
mansion
de subastas

Mansion de Subastas

You have been hired as a lead developer for an art-selling company called **Mansion de Subastas**. They are obsessed with **NodeJS** and will not accept anything else as their tech stack. Along with **NodeJS** they want **ExpressJS, MongoDB** (*with **Mongoose***). As always failure is not an option, these artist sellers demand perfection!

Rules

This applies for all routes:

- If a resource is not found, the web service should always return a status code 404
- In POST requests which include ids from another resource, the ids should be validated by checking if they exist. If they don't exist the web service should return a status code 400
- If an error occurs on the database level, the web service should return a status code 500

Assignment description

Below the functionality of the assignment is described:

Functionality (70%)

- Routes (*all route declarations should reside in **index.js***)
  - **/api/arts [GET]** - Gets all arts


```
[
    {
        "images": [
            "http://example.com/profile.jpg",
            "http://example2.com/profile.jpg"
        ],
        "isAuctionItem": false,
        "_id": "5bad660b5e40653b18e60e64",
        "title": "Nice",
        "artistId": "5bad25d988531b2e869ee144",
        "date": "2018-09-27T23:21:47.565Z",
        "description": "Nice painting!",
        "__v": 0
    },
    {
        "images": [
            "http://example.com/profile.jpg",
            "http://example2.com/profile.jpg"
        ],
        "isAuctionItem": false,
        "_id": "5bad68c726eb083d4cdddd90",
        "title": "Nice",
        "artistId": "5bad25d988531b2e869ee149",
        "date": "2018-09-27T23:33:27.595Z",
        "description": "Nice painting!",
        "__v": 0
    }
]
```


  - **/api/arts/:id [GET]** - Gets an art by id


```
{
    "images": [
        "http://example.com/profile.jpg",
        "http://example2.com/profile.jpg"
    ],
    "isAuctionItem": false,
    "_id": "5bad68c726eb083d4cdddd90",
    "title": "Nice",
    "artistId": "5bad25d988531b2e869ee149",
    "date": "2018-09-27T23:33:27.595Z",
    "description": "Nice painting!",
    "__v": 0
}
```

- **/api/arts [POST]** - Creates a new art (*see how model should look like in **Model** section*)
- **/api/artists [GET]** - Gets all artists


```
[
    {
        "_id": "5bad25d988531b2e869ee144",
        "name": "A",
        "nickname": "B",
        "address": "C",
        "memberSince": "2018-08-13T00:00:00.000Z",
        "__v": 0
    },
    {
        "_id": "5bafa7694dcbb30781bd9be2",
        "name": "A",
        "nickname": "B",
        "address": "C",
        "memberSince": "2018-08-13T00:00:00.000Z",
        "__v": 0
    }
]
```


- **/api/artists/:id [GET]** - Gets an artist by id


```
{
    "_id": "5bafa7694dcbb30781bd9be2",
    "name": "A",
    "nickname": "B",
    "address": "C",
    "memberSince": "2018-08-13T00:00:00.000Z",
    "__v": 0
}
```


- **/api/artists [POST]** - Creates a new artist (*see how model should look like in **Model** section*)
- **/api/customers [GET]** - Gets all customers


```
[
    {
        "_id": "5bafacd34dcbb30781bd9be3",
        "name": "Customer 1",
        "username": "customer1",
        "email": "customer1@corp.com",
        "address": "Customer street 1",
        "__v": 0
    }
]
```


- **/api/customers/:id [GET]** - Gets a customer by id


```
{
    "_id": "5bafacd34dcbb30781bd9be3",
    "name": "Customer 1",
    "username": "customer1",
    "email": "customer1@corp.com",
    "address": "Customer street 1",
    "__v": 0
}
```


- **/api/customers [POST]** - Creates a new customer (*see how model should look like in **Model** section*)

- **/api/customers/:id/auction-bids [GET]** - Gets all auction bids associated with a customer


```
[
    {
        "_id": "5bafae3b5ac5d609ed293e47",
        "auctionId": "5bafad524dcbb30781bd9be4",
        "customerId": "5bafacd34dcbb30781bd9be3",
        "price": 20,
        "__v": 0
    },
    {
        "_id": "5bafb0bd4439df0a9ead1ba8",
        "auctionId": "5bafad524dcbb30781bd9be4",
        "customerId": "5bafacd34dcbb30781bd9be3",
        "price": 20,
        "__v": 0
    }
]
```


- **/api/auctions [GET]** - Gets all auctions


```
[
    {
        "_id": "5bafad524dcbb30781bd9be9",
        "artId": "5bad68c726eb083d4cdddd90",
        "minimumPrice": 10,
        "endDate": "09/09/2018 10:00:00",
        "auctionWinner": "5bafacd34dcbb30781bd9be3",
        "__v": 0
    }
]
```


- **/api/auctions/:id [GET]** - Gets an auction by id


```
{
    "_id": "5bafad524dcbb30781bd9be9",
    "artId": "5bad68c726eb083d4cdddd90",
    "minimumPrice": 10,
    "endDate": "09/09/2018 10:00:00",
    "auctionWinner": "5bafacd34dcbb30781bd9be3",
    "__v": 0
}
```


- **/api/auctions/:id/winner [GET]** - Gets the winner of the auction. If the auction is not finished the web service should return a status code 409 (Conflict), otherwise it should return the customer which holds the highest bid. If the auction had no bids, it should return a status code 200 (OK) with the message: ‘This auction had no bids.’.


```
{
    "_id": "5bafacd34dcbb30781bd9be3",
    "name": "Customer 1",
    "username": "customer1",
    "email": "customer1@corp.com",
    "address": "Customer street 1",
    "__v": 0
}
```

- **/api/auctions [POST]** - Create a new auction (*see how model should look like in **Model** section*). The art id provided within the body must be a valid art id with its property **isAuctionItem** set to true. If the **isAuctionItem** is set to false, the web service should return a status code 412 (Precondition failed).
- **/api/auctions/:id/bids [GET]** - Gets all auction bids associated with an auction


```
[
    {
        "_id": "5bafae3b5ac5d609ed293e47",
        "auctionId": "5bafad524dcbb30781bd9be4",
        "customerId": "5bafacd34dcbb30781bd9be3",
        "price": 20,
        "__v": 0
    },
    {
        "_id": "5bafb0bd4439df0a9ead1ba8",
        "auctionId": "5bafad524dcbb30781bd9be4",
        "customerId": "5bafacd34dcbb30781bd9be3",
        "price": 20,
        "__v": 0
    }
]
```


- **/api/auctions/:id/bids [POST]** - Creates a new auction bid (*see how model should look like in **Model** section*). Auction bids must be higher than the minimum price and must also be higher than the current highest bid. If the auction bid price is lower than the minimum price or current highest bid, the web service should return a status code 412 (Precondition failed). If the auction has already passed its end date, the web service should return a status code 403 (Forbidden). As a side-effect the **auctionWinner** property in the **Auction** schema should be updated to the latest highest bidder.

- Services (*all service implementations should reside in **services/** folder for each corresponding service*)
  - **artService** - Should be used to get data from the database using the schema **Art** and tailor the data according to the endpoints needs
  - **artistService** - Should be used to get data from the database using the schema **Artist** and tailor the data according to the endpoints needs
  - **auctionService** - Should be used to get data from the database using the schemas **Auction** and **AuctionBids** and tailor the data according to the endpoints needs
  - **customerService** - Should be used to get data from the database using the schema **Customer** and tailor the data according to the endpoints needs

- Schemas *(all Mongoose schemas should be declared within **schemas/** folder).* The red asterisk means that the field is required.
  - **Art** - title* (*String*), artistId* *(ObjectId*), date* (*Date, defaults to now*), images (*A list of String*), description (*String*), isAuctionItem (*Boolean, defaults to false*)
  - **Artist** - name* (*String*), nickname* (*String*), address* (*String*), memberSince* (*Date, defaults to now*)
  - **Auction** - artId* (*ObjectId*), minimumPrice (*Number, defaults to 1000*), endDate* (Date), auctionWinner (ObjectId, which should be a valid id of a customer, which holds the highest bid)
  - **AuctionBid** - auctionId* (*ObjectId*), customerId* (*ObjectId*), price* (*Number*)
  - **Customer** - name* (*String*), username* (*String*), email* (*String*), address* (*String*)

- Database (*database connection should reside within data/ folder*)
  - **db.js** - The connection to the database should be setup within this folder and connected to the schemas. The schemas should then be exported from this file.

Structure (30%)

- **(30%)** The solution must be implemented using **Node** with **ExpressJS**
- **(30%)** The solution must make use of **MongoDB** with **Mongoose** to retrieve/store data
- **(30%)** The solution should be modular and follow the **SRP** principle, what this means is that it should follow the directory structure seen in the **Folder structure** section below
- **(10%)** All packages should be installed using either **NPM** or **Yarn** and they should be tracked within package.json

Folder structure

- mansion_de_subastas/
  - schemas/
    - art.js
    - artist.js
    - auction.js
    - auctionBid.js
    - customer.js
  - data/
    - db.js
  - services/
    - artService.js
    - artistService.js
    - auctionService.js
    - customerService.js
  - index.js

Models

ART

```
{
    "images": [
        "http://example.com/profile.jpg",
        "http://example2.com/profile.jpg"
    ],
    "isAuctionItem": false,
    "title": "Nice",
    "artistId": "5bad25d988531b2e869ee144",
    "date": "2018-09-27T23:33:27.595Z",
    "description": "Nice painting!"
}
```

ARTIST

```
{
    "name": "A",
    "nickname": "B",
    "address": "C",
    "memberSince": "2018-08-13T00:00:00.000Z"
}
```

CUSTOMER

```
{
    "name": "Customer 1",
    "username": "customer1",
    "email": "customer1@corp.com",
    "address": "Customer street 1"
}
```

AUCTION

```
{
    "artId": "5bad68c726eb083d4cdddd90",
    "minimumPrice": 10,
    "endDate": "09/09/2018 10:00:00"
}
```

AUCTION BID

```
{
    "customerId": "5bafacd34dcbb30781bd9be3",
    "price": 20
}
```

Submission

A single compressed (***.zip, *.rar**) file should be submitted on **Canvas**. Don't forget to comment the name of each group member (*excluding the one who submitted*) and remove the **node_modules/** folder.