代码Lab03-订单模块

组员信息

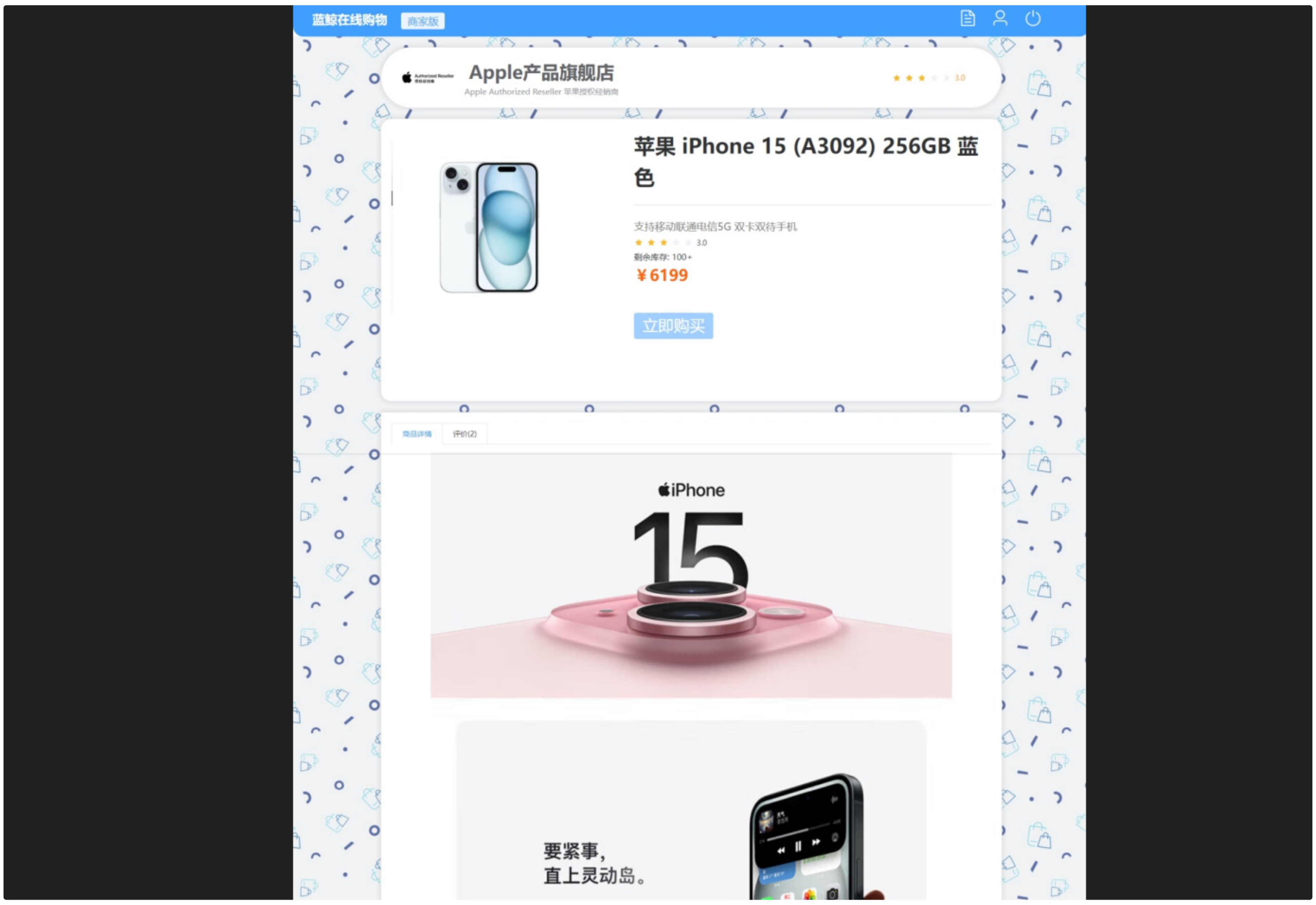
姓名	学号	电话
刘存玺	221850148	17318601136
董天诺	221250004	13264512313
史创屹	221250106	15385019512
杨枫	221250155	18976304216

前端具体实现

商品详细页面

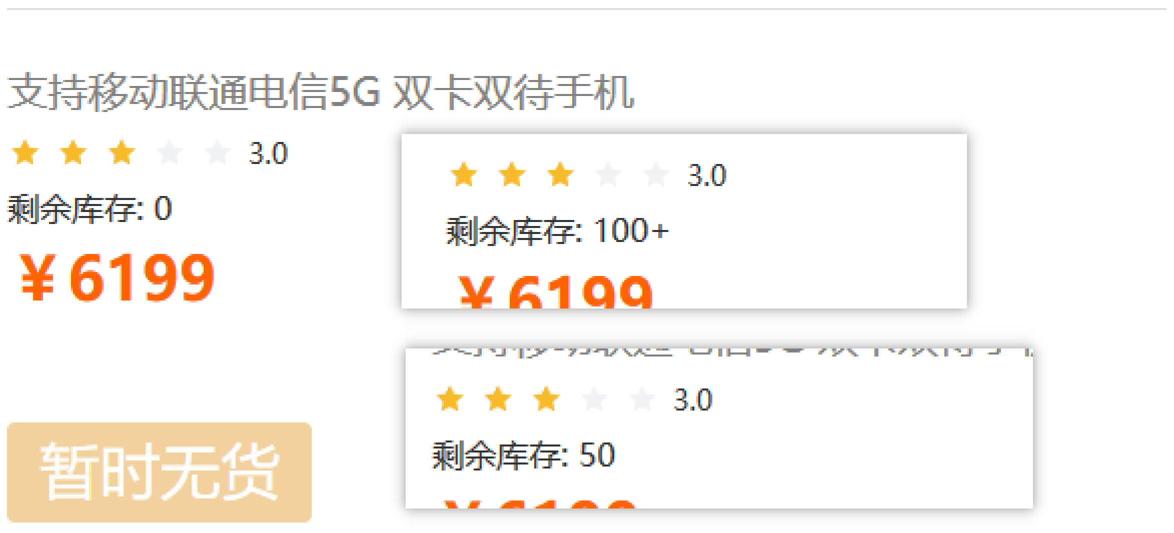
商品详情图

- 商品除了展示图之外，还可以有二级详情介绍图。
- 商品创建时可以上传详情介绍图



库存显示

- 商品拥有库存的显示
- 库存大于100时，显示为100+；库存小于100时，显示具体数字；库存为0时，显示为0，并且购买按钮变为无货按钮



- 使用v-if实现

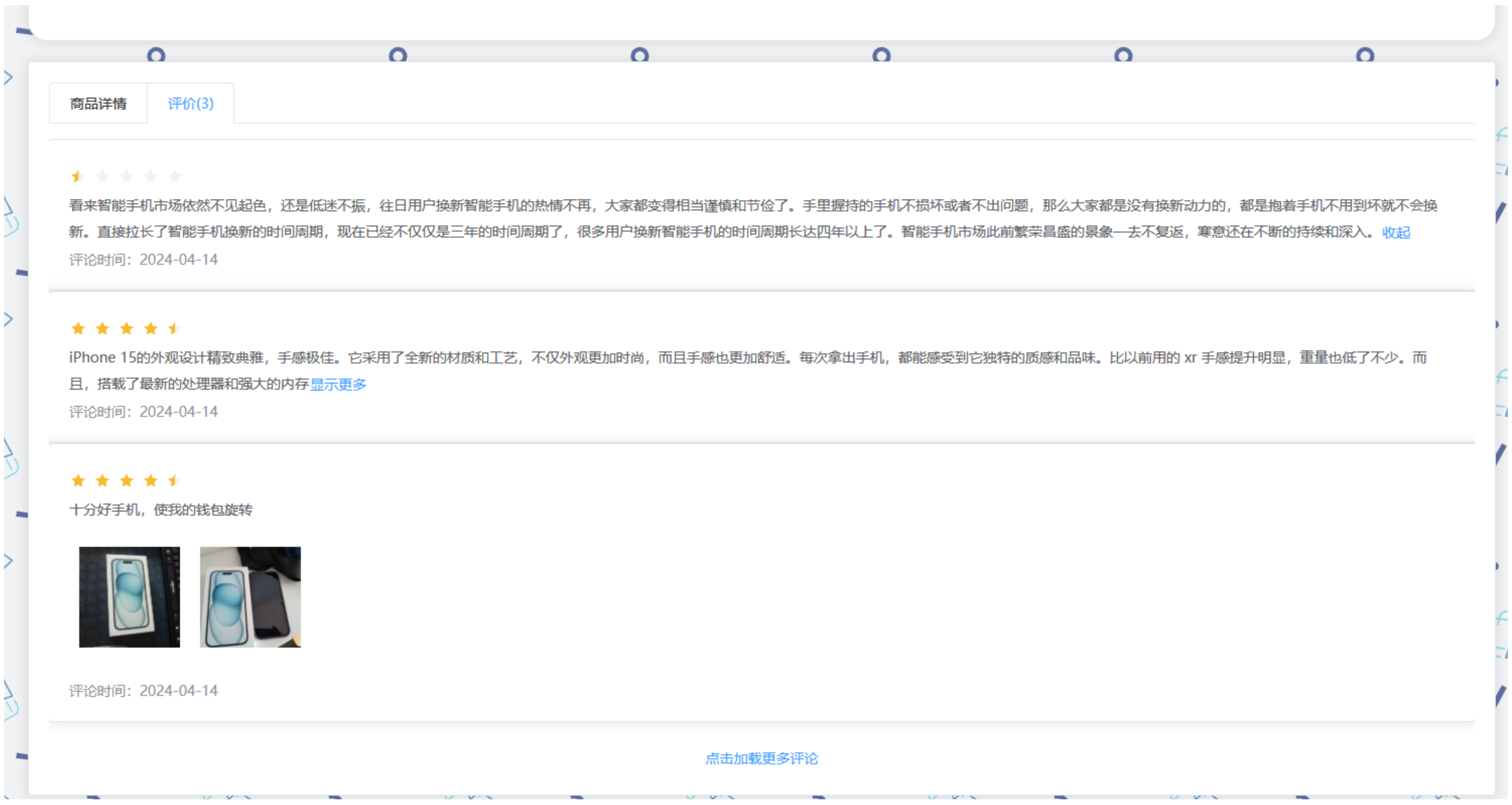
```
<el-row v-if="product.inventory > 100">剩余库存: 100+</el-row>
<el-row v-if="product.inventory ≤ 100">剩余库存: {{product.inventory}}</el-
row>
<el-row class="price-text">¥{{product.price}}</el-row>
<el-button type="warning" v-if="product.inventory ≤ 0" :disabled=true
 @click="handleToPurchase">暂时无货</el-button>
<el-button type="primary" v-if="product.inventory > 0" :disabled="role ≠
'CUSTOMER'" @click="handleToPurchase">立即购买</el-button>
```

购买权限

- **对应lab3文档的需求1**
- 只有顾客可以购买商品，其他Role在商品详细界面的购买按钮不可用

评论区

- **对应lab3文档的需求7**
- 商品的评论区也处于商品详细界面内。与商品详情介绍处于同一个标签页下，可以通过标签页头进行切换浏览。
- 众多评论采用懒加载，以应对评论数量巨大的情况。点击加载更多评论 可以实现评论的加载
  - 加载到底时，按钮文字会变为到底啦 并且不可继续点击
- 评论字数过多会进行折叠，可以点击显示更多 来进行展开。展开之后也可以点击收起 来关闭
- 评论的标签栏处会显示商品总评论个数（是总个数，不是当前懒加载了多少条），**对应lab3文档的需求6**



商品详情 评价(3)

看来智能手机市场依然不见起色，还是低迷不振，往日用户换新智能手机的热情不再，大家都变得相当谨慎和节俭了。手里握持的手机不损坏或者不出问题，那么大家都是没有换新动力的，都是抱着手机不用到坏就不会换新。直接拉长了智能手机换新的时间周期，现在已经不仅仅是三年的时间周期了，很多用户换新智能手机的时间周期长达四年以上了。智能手机市场此前繁荣昌盛的景象一去不复返，寒意还在不断的持续和深入。收起

评论时间：2024-04-14

iPhone 15的外观设计精致典雅，手感极佳。它采用了全新的材质和工艺，不仅外观更加时尚，而且手感也更加舒适。每次拿出手机，都能感受到它独特的质感和品味。比以前用的 xr 手感提升明显，重量也低了不少。而且，搭载了最新的处理器和强大的内存显示更多

评论时间：2024-04-14

十分好手机，使我的钱包旋转

评论时间：2024-04-14

点击加载更多评论

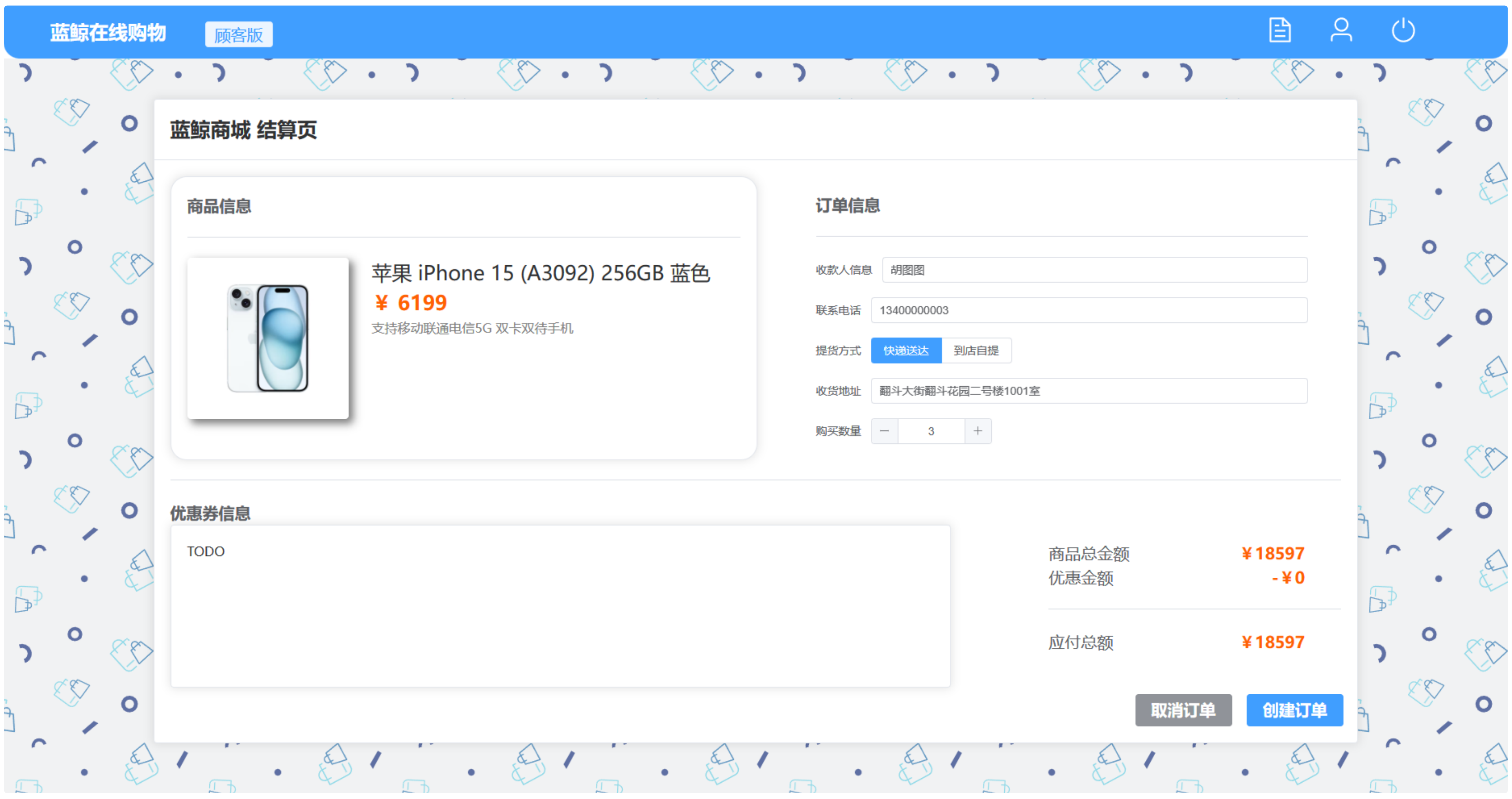
```
function handleLoadComment(){
  reqCommentList({
    itemId: parseInt(productId.value),
    startInd: commentList.value.length
  }).then(res => {
    if(res.data.result.length == 0){
        // 评论区到底了
      commentListEnd.value = true
    }
    else{
      commentList.value.push(...res.data.result)
    }
  })
}
```

```
<el-tab-pane :label="'评价(' + commentCount + ')'">
  <el-container style="display: flex; flex-direction: column; align-items: center">
    <CommentItem style="width: 100%" v-for="item in commentList" :key="item.id" :comment-info="item"></CommentItem>
    <el-button style="margin-top: 20px" type="text" @click="handleLoadComment" v-if="!commentListEnd">点击加载更多评论</el-button>
    <el-text style="margin-top: 20px" v-if="commentListEnd">到底啦</el-text>
    </el-container>
</el-tab-pane>
```

创建订单页面

订单创建

- **对应lab文档需求1**
- 订单信息的表单会自动读取用户注册时的信息
- 顾客也可以临时更改信息
- 顾客可以选择提货方式为快递或自提
    - 选择自提时，收货地址一栏自动消失
- 购买数量最小为1，最大不超过库存数量
- 显示当前应付总额



苹果 iPhone 15 (A3092) 256GB 蓝色
¥ 6199
支持移动联通电信5G 双卡双待手机

收款人信息	胡图图
联系电话	13400000003
提货方式	[✓ 快递送达] 到店自提
收货地址	翻斗大街翻斗花园二号楼1001室
购买数量	3

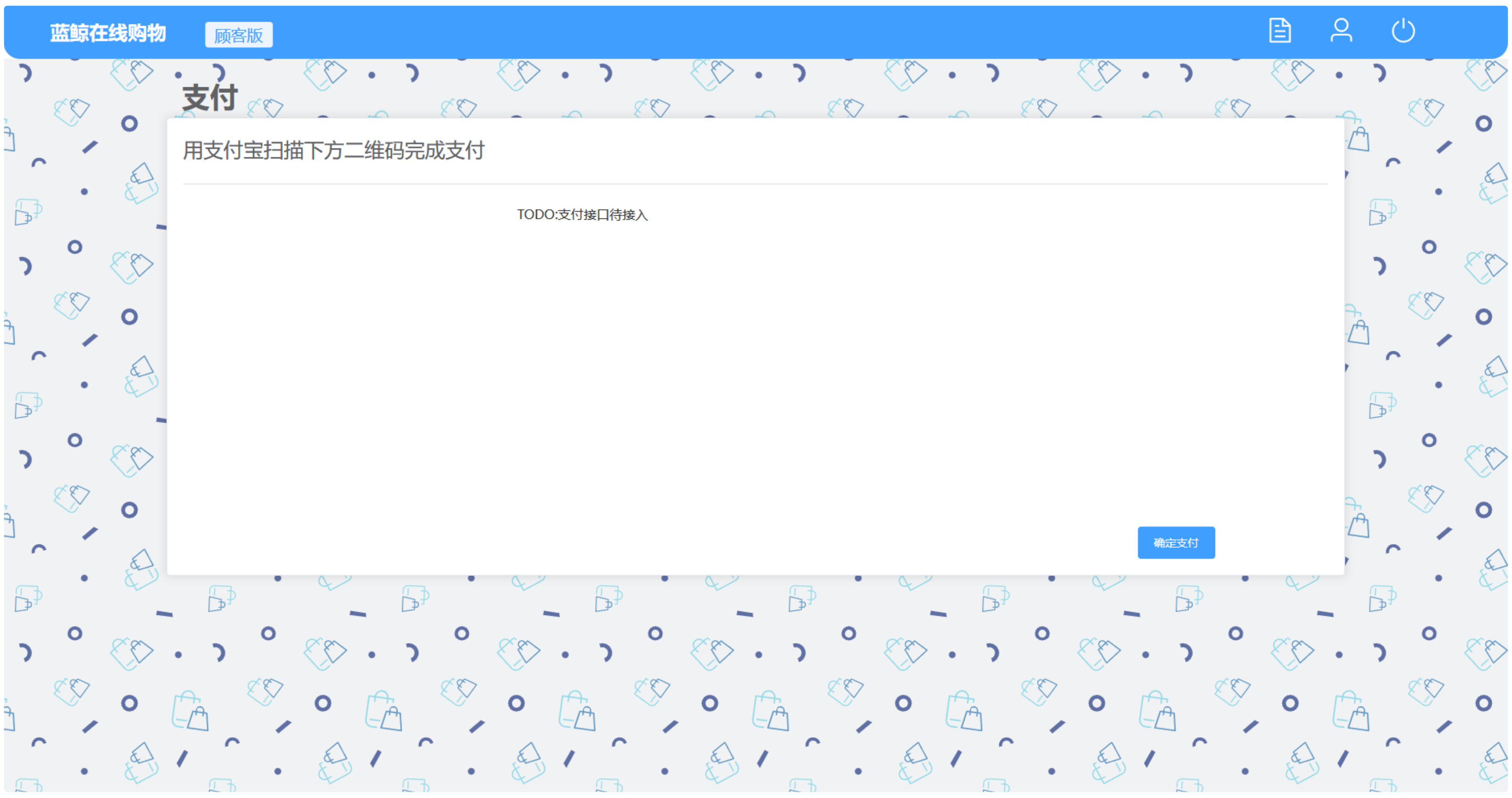
TODO

商品总金额	¥18597
优惠金额	-¥0
应付总额	¥18597

```javascript
// 创建订单的请求方法
function createOrder() {
  reqOrderAdd({
    id: 0,
    itemId: productId.value,
    userId: userId.value,
    status: 'UNPAID',
    delivery: delivery.value,
    storeId: product.value.storeId,
    price: product.value.price,
    quantity: quantity.value
  }).then(res => {
    if (res.data.code === '000'){
      router.push({path: "/Payment/" + res.data.result})
    }
    else {
      ElMessage({
        message: "订单创建失败！",
        type: 'error',
        center: true
      })
    }
  }).catch(err => {
    ElMessage({
      message: "订单创建失败！",
      type: 'error',
      center: true
    })
  })
}
```

订单支付

- **对应lab文档需求2**
- 顾客点击创建订单 之后，订单被创建，并且进入支付页面
- 点击确定支付 可以立即下单
  - 如果不点，直接返回的话订单也会被保存，状态为未支付



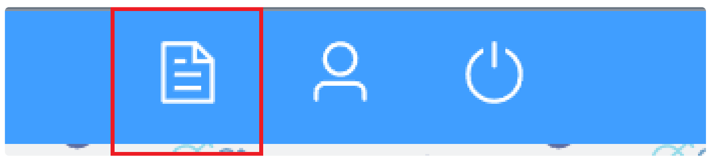
用支付宝扫描下方二维码完成支付

TODO:支付接口待接入

```
function payOrder() {
  reqOrderPay({
    orderId: parseInt(orderId.value)
  }).then(res => {
    if (res.data.code === '000') {
      ElMessage({
        message: '支付成功',
        type: 'success'
      })
      router.push('/orders')
    } else {
      ElMessage({
        message: '支付失败',
        type: 'error'
      })
    }
  })
}
```

订单页面

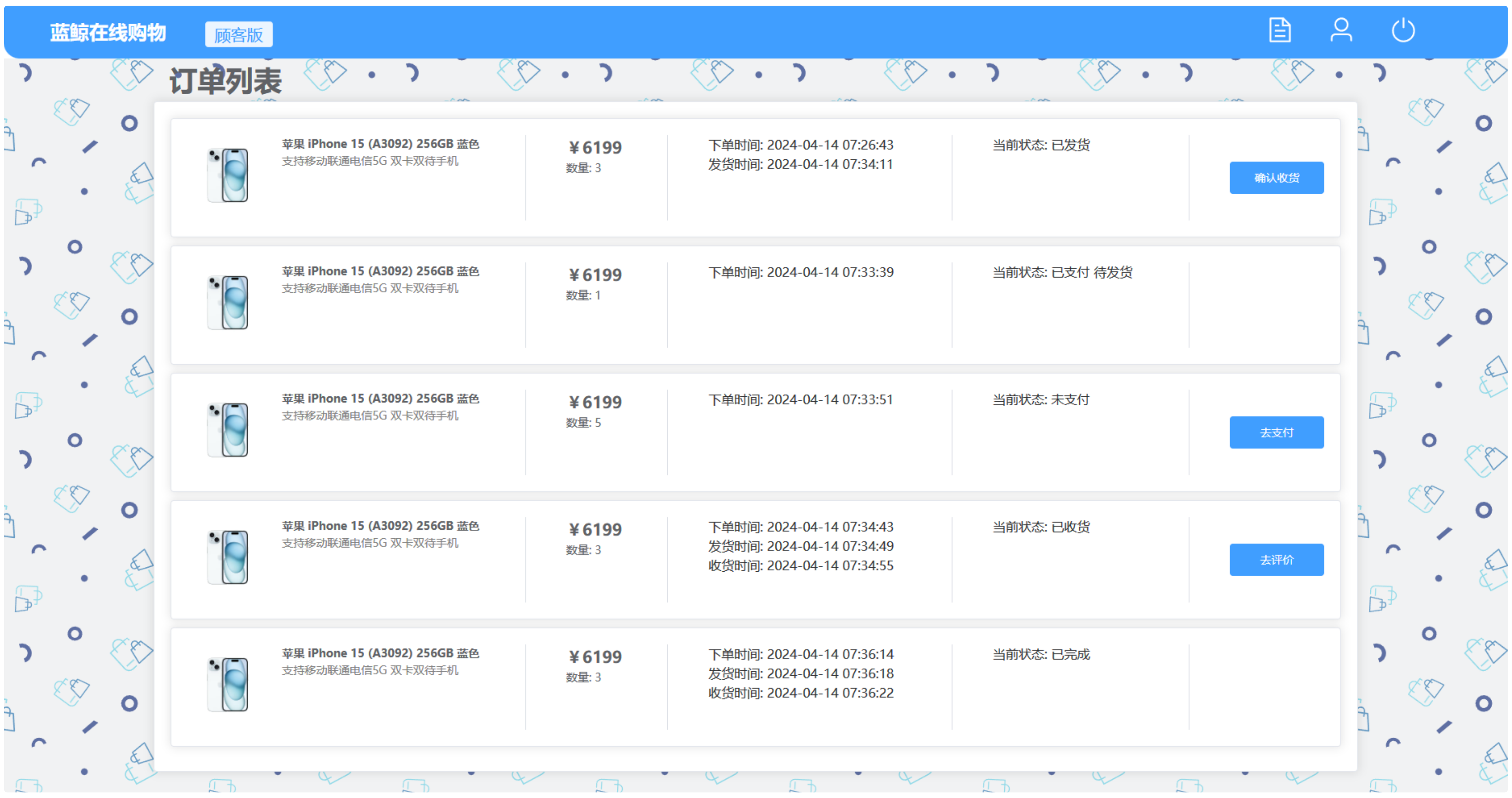
- 订单页面可以通过Header中的图标访问



```
<el-col :span="1" class="header-icon">
  <router-link to="/orders" v-slot="{navigate}">
    <el-icon @click="navigate" :size="35" color="white" ><Document /></el-icon>
  </router-link>
</el-col>
```

顾客订单页面

- **对应lab文档需求3**
- 顾客能够访问自己的订单列表
- 订单列表包含一些必要的信息
- 针对不同状态的订单，用户可能会有相关操作，订单右端会出现操作的按钮
  - 操作包括确认收货，**对应lab文档需求5**



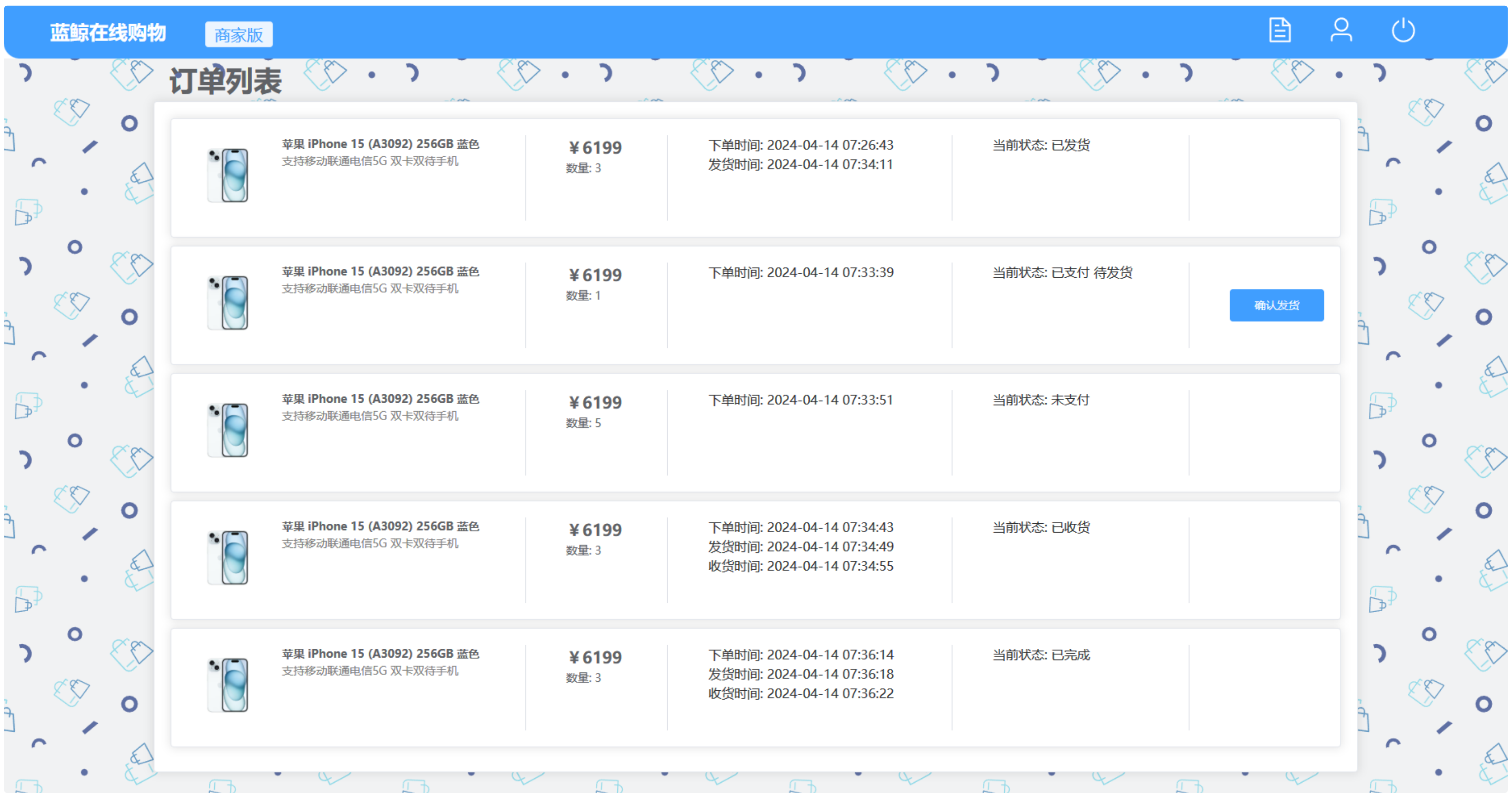
订单列表

苹果 iPhone 15 (A3092) 256GB 蓝色 支持移动联通电信5G 双卡双待手机	¥6199 数量: 3	下单时间: 2024-04-14 07:26:43 发货时间: 2024-04-14 07:34:11	当前状态: 已发货	确认收货
苹果 iPhone 15 (A3092) 256GB 蓝色 支持移动联通电信5G 双卡双待手机	¥6199 数量: 1	下单时间: 2024-04-14 07:33:39	当前状态: 已支付 待发货	
苹果 iPhone 15 (A3092) 256GB 蓝色 支持移动联通电信5G 双卡双待手机	¥6199 数量: 5	下单时间: 2024-04-14 07:33:51	当前状态: 未支付	去支付
苹果 iPhone 15 (A3092) 256GB 蓝色 支持移动联通电信5G 双卡双待手机	¥6199 数量: 3	下单时间: 2024-04-14 07:34:43 发货时间: 2024-04-14 07:34:49 收货时间: 2024-04-14 07:34:55	当前状态: 已收货	去评价
苹果 iPhone 15 (A3092) 256GB 蓝色 支持移动联通电信5G 双卡双待手机	¥6199 数量: 3	下单时间: 2024-04-14 07:36:14 发货时间: 2024-04-14 07:36:18 收货时间: 2024-04-14 07:36:22	当前状态: 已完成	

商店工作人员订单页面

- **对应lab文档需求3**
- 商店工作人员能访问自己商店的订单列表，其他商店是看不到的
- 看到的信息基本和顾客是一样的，只是操作的按钮不一样，商家只有一个确认发货的按钮

- 点击确认发货后库存会进行修改



蓝鲸在线购物 商家版

订单列表

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 3
下单时间: 2024-04-14 07:26:43
发货时间: 2024-04-14 07:34:11
当前状态: 已发货

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 1
下单时间: 2024-04-14 07:33:39
当前状态: 已支付 待发货
确认发货

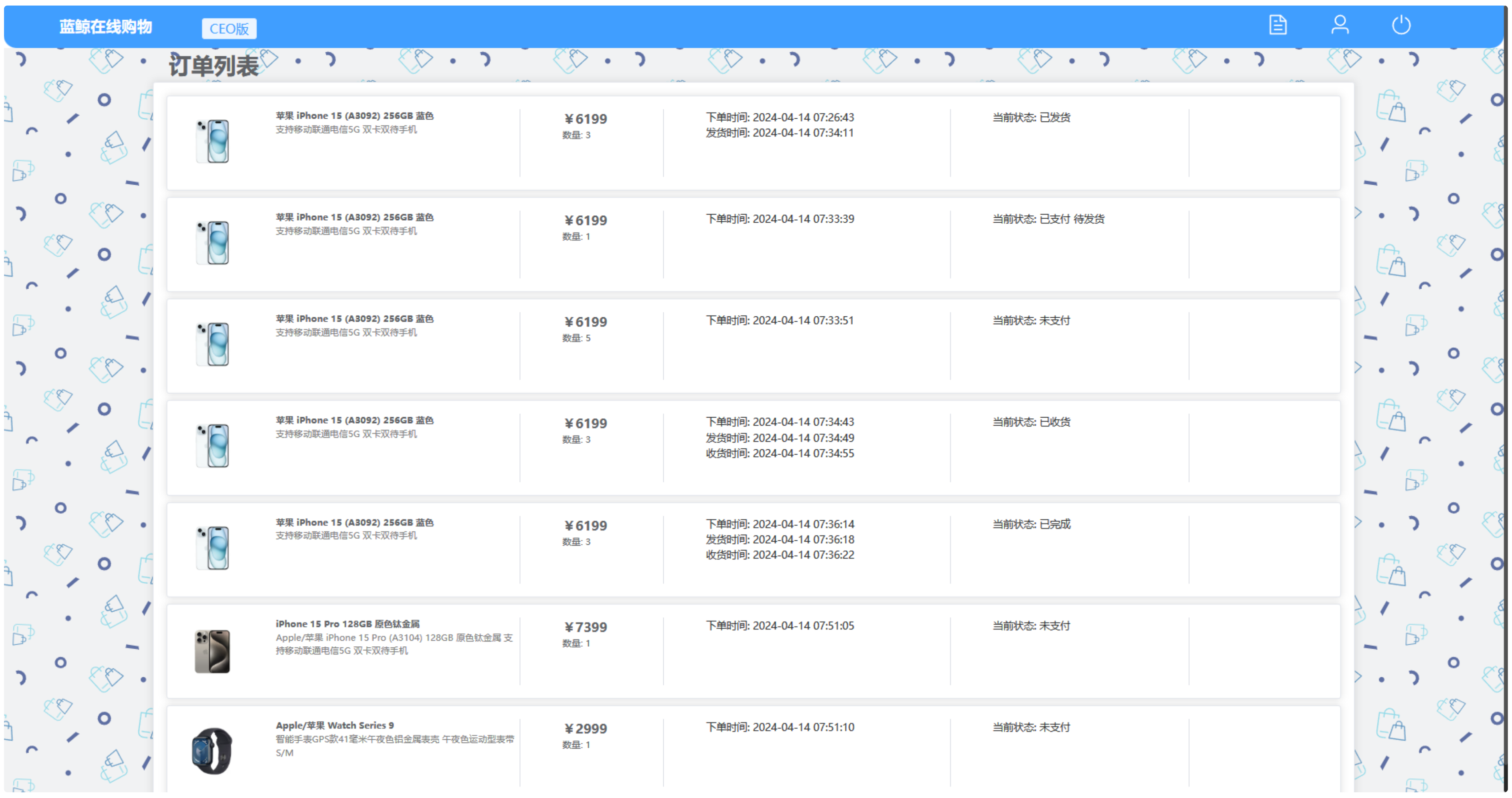
苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 5
下单时间: 2024-04-14 07:33:51
当前状态: 未支付

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 3
下单时间: 2024-04-14 07:34:43
发货时间: 2024-04-14 07:34:49
收货时间: 2024-04-14 07:34:55
当前状态: 已收货

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 3
下单时间: 2024-04-14 07:36:14
发货时间: 2024-04-14 07:36:18
收货时间: 2024-04-14 07:36:22
当前状态: 已完成

商场管理人员和经理订单页面

- 可以看到整个商场所有的订单
- 商场管理人员和经理无法对订单做任何操作



蓝鲸在线购物 CEO版

订单列表

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 3
下单时间: 2024-04-14 07:26:43
发货时间: 2024-04-14 07:34:11
当前状态: 已发货

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 1
下单时间: 2024-04-14 07:33:39
当前状态: 已支付 待发货

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 5
下单时间: 2024-04-14 07:33:51
当前状态: 未支付

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 3
下单时间: 2024-04-14 07:34:43
发货时间: 2024-04-14 07:34:49
收货时间: 2024-04-14 07:34:55
当前状态: 已收货

苹果 iPhone 15 (A3092) 256GB 蓝色
支持移动联通电信5G 双卡双待手机
¥6199
数量: 3
下单时间: 2024-04-14 07:36:14
发货时间: 2024-04-14 07:36:18
收货时间: 2024-04-14 07:36:22
当前状态: 已完成

iPhone 15 Pro 128GB 原色钛金属
Apple/苹果 iPhone 15 Pro (A3104) 128GB 原色钛金属 支持移动联通电信5G 双卡双待手机
¥7399
数量: 1
下单时间: 2024-04-14 07:51:05
当前状态: 未支付

Apple/苹果 Watch Series 9
智能手表GPS款41毫米午夜色铝金属表壳 午夜色运动型表带 S/M
¥2999
数量: 1
下单时间: 2024-04-14 07:51:10
当前状态: 未支付

- 订单页面实现

```
<script setup lang="ts">
import OrderItem from "../../components/OrderItem.vue";
import {reqAllOrder, reqOrderList, reqStoreOrderList} from
"../../api/order.ts";
import {ref} from "vue";

const orderList = ref([])
const role = ref(sessionStorage.getItem('role'))
```

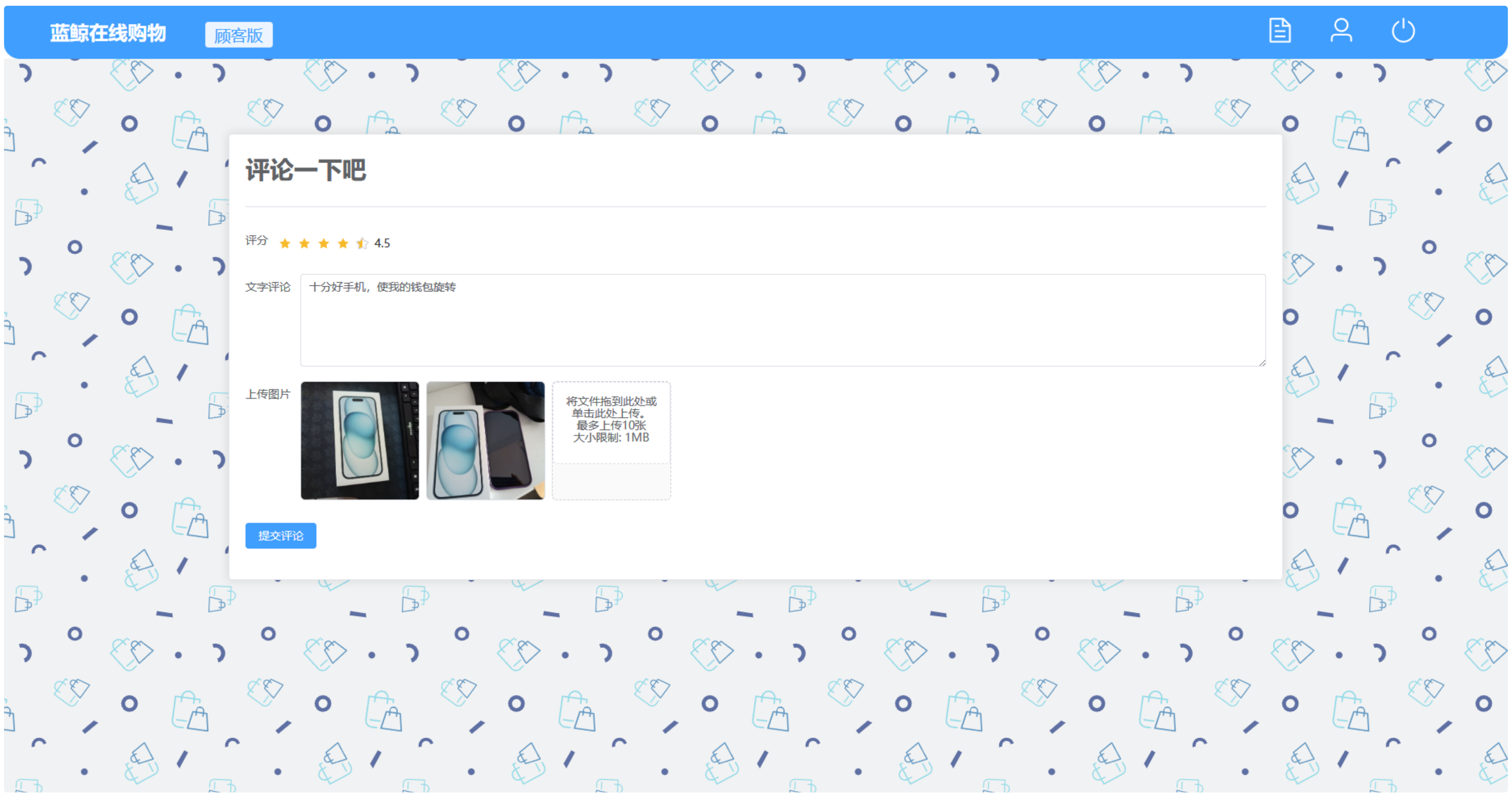
```
getOrders()

function getOrders(){
  // 根据用户角色获取订单列表
  if(role.value == 'CUSTOMER'){
    reqOrderList().then(res => {
      orderList.value = res.data.result
    })
  }
  if(role.value == 'STAFF'){
    reqStoreOrderList().then(res => {
      orderList.value = res.data.result
    })
  }
  if(role.value == 'CEO' || role.value == 'MANAGER'){
    reqAllOrder().then(res => {
      orderList.value = res.data.result
    })
  }
}
</script>

<template>
  <el-container style="display: flex; flex-direction: column; align-items: center" class="bgimage">
    <el-text style="width: 78%;text-align: left; font-size: 35px; font-weight: bold">订单列表</el-text>
    <el-card style="width: 80%">
      <OrderItem v-for="item in orderList" :key="item.createTime"  :order-info="item" :is-store-order=true></OrderItem>
    </el-card>
  </el-container>
</template>

<style scoped>
.bgimage {
  background-image: url("../../assets/shopping-1s-1084px.svg");
}
</style>
```

评论页面

新建评论

- **对应lab文档需求6**
- 顾客可以对已收货的订单进行评论
- 需要输入的信息包括:

    - 评分
    - 文字评论
    - 图片评论（可选）
- 评论后后端会自动计算评分，所以前端只需要上传评论就行啦



评论一下吧

评分 ★★★★☆ 4.5

文字评论 十分好手机，使我的钱包旋转

上传图片

- 实现上和新建商品、新建商店类似

后端具体实现

评论模块

- 评论（comment）

字段	类型	备注
id	unique identifier & not null & primary key	评论id
content	varchar(255) & not null	评论内容
itemId	int & not null	评论所属商品id
rating	float & not null	评分
userId	int & not null	发表评论用户id
create_time	varchar(255)	评论时间
order_id	int & not null	评论所属订单id

获取商品评论列表

- 商品详情界面会请求评论列表。
- 采用懒加载的策略，请求时只会返回部分评论。
- **实现细节**
    - 请求体要包含商品*id*和初始下标

  - 只返回商品所有评论列表中从初始下标开始的评论。返回个数可以修改，为了测试方便，一次懒加载只返回两个评论。

```
    @Override
    public List<CommentVO> getCommentByItemIdAndCount(Integer itemId, Integer
startInd) {
        // 一次返回的评论数量，为了保证展示效果，这里设置为2
        int loadCount = 2;
        List<Comment> allComments = commentRepository.findByItemId(itemId);
        List<Comment> poList;
        List<CommentVO> voList = new ArrayList<>();
        if (startInd + loadCount <= allComments.size()) {
            // 如果还有更多评论未加载
            poList = allComments.subList(startInd, startInd + loadCount);
        }
        else{
            // 如果所有评论都已加载
            poList = allComments.subList(startInd, allComments.size());
        }
        for(Comment comment : poList){
            voList.add(comment.toVO());
        }
        return voList;
    }
```

获取商品评论总条数

- 因为采用了懒加载的策略，前端基本无法通过一次请求就获取到所有的评论，从而就无法得到评论总条数
- **实现细节**
  - 统计总条数并返回就行

```
    @Override
    public Integer getCommentCountByItemId(Integer itemId) {
        List<Comment> comments = commentRepository.findByItemId(itemId);
        return comments.size();
    }
```

新建评论

- 新增评论，并同步更新商店和商品的评分
- **实现细节**
  - 更新商品评分时，直接遍历商品的所有评论算一遍就行。这样虽然牺牲了性能，但保证了数据不会出问题。（直接用公式计算新的评分在不当操作下会出bug）
  - 更新商店评分时，要将没有评论的商品（即评分为0）过滤掉，否则会降低商家的评分参考性。

```
@Override
    public Integer appendComment(CommentVO commentVO) {
```

```java
        Comment newComment = commentVO.toPO();
        newComment.setCreateTime(new Date());
        Comment saveComment = commentRepository.save(newComment);
        // 更新相关Rating
        updateItemRating(saveComment.getItemId(), saveComment.getRating());
        updateStoreRating(saveComment.getItemId(), saveComment.getRating());
        return saveComment.getId();
    }

    // 根据评论更新商品Rating
    private void updateItemRating(Integer itemId, float rating){
        Item item = itemRepository.findById(itemId.intValue());
        List<Comment> comments = commentRepository.findByItemId(itemId);

        float ratingSum = 0;
        for(Comment c : comments){
            ratingSum += c.getRating();
        }
        float newRating = (ratingSum + rating) / (comments.size() + 1);
        item.setRating(newRating);
        itemRepository.save(item);
    }

    // 根据评论更新商家Rating
    private void updateStoreRating(Integer itemId, float rating){
        Item item = itemRepository.findById(itemId.intValue());
        int storeId = item.getStoreId();

        Store store = storeRepository.findById(storeId);
        List<Item> items = itemRepository.findByStoreId(storeId);

        // 只计算有评分的商品，否则商家太惨了
        float ratingSum = 0;
        int validRatingCount = 0;

        for(Item i : items){
            if(i.getRating() != 0){
                validRatingCount ++;
                ratingSum += i.getRating();
            }
        }
        float newRating = ratingSum / validRatingCount;
        store.setRating(newRating);
        storeRepository.save(store);
    }
```

订单模块

- 订单 (order)

字段	类型	备注
id	unique identifier & not null & primary key	订单id
itemId	int & not null	订单商品id
userId	int & not null	购买者id
status	varchar(255) & not null	订单状态
delivery	varchar(255) & not null	提货方式
storeId	int & not null	所属商店id
coupon	varchar(255) & not null	优惠券使用情况(待定)
price	float & not null	价格(商品单价)
quantity	int & not null	数量
create_time	varchar(255)	订单创建时间
send_time	varchar(255)	订单发货时间
receive_time	varchar(255)	订单收货时间

获取订单列表

- 前端会请求订单列表。分三种请求：请求所有订单、请求某个商店的订单、请求某个用户的订单。
- **实现细节**
    - 对这三种请求，面向前端写三个请求接口。
    - 请求订单列表前，后端检查用户身份或所属商店是否与请求符合。
    - 若符合，则返回对应列表

```
    @Override
    public List<OrderVO> getAll() {
        User user = securityUtil.getCurrentUser();
        // 只有CEO和MANAGER可以查看所有订单
        if(user.getRole() ≠ RoleEnum.CEO && user.getRole() ≠ 
RoleEnum.MANAGER){
            throw AuthorityException.notAllowToOperate();
        }
        List<Order> poList = orderRepository.findAll();
        List<OrderVO> voList = new ArrayList<>();
        for(Order po : poList){
            voList.add(po.toVO());
        }
        return voList;
    }

    @Override
    public List<OrderVO> getByUserId() {
        User user = securityUtil.getCurrentUser();
        int userId = user.getId();
        List<Order> poList = orderRepository.findByUserId(userId);
        List<OrderVO> voList = new ArrayList<>();
        for(Order po : poList){
            voList.add(po.toVO());
        }
        return voList;
    }

    @Override
    public List<OrderVO> getByStoreId() {
        User user = securityUtil.getCurrentUser();
        int storeId = user.getStoreId();
        List<Order> poList = orderRepository.findByStoreId(storeId);
        List<OrderVO> voList = new ArrayList<>();
        for(Order po : poList){
            voList.add(po.toVO());
        }
        return voList;
    }
```

创建订单

- 用户点击购买并确认信息之后订单被创建。
- 订单状态初始值为*UNPAID*
- **实现细节**
    - 先校验用户身份，非本人无法创建订单
    - 尽管前端也对库存做了校验，后端再校验一遍会更安全
    - 校验后创建订单，给订单标记创建时间，返回订单id

```
@Override
    public Integer createOrder(OrderVO orderVO) {
        // 安全校验
        User user = securityUtil.getCurrentUser();
        Item item = itemRepository.findById(orderVO.getItemId().intValue());
        if(!authorityCheck(user.getId(), orderVO.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(item.getInventory() < orderVO.getQuantity()){
            // 库存不足
            throw ItemException.inventoryNotEnough();
        }
        // 附上系统时间
        orderVO.setCreateTime(new Date());
        Order orderToCreate = orderVO.toPO();
        Order orderSaved = orderRepository.save(orderToCreate);
        return orderSaved.getId();
    }
```

支付订单

- 前端支付订单的请求接口
- 目前只有如下功能，后续随着支付API的提供会增加功能：
  - 校验用户信息
  - 将订单状态从 *UNPAID* 转换到 *UNSEND* 的功能
- **实现细节**

```
@Override
    public Boolean purchaseOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNPAID){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为未发货
        switchOrderStatus(order, OrderStatus.UNSEND);
        return true;
    }
```

订单确认发货

- 商家发货后，在平台上将订单状态更新为已发货(*UNGET*)
- **实现细节**
  - 校验店员身份
  - 校验订单状态
  - 改变订单状态

```
      @Override
    public Boolean sendOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(user.getStoreId().intValue() != order.getStoreId().intValue()){
            // 非本店无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNSEND){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为已发货
        switchOrderStatus(order, OrderStatus.UNGET);
        // 将订单发货时间更新为当前时间
        order.setSendTime(new Date());
        orderRepository.save(order);
        return true;
    }
```

订单确认收货

- 顾客收货后，在平台上点击确认收货，订单状态将改变为已收货(UNCOMMENT)
- **实现细节**
  - 校验操作者身份
  - 校验订单状态
  - 改变订单状态

```
@Override
    public Boolean receiveOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNGET){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为已收货
        switchOrderStatus(order, OrderStatus.UNCOMMENT);
        // 将订单收货时间更新为当前时间
        order.setReceiveTime(new Date());
        orderRepository.save(order);
        return true;
    }
```

评价订单

- 注意：此接口只负责改变订单状态，并不负责评论模块的相关功能
- 用户评价订单后，订单的状态将改变为已评价(DONE)
- **实现细节**
  - 校验操作者身份
  - 校验订单状态
  - 改变订单状态

```
     @Override
    public Boolean commentOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNCOMMENT){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为已评论
        switchOrderStatus(order, OrderStatus.DONE);
        return true;
    }
```

前端订单模块实现代码

- /src/api/order.ts
- 订单网络请求

```
import {axios} from "../utils/request.ts";
import {ORDER_MODULE} from "./_prefix";

export type OrderInfo = {
id: number,
itemId: number,
userId: number,
status: string,
delivery: string,
storeId: number,
price: number,
quantity: number
}

// 请求订单
export const reqOrder = (orderId: {orderId: number}) ⇒ {
return axios.get(`${ORDER_MODULE}/id`, {params: orderId})
.then(res ⇒ {
return res;
})
}

// 请求所有订单
export const reqAllOrder = () ⇒ {
return axios.get(`${ORDER_MODULE}/all`)
.then(res ⇒ {
return res;
})
}

// 请求订单列表(根据用户id)
export const reqOrderList = () ⇒ {
return axios.get(`${ORDER_MODULE}/user`)
.then(res ⇒ {
return res;
})
}

// 请求订单列表(根据商店id)
export const reqStoreOrderList = () ⇒ {
return axios.get(`${ORDER_MODULE}/store`, )
.then(res ⇒ {
return res;
})
}

// 请求创建订单
export const reqOrderAdd = (orderInfo: OrderInfo) ⇒ {
return axios.post(`${ORDER_MODULE}/create`, orderInfo)
.then(res ⇒ {
return res;
})
}

// 请求删除订单
export const reqOrderDelete = (orderId: {orderId: number}) ⇒ {
return axios.post(`${ORDER_MODULE}/delete`, null, {params: orderId})
.then(res ⇒ {
return res;
})
}

// 请求支付订单
export const reqOrderPay = (orderId: {orderId: number}) ⇒ {
return axios.post(`${ORDER_MODULE}/purchase`, null, {params: orderId})
.then(res ⇒ {
return res;
})
}

// 请求发货
export const reqOrderSend = (orderId: {orderId: number}) ⇒ {
return axios.post(`${ORDER_MODULE}/send`, null, {params: orderId})
.then(res ⇒ {
return res;
})
}

// 请求收货
export const reqOrderReceive = (orderId: {orderId: number}) ⇒ {
return axios.post(`${ORDER_MODULE}/receive`, null, {params: orderId})
.then(res ⇒ {
return res;
})
}

// 请求评论
export const reqOrderComment = (orderId: {orderId: number}) ⇒ {
return axios.post(`${ORDER_MODULE}/comment`, null, {params: orderId})
.then(res ⇒ {
return res;
})
}
```

- src/components/OrderItem.vue
- 订单组件

```
<script setup lang="ts">

import {reqImageList} from "../api/tools.ts";
import {ref} from "vue";
import {reqProduct} from "../api/product.ts";
import {router} from "../router";
import {reqOrderReceive, reqOrderSend} from "../api/order.ts";

const props = defineProps(['orderInfo'])
const role = ref(sessionStorage.getItem('role'))

const orderId = ref(props.orderInfo.id)
const orderCreateTime = ref(props.orderInfo.createTime)
const orderSendTime = ref(props.orderInfo.sendTime)
const orderReceiveTime = ref(props.orderInfo.receiveTime)
const orderStatus = ref(props.orderInfo.status)

const iconUrl = ref('')
const productId = ref(props.orderInfo.itemId)
const productName = ref('')
const productDesciption = ref('')

getOrderIconUrl()
getProductInfo()
formatTime()

function getOrderIconUrl(){
  reqImageList({
    type: 'ITEM_DISP',
    parentId: props.orderInfo.itemId
  }).then(res => {
    res.data.result.sort((a: any, b: any) => a.ind - b.ind)
    iconUrl.value = res.data.result[0].url
  })
}

function getProductInfo(){
  reqProduct({itemId: parseInt(productId.value)}).then(res => {
    productName.value = res.data.result.name
    productDesciption.value = res.data.result.description
  })
}

function formatTime() {
  orderCreateTime.value = props.orderInfo.createTime.substring(0,
19).replace('T', ' ')
  if (orderSendTime.value !== null) {
    orderSendTime.value = props.orderInfo.sendTime.substring(0,
19).replace('T', ' ')
  }
  if (orderReceiveTime.value !== null) {
    orderReceiveTime.value = props.orderInfo.receiveTime.substring(0,
19).replace('T', ' ')
  }
}

function statusToString(status: string){
  switch (status) {
    case 'UNPAID':
      return '未支付'
    case 'UNSEND':
      return '已支付 待发货'
    case 'UNGET':
      return '已发货'
    case 'UNCOMMENT':
      return '已收货'
    case 'DONE':
      return '已完成'
    default:
      return '未知状态'
  }
}

function toPayment(){
  router.push({path: "/Payment/" + orderId.value})
}

function orderSend(){
  reqOrderSend({
    orderId: orderId.value
  }).then(res => {
    if (res.data.code === '000') {
      ElMessage({
        message: '发货成功',
        type: 'success'
      })
      router.go(0)
    } else {
      ElMessage({
        message: '发货失败',
        type: 'error'
      })
    }
  })
}

function orderReceive(){
  reqOrderReceive({
    orderId: orderId.value
  }).then(res => {
    if (res.data.code === '000') {
      ElMessage({
        message: '收货成功',
        type: 'success'
      })
      router.go(0)
    } else {
      ElMessage({
        message: '收货失败',
        type: 'error'
      })
    }
  })
}

function toWriteComment(){
  router.push({path: "/Comment/" + orderId.value})
}

</script>

<template>
  <el-card style="margin-bottom: 10px">
    <el-row>
      <el-col :span="2">
        <el-image :src="iconUrl" style="width: 100px"></el-image>
      </el-col>
      <el-col :span="5">
        <el-row>
          <el-text style="font-weight: bold">{{productName}}</el-text>
        </el-row>
        <el-row>
          <el-text style="color: gray">{{productDesciption}}</el-text>
        </el-row>

      </el-col>
      <el-col :span="1">
        <el-divider direction="vertical" style="height: 100%"></el-divider>
      </el-col>
      <el-col :span="2">
        <el-row>
          <el-text style="font-weight: bold; font-size: 20px">¥
{{props.orderInfo.price}}</el-text>
        </el-row>
        <el-row>
          <el-text>数量: {{props.orderInfo.quantity}}</el-text>
        </el-row>
      </el-col>
      <el-col :span="1">
        <el-divider direction="vertical" style="height: 100%"></el-divider>
      </el-col>
      <el-col :span="5">
        <el-row>下单时间: {{orderCreateTime}}</el-row>
        <el-row v-if="orderSendTime">发货时间: {{orderSendTime}}</el-row>
        <el-row v-if="orderReceiveTime">收货时间: {{orderReceiveTime}}</el-row>
      </el-col>
      <el-col :span="1">
        <el-divider direction="vertical" style="height: 100%"></el-divider>
      </el-col>
      <el-col :span="4">
        当前状态: {{statusToString(orderStatus)}}
      </el-col>
      <el-col :span="1">
        <el-divider direction="vertical" style="height: 100%"></el-divider>
      </el-col>
      <el-col :span="2" v-if="role == 'CUSTOMER'" style="display: flex; flex-
direction: column; justify-content: center">
        <el-button type="primary" size="large" v-if="orderStatus == 'UNPAID'"
@click="toPayment">去支付</el-button>
        <el-button type="primary" size="large" v-if="orderStatus == 'UNGET'"
@click="orderReceive">确认收货</el-button>
        <el-button type="primary" size="large" v-if="orderStatus ==
'UNCOMMENT'" @click="toWriteComment">去评价</el-button>
      </el-col>
      <el-col :span="2" v-if="role == 'STAFF'" style="display: flex; flex-
direction: column; justify-content: center">
        <el-button type="primary" size="large" v-if="orderStatus == 'UNSEND'"
@click="orderSend">确认发货</el-button>
      </el-col>
    </el-row>
  </el-card>
</template>

<style scoped>

</style>
```

- /src/views/product/Payment.vue
- 支付页面

```
<script setup lang="ts">
import {ref} from "vue";
import {router} from "../../router";
import {reqOrderPay} from "../../api/order.ts";


const orderId = ref(router.currentRoute.value.params.orderId)

function payOrder() {
  reqOrderPay({
    orderId: parseInt(orderId.value)
  }).then(res => {
    if (res.data.code === '000') {
      ElMessage({
        message: '支付成功',
        type: 'success'
      })
      router.push('/orders')
    } else {
      ElMessage({
        message: '支付失败',
        type: 'error'
      })
    }
  })
}

</script>

<template>
  <el-main class="bgimage">
    <el-container style="display: flex; flex-direction: column; align-items: center">
      <el-text style="width: 78%;text-align: left; font-size: 35px; font-weight: bold">支付</el-text>
      <el-card style="width: 80%">
        <el-row style="font-size: 25px; color: #606266">
          用支付宝扫描下方二维码完成支付
        </el-row>
        <el-divider></el-divider>
        <el-row>
          <el-col :span="10" :offset="7">
            <el-container style="height: 400px">TODO:支付接口待接入</el-container>
          </el-col>
        </el-row>
        <el-row>
          <el-col :span="5" :offset="20">
            <el-button size="large" type="primary" @click="payOrder">确定支付</el-button>
          </el-col>
        </el-row>
      </el-card>
    </el-container>
  </el-main>
</template>

<style scoped>
.bgimage {
  background-image: url("../../assets/shopping-1s-1084px.svg");
}
</style>
```

- src/views/product/Purchase.vue
- 创建订单界面

```
<script setup lang="ts">

import {ref} from "vue";
import {router} from "../../router";
import {reqProduct} from "../../api/product.ts";
import {userInfo} from "../../api/user.ts";
import {reqImageList} from "../../api/tools.ts";
import {reqOrderAdd} from "../../api/order.ts";
import {ElMessage} from "element-plus";

const productId = ref(router.currentRoute.value.params.itemId)
const product = ref('')
const productIconUrl = ref('')

const userId = ref(1)
const name = ref('')
const tel = ref('')
const address = ref('')
const quantity = ref(1)
const delivery = ref('DELIVERY')

getProduct()
getProductIconUrl()
getUserInfo()

function getProduct() {
  reqProduct({itemId: parseInt(productId.value)}).then(res => {
    product.value = res.data.result
  })
}

function getProductIconUrl(){
  reqImageList({
    type: 'ITEM_DISP',
    parentId: parseInt(productId.value)
  }).then(res => {
    res.data.result.sort((a, b) => a.ind - b.ind)
    productIconUrl.value = res.data.result[0].url
  })
}

function getUserInfo() {
  userInfo().then(res => {
    userId.value = res.data.result.id
    name.value = res.data.result.name
    tel.value = res.data.result.phone
    address.value = res.data.result.address
  })
}

function createOrder() {
  reqOrderAdd({
    id: 0,
    itemId: productId.value,
    userId: userId.value,
    status: 'UNPAID',
    delivery: delivery.value,
    storeId: product.value.storeId,
    price: product.value.price,
    quantity: quantity.value
  }).then(res => {
    if (res.data.code === '000'){
      router.push({path: "/Payment/" + res.data.result})
    }
    else {
      ElMessage({
        message: "订单创建失败！",
        type: 'error',
        center: true
      })
    }
  }).catch(err => {
    ElMessage({
      message: "订单创建失败！",
      type: 'error',
      center: true
    })
  })
}

function cancelOrder() {
  router.back()
}

</script>

<template>
  <el-container style="display: flex; flex-direction: column; align-items:
center" class="bgimage">
    <el-card style="width: 80%; margin-top: 50px">
      <template #header>
        <div style="font-size: 25px; font-weight: bold">
          <span>蓝鲸商城 结算页</span>
        </div>
      </template>
      <el-row>
        <el-card style="width: 50%; margin-right: 5%;border-radius: 20px;
min-height: 350px">
          <el-text class="section-title">商品信息</el-text>
          <el-divider></el-divider>
          <el-row>
            <el-col :span="8">
              <el-image :src="productIconUrl" fit="contain" style="width:
200px; height: 200px; box-shadow: 5px 5px 10px 1px gray; border-radius: 5px">
</el-image>
            </el-col>
            <el-col :span="16">
              <el-row style="font-size: 25px">{{product.name}}</el-row>
              <el-row style="font-size: 25px;" class="price-text">¥
{{product.price}}</el-row>
              <el-row style="color: gray">{{product.description}}</el-row>
            </el-col>
          </el-row>
        </el-card>
        <el-form style="width: 42%; margin-top: 20px; margin-bottom: auto">
          <el-text class="section-title">订单信息</el-text>
          <el-divider></el-divider>
          <el-form-item label="收款人信息">
            <el-input v-model="name" placeholder="请输入收款人姓名"></el-input>
          </el-form-item>
          <el-form-item label="联系电话">
            <el-input v-model="tel" placeholder="请输入联系电话"></el-input>
          </el-form-item>
          <el-form-item label="提货方式">
            <el-radio-group v-model="delivery">
              <el-radio-button label="DELIVERY">快递送达</el-radio-button>
              <el-radio-button label="PICKUP">到店自提</el-radio-button>
            </el-radio-group>
          </el-form-item>
          <el-form-item label="收货地址" v-if="delivery=='DELIVERY'">
            <el-input v-model="address" placeholder="请输入收货地址"></el-input>
          </el-form-item>
          <el-form-item label="购买数量">
            <el-input-number v-model="quantity" :min="1"
:max="product.inventory"></el-input-number>
          </el-form-item>
        </el-form>
      </el-row>
      <el-divider></el-divider>
      <el-row justify="space-between">
        <el-col :span="16">
          <el-text class="section-title">优惠券信息</el-text>
          <el-card style="height: 200px;">TODO</el-card>
        </el-col>
        <el-col :span="6" style="margin-top: 50px">
          <el-row>
            <el-col :span="14" class="hint-text">商品总金额</el-col>
            <el-col :span="7" class="price-text" style="text-align: right">¥
{{product.price * quantity}}</el-col>
          </el-row>
          <el-row>
            <el-col :span="14" class="hint-text">优惠金额</el-col>
            <el-col :span="7" class="price-text" style="text-align: right">-
¥0</el-col>
          </el-row>
          <el-divider></el-divider>
          <el-row>
            <el-col :span="14" class="hint-text">应付总额</el-col>
            <el-col :span="7" class="price-text" style="text-align: right">¥
{{product.price * quantity}}</el-col>
          </el-row>
          <el-row :gutter="5" style="margin-top: 50px">
            <el-col :span="5" :offset="7">
              <el-button type="info" size="large" class="button-text"
@click="cancelOrder">取消订单</el-button>
            </el-col>
            <el-col :span="5" :offset="4">
              <el-button type="primary" size="large" class="button-text"
@click="createOrder">创建订单</el-button>
            </el-col>
          </el-row>
        </el-col>
      </el-row>
    </el-card>
  </el-container>
</template>

<style scoped>
.bgimage {
  background-image: url("../../assets/shopping-1s-1084px.svg");
}
.section-title{
  font-size: 20px;
  font-weight: bold;
  color: #606266;
}
.hint-text{
  font-size: 20px;
  color: #606266;
}
.price-text{
  font-size: 20px;
  font-weight: bold;
  color: #ff6200;
}
.button-text{
  font-weight: bold;
  font-size: 20px;
}
</style>
```

- src/views/user/AllOrder.vue
- 订单列表页面

```
<script setup lang="ts">
import OrderItem from "../../components/OrderItem.vue";
import {reqAllOrder, reqOrderList, reqStoreOrderList} from
"../../api/order.ts";
import {ref} from "vue";

const orderList = ref([])
const role = ref(sessionStorage.getItem('role'))

getOrders()

function getOrders(){
  // 根据用户角色获取订单列表
  if(role.value == 'CUSTOMER'){
    reqOrderList().then(res => {
      orderList.value = res.data.result
    })
  }
  if(role.value == 'STAFF'){
    reqStoreOrderList().then(res => {
      orderList.value = res.data.result
    })
  }
  if(role.value == 'CEO' || role.value == 'MANAGER'){
    reqAllOrder().then(res => {
      orderList.value = res.data.result
    })
  }
}
</script>

<template>
  <el-container style="display: flex; flex-direction: column; align-items:
center" class="bgimage">
    <el-text style="width: 78%;text-align: left; font-size: 35px; font-
weight: bold">订单列表</el-text>
    <el-card style="width: 80%">
      <OrderItem v-for="item in orderList" :key="item.createTime"  :order-
info="item" :is-store-order=true></OrderItem>
    </el-card>
  </el-container>
</template>

<style scoped>
.bgimage {
  background-image: url("../../assets/shopping-1s-1084px.svg");
}
</style>
```

后端订单模块实现代码

- src/main/java/com/seecoder/BlueWhale/controller/OrderController.java

```java
package com.seecoder.BlueWhale.controller;

import com.seecoder.BlueWhale.service.OrderService;
import com.seecoder.BlueWhale.vo.OrderVO;
import com.seecoder.BlueWhale.vo.ResultVO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/orders")
public class OrderController {
    @Autowired
    OrderService orderService;

    // 根据订单id获取订单
    @GetMapping("/id")
    public ResultVO<OrderVO> getOrderById(@RequestParam int orderId) {
        return ResultVO.buildSuccess(orderService.getById(orderId));
    }

    // 获取所有订单
    @GetMapping("/all")
    public ResultVO<List<OrderVO>> getAllOrders() {
        return ResultVO.buildSuccess(orderService.getAll());
    }

    // 根据用户id获取订单
    @GetMapping("/user")
    public ResultVO<List<OrderVO>> getOrdersByUserId() {
        return ResultVO.buildSuccess(orderService.getByUserId());
    }

    // 根据商店id获取订单
    @GetMapping("/store")
    public ResultVO<List<OrderVO>> getOrdersByStoreId() {
        return ResultVO.buildSuccess(orderService.getByStoreId());
    }

    // 创建订单
    @PostMapping("/create")
    public ResultVO<Integer> createOrder(@RequestBody OrderVO orderVO) {
        return ResultVO.buildSuccess(orderService.createOrder(orderVO));
    }
    // 删除订单
    @PostMapping("/delete")
    public ResultVO<Boolean> deleteOrder(@RequestParam int orderId) {
        return ResultVO.buildSuccess(orderService.deleteOrder(orderId));
    }

    // 购买订单
    @PostMapping("/purchase")
    public ResultVO<Boolean> purchaseOrder(@RequestParam int orderId) {
        return ResultVO.buildSuccess(orderService.purchaseOrder(orderId));
    }

    // 订单发货
    @PostMapping("/send")
    public ResultVO<Boolean> sendOrder(@RequestParam int orderId) {
        return ResultVO.buildSuccess(orderService.sendOrder(orderId));
    }

    // 确认订单收货
    @PostMapping("/receive")
    public ResultVO<Boolean> receiveOrder(@RequestParam int orderId) {
        return ResultVO.buildSuccess(orderService.receiveOrder(orderId));
    }

    // 评论订单
    @PostMapping("/comment")
    public ResultVO<Boolean> commentOrder(@RequestParam int orderId) {
        return ResultVO.buildSuccess(orderService.commentOrder(orderId));
    }
}
```

- src/main/java/com/seecoder/BlueWhale/serviceImpl/OrderServiceImpl.java

```java
package com.seecoder.BlueWhale.serviceImpl;

import com.seecoder.BlueWhale.enums.OrderStatus;
import com.seecoder.BlueWhale.enums.RoleEnum;
import com.seecoder.BlueWhale.exception.AuthorityException;
import com.seecoder.BlueWhale.exception.ItemException;
import com.seecoder.BlueWhale.exception.OrderException;
import com.seecoder.BlueWhale.po.Item;
import com.seecoder.BlueWhale.po.Order;
import com.seecoder.BlueWhale.po.User;
import com.seecoder.BlueWhale.repository.ItemRepository;
import com.seecoder.BlueWhale.repository.OrderRepository;
import com.seecoder.BlueWhale.service.OrderService;
import com.seecoder.BlueWhale.util.SecurityUtil;
import com.seecoder.BlueWhale.util.TokenUtil;
import com.seecoder.BlueWhale.vo.OrderVO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Service
public class OrderServiceImpl implements OrderService {
    @Autowired
    OrderRepository orderRepository;

    @Autowired
    ItemRepository itemRepository;

    @Autowired
    TokenUtil tokenUtil;

    @Autowired
    SecurityUtil securityUtil;

    @Override
    public OrderVO getById(int orderId) {
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        return order.toVO();
    }

    @Override
    public List<OrderVO> getAll() {
        User user = securityUtil.getCurrentUser();
        // 只有CEO和MANAGER可以查看所有订单
        if(user.getRole() != RoleEnum.CEO && user.getRole() !=
RoleEnum.MANAGER){
            throw AuthorityException.notAllowToOperate();
        }
        List<Order> poList = orderRepository.findAll();
        List<OrderVO> voList = new ArrayList<>();
        for(Order po : poList){
            voList.add(po.toVO());
        }
        return voList;
    }

    @Override
    public List<OrderVO> getByUserId() {
        User user = securityUtil.getCurrentUser();
        int userId = user.getId();
        List<Order> poList = orderRepository.findByUserId(userId);
        List<OrderVO> voList = new ArrayList<>();
        for(Order po : poList){
            voList.add(po.toVO());
        }
        return voList;
    }

    @Override
    public List<OrderVO> getByStoreId() {
        User user = securityUtil.getCurrentUser();
        int storeId = user.getStoreId();
        List<Order> poList = orderRepository.findByStoreId(storeId);
        List<OrderVO> voList = new ArrayList<>();
        for(Order po : poList){
            voList.add(po.toVO());
        }
        return voList;
    }

    @Override
    public Integer createOrder(OrderVO orderVO) {
        // 安全校验
        User user = securityUtil.getCurrentUser();
        Item item = itemRepository.findById(orderVO.getItemId().intValue());
        if(!authorityCheck(user.getId(), orderVO.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(item.getInventory() < orderVO.getQuantity()){
            // 库存不足
            throw ItemException.inventoryNotEnough();
        }
        // 附上系统时间
        orderVO.setCreateTime(new Date());
        Order orderToCreate = orderVO.toPO();
        Order orderSaved = orderRepository.save(orderToCreate);
        return orderSaved.getId();
    }

    @Override
    public Boolean deleteOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        orderRepository.delete(order);
        return true;
    }

    @Override
    public Boolean purchaseOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNPAID){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为未发货
        switchOrderStatus(order, OrderStatus.UNSEND);
        return true;
    }

    @Override
    public Boolean sendOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(user.getStoreId().intValue() != order.getStoreId().intValue()){
            // 非本店无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNSEND){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为已发货
        switchOrderStatus(order, OrderStatus.UNGET);
        // 将订单发货时间更新为当前时间
        order.setSendTime(new Date());
        orderRepository.save(order);
        return true;
    }

    @Override
    public Boolean receiveOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNGET){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为已收货
        switchOrderStatus(order, OrderStatus.UNCOMMENT);
        // 将订单收货时间更新为当前时间
        order.setReceiveTime(new Date());
        orderRepository.save(order);
        return true;
    }

    @Override
    public Boolean commentOrder(int orderId) {
        User user = securityUtil.getCurrentUser();
        Order order = orderRepository.findById(orderId);
        if(order == null){
            // 订单不存在
            throw OrderException.orderNotExist();
        }
        if(!authorityCheck(user.getId(), order.getUserId())){
            // 非本人无法操作
            throw AuthorityException.notAllowToOperate();
        }
        if(order.getStatus() != OrderStatus.UNCOMMENT){
            // 订单状态不符合要求
            throw OrderException.orderStatusError();
        }
        // 将订单状态更新为已评论
        switchOrderStatus(order, OrderStatus.DONE);
        return true;
    }

    private void switchOrderStatus(Order order, OrderStatus newStatus){
        order.setStatus(newStatus);
        orderRepository.save(order);
    }
    private boolean authorityCheck(Integer userId, Integer orderUserId){
        return userId.intValue() == orderUserId.intValue();
    }
}
```

- src/main/java/com/seecoder/BlueWhale/vo/OrderVO.java

```java
package com.seecoder.BlueWhale.vo;

import com.seecoder.BlueWhale.enums.OrderStatus;
import com.seecoder.BlueWhale.po.Order;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.Date;

@Getter
@Setter
@NoArgsConstructor
public class OrderVO {
    private Integer id;

    private Integer itemId;

    private Integer userId;

    private OrderStatus status;

    private String delivery;

    private Integer storeId;

    private Float price;

    private Integer quantity;

    private Date createTime;

    private Date sendTime;

    private Date receiveTime;

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

- src/main/java/com/seecoder/BlueWhale/po/Order.java

```
package com.seecoder.BlueWhale.po;

import com.seecoder.BlueWhale.enums.OrderStatus;
import com.seecoder.BlueWhale.vo.OrderVO;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.persistence.*;
import java.util.Date;

@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name = "`order`")
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
```

```java
    @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;
    }
}
```

- src/main/java/com/seecoder/BlueWhale/exception/OrderException.java

```
package com.seecoder.BlueWhale.exception;
/**
 * 和订单有关的Exception
 */

public class OrderException extends RuntimeException {
    public OrderException(String message) {
        super(message);
    }

    public static OrderException orderNotExist() {
        return new OrderException("无法找到该订单！");
    }

    public static OrderException orderStatusError() {
        return new OrderException("订单状态错误！");
    }

}
```