



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Implementación e integración de APIs para el manejo de rutas y puntos de interés en plataforma web Amukan

Natanael Eliab Roque Ramírez

natanael.roque@usm.cl

Pamela Gatica
Profesora Guía

Franco Jorquera
Profesor Correferente

Resumen: El turismo en Viña del Mar enfrenta el desafío de la fragmentación de la información, ya que los visitantes deben recurrir a múltiples plataformas para acceder a datos sobre alojamientos, actividades y eventos. Esta dispersión dificulta la planificación de viajes y reduce la visibilidad de comercios locales y panoramas culturales. Con el objetivo de centralizar y personalizar la experiencia turística, este trabajo propone la implementación e integración de diversas APIs dentro de la plataforma web Amukan. El enfoque principal se orienta al cálculo de rutas en vehículo o a pie y la visualización de puntos de interés en base a un itinerario o la ubicación actual del usuario, ambos adaptados a las preferencias de este. La metodología incluyó un proceso de investigación y análisis de servicios disponibles, diseño de arquitectura basada en APIs REST, implementación de módulos de georreferenciación y cálculo de rutas mediante el algoritmo de Dijkstra, así como pruebas funcionales para validar la precisión de la información desplegada.

Palabras Clave: API, Mapa Interactivo, Plataforma Web, Puntos de interés, Rutas

1 Introducción

Al momento de planificar un viaje, los turistas se ven obligados a saltar constantemente entre diferentes páginas web (hospedaje, panoramas, restaurantes, eventos, entre otras), lo que genera una pérdida de tiempo y una experiencia fragmentada. Además, las plataformas actuales carecen de un nivel adecuado de personalización en sus recomendaciones, ya que no siempre consideran aspectos relevantes como presupuesto, edad, intereses o preferencias del usuario.

Por otro lado, los residentes de Viña del Mar enfrentan un desafío relacionado con el escaso nivel de involucramiento ciudadano en el desarrollo turístico de la ciudad. Según



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Implementación e integración de APIs para el manejo de rutas y puntos de interés en plataforma web Amukan

Nombre del candidato(a): Natanael Eliab Roque Ramírez

Carrera / Grado: Ingeniería en informática

Campus: Sede Viña del Mar

Departamento: Electrotecnia e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Pamela Alexandra Gatica Caballero en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 19/06/2026 Firma: 

Estudiante o Candidato(a):

Fecha: 19/06/2026 Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.



el PLADETUR 2025, más de la mitad de los vecinos y vecinas expresan una preocupación relacionada con la falta de involucramiento de la ciudadanía en el desarrollo del turismo, resaltando la necesidad de descentralizar la oferta turística existente [1][2]. Esto refleja también la importancia de fortalecer la promoción de actividades culturales, artísticas y de esparcimiento en el borde costero, fomentando así un mayor sentido de pertenencia y participación local.

A este panorama se suman los comercios locales, que padecen la falta de visibilidad digital, especialmente en temporada baja o cuando no se encuentran en sectores céntricos de la ciudad. Finalmente, instituciones y organizaciones vinculadas al turismo carecen de herramientas eficaces para analizar la información recolectada. La ausencia de paneles con indicadores clave (KPI) y datos sobre el comportamiento de los usuarios limita la capacidad de mejorar los servicios y optimizar la gestión turística en el futuro.

En base a estas problemáticas nace Amukan, la cual es una plataforma web que busca dar solución a estos desafíos dentro del turismo en Viña del Mar. Su propósito es centralizar en un solo lugar toda la información relevante para viajeros y residentes, abarcando alojamientos, transporte, actividades, eventos y comercios locales. De esta forma, elimina la necesidad de que los turistas deban navegar entre múltiples páginas, ofreciendo una experiencia simplificada y coherente.

La plataforma combina recopilación automática de datos con la colaboración activa de socios estratégicos, como instituciones públicas, organizaciones culturales y emprendimientos locales. Esta integración permite mantener una base de información actualizada, junto con potenciar la visibilidad digital de pequeños negocios que, como se mencionó anteriormente, carecen de canales de promoción efectivos, especialmente en temporada baja.

En el núcleo del sistema se encuentra un motor de recomendación y un perfilado dinámico de usuarios, diseñado para adaptarse a las preferencias, el presupuesto y la cantidad de días de viaje de cada persona. Gracias a ello, Amukan puede sugerir tanto panoramas espontáneos para el mismo día como itinerarios completos de viaje, ajustados a los intereses particulares de cada visitante. Junto con esto, Amukan cuenta con recomendación basada en la ubicación actual del usuario, por lo que residentes tienen la oportunidad de conocer actividades y eventos cercanos a ellos sin la necesidad de búsqueda fuera de la página. Esta personalización eleva la calidad de la experiencia turística, a la vez que fomenta la participación de los residentes en actividades culturales y artísticas locales.

Además de facilitar la organización de viajes y el involucramiento de residentes, Amukan genera un valor adicional para empresas, instituciones y organismos relacionados al turismo, al proporcionar paneles con indicadores clave (KPI) y estadísticas de comportamiento de usuarios. Esta información resulta fundamental para mejorar la gestión de servicios, planificar estrategias de promoción y tomar decisiones basadas en datos.

En resumen, Amukan simplifica la planificación turística en un único sitio y, al mismo tiempo, también contribuye a la descentralización de la oferta, fortalece el ecosistema



local y genera una experiencia de viaje ágil, cómoda y personalizada, beneficiando por igual a turistas, residentes, comercios e instituciones.

La importancia de este trabajo recae principalmente en cómo la implementación e integración de APIs y el uso de un mapa interactivo aporta significativamente al punto más importante de la plataforma, la creación de itinerarios personalizados, optimizando el manejo y visualización de la información turística transformando datos estáticos en datos dinámicos centrados en la experiencia, incrementando la inmersión de esta y ayudando a la retención de usuarios dentro de la plataforma.

El despliegue de ubicaciones de puntos de interés y la generación de rutas tanto en vehículo como a pie facilita el desplazamiento a las actividades dentro del itinerario, al mismo tiempo que enriquece la experiencia de navegación y recomendación dentro de la página web. De esta manera se entrega un valor práctico a los turistas que desconocen la ciudad, ofreciendo un contexto más claro sobre distancias y proximidad que un texto por sí solo no lograría transmitir, y además se potencia la visibilidad de comercios y panoramas locales que podrían pasar desapercibidos en otras plataformas, las cuales suelen ser generalistas. [9]

Por lo tanto, el trabajo realizado se asegura de que los usuarios cuenten con una herramienta que convierta datos turísticos abstractos en una visualización de la información clara y actualizada sobre cómo llegar a sus destinos, optimizando el tiempo de traslado, mejorando la organización de su viaje e impulsando la capacidad del usuario de tomar decisiones informadas basadas en la información visual que ofrece un mapa interactivo con las funcionalidades mencionadas.

Por último, es importante señalar que para el desarrollo del proyecto fue necesario considerar diversas limitaciones para llevarlo a cabo, lo cual influyó en la toma de decisiones tanto técnicas como estratégicas del equipo. En primer lugar, el tiempo de desarrollo disponible, se contó con un plazo acotado de tiempo con entregas iterativas de un producto mínimo viable (MVP) para realizar una evaluación y ajuste de funcionalidades en cada una. En segundo lugar, se encontraron restricciones relacionadas con el hardware donde se está desarrollando el sistema, la extracción, procesamiento de datos y el uso de contenedores puede llegar a ser muy demandante dependiendo del equipo que se esté usando, dado que el desarrollo se realizó utilizando equipos personales, el procesamiento varió según el hardware disponible en donde se realizó cada tarea. Finalmente, se encuentra una falta de presupuesto o un presupuesto nulo, lo que llevó a optar por recurrir a recursos y herramientas de código abierto que aseguren una sostenibilidad técnica en la plataforma sin comprometer la calidad entregada en esta.

2 Marco Conceptual

2.1 Plataformas digitales de turismo

Son sitios y aplicaciones que facilitan la planificación, reserva, ejecución de viajes, búsqueda de panoramas, etc. Ofreciendo reseñas, opiniones y recomendaciones de otros viajeros sobre las distintas aristas de un viaje. Estas páginas si bien son útiles en conjunto, individualmente no logran satisfacer todas las necesidades de un viajero, sufriendo una falta de centralización de estas características y contando con una falta de mayor personalización de las preferencias del usuario. [3]

La plataforma busca resolver precisamente esta fragmentación, centralizando en un solo sitio la información turística de Viña del Mar. A diferencia de otras plataformas, Amukan no solo recopila información de distintas fuentes, sino que además la organiza y personaliza mediante su motor de recomendaciones y la integración de APIs, ofreciendo a los usuarios itinerarios completos adaptados a sus preferencias, presupuesto y tiempo de viaje.

2.2 API

API (Application Programming Interface) o interfaz de programación de aplicaciones, es un conjunto de reglas o protocolos que permiten que las aplicaciones de software se comuniquen entre sí para intercambiar datos, características y funcionalidades. Permitiendo a los desarrolladores integrar datos, servicios y capacidades de otras aplicaciones, en lugar de desarrollarlas desde cero. [4]

2.2.1 Arquitectura de APIs

Define el diseño y los procesos para crear y gestionar Interfaces de Programación de Aplicaciones (APIs), se centra en la interacción entre sistemas y la creación de componentes reutilizables. Entre sus principios se incluye la definición de endpoints claros, la forma de la solicitud/respuesta y el uso de protocolos estándar como HTTP. Las arquitecturas de API modernas facilitan la creación de microservicios, la gestión de la seguridad y la escalabilidad de las aplicaciones, a menudo a través de gateways de API. [5]

2.2.2 API REST

Es un estilo de arquitectura que sigue principios como la Transferencia de Estado Representacional (REST), utilizando HTTP para las operaciones (GET, POST, PUT, DELETE) y URLs para identificar recursos. [6]

El trabajo hizo uso de APIs como pieza fundamental para su funcionamiento. A través de ellas se integran servicios de georreferenciación (mapas), cálculo de rutas y visualización de puntos de interés cercanos. Se adopta el modelo REST por su simplicidad y estandarización, lo que facilita la comunicación entre la plataforma y las distintas fuentes de datos externas, asegurando una implementación más fácil y una actualización constante de la información.

2.3 Teoría de Grafos

Es una rama de las matemáticas y la informática que estudia los grafos, estructuras compuestas por vértices (puntos) y aristas (líneas) que los conectan, las cuales representan relaciones o interacciones entre objetos.

Las rutas son modeladas mediante esta teoría, en donde las carreteras son aristas con pesos (distancia, tiempo, tráfico) y los puntos o nodos representan intersecciones.

2.3.1 Dijkstra

Técnica de teoría de grafos para encontrar el camino más corto entre un nodo origen y todos los demás nodos en un grafo, siempre y cuando los pesos de las aristas no sean negativos.

2.3.2 A*

Algoritmo de búsqueda informada o heurística que encuentra el camino más corto entre un punto de inicio y un punto de destino en un grafo.

2.3.3 OpenStreetMap (OSM)

Es una web que proporciona datos de mapas abiertos gratuitos y editables a todo el mundo, está construido por una comunidad global de voluntarios los cuales contribuyen para mantener datos de mapas sobre carreteras, senderos, estaciones, etc [13][14].

2.3.4 OSRM (Open Source Routing Machine)

Es un proyecto que consiste en un motor de alto rendimiento para el cálculo de rutas óptimas utilizando los datos de OpenStreetMap (OSM), siendo capaz de determinar la ruta más rápida o corta para autos, bicicletas y peatones. Es usado por desarrolladores para aplicaciones de navegación, optimización de logística y análisis de transporte [12][17].

La teoría de grafos es la base matemática detrás de los servicios de cálculo de rutas que se integran a través de APIs. Aunque Amukan no implementa directamente algoritmos como A* o Dijkstra, consume una variante de este último mediante proveedores externos. La decisión de usar el algoritmo de Dijkstra responde a la necesidad de trabajar con múltiples puntos de interés, donde se buscan calcular rutas óptimas que conecten distintos nodos intermedios, contrario a A* que es más eficiente al momento de calcular una ruta única desde un origen a un destino específico [7].

Esto permitió a la plataforma ofrecer a los usuarios rutas en vehículo o a pie, optimizadas en base a la distancia entre puntos de interés, facilitando la navegación del itinerario personalizado.

2.4 Integración en Plataformas Web

Consiste en la conexión y sincronización de forma efectiva de dos o más sistemas, aplicaciones o servicios para que trabajen juntos, compartan información y automaticen procesos. Estas integraciones se logran principalmente a través de APIs y plataformas de integración que actúan como intermediarios para facilitar la comunicación y la transferencia de datos entre sistemas, ya sean locales o en la nube.



La plataforma integra múltiples servicios en un único entorno digital, permitiendo a los usuarios acceder a toda la información turística desde un solo lugar. Mediante la integración de APIs, Amukan conecta mapas interactivos, puntos de interés y rutas. Este enfoque modular permite que la plataforma pueda evolucionar en el futuro, incorporando nuevas fuentes de datos y funcionalidades sin necesidad de rediseñar todo el sistema.

2.5 Georreferenciación

Consiste en el proceso de asignar coordenadas geográficas (latitud y longitud) a datos espaciales, tales como mapas, imágenes satelitales o fotografías, con el fin de alinearlos a ubicaciones reales en la superficie terrestre [10].

Para el contexto de Amukan, la georreferenciación es utilizada para transformar las coordenadas de los puntos de interés en marcadores colocados en ubicaciones reales dentro de la ciudad de Viña del Mar para que el usuario tenga una noción de la distancia entre distintos puntos de interés junto con la ubicación exacta de cada uno de estos.

2.6 Geolocalización

Capacidad de determinar la ubicación geográfica real de un dispositivo, objeto o persona mediante tecnologías como el GPS, señales Wi-Fi, redes celulares o direcciones IP, lo que permite su uso en aplicaciones para la navegación y/o seguridad [11].

Dentro de la plataforma, la geolocalización es utilizada para conocer la ubicación en tiempo real del usuario, los puntos de interés son desplegados al usuario en base a un radio de 500 metros acorde a esta ubicación.

3 Objetivos

3.1 Objetivo General

Implementar e integrar APIs para la visualización de información turística y la mejora de experiencia del usuario para la planificación y el desarrollo de sus viajes dentro de la plataforma web.

3.2 Objetivos Específicos

1. Integrar API de mapa y georreferenciación de código abierto enfocado en la ciudad de Viña del Mar que permita mostrar de manera clara y dinámica la ubicación de puntos de interés turístico, culturales y comerciales dentro de la plataforma.
2. Desplegar marcadores para las ubicaciones sugeridas según la selección de itinerarios personalizados
3. Desplegar marcadores para las ubicaciones de la categoría seleccionada en base a los datos recopilados

4. Integrar API de cálculo de rutas en vehículo y a pie para la posterior generación de rutas
5. Desplegar rutas entre distintas ubicaciones (Ruta sugerida para el usuario) según el itinerario y el día seleccionado en este
6. Integrar API de geolocalización para conocer la ubicación actual del usuario
7. Desplegar marcadores para ubicaciones de puntos de interés sugeridos teniendo en cuenta la ubicación actual del usuario

4 Metodología

El desarrollo general del proyecto para el sistema Amukan se apoyó en la metodología ágil Scrum, debido a la adaptabilidad y habilidad que puede llegar a lograr para los proyectos. Dividido en ciclos cortos para el trabajo llamados "Sprints", Scrum se basa en la iteración y la entrega continua de valor, facilitando la entrega incremental de funcionalidades completas y verificables. [8]

También, con el fin de mantener un registro y control de los avances de los integrantes del equipo, se hizo uso de la plataforma GitHub específicamente las herramientas de Git para el control de versiones y GitHub Project para el manejo del backlog.

4.1 Metodología de Desarrollo / Plan de Trabajo

Aplicando Scrum para el desarrollo del proyecto, con Sprints de 2 semanas:

- **Sprint 1:** Integración e implementación de API para mapa interactivo y despliegue en su vista correspondiente: Recolección de tiles de mapa y desarrollo del componente de mapa interactivo para su uso e integración en el frontend.
- **Sprint 2:** Despliegue de puntos de interés en el mapa usando base de datos local: Recolección de puntos de interés desde la base de datos y desarrollo de funciones para extracción de coordenadas y despliegue de marcadores en el mapa interactivo.
- **Sprint 3:** Integración e implementación de API de modelo de recomendaciones y despliegue de puntos de interés en base a de itinerarios y datos recopilados: Implementación de API y desarrollo de funciones para extracción de coordenadas, diferenciación entre categorías seleccionadas y despliegue acorde a la selección.
- **Sprint 4:** Integración e implementación de API de cálculo de rutas en vehículo y a pie y despliegue de estas en base a las actividades y puntos de interés dentro del itinerario y día seleccionado: Recolección y extracción de datos de rutas tanto en vehículo como a pie, implementación de contenedores para host local de cálculo de rutas, desarrollo de funciones de diferenciación de itinerario y día seleccionado, cálculo y despliegue de rutas.



- **Sprint 5:** Implementación de API de geolocalización y despliegue de la ubicación actual del usuario y puntos de interés cercanos a él: Implementación de API, desarrollo de funciones para extraer la ubicación del usuario, despliegue de ubicación, despliegue de radio relacionado a la posición del usuario y despliegue de puntos de interés dentro del radio.
- **Sprint 6:** Pruebas finales de la plataforma web ya desplegada y arreglo de bugs menores

A continuación, se muestran las etapas que se realizaron dentro de cada sprint:

4.1.1 Investigación y Análisis

En esta etapa se recopiló y estudió la información necesaria para comprender el problema y definir los requerimientos de la plataforma. Incluyó la investigación de APIs disponibles (mapas, ubicación del usuario, puntos de interés, etc.), la identificación de limitaciones técnicas y análisis de compatibilidad con Amukan. Se tuvo como objetivo asegurar que las acciones posteriores a esta etapa tuvieran un funcionamiento garantizado.

4.1.2 Implementación

Corresponde al desarrollo práctico de la integración de APIs y funciones relacionadas al procesamiento de los datos dentro de la plataforma web Amukan. Se programaron los módulos encargados de consumir los datos externos para luego trabajarlos y desplegarlos de manera interactiva en la interfaz web. En esta etapa se utilizaron tecnologías web, librerías de mapas y herramientas de desarrollo que aseguren eficiencia y un correcto manejo de la información turística brindada por la base de datos y los recursos de código abierto.

4.1.3 Pruebas

Una vez implementadas las funcionalidades, se realizaron pruebas individuales y controladas para verificar el correcto funcionamiento de estas antes de su integración. Se evaluaron aspectos como la exactitud de las rutas, el despliegue de los puntos de interés, la velocidad de carga y la estabilidad del sistema frente a diferentes escenarios de uso. También se simulan posibles errores, como la petición repetitiva de una ruta o el despliegue entre categorías, para comprobar que el sistema tenga mecanismos de respuesta adecuados y mantenga una experiencia fluida para el usuario.

4.1.4 Mejora Continua

La plataforma se concibe como un sistema en constante evolución. Por ello, tras las pruebas y la integración con otras funcionalidades de la plataforma, se planteó una etapa de mejora continua que incluye el manejo de errores menores y la adaptación de cada funcionalidad acorde a necesidades futuras del proyecto y al feedback de los usuarios. Este proceso asegura que Amukan se mantenga actualizado, alineado con las demandas del turismo, generando valor tanto para visitantes como para residentes.

4.2 APIs y Tecnologías Utilizadas

React

React es una biblioteca de JavaScript enfocada en la construcción de interfaces de usuario basada en componentes reutilizables, uno de los puntos más importantes de su uso en Amukan es el manejo y el renderizado dinámico de la interfaz DOM, permitiendo construir interfaces que requieren de una actualización constante, pero sin la necesidad de recargar la página en la que se visualiza la información. Dentro de las ventajas de React se encuentra la facilidad que se tiene para manejar APIs REST, por lo que la integración de estas se vuelve un proceso más ameno para las llamadas y recolección de los datos brindados por estas, y también la integración de bibliotecas de manejo de mapas.

Django

El backend de la plataforma se encuentra construido en Django, un framework orientado al desarrollo de aplicaciones web, este funciona como el núcleo para gestionar los datos provenientes de los servicios turísticos y la comunicación con la base de datos, Django expone esta información mediante APIs REST, limpiando y unificando los datos antes de enviarlos al frontend, con el fin de que React reciba información consistente.

MapLibre GL JS

Es una librería para renderizar mapas interactivos en el navegador haciendo uso de WebGL, por lo que es el motor que permite mostrar los mapas, y en estos, añadir toda la información que permita al usuario tener una buena experiencia a la hora de revisar sus itinerarios personalizados o estar en búsqueda de panoramas cerca de él. Cualquier tipo de acción del usuario o cambio de estado se ve reflejado gracias a esta librería, la cual, comparada a otras, tiene una mayor precisión y otorga mayor control de la

personalización de aspectos visuales y de interacción necesarios para la correcta navegación del usuario.

OpenStreetMap API/OpenFreeMap

Para desplegar el mapa interactivo dentro de Amukan, la plataforma hace uso de tiles vectoriales obtenidos de un servidor compatible con OpenStreetMap. Estos tiles corresponden a pequeños fragmentos del mapa que al unirse, forman el mapa completo visible para el usuario.

La decisión de usar tiles vectoriales recae en las siguientes características:

- **Menor Peso Comparado con Otros Tipos de Tiles:** Los tiles vectoriales tienen un tamaño de archivo considerablemente menor en comparación a otros tiles como por ejemplo los raster tiles (imágenes pre-renderizadas), ya que estos almacenan datos geométricos en lugar de píxeles. Y, en lugar de descargar imágenes grandes para cada nivel de zoom, los tiles vectoriales usan puntos de datos compactos renderizados en el cliente, lo que facilita el almacenamiento temporal de copias de datos en caché y los mapas de alta resolución global sin sobrecargar los recursos.
- **Mayor Personalización:** Los tiles vectoriales permiten una personalización dinámica, ya que los estilos se aplican en tiempo real en el lado del cliente sin necesidad de descargar tiles nuevos. Además, se logran interacciones avanzadas que no son posibles con raster tiles, donde los cambios requieren regenerar las imágenes.
- **Mejor Rendimiento:** El rendimiento superior se consigue gracias al punto mencionado anteriormente, el tamaño reducido de las tiles, este menor tamaño de archivo y el renderizado en el cliente, se traduce en cargas más rápidas, mapas más eficientes y un menor consumo de recursos en general. No se requiere un procesamiento adicional para distintos tamaños de mapa o niveles de zoom dentro de estos, permitiendo mostrar detalles sin la necesidad de descargar datos extras. Para el caso de Amukan, esto lo hace muy eficiente con conjuntos de datos grandes y en dispositivos con recursos limitados, superando totalmente a los raster tiles que pueden causar demoras en la carga.
- **Animaciones más Suaves:** Como el rendimiento de los tiles vectoriales es muy alto, el despliegue en mapa es capaz de ofrecer animaciones más fluidas gracias al renderizado en tiempo real en el cliente, lo que permite que el zoom, rotaciones y transiciones entre puntos puedan llevarse a cabo sin perder calidad. Lo que ayuda a ofrecer una experiencia interactiva más inmersiva y cómoda al usuario, la cual puede adaptarse al dispositivo utilizado para usar la plataforma web[15][21].

Endpoint	Tipo de Request	Descripción
https://tiles.openfreemap.org/styles/liberty	GET	Realiza una petición a la API de OpenStreetMap para conseguir los tiles utilizados en cada vista que se necesite de un mapa.

APIs de Puntos de Interés

Dentro del backend de Amukan, la información turística se organiza en modelos independientes, ya que cada tipo de punto de interés contiene atributos distintos, y se requiere de campos específicos de cada uno para su adición a la base de datos. Por este motivo, se diseñaron distintos endpoints separados para cada categoría y estos son llamados por el frontend para consultar los distintos datos necesarios (Puntos de interés para cada categoría específica o itinerarios personalizados).

Los endpoints creados son los siguientes:

Endpoint	Tipo de Request	Descripción
api/destination/accommodations/	GET	Devuelve una lista de alojamientos disponibles dentro de la ciudad de Viña del Mar.
api/destination/activities/	GET	Obtiene actividades y puntos de interés en Viña del Mar, esto incluye centros comerciales, sitios turísticos, miradores, etc.
api/destination/events/	GET	Muestra eventos turísticos activos y programados dentro de la ciudad de Viña del Mar, junto con eventos que los usuarios con cuenta de organización hayan subido a la plataforma.
api/destination/places/?type=restaurant	GET	Este endpoint realiza una petición al backend para conseguir restaurantes y gastronomía en general dentro de la ciudad de Viña del Mar.
api/travel/preview-itinerary/	GET	Genera itinerarios personalizados en base a preferencias del usuario (presupuesto, número de personas, cantidad de días). Estos varían según el presupuesto teniendo la opción

		más económica, una estándar y una más costosa.
api/recommendations/services/?type=	GET	Genera recomendaciones personalizadas de puntos de interés asociados a las preferencias personales del usuario y la categoría especificada por este.

Esta distribución de endpoints otorga múltiples ventajas que permiten a la plataforma funcionar de mejor manera, como por ejemplo la eficiencia de las consultas que se hacen a la base de datos, ya que estas están filtradas a nivel de endpoint, y también, se encuentra la optimización del mapa interactivo, ya que el frontend puede solicitar solamente los puntos de interés necesarios según la vista actual, evitando consultas masivas o innecesarias

APIs de Rutas

Para el cálculo de rutas, Amukan usa un contenedor local basado en OSRM, la función de este contenedor es proveer endpoints que reciben coordenadas (latitud y longitud) enviadas por el frontend para retornar la mejor ruta posible ya sea en modo vehículo o a pie, dependiendo de las necesidades del usuario.

Construcción del Contenedor

Antes de que la plataforma sea capaz de solicitar rutas, el armado del contenedor sigue una serie de pasos internos obligatorios para que este contenga el motor de enrutamiento:

- **Carga de Datos OSM:** Se descargan archivos tipo “.osm.pbf”, un tipo de formato concebido como una alternativa al formato XML, este es usado comúnmente para el almacenamiento de grandes conjuntos de datos y para la distribución de la descarga de extractos de datos OSM, reduciendo el tamaño de los archivos y acelerando el procesamiento en aplicaciones de mapeo o análisis geográfico[16]. Para Amukan, se descargaron estos archivos acotando a la región de Valparaíso a áreas más pequeñas, estas siendo Quilpué-Viña del Mar-Concón, estos archivos contienen datos como las calles, sentidos de tránsito, restricciones, veredas y caminos, etc.
- **Extracción y Conversión:** Mediante comandos, se realiza la transformación de los datos, donde estos son modificados para obtener un grafo navegable y generando un archivo intermedio de tipo “.osrm” el cual contiene toda la red de caminos incluidos.
- **Levantamiento del Contenedor OSRM:** Finalmente se levanta el contenedor de enrutamiento, exponiendo determinados puertos dentro de la máquina local y

que al iniciarlo se estará ejecutando en segundo plano y escuchará peticiones a los puertos seleccionados.

Una vez que el contenedor se encuentra corriendo, el servidor OSRM expone una API REST en los puertos determinados, en donde se pueden hacer llamadas directamente desde el frontend, donde envían las coordenadas al contenedor, se reciben las rutas en formato GeoJSON y luego se renderizan.

Endpoint	Tipo de Request	Descripción
/route/v1/driving/	GET	Calcula una ruta vehicular entre los distintos puntos de interés definidos en un itinerario.
/route/v1/walking/	GET	Calcula una ruta peatonal (a pie) entre los distintos puntos de interés definidos en un itinerario.

API de Geolocalización

Para conocer la ubicación actual del usuario, se usa Web Geolocation API, que es una interfaz nativa de los navegadores que permite obtener la ubicación geográfica aproximada o precisa del dispositivo en el que se esté accediendo a la plataforma, utilizando diferentes mecanismos dependiendo del dispositivo, ya sea GPS, Wi-Fi, redes celulares, etc. El propósito de uso de esta API es que Amukan pueda acceder a latitud, longitud, precisión y dirección en la que el usuario se está moviendo a tiempo real o con un corto periodo de respuesta para asegurar que los puntos de interés se estén mostrando con una alta precisión basado en esta información.

A diferencia de otras APIs utilizadas en Amukan, Web Geolocation API no requiere endpoints propios, ya que su funcionamiento ocurre totalmente en el navegador del usuario.

4.3 Flujo General del Sistema

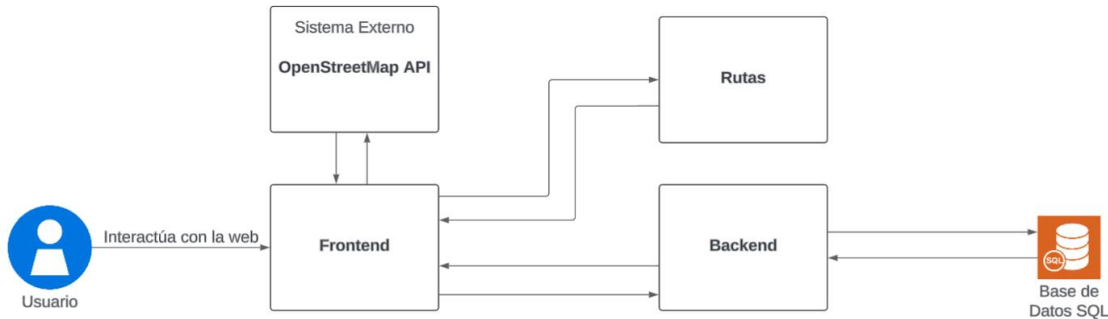


Figura 1. Diagrama de contenedores plataforma web Amukan

Fuente: Elaboración propia

El diagrama representa la comunicación entre los distintos contenedores que permiten que la plataforma funcione, desde que el usuario interactúa con la web hasta la recopilación de datos necesarios para el mapa, recopilación de datos turísticos y la generación de rutas.

A continuación, se explica más detalladamente el uso y funcionalidad de cada contenedor y sus interacciones con la plataforma.

Contenedor Frontend

Contiene todos los elementos mostrados en pantalla junto con la lógica de despliegue y manejo de los diversos datos conseguidos gracias a los demás contenedores, a medida que el usuario interactúa con este, se realizan las peticiones correspondientes para desplegar mapa, itinerarios, puntos de interés, rutas, etc.

Amukan hace uso de la API de OpenStreetMap para conseguir los tiles que luego se usan para desplegar el mapa en las vistas que lo necesiten, para esto, se realiza una petición al momento que se carga la vista por primera vez, donde se recolectan estos tiles y quedan almacenados en almacenamiento local para su uso, incluso si el usuario refresca la página. Por último, estos tiles son renderizados mediante MapLibre GL JS.

Para conseguir todos los puntos de interés que se deben mostrar al usuario, la plataforma realiza peticiones hacia el backend, el cual se alimenta con la base de datos de Amukan, estas peticiones se realizan mediante los distintos endpoints listados anteriormente dependiendo del punto de interés buscado en cada petición.

Por último, el contenedor de rutas es utilizado al momento de que el usuario busca conocer detalles dentro de un itinerario específico, por lo que la petición se realiza en ese instante y se "dibujan" mediante polígonos posicionados encima de la capa del mapa interactivo y, dependiendo de si el usuario desea ver las rutas entre puntos de interés a vehículo o a pie, estas se van refrescando en base a esto o al itinerario seleccionado.



Contenedor de Sistema Externo OpenStreetMap API

El sistema externo se encarga de proveer al frontend los tiles necesarios para el correcto despliegue del mapa interactivo, estos tiles no solo son estándar, sino que también pueden contener estilos o elementos personalizados que pueden ser útiles al momento del despliegue.

Contenedor Backend

El backend es encargado de interactuar con la base de datos con el fin de proveer toda la información necesaria de los puntos de interés para el frontend y su procesamiento, en él, se encuentra el modelo de recomendación el cual preprocesa los datos basados en los intereses y las preferencias del usuario, para luego enviársela al frontend al momento de realizar una petición.

Contenedor Rutas

Este contenedor es el encargado de devolver las rutas óptimas dependiendo de las coordenadas entregadas mediante el contenedor de frontend, al momento de realizar una petición hacia este contenedor, este procesa las coordenadas enviadas dependiendo del modo que el usuario haya elegido, por lo que el procesamiento de las coordenadas se hace en sub contenedores de rutas en vehículo y rutas a pie.

4.4 Preprocesamiento de Datos

Los datos recibidos desde el backend y el contenedor de rutas corresponden a objetos con formato JSON, por lo que antes de desplegarlos al usuario, se necesita de un preprocesamiento para extraer los datos importantes de cada punto de interés. Este preprocesamiento engloba las transformaciones, validaciones y adaptaciones que se realizan antes de que la información pueda ser desplegada en el mapa interactivo.

Extracción y Parsing de Coordenadas

El principal problema de las coordenadas es que son entregadas en formatos WKT, un formato de texto estandarizado para representar objetos geométricos y sistemas de referencia espacial de manera legible para las personas [18]. Esto es un problema ya que, para ser capaces de desplegar marcadores de puntos de interés, se necesitan datos numéricos, por lo que hay que formatear las coordenadas entregadas por el backend para su uso en el frontend.

La función utilizada para formatear las coordenadas es la siguiente:

```
const parseCoordinates = (wkt) => {  
  
  const match = wkt?.match(/POINT\s*(\s*(-?\d+(\.\d+)?)\s+(-?\d+(\.\d+)?)\s*)/i);  
  
  if (!match) return null;  
  
  const lon = parseFloat(match[1]);  
  const lat = parseFloat(match[3]);  
  
  if (isNaN(lat) || isNaN(lon)) return null;  
  
  return [lat, lon];  
};
```

Figura 2. Código utilizado para función de formateo de coordenadas

Fuente: Elaboración Propia

La función recibe las coordenadas en el formato mencionado ("POINT(-71.5516 - 33.0293)" por ejemplo), la expresión ".match" dentro de la variable busca coincidencias dentro de un string, todos los parámetros utilizados dentro de esta expresión buscan que las coordenadas no cuenten con caracteres innecesarios ni que puedan afectar al valor de las coordenadas finales, cada parámetro es explicado en la siguiente tabla:

Parámetro	Descripción
POINT	Busca exactamente la palabra POINT dentro de las coordenadas entregadas
\s*	La primera expresión \s hace referencia a espacio, y el * indica 0 o más veces, por lo tanto, permite que haya (o no) espacios después de la palabra POINT
\(	Se busca el carácter "(" de manera literal
\)	Se busca el carácter ")" de manera literal

Luego se encuentra la expresión "(-?\d+(\.\d+)?)", la cual es una fórmula utilizada para detectar números decimales [19][22][23].

Parámetro	Descripción
-?	Detecta números positivos o negativos (Los signos de estos)

<code>\d+</code>	Detecta números enteros, <code>\d</code> es cualquier número de (0-9) y <code>+</code> significa uno o más
Los siguientes puntos son para el parámetro (<code>\. \d+</code>)	
<code>\.</code>	Se busca el carácter “.” de manera literal
<code>\d+</code>	Busca dígitos después del punto
<code>?</code> (Final)	Hace que todo el bloque decimal sea opcional
<code>/i</code>	Ignora mayúsculas y minúsculas

Luego de finalizada esta expresión, se confirma si es que existe algún tipo de match en las coordenadas, o sea, si la entrada tenía el formato esperado, en caso de no tenerlo simplemente no se entrega nada, en caso de que sí, se extraen las coordenadas y se convierten en números utilizados en variables de latitud y longitud. Antes de retornar las coordenadas, se confirma que ninguna de estas sea nula para evitar errores posteriores, y finalmente se retorna un array con la latitud y longitud.

4.5 Despliegue en Mapa

Una vez ya preprocesados los datos, estos se transforman en elementos visuales dentro del mapa interactivo. A continuación, se describe de manera más detallada el proceso de despliegue de los puntos de interés:

- **Inicialización del Mapa:** Al momento que se carga el componente, se crea una instancia de “`maplibregl.Map`” y se le definen parámetros como el estilo base, el centro inicial del mapa, el zoom predeterminado, límites geográficos, etc.
- **Limpieza de Mapa:** Antes de mostrar cualquier punto de interés, el mapa elimina posibles marcadores existentes, posible marcador seleccionado y rutas previamente dibujadas, con el fin de asegurar que el mapa no muestre información residual.
- **Detección de contenido a desplegar:** Dentro de esta vista se manejan dos posibles escenarios:
 1. **Categorías Específicas (Gastronomía, actividades y eventos):** Cada punto de interés maneja coordenadas individuales y todos los marcadores añadidos tendrán el mismo color, ya que pertenecen a una misma categoría.
 2. **Categoría “Paquetes”:** Cada coordenada incluye el valor “Día del itinerario” a modo de identificador para el despliegue, el sistema agrupa automáticamente los puntos de interés por día mediante una función, se determina si la selección del usuario es para un día específico o para todos los días del itinerario y se les asigna distintos colores a los marcadores según el día para su diferenciación dentro del mapa.



- **Creación y Despliegue de Marcadores:** Una vez la lógica anterior decide que tipos de puntos de interés mostrar, se llama a la función general para agregar marcadores, la cual crea una instancia de nuevos objetos de tipo "maplibregl.Marker" y estos son configurados con los parámetros deseados (color, coordenadas, movimiento), luego son insertados dentro del DOM agregando el elemento HTML del marcador dentro del contenedor del mapa interactivo y por último se guarda la instancia de cada uno de los marcadores para limpiarla en el momento que se haga un cambio.
- **Ajuste de Cámara:** A modo de brindar una buena experiencia de usuario, la cámara general en la que se enfoca el mapa es cambiada para que todos los marcadores se muestren al mismo tiempo, por lo que usando las coordenadas de todos los marcadores recién creados se realiza un cálculo para "extender" los límites antes puestos y luego se realiza un ajuste automático para poner la cámara en un punto medio entre todos los marcadores.

Este proceso se realiza constantemente a medida que el usuario navega por las recomendaciones y puntos de interés brindados por la plataforma, sin contar la inicialización del mapa.

Proceso de Despliegue de Puntos de Interés

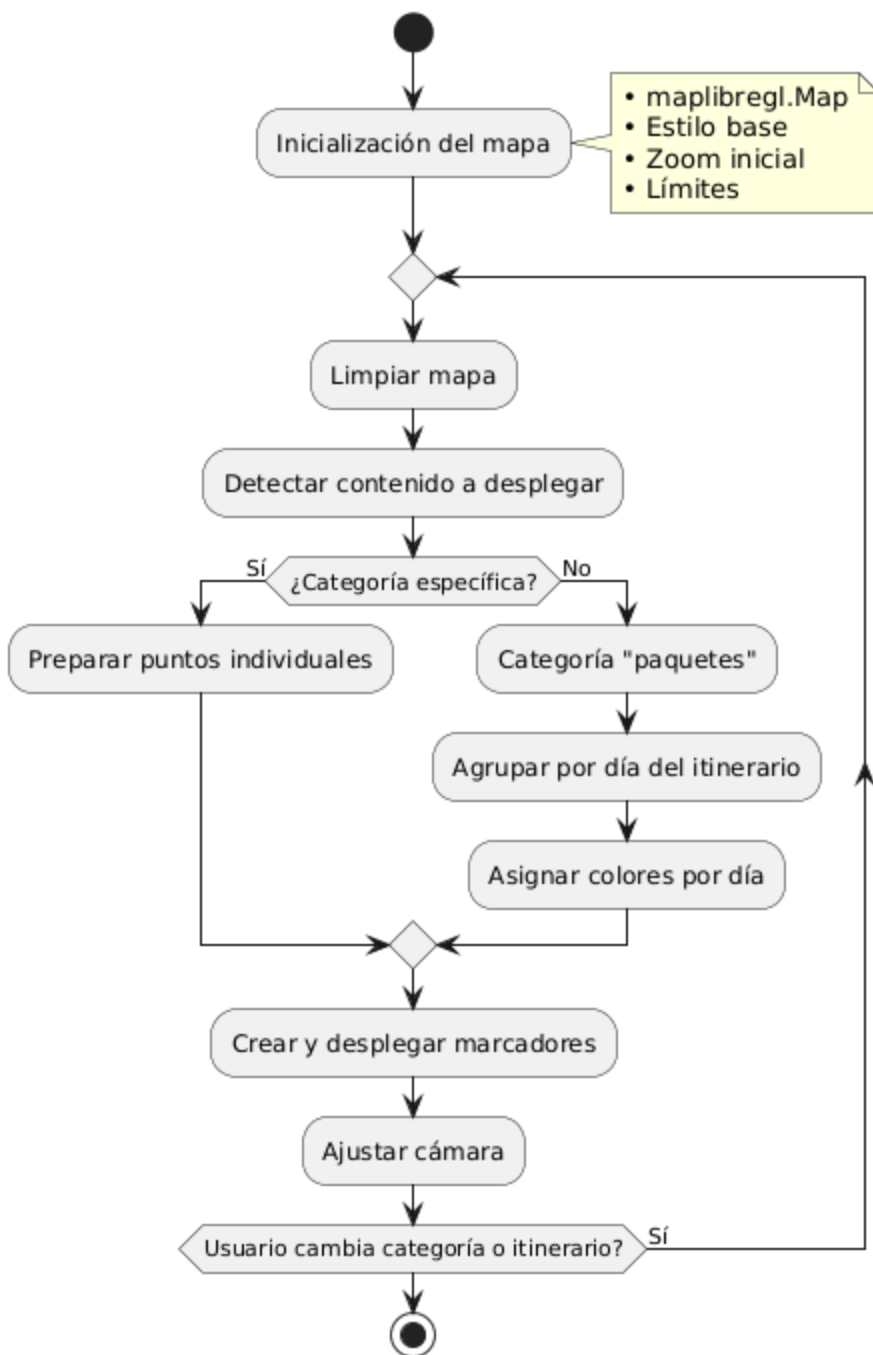


Figura 3. Diagrama de proceso de despliegue

Fuente: Elaboración propia

Las siguientes imágenes muestran el resultado final del despliegue de puntos de interés a partir de itinerarios y categorías específicas, junto con la selección de un punto de interés en específico.

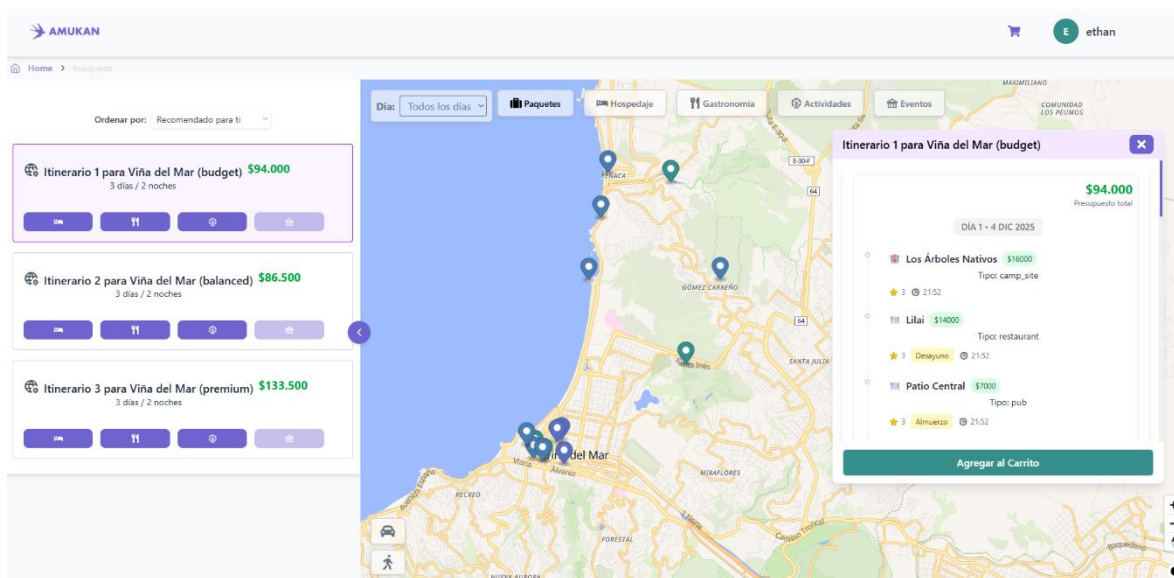


Figura 4. Despliegue de puntos de interés en un itinerario

Fuente: Elaboración propia

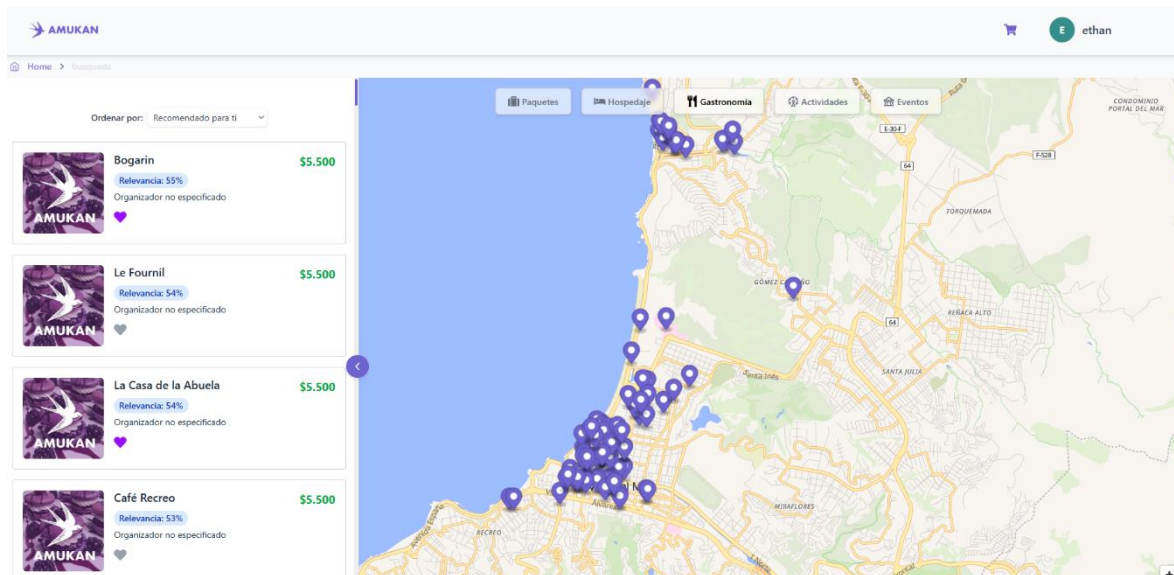


Figura 5. Despliegue de puntos de interés para una categoría específica

Fuente: Elaboración Propia

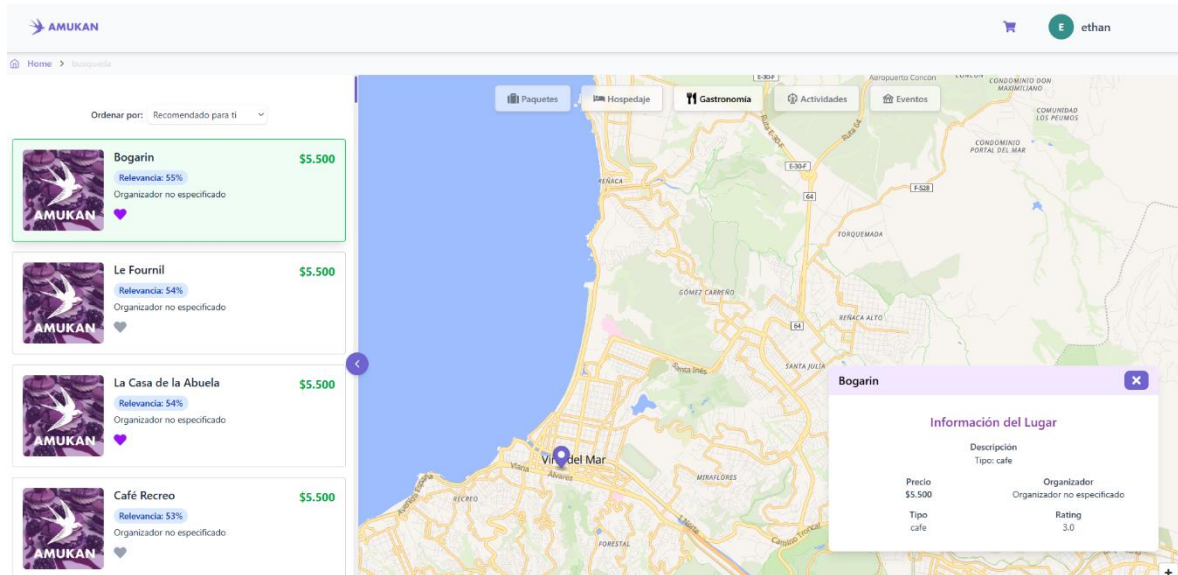


Figura 6. Selección de punto de interés específico

Fuente: Elaboración propia

4.6 Despliegue de Rutas

El proceso de despliegue de rutas es gatillado solamente cuando el usuario selecciona un día específico mientras se encuentra en la categoría "paquetes", la cual cuenta con 3 itinerarios que contienen múltiples días con puntos de interés asociados para distintas actividades recomendadas. El sistema utiliza estos puntos para calcular la ruta en el modo seleccionado por el usuario.

- **Preparación de Coordenadas:** Antes de realizar una llamada a los endpoints del contenedor de rutas, las coordenadas del día seleccionado deben ser transformadas a un formato específico, el cual requiere que todas las coordenadas sean un string separado por puntos y coma en el orden longitud-latitud.
- **Petición al Contenedor de Rutas:** El sistema realiza una petición al contenedor de rutas añadiendo como parámetro las coordenadas anteriormente formateadas y se añaden parámetros al endpoint para recibir la geometría de la ruta generada ("steps=true", "geometries=polyline"), optimizando el tamaño de la respuesta.
- **Transformación Geométrica:** Cuando se recibe la respuesta del contenedor confirmando la petición, se debe procesar la geometría de la ruta antes de poder dibujarla. La ruta que se recibe viene comprimida en un string codificado, por lo que se hace uso de la librería "mapbox/polyline" para transformar ese string en un array de coordenadas manejables. Por último,



el array es transformado nuevamente para ajustarse al formato apto para MapLibre.

- **Renderizado:** Para permitir la visualización de las líneas en el mapa, se verifica la existencia de una fuente de datos para el tipo de ruta mediante `"mapRef.current.getSource"`, en caso de existir, se actualizan los datos usando `".setData(geojson)"`, algo más eficiente que borrar y crear de nuevo, y en caso de que no exista, se añade una nueva fuente y capa para poder "dibujar" encima de esta. Por último, se asignan colores distintos para cada ruta para facilitar el entendimiento del tipo que se está visualizando.
- **Limpieza de Ruta Opuesta:** Para evitar el ruido visual, la web implementa una lógica de exclusión mutua, por lo que, si se dibuja un tipo de ruta, la otra será eliminada inmediatamente y viceversa, esto asegura que solo exista un trazado activo a la vez.

Proceso de Despliegue de Rutas en Mapa

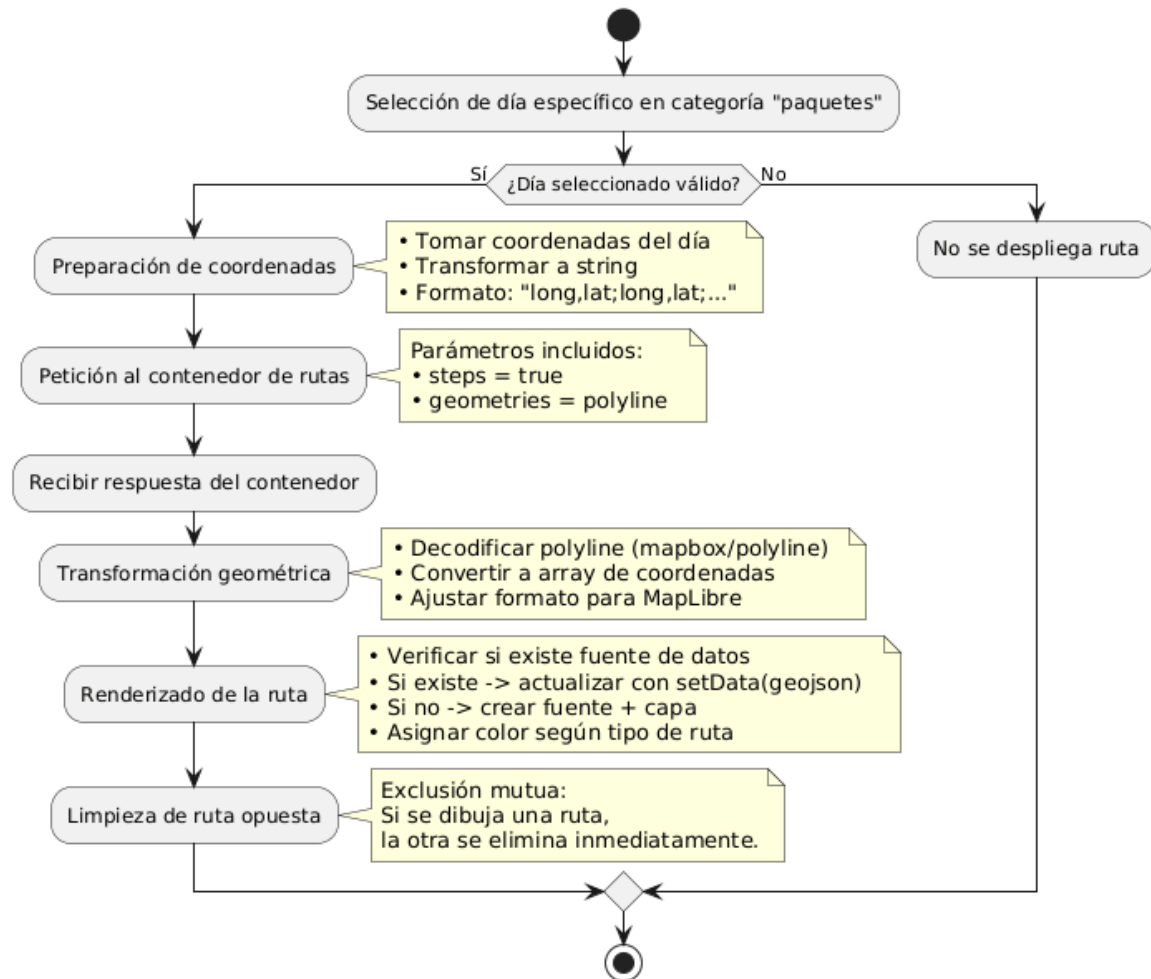


Figura 7. Diagrama de proceso de despliegue de rutas
Fuente: Elaboración Propia

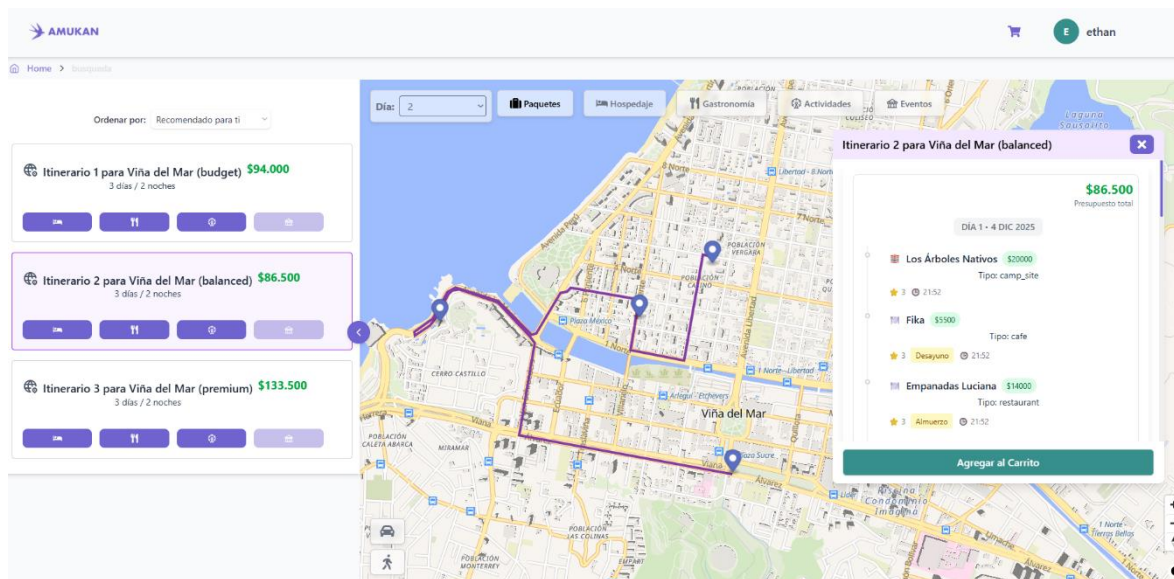


Figura 8. Despliegue de ruta vehicular

Fuente: Elaboración Propia

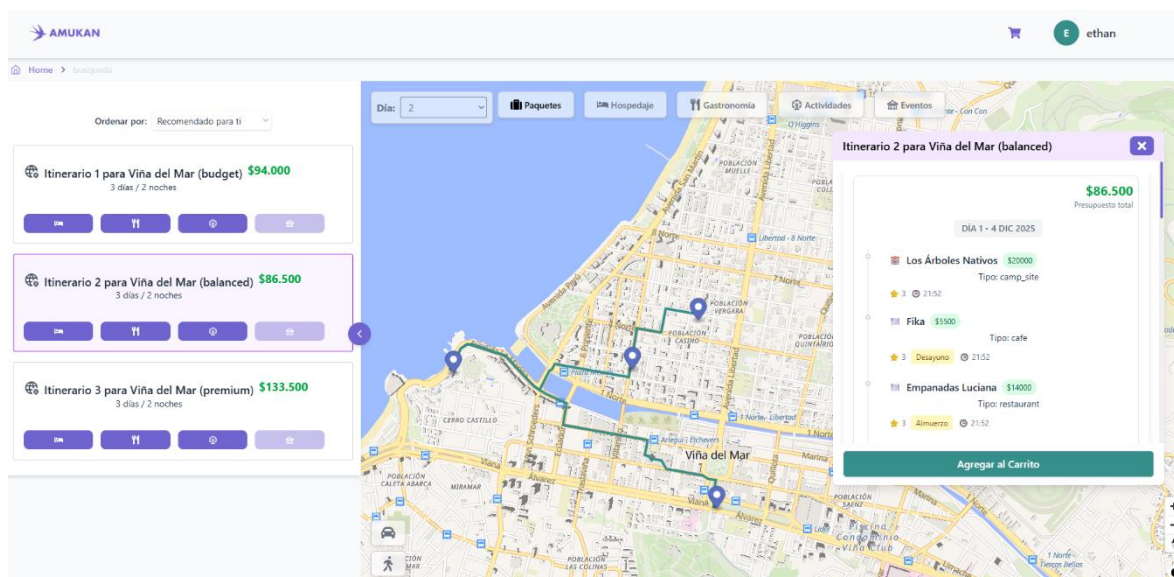


Figura 9. Despliegue de ruta peatonal

Fuente: Elaboración propia

4.7 Despliegue de Puntos Cercanos al Usuario

Con el fin de que los usuarios puedan visualizar puntos de interés que se encuentren próximos a su ubicación actual, Amukan ofrece un apartado dedicado a esto, siendo una experiencia más simple y rápida que la sección de itinerarios, presentando solo 3 categorías: gastronomía, actividades y eventos. Trabajos como la recolección y preprocesamiento de los datos se realizan de la forma explicada anteriormente, verificando que si el usuario se encuentra o no registrado para realizar las recomendaciones personalizadas y haciendo uso del endpoint correspondiente.

- **Inicialización y Geolocalización:** Si bien, al igual que en el proceso anterior, se inicializa una instancia de mapa tipo "maplibregl.Map", esta vez no se configura un centro con una coordenada fija, sino que se le asignan constantemente las coordenadas del usuario usando un listener de eventos que actualiza el estado en el momento que se reporta una nueva posición. Implementando el control nativo del navegador, el sistema permite mantener el seguimiento al usuario, y en caso de percibir movimiento, se solicitan nuevas coordenadas al GPS del dispositivo.
- **Visualización del Radio de Alcance:** Usando como referencia la ubicación del usuario como centro, en el mapa se dibuja un área que representa el rango en el que aparecerán puntos de interés. Se hace uso de la librería "Turf.js", una librería de análisis espacial que provee pequeños módulos para realizar operaciones espaciales, para calcular un polígono circular de 500 metros alrededor del usuario, el cual finalmente es insertado en el mapa.

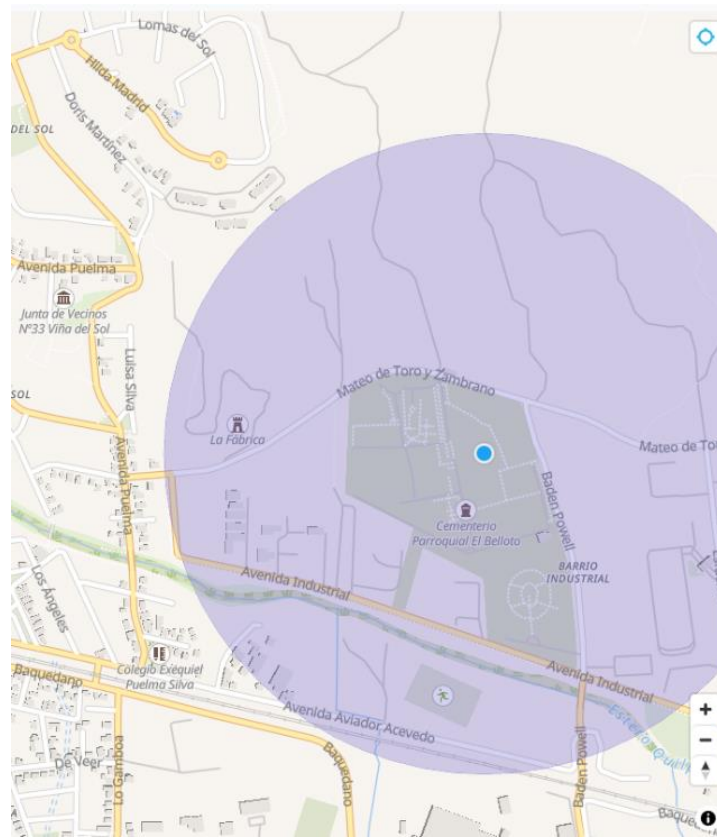


Figura 10. Radio creado a partir de la ubicación del usuario
Fuente: Elaboración propia

- **Filtrado Dentro del Radio:** Para encontrar todos los puntos de interés que están dentro del rango del usuario, se hace uso de la fórmula de haversine o fórmula del semiverseno, la cual se utiliza para calcular la distancia más corta entre dos puntos a partir de sus coordenadas [20]. Haciendo uso de esta fórmula, mediante una función se comparan las coordenadas de todos los puntos de interés encontrados, con las coordenadas de la ubicación actual del usuario para que estos sean visibles dentro del rango que marca el polígono creado anteriormente.
- **Marcadores:** Dentro de los datos de recomendación brindados por el backend, los puntos de interés se encuentran ordenados mediante un puntaje de mayor a menor que representa la similitud con los intereses del usuario, por lo que el primer elemento de cada array es la recomendación más ajustada a los intereses de este. Este punto de interés aparece dentro del mapa con un marcador especial para representar la mejor recomendación para el usuario, por lo que siempre será visible sin importar si se encuentra dentro del rango del usuario, mientras que los siguientes elementos si deberán estar en este rango para poder ser visibles.

- **Información en Marcadores:** Con el fin de mejorar la experiencia del usuario, se decidió implementar que, dentro de esta vista, los marcadores fueran clickeables para poder ver la información de cada punto de interés y que el usuario pueda tomar una decisión un poco más informada sobre lo que quiere visitar.

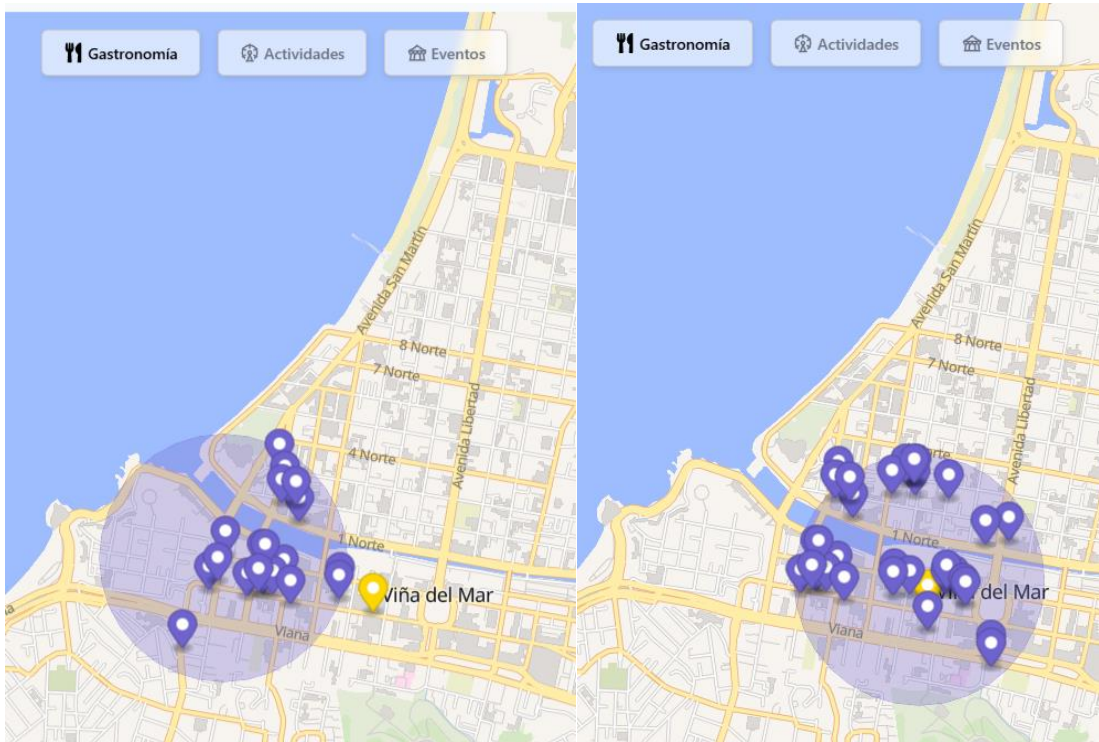


Figura 11 y 12. Despliegue de puntos cercanos
Fuente: Elaboración propia

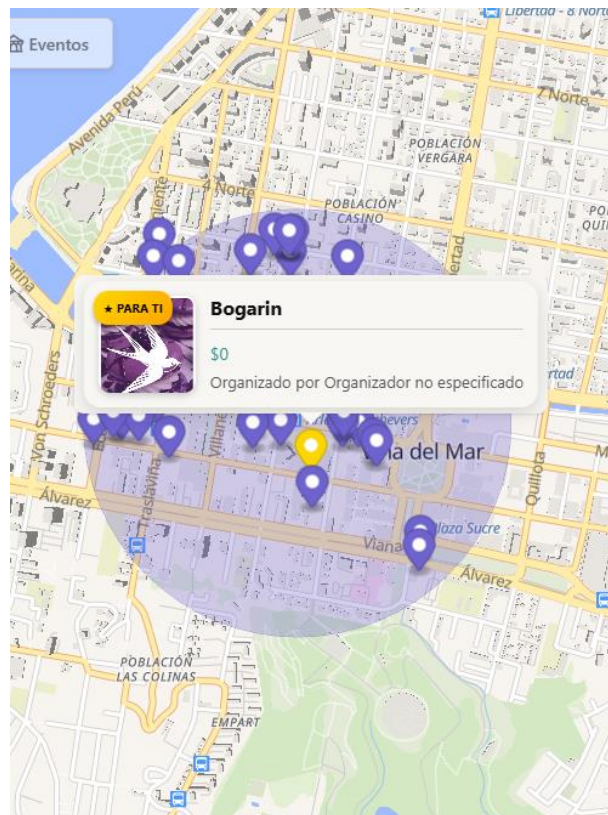


Figura 13. Popup de información de punto de interés más recomendado

Fuente: Elaboración propia

5 Pruebas de Sistema y Validación

Las pruebas realizadas dentro de la plataforma web tuvieron como objetivo comprobar la correcta integración y funcionamiento de las APIs de mapas, cálculo de rutas, geolocalización y recolección de puntos de interés. Dado las limitaciones técnicas de hardware, estas pruebas se realizaron en funciones y escenarios controlados, enfocados en evaluar la coherencia, precisión y utilidad de los datos obtenidos y las funciones implementadas.

5.1 Alcance de la Validación

Dentro del proceso de validación se tuvieron en cuenta los siguientes componentes del sistema con el fin de responder directamente a los objetivos específicos definidos para el trabajo:

- Visualización del mapa interactivo y carga de tiles vectoriales

- Despliegue de puntos de interés según itinerario o categoría seleccionada
- Cálculo y despliegue de rutas vehiculares y peatonales
- Obtención y uso de la ubicación actual del usuario
- Filtrado y coherencia de puntos de interés cercanos al usuario

5.2 Métricas y Pruebas Realizadas

Para evaluar el correcto funcionamiento del sistema, se definieron métricas básicas tanto cualitativas como cuantitativas:

- **Tiempo de Respuesta:** Se observó el tiempo requerido para la carga inicial del mapa, aparición de marcadores y generación de rutas tras la solicitud del usuario.

En las pruebas realizadas se midió el tiempo transcurrido desde que el cliente realiza una solicitud hasta que se recibe una respuesta de alguno de los contenedores o servicios y se despliega información en pantalla.

Servicio/Endpoint	Resultado Esperado	Resultado Real
Carga de Mapa (Tiles Vectoriales)	5 segundos	1-2 segundos aprox.
Cálculo y Despliegue de rutas	15 segundos	3 segundos aprox.
Despliegue de Puntos de Interés (Primera carga de la página)	10 segundos	5 segundos aprox.
Despliegue de Puntos de Interés (Navegación entre itinerarios y categorías)	3 segundos	<= 1 segundo

Tabla de comparación entre resultados

La tabla anterior muestra una comparación de los tiempos esperados y los resultados reales obtenidos para los principales servicios integrados en la plataforma web. Dentro de los resultados se logra apreciar la eficiencia del uso de tiles vectoriales y el renderizado en cliente mediante la librería MapLibre GL JS, lo que reduce considerablemente el tiempo de carga inicial del mapa.

Para el cálculo y despliegue de rutas, se estimó un tiempo de respuesta alto debido a la carga computacional asociada al procesamiento y transformación de datos geoespaciales, el resultado final evidencia un rendimiento muy favorable del motor de enrutamiento OSRM, validando su integración y su capacidad de generar rutas vehiculares y peatonales de forma eficiente.

Por último, para el despliegue de puntos de interés en la primera carga de la página se demuestra una eficiente comunicación entre el backend, el procesamiento de datos geoespaciales y el renderizado inicial de marcadores. Y al momento de navegar entre puntos de interés, ya sea entre itinerarios o

categorías, los tiempos de respuesta se encuentran bastante bajos, confirmando la actualización dinámica de marcadores, reutilización de la instancia del mapa y la limpieza de capas previas, permitiendo interacciones rápidas y fluidas sin la necesidad de recargar completamente la vista.

- **Precisión Visual de Rutas:** Se verificó que las rutas generadas fueran coherentes en términos de recorrido, continuidad y conexión entre puntos de interés.

Para validar la fiabilidad del motor de enrutamiento OSRM, se compararon múltiples rutas generadas en Amukan con el servicio de Google Maps en la ciudad de Viña del Mar.

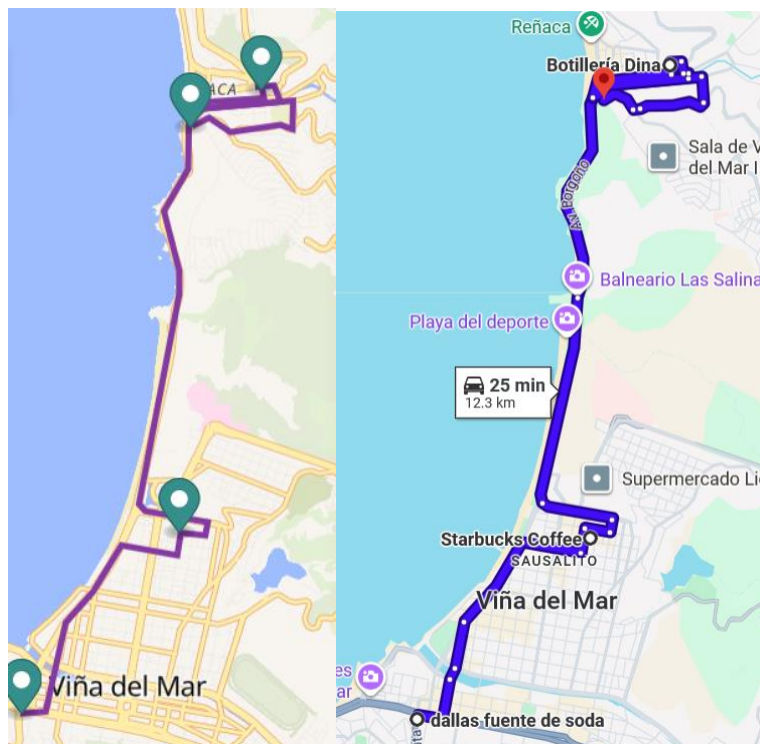


Figura 14. Ejemplo de comparación de ruta vehicular

Fuente: Elaboración Propia

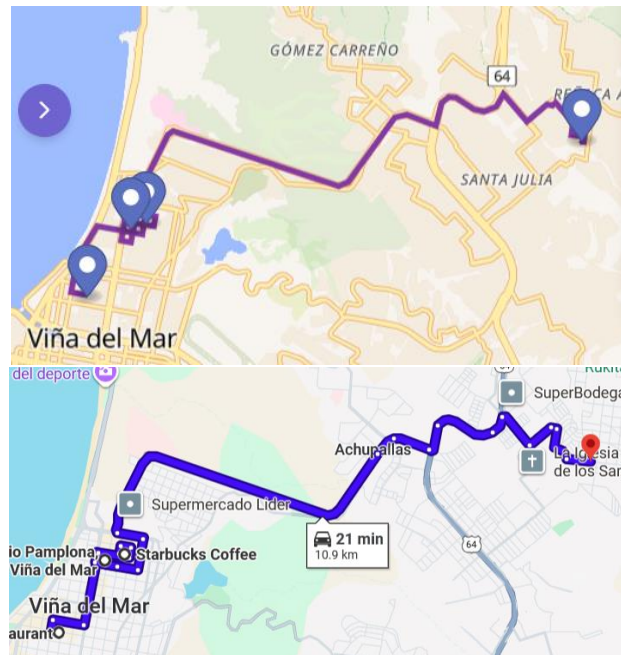


Figura 15. Ejemplo N°2 de comparación de ruta vehicular

Fuente: Elaboración Propia

En base a las pruebas realizadas, se demuestra que las rutas generadas mediante el motor de enrutamiento OSRM en Amukan poseen una coherencia de los recorridos obtenidos, las rutas muestran una continuidad, conexión lógica entre puntos de interés y trayectos equivalente a los propuestos por herramientas más utilizadas como lo fue en este caso Google Maps, por lo que se confirma la confiabilidad del motor de enrutamiento.

- **Posicionamiento Geográfico:** La validación de este módulo se centró en asegurar que la ubicación del usuario se obtenga y despliegue en posiciones geográficas coherentes y consistentes dentro del mapa, junto con garantizar que el filtrado de puntos de interés dentro del radio de 500 metros fuera preciso.

Se realizaron pruebas de campo utilizando dispositivos móviles y equipos de escritorio para obtener las coordenadas del usuario, junto con ver el posicionamiento del usuario en pantalla, estas coordenadas fueron ingresadas en un buscador de mapa mediante coordenadas GPS para comparar con el resultado entregado y la exactitud de la ubicación real.

Debido a que la intención inicial del componente de geolocalización es ser utilizado en dispositivos móviles, los resultados muestran que la API de geolocalización utilizada es eficaz para este propósito, esto gracias a que el

margen de error se mantuvo dentro de un rango más que satisfactorio para que el usuario identifique su posición relativa a los puntos de interés desplegados, no superando más allá de los 5 metros de diferencia. Por otro lado, el uso de la API de geolocalización varía mucho en la posición relativa del usuario dentro de equipos de escritorio, donde el hardware y la conexión utilizada son variables que afectan considerablemente al resultado final, este teniendo un margen de error de hasta 500 metros.

En cuanto al filtrado de puntos de interés, se aisló la función del cálculo de distancia basada en la fórmula del semiverseno. Se seleccionó una ubicación fija para el usuario y se comprobó que los puntos de interés desplegados correspondían al radio establecido.

Dentro del escenario de prueba, se posicionó al usuario en la Plaza de Viña con el radio de despliegue configurado a 500 metros, para realizar la validación, se corroboró que los puntos de interés que estuvieran a una distancia de hasta 500 metros del usuario estuvieran visibles, mientras que los que se encontraran en una distancia mayor se excluyeran del renderizado.

Estas pruebas confirmaron la correcta integración lógica entre la obtención de coordenadas del usuario y el filtrado de datos geoespaciales, asegurando que el usuario reciba la información correctamente.

En conjunto, los resultados obtenidos mediante las pruebas realizadas demuestran que la plataforma logra ofrecer una experiencia de usuario fluida y estable, con tiempos de respuesta bajos y una interacción dinámica del mapa, en donde la actualización eficiente de puntos de interés y la generación confiable de rutas, ofrecen una visión unificada de la ciudad de Viña del Mar y facilitan la planificación de actividades tanto para visitantes como para residentes.

6 Conclusiones

El desarrollo de Amukan permitió validar una viabilidad técnica y funcional de una plataforma web turística para la centralización de información, dentro del ámbito del trabajo realizado, esta viabilidad se ve demostrada en la exitosa integración de las múltiples tecnologías de código abierto dentro de una arquitectura modular, logrando construir un sistema funcional sin dependencia de servicios propietarios.

En cuanto al alcance del proyecto, se logró cubrir la necesidad de centralizar información turística en la ciudad de Viña del Mar, integrando mapas interactivos, cálculo de rutas y despliegue de puntos de interés en un entorno unificado. Sin embargo, estos resultados se vieron condicionados debido a las limitaciones técnicas mencionadas. Una limitante bastante significativa fue la dependencia de equipos personales para el desarrollo y despliegue de contenedores, especialmente para los contenedores en donde se encuentra el motor de enrutamiento y la extracción de mapas, ya que el procesamiento de datos geoespaciales tiene un gran costo computacional, esto influye directamente en la escalabilidad del sistema, donde a mayor área de descarga de datos OSM, se requiere de un mayor procesamiento y por lo tanto un hardware más robusto, de ahí la decisión

de acotar el área a la ciudad de Viña del Mar en específico, para mantener viabilidad dentro de un entorno local.

Dentro de otra de las limitantes, se encuentra el tiempo y metodología de trabajo, ya que el plazo acotado de desarrollo obliga a realizar un producto mínimo viable. Si bien, el uso de la metodología Scrum permitió entregas continuas funcionales, algunas funcionalidades más avanzadas quedaron obsoletas al momento de priorizarlas con otras más esenciales como la visualización de los puntos de interés y el ruteo.

Si bien se mencionó anteriormente la necesidad de un hardware más robusto para el cálculo de rutas, respecto a las tecnologías y herramientas utilizadas para el despliegue en sí, las restricciones del hardware y presupuesto, fuera de ser negativas, demuestran que es posible construir soluciones robustas sin costos de uso o licenciamiento, asegurando una sostenibilidad técnica del proyecto a largo plazo en cuanto al uso de estas.

Gracias a la arquitectura implementada en el trabajo realizado, se proponen diversas líneas de desarrollo a modo de trabajo futuro, especialmente haciendo uso del motor de enrutamiento OSRM, como por ejemplo, la implementación de datos de tráfico en tiempo real o información sobre el transporte público para el traslado entre puntos de interés y la creación de rutas en base a puntos de interés cercanos, lo que enriquecería la experiencia del usuario junto con añadir más valor al cálculo y despliegue de rutas.

Finalmente, el desarrollo de Amukan supuso un desafío técnico que requirió adquirir conocimientos de diversas áreas para llevarse a cabo, como lo fueron la integración de diversas tecnologías y su comunicación mediante APIs, buscando la consistencia de los datos intercambiados entre estas y el manejo de datos geoespaciales, enfrentando la complejidad de transformar coordenadas para su procesamiento y manejo en los distintos módulos de la plataforma.

Referencias

[1] Municipalidad de Viña del Mar and DTS Consultores, "INFORME FINAL PLADETUR SUSTENTABLE COMUNA DE VIÑA DEL MAR" Viña del Mar, Jun. 2025.

<https://visitavina.munivina.cl/wp-content/uploads/2025/10/INFORME-FINAL - PLADETUR-VDM.pdf#page=1>

[2] Municipalidad de Viña del Mar, "Informe-final_VD3-PLADECO.pdf" Viña del Mar, 2023. http://munivina.cl/wp-content/uploads/2023/11/Informe-final_VD3-PLADECO.pdf

[3] Doxee, "What is digital tourism? Much more than a definition" Oct. 2020. <https://www.doxee.com/blog/customer-experience/what-is-digital-tourism/>

[4] IBM, "What Is an API (Application Programming Interface)?" Sep. 2025. <https://www.ibm.com/think/topics/api>

[5] Catchpoint, "API Architecture Patterns and Best Practices" Oct. 2025. <https://www.catchpoint.com/api-monitoring-tools/api-architecture>



- [6] IBM, "What Is a REST API (RESTful API)?" <https://www.ibm.com/think/topics/rest-apis>
- [7] GeeksforGeeks, "Difference Between Dijkstra's Algorithm and A Search Algorithm" Jul. 2025. <https://www.geeksforgeeks.org/dsa/difference-between-dijkstras-algorithm-and-a-search-algorithm/>*
- [8] Atlassian, "What is Scrum?" <https://www.atlassian.com/agile/scrum>
- [9] Crocoblock, "Interactive Maps in Web Design: Benefits, Tools and Best Practices" Jan. 2025. <https://crocoblock.com/blog/interactive-maps-in-web-design/>
- [10] Alura Cursos Online, "Georreferenciación: qué es, sus usos y herramientas" São Paulo, Dec. 2023. <https://www.aluracursos.com/blog/georreferenciacion-que-es-sus-usos-y-herramientas#que-es-la-georreferenciacion>
- [11] INCIBE, "Geolocalización y privacidad en dispositivos móviles" Jul. 2024. <https://www.incibe.es/ciudadania/blog/geolocalizacion-y-privacidad-en-dispositivos-moviles>
- [12] LocationIQ, "What is OSRM?" 2025. <https://locationiq.com/glossary/osrm#>
- [13] OpenStreetMap Wiki, "About OpenStreetMap" Mar. 2025. https://wiki.openstreetmap.org/wiki/About_OpenStreetMap
- [14] OpenStreetMap, "Copyright and License" 2025. <https://www.openstreetmap.org/copyright>
- [15] OpenStreetMap Community, "OSM Vector Tile Capabilities and Advancements as of 2023?" Sep. 2023. <https://community.openstreetmap.org/t/osm-vector-tile-capabilities-and-advancements-as-of-2023/103620/4>
- [16] OpenStreetMap Wiki, "PBF Format" Aug. 2025. https://wiki.openstreetmap.org/wiki/PBF_Format
- [17] GitHub, "Project-OSRM/osrm-backend: Open Source Routing Machine" 2025. <https://github.com/Project-OSRM/osrm-backend>
- [18] Wikipedia, "Well-known text representation of geometry" 2025. https://es.wikipedia.org/wiki/Well_Known_Text
- [20] Wikipedia, "Fórmula del semiverseno" 2025. https://es.wikipedia.org/wiki/F%C3%B3rmula_del_semiverseno
- [21] Jawg Maps, "Why you should use vector tiles rather than raster tiles for your maps" Feb. 2024. <https://docs.mapbox.com/data/tilesets/guides/vector-tiles-introduction/>
- [22] MDN Web Docs, "Regular expressions" Jul. 2025. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_expressions



[23] MDN Web Docs, “Character classes” Jun. 2025.

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_expressions/Character_classes