RESTful API built in NODE.JS. The concept is built around an API that would serve a store. The user could do simple tasks like create products and costumers, as well modify them. Also, after completing a sale, a new object is created using data from the product and the customer.

Initialization of the App:

- node index.js

Database:

- Local: MongoDB Compass
- Cloud: MongoDB Atlas

NPM:

- compression@1.7.4
- config@3.3.7
- express@4.17.3
- fawn@2.1.5
- helmet@5.0.2
- joi@17.6.0
- lodash@4.17.21
- mongoose@6.2.4

The app consists of three routes and three models. One for customers, one for products and one for sales.

Customers:

MongoDB Schema:

- name: type:String / required: true / minlength: 2 / maxlength: 50;
- email: type:String / minlength: 2 /maxlength: 50;
- phone: type:String / minlength: 2 /maxlength: 50

A second validation is done by Joi.

Routes:

GET /api/customers – Returns all the registered costumers.

GET/api/customers/customerID?:email - returns the customer ID using the registered email. The method _.pick (lodash) is used to send only the customer ID instead the complete information about the customer.

POST /api/customers – Creates a new costumer and saves it to the database.

PUT /api/customers/:id – Updates an already existing customer. – each field can be updated individually. All the restrictions apply except the required().

DELETE /api/customers/:id – Deletes the costumer with the given ID.

Products:

MongoDB Schema:

- product_id: type:String/required: true/minlength: 2/maxlength: 50;
- item: type:String/required: true/minlength: 2/maxlength: 50;
- price: type:mongoose.Types.Decimal128/required: true/minlength: 2/maxlength: 50;
- description: type: String/required: true/minlength: 2/maxlength: 250;

A second validation is done by Joi.

Routes:

GET /api/products – Returns all the registered products.

GET/api/customers/productID?:product_id – Returns the product with the given product_id.

POST /api/products – Creates a new product and saves it to the database.

PUT /api/products/:id – Updates an already existing product. – each field can be updated individually. All the restrictions apply except the required().

DELETE /api/products/:id – deletes a product based on the ID.

Sales:

MongoDB Schema:

- customerId: type:String/required: true/minlength: 2/maxlength: 50;
- itemId: type:String/required: true/minlength: 2/maxlength: 50;
- date: type:Date / default: Date.now;
- amount: type:Number/required: true;

A second validation is done by Joi.

Routes:

GET /api/sales – Returns all the sales.

GET/api/sales/:id – Returns the sale with the given product_id.

POST /api/sales – Creates a new sale and saves it to the database. The object will include information taken from one costumer and one product. The amount of the sale will be deducted from the stock of the product.

Implementation of Authentication and Authorization.

Authentication and Authorization were introduced to the API. Some routes are now protected and can only be accessed by authenticated users. The integrity of the data is protected by JSON WEB TOKENS. Some routes were also protected using a function that will only allow authenticated users that have admin privileges.

Added NPM:

- bcrypt@5.0.1
- jsonwebtoken@8.5.1

Creating a new User:

- When creating a new user, the password is salted and hashed and only after is stored in the database. The process is done using bcrypt@5.0.1.