## Base de Datos

Para este proyecto hemos utilizado una base de datos no relacional con ayuda de MongoDB atlas. Para su conexión hemos creado un *.env* en el que hemos creado una variable *MONGO_URI*, con el valor de la uri de conexión proporcionada por Mongo. Para esta uri utilizamos un user creado para el acceso general a la bd sin más permisos que lectura y escritura, además de permitir el acceso a través de cualquier ip para que pueda probarla el profesorado.

```
MONGO_URI = "mongodb+srv://user:OSm3qwLvpOs2Prar@cluster0.vikio.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0"
```

*(.env)*

Con la uri importada en *.env*, creamos nuestro archivo de configuración de la base de datos para hacer la conexión mediante un *MongoClient* inicializado con la uri y rescatando la base de datos. En el caso de no existir se creara una con el nombre que se haya requerido. Por último tenemos un paso opcional que nos permitirá una escritura posterior, consiste en obtener las colecciones que vayamos a usar referenciando nuestro objeto colección a la colección de la base de datos. Esta también se creará si la colección pedida no existe.

```python
import os
from dotenv import load_dotenv
from pymongo import MongoClient

load_dotenv()

MONGO_URI = os.getenv("MONGO_URI")

mongo = MongoClient(MONGO_URI)
db = mongo.crowdsourcing_database

colaboradores = db.colaboradores
tareas = db.tareas
```

*(db.py)*

En este caso nuestra base de datos será *db* y nuestras colecciones *colaboradores* y *tareas*. Lo siguiente será crear nuestros modelos u objetos que serán insertados en las respectivas colecciones. Para ello es tan simple como crear una clase *BaseModel* e introducir los nombres de las variables y sus tipos. Mongo proporciona un _id automático por lo que utilizar otra variable como id dependerá de nuestra lógica de implementación. En este caso tenemos tres modelos aunque el tercero se utiliza solo para formatear la salida para los contactos de un usuario.

```python
from typing import List
from pydantic import BaseModel, EmailStr

class Colaborador(BaseModel):
    email: EmailStr
    nombre: str
    habilidades: List[str]
```

```python
import datetime
from typing import List, Tuple
from pydantic import BaseModel, EmailStr, Field

class Tarea(BaseModel):
    responsable: EmailStr
    descripcion: str
    habilidades: List[str]
    segmentos: int = Field(..., gt=0)
    colaboradores: List[EmailStr]
```

## Tecnologías Utilizadas

He desarrollado los microservicios con el framework FastAPI de Python, la librería de Uvicorn para lanzar el programa además de docker y la base de datos ya mencionada. He utilizado estás tecnologías ya que me parecen más cómodas para el desarrollo backend además de tener un conocimiento medio en python y la facilidad y comodidad de su uso.

Para ejecutar los microservicios hemos utilizado Docker, en el que para cada microservicio tenemos un contenedor para mantener el aislamiento de estos. Nuestra estructura nos permite comodamente esta ejecución aislada, en esta tenemos un directorio services y un directorio por microservicio en el que tenemos un dockerfile para configurar los contenedores.

```dockerfile
FROM python:3.12

WORKDIR /app
COPY . /app

RUN pip install -r requirements.txt

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000", "--reload"]
```

*Dockerfile*

Para lanzar estos contenedores tenemos un archivo *docker-compose.yml* que lanzará todos los contenedores a la vez.

```yaml
services:
  usuario_service:
    build: ./services/crowdsourcing_service
    ports:
      - "8000:8000"
    environment:
      - MONGO_URI=${MONGO_URI}
    volumes:
      - ./services/crowdsourcing_service:/app
```

*Docker-compose.yml*

Para las pruebas hemos utilizado tanto *Swagger* (accediendo con localhost:{port}/docs) y *Postman*, posteriormente daremos más detalle al respecto.

## Ejecución del servicios

Como ya hemos mencionado antes usaremos docker de la manera explicada, teniendo que escribir en la ventana de comandos del directorio raíz del proyecto:

```
docker-compose up --build
```

Una vez se le notifique la confirmación del correcto despliegue de los contenedores irá teniendo los microservicios desplegados en localhost:8000 y puertos sucesivos en el caso de tener mas microservicios desplegados.

## Pruebas con Postman

Para la prueba y ejecución del profesorado tenemos creada una colección de pruebas con una carpeta por CRUD pedido que contiene peticiones http, en el que solo tiene que pulsar la carpeta y petición http que quiere probar y pulsar enviar en la ventana que le aparece. Alternativamente puede pulsar click derecho en la colección de las pruebas y pulsar documentación, ahi verá todas las peticiones con las llamadas ya definidas y solo tendrá que pulsar open request y send.