Documentación Técnica : *[Quantum]Microservice* - Versión 1.0

Barcelona, 12 de Marzo de 2025.

Autor: *Héctor Ayuso Martín.*

Introducción:

Este documento describe la arquitectura, funcionalidades y detalles técnicos del microservicio desarrollado con **FastAPI, MongoDB Atlas y Qiskit**. El objetivo principal de este microservicio es proporcionar autenticación segura mediante **JWT**, gestión de usuarios y generación de números aleatorios basados en mecánica cuántica.

El diseño modular y escalable permite su integración con otros servicios, asegurando seguridad y eficiencia en el manejo de datos sensibles.

Arquitectura del Proyecto:

El microservicio está diseñado bajo una arquitectura **modular** que facilita la **escalabilidad y mantenimiento**. Se compone de diversos módulos responsables de la gestión de usuarios, autenticación, almacenamiento de datos y generación de números aleatorios cuánticos.

El esquema general es el siguiente:

- *FastAPI*: Framework principal para el desarrollo de la **API REST.**
- *MongoDB Atlas*: **Base de datos NoSQL** utilizada para almacenar información de los usuarios.
- *Motor*: Cliente **asíncrono** para la interacción con MongoDB.
- *Bcrypt*: Biblioteca para la gestión de contraseñas seguras mediante **hashing.**
- *PyJWT*: Gestor de tokens JWT para **autenticación segura**.
- *Qiskit*: Framework para simulación y uso de mecánica cuántica de IBM.

Gestión de Base de Datos:

El microservicio utiliza **MongoDB Atlas** como base de datos principal, permitiendo almacenamiento flexible y eficiente. Las operaciones **CRUD** (Crear, Leer, Actualizar, Eliminar) son implementadas en core/db.py utilizando motor. Los datos almacenados incluyen:

- *Usuarios*: Con información de autenticación segura (hash de contraseñas) y roles de acceso.
- *Tokens de autenticación*: Para validar el acceso seguro a los endpoints.

Seguridad y Autenticación:

La seguridad del sistema se basa en la implementación de autenticación con **tokens JWT**. Los principales aspectos incluyen:

- *Hashing de contraseñas*: Se utiliza **bcrypt** para almacenar contraseñas de manera segura.
- *Generación y validación de tokens JWT*: Implementado con **PyJWT** para garantizar accesos autenticados.
- *Autorización basada en roles*: **Control de acceso** en los endpoints críticos.

El archivo core/security.py gestiona estas funcionalidades, asegurando que solo usuarios autenticados puedan acceder a las rutas protegidas.

Módulos Principales:

El microservicio se organiza en **módulos** especializados:

1. Autenticación (routers/auth.py):
    - Endpoint para inicio de sesión con validación de credenciales.
    - Generación y entrega de tokens JWT.

2. Gestión de Usuarios (routers/users.py):
    - Permite creación, consulta, actualización y eliminación de usuarios.
    - Solo accesible para usuarios con permisos adecuados.

3. Generación de Claves Criptográficas (routers/keys.py):
    - Implementa generación de claves **AES, RSA, UUID y OTP**.
    - Utiliza números aleatorios obtenidos de la mecánica cuántica con Qiskit.

4. Generación de Números Aleatorios Cuánticos (routers/quantum_random.py):
    - Uso de Qiskit para obtener bits aleatorios a partir de estados cuánticos.
    - Implementación de funciones para generar números aleatorios reales.

Implementación de Qiskit en el Microservicio:

La generación de números aleatorios se realiza mediante **Qiskit**, utilizando qubits en superposición y mediciones en la base computacional. Los principales métodos utilizados son:

- Qubit en superposición: Se crea un estado cuántico con una distribución equitativa entre |0> y |1>.
- Medición del qubit: Al colapsar el estado, se obtiene un bit aleatorio.
- Expansión a números completos: Se generan secuencias de bits para formar números aleatorios más grandes.

La implementación **garantiza una fuente de entropía confiable para aplicaciones de seguridad criptográfica.**

Despliegue y Contenedorización:

El microservicio ha sido **dockerizado** para facilitar su despliegue en cualquier entorno compatible con contenedores. Las principales características del despliegue son:

- *Uso de Docker*: Definición en un **Dockerfile** para encapsular todas las dependencias y configuraciones.
- *Orquestación con Render*: El servicio se encuentra desplegado en Render para **disponibilidad y escalabilidad**.
- *Configuración de variables de entorno*: Manejo seguro de credenciales y secretos mediante .env.

El despliegue permite la integración con otros servicios de manera sencilla y escalable.

Conclusión:

El microservicio desarrollado ofrece una soluciónde ayuda para la autenticación segura y la generación de números aleatorios basados en mecánica cuántica. La combinación de FastAPI, MongoDB y Qiskit permite garantizar eficiencia, seguridad y escalabilidad en aplicaciones que requieran aleatoriedad y gestión de usuarios segura.

Gracias a su diseño modular y su despliegue en contenedores, el servicio puede ser fácilmente extendido y adaptado a nuevas necesidades sin comprometer su estabilidad, lo cual simepre es muy agradecido a la hora de mantener el proyecto.

Este proyecto me ha permitido consolidar mis habilidades en el desarrollo de microservicios con FastAPI, la gestión de bases de datos NoSQL con MongoDB Atlas y la implementación de seguridad mediante autenticación JWT. Además, he trabajado con Qiskit para generar números aleatorios cuánticos, lo que me ha dado mas soltura y conocimientos en computación cuántica.

Durante este desarrollo, he aplicado metodologías de contenedorización con Docker y despliegue en plataformas cloud como Render, lo que me ha permitido familiarizarme con buenas prácticas de DevOps y despliegue continuo. La modularidad y escalabilidad del proyecto reflejan mi enfoque en la creación de software mantenible y adaptable.

Este microservicio es un ejemplo de mi capacidad para diseñar, desarrollar y desplegar soluciones tecnológicas eficientes y seguras. Estoy entusiasmado por seguir explorando nuevas tecnologías y aportar mis conocimientos en entornos profesionales.

En futuras versiones, tengo la idea de perfeccionar los algoritmos cuánticos de QNRG, y mejorar asi su velocidad, ademas de implementar nuevos endpoints con diferentes circuitos cuánticos para que la API sea capaz de servir para mas funcionalidades criptográficas cuánticas.

Héctor.