TRABAJO DE GRADO
Opción Seminario-Diplomado.

Sistema de Marketplace para experiencias turísticas locales con arquitectura de microservicios

Corporación Universitaria Remington
Facultad de ingeniería
Ingeniería en sistemas

Daniel Camilo Rosero Pantoja
Tutor: Diego Fernando Marin Lozano

Opción de Trabajo de grado Seminario-Diplomado
2025

Dedicatoria

Este proyecto está dedicado a Dios y a mi familia, por su amor incondicional y apoyo constante, y a mí mismo, por la perseverancia y dedicación que me ha permitido llegar hasta aquí, y poder lograr una meta más, para así seguir cumpliendo cada meta con cada paso que doy.

Agradecimientos

Agradezco a Dios por permitirme llegar hasta acá, a mi familia por su apoyo incondicional y motivación constante, que me dio los ánimos y valentía para sacar este proyecto adelante. A mis profesores y compañeros, por compartir sus conocimientos y experiencias, que permitieron fortalecer las bases para este proyecto y me guiaron con los primeros pasos para entrar al mundo profesional. Y a todas las personas que, de alguna manera, contribuyeron a la realización de este trabajo.

Tabla de Contenidos

Resumen 5
Palabras clave 5
Pregunta orientadora de la búsqueda 6
Sustentación teórica de la pregunta 6
Problema 9
Metodología 10
Desarrollo 11
Componentes del sistema 11
Flujo del sistema 20
Conclusiones. 29
Referencias 30

Resumen

Este proyecto es un sistema de Marketplace especialmente orientado a la conexión entre guías locales y turistas para ofrecer experiencias turísticas, enfocado en una arquitectura de microservicios, con una gestión directa de las ofertas de experiencias de cada guía, la reserva y las valoraciones de estas experiencias.

Además, la implementación de contendores en Docker es fundamental para la instalación, despliegue y las pruebas del sistema en cualquier momento y en cualquier entorno, lo que permite levantar servicios y paquetes sin la necesidad de dependencias propias o locales.

El principal problema actual es la falta de un sistema organizado de experiencias turísticas, que ocasiona que muchos turistas lleguen a destinos sin planes definidos, enfrentando a guías locales informales que pueden aprovecharse de la condición de extranjeros y cobrar cifras exorbitantes por un servicio, lo que dificulta la organización y planificación de guías, tanto para los guías y para los turistas generando desconfianza e insatisfacción. (de Duque, Plazas, & Velez Rivas, 2011).

El objetivo de este proyecto es diseñar un sistema que permita conectar a los turistas con los guías, permitiendo crear experiencias con precios igualitarios para todos, y que los turistas pueden elegir cual es el más conveniente o atractivo para el y su familia, con la opción de poder reservar su experiencia de forma fácil y clara, y poder valorar su experiencia junto con un comentario; lo que permite la mejora en los servicios de los guías locales.

Palabras clave

Marketplace, Turismo, Microservicos, Docker, FastAPI, Flask

Pregunta orientadora de la búsqueda

¿Cómo puede ayudar un sistema basado en microservicios, en la gestión de experiencias turísticas conectando a guías con turistas resaltando la protección, facilidad y confianza entre estos?

Sustentación teórica de la pregunta

La gestión y organización de experiencias en el sector del turismo mediante medios tecnológicos ha sido un tema de debate entre diversos autores, lo cuales destacan que la idea de la digitalización de procesos de turismo es un gran avance, que permite mejorar la experiencia, seguridad y eficacia entre los involucrados, según autores como (Palomeque, Delgado, Vera, & Baidal, 2022) que indican que la digitalización del sector turismo representa un importante avance ya que permite una comunicación más formal y clara entre los guías y los turistas, lo que permite mayor seguridad y confianza entre las partes, esto es fundamental cuando vemos que los turistas quieren y deben tener una información clara, y a su vez lo guías no cuentan con los medios tecnológicos que les permitan solventar cierta demanda y poder ofrecer sus servicios.

En el turismo actual y local, un gran problema que se ve es la falta de plataformas turísticas que regulen y organicen ofertas de experiencias, lo cual genera en los turistas la duda en elegir un guía que sea totalmente confiable, ya que los precios siempre varían por parte de los guías cuando ven personas extranjeras, y los turistas no pueden retroalimentar a los servicios que se ofrecen para su posible mejora.

Cabe resaltar, que estudios han demostrado que una herramienta tecnológica que genere confianza entre las partes que interactúan, y más en lo de turismo es fundamental para un crecimiento se usuarios y generación de experiencias y reservas en la app (Marquez, 2019), permiten mayor aceptación entre los usuarios y mucha más visibilidad y usabilidad. Una plataforma tecnológica que permita dejar calificaciones, comentarios o

reseñas a una experiencia turística que tuvieron permite ayudar a disminuir la incertidumbre de este servicio, y mejorar la percepción del servicio para el guía, dando las bases para mejorar los errores, y felicitar todo lo bueno que ha realizado.

Para resolver esta problemática se propone solucionarla con el desarrollo de una plataforma basada en arquitectura de microservicios, de acuerdo con (Raj & David, 2020), los microservicios permiten separar un sistema completo, en demás partes o componentes independientes uno del otro, que cumplen con funciones definidas pero que se pueden comunicar entre ellos lo que permite seguir escalando el sistema, añadiendo mas microservicios o aumentando la capacidad o funcionalidad de cada uno, esto quiere decir que cada servicio puede seguir en constante crecimiento si modificar o cambiar la estructura total del sistema.

El uso de microservicios como respuesta a la pregunta, es que la implementación de este tiene una gran ventaja en cuanto a escalabilidad, ya que cada microservicio se encarga de una acción en específico, lo cual permite añadir módulos en un microservicio en espacial y esto hace que se pueda crecer y manjar de forma ordenada y más si hablamos en del tema de un Marketplace turístico ya que las reservas, valoraciones o autenticación pueden ampliarse o corregirse independientemente sin afectar el servicio general del sistema.

Como ayuda al sistema también se tiene la implementación de tecnologías como FastApi, que es forma una base sólida para construcción de servicios modernes destacando su rapidez de respuesta y su desarrollo limpio, al igual que la implementación de flask que aporta gran sencillez y facilidad para la creación de interfaces web ligeras, orientadas más que todo a la experiencia al usuario destacando la simplicidad y estilo minimalista para que cada usuario pueda usar el sistema sin perderse, la implementación de estas tecnologías permiten crear un estructura organizada tanto internamente, como externamente con la capa frontend que es la capa visual con la que interactúa el usuario.

Otro tema clave es el uso de contenedores Docker que según (Christopher Negus, 2015), permiten encapsular cada microservicio junto con sus dependencias o configuraciones garantizando el funcionamiento del sistema en cual entorno. Esto es importante ya que se puede crear un contenedor de cada microservicio para que opere aislado con sus dependencias o ajustes, pero permitiendo una comunicación entre ellos, además con Docker se tiene une ejecución estable, segura y ágil a la hora de desplegarlo y a su vez un despliegue en cualquier dispositivo, dejando atrás incompatibilidades o errores, o la típica frase “en mi computador si funciona”.

Finalmente, se puede decir que la sustentación teórica, respalda la idea de la creación del sistema ya mencionado, mejorando la comercialización en este asunto dando como resultado una superior experiencia a la hora de reservas, experiencias y valoraciones, con la ayuda de implementación de tecnologías como arquitectura de microservicios, FastAPI, Flask y Docker que permiten del desarrollo del sistema, un sistema escalable, modular y seguro, con la ayuda de los microservicios nos da un crecimiento, mejora y corrección de cada componente independientemente, con FastAPI y Flask conseguimos la creación de servicios rápidos y ligeros orientados al usuario, a vez que la implementación de Docker nos permite encapsular cada microservicio y un despliegue en cualquier entorno.

Problema

En el contexto actual, el turismo es una gran fuente de ingreso para los países promoviendo su economía interna y siendo una gran fuente de empleo para las personas locales, tanto guías turísticos como vendedores informales (Molina, 2025)**.** Sin embargo, la organización y gestión para la creación de experiencias por parte de los guías locales informales presenta muchas falencias y problemas, como el hecho que no cuentan con la tecnología para poder ofrecer sus servicios a los turistas.

Dicha falta de estas tecnologías y plataformas disminuye el desarrollo económico y genera barreras en las conexiones turísticas. Además, los turistas llegan con falta de confianza con los guías informales, ya que nadie regula sus precios y ofertas (Maul), creándose aquí la dificultad de los guías en no tener una herramienta tecnología para poder ofrecer sus experiencias, lo que impide un mayor número de reservas y retroalimentaciones de parte de los usuarios con sus valoraciones.

Según esto nace la necesidad de la creación de una herramienta tecnológica que permita a los guías locales informales, tener un medio por el cual pueden ofrecer sus guías y experiencias a los turistas, fomentando la transparencia, facilidad y seguridad tanto para los guías como para los turistas, permitiendo un relación más cercana y más confiable entre estas partes, de esto nace la pregunta orientada a la búsqueda.

La gestión y organización de experiencias en el sector del turismo mediante medios tecnológicos ha sido un tema de debate entre diversos autores, lo cuales destacan que la idea de la digitalización de procesos de turismo es un gran avance, que permite mejorar la experiencia, seguridad y eficacia entre los involucrados (Palomeque, Delgado, Vera, & Baidal, 2022).

Metodología

La metodología se basa en la creación de una arquitectura de microservicios en Python. Cada servicio tiene un rol especifico y corre dentro de un contenedor individual a través de Docker, lo que permite que el proyecto sea fácil de escalar, mantener y de desplegar (Martinez, Valderas, & Bosh, 2018). El diseño de cada microservicio en su contenedor es fundamental para el funcionamiento general del sistema, permitiendo la comunicación entre ellos y la integración con el frontend con interfaces intuitivas y fáciles de usar para que tanto el guía como el turista tengan las opciones básicas directas al grano, y les sea fácil la búsqueda y reserva de una experiencia u la personalización de una de estas (Aggarwal, 2023).

En los microservicios se puede destacar cada una de sus funcionalidades como en el de autenticación el cual permite distinguir y gestionar cada uno de los roles en el sistema, encargado de la seguridad en el ingreso personalizado, así como el de experiencias el cual permite la creación, edición y eliminación de las experiencias que el guía desee ofrecer junto con su descripción, precio y disponibilidad, este es indispensable para la organización de cada una de las experiencias, también el microservicio de reservas el cual permite gestionar y organizar la reservas de experiencias para los usuarios, pero con la excepción de que haya disponibilidad de este, y por último el de valoraciones que permite la recopilación y muestras de las valoraciones que cada turista hace a las experiencias y le muestra a cada guía para que pueda implementar mejoras o recomendaciones.

Para la organización del desarrollo se utilizó Scrumban, con la implementación de 3 sprints dividiendo el trabajo para su mejor gestión (Turley & Rad, 2019), los sprints son:

1. Desarrollo del microservicio de Autenticación.
2. Implementación de los microservicios de Experiencias, Reservas y Valoraciones.
3. Creación del API Gateway y desarrollo del frontend, integrando todos los servicios y haciendo pruebas de conectividad y funcionamiento.

Desarrollo

Componentes del sistema

El sistema en general cuanto con una estructura ya definida que permite una gestión integral de los archivos, y una organización de estos mediante carpetas.

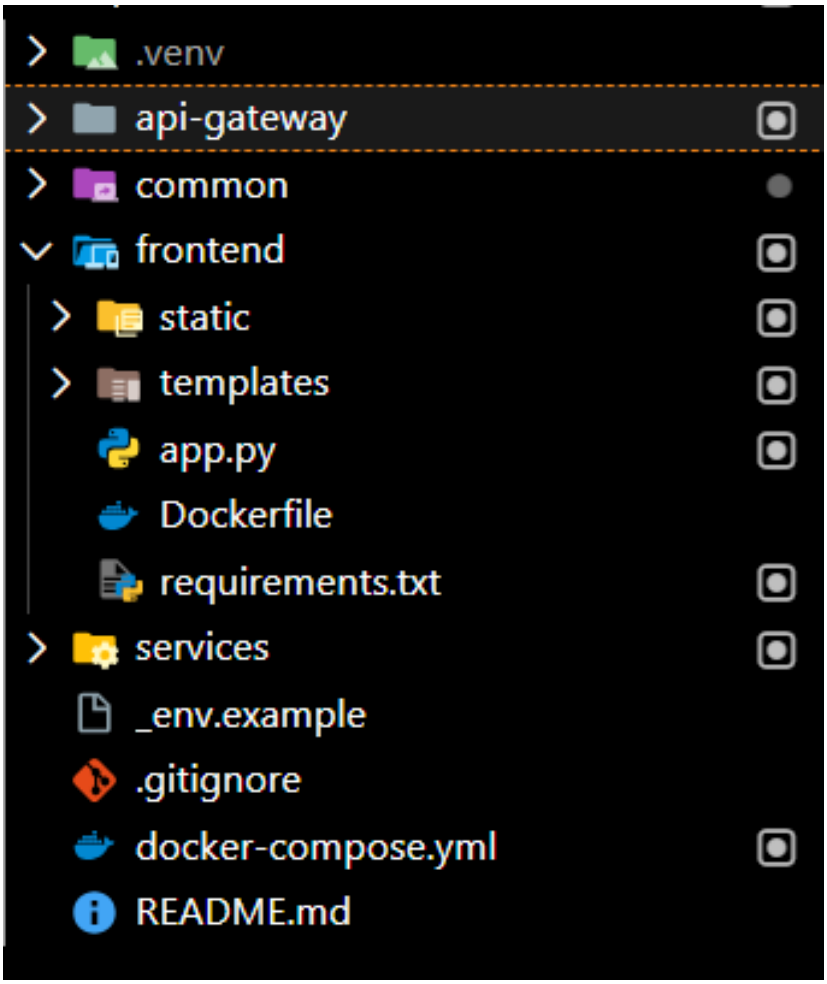

Figura 1. Estructura del proyecto

En esta imagen podemos observar las carpetas principales como api-getway en la cual se encuentra el punto de entrada principal para las solicitudes externas, este se encarga

de enrutar las peticiones a los microservicios correspondientes y aplicar lógica de autenticación y autorización.

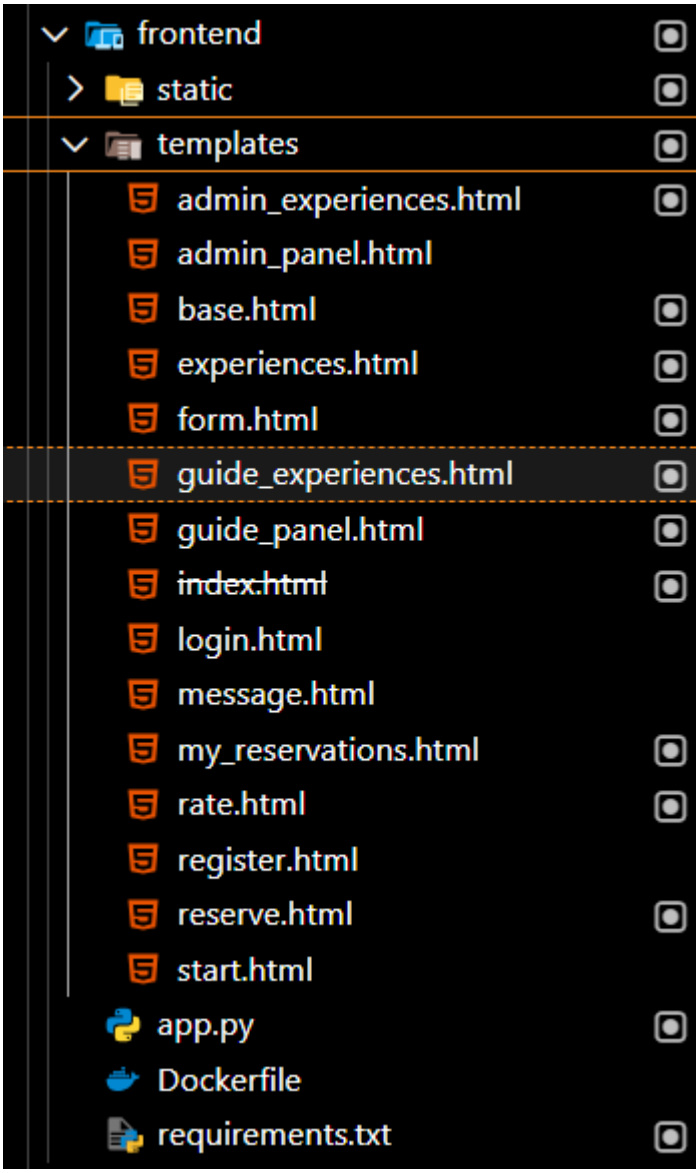

Figura 2. Carpeta frontend

Aquí también tenemos la carpeta frontend la cual contiene como principal aspecto los templates, es aquí donde están todas plantillas html las cuales son la base de nuestras

interfaces, que permiten la integración entre el usuario y el backend, permitiendo visualizar las opciones y realizar todas las tareas.

Estas plantillas permiten que el usuario interactúe con el backend, actuando como puente, para que el usuario visualice la información dependiendo de su rol, le permite también interactuar de forma dinámica con la ayuda de formularios para llenar y enviar las solicitudes a los microservicios, por ejemplo:

```
<h2>Reservar experiencia {{ exp_id }}</h2>
<form method="post">
    {% if session.get('username') %}
        <p>Reservando como: {{ session.get('username') </p>
        <input type="hidden" id="user_id" name="user_id" value="
            {{ session.get('username') }}">
    {% else %}
        <label for="user_id">Tu nombre (id):</label>
        <input type="text" id="user_id" name="user_id"
            placeholder="usuario123">
    {% endif %}

    <label for="date">Fecha:</label>
    <input type="date" id="date" name="date" required>
    <p id="date-error" style="display:none;">Fecha no válida</p>

    <label for="num_personas">Cantidad de personas:</label>
    <input type="number" name="num_personas" min="1" value="1" required>

    <label for="notes">Notas:</label>
    <textarea id="notes" name="notes"></textarea>

    <button class="btn" type="submit">Confirmar reserva</button>
</form>
```

El anterior código es del apartado de reserves.html, este está encargado de la interfaz gráfica para la reserva de una experiencia, si el usuario ya se está autenticado le muestra el nombre, el formulario incluye el selector de fecha, especificando que sea después del día en el que esta, también tiene el apartado de numero de personas y notas, al presionar el botón confirmar reserva, estos datos se envían mediante POST a la ruta del backend que corresponde a la experiencia

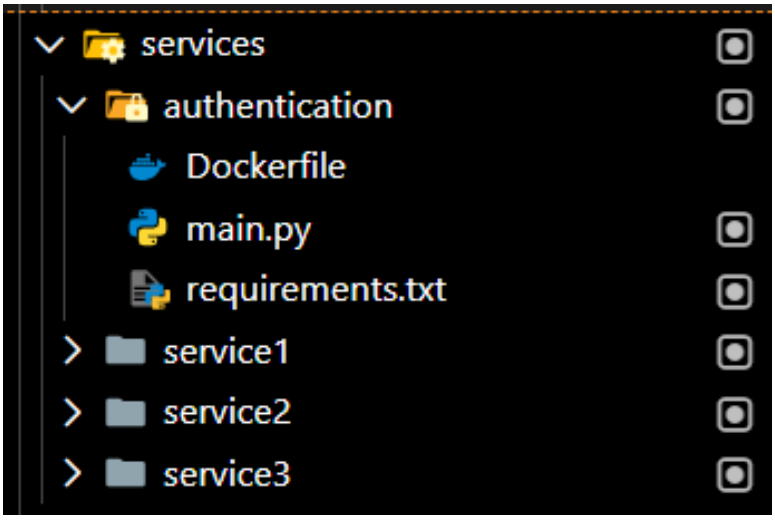

Figura 3. Carpeta services

En esta imagen podemos observar las carpetas de services, en esta parte vemos los microservicios, cada uno en una carpeta especifica con las funciones necesario para montar cada servicio.

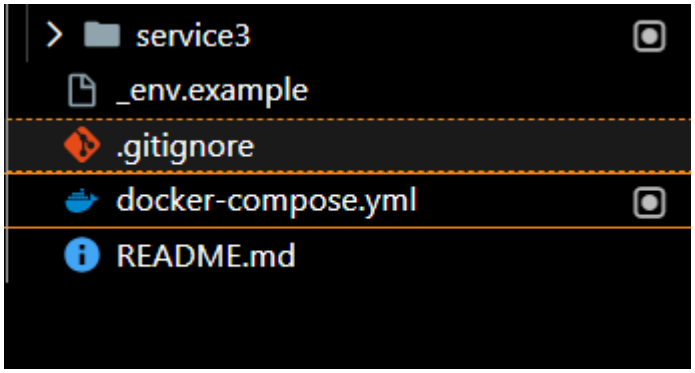

Figura 4. Archivo Docker-compose

Es el archivo que orquesta el despliegue de todos los servicios y la base de datos usando Docker, facilitando la ejecución y escalabilidad del sistema. Para ser más específicos miremos que tenemos en este archivo y ver que función cumple cada apartado.

```
frontend:
  build: ./frontend
  container_name: frontend
  ports:
    - "5000:5000"
  environment:
    - API_GATEWAY_URL=http://api-gateway:8000
  depends_on:
    - api-gateway
```

Este fragmento de código construye el contenedor del Frontend desde la carpeta /frontend y exponiendo el puerto 5000 para poder acceder a web desde ahí, y se configura URL del API Gateway para su acceso interno, pero espera a que este este disponible antes de iniciar.

```
api-gateway:
  build: ./api-gateway
  container_name: api-gateway
  ports:
    - "8000:8000"
  depends_on:
    - auth-service
```

En el código anterior esta la construcción de api-getway en el cual exponemos el puerto 8000 para su ejecución, este enruta todas las peticiones a su microservicio correspondiente, después tenemos el microservicio de autenticación:

```
auth-service:
  build: ./services/authentication
  container_name: auth-service
  ports:
    - "8001:8001"
  environment:
    - DATABASE_URL=mongodb://auth-db:27017/auth_db
  depends_on:
    - auth-db

auth-db:
  image: mongo:latest
  container_name: auth-db
  ports:
    - "27017:27017"
  volumes:
    - auth_data:/data/db
```

Como podemos ver en este código esta la configuración del contendor de autenticación junto con su base de datos, este servicio gestiona el registro y login de los usuarios mediante el puerto 8001, conectándose a su base de datos MongoDB dedicada y con su volumen auth_data para la persistencia de los datos, después de este tenemos los contendores de los microservicios, lo cuales tienen la siguiente estructura:

```
experiences-service:
  build: ./services/service1
  container_name: experiences-service
  ports:
    - "8002:8002"
  environment:
    - DATABASE_URL=mongodb://experiences-
db:27017/experiences_db
  depends_on:
    - experiences-db

experiences-db:
  image: mongo:latest
  container_name: experiences-db
  ports:
    - "27018:27017"
  volumes:
    - experiences_data:/data/db
```

En este código esta especificado el servicio de experiencia, la configuración para el contenedor gestiona el crud de las experiencias turísticas a través del puerto 8002, conectándose a su base de datos y con el volumen de datos en experiences_data para su persistencia. Así como este tenemos los demás microservicios como el de reservas y valoraciones, los cuales cuentan con la misma estructura que el de experiencias, lo que cambian son los puertos y sus volúmenes (reservas en el puerto 8004 y valoraciones en el 8005).

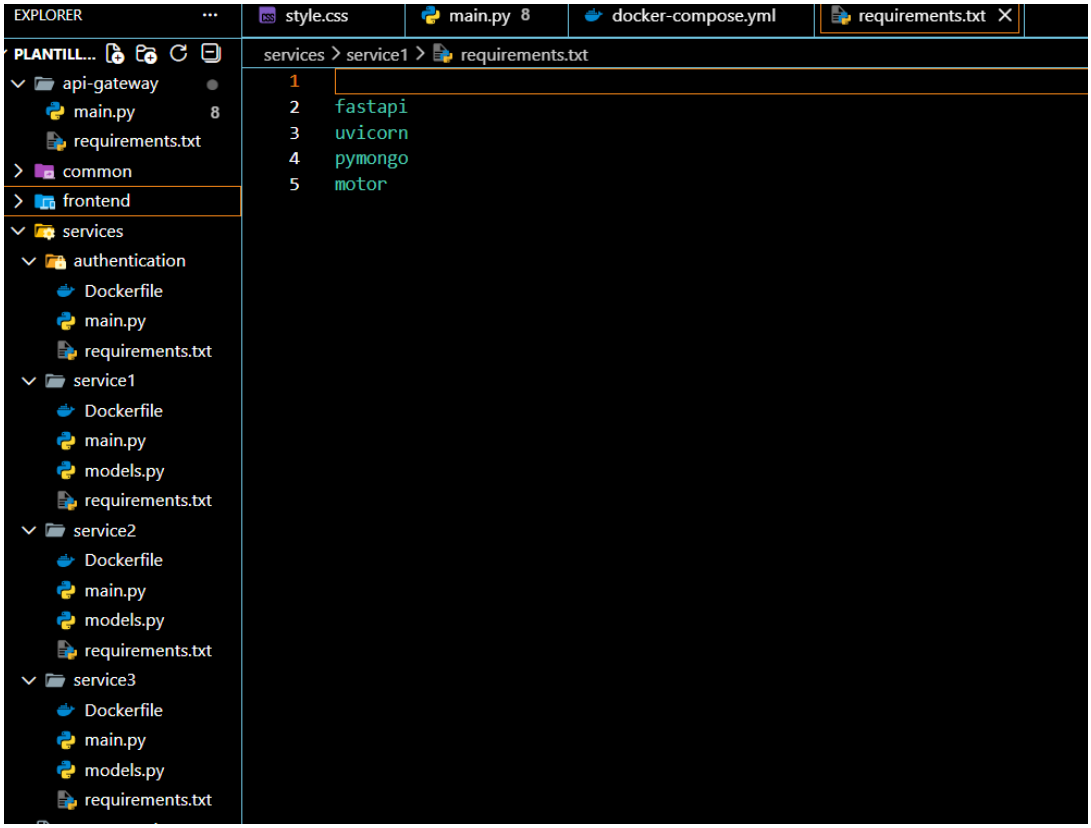

Figura 5. Archivo requirements

Este archivo se encuentra en el frontend y en cada uno de los servicios y es el que tiene la lista de las dependencias necesarias para ejecutar cada componente del sistema, en este caso se muestra el del microservicio 1 (Experiencias) que necesita FastAPI para la API REST, Uvicorn para ejecutarla, PyMongo y Motor para interactuar con MongoDB según sea necesario.

Antes de mostrar el flujo del sistema es necesario poner en funcionamiento, para esto es necesario usar el comando docker-compose up –build, el cual se encarga de primeramente construir todos contenedores según la configuración del archivo Dockerfile de cada microservicio por ejemplo en el servicio 1 tenemos el siguiente Dockerfile:

```
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port",
"8002"]
```

Aquí primero definimos Python 3.12 como entorno base, luego define /app como carpeta de trabajo donde estará todo el código, después de esto copia solo el archivo de dependencias (requirements.txt) e instala las librerías necesarias para el funcionamiento para después copiar todo el código del microservicio al contendor y poder levantar el servicio de la aplicación FastAPI en el puerto 8002.

Ya sabiendo lo que contiene el Docker file se levantan todos los contendores que previamente definidos en el Docker-compose de forma orquestada, en la siguiente iamegn podemos ver el resultado de ejecutar el comando (docker-compose up –build):

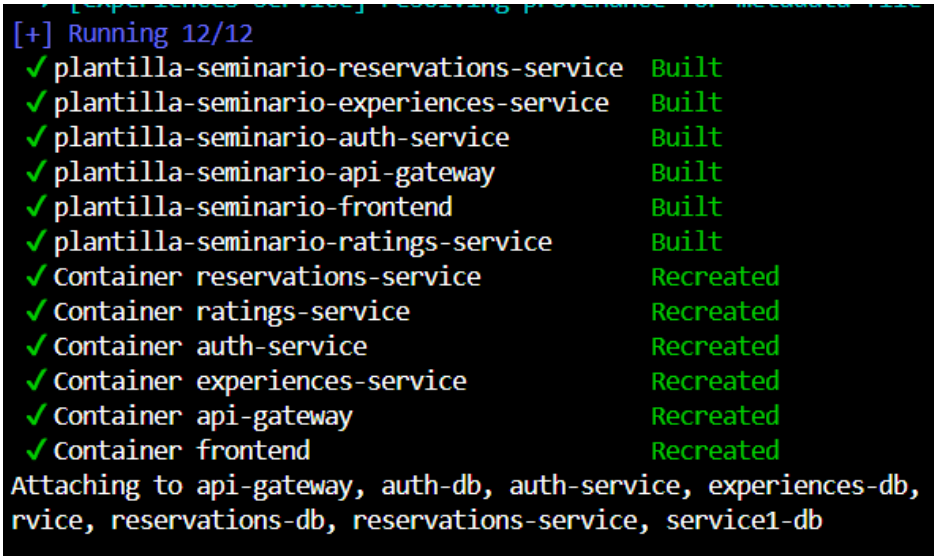

Figura 6. *Resultado comando*

Como podemos ver en la anterior imagen se puede encontrar que se están construyendo las imágenes Docker de los microservicios con lo demás que definimos, y en lo ultimo podemos observar queel sistema está conectando los contenedores a la red interna de Docker, donde se obtiene la comunicación entre el API Gateway, las bases de datos, los microservicios y el frontend permitiendo que toda esta funcionando y operando correctamente. Para comprobación del funcionamiento podemos ver el estado de los contendores y su funcionamiento en http://localhost:5000/ como podemos ver en las figuras 7 y 8)

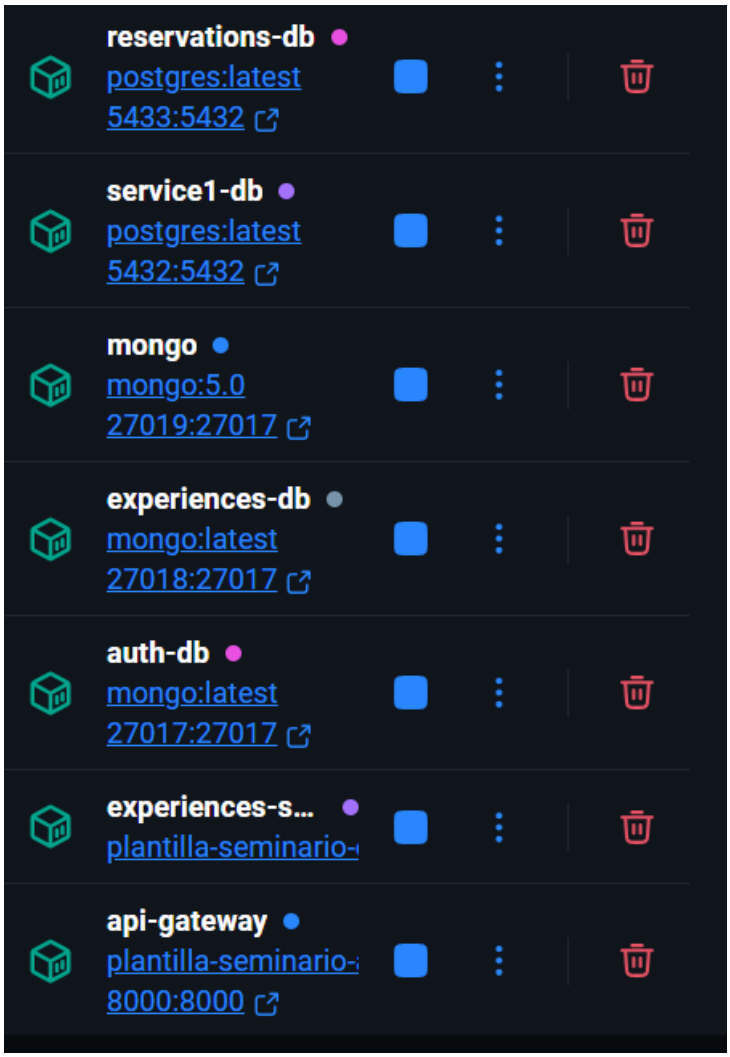

Figura 7. *Contendores corriendo*

Flujo del sistema

El sistema se compone del siguiente flujo:

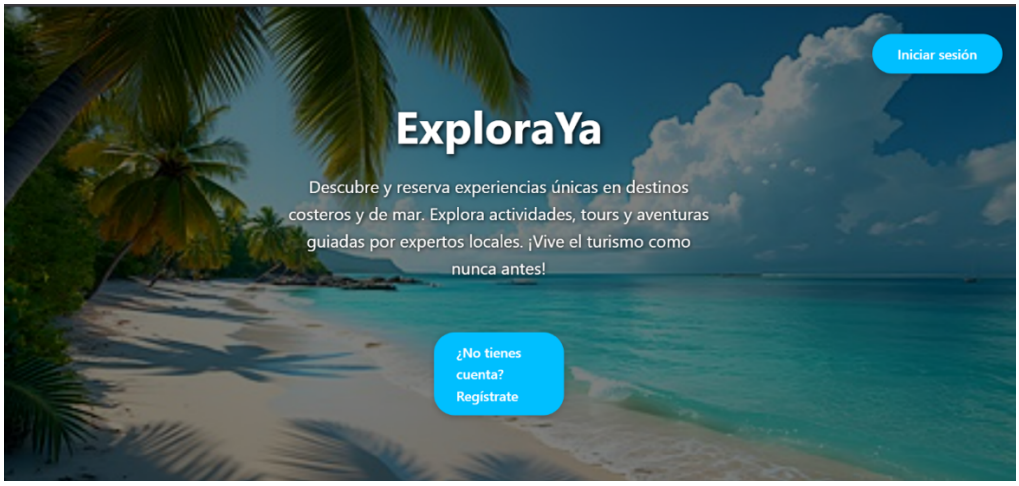

Figura 8. *Pantalla principal*

Aquí ya estamos en http://localhost:5000/ el cual nos muestra nuestro sistema como tal y nos permite iniciar sesión o crear una cuenta.

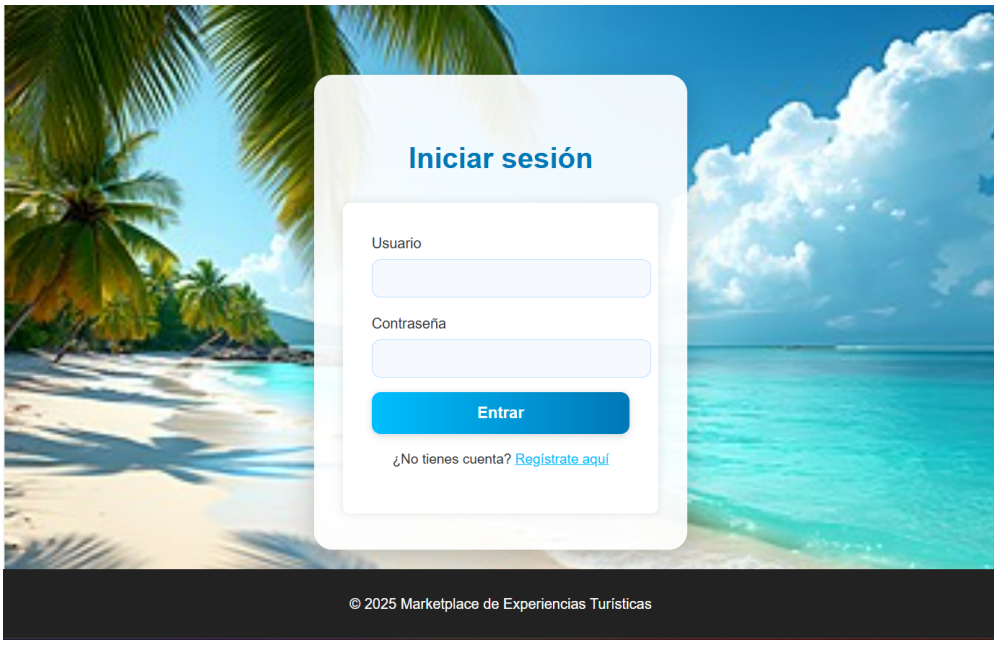

Figura 9. *Inicio de sesión*

En la figura 9 podemos ver la interfaz de inicios de sesión y si no tenemos cuenta, está el apartado para crearnos una cuenta.

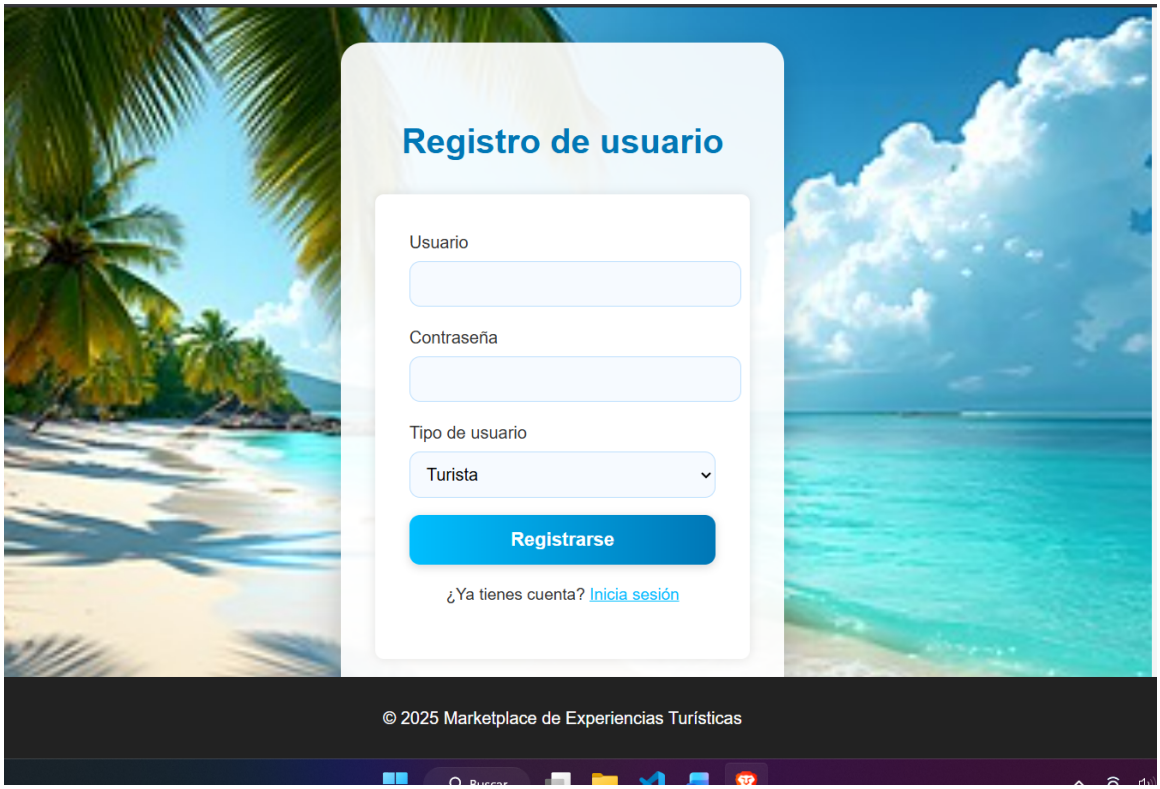

Figura 10. *Registrar usuario*

Si no tenemos una cuenta, en este apartado nos permite crear un nombre, contraseña y tipo de usuario para nuestra cuenta y permite crear el usuario para ingresar posteriormente, aquí podemos ver una parte importante del proyecto ya que a la hora de enviar el formulario se agarra todos los datos y se los envía a su microservicio que sería el de autenticación mediante una petición POST, gracias al api getway que es el que conoce a donde deben ir los datos y los enruta.

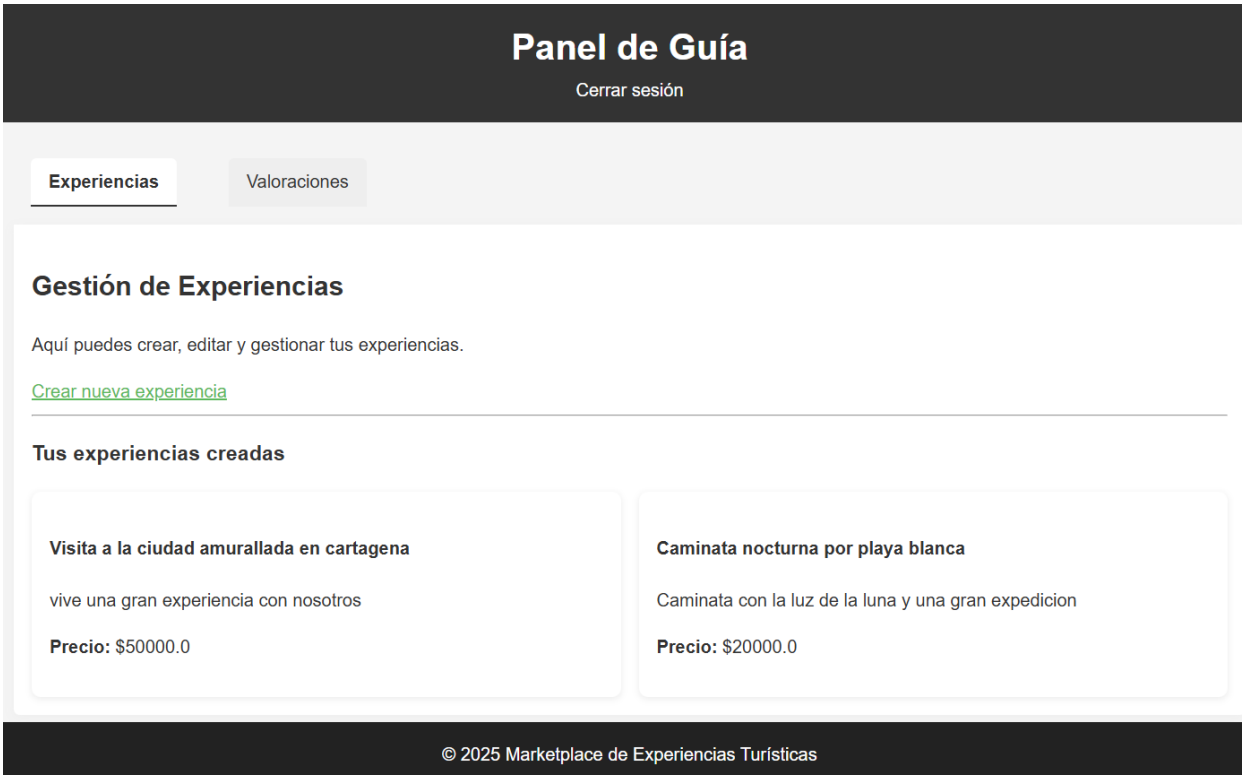

Figura 11. *Panel de guía*

En este apartado ingresamos con una cuenta de guía, aquí como primera opción podemos ver las experiencias que hemos creado, y nos permite crear nuevas, y también tenemos el apartado de las valoraciones, otro punto esencial ya que el usuario guía interactúa con el backend permitiendo ver, editar, eliminar y crear experiencias por el puente que sería el frontend.

Figura 12. *Crear nueva experiencia*

En la figura 12 podemos ver que nos permite crear una nueva experiencia tan solo con llenar el formulario y presionar el botón, mediante el cual se hace el proceso de peticiones para guardar estos datos.

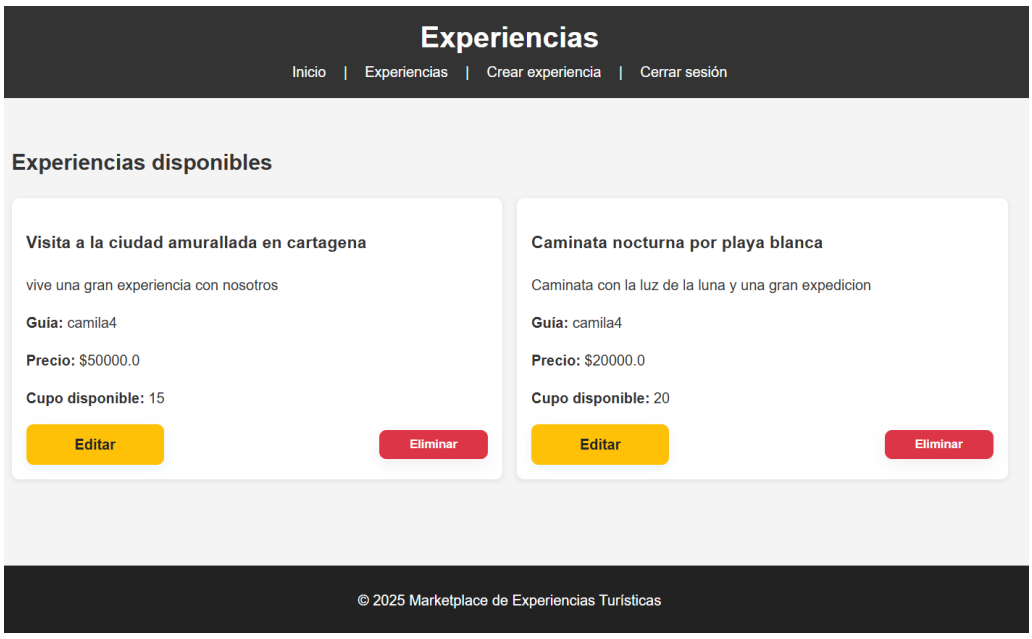

Figura 13. *Panel de experiencias disponibles*

Aquí podemos gestionar nuestras experiencias disponibles, si queremos editar alguna variable en especifico o ya eliminar la experiencia por completo.

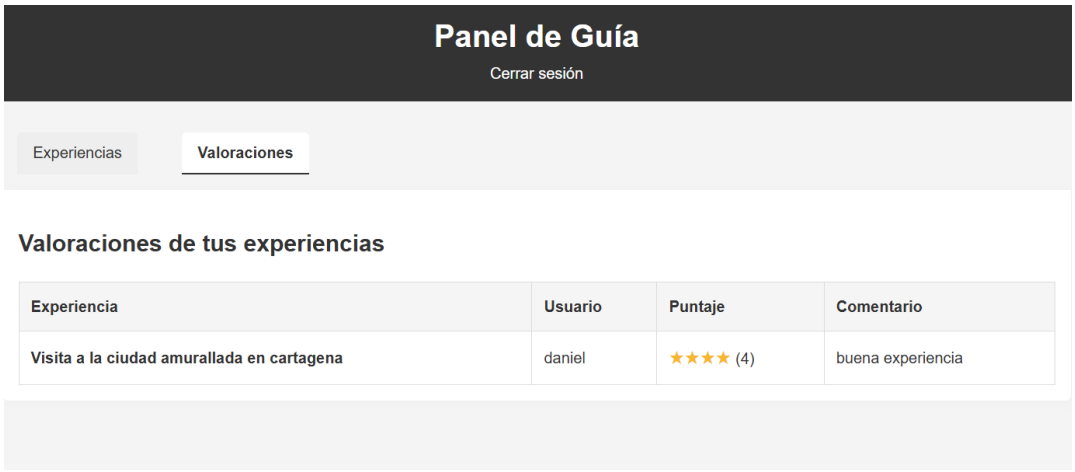

Figura 14. *Panel de valoraciones*

En este panel se pueden visualizar todas las valoraciones de los turistas junto con un comentario, se puede ver quien hizo la valoración, que puntaje le día a la experiencia y que comentario dejó.

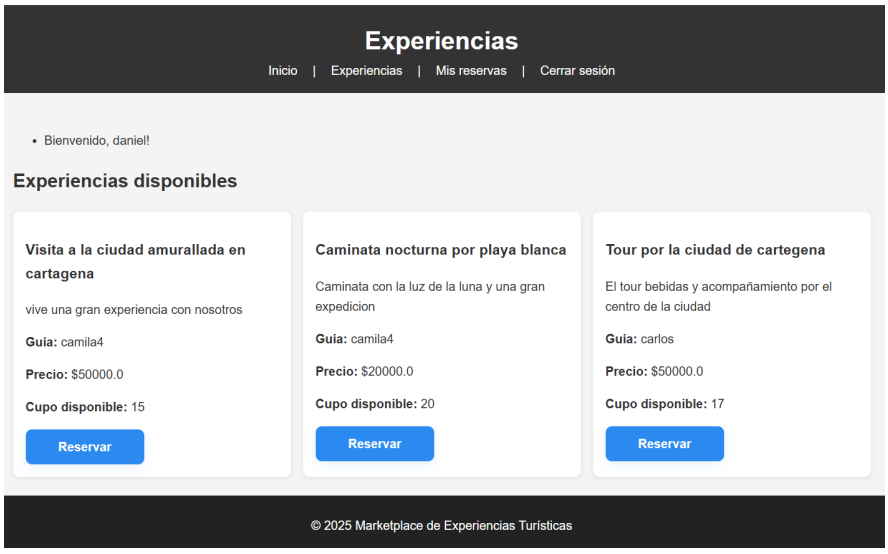

Figura 15. *Panel para turista*

En la figura 15 podemos ver que ingresamos como usuario turista con nuestra cuenta, y aquí podemos visualizar todas las experiencias disponibles por todos los guías, junto con su descripción, el precio, el cupo disponible y el nombre del guía, aquí se puede reservar una experiencia al presionar el botón. En este apartado también tenemos la opción de ver mis reservas donde se mostrará cuantas reservas hemos hecho.

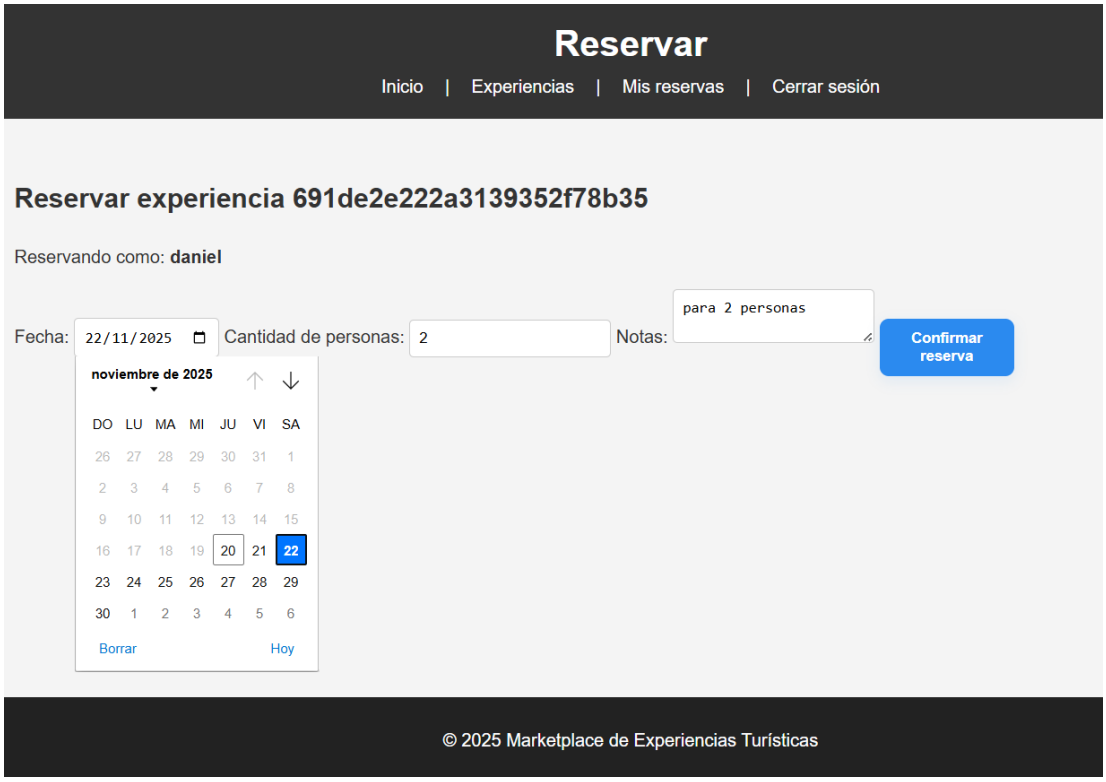

Figura 16. *Panel para reservar*

Al presionar el botón de reservar una experiencia tenemos la opción de seleccionar la fecha, la cantidad de personas para que valide si hay disponibilidad y una nota por si queremos decir algo, y estos se envían al microservicio de reservas especializado para generar las reservas dependiendo si hay disponibilidad.

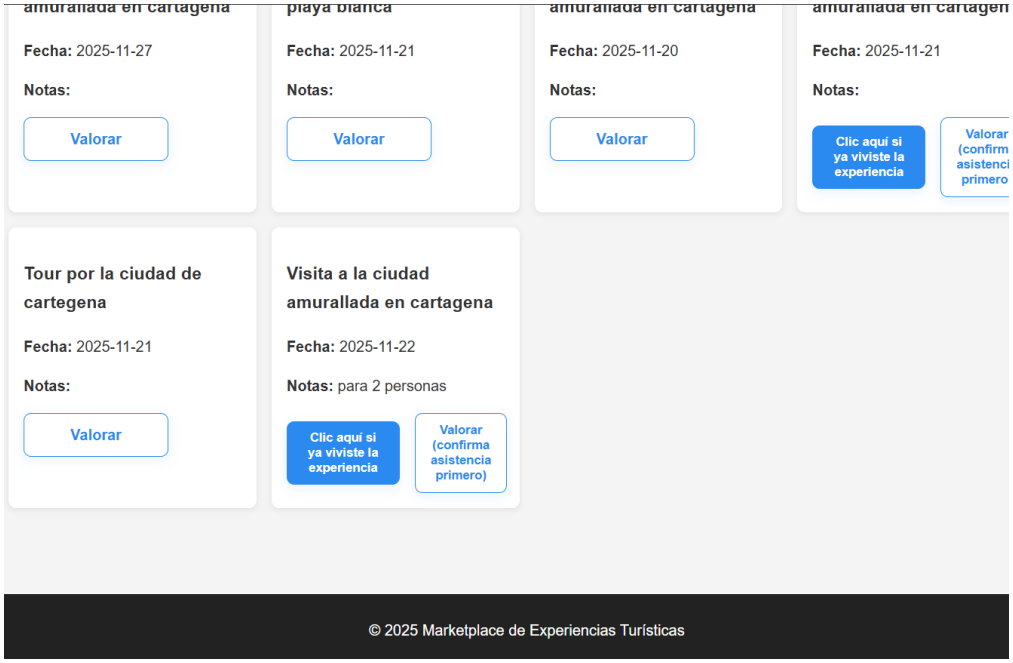

Figura 17. *Panel de mis reservas*

Después de haber echo la reserva podemos revisarla en nuestro panel de reservas, donde se los mostrar todas las reservas que hemos reservado, y si ya vivimos la experiencia podemos marcarla para poder hacer una valoración.

Figura 18. *Panel valorar experiencia*

Den la figura 18 ya podemos ver que nos deja hacer la valoración del servicio, y podemos rellenar los datos y enviar par que le llegue al guía y el puede aplicar mejoras o retroalimentarse.

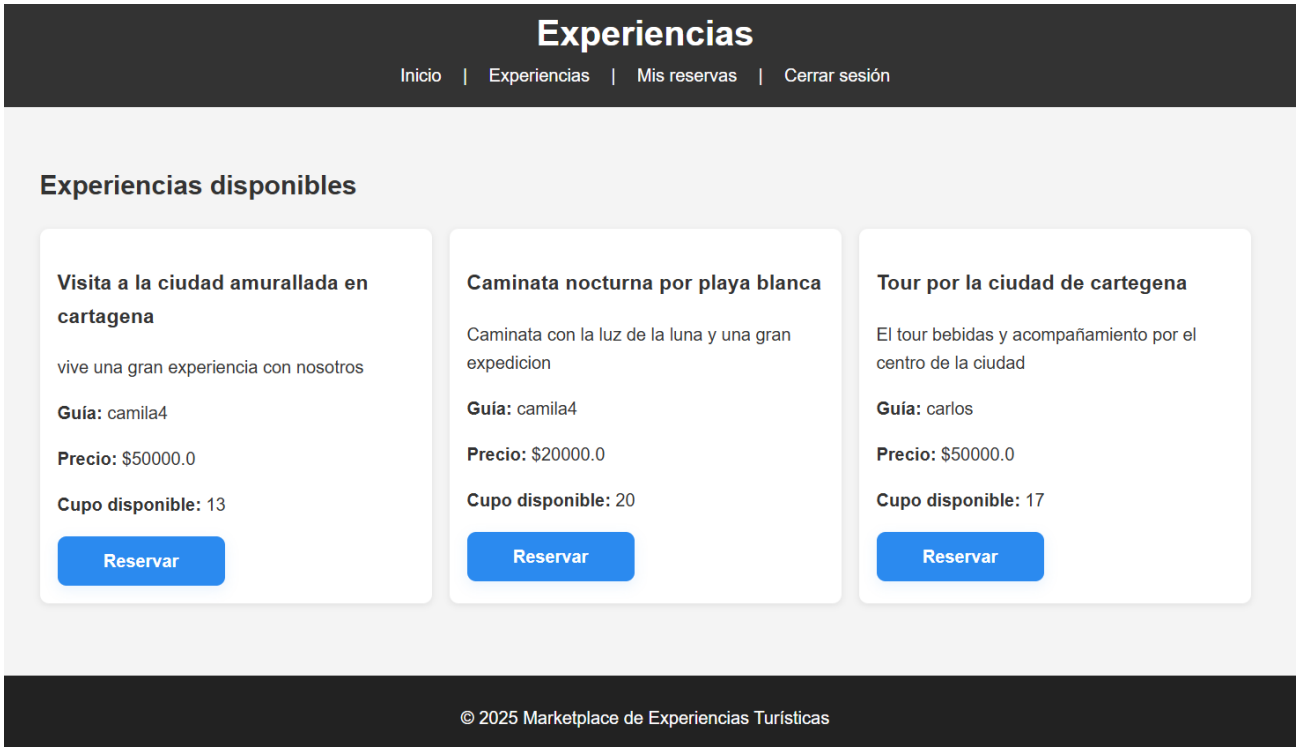

Figura 19. *Prueba de cupos*

En la figura 19 podemos ver que se descontó el cupo de 15 a 13 porque la reserva fue para 2 personas, esto gracias a que si se hizo la petición y salió correcta y por eso se descontó la cantidad de la base de datos.

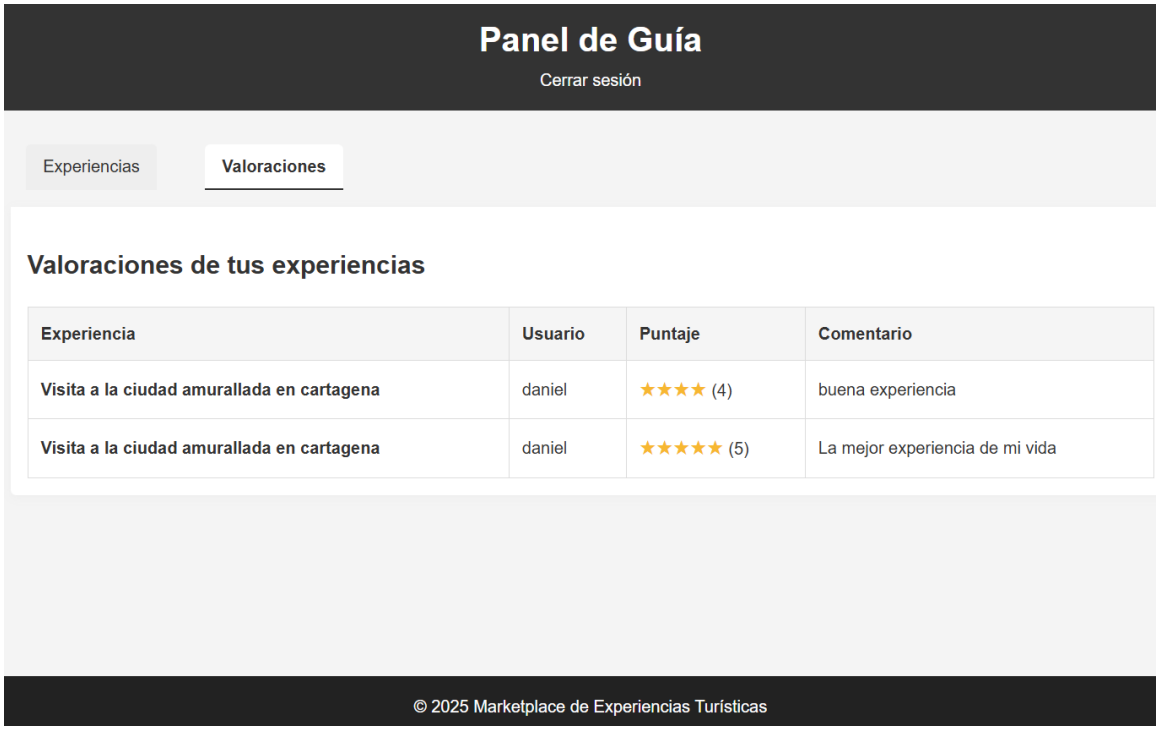

Figura 20. *Prueba de valoración*

Aquí se puede ver reflejada la valoración que hizo el turista en la experiencia que tuvo y que fue de 5 estrellas, este punto nos muestra que el sistema esta funcionando por completo, de forma organizado y precisa, ejecutando cada microservicio y asignando las peticiones a su respectiva ruta y almacenamiento en la base de datos.

Conclusiones.

La implementación del sistema Marketplace de para experiencias turísticas locales con arquitectura de microservicios se logro realizar gracias a la búsqueda de información por diferentes medios, que permitió contextualizarse en el campo de turismo, y entender el funcionamiento de los microservicios y Docker, a su vez que gracias a la implementación de estos se logró un sistema muy escalable a futuro, con una interfaz sencilla y practica hacia los usuarios finales y por último y más importante, funcional.

Ya que permite la conexión entre turistas y guías para ofrecer y poder reservas experiencias según disponibilidad, con la opción de valorar el servicio, dando así solución y respuesta a la pregunta orientadora.

Referencias

Martinez, D. R., Valderas, P. J., & Bosh, V. T. (2018). *Microservicios Un enfoque integrado.* RA-MA S.A. Editorial y Publicaciones.

Aggarwal, S. (2023). *Flask Framework Cookbook.* Packt Publishing.

de Duque, R. R., Plazas, R. R., & Velez Rivas, M. L. (2011). *La cadena de valor en turismo.* Universidad Externado de Colombia.

Gouigoux, J. P. (2018). *Docker primeros pasos y puesta en práctica de una arquitectura basada en micro-servicios.* Ediciones ENI.

Janev, M. (2025). *Desarrollo de aplicaciones de inteligencia artificial con las API de OpenAI* (2nd ed.). Marcombo.

Kroger, J., & Marx, S. (2025). *Strategic Agility in Marketing.* Springer Fachmedien Wiesbaden.

Marquez, J. V. (2019). *Confianza e información digital: bibliotecas, archivos y web.* UNAM, Instituto de Investigaciones Bibliotecológicas y de la Información.

Martin, P. P. (2017). *Tecnología de Contenedores Docker.* Universitat Politècnica de Catalunya. Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona.

Maul, H. (s.f.). *Economía informal superando las barreras de un estado excluyente* (Vol. 1). 2006: Universidad de Texas.

Molina, P. G. (2025). *Productos, Servicios Y Destinos Turísticos. Uf0073.* Editorial Tutor Formación.

Ortega, J. M. (2023). *Desarrollo de microservicios con Python.* Marcombo.

Palomeque, F. L., Delgado, A. T., Vera, J. F., & Baidal, J. A. (2022). *El turismo ¿fin de una epoca? Desafíos de España como destino turístico en un nuevo escenario* (Vols. 270-290). Universidad de València Servicio de Publicaciones.

Surianarayanan, C., Ganapathy, G., & Pethuru, R. (2019). *Essentials of Microservices Architecture.* CRC Press.

Turley, F., & Rad, N. K. (2019). *Los Fundamentos de Agile Scrum* . Van Haren Publishing.

Yoris, A. C. (2018). *Primeros pasos con FastApi.* Andres Cruz.