САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:
Куцало Александр
Группа
M3221d

Проверил:
Добряков Д. И.

Санкт-Петербург

Задача

Создать RESTful сервис для работы с магазином одежды. Требуемый функционал: регистрация, авторизация, создание профиля, работа с товарами, просмотр количества единиц товара, управление скидками и акциями, работа с базой клиентов.

Ход работы

Для выполнения была изменена файловая структура проекта, добавлены новые сущности:

```
src
|-- controllers
|   |-- shop.ts
|   `-- user.ts
|-- errors
|   `-- userErrors.ts
|-- index.ts
|-- instances
|   `-- db.ts
|-- middleware
|   `-- authMiddleware.ts
|-- models
|   |-- Item.ts
|   |-- ItemTag.ts
|   |-- RefreshToken.ts
|   |-- ShoppingCartItem.ts
|   |-- tag.ts
|   `-- user.ts
|-- routers
|   |-- _admin.ts
|   |-- admin.ts
|   |-- router.ts
|   |-- shop.ts
|   `-- users.ts
|-- services
|   |-- shop.ts
|   `-- user.ts
|-- utility
|   |-- createToken.ts
|   |-- destroyTokens.ts
|   |-- makeTokens.ts
|   |-- passwordCheck.ts
|   |-- verifyExpiration.ts
|   `-- verifyRefreshToken.ts
```

Была создана модель товара, тега товара, единицы в корзине, а также необходимые для их взаимодействия таблицы отношений товаров в корзине и пользователей, товаров и тегов.

```
import { BelongsTo, Column, DataType, ForeignKey, Model, Table } from
"sequelize-typescript";
import Item from "./Item";
import Tag from "./Tag";

@Table
class ItemTag extends Model {
    @ForeignKey(() => Item)
    @Column(DataType.INTEGER)
    itemId: number

    @ForeignKey(() => Tag)
    @Column(DataType.INTEGER)
    tagId: number

    @BelongsTo(() => Item)
    item: Item

    @BelongsTo(() => Tag)
    tag: Tag
}

export default ItemTag
```

```
import { AllowNull, BelongsToMany, Column, DataType, Model, Table } from
"sequelize-typescript";
import Tag from "./Tag";
import ItemTag from "./ItemTag";


@Table
class Item extends Model {

    @AllowNull(false)
    @Column(DataType.STRING)
    name: string

    @Column(DataType.STRING)
    description: string

    @AllowNull(false)
    @Column(DataType.INTEGER)
    price: number

    @AllowNull(false)
    @Column(DataType.STRING)
    imageURL: string

    @BelongsToMany(() => Tag, () => ItemTag)
    tags: Tag[]

    @Column(DataType.INTEGER)
    quantity: number
}
```

```
export default Item
```

```
import { BelongsTo, Column, DataType, ForeignKey, Model, Table } from
"sequelize-typescript";
import User from "./User";
import Item from "./Item";

@Table
class ShoppingCartItem extends Model {

    @ForeignKey(() => User)
    @Column(DataType.INTEGER)
    userId: number

    @BelongsTo(() => User, 'userId')
    user: User

    @ForeignKey(() => Item)
    @Column(DataType.INTEGER)
    itemId: number

    @Column(DataType.INTEGER)
    quantity: number

    @BelongsTo(() => Item, 'itemId')
    item: Item

}


export default ShoppingCartItem
```

```
import {AllowNull, BelongsToMany, Column, DataType, Model, Table } from
"sequelize-typescript";
import ItemTag from "./ItemTag";
import Item from "./Item";

@Table
class Tag extends Model {
    @AllowNull(false)
    @Column(DataType.STRING)
    name: string

    @BelongsToMany(() => Item, () => ItemTag)
    items: Item[]

}

export default Tag
```

```
import { Table, Column, Model, Unique, AllowNull, BeforeCreate,
BeforeUpdate, DataType, IsEmail, HasMany } from 'sequelize-typescript'
import {hashSync} from 'bcrypt'
import ShoppingCartItem from './ShoppingCartItem'
import Item from './Item'
```

```
@Table
class User extends Model {

    @Column(DataType.STRING)
    name: string

    @Unique
    @AllowNull(false)
    @IsEmail
    @Column(DataType.STRING)
    email: string

    @AllowNull(false)
    @Column(DataType.STRING)
    password: string

    @HasMany(() => ShoppingCartItem, 'userId')
    shoppingCartItems: Item[]

    @BeforeCreate
    @BeforeUpdate
    static generatePasswordHash(instance: User) {
        const { password } = instance

        if (instance.changed('password')) {
            instance.password = hashSync(password, 5)

        }
    }
}

export default User
```

Также были изменены эндпоинты:

```
import { Router } from "express";
import userRouter from "./users";
import shopRouter from "./shop";
import adminRouter from "./admin";


const mainRouter = Router()

mainRouter.use('/users', userRouter)

mainRouter.use('/shop', shopRouter)

mainRouter.use('/admin', adminRouter)


export default mainRouter
```

```
import { Router } from "express";
import ShopController from "../controllers/shop";
import requireAuth from "../middleware/authMiddleware";
```

```
const shopRouter = Router()
const shopController = new ShopController()

shopRouter.route('/')
    .get(shopController.index)

shopRouter.route('/item')
    .get(shopController.getAllItems) // {tagId: number[]}

shopRouter.route('/item/:id')
    .get(shopController.getItem)
    .post(requireAuth, shopController.addToShoppingCart) // {'quantity':
number}

shopRouter.route('/cart')
.get(requireAuth, shopController.getShoppingCartContents)

export default shopRouter
```

```
import { Router } from "express";
import UserController from "../controllers/user";
import requireAuth from "../middleware/authMiddleware";

const userRouter: Router = Router()

const userController: UserController = new UserController()


userRouter.route('/id/:id')
    .get(userController.get)

userRouter.route('/create/')
    .post(userController.create)

userRouter.route('/login')
    .post(userController.login)

userRouter.route('/auth_only')
    .get(requireAuth, userController.privatePage)

userRouter.route('/logout')
    .post(userController.logout)
export default userRouter
```

Также была изменена система авторизации и аутентификации, теперь используется система из 2 токенов, один из которых также хранится в базе данных:

```
import { AllowNull, BeforeCreate, BeforeUpdate, BelongsTo, Column,
DataType, Default, Model, Table } from "sequelize-typescript";
import User from "./User";
import { randomUUID } from "crypto";

@Table
class RefreshToken extends Model {

    @Default(DataType.UUIDV4)
    @Column(DataType.UUID)
    token: string

    @AllowNull(false)
    @Column(DataType.NUMBER)
    userId: number

    @Column(DataType.DATE)
    expirationDate: Date

    @BelongsTo(() => User, 'userId')
    user: User

    @BeforeUpdate
    @BeforeCreate
    static generateDate(instance: RefreshToken) {
        instance.expirationDate = new Date(Date.now() +
Number(process.env.REFRESH_TOKEN_AGE_MS))
    }

    @BeforeUpdate
    static setNewToken(instance: RefreshToken) {
        instance.token = randomUUID()
    }
}

export default RefreshToken
```

Вывод

В ходе выполнения данной работы была спроектирована и имплементирована система для менеджмента онлайн-магазина одежды