

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 2

Выполнил:

Пономарев Константин

Группа К33402

Проверил:

Добряков Д. И.

**Санкт-Петербург
2024 г.**

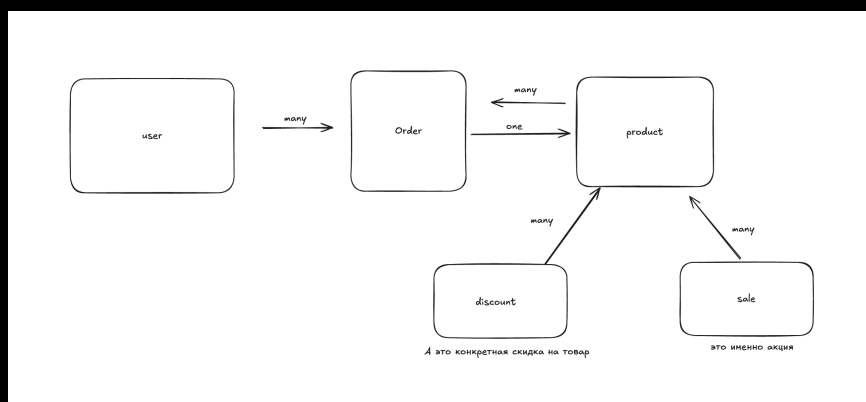
Задача




Написать свой сервис. 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 начала надо хотя вершнеуровнево составить модели, чтобы понять как определить связи. Помимо этого можно взять бойлерплейты с 1лр, поскольку там все настроено для работы с юзером

Пробежимся вообще по онлайн маркетам, за основу взял ламоду, но сильно ее упростил по заданию. У нас есть пользователь, работа с товарами осуществляться будет через ручки, связанные с товарами. Сам товар часто бывает с каким-то предложением (скидкой), соответственно надо продумать еще момент со связями. Далее я просто приложу скрин с excalidraw , где просто примерно раскидал модели



```
@Table({ Show usages  ko.ponomarev
    tableName: 'Sale'
})
export class Sale extends Model<Sale> {
    @PrimaryKey
    @AutoIncrement
    @Column
    id: number;

    @AllowNull(false)
    @Column
    title: string;

    @Column(DataType.TEXT)
    description: string;

    @Column(DataType.DATE)
    startDate: Date;

    @Column(DataType.DATE)
    endDate: Date;

    @HasMany(() => Product)
    products: Product[];
}
```



```
6  ✓ @Table({ Show usages  ko.ponomarev
7  ↗  tableName: 'Products'
8  })
9  ✓ export class Product extends Model<Product> {
10     @PrimaryKey
11     @AutoIncrement
12     @Column
13     ↗  id: number;
14
15     @AllowNull(false)
16     @Column
17     ↗  name: string;
18
19     @AllowNull(false)
20     @Column
21     price: number;
22
23     @Column
24     stockQuantity: number;
25
26     @ForeignKey(() => Discount)
27     @Column
28     discountId?: number;
29
30     @ForeignKey(() => Sale)
31     ⚡ @Column
32     saleId?: number;
33
34     @HasMany(() => Order)
35     orders: Order[];
36 }
37
```

После чего начал делать логику

```
> import ...

export class DiscountService { Show usages  ko.ponomarev

  async createDiscount(discountData: any): Promise<Discount> { Show usages  ko.ponomarev
    const discount = await Discount.create(discountData);
    return discount;
  }

  async getDiscountById(discountId: number): Promise<Discount | null> { Show usages  ko.ponomarev
    const discount = await Discount.findByPk(discountId, {
      include: [Product]
    });
    return discount;
  }

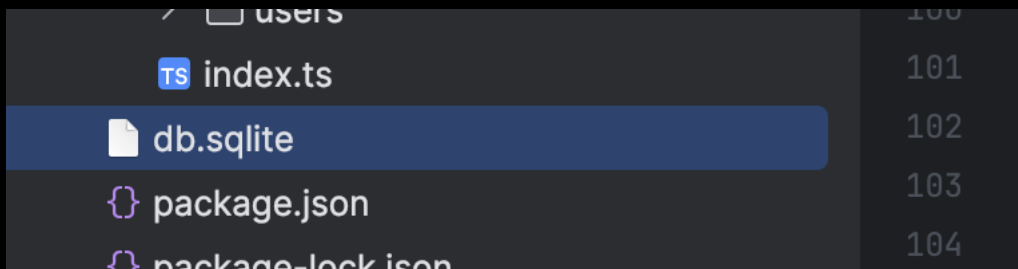
  async getAllDiscounts(): Promise<Discount[]> { Show usages  ko.ponomarev
    const discounts = await Discount.findAll({
      include: [Product]
    });
    return discounts;
  }

  async updateDiscount(discountId: number, discountData: any): Promise<Discount | null> { Show usages  ko.ponomarev
    const discount = await Discount.findByPk(discountId);
    if (discount) {
      await discount.update(discountData);
    }
    return discount;
  }

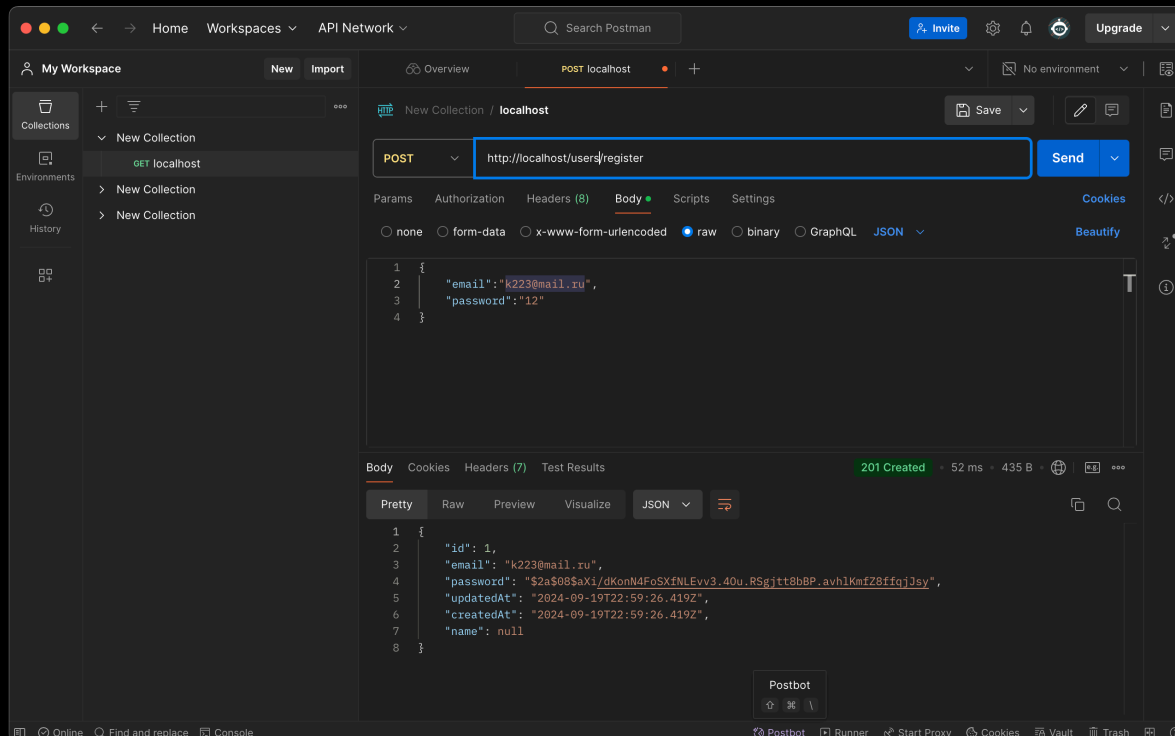
  async deleteDiscount(discountId: number): Promise<boolean> { Show usages  ko.ponomarev
    const deleted = await Discount.destroy({
      where: { id: discountId }
    });
    return deleted > 0;
  }
}
```

Делал дальше все как у себя, api (service) -> controller (repository) -> route

Потом это все накрутил на роуты и сделал базу sqlite. Постгрес посчитал лишним, поскольку решил с ним помучаться в докере (легче ставить, не надо париться, запускать локально)



Ну а далее просто запускаем и убеждаемся что все работает



Вывод:

Научился базово проектировать api сервисы, поработал все с тем же express + sequelize + sqlite, подготовил основу для будущих лр, где надо разбивать на микросервисы

