文档Lab04-体系结构设计文档

文档作者

主要编写者：史创屹、杨枫、刘存玺

其他编写者：董天诺

文档修改历史

修改人员	日期	修改原因	版本号
刘存玺	2024-4-17	根据已经开发的系统完善文档5.1 5.2	
史创屹	2024-4-17	根据已经开发的系统完善文档1到4	
杨枫	2024-4-17	根据已经开发的系统完善文档5.3及之后所有的内容	

1. 引言

1.1 编制目的

- 本报告详细完成对 **蓝鲸网购** 的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。
- 本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

1.2 词汇表

词汇名称	词汇含义	备注
ServiceImpl	Service Implementation	对Service接口的具体实现
oss	Object Storage Service	对象存储服务

1.3 参考资料

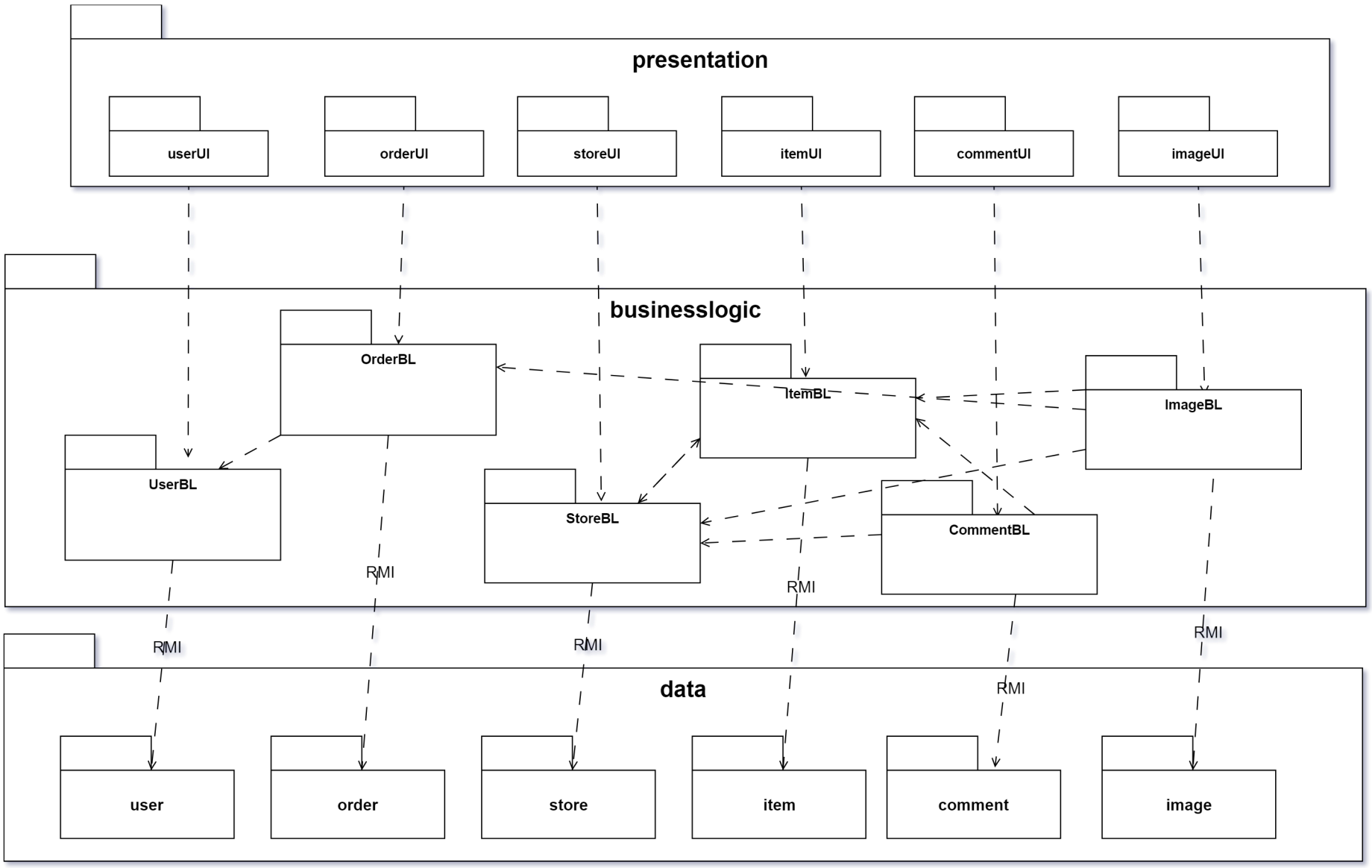
- 丁二玉 刘钦《软件工程与计算 卷2 软件开发的技术基础 》

2. 产品概述

- 参考用例文档以及需求规格说明书中对系统的概括描述

3. 逻辑视角

- 选用分层体系结构风格，将系统分为3层（展示层、业务层、数据层）。
- 软件体系结构逻辑设计方案



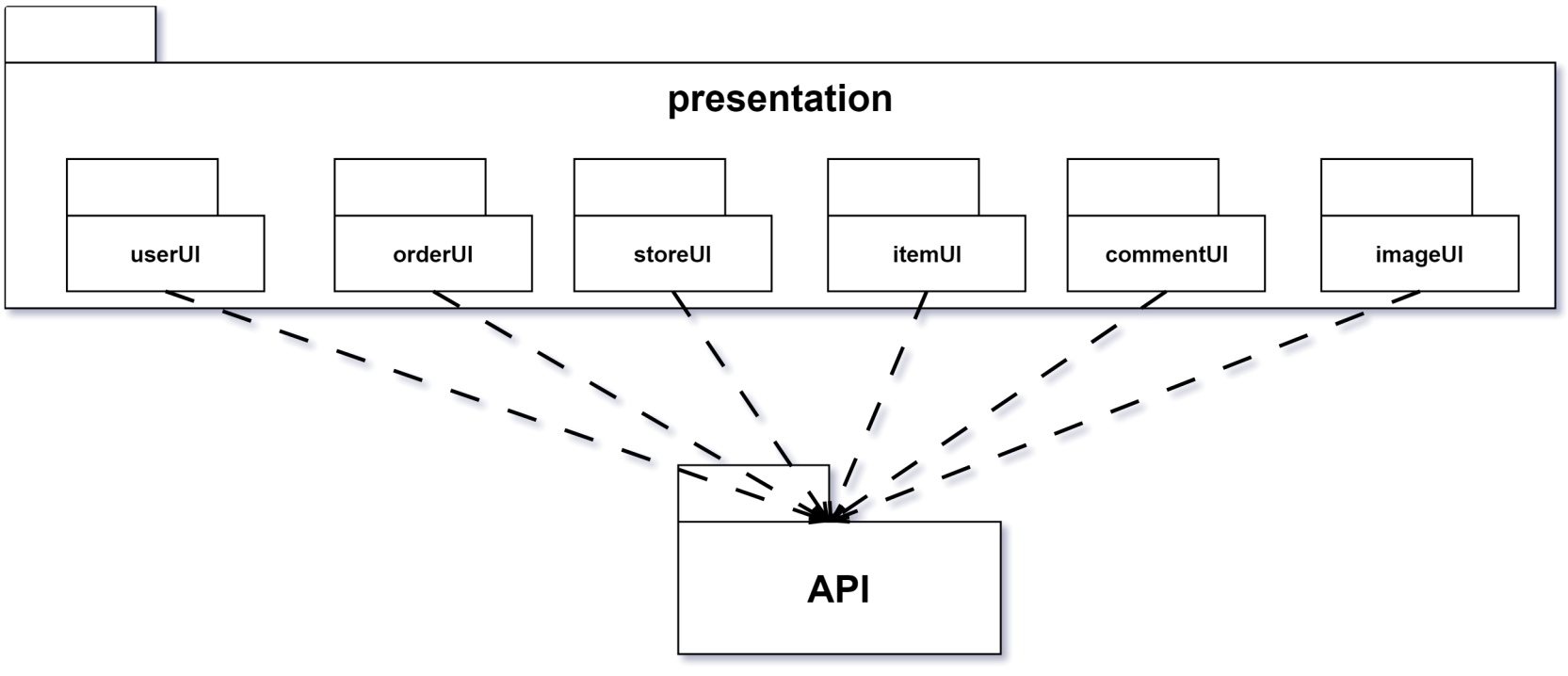
组合视角

4.1 开发包图

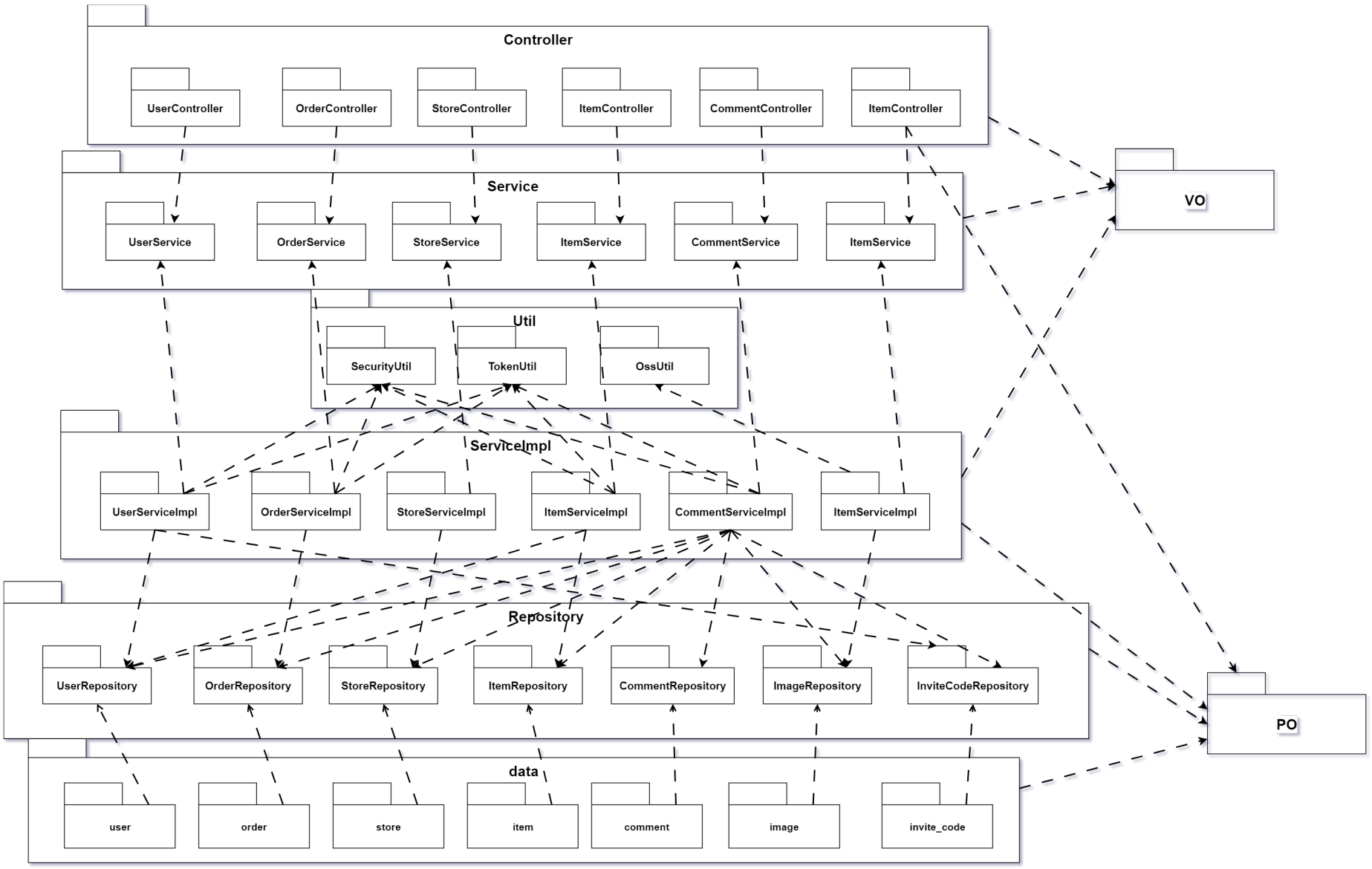
- 表1 开发包设计

开发(物理)包	依赖的其他开发包
user(UI)	api,router
UserController	UserService,vo
UserService	vo
UserServiceImpl	UserService,po,vo,UserRepository,InviteCodeRepository,SecurityUtil,TokenUtil
UserRepository	po
user(data)	/
store(UI)	api,router
StoreController	StoreService,vo
StoreService	vo
StoreServiceImpl	StoreService,po,vo,StoreRepository
StoreRepository	po
store(data)	/
order(UI)	api,router
OrderController	OrderService,vo
OrderService	vo
OrderServiceImpl	OrderService,po,vo,OrderRepository,SecurityUtil,TokenUtil
OrderRepository	po
order(data)	/
item(UI)	api,router
ItemController	ItemService,po,vo
ItemService	po,vo
ItemServiceImpl	ItemService,po,vo,UserRepository,ItemRepository,SecurityUtil,TokenUtil
ItemRepository	po
item(data)	/
image(UI)	api,router
ImageController	ImageService,vo
ImageService	po,vo
ImageServiceImpl	ImageService,po,vo,ImageRepository,OssUtil
ImageRepository	po
image(data)	/
comment(UI)	api,router
CommentController	CommentService,vo
CommentService	vo
CommentServiceImpl	CommentService,po,vo,repository,SecurityUtil,TokenUtil
CommentRepository	po
comment(data)	/
InviteCodeRepository	po
api	/
router	/
po	/
vo	/
OssUtil	/
SecurityUtil	/
TokenUtil	/
enums	/
exception	/

- 客户端开发包图

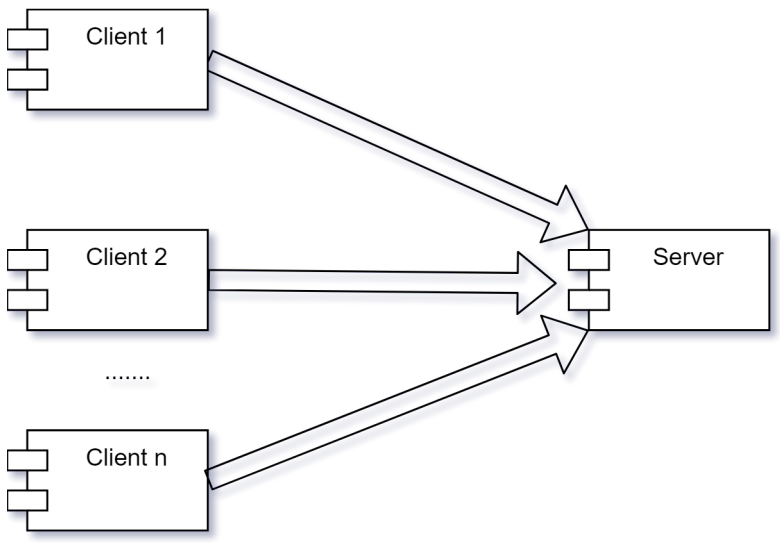


- 服务端开发包图



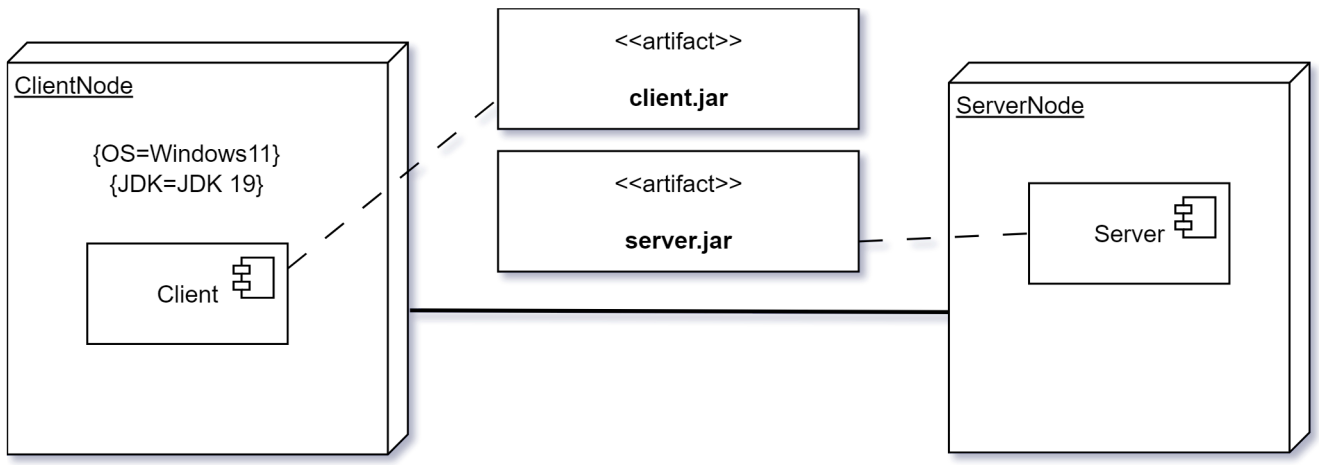
4.2 运行时进程

- 在蓝鲸商城网购平台系统中，会有多个客户端进程和一个服务端进程，如下图所示



4.3 物理部署

- 部署图



接口视角

5.1 模块的职责

- 模块视图

- 表2 客户端各层职责

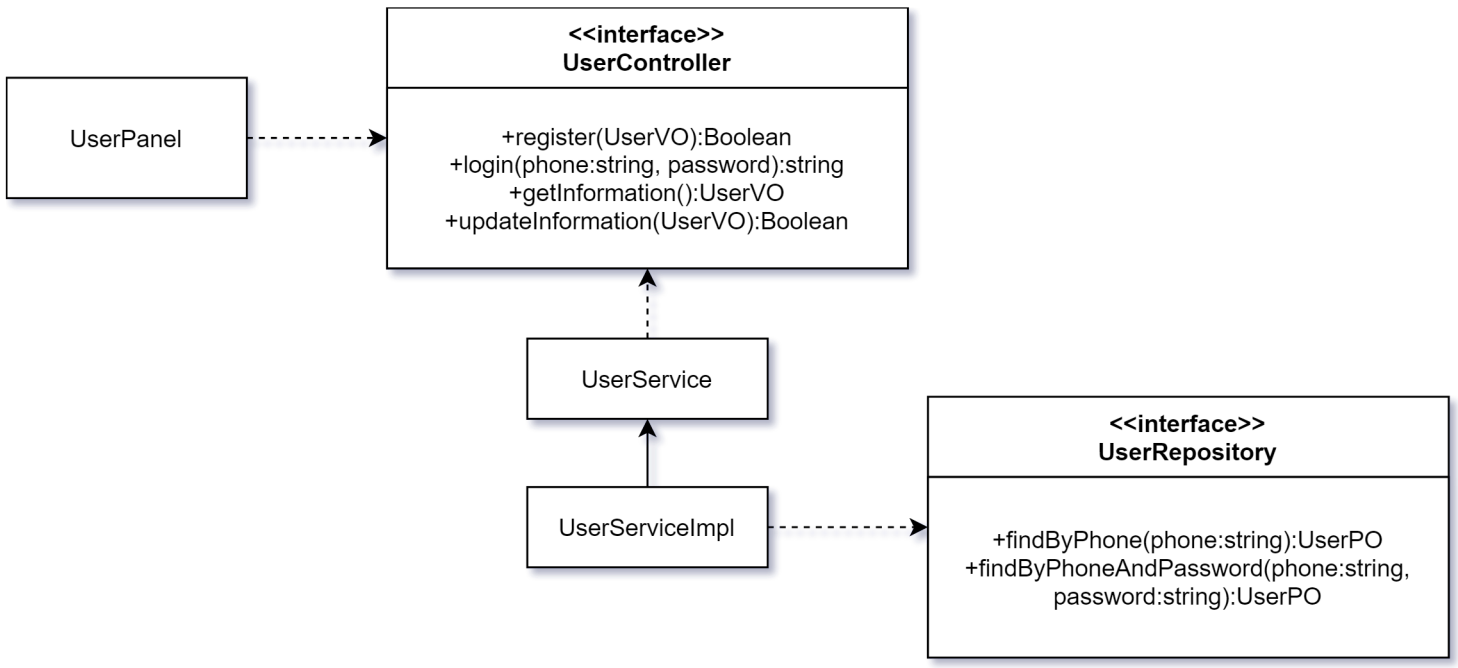
层	职责
用户界面层	基于web的网购商城客户端界面
业务逻辑层	对网络模块接受的数据与用户界面的输入进行响应并进行业务逻辑处理
客户端网络模块	发送网络请求

- 表3 服务端各层职责

层	职责
接口层Controller	负责响应客户端传来的网络请求
业务逻辑层Service/ServiceImpl	负责接收并处理来自接口层的请求
数据接口层Repository	负责数据的持久化和数据访问接口

- 表4 层之间调用接口，下表为一个例子

接口	服务调用方	服务提供方
UserController	客户端网络模块	业务逻辑层
UserRepository	业务逻辑层	数据接口层

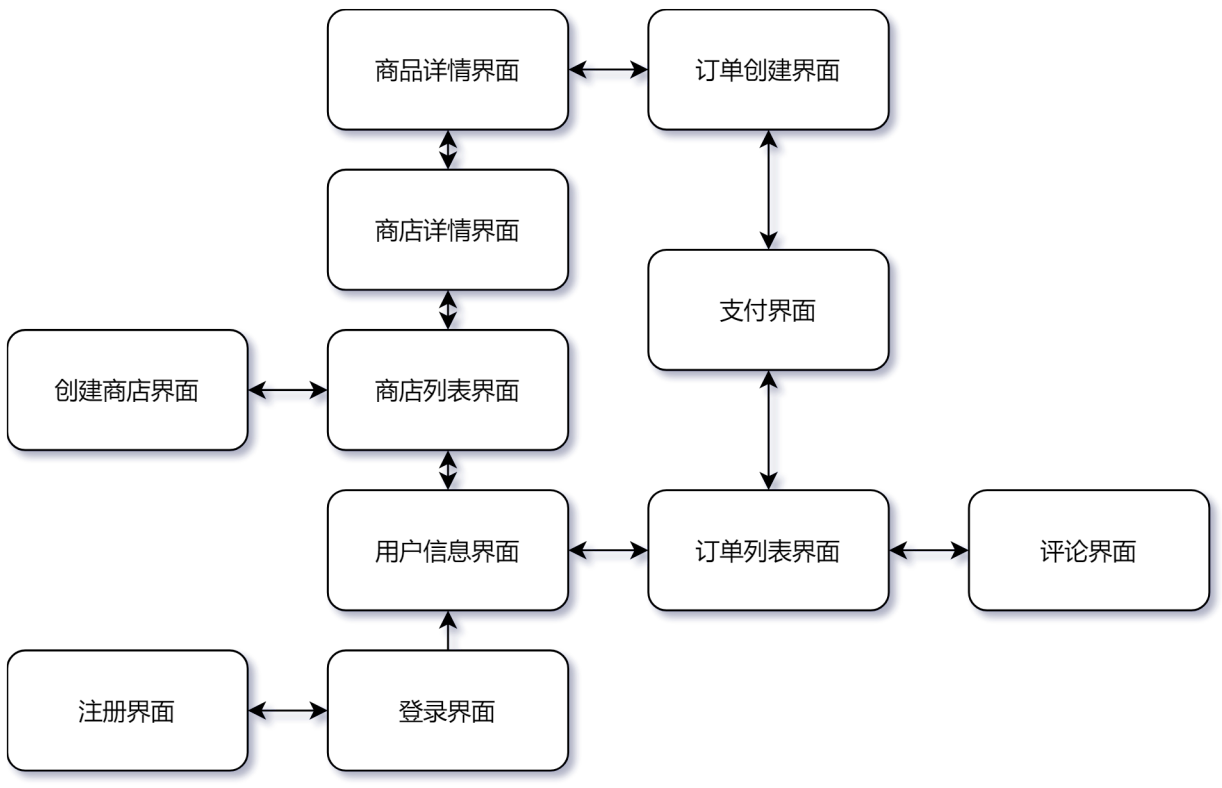


5.2 用户界面层的分解

根据需求，系统存在11个用户界面：

- 登录界面
- 注册界面
- 用户信息界面
- 商店列表界面
- 商店详情界面
- 商品详情界面
- 订单创建界面
- 支付界面
- 订单列表界面
- 评论界面
- 创建商店界面

这些界面的跳转如下图所示：



5.2.1 职责

模块	职责
用户界面层	负责界面的显示和界面的跳转，以及与用户的交互

5.2.2 接口规范

用户界面层模块的接口规范如下表所示。

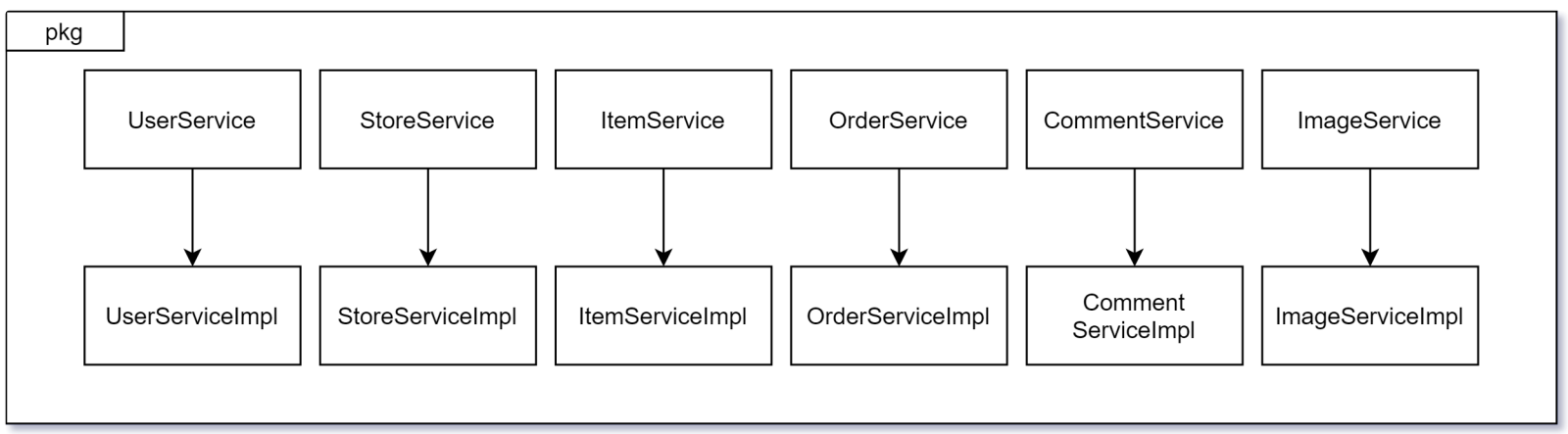
名称	语法	前置条件	后置条件
用户界面层	init(args:String[])	无	显示Frame

用户界面层需要的服务接口如下表所示。

服务名	服务
service.*Service	每个节目对应的业务逻辑接口

5.3 业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，UserService对象负责处理登录界面的业务逻辑；StoreService对象负责处理商店页面的业务逻辑；ItemService对象负责处理商品页面的业务逻辑；OrderService对象负责处理订单页面的业务逻辑；CommentService对象负责处理评论页面的业务逻辑。业务逻辑层的设计图如下所示：



5.3.1 职责

模块	职责
UserService	负责实现对应与登陆界面所需要的服务
StoreService	负责实现商店页面所需要的服务
ItemService	负责实现商品页面所需要的服务
OrderService	负责实现订单页面所需要的服务
CommentService	负责实现评论界面所需要的服务
ImageService	负责实现所有页面的图片服务

5.3.2 接口规范

5.3.2.1 UserService模块的接口规范

提供的服务（供接口）

名称	语法	前置条件	后置条件
UserService.register	public Boolean register(UserVO userVO)	注册信息符合输入规则	新建一个持久化对象User
UserService.login	public String login(String phone,String password)	电话和密码信息符合输入规则	查找是否存在对应的User，根据输入的密码返回登录令牌
UserService.getInformation	public UserVO getInformation();	无	调用数据库返回用户VO
UserService.updateInformation	public Boolean updateInformation(UserVO userVO);	用户更新信息符合规则	更新数据库中持久对象的内容
UserService.getInviCode	public String getInviteCode(RoleEnum role);	role是合法角色	返回一个邀请码

需要的服务（需接口）

服务名	服务
UserRepository.findByPhone(String phone)	根据phone查找单一持久化对象
UserRepository.save(User user)	保存单一持久化对象
UserRepository.findByPhoneAndPassword(String phone,String password)	根据phone和password查找单一持久化对象
InviteRepository.findByCode(String code)	根据code查找单一持久对象
InviteRepository.delete(String code)	删除单一持久化对象
TokenUtil.getToken(User user)	获取用户令牌
SecurityUtil.getCurrentUser()	获取当前登录用户信息
InviteCodeRepository.save(InviteCode inviteCode)	保存单一持久化对象

5.3.2.2 StoreService模块的接口规范

提供的接口(供接口)

名称	语法	前置条件	后置条件
StoreService.getStores	public List getStores()	无	返回所有商店VO
StoreService.getStore	public StoreVO getStore(int storeId)	提供的商店id合法有效	返回对应商店VO
StoreService.createStore	public Integer createStore(StoreVO storeVO)	输入商店信息符合规则	创建一个持久化商店对象
StoreService.deleteStore	public boolean deleteStore(int storeId)	提供的商店id合法有效	删除对应的持久化商店对象

需要的接口(需接口)

服务名	服务
StoreRepository.findAll()	查找Store表中所有持久化对象
StoreRepository.findById(int storeId)	根据商店id查找单一持久化对象
StoreRepository.findByName(String name)	根据商店名称查找单一持久化对象
StoreRepository.save(Store store)	保存单一持久化对象
StoreRepository.delete(Store store)	删除单一持久化对象

5.3.2.3 Item模块的接口规范

提供的接口(供接口)

名称	语法	前置条件	后置条件
ItemService.appendItem	public Integer appendItem(ItemVO itemVO)	商品信息符合规则	创建持久化商品对象并返回商品id
ItemService.getItemByItemId	public ItemVO getItemById(Integer itemId)	输入的商品id合法有效	返回对应商品VO
ItemService.getItemListByStoreId	public List getItemListByStoreId(Integer storeId)	输入的商店id合法有效	返回该商店所有商品VO
ItemService.updateInventory	public Boolean updateInventory(Integer itemId, Integer inventory)	输入的商品id与库存数量合法有效	更新持久化对象信息
ItemService.getItemCategories	public List getItemCategories()	无	返回商品分类的枚举类信息

需要的接口(需接口)

服务名	服务
ItemRepository.save(item)	保存单一持久化对象
ItemRepository.findById(int id)	根据商品id查找单一持久化对象
ItemRepository.findByStoreId(int storeId)	根据商店id查找多个持久化对象
SecurityUtil.getCurrentUser()	获取当前用户信息

5.3.2.4 Order模块的接口规范

提供的接口(供接口)

名称	语法	前置条件	后置条件
OrderService.getById	public OrderVO getById(int orderId)	输入的订单id合法有效	返回对应订单的VO
OrderService.getAll	public List getAll()	无	返回所有订单的VO
OrderService.getByUserId	public List getByUserId()	无	返回当前用户所有订单VO
OrderService..getByStoreId	public List getByStoreId()	无	返回当前用户所属商店的所有订单VO
OrderService.createOrder	public Integer createOrder(OrderVO orderVO)	输入的订单信息合法有效	创建一个单一订单持久化对象并返回订单id
OrderService.deleteOrder	public Boolean deleteOrder(int orderId)	输入的订单id合法有效	删除单一订单持久化对象
OrderService.purchaseOrder	public Boolean purchaseOrder(int orderId)	输入的订单id合法有效且订单状态为未付款	更新单一订单持久化对象状态为未发货
OrderService.sendOrder	public Boolean sendOrder(int orderId)	输入的订单id合法有效且状态为未发货	更新单一订单持久化对象状态为已发货
OrderService.receiveOrder	public Boolean receiveOrder(int orderId)	输入的订单id合法有效且订单状态为未收货	更新单一订单持久化对象状态为已收货
OrderService.commentOrder	public Boolean commentOrder(int orderId)	输入的订单id合法有效且订单状态为未发货	更新单一订单持久化对象状态为已评论

需要的接口(需接口)

服务名	服务
OrderRepository.findById(int id)	根据订单id查找单一持久化对象
OrderRepository.findAll()	查找所有订单持久化对象
OrderRepository.findByUserId(int userId)	根据用户id查找多个持久化对象
OrderRepository.findByStoreId(int storeId)	根据商店id查找多个持久化对象
OrderRepository.save(Order order)	保存单一持久化对象
OrderRepository.delete(Order order)	删除单一持久化对象
SecurityUtil.getCurrentUser()	获取当前用户信息

5.3.2.5 Comment模块的接口规范

提供的接口(供接口)

名称	语法	前置条件	后置条件
CommentService.getCommentById	public CommentVO getCommentById(Integer commentId)	输入的id是合法有效的	返回对应的评论VO
CommentService.getCommentByItemAndCount	public List getCommentByItemIdAndCount(Integer itemId, Integer startInd)	输入的商品id和index是合法有效的	返回对应商品对应数量的评论VO
CommentService.getCommentByItemId	public Integer getCommentCountByItemId(Integer itemId)	输入的商品id是合法有效的	返回对应商品的所有评论VO
CommentService.appendComment	public Integer appendComment(CommentVO commentVO)	输入的评论信息是合法有效的	创建持久化对象,持久化更新对应商品及商店的评分,并返回该评论的id
CommentService.updateItemRating	private void updateItemRating(Integer itemId, float rating)	输入的商品id以及新增评价的评分是合法有效的	持久化更新该商品评分
CommentService.updateStoreRating	private void updateStoreRating(Integer itemId, float rating)	输入的商店id以及新增评价的评分是合法有效的	持久化更新该商店评分

需要的接口(需接口)

服务名	服务
CommentRepository.findById(int id)	根据评论id查找单一持久化对象
CommentRepository.findByItemId(int ItemId)	根据商品id查找多个持久化对象
CommentRepository.save(Comment comment)	保存单一持久化对象
SecurityUtil.getCurrentUser()	查询当前用户信息
ItemRepository.findById(int id)	根据商品id查找单一持久化对象
ItemRepository.findByStoreId(int id)	根据商店id查找多个持久化对象
ItemRepository.save(Item item)	保存单一持久化对象
StoreRepository(Store store)	保存单一持久化对象

5.3.2.6 Image模块的接口规范

提供的接口(供接口)

名称	语法	前置条件	后置条件
ImageService.upload	public String upload(MultipartFile file, ImageType type, int parentId, int ind)	输入的图片信息有效	将图片上传至oss并且更新image数据库
ImageService.delete	public Boolean delete(ImageType type, int parentId)	要删除的图片信息存在	同时删除oss和数据库中的图片信息
ImageService.download	public List download(ImageType type, int parentId)	要下载的图片信息存在	读取数据库中图片的url并返回

需要的接口(需接口)

服务名	服务
OssUtil.upload(String objectName,InputStream inputStream)	上传图片到oss
OssUtil.delete(String objectUrl)	根据url删除oss中的图片
ImageService.findAllByTypeAndParentId(ImageType type,int parentId)	根据图片类型和父辈id查找多个持久化对象
ImageService.save(Image image)	保存单个持久化对象
ImageService.delete(Image image)	删除单个持久化对象

5.4 数据层的分解

数据层主要给业务逻辑层提供数据访问服务,包括对于持久化数据的增、删、改、查，各个业务逻辑需要的服务由其对应的Repository接口提供。

5.4.1 职责

模块	职责
JPARepository	持久化数据库的接口,提供增、删、改、查的服务
各模块Repository	继承自JPARepository，根据逻辑业务需要提供更加精细的增、删、改、查服务。

5.4.2 接口规范

提供的服务（供接口）

名称	语法	前置条件	后置条件
UserRepository.save	void save(User user)	无	若数据库无该PO,新建一个,否则更新该PO
UserRepository.delete	void delete(User user)	数据库存在该PO	删除一个PO
UserRepository.findByPhone	User findByPhone(String phone)	无	按电话号码查询返回相应的UserPO
UserRepository.findByPhoneAndPassword	User findByPhoneAndPassword(String phone, String password)	无	按电话号码和密码查询返回相应的UserPO
UserRepository.findAll	List findAll()	无	返回所有UserPO

信息视角

6.1 描述数据持久化对象(PO)

系统的PO类就是对应的相关的实体类。在此只做简单的介绍

- UserPO类包含用户的id、名称、电话号码、密码、创建时间、所属商店id、地址、角色。定义如下:

```
public class User {

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
```

```
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name = "name")
    private String name;

    @Basic
    @Column(name = "phone")
    private String phone;

    @Basic
    @Column(name = "password")
    private String password;

    //必须注意，在Java中用驼峰，在MySQL字段中用连字符_
    @Basic
    @Column(name = "create_time")
    private Date createTime;

    @Basic
    @Column(name = "store_id")
    private Integer storeId;

    @Basic
    @Column(name = "address")
    private String address;

    @Basic
    @Column(name = "role")
    @Enumerated(EnumType.STRING)
    private RoleEnum role;

    public UserVO toVO(){
        UserVO userVO=new UserVO();
        userVO.setId(this.id);
        userVO.setAddress(this.address);
        userVO.setName(this.name);
        userVO.setRole(this.role);
        userVO.setStoreId(this.storeId);
        userVO.setPhone(this.phone);
        userVO.setPassword(this.password);
        userVO.setCreateTime(this.createTime);
        return userVO;
    }
}
```

- StorePO类包含商店的id、名称、描述、评分。定义如下：

```
public class Store {
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name = "name")
    private String name;

    @Basic
    @Column(name = "description")
    private String description;

    @Basic
    @Column(name = "rating")
    private float rating;

    public StoreVO toVO(){
        StoreVO storeVO = new StoreVO();
        storeVO.setId(this.id);
        storeVO.setName(this.name);
        storeVO.setDescription(this.description);
        storeVO.setRating(this.rating);

        return storeVO;
    }
}
```

- OrderPO类包含订单的id、商品id、用户id、状态、配送、商店id、优惠券、价格、数量、创建时间、发货时间、签收时间。定义如下：

```
public class Order {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name = "item_id")
    private Integer itemId;

    @Basic
    @Column(name = "user_id")
    private Integer userId;

    @Basic
    @Column(name = "status")
    private OrderStatus status;

    @Basic
```

```
    @Column(name = "delivery")
    private String delivery;

    @Basic
    @Column(name = "store_id")
    private Integer storeId;

    @Basic
    @Column(name = "coupon")
    private Integer coupon;
    // TODO: 2021/6/1 优惠券

    @Basic
    @Column(name = "price")
    private Float price;

    @Basic
    @Column(name = "quantity")
    private Integer quantity;

    @Basic
    @Column(name = "create_time")
    private Date createTime;

    @Basic
    @Column(name = "send_time")
    private Date sendTime;

    @Basic
    @Column(name = "receive_time")
    private Date receiveTime;

    public OrderVO toVO(){
        OrderVO orderVO = new OrderVO();
        orderVO.setId(this.id);
        orderVO.setItemId(this.itemId);
        orderVO.setUserId(this.userId);
        orderVO.setStatus(this.status);
        orderVO.setDelivery(this.delivery);
        orderVO.setStoreId(this.storeId);
        orderVO.setPrice(this.price);
        orderVO.setQuantity(this.quantity);
        orderVO.setCreateTime(this.createTime);
        orderVO.setSendTime(this.sendTime);
        orderVO.setReceiveTime(this.receiveTime);
        return orderVO;
    }
}
```

- ItemPO类包含了商品的id、名称、描述、价格、评分、所属类别、所属商店id、库存。定义如下

```java
public class Item {

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name = "name")
    private String name;

    @Basic
    @Column(name = "description")
    private String description;

    @Basic
    @Column(name = "price")
    private float price;

    @Basic
    @Column(name = "rating")
    private float rating;

    @Basic
    @Column(name = "category")
    private CategoryEnum category;

    @Basic
    @Column(name = "store_id")
    private int storeId;

    @Basic
    @Column(name = "inventory")
    private int inventory;

    public ItemVO toVO(){
        ItemVO itemVO = new ItemVO();
        itemVO.setId(this.id);
        itemVO.setName(this.name);
        itemVO.setDescription(this.description);
        itemVO.setPrice(this.price);
        itemVO.setRating(this.rating);
        itemVO.setCategory(this.category);
        itemVO.setStoreId(this.storeId);
        itemVO.setInventory(this.inventory);
        return itemVO;
    }
}
```

- InviteCodePO类包含了邀请码的id、值、被邀请者角色。定义如下

```
public class InviteCode {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name = "code")
    private String code;

    @Basic
    @Column(name = "role")
    @Enumerated(EnumType.STRING)
    private RoleEnum role;

}
```

- CommentPO类包含了评价的id、内容、被评价商品id、评价者id、订单id、商品评分、评价创建时间。定义如下：

```
public class Comment {

    @GeneratedValue(strategy =  GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name ="content" )
    private String content;

    @Basic
    @Column(name = "itemId")
    private Integer itemId;

    @Basic
    @Column(name = "userId")
    private Integer userId;

    @Basic
    @Column(name = "orderId")
    private Integer orderId;

    @Basic
    @Column(name = "rating")
    private float rating;

    @Basic
    @Column(name = "createTime")
```

```
    private Date createTime;

    public CommentVO toVO(){
        CommentVO commentVO = new CommentVO();
        commentVO.setId(this.id);
        commentVO.setContent(this.content);
        commentVO.setUserId(this.userId);
        commentVO.setOrderId(this.orderId);
        commentVO.setItemId(this.itemId);
        commentVO.setRating(this.rating);
        commentVO.setCreateTime(this.createTime);
        return commentVO;

    }
```

- ImagePO类包含了图片的id、url、类型、所属父辈id、排序。定义如下：

```
public class Image {

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;

    @Basic
    @Column(name = "url")
    private String url;

    @Basic
    @Column(name = "type")
    @Enumerated(EnumType.STRING)
    private ImageType type;

    @Basic
    @Column(name = "parent_id")
    private Integer parentId;

    @Basic
    @Column(name = "ind")
    private Integer ind;

    public ImageVO toVO(){
        ImageVO imageVO = new ImageVO();
        imageVO.setId(this.id);
        imageVO.setUrl(this.url);
        imageVO.setParentId(this.parentId);
        imageVO.setInd(this.ind);
        return imageVO;
    }
```

```
}
```

6.2 数据库表

数据库中包含User表、Store表、Item表、Order表、Comment表、InviteCode表、Image表。