

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа

Выполнила:

Саунин Антон

Группа К33402

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

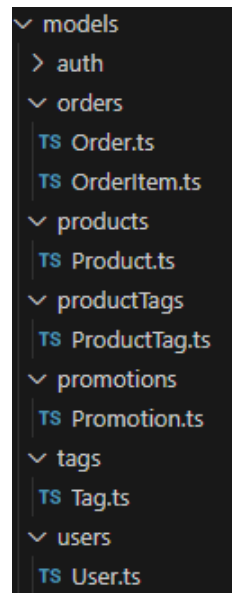
Задача

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

Вариант: интернет-магазин.

Ход работы

Созданы модели:



Для примера реализации используем продукт:

```
export type ProductAttributes = {
  id: number;
  name: string;
  price: number;
};

export type ProductCreationAttributes = Optional<ProductAttributes, "id">;

@Table
export class Product extends Model<
  ProductAttributes,
  ProductCreationAttributes
> {
  @AutoIncrement
  @AllowNull(false)
  @PrimaryKey
  @Column
  declare id: number;

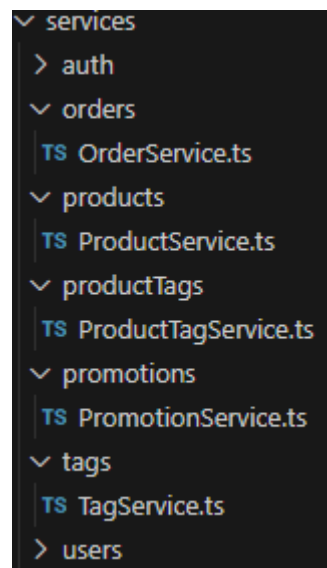
  @AllowNull(false)
  @Column
  declare name: string;

  @AllowNull(false)
  @Column
  declare price: number;

  @BelongsToMany(() => Tag, () => ProductTag)
  tags: Tag[];
}

export default Product;
```

Созданы соответствующие сервисы:



Для примера реализации сейчас и далее будем также использовать продукт:

```
class ProductService {
  async getAll() {
    return ProductRepository.getAll();
  }

  async getById(id: number) {
    return ProductRepository.getById(id);
  }

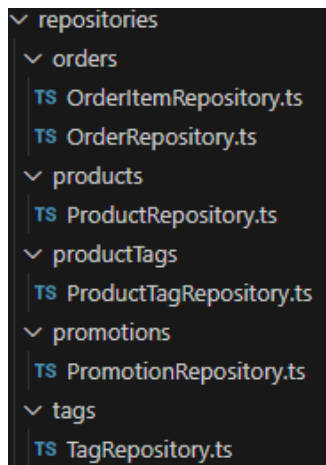
  async create(productData: any) {
    return ProductRepository.create(productData);
  }

  async update(id: number, productData: any) {
    return ProductRepository.update(id, productData);
  }

  async delete(id: number) {
    return ProductRepository.delete(id);
  }
}

export default new ProductService();
```

Репозитории:



Реализация:

```
class ProductRepository {
  async getAll() {
    return Product.findAll();
  }

  async getById(id: number) {
    return Product.findPk(id);
  }

  async create(productData: any) {
    return Product.create(productData);
  }

  async update(id: number, productData: any) {
    const product = await Product.findPk(id);
    if (product) {
      return product.update(productData);
    }
    return null;
  }

  async delete(id: number) {
    const product = await Product.findPk(id);
    if (product) {
      return product.destroy();
    }
    return null;
  }
}

export default new ProductRepository();
```

Контроллеры:

```

└─ controllers
  └─ > auth
  └─ orders
    └─ TS OrderController.ts
  └─ products
    └─ TS ProductController.ts
  └─ productTags
    └─ TS ProductTagController.ts
  └─ promotions
    └─ TS PromotionController.ts
  └─ tags
    └─ TS TagController.ts
  └─ > users
```

Реализация:

```

class ProductController {
  async getAll(req: Request, res: Response) {
    const products = await ProductService.getAll();
    res.json(products);
  }

  async getById(req: Request, res: Response) {
    const product = await ProductService.getById(parseInt(req.params.id, 10));
    res.json(product);
  }

  async create(req: Request, res: Response) {
    console.log(req.body);
    const newProduct = await ProductService.create(req.body);
    res.status(201).json(newProduct);
  }

  async update(req: Request, res: Response) {
    const updatedProduct = await ProductService.update(
      parseInt(req.params.id, 10),
      req.body
    );
    res.json(updatedProduct);
  }

  async delete(req: Request, res: Response) {
    await ProductService.delete(parseInt(req.params.id, 10));
    res.status(204).send();
  }
}

export default new ProductController();
```

Роутер делится на отдельные файлы для соответствующих разделов:

```
▼ routes
  > auth
  ▼ orders
    TS OrderRoutes.ts
  ▼ products
    TS ProductRoutes.ts
  ▼ productTags
    TS ProductTagRoutes.ts
  ▼ promotions
    TS PromotionRoutes.ts
  ▼ tags
    TS TagRoutes.ts
  > users
  TS index.ts
```

Реализация:

productRoutes.ts:

```
const router = Router();

router.get("/", ProductController.getAll);
router.get("/:id", ProductController.getById);
router.post("/", ProductController.create);
router.patch("/:id", ProductController.update);
router.delete("/:id", ProductController.delete);

export default router;
```

Index.ts:

```
const router = Router();

router.use("/products", auth, ProductRoutes);
router.use("/tags", auth, TagRoutes);
router.use("/promotions", auth, PromotionRoutes);
router.use("/orders", auth, OrderRoutes);
router.use("/users", auth, UserRoutes);
router.use("/", AuthRoutes);
router.use("/productTags", auth, productTagRoutes);

export default router;
```

Пример запроса:

GET

http://localhost:5000/products/

Send

ParamsAuthorization●Headers (7)BodyScriptsSettingsCookies

☒ none☐ form-data☐ x-www-form-urlencoded☐ raw☐ binary☐ GraphQL

This request does not have a body

BodyCookiesHeaders (8)Test Results200 OK · 43 ms · 765 B · 🌐 📄 ⋮

PrettyRawPreviewVisualizeJSON📄🔍

```
1  [  
2    {  
3      "id": 1,  
4      "name": "Levis Jeans",  
5      "price": 250,  
6      "createdAt": "2024-09-12T23:23:27.548Z",  
7      "updatedAt": "2024-09-13T00:13:09.286Z"  
8    },  
9    {  
10     "id": 2,  
11     "name": "Armani t-shirt",  
12     "price": 370,  
13     "createdAt": "2024-09-13T00:10:44.532Z",  
14     "updatedAt": "2024-09-13T00:13:17.617Z"  
15   },  
16   {  
17     "id": 3,  
18     "name": "Palm Angel hoodie",  
19     "price": 299,  
20     "createdAt": "2024-09-17T14:58:18.450Z",  
21     "updatedAt": "2024-09-17T14:58:18.450Z"  
22   },  
23  ]
```



Вывод

В ходе данной работы был написан свой RESTful API средствами express + typescript