Memoria Técnica

Aplicación REST con FastAPI y MongoDB Atlas

1. Objetivo del Proyecto

Desarrollar una API REST funcional, ligera y escalable en Python que permita gestionar recursos (en este caso, una lista de tareas – *to-do list*) mediante operaciones CRUD (Crear, Leer, Actualizar, Eliminar), utilizando una base de datos NoSQL en la nube: **MongoDB Atlas**.

2. Tecnologías Empleadas

Componente	**Tecnología**	**Versión Recomendada**
Lenguaje	Python	3.9+
Framework Web	FastAPI	0.100+
Servidor ASGI	Uvicorn	0.23+
Cliente de MongoDB	Motor (basado en PyMongo)	3.3+
Base de datos	MongoDB Atlas (MongoDB en la nube)	MongoDB 6.0+
Validación de datos	Pydantic	2.0+
Documentación API	OpenAPI / Swagger UI (integrado)	—

3. Arquitectura General

```
[ Cliente HTTP (Postman, navegador, etc.) ]
            ↓
[ FastAPI (API REST en Python) ]
            ↓
[ Motor (cliente asíncrono) ]
            ↓
[ MongoDB Atlas (base de datos en la nube) ]
```

- La comunicación es **asíncrona** para mejorar el rendimiento.
- La API sigue el estándar RESTful.
- La base de datos está alojada en la nube (SaaS), sin necesidad de infraestructura local.

4. Pasos para Crear y Desarrollar la Aplicación

4.1. Entorno de Desarrollo Local

1. **Instalar Python** (3.9 o superior) desde python.org.
2. **Crear un entorno virtual** (recomendado):

```
python -m venv venv
source venv/bin/activate  # Linux/macOS
# o
venv\Scripts\activate     # Windows
```

3. **Instalar dependencias**:

```
pip install fastapi uvicorn motor python-dotenv
```

    1. `python-dotenv`: para manejar variables de entorno de forma segura (opcional, pero recomendado).

4.2. Crear la Aplicación en FastAPI

Crear un archivo `main.py` con la lógica de la API (ver código completo en sección 5).

Características clave:

- Uso de modelos Pydantic para validación.
- Operaciones CRUD asíncronas con Motor.
- Manejo de errores HTTP adecuado (404, 400, etc.).
- Serialización segura de ObjectId de MongoDB.

5. Configuración y Conexión con MongoDB Atlas

5.1. Crear una Cuenta en MongoDB Atlas

1. Ir a https://www.mongodb.com/atlas/database.
2. Registrarse o iniciar sesión.
3. Hacer clic en **"Create a Free Cluster"**.

5.2. Configurar el Clúster

1. **Seleccionar proveedor y región**:
    1. Recomendado: AWS, Google Cloud o Azure (elige la más cercana a tus usuarios).
    2. Para uso gratuito: *Shared Cluster (M0)*.
2. **Nombre del clúster**: `ToDoCluster` (o el que desees).
3. **Esperar a que se aprovisione** (puede tardar 2-5 minutos).

5.3. Configurar Acceso a la Base de Datos

a) Crear un usuario de base de datos

1. Ir a **Database Access** (en el menú izquierdo).
2. Clic en **+ Add New Database User**.
3. Seleccionar **Password** como método de autenticación.
4. Ingresar:
    1. **Nombre de usuario**: `fastapi_user`
    2. **Contraseña**: usa una fuerte (guárdala en un gestor de contraseñas).
5. Asignar permisos: **Read and write to any database** (o personalizar si es necesario).
6. Guardar.

b) Configurar IP de acceso (Network Access)

1. Ir a **Network Access**.
2. Clic en **+ Add IP Address**.
3. Opciones:
    1. **Acceso desde cualquier IP**: `0.0.0.0/0` (solo para desarrollo).
    2. **Acceso desde tu IP actual**: más seguro (recomendado en producción).

4. Guardar.

⚠ ***Advertencia**: `0.0.0.0/0` expone tu base de datos a Internet. Úsalo solo temporalmente en desarrollo.*

5.4. Obtener la URI de Conexión

1. Ir a **Database** > **Clusters** > **Connect**.
2. Seleccionar **"Connect your application"**.
3. Copiar la **Connection String** (URI), que se verá así:

```
MONGO_URI=mongodb+srv://
fastapi_user:<password>@@clusterbase0.kr4et81.mongodb.net/?appName
=ClusterBase0
```

4. **Reemplaza `<password>`** por la contraseña real del usuario de base de datos.

5.5. Conectar la Aplicación a Atlas

En `main.py`, define la URI:

```
# Con variables de entorno (recomendado)
import os
from dotenv import load_dotenv

load_dotenv()
MONGO_URI = os.getenv("MONGO_URI")
```

Y crea el archivo `.env` (agrega a `.gitignore`):

`.env`

```
MONGO_URI=mongodb+srv://
fastapi_user:<password>@@clusterbase0.kr4et81.mongodb.net/?appName
=ClusterBase0
```

☑ ***Buena práctica**: Nunca incluyas credenciales en el código fuente.*

6. Ejecución y Pruebas

6.1. Iniciar el Servidor

bash

```
uvicorn main:app --reload
```

6.2. Acceder a la Documentación Automática

Abrir en el navegador:

- http://localhost:8000/docs → Interfaz Swagger
- http://localhost:8000/redoc → Documentación alternativa

6.3. Probar Endpoints

Ejemplo: Crear una tarea

http

```
POST /tasks/
Content-Type: application/json

{
  "title": "Aprender MongoDB Atlas",
  "description": "Conectar FastAPI a la nube"
}
```

Verificar que se guarda en Atlas usando el **Compass** (cliente gráfico) o el **Data Explorer** en la web.

7. Consideraciones de Seguridad

- 🔒 **Nunca expongas credenciales** en código o repositorios públicos.
- 🌐 **Restringe las IPs** en producción (no uses `0.0.0.0/0`).
- 🔁 **Usa conexiones TLS/SSL**: MongoDB Atlas lo hace por defecto (`+srv` en la URI).
- 💉 **Valida entradas**: Pydantic ya ayuda, pero añade reglas de negocio si es necesario.
- 🛡️ **Considera autenticación** (JWT, OAuth2) si la API será pública.

8. Posibles Mejoras Futuras

Área	Mejora Sugerida
Autenticación	Integrar JWT con `fastapi.security`
Validación	Añadir longitud mínima, caracteres, etc.
Logging	Registrar operaciones críticas
Paginación	Limitar resultados en `/tasks/`
Tests	Agregar pruebas unitarias con `pytest`
Dockerización	Empaquetar en contenedor
Despliegue	Publicar en Render, Railway o AWS

9. Resolución de Errores Comunes

Error	**Causa Probable**	**Solución**
`NameError: name 'pymongo' is not defined`	Uso de `pymongo` sin importar o sin instalación	Usa `motor` + `AsyncIOMotorClient`
`ServerSelectionTimeoutError`	IP no autorizada o credenciales incorrectas	Revisa Network Access y usuario
`ObjectId is not valid`	ID mal formado en ruta	Validar con `ObjectId.is_valid()`
`403 Forbidden`	Usuario sin permisos	Revisa Database Access en Atlas

10. Conclusión

Esta aplicación demuestra una integración moderna, segura y escalable entre un backend en Python (FastAPI) y una base de datos en la nube (MongoDB Atlas). Al seguir buenas prácticas de desarrollo, seguridad y documentación, se obtiene una solución lista para evolucionar hacia entornos productivos.

Anexos

A. Estructura de Archivos

```
proyecto-todo/
├── main.py
├── .env
├── .gitignore
├── requirements.txt
└── README.md
```

B. `requirements.txt`

txt

```
fastapi==0.104.1
uvicorn==0.24.0
motor==3.3.0
python-dotenv==1.0.0
```

C. `.gitignore`

gitignore

```
__pycache__
*.pyc
.env
venv/
```