Lab 6.1 - Implement a RESTful JSON POST

The labs-1 folder contains the following files:

- `model.js`
- `package.json`
- `validate.js`

The `model.js` file and the `package.json` file are exactly the same as the first lab exercise in the previous chapter.

The `model.js` file has the following content:

```
'use strict'

module.exports = {
  boat: boatModel()
}

function boatModel () {
  const db = {
    1: { brand: 'Chaparral', color: 'red' },
    2: { brand: 'Chaparral', color: 'blue' }
  }

  return {
    uid,
    create,
    read,
    update,
    del
```



```
}

function uid () {
  return Object.keys(db)
    .sort((a, b) => a - b)
    .map(Number)
    .filter((n) => !isNaN(n))
    .pop() + 1 + ''
}

function create (id, data, cb) {
  if (db.hasOwnProperty(id)) {
    const err = Error('resource exists')
    err.code = 'E_RESOURCE_EXISTS'
    setImmediate(() => cb(err))
    return
  }
  db[id] = data
  setImmediate(() => cb(null, id))
}

function read (id, cb) {
  if (!(db.hasOwnProperty(id))) {
    const err = Error('not found')
    err.code = 'E_NOT_FOUND'
    setImmediate(() => cb(err))
    return
  }
  setImmediate(() => cb(null, db[id]))
}

function update (id, data, cb) {
  if (!(db.hasOwnProperty(id))) {
    const err = Error('not found')
    err.code = 'E_NOT_FOUND'
    setImmediate(() => cb(err))
    return
  }
  db[id] = data
  setImmediate(() => cb())
}
```



```
  function del (id, cb) {
    if (!(db.hasOwnProperty(id))) {
      const err = Error('not found')
      err.code = 'E_NOT_FOUND'
      setImmediate(() => cb(err))
      return
    }
    delete db[id]
    setImmediate(() => cb())
  }
}
```

The `package.json` file looks as follows:

```
{
  "name": "labs-1",
  "version": "1.0.0",
  "description": "",
  "scripts": {
    "start": "echo \"Error: start script not specified\" && exit 1",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Use either Fastify or Express to implement a RESTful HTTP server so that when the command **`npm start`** is executed it starts a server that listens on **`process.env.PORT`**.

The server should support a **`POST`** request to **`/boat`** that uses the `model.js` file to create a new entry. The route should only accept **`application/json`** mime-type requests and should respond with **`application/json`** content-type responses.

The **`POST`** request should expect JSON data to be sent in the following format:

```
{ data: { brand, color } }
```

A successful request should respond with a 201 Created status code. Unexpected errors should result in a 500 Server Error response.

The service must also support the same **`GET /boat/{id}`** route as implemented in the previous chapter.

It is not necessary to validate user input for this exercise.

Feel free to copy the files and folders from the labs-1 answer of the previous chapter into the labs-1 folder of this chapter and then build upon, or else start from scratch, as preferred.

Making sure that the labs-1 folder is the current working directory, run the following command to validate the completed exercise:

```
node validate
```

When correctly implemented, this command should output the following: