Welcome, this is the API documentation for the Water jug REST API, this documentation is aiming to help the user to use and understand correctly the proper behavior of the endpoint/s present at the app with basic usage examples.

This API provides an algorithmic solution to the water jug riddle, which consist of measuring a specific amount of water using only 2 jugs of x and y capacity, this API was build using NodeJS, TypeScript and Fastify.

Endpoints

baseUrl/api/v1/water

```
POST /water
```

Description

This endpoint accepts a request with the capacities of two jugs and the target amount of water to be measured. The API runs the data through an algorithm to calculate the best possible solution to the riddle, returning either the steps required to measure the exact amount of water or a message indicating that the problem has no solution.

Body Schema

The body of the request must be in JSON format and should follow this schema:

Parameter	Type	Required	Description
x_capacity	Number	Yes	The capacity of the first jug (Positive Integer)
y_capacity	Number	Yes	The capacity of the second jug (Positive Integer)
z_amount_wanted	Number	Yes	The target amount of water to be measured (positive integer)

Response

Success response

Status code: 200

If the riddle has a solution, the response will include the steps to achieve the desired water amount in the best possible solution:


```
{
  "message": "Water jug problem solved",
  "solution": {
    "steps": [
      {
        "step": 1,
        "bucketX": 0,
        "bucketY": 5,
        "action": "Fill Y with 5L"
      },
      {
        "step": 2,
        "bucketX": 3,
        "bucketY": 2,
        "action": "Transfer from Y to X"
      },
      {
        "step": 3,
        "bucketX": 0,
        "bucketY": 2,
        "action": "Empty X"
      },
      {
        "step": 4,
        "bucketX": 2,
        "bucketY": 0,
        "action": "Transfer from Y to X"
      }
    ],
    "totalSteps": 4
  }
}
```


Error responses

Status code: 400

If the request body is missing required parameters or includes invalid data, the response will include an error message:

```
{
  "statusCode": 400,
  "code": "FST_ERR_VALIDATION",
  "error": "Bad Request",
  "message": "error message"
}
```


Error cases:

- - **Invalid Parameter (Negative Number)**
  If the request body is missing required parameters or includes invalid data, the response will include an error message:


```
{
  "statusCode": 400,
  "code": "FST_ERR_VALIDATION",
  "error": "Bad Request",
  "message": "body y_capacity is a required field."
}
```


- - **Invalid Parameter (Negative Number)**


```
{
  "statusCode": 400,
  "code": "FST_ERR_VALIDATION",
  "error": "Bad Request",
  "message": "body/x_capacity x_capacity must be a positive number."
}
```


Status code: 422

If the data sent by the user cannot be used to find a solution, the server will return a `422 Unprocessable Entity` error with a message specifying the issue:


```
{
  "statusCode": 422,
  "error": "Unprocessable Entity",
```

```
  "message": "error"
}
```

Error cases:

- - `z_amount_wanted` **is greater than the capacity of** `x_capacity` **and** `y_capacity`**:**

```
{
  "statusCode": 422,
  "error": "Unprocessable Entity",
  "message": "Amount wanted is greater than the capacity of both jugs. There
is not a possible solution"
}
```

- - `z_amount_wanted` is not a multiple of the greatest common divisor (GCD) of `x_capacity` and `y_capacity`

```
{
  "statusCode": 422,
  "error": "Unprocessable Entity",
  "message": "Amount wanted is not a multiple of the greatest common divisor
of the capacities of the jugs.  There is not a possible solution"
}
```

How to test the API

To test the API, you can use a tool like Postman or the ThunderClient.extension in Visual Studio Code. You can also use the command line with `curl`.

Example with curl

```
curl -X POST http://localhost:7338/api/v1/water \
-H "Content-Type: application/json" \
-d '{"x_capacity": 3, "y_capacity": 5, "z_amount_wanted": 4}'
```

Relevant Commands

Install Project Dependencies

```
npm install
```

Start the Server

```
npm run start
```

Run Unit Tests

```
npm run testing
```

made by Andrés Álvarez