

API calls:

POST /plateau/create

On start, we define Plateau with dimensions width and height (Note: If plateau is already created, this call will create new Plateau and restart game).

Request: text with space-separated values for width and height (max 1000)

Response: matrix

The screenshot shows an API client interface with the following components:

- Method and URL:** POST http://localhost:8080/plateau/create
- Params:** none
- Authorization:** none
- Headers (9):** none
- Body:** 3 3
- Pre-request Script:** none
- Tests:** none
- Settings:** none

The response is displayed in the **Body** tab, showing a JSON object:

```
{
  "plateau": [
    {
      "x": 0,
      "y": 0,
      "rover": null,
      "visited": false
    },
    {
      "x": 0,
      "y": 1,
      "rover": null,
      "visited": false
    }
  ]
}
```

GET /plateau/read

Response: current plateau matrix

POST /rover/create

Request: Space-separated command. First is X-assis, second is Y-assis and third is Orientation.

Response: Rover details

You can create multiple rovers on different positions (If we try to deploy rover on occupied position Exception is thrown)

The screenshot shows a REST client interface with the following components:

- Method and URL:** POST `http://localhost:8080/rover/create`
- Tabs:** Params, Authorization, Headers (9), **Body** (selected), Pre-request Script, Tests, Settings.
- Body Type:** none, form-data, x-www-form-urlencoded, **raw** (selected), binary, GraphQL, Text.
- Body Content:**

```
1 1 1 N
```
- Response Tabs:** **Body** (selected), Cookies, Headers (5), Test Results.
- Response Format:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh icon.
- Response Content:**

```
{
  "x": 1,
  "y": 1,
  "id": 2,
  "orientation": "NORTH"
}
```

GET /plateau/print

Response: Plateau where rovers are represented by their orientation, and free fields are marked as '- '.

GET

▼

http://localhost:8080/plateau/print

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (5)

Test Results


Pretty

Raw

Preview

Visualize

Text ▼



1

[-][N][N]

2

[-][N][-]

3

[-][-][-]

4

POST /rover/move

Request: Id of rover, Command string (L for orientating left, R for right, and M for make a move)

POST

http://localhost:8080/rover/move

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

Text ▼

1

0 LLM

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼



1

{

2

"x": 1,

3

"y": 2,

4

"id": 0,

5

"orientation": "SOUTH"

6

}

After move is done, here is plateau with moved rover:

GET

▼

http://localhost:8080/plateau/print

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (5)

Test Results


Pretty

Raw

Preview

Visualize

Text ▼



```
1 [{"N"}]
2 [{"N"}]
3 [{"N"}]
4 [{"N"}]
```

GET /rover/printRoute

Request: Id of rover to move to specific field, field x coordinate, field y coordinate

Response: Preview of plateau matrix, where S is start field, D is destination field, and * is **shortest route**

GET http://localhost:8080/rover/printRoute

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL Text ▼

```
1 0 0 0
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text ▼ 

```
1 [{"D"}][ ][-]
2 [{"*"}][ ][S]
3 [{"*"}][*][*]
4
```

GET /rover/getDirections

Request: Id of rover to move to specific field, field x coordinate, field y coordinate

Response: Command string, which we can use for moving rover via **rover/move**

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method `GET` and URL `http://localhost:8080/rover/getDirections`.
- Request Body:** The `Body` tab is selected, showing a single line of text: `1 0 0 0`.
- Response Body:** The `Body` tab is selected, showing a single line of text: `1 MRMMRMM`. The response is displayed in a "Text" format.

In project, there is a test **getDirections_happyPath()** which can serve as a mock documentation of how the game should be used.