaciones en historia de usuario.

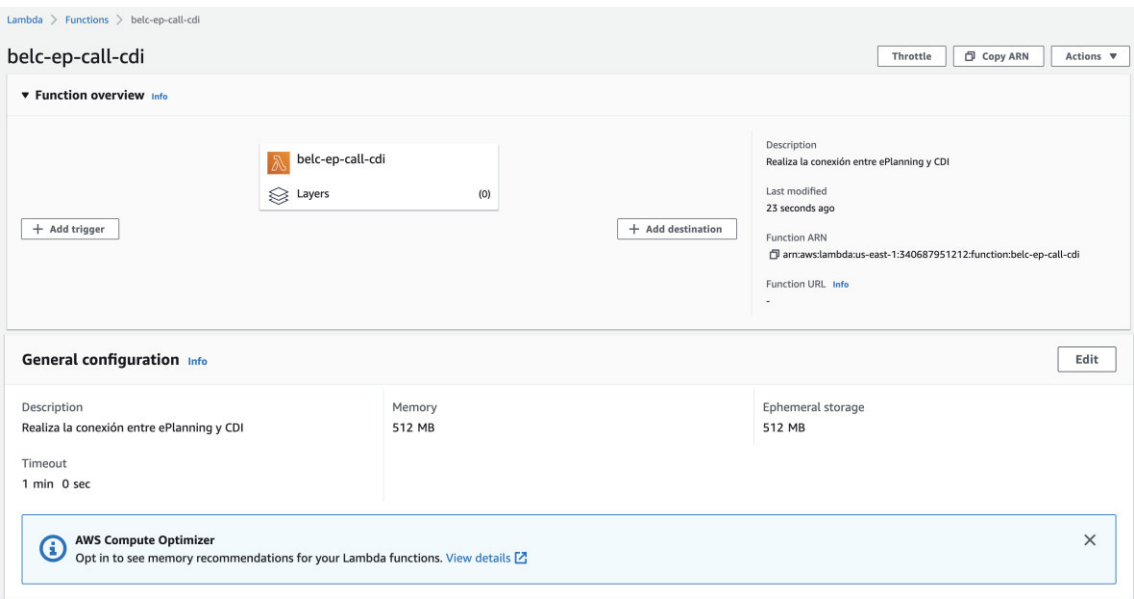


Figura 22. Lambda belc-ep-call-cdi.

Fuente: Autoría propia.

3.2.5.3.4.3 Implementación de sistema dockerizado belc-eplanning-task

Inicialmente se debe crear el componente dockerizado y luego subirlo a ECR desde donde el task lo ejecutará.

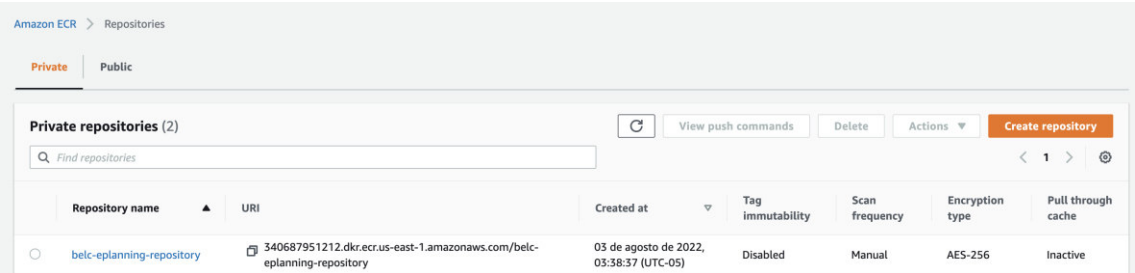


Figura 23. Repostorio docker cargado en ECR.
Fuente: Autoría propia.

Luego se debe crear el cluster

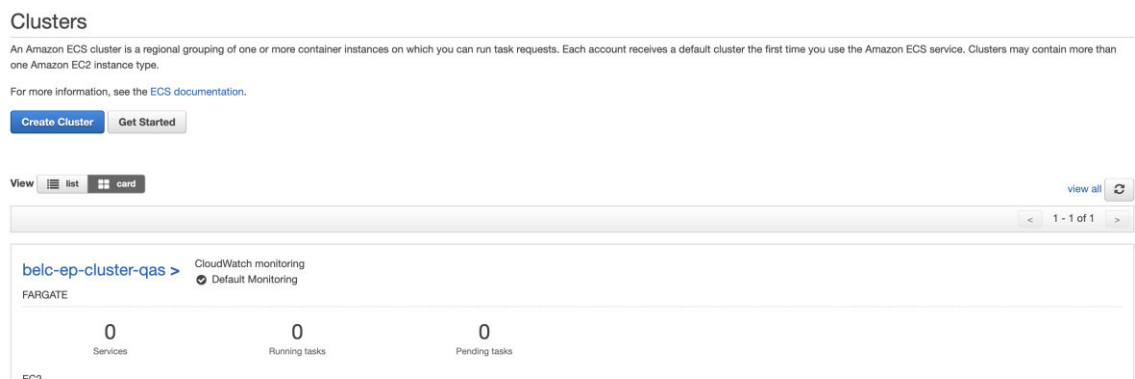


Figura 24. Cluster belc-ep-cluster.
Fuente: Autoría propia.

Y finalmente se crear el task Fargate dentro del cluster.

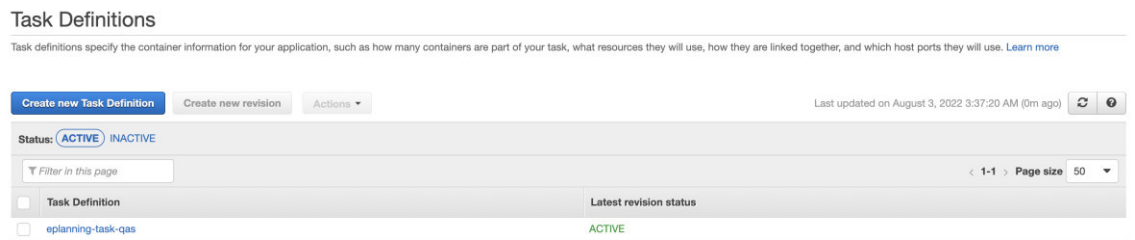


Figura 25. Task Defintion eplanning-task.
Fuente: Autoría propia.

3.2.5.3.5 Sprint V

3.2.5.3.5.1 Implementación de sistema web ePlanning

Este sistema es una web que por fuera expone un Api Gateway con un front hecho en Razor y en el backend ejecuta una lambda, la cual por metodos Post y Get ejecuta las tareas, es la base para las funcionalidades de las siguientes historias.

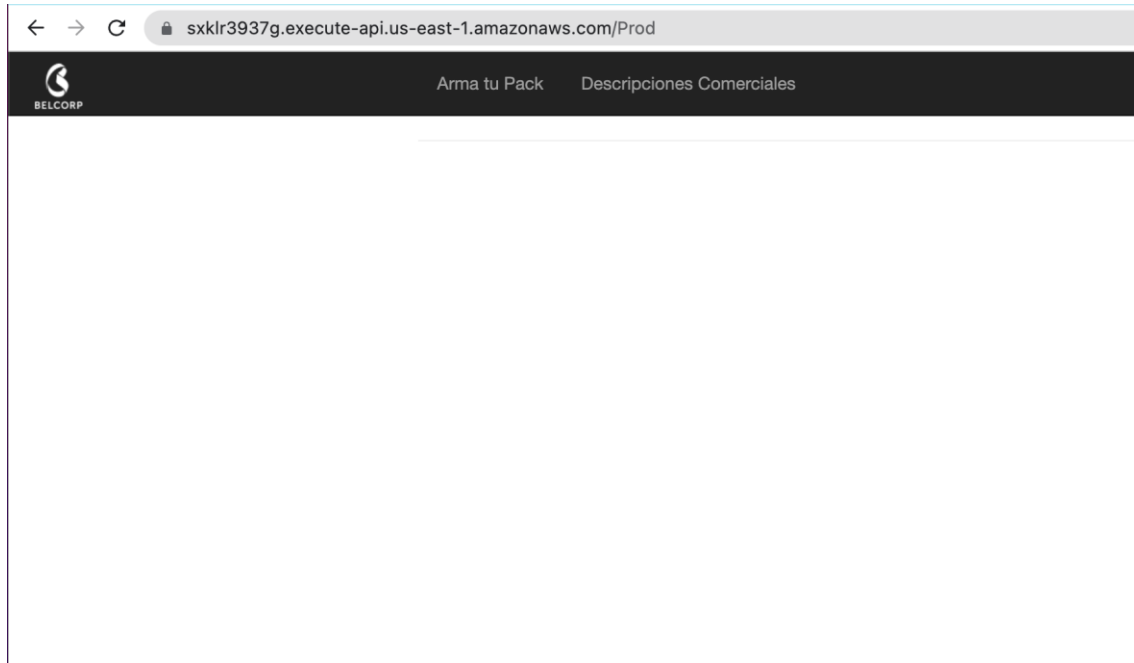


Figura 26. Web ePlanning.

Fuente: Autoría propia.

3.2.5.3.5.2 Implementación módulo ArmatuPack en web ePlanning

Desde este módulo se puede realizar la carga de ofertas para la palanca Arma tu Pack, luego de ser cargadas y ejecutar las validaciones se envían al lambda de Cuvs.

País	Campaña	IdProceso	Fecha Carga	Archivo Carga	ARP
PE	202202	c8d382d5-d457-453c-a717-cdd6095bd9cb	1/2/2020 0:0:0	<button>Descargar</button>	<button>ARP</button>

Figura 27. Módulo ArmatuPack en web ePlanning.

Fuente: Autoría propia.

3.2.5.3.6 Sprint VI

3.2.5.3.6.1 Implementación módulo Descarga de descripciones Comerciales en web ePlanning

Desde este módulo se puede realizar la consulta de las descripciones comerciales generadas para las ofertas, en las cargas realizadas, esto con el fin de realizar alguna validación.

opComercialFiltro

Arma tu Pack Descripciones Comerciales

Descripciones Comerciales

País: Campaña:

País	Campaña	IdProceso	Fecha Carga	Descarga
PE	202202	c8d382d5-d457-453c-a717-cdd6095bd9cb	1/2/2020 0:0:0	Descargar Descripciones

Figura 28. Módulo Descarga de descripciones Comerciales en web ePlanning.
Fuente: Autoría propia.

Adicionalmente a el equipo SCRUM, para la realización de este proyecto se necesitó la colaboración de otras áreas de TI mencionadas a continuación:

Tabla 33 - Apoyo áreas implementación

ROL	Responsabilidades	Responsable
Comité de Arquitectura	Evaluar y aprobar la arquitectura propuesta, para su próxima implementación.	Arquitectos asignados.
Devops	Garantizar que los despliegues se realicen de forma automatizada y de su correcta configuración.	DevOps asignado.
Equipo SICC	Garantizar la correcta integración con sistema SICC, durante la fase de pruebas.	Analista asignado.
Equipo CDI	Garantizar la correcta integración con sistema CDI, durante la fase de pruebas.	Analista asignado.
Equipo Portal Ventas	Garantizar la correcta integración con el sistema Portal de ventas, durante la fase de pruebas.	Analista asignado.

Nota: Autoría propia

3.2.5.4 REVISIÓN Y RETROSPECTIVA

Para la revisión y la retrospectiva de cada Sprint el Scrum Master junto con equipo Scrum mostraban en la Sprint Review los componentes desarrollados al Product Owner, el cual luego daba su conformidad. Adicionalmente a esto se realizaba la retrospectiva, en la cual el Scrum Master y el equipo Scrum, evaluaban el desempeño durante el Sprint, detectando oportunidades y riesgos a tomar en cuenta para los próximos Sprints.

3.2.5.5 LIBERACIÓN

Con los entregables ya cargados en los repositorios, se procede a coordinar el pase a producción con las áreas asignadas. El área de DevOps es el encargado de configurar los ambientes, pipelines y despliegue automático. Así mismo, para la puesta en producción del proyecto, se pidió el apoyo de las siguientes gerencias, con el fin de realizar un despliegue exitoso:

Tabla 34 - Apoyo áreas implementación

ROL	Responsabilidades	Responsable
Devops	Garantizar el correcto despliegue del sistema en producción.	DevOps asignado.
Equipo SICC	Garantizar la correcta integración con sistema SICC	Analista asignado.
Equipo CDI	Garantizar la correcta integración con sistema CDI	Analista asignado.
Equipo Portal de Ventas	Garantizar la correcta integración con sistema Portal de ventas, después del pase a producción.	Analista asignado.
Planning	Validación del flujo de trabajo, mediante ejecuciones en productivo con carga real.	Planner asignado.

Nota: Autoría propia

3.3 EVALUACIÓN

3.3.1 EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO - BENEFICIO

La propuesta económica presentada está basado en calculado de ejecuciones y utilización aproximado, basado en records de procesos a lo largo de la vida de ePlanning. Estos costos son:

- **Costos de implementación – Recurso Humano**
- **Costo de implementación - Tecnología**
- **Costo de operación – Pre Migración.**
- **Costo Operación Post-Migración**

Tabla 35 - Costo de implementación – Recurso Humano

Recurso Requerido	Cantidad	Mes 1	Mes 2	Mes 3	
Desarrollador Senior	1	\$ 2000	\$ 2000	\$ 2000	
Analista Funcional	1	\$ 2000	\$ 2000	\$ 2000	
Calidad	1	\$ 1500	\$ 1500	\$ 1500	
Total		\$ 5500	\$ 5500	\$ 5500	\$ 16500

Nota: Autoría propia.

Tabla 36 - Costo de implementación - Tecnología

Recurso Requerido	Cantidad	Mes 1	
AWS Cloud	1	\$ 340.98	
Total		\$ 340.98	\$ 340.98

Nota: Autoría propia.

Una vez implementada la migración se redujeron los precios de operación hasta casi un 50% por mes, y se demuestra en la siguiente tabla detallada de recursos:

Tabla 37 - Costo Operación Pre-Migración

Recurso	Características	Monto mensual	Monto anual
AWS EC2 Aplicación	Sistema Operativo: Windows Server 2019 Tipo de instancia: m5.xlarge Almacenamiento: 200 Gb	\$ 242.65	\$ 2911.8
AWS EC2 Base de datos	Sistema Operativo: Windows Server 2019 Tipo de instancia: m5.2xlarge Almacenamiento: 200 Gb	\$ 465.30	\$ 5583.6
Total		\$ 707.95	\$ 8495.4

Nota: Autoría propia basada en AWS Pricing Calculator.

Tabla 38 - Costo Operación Post-Migración

Recurso	Cantidad	Características	Monto mensual	Monto anual
AWS Lambda	10	Arquitectura: x86 Número de solicitudes: 10000 Duración de solicitud: 1 minuto Memoria: 512mb Almacenamiento: 512mb	\$ 100	\$ 1200
AWS API Gateway	2	Solicitudes por mes: 5000 Cache: 0	\$ 35	\$ 420
AWS SQS	2	Solicitudes por mes: 10000	\$ 0	\$ 0
AWS Fargate	1	Sistema Operativo: Linux Arquitectura CPU: x86 Número de tareas: 10000 Tiempo ejecución: 30 minutos Monto de vCPU: 0.5 Memoria: 4Gb Almacenamiento: 20	\$190.1	2281.2

AWS ECR	1	Almacenamiento: 100mb	\$ 0.0098	\$ 0.1176
AWS S3	1	Almacenamiento: 500mb Cantidad solicitudes PUT, COPY, POST, LIST: 80000 Cantidad solicitudes GET, SELECT: 80000	\$15	\$180
Total			\$ 340.1098	\$ 4081.3176

Nota: Autoría propia basada en AWS Pricing Calculator.

3.3.2 INTERPRETACIÓN DEL VAN Y DEL TIR

La corporación espera lograr un VAN positivo para este proyecto, este proyecto se debe soportar para los próximos 10 años, así que definiremos 10 periodos de 1 año, tiene como inversión inicial \$ 17000 que se debe retornar al quinto año, y se espera un retorno mensual de \$4500.

Tabla 39 - Flujo de caja

Inversión Inicial	Año 1	Año 2	Año 3	Año 4	Año 5
17000	\$ 4500	\$ 4500	\$ 4500	\$ 4500	\$ 4500
	Año 6	Año 7	Año 8	Año 9	Año 10
	\$ 4500	\$ 4500	\$ 4500	\$ 4500	\$ 4500

Nota: Autoría propia.

El VAN obtenido para el proyecto es de \$ 5584.46, con este resultado se concluye que el proyecto de migración genera rentabilidad, lo que quiere decir que se estarían generando utilidades para este proyecto en un tiempo estimado de 10 años.

Tabla 40 - Comparación VAN/TIR

Tasa de descuento	15 %
VAN	\$ 5584.46
TIR	23.179 %

Nota: Autoría propia.

CAPÍTULO IV: REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

El presente trabajo significó un reto, debido al uso de nuevas tecnologías y personalmente a la inmersión total al mundo de los servicios en la nube, adicionalmente a esto se consolidaron con los marcos de buena arquitectura de AWS y los lineamientos de la corporación.

Durante la fase de análisis y diseño tuvieron ciertos problemas debido a que batallábamos con el costo de la arquitectura, ya que se tenía que lograr el ahorro al máximo, y la corporación fue muy enfática con el presupuesto, existía muy poca documentación funcional de ePlanning lo que incrementaba el riesgo para definir las funcionalidades sin embargo esto pudo ser superado por el conocimiento del sistema del equipo SCRUM y los involucrados que nos apoyaron con un correcto análisis y así poder definir la mayor cantidad de módulos y procesos importantes para el sistema.

El proceso de migración fue paulatina, migrando los países con menos carga y así sucesivamente hasta los países con la carga más pesada, lo que permitía un control sobre cómo iba respondiendo el sistema mientras se aumentaba la carga. Se identificaron un par de bugs mientras se realizaron las pruebas que fueron solucionados con walk-arounds y luego generado tickets para la atención y su resolución.

El proyecto se vio enriquecido por utilizar, marcos y metodologías robustas, lo que facilitaba la implementación y la correcta comunicación con otros equipos.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se logró diseñar una arquitectura orientada a microservicios y sin servidor apta para bajas y altas cargas de trabajo, con buen performance y escalabilidad.
- La migración de ePlanning se realizó de manera satisfactoria y sin contratiempos, logrando implementar el total de los módulos y procesos que tenía este sistema en su arquitectura anterior.
- El sistema funciona con fluidez ante cargas más pesadas, ahora se realizan con más rapidez, y sin interrupciones.
- Se ha obtenido un ahorro de costes de operación aproximado del 50%, debido al enfoque sin servidor, solo se paga mientras se está ejecutando la tarea.

5.2 RECOMENDACIONES

- Actualmente la solución se encuentra en una sola región de AWS, lo que la hace vulnerable a una caída de los servicios en esa región, se podría complementar con una arquitectura multi región, para garantizar una mayor disponibilidad del servicio.
- Se puede agregar algún proceso de notificación para el usuario de que culminó su proceso de carga, ya que actualmente el usuario debe dar seguimiento al proceso desde la aplicación.

FUENTES DE INFORMACIÓN

- Amazon Web Services. (2022) https://aws.amazon.com/es/?nc2=h_lg
- Amazon Web Services - AWS Well-Architected Framework (December 2021) <https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html>
- SCRUM <https://www.scrum.org/>

- Microservicios en AWS
https://docs.aws.amazon.com/es_es/whitepapers/latest/microservices-on-aws/microservices.html
- Interfaz de Usuario en AWS estático
https://docs.aws.amazon.com/es_es/whitepapers/latest/microservices-on-aws/user-interface.html
- ¿Qué son y para qué sirven los microservicios?
<https://www.redhat.com/es/topics/microservices>
- ¿Qué es la informática sin servidor? <https://www.redhat.com/es/topics/cloud-native-apps/what-is-serverless>
- Amazon Leads \$180-Billion Cloud Market
<https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>
- Tecnología sin servidor en AWS <https://aws.amazon.com/es/getting-started/deep-dive-serverless/>
- CLI de .NET Core https://docs.aws.amazon.com/es_es/lambda/latest/dg/csharp-package-cli.html
- .NET Core 3.1 <https://dotnet.microsoft.com/en-us/download/dotnet/3.1>
- Apache Parquet for .NET <https://github.com/alonguid/parquet-dotnet>
- CyberTesis UNMSM <https://cybertesis.unmsm.edu.pe/>
- Van y TIR: Concepto, diferencias y cómo calcularlos
<https://economia3.com/van-tir-concepto-diferencias-como-calcularlos/>
- Qué es .NET Core <https://openwebinars.net/blog/que-es-net-core/>

5.4 GLOSARIO

- **Oferta:** Es el producto ya sea individual o agrupamiento de productos que serán presentados a las consultoras.
- **Descripción Comercial:** Es el nombre final que tendrá la oferta, puede ser un pack, individual, etc.

- **Ranking Tono:** Es el tono más vendido de cierto producto que cuenta con tonos.
- **Tipo de oferta:** Según la composición de la oferta, es categorizada en un tipo de oferta.
- **Consultora:** Son las personas que realizan los pedidos.
- **Planner:** Es el usuario del sistema ePlanning, específicamente personal de esta empresa de maquillaje del área de planeamiento.
- **SICC:** Sistema de Información Comercial Corporativo, es el sistema legado y Core de la corporación, es el repositorio final de las ofertas con los datos cargados correctamente y el encargado de generar el CUV.
- **CDI:** Composición de imágenes, es el sistema encargado de graficar la imagen de la oferta que se mostrará.
- **Portal de Ventas:** Es el sistema web donde las consultoras pueden realizar sus pedidos.
- **CUV:** Código único de venta, es el código generado por el sistema SICC, es el código final de oferta por cada campaña y país.
- **Parquet:** Parquet es un formato de columnas que es compatible con muchos otros sistemas de procesamiento de datos. Spark SQL brinda soporte para leer y escribir archivos de Parquet que conservan automáticamente el esquema de los datos originales

ANEXOS

ANEXO 1: ARCHIVO. YML QUE GENERA LA ARQUITECTURA POR CLOUDFORMATION PARA DESPLIEGUE AUTOMATICO.

```
---
AWSTemplateFormatVersion: '2010-09-09'
```

```

Description:ArquitecturaEplanning

#Parameters:
#SubnetParameter:
#Type:String
#Description:EntersubnetforEplanningAPI

#SecurityGroupParameter:
#Type:String
#Description:EnterSecurityGroupforEplanningAPI

Resources:

EPSQSProceso:
Type::AWS::SQS::Queue
Properties:
ContentBasedDeduplication:false
DelaySeconds:0
FifoQueue:true
QueueName:SQS-ep-sicc-proceso-prd.fifo
ReceiveMessageWaitTimeSeconds:0
VisibilityTimeout:60
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPSQSControl:
Type::AWS::SQS::Queue
Properties:
ContentBasedDeduplication:false
DelaySeconds:0
FifoQueue:true
QueueName:SQS-ep-sicc-control-prd.fifo
ReceiveMessageWaitTimeSeconds:0
VisibilityTimeout:0
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPParquetReader:
Type::AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-parquetreader.zip
Description:LeearchivosParquetygeneraarchivojson
FunctionName:belc-ep-parquetreader-prd
Environment:
Variables:
BUCKET_EPLANNING:belc-eplanning-base-prd
Handler:ParquetReader::ParquetReader.Function::FunctionHandler

```

```

MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleS3EplanningPRD
Runtime:dotnetcore3.1
Timeout:180
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPDescripcionComercial:
Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-desccomercial.zip
Description:Calculaladescripcioncomercialdecadaoferta
FunctionName:belc-ep-desccomercial-prd
Environment:
Variables:
BUCKET_EPLANNING:belc-eplanning-base-prd
Handler:XXXXXX.ePlanning.API.DescripcionComercial::XXXXXX.ePlanning.API.DescripcionComercial.Function::FunctionHandler
MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleS3EplanningPRD
Runtime:dotnetcore3.1
Timeout:60
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPTipoOferta:
Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-tipooferata.zip
Description:Calculalatipodeofertadecadaoferta
FunctionName:belc-ep-tipooferata-prd
Environment:
Variables:
BUCKET_EPLANNING:belc-eplanning-base-prd
Handler:XXXXXX.ePlanning.API.DescripcionComercial::XXXXXX.ePlanning.API.DescripcionComercial.Function::FunctionHandler
MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleS3EplanningPRD
Runtime:dotnetcore3.1
Timeout:60
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPRankingTono:

```

```

Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-rankingtono.zip
Description:CalculaelSAPMinRankingTonodecadaOferta
FunctionName:belc-ep-rankingtono-prd
Environment:
Variables:
BUCKET_EPLANNING:belc-eplanning-base-prd
Handler:XXXXXX.ePlanning.API.Ranking::XXXXXX.ePlanning.API.Ranking.Function::FunctionHandler
MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleS3EplanningPRD
Runtime:dotnetcore3.1
Timeout:60
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPCuv:
Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-cuv.zip
Description:RealizaelregistroparaenviarofertasaSICC
FunctionName:belc-ep-cuv-prd
Environment:
Variables:
SQS_PROCESO:SQS-ep-sicc-proceso-prd.fifo
Handler:XXXXXX.ePlanning.API.Cuv::XXXXXX.ePlanning.API.Cuv.Function::FunctionHandler
MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleInsertSQSEplanningPRD
Runtime:dotnetcore3.1
Timeout:60
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPSendSomosBelcorp:
Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-sendsm.zip
Description:RealizaelregistroparaenviarofertasaSICC
FunctionName:belc-ep-sendsm-prd
Environment:
Variables:
BUCKET_EPLANNING:belc-eplanning-base-prd
Handler:XXXXXX.ePlanning.API.sendsm::XXXXXX.ePlanning.API.sendsm.Function::FunctionHandler

```

```

MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleInsertSQSEplanningPRD
Runtime:dotnetcore3.1
Timeout:60
Tags:
- Key:Grupo
  Value:EPLANNING-V2

EPSendCDI:
Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-sendedi.zip
Description:RealizaelenvioaCDI
FunctionName:belc-ep-sendedi-prd
Environment:
Variables:
BUCKET_EPLANNING:belc-eplanning-base-prd
Handler:XXXXXX.ePlanning.API.sendedi::XXXXXX.ePlanning.sendedi.Cuv.Function::FunctionHandler
MemorySize:1024
Role:arn:aws:iam::820233355588:role/RoleInsertSQSEplanningPRD
Runtime:dotnetcore3.1
Timeout:60
Tags:
- Key:Grupo
  Value:EPLANNING-V2

EPCallStepFunction:
Type:AWS::Lambda::Function
Properties:
Code:
S3Bucket:XXXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-callstepfunction.zip
Description:ConexionentreApiGatewayyStepFunction
FunctionName:belc-ep-callstepfunction-prd
Environment:
Variables:
STATE_MACHINE:SF-eplanning-machine-prd
Handler:CallStepFunction::CallStepFunction.Function::FunctionHandler
MemorySize:512
Role:arn:aws:iam::820233355588:role/RoleExecStepEplanningPRD
Runtime:dotnetcore3.1
Timeout:30
Tags:
- Key:Grupo
  Value:EPLANNING-V2

EPCallFargate:
Type:AWS::Lambda::Function
Properties:

```

```

Code:
S3Bucket:XXXXX-codedeploy
S3Key:eplanning/PRD/belc-ep-callfargate.zip
Description:ConexionentreSQSProcesoyFargate
FunctionName:belc-ep-callfargate-prd
Environment:
Variables:
SQS_CONTROL:SQS-ep-sicc-control-prd.fifo
SQS_PROCESO:SQS-ep-sicc-proceso-prd.fifo
TIEMPO_REINTENTO_MS:'120000'
SUBNET:"
SECURITYGROUP:"
TASKDEFINITION:"
CLUSTER:belc-ep-cluster-prd
Handler:CallFargate::CallFargate.Function::FunctionHandler
MemorySize:512
Role:arn:aws:iam::820233355588:role/RoleSQSEplanningPRD
Runtime:dotnetcore3.1
Timeout:30
Tags:
-Key:Grupo
Value:EPLANNING-V2

EPStateMachine:
Type:AWS::StepFunctions::StateMachine
Properties:
StateMachineName:SF-eplanning-machine-prd
StateMachineType:STANDARD
DefinitionString:
Fn::Sub:
-{\nComment:\EplanningAmazonStatesLanguageusingPassstates\,\n
\StartAt:\CallLambdaParquetReader\,\nStates:{\n\CallLambdaParquetReader\:
{\n\Type:\Task\,\n\Resource:\${lambdaParquetReader}\,\n\Next\:
\CallLambdaDesc_Comer\,\n\CallLambdaDesc_Comer:{\n\Type:\Task\,\n\Resource:\${lambdadesccomercial}\,\n
\Next:\CallLambdaRanking\,\n\CallLambdaRanking:{\n\Type:
\Task\,\n\Resource:\${lambdarankingtono}\,\n\Next:\CallLambdaCuv\
\,\n\CallLambdaCuv:{\n\Type:\Task\,\n\Resource:\${lambdacuv}\,\n
\End\:true\,\n}\,\n}
-lambdaParquetReader:
Fn::GetAtt:
-EPParquetReader
-Arn
lambdadesccomercial:
Fn::GetAtt:
-EPDescripcionComercial
-Arn
lambdarankingtono:
Fn::GetAtt:
-EPRankingTono
-Arn

```

```

lambdacuv:
  Fn::GetAtt:
  - EPCuv
  - Arn
  RoleArn:arn:aws:iam::820233355588:role/RoleStepFunctionEplanningPRD
  Tags:
  - Key:Grupo
    Value:EPLANNING-V2

EPEventSource:
  Type:AWS::Lambda::EventSourceMapping
  Properties:
    BatchSize:1
    Enabled:true
    EventSourceArn:
    Fn::GetAtt:
    - EPSQSProceso
    - Arn
  FunctionName:
  Fn::GetAtt:
  - EPCallFargate
  - Arn

EPAPI:
  Type:AWS::ApiGateway::RestApi
  Properties:
    EndpointConfiguration:
    Types:
    - REGIONAL
    Description:EndPointEplanning
    Name:belc-ep-api-prd
    Tags:
    - Key:Grupo
      Value:EPLANNING-V2
  Body:
    swagger:'2.0'
    info:
      description:API
      version:1.0.0
      title:EplanningAPI
    schemes:
    - https
    paths:
    /:{}
  /Eplanning:
    post:
      produces:
      - application/json
      - application/customer-error
      - application/api_not_found
      - application/no_record

```

```

-application/api_error
responses:
'200':
description:200response
schema:
$ref:#/definitions/Empty
'201':
description:201response
'204':
description:204response
'230':
description:230response
'231':
description:231response
'232':
description:232response
'233':
description:233response
x-amazon-apigateway-integration:
responses:
default:
statusCode:'200'
uri:
Fn::Join:
-:
--arn:aws:apigateway
-Ref:AWS::Region
-lambda:path/2015-03-31/functions/arn:aws:lambda
-Ref:AWS::Region
-Ref:AWS::AccountId
-function
-Fn::Join:
_"
--Ref:EPCallStepFunction
-/invocations
passthroughBehavior:when_no_match
httpMethod:POST
contentHandling:CONVERT_TO_TEXT
type:aws
definitions:
Empty:
type:object
title:EmptySchema

EPApiStage:
Type:AWS::ApiGateway::Stage
Properties:
DeploymentId:
Ref:EPApiDeployment
Tags:
-Key:Grupo

```

```

Value:EPLANNING-V2
MethodSettings:
-DataTraceEnabled:false
HttpMethod:*
ResourcePath:/*
RestApiId:
Ref:EPAPI
StageName:prd

EPApiDeployment:
Type:AWS::ApiGateway::Deployment
DependsOn:
-EPAPI
Properties:
RestApiId:
Ref:EPAPI
StageName:dummy

EPLambdaPermissions:
Type:AWS::Lambda::Permission
DependsOn:
-EPCallStepFunction
Properties:
Action:lambda:InvokeFunction
FunctionName:
Ref:EPCallStepFunction
Principal:apigateway.amazonaws.com

EPCluster:
Type:AWS::ECS::Cluster
Properties:
ClusterName:belc-ep-cluster-prd
Tags:
-Key:Grupo
Value:EPLANNING-V2

LogGroup:
Type:AWS::Logs::LogGroup
Properties:
LogGroupName:ecs/td-eplaning-prd

EPTask:
Type:AWS::ECS::TaskDefinition
Properties:
Family:eplanning-task-prd
Cpu:256
Memory:512
NetworkMode:awsvpc
RequiresCompatibilities:
-FARGATE
ExecutionRoleArn:arn:aws:iam::820233355588:role/ecsTaskExecutionRole

```

```

TaskRoleArn:arn:aws:iam::820233355588:role/RoleExecFargateEplanningPRD
Tags:
- Key:Grupo
  Value:EPLANNING-V2
ContainerDefinitions:
- Name:eplanning-api-container
  Image:820233355588.dkr.ecr.us-east-1.amazonaws.com/eplanning/siccservice:latest#RevisarImage
  Cpu:256
  Memory:512
LogConfiguration:
  LogDriver:awslogs
Options:
  awslogs-group:!RefLogGroup
  awslogs-region:us-east-1
  awslogs-stream-prefix:ecs

```

ANEXO 2: ARCHIVO .JSON QUE GENERA LOS ROLES Y POLÍTICAS DE ACCESO ENTRE RECURSOS DE AWS.

```

-->RolEjecucionLambda(StepFunction)
Role:RoleStepFunctionEplanningPRD
Policy:AWSLambdaRole
{
  Version:2012-10-17,
  Statement:[
    {
      Effect:Allow,
      Action:[
        lambda:InvokeFunction
      ],
      Resource:*
    }
  ]
}
TrustRelationShip:states.amazonaws.com

-->RolEjecucionStepFunction(Lambda)
Role:RoleExecStepEplanningPRD
Policy:AWSLambdaBasicExecutionRole
InlinePolicy:
{
  Version:2012-10-17,
  Statement:[
    {
      Effect:Allow,
      Action:[
        states:DescribeExecution,

```

```

states:ListExecutions,
states:GetExecutionHistory,
states:StartExecution
],
Resource:[arn:aws:states:us-east-1:820233355588:stateMachine:SF-eplanning-machine-prd,
arn:aws:states:us-east-1:820233355588:execution:SF-eplanning-machine-prd/*]
}
]
}
TrustRelationShip:lambda.amazonaws.com

-->RolAccesoS3(ParquetReader,DescripcionComercial,Ranking,Offertype,sendCDI,sendSomosBelcorp)(Lambda)
Role:RoleS3EplanningPRD
Policy:AWSLambdaBasicExecutionRole
InlinePolicy:
{
Version:2012-10-17,
Statement:[
{
Effect:Allow,
Action:[
s3:PutObject,
s3:GetObject,
s3:ListBucket,
s3:PutObjectAcl
],
Resource:[
arn:aws:s3:::belc-eplanning-base-prd,
arn:aws:s3:::belc-eplanning-base-prd/*,
arn:aws:s3:::belc-ccss-cm-data-prd/*,
arn:aws:s3:::belc-ccss-cm-data-prd
]]}]
TrustRelationShip:lambda.amazonaws.com

-->RolCuv(Lambda)
Role:RoleInsertSQSEplanningPRD
Policy:AWSLambdaBasicExecutionRole
InlinePolicy:
{
Version:2012-10-17,
Statement:[
{
Effect:Allow,
Action:[
sqs:GetQueueUrl,
sqs:SendMessage
],
Resource:arn:aws:sqs:us-east-1:820233355588:SQS-ep-sicc-control-prd.fifo,
arn:aws:sqs:us-east-1:820233355588:SQS-ep-sicc-proceso-prd.fifo
}
]

```

```

}
TrustRelationShip:lambda.amazonaws.com

-->RolInvocaFargate(Lambda)
Role:RoleSQSEplanningPRD
Policy:AWSLambdaBasicExecutionRole
InlinePolicy:
{
Version:2012-10-17,
Statement:[
{
Sid:VisualEditor0,
Effect:Allow,
Action:ecs:RunTask,
Resource:arn:aws:ecs:us-east-1:820233355588:task-definition/eplanning-task-prd,
Condition:{
ArnEquals:{
ecs:cluster:arn:aws:ecs:us-east-1:820233355588:cluster/belc-ep-cluster-prd}
}
},
{
Sid:VisualEditor1,
Effect:Allow,
Action:[
sqs:DeleteMessage,
sqs:GetQueueUrl,
iam:PassRole,
sqs:ReceiveMessage,
sqs:SendMessage,
sqs:GetQueueAttributes
],
Resource:[
arn:aws:sqs:us-east-1:820233355588:SQS-ep-sicc-proceso-prd.fifo,
arn:aws:sqs:us-east-1:820233355588:SQS-ep-sicc-control-prd.fifo,
arn:aws:iam:820233355588:role/RoleExecFargateEplanningPRD,
arn:aws:iam:820233355588:role/ecsTaskExecutionRole
]
}
]
}
TrustRelationShip:lambda.amazonaws.com

-->EnvioSICC(EC2Fargate)
Role:RoleExecFargateEplanningPRD
Policy:AmazonECSTaskExecutionRolePolicy
InlinePolicy:
{
Version:2012-10-17,
Statement:[
{
Effect:Allow,

```

```

Action:[
sqs:GetQueueUrl,
sqs:ReceiveMessage,
sqs:SendMessage,
sqs:DeleteMessage
s3:PutObject,
s3:GetObject,
s3:ListBucket,
s3:PutObjectAcl,
sts:AssumeRole
],
Resource:[arn:aws:iam::820233355588:role/RoleSQSEplanningPRD,
arn:aws:iam::820233355588:role/ecsTaskExecutionRole,
arn:aws:sqs:us-east-1:820233355588:SQS-ep-sicc-proceso-prd.fifo,
arn:aws:sqs:us-east-1:820233355588:SQS-ep-sicc-control-prd.fifo,
arn:aws:s3:::belc-eplanning-base-prd,
arn:aws:s3:::belc-imagecompositor-base-prd,
arn:aws:s3:::belc-ccss-cm-data-prd,
arn:aws:s3:::belc-eplanning-base-prd/*,
arn:aws:s3:::belc-imagecompositor-base-prd/*,
arn:aws:s3:::belc-ccss-cm-data-prd/*]
}
]
}
TrustRelationShip:ecs-tasks.amazonaws.com

```

ANEXO 3: ARCHIVO CONFIG .JSON DEL SISTEMA

```

{
MX:{
Host: XXXXXXXX,
Puerto:9999,
ServiceName:svha_mxl,
Usuario:XXXXXXX,
Password:XXXXXXX$,
Esquema:SSICC_MX_LB
},
CL:{

```

```
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_cle,  
Usuario:XXXXXXX,  
Password:XXXXXXX$,  
Esquema:SSICC_CL_ES  
},  
CO:{  
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_coe,  
Usuario:XXXXXXX,  
Password:XXXXXXX$,  
Esquema:SSICC_CO_ES  
},  
EC:{  
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_ecl,  
Usuario:XXXXXXX,  
Password:XXXXXXX$,  
Esquema:SSICC_EC_LB  
},  
GT:{  
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_gte,  
Usuario:XXXXXXX,  
Password:XXXXXXX$,  
Esquema:SSICC_GT_ES  
},  
CR:{  
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_crl,  
Usuario:XXXXXXX,  
Password:XXXXXXX$,  
Esquema:SSICC_CR_LB  
},  
PA:{  
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_pal,  
Usuario:XXXXXXX,  
Password:XXXXXXX$,  
Esquema:SSICC_PA_LB  
},  
PR:{  
Host: XXXXXXXX,  
Puerto: 9999,  
ServiceName:srvha_prl,
```

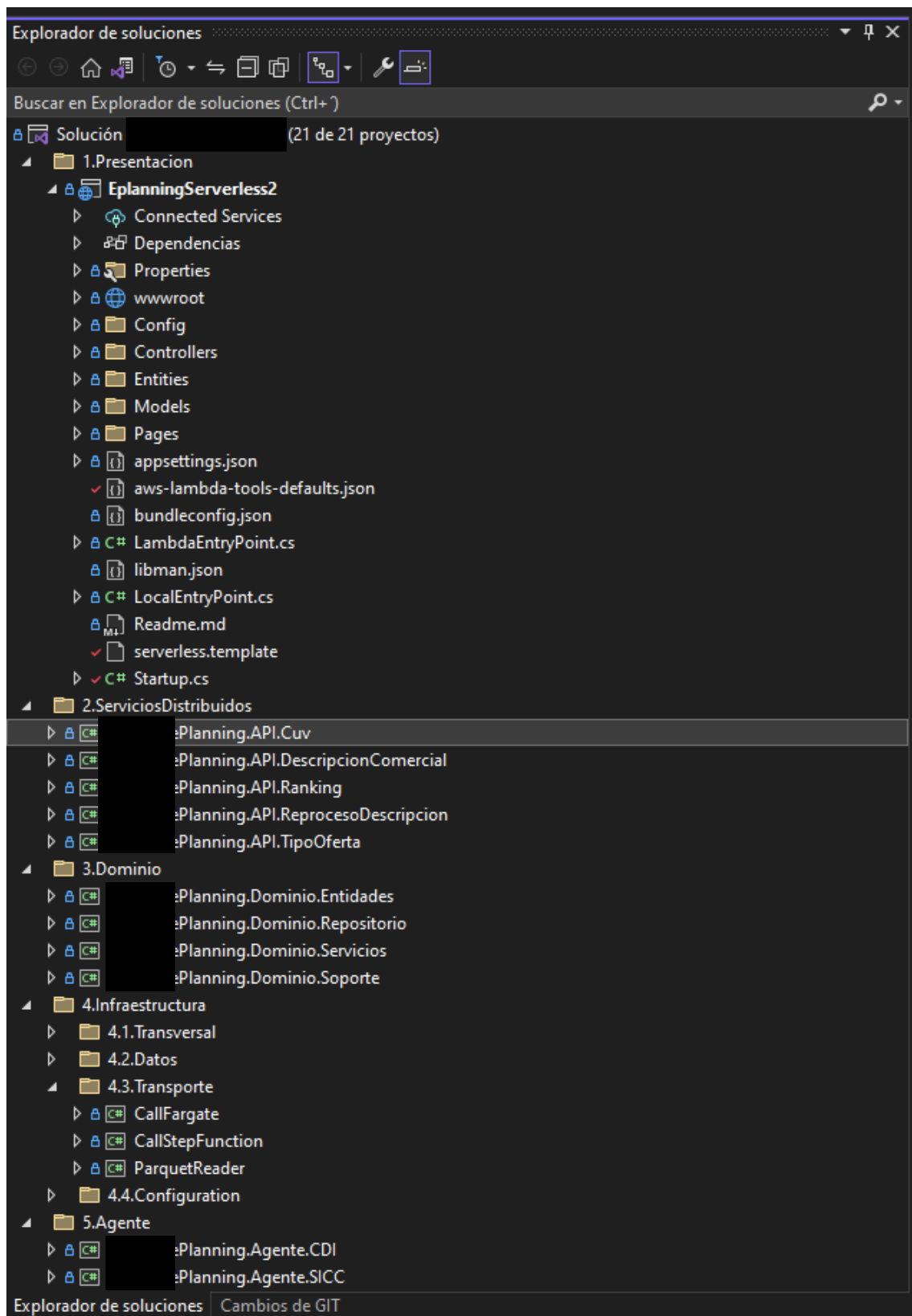
```

Usuario:XXXXXXX,
Password:XXXXXXX$,
Esquema:SSICC_PR_LB
},
DO:{
Host: XXXXXXXX,
Puerto: 9999,
ServiceName:svha_dol,
Usuario:XXXXXXX,
Password:XXXXXXX$,
Esquema:SSICC_DO_LB
},
SV:{
Host: XXXXXXXX,
Puerto: 9999,
ServiceName:svha_sve,
Usuario:XXXXXXX,
Password:XXXXXXX$,
Esquema:SSICC_SV_ES
},
PE:{
Host: XXXXXXXX,
Puerto: 9999,
ServiceName:svha_pee,
Usuario:XXXXXXX,
Password:XXXXXXX$,
Esquema:SSICC_PE_ES
},
BO:{
Host: XXXXXXXX,
Puerto: 9999,
ServiceName:svha_boe,
Usuario:XXXXXXX,
Password:XXXXXXX$,
Esquema:SSICC_BO_ES
},
API-BDI:{
BUCKET:belc-eplanning-base-prd
},
PortalVentasPath:{
PathPortalVentas:DescripcionesComerciales/$Pais/,
FilePortalVentasB:$Pais_DescripcionesSet_$Palanca_$Campana_$uuid.csv,
FilePortalVentasD:$Pais_DescripcionesSet_$Palanca_$Campana_$uuid.csv
},
CDI:{
Bucket:belc-imagecompositor-base-prd,
InputPath:hoftolder/ap/$Pais_$Campana_$uuid.json,
InputFile:$Pais_$Campana_$uuid.json,
EndpointURL:https://XXXXXXXXX.execute-api.us-east-1.amazonaws.com/prd/Alg_Pos,
ApiKey:xtdC3BMwLz9yQa772qJNe7JMcQ2h8hT74m7n5JrJ,
BucketDestinoSB:xxxxxxx/Matriz/$Pais

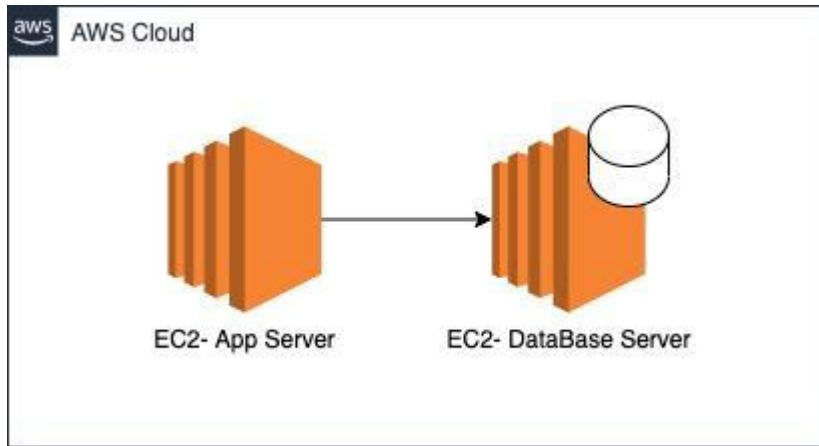
```

```
},  
APILOG:{  
  Url:http://XXXXXXXXXXXXXXXXXXXX.us-east-1.elb.amazonaws.com:8090,  
  App:API-CUVS,  
  Servicio:GeneracionCuvs,  
  TipoEjecucion:Manual,  
  Usuario:xxxxxxx  
}  
}
```

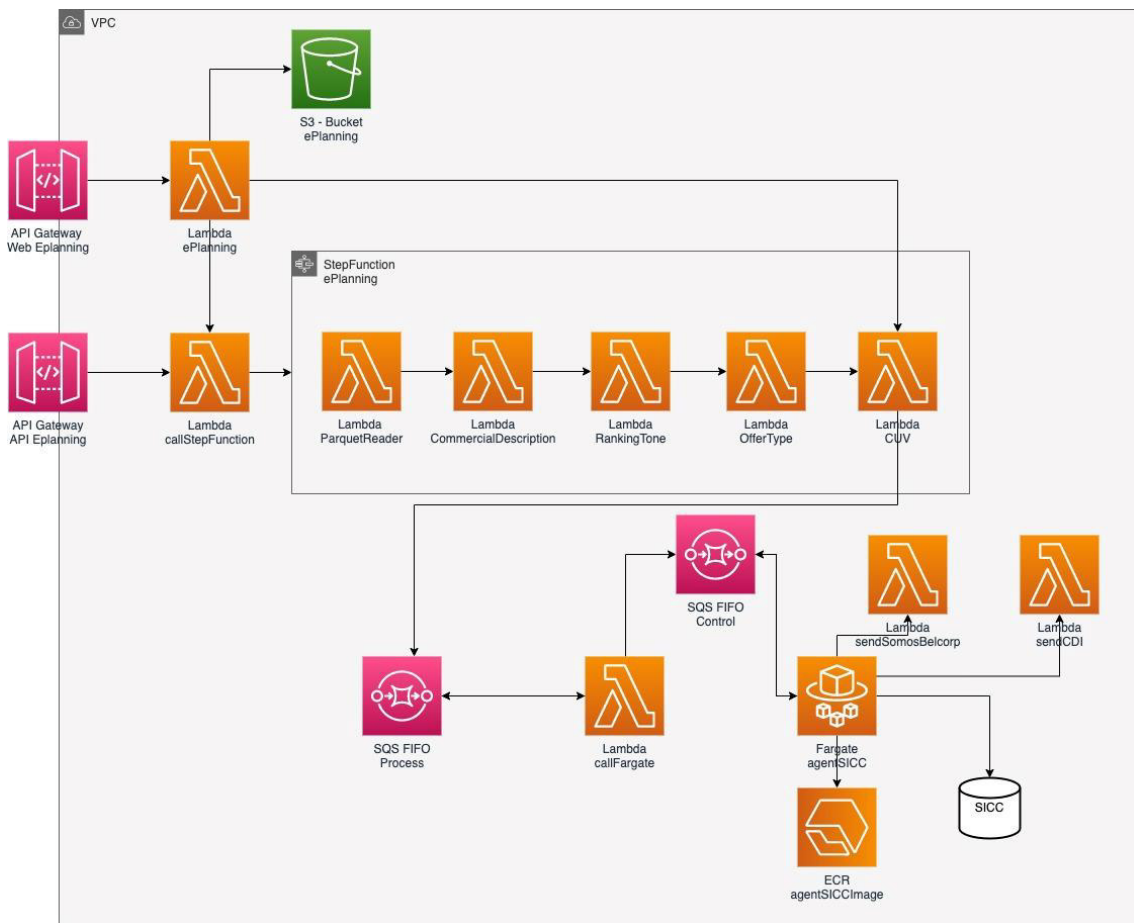
ANEXO 5. SOLUCIÓN EPLANNING EN VISUAL STUDIO



ANEXO 6: ARQUITECTURA INICIAL EPLANNING



ANEXO 7: ARQUITECTURA EPLANNING ORIENTADA A MICROSERVICIOS



ANEXO 8: CALCULO DE COSTOS DE OPERACIÓN EN AWS PRICING CALCULATOR

LAMBDA:

Service settings

info

The calculations below exclude free tier discounts.

Architecture

x86

Number of requests

10000

per month

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

60000

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

512

MB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

512

MB

Show calculations

Unit conversions

Amount of memory allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB

Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB

Pricing calculations

10,000 requests x 60,000 ms x 0.001 ms to sec conversion factor = 600,000.00 total compute (seconds)

0.50 GB x 600,000.00 seconds = 300,000.00 total compute (GB-s)

300,000.00 GB-s x 0.0000166667 USD = 5.00 USD (monthly compute charges)

10,000 requests x 0.0000002 USD = 0.00 USD (monthly request charges)

0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function

Lambda costs - Without Free Tier (monthly): 5.00 USD

API GATEWAY:

REST APIs

info

Select the units, number, and frequency for REST API requests based on expected volume. The calculations below exclude free tier discounts.

REST API request units

The unit will apply to the number of requests you will specify below.

thousands

Requests

10000

per month

Cache memory size (GB)

Optional. Choose cache memory size in GB. Dedicated cache memory is billed at an hourly rate.

None

Show calculations

10,000 requests x 1,000 unit multiplier = 10,000,000 total REST API requests

Tiered price for: 10000000 requests

10000000 requests x 0.0000035000 USD = 35.00 USD

Total tier cost = 35.0000 USD (REST API requests)

Tiered price total for REST API requests: 35 USD

0 USD per hour x 730 hours in a month = 0.00 USD for cache memory

Dedicated cache memory total price: 0.00 USD

REST API cost (monthly): 35.00 USD

SQS:

Standard queue requests

Enter amount million per month ▼

FIFO queue requests

1 million per month ▼

▼ Show calculations

1 requests per month x 1000000 multiplier for million = 1,000,000.00 total FIFO queue requests
 Tiered price for: 1000000.00 requests
 1000000 requests x 0.0000000000 USD = 0.00 USD
 Total tier cost = 0.0000 USD (FIFO queue requests cost)
Total SQS cost (monthly): 0.00 USD

FARGATE:

On Demand [Info](#)

🔔 Fargate Spot and Compute Savings Plans are currently not supported on the AWS Pricing Calculator.

With AWS Fargate, there are no upfront payments and you only pay for the resources that you use. Pricing is based on requested vCPU, memory, and storage resources consumed by your Amazon ECS Task or Amazon EKS Pod.

Operating system

Linux ▼

CPU Architecture

x86 ▼

Number of tasks or pods
 Enter the number of tasks or pods running for your application

10000 per month ▼

Average duration
 Enter the time period for which your tasks or pods are running. Pricing is per second with a 1-minute minimum (Linux) and a 15-minute minimum (Windows). Duration is calculated from the time you start to download your container image (docker pull) until the Task or Pod terminates, rounded up to the nearest second.

30 minutes ▼

Amount of vCPU allocated
 Enter the amount between 0.25 vCPU and 4 vCPU

0.5 ▼

🔔 vCPU selected supports memory values between 1 GB and 4 GB, in 1 GB increments

Amount of memory allocated

4 GB ▼

Amount of ephemeral storage allocated for Amazon ECS
 Enter the amount between 20 GB and 200 GB. The first 20 GB are at no additional charge, you only pay for any additional storage that you configure for the Task.

20 GB ▼

▼ Show calculations

Unit conversions management events
 Average duration: 30 minutes = 0.5 hours

Pricing calculations
 10,000 tasks x 0.50 vCPU x 0.50 hours x 0.04048 USD per hour = 101.20 USD for vCPU hours
 10,000 tasks x 4.00 GB x 0.50 hours x 0.004445 USD per GB per hour = 88.90 USD for GB hours
 20 GB - 20 GB (no additional charge) = 0.00 GB billable ephemeral storage per task
 101.20 USD for vCPU hours + 88.90 USD for GB hours = 190.10 USD total
Fargate cost (monthly): 190.10 USD

ECR:

Storage settings [info](#)

To estimate your monthly cost, enter the amount of data you will store in your repositories. The calculations below exclude Free Tier discounts.

Amount of data stored

MB per month

▼ Show calculations

Unit conversions

Amount of data stored: 100 MB per month x 0.0009765625 GB in a MB = 0.09765625 GB per month

Pricing calculations

0.09765625 GB per month x 0.10 USD = 0.0098 USD

Elastic Container Registry pricing (monthly): 0.0098 USD

S3:

▼ S3 Standard [info](#)

The calculations below exclude Free Tier discounts.

S3 Standard storage

TB per month

PUT, COPY, POST, LIST requests to S3 Standard

GET, SELECT, and all other requests from S3 Standard

Data returned by S3 Select

TB per month

Data scanned by S3 Select

GB per month

▼ Show calculations

Unit conversions

S3 Standard storage: 1 TB per month x 1024 GB in a TB = 1024 GB per month

Pricing calculations

Tiered price for: 1024 GB

1024 GB x 0.0230000000 USD = 23.55 USD

Total tier cost = 23.5520 USD (S3 Standard storage cost)

80,000 PUT requests for S3 Storage x 0.000005 USD per request = 0.40 USD (S3 Standard PUT requests cost)

80,000 GET requests in a month x 0.0000004 USD per request = 0.032 USD (S3 Standard GET requests cost)

23.552 USD + 0.032 USD + 0.40 USD = 23.98 USD (Total S3 Standard Storage, data requests, S3 select cost)

S3 Standard cost (monthly): 23.98 USD

Step Functions:

Standard Workflows [info](#)

The AWS Step Functions Free Tier does not automatically expire at the end of your 12 month AWS Free Tier term, and is available to both existing and new AWS customers indefinitely.

Workflow requests

AWS Step Functions Standard Workflows counts a request each time it starts executing a workflow, including tests from the console.

per month

State transitions per workflow

A state transition is counted each time a step in your execution is completed.

▼ Show calculations

5 state transitions per workflow x 10,000 workflow requests = 50,000.00 state transitions per month

50,000.00 state transitions per month - 4000 Free Tier state transitions = 46,000.00 billable state transitions

Max (46000.00 billable state transitions, 0) = 46,000.00 billable state transitions per month

46,000.00 billable state transitions x 0.000025 USD per state transition = 1.15 USD per month

Standard Workflows pricing (monthly): 1.15 USD