API calls:

POST /plateau/create

On start, we define Plateau with dimensions widht 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

POST http://localhost:8080/plateau/create

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

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

```
3 3
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


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

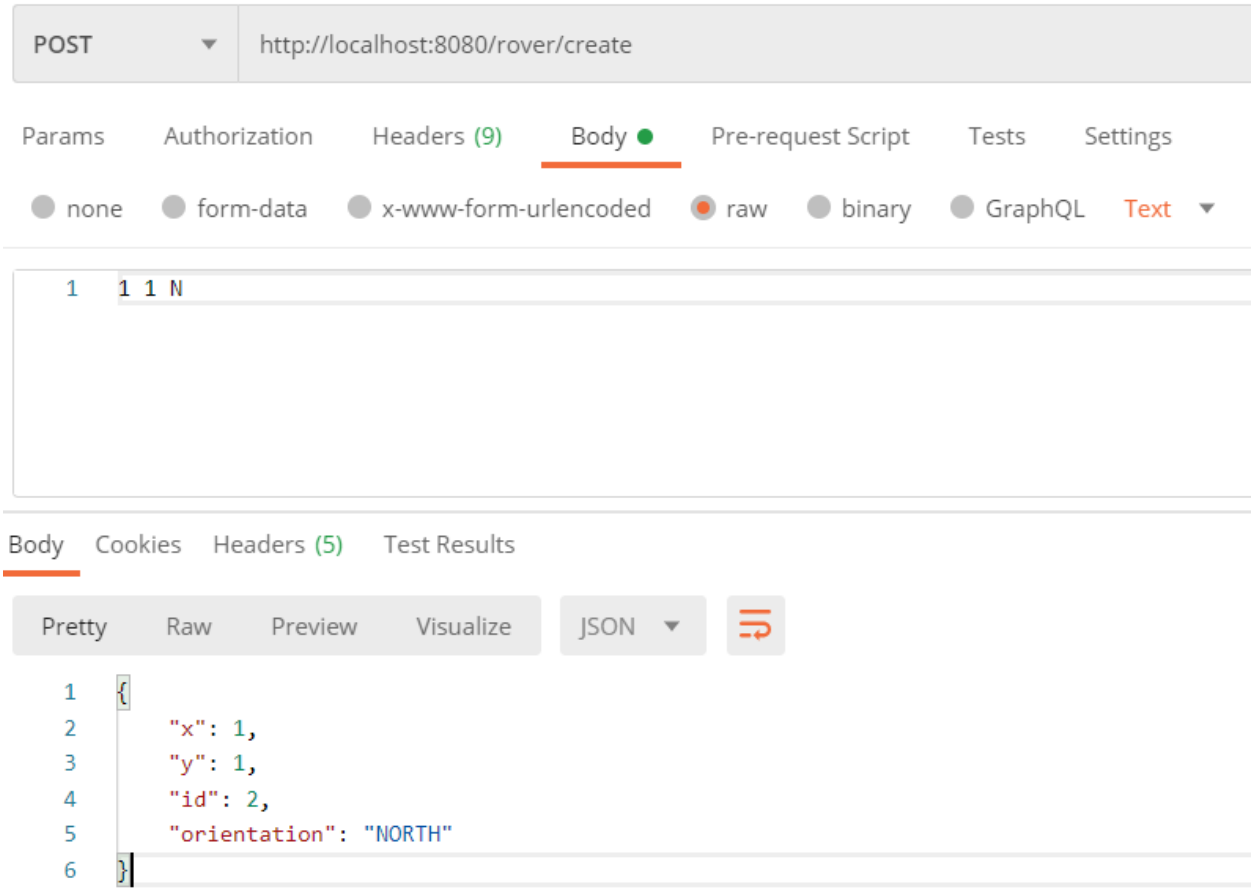
GET /plateau/read

Response: current plateau matrix

POST /rover/create

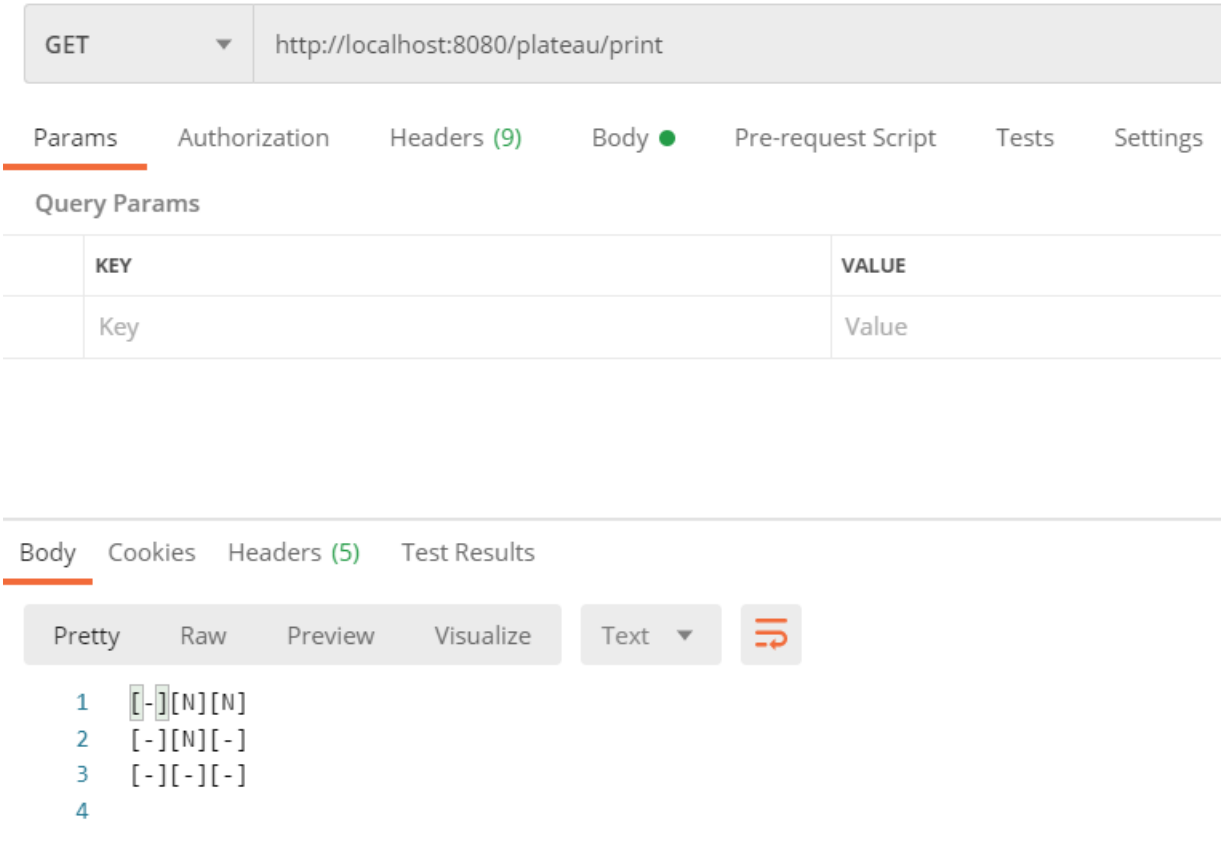
Request: Space-separated command. First is X-assis, second is Y-assis and third is Orientation.
Response: Rover details

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



GET /plateau/print

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



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

```
0 LLM
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "x": 1,
    "y": 2,
    "id": 0,
    "orientation": "SOUTH"
}
```

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

```
[-][N][-]
[-][N][S]
[-][-][-]

```

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

```
0 0 0
```

Body Cookies Headers (5) Test Results

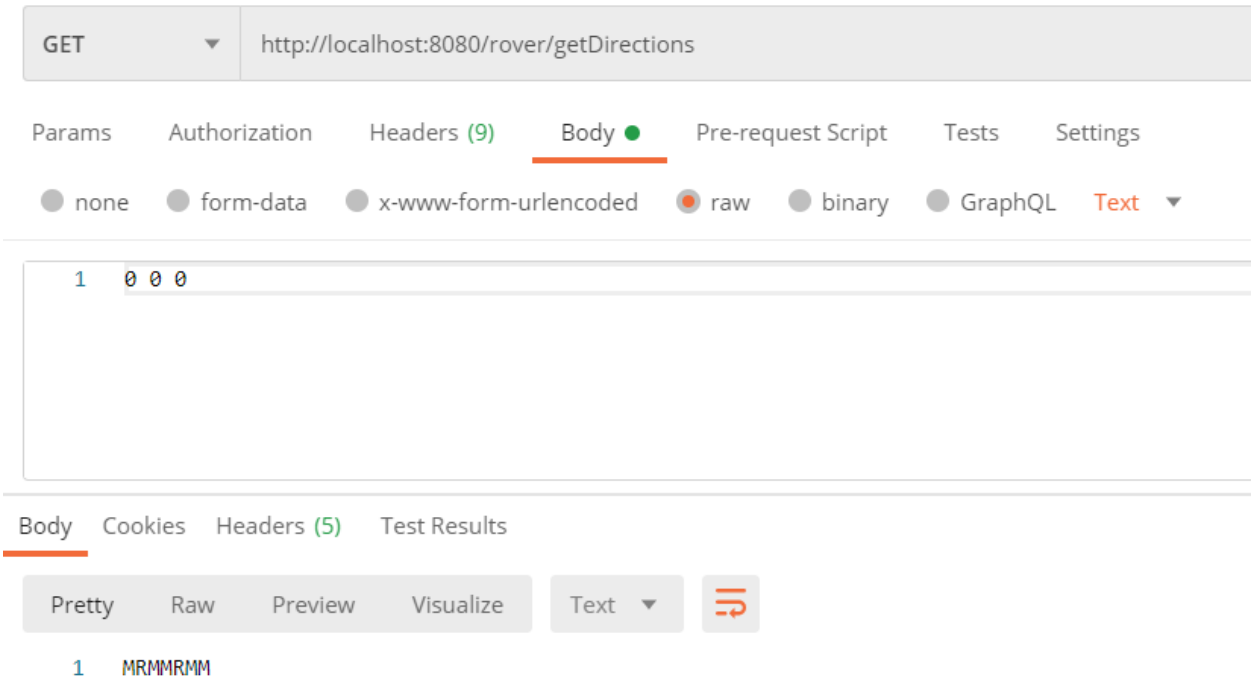
Pretty Raw Preview Visualize Text

```
[D][-][-]
[*][-][S]
[*][*][*]

```

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**



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