

WORKFLOW DEMONSTRATION

Example 1

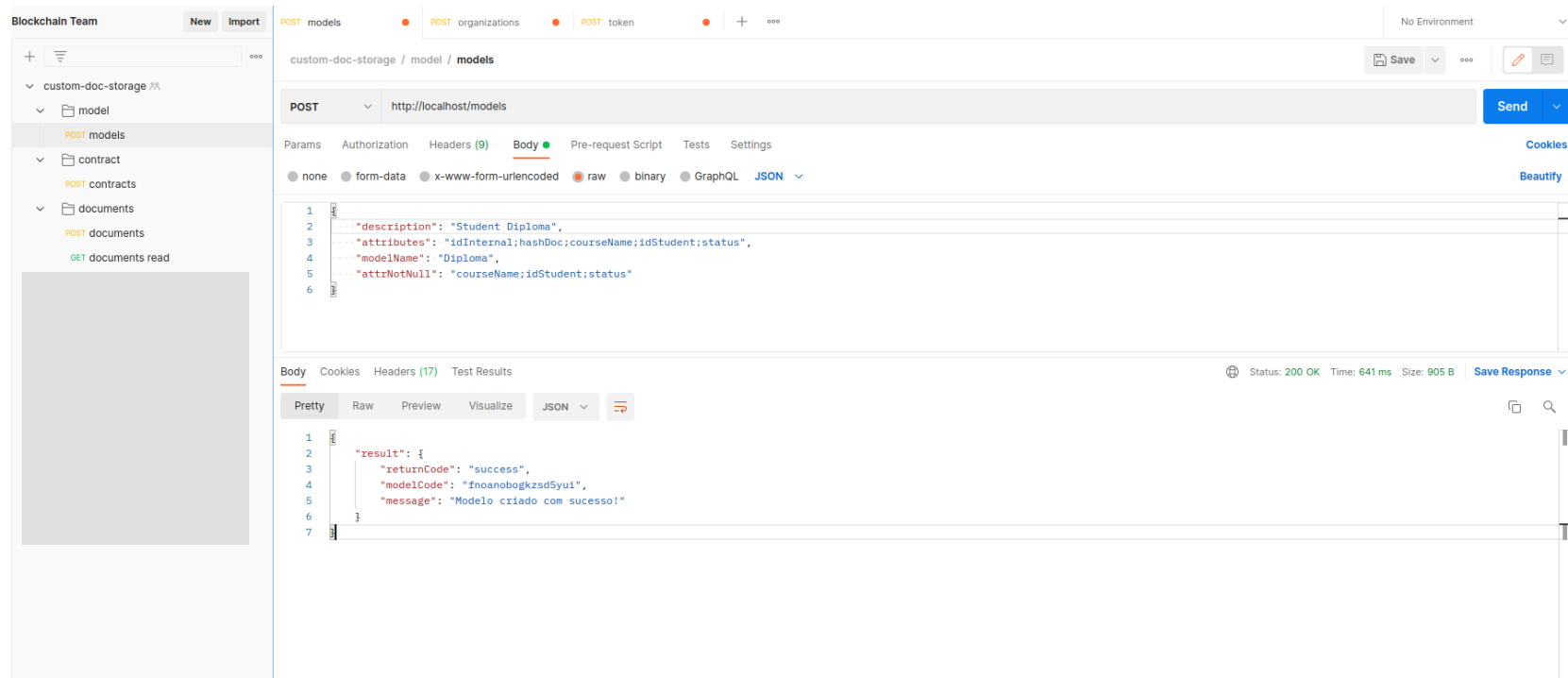
Client Application

Undergraduate Course System

Document

Student Diploma

Step 1 - Create Template



Step 2 - Deploy Contract (Polygon Blockchain)

The screenshot displays a REST client interface for a project named "Blockchain Team". The left sidebar shows a file explorer with the following structure:

- custom-doc-storage /
 - model
 - POST models
 - contract
 - POST contracts
 - documents
 - POST documents
 - GET documents read

The main panel shows a POST request to `http://localhost/contracts`. The request body is a JSON object:

```
{  "blockchain": "3",  "modelCode": "fnoanobogkzsd5yui"}
```

The response is displayed in the "Body" tab, showing a successful deployment with the following JSON:

```
{  "result": {    "returnCode": "success",    "contractDeployed": {      "idContract": "fnoanobogkzsdnks3",      "address": "0x97991F8f4cB82D0eab44c351732232FC5Ec38dfc",      "ABI": [        {          "constant": true,          "inputs": [            {              "name": "_idDocument",              "type": "string"            }          ],          "name": "readDocument",          "outputs": [            {              "name": "",              "type": "string[]"            }          ]        }      ]    }  } }
```

The status bar at the bottom indicates a successful response with status 200 OK, a time of 7.40 s, and a size of 7.21 KB.

Step 3 - Insert Document

The screenshot displays a REST client interface for a project named "Blockchain Team". The left sidebar shows a file tree with the following structure:

- custom-doc-storage
 - model
 - POST models
 - contract
 - POST contracts
 - documents
 - POST documents
 - GET documents read

The main panel shows a POST request to `http://localhost/documents`. The request body is a JSON object:

```
{  "idContract": "fnoanobogkzsdnks3",  "value": ["idInternal001", "hashMedia001", "Software Engineering", "000001", "APPROVED"]}
```

The response is displayed in the "Body" tab, showing a successful status (200 OK) and the following JSON:

```
{  "result": {    "returnCode": "success",    "blockchain": "3",    "idDocument": "fnoanobogkzsdos61",    "blockNumber": 25060195,    "txId": "0x7fdb83ff4e3c81982757fd3285b8825fa018410dc6dd3b798abd901a4108cd"  }}
```

The interface includes various tabs for request configuration (Params, Authorization, Headers, Body, Pre-request Script, Tests, Settings) and response viewing (Body, Cookies, Headers, Test Results). The response status is 200 OK, with a time of 9.74 s and a size of 983 B.

Step 4 - Read Document

The screenshot displays a REST client interface for a project named "Blockchain Team". The left sidebar shows a file tree with the following structure:

- custom-doc-storage
 - model
 - POST models
 - contract
 - POST contracts
 - documents
 - POST documents
 - GET documents read

The main panel shows a selected endpoint: `custom-doc-storage / documents / documents read`. The request is a **POST** to `http://localhost/documents_read`. The request body is in **JSON** format:

```
1 {
2   "idContract": "fnoanobogkzsdnks3",
3   "idDocument": "fnoanobogkzsdos61"
4 }
```

The response is also in **JSON** format, with a status of 200 OK, time of 867 ms, and size of 1.09 KB. The response body is:

```
1 {
2   "result": {
3     "returnCode": "success",
4     "blockchain": "3",
5     "document": {
6       "idInternal": "idInternal001",
7       "hashDoc": "hashMedia001",
8       "courseName": "Software Engineering",
9       "idStudent": "000001",
10      "status": "APPROVED"
11    },
12    "transactionId": "0x7fdb83ff4e3c81982757fd3285b8825fa018410dc6dd3b798abd901a4108cd",
13    "timestamp": "2022-02-18T12:18:33.000Z"
14  }
15 }
```

Example 2

Client Application

System for Generating a Copyright Certificate for Artworks

Document

ArtWork Certificate

Step 1 - Create Template

The screenshot displays a REST client interface for a team named "Blockchain Team". The left sidebar shows a project structure with folders for "custom-doc-storage", "model", "contract", and "documents". The "POST models" endpoint is selected. The main panel shows a POST request to "http://localhost/models" with a JSON body. The response is a 200 OK status with a JSON body indicating success.

Request:

```
POST http://localhost/models
{
  "description": "Artwork Certificate in General",
  "attributes": "idInternal;hashDoc;idArtwork;idAuthor;description;artworkType",
  "modelName": "Artwork Certificate",
  "attrNotNull": "idArtwork;idAuthor;description;artworkType"
}
```

Response:

```
{
  "result": {
    "returnCode": "success",
    "modelCode": "fnoanobgkzse8ux1",
    "message": "Modelo criado com sucesso!"
  }
}
```

Step 2 - Deploy Contract (Ethereum Blockchain)

The screenshot displays a REST client interface for a team named "Blockchain Team". The left sidebar shows a project structure with folders for "model", "contract", "documents", and "contracts". The "contracts" folder is selected, showing a "POST contracts" endpoint. The main panel shows the details of this endpoint, including the URL "http://localhost/contracts" and the request body in JSON format:

```
{  "blockchain": "0",  "modelCode": "fnoanobgkzse8ux1"}
```

The response is displayed in the bottom panel, showing a status of 200 OK, a time of 58.69 s, and a size of 7.21 KB. The response body is in JSON format:

```
{  "result": {    "returnValue": "success",    "contractDeployed": {      "idContract": "fnoanobgkzsearch",      "address": "0xaF4c91Bb8b5F6f91EE01009a42844F3BBeDDDeb5",      "ABI": [        {          "constant": true,          "inputs": [            {              "name": "_idDocument",              "type": "string"            ]          },          {            "name": "readDocument",            "outputs": [              {                "name": "",                "type": "string[]"              ]            }          ]        ]      }    }  }
```

Step 3 - Insert Document

Blockchain Team New Import POST models POST organizations POST token POST contracts POST documents POST documents read + ... No Environment

custom-doc-storage / documents / documents Save

POST http://localhost/documents Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "idContract": "fnoanobogkzsearch",
3   "value": ["idInternal002", "698dc19d489c4e4db73e28a713eab07b", "000001", "000002", "A digital drawing of the Eiffel Tower landscape.", "Digital Drawing"]
4 }
```

Body Cookies Headers (17) Test Results Status: 200 OK Time: 19.82 s Size: 983 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": {
3     "returnCode": "success",
4     "blockchain": "0",
5     "idDocument": "fnoanobogkzsegfkt",
6     "blockNumber": 10189492,
7     "txId": "0xd9ce8c45813668638dbc3b338af5e376c940034543f6df597476b055b480843d"
8   }
9 }
```

Step 4 - Read Document

The screenshot displays a REST client interface for a project named "Blockchain Team". The breadcrumb path is "custom-doc-storage / documents / documents read". The selected endpoint is a POST request to "http://localhost/documents_read".

Request Details:

- Method: POST
- URL: http://localhost/documents_read
- Body (JSON):

```
1 {
2   "idContract": "fnoanobogkzsearch",
3   "idDocument": "fnoanobogkzsegfkt"
4 }
```

Response Details:

- Status: 200 OK
- Time: 937 ms
- Size: 1.17 KB
- Body (JSON):

```
1 {
2   "result": {
3     "returnCode": "success",
4     "blockchain": "0",
5     "document": {
6       "idInternal": "idInternal002",
7       "hashDoc": "698dc19d489c4e4db73e28a713eab07b",
8       "idArtwork": "000001",
9       "idAuthor": "000002",
10      "description": "A digital drawing of the Eiffel Tower landscape.",
11      "artworkType": "Digital Drawing"
12    },
13    "transactionId": "0xd9ce8c45813668638dbc3b338af5e376c940034543f6df597476b055b480843d",
14    "timestamp": "2022-02-18T12:40:13.000Z"
15  }
16 }
```