

1. Listar todos los estudiantes

The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'History' section listing several requests made today and on January 28. A 'Create collections in Postman' section is also present. The main area displays a request configuration for a 'GET' method to 'http://localhost:8080/Estudiantes'. The 'Params' tab is selected. In the 'Body' tab, the response is shown in JSON format:

```
1 {
2   "nombre": "Juan Modificado",
3   "email": "juan.nuevo@email.com"
4 },
5 {
6   "nombre": "Juan Perez",
7   "email": null
8 },
9 {
10  "nombre": "Juan Perez",
11  "email": null
12 }
```

The status bar at the bottom indicates a 200 OK response with 62 ms latency and 576 B size.

2. Eliminar estudiantes por su id

The screenshot shows the Postman application interface. The 'History' sidebar lists various requests. The main area shows a request configuration for a 'DELETE' method to 'http://localhost:8080/Estudiantes/1'. The 'Params' tab is selected. In the 'Body' tab, the response is shown as a single digit '1'.

3. Obtener estudiante por su id

The screenshot shows the Postman interface. On the left, the 'History' sidebar lists several requests, including a recent 'GET http://localhost:8080/estudiantes/3'. The main panel displays a 'GET' request to 'http://localhost:8080/estudiantes/3'. The 'Params' tab is selected, showing an empty table. The 'Body' tab is selected, showing a JSON response with the following content:

```
1
2
3
4
5
6
7
8
9
10
11
```

The response body is a JSON object with the following structure:

```
{"nombre": "Juan Perez", "email": null, "_links": {"self": {"href": "http://localhost:8080/estudiantes/3"}, "estudiante": {"href": "http://localhost:8080/estudiantes/3"}}
```

4. insertar estudiantes

The screenshot shows the Postman interface. On the left, the 'History' sidebar lists several requests, including a recent 'POST http://localhost:8080/Estudiantes'. The main panel displays a 'POST' request to 'http://localhost:8080/Estudiantes'. The 'Body' tab is selected, showing the 'raw' option selected. The 'Body' field contains the following JSON data:

```
1
2
3
4
```

The JSON data is:

```
{"nombre": "Luis Gomez", "email": "luis.gomez@gmail.com"}
```

5. Actualizar estudiante por su id

The screenshot shows the Postman application interface. On the left, there's a sidebar with a history of requests and a 'Create collections' section. The main area shows a 'PUT' request to 'http://localhost:8080/Estudiantes/2'. The 'Body' tab is selected, displaying the following JSON payload:

```

1  {
2     "nombre": "Juan Modificado",
3     "email": "juan.nuevo@email.com"
4 }

```

Below the body, the response status is shown as 200 OK with a duration of 252 ms and a size of 223 B. The response body is also displayed in JSON format:

```

1  {
2     "nombre": "Juan Modificado",
3     "email": "juan.nuevo@email.com"
4 }

```

Mi Base de datos:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables like 'ciudades', 'enlaces', 'estudiante', and 'ddi'. The main area shows the results of a query: 'SELECT * FROM ddi.estudiante;'. The results grid contains the following data:

	id	email	nombre
1	2	juan.nuevo@email.com	Juan Modificado
2	3	NULL	Juan Perez
3	4	NULL	Juan Perez
4	5	NULL	Juan Perez
5	6	NULL	Juan Perez
6	7	NULL	Juan Perez
7	8	NULL	Juan Perez
8	9	NULL	Juan Perez
9	10	NULL	Juan Perez
10	11	luis.gomez@gmail.com	Luis Gomez

At the bottom, the 'Action Output' section shows the execution details for the query:

- Time Action Message Duration / Fetch
- 1 08:14:33 SELECT * FROM ddi.estudiante LIMIT 0, 1000 9 row(s) returned 0.016 sec / 0.000 sec
- 2 08:22:41 SELECT * FROM ddi.estudiante LIMIT 0, 1000 11 row(s) returned 0.000 sec / 0.000 sec
- 3 08:38:04 SELECT * FROM ddi.estudiante LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
- 4 08:40:29 SELECT * FROM ddi.estudiante LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec