Arquitectura de Software Para Vaova Travel

Richard Anderson Mejia Medina

Nota de Aceptación

Certificamos que el presente Trabajo de Grado Satisface,
en alcances y calidad, todos los requisitos que demanda
un Trabajo de Grado de Maestría.

JUAN CAMILO PARRA MARTÍNEZ
Director

LUISA RINCÓN
Jurado 1

ANDRÉS COLLAZOS
Jurado 2

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia
Universidad Javeriana Cali, para optar el título de Magister en
Ingeniería de Software.

HERNAN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Santiago de Cali, 15 de enero de 2024

Acta de Correcciones al Documento de Trabajo de Grado

Santiago de Cali, 25 de enero de 2024

Autor: Richard Anderson Mejia Medina

Título del Trabajo de Grado: "Arquitectura de Software Para Vaova Travel"

Director:

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.

Firma del Director del Trabajo de Grado

Santiago de Cali, 25 de enero de 2024

Ingeniero:
Juan Carlos Martínez Arias
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ingeniería de Software, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado *Arquitectura de Software para Vaova Travel*, el cual fue desarrollado por el (la) estudiante Richard Anderson Mejia Medina con código 8973091 perteneciente al énfasis en Software, bajo la dirección del profesor Juan Camilo Parra Martínez.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Firma
Richard Anderson Mejia Medina
C.C. 1118298344 de Yumbo

Firma
Juan Camilo Parra Martínez
C.C. 1014199525 de Bogotá

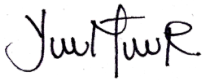
VAOVA
THE ART OF HAPPENINGS

Bogotá, D.C, 11 de Diciembre de 2022

A QUIEN INTERESE

Por medio de la presente yo, Yady Marcela Tarazona Rico identificada con cedula de ciudadanía No. 1030548802 de Bogotá y en calidad de Gerente de tecnología de la empresa VAOVA S.A.S, autorizo a Richard Anderson Mejía Medina identificado con cedula de ciudadanía No. 1118298344 de Yumbo, estudiante de la Maestría en Ingeniería de Software de la Pontificia Universidad Javeriana de Cali para hacer uso del nombre de VAOVA S.A.S en su proyecto de grado, llamado Arquitectura de Software para VAOVA Travel.

Cordialmente,



Marcela Tarazona Rico
Gerente de tecnología
3115076813

Maestría en Ingeniería de Software
Facultad de Ingeniería y Ciencias

FICHA RESUMEN
TRABAJO DE GRADO DE MAESTRÍA

TITULO: "*Arquitectura de Software Para Vaova Travel*"

1. ÉNFASIS: Ingeniería de Software
2. TIPO DE PROYECTO: Aplicado
3. ÁREA DE TRABAJO: Arquitectura de Software
4. ESTUDIANTE (S): Richard Anderson Mejia Medina
5. CORREO ELECTRÓNICO: richie8344@javerianacali.edu.co
6. DIRECCIÓN Y TELÉFONO: Crr 83E # 54-79 Apto 806B, Cali. 3217816705.
7. DIRECTOR: Juan Camilo Parra Martínez
8. VINCULACIÓN DEL DIRECTOR (en la universidad): Cátedra
9. CORREO ELECTRÓNICO DEL DIRECTOR: juancamilo.parra@javerianacali.edu.co
10. CO-DIRECTOR(ES) (Si aplica): No aplica.
11. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Vaova Travel
12. OTROS GRUPOS O EMPRESAS: No aplica.
13. PALABRAS CLAVE (al menos 5): Vaova Travel, Arquitectura de Software, Domain-Driven Design (DDD), Attribute-Driven Design (ADD), Software Architecture Method (SAAM).
14. ODS QUE APLICA EL PROYECTO (Agenda 2030): Educación de Calidad
15. FECHA DE INICIO (Desarrollo del proyecto): 2/01/2023
16. RESUMEN (máximo 400 palabras): Este proyecto de grado propone el diseño de una arquitectura de software para la compañía de turismo Vaova Travel, enfocada en satisfacer los atributos de calidad de disponibilidad e interoperabilidad con plataformas de terceros. Para el desarrollo del proyecto propone realizar un inventario de activos de software e integraciones existentes. La definición de historias épicas como insumo para el diseño de la arquitectura. El diseño de la arquitectura se realiza utilizando los modelos de Domain-Driven Design (DDD) y Attribute-Driven Design (ADD). La evaluación del diseño de la arquitectura de software se divide en dos: una evaluación técnica empleando el modelo analítico Software Architecture Analysis Method (SAAM) y otra evaluación financiera empleando el análisis de retorno de inversión.

En conjunto, este proyecto establece una base sólida para mejorar la operación de Vaova y su capacidad para proporcionar servicios de alta calidad. La arquitectura propuesta cumple con los requisitos técnicos y estratégicos, asegurando la sostenibilidad y competitividad futura de Vaova en el mercado.

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Ingeniería de Sistemas y Computación.
Proyecto de Grado.

Arquitectura de Software Para Vaova Travel

Richard Anderson Mejia Medina

Director: MSc Juan Camilo Parra Martínez

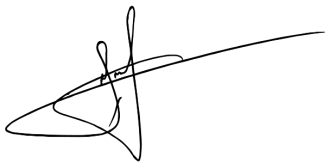
25 de enero de 2024

Santiago de Cali, 25 de enero de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Arquitectura de Software Para Vaova Travel" realizado por el estudiante de la Maestría en Ingeniería de Software, Richard Anderson Mejia Medina (cod: 8973091), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,



MSc Juan Camilo Parra Martínez

Santiago de Cali, 25 de enero de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Arquitectura de Software Para Vaova Travel" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,

Richard Anderson Mejia Medina
Código: 8973091

0.1. Resumen

Este proyecto de grado propone el diseño de una arquitectura de software para la compañía de turismo Vaova Travel, enfocada en satisfacer los atributos de calidad de disponibilidad e interoperabilidad con plataformas de terceros.

Para el desarrollo del proyecto propone realizar un inventario de activos de software e integraciones existentes. La definición de historias épicas como insumo para el diseño de la arquitectura. El diseño de la arquitectura se realiza utilizando los modelos de *Domain-Driven Design* (DDD) y *Attribute-Driven Design* (ADD). La evaluación del diseño de la arquitectura de software se divide en dos: una evaluación técnica empleando el modelo analítico *Software Architecture Analysis Method* (SAAM) y otra evaluación financiera empleando el *análisis de retorno de inversión.*

En conjunto, este proyecto establece una base sólida para mejorar la operación de Vaova y su capacidad para proporcionar servicios de alta calidad. La arquitectura propuesta cumple con los requisitos técnicos y estratégicos, asegurando la sostenibilidad y competitividad futura de Vaova en el mercado.

Palabras Clave: Vaova Travel, Arquitectura de Software, Domain-Driven Design (DDD), Attribute-Driven Design (ADD), Software Architecture Analysis Method (SAAM).

0.2. Abstract

This degree project proposes the design of a software architecture for the tourism company Vaova Travel, focused on satisfying the quality attributes of availability and interoperability with third-party platforms.

For the development of the project, it is proposed to carry out an inventory of existing software assets and integrations. The definition of epic stories as input for architectural design. The architectural design is carried out using the *Domain-Driven Design* (DDD) and *Attribute-Driven Design* (ADD) models. The evaluation of the software architecture design is divided into two: a technical evaluation using the analytical model *Software Architecture Analysis Method* (SAAM) and another financial evaluation using *analysis of return on investment.*

Together, this project establishes a solid foundation to improve Vaova's operation and its ability to provide high-quality services. The proposed architecture meets the technical and strategic requirements, ensuring Vaova's sustainability and future competitiveness in the market.

Keywords: Vaova Travel, Software Architecture, Domain-Driven Design (DDD), Attribute-Driven Design (ADD), Software Architecture Analysis Method (SAAM).

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El turismo es una de las industrias más importantes tanto a nivel nacional como internacional. Desde hace años se ha venido impulsando al país como un destino turístico atractivo, gracias a la diversidad de paísajes que se pueden encontrar en Colombia. Vaova nació con la idea de aprovechar dichos paísajes para brindar experiencias únicas a los viajeros por medio del turismo cultural. Ofrecer a los viajeros la posibilidad de interactuar con las costumbres del lugar que se visita, disfrutar la gastronomía locals, conocer la historia y visitar lugares maravillosos es la propuesta de valor de Vaova.

Para realizar un viaje la compañía debe ejecutar una serie de pasos de su proceso operativo, entre estos podemos encontrar la gestión de cobros, la planeación y la ejecución. En la actualidad Vaova no puede ejecutar por sí misma todos estos pasos, por lo que ha recurrido a acuerdos con terceros para suplir algunas de sus necesidades. Es por esto que utiliza las plataformas de *TourHero* y *WeTravel* para recolectar los datos y pagos de los viajeros que desean contratar viajes con Vaova. Durante la ejecución de los viajes, *Viaxlab* le ofrece la posibilidad de utilizar su aplicación móvil para que los viajeros cuenten con los servicios de itinerarios y recordatorios de actividades.

La empresa ha presentado dificultades durante su operación en los últimos años porque los sistemas propios e integraciones que manejan no eran confiables. Los sistemas propios presentaban errores en su desarrollo y las integraciones muchas veces dejaban de funcionar por cambios en las plataformas de los terceros o simplemente no estaban disponibles cuando se necesitaban. La combinación de plataformas propias con errores e integraciones poco confiables ha generado una operación con reprocesos, inconsistencia en los datos de los viajeros e incremento en gastos operativos de la compañía.

Con el desarrollo de este proyecto de grado se busca establecer claramente la causa de los problemas de Vaova y plantear una solución desde la arquitectura de software. La solución debe garantizar a la compañía los siguientes aspectos claves: la disponibilidad de los servicios propios y la integración con las plataformas de terceros que se requieren para el proceso operativo. Es importante aclarar que el proyecto no brindará solución a los diferentes errores en la operación que tiene hoy Vaova, pero es un inicio para la transformación que requiere para operar de una manera más efectiva.

1.1. Definición del problema de investigación

Una de las industrias más importante a nivel global es el turismo internacional. En el año 2019 esta industria aportó $9.17 billones de dólares a la economía mundial, lo cual fue equivalente al 10.4 % del PIB global, contando en ese momento con 334 millones de empleos (Jus and Misrahi, 2021). Además, para el mismo año, la Organización Mundial del Turismo reportó que el flujo de viajeros fue de 1466 millones, siendo Europa el continente con mayor número de visitantes con el 51 % y Francia el país más visitado con el 6 % del total de turistas a nivel mundial recibiendo 90.9 millones de visitantes, lo que equivale al 0.002 % (UNWTO, 2022).

En Colombia la industria del turismo había tenido un crecimiento constante desde el año 2012 hasta el año 2019, alcanzando cifras históricas en este último. De acuerdo con el reporte del Departamento Administrativo Nacional de Estadística (DANE, 2020) las cifras económicas de la recepción de turistas se resumen en 4.5 millones de viajeros, con una inversión de $20.7 billones de pesos, que representaron 2.15 % del PIB de Colombia en el año 2019.

El país se ha esforzado en ser un destino atractivo para los visitantes de todo el mundo. Por lo tanto se ha enfocado en crear políticas que impulsen el turismo cultural, este se basa en la motivación de las personas extranjeras en aprender, descubrir y vivir experiencias culturales del lugar que visitan (Procolombia, 2021). En el año 2007 se creó la política *Identidad y desarrollo competitivo del patrimonio* y en el año 2019 se actualizó el plan *Turismo: un propósito que nos une* (Ministerio De Comercio, 2021).

Gracias a las políticas nacionales se han creado modelos de negocio que combinan turismo, cultura y tecnología para ofrecer experiencias únicas a los visitantes. Un ejemplo es el caso de Vaova, una compañía de turismo colombianan fundada en el año 2013 cuya propuesta se basa en ofrecer experiencias de turismo alternativo y sostenible por medio de los *happenings*: movimiento artístico de los años 60 que trasladó el arte de las galerías, los teatros y de su zona de confort a la calle (Vaova, 2022). Desde sus inicios la compañía ha logrado traer más de 3000 estudiantes de las escuelas y universidades más prestigiosas de los Estados Unidos y Europa, y cerraron el año 2022 con más de 4000 visitantes. Aunque la compañía ha tenido un crecimiento notable, actualmente cuenta con dificultades en todas las etapas de su proceso operativo: demostración, venta, reservación, planeación y ejecuciójn de los viajes.

En sus inicios la compañía no contaba con un equipo de tecnología que desarrollara los sistemas que se necesitaban para la operación de los viajes, por ende se realizaron acuerdos comerciales con terceros para utilizar sus plataformas. Con el paso de los años se desarrollaron algunos sistemas propios, pero estos no podían soportar todo el proceso operativo de un viaje. Actualmente la compañía depende de varias plataformas y aplicaciones propias y externas para su operación y funcionamiento. A continuación se describen las más importantes:

- No cuenta con un sistema que le permita realizar demostraciones y cotizaciones, por lo que

estas se realizan en archivos de *PowerPoint* de más de 60 diapositivas. En promedio se realizan dos nuevas cotizaciones diarias y se ajustan cinco ya existentes, por lo que se tardan hasta 4 horas en dar respuesta a los clientes.

- Tiene acuerdos comerciales con *TourHero* y *WeTravel*. Estas plataformas se encargan de recibir y procesar los pagos, recolectar los datos personales de los viajeros, así como la información del paquete, destinos, actividaes y restricciones de cada uno. Sin embargo, estas plataformas no envían la información de las ventas a Vaova, por lo que la compañía desarrolló unos procedimientos automáticos que cada 15 minutos descargan la información en archivos de *Excel* que luego se cargan a una base de datos MongoDB.

- En la etapa de planeación del viaje, la información de cada viajero se carga a una aplicación llamada *MyTrip*. Esta aplicación web es propia de la compañía, y cada viajero debe registrar para verificar sus datos personales, así como el paquete, destinos y actividades que adquirió. También permite elegir los vuelos, los hoteles y la persona on la que desea compartir habitación. Este sistema se desarrolló sin buenas prácticas de ingeniería de software, por lo que su funcionamiento es erróneo y no soporta más de 30 usuarios de manera concurrente.

- Finalmente, durante la ejecución de los viajes se ofrece la posibilidad a los viajeros de descargar la aplicación móvil de *Viaxlab*. Esta les ofrece servicios de itinerarios, notificaciones de actividades y ficha médica; asimismo permite a Vaova monitorear y controlar las personas que asisten a cada una de las actividades. Esta aplicación ha generado problemas por su baja disponibilidad y mala calidad. Uno de los errores más frecuentes es que se debe generar un código QA por viajero para ingresar a las actividades, pero este código en la mayoría de ocasiones no se genera, y si se genera este no se puede validar; esto hace que la compañía no pueda verificar si una persona contrató o no una actividad. En otras ocasiones la aplicación simplemente deja de responder y no funciona ningún servicio.

Depender de las plataformas mencionadas, que tienen no se integran de manera automática y tienen problemas de disponibilidad, ha generado las siguientes consecuencias negativas para la empresa:

- **Inconsistencia en la información**. Los viajeros se deben registrar en diferentes plataformas para viajar con la compañía. Adquieren el viaje por la plataforma de *TourHero* o *WeTravel*, se registran en *MyTrip* para verificar sus datos y utilizan *Viaxlab* para visualizar su itinerario durante el viaje. Esto genera inconsistencia en la información, ya que los viajeros no siempre utilizan los mismos datos en los registros de las aplicaciones. Por ejemplo, se pueden registrar con direcciones de correo electrónico diferente o pueden cometer un error en la digitación de los datos.

- **Operación casi manual con reprocesos**. Debido a las fallas en los sistemas de información, la compañía debe realizar muchos procesos operativos de manera manual o realizar controles y verificaciones con hojas de papel. Un ejemplo es que en ocasiones las integraciones con

TourHero o *WeTravel* dejan de funcionar y la información de los viajeros se debe cargar manualmente a la base de datos. Otro ejemplo, durante la ejecución de las actividades *Viaxlab* no está disponible y un funcionario debe imprimir en papel los nombres de las personas que están autorizadas para ingresar a una actividad específica.

- **Incremento de costos operativos**. Es el caso con *Viaxlab* que cobra $9 dólares por cada viajero que Vaova registra en la aplicación. Asumir este costo no se justifica porque la aplicación no funciona como se espera.

El plan estratégico de Vaova es convertirse en la primera compañía colombiana del sector turismo que tiene una base tecnológica para su operación. Para hacer realidad el plan, la compañía ha explorado el mercado nacional e internacional en búsqueda de una solución que le permita operar de manera efectiva, pero no ha encontrado ninguna que se ajuste a sus necesidades. La razón por la cual las soluciones del mercado no se ajustan a las necesidades de la compañía es que Vaova le brinda a sus clientes experiencias totalmente personalizadas; por ende el cliente es quien decide las fechas, destinos, actividades, vuelos, hoteles y demás detalles entre las opciones que Vaova le ofrece. Por lo tanto la compañía ha decidido innovar en el mercado, desarrollando sistemas de software a la medida que le garanticen operar de forma confiable, integrarse automáticamente con proveedores y reducir costos operativos de terceros que no funcionan.

La propuesta de este proyecto de grado es diseñar una arquitectura de software que le permita a Vaova desarrollar los sistemas de información que require para funcionar de manera efectiva en todas las etapas de su operación como son cotizaciones, reservas, pagos, planeación y ejecución de viajes. El diseño de la arquitectura debe contemplar la alta disponibilidad de las operaciones y facilitar la interoperabilidad con plataformas de terceros.

1.2. Planteamiento del problema

¿Cómo generar una solución integral desde la ingeniería de software que evalúe los antecedentes, la situación actual y la visión de Vaova y permita a la compañía superar las dificultades que tiene actualmente en su proceso operativo, para desarrollar satisfactoriamente su plan estratégico de negocio?

1.3. Sistematización

1. ¿Qué activos de software con los que actualmente cuenta Vaova son los que le generan valor en su proceso operativo?
2. ¿Que funcionalidades son las más importantes en la operación de Vaova para poder llevar a cabo su plan estratégico?
3. ¿Como definir la refactorización tecnológica que necesita Vaova para crear un ecosistema tecnológico que le permita optimizar su operación?

4. ¿Que estrategías utilizar para garantizar los niveles de calidad que Vaova requiere en su ecosistema tecnológico?

1.4. Objetivos

1.4.1. Objetivo General

Proponer una arquitectura de software de un sistema de información que soporte la operación de Vaova, teniendo en cuenta las necesidades de alta disponibilidad en los servicios propios de Vaova y la interoperabilidad con plataformas de terceros.

1.4.2. Objetivos Específicos

1. Realizar un inventario de activos de software e integraciones con los que actualmente cuenta Vaova para determinar cuáles se pueden reutilizar, refactorizar o retirar de la operación.

2. Definir las historias épicas que se requieren para diseñar la arquitectura de software.

3. Diseñar la arquitectura de software, contemplando los escenarios de calidad para disponibilidad e interoperabilidad empleando los modelos de diseño de *Domain-Driven Design* (DDD) y *Attribute-Driven Design* (ADD).

4. Evaluar la arquitectura de software utilizando el modelo analítico de *Software Architecture Analysis Method* (SAAM).

1.5. Justificación

En la sección 1.1 se describieron dos temas de vital importancia: la industria del turismo en los ámbitos nacional e internacional y los diferentes problemas que tiene Vaova para operar de manera eficiente. La compañía quiere aprovechar los beneficios que el gobierno ofrece al sector turístico, pero los problemas en su operación no le permiten explotar todas sus capacidades.

El desarrollo de este proyecto de grado será relevante para Vaova porque le permitirá contar con un análisis de su situación actual y se propondrá el diseño de arquitectura de software que podrá soportar su operación y le garantizará los atributos de calidad y disponibilidad para los servicios propios y la interoperabilidad con plataformas de terceros.

Los dos primeros objetivos específicos de este proyecto están enfocados en establecer el porqué de las dificultades de Vaova. Al analizar los sistemas de información se busca establecer los puntos en los cuáles estos presentan los mayores problemas en los procesos de la compañía. Con el análisis de historias épicas se determinará si los sistemas actuales están en condiciones de seguir operando, deben ser rediseñados o se deben retirar definitivamente.

Los dos últimos objetivos específicos se enfocan en el diseño de una nueva arquitectura de software para Vaova. Para el diseño de la arquitectura de software se emplearán procesos formales y rigurosos, con un marco de diseño que utiliza las herramientas que la ingeniería de software ofrece como lo son los patrones, las tácticas y los atributos de calidad. Por su parte, la evaluación de la arquitectura de software se realizará empleando métodos formales que permitan garantizar que los dos atributos de calidad deseados se pueden satisfacer de la mejor manera.

Al finalizar este proyecto de grado, Vaova contará con el diseño de una arquitectura de software que soporta su operación y emplea las mejores prácticas de ingeniería para garantizar los atributos de disponibilidad e interoperabilidad. Además, el documento de arquitectura generado puede servir como guía para futuros diseños y desarrollos en la compañía con relación a documentación, diagramas, diseños, decisiones, metodologías y tecnologías.

1.6. Delimitaciones y Alcances

Este trabajo de grado tiene como alcance el diseño y evaluación de una arquitectura de software que soporte la operación de Vaova. A continuación se describen los aspectos que se incluyen en el alcance y cuya ejecución es necesaria para alcanzar los objetivos:

- Análisis de los procesos operativos de Vaova, para entender cómo funciona el negocio y determinar posibles mejoras.
- Revisión de los sistemas propios que posee Vaova actualmente para determinar cuales se deben conservar, rediseñar o descartar.

- Revisión de las integraciones con las que actualmente opera la compañía. Sólo se deben conservar las que Vaova considere por cuestiones contractuales; las funciones de las integraciones que se descarten serán reemplazadas por servicios propios en el diseño de la arquitectura.
- El diseño de la arquitectura se realizará teniendo en cuenta las restricciones que tiene Vaova con relación a objetivos de negocio, requerimientos funcionales, requerimientos no funcionales, stack tecnológico y presupuesto.
- La evaluación de la arquitectura de software se realizará utilizando un método formal que permita garantizar que los atributos de calidad de disponibilidad e interoperabilidad se satisfacen con el diseño propuesto.

Por fuera del alcance de este proyecto de grado están los siguientes aspectos:

- Implementación de la arquitectura de software en ambientes de desarrollo, aceptación y producción al interior de Vaova.
- No se tendrán en cuenta cambios o nuevos requerimientos que puedan surgir una vez se realice la definición de las historias épicas, diseño de escenarios de calidad o evaluación de la arquitectura de software.

Este proyecto sentará las bases para que la compañía pueda realizar los cambios tecnológicos necesarios a través de la implementación de los sistemas propuestos en la arquitectura de software, aunque dicha implementación está fuera del alcance de este proyecto de grado. Además, el documento de arquitectura puede ser utilizado para que Vaova formalice y establezca las bases del proceso de diseño arquitectónico dentro del área de tecnología.

1.7. Resultados esperados

Con la realización de este proyecto de grado se obtendrán los siguientes resultados.

- Inventario de los sistemas con los que actualmente Vaova realiza su operación, sean propios o de terceros.
- Requerimientos funcionales utilizando el modelo de historias épicas.
- Documento de arquitectura de software para Vaova con análisis, decisiones, diagramas, diseños, escenarios de calidad, patrones y tácticas de arquitectura empleadas.
- Informe detallado con los resultados obtenidos durante la evaluación de la arquitectura de software.
- Caso de negocio basado en la viabilidad de implementar la arquitectura de software en Vaova.

Capítulo 2

Marco de referencia

2.1. Bases teóricas

2.1.1. Antecedentes y Estado de Vaova

Para entender el estado actual de Vaova es necesario conocer los sucesos más importantes que han ocurrido desde su fundación en el año 2013. A continuación se describen los años y los hechos que han marcado a la compañía y la Figura 2.1 muestra un gráfico con la evolución desde su fundación:

- **2013**: La historia de Vaova inicia cuando su fundador y *CEO* decide iniciar su emprendimiento basado en una de sus grandes pasiones como son los viajes.
- **2014**: Inicia la consolidación de Vaova como empresa y logra armar su primer equipo de trabajo, e inicia operaciones gracias a los conocimientos de su *CEO* y los destinos tan atractivos que tiene Colombia.
- **2016**: Logra su mayor viaje hasta ese momento, el cual consistió en traer al país a 300 estudiantes de las escuelas de negocios de las universidades de *Harvard* y *Chicago*.
- **2017**: La empresa tuvo un crecimiento sostenido que le permitió consolidarse como un competidor fuerte en el área de viajes culturales.
- **2020**: Ocurren dos sucesos muy importantes para la empresa. La pandemia generada por el *COVID-19* hace que la empresa deba detener toda su operación debido a las cuarentenas impuestas por los gobiernos en la mayoría de los países. Lo anterior no impidió que la empresa ganara el *Premio Nacional de Turismo* de Procolombia en la categoría de *Liderazgo en Estrategías Innovadoras*. Vaova pudo obtener este premio gracias al proyecto *Eventos de Promoción*, que consistió en promocionar a Colombia como destino turístico por medio de eventos digítales.
- **2022**: Desde el año 2020 hasta el año 2022, la compañía se ha enfocado en solventar los retos que debe superar para convertirse en una referente del turismo cultural a nivel nacional e internacional.

A pesar del crecimiento que tuvo desde sus inicios, la operación de Vaova nunca fue sencilla. Internamente se presentaban dificultades que generaban problemas en la operación. El equipo de

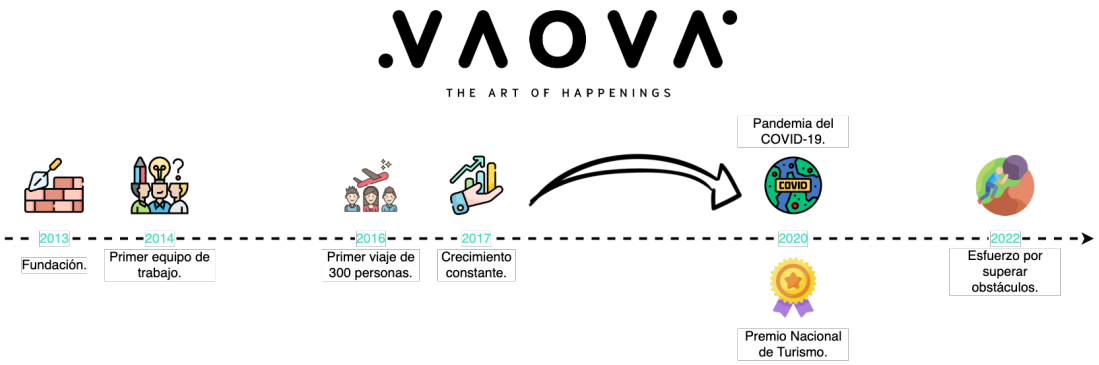


Figura 2.1: Evolución de Vaova

tecnología, que es crucial para el buen funcionamiento de la compañía, no funcionaba como se esperaba y eso generaba conflictos con otros equipos.

El equipo de tecnología a inicios del año 2022 estaba conformado de la siguiente manera:

- Un grupo de 3 desarrolladores de *Backend*: Un *Master Developer*, un *Senior Developer* y un *Junior Developer*.
- Un grupo de 2 desarrolladores de *Frontend*: Un *Senior Developer* y un *Semi Senior Developer*.
- Un *QA Tester*.
- Un *UX Design Junior*.

El equipo de tecnología estaba liderado directamente por el *CEO* y el rol de Líder Técnico era ejercido por el *Backend Master Developer*. El organigrama del equipo de tecnología de Vaova o inicios del año 2022 se puede ver en la Figura 2.2.

La falta de un rol definido que liderara al equipo afectó de manera notable al equipo de tecnología, ya que el *CEO* no es experto en el área. Entre los muchos problemas que existían a inicios del año 2022 se pueden encontrar:

- La persona que ocupaba el rol de *Backend Master Developer* no tenía ni las habilidades de liderazgo ni técnicas que el área de tecnología necesitaba.
- Falta de definición de la metodología del proceso de desarrollo. Los proyectos se desarrollaban según podían los desarrolladores, sin ninguna supervisión o control por parte de los líderes.
- No se definió un stack tecnológico para la empresa. Los proyectos se iban desarrollando según el gusto o experiencia del desarrollador asignado.

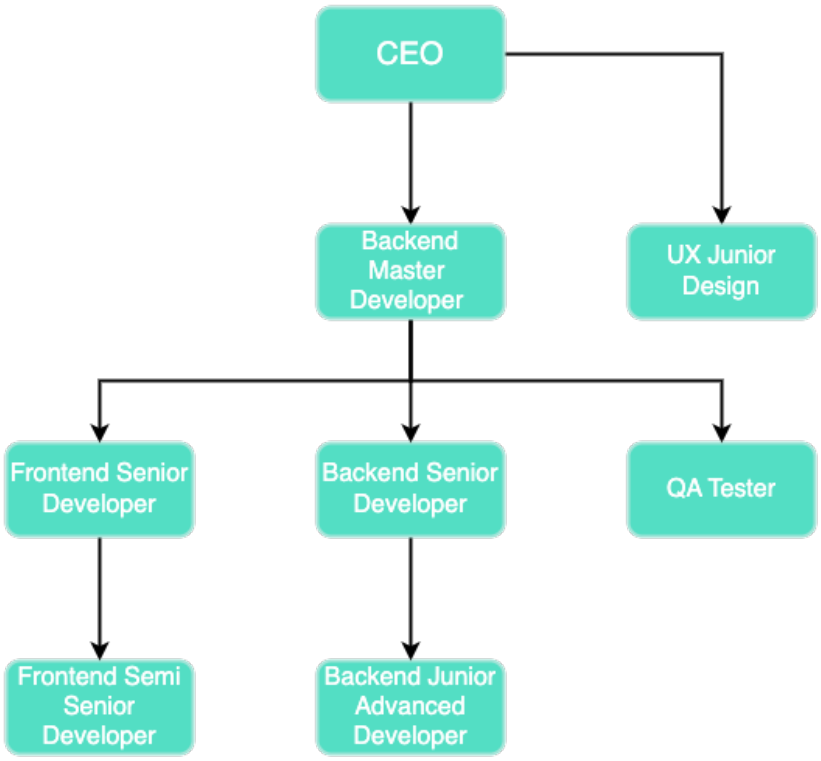


Figura 2.2: Equipo de tecnología de Vaova a inicios del 2022.

- Malas prácticas de desarrollo. No existía documentación de proeyctos, el código no se documentaba, no se hacían pruebas unitarias.
- Mal versionamiento de los proyectos. El código en los ambientes de desarrollo, aceptación y producción era diferente.
- No existía automatización. El desarrollo y despliegue del código se hacía de manera manual.
- A nivel interno, se tenían diferencias entre algunos miembros del equipo. Estas diferencias generaban que la comunicación con otras áreas de la empresa fuera difícil.

A mediados del año 2022 el *CEO* decide contratar a una persona que se encargue de liderar al equpo de tecnología y pueda dar solución a mediado y largo plazo a los problemas tecnológicos de la empresa. El rol de esta persona fue el de *Gerente de Producto (GP)* y tiene funciones de *Chief Product Officer (CPO)* y *Chief Technology Officer (CTO)*. El *GP*, al hacer una evaluación inicial del equipo y los activos, determina lo siguiente:

- El equipo que existía no tenía la experiencia ni los conocimientos técnicos suficientes para solventar los problemas de la empresa.

- La organización del equipo debía cambiar para tener más control sobre el proceso de desarrollo.
- Era indispensable definir la metodología del proceso de desarrollo, adoptar un stack tecnológico e iniciar la automatización de los procesos del área.

Después de los cambios que se realizaron en el organigrama del equipo de tecnología, el organigrama quedó de la siguiente manera:

- Un *GP* que se encarga de dirigir el producto y al área de tecnología.
- Un *Technical Leader* que se encarga de las definiciones técnicas y tiene tareas de desarrollador.
- Un *Backend Junior Advanced Developer*.
- Dos *Frontend Junior Advanced Developer*.
- Un *QA Tester*.
- Un *UX/UI Designer*.

El área de tecnología empezó a tener resultados positivos y ha solucionado varios de los problemas que tenía. El nuevo organigrama del área de tecnología de Vaova a finales del año 2022 se puede ver en la Figura 2.3.

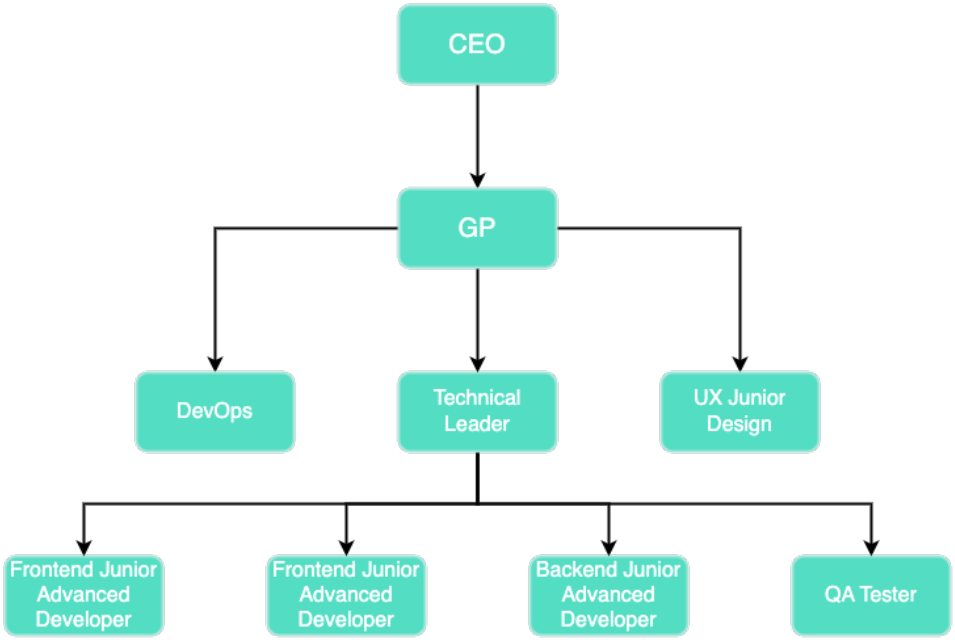


Figura 2.3: Equipo de tecnología de Vaova a finales del 2022.

Una vez realizados los cambios estructurales en el equipo era necesario definir los cambios que se debían realizar sobre los sistemas de software. Los nuevos sistemas se deben basar en una arquitectura de software que siga las buenas prácticas de la industria y que permita alcanzar los atributos de calidad de disponibilidad e interoperabilidad.

2.1.2. Arquitectura de Software

A lo largo del tiempo, muchos autores e instituciones han dado su propia definición de lo que es la arquitectura de software. En este proyecto se utilizará la definición de Bass et al. (2012): *la arquitectura de software es el conjunto de estructuras que se necesitan para razonar sobre un sistema. Estas estructuras comprenden elementos de software, las relaciones entre estos y propiedades tanto de los elementos de software como de las relaciones.*

Los autores explican que otras definiciones de arquitectura de software hablan sobre las decisiones "tempranas", "mayores", o "importantes". Identificar si una decisión tomada sobre una arquitectura de software es "mayor"no es fácil, sobretodo si se está en un modelo de desarrollo ágil. Por lo que se recomienda iniciar por las estructuras, ya que estas son más fáciles de identificar desde el inicio, por lo que los autores determinan que *la arquitectura de software se trata de estructuras que permiten el razonamiento sobre los sistemas.*

Para complementar la definición de arquitectura de software los autores hacen dos aclaraciones importantes:

- **La arquitectura es un conjunto de estructuras**. Las estructuras son un conjunto de elementos unidos por relaciones, por lo cual los sistemas de software pueden estar compuestos por muchas estructuras.
- **La arquitectura es una abstracción**. La arquitectura de software sólo debe exponer comportamientos y propiedades que son significativos para el entendimientos de la misma y omitir todos aquellos que no lo son y sólo agregan complejidad para entenderla.

Teniendo en cuenta la definición anterior, la arquitectura de software que se diseñe deberá contemplar a un alto nivel los componentes de software y las relaciones necesarias para soportar la operación de Vaova. Esta solución no sólo debe soportar las operaciones propias de la compañía, también deberá operar con las plataformas de terceros que le ayudan en una parte de su proceso operativo. Por lo que es necesario definir el concepto de interoperabilidad.

2.1.3. Interoperabilidad

En el caso de estudio de Henttonen et al. (2007) se hace una separación de los siguientes conceptos:

- **Integrabilidad**. Es la capacidad para desarrollar componentes de software de manera independiente que pueden trabajar juntos una vez finalizados.

- **Interoperabilidad**. Es la capacidad del software para utilizar la información que es intercambiada y proporcionar algo nuevo a partir de dicha información.
- **Interconectividad**. Es la capacidad de los componentes de software para comunicarse e intercambiar información.

Kazman et al. (2020) cuestionan esta definición en su artículo, porque integrar sistemas de software es más que desarrollarlos por separado y lograr que cooperen una vez terminados. Afirman que los arquitectos de software que realizan integraciones entre sistemas deben estar conscientes de los costos y los riesgos técnicos de tareas de integración previstas e imprevistas. Los riesgos pueden estar asociados a tiempos, rendimiento, entre otros. El problema de integrabilidad se puede representar de la siguiente manera:

- Se tiene una unidad de software C o un conjunto de unidades de software $C_1, C_2...C_n$ que deben ser integradas al sistema S.
- El sistema S ya está desarrollado. Este sistema puede ser una plataforma en la que se necesita integrar C_i o puede ser una sistema existente que ya se encuentra integrado con $C_1, C_2...C_n$.
- La tarea del arquitecto de software es analizar los costos y riesgos técnicos de realizar la integración de $C_{n+1}...C_m$.

En la Figura 2.4 se aprecia el problema de integración. Se hace una abstracción de un sistema de software S que ya está terminado y unidades de software C_1, C_2, C_3 que se deben adaptar a las necesidades del sistema S para poder integrarse a este.

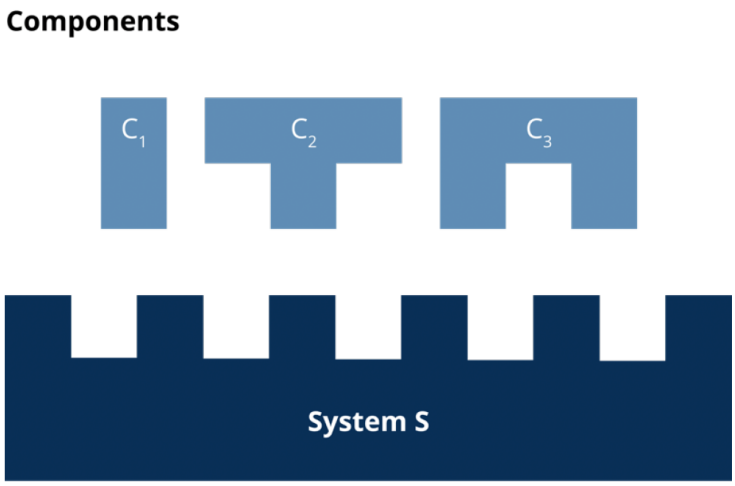


Figura 2.4: Representación abstracta del problema de integración (Kazman et al., 2020)

Kazman et al. (2020) también separan los conceptos claves para explicar mejor el problema de integración:

- **Integrabilidad**. Se refiere al grado en que un arquitecto de software ha anticipado y diseñado las tareas de integración que el sistema puede experimentar. Es esta etapa se trata de predecir los costos y riesgos de integrar alguna unidad de software en el futuro.
- **Integración**. Es una tarea específica en la que dos o más entidades de software son hechas para trabajar en conjunto. El grado en que una tarea se pueda lograr en niveles aceptables de costos y riesgos depende del grado en el que los mecanismos de integrabilidad de una arquitectura de software fueron diseñados.

Así bien, una arquitectura de software que soporte la operación de Vaova debe tener en cuenta los costos y riesgos en las integraciones que se deben realizar con *TourHero* y *WeTravel*. Estos dos proveedores son necesarios en la operación de la compañía y la arquitectura que se diseñe debe ser capaz de integrarse con las plataformas de estos proveedores.

2.1.4. Atributos de Calidad

En el reporte de Barbacci et al. (2001) se realiza una introducción a la manera en la que la arquitectura de software se relaciona con los atributos de calidad deseados para un sistema. Estos atributos de calidad pueden ser rendimiento, disponibilidad, modificabilidad, interoperabilidad y seguridad. La arquitectura de software ofrece una perspectiva global de la manera en la que estos atributos interactúan, formando una base para hacer concesiones entre estos atributos.

Uno de los objetivos de este proyecto es diseñar una arquitectura de software que soporte las operaciones de Vaova y los atributos de calidad de **disponibilidad** e **interoperabilidad** son los más relevantes. A continuación se define de manera general el marco de los atributos de calidad que son utilizados al momento de diseñar una arquitectura de software.

Bass et al. (2012) definen un atributo de calidad como *"una propiedad medible o comprobable que se utiliza para indicar que tan bien el sistema satisface las necesidades de los interesados"*. La definición es clara en señalar que los atributos de calidad de un sistema se pueden medir y ayudan a satisfacer los requerimientos funcionales de un sistema de software. Los autores ofrecen tres definiciones para complementar su punto de vista:

- **Requerimientos Funcionales**. Estos requerimientos determinan lo que el sistema debe hacer y la manera en la que se debe comportar o reaccionar ante un estímulo.
- **Requerimientos de los atributos de calidad**. Estos requerimientos son calificaciones de los requerimientos funcionales o de todo el sistema en sí. Una calificación de un requerimiento funcional es un elemento del tipo *qué tan rápido* se debe realizar una función o *qué tan resistente* debe ser a una entrada errónea.
- **Restricciones**. Una restricción es una decisión de diseño sin ningún grado de discusión. Por lo tanto son decisiones que ya fueron tomadas con anterioridad. Estas pueden involucrar el uso de un determinado lenguaje de programación o la reutilización de un componente específico.

Para complementar la definición de Bass et al. (2012) se puede recurrir al artículo de Esaki et al. (2013) en el cual explican la norma ISO/IEC25010:2011, que define el modelo de los atributos de calidad de un sistema de software. El modelo de calidad se encuentra compuesto por ocho características de principales y cada una de estas se compone a su vez de varias subcaracterísticas. En la Figura 2.5 se muestra el modelo completo.

Figura 2.5: Norma ISO/IEC25010:2011 (ISO, 2022)

En el diseño de la arquitectura de software para Vaova se tendrán en cuenta dos características: **Fiabilidad**, con la subcaracterística de *Disponibilidad*; y **Compatibilidad**, con la subcaracterística de *Interoperabilidad*:

- **Fiabilidad**. Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y período de tiempo determinados.

  - *Disponibilidad*. Capacidad del sistema o componente de estar operativo y accessible para su uso cuando se requiere.

- **Compatibilidad**. Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno de hardware o software.

  - *Interoperabilidad*. Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

El atributo de disponibilidad se empleará para que los servicios de Vaova estén operativos cuando se requieren, por ejemplo antes y durante la ejecución de los viajes. Por su parte, el atributo de interoperabilidad se utilizará para que los servicios de Vaova puedan interactuar con las plataformas de *TourHero* y *WeTravel*, que son los terceros que la empresa utiliza para recibir los pagos y la información de los viajeros.

2.1.5. Patrones y Tácticas de Arquitectura

Una vez que se han definido los atributos de calidad que se quieren alcanzar con la arquitectura que se va a proponer, se deben definir los patrones y tácticas de arquitectura que permitan satisfacer dichos atributos de calidad.

Bass et al. (2012) afirman que existen muchas maneras de realizar un mal diseño y unas pocas para hacerlo bien. Los patrones y las tácticas son formas de capturar estructuras de buen diseño que ya han sido probados y pueden reutilizarse.

En el libro de Gamma et al. (1994) se define por primera vez y de manera formal lo que es un patrón de diseño de software: *cada patrón describe un problema que ocurre una y otra vez en un entorno y describe la solución del mismo, de tal manera que la solución se puede utilizar un millón de veces, sin tener que hacerlo de la misma manera.* La descripción del patrón de diseño debe incluir:

- **Nombre**. Debe describir el problema de diseño, la solución y las consecuencias en una o dos palabras.
- **Problema**. Describe cuándo aplicar el patrón. Este explica el problema y su contexto.
- **Solución**. Describe los elementos que componen la solución, relaciones, responsabilidades y colaboraciones.
- **Consecuencias**. Son los resultados y las compensaciones de aplicar el patrón de diseño.

Bass et al. (2012) utilizan un esquema similar para describir como se documentan los patrones de arquitectura, pero lo complementan con otros elementos dentro de la solución:

- **Un contexto**. Situación común y recurrente que da lugar a un problema.
- **Un problema**. El problema, que apropiadamente generalizado, se presenta en el contexto dado. El patrón describe el problema y sus variantes, y describe las fuerzas complementarias u opuestas. Las descripción del problema a menudo incluye los atributos de calidad que el patrón ayuda a cumplir.
- **Una solución**. Una solución arquitectural exitosa para el problema, abstraída apropiadamente. La solución describe las estructuras arquitectónicas para solucionar el problema, incluyendo cómo hacer un balance entre las diferentes fuerzas. La solución puede describir las responsabilidades y las relaciones estáticas entre los elementos, o el comportamiento en tiempo de ejecución y las interacciones entre los elementos. La solución para un patrón arquitectónico es determinada y descrita por:
  - Un conjunto de tipos de elementos.

- Un conjunto de mecanismos de interacción o conectores.
- Una organización topológica de los elementos.
- Un conjunto de restricciones semánticas que cubren la topología, el comportamiento de los elementos y los mecanismos de interacción.

A continuación se describen brevemente los patrones definidos en el libro de Bass et al. (2012), aunque estos no son todos los patrones de arquitectura que existen:

- **Layered Pattern**. Define capas (agrupamiento de módulos que ofrecen un conjunto cohesivo de servicios) y una relación unidireccional de *puede-usar* a traves de las capas.
- **Broker Pattern**. Define un componente que se encarga de intermediar en la comunicación de varios clientes y servidores.
- **Model-View-Controller Pattern - MVC**. Descompone las funcionalidades de un sistema en tres componentes: un modelo, una vista y un controlador que comunica al modelo con la vista.
- **Pipe-and-Filter Pattern**. Los datos viajan por canales de comunicación denominados tuberías y son procesados en componentes llamados filtros. Los datos de salida de un filtro se convierten en los datos de entrada de otro filtro.
- **Client-Server Patter**. El cliente inicia la interacción con los servidores, invocando los servicios a medida que los necesita y esperando la respuesta de cada servicio.
- **Peer-to-Peer Pattern**. La computación se logra mediante la cooperación de pares que solicitan y brindan servicios entre sí a través de una red.
- **Service-Oriented Architecture Pattern - SOA**. La computación se logra por medio de un conjunto de componentes cooperantes que proporcionan y/o consumen servicios a través de una red. La computación a menudo es descrita utilizando un lenguaje de flujo de trabajo.
- **Publish-Subscribe Pattern**. Los componentes publican y se subscriben a eventos. Cuando un evento es anunciado por un componente, el conector de infraestructura despacha el evento a todos los subscriptores.
- **Shared-Data Pattern**. La comunicación entre quienes acceden a los datos está mediada por un almacén de datos compartidos. El control puede ser iniciado por quienes acceden a los datos o por el almacén de datos. El almacén de datos hace que los datos sean persistentes.
- **Map-Reduce Pattern**. El patrón mapa reducido proporciona un marco de trabajo para el análisis de un conjunto distribuido de datos muy grande que se ejecutará en paralelo, en un conjunto de procesadores. Esta paralelización permite una baja latencia y una alta disponibilidad. El mapa realiza los procesos de extracción y transformación de los datos, mientras el proceso de reducir realiza la carga de resultados.

- **Multi-Tier Pattern**. Las estructuras de ejecución de muchos sistemas están organizadas como un conjunto lógico de componentes agrupados. Cada agrupamiento es llamado un nivel. El agrupamiento de componentes en niveles se puede basar en varios criterios como el tipo de componente, compartir el mismo ambiente de ejecución, o tener el mismo objetivo en tiempo de ejecución.

Siguiendo con el libro de Bass et al. (2012), los autores definen las tácticas como *"bloques de construcción"* del diseño. Si los patrones de arquitectura fueran moléculas, las tácticas serían átomos. La mayoría de patrones de arquitectura están conformados por diferentes tácticas, entonces los patrones de arquitectura empaquetan varias tácticas.

Las tácticas tienen un nivel de granularidad más fino que los patrones, por lo general una táctica ayuda a alcanzar un atributo de calidad determinado o pueden ayudar a mitigar efectos negativos que un patrón de arquitectura puede tener sobre un atributo de calidad determinado (Bass et al., 2012).

Para el atributo de calidad de **disponibilidad**, Bass et al. (2012) definen las siguientes tácticas:

- **Detectar fallos**

  - **Ping/Eco**. Se refiere a un par de mensajes de solicitud y respuesta que se realizan de manera asíncrona que se utilizan para determinar la accesibilidad y el tipo de ida y vuelta de una petición.
  - **Monitor**. Es un componente que se utiliza para monitorear el estado de salud de otros componentes con relación al consumo de procesadores, procesos, entradas/salidas, memoria, entre otros.
  - **Heartbeat**. Es un mecanismo de detección de fallos que utiliza el intercambio de un mensaje periódico entre un sistema monitor y un sistema monitoreado.
  - **Timestamp**. Se utiliza para detectar una secuencia incorrecta de eventos, principalmente en sistemas distribuidos.
  - **Sanity Checking**. Verifica la validez o razonabilidad de operaciones específicas o salidas de un componente.
  - **Condition Monitoring**. Implica verificar las condiciones en un proceso o dispositivo, o validar las suposiciones realizadas durante el diseño. Al monitorear las condiciones, esta táctica evita que un sistema produzca un comportamiento defectuoso.
  - **Voting**. Se emplean componentes que hacen lo mismo, cada uno de los cuales recibe entradas idénticas y reenvía su resultado al componente de votación que se encarga de detectar cualquier inconsistencia entre los resultados por medio de la lógica de votación. El componente de votación debe decidir qué resultado utilizar.

    En la táctica de votación podemos encontrar las siguientes implementaciones:

- **Replication**. Es la forma más simple de votación ya que los componentes son clones exactos. Tener múltiples copias idénticas de un componente puede ser efectivo en la protección contra fallas de hardware, pero esta táctica no puede proteger contra errores en diseño o implementación en software o hardware, debido a que no hay diversidad embebida en esta táctica.
- **Functional Redundancy**. Es una forma de votación destinada a abordar el problema de las fallas de modo común (fallas de diseño o implementación) en componentes de hardware o software. Aquí, los componentes siempre deben dar la misma respuesta dada la misma entrada, pero están diseñados e implementados de manera diversa.
- **Analytic Redundancy**. Permite no sólo la diversidad entre los lados privados de los componentes, sino también la diversidad entre las entradas y las respuestas de los componentes. Esta táctica está destinada a tolerar errores de especificación mediante el uso de especificaciones de requisitos separadas.

- **Exception Detection**. Se refiere a la detección de las condiciones de un sistema que altera el flujo normal de ejecución.
- **Self-Test**. Los componentes pueden ejecutar procedimientos para autoverificar su correcta operación. Los procedimientos de autoevaluación pueden ser iniciados por el propio componente, o ser invocados periódicamente por un sistema monitor.

- **Recuperarse de fallos**

  - **Preparación y Reparación**
    - **Active Redundancy**. Se refiere a una configuración en la que todos los nodos (activos o de repuesto redundantes) en un grupo de protección reciben y procesan entradas idénticas en paralelo, lo que permite que los repuestos redundantes mantengan un estado sincrónico con los nodos activos.
    - **Passive Redundancy**. Se refiere a una configuración donde sólo los miembros activos de un grupo de protección procesan el tráfico de entrada; una de sus funciones es proporcionar a los repuestos redundantes actualizaciones periódicas de estado.
    - **Cold Spare**. Se refiere a una configuración donde los repuestos redundantes de un grupo de protección permanecen fuera de servicio hasta que ocurre una falla, momento en el que se inicia un procedimiento de encendido y reinicio en el repuesto redundante antes de ponerlo en servicio.
    - **Exception Handling**. Cuando una excepción se detecta, el sistema debe manejarla de alguna manera. Lo más simple sería permitir que el sistema falle, pero es una idea terrible desde el punto de vista de disponibilidad, usabilidad, comprobabilidad y sentido común.
    - **Rollback**. Permite al sistema retroceder a un buen estado previo conocido al detectar una falla. Una vez se alcanza el buen estado, la ejecución puede continuar.

- **Software Upgrade**. Permite lograr actualizaciones en el software sin afectar la disponibilidad de los servicios.
- **Retry**. Esta táctica asume que las causas que causaron fallos es temporal y al reintentar la operación se podrá completar.
- **Ignore Faulty Behavior**. Exige ignorar los mensajes enviados desde una fuente en particular cuando se determina que esos mensajes son falsos.
- **Degradation**. Permite operar las funciones más importantes de un sistema en presencia de fallos, eliminando las funciones menos importantes.
- **Reconfiguration**. Intenta recuperarse de las fallas de los componentes al reasignar responsabilidades a los recursos (potencialmente restringidos) que quedaron en funcionamiento, mientras se mantiene la mayor funcionalidad posible.

- **Reintroducción**

  - **Shadow**. Se refiere a operar un componente que falló anteriormente o que se actualizó mientras estaba en servicio en un "modo de sombra"durante un período de tiempo predefinido antes de revertir el componente a un rol activo.
  - **State Resynchronization**. Es un socio de reintroducción de las tácticas de preparación y reparación de la redundancia activa y pasiva. Cuando se usa junto con la táctica de redundancia activa, la resincronización de estado ocurre orgánicamente, porque los componentes activos y de reserva reciben y procesan entradas idénticas en paralelo.
  - **Escalating Restart**. Permite que un sistema se recupere de fallas variando la granularidad de los componentes reiniciados y minimizando el nivel de servicio afectado.
  - **Non-Stop Forwarding**. Es un concepto que se originó en el diseño de enrutadores. En este diseño, la funcionalidad se divide en dos partes, el plano supervisor o de control (que gestiona la conectividad y la información de enrutamiento) y el plano de datos (que hace el trabajo real de enrutar los paquetes del remitente al receptor).

- **Prevenir fallos**

  - **Removal from Service**. Se refiere a colocar temporalmente un componente del sistema en un estado fuera de servicio con el fin de mitigar posibles fallas del sistema.
  - **Transactions**. Los sistemas destinados a servicios de alta disponibilidad aprovechan la semántica transaccional para garantizar que los mensajes asíncronos intercambiados entre componentes distribuidos sean atómicos, consistentes, independientes y duraderos.
  - **Predictive Model**. Un modelo predictivo, cuando se combina con un monitor, es utilizado para monitorear el estado de salud de un proceso del sistema para garantizar que el sistema está funcionando dentro de sus parámetros operativos nominales y para tomar medidas correctivas cuando se detectan condiciones que predicen fallas futuras.

- **Exception Prevention**. Esta táctica se emplea con el propósito de evitar que ocurran excepciones del sistema.
- **Increase Competence Set**. El conjunto de competencias de un programa es el conjunto de estados en los que es "competente"para operar.

Para el atributo de calidad de **interoperabilidad**, Bass et al. (2012) definen las siguientes tácticas:

- **Localizar**
  - **Discover Service**: ubicar un servicio por medio de la búsqueda de directorio de servicios. Pueden existir múltiples niveles de indirección en el proceso de localización, esto se refiere que a una ubicación conocida apunta a otra ubicación que a su vez se puede buscar para el servicio. El servicio puede ser localizado por el tipo de servicio, nombre, ubicación, o algún otro atributo.
- **Administrar interfaces**
  - **Orchestrate**: es una táctica que utiliza el mecanismo de control para coordinar, administrar y secuenciar la invocación de servicios particulares (que podrían ignorarse entre sí). La orquestación se utiliza cuando los sistemas interoperativos deben interactuar para realizar una tarea compleja; la orquestación "escribe"la interacción.
  - **Interfaz a medida (Tailor Interfaces)**: es una táctica que agrega o elimina capacidades a una interfaz. Se pueden agregar capacidades como traducción, adición de almacenamiento en buffer o suavizado de datos. Las capacidades también pueden eliminarse.

2.1.6. Evaluar una Arquitectura de Software

Garantizar que una arquitectura de software cumple los requerimientos funcionales como los no funcionales es un aspecto fundamental en el diseño arquitectónico. Se debe asegurar que una vez implementada la arquitectura, el sistema funcionará tal como se espera. Por este motivo, uno de los objetivos de este proyecto de grado es evaluar de manera formal la arquitectura que se diseñe.

En el estudio comparativo realizado por Raza et al. (2019) analizaron los diferentes métodos propuestos para la evaluación de arquitecturas de software. Los autores afirman que *"el objetivo de la evaluación de la arquitectura de software es estudiar las capacidades de la arquitectura seleccionada para transmitir un sistema apto para cumplir con las necesidades de calidad requeridas y reconocer cualquier riesgo potencial"*. Los errores detectados durante la evaluación de la arquitectura pueden ser resueltos antes de que la arquitectura empiece a ser implementada.

Raza et al. (2019) presentan una comparación exhaustiva de los métodos de evaluación de arquitectura de software más utilizados tanto en el ámbito académico como en la industria:

- **Software Architecture Analysis Method (SAAM)**. Propuesto en 1993. SAAM en un principio se llamó *Scenario-base Architecture Evaluation Method*. Es útil para realizar análisis rápidos de numerosos atributos de calidad tales como modificabilidad, portabilidad, extensibilidad, integrabilidad y cobertura funcional.

- **Architecture based Tradeoff Analysis Method (ATAM)**. Propuesto en el 2000. El propósito de ATAM es brindar un método de orientación para comprender la capacidad de numerosos atributos de calidad en competencia en la arquitectura de software. ATAM percibe la necesidad de un compromiso entre diferentes atributos de calidad cuando se determina la arquitectura de software de un sistema y antes de que se produzca el sistema.

- **Architecture-Level Modifiability Analysis (ALMA)**. Propuesto en 1996. ALMA es una técnica basada en escenarios apropiada para el cálculo de la modificabilidad del diseño de software mediante el empleo de un conjunto de indicadores: predicción del presupuesto de mantenimiento y evaluación de peligros. En la situación de evaluar y relacionar diferentes sistemas, los exámenes de modificabilidad realizados con ALMA también ayudan a elegir la arquitectura de software.

- **SAAM for Complex Scenarios (SAAMCS)**. SAAMCS es una ampliación de SAAM. El objetivo fundamental de SAAMCS es evaluar los peligros en medio del marco del cambio. Esta estrategia maneja detalles particulares; en consecuencia, se pretende ejecutar atributos de calidad de modificabilidad o adaptabilidad.

- **Cost Benefit Analysis Method (CBAM)**. CBAM simplifica los análisis financieros basados en la arquitectura de los sistemas intensivos de software. Esta técnica ayuda a los inversores de sistemas a seleccionar entre opciones arquitectónicas para mejorar el diseño y las diferentes fases de mantenimiento del sistema. Se apoya en un método de análisis basado en escenarios adecuado para evaluar el costo, los beneficios y las implicaciones de programación de las decisiones de arquitectura.

- **Family-Architecture Assessment Method (FAAM)**. El propósito de FAAM es centrarse en los atributos de calidad y la evaluación de la arquitectura. En el proceso de creación de un producto, FAAM involucra a todas las partes interesadas del producto.

Para lograr el objetivo de este proyecto proyecto se debe seleccionar el método adecuado para realizar la evaluación de la arquitectura. Con esto se podrá verificar que los atributos de disponibilidad e interoperabilidad se logran de manera satisfactoria con la arquitectura diseñada.

2.2. Estado del arte

Vaova actualmente tiene un funcionamiento mixto en su operación. Es decir, depende de sistemas propios y de terceros para llevar a cabo los viajes que realiza. Los sistemas propios y las integraciones le permiten realizar la venta, reservación, planeación y ejecución de los viajes. Ya se han mencionado

los muchos problemas que tiene la compañía. En esta sección se explicarán los esfuerzos realizados para comprender mejor la situación actual.

2.2.1. Sistemas Propios

El anterior equipo de tecnología desarrolló algunos sistemas con el objetivo de facilitar la operación de la compañía. Por diversas circunstancias, varios de los proyectos que se iniciaban no se terminaban y los que se lograban terminar tenían una baja calidad. Existen dos proyectos terminados dentro de Vaova:

- **MyTrip**. Este sistema se utiliza en la etapa de reservación y planeación de los viajes. Los viajeros deben ingresar a la plataforma con un usuario y contraseña que se genera automáticamente. El viajero debe elegir los vuelos en los cuales se desea desplazar, ya que Vaova ofrece múltiples opciones de aerolíneas. Debe elegir la acomodación, para esto debe seleccionar el hotel, la habitación y si desea compartir la habitación debe elegir el compañero de cuarto. Finalmente, debe verificar que las actividades que eligió durante la compra están acorde a las actividades reservadas. Esta plataforma es funcional y hacer parte de la operación, tiene errores identificados que el nuevo equipo de tecnología está tratando de solucionar.

- **Backoffice**. Se cuenta con un sistema de Backoffice, cuya idea inicial era permitir la administración de clientes, aerolíneas, hoteles, actividades, destinos, etc. Este sistema no es funcional y no tiene mecanismos de seguridad. El nuevo equipo de tecnología está considerando retirar este Backoffice y hacer uno nuevo que sí ayude a la compañía en su operación.

Vaova complementa su operación con aplicaciones de ofimática como Excel para llevar registro de los viajes, viajeros, itinerarios, reservaciones, hoteles, vuelos, actividades, entre otros. También utiliza PowerPoint para realizar las presentaciones o demostraciones de todo su portafolio de servicios a los promotores.

2.2.2. Integraciones

En los esfuerzos que se realizaron para operar con las plataformas de terceros, se realizaron integraciones. Estas integraciones se ha ido mejorando desde que se estableció el nuevo equipo de tecnología. Las siguientes son las integraciones con las que Vaova cuenta hasta diciembre de 2022:

- **TourHero y WeTravel**. Las plataformas de estos proveedores se utilizan para la venta de los viajes. Un viajero realiza la compara desde alguna de estas plataformas y registra su información. Estas plataformas no envían la información de las ventas a Vaova, por lo tanto es Vaova la que debe consultar esta información de manera constante. Se desarrollaron proyectos en Cloud Functions de Google Cloud que cada 15 minutos se conectan a las plataformas de TourHero y WeTravel y descargan la información de las ventas en un archivo de Excel que luego es utilizado para la reservación, planeación y ejecución de los viajes.

- **Viaxlab**. Esta integración permite cargar la información de viajes, viajeros, destinos y actividades a la aplicación móvil que también es de Viaxlab. Esta integración es inestable por los problemas de disponibilidad que tiene Viaxlab con sus servicios, además de lo costoso que es utilizar la aplicación móvil.

Con los sistemas propios y las integraciones Vaova ha podido funcionar a pesar de los problemas, pero sabe que debe cambiar muchas cosas tanto internas como externas para poder operar de una manera más efectiva. El propósito general de este proyecto es diseñar una arquitectura que pueda guiar a Vaova a desarrollar los sistemas que necesita para mejorar su operación diaria.

2.3. Metodología de la investigación

El objetivo central de este proyecto se centra en el diseño de una arquitectura de software, por ende la metodología que se utilizará será una que faciliten el diseño y la evaluación de la arquitectura. Por este motivo se ha decidido utilizar el modelo Attribute-Driven Design (ADD) para el diseño y Software Architecture Analysis Method (SAAM) como complemento para realizar la evaluación.

2.3.1. Domain-Driven Design - DDD

El diseño dirigido por dominios es una filosofía de desarrollo definida por Eric Evans en el libro *Domain-Driven Design: Tackling Complexity in the Heart of Software*. DDD es un enfoque que permite a los equipos administrar de manera efectiva la construcción y el mantenimientos de software para dominios de problemas complejos (Millett and Tune, 2015).

Evans (2015) elaboró un glosario con los términos más importantes para entender DDD:

- **Dominio**: campo de conocimiento, influencia o actividad. El área temática a la que el usuario aplica un programa es el dominio del software.
- **Modelo**: sistema de abstracciones que describen los aspectos seleccionados de un dominio y puede se utilizado para solucionar problemas relacionados a dicho dominio.
- **Lenguaje ubiquo**: lenguaje estructurado alrededor del modelo de dominio y utilizado por todos los miembros del equipo dentro de un contexto acotado para conectar todas las actividades del equipo de software.
- **Contexto**: entorno en el que aparece una palabra o declaración que determina su significado. Las declaraciones sobre un modelo sólo pueden entenderse en un contexto.
- **Contexto delimitado**: descripción de un límite (típicamente un subsistema, o el trabajo de un equipo particular) dentro del cual se define y aplica un modelo particular.

Para trabajar con DDD, es necesario emplear un conjunto de patrones tanto a nivel estratégico como a nivel táctico. Estos patrones son explicados por Millett and Tune (2015):

- Los patrones estratégicos destilan el dominio del problema y dan forma a la arquitectura de una aplicación. Estos son los patrones que se aplican para diseñar la estrategia:
  - Destilar el dominio del problema para revelar lo qué es importante.
  - Crear un modelo para solucionar problemas del dominio.
  - Usar un lenguaje compartido para habilitar la colaboración de modelado.
  - Aislar los modelos de la ambigüedad y corrupción.
  - Comprender las relaciones entre los contextos.
- Los patrones tácticos, también conocidos modelo de bloques de construcción, son una colección de patrones que ayudan a crear modelos efectivos para contextos delimitados complejos. Muchos de los patrones de codificación presentados dentro de la colección de patrones tácticos han sido previamente definidos por otros autores como Martin Fowler en *Patterns of Enterprise Application Architecture* y Erich Gamma et al. en *Design Patterns: Elements of Reusable Object-Oriented Software*. Estos patrones no son aplicables a todos los modelos, y cada uno debe tomarse por sus propios méritos con el estilo arquitectónico correcto aplicado.

En la Figura 2.6 se muestran todos los elementos que se utilizan al momento de diseñar una arquitectura de software con DDD, entre ellos se pueden ver el lenguaje ubicuo, los contextos delimitados y algunos de los patrones estratégicos y tácticos. Esta imagen resume todos los pasos, beneficios y consecuencias de utilizar DDD.

2.3.2. Attribute-Driven Design - ADD

Barbacci et al. (2001) plantea en su reporte técnico que Attribute-Driven Design es un enfoque para definir una arquitectura de software en la cual el proceso de diseño se basa en los requerimientos de los atributos de calidad del software. Sigue un proceso de diseño recursivo que descompone un sistema o elemento de un sistema aplicando tácticas y patrones de arquitectura para satisfacer sus requisitos. ADD sigue en esencia una secuencia de "Planear, Hacer y Verificar":

- **Planear**. Atributos de calidad y restricciones de diseño que son considerados para seleccionar qué tipos de elementos serán utilizados en la arquitectura.
- **Hacer**. Los elementos se instancian para satisfacer los requerimientos de los atributos de calidad así como los requerimientos funcionales.
- **Verificar**. El diseño resultante se analiza para determinar si los requerimientos se cumplen.

El proceso de "Planear, Hacer y Verificar"se repite hasta que los requerimientos arquitectónicamente significativos se cumple a cabalidad. En la Figura 2.7 se describe el proceso de "Planear, Hacer y Verificar", cada etapa de este proceso consta de insumos, pasos y resultados.

Figura 2.6: Elementos de DDD (Evans, 2015)

2.3.2.1. Entradas de ADD

Las entradas que ADD necesita son los requerimientos funcionales, restricciones de diseño y los requerimientos de atributos de calidad que las partes interesadas del sistema han priorizado de acuerdo a las necesidades del negocio.

- Los requerimientos funcionales especifican qué funciones debe proporcionar un sistema para satisfacer las necesidades establecidas e implícitas de las partes interesadas cuando el software se utiliza en condiciones específicas.
- Las restricciones de diseño son decisiones sobre el diseño del sistema que deben ser incorporadas en el diseño final del sistema. Representan una decisión de diseño con un resultado predeterminado.
- Los requerimientos de atributos de calidad son requerimientos que indican el grado en el cual un sistema debe exhibir varias propiedades.

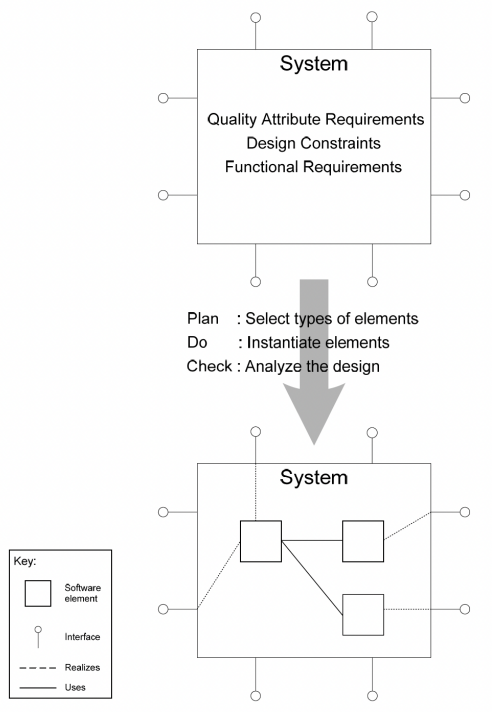


Figura 2.7: ADD - Planear, Hacer y Verificar (Barbacci et al., 2001)

2.3.2.2. Salidas Esperadas de ADD

La salida de ADD es un diseño del sistema en términos de los roles, responsabilidades, propiedades y relaciones entre los elementos del software.

- **Elemento de software**: un artefacto computacional o de desarrollo que cumple varios roles y responsabilidades, tiene propiedades definidas y se relaciona con otros elementos de software para componer la arquitectura de un sistema.
- **Rol**: es un conjunto de responsabilidades relacionadas.
- **Responsabilidad**: funcionalidad, datos, or información que un elemento de software proporciona.
- **Propiedad**: información adicional de un elemento de software como el nombre, tipo, atributo de calidad característico, protocolo y demás.
- **Relación**: definición de cómo dos elementos de software están asociados o interactúan entre sí.

El diseño que resulta de ADD es documentado utilizando varios tipos de vistas arquitectónicas tales como vista de módulo, vista de componente-conector, vista de asignación, entre otras.

2.3.2.3. Pasos de ADD

1. **Confirmar que hay suficiente información de los requerimientos**: se debe confirmar si existe información suficiente acerca de los requerimientos para proceder con ADD. En esencia, se debe asegurar que los interesados en el sistema han priorizado los requerimientos de acuerdo a las necesidades y objetivos del negocio. También se debe confirmar que existe suficiente información sobre los requerimientos de los atributos de calidad para proceder.

2. **Seleccionar un elemento del sistema a descomponer**: se selecciona qué elemento del sistema será el centro de atención del diseño en los pasos posteriores. Se puede llegar a este paso de dos formas:

   Se ejecuta el paso 2 por primera vez, por lo que el único elemento que se puede descomponer es el sistema en sí mismo.

   Se está refinando un sistema parcialmente diseñado y ya se ha pasado antes por el paso 2. En este caso, el sistema ha sido particionado en dos o más elementos y los requerimientos han sido asignado a dichos elementos. Se debe seleccionar uno de estos elementos como el centro de atención de los siguientes pasos. En este caso se debe elegir el elemento a centrarse basado en uno de cuatro áreas de interés:

   - Conocimiento actual de la arquitectura.
   - Riesgo y dificultad.
   - Criterios del negocio.
   - Criterios de la organización.

3. **Identificar los controladores arquitectónicos candidatos**: en este punto se ha seleccionado un elemento del sistema para descomponer y los interesados del sistema han priorizado algún requerimiento que afecta a dicho elemento. Durante este paso, se deben clasificar los mismos requerimientos por segunda vez en función de su impacto relativo en la arquitectura.

4. **Seleccionar un concepto de diseño que satisfaga los *drivers* arquitectónicos**: se debe elegir los principales tipos de elementos que aparecerán en la arquitectura y los tipos de relaciones entre ellos. Las restricciones de diseño y los requerimientos de atributos de calidad son utilizados para determinar los tipos de elementos, relaciones y sus interacciones.

5. **Crear instancias de elementos arquitectónicos y asignar responsabilidades**: se instancian los tipos de elementos de software que se eligieron en el paso anterior. A los elementos instanciados se le asignan responsabilidades de acuerdo a su tipo. Las responsabilidades para los elementos instanciados también se derivan de los requerimientos funcionales asociados con los controladores arquitectónicos candidatos y los requisitos funcionales asociados con el elemento principal. Al final de este paso, cada requisito funcional asociado con el elemento principal debe estar representado por una secuencia de responsabilidades dentro de los elementos secundarios.

6. **Definir interfaces para los elementos instanciados**: se definen los servicios y propiedades requeridas y proporcionados por los elementos de software del diseño. En ADD, estos servicios y propiedades se conocen como la interfaz del elemento. Se debe tener presente que una interfaz no es simplemente una lista de firmas de operaciones. Las interfaces describen las suposiciones *proporcionadas* y *requeridas* que los elementos de software hacen entre sí.

7. **Verificar y refinar los requerimientos y convertirlos en restricciones para los elementos instanciados**: se verifica que la descomposición de elementos hasta el momento cumple con los requisitos funcionales, los requisitos de los atributos de calidad y las restricciones de diseño. También se preparan los elementos secundarios para una mayor descomposición.

8. **Repetir pasos 2 a 7 para el siguiente elemento del sistema que se desea descomponer**: una vez se han completado los pasos 1 a 7, se tiene una descomposición del elemento padre en elementos hijos. Cada elemento hijo es una colección de responsabilidades, cada uno teniendo una descripción de interfaz, requerimientos funcionales, requerimientos de atributos de calidad y restricciones de diseño. Ahora se puede pasar al proceso de descomposición en el paso 2 donde se debe seleccionar el siguiente elemento a descomponer.

En la Figura 2.8 se resumen todos los pasos de ADD, en esta se muestran las entradas necesarias, la sucesión de etapas, la manera de realizar las iteraciones y las salidas que se generan con ADD.

En el marco de este proyecto de grado se harán las iteraciones necesarias para satisfacer los dos atributos de calidad se desean satisfacer con el diseño de la arquitectura de software.

2.3.3. Software Architecture Analysis Method - SAAM

Ionita et al. (2002) realizaron un estudio detallado de múltiples métodos para realizar la evaluación de arquitecturas de software, uno de esos métodos es SAAM. Los autores describen que SAAM es un método capaz de expresar los diferentes atributos de calidad de las arquitecturas de software por medio de escenarios y puede evaluarlos contra los reales. SAAM ha demostrado ser útil para evaluar rápidamente muchos atributos de calidad como la modificabilidad, la portabilidad, la extensibilidad, la integrabilidad y la cobertura funcional. El método también se puede utilizar para evaluar los aspectos de calidad de las arquitecturas de software como el rendimiento o la fiabilidad.

Este método se puede emplear para analizar una o más arquitecturas de software. Si sólo una arquitectura es analizada, SAAM indica los puntos débiles o los puntos fuertes, junto con los puntos donde la arquitectura falla en satisfacer los requerimientos de modificabilidad. Si dos o más arquitecturas son analizadas, el método genera un ranking entre ellas con respecto a su modificabilidad.

Al igual que ADD, SAAM involucra una serie de pasos iterativos para su ejecución. Este método consiste en seis pasos principales, que típicamente están precedidos por una corta revisión general del contexto del negocio y las funcionalidades requeridas del sistema. Los pasos son:

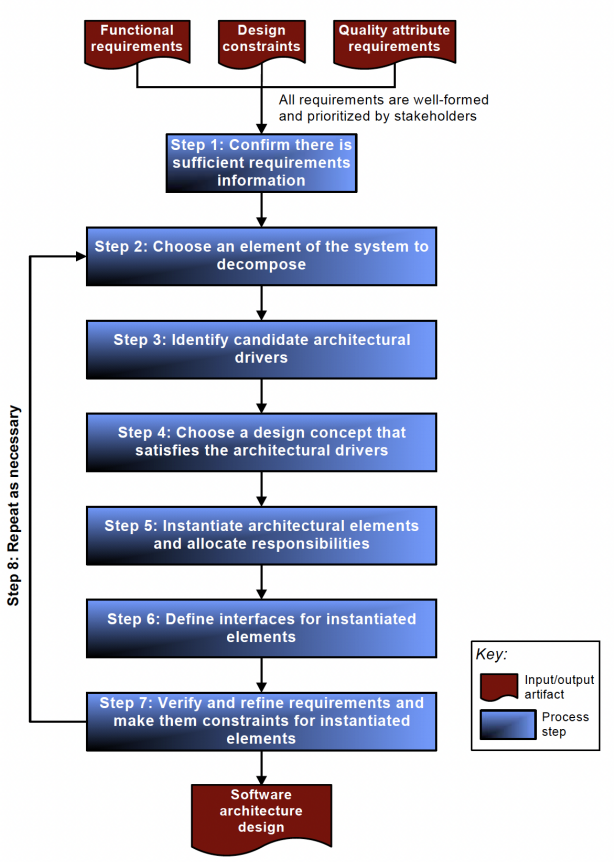


Figura 2.8: ADD - Pasos (Barbacci et al., 2001)

1. **Desarrollo de los escenarios**. Se debe realizar un ejercicio de lluvia de ideas con el objetivo de identificar los tipos de actividades que el sistema debe soportar. Estas actividades junto con las posibles modificaciones que los interesados en el sistema pueden anticipar se agrupan en los llamados *escenarios del sistema*. El reto en el desarrollo de los escenarios es capturar los mayores usos y usuarios del sistema, todos los atributos de calidad y el nivel asociado que el sistema debe alcanzar, además de todos los cambios futuros previsibles en el sistema.

2. **Describir la arquitectura(s)**. Se presenta la arquitectura o arquitecturas. Las notaciones utilizadas en la arquitectura deben ser entendidas por los involucrados y deben indicar la representación estática (componentes, sus conexiones y la relación con el ambiente) así como el comportamiento dinámico del sistema.

3. **Clasificación y priorización de escenarios**. Los escenarios del sistema son clasificados en *directos* e *indirectos*. Un escenario directo es soportado por la arquitectura candidata porque se basa en los requerimientos funcionales. Los escenarios directos son candidatos a ser métricas del rendimiento o la confiabilidad de la arquitectura. Un escenario indirecto

es aquella secuencia de eventos que para su realización la arquitectura debe ser modificada con cambios menores o mayores. La priorización de los escenarios se basa en un proceso de votación.

4. **Evaluación individual de escenarios indirectos**. En el caso de los escenarios directos el arquitecto debe demostrar cómo el escenario sería ejecutado por la arquitectura. Para los escenarios indirectos el arquitecto describe la manera en que la arquitectura debería ser modificada para incluir el escenario. Por cada escenario indirecto se deben identificar las modificaciones arquitectónicas necesarias para facilitar ese escenario junto con los componentes del sistema que se impactan o los nuevos que se agregan y el costo y esfuerzo estimados para implementar la modificación.

5. **Evaluar la interacción del escenario**. Cuando dos o más escenarios implican cambios en el mismo componente o componentes de la arquitectura, se dice que esos escenarios interactúan. En este caso, los componentes afectados se deben modificar o dividir en subcomponentes con la idea de evitar la interacción de escenarios diferentes.

6. **Crear una evaluación general**. Se asigna un peso a cada escenario en términos de su importancia relativa para el éxito del sistema. La ponderación se relaciona con los objetivos comerciales respaldados por un escenario u otros criterios como costos, riesgos, tiempo de comercialización, etc. Se pueden proponer alternativas para la arquitectura más adecuada, cubriendo los escenarios directos y requiriendo menos cambios para soportar los escenarios indirectos.

En la Figura 2.9 se muestran los pasos de SAAM. Se puede notar que los pasos de *Desarrollo de los escenarios* y *Describir la arquitectura(s)* pueden pasar a *Clasificación y priorización de los escenarios* o *Evaluación individual de los escenarios indirectos*. Esto es así porque en el artículo de Kazman et al. (1996) no se incluía el paso de *Clasificación y priorización de los escenarios*.

El método de SAAM se empleará como un complemento a ADD, para verificar que realmente se cumplen tanto con los requerimientos funcionales del sistema así como los atributos de calidad de disponibilidad e interoperabilidad propuestos.

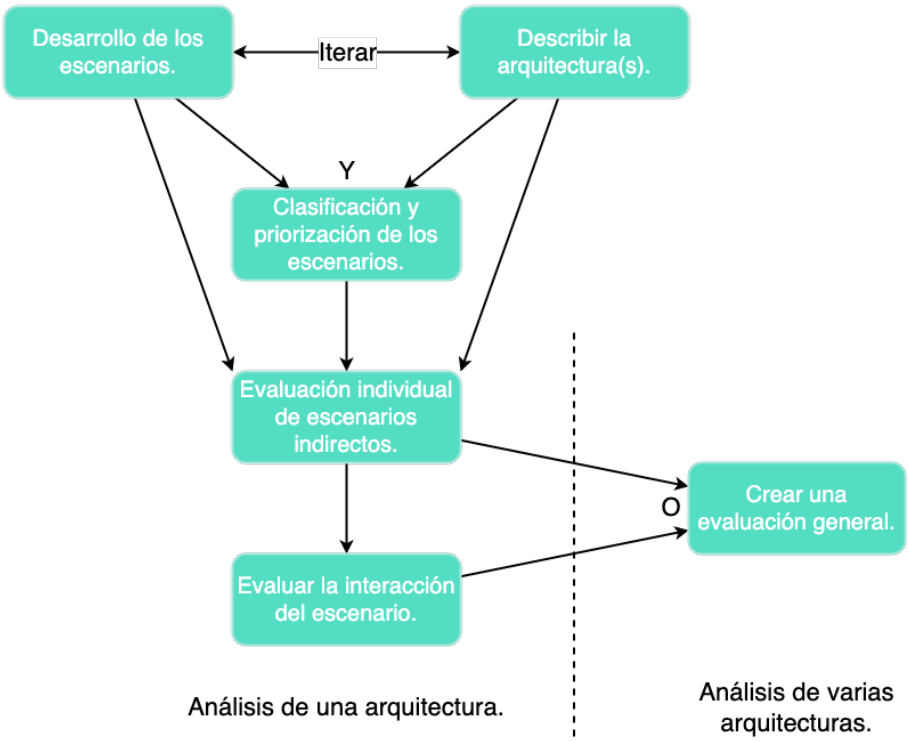


Figura 2.9: SAAM - Pasos

Capítulo 3

Desarrollo del Proyecto

3.1. Negocio de Vaova Travel

Antes de evaluar el estado actual de la compañía o diseñar una arquitectura de software era importante conocer cómo funciona el modelo de negocio de Vaova. Para resolver un problema es necesario comprenderlo muy bien, por lo cual se inició con la elaboración de un *Business Model Canvas* que permite conocer de manera rápida y concreta aspectos como propuesta de valor, segmentos de clientes, canales, relación con los clientes, fuentes de ingresos, recursos clave, actividades clave, alianzas clave y estructura de costos.

A continuación se describen los resultados encontrados. La Figura 3.1 cuanta con un resumen gráfico del *Business Model Canvas*:

- **Propuesta de valor**. Es el Turismo Cultural basado en los happenings.
- **Segmentos de clientes**. La compañía cuenta con dos segmentos de clientes. Los principales son los universitarios, principalmente los Promotores y los Viajeros. El segmento secundario son las empresas.
- **Canales**. Los canales o medios por los que la compañía promociona sus productos son las plataformas de venta de *TourHero* y *WeTravel*. No tiene otro medio por el cual pueda vender sus productos.
- **Relación con los clientes**. La relación con los clientes no es duradera, ya que la mayoría son estudiantes universitarios. La compañía quiere ofrecer la mejor experiencia a los viajeros, por lo cual la relación con estos se basa en tres aspectos:
  - *Soporte previo al viaje*. En esta fase la compañía se enfoca en resolver cualquier duda que pueda surgir con respecto al viaje.
  - *Acompañamiento durante el viaje*. Durante el viaje los viajeros siempre tendrán a un guía que tratará de solucionar cualquier duda o inconformidad que se pueda presentar.
  - *Feedback después del viaje*. Vaova recibe los comentarios y recomendaciones de los viajeros para mejorar.
- **Fuentes de ingresos**. Los ingresos provienen en gran medida de la venta de *Treks* universitarios. En menor proporción de la venta de paquetes de reuniones empresariales.

- **Recursos clave**. Los recursos claves con los que cuenta la compañía son:
  - *Reservaciones*. Estas pueden ser de talento humano, de infraestructura, de locaciones, etc.
  - *Transporte interno*. Se garantiza a los viajeros que siempre tendrán transporte aéreo, terrestre y marítimo durante el viaje en el país de destino.
  - *Acomodación*. Los viajeros saben de antemano la oferta hotelera y de acomodaciones que la compañía puede ofrecer.
  - *Actividades*. Las actividades que Vaova organiza garantiza experiencias únicas a los viajeros.
- **Actividades clave**. En este apartado encontramos que las actividades clave de Vaova son:
  - *Diseño de actividades y paquetes de viajes*. Su propuesta de valor se basa en turismo cultura, por eso debe ofrecer las mejores experiencias en el lugar de destino. La venta de paquetes son su principal fuente de ingreso, estos paquetes deben ser lo suficientemente atractivos para que los viajeros los deseen adquirir.
  - *Negociación con proveedores*. Conseguir los mejores precios posibles sin comprometer la calidad y experiencia de viaje.
  - *Reservación de recursos*. El talento humano, las locaciones, la infraestructura, el transporte y las acomodaciones son recursos que Vaova debe garantizar en cada viaje.
  - *Inversión tecnológica*. La inversión en tecnología es la apuesta de la compañía para poder ser más competitivo en el mercado.
- **Alianzas clave**. Vaova tiene alianzas con múltiples compañías para ofrecer los viajes:
  - *Reservaciones*. Alianzas con plataformas de venta y recaudo como *TourHero* y *WeTravel*.
  - *Transporte interno*. Acuerdos con aerolíneas, empresas de transportes especiales y Tour Operadores.
  - *Acomodación*. Acuerdos con hoteles. Vaova no ofrece acomodaciones que no sean en hoteles.
  - *Actividades*. Cuenta con un grupo de guías turísticos, conferencistas y restaurantes que son de la confianza de Vaova.
- **Estructura de costos**. Los costos de Vaova se distribuyen de la siguiente manera:
  - *Comisiones de los Promotores*. Los promotores no deben pagar los viajes que organizan y Vaova les permite viajar con 5 acompañantes.
  - *Comisiones de los partners*. Vaova paga comisiones y facturas a las plataformas, servicios de transporte, hoteles, restaurantes, infraestructura, locaciones y talento humano que participan en los viajes.

- *Inversión tecnológica.* La inversión en tecnología es fuerte para poder solucionar los problemas de funcionamiento que se tienen de años pasados.
- *Salarios de los empleados.* La nómina que mes a mes debe pagar la compañía a sus empleados.

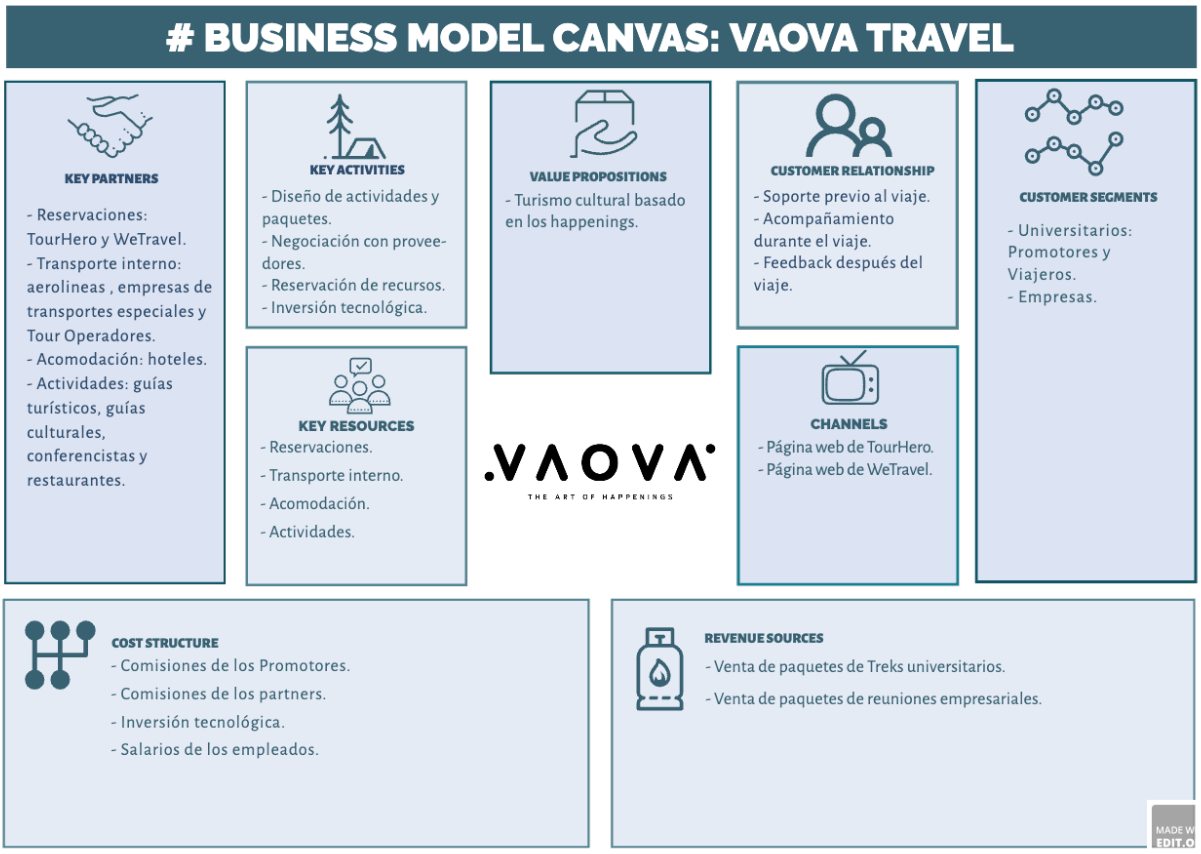


Figura 3.1: *Business Model Canvas* de Vaova

NOTA: La documentación completa del *Business Model Canvas* se puede encontrar en la Sección 6.1 de Anexos.

3.2. Procesos Operativos de Vaova

Lo siguiente que se analizó fueron los procesos operativos de Vaova. Este análisis se realizó con base al funcionamiento la compañía tenía en ese momento. Sólo se describen los procesos que hacen

parte de la planeación de los viajes, ya que estos son los que causas más problemas al momento de la ejecución. A continuación se describen los procesos encontrados:

- **Itinerarios**. Es el proceso en el cual los promotores crean los itinerarios y luego estos son revisados por los asesores comerciales de Vaova. La Figura 3.2 muestra el diagrama BPMN del proceso de Itinerarios.

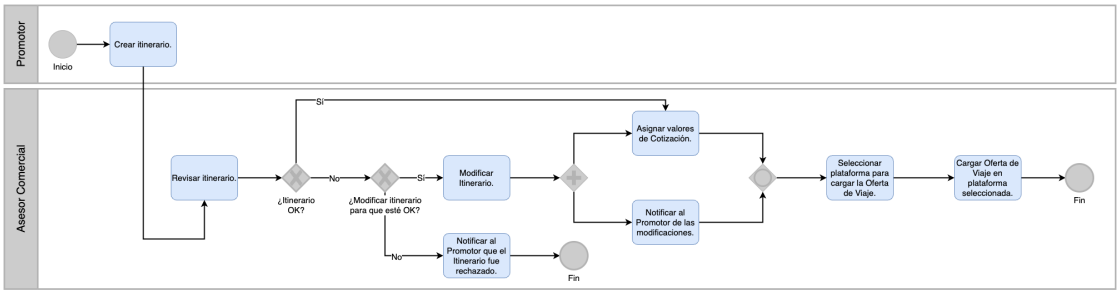


Figura 3.2: Diagrama BPMN de Itinerarios en Vaova

- **Sincronización con *WeTravel*** . El proceso de sincronización de información con *WeTravel* incluye los siguientes pasos:

  1. *Solicitud de información*. Períodicamente un componente de Vaova solícita la información a la plataforma de *WeTravel*. La Figura 3.3 muestra el diagrama BPMN de solicitud de información a *WeTravel*.

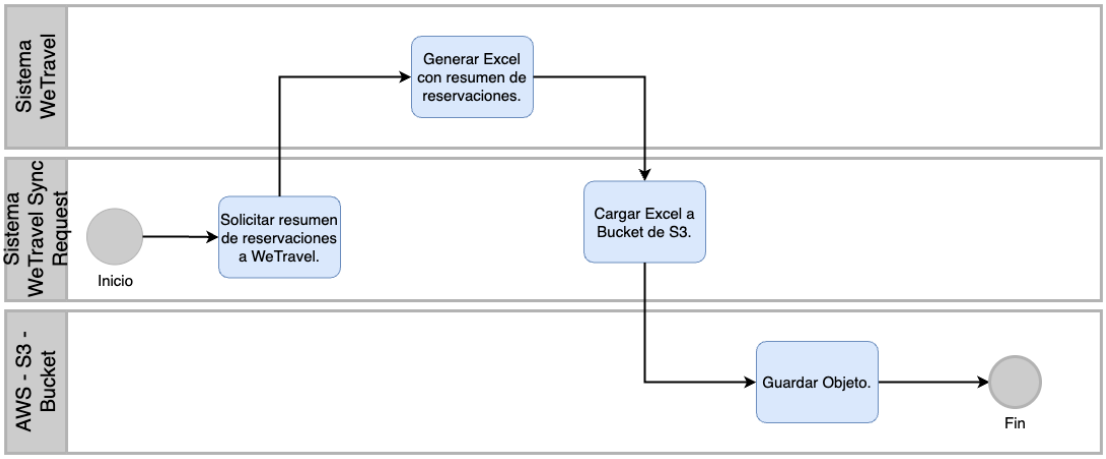


Figura 3.3: Diagrama BPMN Sincronización de *WeTravel* - Solicitud de Información

  2. *Procesamiento de la información*. Un componente de Vaova procesa la información obtenida desde la plataforma de *WeTravel*. La Figura 3.4 muestra el diagrama BPMN del procesamiento de la información obtenida desde *WeTravel*.

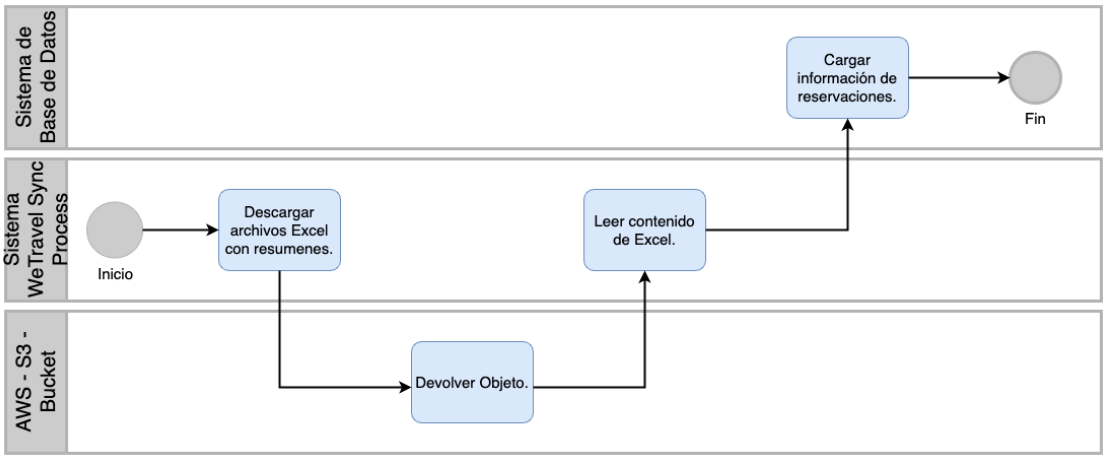


Figura 3.4: Diagrama BPMN Sincronización de *WeTravel* - Procesamiento de Información

- **Sincronización de *TourHero***. El proceso de sincronización de información con *TourHero* incluye los siguientes pasos:

  1. *Solicitud de información*. Periódicamente un componente de Vaova solícita la información a la plataforma de *TourHero*. La Figura 3.5 muestra el diagrama BPMN de solicitud de información a *TourHero*.

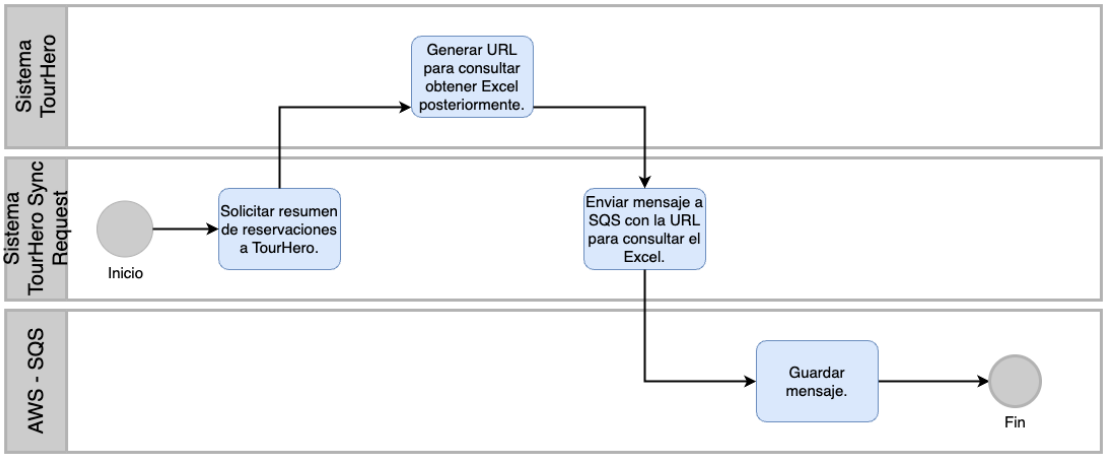


Figura 3.5: Diagrama BPMN de Sincronización de *TourHero* - Solicitud de Información

  2. *Descarga de información*. Un componente de Vaova se encarga de descargar la información desde *TourHero* una vez es notificado por el proceso de *Solicitud de información*. La Figura 3.6 muestra el diagrama BPMN de la descarga de información desde *TourHero*.

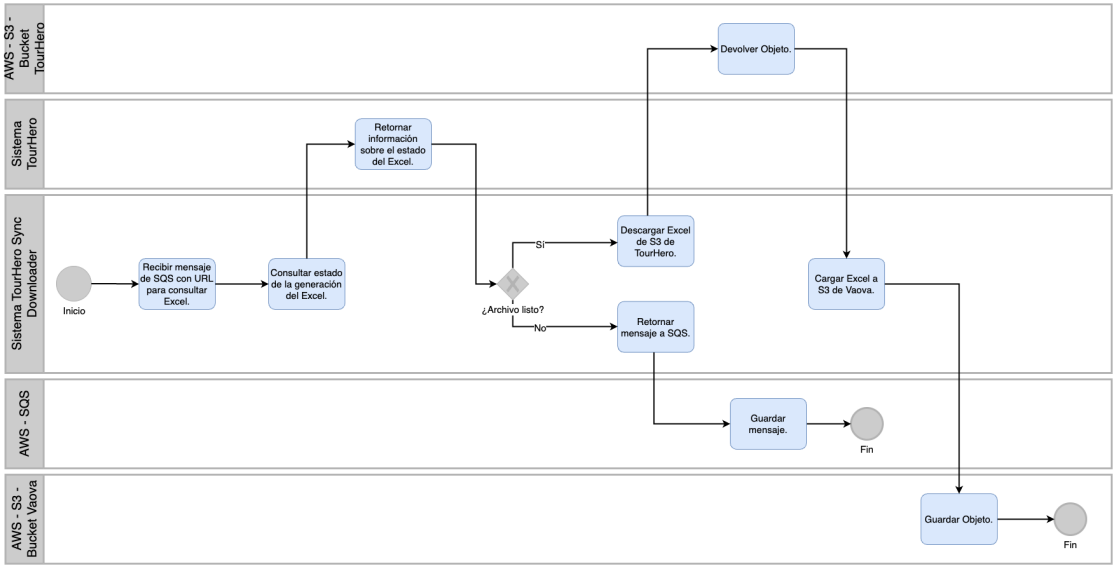


Figura 3.6: Diagrama BPMN de Sincronización de *TourHero* - Descarga de Información

3. *Procesamiento de información*. Un componente de Vaova procesa la información obtenidad desde la plataforma de *TourHero*. La Figura 3.7 muestra el diagrama BPMN del procesamiento de la información.

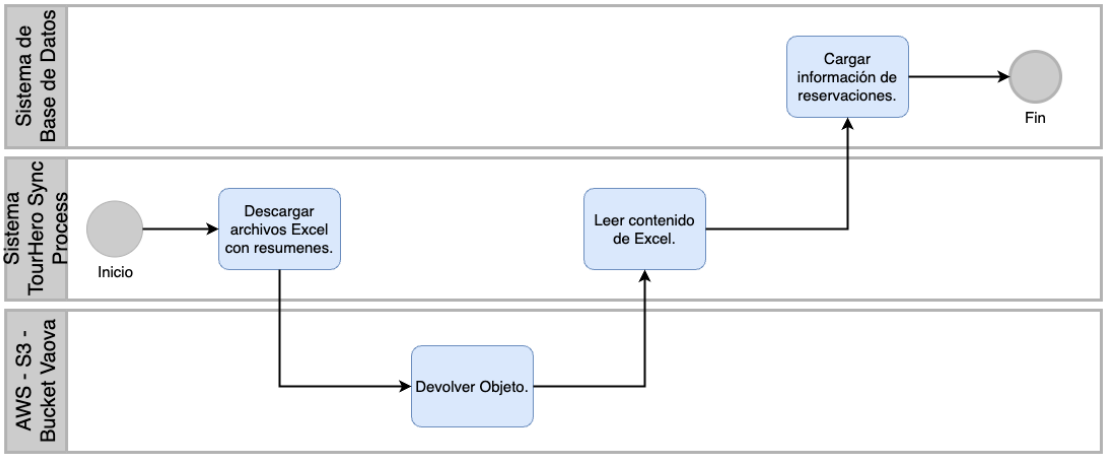


Figura 3.7: Diagrama BPMN de Sincronización de *TourHero* - Procesamiento de Información

- ***Join***. Los viajeros pueden elegir hoteles, acompañante y vuelos. La Figura 3.8 muestra el diagrama BPMN del proceso de *Join*.

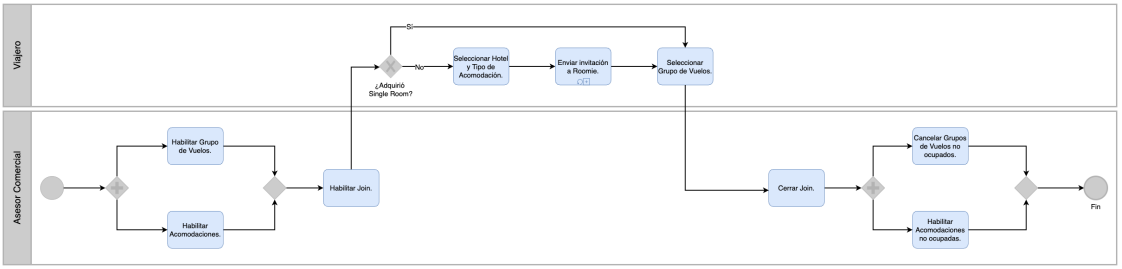


Figura 3.8: Diagrama BPMN *Join* de Vaova

- ***Roomie***. Los viajeros pueden aceptar o rechazar las invitaciones a ser *Roomie* de otro viajero. La Figura 3.9 muestra el diagrama BPMN del proceso de *Roomie*.

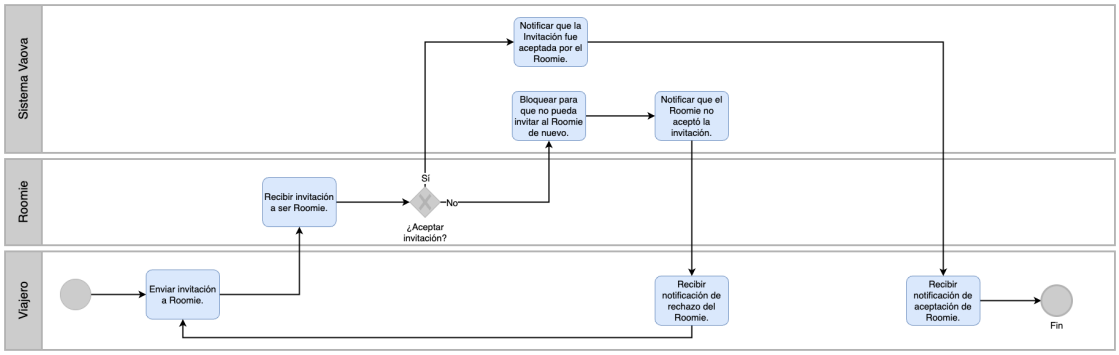


Figura 3.9: Diagrama BPMN *Roomie* de Vaova

NOTA: Los diagramas BPMN de los procesos se pueden encontrar en la Sección 6.2 de Anexos.

3.3. Evaluación del Estado Actual de Vaova

En esta sección se muestran los resultados de las actividades que se realizaron para analizar el estado actual de la compañía con relación a los sistemas de software.

3.3.1. Contexto Actual

Para el contexto actual se diseñaron diagramas con el Modelo C4, ya que estos permiten ir de lo más general a lo particular en los sistemas que se analizan. Para el análisis se realizaron los diagramas de contexto del sistema, contenedores y componentes. A continuación se presentan los diagramas, pero la explicación detallada de cada componente de software se realizará en la subsección 3.3.2.

- **Diagrama de contexto del sistema**. Muestra de manera general el ecosistema de Vaova y la manera en la que interactúan con los viajeros y las integraciones. La Figura 3.10 muestra el diagrama de contexto actual de Vaova.

  - *El viajero*. Compra el viaje en las plataformas de *TourHero* o *WeTravel*. Consolida su información personal y del viaje en los sistemas de Vaova. Durante el viaje consulta su cronograma en la aplicación de *Viaxlab*.

  - *Vaova*. Interactúa con las integraciones de *TourHero*, *WeTravel* y *Viaxlab* para consolidar la información de los viajeros.

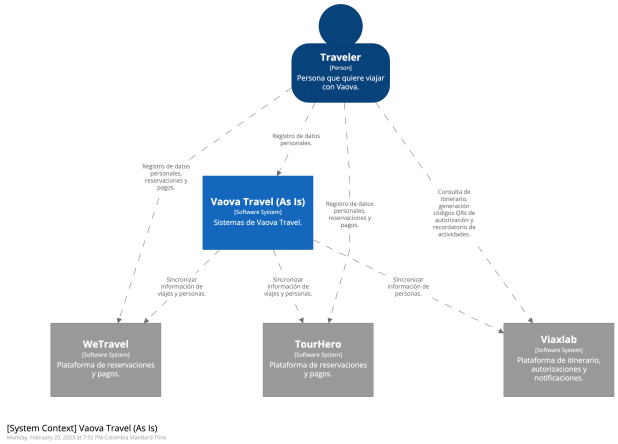


Figura 3.10: Diagrama C4 - Contexto del Sistema Actual de Vaova

- **Diagrama de contenedores**. Muestra los subsistemas con los que cuenta la compañía, además de la base de datos central que se utiliza. A este detalle todavía se muestran las interacciones con los viajeros y las integraciones. Los subsistemas son: *MyTrip*, *TourHero Sync*, *WeTravel Sync*, *Viaxlab Sync* y *Base de datos MongoDB*. La Figura 3.11 muestra el diagrama de contenedores actual de Vaova.

- **Diagrama de componente de *MyTrip***. El subsistema de *MyTrip* está compuesto de una capa *BackEnd* y una capa *FrontEnd*. Este sistema se comunica directamente con la base de datos *MongoDB*. La Figura 3.12 muestra el diagrama del componente de *MyTrip*.

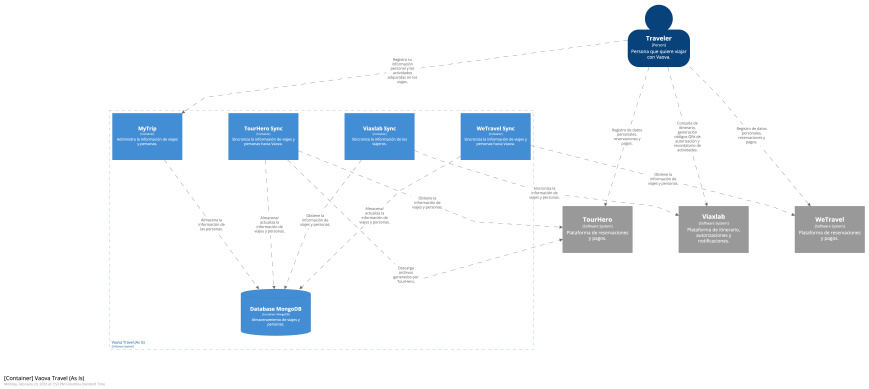


Figura 3.11: Diagrama C4 - Contenedores de Vaova

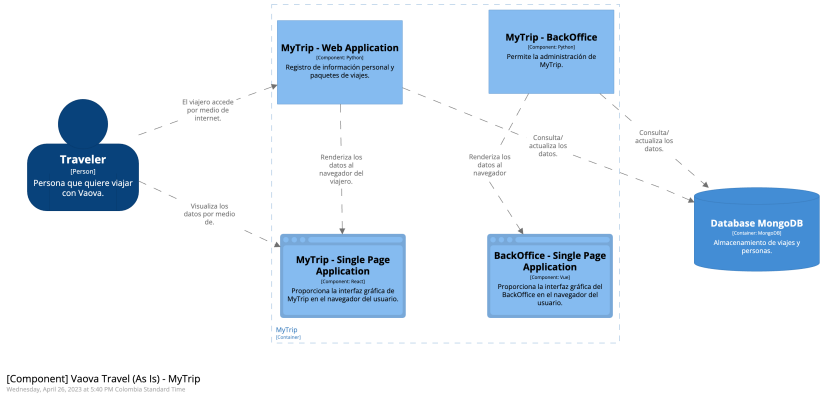


Figura 3.12: Diagrama C4 - Componente de *MyTrip*

- **Diagrama de componente de *TourHero Sync***. El subsistema de *TourHero Sync* está compuesto por tres módulos que permiten sincronizar la información desde *TourHero*: *Information Request*, *Information Downloader* y *Information Process*. La Figura 3.13 muestra el diagrama del componente de *TourHero Sync*.

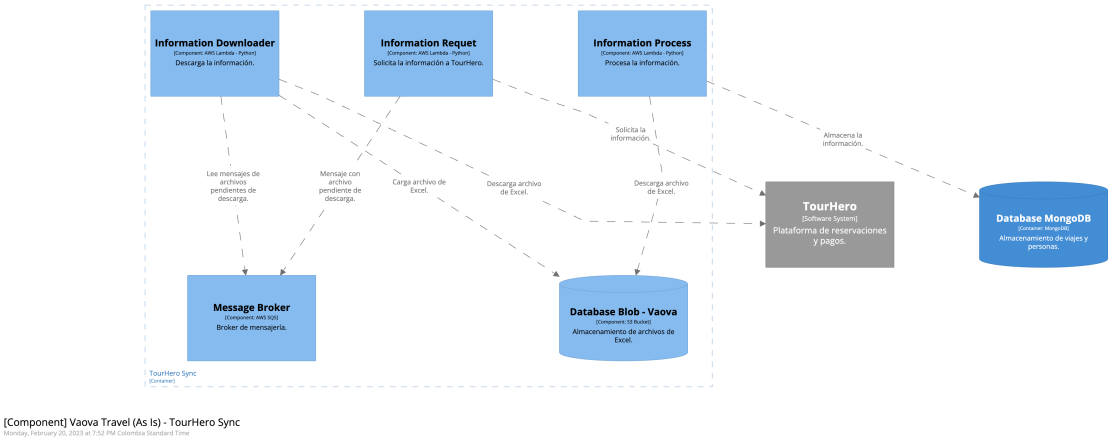


Figura 3.13: Diagrama C4 - Componente de *TourHero Sync*

- **Diagrama de componente de *WeTravel Sync***. El subsistema de *WeTravel Sync* está compuesto por dos módulos que permiten sincronizar la información desde *WeTravel*: *Information Request* y *Information Process*. La Figura 3.14 muestra el diagrama del componente de *WeTravel Sync*.

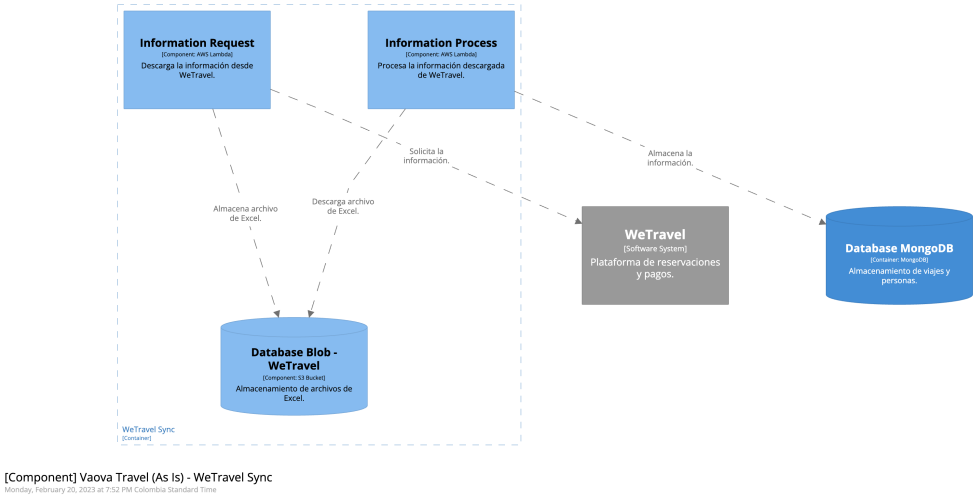


Figura 3.14: Diagrama C4 - Componente de *WeTravel Sync*

- **Diagrama de componente de *Viaxlab Sync***. Esta integración está conformada por 7 módulos que se encargan de consultar una parte de la información y sincronizarla en los sistemas de *Viaxlab*. Los módulos son los siguientes: *Itinerary Viaxlab*, *Group Viaxlab*, *Activity Viaxlab*, *User Viaxlab*, *Trip Viaxlab*, *Passenger Activities Vialab* y *Modify Passenger*. La

Figura 3.15 muestra el diagrama del componente de *Viaxlab Sync*.

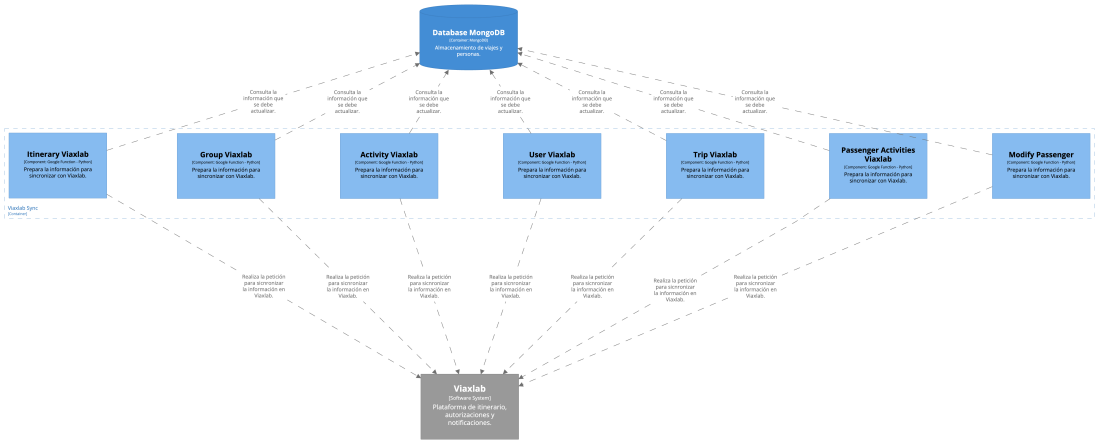


Figura 3.15: Diagrama C4 - Componente de *Viaxlab Sync*

NOTA: Los diagramas en el modelo C4 del Contexto Actual se pueden encontrar en la Sección 6.3 de Anexos.

3.3.2. Análisis de sistemas actuales

Para presentar los resultados del análisis de los sistemas e integraciones de Vaova, se mostrará en una tabla el análisis de cada sistema y se finaliza con una tabla comparativa que resume la información. Los sistemas analizados fueron:

- *MyTrip*
- *MyTrip - BackOffiece*
- Integración *TourHero*
- Integración *WeTravel*
- Integración *Viaxlab*
- Base de datos *MongoDB*

3.3.2.1. Sistemas propios

- ***MyTrip***

  La Tabla 3.1 contiene el resumen del análisis realizado al sistema *MyTrip*.

MyTrip	
Descripción	Este sistema fue pensado para que los viajeros pudieran realizar el *Join* de sus viajes. Para el *Join*, el viajero debía registrarse e iniciar sesión y realizar el siguiente proceso: verificar que sus datos personales están actualizados; verificar que los destinos y actividades del viaje son los mismos que seleccionó en la plataforma de venta (*TourHero* o *WeTravel*); seleccionar grupo de vuelos, acomodaciones y compañero de cuarto.
Problemas encontrados	El sistema fue construido con malas prácticas de diseño y desarrollo. No soporta más de 30 usuarios concurrentes y los tiempos de respuesta no son adecuados. No cuenta con mecanismos de seguridad, por lo que la información de los viajeros no se encuentra protegida.
BackEnd	Lenguaje *Python*.
FrontEnd	Lenguaje *JavaScript* con *Framework React*.
Base de datos	*MongoDB*.
Estado	Retirado de operación.
Reemplazo	Actualmente el *Join* se hace por medio de un archivo de *Excel* que se comparte a los viajeros.
Futuro	Se desea que la arquitectura incluya una solución que soporte el *Join* de los viajeros. Los viajeros deberán realizar el *Join* desde *Vaovapp* (ver Sección 3.4).

Tabla 3.1: Análisis Sistema *MyTrip*

- ***MyTrip - BackOffice***

  La Tabla 3.2 contiene el resumen del análisis realizado al sistema *MyTrip - BackOffice*.

MyTrip - BackOffice	
Descripción	Este sistema fue pensado para que los funcionarios de Vaova pudieran administrar los viajes, los usuarios de los viajeros, los países y destinos, las actividades que ofrece la compañía, las aerolíneas y los hoteles con los que se tienen acuerdos comerciales.
Problemas encontrados	El sistema fue desarrollado con malas prácticas de diseño y desarrollo; sólo permitía administrar actividades y hoteles; los demás servicios no funcionaban. No cumple funcionalmente con las necesidades para las cuales fue desarrollado y no cuenta con mecanismos de seguridad.
BackEnd	Lenguaje *Python*.
FrontEnd	Lenguaje *JavaScript* con *Framework Vue*.

Continúa en la siguiente página

Tabla 3.2 – Continuación de la página previa

MyTrip - BackOffice	
Base de datos	*MongoDB*.
Estado	Retirado de operación.
Reemplazo	Actualmente la administración de viajes, viajeros, destinos, actividades, aerolíneas y hoteles se realiza directamente en la base de datos.
Futuro	Se desea que la arquitectura incluya una solución que permita la administración de viajes, viajeros, destinos, actividades, aerolíneas y hoteles que permita facilitar la creación de Itinerarios y el *Join* de viajeros.

Tabla 3.2: Análisis Sistema *MyTrip - BackOffice*

3.3.2.2. Integraciones

- **TourHero**

  La Tabla 3.3 contiene el resumen del análisis realizado a la Integración *TourHero*.

Integración *TourHero*	
Descripción	Vaova tiene un acuerdo comercial que le permite utilizar la plataforma de *TourHero* como pasarela de ventas. Vaova carga los viajes y *TourHero* se encarga de la venta y recolección de datos. Esta integración permite a Vaova obtener la información de las ventas que se han realizado en la plataforma de *TourHero*.
Funcionamiento	*TourHero* no cuenta con un mecanismo para cargar los viajes de manera automática, estos deben ser ingresados manualmente por un funcionario de Vaova. *TourHero* sí cuenta con un proceso de sincronización inverso para informar las ventas que se han realizado, es decir, los sistemas de Vaova deben estar consultando activamente a los sistemas de *TourHero* para saber si se han realizado nuevas ventas y de esta manera poder sincronizar la información (*TourHero* no notifica, espera a ser consultado).
Continúa en la siguiente página	

Tabla 3.3 – Continuación de la página previa

Integración *TourHero*	
Componentes	La integración con *TourHero* cuenta con tres componentes de software desarrollados en *AWS Lambda* que se activan cada 15 minutos: • *Information Request*: Solicita a *TourHero* la información actualizada de las ventas realizadas para los viajes. *TourHero* responde con una URL que puede ser consultada después para conocer el estado de generación de un archivo de *Excel* que contiene la información solicitada. Esta URL se almacena en una cola de *AWS SQS* para que el siguiente componente la pueda consultar. • *Information Downloader*: Obtiene los mensajes de *AWS SQS* y consulta el estado de generación del archivo de *Excel*. Si el archivo está listo, este se puede descargar desde un *Bucket* de *AWS S3* que es propiedad de *TourHero*. El archivo se descarga y se carga a un *Bucket* de *AWS S3* que es propiedad de Vaova. • *Information Process*: Descarga los archivos de *Excel* que se encuentran en el *Bucket* de *AWS S3* de Vaova. Luego lee el contenido del archivo y carga la información a la base de datos. Finalmente, el archivo procesado es eliminado del *Bucket* de *AWS S3*.
Problemas encontrados	No tiene ningún problema. En la actualidad la integración funciona correctamente con algunas modificaciones que se realizaron por parte de Vaova.
BackEnd	*AWS Lambda* con *Python*.
Base de datos	*MongoDB*.
Estado	En funcionamiento.
Reemplazo	No aplica.
Futuro	Se desea que la arquitectura mantenga esta solución. La solución se debe refactorizar respecto a la base de datos.

Tabla 3.3: Análisis Integración *TourHero*

- **WeTravel**

  La Tabla 3.4 contiene el resumen del análisis realizado a la Integración *WeTravel*.

Integración *WeTravel*	
Descripción	Vaova tiene un acuerdo comercial que le permite utilizar la plataforma de *WeTravel* como pasarela de ventas. Vaova envía la información de los viajes a *WeTravel* en un archivo de *Excel* y ellos se encargan de cargarlos a su plataforma, de la venta y recolección de datos. Esta integración permite a Vaova obtener la información de las ventas que se han realizado en la plataforma de *WeTravel*.
Funcionamiento	*WeTravel* no cuenta con un mecanismo para cargar los viajes de manera automática, estos deben ser enviados en un archivo para que puedan ser cargados. *WeTravel* sí cuenta con un proceso de sincronización inverso para informar las ventas que se han realizado, es decir, los sistemas de Vaova deben estar consultando activamente a los sistemas de *WeTravel* para saber si se han realizado nuevas ventas y de esta manera poder sincronizar la información (*WeTravel* no notifica, espera a ser consultado).
Componentes	La integración con *WeTravel* cuenta con dos componentes de software desarrollados en *AWS Lambda* que se activan cada 15 minutos: • *Information Request*: Solicita a *WeTravel* la información actualizada de las ventas realizadas para los viajes. *WeTravel* responde con un archivo de *Excel* que contiene la información solicitada. El archivo se carga a un *Bucket* de *AWS S3* que es propiedad de Vaova. • *Information Process*: Descarga los archivos de *Excel* que se encuentran en el *Bucket* de *AWS S3* de Vaova. Luego lee el contenido del archivo y carga la información a la base de datos. Finalmente, el archivo procesado es eliminado del *Bucket* de *AWS S3*.
Problemas encontrados	No tiene ningún problema. En la actualidad la integración funciona correctamente con algunas modificaciones que se realizaron por parte de Vaova.
BackEnd	*AWS Lambda* con *Python*.
Base de datos	*MongoDB*.
Estado	En funcionamiento.
Reemplazo	No aplica.
Futuro	Se desea que la arquitectura mantenga esta solución. La solución se debe refactorizar respecto a la base de datos.

Tabla 3.4: Análisis Integración *WeTravel*

- **Viaxlab**

  La Tabla 3.5 contiene el resumen del análisis realizado a la Integración *Viaxlab*.

Integración *Viaxlab*	
Descripción	Vaova ofrece la posibilidad a los viajeros de utilizar la aplicación móvil de *Viaxlab*. Esta aplicación permite tener a disposición la información de viaje, itinerario, cronograma de actividades y ficha médica.
Funcionamiento	*Viaxlab* ofrece un mecanismo de integración para cargar la información de viajes y viajeros a su plataforma. Una vez la información ha sido cargada, los viajeros pueden descargar la aplicación, registrarse y utilizarla.
Componentes	La integración con *Viaxlab* cuenta con los siguientes componentes desarrollados en *Google Cloud Functions* que realizan el proceso de sincronización de la información cada que se registran cambios en la base de datos. Cada componente se encarga de sincronizar con *Viaxlab*: • *Trip Viaxlab*: Los viajes de Vaova. • *Activity Viaxlba*: Todas las actividades que Vaova puede ofrecer en los viajes. • *Itinerary Viaxlab*: Itinerarios de los viajes. En este contexto un itinerario es el conjunto de lugares que se visitan y actividades que se realizan durante el viaje. • *Group Viaxlab*: Grupo de los viajes. En este contexto un grupo son los paquetes que ofrece Vaova para los viajes. • *User Viaxlab*: Los viajeros asociados a un viaje. • *Modify Passenger*: Información personal actualizada de los viajeros. • *Passenger Activies Viaxlab*: Itinerarios de los viajeros.
Problemas encontrados	Vaova no logró operar la integración de manera estable en ningún momento. Actualmente la integración se encuentra apagada, pero se siguen utilizando los servicios de este proveedor. La información se ingresa manualmente por medio de la consola web de *Viaxlab*.
BackEnd	*Google Cloud Functions* con *Python*.
Base de datos	*MongoDB*.
Estado	Integración automática retirada. La información se sincroniza de manera manual.
Reemplazo	Proceso manual realizado por funcionario de Vaova.
Futuro	Se desea eliminar esta integración por su mal funcionamiento y los altos costos asociados de utilizar los servicios de *Viaxlab*. Se desea que la arquitectura incluya una solución que reemplace los servicios de esta integración.

Tabla 3.5: Análisis Integración *Viaxlab*

3.3.2.3. Modelo de datos

- **MongoDB**: El modelo de datos que maneja actualmente Vaova se encuentra en una base de datos *MongoDB*. La Tabla 3.6 muestra para cada colección el nombre, la descripción, el número de documentos, si estaba en uso y la referencia a la figura con el esquema de escritura de la colección.

Colección	Descripción	Docs.	En Uso	Esquema
configs	Información de países y aeropuertos. Cada país contiene una lista de ciudades, una lista de URLs que referencian imágenes y campos de tipo booleano. Por su parte, cada aeropuerto contiene su nombre y la ciudad en la que está ubicado.	6	No	Figura 3.16
flights_info	Vuelos asociados a un viaje. Cada vuelo contiene rutas y cada ruta contiene información de los lugares de origen y destino, la aerolínea, entre otros datos.	87	Sí	Figura 3.17
hotels_info	Hoteles asociados a un viaje. Cada hotel contiene las habitaciones disponibles, la ciudad, fechas de inicio y fin, entre otros datos.	130	Sí	Figura 3.18
join_history	Se diseñó para almacenar el histórico de *Joins* realizado por un viajero. La colección sólo contenía 2 registros, lo que indica que en realidad nunca se llegó a utilizar.	2	No	Figura 3.19
join_users	Información del *Join* realizado por los viajeros. Se detallan los hoteles, vuelos, roomie, entre otros.	4297	Sí	Figura 3.20
join_users_history	Relaciona las invitaciones entre usuarios para ser compañeros de cuarto. Los documentos de la colección sólo contenían datos de pruebas de desarrollo.	742	No	Figura 3.21
report_info	Guarda los informes de la información que se sincroniza desde la plataforma de *WeTravel*.	98 769	Sí	Figura 3.22

Continúa en la siguiente página

Tabla 3.6 – Continuación de la página previa

Colección	Descripción	Docs.	En Uso	Esquema
trips_info	Información de los viajes. Cada viaje contiene datos generales como nombre y descripción. Tiene listas con actividades, complementos y paquetes asociados al viaje.	68	Sí	Figura 3.23
trips_viaxlab	Viajes que se han sincronizado en la plataforma de *Viaxlab* por medio de la integración.	29	No	Figura 3.24
users	Información de los viajeros. Cada viajero queda asociado a un viaje específico y contiene actividades, complementos y reserva de habitaciones. Entre los datos personales del viajero se encuentran sus nombres, apellidos, fecha de nacimiento, dirección, número de teléfono, dirección de correo electrónico, país de origen, información de pasaporte, entre otros.	4895	Sí	Figura 3.25
users_history	Contiene la misma información que la colección de *users*, pero incluye información de pagos exitosos y pagos pendientes.	1 116 438	Sí	Figura 3.26
users_overrides	Esta colección no contenía ningún documento.	0	No	No definido
users_viaxlab	Viajeros que se sincronizaron en la plataforma de *Viaxlab* por medio de la integración.	2799	No	Figura 3.27

Tabla 3.6: Análisis Base de Datos MongoDB

La notación utilizada para realizar los diagramas de las colecciones de la base es la siguiente:

- En la parte izquierda se muestra el objeto JSON principal.
- Los atributos simples se representan en color púrpura dentro del JSON principal.
- El valor de cada atributo simple representa su tipo de dato.
- Los objetos anidados se representan como una asociación en color púrpura.
- Las listas se representan como una asociación de color naranja.

A continuación se muestran los esquemas de escritura de las colecciones de la base de datos *MongDB*.

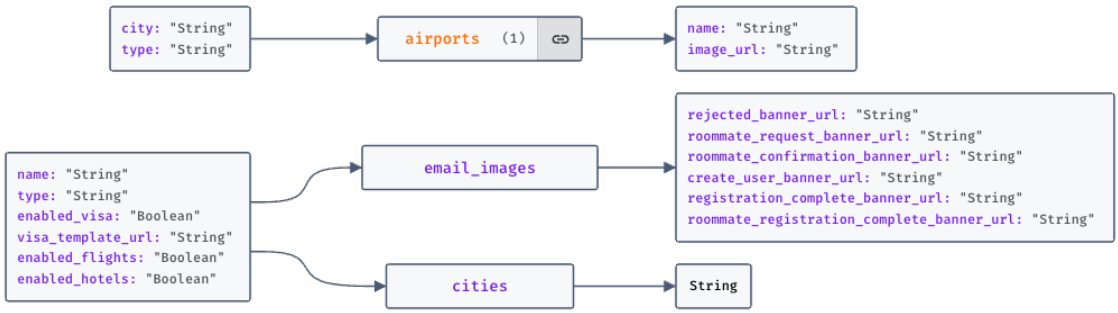


Figura 3.16: MongoDB - Colección *configs*

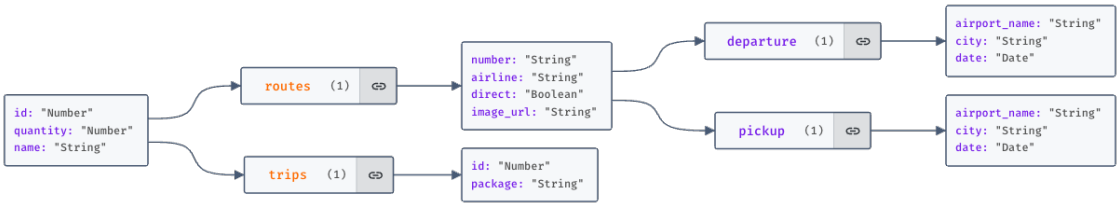


Figura 3.17: MongoDB - Colección *flights_info*

Figura 3.18: MongoDB - Colección *hotels_info*

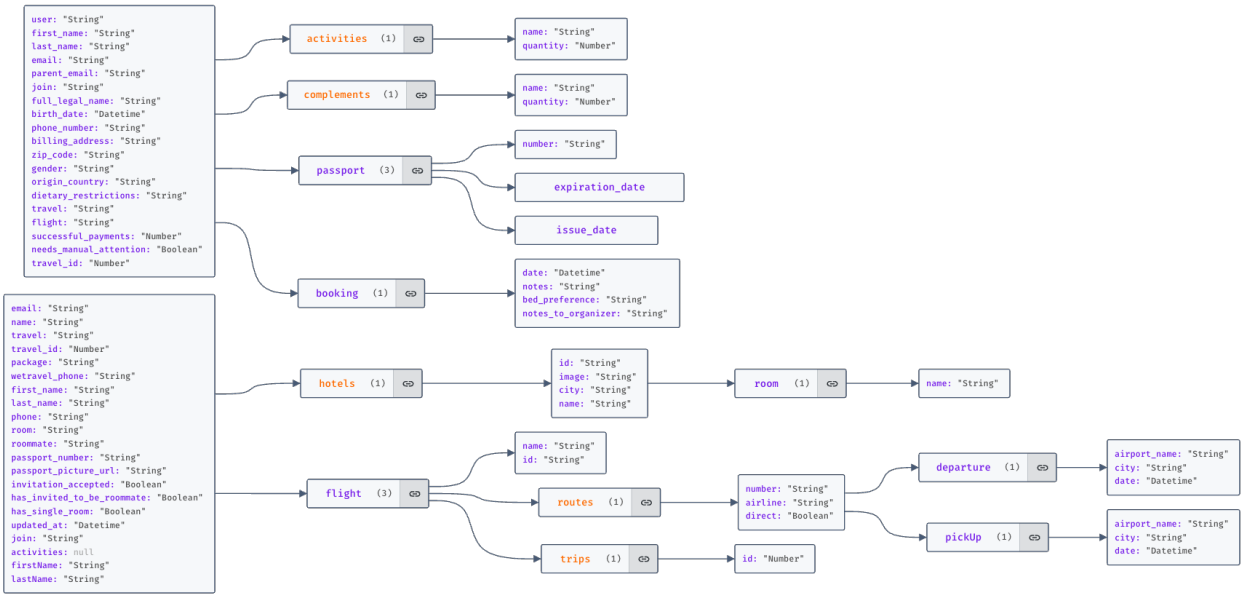


Figura 3.19: MongoDB - Colección *join_history*

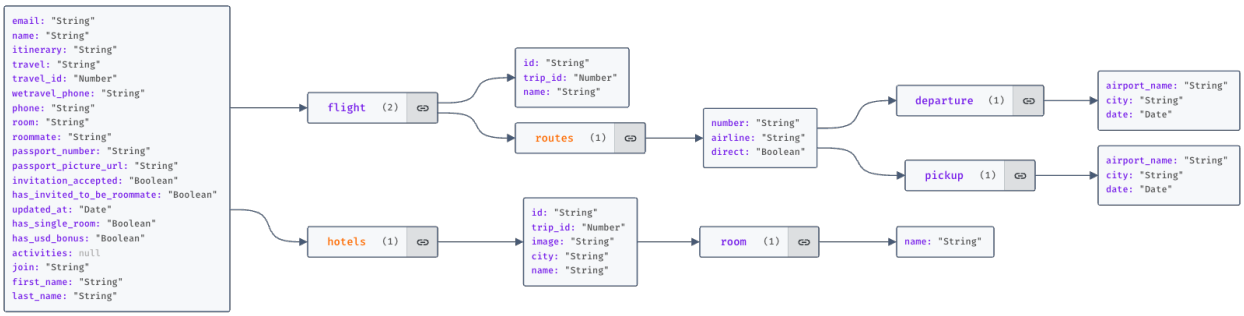


Figura 3.20: MongoDB - Colección *join_users*

Figura 3.21: MongoDB - Colección *join_users_history*

Figura 3.22: MongoDB - Colección *report_info*

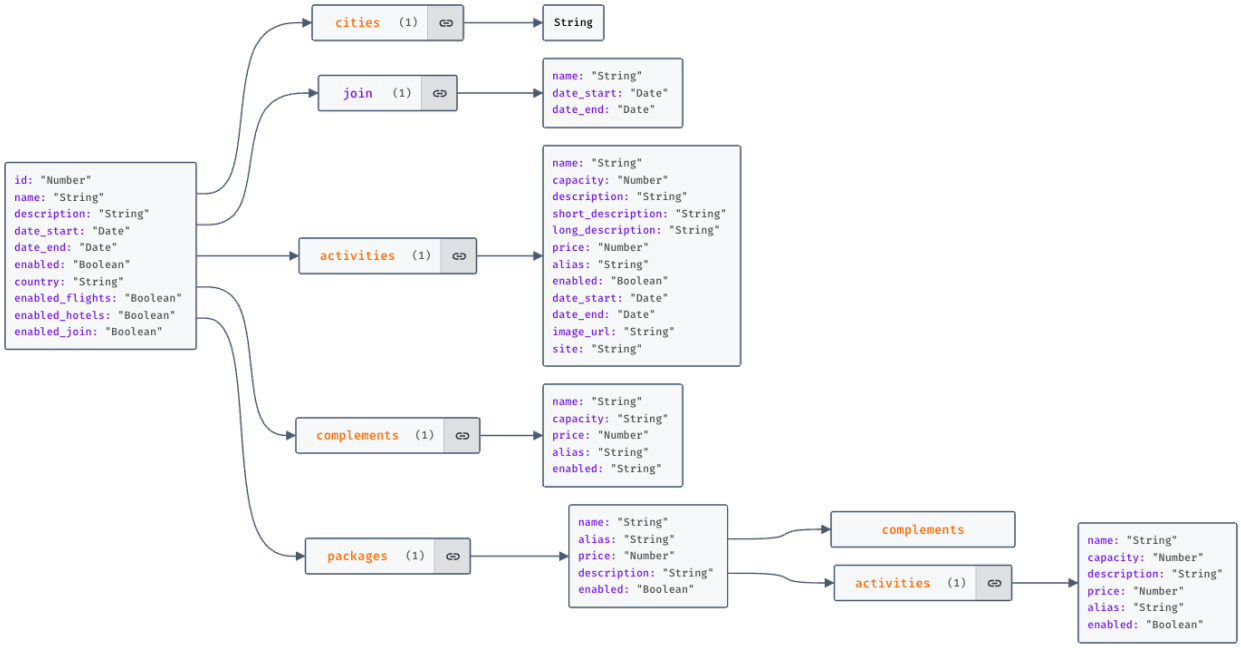


Figura 3.23: MongoDB - Colección *trips_info*

Figura 3.24: MongoDB - Colección *trips_viaxlab*

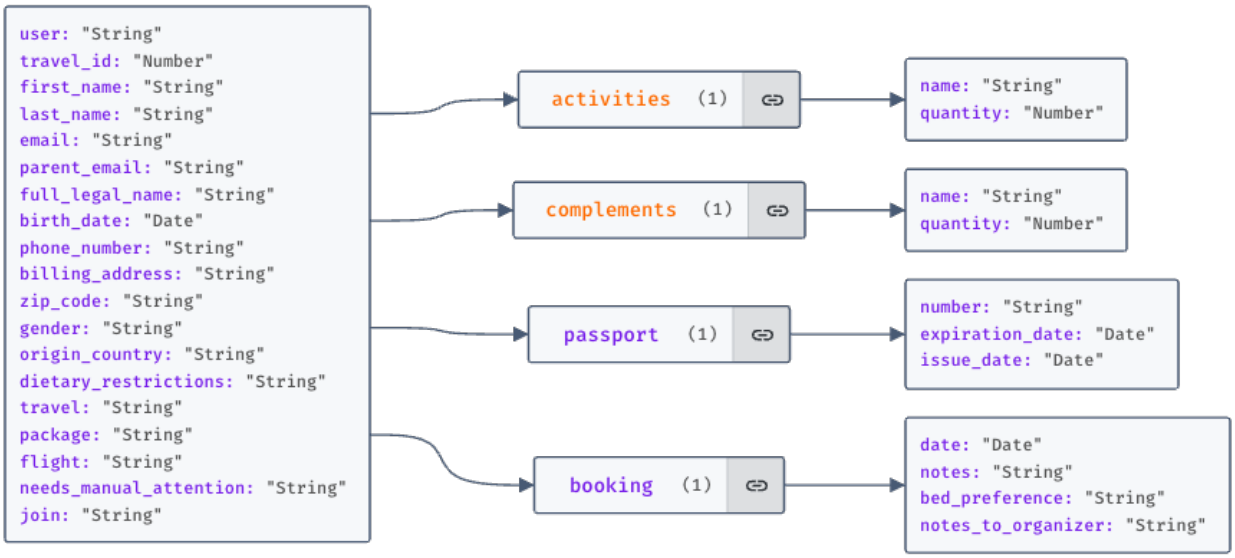


Figura 3.25: MongoDB - Colección *users*

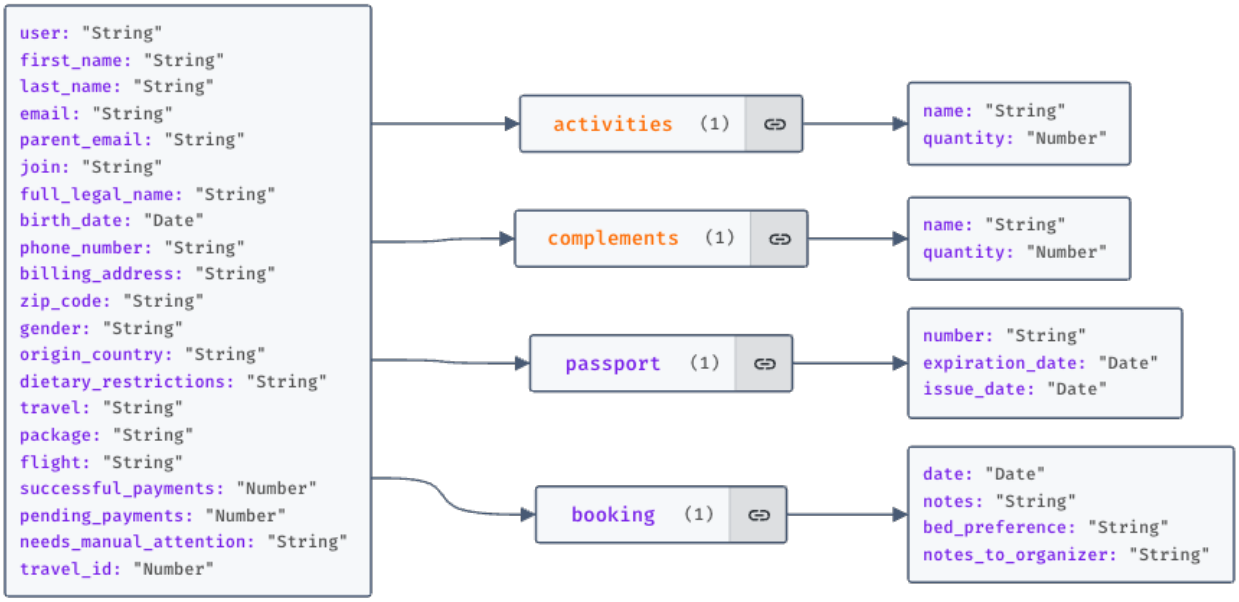


Figura 3.26: MongoDB - Colección *users_history*

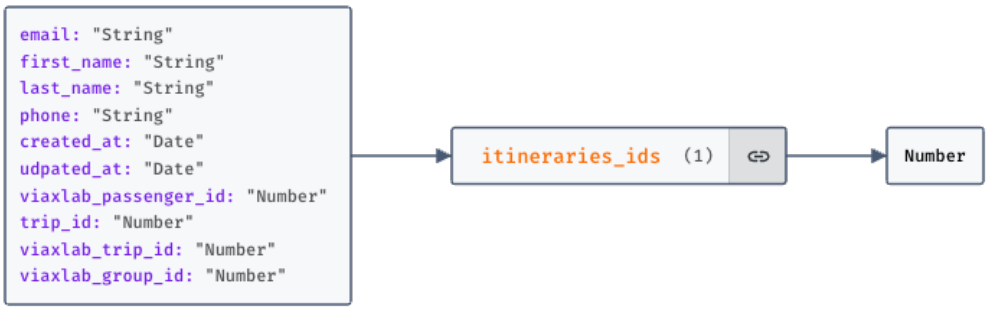


Figura 3.27: MongoDB - Colección *users_viaxlab*

NOTA: Los diagramas de las colecciones de *MongoDB* se pueden en la Sección 6.4 de Anexos.

3.3.2.4. Resumen Estado Actual de Vaova

La Tabla 3.7 resume el estado actual de los sistemas propios, integraciones y bases de datos de Vaova.

Sistema	Estado	Futuro	Observaciones
MyTrip	Retirado.	Rediseñar.	Las funcionalidades están contempladas para el sistema web de Itinerarios y Vaovapp.
MyTrip - BackOffice	Retirado.	Rediseñar.	Las funcionalidades están contempladas para el sistema web de Itinerarios y Vaovapp.
Integración *TourHero*	Operando.	Retener.	La integración funciona correctamente. Se aceptan cambios para la nueva arquitectura.
Integración *WeTravel*	Operando.	Retener.	La integración funciona correctamente. Se aceptan cambios para la nueva arquitectura.
Integración *Viaxlab*	Retirado.	Reemplazar.	La integración no se puedo estabilizar y se apagó definitivamente. Los servicios de Viaxlab se usan de manera manual por medio de su consola web. La decisión de Vaova es terminar el vínculo comercial con Viaxlab y desarrollar estos servicios por su cuenta.
MongoDB	Operando.	Rediseñar.	Si es posible, utilizar un motor de base de datos relacional.

Tabla 3.7: Evaluación Estado de Vaova - Resumen

3.4. Historias Épicas

En las subsecciones 3.3.2.1 y 3.3.2.4 se mencionaron dos sistemas que no fueron definidos: sistema web de Itinerarios y la aplicación móvil Vaovapp. Estos sistemas son las soluciones que Vaova desea implementar para reemplazar los sistemas actuales. También, existen algunos tableros de información que se manejan actualmente por medio de la base de datos de *MongoDB* que se desean conservar. En esta sección, antes de definir las historias épicas, se hará la descripción de estos sistemas y cuáles serán sus responsabilidades.

> *Nota informativa:* el diseño del sistema web de Itinerarios y la aplicación móvil Vaovapp están fuera del alcance del proyecto de grado. El diseño de la arquitectura contemplará los servicios necesarios, pero no se diseñará la arquitectura para estas aplicaciones.

3.4.1. Sistema Web de Itinerarios

Este sistema será el reemplazo de *MyTrip* y *MyTrip - BackOffice*. Será el encargado de:

- A los funcionarios de Vaova, les permitirá la creación de usuarios que corresponden a los Promotores.
- A los funcionarios de Vaova, les permitirá administrar destinos, hoteles, medios de transporte, actividades entre otros recursos.
- A los promotores, les permitirá crear los Itinerarios con la información necesaria como fechas, destinos y actividades.
- A los funcionarios de Vaova, les permitirá revisar Itinerarios y aprobarlos o rechazarlos. Si el Itinerario es aprobado, puede generar el insumo para cargar el viaje a la plataforma externa de *TourHero* o *WeTravel*.
- A los funcionarios de Vaova, les permitirá añadir los recursos de viajes aéreos y hoteles que son necesarios para iniciar y cerrar el Join.

3.4.2. Vaovapp

Esta aplicación móvil será ofrecida a los viajeros, y será el reemplazo de *MyTrip* y *Viaxlab*. Permitirá realizar los siguientes procesos a los viajeros:

- Ingresar y revisar sus viajes programados.
- Realizar el *Join* de su viaje siguiente: seleccionar acomodación, compañero de cuarto y viajes.
- Durante los viajes, cronograma de actividades y recordatorios.

En una primera instancia, Vaovapp no ofrecerá la posibilidad de validar si un usuario está autorizado para ingresar a una actividad. Este servicio se prevé para versiones futuras, por lo que no será tenido en cuenta en el diseño de la arquitectura.

3.4.3. Tableros de Información

Vaova monitorea por medio de tableros informativos en *MongDB* los viajes que están próximos a ejecutarse. Estos tableros permiten conocer de manera muy sencilla el nivel de ocupación de recursos para cada viaje y permiten tomar decisiones de manera rápida a los operarios de la compañía. Las siguientes son algunas de las métricas que actualmente se miden de manera constante en dichos tableros:

- Cantidad de paquetes vendidos de un viaje
- Ranking de destinos en un viaje
- Ranking de actividades de un viaje
- Ranking de complementos vendidos en un viaje
- Cantidad de habitaciones ocupadas de un viaje
- Cantidad de habitaciones disponibles de un viaje
- Cantidad de vuelos ocupados en un viaje
- Cantidad de vuelos disponibles en un viaje
- Ranking de aerolíneas
- Ranking de hoteles
- Entre otros reportes

En el diseño de la nueva arquitectura es necesario generar una solución que permita reemplazar estos tableros si el motor de base de datos no fuera *MongoDB*.

3.4.4. Definición de Historias Épicas

Las historias épicas están divididas para Sistema Web de Itinerarios, Vaovapp y Tableros de Información. El formato que se utilizó para definir las historia épicas fue *Gherkin*, que permite se basa en características con escenarios.

> *Nota informativa*: Sólo se mostrarán 5 de las 39 historias épicas en este documento. Las demás historias épicas pueden ser encontradas en la Sección 6.5 de Anexos.

3.4.4.1. Sistema Web de Itinerarios

Se muestran las historias épicas 4 y 11 de las 20 que fueron definidas para el subsistema de Itinerarios. La Tabla 3.8 contiene todos los títulos de las historias épicas del subsistema de Itinerarios.

```
Feature: Itinerario 4 - Crear Itinerario
  Los usuarios (Promotores y Asesores Comerciales) pueden crear
     itinerarios en la aplicación web de Itinerarios.

  Scenario: Como usuario de la aplicación web de Itinerarios puedo crear
     un nuevo itinerario sin finalizar seleccionando la opción de Crear
     Itinerario.
    Given Un usuario con rol de viajero
    When Solicita crear un itinerario
    And Ingresa la información del itinerario
    And Elige la opción de guardar sin finalizar
    And Envía la petición
    Then El itinerario se guarda en el sistema de itinerarios con estado
       sin finalizar

  Scenario: Como usuario de la aplicación web de Itinerarios puedo crear
     un nuevo itinerario finalizado seleccionando la opción de Crear
     Itinerario.
    Given Un usuario con rol de viajero
    When Solicita crear un itinerario
    And Ingresa la información del itinerario
    And Elige la opción de guardar y finalizar
    And Envía la petición
    Then El itinerario se guarda en el sistema de itinerarios con estado
       finalizado
```

```
Feature: Itinerarios 11 - Fechas de Viaje
  Los itinerarios se crean con varios meses de anticipación, esto se debe
     a que operaciones debe validar, aceptar y planear el viajeo.
  El tiempo mínimo para aceptar un itinerario es de 4 meses a partir de
     la fecha de creación. Aunque hay excepciones a esta regla.
  Esta regla no es una restricción para crear los itinerario, en todo
     caso el itinerario podría ser rechazado por el asesor comercial si
     no se cumple el tiempo minimo.
  Existe una regla de negocio que indica el número de destinos permitidos
     a partir del número de noches:
      - Entre 1-4 noches se permite un único destino dentro del país
      - Entre 5-7 noches se permiten hasta dos destinos dentro del país
      - Entre 7-9 noches se permiten hasta tres destinos dentro del país
      - Entre 10-15 noches se permiten hasta cuatro destinos dentro del
         país
```

```
  Con lo anterior se define que los viajes deben tener una duración de
     entre 1 a 15 noches.

  Scenario: Como viajero debo poder elegir las fechas de inicio y fin
     globales de mi viaje, cuando elija esto se deberá calcular el total
     de noches de mi viaje. La fecha de fin debe ser al menos un día
     después de la fecha de inicio.
    Given Un usuario creando un itinerario
    When Elige las fechas de inicio y fin globales del viaje
    Then El sistema debe calcular automáticamente la cantidad de noches
       que durará el viaje

  Scenario: Como viajero requiero que el sistema me indique de forma
     automática la cantidad de destinos que puedo visitar dentro del
     país, teniendo en cuenta las fechas globales y noches de mi viaje.
    Given Un usuario creando un itinerario
    When Elige las fechas de inicio y fin globales del viaje
    Then El sistema debe calcular automáticamente la cantidad de destinos
       disponibles que el viaje puede tener
```

3.4.4.2. Vaovapp

Se muestran las historias épicas 6 y 11 de las 18 que fueron definidas para el subsistema de Vaovapp. La Tabla 3.9 contiene todos los títulos de las historias épicas del subsistema de Vaovapp.

```
Feature: Vaovapp 6 - Upgrade Single Room
  Los viajes de Vaova se programan para que los viajeros compartan
     habitación.
  Los viajeros que no quieran compartir habitación pueden comprar una
     mejora para tener una habitación sin acompañante.
  Desde TourHero y WeTravel, al momento de hacer la compra del paquete,
     los viajeros pueden elegir la mejora de Single Room para no
     compartir habitación.

  Scenario: Como viajero, requiero que al comprar el upgrade a Single
     Room sin importar el paquete al que pertenezca, no debo tener la
     opción de elegir Roomie.
    Given Un usuario haciendo Join
    When Adquiere la mejora de Single Room
    Then El sistema no le permite elegir Roomie

  Scenario: Como viajero, requiero que al comprar el upgrade a Single
     Room sin importar el paquete al que pertenezca, no seré invitado por
     ninguna persona del viaje para ser su roomie.
    Given Un usuario haciendo Join
```

```
    When Adquiere la mejora de Single Room
    Then El sistema no le permite a otros viajeros del mismo viaje
       elegirlo como Roomie

  Scenario: Como viajero, requiero que al comprar el upgrade a Single
     Room sin importar el paquete al que pertenezca, no puedo elegir tipo
     de acomodación.
    Given Un usuario haciendo Join
    When Adquiere la mejora de Single Room
    Then El sistema no le permite elegir tipo de acomodación, ya que el
       tipo de acomodación será Single Room
```

```
Feature: Vaovapp 11 - Invitación

  Scenario: Como viajero necesito saber cuando he invitado a un Roomie y
     esa persona no ha aceptado mi invitación
    Given Un usuario
    When Ha enviado una invitación de Roomie a otro viajero
    Then El sistema automáticamente muestra el estado de la invitación

  Scenario: Como viajero necesito saber cuando me han invitado a ser
     Roomie y no he aceptado la invitación.
    Given Un usuario
    When Ha recibido una invitación a ser Roomie de otro viajero
    Then El sistema automáticamente muestra el estado de la invitación

  Scenario: Como viajero que ha enviado invitación a otro para que sea mi
     Roomie, no puedo enviar invitación a otro viajero hasta cancelar la
     invitación pendiente.
    Given Un usuario
    When Ha enviado una invitación de Roomie a otro viajero
    And La invitación está pendiente de ser aceptada
    Then El sistema no le permite enviar invitación de Roomie a ningún
       otro viajero

   Scenario: Como viajero que he recibido una invitación de otro para ser
      su Roomie, no puedo recibir nuevas invitaciones hasta cancelar la
      invitación pendiente.
    Given Un usuario
    When Ha recibido una invitación de Roomie de otro viajero
    And La invitación está pendiente de ser aceptada
    Then El sistema no le permite recibir invitaciones de Roomie de
       ningún otro viajero
```

3.4.4.3. Tableros de Información

Para el subsistema de Tableros de Información sólo se definió una historia épica.

```
Feature: Tableros 1 - Tableros de Información
  Vaova necesita monitorear varios aspectos relacionados con los vuelos,
     ya que esta información le permite tomar decisiones sobre los mismos.
  La información que se monitorea es la siguiente:
      - Cantidad de paquetes vendidos de un viaje
      - Ranking de destinos en un viaje
      - Ranking de actividades de un viaje
      - Ranking de complementos vendidos en un viaje
      - Cantidad de habitaciones ocupadas de un viaje
      - Cantidad de habitaciones disponibles de un viaje
      - Cantidad de vuelos ocupados en un viaje
      - Cantidad de vuelos disponibles en un viaje
      - Ranking de aerolíneas
      - Ranking de hoteles
      - Entre otros reportes

  Scenario: Como promotor comercial y usuario de operaciones, requiero de
     tableros informativos que me permitan conocer de manera rápida el
     estado de las ventas de los viajes.
    Given Un viaje cargado en la plataforma de TourHero/WeTravel
    When Se realizan ventas de los paquetes
    Then La información de las ventas alimenta y actualiza los tableros
       de información de los viajes

  Scenario: Como promotor comercial y usuario de operaciones, requiero de
     tableros informativos que me permitan conocer de manera rápida el
     estado del Join para viajes cercanos.
    Given Un viaje con el Join habilitado
    When Los viajeros completan su Join
    Then La información del Join de los viajeros alimenta y actualiza los
       tableros de información del Join de los viajes
```

3.4.4.4. Resumen Historias Épicas

Las Tablas 3.8 y 3.9 contienen el resumen de todas las historias épicas definidas para los subsistemas de Itinerarios y Vaoapp. No se incluye tabla para el subsistema de Tableros de Información porque sólo contiene una historia épica.

Número	Título
1	Itinerarios - Login de Usuario
2	Itinerarios - Recuperar Contraseña
3	Itinerarios - Crear Usuario
4	Itinerario - Crear Itinerario
5	Itinerario - Editar Itinerario
6	Itinerarios - Listar de Itinerarios
7	Itinerarios - Filtrar Itinerario
8	Itinerarios - Eliminar Itinerario
9	Itinerarios - Número Estimado de Viajeros
10	Itinerarios - País de Destino
11	Itinerarios - Fechas de Viaje
12	Itinerarios - Cantidad de Destinos Disponibles
13	Itinerarios - Paso a Paso
14	Itinerarios - Listado de Destinos
15	Itinerarios - Búsqueda de un Destino
16	Itinerarios - Detalle Destino
17	Itinerarios - Calendario de Actividades
18	Itinerarios - Listado de Actividades
19	Itinerarios - Detalle Destino
20	Itinerarios - Resumen

Tabla 3.8: Historias épicas de Itinerarios

Número	Título
1	Vaovapp - Login de Usuario
2	Vaovapp - Recuperar Contraseña
3	Vaovapp - Información de Usuario
4	Vaovapp - Editar Información del Usuario
5	Vaovapp - MyTrips
6	Vaovapp - Upgrade Single Room
7	Vaovapp - Tipo de Acomodación
8	Vaovapp - Vuelos
9	Vaovapp - Información de Vuelos
10	Vaovapp - Invitación de Roomie
11	Vaovapp - Invitación
Continúa en la siguiente página	

Tabla 3.9 – Continuación de la página previa

Número	Título
12	Vaovapp - Cancelar Invitación
13	Vaovapp - Rechazar Invitación
14	Vaovapp - Última Mujer
15	Vaovapp - Invitación Aceptada
16	Vaovapp - Join
17	Vaovapp - Inventario de Habitaciones
18	Vaovapp - Inventario de Vuelos

Tabla 3.9: Historias épicas de Vaovapp

3.5. Diseño de la Arquitectura de Software

En esta sección [1] se describen los pasos que se realizaron para realizar el diseño de la arquitectura. Siguiendo la metodología de DDD, se dividieron las tareas en diseño estratégico y diseño táctico. En la fase de diseño estratégico se realizaron tareas relacionadas con el negocio de Vaova (por eso fueron importantes las tareas realizadas en las subsecciones 3.1, 3.2 y 3.3); en la fase de diseño táctico se realizaron todas las tareas técnicas para proponer el diseño de la arquitectura de software.

3.5.1. Diseño Estratégico

En la fase de diseño estratégico se realizó el análisis detallado de negocio, las necesidades y los problemas que se buscaban resolver con el diseño de la arquitectura de software. En esta etapa se definieron el dominio del problema, los subdominios, los contextos delimitados, los lenguajes ubicuos de cada contexto. En la fase de diseño estratégico también se deberían definir los *drivers arquitectónicos*, pero se decidió por dedicarle una sección completa a dicho tema.

3.5.1.1. Dominio

En la etapa de definición del dominio se realizó la siguiente actividad: comprender enteramente el negocio y los procesos operativos de la compañía, para diseñar los sistemas que fueran necesarios. El dominio del negocio de Vaova incluye tres etapas:

- **Administración:** Se manejan los recursos como destinos, agencias de transporte terrestre, aerolíneas, hoteles y las actividades que Vaova diseña para ofrecer en los viajes.

[1] Sólo se describen los apartados más importantes y significativos del documento de arquitectura de software. El documento de arquitectura de software se puede encontrar en la Sección 6.6.

- **Planeación:** En esta fase se diseñan los itinerarios de viaje, se aprueban o rechazan los itinerarios, se administran los viajes, se sincronizan las ventas y datos de viajeros, se asignan los recursos a los viajes programados y se realiza el Join por parte de los viajeros.

- **Ejecución:** Se realizan todas las actividades de los viajes en las fechas establecidas.

Por fuera del dominio de negocio está la venta directa de los viajes, el recaudo de los paquetes vendidos y la recolección de la información personal de los viajeros. De estos procesos se encargan *TourHero* y *WeTravel*, que son las compañías con las que se tienen acuerdos para realizar estas actividades. La información que estás compañías recolectan llega a Vaova por medio de integraciones.

3.5.1.2. Subdominios

Con el dominio definido, que a su vez delimita el contexto de negocio, se definieron los subdominios presentados en la Figura 3.28. Estos subdominios fueron definidos a partir del análisis realizado en la Sección 3.2.

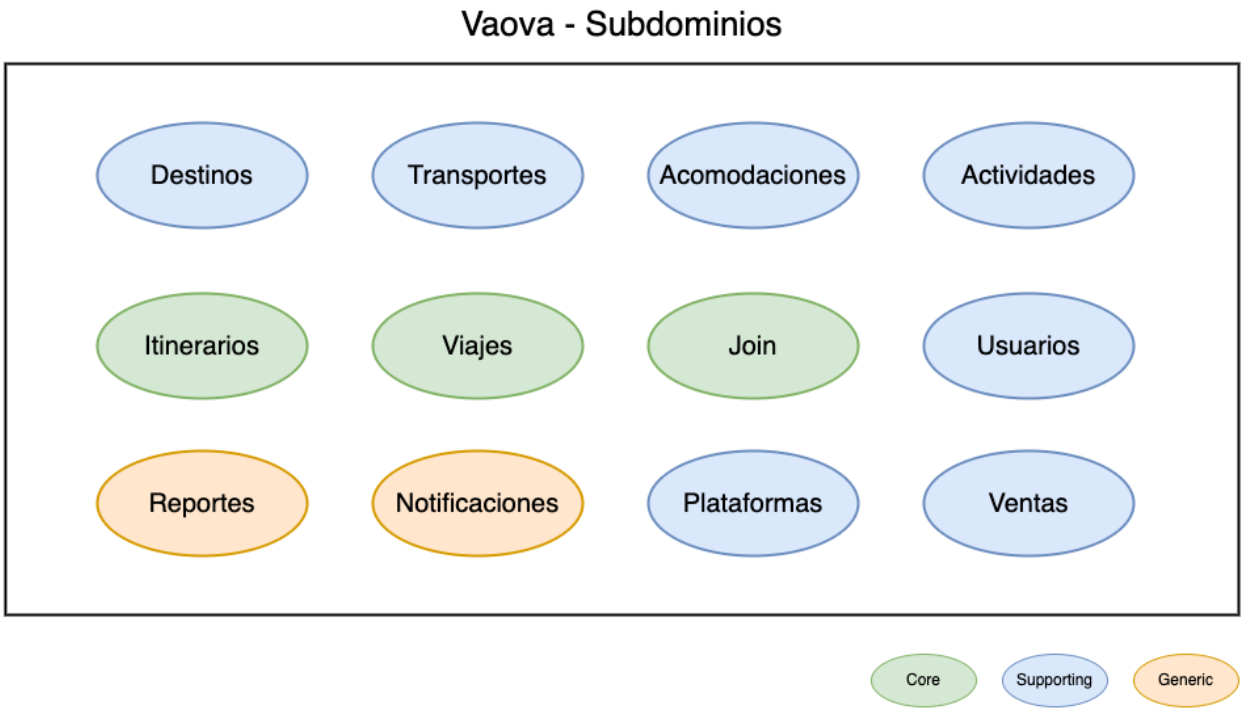


Figura 3.28: Subdominios del negocio de Vaova

En la Sección 3.5.1.3 se define de manera detallada el alcance y los limites de cada subdominio.

3.5.1.3. Contextos Delimitados

En esta sección se definen los contextos delimitados de cada subdominio. El contexto delimitado define de manera clara las operaciones que el subdominio puede realizar, los atributos más importantes de las entidades, y las relaciones y dependencias con otros subdominios.

Para cada subdominio se definió un gráfico de cajas y flechas, que es una combinación entre diagrama de flujo y diagramas BPMN. La nomenclatura es la siguiente:

- **Caja roja:** Representa el subdominio al que se le está definiendo su contexto delimitado.
- **Cajas verdes:** Son atributos del subdominio.
- **Cajas azules:** Son actividades que se realizan dentro del subdominio.
- **Cajas púrpuras:** Son dependencias con otros subdominios.
- **Flechas:** Son acciones que se realizan entre atributos o actividades del subdominio.

También se le asigna la clasificación estratégica a cada subdominio. *DDD* define tres categorías para clasificar a los subdominios (Evans, 2015):

- **Subdominios Core:** Son los subdominios fundamentales para el dominio principal. Concentran la mayor parte de la lógica de negocio y no se puede delegar su desarrollo a un tercero o buscar una solución en el mercado. En el caso de Vaova, los subdominios de Itinerarios, Viajes y Join se clasificarion como Core.
- **Subdominios de Soporte:** Son subdominios auxiliares para los subdominios core. No son parte del negocio principal, pero sin estas el negocio no podría funcionar. Al ser de soporte, es posible delegar su desarrollo a un tercero o buscar una solución que ya se encuentre desarrollada en el mercado. Los subdominios de Destinos, Transportes, Acomodaciones, Actividades, Usuarios, Plataformas y Ventas se clasificaron como Soporte.
- **Subdominios Genéricos:** Son subdominios que no son específicas del negocio, por lo tanto se pueden emplear soluciones genéricas ya desarrolladas y probadas. En esta categoría se definieron los subdominios de Reportes y Notificaciones.

A continuación, se definen los contextos delimitados que pertenecen a la clasificación estratégica de Core.

- **Itinerarios:** Se muestra en la Tabla 3.10.

Contexto Delimitado de Itinerarios	
Subdominio	Itinerarios.
Clasificación Estratégica	Core.
Descripción	En este subdominio se diseñan los viajes por parte de los Promotores. Un Itinerario es un viaje potencial y define fechas, destinos y actividades que se realizarán en caso de que el Itinerario sea aprobado. Una vez el Promotor termina de elaborar el Itinerario, este pasa a revisión por parte de un Asesor Comercial que debe verificar si el viaje es idóneo para ser llevado a cabo. En la revisión del Itinerario, el Asesor Comercial puede aceptar, modificar y aceptar, o rechazar el Itinerario.
Diagrama de Contexto Delimitado	Figura 3.29.

Tabla 3.10: Contexto Delimitado de Itinerarios

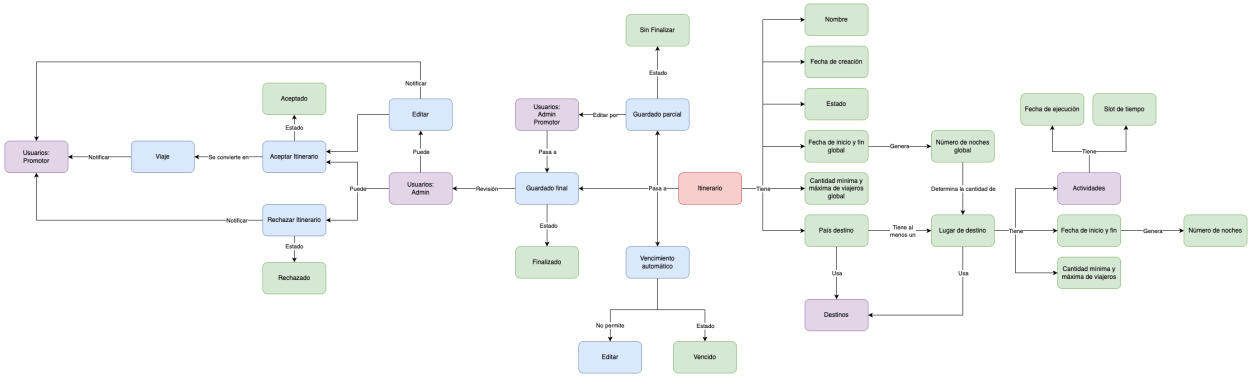


Figura 3.29: Contexto Delimitado de Itinerarios

- **Viajes:** Se muestra en la Tabla 3.11.

Contexto Delimitado de Viajes	
Subdominio	Viajes.
Clasificación Estratégica	Core.
Descripción	Una vez un Itinerario de un Promotor es revisado y aprobado por el Asesor Comercial, este pasa a ser un viaje en planeación. En esta etapa, se definen los paquetes del viaje (un paquete es un conjunto de destinos y actividades dentro del viaje) y se debe definir la plataforma en la que el viaje será cargado para que pueda ser vendido a los viajeros (se carga a *TourHero* o *WeTravel*).
Diagrama de Contexto Delimitado	Figura 3.30.

Tabla 3.11: Contexto Delimitado de Viajes

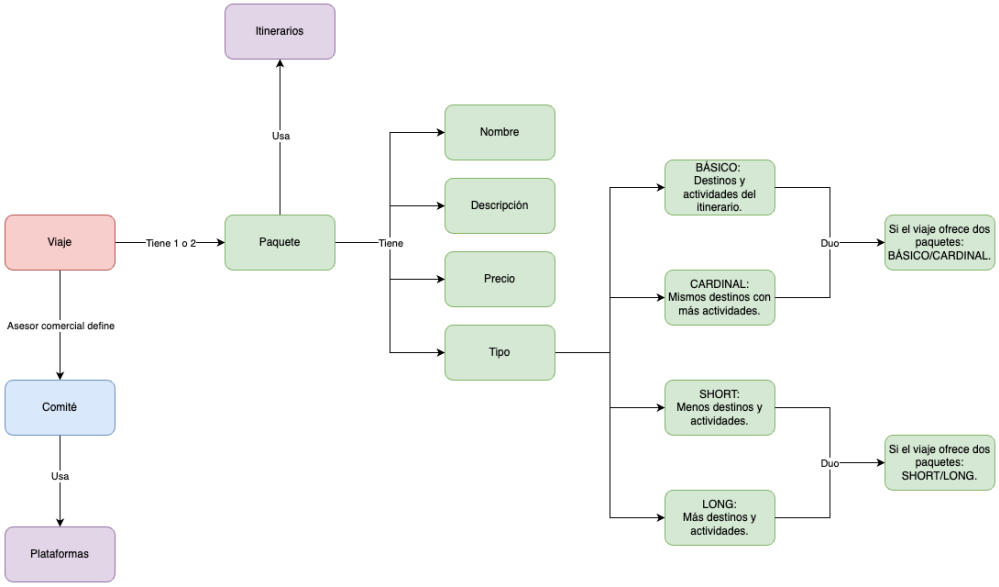


Figura 3.30: Contexto Delimitado de Viajes

- **Join:** Se muestra en la Tabla 3.12.

Contexto Delimitado de Join	
Subdominio	Viajes.
Clasificación Estratégica	Core.
Descripción	El Join es un proceso operativo muy importante para Vaova. Es el momento en el cual los viajeros seleccionan los vuelos para desplazarse dentro del país de destino, los hoteles en los que se hospedarán y su compañero de cuarto. Para iniciar el Join, se deben configurar los vuelos y los hoteles con sus respectivas habitaciones para que los viajeros las puedan seleccionar. Este detalle es de vital importancia, porque el viaje se sigue vendiendo en la plataforma durante el Join y estos recursos se pueden agotar.
Diagrama de Contexto Delimitado	Figura 3.31.

Tabla 3.12: Contexto Delimitado de Join

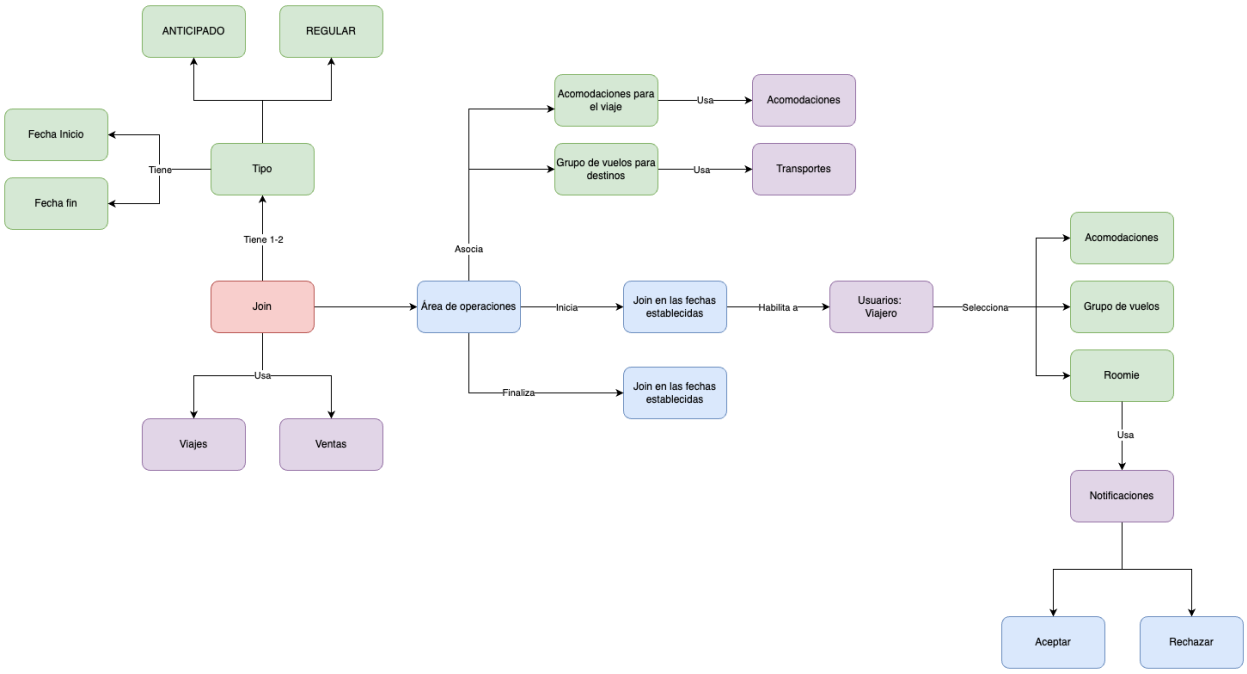


Figura 3.31: Contexto Delimitado de Join

Ahora, se presentan los contextos delimitados que corresponden a la clasificación estratégica de Soporte.

- **Destinos:**: Se muestra en la Tabla 3.13.

Contexto Delimitado de Destinos	
Subdominio	Viajes.
Clasificación Estratégica	Soporte.
Descripción	Este subdominio presenta los lugares a los cuales se pueden realizar los viajes. Consiste en los países y los pueblos, ciudades y regiones dentro de estos en los cuales Vaova opera.
Diagrama de Contexto Delimitado	Figura 3.32.

Tabla 3.13: Contexto Delimitado de Destinos

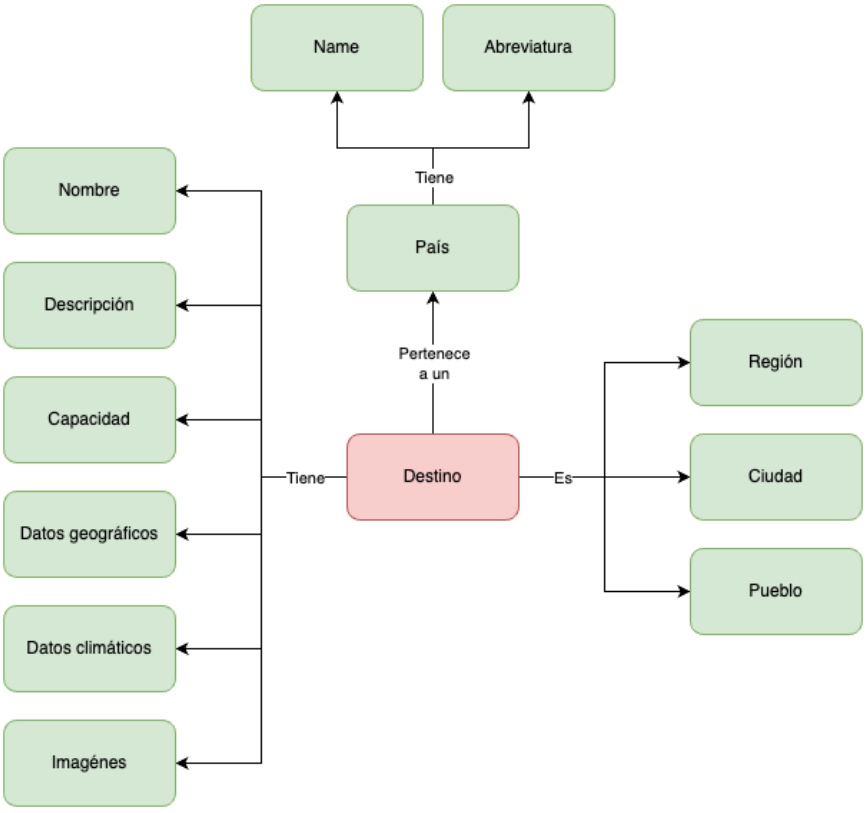


Figura 3.32: Contexto Delimitado de Destinos

- **Transportes:** Se muestra en la Tabla 3.14.

Contexto Delimitado de Transportes	
Subdominio	Viajes.
Clasificación Estratégica	Soporte.
Descripción	Vaova ofrece transporte terrestre y aéreo a los viajeros en los destinos de los viajes. Por lo tanto, estos pueden seleccionar las aerolíneas y los vuelos en los que quieren desplazarse dentro del país de destino (el transporte internacional, del país de origen al país de destino es responsabilidad de los viajeros). El transporte terrestre también se ofrece a los viajeros, pero los viajeros tienen la potestad de decidir si lo utilizan o no. El transporte terrestre no se selecciona durante el Join.
Diagrama de Contexto Delimitado	Figura 3.33.

Tabla 3.14: Contexto Delimitado de Transportes

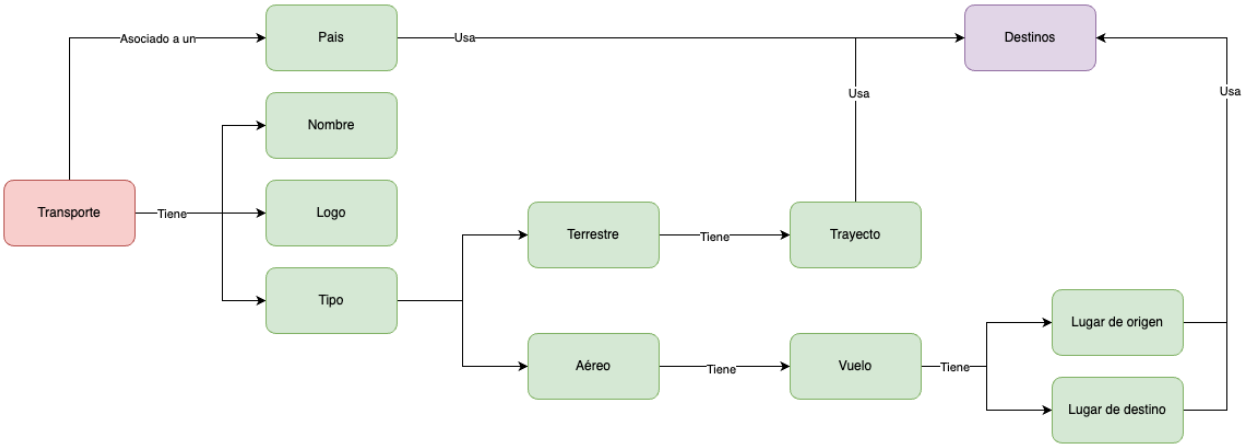


Figura 3.33: Contexto Delimitado de Transportes

- **Acomodaciones:** Se muestra en la Tabla 3.15.

Contexto Delimitado de Acomodaciones	
Subdominio	Viajes.
Clasificación Estratégica	Soporte.
Descripción	Las acomodaciones son los cuartos en los hoteles que ofrece Vaova a los viajeros. Se manejan diferentes hoteles y tres tipos de habitaciones para que los viajeros elijan.
Diagrama de Contexto Delimitado	Figura 3.34.

Tabla 3.15: Contexto Delimitado de Acomodaciones

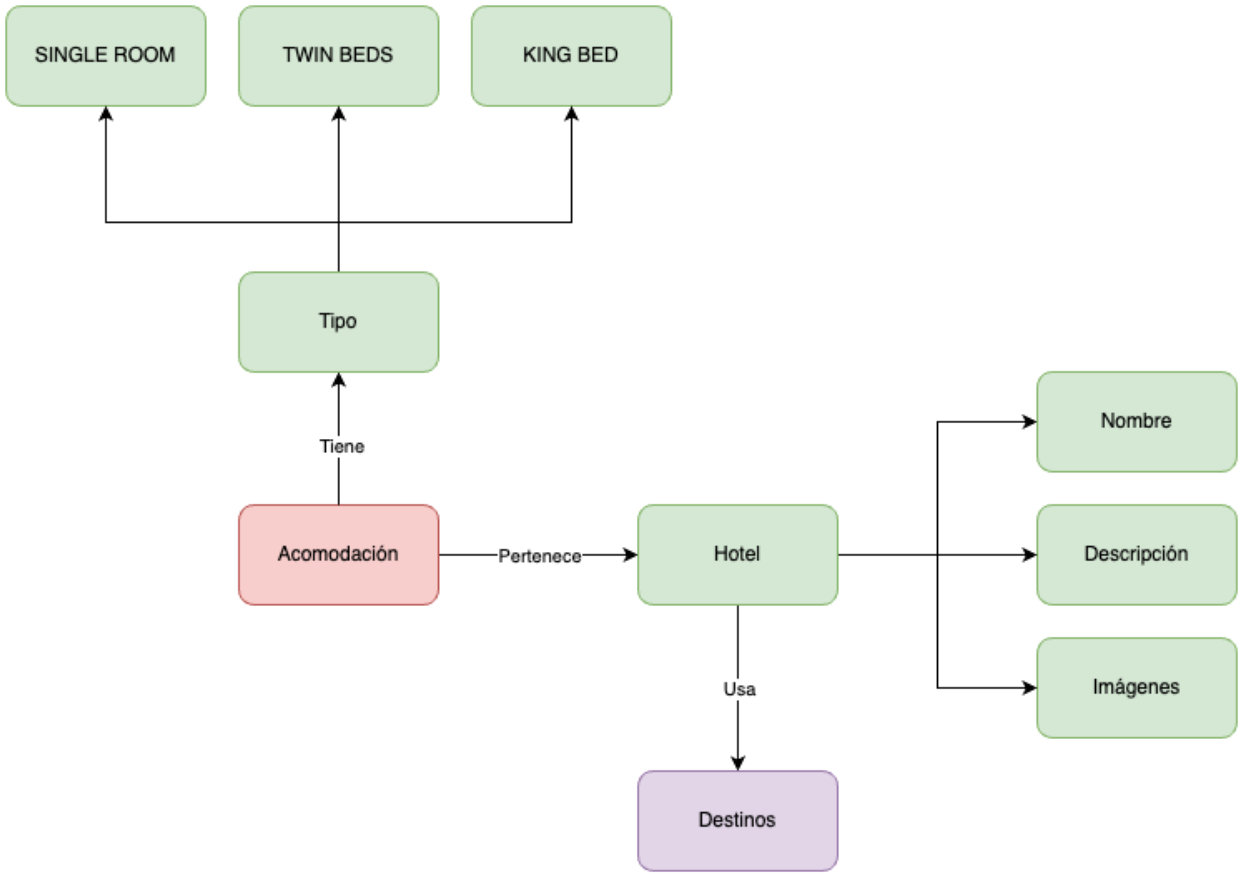


Figura 3.34: Contexto Delimitado de Acomodaciones

- **Actividades:** Se muestra en la Tabla 3.16.

Contexto Delimitado de Actividades	
Subdominio	Viajes.
Clasificación Estratégica	Soporte.
Descripción	Este subdominio presenta las actividades que Vaova diseña para que los viajeros puedan realizar en los destinos que visitan.
Diagrama de Contexto Delimitado	Figura 3.35.

Tabla 3.16: Contexto Delimitado de Actividades

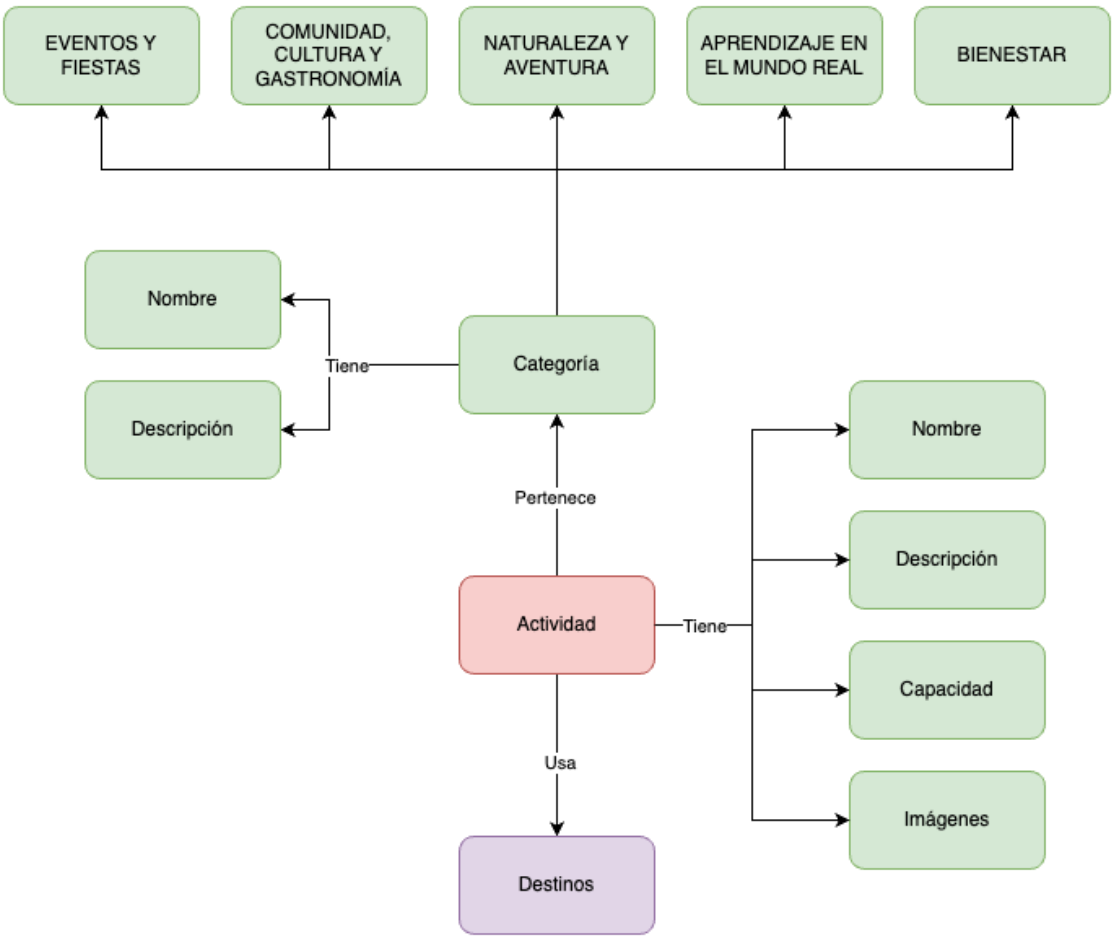


Figura 3.35: Contexto Delimitado de Actividades

- **Usuarios:** Se muestra en la Tabla 3.17.

Contexto Delimitado de Usuarios	
Subdominio	Usuarios.
Clasificación Estratégica	Soporte.
Descripción	Este subdominio de Usuarios se encarga de manejar la información de Promotores y Viajeros. También, los Asesores Comerciales cuentan con usuarios para la gestión de Itinerarios.
Diagrama de Contexto Delimitado	Figura 3.36.

Tabla 3.17: Contexto Delimitado de Usuarios

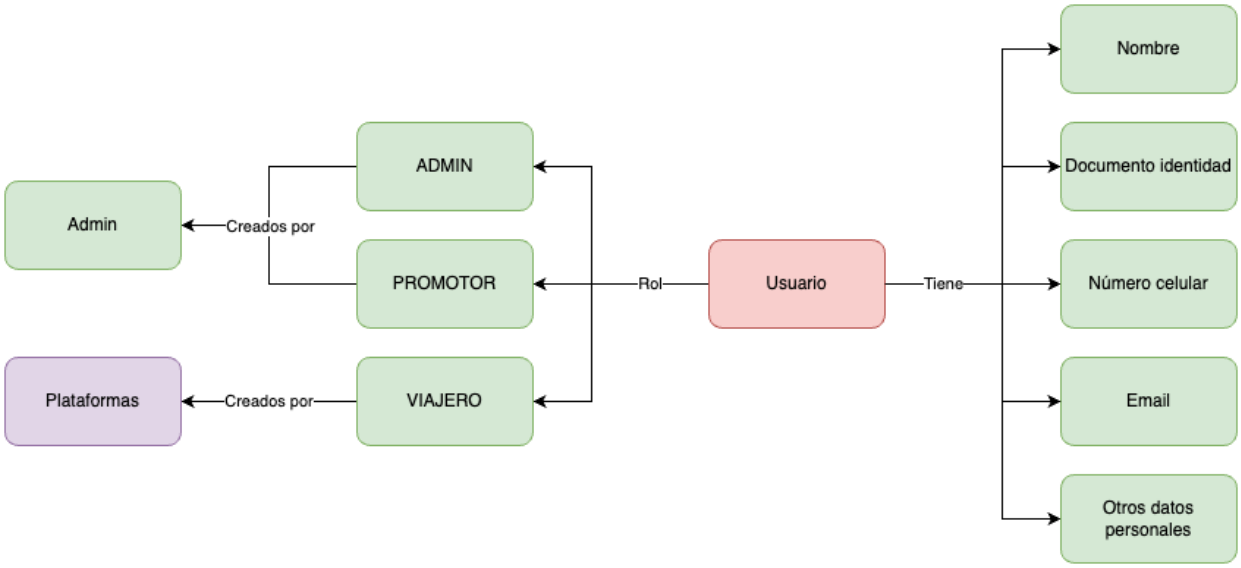


Figura 3.36: Contexto Delimitado de Usuarios

- **Plataformas:** Se muestra en la Tabla 3.18.

Contexto Delimitado de Plataformas	
Subdominio	Plataformas.
Clasificación Estratégica	Soporte.
Descripción	Este subdominio administra las plataformas externas con las que Vaova se integra para cargar y vender los viajes. Actualmente sólo son dos plataformas: *TourHero* y *WeTravel*.
Diagrama de Contexto Delimitado	Figura 3.37.

Tabla 3.18: Contexto Delimitado de Plataformas

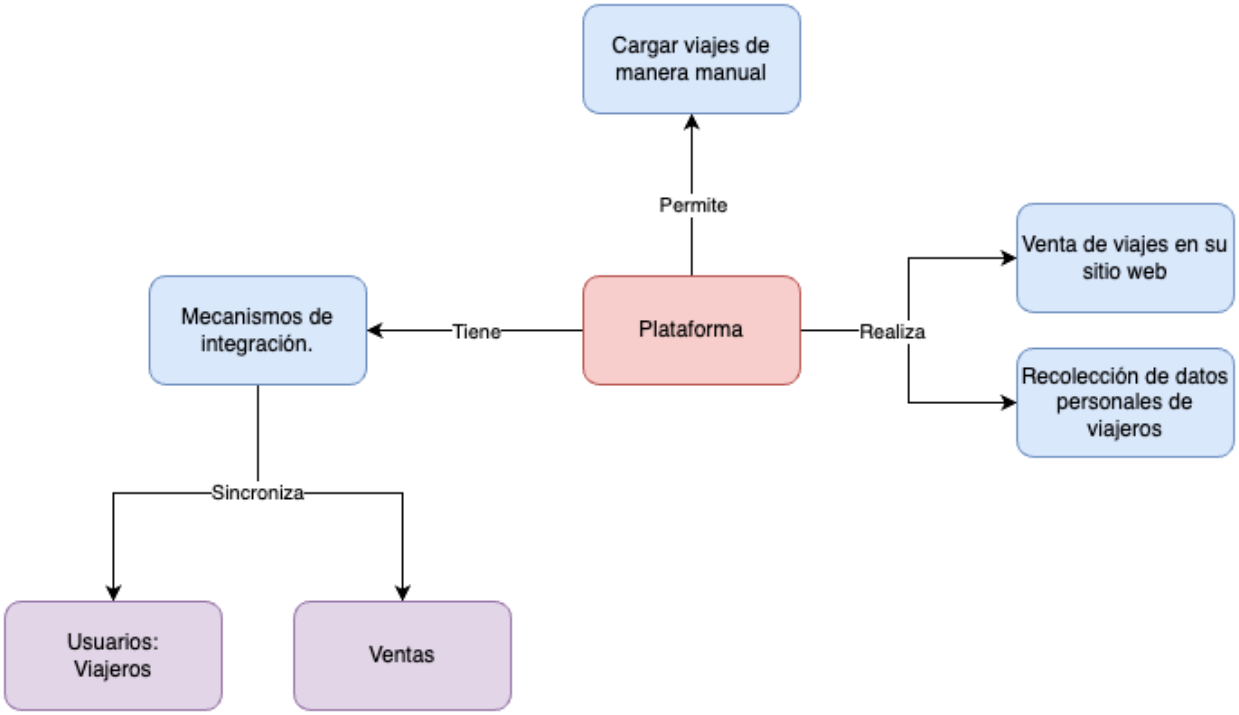


Figura 3.37: Contexto Delimitado de Plataformas

- **Ventas:** Se muestra en la Tabla 3.19.

Contexto Delimitado de Ventas	
Subdominio	Ventas.
Clasificación Estratégica	Soporte.
Descripción	El subdominio de Ventas se encarga de sincronizar la información de los viajeros y los paquetes que estos adquieren de los viajes. Los paquetes de los viajes se venden por medio de las plataformas externas (*TourHero* y *WeTravel*). Por lo tanto, esta información se debe obtener y sincronizar desde los sistemas externos de estas plataformas.
Diagrama de Contexto Delimitado	Figura 3.38.

Tabla 3.19: Contexto Delimitado de Ventas

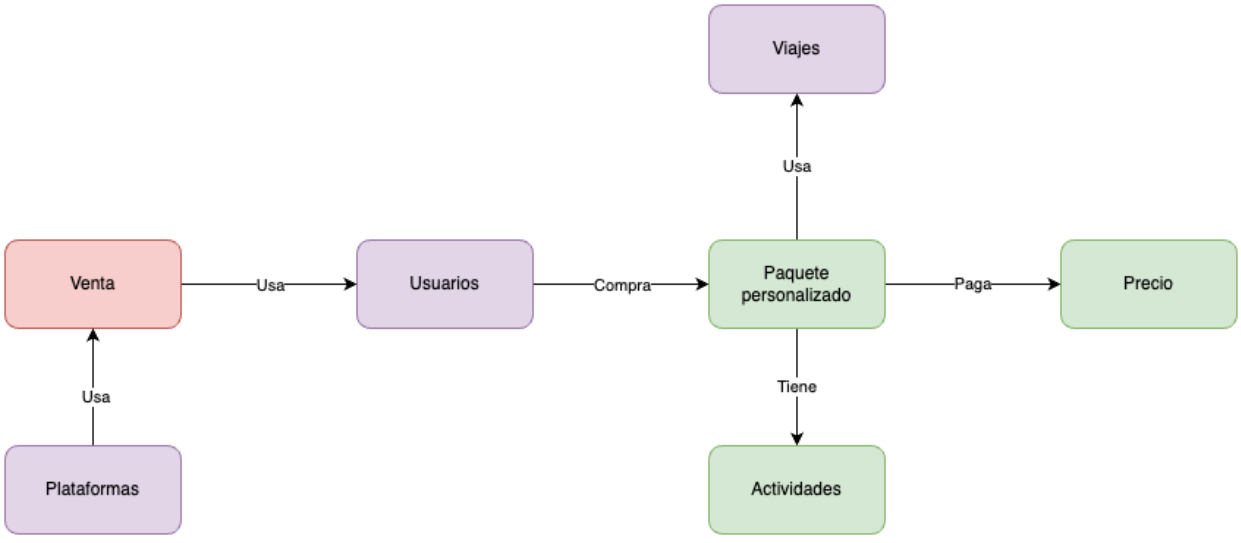


Figura 3.38: Contexto Delimitado de Ventas

A continuación, se definen los contextos delimitados que pertenecen a la clasificación estratégica de Genéricos.

- **Reportes:** Se muestra en la Tabla 3.20.

Contexto Delimitado de Reportes	
Subdominio	Reportes.
Clasificación Estratégica	Genérico.
Descripción	El subdominio de Reportes se encarga de mostrar información de interés para Vaova de forma sencilla y rápida. Por ejemplo, nivel de venta en los viajes, nivel de ocupación de vuelos o habitaciones, etc.
Diagrama de Contexto Delimitado	Figura 3.39.

Tabla 3.20: Contexto Delimitado de Reportes

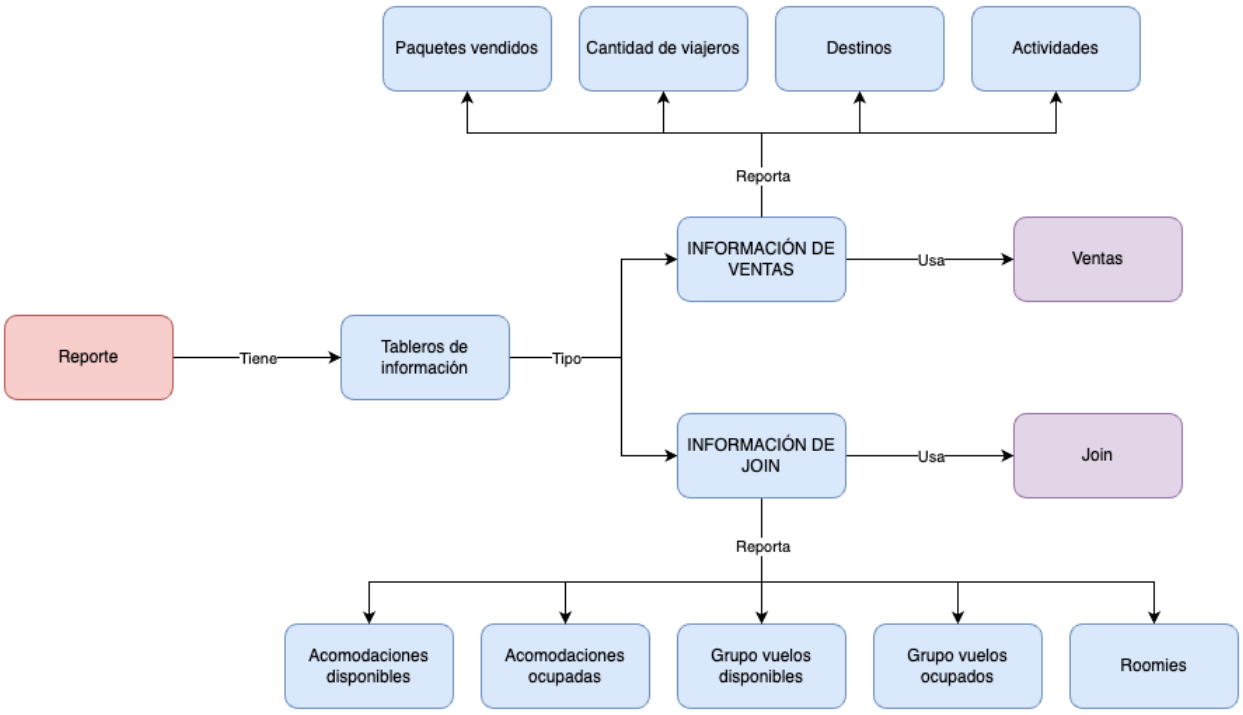


Figura 3.39: Contexto Delimitado de Reportes

- **Notificaciones:** Se muestra en la Tabla 3.21.

Contexto Delimitado de Notificaciones	
Subdominio	Notificaciones.
Clasificación Estratégica	Genérico.
Descripción	El subdominio permite enviar a los usuarios diferentes tipos de notificaciones como correos electrónicos o mensajes de texto. Estas notificaciones son importantes para el proceso automático de registro de viajeros y el Join.
Diagrama de Contexto Delimitado	Figura 3.40.

Tabla 3.21: Contexto Delimitado de Notificaciones

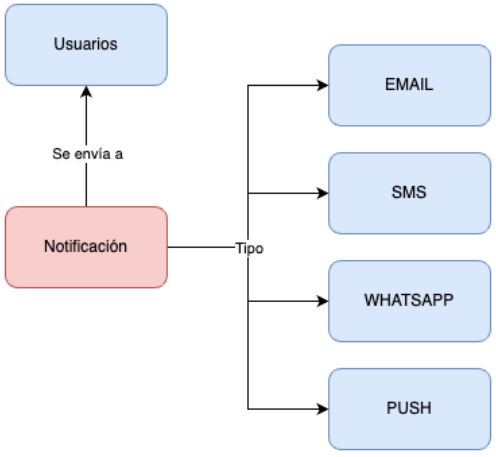


Figura 3.40: Contexto Delimitado de Notificaciones

3.5.1.4. Lenguajes Ubicuos

Evans (2015) define el lenguaje ubicuo de la siguiente manera: *lenguaje estructurado en torno al modelo de dominio y utilizado por todos los miembros del equipo dentro de un contexto delimitado para conectar todas las actividades del equipo de software.* En otras palabras, el lenguaje ubico es aquel que utilizan los expertos de negocio y las personas técnicas (arquitectos de software, líderes técnicos, desarrolladores, etc.) para entenderse mutuamente.

Como se mencionó en la nota informativa del inicio de la Sección 3.5, en este documento no se pretende trasladar todo lo que se plasmó en el documento de arquitectura de software. Por lo

tanto, sólo se definirá el lenguaje ubicuo para el subdominio de Itinerarios. Los lenguajes ubicuos de los otros subdominios se pueden consultar en el documento de arquitectura de software.

En la Tabla 3.22 se muestran todos los términos definidos en el lenguaje ubicuo para el subdominio de Itinerarios.

Lenguaje Ubicuo - Subdominio Itinerarios	
Itinerario	Es la cotización de un viaje, que no incluye precios, creada por un Promotor o un Asesor Comercial. En un itinerario se definen: las fechas globales de inicio y fin del viaje, cantidad mínima y máxima de viajeros permitidos, cantidad de noches globales del viaje, el país de destino, lugares del destino a visitar, actividades a realizar durante el viaje con su slot de tiempo asignado. En un Itinerario no se incluyen precios porque estos son calculados y asignados manualmente por el Asesor Comercial durante la revisión de este. Los Itinerarios son aceptados o rechazados por los Asesores Comerciales de Vaova.
Promotor	Es un estudiante, por lo general de nacionalidad colombiana o mexicana, de alguna universidad de Estados Unidos, que contacta con Vaova para organizar viajes a Colombia o México. Esta persona es la encargada de diseñar los itinerarios de los viajes que organiza junto con Vaova.
Asesor Comercial	Funcionario de Vaova encargado de revisar los Itinerarios propuestos por los Promotores. Si el funcionario acepta el Itinerario, le asigna precios al Itinerario y este pasa a ser un viaje.
Fechas globales	Un Itinerario tiene una fecha de inicio global y una fecha de fin global. Estas fechas definen el inicio y fin del viaje y no se realizan actividades antes o después de estas fechas. Se definen de esta manera porque cada destino tiene fechas de inicio y fin propias.
Cantidad de noches global	Un Itinerario tiene una cantidad global de noches. La cantidad de noches global se calcula a partir de las fechas globales. Se define de esta manera porque cada destino tiene una cantidad de noches propias. La cantidad global de noches define el número de destinos que se pueden visitar durante el viaje.
Lugar de destino	Es un pueblo, ciudad o región a visitar dentro del país de destino. Cada lugar de destino (o simplemente destino) define una fecha de inicio y fin, una cantidad de noches, cantidad mínima y máxima de viajeros, y un conjunto de actividades a realizar.

Continúa en la siguiente página

Tabla 3.22 – Continuación de la página previa

Lenguaje Ubicuo - Subdominio Itinerarios	
Actividad	Es una tarea, salida, reunión, conferencia o evento social diseñado por Vaova para el desarrollo y el disfrute de los viajeros.
Slot de tiempo	Es un rango de horas en las cuales se pueden realizar las actividades: ▪ *Mañana:* 8 am a 12 pm del mismo día. ▪ *Tarde:* 2 pm a 6 pm del mismo día. ▪ *Noche:* 8 pm a 3 am del día siguiente. Existen Slots de tiempo que no se pueden utilizar para planear actividades por parte de los Promotores al momento de diseñar el Itinerario: en la fecha global de inicio, no se permiten actividades en la mañana y tarde; en la fecha global de fin, no se permiten actividades en la tarde o noche. Reglas similares (pero no iguales), se aplican a las fechas de inicio y fin de cada destino. Estas restricciones se deben a los desplazamientos que deben realizar los viajeros.
Estado de Itinerario	Los Itinerarios pueden pasar por varios estados: ▪ *En progreso:* El Promotor no ha finalizado el Itinerario, pero lo ha guardado para no perder los cambios. ▪ *Finalizado:* El Promotor ha terminado el Itinerario. Este pasa a revisión por parte del Asesor Comercial. ▪ *Aprobado:* El Asesor Comercial revisó el Itinerario y lo aceptó. Cuando el Itinerario se acepta, el Asesor Comercial debe asignar precios a los destinos y actividades; el Itinerario pasa a ser un viaje. ▪ *Rechazado:* El Asesor Comercial revisó el Itinerario y lo rechazó. El Itinerario ya no puede ser editado. ▪ *Vencido:* La fecha de inicio del viaje se alcanzó y el Itinerario nunca se finalizó. El Itinerario ya no puede ser editado. ▪ *En revisión:* El Asesor Comercial está realizando cambios al Itinerario durante la revisión. ▪ *Cargado en plataforma:* El Itinerario se convirtió en viaje y fue cargado a una plataforma externa para ser vendido.

Tabla 3.22: Lenguaje Ubicuo - Subdominio Itinerarios

3.5.2. *Drivers* Arquitectónicos

Bass et al. (2021) definen que el diseño arquitectónico para sistemas de software es dirigido por los *drivers arquitectónicos*. Estos *drivers* consisten en requisitos arquitectónicamente significativos (*Arquitecturally Significant Requirements*), funcionalidades, restricciones, preocupaciones arquitectónicas y propósitos de diseño.

Teniendo en cuenta la definición anterior, los drivers arquitectónicos que se definieron para el diseño de la arquitectura de software fueron los siguientes: objetivos de negocio de Vaova, atributos de calidad, escenarios de calidad, tácticas de arquitectura y restricciones de arquitectura.

3.5.2.1. Objetivos de Negocio

Describen las metas que la organización quiere alcanzar o el estado al que se desea llegar. Los objetivos de negocio que se definieron para la arquitectura de software constan de cuatro partes:

- Se describen los antecedentes, la actualidad o los casos que más están causando problemas.
- Justificar el porqué se desea alcanzar esa meta o llegar a dicho estado, o cuáles serían los beneficios de solucionar dichos problemas.
- Definir las restricciones técnicas o tecnológicas que puede tener el objetivo de negocio.
- Indicar los atributos de calidad deseados.

Las Tablas 3.23, 3.24 y 3.25 muestran los objetivos de negocio definidos para el diseño de la arquitectura de software.

Objetivo de Negocio - Automatizar Procesos Operativos	
Objetivo de Negocio	Automatizar procesos operativos.
Código	BO_01
Antecedentes y detalles	
Continúa en la siguiente página	

Tabla 3.23 – Continuación de la página previa

Objetivo de Negocio - Automatizar Procesos Operativos	
1. No se cuenta con un mecanismo que permita a los Promotores diseñar los Itinerarios de manera autónoma. Un Asesor Comercial debe realizar una presentación con diapositivas de PowerPoint al Promotor que desea organizar un viaje, para que entonces el Promotor pueda diseñar el Itinerario. Este proceso puede durar varios días, por las validaciones posteriores que debe realizar el Asesor Comercial al momento de revisar el Itinerario. Por el lapso de espera tan largo, algunos Promotores desisten del proceso y los viajes se pierden. 2. El *Join* se maneja actualmente en un archivo de Excel. Los viajeros deben ingresar en las fechas establecidas a un archivo compartido por Vaova y allí gestionar sus grupos de vuelos, acomodaciones y selección de *Roomie*. Este proceso por medio del archivo de Excel genera inconsistencias en la información debido a que muchos viajeros editan el archivo simultáneamente, lo que causa que algunos cambios se sobrescriban y el *Join* quede inconsistente. 3. Se tiene un acuerdo con *Viaxlab* para que los viajeros puedan utilizar la aplicación móvil de esta compañía y tener su información personal, itinerario y recordatorios durante el viaje. La integración con *Viaxlab* no funciona y esta empresa no provee una documentación clara para automatizar el proceso. Actualmente la información se registra en la plataforma de manera manual. Vaova decidió prescindir de los servicios de *Viaxlab* y para ello necesita un reemplazo que ofrezca los mismos servicios y que opere de manera automatizada.	
¿Qué rol desempeñará?	Vaova quiere automatizar sus procesos operativos para ser más ágil. Se requiere una solución que permita automatizar los procesos de manera confiable, ya que realizar estos procesos de manera manual requiere mucho tiempo de los colaboradores y es propenso a errores.
¿Tiene restricciones?	No se van a automatizar todos los procesos operativos de la compañía. Sólo aquellos que no requieren intervención o decisiones humanas.
¿Qué atributos de calidad se requieren?	• **Adecuación funcional:** Pertinencia funcional • **Fiabilidad:** Madurez y Disponibilidad

Tabla 3.23: Objetivo de Negocio BO_01

Objetivo de Negocio - Mejorar la Experiencia de los Viajeros	
Objetivo de Negocio	Mejorar la experiencia de los viajeros.
Código	BO_02
Antecedentes y detalles	
Continúa en la siguiente página	

Tabla 3.24 – Continuación de la página previa

Objetivo de Negocio - Mejorar la Experiencia de los Viajeros	
1. Actualmente, para que una persona pueda viajar con Vaova debe realizar los siguientes pasos: ▪ Comprar el paquete de viaje en la plataforma de *TourHero* o *WeTravel*. Durante la compra debe seleccionar el paquete, los destinos, las actividades y los complementos del viaje que desea adquirir. En este paso realiza el pago e ingresa su información personal. ▪ Registrarse e ingresar a la aplicación web *MyTrip* para validar que su información personal y la información de su paquete de viaje están correctos. ▪ Realizar el Join por medio de un Excel compartido. ▪ Registrar en la aplicación móvil de *Viaxlab* para tener su información personal e itinerario de viaje. 2. Cuando Vaova ingresa manualmente la información de los viajeros a la plataforma de *Viaxlab*, esta genera un código QR para cada viajero con la información del viaje. Este código se debe utilizar para verificar si un viajero está registrado para una actividad. La disponibilidad de los servicios de *Viaxlab* es baja y cuando un viajero desea ingresar a una actividad, muchas veces, este código QR no se puede validar. Si no es posible validar los códigos QR, los operarios de Vaova deben realizar la validación de los viajeros registrados en las actividades de manera manual, y estos en viajes grandes (más de 400 viajeros) genera un cuello de botella en el acceso a las actividades y disgusto en las personas.	
¿Qué rol desempeñará?	Vaova quiere ofrecer una mejor experiencia a los viajeros, simplificando el proceso: • Comprar el viaje en la plataforma de *TourHero* o *WeTravel*. • Ofrecer una aplicación móvil propia con registro automático a partir de la compra realizada en alguna de las plataformas externas. Por medio de esta aplicación se podrá realizar el Join. Toda la información del viajero y sus viajes estará disponible en la misma aplicación.
¿Tiene restricciones?	• La información de los viajeros debe garantizarse desde la compra. • Los servicios de la aplicación móvil deben estar disponibles cuando se requieren. • Los operarios deben intervenir en los procesos cuando sea necesario.
¿Qué atributos de calidad se requieren?	• **Interoperabilidad:** Compatibilidad • **Fiabilidad:** Madurez y Disponibilidad

Tabla 3.24: Objetivo de Negocio BO_02

Objetivo de Negocio - Reducir Costos	
Objetivo de Negocio	Reducir costos.
Código	BO_03
Antecedentes y detalles	
1. Los Asesores Comerciales invierten mucho tiempo realizando presentaciones a los Promotores y revisando las propuestas de Itinerarios que estos realizan. 2. Los Operarios invierten mucho tiempo durante el Join, verificando que la información registrada en el archivo de Excel es correcta y no hubo modificaciones que no se debían realizar. 3. Si los códigos QR de *Viaxlab* no se pueden validar, se deben imprimir listas de actividades por viajero. Esto implica gastos en papel, tinta de impresoras y el tiempo que se invierte en las validaciones manuales. Aquí también se ve comprometida la imagen reputacional de la compañía ante los viajeros. 4. *Viaxlab* cobra a Vaova $9 dólares por viajero registrado, lo que incrementa mucho los costos operativos. Por ejemplo, para un viaje de 350 viajeros Vaova debe pagar $3150 dólares a *Viaxlab* por el uso de la aplicación móvil.	
¿Qué rol desempeñará?	• Reducir el tiempo que los Asesores Comerciales invierten en presentaciones, revisando y corrigiendo propuestas de Itinerarios. Las presentaciones a los Promotores no deberían tardar más de 30 minutos. La revisión y validación de los Itinerarios debería completarse en menos de un día. • El Join debe ser automático y no generar errores. Los operarios deben vigilar el Join sólo para gestionar los grupos de vuelos y las acomodaciones cuando se están agotando. • La integración con *Viaxlab* se debe retirar por los altos costos y el mal funcionamiento, y se deben proveer servicios que la reemplacen.
¿Tiene restricciones?	• Se contempla una aplicación web para el manejo de Itinerarios. • Se contempla una aplicación móvil para que los viajeros realicen el Join. • Los servicios deben estar separados del canal (web o móvil), y estar disponibles cuando se requieren.
¿Qué atributos de calidad se requieren?	• **Pertinencia funcional:** Adecuación funcional • **Fiabilidad:** Madurez y Disponibilidad

Tabla 3.25: Objetivo de Negocio BO_03

3.5.2.2. Atributos de Calidad

En la Sección 2.1.4 se describió como los atributos de calidad se relacionan con la arquitectura de software. El marco utilizado para definir los atributos de calidad en la arquitectura de software

para este proyecto de grado es el ISO/IEC25010:2011. Bass et al. (2012) definen que los atributos de calidad son requerimientos sobre cómo el sistema se debe comportar para satisfacer las necesidades de las partes interesadas.

Inicialmente, se propuso que la arquitectura de software se centraría en los atributos de calidad de *Disponibilidad* (Fiabilidad) e *Interoperabilidad* (Compatibilidad). Pero, haciendo el análisis de los objetivos de negocio, se encontraron atributos de calidad que no se habían contemplado. Se incluyen los atributos de calidad de *Pertinencia Funcional* (Adecuación Funcional) y Madurez (Fiabilidad) al listado de atributos de calidad que fueron contemplados en el diseño de la arquitectura de software. La Tabla 3.26

Característica de Calidad	Atributo de Calidad
Adecuación Funcional	Pertinencia Funcional
Compatibilidad	Interoperabilidad
Fiabilidad	Madurez, Disponibilidad

Tabla 3.26: Drivers Arquitectónicos - Atributos de Calidad

A continuación, se presentan las definiciones de cada característica y atributo de calidad de acuerdo al modelo ISO/IEC25010:2011 (ISO, 2022).

- **Adecuación Funcional:** Representa la capacidad del producto de software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas.

  - **Pertinencia Funcional:** Capacidad del producto de software para proporcionar un conjunto de funciones para tareas y objetivos de usuario especificados.

- **Compatibilidad**: Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software.

  - **Interoperabilidad:** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

  Fiabilidad: Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados.

  - **Madurez:** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
  - **Disponibilidad:** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.

3.5.2.3. Escenarios de Calidad

Una vez definidos los atributos de calidad que deben guiar el diseño de la arquitectura, se deben definir los escenarios de calidad con los cuales se busca satisfacer dichos atributos de calidad. Bass et al. (2021) ofrecen una plantilla para la definición de atributos de calidad que consiste de los siguientes elementos:

- **Fuente del estímulo:** Es alguna entidad (un humano, un sistema de computación, o cualquier otro actor) que genera el estímulo.
- **Estímulo:** Es una condición o evento que requiere una respuesta cuando llega al sistema.
- **Ambiente:** El estímulo ocurre bajo unas condiciones específicas. El sistema puede estar en condición de operación normal o sobrecarga, o cualquier otro estado relevante.
- **Artefacto:** Algún artefacto es estimulado. Este artefacto puede ser una colección de sistemas, el sistema entero, o alguna pieza o piezas de este.
- **Respuesta:** La respuesta es la actividad realizada como resultado de la llegada del estímulo.
- **Medición de la respuesta:** Cuando se produce la respuesta, esta debería ser medida de alguna manera de tal manera que el requerimiento pueda ser probado.

El último elemento, *medición de la respuesta*, es especialmente importante en la definición de los escenarios de calidad. Ya que expresa en valores cuantificables el resultado esperado, de esta manera es sencillo determinar si la arquitectura satisface el escenario de calidad una vez este es probado y medido.

Para el diseño de la arquitectura de software se definieron diez escenarios de calidad: tres escenarios para adecuación funcional, tres escenarios para interoperabilidad, tres escenarios para madurez y un escenario para disponibilidad. Las Tablas 3.27 a 3.36 muestran los escenarios definidos.

Escenario de Calidad QS_01 - Adecuación-Funcional	
Fuente	Un usuario con Rol de Promotor.
Estímulo	Selecciona la opción de *Crear Itinerario.*
Artefacto	Aplicación web de Itinerarios.
Ambiente	Dispositivo del cliente con conexión a internet.
Continúa en la siguiente página	

Tabla 3.27 – Continuación de la página previa

Escenario de Calidad QS_01 - Adecuación-Funcional	
Respuesta	El sistema debe permitir al usuario gestionar la siguiente información para crear el Itinerario: 1. Nombre del viaje, país de destino, fechas globales, y número estimado de viajeros. 2. Seleccionar la cantidad de destinos dentro del país. 3. Para cada destino seleccionado: a. Elegir fechas del destino. b. Elegir actividades del destino con su *slot* de tiempo. 4. Mostrar resumen del Itinerario. 5. Terminar el Itinerario.
Medida de Respuesta	La tasa de éxito de Itinerarios terminados debe ser mínimo del 70 %.

Tabla 3.27: Escenario de Calidad QS_01

Escenario de Calidad QS_02 - Adecuación-Funcional	
Fuente	Un usuario con rol de Asesor Comercial.
Estímulo	Selecciona *Validar Itinerario*.
Artefacto	Aplicación web de Itinerarios.
Ambiente	Dispositivo del cliente con conexión a internet.
Respuesta	El sistema debe permitir al usuario realizar lo siguiente: 1. Editar el Itinerario si es necesario. 2. Aceptar o Rechazar el Itinerario. 3. Si Acepta el Itinerario: a. Asignar precios a cada destino y actividad. 4. Terminar la validación del Itinerario.
Medida de Respuesta	La tasa de éxito de Itinerarios validados (aceptados o rechazados) debe ser mínimo del 90 %.

Tabla 3.28: Escenario de Calidad QS_02

Escenario de Calidad QS_03 - Adecuación-Funcional	
Fuente	Un usuario con rol de Viajero.
Estímulo	Selecciona *Realizar Join*.
Artefacto	Aplicación móvil Vaovapp.
Ambiente	Dispositivo del cliente con conexión a internet.
Respuesta	El sistema debe permitir al usuario realizar lo siguiente: 1. Seleccionar tipo de acomodación. 2. Seleccionar hoteles y habitaciones. 3. Si aplica, seleccionar *Roomie*. 4. Seleccionar grupo de vuelos. 5. Finalizar Join.
Medida de Respuesta	La tasa de éxito de Join terminados debe ser mínimo del 90 %.

Tabla 3.29: Escenario de Calidad QS_03

Escenario de Calidad QS_04 - Interoperabilidad	
Fuente	Un usuario con cualquier rol.
Estímulo	Selecciona *Iniciar Sesión* en la aplicación web de Itinerarios o en la aplicación móvil Vaovapp.
Artefacto	Servicio de autenticación de Auth0.
Ambiente	Se tiene integración definida con Auth0 y sus servicios son conocidos de antemano.
Respuesta	El sistema Auth0 válida las credenciales del usuario y le permite ingresar a la aplicación.
Medida de Respuesta	Los intentos de autenticación con credenciales válidas de Auth0 son exitosos el 99 % de las veces.

Tabla 3.30: Escenario de Calidad QS_04

Escenario de Calidad QS_05 - Interoperabilidad	
Fuente	Integración *TourHero Sync*.
Estímulo	Solicita a la plataforma de *TourHero* la información de las ventas de los viajes programados por Vaova.

Continúa en la siguiente página

Tabla 3.31 – Continuación de la página previa

Escenario de Calidad QS_05 - Interoperabilidad	
Artefacto	Plataforma externa de *TourHero*.
Ambiente	La plataforma de *TourHero* es conocida de antemano.
Respuesta	La plataforma de *TourHero* envía la información de las ventas actualizadas hasta ese momento.
Medida de Respuesta	La información que devuelve la plataforma de *TourHero* se interpreta y procesa de manera correcta el 99 % de las veces.

Tabla 3.31: Escenario de Calidad QS_05

Escenario de Calidad QS_06 - Interoperabilidad	
Fuente	Integración *WeTravel Sync*.
Estímulo	Solicita a la plataforma de *WeTravel* la información de las ventas de los viajes programados por Vaova.
Artefacto	Plataforma externa de *WeTravel*.
Ambiente	La plataforma de *WeTravel* es conocida de antemano.
Respuesta	La plataforma de *WeTravel* envía la información de las ventas actualizadas hasta ese momento.
Medida de Respuesta	La información que devuelve la plataforma de *WeTravel* se interpreta y procesa de manera correcta el 99 % de las veces.

Tabla 3.32: Escenario de Calidad QS_06

Escenario de Calidad QS_07 - Madurez	
Fuente	Transacciones en el sistema.
Estímulo	Los usuarios utilizan los servicios de Itinerarios y Vaovapp.
Artefacto	Aplicación web de Itinerarios y aplicación móvil Vaovapp.
Ambiente	Los sistemas de Itinerarios y Vaovapp se encuentran en operación normal.
Respuesta	Las transacciones se procesan exitosamente cuando se invocan, sin generar errores o retrasos en los tiempos de respuesta.
Medida de Respuesta	El MTBF semanal debe ser de 6,65 días.

Tabla 3.33: Escenario de Calidad QS_07

Escenario de Calidad QS_08 - Madurez	
Fuente	Externa, fallos de hardware o red.
Estímulo	Los usuarios utilizan los servicios de Itinerarios y Vaovapp.
Artefacto	Aplicación web de Itinerarios y aplicación móvil Vaovapp.
Ambiente	Los sistemas de Itinerarios y Vaovapp sufren un error que provoca que se encuentren por fuera de operación normal.
Respuesta	Los usuarios no pueden consultar los servicios de Itinerarios y Vaovapp.
Medida de Respuesta	El MTTR (Resolución) semanal debe ser máximo de 5,4 horas.

Tabla 3.34: Escenario de Calidad QS_08

Escenario de Calidad QS_09 - Madurez	
Fuente	External, equipo de desarrollo.
Estímulo	Mantenimiento y actualizaciones.
Artefacto	Aplicación web de Itinerarios y aplicación móvil Vaovapp.
Ambiente	Los sistemas de Itinerarios y Vaovapp se suspenden temporalmente durante la ventana de mantenimiento.
Respuesta	El sistema se encuentra fuera de operación durante la ventana de mantenimiento y los usuarios no pueden acceder a los servicios.
Medida de Respuesta	La ventana de mantenimiento debe durar máximo 3 horas. Esta ventana de mantenimiento se programará cada dos semanas los lunes entre 6 P.M. a 9 P.M. con la condición de que no esté en proceso un Join.

Tabla 3.35: Escenario de Calidad QS_09

Escenario de Calidad QS_10 - Disponibilidad	
Fuente	Transacciones en el sistema.
Estímulo	Los usuarios utilizan los servicios de Itinerarios y Vaovapp.
Artefacto	Aplicación web de Itinerarios y aplicación móvil Vaovapp.
Ambiente	Los sistemas de Itinerarios y Vaovapp se encuentran en operación normal.

Continúa en la siguiente página

Tabla 3.36 – Continuación de la página previa

Escenario de Calidad QS_10 - Disponibilidad	
Respuesta	Las transacciones se procesan exitosamente cuando se invocan sin generar errores o retrasos en los tiempos de respuesta.
Medida de Respuesta	El tiempo de respuesta de cada transacción es de máximo 2 segundos para el percentil 95 %.

Tabla 3.36: Escenario de Calidad QS_10

En los escenarios de calidad presentados en las Tablas 3.33, 3.34 y 3.35 se introducen los conceptos de MTBF, MTTR y ventana de mantenimiento. Atlassian (2023) ofrece la definición de un grupo de conceptos que son muy importantes para medir la fiabilidad y madurez de los sistemas de software: MTBF, MTTR, MTTA y MTTF.

- ***Mean Time Between Failures* (MTBF):** El tiempo medio entre fallos es la medida de tiempo entre fallos reparables de un producto tecnológico. La métrica se utiliza para controlar tanto la disponibilidad como la fiabilidad de un producto. Cuanto mayor sea el tiempo entre fallos, más fiable será el sistema.

  El MTBF se calcula utilizando una media aritmética: se toman los datos del período que se quiere calcular (horas, días, semanas, meses, años) y se divide el tiempo de funcionamiento total de ese período por el número de fallos.

  $$MTBF = \frac{Periodo - TiempoTotalInactividad}{NumeroTotalIncidentes}$$

- **MTTR:** Esta no es una métrica única. La R puede tener cuatro significados que representan mediciones relacionadas pero diferentes.

  - ***Mean Time To Repair*:** El tiempo medio de reparación es la media de tiempo que se tarda en reparar un sistema. Incluye tanto el tiempo de reparación como el tiempo de prueba. El reloj no se detiene en esta métrica hasta que el sistema vuelve a funcionar por completo.

    Se puede calcular este MTTR sumando el tiempo total dedicado a las reparaciones durante un período determinado y dividiendo ese tiempo por el número de reparaciones.

    $$MTTR_{Reparacion} = \frac{TiempoTotalReparacion}{NumeroTotalIncidentes}$$

  - ***Mean Time To Respond*:** El tiempo medio de respuesta es la media de tiempo que se tarda en recuperarse un sistema de un fallo desde el momento en que se avisa por

primera vez de dicho fallo. Este tiempo no incluye tiempos de retraso en los sistemas de monitoreo y alertas.
Este MTTR se calcula sumando el tiempo de respuesta total desde la alerta hasta el momento en que el sistema vuelve a funcionar por completo dividido por el número de incidentes.

$$MTTR_{Respuesta} = \frac{TiempoDeAleta + TiempoTotalReparacion}{NumeroTotalIncidentes}$$

- ***Mean Time To Recovery*:** El tiempo medio de recuperación o tiempo medio de restauración es la media de tiempo que se tarda en recuperarse de un fallo un sistema. Esto incluye todo el tiempo de la interrupción, desde el momento en que el sistema falla hasta que vuelve a funcionar por completo.

  El tiempo medio de recuperación se calcula sumando todo el tiempo de inactividad en un período concreto y dividiéndolo por el número de incidentes.

$$MTTR_{Recuperacion} = \frac{TiempoTotalInactividad}{NumeroTotalIncidentes}$$

- ***Mean Time To Resolve*:** El tiempo medio de resolución es la media de tiempo que se tarda en resolver completamente un fallo. Esto incluye no sólo el tiempo dedicado a detectar el fallo, diagnosticar el problema y reparar la incidencia, además del tiempo dedicado a garantizar que el fallo no vuelva a presentarse.

  Para calcular este MTTR, se suma el tiempo de resolución total durante el período en el que se quiere hacer el seguimiento y se divide por el número de incidentes.

$$MTTR_{Resolucion} = \frac{TiempoTotalReparacion + TiempoTotalMonitoreo}{NumeroTotalIncidentes}$$

Las cuatro medidas de MTTR están relacionadas y son incrementales, esto quiere decir que una medida de MTTR incluye o tiene en cuenta datos que otras medidas de MTTR no. En este sentido, es posible realizar la siguiente afirmación:

$$MTTR_{Reparacion} < MTTR_{Respuesta} < MTTR_{Recuperacion} < MTTR_{Resolucion}$$

- ***Mean Time To Acknowledge* (MTTA):** El tiempo medio de confirmación de recepción es la media de tiempo que transcurre desde que se activa una alerta hasta que se empieza a trabajar en la incidencia. Esta métrica es útil para hacer un seguimiento de la capacidad de respuesta del equipo y de la eficiencia del sistema de alertas.

  Para calcular el MTTA se suma el tiempo transcurrido entre la alerta y la confirmación de recepción y se divide por el número de incidentes.

$$MTTA = \frac{TiempoTotalAlertas}{NumeroTotalIncidentes}$$

- ***Mean Time To Failure* (MTTF):** El tiempo medio sin averías es la media de tiempo que transcurre sin averías no reparables de un producto tecnológico. Este indicador se utiliza para conocer la duración típica de un sistema, determinar si una nueva versión de un sistema supera a la anterior y ofrecer a los clientes información sobre la vida útil prevista y cuándo deben programar las revisiones de sus sistemas.

  El tiempo medio sin averías es una media aritmética, por lo que se calcula sumando el tiempo de funcionamiento total de los sistemas que se están evaluando y dividiendo esa cifra por el número de dispositivos.

$$MTTF = \frac{TiempoTotalFuncionamiento}{NumeroTotalDispositivos}$$

La Figura 3.41 muestra un resumen sobre la relación entre los indicadores MTTR, MTBF y MTTF.

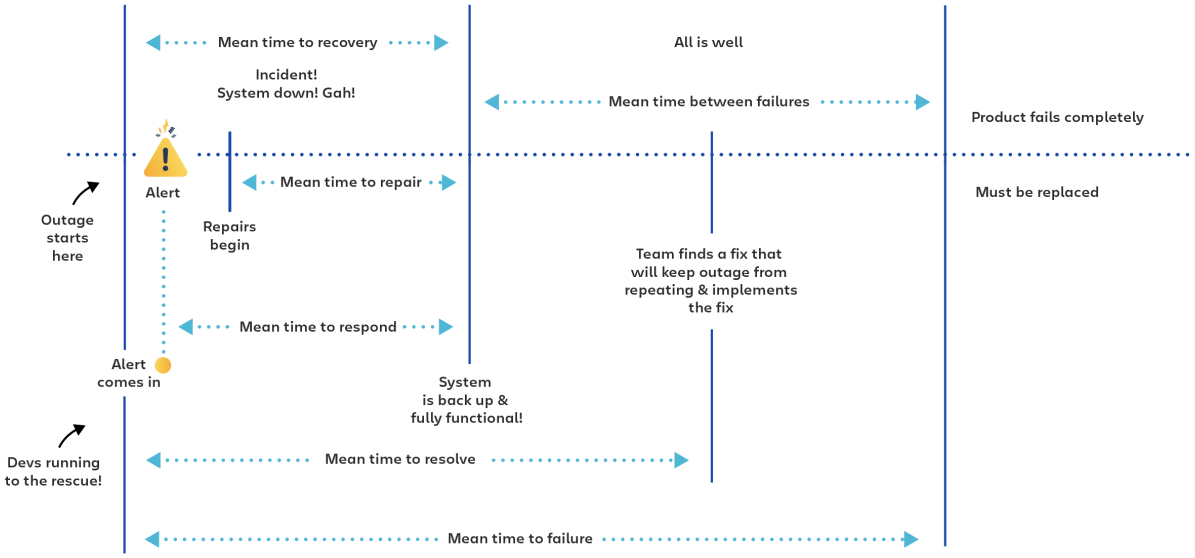


Figura 3.41: Relación entre MTTR, MTBF y MTTF (Atlassian, 2023)

3.5.2.4. Tácticas

Las tácticas de arquitectura fueron explicadas en la Sección 2.1.5, por lo que no se requiere realizar definiciones de este concepto. En la Tabla 3.37 se muestran las tácticas que fueron elegidas para el diseño de la arquitectura. Cada táctica define su nombre, el escenario de calidad que ayuda a resolver y la descripción de la táctica.

Táctica	Resuelve	Descripción
AT_01 - Modularidad	QS_01 QS_02 QS_03	Cada subdominio se implementará como un módulo independiente de los demás. Para los módulos que se comunican con otros o con servicios externos, se deben definir contratos de intercambio de información.
AT_02 - Separación de Responsabilidades	QS_01 QS_02 QS_03	Cada módulo será el encargado de ejecutar las operaciones de acuerdo con su contexto delimitado y proveerá los servicios a los demás módulos para que puedan cumplir con sus procesos.
AT_03 - Contratos Bien Definidos	QS_01 QS_02 QS_03	Cada módulo deberá exponer contratos con las restricciones que se deben cumplir para poder ser consumidos por otros módulos. Estos contratos se definirán a nivel arquitectónico.
AT_04 - Estandarización de Prácticas de Desarrollo	QS_07 QS_08 QS_09	Se definirán los estándares de programación como patrones de diseño, convenciones de código, lenguajes de programación, *frameworks* y herramientas a utilizar durante el desarrollo. Inicialmente, sólo se utilizará el lenguaje Python para el desarrollo de los módulos de cada subdominio.
AT_05 - Documentación	QS_07 QS_08 QS_09	La documentación a cada nivel se centrará en los siguientes aspectos: • Arquitectura: Diseños, cambios y actualizaciones. • Módulos: Cada módulo contará con la documentación de sus contratos. Estos deben incluir parámetros, cuerpo de petición, obligatoriedad, longitud de campos, valores permitidos, códigos de respuesta, etc. • Código: Se documentan los servicios o métodos que conllevan lógica de negocio especifica y por ende no son tan sencillos de comprender.
AT_06 - Integración Continua	QS_07 QS_08 QS_09	Se implementarán *Pipelines* que permitan realizar análisis estático de código y ejecutar pruebas automáticas. Las pruebas que se deben ejecutar en los *Pipelines* serán: • Pruebas unitarias • Pruebas de integración • Pruebas de sistema • Pruebas E2E
AT_07 - Monitoreo y Registro	QS_04 QS_05 QS_06	Se implementarán herramientas que permitan consultar los logs del sistema. Para lo anterior se debe estandarizar el manejo y registro de errores y excepciones en el código.

Continúa en la siguiente página

Tabla 3.37 – Continuación de la página previa

Táctica	Resuelve	Descripción
AT_08 - Protocolos y Formatos Estandarizados	QS_04 QS_05 QS_06	Se emplearán los protocolos y estándares que más se utilizan en la industria: • El formato de intercambio de datos será JSON. • El protocolo de comunicación será HTTP/HTTPS.
AT_09 - Mapeo y Transformación de Datos	QS_04 QS_05 QS_06	Los sistemas externos (Auth0, TourHero, WeTravel) son sistemas que ya están desarrollados. Los sistemas internos se deben adaptar a los datos que esos sistemas retornan, por lo cual se deben mapear o transformar antes de ser utilizados (para evitar acoplamiento a nivel de campos). TourHero y WeTravel retornan la información en archivos de Excel. Para este caso, el contenido del Excel se debe leer y convertir en JSON antes de ser procesado. Si un sistema externo retorna datos en un formato diferente a JSON, estos datos deben pasar por un adaptador para ser convertidos a JSON.
AT_10 - Replicación	QS_10	Se implementará redundancia activa en los componentes de cada módulo para mejorar la disponibilidad y los tiempos de respuesta.
AT_11 - Balanceo de Carga	QS_10	Se debe contar con un componente que pueda distribuir de manera uniforme las peticiones entre los componentes.
AT_12 - Escalamiento Horizontal	QS_10	El sistema debe estar en la capacidad agregar (*scale-out*) o retirar (*scale-in*) componentes de software según lo requiera la demanda.

Tabla 3.37: Tácticas de Arquitectura

3.5.2.5. Restricciones de Arquitectura

Bass et al. (2012) definen una restricción como *una decisión de diseño cero grados de libertad*. En este sentido, las restricciones son imposiciones sobre el diseño de la arquitectura. Estas restricciones se pueden generar por requerimientos del negocio, petición de los usuarios, presupuesto, tecnología, regulaciones nacionales o internacionales, entre otras.

Las restricciones de arquitectura definidas para el diseño de la arquitectura de software se muestran en la Tabla 3.38.

Restricciones de Arquitectura	
AC_01 - Auth0	Se debe utilizar la plataforma *Auth0* para realizar la autenticación y autorización en aplicación web de Itinerarios y la aplicación móvil Vaovapp.
AC_02 - Proveedor de Servicios de Nube	La arquitectura se debe diseñar teniendo en cuenta que se utilizará el proveedor AWS para desplegar los servicios.
AC_03 - Lenguaje de Programación	El lenguaje de programación que se utilice debe ser Python.
AC_04 - Integración con Plataformas de Ventas	Se deben tener en cuenta las integraciones con *TourHero* y *WeTravel* y su funcionamiento, para poder sincronizar los datos de viejeros y paquetes vendidos.

Tabla 3.38: Restricciones de Arquitectura

3.5.3. Diseño Táctico

En la etapa de diseño táctico se pasó de definiciones de negocio a definiciones técnicas. Toda la información y datos recolectados en la Sección 3.5.1 se toman en cuenta para realizar el diseño de la arquitectura a nivel técnico. En esta sección y en la siguiente (Sección 3.5.4) se definen las decisiones a nivel de arquitectura que fueron tomadas y se definen los diagramas de arquitectura.

3.5.3.1. Decisiones de Arquitectura

El diseño de un sistema consiste en un conjunto de decisiones. Algunas de estas decisiones ayudan a controlar las respuestas a los atributos de calidad; otros aseguran el logro de las funcionalidades del sistema (Bass et al., 2021).

Teniendo en cuenta la definición del autor, todas las decisiones que se tomen al momento de realizar el diseño de la arquitectura se deben documentar [2] y mantener actualizadas. Es indispensable que las decisiones de arquitectura tengan una justificación o un porqué, para que puedan ser entendidas por las personas que deben mantener o realizar mantenimiento al sistema en el futuro. Las decisiones de arquitectura para el diseño se muestran en la Tabla 3.39.

[2]El documento de registro de decisiones de arquitectura de software puede ser encontrado en la Sección 6.6. La Tabla 3.39 sólo muestra de manera resumida cada una de las decisiones de arquitectura.

Decisiones de Arquitectura	
ADR_01 - Estilo Arquitectónico	Se elije el estilo de microservicios. Este estilo es adecuado cuando se ha utilizado *Domain Driven Design*, ya que cada subdominio se puede convertir de manera sencilla en un microservicio.
ADR_02 - Frameworks	Se elijen los siguientes *frameworks* que son compatibles con el lenguaje *Python*: • **Desarrollo:** *FastAPI*. • **Pruebas de software**: *Pytest*. • **Pruebas E2E**: *Karate*.
ADR_03 - Estructura de Microservicios	Se define una estructura de capas en los microservicios. Esta estructura permite realizar una división lógica y establecer las responsabilidades de cada capa.
ADR_04 - BackOffice	Se define un componente llamado *BackOffice*. Este será un componente de software (monolito) que administrará los subdominios de Destinos, Actividades, Acomodaciones y Transportes.
ADR_05 - Itinerarios	Los subdominios de Itinerarios y Viajes se manejarán en el componente de Itinerarios. Este componente tendrá las capacidades para: • Administrar los Itinerarios. • Aprobar o rechazar los Itinerarios. • Administrar los Viajes.
ADR_06 - Serverless	Se deben utilizar tecnologías *Serverless* siempre que sea posible. Los servicios tipo *Serverless* (sin servidor) son soluciones que permiten crear y ejecutar servicios en la nube de manera sencilla y rápida.
ADR_07 - Lambda	Se elige el servicio AWS Lambda para ejecutar los microservicios. Se estima que AWS Lambda es el más adecuado para el nivel de carga y funcionalidades que debe realizar la arquitectura.
ADR_08 - Motor de Base de Datos	Se elige el motor de bases de datos relacional *MySQL* para el almacenamiento de cada microservicio. El servicio de AWS a utilizar será Amazon RDS Aurora con *MySQL*.
ADR_09 - Colas de Mensajería	Se elige Amazon SQS. *Amazon Simple Queue Service* (SQS) es un servicio que permite enviar mensajes entre componentes de software por medio de colas de mensajería.
ADR_10 - Logs	Se deben manejar logs de auditoria en los microservicios para poder rastrear errores y solucionarlos de manera sencilla. Las estructura de los mensajes que se envían como log tendrá la siguiente estructura: componente, nivel de severidad, tiempo, mensaje, parámetros y resultado.
ADR_11 - Diagramas como Código	Los diagramas se deben realizar con herramientas que permitan generarlos a partir de código (*Diagrams as Code*).

Continúa en la siguiente página

Tabla 3.39 – Continuación de la página previa

Decisiones de Arquitectura	
ADR_12 - Componente Aggregator	Se incluye un componente agregador que permite recolectar la información de diferentes fuentes y retornarla para que las GUI tengan la capacidad de mostrar los datos de manera que los usuarios la puedan entender. Para este componente se utilizará el servicio *AWS AppSync*, que ofrece capacidades de *GraphQL*, almacenamiento en caché y sincronización de datos sin conexión.
ADR_13 - Caché en Componente de Autenticacion	Se debe incluir una base de datos de tipo caché (o similar) en el componente Autenticacion. Esta base de datos permitirá almacenar los *tokens* generados en *Auth0* para los usuarios, ya que estos *tokens* tienen una vida útil larga. Para este propósito se utilizará el servicio AWS DynamoDB que es un servicio *Serverless* de base de datos *NoSQL* del tipo clave-valor (*key-value*).
ADR_14 - Componente Recordatorios	Se debe agregar el componente Recordatorios. Este componente se debe activar periódicamente de manera automática y buscar los Itinerarios que estén sin finalizar y enviar notificaciones a los Promotores para que los finalicen antes de su vencimiento.

Tabla 3.39: Decisiones de Arquitectura

Es importante aclarar que las decisiones ADR_12, ADR_13 y ADR_14 se agregaron después de la evaluación técnica de la arquitectura (Sección 4.1). Por este motivo no aparecen en el primer diseño de la arquitectura.

3.5.4. Arquitectura de Solución

En esta sección se muestran las diferentes vistas de arquitectura que fueron generadas para representar a la arquitectura. Rozanski and Woods (2012) definen una vista como *una representación de uno o más aspectos estructurales de una arquitectura que ilustra cómo una arquitectura aborda una o más preocupaciones de una o más de sus partes interesadas.* Así, cada una de las vistas que se presenta para el diseño de la arquitectura puede estar dirigida a una o varias partes interesadas, pero responde a una pregunta en concreto.

3.5.4.1. Modelo de Contexto

La Vista de Contexto describe las relaciones, dependencias e interacciones entre los sistemas y su entorno (las personas, sistemas y entidades externas con las que interactúa) (Rozanski and Woods, 2012). Para realizar el diagrama de se utilizó una nomenclatura en la que se define el significado de cada elemento. La Tabla 3.40 muestra a quién va dirigida la vista y las preocupaciones que responde esta vista.

Vista de Contexto	
Objetivo	Describir las relaciones, dependencias e interacciones entre los sistemas y su entorno (las personas, sistemas y entidades externas con las que interactúa).
Dirigida a	A todas las partes interesadas, con énfasis a los equipos de arquitectura y de desarrollo.
Responde a	• Alcance del sistema y responsabilidades. • Definición de componentes y relaciones. • Agrupación lógica de componentes. • Dependencias con sistemas externos.

Tabla 3.40: Vista de Contexto

En el diseño de la arquitectura se realizó una agrupación lógica de los componentes. De esta manera, los componentes se agruparon de tal manera que se lograra una alta cohesión en los procesos que realizan. Los grupos o capas definidos fueron los siguientes:

- **Capa de redireccionamiento:** En esta capa se reciben todas las peticiones que llegan al sistema y se realiza la autenticación en *Auth0*. Se definen los componentes Gateway, Aggregator y Autenticacion.

- **Capa de procesamiento:** En esta capa se encuentra la lógica del negocio de Vaova. Se definen los componentes BackOffice, Itinerarios, Ventas, Usuarios y Join.

- **Capa de integraciones:** Esta capa es la encargada de integrar las plataformas externas (*TourHero* y *WeTravel*) con Vaova. Define los componentes Plataformas, TourHeroSync, TourHeroDownloader y WeTravelSync.

- **Capa de notificaciones:** Esta capa se encarga de enviar las notificaciones por correo electrónico, mensajes de texto y notificaciones *push* que se producen en el sistema. Define el componente Notificaciones.

- **Capa de tableros:** Es la capa de inteligencia de negocio. Procesa la información de tal manera que se puede mostrar en tableros informativos para que los operarios de Vaova puedan tomar las decisiones que requieren. Define el componente Reportes.

El diagrama de la vista de contexto se muestra en la Figura 3.42.

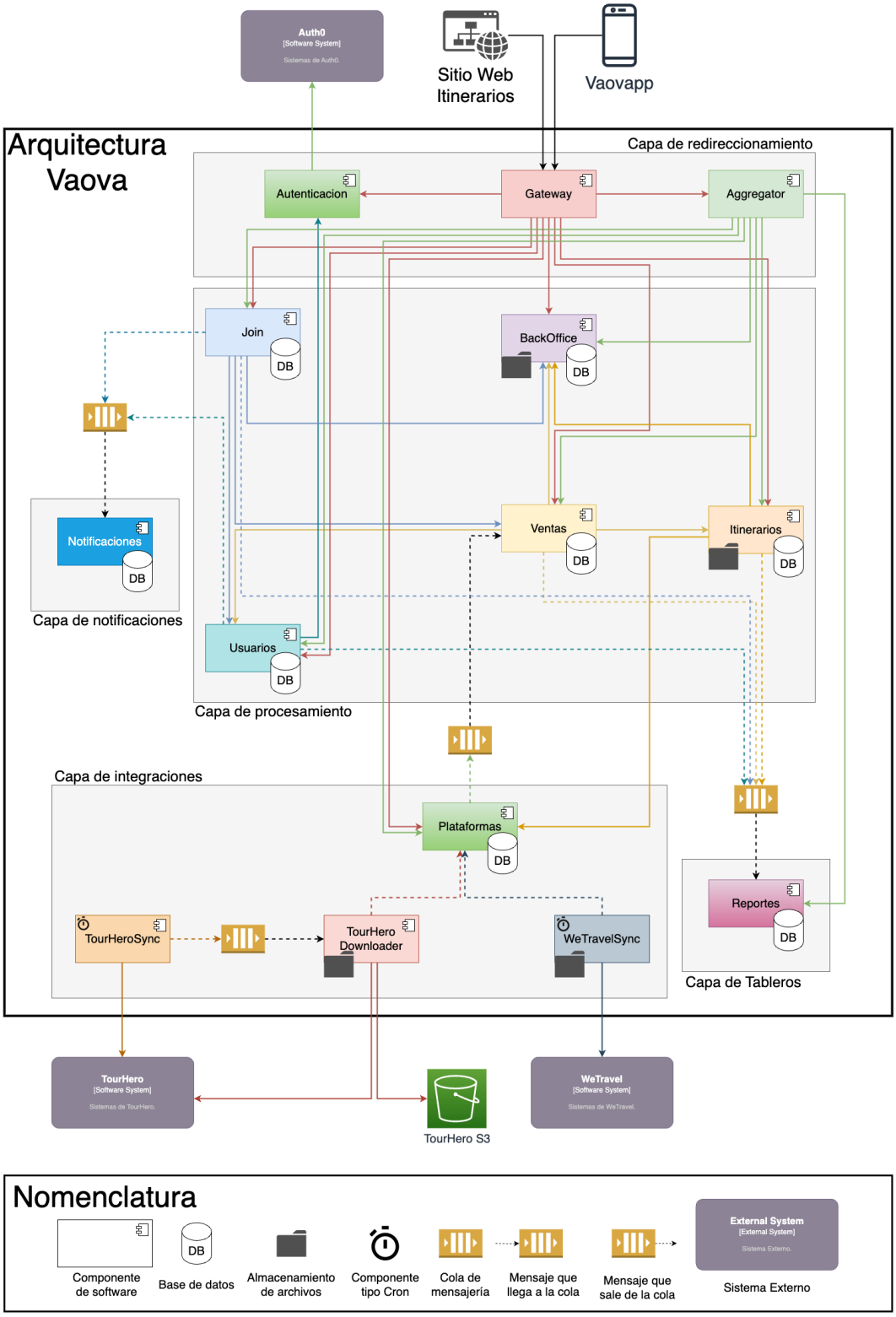


Figura 3.42: Diagrama de Contexto - Versión 1

3.5.4.2. Modelo Funcional

La Vista de Contexto define los elementos estructurales y las relaciones entre estos. Por su parte, la Vista Funcional [3] describe los elementos arquitectónicos que brindan las funciones del sistema que describe. En esta vista se documentan las funcionalidades del sistema, sus responsabilidades, las interfaces que exponen y las interacciones entre componentes (Rozanski and Woods, 2012).

Una característica muy importante de esta vista es que el sistema que se está diseñando no define tecnologías con nombres propios, como servicios específicos de proveedores tecnológicos. Los componentes se definen de manera genérica y la asignación a servicios de proveedores tecnológicos se realiza en otra vista. Los servicios externos sí pueden tener definidos nombres propios con relación a servicios de proveedores tecnológicos. La Tabla 3.41 muestra a quién va dirigida la vista y qué responde.

Vista Funcional	
Objetivo	Describe los elementos arquitectónicos que brindan las funcionalidades del sistema que describe. Se documentan las funcionalidades del sistema, sus responsabilidades, las interfaces que exponen y las interacciones entre componentes.
Dirigida a	A todas las partes interesadas.
Responde a	• Capacidades funcionales. • Relación entre componentes. • Interfaces de comunicación. • Comunicaciones internas y externas.

Tabla 3.41: Vista Funcional

Modelo Funcional del Componente *BackOffice*

El componente de *BackOffice* se encargará de administrar los subdominios de Destinos, Transportes, Acomodaciones y Actividades. Ofrecerá servicios para administrar y consultar las entidades de dichos subdominios. El componente tiene su propia base de datos y acceso a un sistema de almacenamiento de archivos para guardar las imágenes relacionadas con las entidades (estas imágenes pueden ser logos de compañías o fotografías ilustrativas). La Tabla 3.42 muestra el diseño funcional para el componente de *BackOffice*.

[3] Sólo se muestra la Vista Funcional para el componente *BackOffice*. Las vistas funcionales de los demás componentes se pueden encontrar en el documento de arquitectura en la Sección 6.6.

Vista Funcional - BackOffice				
Componente		*BackOffice*		
Versión		1.0		
Responsabilidad	Invocación			Colaboradores
	Rest	*Queue*	Tipo	
CRUD de países	X		Interno	*BackOfficeDB*
CRUD de destinos	X		Interno	*BackOfficeDB* y Sistema de almacenamiento de archivos
CRUD de aeropuertos	X		Interno	*BackOfficeDB*
CRUD de aeropuertos por destino	X		Interno	*BackOfficeDB*
CRUD de aerolíneas	X		Interno	*BackOfficeDB* y sistema de almacenamiento de archivos
CRUD de aerolíneas por destino	X		Interno	*BackOfficeDB*
CRUD de transportes	X		Interno	*BackOfficeDB*
CRUD de transporte por destino	X		Interno	*BackOfficeDB*
CRUD de actividades	X		Interno	*BackOfficeDB* y sistema de almacenamiento de archivos
CRUD de actividades por destino	X		Interno	*BackOfficeDB*
CRUD de categoría de actividades	X		Interno	*BackOfficeDB*
CRUD de *time slots* de actividades	X		Interno	*BackOfficeDB*
CRUD de hoteles	X		Interno	*BackOfficeDB* y sistema de almacenamiento de archivos
CRUD de hoteles por destino	X		Interno	*BackOfficeDB*

Tabla 3.42: Vista Funcional - BackOffice

3.5.4.3. Modelo de Despliegue

La vista de despliegue (también conocida como vista de *deployement*) describe el ambiente en el cual el sistema será desplegado, incluyendo dependencias que el sistema tiene con su ambiente en tiempo de ejecución (Rozanski and Woods, 2012).

Vista De Despliegue	
Objetivo	Describe el ambiente en el cual el sistema será desplegado, incluyendo dependencias que el sistema tiene con su ambiente en tiempo de ejecución.
Dirigida a	Equipos de arquitectura, desarrollo y DevSecOps.
Responde a	• Servicios de nube requeridos. • Permisos para comunicación interna y externa. • Segmentaciones de red. • Restricciones de seguridad que se deben implementar.

Tabla 3.43: Vista De Despliegue

Esta vista, por la restricción AC_02 de la Sección 3.5.2.5, está muy enfocada a los servicios de AWS que se van a utilizar en el despliegue final. Se utilizó una estrategia de crear diagramas enfocados en un aspecto del despliegue, porque el diagrama final era muy grande y difícil de entender. Los diez diagramas generados para la vista de despliegue son pequeños y se enfocan en un único aspecto de la arquitectura, lo que facilita su interpretación y entendimiento.

Despliegue de Red: Este diagrama muestra la infraestructura de red que se realiza dentro de la VPC en la región elegida, incluye las subredes que se derivan, grupos de seguridad y zonas de disponibilidad. La Figura 3.43 muestra el diagrama de despliegue de red.

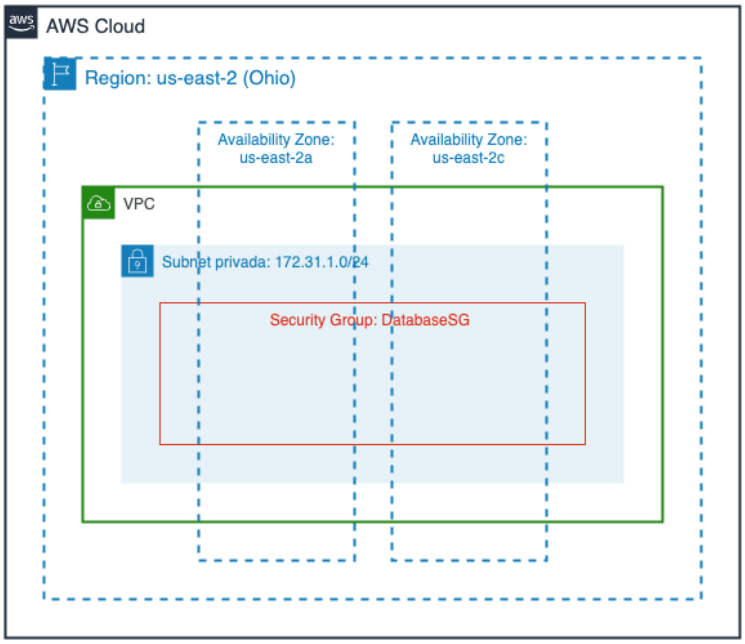


Figura 3.43: Diagrama de Despliegue de Red

Despliegue de Servidores: Este diagrama muestra cómo las instancias de RDS se ubican en las zonas de disponibilidad para realizar la estrategia de replicación de la información. La Figura 3.44 muestra el diagrama de despliegue de servidores.

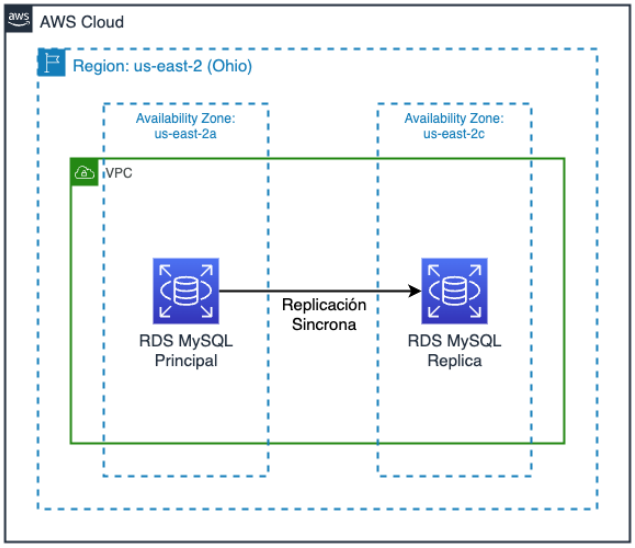


Figura 3.44: Diagrama de Despliegue de Servidores

Despliegue de Almacenamiento: Este diagrama muestra todos los servicios que se utilizaran para el almacenamiento de la información. En este caso se utilizan RDS para las bases de datos y S3 para almacenar archivos. La Figura 3.45 muestra el diagrama de despliegue de almacenamiento.

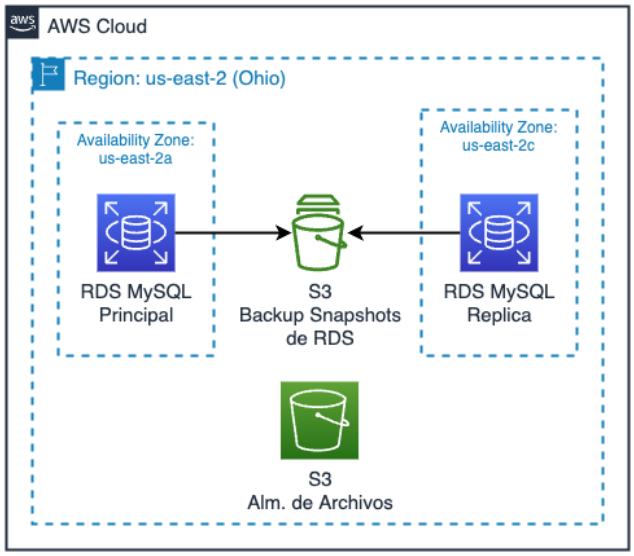


Figura 3.45: Diagrama de Despliegue de Almacenamiento

Despliegue de Componentes y Dependencias: Este diagrama muestra todos los componentes que se deben desarrollar, y las dependencias que tienen con servicios de bases de datos (RDS), colas de mensajería (SQS), servicios de notificaciones (SNS) y administración de parámetros y secretos (*System Manager Parameter Store*). La Figura 3.46 muestra el diagrama de componentes y dependencias.

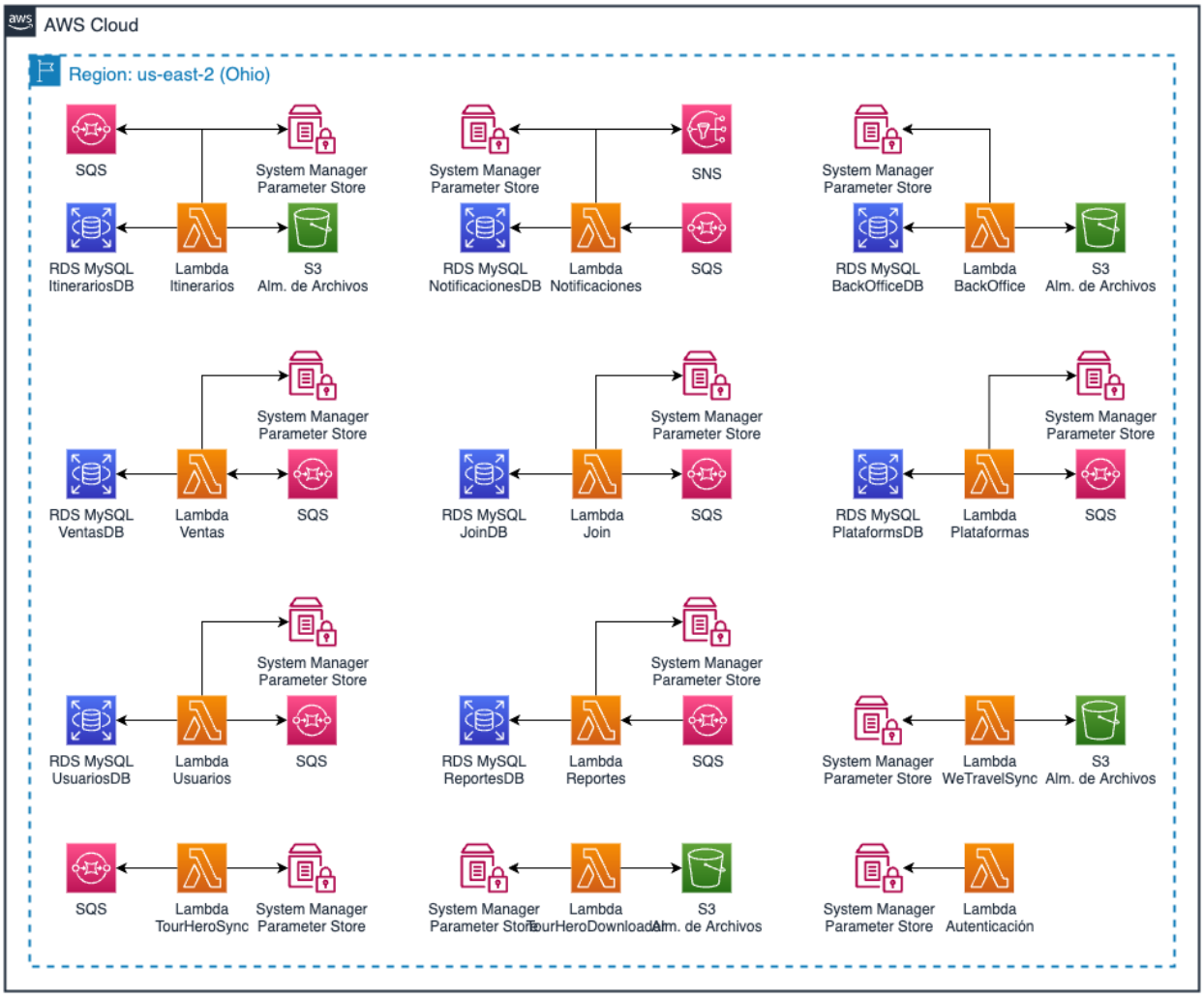


Figura 3.46: Diagrama de Despliegue de Componentes y Dependencias

Diagrama de Despliegue de Comunicación Interna Síncrona: Este diagrama muestra las llamadas internas que se realizan entre los componentes de la arquitectura de manera síncrona (REST). Se omiten las llamadas externas porque tendrá su propio diagrama. La Figura 3.47 muestra el diagrama de comunicación interna síncrona.

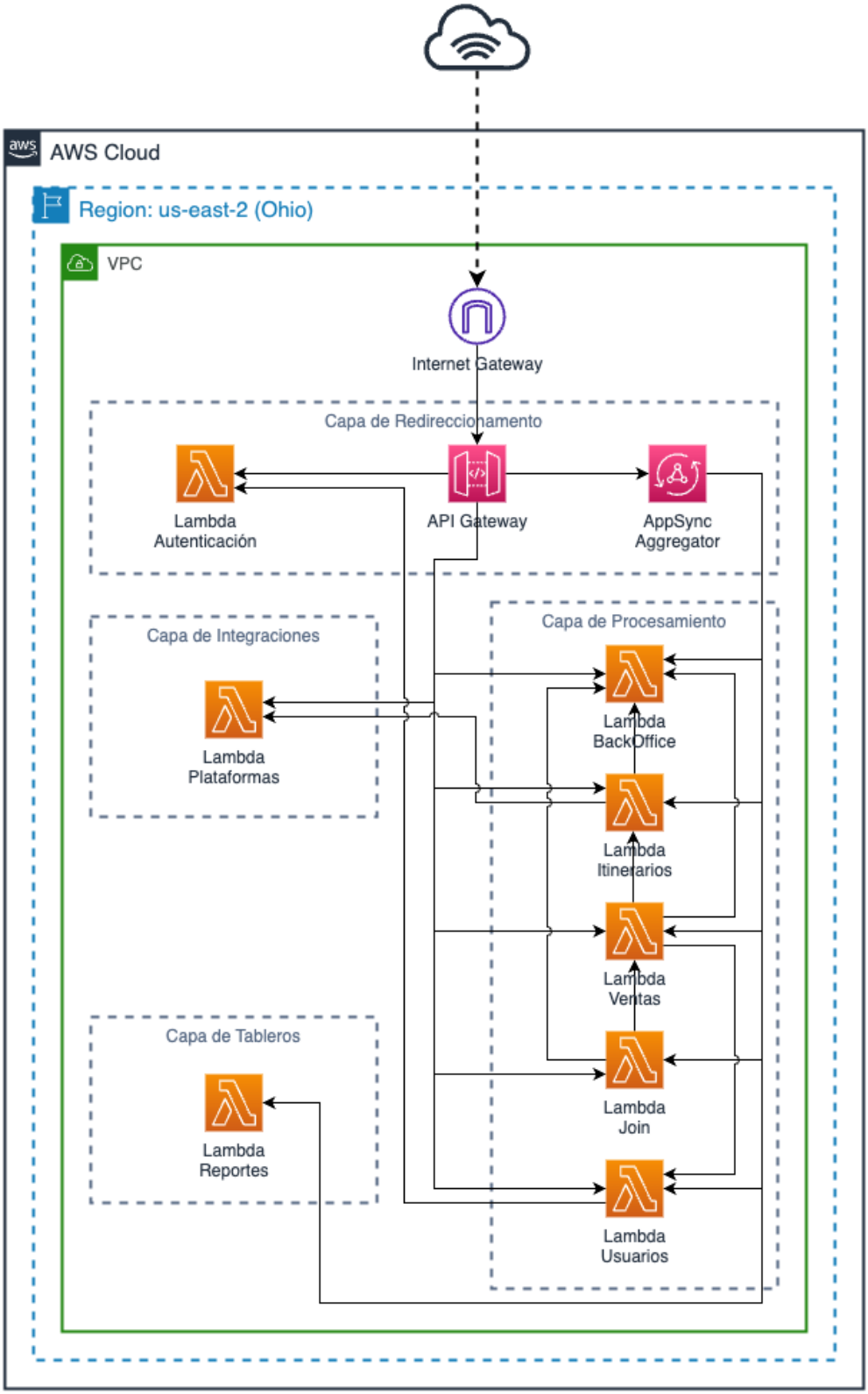


Figura 3.47: Diagrama de Despliegue de Comunicación Interna Síncrona

Diagrama de Comunicación Interna Asíncrona: Este diagrama muestra las llamadas internas que se realizan entre los componentes de la arquitectura de manera asíncrona (colas de mensajería). La Figura 3.48 muestra el diagrama de comunicación interna asíncrona.

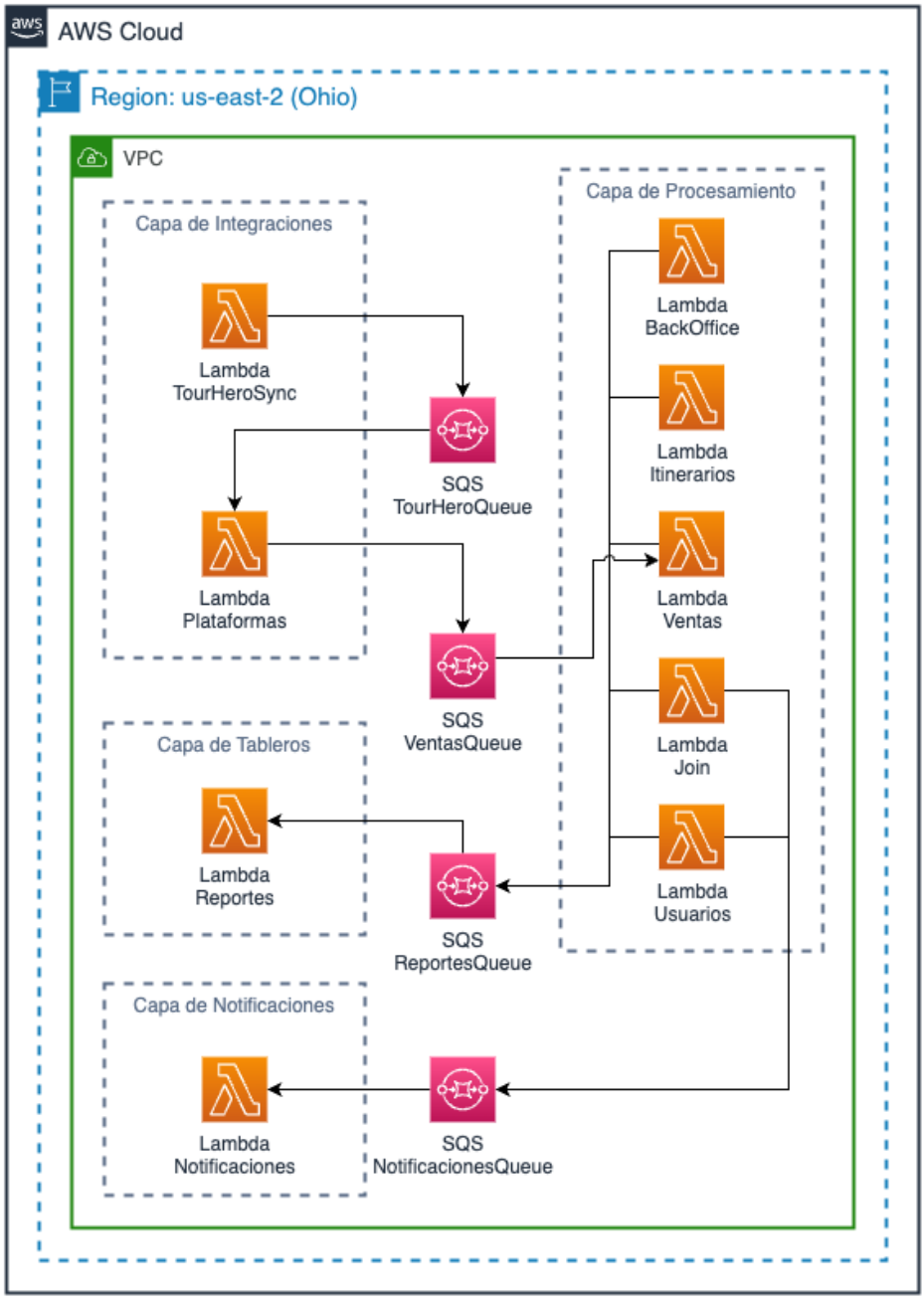


Figura 3.48: Diagrama de Despliegue de Comunicación Interna Asíncrona

Diagrama de Despliegue de Comunicación por Eventos: Este diagrama muestra las llamadas internas que se realizan entre los componentes por medio de la activación de eventos configurados (Event Bridge). La Figura 3.49 muestra el diagrama de comunicación por eventos.

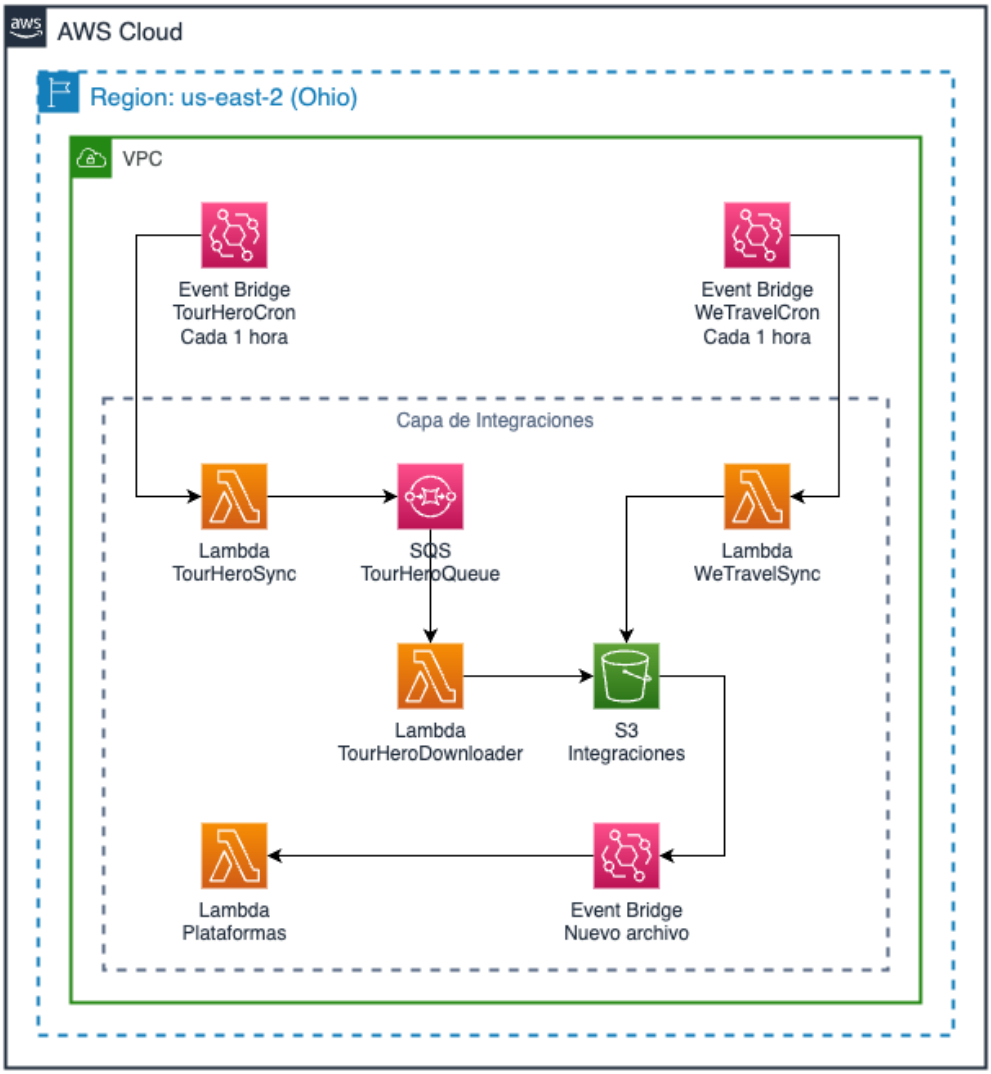


Figura 3.49: Diagrama de Despliegue de Comunicación Interna por Eventos

Diagrama de Despliegue de Comunicación Externa: Este diagrama muestra las llamadas externas que realizan los componentes de la arquitectura de manera síncrona (REST). La Figura 3.50 muestra el diagrama de comunicación externa.

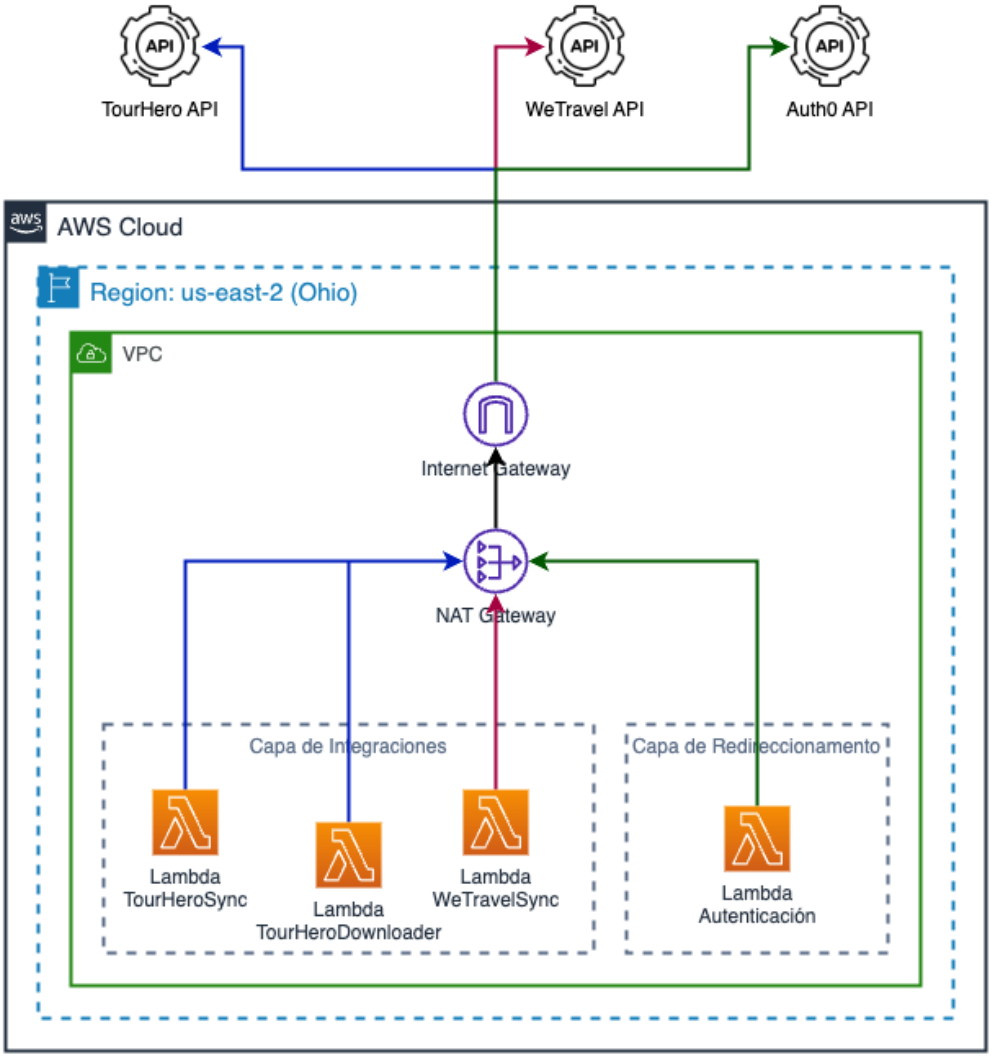


Figura 3.50: Diagrama de Despliegue de Comunicación Externa

Diagrama de Despliegue de Monitoreo y Registro: Este diagrama muestra la manera en las que los componentes registran las transacciones en la cuenta de AWS (CloudTrail) y la manera en la que guardan los logs transaccionales (CloudWatch). La Figura 3.51 muestra el diagrama de monitoreo y registro.

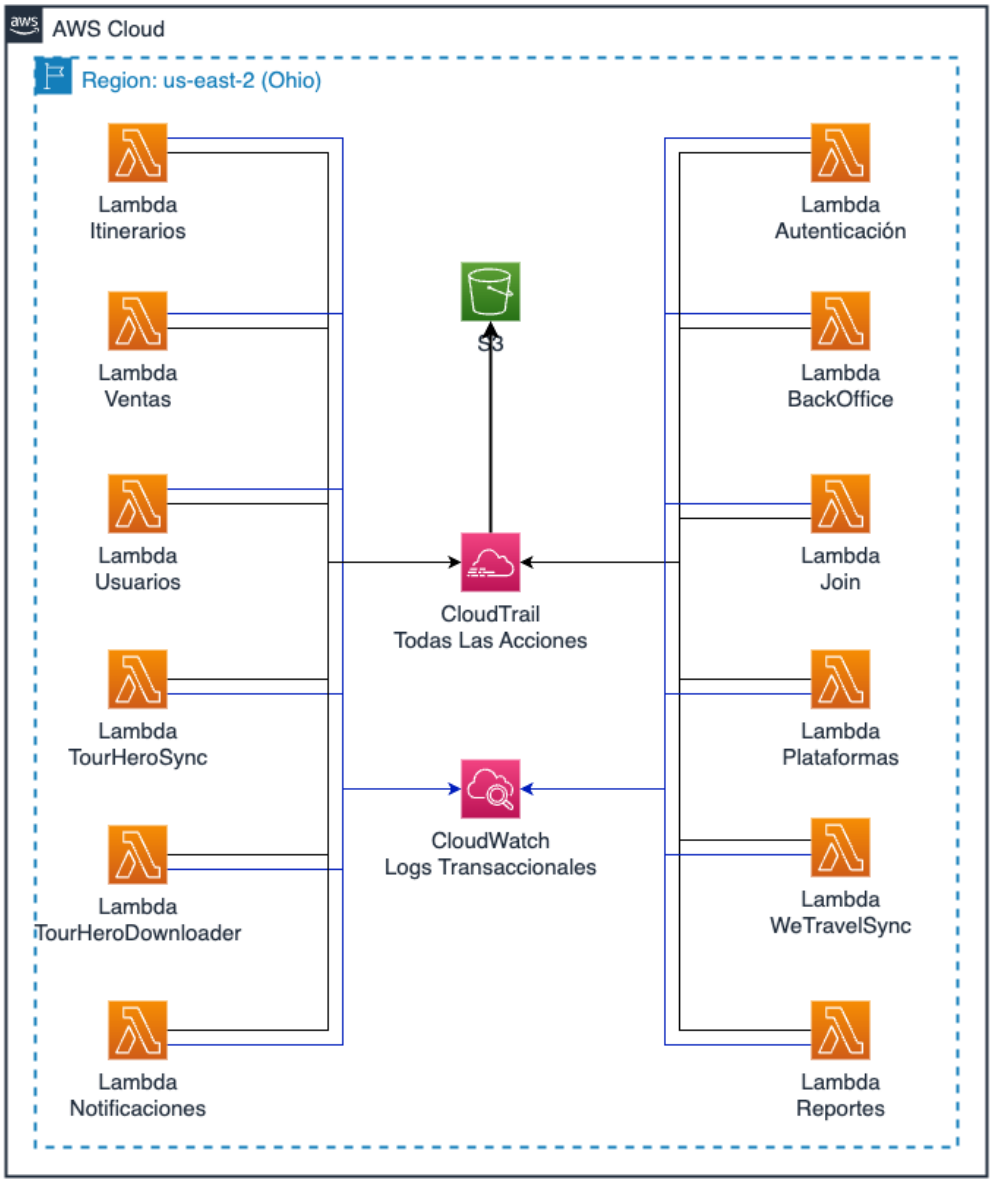


Figura 3.51: Diagrama de Despliegue de Monitoreo y Registro

Diagrama de Despliegue de Seguridad: Este diagrama muestra la manera en la que se configuran los elementos de seguridad dentro de la arquitectura. Se muestran los servicios de Route53, Shield, WAF, Parameter Store, grupos de seguridad, las subredes y la comunicación con *Auth0*. La Figura 3.52 muestra el diagrama de seguridad.

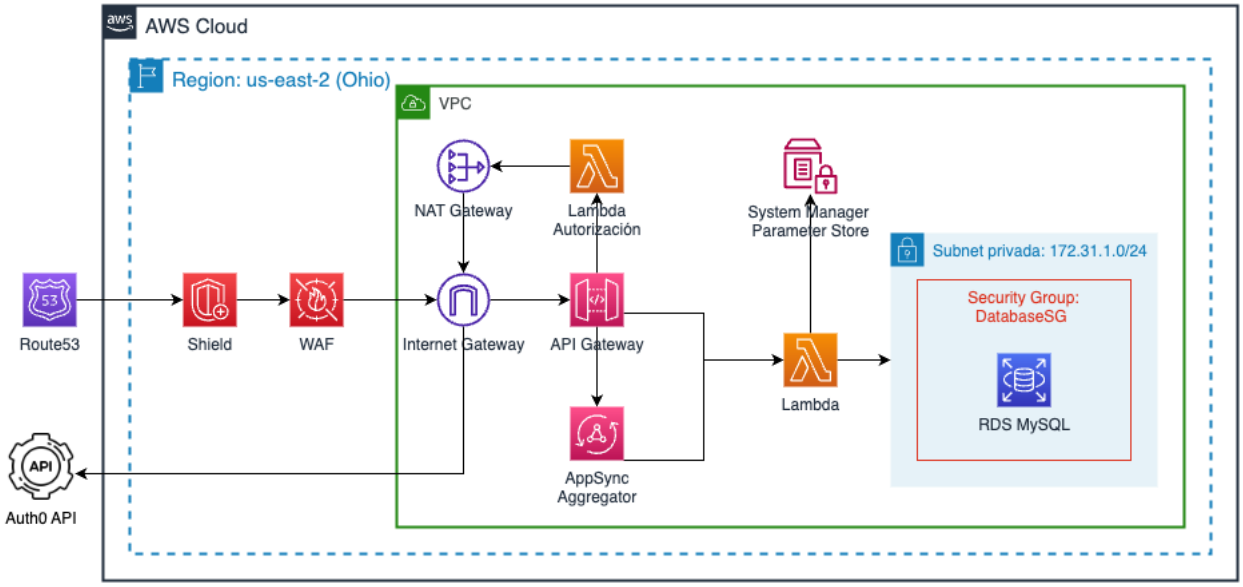


Figura 3.52: Diagrama de Despliegue de Seguridad

3.5.4.4. Modelo de Información

La vista de información describe la manera en la que la arquitectura almacena, manipula, administra y distribuye la información entre los componentes de software (Rozanski and Woods, 2012). La Tabla 3.44 muestra qué resuelve y hacia quién va dirigida la vista de información.

Vista De Información	
Objetivo	Describe la manera en la que la arquitectura almacena, manipula, administra y distribuye la información entre los componentes de software.
Dirigida a	Equipos de arquitectura, desarrollo y DevSecOps.
Responde a	• Estructura y contenido de la información. • Flujo de la información. • Cantidad y volumen de información. • Políticas de retención de datos.

Tabla 3.44: Vista De Información

En el diseño de la vista de información para la arquitectura de software, se utilizaron los siguientes elementos para generar la vista de información. Estos elementos relacionan el origen de la

información, los servicios o componentes que la procesan y la manera en la que se almacena. La Figura 3.54 muestra el diagrama general del catálogo de componentes y entidades de información.

- **Catálogo de Componentes de Información:** Los componentes de información son las herramientas y aplicaciones que se encargan de administrar la información generada o recibida por el sistema. La Tabla 3.45 muestra los componentes de información identificados para la arquitectura de software.

Componente	Ubicación	Propósito
Git	GitHub	Es el código que se utiliza para desplegar las funciones Lambda en AWS.
AWS Lambda	AWS	Ejecución de las funciones Serverless.
AWS Parameter Store	AWS	Almacenamiento de variables de entorno y secretos.
AWS RDS	AWS	Base de datos relacional.
AWS S3	AWS	Almacenamiento de archivos en la nube.
AWS DynamoDB	AWS	Base de datos NoSQL del tipo Key-Value Store.
AWS SQS	AWS	Sistema de colas de mensajería en la nube.
AWS SNS	AWS	Sistema de notificaciones en la nube.
AWS CloudWatch	AWS	Sistema de registro de logs transaccionales de la nube de AWS.

Tabla 3.45: Componentes de Información

- **Catálogo de Entidades de Información:** Las entidades de información son las fuentes internas o externas que generan o reciben la información que el sistema debe procesar. La Tabla 3.46 muestra las entidades de información identificadas para la arquitectura de software.

Entidad	Ubicación	Propósito
Código fuente	GitHub	Es el código que se utiliza para desplegar las funciones Lambda en AWS.
Variables de entorno	AWS Parameter Store	Datos de configuración que se utilizan para configurar las funciones Lambda.
Data Entry	AWS RDS	Son los valores que los funcionarios de Vaova generan para configurar destinos, transportes, acomodaciones y usuarios.

Continúa en la siguiente página

Tabla 3.46 – Continuación de la página previa

Entidad	Ubicación	Propósito
TourHero	AWS S3	Archivos en Excel que contienen la información de viajeros y ventas realizadas por medio de la plataforma de TourHero.
WeTravel	AWS S3	Archivos de Excel que contienen la información de viajeros y ventas realizadas por medio de la plataforma de WeTravel.
Auth0	AWS DynamoDB	Tokens generados por Auth0 para autenticar y autorizar a los usuarios y acciones en el sistema.
Eventos	AWS SQS	Mensajes que se envían por colas de mensajería para realizar procesamiento de manera asíncrona.
Notificaciones	AWS SNS	Notificaciones por correo electrónico, mensajes de texto y notificaciones push que se envían a los usuarios del sistema.
Secretos	AWS Parameter Store	Son variables de entorno sensibles de ser reveladas que se utilizan para la configuración de las funciones Lambda. Se pueden encontrar cadenas de conexión a bases de datos, credenciales para comunicarse con servicios de terceros, etc.
Logs Transaccionales	AWS CloudWatch	Son los registros de las transacciones que dejan todos los servicios de AWS, entre ellos las funciones Lambda. Sirven para determinar las acciones realizadas en los servicios o establecer las causas de posibles errores.

Tabla 3.46: Entidades de Información

- **Modelos de Entidad-Relación:** Los modelos Entidad-Relación [4]. muestran las entidades, atributos y relaciones de la información. Para el diseño de esta arquitectura, se complementan con diagramas de estado que muestran el flujo de estados de la información. Las Figura 3.53 y la Tabla 3.47 muestran el diagrama Modelo-Entidad y la secuencia de estados de la entidades del subdominio de Itinerarios.

[4]Sólo se muestra el modelo entidad-relación del subdominio de Itinerarios. Los modelos entidad-relación de los demás subdominios se pueden encontrar en la Sección 6.6

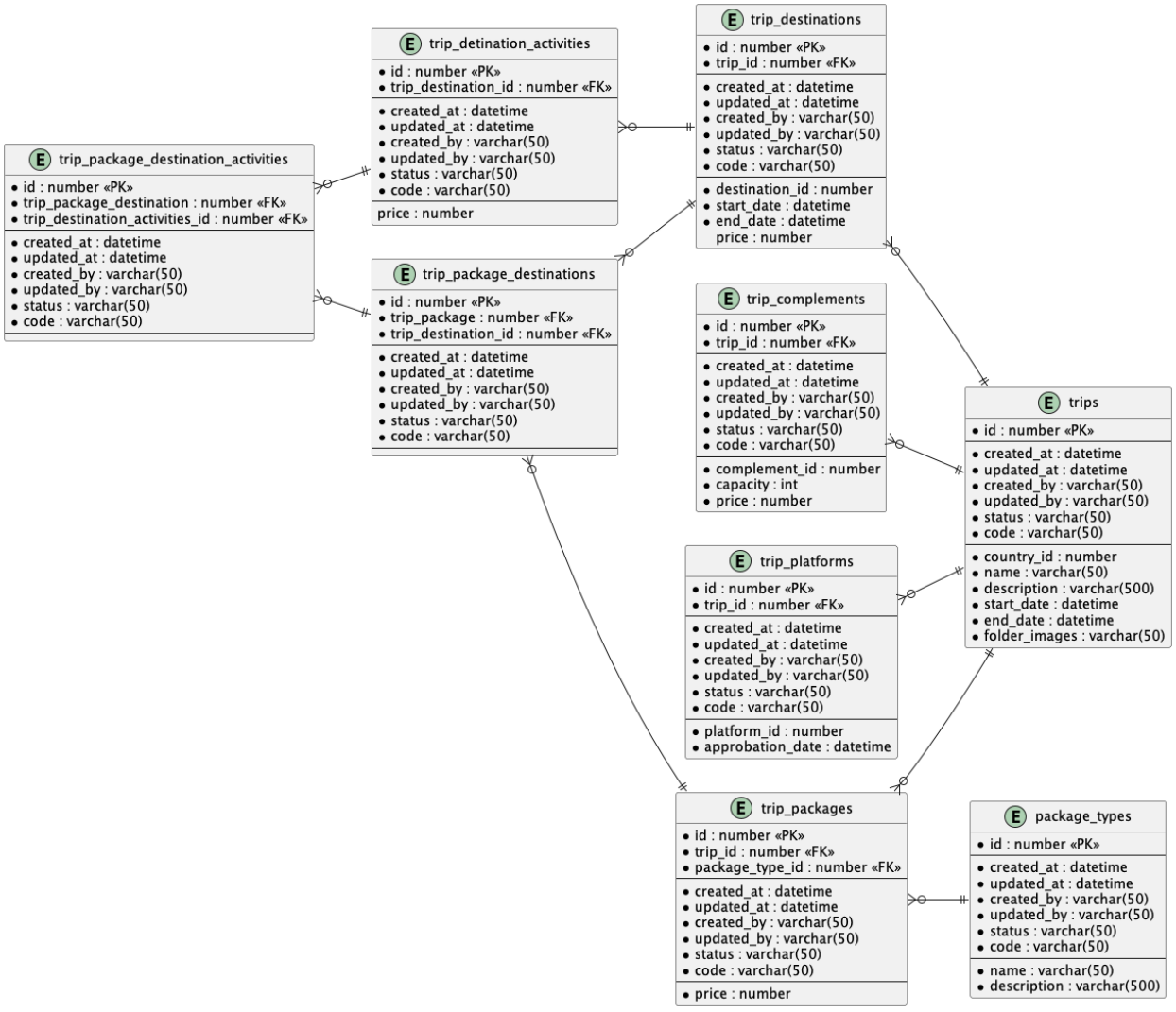
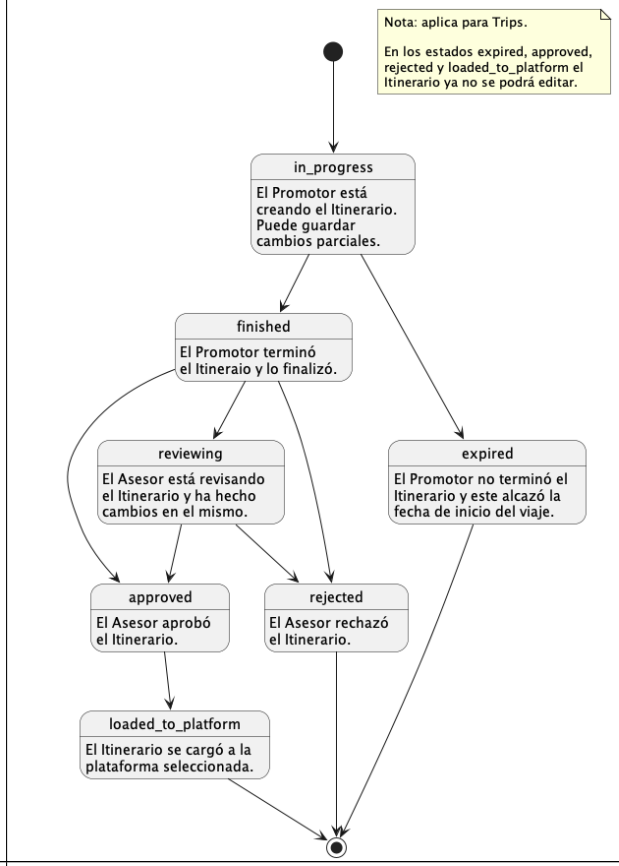
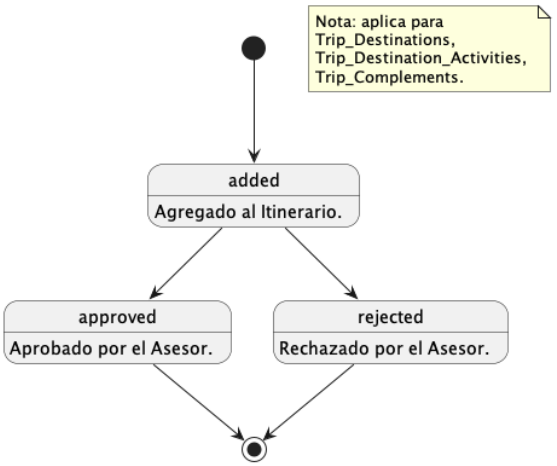


Figura 3.53: Diagrama Entidad-Relación de Itinerarios

Secuencia de Estados de Entidades - Itinerarios	
Entidad	***Secuencia de Estados***
	Continúa en la siguiente página

Tabla 3.47 – Continuación de la página previa

Secuencia de Estados de Entidades - Itinerarios	
Trips	
Trip_Destinations, *Trip_Destination_Activities,* *Trip_Complements*	

Continúa en la siguiente página

Tabla 3.47 – Continuación de la página previa

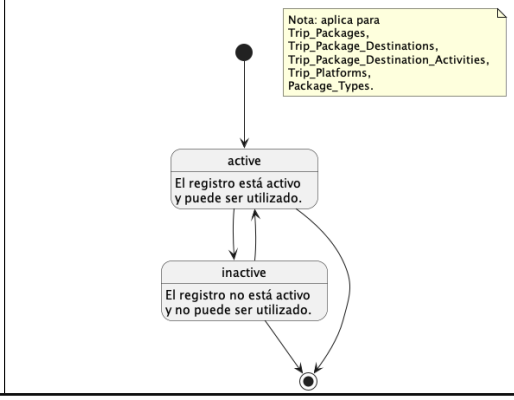
Secuencia de Estados de Entidades - Itinerarios	
Trip_Packages, Trip_Package_Destinations, Trip_Package_Destination_Activities, Trip_Platforms, Package_Types	

Tabla 3.47: Secuencia de Estados de Entidades - Itinerarios

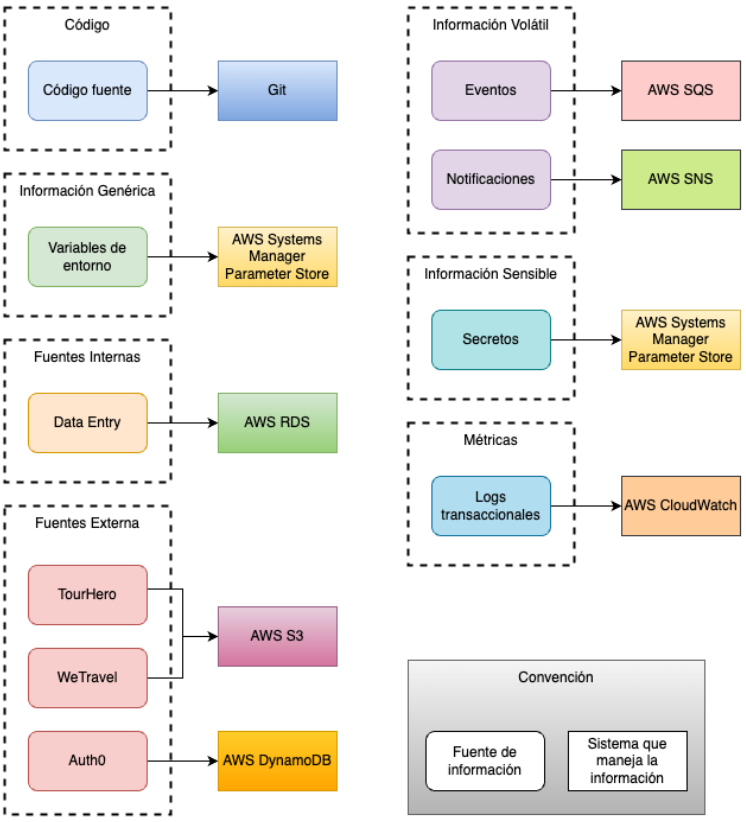


Figura 3.54: Diagrama de Fuentes y Componentes de Información

3.5.5. Modelo DevSecOps

La última sección en el documento de arquitectura de software son recomendaciones sobre el modelo *DevSecOps* que Vaova debería implementar junto a la arquitectura de software. Se incluyó este aparto debido a que la compañía no cuenta con un modelo maduro de *DevOps* o *DevSecOps*. Vale la pena aclarar que sólo son recomendaciones y la implementación queda a total disposición de Vaova.

Antes de iniciar, se debe definir qué es *DevSecOps*. Myrbakken and Colomo-Palacios (2017) en su artículo de revisión de literatura realizaron un estudio sobre *DevSecOps* y llegaron a las siguientes definiciones:

- ***DevOps*:** Se describe como la "*fusión conceptual y operativa de las necesidades, equipos y tecnologías de desarrollo y operaciones*". Esta fusión tiene como objetivo alinear las prioridades de los equipos de desarrollo y operaciones para que trabajen juntos hacia un objetivo común, cooperando con el desarrollo de software e implementando ese software en producción. Esto se puede realizar involucrando al equipo de operaciones en todas las etapas de desarrollo, mediante la colaboración de desarrolladores y operadores para resolver problemas, crear procesos y productos que puedan **automatizarse** y acordar desarrollar métricas que todos puedan utilizar. Lo anterior refleja los cuatro principios fundamentales de *DevOps*:

  - *Cultura:* Una cultura *DevOps* promueve la colaboración entre los equipos de desarrollo y operaciones, donde todos aceptan que son responsables de entregar software al usuario final.
  - *Automatización:* La automatización de construir, desplegar y probar es importante para alcanzar rápidamente el desarrollo, despliegue y retroalimentación de los usuarios finales.
  - *Medición:* Las mediciones incluyen el monitoreo de métricas de negocio como los ingresos y los indicadores clave de rendimiento (como el efecto que tienen los nuevos lanzamientos en la estabilidad de un sistema) para conocer el estado actual y descubrir cómo mejorarlo.
  - *Uso compartido:* Los desarrolladores y operadores comparten el conocimiento, herramientas de desarrollo, y técnicas para manejar los procesos.

  Aunque la implementación de *DevOps* se ha vuelto muy importante, esta no contempla prácticas de seguridad. Muchos gerentes, desarrolladores y operadores consideran que la seguridad es un obstáculo de la agilidad y velocidad requeridas en las prácticas *DevOps*.

- ***DevSecOps*:** Es un nuevo concepto que soluciona las necesidades de seguridad en el modelo de *DevOps*. El principal objetivo de *DevSecOps* es crear e incluir prácticas de seguridad modernas que puedan incorporarse en el rápido y ágil mundo de *DevOps*. Es una extensión del objetivo de *DevOps* de promover la colaboración entre desarrolladores y operadores involucrando también a expertos en seguridad desde el principio. *DevSecOps* comparte los mismos cuatro principios de *DevOps*, pero incluye uno adicional:

- *Cultura:* Incluye colaborar con el equipo de seguridad, así como promover una cultura en la que las operaciones y el desarrollo también trabajan para integrar la seguridad en su trabajo. Esto significa involucrar al equipo de seguridad desde las etapas de planificación, y asegurando que todos acepten que la seguridad es responsabilidad de todos.
- *Automatización: DevSecOps* también promueve un enfoque en la automatización de la seguridad, para poder mantenerse al día con la velocidad y la escala lograda por *DevOps*. El objetivo debería ser la automatización del 100 % de los controles de seguridad, donde los controles puedan implementarse y gestionar sin interferencia manual. Es importante implementar la seguridad automática de una manera que no obstaculice la agilidad de *DevOps* de ninguna manera.
- *Medición:* Promueve el uso y desarrollo de métricas que rastrea amenazas y vulnerabilidades durante todo el proceso de desarrollo de software. Los controles de seguridad automáticos durante todo el proceso de desarrollo de software significa que hay métricas disponibles para rastrear amenazas y vulnerabilidades en tiempo real y eso permite a la organización verificar qué tan buena es una aplicación bajo demanda.
- *Uso compartido:* Promueve la inclusión del equipo de seguridad en el intercambio promovido en un entorno *DevOps*. Al informar al equipo de seguridad sobre los desafíos que enfrentan los operadores y desarrolladores, y viceversa, se mejorarán los procesos de seguridad que se desarrollan.
- *Desplazar la seguridad hacía la izquierda:* En los procesos de desarrollo de software tradicionales, la seguridad es un paso de cierre al final del proceso. *DevSecOps* promueve un desplazamiento a la izquierda para la seguridad, donde se debe incluir desde el principio en todos los pasos del proceso de desarrollo de software. Esto significa que el equipo de seguridad es incluido desde el primer paso de planificación y es parte de la planificación de cada iteración del ciclo de desarrollo. También significa que la seguridad está allí para ayudar a desarrolladores y operadores en cuestiones de seguridad.

Entre las recomendaciones para el modelo *DevSecOps* [5] que Vaova debería implementar se encuentran los siguientes apartados:

- **Herramientas de desarrollo:** Se describen las herramientas sugeridas para utilizar durante el desarrollo del sistema. Se encuentran herramientas como VisualStudio Code, DBeaver, Python 3, Poetry, Pytest, Karate, Postman y Mockoon.
- **Herramientas de integración continua:** Se recomiendan los servicios para la integración continua. GitHub, GitHub Actions y SonarCloud son las herramientas que se pueden utilizar para lograr la integración continua en un primer paso.
- **Servicios de nube:** Se describen todos los servicios en la nube de AWS que son necesarios para el desarrollo de la arquitectura de software.

[5] De nuevo, sólo se muestra un breve resumen de lo que se encuentra en en documento de arquitectura de software. Los detalles del modelo *DevSecOps* completo se pueden encontrar en la Sección 6.6.

- **Herramientas de documentación:** Son las herramientas que el equipo de arquitectura y de desarrollo pueden utilizar para realizar la documentación de arquitecturas, código y documentos relacionados con el software. También se incluyen herramientas para realizar diagramas.
- **Herramientas de seguridad:** Se ofrece un listado de todos los servicios que AWS ofrece para aumentar la seguridad dentro de su nube. Asimismo, algunos servicios evitar ataques de denegación de servicio y disminuir el riesgo de pérdida de información. Se incluye el servicio de *Auth0*, ya que es una restricción de la arquitectura (AC_01).
- **Ambientes separados:** Se describen las ventajas que ofrece tener ambientes separados para desarrollo, aceptación y producción. Se describen las regiones en las que estos deberían ser desplegados y las zonas de disponibilidad que AWS ofrece en cada región.
- **Estrategia de integración continua:** Se describen las diferentes estrategias que se deberían adoptar para crear ramas en Git, realizar los *commits* y crear los *pull requests*.
- **Plan de despliegue:** Se detallan todos los pasos que se deberían seguir para preparar un despliegue a producción y la elaboración del documento de plan de despliegue,
- **Plan de rollback:** Se detallan todos los pasos que se deberían seguir en caso de que un despliegue a producción falle y se deba realizar un *rollback* del sistema. Un aspecto a tener en cuenta en el plan de *rollback* es que se debe probar y garantizar que funciona antes de iniciar el despliegue a producción.
- **Estrategia de despliegue a producción:** Se detallan las pautas y recomendaciones que se deben seguir durante un despliegue a producción.
- **Plan de recuperación de desastres:** Se realizan recomendaciones sobre herramientas y estrategias que se pueden utilizar en caso de que sea necesario recuperarse de un fallo o desastre. En este caso, al ser una arquitectura que se despliega en la nube de AWS se puede entender un desastre como la no disponibilidad de un servicio, una zona de disponibilidad o toda una región de AWS.
- **Monitoreo continuo:** Se describen las herramientas que AWS ofrece para monitorear activamente el estado de servicios, aplicaciones y componentes de software desplegados.

Capítulo 4

Evaluación

En este capítulo se realiza la evaluación de la arquitectura de software, tal como se planteó en el cuarto objetivo específico (Sección 1.4.2). Teniendo en cuenta dicho objetivo, se realizaron dos tipos de evaluación al diseño de la arquitectura: desde el punto de vista técnico y desde el punto de vista financiero.

En la evaluación técnica se analizó el diseño para verificar que podía satisfacer los requerimientos tanto funcionales como no funcionales para los cuales la arquitectura fue diseñada. En la evaluación financiera, se analizó el nivel de inversión que sería necesario realizar y en cuánto tiempo esa inversión se podría recuperar.

Este capítulo, al igual que el Capitulo 3, muestra sólo un resumen de los resultados. El análisis completo se puede encontrar en la Sección 6.7.

4.1. Evaluación Técnica

En esta sección se muestran los resultados de la evaluación a nivel técnico. La evaluación técnica de la arquitectura se dividió en *evaluación funcional* y *evaluación no funcional*.

4.1.1. Evaluación Funcional

El propósito de la evaluación funcional [1] era verificar que el diseño de la arquitectura tuviera la estructura necesaria (componentes y comunicaciones) para satisfacer a alto nivel las historias épicas (Sección 3.4). Cada historia épica se evaluó para verificar su funcionamiento dentro del diseño de la arquitectura, para esto se definió el siguiente contrato de evaluación a nivel funcional:

- Historia épica que se evalúa.
- Determinar si la historia épica se debe satisfacer en la arquitectura.
- Diagrama de secuencia de alto nivel con los componentes que resuelven la historia épica.

El segundo punto del contrato de evaluación funcional se definió porque existen historias que están más enfocadas al comportamiento de una GUI (sistema web de Itinerarios y aplicación móvil

[1] No se mostrarán los escenarios de evaluación de todas las historias épicas, sólo 2 escenarios. La evaluación de las demás historias épicas se puede encontrar en la Sección 6.7.

Vaovapp). Como el diseño de la arquitectura sólo contempla servicios de *backend*, esas historias épicas no se podían evaluar a este nivel.

El tercer punto del contrato de evaluación funcional es un diagrama de secuencia. Este diagrama muestra el recorrido que se debe realizar por los componentes definidos para lograr cumplir la historia épica. El diagrama de secuencia no entra mucho en detalle sobre validaciones o lógica de negocio (sólo algunos, donde se consideró necesario), ya que es una evaluación funcional a nivel arquitectónico y no una evaluación a bajo nivel donde se requiere mucho más detalle.

A continuación, se muestran los escenarios de evaluación funcional para las historias épicas de *Crear Usuario* (Tabla 4.1) y *Rechazar Invitación* (Tabla 4.2).

Evaluación Funcional - Historia Épica - Crear Usuario	
Historia Épica	**¿Se debe cumplir en la arquitectura?**
Crear Usuario	Sí

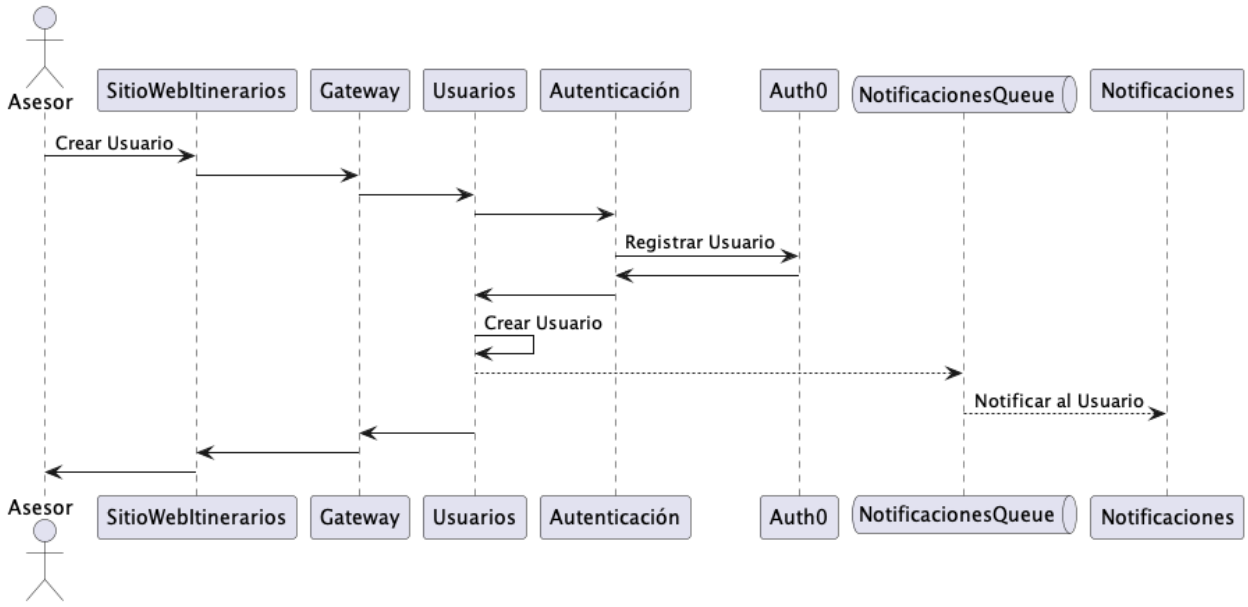


Tabla 4.1: Evaluación Funcional - Crear Usuario

Evaluación Funcional - Historia Épica - Rechazar Invitación	
Historia Épica	**¿Se debe cumplir en la arquitectura?**
Rechazar Invitación	Sí

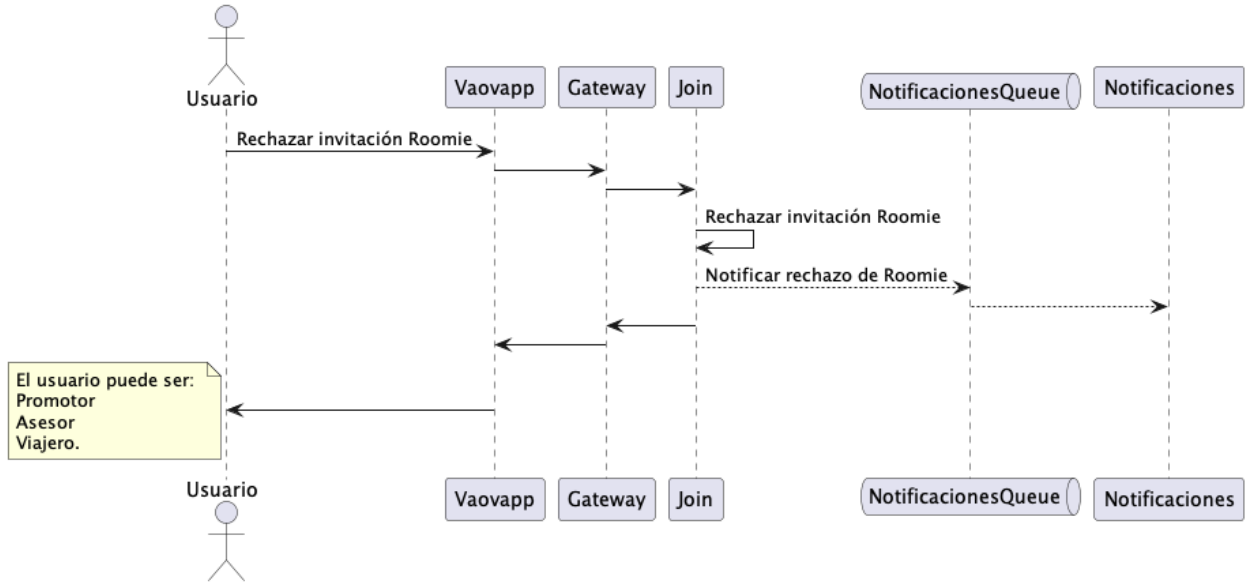


Tabla 4.2: Evaluación Funcional - Rechazar Invitación

Al realizar la evaluación funcional de cada historia épica con el diseño de la arquitectura, se puede concluir que el diseño arquitectónico ofrece la estructura necesaria, componentes y relaciones, para satisfacer las funcionalidades para la que fue diseñada. A nivel funcional no es necesario alterar o introducir cambios en el diseño de la arquitectura, ya que las historias épicas de Itinerarios, Vaovapp y Reportes se satisfacen con el diseño propuesto.

4.1.2. Evaluación No Funcional

Como se mencionó en la Sección 2.3.3, el método analítico elegido para realizar la evaluación técnica de la arquitectura de software es SAAM (*Software Architecture Analysis Method*), pero SAAM se aplica para evaluar los escenarios de calidad con el diseño de la arquitectura de software. Para realizar la evaluación técnica con SAAM, se consultaron las siguientes fuentes:

- Kazman et al. (1996): Presentan la guía general para entender el funcionamiento de SAAM. Los autores afirman que existen tres perspectivas para entender y describir arquitecturas.
    - *Funcionalidad:* La funcionalidad de un sistema es lo que el sistema hace. Esta puede ser una función simple o un conjunto de funcionalidades relacionadas que juntas describen el comportamiento general del sistema.

- *Estructura:* La estructura de un sistema de software revela cómo está construido a partir de pequeñas piezas conectadas. La estructura se describe en función de los siguientes elementos.
  1. Una colección de **componentes** que representan las entidades computacionales o repositorios de datos persistentes.
  2. Una representación de las **conexiones** entre los componentes, que es la comunicación y relaciones de control entre los componentes.
- *Asignación:* La asignación de funciones a la estructura identifica cómo las funcionalidades de dominio se realizan dentro de la estructura de software. El propósito de hacer explícita esta asignación es comprender la forma en que el sistema desarrollado logra la funcionalidad prevista.

- Patidar and Suman (2015): Los autores definen la manera de utilizar SAAM para realizar el análisis de arquitecturas de software. *SAAM fue desarrollado para ayudar a comparar soluciones arquitectónicas. SAAM realiza la evaluación de la arquitectura para identificar riesgos inherentes a la arquitectura y además es capaz de expresar múltiples atributos de calidad, como modificabilidad, portabilidad, etc.* La evaluación de una arquitectura utilizando SAAM consiste de cinco pasos:

  1. Se describen las arquitecturas candidatas.
  2. Se describen escenarios para ilustrar las actividades y cambios del sistema a lo largo del tiempo.
  3. Los escenarios se clasifican en escenarios directos o indirectos. Los escenarios directos se pueden ejecutar directamente y los escenarios indirectos requieren cambios arquitectónicos para ejecutarse.
  4. Se identifican las interacciones entre escenarios para la medición de la modularidad de la arquitectura.
  5. Cada escenario se pondera para determinar la evaluación de todas las arquitecturas.

4.1.2.1. Metodología de Evaluación

Como ya se mencionó, la metodología que se utilizó para realizar la evaluación no funcional de la arquitectura fue el modelo SAAM. Los siguientes son los criterios que se definieron a partir del modelo seleccionado:

- Todos los escenarios de calidad (Sección 3.5.2.3) se pueden evaluar con el diseño de la arquitectura. Los escenarios se clasifican como directos, indirectos o parciales dependiendo del grado en el que el escenario se puede cumplir dentro del diseño de la arquitectura.
- Si un escenario de calidad se satisface en la arquitectura sin necesidad de cambios, se denomina un *escenario directo.*

- Si un escenario de calidad no se satisface y se deben realizar cambios al diseño de la arquitectura, se denomina un *escenario indirecto*.
- Si un escenario es indirecto, se deben describir y realizar los cambios necesarios para satisfacerlo y determinar si es necesario realizar una nueva evaluación después de implementar los cambios.
- Si un escenario no depende 100 % del diseño de la arquitectura, se denomina un *escenario parcial*. Con este tipo de escenarios se debe documentar cómo el diseño de la arquitectura ayuda a satisfacerlo.

4.1.2.2. Contrato de Evaluación de Escenarios de Calidad

Para la evaluación no funcional de los escenarios de calidad se utilizó el siguiente contrato:

1. Escenario de calidad que se evalúa.
2. Definir si es un escenario directo, indirecto o parcial.
    a. Si es un escenario directo, explicar el funcionamiento de dicho escenario en el diseño de la arquitectura.
    b. Si es un escenario indirecto, explicar la razón por la cual el escenario no se satisface directamente. Luego, documentar los cambios que se deben realizar para satisfacer el escenario. Después, alterar el diseño de la arquitectura para incluir los cambios generados por el escenario indirecto. Finalmente, determinar si se requiere una nueva evaluación del escenario después de implementar los cambios.
    c. Si es un escenario parcial, aportar las razones de cómo el diseño de la arquitectura ayuda a satisfacerlo y definir el alcance que tiene el diseño de la arquitectura en el escenario.
3. Al finalizar la evaluación, mostrar un resumen con la clasificación de los escenarios y los cambios realizados en el diseño de la arquitectura.
4. Mostrar los cambios realizados en los modelos de Contexto y Despliegue.

4.1.2.3. Evaluación de Escenarios de Calidad

A continuación, se muestran los resultados de la evaluación de los escenarios de calidad. Las Tablas 4.3 a 4.12 contienen la evaluación individual de cada escenario de calidad.

Evaluación Escenario Calidad QS_01 - Adecuación-Funcional	
Escenario de Calidad	**Tipo de Escenario**
QS_01 – Adecuación-Funcional	Indirecto
¿Por qué es un escenario indirecto?	
En el diseño de la arquitectura no se encuentra ningún componente que pueda enviar recordatorios de manera automática a los Promotores para que terminen los Itinerarios que han iniciado, pero no terminado.	
Cambios en la arquitectura	
Incluir un nuevo componente que pueda acceder al listado de Itinerarios sin finalizar y enviar notificaciones al Promotor para que lo termine y pueda ser validado por el asesor comercial. Detalle del cambio: • Incluir un nuevo componente en la arquitectura que se llame Recordatorios. • Este componente será una Lambda que se active de manera automática cada día (Event Bridge). • Será un componente que no sólo funcionará para Itinerarios, sino para cualquier proceso que requiera de recordatorios automáticos para ser terminado. • La comunicación con Itinerarios será síncrona (REST) y la comunicación con el subdominio de Notificaciones será asíncrona (Cola de mensajería).	
¿El escenario de calidad requiere nueva evaluación?	
No, con los cambios descritos se cumplen las condiciones para notificar a los Promotores para que terminen el diseño de los Itinerarios y alcanzar una tasa de éxito del 70 %.	

Tabla 4.3: Evaluación Escenario Calidad QS_01

Evaluación Escenario Calidad QS_02 - Adecuación-Funcional	
Escenario de Calidad	**Tipo de Escenario**
QS_02 – Adecuación-Funcional	Indirecto
¿Por qué es un escenario indirecto?	
En el diseño de la arquitectura no se encuentra ninguna acción que notifique a los Asesores Comerciales que un Itinerario fue finalizado y está listo para ser revisado.	
Cambios en la arquitectura	
Continúa en la siguiente página	

Tabla 4.4 – Continuación de la página previa

Evaluación Escenario Calidad QS_02 - Adecuación-Funcional
Incluir una relación de Itinerarios a Notificaciones para alertar a un grupo de personas interesadas cuando un Itinerario esté finalizado y listo para ser revisado. Detalles de cambios: • Incluir una relación de comunicación asíncrona (Cola de mensajería) entre Itinerarios y Notificaciones, para enviar una notificación a los Asesores Comerciales cuando un Promotor finalice un Itinerario. • Aprovechar el nuevo componente Recordatorios para alertar de manera diaria a los Asesores Comerciales de que existen Itinerarios terminados que están pendientes de revisión.
¿El escenario de calidad requiere nueva evaluación?
No, con los cambios descritos se cumplen las condiciones para notificar a los Asesores Comerciales para que terminen la revisión de los Itinerarios que están terminados y alcanzar una tasa de éxito del 90

Tabla 4.4: Evaluación Escenario Calidad QS_02

Evaluación Escenario Calidad QS_03 - Adecuación-Funcional	
Escenario de Calidad	**Tipo de Escenario**
QS_03 – Adecuación-Funcional	Indirecto
¿Por qué es un escenario indirecto?	
En el diseño de la arquitectura se encuentra la relación entre Join y Notificaciones. Por lo tanto, se podrían enviar notificaciones a los Viajeros antes y durante el Join para que lo inicien y terminen de manera satisfactoria.	
Cambios en la arquitectura	
Si bien existe una relación entre Join y Notificaciones para alertar a los viajeros cuando un Join es creado, no existe un proceso automático que les envía notificaciones con frecuencia durante Join para recordar que deben realizarlo. Se necesita un proceso que envíe notificaciones durante el Join para que todos los viajeros lo puedan realizar. Detalle del cambio: • Utilizar el nuevo componente de Recordatorios. • Este componente debe enviar notificaciones reiteradas a los viajeros que no han terminado su Join para que estos lo terminen. • La comunicación de Recordatorios con Join será síncrona (REST) y la comunicación con el subdominio de Notificaciones será asíncrona (Cola de mensajería).	
Continúa en la siguiente página	

Tabla 4.5 – Continuación de la página previa

Evaluación Escenario Calidad QS_03 - Adecuación-Funcional
¿El escenario de calidad requiere nueva evaluación?
No, con los cambios descritos se cumplen las condiciones para notificar a los Viajeros para que realicen el Join de sus viajes y alcanzar una tasa de éxito del 90 %.

Tabla 4.5: Evaluación Escenario Calidad QS_03

Evaluación Escenario Calidad QS_04 - Interoperabilidad	
Escenario de Calidad	**Tipo de Escenario**
QS_04 – Interoperabilidad	Directo
¿Por qué es un escenario directo?	
En el diseño de la arquitectura se contempla de manera clara la integración con *Auth0* y que el componente Gateway siempre se comunica con el componente Autenticacion.	

Tabla 4.6: Evaluación Escenario Calidad QS_04

Evaluación Escenario Calidad QS_05 - Interoperabilidad	
Escenario de Calidad	**Tipo de Escenario**
QS_05 – Interoperabilidad	Directo
¿Por qué es un escenario directo?	
En el diseño de la arquitectura se contempla de manera clara la integración con *TourHero* por medio de los componentes TourHeroSync, TourHeroDownloader y Plataformas. Estos componentes se comunican con el componente Ventas (y este con el componente Usuarios) para el registro de viajeros y ventas desde *TourHero*.	

Tabla 4.7: Evaluación Escenario Calidad QS_05

Evaluación Escenario Calidad QS_06 - Interoperabilidad	
Escenario de Calidad	**Tipo de Escenario**
QS_06 – Interoperabilidad	Directo
¿Por qué es un escenario directo?	
Continúa en la siguiente página	

Tabla 4.8 – Continuación de la página previa

Evaluación Escenario Calidad QS_06 - Interoperabilidad
En el diseño de la arquitectura se contempla de manera clara la integración con *WeTravel* por medio de los componentes WeTravelSync y Plataformas. Estos componentes se comunican con el componente Ventas (y este con el componente Usuarios) para el registro de viajeros y ventas desde *WeTravel*.

Tabla 4.8: Evaluación Escenario Calidad QS_06

Evaluación Escenario Calidad QS_07 - Madurez	
Escenario de Calidad	**Tipo de Escenario**
QS_07 – Madurez	Parcial
¿Por qué es un escenario parcial?	
Hay dos maneras de medir el MTBF (tiempo medio entre fallos semanal) de la arquitectura: por los servicios de Vaova y por los servicios de proveedores. Servicios de Vaova, el MTBF semanal el MTBF se propuso en los escenarios de calidad en el documento de la arquitectura, y este se podrá medir al hacer seguimiento por medio de métricas y procesos. El cumplimiento del MTBF a nivel Vaova depende de que el sistema esté en marcha y se sigan los lineamientos establecidos. Servicios de proveedores, se puede consultar el histórico de problemas de disponibilidad que hayan sufrido AWS en la región seleccionada, GitHub y Auth0.	
Histórico de Disponibilidad de AWS en Ohio (us-east-2)	
El servicio AWS Health Dashboard proporciona los reportes de problemas de disponibilidad para los servicios que ofrece AWS en la región. A continuación, se presentan los datos de disponibilidad (06-08-2022 a 11-08-2023) para los servicios propuestos para el despliegue de la arquitectura: • Los servicios de API Gateway, Lambda, S3, SQS y SNS no tuvieron errores reportados. • RDS: el 30-06-2023 se reportó un problema se involucraba una pérdida de poder en la zona de disponibilidad 1 (us-east-2-az1). El problema afectó a algunos usuarios de los servicios de EC2 y EBS. El tiempo para detectar y resolver el problema fue de 10:16 AM PDT a 11:56 AM PDT, un total de 1 hora y 40 minutos. Por la frecuencia de incidentes (1 en un año) y el tiempo de recuperación (menos de 2 horas), AWS en la región de Ohio puede ayudar a satisfacer el MTBF propuesto en el escenario de calidad del documento de arquitectura.	

Continúa en la siguiente página

Tabla 4.9 – Continuación de la página previa

Evaluación Escenario Calidad QS_07 - Madurez
Histórico de Incidentes de GitHub
GitHub ofrece un servicio que permite conocer los incidentes de disponibilidad que han presentado sus diferentes servicios: • Agosto de 2022: 11 incidentes reportados (Codespaces, Actions, Pages, API Requests y Pull Requests). • Septiembre de 2022: 15 incidentes reportados (Packages, Issues, Actions, Pull Requests, Codespaces). • Octubre de 2022: 17 incidentes reportados (Codespaces, Actions, Issues, API Requests, Git Operations). • Noviembre de 2022: 19 incidentes reportados (Issues, Pull Requests, Actions, Codespaces, Git Operations, API Requests, Copilot, Packages, Pages, Webhooks). • Diciembre de 2022: 9 incidentes reportados (Git Operations, Codespaces, Issues, Pull Request, Git Operations, API Requests, Actions, Codespaces). • Enero de 2023: 17 incidentes reportados (Codespaces, Actions, Git Operations, Copilot, Pages, API Requests, Pull Requests, Packages). • Febrero de 2023: 19 incidentes reportados (Webhooks, Codespaces, Actions, Issues, Pull Requests, Git Operations, API Requests, code-scanning, Issues, orgs, Packages, Pages, repositories, teams). • Marzo de 2023: 20 incidentes reportados (Packages, Pages, Git Operations, Actions, Codespaces, Webhooks, API Requests, Issues, Pull Requests). • Abril de 2023: 12 incidentes reportados (Actions, Pages, Git Operations, Codespaces, Issues, Pull Requests, Webhooks, Copilot, Packages). • Mayo de 2023: 10 incidentes reportados (Actions, API Requests, Codespaces, Copilot, Git Operations, Issues, Pages, Webhooks, Pull Requests, Copilot). • Junio de 2023: 13 incidentes reportados (Actions, Pull Requests, Webhooks, Pages, Copilot, Issues, API Requests). • Julio de 2023: 11 incidentes reportados (Pull Requests, Codespaces, Git Operations, Actions, API Requests, Packages, Pages, Issues). • Agosto 11 de 2023: 4 incidentes reportados (Copilot, Git Opeations, Actions, Webhooks). GitHub es un proveedor que no es muy estable. Presenta un promedio de 13,6 errores por mes en el último año y cada mes los servicios más críticos como Actions o Pull Requests presentan problemas en su operación. Los problemas de disponibilidad de los servicios de GitHub podrían dificultar alcanzar el MTBF propuesto en los escenarios de calidad del documento de arquitectura.
Histórico de Disponibilidad de Auth0
Continúa en la siguiente página

Tabla 4.9 – Continuación de la página previa

Evaluación Escenario Calidad QS_07 - Madurez
Auth0 ofrece un servicio para conocer el estado de sus servicios y el histórico de disponibilidad de estos en los últimos 12 meses para las regiones que ofrece. En general, el tiempo de disponibilidad de los servicios es de 99,99 % para la región us-2. No se han presentado incidentes o interrupciones en los últimos 90 (11-08-2023). El nivel de disponibilidad ofrecido por Auth0 permitirá alcanzar el MTBF propuesto en los escenarios de calidad del documento de arquitectura.

Tabla 4.9: Evaluación Escenario Calidad QS_07

Evaluación Escenario Calidad QS_08 - Madurez	
Escenario de Calidad	**Tipo de Escenario**
QS_08 – Madurez	Parcial
¿Por qué es un escenario parcial?	
El MTTR semanal (tiempo medio de reparación semanal) se propuso a nivel de documentación en el documento de arquitectura de software. Este se podrá medir por medio de métricas y procesos una vez la arquitectura esté implementada y en producción. En el documento de arquitectura se definen los pasos y estrategias que se deben seguir antes de un despliegue, los planes de Rollback que se deben tener en cuenta, las políticas de Backup de los datos, los planes de recuperación, entre otros. El cumplimiento de políticas y controles para alcanzar el MTTR ya es un proceso que el equipo de desarrollo y operaciones de Vaova deben cumplir para satisfacer los indicadores propuestos. La evaluación de la arquitectura para este escenario de calidad llega hasta este punto, la siguiente medición que se debe realizar es a nivel operativo cuando el sistema esté en funcionamiento.	

Tabla 4.10: Evaluación Escenario Calidad QS_08

Evaluación Escenario Calidad QS_09 - Madurez	
Escenario de Calidad	**Tipo de Escenario**
QS_09 – Madurez	Parcial
¿Por qué es un escenario parcial?	
La ventana de mantenimiento propuesta es de 6 PM a 9 PM los lunes. Este es un proceso operativo que depende mucho del equipo de desarrollo en los momentos de despliegue al final de cada Sprint. Se puede medir este proceso con métricas cuando inicien los despliegues del sistema, no antes. En el documento de arquitectura se definieron las políticas y controles que se deben realizar para cada despliegue como horario, documento de plan de despliegue, documento de Rollback, pruebas antes y durante el despliegue, entre otros. El cumplimiento de políticas y controles para que las ventanas de mantenimiento duren sólo 3 horas ya es un proceso que el equipo de desarrollo y operaciones de Vaova deben cumplir para satisfacer los indicadores propuestos. La evaluación de la arquitectura para este escenario de calidad llega hasta este punto, la siguiente medición que se debe realizar es a nivel operativo cuando el sistema esté en funcionamiento.	

Tabla 4.11: Evaluación Escenario Calidad QS_09

Evaluación Escenario Calidad QS_10 - Disponibilidad	
Escenario de Calidad	**Tipo de Escenario**
QS_10 – Disponibilidad	Indirecto
¿Por qué es un escenario indirecto?	
Se percibe una dependencia muy fuerte con la integración de Auth0 en todas las transacciones. En caso de que Auth0 presente problemas de disponibilidad, los servicios de la arquitectura no estarán disponibles porque no se podrán realizar validaciones de autenticación y autorización en las peticiones.	
Cambios en la arquitectura	
Continúa en la siguiente página	

Tabla 4.12 – Continuación de la página previa

Evaluación Escenario Calidad QS_10 - Disponibilidad
Incluir una base de datos de tipo caché en el componente Autenticacion para guardar los tokens que retorna Auth0. Estos tokens no expiran de inmediato, si se consultan en caché se puede seguir operando hasta que Auth0 vuelva a estar disponible. La consulta en caché también ayuda a mejorar el rendimiento de las transacciones. Detalles del cambio: • En el componente Autenticacion incluir una base de datos DynamoDB para almacenar los tokens obtenidos para los usuarios. • Alterar el proceso para las transacciones: el token de Auth0 primero se deben consultar en DynamoDB. Si el token no se encuentra, entonces se consulta en Auth0.

Tabla 4.12: Evaluación Escenario Calidad QS_10

4.1.2.4. Resumen de la Evaluación de Escenarios de Calidad

La Tabla 4.13 muestra el resumen que se obtuvo después de realizar la evaluación no funcional aplicando el método SAAM.

Resumen Evaluación No Funcional		
Escenario de Calidad	**Tipo**	**Cambios realizados**
QS_01 – Adecuación-Funcional	Indirecto	Agregar componente Recordatorios con relación
QS_02 – Adecuación-Funcional	Indirecto	sínc. a Itinerarios y asínc. a Notificaciones.
QS_03 – Adecuación-Funcional	Indirecto	Agregar relación síncrona entre Join y Recordatorios.
QS_04 – Interoperabilidad	Directo	No aplica.
QS_05 – Interoperabilidad	Directo	No aplica.
QS_06 – Interoperabilidad	Directo	No aplica.
QS_07 – Madurez	Parcial	Se definen las métricas deseadas, las políticas
QS_08 – Madurez	Parcial	y controles a cumplir. El cumplimiento depende
QS_09 – Madurez	Parcial	del seguimiento que se realice.
QS_10 – Disponibilidad	Indirecto	Agregar una base de datos DynamoDB en el componente Autenticacion para almacenar los tokens obtenidos desde Auth0.

Tabla 4.13: Resumen Evaluación No Funcional

Los cambios que se deben aplicar en los modelos de Contexto y Despliegue son los siguientes:

- Agregar el componente Recordatorios, que es de tipo Cron.
- Relación síncrona entre los componentes Recordatorios e Itinerarios.
- Relación síncrona entre los componentes Recordatorios y Join.
- Relación asíncrona entre los componentes Itinerarios y Notificaciones.
- Se agrega base de datos en el componente Autenticacion.

Estos cambios son para satisfacer los escenarios indirectos QS_01, QS_02, QS_03 y QS_10 y se deben reflejar en los diagramas de estos modelos para finalizar la evaluación no funcional del diseño de la arquitectura de software.

4.1.2.5. Nuevo Modelo de Contexto

Se presenta el nuevo modelo de Contexto con los cambios aplicados después de la evaluación no funcional. Se agregó el componente Recordatorios (en color púrpura) con las relaciones hacia los componentes Itinerarios, Join y Notificaciones. La Figura 4.1 muestra el nuevo diagrama de contexto con los cambios aplicados.

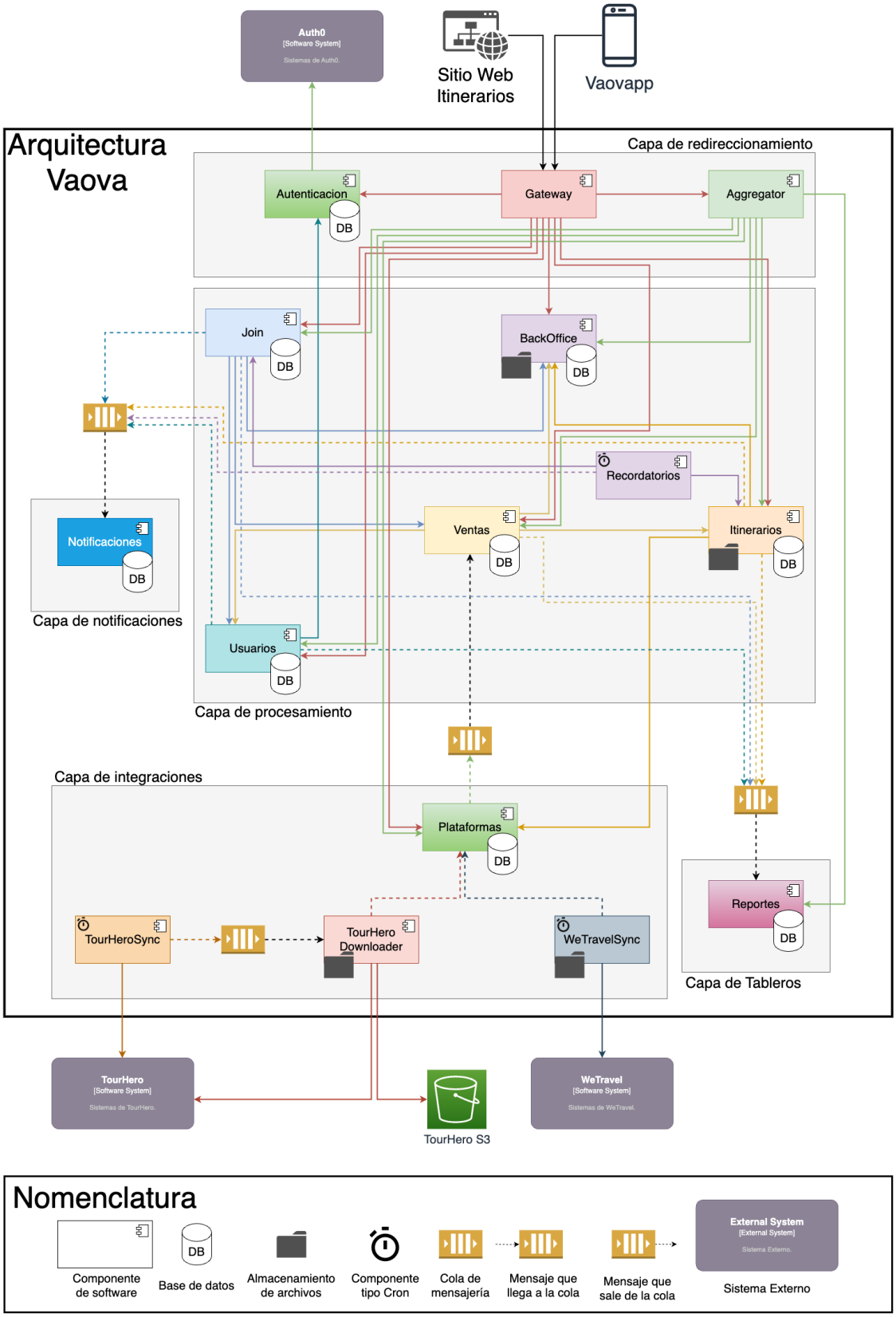


Figura 4.1: Diagrama de Contexto - Cambios Aplicados

4.1.2.6. Nuevo Modelo de Despliegue

Se presenta el nuevo modelo de Despliegue con los cambios aplicados después de la evaluación no funcional. En cada subdiagrama se muestran los cambios realizados después de la evaluación no funcional de la arquitectura.

Diagrama de Despliegue de Red

Después de la evaluación de la arquitectura, el Diagrama de Despliegue de Red no sufre cambios.

Diagrama de Despliegue de Servidores

Después de la evaluación de la arquitectura, el Diagrama de Despliegue de Servidores no sufre cambios.

Diagrama de Despliegue de Almacenamiento

En el Diagrama de Despliegue de Almacenamiento se incluye la base de datos DynamoDB. La Figura 4.2 muestra el nuevo Diagrama de Despliegue de Almacenamiento.

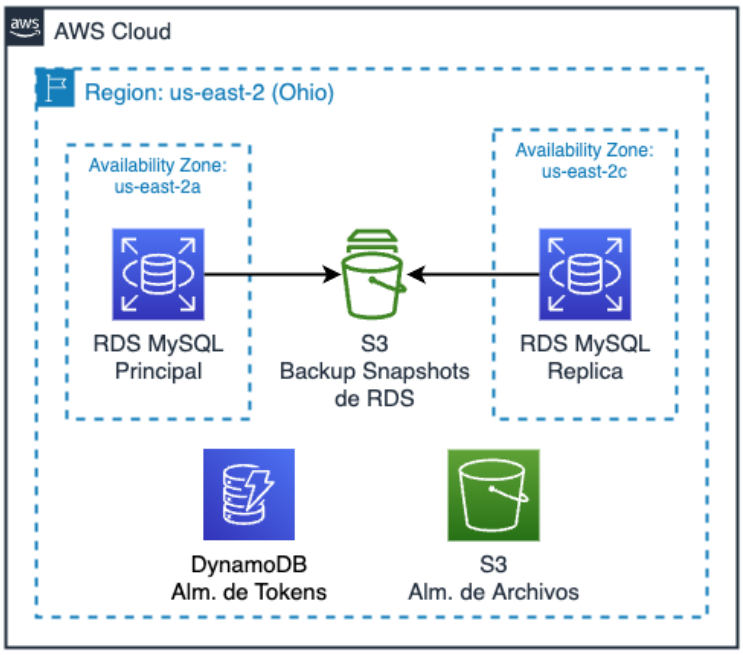


Figura 4.2: Diagrama de Despliegue de Almacenamiento - Cambios Aplicados

Diagrama de Despliegue de Componentes y Dependencias

En el Diagrama de Despliegue de Componentes y Dependencias, se agrega la base de datos DynamoDB al componente Autentication. La Figura 4.3 muestra el nuevo Diagrama de Despliegue de Componentes y Dependencias.

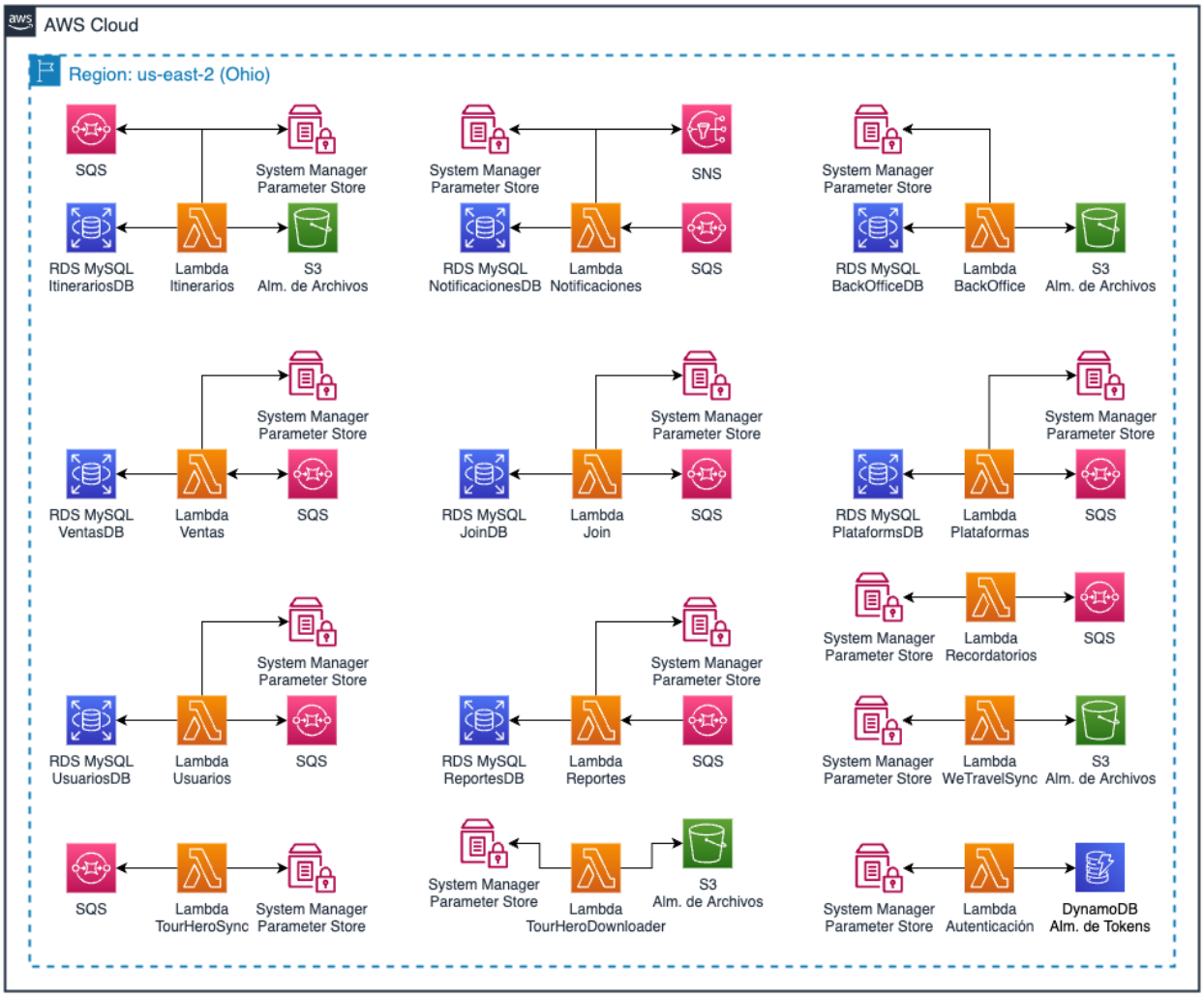


Figura 4.3: Diagrama de Despliegue de Componentes y Dependencias - Cambios Aplicados

Diagrama de Despliegue de Comunicación Interna Síncrona

En el Diagrama de Despliegue de Comunicación Interna Síncrona se agrega el componente Recordatorios y sus relaciones con los componentes de Itinerarios y Join. La Figura **??** muestra el nuevo Diagrama de Comunicación Interna Síncrona.

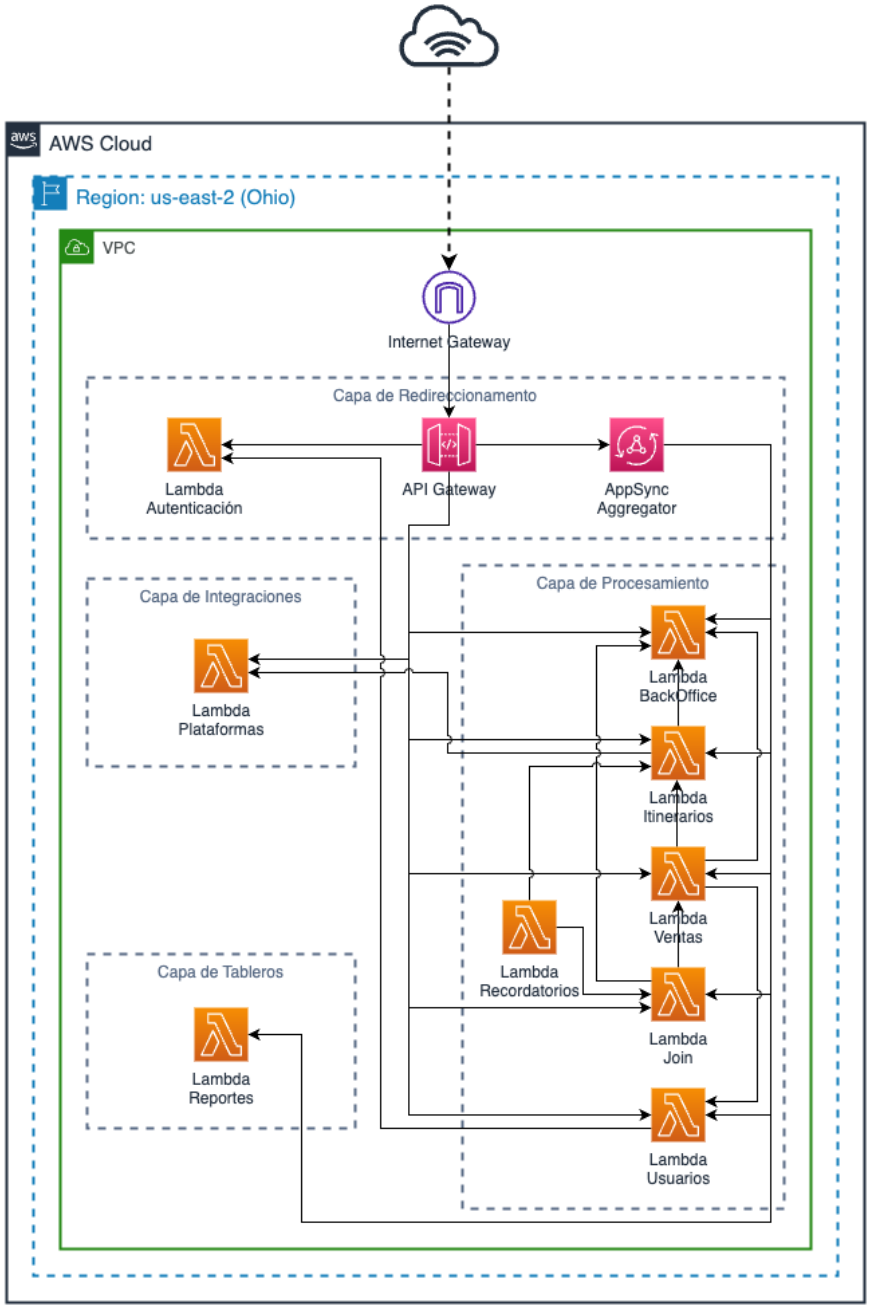


Figura 4.4: Diagrama de Despliegue de Comunicación Interna Síncrona - Cambios Aplicados

Diagrama de Despliegue de Comunicación Interna Asíncrona

En el Diagrama de Comunicación Interna Asíncrona se agrega el componente de Recordatorios y su relación con el componente de Notificaciones. Asimismo, se agrega la relación del componente

Itinerarios hacia el componente de Notificaciones. La Figura 4.5 muestra el nuevo Diagrama de Comunicación Interna Asíncrona.

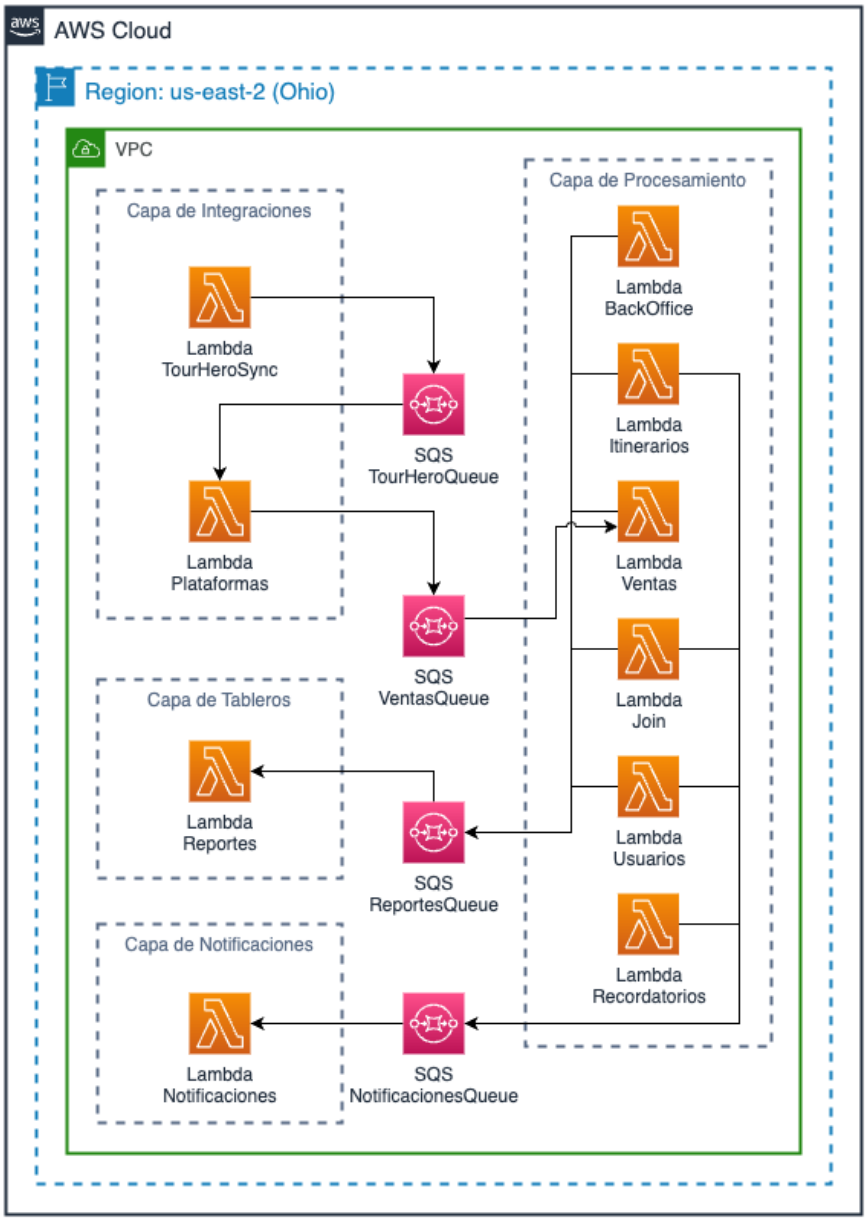


Figura 4.5: Diagrama de Despliegue de Comunicación Interna Asíncrona - Cambios Aplicados

Diagrama de Despliegue de Comunicación por Eventos

En el Diagrama de Despliegue de Comunicación por Eventos se agrega el componente Recorda-

torios. Este componente se activa por medio de un evento diario configurado en Event Bridge. La Figura 4.6 muestra el nuevo Diagrama de Comunicación por Eventos.

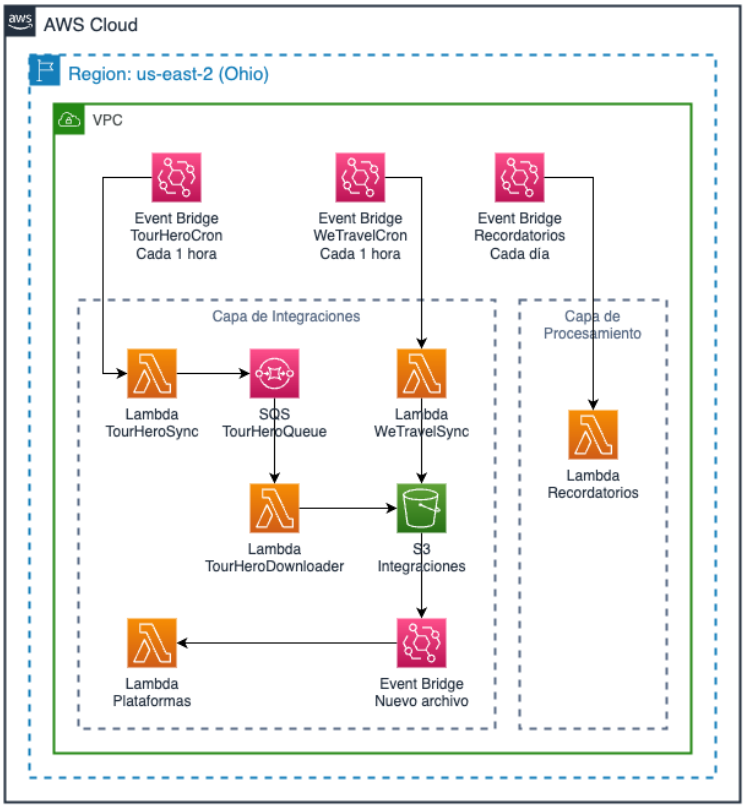


Figura 4.6: Diagrama de Despliegue de Comunicación por Eventos - Cambios Aplicados

Diagrama de Despliegue de Comunicación Externa

Después de la evaluación de la arquitectura, el Diagrama de Despliegue de Comunicación Externa no sufre cambios.

Diagrama de Despliegue de Monitoreo y Registro

Después de la evaluación de la arquitectura, el Diagrama de Despliegue de Monitoreo y Registro no sufre cambios.

Diagrama de Despliegue de Seguridad

Después de la evaluación de la arquitectura, el Diagrama de Despliegue de Seguridad no sufre cambios.

4.1.2.7. Conclusiones Evaluación a Nivel No Funcional

Después de realizar la evaluación al diseño de la arquitectura de software a nivel no funcional, se concluye que el diseño es capaz de satisfacer los escenarios de calidad definidos. Los cambios después de la evaluación fueron sólo para agregar nuevos elementos:

- Una base de datos DynamoDB para almacenar los tokens de Auth0.
- Un componente de Recordatorios para enviar notificaciones sobre los procesos que lo requerían.
- Dos relaciones síncronas, Recordatorios-Itinerarios y Recordatorios-Join.
- Dos relaciones asíncronas, Itinerarios-Notificaciones y Recordatorios-Notificaciones.

No hubo cambios en la estructura de componentes y relaciones definidas desde el inicio, lo que refleja que el diseño arquitectónico tuvo en cuenta tanto las historias épicas como los escenarios de calidad definidos.

4.2. Evaluación Financiera

El objetivo de la evaluación financiera era determinar el nivel de la inversión que tendría que hacer Vaova para implementar el diseño de la arquitectura y poner en producción el nuevo sistema. Para esto se analizaron los costos directos, indirectos y de mantenimiento de la inversión; asimismo, los beneficios generados y el tiempo que se tardaría en recuperar la inversión.

Esta evaluación se realizó a modo de caso de negocio, por lo tanto se definieron recursos necesarios, objetivos de negocio, alcance, cronograma, costos, beneficios y retorno de inversión. El documento de la evaluación financiera se puede encontrar en 6.7.

4.2.1. Descripción del Proyecto

Se presentan los objetivos, el alcance, los requerimientos y restricciones del proyecto de implementación del proyecto Arquitectura de Software Para Vaova Travel.

4.2.1.1. Objetivos de Negocio

Los objetivos de negocio del proyecto de implementación son los mismos que se definieron en la Sección 3.5.2.1:

- **BO_01** - Automatizar procesos operativos.
- **BO_02** - Mejorar la experiencia de los viajeros.
- **BO_03** - Reducir costos.

4.2.1.2. Alcance

La arquitectura de software contempla el diseño de todos los servicios e integraciones necesarios para que el sistema web de Itinerarios y la aplicación móvil Vaovapp puedan funcionar. Sólo se diseñan los servicios que estos sistemas necesitan, pero no se tiene en cuenta la propia implementación de estos sistemas.

Dentro del alcance del proyecto están los siguientes elementos:

- Servicios Backend para aplicación web de Itinerarios.
- Servicios Backend para la aplicación móvil Vaovapp.
- Integración con TourHero.
- Integración con WeTravel.
- Integración con aplicación web de Itineraios.
- Integración con aplicación móvil Vaovapp.

Por fuera del alcance del proyecto están los siguientes elementos:

- Desarrollo de aplicación web de Itinerarios.
- Desarrollo de aplicación móvil Vaovapp.

4.2.1.3. Recursos Humanos

Para el desarrollo de la arquitectura de software se necesita del siguiente talento humano:

- 2 ingenieros de desarrollo de Python con nivel Senior.
- 2 ingenieros de desarrollo de Python con nivel Semi Senior.
- 1 líder técnico.
- 1 ingeniero de DevOps.
- 1 ingeniero de QA.
- 1 Product Owner.

4.2.1.4. Recursos Tecnológicos

- Cuenta de AWS para la organización.
- Cuenta empresarial de Google Workspace.
- Cuenta empresarial de Attlassian que incluya Jira y Confluence.
- Cuenta de GitHub.
- Cuenta empresarial de Auth0.
- Cuenta empresarial de Postman.

4.2.1.5. Cronograma

Como se ha mencionado antes, esta arquitectura de software es el insumo para la aplicación web de Itinerarios y la aplicación móvil Vaovapp. Por esto, es necesario tener en cuenta que, al finalizar el desarrollo de esta arquitectura, se debe tener una fase de integración con dichas aplicaciones. El cronograma propuesto para el desarrollo de esta arquitectura se muestra en la Figura 4.7:

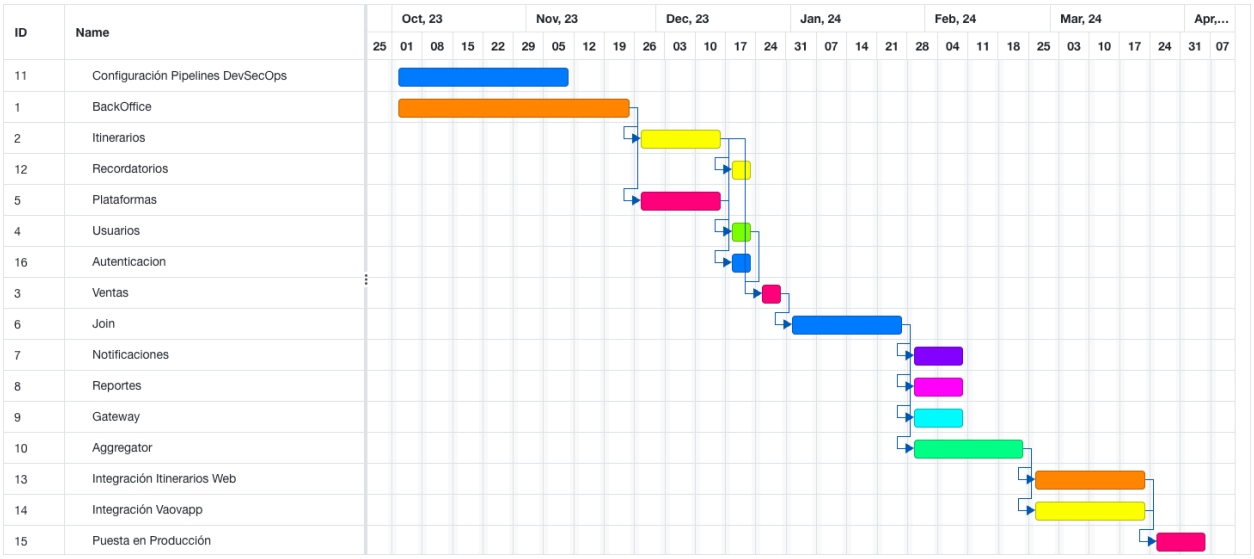


Figura 4.7: Caso de Negocio - Road Map Proyecto de Implementación

A continuación se explica el cronograma propuesto:

- Se presupuestan 2 trimestres para el desarrollo y salida a producción. Se inicia el 2 de octubre de 2023 y se finaliza el 5 de abril de 2023, con Sprints de 2 semanas de trabajo.

- En cada Sprint se cumplen las tareas de estimación, planeación, desarrollo, pruebas de desarrolladores, pruebas y certificación de QA, regresión, integración a *develop*, integración a *master*.
- Durante los Sprints se pueden planear tareas de integración con Itinerarios y Vaovapp, aunque al final se estiman 2 Sprints para hacerlo y dejar todo integrado y probado.
- Se estima un Sprint de preparación para el lanzamiento a producción final. Pero se pueden realizar liberaciones a producción según el avance de Itinerarios y Vaovapp.

4.2.2. Costos y Gastos

Se describen los costos asociados al desarrollo y mantenimiento del proyecto. Se dividen costos directos, costos indirectos y costos de operación.

4.2.2.1. Costos Directos

Son los costos en los que debe incurrir Vaova directamente para el desarrollo del proyecto. La Tabla 4.14 muestra el detalle de los costos directos asociados al proyecto.

Caso de Negocio - Costos Directos				
Recurso	**Detalle**	**Cantidad**	**Mes (USD)**	**Total (USD)**
Líder Técnico		1	$3.500,00	$3.500,00
Desarrollador Senior		2	$3.000,00	$6.000,00
Desarrollador Semi Senior		2	$2.500,00	$5.000,00
Ingeniero DevSecOps		1	$2.800,00	$2.800,00
Ingeniero QA		1	$2.500,00	$2.500,00
Licencia Auth0	Licencia B2C Essentials.	1	$23,00	$23,00
AWS API Gateway	5000 peticiones al mes.	1	$22,50	$22,50
AWS Lambda	10000 peticiones al mes. Concurrencia de 100 horas.	1	$15,69	$15,69
AWS RDS for MySQL	Tipo db.t4g.medium, Single AZ.	1	$63,31	$63,31
AWS S3	5 GB de transferencia.	1	$0,82	$0,82

Continúa en la siguiente página

Tabla 4.14 – Continuación de la página previa

Caso de Negocio - Costos Directos				
AWS DynamoDB	1 GB de almacenamiento.	1	$0,45	$0,45
AWS SQS	1 millón de peticiones.	1	$0,09	$0,09
AWS SNS	5000 peticiones al mes.	1	$0,53	$0,53
Amazon EventBridge	5000 eventos al mes.	1	$0,03	$0,03
Amazon CloudWatch	10 métricas.	1	$16,50	$16,50
AWS Systems Manager	50 configuraciones de aplicaciones.	1	$13,70	$13,70
AWS AppSync	1000 peticiones al mes.	1	$10,55	$10,50
Valor Total Mensual (USD)				**$19.967,17**

Tabla 4.14: Caso de Negocio - Costos Directos

4.2.2.2. Costos Indirectos

Son costos asociados, pero no exclusivos, al proyecto o que se pueden asociar a este de manera indirecta. La Tabla 4.15 tabla muestra el detalle de los costos indirectos asociados al proyecto.

Caso de Negocio - Costos Indirectos				
Recurso	**Detalle**	**Cantidad**	**Mes (USD)**	**Total (USD)**
Gerente de Producto	A un 25 % de su capacidad.	1	$1.250,00	$1.250,00
Licencia Jira	Licencia Standard	8	$7,75	$62,00
Licencia Confluence	Licencia Standard	8	$5,75	$46,00
Licencia GitHub	Licencia Free	8	$0,00	$0,00
Licencia Postman	Licencia Basic	2	$12,00	$24,00
Google Workspace	Licencia Starter	8	$6,90	$55,20
SonarCloud	Licencia Mensual (As a Service)	1	$10,00	$10,00

Continúa en la siguiente página

Tabla 4.15 – Continuación de la página previa

Caso de Negocio - Costos Indirectos	
Valor Total Mensual (USD)	**$1.447,20**

Tabla 4.15: Caso de Negocio - Costos Indirectos

4.2.2.3. Costos de Operación Tecnológicos Actuales

Se muestran los costos de operación tecnológicos mensuales que Vaova tiene actualmente para que los sistemas funcionen. Estos costos son por la infraestructura y el pago al proveedor *Viaxlab*. La Tabla 4.16 muestra los costos de operación tecnológicos actuales.

Caso de Negocio - Costos de Operación Tecnológicos Actuales				
Recurso	**Detalle**	**Cantidad**	**Mes (USD)**	**Total (USD)**
Google Platform	Infraestructura tecnológica.	1	$190,00	$190,00
MongoDB	Base de datos.	1	$85,00	$85,00
Viaxlab	Precio por cada pasajero.	9	$9,00	$9,00
Valor Total Mensual (USD)				**$284,00**

Tabla 4.16: Caso de Negocio - Costos de Operación Tecnológicos Actuales

4.2.2.4. Costos de Operación Tecnológicos Futuros

En los costos de operación tecnológicos futuros se incluyen los costos mensuales que debe asumir Vaova para mantener el sistema en operación una vez se concluye la etapa de desarrollo. Se incluyen todos los servicios que se deben contratar con AWS y una porción del tiempo de un desarrollador Semi Senior y un ingeniero QA para solucionar errores o fallas que puedan presentar durante la operación. La Tabla 4.17 muestra el resumen de los costos de operación tecnológicos futuros.

Caso de Negocio - Costos de Operación Tecnológicos Futuros				
Recurso	**Detalle**	**Cantidad**	**Mes (USD)**	**Total (USD)**
				Continúa en la siguiente página

Tabla 4.17 – Continuación de la página previa

Caso de Negocio - Costos de Operación Tecnológicos Futuros				
Licencia Auth0	Licencia B2C Essentials.	1	$23,00	$23,00
AWS API Gateway	5000 peticiones al mes.	1	$22,50	$22,50
AWS Lambda	10000 peticiones al mes. Concurrencia de 100 horas.	1	$15,69	$15,69
AWS RDS for MySQL	Tipo db.t4g.medium, Single AZ.	1	$83,29	$83,29
AWS S3	5 GB de transferencia.	1	$0,82	$0,82
AWS DynamoDB	1 GB de almacenamiento.	1	$0,45	$0,45
AWS SQS	1 millón de peticiones.	1	$0,09	$0,09
AWS SNS	5000 peticiones al mes.	1	$0,53	$0,53
Amazon EventBridge	5000 eventos al mes.	1	$0,03	$0,03
Amazon CloudWatch	10 métricas.	1	$16,50	$16,50
AWS Systems Manager	50 configuraciones de aplicaciones.	1	$13,70	$13,70
AWS AppSync	1000 peticiones al mes.	1	$10,55	$10,50
Soporte Desarrollo	Desarrollador Semi Senior al 25 % de su tiempo.	1	$625,00	$625,00
Soporte QA	Ingeniero QA al 10 % de su tiempo.	1	$250,00	$250,00
Valor Total Mensual (USD)				**$1.062,15**

Tabla 4.17: Caso de Negocio - Costos de Operación Tecnológicos Futuros

4.2.2.5. Inversión Total

Con la descripción de costos directos, costos indirectos y costos de operación se tiene que la inversión total en el desarrollo del proyecto se distribuirá de la siguiente manera durante los próximos 5 años. La Tabla 4.18 muestra la distribución de la inversión total.

Caso de Negocio - Inversión Total						
Costos	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
Desarrollo	$64.243,11	$64.764,10	$0,00	$0,00	$0,00	$0,00
Operación y Mantenimiento	$0,00	$11.896,08	$15.988,33	$17.906,93	$20.055,76	$22.462,45
Total x Año	$64.243,11	$76.660,18	$15.988,33	$17.906,93	$20.055,76	$22.462,45
Valor Total Inversión (USD): $217.316,77						

Tabla 4.18: Caso de Negocio - Inversión Total

Los valores de 2023 y 2024 incluyen los costos del desarrollo del proyecto, del año 2025 en adelante sólo son los costos de infraestructura y mantenimiento del sistema. Por este motivo el valor en el 2024 es el mayor de todos, porque incluye la mitad del costo de desarrollo y la infraestructura de producción de 10 meses. Los valores están calculados utilizando la fórmula de valor presente con una inflación del 12 % (es el promedio que se ha venido manejando en Colombia durante los últimos años).

4.2.3. Análisis de Retorno de Inversión

En el análisis de *Retorno de Inversión* (ROI), se van a comparar los gastos que actualmente realiza Vaova contra el ahorro que significaría el desarrollo y puesta en producción de la arquitectura.

4.2.3.1. Cálculo de Beneficios

En el cálculo de beneficios se tienen en cuenta el dinero ahorrado por retirar de producción lo que se tiene actualmente. Adicionalmente, se suman los ingresos que se generan por la puesta en producción de la nueva arquitectura. La Tabla 4.19 muestra el cálculo de beneficios desde el año 2023 hasta el año 2028.

Caso de Negocio - Cálculo de Beneficios							
Costos	**Valor**	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
Google Platform	$190,00	$0,00	$1.915,20	$2.860,03	$3.203,24	$3.587,62	$4.018,14
MongoDB	$85,00	$0,00	$856,80	$959,62	$1.074,77	$1.203,74	$1.348,19
Viaxlab	$9,00	$0,00	$22.680,00	$36.126,72	$42.990,80	$50.982,03	$60.272,09
Fidelización	$0,00	$0,00	$0,00	$2.312,00	$5.235,84	$7.908,07	$11.210,77

Continúa en la siguiente página

Tabla 4.19 – Continuación de la página previa

Caso de Negocio - Cálculo de Beneficios						
Valor Total (USD)	**$0,00**	**$25.452**	**$42.258**	**$52.505**	**$63.681**	**$76.849**

Tabla 4.19: Caso de Negocio - Cálculo de Beneficios

De manera permanente se tendría el ahorro que representa dejar de utilizar *Google Platform*, *MongoDB* y *Viaxlab*. Se proyecta un beneficio de *Fidelización* a partir del año 2025. Este el beneficio consiste en que se ganará el 1 % de pasajeros adicionales cada año a partir de 2025. Este es el resumen del cálculo de beneficios:

- Vaova espera movilizar a 3000 pasajeros en 2024 y aumentar este número de manera gradual hasta 4000 pasajeros para el año 2028.
- Con la salida a producción, los costos de infraestructura generados por *Google Platform* y *MongoDB* serán reemplazados por AWS.
- La integración con *Viaxlab* se cancelará, y la arquitectura proveerá los servicios necesarios.

4.2.3.2. Análisis del Flujo de Dinero

El análisis del flujo de dinero corresponde a la posibilidad de empezar a recuperar la inversión lo más pronto que se pueda. Con la estructura de costos directos, indirectos y de operación se tiene el análisis de flujo de dinero que se muestra en la Tabla 4.20.

Caso de Negocio - Análisis de Flujo de Dinero						
Detalle	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
Inversión	-$64.243,11	-$76.660,18	-$15.988,33	-$17.906,93	-$20.055,76	-$22.462,45
Beneficios	$0,00	$25.452,00	$42.258,37	$52.504,64	$63.681,47	$76.849,19
Flujo de dinero	-$64.243,11	-$51.208,18	$26.270,04	$34.597,71	$43.625,70	$54.386,74

Tabla 4.20: Caso de Negocio - Análisis de Flujo de Dinero

A partir de la Tabla 4.20, se generan las siguientes gráficas que muestran el estado de la inversión contra los beneficios generados, y el flujo de dinero durante los años 2023-2028. La Figura 4.8 muestra la evolución de la inversión realizada contra los beneficios generados, y la Figura 4.9 muestra la evolución del flujo de dinero.

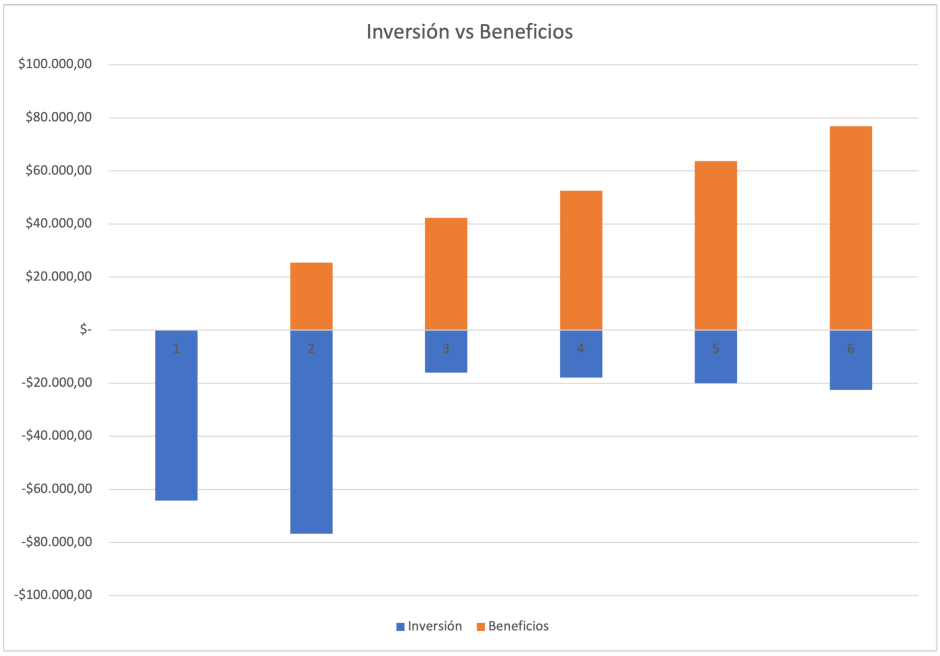


Figura 4.8: Caso de Negocio - Inversión vs. Beneficios

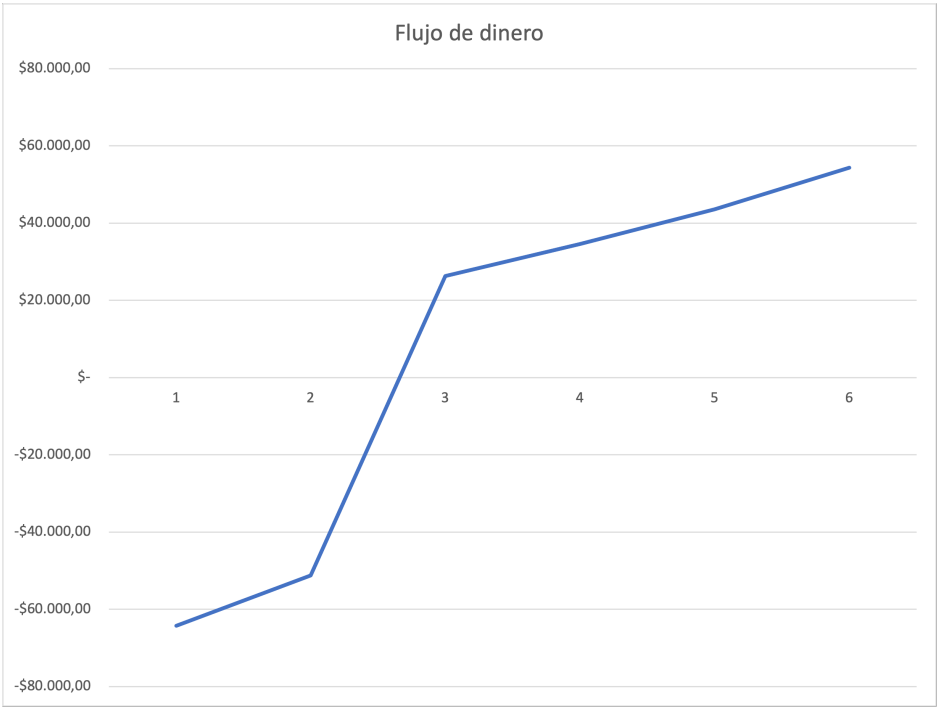


Figura 4.9: Caso de Negocio - Flujo de Dinero

A partir de las gráficas en las Figuras 4.8 y 4.9, se obtienen las siguientes conclusiones:

- En el tercer año (2025) se empieza a tener flujo positivo de dinero. Finalizando el año con un capital de $26.270,04 dólares.
- La inversión total se recupera al término de 5 años. Al finalizar el año 2028 el costo total de la inversión es de $217.316,77 dólares. Por otra parte, al finalizar el mismo año los beneficios serán de $260.745,67. Esto significa que se recupera el total de la inversión y se obtiene un beneficio del 19,98 %.

4.2.4. Beneficios Esperados

A partir de los objetivos de negocio (Sección 3.5.2.1) que busca satisfacer el diseño de la arquitectura, los mayores beneficios de la implementación de este proyecto son una operación más efectiva para Vaova. Aunque también se podrían obtener beneficios económicos gracias a la mejora en la operación. Los objetivos de negocio que se propusieron en el diseño de la arquitectura son los siguientes:

- **BO_01** – Automatizar Procesos Operativos.
- **BO_02** – Mejorar la Experiencia de los Viajeros.
- **BO_03** – Reducir Costos.

El siguiente análisis se podría realizar de manera cuantitativa, pero no se cuenta con los datos suficientes para realizarlo. Por lo anterior, el análisis se realiza de manera cualitativa en la mayoría de los casos, sin profundizar en cifras económicas.

4.2.4.1. Beneficios Para el Objetivo de Negocio BO_01

La Tabla 4.21 muestra los beneficios esperados para el objetivo de negocio BO_01.

Beneficios Objetivo de Negocio BO_01
Se implementan los servicios para que los Promotores puedan tener una herramienta que permita diseñar los Itinerarios de manera sencilla sin la asistencia de los Asesores Comerciales. Esto facilitará la revisión y aprobación de Itinerarios por parte de los Asesores Comerciales, ya que los Promotores sólo podrán crear los Itinerarios siguiendo las reglas del sistema.
Continúa en la siguiente página

Tabla 4.21 – Continuación de la página previa

Beneficios Objetivo de Negocio BO_01
Los servicios de Join tendrán la capacidad de manejar a los viajeros de manera concurrente sin presentar errores. Las inconsistencias que se presentan actualmente desaparecerán y la operación del Join se simplificará de manera significativa. Los operarios de Vaova ya no tendrán que gastar su tiempo en la revisión del Join, para asegurar que cada viajero tiene sus hoteles, vuelos y compañero de cuarto. Esta tarea está garantizada en el diseño de la arquitectura.
La integración con Viaxlab se cancelará y la arquitectura brindará los servicios que sean necesarios. Esto ofrecerá el mayor ahorro económico posible a Vaova, ya que el costo de $9 dólares por pasajero será eliminado. El beneficio no es sólo económico, ya que los Operarios de Vaova no tendrán que cargar de manera manual los datos de los viajeros a la plataforma de Viaxlab, dicha tarea desaparece totalmente de la operación de Vaova.
Mayor Beneficio Obtenido
Al implementar la arquitectura en un ambiente productivo, el mayor beneficio para el objetivo de negocio BO_01 es el ahorro de tiempo. Los Asesores Comerciales y los Operarios de Vaova ya no tendrán que emplear su tiempo en tareas de validación, vigilancia y carga de datos porque todas estas tareas estarán automatizadas. Con este ahorro de tiempo, estas personas podrán realizar otras tareas o actividades que generen mayor beneficio a la compañía.

Tabla 4.21: Beneficios Objetivo de Negocio BO_01

4.2.4.2. Beneficios Para el Objetivo de Negocio BO_02

La Tabla 4.22 muestra los beneficios esperados para el objetivo de negocio BO_02.

Beneficios Objetivo de Negocio BO_02
Se elimina la complejidad que tenían los viajeros al momento de comprar un viaje con Vaova. Con la implementación de este proyecto, sólo deberán comprar su viaje en la plataforma *TourHero/ WeTravel* y tener la aplicación Vaovapp. El proceso de registro en Vaova se hará automáticamente y de manera transparente para el viajero. Desde Vaovapp podrá ver su información, los viajes próximos, realizar el Join y ver su itinerario de viaje.
Como la integración con Viaxlab se cancela, la verificación de viajeros habilitados para un evento se podrá hacer desde Vaovapp de manera sencilla con una alta disponibilidad en los servicios. Ya no se tendrán retrasos para el acceso a las actividades de los viajeros por falta de disponibilidad en los servicios.
Eliminación de la inconsistencia de la información. Anteriormente, el viajero debía registrarse en tres plataformas diferentes (*TourHero/ WeTravel*, *MyTrip* y *Viaxlab*). Esto se elimina y el origen de la información serán la plataforma externa de ventas (*TourHero* o *WeTravel*). Esta información se sincronizará de manera automática con Vaova.
Un beneficio que ya me mencionó anteriormente es la Fidelización que se espera obtener, aumentar el número de pasajeros gracias a las facilidades que se ofrecen y la experiencia mejorada.
Mayor Beneficio Obtenido
El beneficio obtenido para el objetivo de negocio BO_02 es la comodidad y la facilidad que se le ofrecerá a los viajeros. Ya se contabilizó un beneficio económico: la fidelización que ayudará a incrementar el número de viajeros que Vaova transporta en un año. Pero se pueden presentar otros beneficios: un mejor posicionamiento de marca, que más Promotores quieran realizar viajes con Vaova y las recomendaciones boca-a-boca de los viajeros que quedan satisfechos.

Tabla 4.22: Beneficios Objetivo de Negocio BO_02

4.2.4.3. Beneficios Para el Objetivo de Negocio BO_03

La Tabla 4.23 muestra los beneficios esperados para el objetivo de negocio BO_03.

Beneficios Objetivo de Negocio BO_03
La revisión de itinerarios por parte de los Asesores Comerciales se simplificará, ya que la creación de estos por parte de los Promotores estará controlada por reglas de negocio definidas en la arquitectura y no se podrán crear itinerarios que no satisfagan dichas reglas. Los Asesores Comerciales podrán realizar la validación de manera ágil y con tiempos de respuesta mucho más cortos. Lo anterior permitirá a los Asesores Comerciales realizar otras actividades de más beneficio para Vaova. Otro escenario es que se podría plantear una reducción en el número de Asesores Comerciales, ya que las revisiones serían más sencillas de realizar y no tomarían tanto tiempo.
El Join no necesitará supervisión humana para controlar que todo esté correcto con respecto a hoteles, vuelos y compañero de cuarto de los viajeros. Este proceso también tendrá reglas claras que permitirán que los viajeros lo hagan de manera correcta y sin usurpación de vuelos, habitaciones o compañeros de viaje. En el Join, los Operarios sólo deben monitorear que las habitaciones y vuelos estén disponibles para los viajeros que realizan el Join. Y, en caso de ser necesario, agregar más recursos cuando se agoten. Los Operarios se podrán concentrar en la planificación del viaje y de monitorear que los recursos asignados al viaje son suficientes. Los Operarios tendrán más tiempo para la planificación del viaje, lo que se puede traducir en mejores negociaciones para obtener precios más favorables en hoteles, aerolíneas, transportes, etc. Estas actividades son de mucho más beneficio para Vaova, que estar vigilando el Join de los viajeros.
La arquitectura está diseñada para ofrecer una alta disponibilidad en sus servicios, por lo tanto, Itinerarios y Vaovapp no deberán presentar fallas al momento de ser requeridas. Esto permitirá agilidad al momento de realizar validaciones durante la ejecución del viaje. Con este beneficio se terminarían los gastos hormiga durante los viajes (compra de resmas de papel, tintas para impresora, etc).
La integración con Viaxlab se cancelará y el costo de $9 dólares por pasajero se convertirán en un ahorro. Adicionalmente, los problemas con la carga de información a Viaxlab también se evitará, ya que todo funcionará de manera automática con TourHero/WeTravel-Vaovapp. Este es el mayor beneficio económico, ya que el dinero que se deja de pagar a Viaxlab puede ser invertido para planes de expansión de la compañía, el diseño y contratación de nuevas actividades, entre otras.
Continúa en la siguiente página

Tabla 4.23 – Continuación de la página previa

Beneficios Objetivo de Negocio BO_03
Mayor Beneficio Obtenido
Los beneficios económicos que se obtienen para el objetivo de negocio BO_03: • El ahorro de tiempo de los Asesores Comerciales y los Operarios. • Evitar gastos hormiga durante los viajes. • La cancelación de la integración con Viaxlab.

Tabla 4.23: Beneficios Objetivo de Negocio BO_03

4.2.5. Criterios de Éxito

Para el desarrollo del proyecto de implementación se deben tener en cuenta estas variables para tener éxito:

- La implementación del proyecto se cumple en tiempo, forma y presupuesto.
- Un factor externo, la implementación de la aplicación web de Itinerarios debe desarrollarse en paralelo y estar lista antes de abril de 2023.
- Un factor externo, la implementación de la aplicación móvil Vaovapp debe desarrollarse en paralelo y estar lista antes de abril de 2023.
- El área comercial y operativa debe planear las estrategias para aumentar gradualmente el número de pasajeros, tal como es el plan general de Vaova hasta 2028.

4.2.6. Conclusiones y Recomendaciones

El diseño de la arquitectura de servicios para Vaova Travel es crucial para simplificar, facilitar y mejorar la operación de Vaova. El diseño e implementación de la arquitectura no es la única tarea, también se deben diseñar la aplicación web de Itinerarios y aplicación móvil Vaovapp. Estos tres proyectos estarán bajo el mando de la Gerente de Producto y debe garantizar la correcta ejecución de los presupuestos y cronogramas de los tres proyectos.

Actualmente Vaova no opera de forma efectiva, tiene márgenes de ganancia muy bajos y los gastos generados por esta operación inefectiva son altos. La implementación de la Arquitectura de Software para Vaova Travel, en conjunto con Itinerarios Web y Vaovapp, ayudará a que Vaova se organice para operar de una mejor manera y después le ayudará en el ahorro de costos y gastos.

Capítulo 5

Conclusiones

En esta sección se presentan las conclusiones para cada objetivo específico y una conclusión para el objetivo general (Sección 1.4). Estas conclusiones se generan a partir de los resultados del desarrollo del proyecto de grado, presentando las dificultades y lecciones aprendidas del mismo.

Proponer una arquitectura de software de un sistema de información que soporte la operación de Vaova, teniendo en cuenta las necesidades de alta disponibilidad en los servicios propios de Vaova y la interoperabilidad con plataformas de terceros.

5.1. Pasos Previos

Antes de la ejecución de los objetivos específicos, se realizaron algunas tareas de contextualización sobre el negocio y la operación de Vaova. Esto fue necesario porque era muy importante conocer su manera de operar antes de realizar cualquier inventario o diseño de arquitectura.

Se realizó el *Business Model Canvas* de Vaova para conocer los segmentos de clientes, la propuesta de valor, los canales de distribución, la relación con los clientes, los ingresos, los recursos, las actividades fundamentales, los socios estratégicos y la estructura de costos. Este diseño proporcionó un conocimiento muy importante para entender el modelo de negocio que maneja Vaova.

Se analizaron los procesos operativos de Vaova que más estaban relacionados con sistemas de información. Estos procesos operativos se trasladaron a diagramas *BPMN*, que permitieron realizar una asiganción procesos-sistemas.

Aunque estos pasos previos no se contemplaron en los objetivos, su realización era fundamental para tener éxito en la ejecución del proyecto de grado. Conocer a Vaova como compañía, su negocio y sus procesos ayudaría a determinar sus mayores dolencias y los puntos claves en los que se presentaban la mayor cantidad de errores y reprocesos en su operación.

5.2. Objetivo Específico 1

Realizar un inventario de activos de software e integraciones con los que actualmente cuenta Vaova para determinar cuáles se pueden reutilizar, refactorizar o retirar de la operación.

Al realizar las actividades para este objetivo, se logró determinar que los sistemas con los que Vaova realizaba su operación no eran los más óptimos. Muchos de los sistemas no funcionaban adecuadamente, e incluso algunos se tuvieron que retirar porque generaban muchos problemas pero ningún valor.

Los análisis realizados proporcionaron una evaluación clara del estado actual de Vaova. Esto permitió establecer que la mayoría de sistemas se debían retirar de operación, y permitió identificar que el proceso de las integraciones con *TourHero* y *WeTravel* se podía reutilizar realizando algunas modificaciones.

Al finalizar este objetivo, se obtuvieron las bases para realizar un diseño de arquitectura que ayudara a Vaova a mejorar su operación.

5.3. Objetivo Específico 2

Definir las historias épicas que se requieren para diseñar la arquitectura de software.

Las historias épicas serían otro insumo para el diseño de la arquitectura. La definición de estas historias épicas era esencial para comprender las necesidades y requerimientos de Vaova, y asegurar que se tendrían en cuenta en el diseño de la arquitectura de software.

La definición de las historias épicas resultó en una comprensión clara y detallada de los requerimientos y necesidades de Vaova. Se definieron utilizando el formato *Gherkin*, lo que facilitó su análisis e incluso ayudó a generar diversos escenarios para una misma historia.

5.4. Objetivo Específico 3

Diseñar la arquitectura de software, contemplando los escenarios de calidad para disponibilidad e interoperabilidad empleando los modelos de diseño de Domain-Driven Design (DDD) y Attribute-Driven Design (ADD).

Al momento de definir este objetivo específico sólo se pensaron los atributos de calidad para *disponibilidad* e *interoperabilidad*, ya que eran los más evidentes al momento de realizar el análisis durante el ante proyecto de grado. Al ejecutar los pasos previos, y los objetivos específicos 1 y 2 se determinó que Vaova necesitaba un diseño de arquitectura que ofreciera más que dichos atributos.

Por el motivo anterior, se incluyeron los atributos de calidad de *madurez* y *pertinencia funcional*. El atributo de calidad de madurez ayudaría a que el diseño de la arquitectura tuviera en cuenta elementos como el MTBF y MTTR que son la base para ofrecer una mejor disponibilidad. El atributo de calidad de pertinencia funcional ayudaría a ofrecer el conjunto básico y esencial de

servicio que Vaova necesitaba para operar más eficientemente.

En el diseño de la arquitectura de software se utilizaron los modelos de Domain-Driven Design (DDD) y Attribute-Driven Design (ADD), estos modelos permitieron tener enfoque efectivo para diseñar una arquitectura sólida y escalable. Se tuvieron en cuenta los objetivos de negocio, los atributos y escenarios de calidad y las restricciones de arquitectura lo que garantiza que la arquitectura propuesta cumpla con los requerimientos técnicos y estratégicos de Vaova.

Este objetivo específico ha proporcionado un diseño de arquitectura que soporta la operación de Vaova, y define una hoja de ruta clara para que Vaova pueda implementar la arquitectura y ponerla en producción de manera exitosa.

5.5. Objetivo Específico 4

Evaluar la arquitectura de software utilizando el modelo analítico de Software Architecture Analysis Method (SAAM).

Se realizaron dos evaluaciones sobre la arquitectura: una técnica y una financiera. Esto permitió establecer si el diseño de la arquitectura era viable tanto desde el punto de vista técnico como económico.

La evaluación técnica del diseño de la arquitectura de software utilizando el modelo analítico de *Software Architecture Analysis Method* (SAAM) proporcionó una visión crítica de la calidad y la robustez de la arquitectura propuesta. Los resultados de la evaluación identificaron puntos de mejora y fortalezas en la arquitectura, lo que permitirá tomar decisiones informadas para mantenimientos futuros.

La evaluación financiera del diseño de la arquitectura de software utilizando el *análisis de retorno de inversión* (ROI) permitió conocer el nivel de inversión, los beneficios esperados y el tiempo necesario para obtener un flujo positivo de efectivo. Este análisis permitirá a Vaova conocer los niveles de riesgo de la inversión que debe realizar para implementar la arquitectura y el tiempo estimado para recuperar dicha inversión y obtener ganancias.

Las evaluaciones técnica y financiera respaldan el diseño de arquitectura realizado. Este diseño tiene en cuenta los aspectos técnicos que necesita Vaova con un nivel de inversión bajo que puede generar grandes beneficios operativos y económicos.

5.6. Objetivo General

Proponer una arquitectura de software de un sistema de información que soporte la operación de Vaova, teniendo en cuenta las necesidades de alta disponibilidad en los servicios propios de Vaova

y la interoperabilidad con plataformas de terceros.

El fin de este proyecto de grado era proponer una arquitectura de software que ayudara a Vaova en su operación. Con la ejecución de este proyecto, se logró proponer un diseño de arquitectura de software sólido y estratégico para Vaova. Este diseño de arquitectura no sólo aborda con éxito las necesidades de alta disponibilidad en los servicios propios de Vaova, sino que brinda las herramientas necesarias para obtener un sistema maduro y con las operaciones esenciales para la operación de la compañía. También, garantiza la interoperabilidad efectiva con las plataformas de terceros que son los aliados estratégicos para la compañía.

A través de un enfoque analítico que incluyó la identificación y evaluación de activos de software existentes, la definición de historias épicas usando el formato *Gherkin*, el diseño basado en *Domain-Driven Design* (DDD) y *Attribute-Driven Design* (ADD), se generó una base sólida para la estabilización, crecimiento y evolución de Vaova. La evaluación técnica de la arquitectura utilizando el modelo analítico de *Software Architecture Analysis Method* (SAAM) y la evaluación financiera proporcionaron la validación crítica de la pertinencia y robustez del diseño de arquitectura de software propuesto.

5.7. Conclusiones Personales

El desarrollo de este proyecto de grado me permitió desempeñar el rol de arquitecto de software. Este es un rol integral que requiere de muchas áreas del conocimiento: ingeniería de software, negociación, finanzas, planeación estratégica, liderazgo, entre otras. Diseñar la arquitectura de software para Vaova requirió adquirir conocimientos que no consideraba importantes hasta ahora y entender que los ingenieros de sistemas nos debemos preparar cada vez más, y no sólo en aspectos técnicos.

Considero que el diseño de la arquitectura utilizando DDD y ADD, y la evaluación técnica utilizando SAAM fueron los aspectos más interesantes del desarrollo del proyecto. Me permitió estudiar a profundidad y poner en práctica dichas metodologías, que sólo había conocido de manera teórica.

Durante la ejecución del proyecto tuve que realizar tareas que no estaba estimadas, pero creo que todo lo que se realizó fue necesario. Entender todos los pasos que se requieren para diseñar una arquitectura de software es fundamental para generar un buen diseño (aunque la literatura diga que no hay buenos ni malos diseños, sólo diseños más adecuados).

Capítulo 6

Anexos

6.1. Business Model Canvas

El diseño del *Business Model Canvas* de Vaova fue realizado en la aplicación web de EDIT.org, que ofrece la posibilidad de realizar el diseño de manera gratuita (sólo agrega una marca de agua en la esquina).

En la carpeta **1 - Business Model Canvas** de OneDrive se puede encontrar el archivo *Vaova Travel - Business Model Canvas.pdf* con una excelente resolución.

6.2. Diagramas BPMN Procesos Operativos

Los diagramas BPMN de Vaova fueron realizados en la aplicación web draw.io.

En la carpeta **2 - Diagramas BPMN** de OneDrive se pueden encontrar los siguientes archivos:

- *1 - Vaova Itinerarios - BPMN.svg*
- *2 - Vaova WeTravel Sync Request - BPMN.svg*
- *3 - Vaova WeTravel Sync Process - BPMN.svg*
- *4 - Vaova TourHero Sync Request - BPMN.svg*
- *5 - Vaova TourHero Sync Downloader - BPMN.svg*
- *6 - Vaova TourHero Sync Process - BPMN.svg*
- *7 - Vaova Join - BPMN.svg*
- *8 - Vaova Invitación Roomie - BPMN.svg*

6.3. Diagramas C4 - Contexto Actual

Los diagramas de contexto fueron realizados en la aplicación web Structurizr, que permite realizar diagramas en C4 en la modalidad de *diagramas como código*.

En la carpeta **3 - Diagramas C4** de OneDrive se pueden encontrar las siguientes carpetas:

- **1 - Imágenes**. Esta carpeta contiene las imágenes en formato SVG de los diagramas en el modelo C4:
  - *01 - Vaova Travel - Context - As Is.svg*
  - *02 - Vaoval Travel - Container - As Is.svg*
  - *03 - Vaoval Travel - Component - MyTrip.svg*
  - *04 - Vaova Travel - Component - Viaxlab.svg*
  - *05 - Vaova Travel - Component - TourHero.svg*
  - *06 - Vaova Travel - Component - WeTravel.svg*
- **2 - Fuente**. Esta carpeta contiene el archivo *Vaova.dsl*, que guarda el código fuente que permite importar y editar los diagramas en la aplicación web Structurizr.

6.4. Colecciones Base de Datos *MongoDB*

Los diagramas de las colecciones de la base de datos de MongoDB fueron realizados en la aplicación web JSON CRACK que permite realizar diagramas a partir de objetos JSON.

En la carpeta **4 - Estado Actual/MongoDB** de OneDrive se pueden encontrar las siguientes carpetas:

- **1 - Imágenes**. Esta carpeta contiene las imágenes en formato SVG de los diagramas de las colecciones de la base de datos *MongoDB*:
  - *configs.svg*
  - *flights_info.svg*
  - *hotels.svg*
  - *join_history.svg*
  - *join_users_history.svg*
  - *join_users.svg*
  - *report_info.svg*
  - *trips_info.svg*
  - *trips_viaxlab.svg*
  - *user_history.svg*
  - *users_viaxlab.svg*
  - *users.svg*

- **2 - Fuente**. Esta carpeta contiene los archivos que se pueden importar en JSON CRACK para visualizar y editar los diagramas de las colecciones:

  - *configs.json*
  - *flights_info.json*
  - *hotels.json*
  - *join_history.json*
  - *join_users_history.json*
  - *join_users.json*
  - *report_info.json*
  - *trips_info.json*
  - *trips_viaxlab.json*
  - *user_history.json*
  - *users_viaxlab.json*
  - *users.json*

6.5. Historias Épicas

Las historias épicas se definieron utilizando el formato *Gherkin*. Se pueden utilizar editores como *Visual Studio Code* o *Sublime Text* para la creación y edición de los archivos *.feature* (extensión de los archivos en formato *Gherkin*).

En la carpeta **5 - Historias Épicas** de OneDrive se pueden encontrar las siguientes carpetas:

- **01 - Itinerarios**. Contiene todos los archivos que definen las historias épicas para el subsistema de Itinerarios:

  - *01 - Login Usuario.feature*
  - *02 - Recuperar Contrasenia.feature*
  - *03 - Crear Usuario.feature*
  - *04 - Crear Itinerario.feature*
  - *05 - Editar Itinerario.feature*
  - *06 - Listar Itinerario.feature*
  - *07 - Filtrar Itinerario.feature*
  - *08 - Eliminar Itinerario.feature*
  - *09 - Numero Estima de Viajeros.feature*

- *10 - Pais de Destino.feature*
- *11 - Fechas de Viaje.feature*
- *12 - Cantidad de Destinos.feature*
- *13 - Paso a Paso.feature*
- *14 - Listado de Destinos.feature*
- *15 - Busqueda de un Destino.feature*
- *16 - Detalle Destino.feature*
- *17 - Calendario de Actividades.feature*
- *18 - Listado de Actividades.feature*
- *19 - Detalle Actividad.feature*
- *20 - Resumen.feature*

- **02 - Vaovapp**. Contiene todos los archivos que definen las historias épicas para el subsistema de Vaovapp:

  - *01 - Login de Usuario.feature*
  - *02 - Recuperar Contrasenia.feature*
  - *03 - Informacion de Usuario.feature*
  - *04 - Editar Información de Usuario.feature*
  - *05 - MyTrips.feature*
  - *06 - Upgrade Single Room.feature*
  - *07 - Tipo de Acomodacion.feature*
  - *08 - Vuelos.feature*
  - *09 - Información de Vuelos.feature*
  - *10 - Invitacion de Roomie.feature*
  - *11 - Invitación.feature*
  - *12 - Cancelar Invitacion.feature*
  - *13 - Rechazar Invitación.feature*
  - *14 - Ultima Mujer.feature*
  - *15 - Invitacion Aceptada.feature*
  - *16 - Join.feature*
  - *17 - Inventario Habitaciones.feature*
  - *18 - Inventario Vuelos.feature*

- **03 - Tableros**. Contiene el archivo que define la historia épica para el subsistema de Tableros de Información:

  - *01 - Tableros de Informacion.feature*

6.6. Documento de Arquitectura de Software

En la carpeta **6 - Documentos de Arquitectura de Software** de OneDrive se pueden encontrar las siguientes carpetas:

- **1 - Documento de Arquitectura de Software.** Contiene el documento de arquitectura de software completo.
  - *Documento de Arquitectura de Software.docx.*
- **2 - Documento de Registro de Decisiones de Arquitectura.** Contiene el documento con el registro de decisiones tomadas sobre la arquitectura.
  - *Documento de Registro de Decisiones de Arquitectura.docx.*
- **2 - Diagramas.** Contiene todos los diagramas que se muestran en el documento de arquitectura de software. Se pueden encontrar las siguientes carpetas:
  - **1 - Subdominios y Contextos Delimitados.** Contiene los diagramas de contexto delimitado de cada subdominio.
    - *1 - Vaova-Subdominios.svg.*
    - *2 - Vaova-Destinos - Bounded Context.svg.*
    - *3 - Vaova-Transportes - Bounded Context.svg.*
    - *4 - Vaova-Acomodaciones - Bounded Context.svg.*
    - *5 - Vaova-Actividades - Bounded Context.svg.*
    - *6 - Vaova-Itinerarios - Bounded Context.svg.*
    - *7 - Vaova-Viajes - Bounded Context.svg.*
    - *8 - Vaova-Ventas - Bounded Context.svg.*
    - *9 - Vaova-Usuarios - Bounded Context.svg.*
    - *10 - Vaova-Join - Bounded Context.svg.*
    - *11 - Vaova-Reportes - Bounded Context.svg*
    - *12 - Vaova-Plataformas - Bounded Context.svg.*
    - *13 - Vaova-Notificaciones - Bounded Context.svg.*
  - **1 - Diagrama de Contexto.** Contiene el diagrama de contexto del diseño de la arquitectura.
    - *Vaova-Arq. - Diagrama de Contexto.svg.*
  - **2 - Diagramas de Despliegue.** Contiene los 10 diagramas generados para la Vista de Despliegue.
    - *1 - Vaova-Deploy - Red.svg.*

- *2 - Vaova-Deploy - Servidores.svg.*
- *3 - Vaova-Deploy - Almacenamiento.svg.*
- *4 - Vaova-Deploy - Componentes y Dependencias.svg.*
- *5 - Vaova-Deploy - Comunicación Interna - Síncrona.svg.*
- *6 - Vaova-Deploy - Comunicación Interna - Asíncrona.svg.*
- *7 - Vaova-Deploy - Comunicación Interna - Eventos.svg.*
- *8 - Vaova-Deploy - Comunicación Externa.svg.*
- *9 - Vaova-Deploy - Monitoreo y Registro.svg.*
- *10 - Vaova-Deploy - Seguridad.svg.*

- **4 - Diagramas de Datos**. Contiene los diagramas del Modelo de Información, se realizaron con la herramienta PlantUML. Contiene las siguientes carpetas:
  - **1 - BackOffice**. Contiene los diagramas relacionados con BackOffice.
    - *1 - BackOffice MER.puml.*
    - *2 - BackOffice MER.svg.*
    - *3 - BackOffice States.puml.*
    - *4 - BackOffice States.svg.*
  - **2 - Itinerarios**. Contiene los diagramas relacionados con Itinerarios.
    - *1 - Itinerarios-Viajes MER.puml.*
    - *2 - Itinerarios-Viajes MER.svg.*
    - *3 - Trips States.puml.*
    - *4 - Trips States.svg.*
    - *5 - Trips-Destinations-Activities-Complements States.puml.*
    - *6 - Trip_Destination-Trip_Destination_Activities-Trip_Complements States.svg.*
    - *7 - Trips Other States.puml.*
    - *8 - Trip_Other States.svg.*
  - **3 - Ventas**. Contiene los diagramas relacionados con Ventas.
    - *1 - Ventas MER.puml.*
    - *2 - Ventas MER.svg.*
    - *3 - Ventas States.puml.*
    - *4 - Ventas States.svg.*
  - **4 - Usuarios**. Contiene los diagramas relacionados con Usuarios.
    - *1 - Usuarios MER.puml.*
    - *2 - Usuarios MER.svg.*
    - *3 - Usuarios States.puml.*
    - *4 - Usuarios States.svg.*
  - **5 - Join**. Contiene los diagramas relacionados con Join.

- ◇ *1 - Join MER.puml.*
- ◇ *2 - Join MER.svg.*
- ◇ *3 - Trip-Join States.puml.*
- ◇ *4 - Join-Other States.svg.*
- ◇ *5 - Join Other States.puml.*
- ◇ *6 - Trip-Join States.svg.*

- **6 - Plataformas**. Contiene los diagramas relacionados con Plataformas.
  - ◇ *1 - Plataformas MER.puml.*
  - ◇ *2 - Plataformas MER.svg.*
  - ◇ *3 - Plataformas States.puml.*
  - ◇ *4 - Plataformas States.svg.*
- **7 - Notificaciones**. Contiene los diagramas relacionados con Notificaciones.
  - ◇ *1 - Notificaciones MER.puml.*
  - ◇ *2 - Notificaciones MER.svg.*
  - ◇ *3 - Notificaciones States.puml.*
  - ◇ *4 - Notificaciones States.svg.*
- **8 - Tableros**. Contiene los diagramas relacionados con Tableros.
  - ◇ *1 - Tableros MER.puml.*
  - ◇ *2 - Tableros MER.svg.*

6.7. Documentos de Evaluación de Arquitectura de Software

En la carpeta **7 - Documentos de Evaluación de Arquitectura** de OneDrive se pueden encontrar las siguientes carpetas:

- **1 - Evaluación Técnica**. Contiene el documento de la evaluación técnica de la arquitectura y los diagramas con los cambios aplicados. En el documento de Evaluación Técnica de la Arquitectura de Software se encuentra la aprobación por parte de un revisor experto externo que revisó y aprobó la evaluación técnica realizada.

  - *1 - Documento de Evaluación Técnica de Arquitectura de Software.docx.*
  - *2 - Vaova-Arq. - Diseño Final.svg.*
  - *3 - Vaova-Deploy - Almacenamiento - Postevaluacion.svg.*
  - *4 - Vaova-Deploy - Componentes y Dependencias - Postevaluacion.svg.*
  - *5 - Vaova-Deploy - Comunicación Interna - Síncrona - Postevaluacion.svg.*
  - *6 - Vaova-Deploy - Comunicación Interna - Asíncrona - Postevaluacion.svg.*
  - *7 - Vaova-Deploy - Comunicación Interna - Eventos - Postevaluacion.svg.*

- **2 - Evaluación Financiera**. Contiene los documentos generados para realizar la evaluación financiera.
  - *1 - Documento de Evaluación Financiera de Arquitectura de Software.docx.*
  - *2 - Documento Análisis Costos-ROI de Arquitectura de Software.xlsx.*
  - *3 - Estimación Vaova Travel - AWS Pricing Calculator.pdf.*
  - *4 - Vaova-Deploy - Almacenamiento - Postevaluacion.svg.*
  - *5 - Vaova-Deploy - Componentes y Dependencias - Postevaluacion.svg.*
  - *6 - Vaova-Deploy - Comunicación Interna - Sincrona - Postevaluacion.svg.*
  - *7 - Vaova-Deploy - Comunicación Interna - Asincrona - Postevaluacion.svg.*
  - *8 - Vaova-Deploy - Comunicación Interna - Eventos - Postevaluacion.svg.*
  - **9 - Itinerarios**. Esta carpeta contiene los diagramas de secuencia para las historias épicas de Itinerarios.
    - *1 - Login Usuario.puml.*
    - *2 - Itinerarios - Login de Usuario.svg.*
    - *3 - Recuperar Contrasenia.puml.*
    - *4 - Itinerarios - Recuperar Contraseña.svg.*
    - *5 - Crear Usuario.puml.*
    - *6 - Itinerarios - Crear Usuario.svg.*
    - *7 - Crear Itinerario.puml.*
    - *8 - Itinerarios - Crear Itinerario.svg.*
    - *9 - Editar Itinerario.puml.*
    - *10 - Itinerarios - Editar Itinerario.svg.*
    - *11 - Listar Itinerarios.puml.*
    - *12 - Itinerarios - Listar Itinerario.svg.*
    - *13 - Filtrar Itinerario.puml.*
    - *14 - Itinerarios - Filtrar Itinerario.svg.*
    - *15 - Eliminar Itinerario.puml.*
    - *16 - Itinerarios - Eliminar Itinerario.svg.*
    - *17 - Numero Estimado de Viajeros.puml.*
    - *18 - Itinerarios - Número Estimado de Viajeros.svg.*
    - *19 - Pais de Destino.puml.*
    - *20 - Itinerarios - País de Destino.svg.*
    - *21 - Fechas de Viaje.puml.*
    - *22 - Itinerarios - Fechas de Viaje.svg.*
    - *23 - Cantidad de Destinos.puml.*

- *24 - Itinerarios - Cantidad de Destinos.svg.*
- *25 - Listar Destinos.puml.*
- *26 - Itinerarios - Listar Destinos.svg.*
- *27 - Busqueda de un Destino.puml.*
- *28 - Itinerarios - Búsqueda de un Destino.svg.*
- *29 - Detalle Destino.puml.*
- *30 - Itinerarios - Detalle Actividad.svg.*
- *31 - Calendario de Actividades.puml.*
- *32 - Itinerarios - Calendario de Actividades.svg.*
- *33 - Listar Actividades.puml.*
- *34 - Itinerarios - Listar Actividades.svg.*
- *35 - Detalle Actividad.puml.*
- *36 - Itinerarios - Detalle Destino.svg.*

- **10 - Vaovapp**. Esta carpeta contiene los diagramas de secuencia para las historias épicas de Vaovapp.
  - *1 - Login Usuario.puml.*
  - *2 - Vaovapp - Login de Usuario.svg.*
  - *3 - Recuperar Contrasenia.puml.*
  - *4 - Vaovapp - Recuperar Contraseña.svg.*
  - *5 - Informacion de Usuario - TourHero.puml.*
  - *6 - Vaovapp - Información de Usuario - TourHero.svg.*
  - *7 - Informacion de Usuario - WeTravel.puml.*
  - *8 - Vaovapp - Información de Usuario - WeTravel.svg.*
  - *9 - Editar Informacion de Usuario.puml.*
  - *10 - Vaovapp - Editar Información de Usuario.svg.*
  - *11 - MyTrips.puml.*
  - *12 - Vaovapp - MyTrips.svg.*
  - *13 - Upgrade Single Room.puml.*
  - *14 - Vaovapp - Upgrade Single Room.svg.*
  - *15 - Tipo de Acomodación.puml.*
  - *16 - Vaovapp - Tipo de Acomodación.svg.*
  - *17 - Vuelos.puml.*
  - *18 - Vaovapp - Vuelos.svg.*
  - *19 - Informacion de Vuelos.puml.*
  - *20 - Vaovapp - Información de Vuelos.svg.*
  - *21 - Invitacion.puml.*

- *22 - Vaovapp - Invitación.svg.*
- *23 - Cancelar Invitacion.puml.*
- *24 - Vaovapp - Cancelar Invitación.svg.*
- *25 - Rechazar Invitacion.puml.*
- *26 - Vaovapp - Rechazar Invitación.svg.*
- *27 - Ultima Mujer.puml.*
- *28 - Vaovapp - Última Mujer.svg.*
- *29 - Invitacion Aceptada.puml.*
- *30 - Vaovapp - Invitación Aceptada.svg.*
- *31 - Join - Habilitar.puml.*
- *32 - Vaovapp - Join - Habilitar.svg.*
- *33 - Join - Terminar.puml.*
- *34 - Vaovapp - Join - Terminar.svg.*
- *35 - Inventario Habitaciones.puml.*
- *36 - Vaovapp - Inventario de Habitaciones.svg.*
- *37 - Inventario Vuelos.puml.*
- *38 - Vaovapp - Inventario de Vuelos.svg.*

- **11 - Tableros**. Esta carpeta contiene los diagramas de secuencia para las historias épicas de Tableros de Información.
  - *1 - Tableros de Informacion.puml.*
  - *2 - Tableros - Tableros de Información.svg.*

Bibliografía

Atlassian (2023). MTBF, MTTR, MTTF, MTTA: comprensión de las métricas de incidentes.

Barbacci, M., Ellison, R., Stafford, J., Weinstock, C., and Wood, W. (2001). Quality attribute workshops. Technical Report CMU/SEI-2001-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice (SEI Series in Software Engineering)*. Addison-Wesley Professional, 3 edition.

Bass, L., Clements, P., and Kazman, R. (2021). *Software Architecture in Practice (SEI Series in Software Engineering)*. SEI Series in Software Engineering, 4 edition.

DANE (2020). Boletín Técnico Cuenta Satélite De Turismo (CST). Technical report, Departamento Administrativo De Estadística Nacional.

Esaki, K., Azuma, M., and Komiyama, T. (2013). Introduction of Quality Requirement and Evaluation Based on ISO/IEC SQuaRE Series of Standard. *Trustworthy Computing and Services*, pages 94–101.

Evans, E. (2015). *Domain-Driven Design Reference*.

Gamma, E., Helm, R., Johnson, R., and Vlissided, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing (Hardcover))*. Pearson Education.

Henttonen, K., Matinlassi, M., Niemeläand, E., and Kanstrén, T. (2007). Integrability and extensibility evaluation from software architectural models - a case study.

Ionita, M. T., Hammer, D. K., and Obbink, H. (2002). Scenario-based software architecture evaluation methods: An overview. In *Workshop on methods and techniques for software architecture review and assessment at the international conference on software engineering*, pages 1–12. Citeseer.

ISO (2022). ISO/IEC 25010.

Jus, N. and Misrahi, T. (2021). Travel & Tourism Economic Impact 2021. Technical report, World Travel & Tourism Council.

Kazman, R., Abowd, G., Bass, L., and Clements, P. (1996). Scenario-based analysis of software architecture. Technical report, IEEE.

Kazman, R., Bianco, P., Ivers, J., and Klein, J. (2020). Integrability. Technical Report CMU/SEI-2020-TR-001, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Millett, S. and Tune, N. (2015). *Patterns, Principles, and Practices of Domain-Driven Design*. John Wiley & Sons.

Ministerio De Comercio, I. y. T. (2021). Política de Turismo Cultural. Technical report, Ministerio De Comercio, Industria y Turismo.

Myrbakken, H. and Colomo-Palacios, R. (2017). *DevSecOps: A Multivocal literature Review*.

Patidar, A. and Suman, U. (2015). A survey on software architecture evaluation methods. *International Conference on Computing for Sustainable Global Development*, pages 967–972.

Procolombia (2021). Turismo Cultural: una oferta atractiva y responsable.

Raza, A., Zafar, S., Rehman, S. U., and Khattak, U. (2019). Software Architecture Evaluation Methods: A Comparative Study. *International Journal of Computing & Communication Networks*, 1(2).

Rozanski, N. and Woods, E. (2012). *Software Systems Architecture*. Addison-Wesley.

UNWTO (2022). Global and regional tourism performance. Technical report, World Tourism Organization a UN Specialized Agency.

Vaova (2022). Los happenings: nuestra forma de hacer un turismo alternativo y sostenible.