API de Morosos con Spring Boot + Postman (LinkedList, GET/POST/DELETE)

Objetivo

Construir una API REST capaz de:

- GET: obtener la lista completa de morosos desde una LinkedList.
- POST: añadir nuevos morosos a la LinkedList.
- DELETE: eliminar un moroso por id de la LinkedList.

Todas las pruebas se realizarán exclusivamente con Postman (sin frontend).

PARTE A — Usar Postman dentro de VS Code

1) Instalar y preparar la extensión

- Abre VS Code y ve a la parte de las extensiones (Ctrl+Shift+X).
- Busca “Postman” y instala la extensión oficial (publisher: Postman).
- Abre el panel Postman (barra lateral izquierda).

2) Crear Workspace, Environment y Collection

En el panel de Postman:

- Crea un Workspace llamado DWES-Morosos.
- Crea un Environment llamado DWES-Morosos (local) con variables:
  + baseUrl = http://localhost:8080
  + morososPath = /api/morosos
- Crea una Collection llamada Morosos API.

3) Añadir Requests y Tests (dentro de VS Code)

- En la Collection Morosos API, crea tres requests. En todos, usa la URL con variables: {{baseUrl}}{{morososPath}}

  A. GET lista completa
  - Method: GET
  - URL: {{baseUrl}}{{morososPath}}
  - Tests (pestaña Tests del request en la extensión):

```
pm.test("GET lista: 200 OK", () => pm.response.to.have.status(200));
pm.test("Devuelve un array", () => {
 const data = pm.response.json();
 pm.expect(Array.isArray(data)).to.be.true;
});
```

B. POST crear moroso

- Method: POST
- URL: {{baseUrl}}{{morososPath}}
- Headers: Content-Type: application/json
- Body (raw/JSON):

```
{
 "nombre": "Ana López",
 "dni": "12345678A",
 "email": "ana@example.com",
 "telefono": "600111222",
 "importe": 120.50,
 "concepto": "Cuota atrasada"
}
```

- Tests:

```
pm.test("POST crear: 201 Created", () => pm.response.to.have.status(201));
pm.test("Devuelve objeto con id", () => {
 const m = pm.response.json();
 pm.expect(m).to.have.property("id");
});
```

C. DELETE por id

- Method: DELETE
- URL: {{baseUrl}}{{morososPath}}/1 ← cambia 1 por el id creado
- Tests:

```
pm.test("DELETE: 204 No Content", () => pm.response.to.have.status(204));
```

PARTE 2 — Controladores que debes implementar (sin código, especificación)

1. Reglas de diseño

- Estructura de datos: una LinkedList<Moroso> que actúe como lista en memoria de la aplicación.
- Puede residir como campo static dentro del controlador o en una clase "repositorio" sencilla.
- Usa un generador de ids (por ejemplo, AtomicLong) para asignar id incrementales en POST.
- Modelo: utilizar Moroso **(se proporciona).**
- Formato: JSON de entrada/salida.

2. Endpoints requeridos

2.1 GET /api/morosos (Devuelve la lista completa (contenido de la LinkedList).)

Respuesta: 200 OK + application/json con un array.

Ejemplo de respuesta:


```
[
 {
  "id": 1,
  "nombre": "Ana López",
  "dni": "12345678A",
  "email": "ana@example.com",
  "telefono": "600111222",
  "importe": 120.5,
  "concepto": "Cuota atrasada",
 }
 {
  "id": 2,
  "nombre": "Lucía Gómez",
  "dni": "12345678B",
  "email": "lucia@example.com",
  "telefono": "600111223",
  "importe": 225.5,
```

```
  "concepto": "Alquiler",
 }
]
```


2.2 POST /api/morosos (añade un nuevo moroso a la LinkedList.)

Body (JSON): debe incluir al menos nombre. importe no puede ser negativo.

Lógica:

- Validar campos mínimos (p. ej., nombre no vacío; importe >= 0 si viene informado).
- Asignar id autoincremental (no aceptar id enviado).
- Añadir al final de la LinkedList.

Respuestas:

- 201 Created + objeto creado (con id).
- 400 Bad Request si validaciones fallan.

Ejemplo de respuesta (201):


```
{
 "id": 2,
 "nombre": "Luis Pérez",
 "dni": "23456789B",
 "email": "luis@example.com",
 "telefono": "600333444",
 "importe": 300.0,
 "concepto": "Servicio impagado",
}
```


2.3 DELETE /api/morosos/{id} (elimina el moroso cuyo id coincida (primer elemento encontrado).)

Lógica: Buscar en la LinkedList por id y si existe, eliminarlo.

Respuestas:

- 204 No Content si se elimina.
- 404 Not Found si no existe.

3. Qué entregar (aplicación)

- Proyecto ejecutable (IDE o mvn spring-boot:run). **(COMPRIMIDO)**
- Capturas con resultados del postman (en formato PDF).

ANEXOS

ESTRUCTURA DEL PROYECTO

```
morosos-app/
├─ pom.xml
├─ src/
│  ├─ main/
│  │  ├─ java/
│  │  │  └─ com/
│  │  │     └─ ejemplo/
│  │  │        └─ morosos/
│  │  │           ├─ MorososAppApplication.java
│  │  │           ├─ controller/
│  │  │           │  └─ MorosoController.java
│  │  │           └─ model/
│  │  │              └─ Moroso.java
│  │  └─ resources/
│  │     └─ application.properties
│  └─ test/
│     └─ java/
```

(SE ENTREGA EN EL COMPRIMIDO JUNTO AL DOCUMENTO)