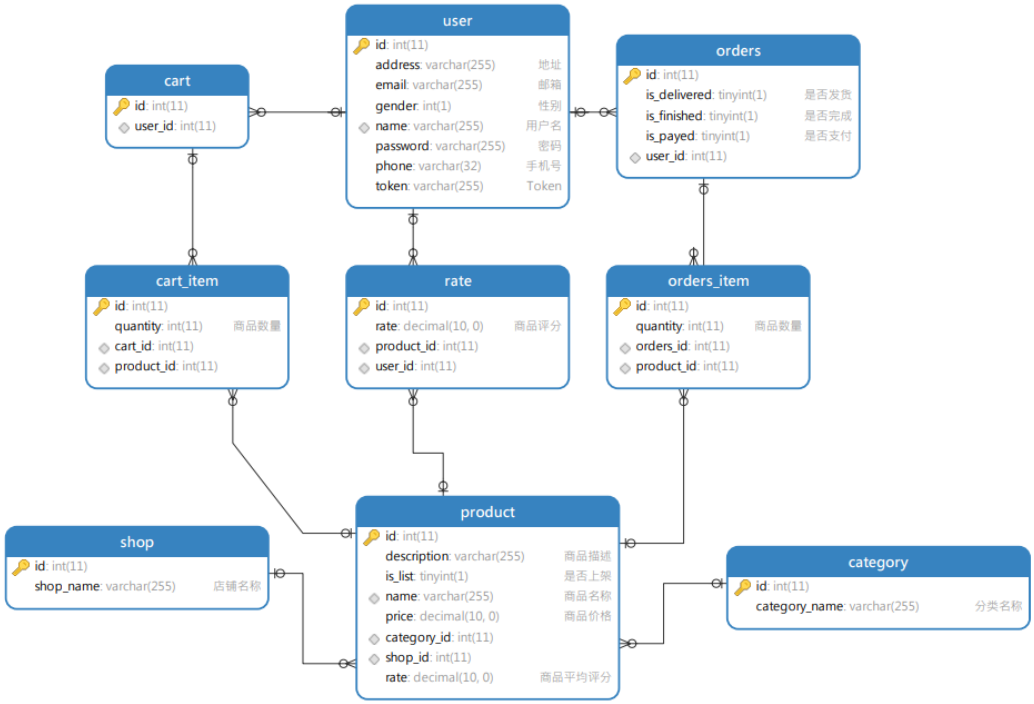
项目亮点

DB设计

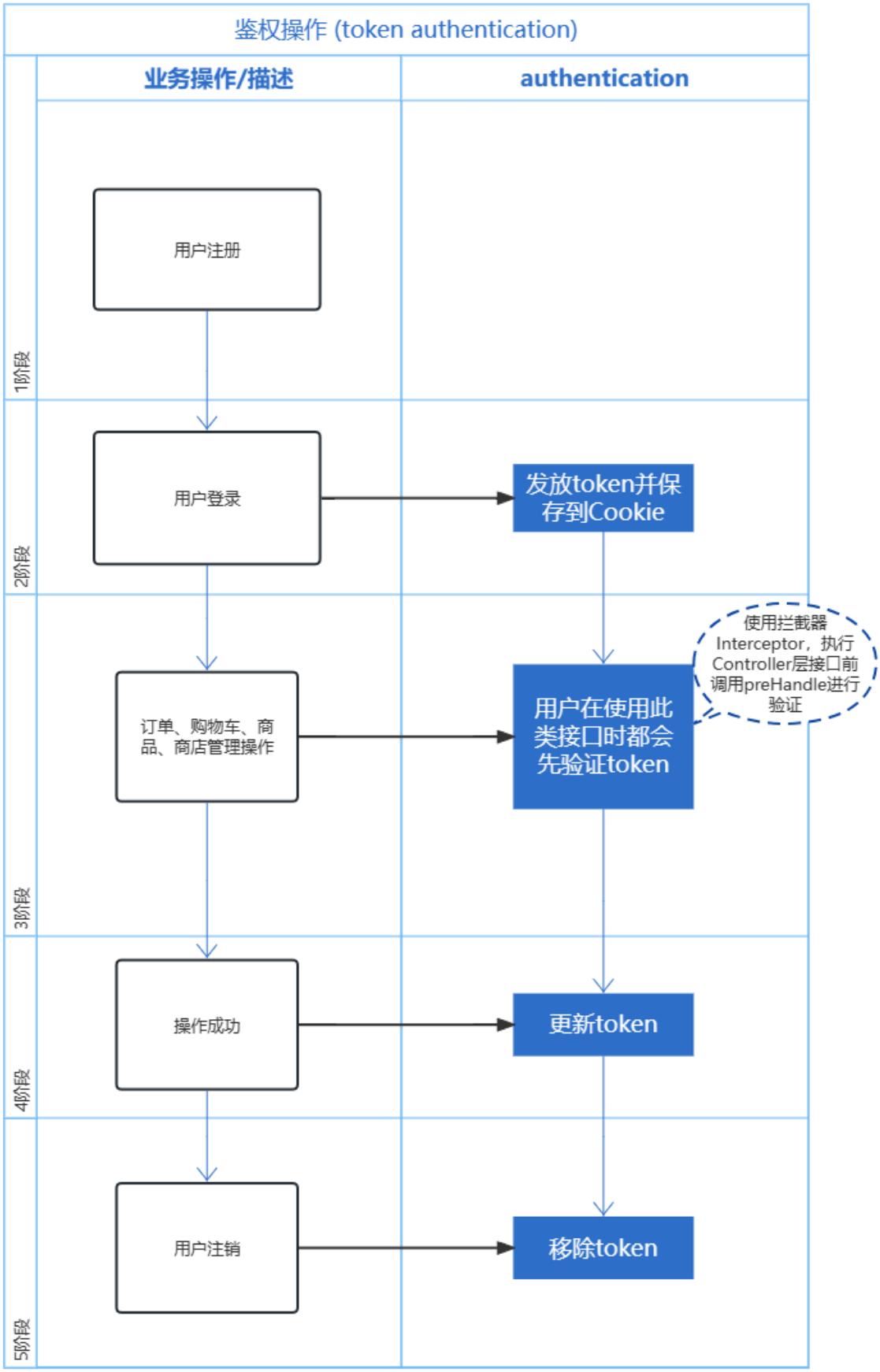


已实现功能

用户

- 用户注册
- 登录
- 访问令牌刷新

鉴权操作



- 鉴权操作流程
  用户在进行登录操作后，后端server会在cookie里保存一个token，接下来用户在调用订单管理、购物车管理、商品管理、商家管理等模块的功能之前，都会先使用Interceptor拦截当前请求，并验证当前token是否合法且未过期，如果当前token合法，则放行，允许用户进行这些功能模块的操作，操作成功后会更新token；用户注销（退出登录状态）后，同步移除Cookie数据里的token。

商品管理

基础功能

- 获取商品列表
- 根据商品名查询详情
- 搜索
- 评价

附加功能

- 根据id查询详情
- 根据店家id查询详情
- 增加商品
- 根据ID删除商品
- 根据商品名删除商品
- 根据ID修改商品信息
- 商品上架
- 商品下架
- 删除所有商品

购物车管理

基础功能

- 添加商品 **通过用户ID，商品ID和数量添加商品**
- 删除商品 **通过用户ID和商品ID删除商品 通过用户ID和商家ID删除商品 通过用户ID和分类ID删除商品**
- 更新商品数量 **更改商品数量**
- 获取详情 **通过用户ID展示购物车商品详情 & 通过用户ID、商店ID、商品类别ID展示购物车商品详情**

附加功能

- 通过用户ID查找对应的购物车item
- 通过用户Id和商家Id查找购物车

订单管理

- 订单创建
- 查询订单列表 **通过用户ID**
- 查询订单商品 **通过订单ID**
- 详细查询
- 删除订单 **通过订单ID**
- 删除所有订单
- 订单设为已支付 **通过订单ID**
- 订单设为已发货 **通过订单ID**
- 订单设为已完成 **通过订单ID**

其余附加功能

商家管理

- 展示商家列表
- 查询商家信息
- 删除商家信息
- 修改商家信息
- 插入商家信息
- 删除所有商家

分类管理

- 展示分类列表
- 查询分类信息
- 删除分类信息
- 修改分类信息
- 插入分类信息
- 删除所有分类信息

用户管理

- 查询所有用户
- 查询单个用户
- 创建新用户
- 删除用户
- 删除所有用户

代码目录介绍

- src
  - main
    - java
      - com/example/ex3_2_back
        - configuration Cors、Swagger等的配置， Spring容器可以使用这些方法来注入Bean
        - controller 控制层
        - domain domain层，定义相关接口的数据结构
        - entity 实体类，此处使用JPA可直接在数据库生成对应表
        - exception ExceptionHandler返回相关异常信息
        - interceptor 拦截器，用于token验证
        - repository 与数据库相连的持久层，部分操作需要使用JPQL语句
        - service 一些复用程度高的功能需要在这实现
        - utils 后端运行入口
      - resources .相关配置文件yaml
  - test
    - java/com/example/ex3_2_back
      - repository 持久层相关方法的测试

- utils 验证是否能够成功加载 Spring 上下文的测试

JPA & JPQL

JPA

Java Persistence API，可以通过注解或者XML描述【对象-关系表】之间的映射关系，并将实体对象持久化到数据库中。 一些注释略解（如下👇）

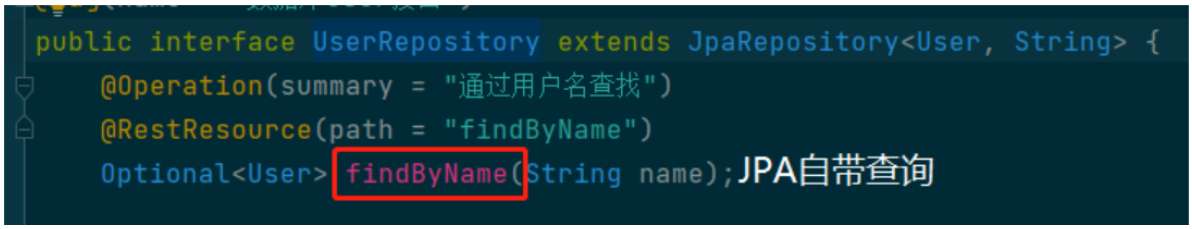

```java
package com.example.ex3_2_back.entity;
import ...


@Builder
@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@ToString
@Entity
@Table(name = "t_User")
@Schema(description = "User")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    Integer id;
    @Column(unique = true, nullable = false)
    @Schema(defaultValue = "test")
    String name;
    @Schema(defaultValue = "test")
    String password;
    @Builder.Default
    Gender gender = Gender.Unknown;
    @Email
    @Schema(defaultValue = "test@qq.com")
    String email;
    @Schema(defaultValue = "111111111111")
    String phone;
    @Schema(defaultValue = "test")
    String address;
}
```

还有其他许多注解对应不同功能，需根据实际情况灵活选用。

JPQL

JpaRepository自带的可直接使用的简单查询（如下👇）有时候满足不了我们的需求



我们经常需要 **@Query结合jpql语句进行查询等操作**

一个更新数据库值的一个简单示例（如下👇）

```
@Transactional
@Modifying
@Query("UPDATE Movie m SET m.seenCount = m.seenCount + 1 WHERE m.id = :movieId")
@RestResource(path = "incrementSeenCount")
void incrementSeenCount(Integer movieId);
```

还有其他许多注解对应不同功能，需根据实际情况灵活选用

团队合作经验

1.明确定义API目标和用途：在开始设计API之前，确保整个团队对API的目标和用途有清晰的共识。这有助于确保设计和开发过程的一致性，并使团队成员在迭代和决策时有一个明确的方向。

2.制定规范和设计原则：定义一套API设计规范和设计原则，以确保团队成员在设计API时遵循一致的方法。这可以包括命名约定、数据结构规范、错误处理、版本控制等方面的指导原则。

3.拥抱开放沟通：建立开放和透明的沟通渠道，鼓励团队成员共享想法、问题和进展。定期召开会议、使用在线协作工具、建立团队聊天群等方式可以促进信息的交流和共享。

4.分工合作：根据团队成员的专长和技能，进行合理的任务分工。确保每个成员在项目中承担适合其能力和兴趣的角色，同时促进协作和知识共享。

5.版本控制和文档管理：使用版本控制系统（如Git）对API代码进行管理，以便团队成员可以协同工作、合并代码和追踪变更。同时，建立完善的API文档，包括清晰的使用说明、示例代码和错误处理信息，以便其他开发人员能够轻松理解和使用API。

个人贡献

姓名	贡献
贺思超	DB设计，框架搭建，整体API设计，视频制作，测试文档
陈增耀	鉴权，用户管理，规范API设计，测试文档，设计文档
韩熔	订单管理，团队合作计划，项目总结报告，设计文档
李易达	购物车管理
王俊铭	商品管理