

Guía API con FastAPI (sin base de datos)

Reglas comunes para **todos** los ejercicios:

- Debes implementar al menos: **1 endpoint con path params**, **1 con query params**, y **1 con JSON en el body** (request body).
 - No usar base de datos. Puedes usar estructuras en memoria (listas/dicts) si lo necesitas.
 - Validación con **Pydantic** (models) para el body.
 - Respuestas con códigos HTTP adecuados (200, 201, 400, 404, 422).
-

Ejercicio 1: Gestor de tareas (To-Do) en memoria

Contexto

Una mini API para gestionar tareas de una lista (sin persistencia).

Endpoints requeridos

1. `POST /tasks`

- **Body (JSON):** `{ "title": str, "description": str | null, "priority": int (1-5) }`
- Crea una tarea con `id` autogenerado y `completed=false`.

2. `GET /tasks/{task_id}`

- **Path param:** `task_id`
- Devuelve la tarea o `404`.

3. `GET /tasks`

- **Query params:**
 - `completed: bool | null`
 - `min_priority: int | null`
- Filtra tareas según parámetros.

4. `PATCH /tasks/{task_id}/complete`

- **Path param:** `task_id`
- Marca la tarea como completada.

Retos extra

- Validar que `priority` esté entre 1 y 5.
 - Soportar paginación simple: `skip` y `limit` (query params).
-

Ejercicio 2: Conversor de unidades (con historial en memoria)

Contexto

API que convierte valores entre unidades (temperatura, distancia, peso) y guarda un historial temporal en memoria.

Endpoints requeridos

1. `POST /convert`

- **Body (JSON):**

JSON

```
{ "category": "temperature|distance|weight", "from_unit": str, "to_unit": str, "value": float }
```

- Devuelve `{ "result": float, "formula": str }`.

2. `GET /history/{category}`

- **Path param:** `category`
- Devuelve conversiones previas de esa categoría.

3. `GET /history`

- **Query params:**
 - `category: str | null`
 - `min_value: float | null`
- Filtra historial.

Retos extra

- Manejar errores: unidades inválidas => `400` con mensaje claro.
 - Agregar `timestamp` a cada conversión.
-

Ejercicio 3: Validador de formulario de registro

Contexto

API que valida datos de registro (sin guardar nada), ideal para practicar validaciones.

Endpoints requeridos

1. `POST /register/validate`

- **Body (JSON):**
 - `username` (3–20)
 - `email` (formato email)
 - `password` (mínimo 8, al menos 1 número)
 - `age` (≥ 13)
- Devuelve:
 - Si válido: `{ "ok": true }`
 - Si inválido: `{ "ok": false, "errors": [...] }`

2. `GET /users/{username}/availability`

- **Path param:** `username`
- Simula disponibilidad comparando contra una lista en memoria.

3. `GET /password/rules`

- **Query params:**
 - `lang`: `"es" | "en"` (default `"es"`)
- Devuelve reglas de contraseña.

Retos extra

- Implementar validaciones con `@field_validator` (Pydantic).
 - Estandarizar el formato de error (schema propio).
-

Ejercicio 4: Catálogo de películas + recomendador simple

Contexto

API que maneja un catálogo en memoria y recomienda películas por criterios.

Endpoints requeridos

1. `POST /movies`

- **Body (JSON):**

JSON

```
{ "title": str, "genres": [str], "year": int, "rating": float (0-10) }
```

- Crea una película con `id`.

2. `GET /movies/{movie_id}`

- **Path param:** `movie_id`

3. `GET /movies`

- **Query params:**
 - `genre: str | null`
 - `min_rating: float | null`
 - `year: int | null`

4. `POST /movies/recommend`

- **Body (JSON):**

JSON

```
{ "preferred_genres": [str], "min_year": int | null, "max_results": int }
```

- Devuelve lista ordenada (por rating desc).

Retos extra

- Rechazar duplicados por `title+year` con `409 Conflict`.
- Respuesta paginada con `skip` y `limit`.

Ejercicio 5: Mini API de “carrito de compras” (sin pagos)

Contexto

API que simula un carrito con productos en memoria.

Endpoints requeridos

1. `POST /products`

- **Body (JSON):** { "name": str, "price": float (>0), "stock": int (>=0) }

2. `POST /cart/{cart_id}/items`

- **Path param:** `cart_id`
- **Body (JSON):** { "product_id": int, "quantity": int (>0) }

3. `GET /cart/{cart_id}`

- **Path param:** `cart_id`
- Devuelve items y total.

4. `GET /products`

- **Query params:**
 - `max_price`: float | null
 - `in_stock`: bool | null

Retos extra

- Validar stock disponible al agregar items.
 - Endpoint `DELETE /cart/{cart_id}/items/{product_id}` (path params múltiples).
-

Ejercicio 6: Analizador de texto (estadísticas + filtros)

Contexto

API que recibe texto y devuelve estadísticas (sin guardar), útil para practicar body + query params.

Endpoints requeridos

1. `POST /text/analyze`

- **Body (JSON):**

JSON

```
{ "text": str, "language": "es|en" }
```

- Devuelve:
 - número de palabras
 - número de caracteres
 - top 5 palabras más repetidas

2. GET /text/{word}/frequency

- **Path param:** `word`
- **Query params:**
 - `text: str` (texto pasado por query param)
 - `case_sensitive: bool` (default false)
- Devuelve frecuencia de `word` en `text`.

3. POST /text/censor

- **Body (JSON):**

JSON

```
{ "text": str, "banned": [str], "mask": "*" }
```

- Devuelve texto censurado.

Retos extra

- Ignorar “stopwords” con query param: `ignore_stopwords=true`.
- Devolver resultados con un modelo de respuesta (Pydantic) para consistencia.