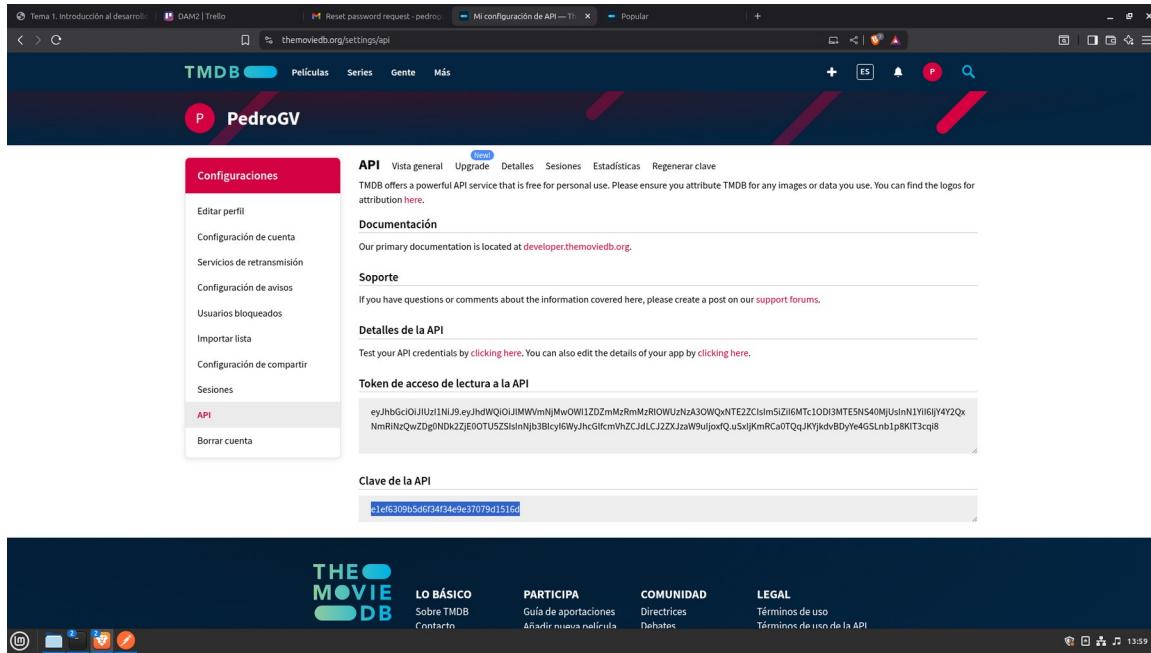


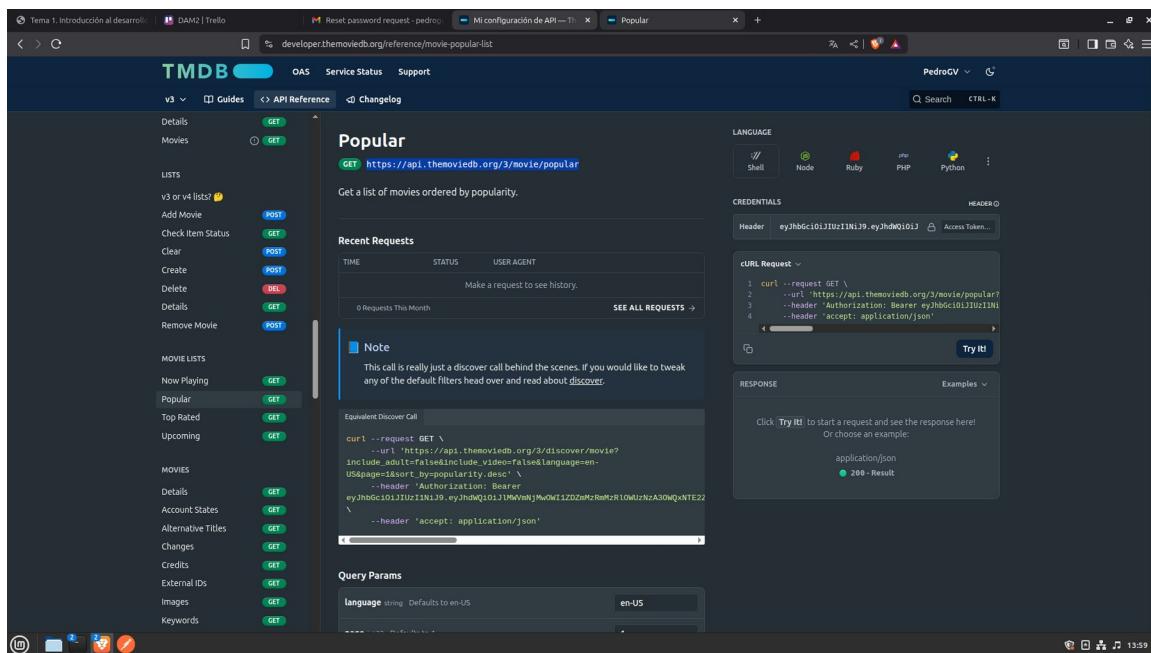
4. Consulta una API de películas con una API Key

a. Regístrate en la página web de The Movie Database (TMDB). Una vez iniciada la sesión, accede a Ajustes en tu perfil y después a la sección API del menú lateral izquierdo para obtener tu API Key.



The screenshot shows the TMDB API configuration page. On the left, there's a sidebar with 'Configuraciones' and 'API' sections. The 'API' section is expanded, showing details like 'Vista general', 'Upgrade', 'Detalles', 'Sesiones', 'Estadísticas', and 'Regenerar clave'. Below this, there's a large text area containing a long API key: eyJhbGciOiJIUzI1NiJ9eyJhdWQiOiJMMWVmNjMwOWI1ZDmMzRmMzRiOWLzNzA3OWQxNTE2ZCisimSiZiG6MtclODI3MTESNS40MjUsInN1YiIjY4Y2QxNmRiNzQwZDg0NDk2ZjE0OTU5ZlslnNjb3BlcyEyWyJhcGlfcmVhZCJdCJ2ZXJzaW9uJoxfQ.uSxlijKmRCa0TQqJKYkdvBdyYe4G5Lnbp8KIT3cq8. At the bottom of this area, there's a 'Clave de la API' input field containing the same key. The TMDB logo is at the top, and the navigation bar includes links for 'LO BÁSICO', 'PARTICIPA', 'COMUNIDAD', and 'LEGAL'.

b. Consulta la referencia de la API y usa Postman para realizar una solicitud HTTP GET que obtenga la lista de películas populares.



The screenshot shows the TMDB API reference page for the 'Popular' endpoint in Postman. The left sidebar lists various endpoints like 'Details', 'Movies', 'Lists', 'MOVIE LISTS', 'MOVIES', etc. The 'Popular' endpoint is selected. The main panel shows the URL <https://api.themoviedb.org/3/movie/popular> and a note: 'Get a list of movies ordered by popularity.' It also shows a 'Recent Requests' section with a history of 0 requests this month. On the right, there are sections for 'LANGUAGE' (Shell, Node, Ruby, PHP, Python), 'CREDENTIALS' (Header: eyJhbGciOiJIUzI1NiJ9eyJhdWQiOiJMMWVmNjMwOWI1ZDmMzRmMzRiOWLzNzA3OWQxNTE2ZCisimSiZiG6MtclODI3MTESNS40MjUsInN1YiIjY4Y2QxNmRiNzQwZDg0NDk2ZjE0OTU5ZlslnNjb3BlcyEyWyJhcGlfcmVhZCJdCJ2ZXJzaW9uJoxfQ.uSxlijKmRCa0TQqJKYkdvBdyYe4G5Lnbp8KIT3cq8.), 'cURL Request', and 'RESPONSE' (with a placeholder message: 'Click Try It! to start a request and see the response here! Or choose an example: application/json 200 - Result').

```
GET https://api.themoviedb.org/3/movie/popular?api\_key=TU\_API\_KEY&language=es-ES&page=1
```

c. Busca en Postman todas las películas del año 2024.

```
GET https://api.themoviedb.org/3/discover/movie?  
api\_key=TU\_API\_KEY&language=es-ES&primary\_release\_year=2024
```

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Edit', 'View', 'Help', 'File', 'Home', 'Workspaces', and 'Explore'. Below that is a search bar with 'Search Postman'. The main area has tabs for 'History' (selected), 'New', and 'Import'. A URL 'https://api.themoviedb.org/3/movie/popular?api_key=e1ef6309b5d6f34e37079d1516d' is entered in the address bar. The status bar at the bottom right shows 'Status: 200 OK', 'Time: 282 ms', 'Size: 5.6 KB', and 'Save Response'. The response body is displayed in a JSON editor with various fields like 'adult', 'backdrop_path', 'genre_ids', 'id', 'original_language', 'original_title', 'overview', 'popularity', 'poster_path', 'release_date', 'title', 'video', 'vote_average', and 'vote_count'. There are buttons for 'Send', 'Save', and 'Copy' on the right side of the JSON editor.

d. Busca las películas de la saga de Star Wars.

GET [https://api.themoviedb.org/3/search/collection?
api_key=TU_API_KEY&language=es-ES&query=Star%20Wars](https://api.themoviedb.org/3/search/collection?api_key=TU_API_KEY&language=es-ES&query=Star%20Wars)

e. Busca las películas en las que haya actuado Cillian Murphy.

GET [https://api.themoviedb.org/3/search/person?
api_key=TU_API_KEY&language=es-ES&query=Cillian%20Murphy](https://api.themoviedb.org/3/search/person?api_key=TU_API_KEY&language=es-ES&query=Cillian%20Murphy)

5. Crea una API de pruebas mediante json-server

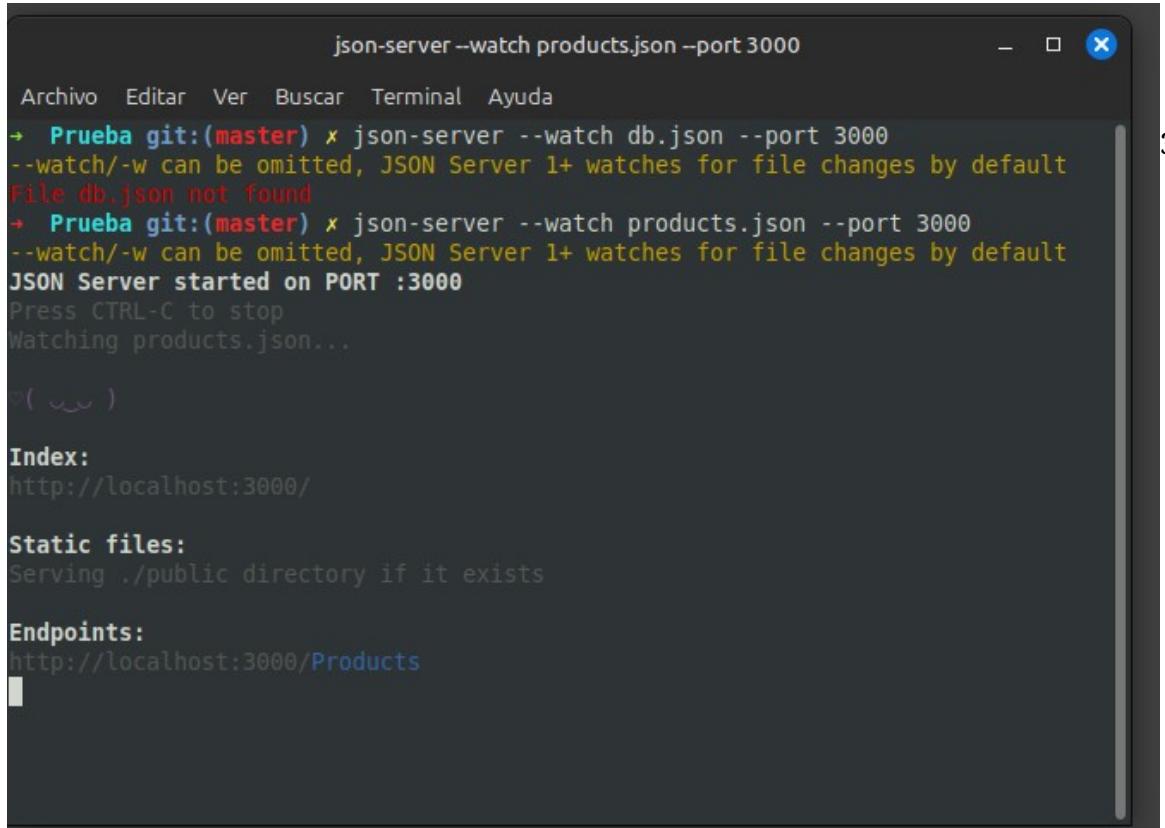
Para crear una API de pruebas con json-server:

1. Crea un fichero JSON con los datos de productos de una tienda de informática (por ejemplo, db.json).

```
{  
  "Products": [  
    {  
      "id": "1",  
      "name": "Monitor de 24 pulgadas",  
      "pvp": 80.51,  
      "description": "Monitor led"  
    },  
    {  
      "id": "2",  
      "name": "Monitor de 24 pulgadas",  
      "pvp": 80.51,  
      "description": "Monitor led"  
    },  
    {  
      "id": "3",  
      "name": "Monitor de 24 pulgadas",  
      "pvp": 80.51,  
      "description": "Monitor led"  
    }  
  ]  
}
```

2. Ejecuta el servidor con el comando:

```
json-server --watch db.json --port 3000
```



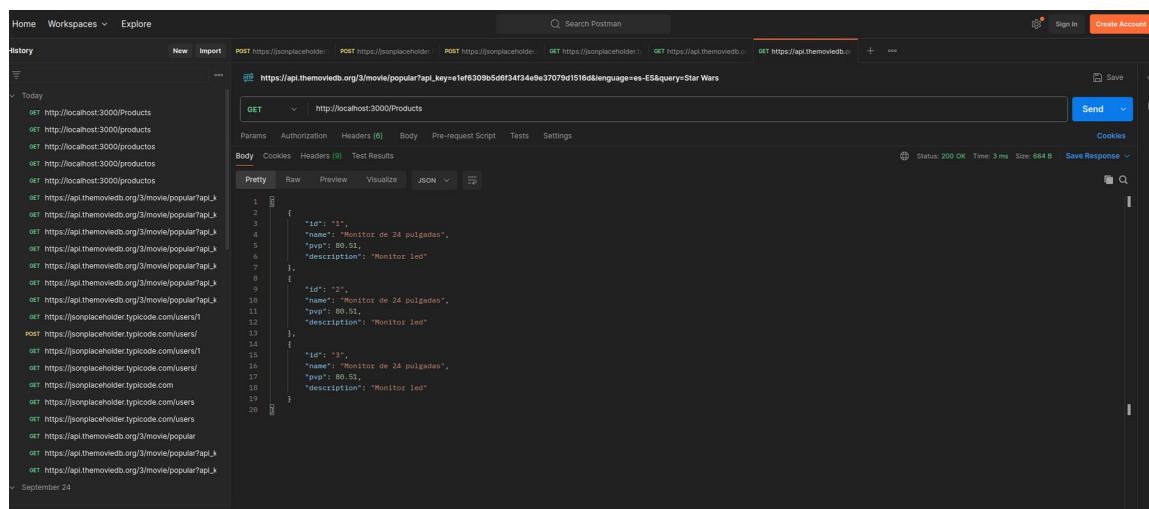
```
json-server --watch products.json --port 3000
Archivo Editar Ver Buscar Terminal Ayuda
→ Prueba git:(master) x json-server --watch db.json --port 3000
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
File db.json not found
→ Prueba git:(master) x json-server --watch products.json --port 3000
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching products.json...
Index:
http://localhost:3000/
Static files:
Serving ./public directory if it exists
Endpoints:
http://localhost:3000/Products
```

3.

Usa Postman para realizar las siguientes solicitudes:

a. Obtener todos los productos:

GET <http://localhost:3000/Products>

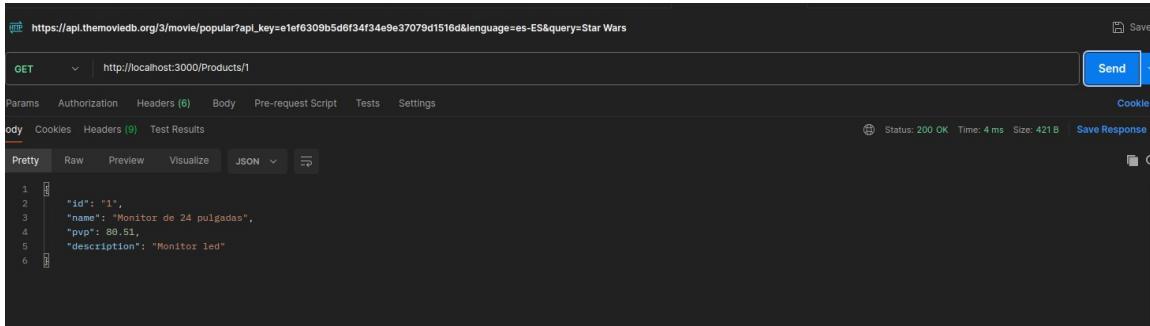


The screenshot shows a Postman interface with a history panel on the left and a main workspace on the right. In the workspace, a GET request is defined for the URL <http://localhost:3000/Products>. The request is set to 'Pretty' JSON format. The response body contains a list of product objects, each with an id, name, pvp, and description. The first few objects are:

```
1 [
2   {
3     "id": "1",
4     "name": "Monitor de 24 pulgadas",
5     "pvp": 88.51,
6     "description": "Monitor led"
7   },
8   {
9     "id": "2",
10    "name": "Monitor de 24 pulgadas",
11    "pvp": 88.51,
12    "description": "Monitor led"
13  },
14  {
15    "id": "3",
16    "name": "Monitor de 24 pulgadas",
17    "pvp": 88.51,
18    "description": "Monitor led"
19  }
20 ]
```

b. Obtener un producto por su id:

GET <http://localhost:3000/Products/1>

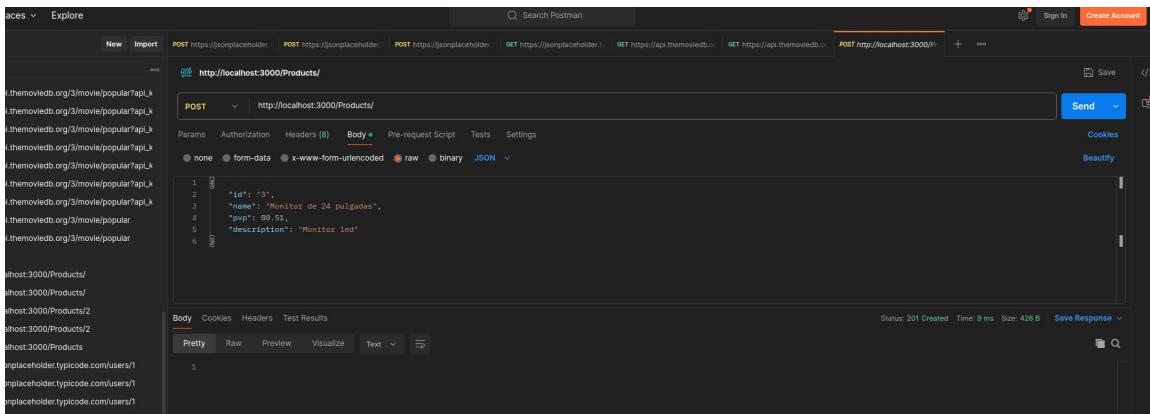


```
1
2   "id": "1",
3   "name": "Monitor de 24 pulgadas",
4   "pvp": 80.51,
5   "description": "Monitor led"
6
```

c. Crear un nuevo producto:

POST <http://localhost:3000/Products>
Content-Type: application/json

```
{
  "nombre": "Impresora",
  "precio": 80
}
```



```
1
2   "id": "3",
3   "name": "Monitor de 24 pulgadas",
4   "pvp": 80.51,
5   "description": "Monitor led"
6
```

d. Actualizar un producto existente con el método PUT:

PUT <http://localhost:3000/Products/1>

Content-Type: application/json

```
{  
  "id": 1,  
  "nombre": "Teclado mecánico",  
  "precio": 50  
}
```

e. Actualizar un producto existente con el método PATCH:

PATCH <http://localhost:3000/Products/2>

Content-Type: application/json

```
{  
  "precio": 18  
}
```

f. Eliminar un producto:

DELETE <http://localhost:3000/Products/3>