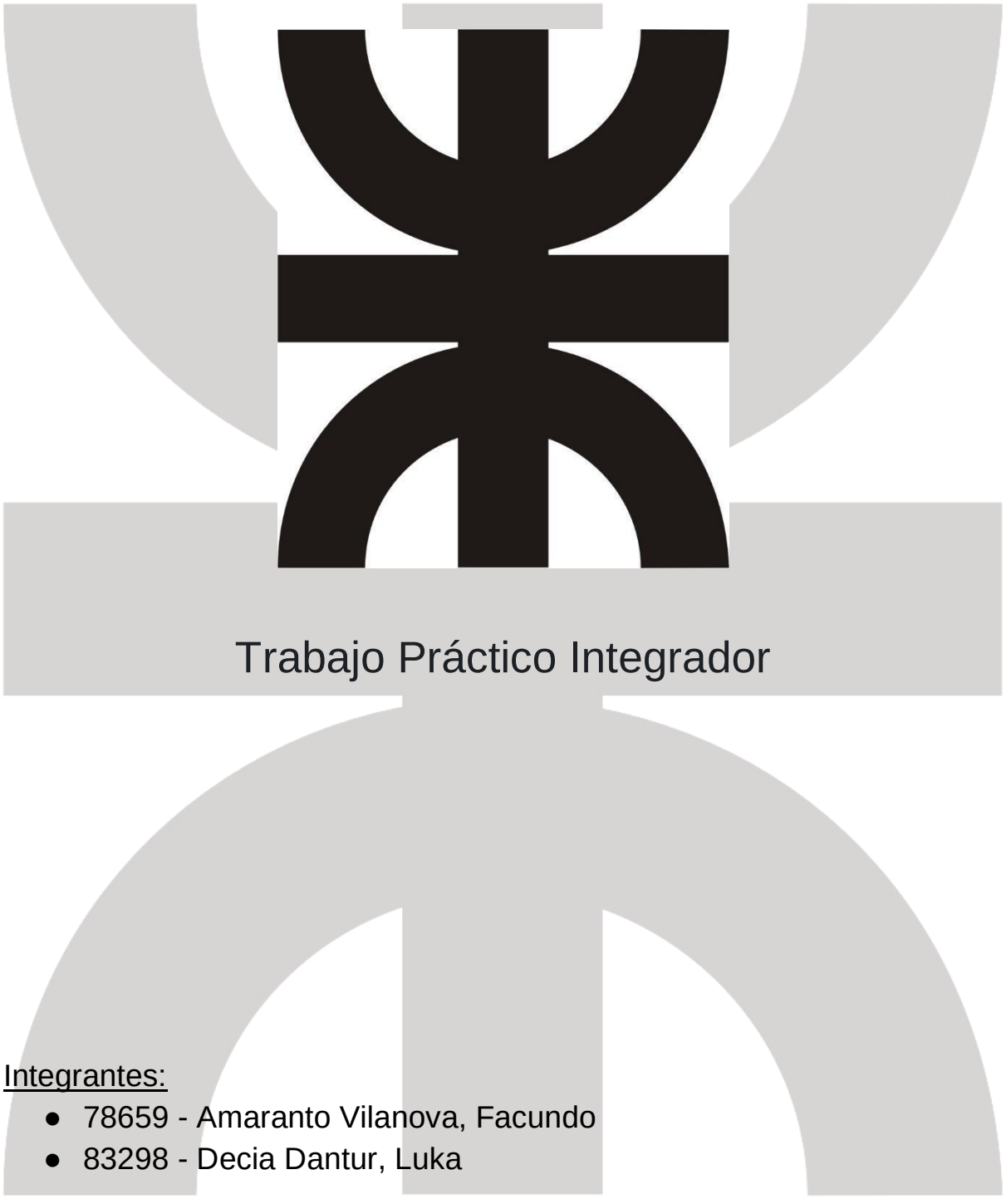
Integración de Aplicaciones en Entorno Web



Trabajo Práctico Integrador

Integrantes:

- 78659 - Amaranto Vilanova, Facundo
- 83298 - Decia Dantur, Luka

Curso: 5k4

Documentación Técnica y de Diseño para Microservicio de Gestión de Experiencias Turísticas

1. Introducción

El Microservicio de Gestión de Experiencias Turísticas es una API REST diseñada para Administrar información sobre tours, actividades, y experiencias turísticas, así como reservas para estas experiencias.

Este documento proporciona información técnica y de diseño sobre la implementación de este servicio.

2. Tecnologías Utilizadas

2.1 Base de Datos: MongoDB Atlas

El Microservicio de Gestión de Experiencias Turísticas utiliza MongoDB Atlas como base de datos principal debido a sus características clave que benefician directamente al proyecto.

2.1.1 Razones de la Elección

1. **Escalabilidad Horizontal:** MongoDB Atlas facilita la expansión horizontal para manejar crecimientos de datos y tráfico sin perder rendimiento.
2. **Disponibilidad y Tolerancia a Fallos:** La arquitectura de Atlas garantiza alta disponibilidad y tolerancia a fallos mediante la replicación automática de datos en diferentes ubicaciones geográficas.
3. **Gestión Simplificada:** Como plataforma en la nube completamente administrada, MongoDB Atlas alivia a los desarrolladores de tareas operativas, permitiéndoles centrarse en el desarrollo de la aplicación.
4. **Seguridad Integral:** Atlas proporciona funciones avanzadas de seguridad, como encriptación de datos, autenticación basada en roles y auditoría de eventos, garantizando la protección integral de los datos.
5. **Ecosistema Flexible:** La flexibilidad del modelo de datos basado en documentos JSON BSON facilita la adaptación a cambios en los requisitos de la aplicación.

2.1.2 Ventajas de Utilizar MongoDB Atlas

1. **Sin Preocupaciones de Infraestructura:** La implementación en la nube elimina la gestión de hardware y configuraciones, permitiendo un enfoque sin preocupaciones en el desarrollo.

2. **Eficiencia en Desarrollo:** La interfaz amigable y las herramientas de desarrollo de Atlas agilizan la creación, prueba y despliegue de aplicaciones.
3. **Escalabilidad Bajo Demanda:** La capacidad de escalar recursos automáticamente asegura que el microservicio pueda adaptarse a diversas cargas de trabajo.
4. **Monitorización y Optimización Continua:** Atlas proporciona herramientas integradas para monitoreo y optimización continua del rendimiento de la base de datos.

La elección de MongoDB Atlas se fundamenta en su capacidad para proporcionar escalabilidad, disponibilidad, seguridad y una gestión simplificada, elementos críticos para el éxito y la longevidad del sistema en un entorno dinámico y exigente.

2.2 Conexión con la Base de Datos

La conexión entre el backend de la API REST, desarrollada en C#, y la base de datos MongoDB Atlas se realiza de manera eficiente a través de las siguientes consideraciones y pasos:

2.2.1 Driver de MongoDB para C#

Para establecer la conexión con MongoDB Atlas desde C#, se utiliza un controlador (driver) de MongoDB específico para este lenguaje. MongoDB proporciona un controlador oficial llamado "MongoDB C# Driver". Este controlador facilita la interacción con la base de datos MongoDB Atlas directamente desde la aplicación escrita en C#.

2.2.2 Configuración de la Conexión

La configuración de la conexión se realiza mediante la especificación de la cadena de conexión de MongoDB Atlas en el código C#. La cadena de conexión incluye información como la dirección del servidor, el nombre de la base de datos y las credenciales de acceso.

Se debe reemplazar **<usuario>**, **<contraseña>**, y **<basededatos>** con las credenciales y el nombre de la base de datos específicos de tu entorno.

2.2.3 Uso del Driver en el Código C#

Una vez configurada la cadena de conexión, se utiliza el driver de MongoDB C# para establecer la conexión y realizar operaciones en la base de datos.

2.3 Backend

El código backend del microservicio está desarrollado en C#, aprovechando las capacidades de este lenguaje para construir una aplicación robusta y eficiente.

2.4 Endpoints

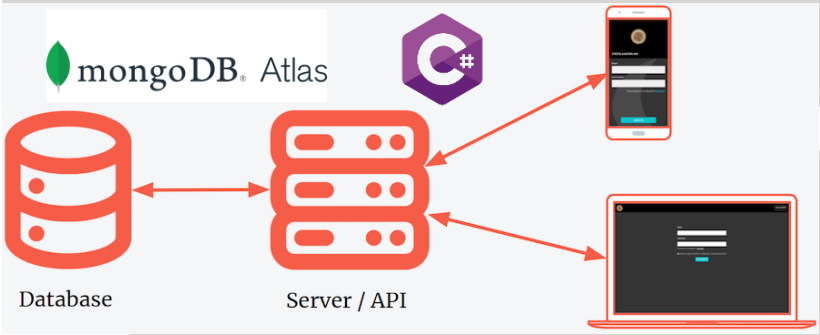
Se han implementado varios endpoints para la interacción con el microservicio:

- **GET /experiencias**: Obtiene la lista de todas las experiencias disponibles.
- **POST /experiencias**: Crea una nueva experiencia.
- **PUT /experiencias/{id}**: Actualiza la información de una experiencia específica.
- **DELETE /experiencias/{id}**: Elimina una experiencia.
- **GET /reservas-experiencias**: Obtiene la lista de todas las reservas de experiencias.
- **POST /reservas-experiencias**: Realiza una nueva reserva de experiencia.
- **GET /clientes/{id_cliente}/reservas-experiencias**: Obtiene las reservas de experiencias asociadas a un cliente específico.

2.5 Documentación de la API

La API se documenta utilizando Swagger, una herramienta que facilita la creación, visualización y consumo de servicios web RESTful. La documentación Swagger describe de manera detallada cada endpoint, los parámetros necesarios, y las respuestas esperadas.

3. Diseño de la Solución

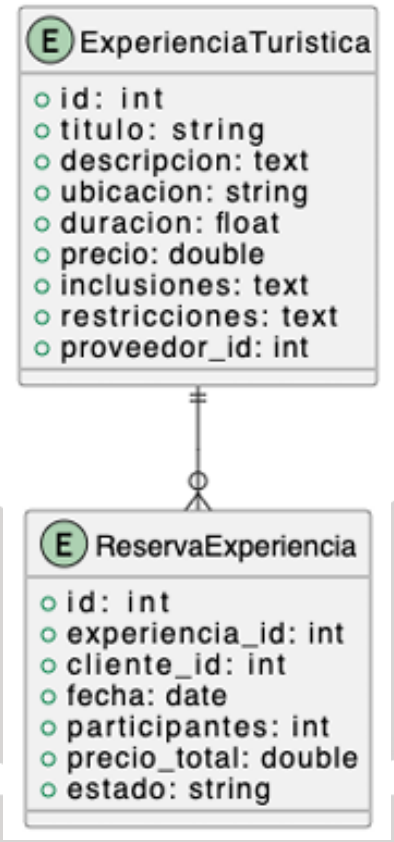


3.1 Estructura del Proyecto

El código fuente está organizado en un formato modular, siguiendo las mejores prácticas de desarrollo. Se han definido módulos para la gestión de experiencias, reservas y clientes, lo que facilita la mantenibilidad y escalabilidad del sistema.

3.2 Modelo de Datos

El modelo de datos utilizado sigue las mejores prácticas de diseño de bases de datos NoSQL. Se han definido colecciones para Experiencias Turisticas y Reserva Experiencia, estableciendo relaciones eficientes para garantizar la integridad de los datos.



3.3 Seguridad

El Microservicio de Gestión de Experiencias Turísticas ha implementado medidas de seguridad integrales para proteger los datos y garantizar el acceso autorizado. La estrategia de seguridad se basa en la implementación de OAuth 2.0 como protocolo de autorización y Auth0 como servidor de autorización.

3.3.1 OAuth 2.0

OAuth 2.0 se ha integrado como el protocolo de autorización principal, proporcionando un marco robusto para la autenticación y autorización. Este protocolo permite un control granular sobre el acceso a los recursos del microservicio, asegurando que solo usuarios autenticados y autorizados puedan realizar operaciones específicas.

3.3.2 Auth0

Auth0 se utiliza como el servidor de autorización central para gestionar la identidad y la seguridad. Esta plataforma ofrece servicios avanzados de gestión de identidad, garantizando un flujo seguro de información de autenticación y autorización. La elección de Auth0 contribuye a una gestión centralizada de identidades, simplificando la administración de usuarios y mejorando la seguridad en el acceso a recursos críticos.

3.3.4 Beneficios de la Estrategia de Seguridad

- **Control de Acceso Granular:** La implementación de OAuth 2.0 y Auth0 permite un control de acceso granular, asegurando que solo los usuarios autorizados puedan acceder a recursos específicos.
- **Seguridad en la Comunicación:** La seguridad en la transmisión de datos se mejora mediante la autenticación y autorización, garantizando la confidencialidad e integridad de la información intercambiada.
- **Auditoría y Rastreo:** Auth0 proporciona capacidades de auditoría y rastreo, permitiendo la monitorización de actividades y la identificación proactiva de posibles amenazas de seguridad.

3.4 Escalabilidad

La arquitectura del microservicio está diseñada para ser escalable horizontalmente, permitiendo la adición de instancias para manejar un mayor volumen de solicitudes y datos.

4. Conclusiones

En conclusión, el Microservicio de Gestión de Experiencias Turísticas se erige como una solución robusta y eficiente para la administración de tours, actividades y reservas. La elección de tecnologías clave, como MongoDB Atlas para la base de datos y C# para el desarrollo del backend, demuestra un enfoque estratégico para garantizar escalabilidad, disponibilidad y seguridad.

La implementación de endpoints bien definidos facilita la interacción con el microservicio, permitiendo operaciones como la obtención, creación, actualización y eliminación de experiencias y reservas. La documentación detallada mediante Swagger añade un valor significativo al proporcionar una guía clara para consumir la API.

El diseño modular y organizado del proyecto, junto con un modelo de datos eficiente, contribuye a la mantenibilidad y escalabilidad a medida que el sistema evoluciona. Además, la estrategia de seguridad basada en OAuth 2.0 y Auth0 garantiza un acceso controlado y seguro a los recursos, con beneficios notables como el control de acceso granular y la mejora en la seguridad de la comunicación.

En términos de escalabilidad, la arquitectura del microservicio está preparada para crecer horizontalmente, permitiendo adaptarse sin problemas a mayores volúmenes de solicitudes y datos. En conjunto, estas decisiones de diseño y tecnológicas proporcionan una base sólida para la implementación y evolución exitosa del Microservicio de Gestión de Experiencias Turísticas en un entorno dinámico y exigente.