Prueba técnica backend

Diseño de una API REST con el framework FastAPI de python

Para esta prueba se requiere conocimiento en python, protocolo http, verbos http (GET, POST, PUT, DELETE). Haciendo uso del framework de desarrollo para python FastAPI, se desea que el participante desarrolle las siguientes funcionalidades:

1. Obtener un listado de <entidad>
2. Obtener la información de <entidad> por id
3. Actualizar la información de una <entidad>
4. Borrar la <entidad>

La <entidad> con la que vamos a trabajar tiene el siguiente modelo de datos:

```
{
   id: string
   name:  string,
   picture: string,
   create_date: string,
   is adopted: bool
}
```

La entidad que vamos a manejar en esta ocasión es Dog. Para cumplir las necesidades listadas en las funcionalidades se requiere tener los siguientes ENPOINTS:

GET -> /api/dogs : Obtener una listado.
GET -> /api/dogs/{name} : Obtener una entrada a partir del nombre.
GET -> /api/dogs/is_adopted : Obtener todas las entradas donde la bandera is_adopted sea True.
POST -> /api/dogs/{name}: Guardar un registro según el esquema de arriba. El campo picture se debe rellenar con una consulta al API externa https://dog.ceo/api/breeds/image/random.
PUT -> /api/dogs/{name}: Actualizar un registro según el nombre.
DELETE -> /api/dogs/{name}: Borrar un registro según el nombre.

Un ejemplo de registro:

```
{
   "name": "Lazy",
   "picture": "https:\/\/images.dog.ceo\/breeds\/papillon\/n02086910_6483.jpg",
    "create_date": "2020-02-20 20:58:55.164954"
    "is_adopted": True,
}
```

Los datos pueden ser guardados utilizando una base de datos (la que el participante desee), o puede optar por hacer uso lista en memoria.

OPCIONAL

1) Las rutas para ingresar un nuevo camino (el POST) debería estar protegida con alguna política de seguridad mediante recomiendo usar JWT.

2) Realizar dockerfile y docker-compose de la aplicación.

3) Desarrollar un Worker con Celery cuando se llame la función POST. Las tareas en segundo plano tienen sentido para actividades que se pueden realizar de forma asíncrona. Puedes simular una latencia en el registro de los perros de unos segundos para verificar que el "Worker" está funcionando.

4) Añadir una nueva entidad llamada User con sus respectivos endpoints de CRUD básico. Esta entidad tendrá una relación de uno a muchos con la entidad Dog, es decir, un User puede tener uno o muchos Dogs (para esto solo se requiere que la entidad Dog se agregue un nuevo campo llamado id_user y guarde el id de un usuario). Los campos para la entidad User pueden ser: id, nombre, apellido, email.

Finalmente, el código debe estar en un repositorio en tu GIT personal llamado "guane-intern-fastapi". También debes seguir el estándar del PEP8 en la codificación con lineamientos de buenas prácticas.