Evaluación práctica (Django + DRF)

Objetivo

Demostrar comprensión de:

- **RESTful API** (recursos, verbos, códigos, stateless, versionado).
- **Arquitectura REST** por capas en un proyecto Django.
- Implementación mínima con **DRF**.

Entregables (3 archivos obligatorios)

1. **README.md** (1 página)
   - Recursos y URIs: `/api/v1/tasks/`, `/api/v1/tasks/{id}/`.
   - Verbos y códigos: qué hace cada endpoint y códigos esperados.
2. **Código Django** (carpeta del proyecto) con:
   - Modelo, serializer, viewset/urls funcionando.
   - Config DRF básica
3. **tests.md** con **5 comandos `curl`** que muestren: listar, crear, detalle, actualizar `done`, 404, o en su defecto capturas de pantalla de Postman con cada endpoint funcionando

Nota: Se evalúa que corra localmente y responda JSON.

Requisitos funcionales mínimos

Recurso principal

Tarea con campos:

- `id` (auto)
- `titulo` (string, requerido)
- `hecho` (boolean, por defecto `false`)
- `created_at` (auto)

Endpoints (DRF)

- `GET /api/v1/tareas/` → lista paginada.
- `POST /api/v1/tareas/` → crea.
- `GET /api/v1/tareas/{id}/` → detalle.
- `PATCH /api/v1/tareas/{id}/` → actualiza `title` o `done`.
- *`DELETE /api/v1/tareas/{id}/` → 204.*

Códigos HTTP esperados

- `200 OK` (GET/PATCH), `201 Created` (POST), `204 No Content`
- `400 Bad Request` (validación), `404 Not Found` (id inexistente).

Reglas REST que debes mencionar y explicar con tus palabras en README

- **Stateless** (sin sesiones para esta práctica).
- **JSON** (`Content-Type: application/json`).
- **Versionado** en la ruta: `/api/v1/...`.
- **Idempotencia**: `GET` no cambia estado; `PATCH` repetido deja mismo estado.

Diagrama de arquitectura (inclúyelo en README)

```
[Cliente (curl/SPA)]
        |
      HTTP/JSON
        |
[ API /api/v1 (DRF ViewSets/URLs) ]
        |
[ Lógica/Serializers (validación) ]
        |
[ Modelo Django (ORM) ]
        |
[ DB SQLite (local) ]
```

Una línea por capa describiendo su función.