САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

RESTful API средствами express + typescript

Выполнила:
Зайцева А. А.
Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

В рамках данной лабораторной работы Вам предложено выбрать один из нескольких вариантов. Выбранный вариант останется единым на весь курс и будет использоваться в последующих лабораторных работах.

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Вариант 4. Сайт администратора интернет-магазина

- Вход
- Регистрация
- Учёт товара на складе
- Графики по продажам тех или иных товаров, по общей выручке предприятия
- Управление сотрудниками

Ход работы

Для реализации функционала входа и регистрации в **package.json** были добавлены пакеты **passport** и **passport-jwt**.

В файл настроек **settings.ini** были добавлены настройки jwt: поля **accessTokenLifetime** и **refreshTokenLifetime**.

Для подключения функционала **passport** был создан **middleware passport.ts** – он содержит базовые настройки jwt авторизации. Данный middleware подключается к express в точке входа приложения index.ts:

```
private createApp(): express.Application {
    const app = express()
    app.use(cors())
    app.use(bodyParser.json())
    app.use(passport.initialize())
    app.use('/v1', routes)

    return app
}
```

Были добавлены новые sequelize модели: refreshToken, product, supply и sale.

- **RefreshToken** – модель для хранения refresh токенов в базе данных. Содержит поля:
  - **token** (string) – само значение токена;
  - **userId** (integer) – связанный пользователь.
- **Product** – товар интернет магазина. Содержит поля:
  - **name** (string) – название товара;
  - **price** (integer) – стоимость одной единицы товара на данный момент.
- **Supply** – поставка товара. Содержит поля:
  - **productId** (integer) – связанный товар;
  - **quantity** (integer) – количество единиц товара в поставке;
  - **dateOfSupply** (date) – дата поставки.
- **Supply** – Продажа товара. Содержит поля:
  - **productId** (integer) – связанный товар;
  - **quantity** (integer) – количество проданных единиц товара;
  - **price** (integer) – полная сумма продажи (кол-во проданных единиц * стоимость единицы товара на дату покупки);
  - **dateOfSale** (date) – дата продажи.

Также в модель пользователя **User** было добавлено **integer** поле **role**, допускающее одно из следующих значений: 1 – admin, 2 – manager, 3 – employee. Поле role служит для ограничения возможностей того или иного пользователя.

Для новых моделей были созданы соответствующие им сервисы-репозитории:

- **ProductRepository** – содержит базовые методы **getAll**, **getById**, **create**, **deleteById**, **updateById**.
- **SupplyRepository** – содержит базовые методы и метод **getSuppliedAmount**, возвращающий количество поставленных единиц конкретного товара за определенный период или за все время.

```
interface SuppliedRequestParams {
    productId: number;
    dateFrom?: Date;
    dateTo?: Date;
}
```

```
async getSuppliedAmount(params: SuppliedRequestParams): Promise<number>
```

- **SaleRepository -** содержит базовые методы и методы:
    - **getSoldRevenue –** возвращает выручку по конкретному товару или по всем товарам за определенный период или за все время (регулируется параметрами);
    - **getSoldAmount –** то же, что и getSoldRevenue, но для количества проданных единиц;
    - **getSalesRevenueGroupedByDay –** возвращает массив продаж, сгруппированный по дням для конкретного товара или всех товаров за определенный период или за все время, таким образом позволяет получить значения для построения графика продаж;
    - **getSalesAmountGroupedByDay –** то же, что и getSalesRevenueGroupedByDay, но для количества проданных единиц.

Для моделей Product, Supply и Sale были созданы соответствующие им контроллеры, предоставляющие доступ к методам репозиториев.

Методы контроллеры в свою очередь были связанный с определенными роутами.

Для тестирования работы приложения были добавлены сидеры, заполняющие базу данных тестовыми данными.

Вывод

В ходе второй лабораторной работы я создала RESTful API средствами express, typescript и typescript-sequelize. Также я освоила способ авторизации по jwt токену и реализовала данный метод авторизации с помощью пакетов passport и passport-jwt.