

Programação de Dispositivos Móveis 2

Sure Rocha Bezerra
P5 - Informática
Avaliação 06

Enunciado:

Desenvolver um Backend Python que usa FASTAPI

Como um programador Python, crie uma aplicação backend que usa o framework FASTAPI seguindo os seguintes passos:

1) Crie um banco de dados SQLITE3 com o nome dbalunos.db.

2) Crie uma entidade aluno que será persistida em uma tabela TB_ALUNO com os seguintes campos:

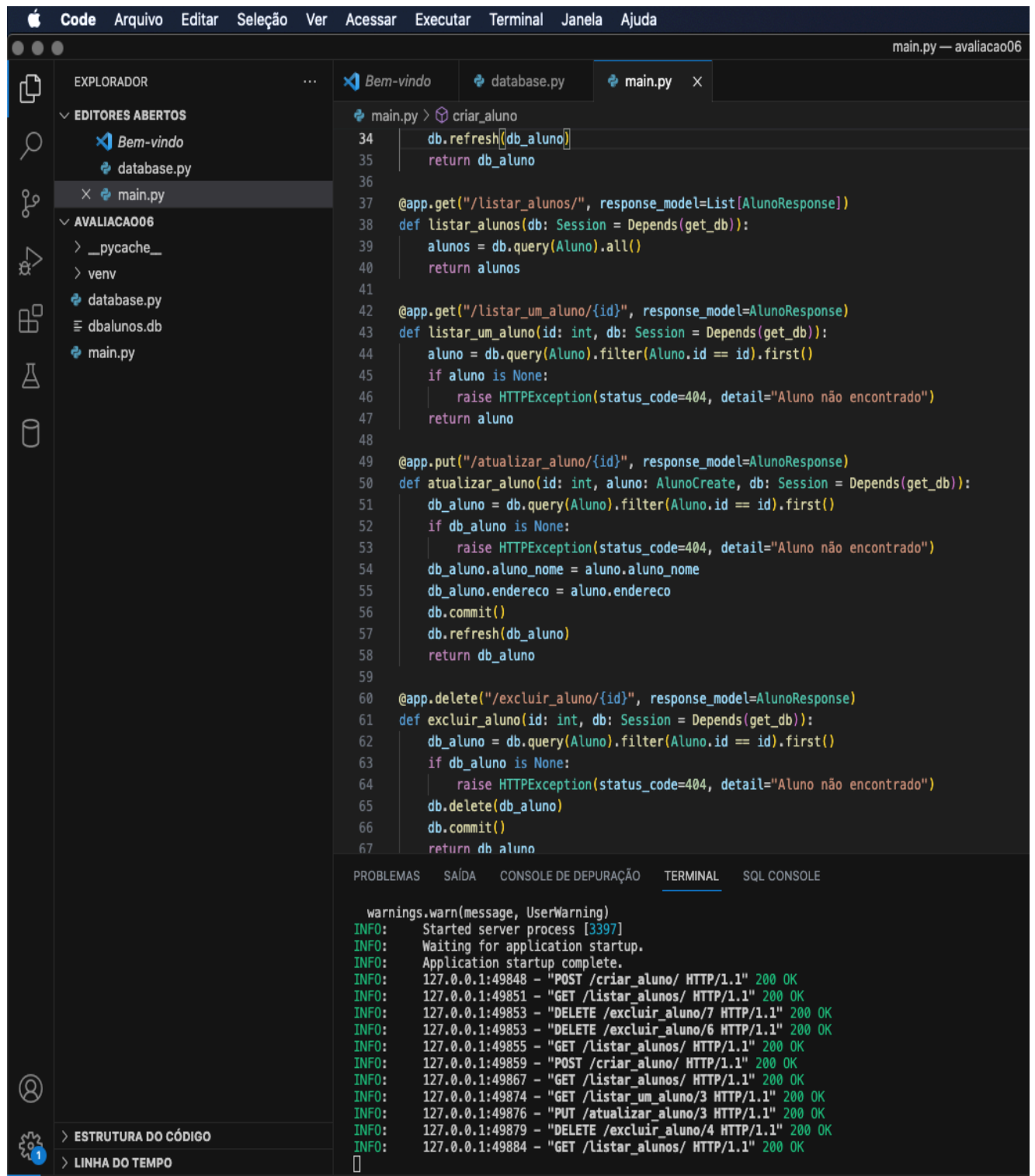
- id chave primária do tipo inteiro com autoincremento;
- aluno_nome do tipo string com tamanho 50;
- endereço do tipo string com tamanho 100;

3) Crie os seguintes endpoints FASTAPI abaixo descritos:

- a) criar_aluno grava dados de um objeto aluno na tabela TB_ALUNO;
- b) listar_alunos ler todos os registros da tabela TB_ALUNO;
- c) listar_um_aluno ler um registro da tabela TB_ALUNO a partir do campo id;
- d) atualizar_aluno atualiza um registro da tabela TB_ALUNO a partir de um campo id e dos dados de uma entidade aluno;
- e) excluir_aluno exclui um registro da tabela TB_ALUNO a partir de um campo id e dos dados de uma entidade aluno;

Próximas páginas com os resultados:

Código funcionando:



The image shows a VS Code editor window with a Python Flask application. The file explorer on the left shows the project structure, including a folder named 'AVALIACAO06' and files like 'database.py' and 'main.py'. The main editor displays the code for 'main.py', which defines several Flask routes for a school database application. The routes include creating a student, listing all students, getting a student by ID, updating a student, and deleting a student. The terminal at the bottom shows the server running and several successful HTTP requests.

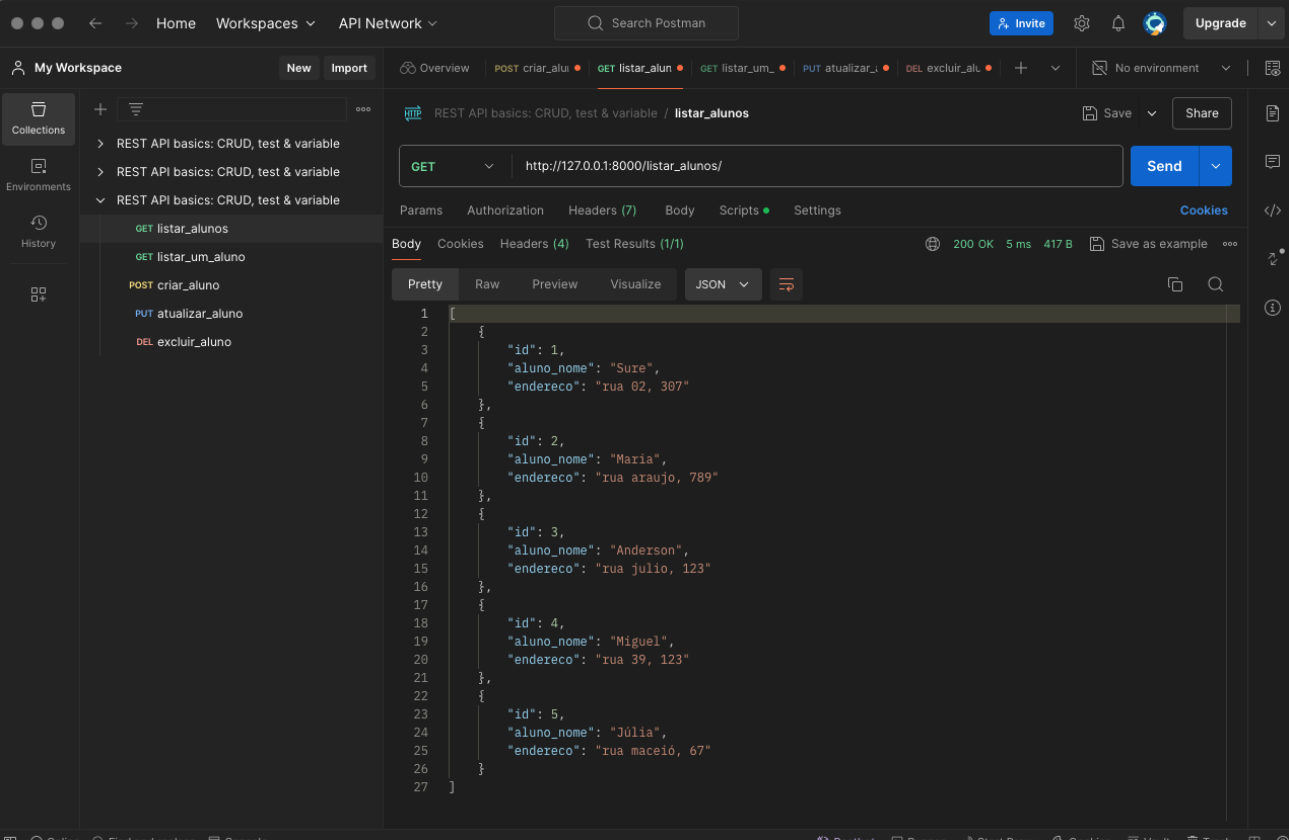
```
main.py — avaliacao06

EXPLORADOR
EDITORES ABERTOS
  Bem-vindo
  database.py
  main.py
AVALIACAO06
  __pycache__
  venv
  database.py
  dbalunos.db
  main.py

main.py > criar_aluno
34 db.refresh(db_aluno)
35 return db_aluno
36
37 @app.get("/listar_alunos/", response_model=List[AlunoResponse])
38 def listar_alunos(db: Session = Depends(get_db)):
39     alunos = db.query(Aluno).all()
40     return alunos
41
42 @app.get("/listar_um_aluno/{id}", response_model=AlunoResponse)
43 def listar_um_aluno(id: int, db: Session = Depends(get_db)):
44     aluno = db.query(Aluno).filter(Aluno.id == id).first()
45     if aluno is None:
46         raise HTTPException(status_code=404, detail="Aluno não encontrado")
47     return aluno
48
49 @app.put("/atualizar_aluno/{id}", response_model=AlunoResponse)
50 def atualizar_aluno(id: int, aluno: AlunoCreate, db: Session = Depends(get_db)):
51     db_aluno = db.query(Aluno).filter(Aluno.id == id).first()
52     if db_aluno is None:
53         raise HTTPException(status_code=404, detail="Aluno não encontrado")
54     db_aluno.aluno_nome = aluno.aluno_nome
55     db_aluno.endereco = aluno.endereco
56     db.commit()
57     db.refresh(db_aluno)
58     return db_aluno
59
60 @app.delete("/excluir_aluno/{id}", response_model=AlunoResponse)
61 def excluir_aluno(id: int, db: Session = Depends(get_db)):
62     db_aluno = db.query(Aluno).filter(Aluno.id == id).first()
63     if db_aluno is None:
64         raise HTTPException(status_code=404, detail="Aluno não encontrado")
65     db.delete(db_aluno)
66     db.commit()
67     return db_aluno

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL SQL CONSOLE
warnings.warn(message, UserWarning)
INFO: Started server process [3397]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:49848 - "POST /criar_aluno/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:49851 - "GET /listar_alunos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:49853 - "DELETE /excluir_aluno/7 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49853 - "DELETE /excluir_aluno/6 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49855 - "GET /listar_alunos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:49859 - "POST /criar_aluno/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:49867 - "GET /listar_alunos/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:49874 - "GET /listar_um_aluno/3 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49876 - "PUT /atualizar_aluno/3 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49879 - "DELETE /excluir_aluno/4 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49884 - "GET /listar_alunos/ HTTP/1.1" 200 OK
```

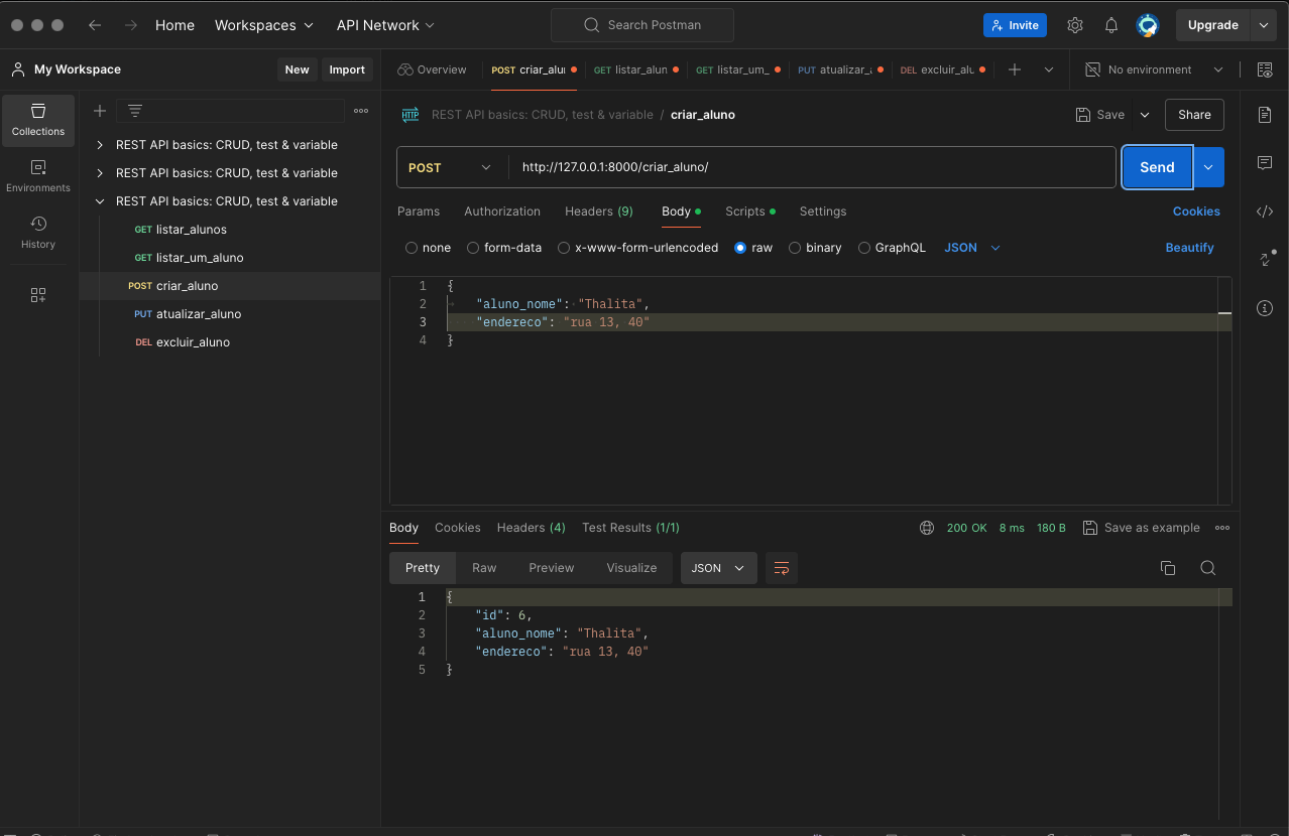
Lista dos alunos antes do POST:



The screenshot shows the Postman interface with a workspace named "My Workspace". The "Collections" panel on the left shows a collection named "REST API basics: CRUD, test & variable" with several endpoints. The "History" panel shows a list of requests, including "GET listar_alunos". The main panel displays a GET request to "http://127.0.0.1:8000/listar_alunos/" with a status of "200 OK", a response time of "5 ms", and a body size of "417 B". The response body is a JSON array of 5 students, each with an "id", "aluno_nome", and "endereco".

```
1 {
2   {
3     "id": 1,
4     "aluno_nome": "Sure",
5     "endereco": "rua 02, 307"
6   },
7   {
8     "id": 2,
9     "aluno_nome": "Maria",
10    "endereco": "rua araujo, 789"
11  },
12  {
13    "id": 3,
14    "aluno_nome": "Anderson",
15    "endereco": "rua julio, 123"
16  },
17  {
18    "id": 4,
19    "aluno_nome": "Miguel",
20    "endereco": "rua 39, 123"
21  },
22  {
23    "id": 5,
24    "aluno_nome": "Júlia",
25    "endereco": "rua maceió, 67"
26  }
27 }
```

Realizando POST:

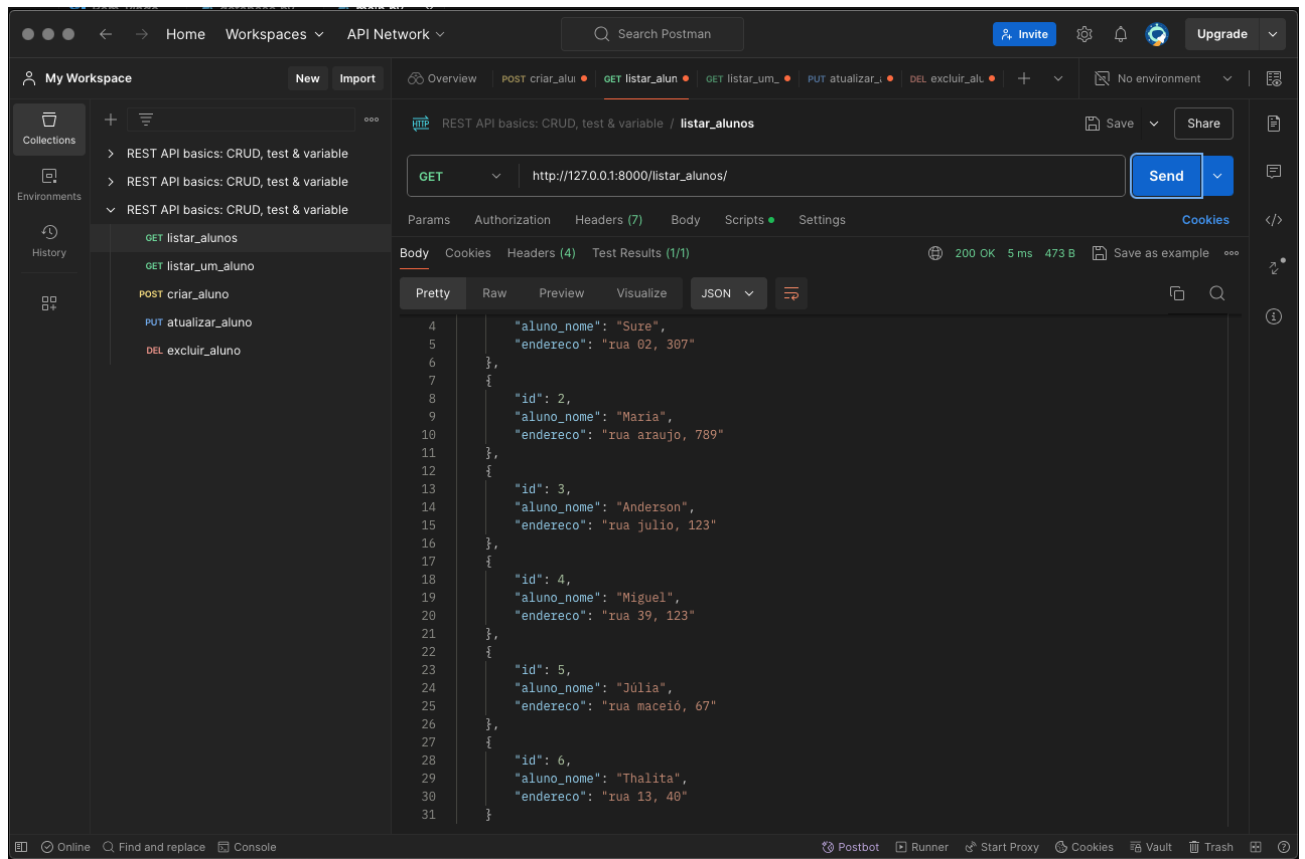


The screenshot shows the Postman interface with the same workspace. The "Collections" panel shows the same collection. The "History" panel shows a list of requests, including "POST criar_aluno". The main panel displays a POST request to "http://127.0.0.1:8000/criar_aluno/" with a status of "200 OK", a response time of "8 ms", and a body size of "180 B". The request body is a JSON object with "aluno_nome": "Thalita" and "endereco": "rua 13, 40". The response body is a JSON object with "id": 6, "aluno_nome": "Thalita", and "endereco": "rua 13, 40".

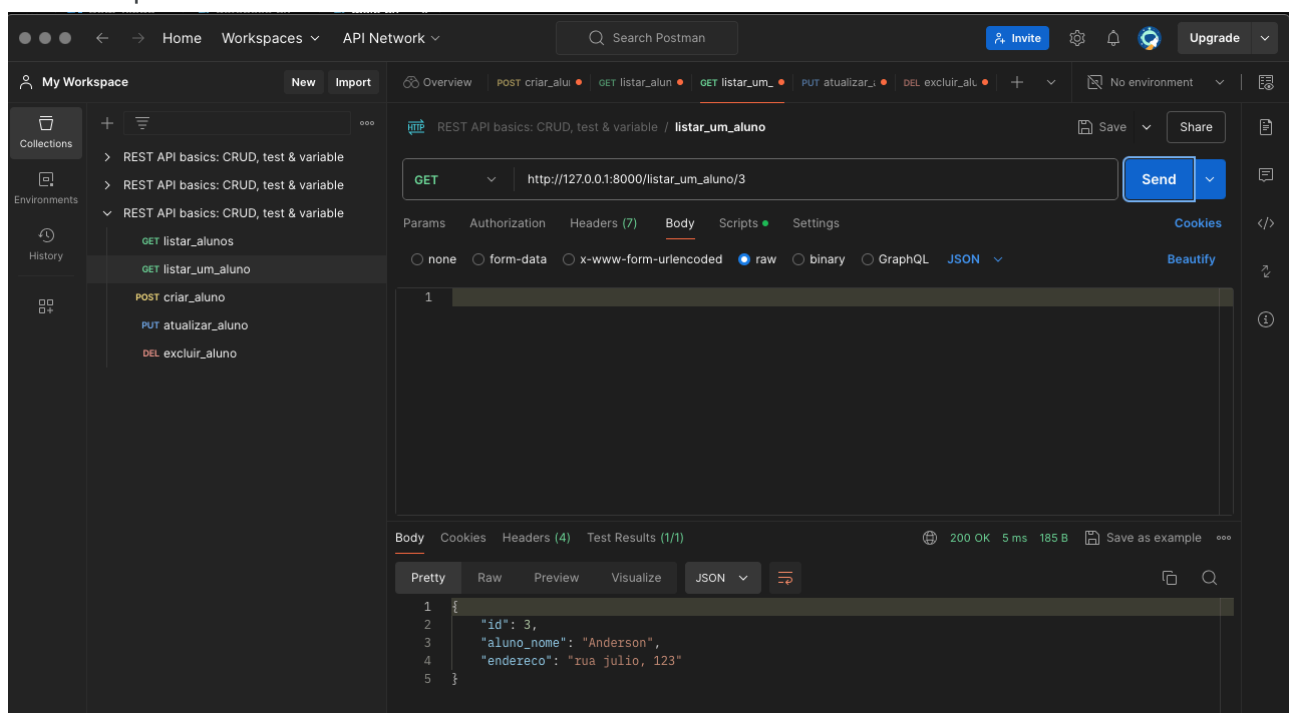
```
1 {
2   "aluno_nome": "Thalita",
3   "endereco": "rua 13, 40"
4 }
```

```
1 {
2   "id": 6,
3   "aluno_nome": "Thalita",
4   "endereco": "rua 13, 40"
5 }
```

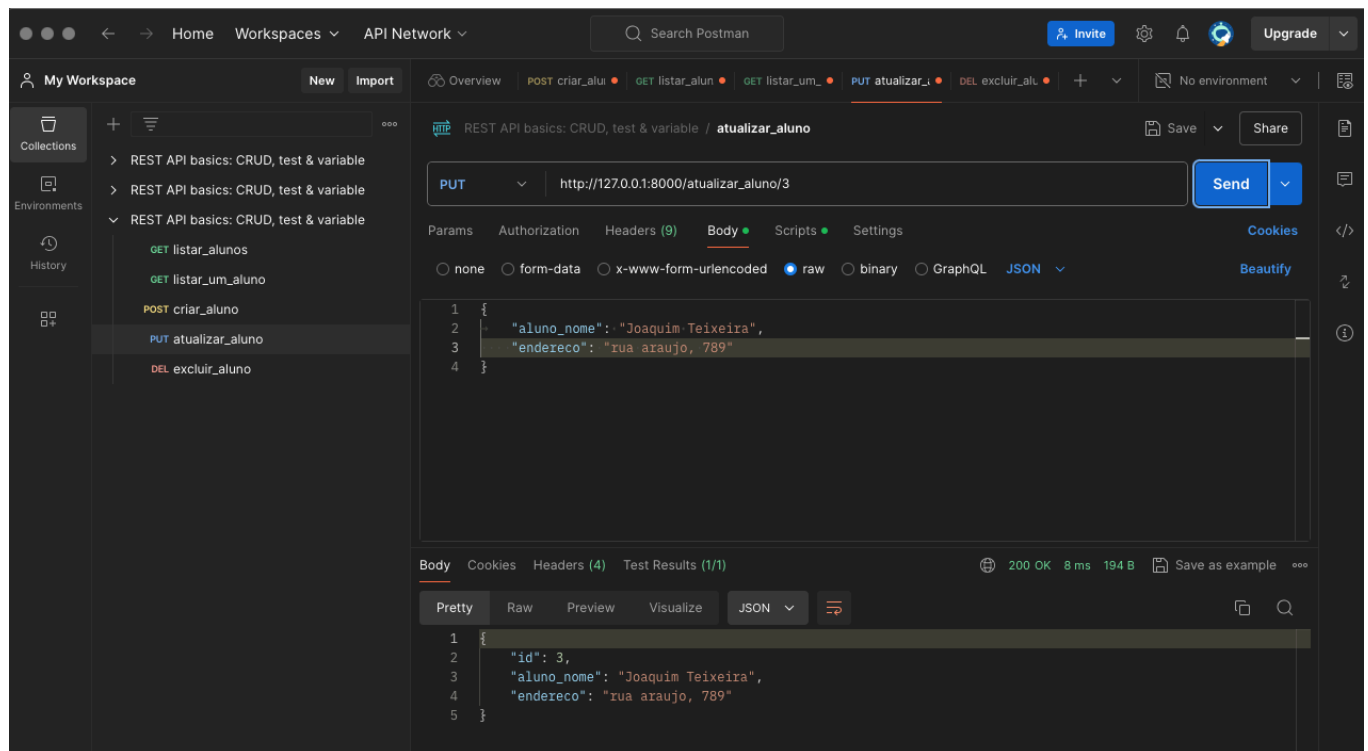
Lista dos alunos depois do POST:



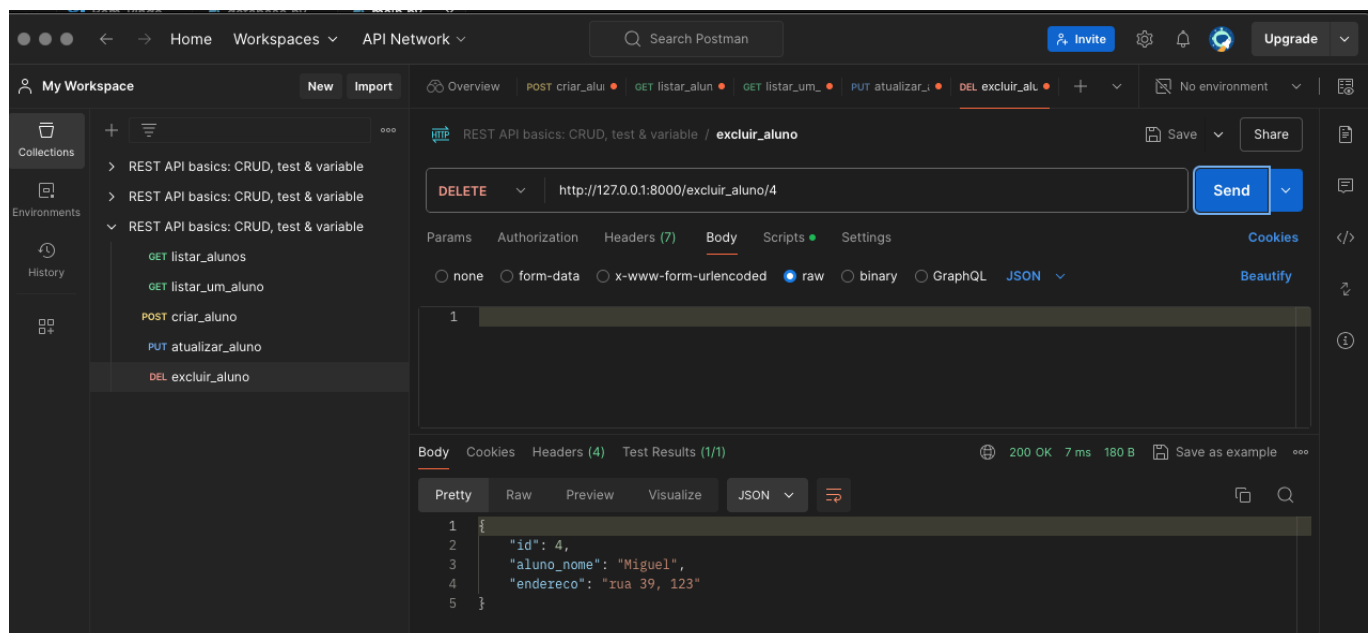
Listando apenas 1 aluno:



Realizando PUT:



Realizando DELETE:



Lista depois do PUT e DELETE:

The screenshot shows the Postman interface with a REST API test named 'listar_alunos' selected. The test is a GET request to the endpoint 'http://127.0.0.1:8000/listar_alunos/'. The response is a JSON array of student records, displayed in the 'Body' tab. The status is 200 OK, with a response time of 6 ms and a body size of 426 B. The left sidebar shows the workspace structure, and the bottom status bar indicates the application is online.

REST API basics: CRUD, test & variable / listar_alunos

GET **http://127.0.0.1:8000/listar_alunos/** **Send**

Params Authorization Headers (7) Body Scripts Settings Cookies

Body Cookies Headers (4) Test Results (1/1) 200 OK 6 ms 426 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "aluno_nome": "Suzie",
5     "endereco": "rua 02, 307"
6   },
7   {
8     "id": 2,
9     "aluno_nome": "Maria",
10    "endereco": "rua araujo, 789"
11  },
12  {
13    "id": 3,
14    "aluno_nome": "Joaquim Teixeira",
15    "endereco": "rua araujo, 789"
16  },
17  {
18    "id": 5,
19    "aluno_nome": "Júlia",
20    "endereco": "rua maceió, 67"
21  },
22  {
23    "id": 6,
24    "aluno_nome": "Thalita",
25    "endereco": "rua 13, 40"
26  }
27 ]
```