Documento de Arquitectura DAS

***Muta APP***

**Historial de Versiones**

| **Fecha** | **Versión** | **Autor** | **Organización** | **Descripción** |
| --- | --- | --- | --- | --- |
| 21/10/2024 | 1.1 | Carlos Troncoso |  | Se integra versión 1 de la arquitectura DAS |

**Información del proyecto**

| Empresa / Organización | Muta App |
| --- | --- |
| Proyecto | Aplicación Mobile Muta App |
| Fecha de preparación | 01/2024 |
| Cliente | Gimnasio Muta |
| Patrocinador principal | Duoc UC |
| Gerente de proyecto | Maximiliano Sánchez |

**1. Introducción**

Este documento describe la arquitectura de la aplicación de fitness impulsada por inteligencia artificial utilizando un enfoque de servicios distribuidos (DAS). El objetivo es proporcionar una visión clara de los componentes del sistema, sus interacciones y las tecnologías utilizadas para implementar la aplicación.

**2. Descripción General de la Arquitectura**

La arquitectura de la aplicación se basa en un modelo de microservicios que permite una escalabilidad eficiente, mantenibilidad y flexibilidad. La aplicación está compuesta por varios servicios independientes que se comunican entre sí a través de APIs RESTful.

**3. Componentes de la Arquitectura**

**3.1. Frontend (Interfaz de Usuario)**

* **Descripción:** La interfaz de usuario es la parte de la aplicación con la que interactúan los usuarios finales.
* **Tecnologías Utilizadas:**
  + **Framework:** React Native (para aplicaciones móviles).
  + **Lenguaje de Programación:** TypeScript.
  + **Gestión de Estado:** Redux o Context API.
* **Funciones Principales:**
  + Registro e inicio de sesión de usuarios.
  + Ingreso de datos personales.
  + Visualización de rutinas y planes de alimentación.

**3.2. Microservicio Principal**

* **Descripción:** Servicio principal desarrollado en .NET 8, encargado de gestionar las funcionalidades clave de la aplicación.
* **Tecnologías Utilizadas:**
  + **Framework:** .NET 8.
  + **Despliegue:** Containerizado en AWS utilizando ECR y ECS.
* **Funciones Principales:**
  + Lógica de negocio central de la aplicación.

**3.3. Servicios de Integración**

* **Descripción:** Este módulo es un servicio de integraciones que se conecta de forma asincrónica con productos externos, utilizando AWS SQS como cola de mensajes. También permite conexiones síncronas, como con OpenAPI.
* **Tecnologías Utilizadas:**
  + **Framework:** Spring Boot con compilación nativa mediante GraalVM.
  + **Cola de Mensajes:** AWS SQS.
  + **Base de Datos:** MySQL 8.0 LTS en AWS RDS, sobre procesadores Graviton.
* **Funciones Principales:**
  + Conexiones asincrónicas y síncronas con servicios externos como OpenAPI.

**3.4. Base de Datos**

* **Descripción:** Almacenamiento de datos estructurados de los usuarios y sus interacciones con la aplicación.
* **Tecnologías Utilizadas:**
  + **Tipo:** MySQL 8.0 LTS.
* **Estructura de Datos:**
  + **Colección de Usuarios:** Información personal, preferencias, historial de escaneos.
  + **Colección de Rutinas:** Rutinas generadas, progreso del usuario.

**4. Diagrama de Arquitectura**

[Incluir un diagrama de arquitectura que represente los componentes del sistema y sus interacciones. Este diagrama puede ser realizado utilizando herramientas como Lucidchart, Draw.io o Microsoft Visio. Asegúrate de mostrar cómo se comunican los diferentes servicios y componentes.]

**5. Comunicación entre Componentes**

**5.1. APIs RESTful**

* **Descripción:** Los componentes del sistema se comunican entre sí utilizando APIs RESTful.
* **Protocolos:** HTTP.
* **Formatos de Datos:** JSON.

**5.2. Mensajería Asincrónica**

* **Descripción:** Para procesos que requieren tiempo de procesamiento, se utiliza mensajería asincrónica (por ejemplo, a través de AWS SQS) para mejorar la escalabilidad y la eficiencia.

**6. Seguridad y Autenticación**

**6.1. Autenticación de Usuarios**

* **Método:** OAuth 2.0 o JWT (JSON Web Tokens) para gestionar la autenticación y autorización de usuarios.
* **Cifrado:** Utilización de HTTPS para todas las comunicaciones y cifrado de contraseñas en la base de datos.

**6.2. Protección de Datos Personales**

* **Cumplimiento:** La arquitectura está diseñada para cumplir con la Ley N° 19.628 sobre Protección de la Vida Privada de Chile y el RGPD.
* **Acceso a Datos:** Implementación de políticas de acceso para limitar el acceso a datos sensibles.

**7. Escalabilidad y Mantenibilidad**

**7.1. Escalabilidad**

* **Horizontal:** Los microservicios permiten la escalabilidad horizontal utilizando AWS ECS y Fargate.

**7.2. Mantenibilidad**

* **containerización:** Utilización de Docker para contenerizar los servicios, lo que facilita el despliegue y la gestión de versiones.
* **Documentación:** Documentación clara de las APIs y la arquitectura para facilitar el mantenimiento y la incorporación de nuevos desarrolladores.

**8. Conclusiones**

Este documento proporciona una descripción detallada de la arquitectura de la aplicación de fitness impulsada por inteligencia artificial. El enfoque en microservicios, seguridad y escalabilidad garantiza que la aplicación sea eficiente, segura y fácil de mantener a medida que crece y se adapta a las necesidades de los usuarios.