Memoria: Práctica de Backend

Ingeniería Web

Diseño De la Base de Datos

El diseño de la base de datos esta hecho dentro de mongoDB con validaciones. Aqui le dejo unas capturas de pantalla con eso para que vea como se ha construido

Entidad colaboradores:

```
{
  $jsonSchema: {
    required: [
      '_id',
      'email',
      'nombre',
      'habilidades'
    ],
    additionalProperties: false,
    properties: {
      _id: {
        bsonType: 'objectId',
        description: 'ID de la version'
      },
      email: {
        bsonType: 'string',
        description: 'email del usuario'
      },
      nombre: {
        bsonType: 'string',
        description: 'nombre del usuario'
      },
      habilidades: {
        bsonType: 'array',
        description: 'conjunto de habilidades del usuario'
      }
    }
  }
}
```

Entidad tareas

```
{
  $jsonSchema: {
    required: [
      '_id',
      'responsable',
      'descripcion',
      'habilidades',
      'segmentos'
    ],
    additionalProperties: false,
    properties: {
      _id: {
        bsonType: 'objectId',
        description: 'ID de la version'
      },
      responsable: {
        bsonType: 'string',
        description: 'email del responsable de la tarea'
      },
      descripcion: {
        bsonType: 'string',
        description: 'descripcion de la tarea'
      },
      habilidades: {
        bsonType: 'array',
        description: 'una serie de habilidades (términos) adecuadas para participar en la tarea.'
      },
      segmentos: {
        bsonType: 'int',
        description: 'duración estimada de la tarea (en segmentos de 1 hora de trabajo).'
      }
    }
  }
}
```

He añadido una tercera identidad que se llama asignaciones, en esta tendremos constancia de todas las asignaciones entre los colaboradores y las tareas

```
{
  $jsonSchema: {
    required: [
      '_id',
      'nombre',
      'tarea',
      'segmento'
    ],
    additionalProperties: false,
    properties: {
      _id: {
        bsonType: 'objectId',
        description: 'ID de la version'
      },
      nombre: {
        bsonType: 'string',
        description: 'nombre del colaborador'
      },
      tarea: {
        bsonType: 'objectId',
        description: 'id de la tarea asignada'
      },
      segmento: {
        bsonType: 'int',
        description: 'segmento en el que trabaja'
      }
    }
  }
}
```

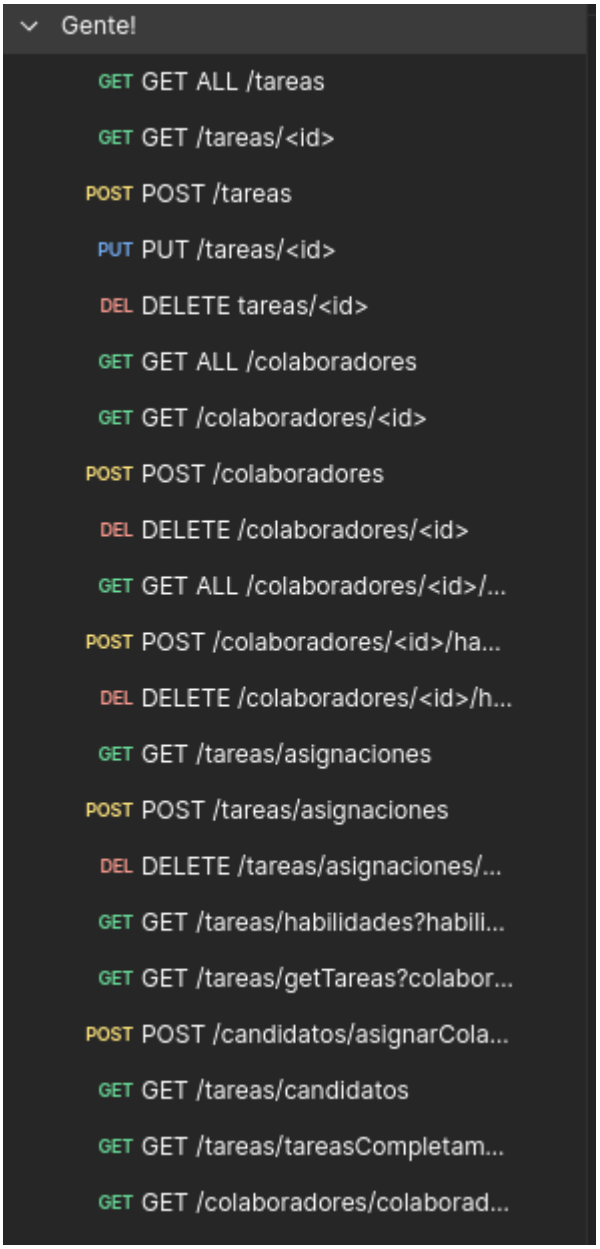
No me va a dar tiempo a hacer el CRUD de esto asi que he hecho el post y el getAll, por si lo quieres tener en cuenta

Tecnologías

Las tecnologías utilizadas en la práctica (lenguajes, bibliotecas, frameworks, base de datos, etc.). Para la base de datos vamos a utilizar MongoDB Atlas. Para construir la API usaremos el framework Flask de Python. Usaremos pymongo para conectarnos a la base de datos. Como ejemplos de pruebas ejecutables se ha realizado un conjunto de pruebas en Postman.

Postman

En el zip hay un archivo json con las pruebas Postman para probar que funciona



En total hay 21 Postmans

Los 5 primeros son del CRUD de la entidad tareas
Los 7 siguientes son del CRUD de la entidad colaboradores, con su CRUD de habilidades del colaborador
Los 3 siguientes son del CRUD de la entidad asignaciones
Y los últimos 6 son de los servicios adicionales

Docker

Solo sería necesario construir el contenedor y ejecutarlo. Para eso siga estos pasos
Descomprimir el archivo .zip
Ejecutar en terminal => docker compose up -build (docker-compose up -build en caso de tener la otra versión)
Todas las pruebas de postman están definidas sobre el puerto que hay en el .env, el 5000. En caso de querer usar otro puerto sería modificarlo en el -env

Endpoints de la entidad tareas

La entidad tareas contiene los puntos de acceso necesarios para gestionar la entidad tareas. A continuación, especificamos los detalles de cada endpoint de la entidad para facilitar su uso.

Tareas

GET ALL
En la entidad tareas el método get all lo hacemos buscando en la base de datos con el método find() por lo tanto obtenemos todo.
GET
En la entidad tareas el método get lo hacemos buscando en la base de datos con el método find() y aplicando los filtros respectivos.
POST
En la entidad tareas el método post lo hacemos insertando los datos previamente obtenidos con el método insert().
PUT
En la entidad tareas el método put lo hacemos actualizando los datos previamente obtenidos con el método update().
DELETE

En la entidad tareas el método delete lo hacemos borrando la entrada con el método delete()

Endpoints de la entidad colaboradores

La entidad colaboradores contiene los puntos de acceso necesarios para gestionar la entidad colaboradores. A continuación, especificamos los detalles de cada endpoint de la entidad para facilitar su uso.

Colaboradores

GET ALL
En la entidad colaboradores el método get all lo hacemos buscando en la base de datos con el método find() por lo tanto obtenemos todo.
GET
En la entidad colaboradores el método get lo hacemos buscando en la base de datos con el método find() y aplicando los filtros respectivos.
POST
En la entidad colaboradores el método post lo hacemos insertando los datos previamente obtenidos con el método insert().
PUT
En la entidad colaboradores el método put lo hacemos actualizando los datos previamente obtenidos con el método update().
DELETE
En la entidad colaboradores el método delete lo hacemos borrando la entrada con el método delete()

Endpoints de la entidad asignaciones

La entidad asignaciones contiene los puntos de acceso necesarios para gestionar la entidad asignaciones. A continuación, especificamos los detalles de cada endpoint de la entidad para facilitar su uso.

Asignaciones

GET ALL
En la entidad asignaciones el método get all lo hacemos buscando en la base de datos con el método find() por lo tanto obtenemos todo.

POST
En la entidad asignaciones el método post lo hacemos insertando los datos previamente obtenidos con el método insert().

Servicios REST adicionales:

Ruta `http://127.0.0.1:5000/tareas?habilidad=nombreDeLaHabilidad`
Explicación: Tareas que requieren una determinada habilidad, devolviendo una lista de tareas.

Ruta `http://127.0.0.1:5000/tareas/getTareas?colaborador=nombreDelColaborador`
Explicación: Tareas asignadas a un determinado colaborador, devolviendo una lista de tareas.

Ruta `/tareas/asignarColaborador`
Explicación: Asignar un colaborador a una tarea, para lo cual se comprobará que el colaborador posea al menos una de las habilidades requeridas por la tarea.

Ruta `/tareas/candidatos?tarea=idDeLaTarea`
Explicación: Buscar candidatos a colaborar en una tarea, devolviendo una lista de emails de colaboradores que posean al menos una de las habilidades requeridas por la tarea

Ruta `/tareas/tareasCompletamenteAsignadas`
Explicación:Tareas completamente asignadas, que devolverá las tareas que tengan asignados tantos colaboradores como segmentos de los que consta la tarea

Ruta `/colaboradores/colaboradoresUsuario?responsable=`
Explicación: Buscar los colaboradores de un determinado usuario, devolviendo una lista de emails de colaboradores que participen en alguna de las tareas de las que el usuario es responsable.