Practice 1
Web App Persistence and REST API Definition

Gabriel Cebrián Márquez
Gabriel.Cebrian@uclm.es
Web Engineering and Services
Faculty of Computer Science Engineering
University of Castilla-La Mancha

Introduction

The goal of this practice is to implement the data model presented in Seminar 1 using MongoDB, and to provide access to its resources by defining a REST API.

In this practice, the students must apply the concepts and technologies studied in Seminar 1. The seminar can be used as a basis for the practice. Remember, however, that students must do this practice for the type of product they selected for their project (not videogames!). Although the organization and structure of the application is up to the students, it is highly recommended that students follow the structure presented in Seminar 1 and in this document.

Procedure

During this procedure student should:

- Implement the data model.
- Develop the REST API in Annex A. Note that some of the operations may have been already implemented in the seminar.
- Test the correctness of the REST API.

The following steps describe the procedure to fulfil this practice goal:

1. Implement the **Order** model in `/src/models/Order.ts`.
2. Modify the `/db/seeder.ts` file, so that the seeder clears the database first, and then populates the database with some sample products. Optionally, the seeder may also add one or more sample users for testing purposes.
   Tip: By using `createCollection()` on a model you can create empty collections, e.g., `Orders.createCollection()`.
3. For each of the REST API endpoint in Annex A, do the following:
   (a) Define what the inputs and the outputs should be for the handler according to the definition of the API.
   (b) Implement an interface for the data type returned by the handler in `/src/lib/handlers.ts`.

(c) Implement the logic of the handler in `/src/lib/handlers.ts`. The handler must take the inputs identified above as arguments, and must return an object conforming the defined interface as a result.
(d) Create a `route.ts` file (if it has not been created previously) in the corresponding folder inside `/src/api` according to the rules of Next.js's app router.
(e) Implement the corresponding operation (**GET**, **POST**…) in the `route.ts` file. Pay special attention to the parameters received in the request's body, and the response codes or headers that must be returned in the response.

4. Test the REST API thoroughly using Postman.

There are some aspects that you will have to take into consideration when carrying out the above procedure:

- Given that the seeder first drops and then recreates the database, the new documents will have different IDs. This must be taken into account when testing the REST API endpoints that take IDs as parameters.
- The **Order** model should have an `orderItems` attribute similar to the `cartItems` attribute in the **User** model, but for storing order items.
- Think the logic of each handler through. For instance, you must consider these aspects (not limited to):
  - When creating a new user, the handler must check if there is a user with the same e-mail already.
  - When creating an order, the date is not provided in the request body. Instead, the date is generated by the server with the current time of the request.
  - When creating an order, each cart item of the user's cart has to be "transformed" into an order item. Remember that order items have a price attribute, which takes the value of the product's price at the time of purchase. After finishing the purchase, the user's cart must be cleared.
  - The `qty` parameter in **PUT /api/users/[userId]/cart/[productId]** must be greater than or equal to 1 (we are creating a cart item or modifying an existing one). To delete a cart item, the **DELETE** operation must be used instead.

Please, ask if you have any questions!

Annex A: REST API

GET /api/products

Description Retrieve the list of products.

Parameters None.

Request body None.

Responses

- **200**: Success.

  Content-type: application/json

```
{
  "products": [
    {
      "_id": "66f34df0d7d45a791b6b14c6",
      "name": "Earthen Bottle",
      "price": 39.95,
      "img": "https://tailwindui.com/img/ecommerce-images/category-page-04-image-card-01.jpg",
      "description": "What a bottle!"
    },
    {
      "_id": "66f34df0d7d45a791b6b14c7",
      "name": "Nomad Tumbler",
      "price": 39.95,
      "img": "https://tailwindui.com/img/ecommerce-images/category-page-04-image-card-02.jpg",
      "description": "Yet another item"
    }
  ]
}
```

GET /api/products/[productId]

Description Retrieve the details of a product.

Parameters

- **productId** (ObjectId): ID of the product.

Request body None

Responses

- **200**: Success.

  Content-type: application/json

```
{
  "_id": "66f34df0d7d45a791b6b14c6",
  "name": "Earthen Bottle",
  "price": 39.95,
  "img": "https://tailwindui.com/img/ecommerce-images/category-page-04-image-card-01.jpg",
  "description": "What a bottle!"
}
```

- **400**: Invalid product ID.
- **404**: Product not found.

POST /api/users

Description Create a new user.

Parameters None.

Request body

Content-type: application/json

```
{
  "email": "foobar@example.com",
  "password": "1234",
  "name": "Foo",
  "surname": "Bar",
  "address": "Unnamed Street 123, 12345 London, UK",
  "birthdate": "1970-01-01"
}
```

Responses

- **201**: User created. The `Location` attribute of the response header should be set to the path of the user created. Example:

  Content-type: application/json

  ```
  {
    "_id": "66f3525beeaacfcebbc1677b"
  }
  ```

  Location: /api/users/66f3525beeaacfcebbc1677b

- **400**: Invalid request. This can happen if the request body is invalid or incomplete, or if the e-mail address of the user is already in use.

Implementation remarks

- Remember that the **User** model has two more attributes apart from the ones provided in the request body: `cartItems` and `orders`. These must be initialized empty when creating the user document.

GET /api/users/[userId]

Description Retrieve the details of an existing user (password, orders and cart items not included).

Parameters

- **userId** (ObjectId): ID of the user.

Request body None.

Responses

- **200**: Success.

  Content-type: application/json


```
{
  "_id": "66f3525beeaacfcebbc1677b",
  "email": "foobar@example.com",
  "name": "Foo",
  "surname": "Bar",
  "address": "Unnamed Street 123, 12345 London, UK",
  "birthdate": "1970-01-01T00:00:00.000Z"
}
```


- **400**: Invalid user ID.
- **404**: User not found.

GET /api/users/[userId]/cart

Description Retrieve the list of cart items of an existing user.

Parameters

- **userId** (ObjectId): ID of the user.

Request body None.

Responses

- **200**: Success.

  Content-type: application/json


```
{
  "cartItems": [
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c6",
        "name": "Earthen Bottle",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-01.jpg",
        "description": "What a bottle!"
      },
```

```
      "qty": 2
    },
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c7",
        "name": "Nomad Tumbler",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-02.jpg",
        "description": "Yet another item"
      },
      "qty": 5
    }
  ]
}
```


- **400**: Invalid user ID.
- **404**: User not found.

Implementation remarks

- As can be seen in the response, the products are not just references (ObjectId). Instead, they have been populated to show their details.

PUT /api/users/[userId]/cart/[productId]

Description Modify the number of items of a given product in the cart of an existing user. If the product does not exist in the cart, a new entry is created.

Parameters

- **userId** (ObjectId): ID of the user.
- **productId** (ObjectId): ID of the product.

Request body

Content-type: application/json


```
{
  "qty": 4
}
```


Responses

- **200**: Success. The product was already in cart and the number of items was updated. The entire updated cart is returned.

  Content-type: application/json


```
{
  "cartItems": [
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c6",
        "name": "Earthen Bottle",
```

```
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-01.jpg",
        "description": "What a bottle!"
      },
      "qty": 4
    },
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c7",
        "name": "Nomad Tumbler",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-02.jpg",
        "description": "Yet another item"
      },
      "qty": 5
    }
  ]
}
```


- **201**: The product was not in the cart, so a new entry was created. The entire updated cart is returned. The `Location` attribute of the response header should be set to the path of the cart item.

  Content-type: application/json


```
{
  "cartItems": [
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c6",
        "name": "Earthen Bottle",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-01.jpg",
        "description": "What a bottle!"
      },
      "qty": 2
    },
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c7",
        "name": "Nomad Tumbler",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-02.jpg",
        "description": "Yet another item"
      },
      "qty": 5
    }
  ]
}
```


  Location: /api/users/66f3525beeaacfcebbc1677b/cart/66f34df0d7d45a791b6b14c6

- **400**: Invalid user ID, invalid product ID or number of items not greater than 0.

- **404**: User not found or product not found.

DELETE /api/users/[userId]/cart/[productId]

Description Remove a product from the cart of an existing user.

Parameters

- **userId** (ObjectId): ID of the user.
- **productId** (ObjectId): ID of the product.

Request body None.

Responses

- **200**: Success. The product was removed from the cart or it did not exist. The entire updated cart is returned.

  Content-type: application/json


```
{
  "cartItems": [
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c7",
        "name": "Nomad Tumbler",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-02.jpg",
        "description": "Yet another item"
      },
      "qty": 5
    }
  ]
}
```


- **400**: Invalid user ID or invalid product ID.
- **404**: User not found or product not found.

GET /api/users/[userId]/orders

Description Retrieve the list of orders of an existing user.

Parameters

- **userId** (ObjectId): ID of the user.

Request body None.

Responses

- **200**: Success.

  Content-type: application/json

```
{
  "orders": [
    {
      "_id": "66f34df0d7d45a791b6b14c9",
      "address": "123 Main St, 12345 New York, United States",
      "date": "2024-09-24T23:40:32.321Z",
      "cardHolder": "John Doe",
      "cardNumber": "1111222233334444",
      "orderItems": [
        {
          "product": "66f34df0d7d45a791b6b14c6",
          "qty": 1,
          "price": 19.99
        },
        {
          "product": "66f34df0d7d45a791b6b14c7",
          "qty": 2,
          "price": 29.99
        }
      ]
    },
    {
      "_id": "66f34df0d7d45a791b6b14ca",
      "address": "456 Test St, 00000 Alabama, United States",
      "date": "2024-09-24T23:40:32.322Z",
      "cardHolder": "John Doe Jr.",
      "cardNumber": "5555666677778888",
      "orderItems": [
        {
          "product": "66f34df0d7d45a791b6b14c7",
          "qty": 3,
          "price": 24.99
        }
      ]
    }
  ]
}
```


- **400**: Invalid user ID.
- **404**: User not found.

Implementation remarks

- As can be seen in the response, the orders have been populated, but the products of each order have not.

POST /api/users/[userId]/orders

Description Create a new order for an existing user.

Parameters

- **userId** (ObjectId): ID of the user.

Request body

Content-type: application/json

```
{
  "address": "Unnamed Street 123, 12345 London, UK",
  "cardHolder": "Foo Bar",
  "cardNumber": "0000111122223333"
}
```

Responses

- **201**: Order created. The `Location` attribute of the response header should be set to the path of the order created. Example:

  Content-type: application/json

  ```
  {
    "_id": "66f358b7eeaacfcebbc1678a"
  }
  ```

  Location: /api/users/66f3525beeaacfcebbc1677b/orders/66f358b7eeaacfcebbc1678a

- **400**: Invalid user ID or invalid request. The latter can happen if the request body is invalid or incomplete, or the cart is empty.

- **404**: User not found.

Implementation remarks

- This operation represents the purchase of the cart contents.
- Creating an order for a user implies transforming the current cart items into order items. Remember that order items have a price attribute that corresponds to the current price of the product. This can be interpreted as the price at which the product was bought.
- The cart of the user must be empty after the order is created successfully.
- The list of orders of the user must be updated accordingly.

GET /api/users/[userId]/orders/[orderId]

Description Find an existing order of an existing user by ID.

Parameters

- **userId** (ObjectId): ID of the user.
- **orderId** (ObjectId): ID of the order.

Request body None.

Responses

- **200**: Success. Example:

  Content-type: application/json


```
{
  "_id": "66f358b7eeaacfcebbc1678a",
  "address": "Unnamed Street 123, 12345 London, UK",
  "date": "2024-09-25T00:26:31.165Z",
  "cardHolder": "Foo Bar",
  "cardNumber": "0000111122223333",
  "orderItems": [
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c6",
        "name": "Earthen Bottle",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-01.jpg",
        "description": "What a bottle!"
      },
      "qty": 4,
      "price": 39.95
    },
    {
      "product": {
        "_id": "66f34df0d7d45a791b6b14c7",
        "name": "Nomad Tumbler",
        "price": 39.95,
        "img": "https://tailwindui.com/img/ecommerce-images/
            category-page-04-image-card-02.jpg",
        "description": "Yet another item"
      },
      "qty": 5,
      "price": 39.95
    }
  ]
}
```


- **400**: Invalid user ID or invalid order ID.
- **404**: User not found or order not found.

Implementation remarks

- As can be seen in the response, the products are not just references (ObjectId). Instead, they have been populated to show their details.