Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelos de Desarrollo Web

NRC: 45316

Actividad 1: API solo GET

Iván Pérez Sánchez

202247984

Mtro. Alfredo García Suárez

15 de junio de 2025

Para esta actividad se realizó una API en la que, a partir de peticiones GET y un manejo de diferentes niveles de la ruta de la URL, se pueda obtener información acerca de los datos almacenados en la base de datos sobre estudiantes de la Facultad de Ciencias de la Computación.

Archivos

Se generaron dos archivos: `models.py` y `main.py`. El primero de ellos es el que contiene el modelo utilizado para almacenar datos, los cuales lee a partir de un archivo CSV de nombre `Base_Datos.csv`; para ello se utilizaron las bibliotecas `pydantic`, `csv` y `sys`.

Con respecto al modelo creado, este es una clase llamada `Estudiante` que hereda de `BaseModel`, clase propia de la biblioteca `pydantic`, y contiene los campos de la base de datos creada en sesiones anteriores. Este modelo se puede comprender de mejor manera con el diagrama ER presentado en la figura 1.

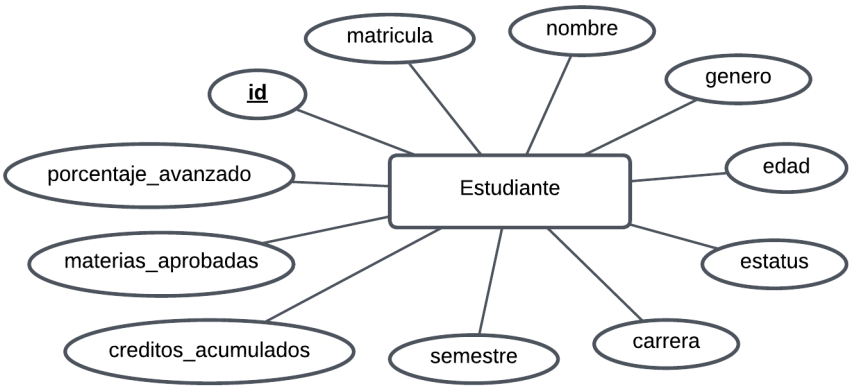

Figura 1. Diagrama ER del modelo Estudiante

Por otro lado, se tiene una función llamada `load_students()` que se encarga de leer el archivo CSV y generar, a partir de él, una lista de objetos de tipo `Estudiante`. Cuando acaba de leer el archivo, retorna dicha lista; sin embargo, si llega a haber un problema al abrirlo o leerlo, se marca una excepción y se finaliza la ejecución del programa.

Pasando a main.py, este importa a model.py y a la biblioteca `fastapi` (específicamente la clase homónima), pues es el archivo que implementa la API. Primero, se cargan los datos en memoria mediante la función `load_students()`, descrita anteriormente; cuando finaliza la carga, se genera una instancia de la clase `FastAPI` de nombre `app` y, con base en ella, se definen las peticiones GET.

Peticiones

Antes de explicar las peticiones, a continuación, se presenta el desglose de los endpoints de la URL en forma de árbol, mostrando los tres niveles solicitados del path:

```
estudiantes/
├── activos/
├── genero/
│   ├── semestre
│   └── creditos_max
├── semestre/
│   └── porcentaje_min
└── carrera/
    └── materias_min
```

Así pues, se tienen tres endpoints en el primer nivel, cuatro en el segundo y cuatro en el tercero. No obstante, algunos no están totalmente implementados o tienen ciertas cuestiones en la implementación. Esto se explicará a continuación.

Primer nivel

En este primer nivel se tienen los siguientes endpoints:

1. `GET /estudiantes`

Este endpoint sirve para proporcionar dos posibles resultados. Si se proporciona el parámetro carrera, filtrará la lista por ese valor; en caso contrario, devolverá la lista completa de estudiantes (es por ello que este endpoint se consideró como dos).

El parámetro proporcionado puede ser de tipo `str` o `None`; y la petición devolverá como resultado un JSON.

Ejemplos de uso son los siguientes:

- `/estudiantes`
- `/estudiantes?carrera=Ingeniería en Ciencias de la Computación`
- `/estudiantes?carrera=Ingeniería en Tecnologías de la Información`

2. `GET /estudiantes/{matricula}`

Este endpoint devuelve un JSON. Si la matrícula dada en el parámetro `{matricula}` existe, el JSON contendrá la información del estudiante asociado; en caso contrario contendrá un mensaje de error.

Ejemplos de uso son los siguientes:

- `/estudiantes/202247984`
- `/estudiantes/202212345`
- `/estudiantes/202312345`

Segundo nivel

En este nivel se agruparon filtros más específicos que se anidan bajo `estudiantes/`. Hay cuatro rutas base, cada una de ellas representa una dimensión adicional de segmentación, no obstante, solo se tomarán en cuenta tres, dado que una de ellas (`/estudiantes/carrera`) solo es usada para implementar una petición del tercer nivel, y ella per se no devuelve nada.

1. `GET /estudiantes/activos`

Devuelve la lista de todos los alumnos cuyo campo estatus sea exactamente "Activo". Si no hay alumnos que tengan ese valor en dicho campo, se devuelve un mensaje de error.

Ejemplo de uso es el siguiente:

- `/estudiantes/activos`

2. `GET /estudiantes/genero/{genero}`

Retorna todos los estudiantes cuyo atributo genero coincida exactamente con el parámetro `{genero}`. Si ocurre algún error en el parámetro, se devuelve un mensaje señalándolo.

Ejemplos de uso son los siguientes:

- `/estudiantes/genero/Femenino`
- `/estudiantes/genero/Masculino`

3. `GET /estudiantes/semestre/{semestre}`

Lista todos los estudiantes que se encuentren en el semestre `{semestre}`, siendo este un parámetro. Si ocurre algún error en el parámetro, se señala en la respuesta de la petición.

Ejemplos de uso son los siguientes:

- `/estudiantes/semestre/6`
- `/estudiantes/semestre/2`
- `/estudiantes/semestre/7`

Tercer nivel

En este nivel se combinan dos criterios de filtrado para obtener segmentos aún más precisos. Hay cuatro endpoints específicos:

1. `GET /estudiantes/genero/{genero}/semestre/{semestre}`

Devuelve los estudiantes que cumplen los criterios dados en los parámetros `{genero}` y `{semestre}`. Si no se cumplen, se devuelve un mensaje indicándolo.

Ejemplos de uso son los siguientes:

- `/estudiantes/genero/Femenino/semestre/6`
- `/estudiantes/genero/Masculino/semestre/6`
- `/estudiantes/genero/Masculino/semestre/2`

2. `/estudiantes/semestre/{semestre}/porcentaje_min/{porcentaje_min}`

Recupera a todos los alumnos en el semestre `{semestre}` cuyo `porcentaje_avanzado` sea mayor o igual a `{porcentaje_min}`. Si no hay alumnos que cumplan dichos criterios, se informa en la respuesta.

Ejemplos de uso son los siguientes:

- `/estudiantes/semestre/4/porcentaje_min/40`
- `/estudiantes/semestre/6/porcentaje_min/70`
- `/estudiantes/semestre/8/porcentaje_min/80`

3. `GET /estudiantes/{carrera}/{materias_min}`

Lista los estudiantes cuya carrera coincide con `{carrera}` y que tienen al menos `{materias_min}` materias aprobadas. Si no hay estudiantes que cumplan con esos requisitos, se informa en el JSON de respuesta.

Ejemplos de uso son los siguientes:

- `/estudiantes/Ingeniería en Tecnologías de la Información/30`
- `/estudiantes/Ingeniería en Ciencias de la Computación/30`
- `/estudiantes/Ingeniería en Tecnologías de la Información/40`

4. `GET /estudiantes/genero/{genero}/creditos_max/{creditos_max}`

Obtiene todos los estudiantes de género `{genero}` cuyo total de `creditos_acumulados` sea menor o igual a `{creditos_max}`. Si no hay alumnos que cumplan con dichas condiciones, se informa en la respuesta.

Ejemplos de uso:

- `/estudiantes/genero/Masculino/creditos_max/180`
- `/estudiantes/genero/Femenino/créditos_max/200`
- `/estudiantes/genero/Femenino/créditos_max/160`