

PING API

Endpoint: GET /api/ping

Function: Check API server is running.

The screenshot shows the Bruno API testing tool interface. On the left, there's a sidebar with various API endpoints listed under 'Collections'. In the center, a request configuration panel shows a 'GET' request to 'http://127.0.0.1:8000/api/ping'. The 'Body' tab is selected, and it displays the response body: a JSON object with a single key 'message' containing the value 'pong'. The status bar at the bottom right indicates a 200 OK response with 478ms latency and 18B body size.

Get Documents

Endpoint:

GET /api/documents

Function:

Retrieve list of documents.

The screenshot shows the Bruno API testing tool interface. On the left, there's a sidebar with various API endpoints listed under 'Collections'. In the center, a request configuration panel shows a 'GET' request to 'http://127.0.0.1:8000/api/documents'. The 'Body' tab is selected, and it displays the response body: a JSON array containing one document object. The object has fields like id, created_at, updated_at, title, description, file_name, and file_path. The status bar at the bottom right indicates a 200 OK response with 280ms latency and 624B body size.

Create Document

Endpoint: POST /api/documents

Function: Upload new document.

The screenshot shows the Postman interface for creating a new document. The left sidebar lists various API endpoints. The main workspace shows a POST request to `http://127.0.0.1:8000/api/documents`. The 'Body' tab is selected, showing a 'Multipart Form' structure with fields: title (Test document), description (test file), file (selected file M11.pdf), category_id (1), department_id (1), and access_level (public). The 'Response' tab displays the JSON response of the created document:

```
1.  {
2.   "title": "Test document",
3.   "description": null,
4.   "file_name": "M11.pdf",
5.   "file_path": "documents/ph4wGyJVAI0mLvccxMzsWsx7jH7oadiwnQKbZMXj.pdf",
6.   "file_type": "File",
7.   "file_size": 1363137,
8.   "category_id": "1",
9.   "department_id": "1",
10.  "uploaded_by": null,
11.  "access_level": "public",
12.  "download_count": 0,
13.  "updated_at": "2026-02-09T15:18:44.000000Z",
14.  "created_at": "2026-02-09T15:18:44.000000Z",
15.  "id": 3
16. }
```

Update Document

Endpoint: POST /api/documents/{id}

Function: Update document information.

The screenshot shows the Postman interface for updating a document. The left sidebar lists various API endpoints. The main workspace shows a GET request to `http://127.0.0.1:8000/api/documents`. The 'Body' tab is selected, showing 'No Body'. The 'Response' tab displays the JSON response of the updated document:

```
1.  [
2.   {
3.     "id": 2,
4.     "created_at": "2026-02-09T15:17:10.000000Z",
5.     "updated_at": "2026-02-09T16:14:08.000000Z",
6.     "title": "Test document",
7.     "description": null,
8.     "file_name": "M11.pdf",
9.     "file_path": "documents/dDqtvrCh4va9EW8gwu9huxzd0rbk94VXInvrWI.pdf",
10.    "file_type": "File",
11.    "file_size": 1363137,
12.    "category_id": 1,
13.    "department_id": 1,
14.    "uploaded_by": null,
15.    "access_level": "public",
16.    "download_count": 1,
17.    "category": {
18.      "id": 1,
19.      "name": "Policy",
20.      "created_at": "2026-02-03T16:29:08.000000Z",
21.      "updated_at": "2026-02-03T16:29:08.000000Z"
22.    }
23.  },
```

Delete Document

Endpoint: DELETE /api/documents/{id}

Function: Delete selected document.

The screenshot shows the Bruno API testing tool interface. The left sidebar lists various API endpoints. The main area shows a DELETE request to `http://127.0.0.1:8000/api/documents/2`. The 'Body' tab is selected, showing 'No Body'. The 'Response' tab displays the JSON response:

```
1 {  
2   "message": "Document deleted successfully"  
3 }
```

. The status bar at the bottom indicates `200 OK 710ms 43B`.

Download Document

Endpoint: GET /api/documents/{id}/download

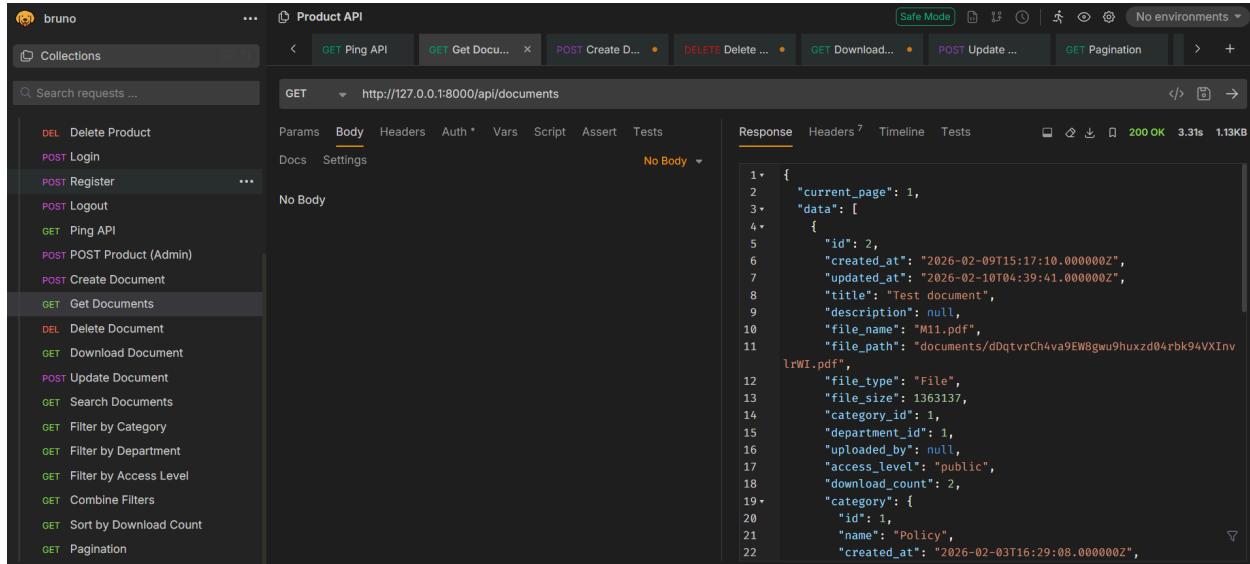
Function: Download document file.

The screenshot shows the Bruno API testing tool interface. The left sidebar lists various API endpoints. The main area shows a GET request to `http://127.0.0.1:8000/api/documents/2/download`. The 'Params' tab is selected, showing 'Query' and 'Path' sections. The 'Response' tab displays a PDF file titled 'Wawasan 2020' which is the document being downloaded. The status bar at the bottom indicates `200 OK 1.3s 1331.19KB PDF Raw`.

Download Count

Endpoint: GET /api/documents

Function: Verify download_count increases after download.



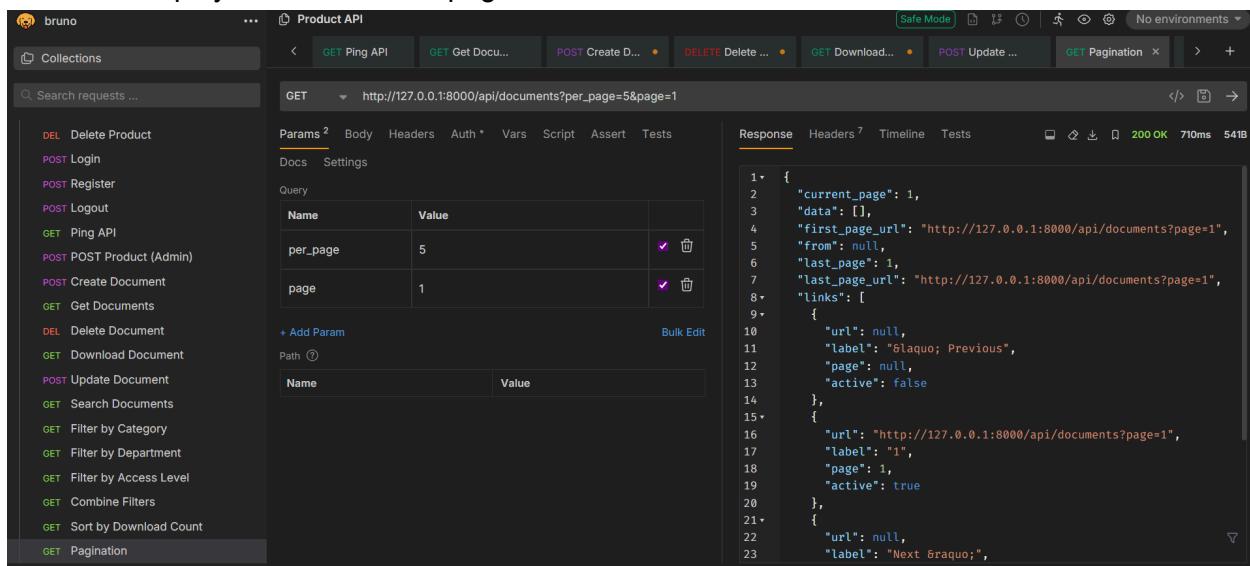
The screenshot shows the Postman interface with a successful API call. The URL is `http://127.0.0.1:8000/api/documents`. The response body is a JSON object representing a document:

```
1. {
2.     "current_page": 1,
3.     "data": [
4.         {
5.             "id": 2,
6.             "created_at": "2026-02-09T15:17:10.00000Z",
7.             "updated_at": "2026-02-10T04:39:41.00000Z",
8.             "title": "Test document",
9.             "description": null,
10.            "file_name": "M11.pdf",
11.            "file_path": "documents/dDqtvCh4va9EW8gwu9hxzd04rbk94VXInvlrWI.pdf",
12.            "file_type": "file",
13.            "file_size": 1363137,
14.            "category_id": 1,
15.            "department_id": 1,
16.            "uploaded_by": null,
17.            "access_level": "public",
18.            "download_count": 2,
19.            "category": {
20.                "id": 1,
21.                "name": "Policy",
22.                "created_at": "2026-02-09T16:29:08.00000Z"
23.            }
24.        }
25.    ],
26.    "last_page": 1,
27.    "next_page": null,
28.    "path": "/api/documents",
29.    "per_page": 5,
30.    "previous_page": null,
31.    "total": 1
32. }
```

Pagination

Endpoint: GET /api/documents?per_page=5&page=1

Function: Display documents with pagination.



The screenshot shows the Postman interface with a successful API call. The URL is `http://127.0.0.1:8000/api/documents?per_page=5&page=1`. The response body is a JSON object representing a paginated list of documents:

```
1. {
2.     "current_page": 1,
3.     "data": [],
4.     "first_page_url": "http://127.0.0.1:8000/api/documents?page=1",
5.     "from": null,
6.     "last_page": 1,
7.     "last_page_url": "http://127.0.0.1:8000/api/documents?page=1",
8.     "links": [
9.         {
10.             "url": null,
11.             "label": "\u25c0; Previous",
12.             "page": null,
13.             "active": false
14.         },
15.         {
16.             "url": "http://127.0.0.1:8000/api/documents?page=1",
17.             "label": "1",
18.             "page": 1,
19.             "active": true
20.         },
21.         {
22.             "url": null,
23.             "label": "Next \u25c1;",
24.         }
25.     ],
26.     "last_page": 1,
27.     "next_page": null,
28.     "path": "/api/documents",
29.     "per_page": 5,
30.     "previous_page": null,
31.     "total": 0
32. }
```

Backend

```
❖ PS C:\Users\User\final-demo\employee-doc-portal-api> php artisan serve
  INFO Server running on [http://127.0.0.1:8000].
  Press Ctrl+C to stop the server
```

Frontend

```
> ▾ TERMINAL
⚠ PS C:\Users\User\final-demo\employee-doc-portal-api\frontend> npm run dev
  VITE v7.3.1 ready in 641 ms
  → Local: http://localhost:5173/
  → Network: use --host to expose
  → press h + enter to show help
```

Testing

```
> ▾ TERMINAL
⚠ Duration: 13.99s
● PS C:\Users\User\final-demo\employee-doc-portal-api> php artisan test
  PASS Tests\Unit\ExampleTest
    ✓ that true is true 0.01s
  PASS Tests\Feature\AuthApiTest
    ✓ user can register 0.69s
    ✓ user can login 0.04s
    ✓ user can logout 0.04s
  PASS Tests\Feature\DocumentApiTest
    ✓ manager can upload document 0.07s
    ✓ employee cannot upload document 0.04s
    ✓ user can search documents 0.05s
    ✓ user can filter documents by category 0.04s
    ✓ admin can delete any document 0.04s
  PASS Tests\Feature\ExampleTest
    ✓ the application returns a successful response 0.05s
  Tests: 10 passed (27 assertions)
  Duration: 1.26s
  ❖ PS C:\Users\User\final-demo\employee-doc-portal-api>
```