代码Lab4-优惠券模块

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

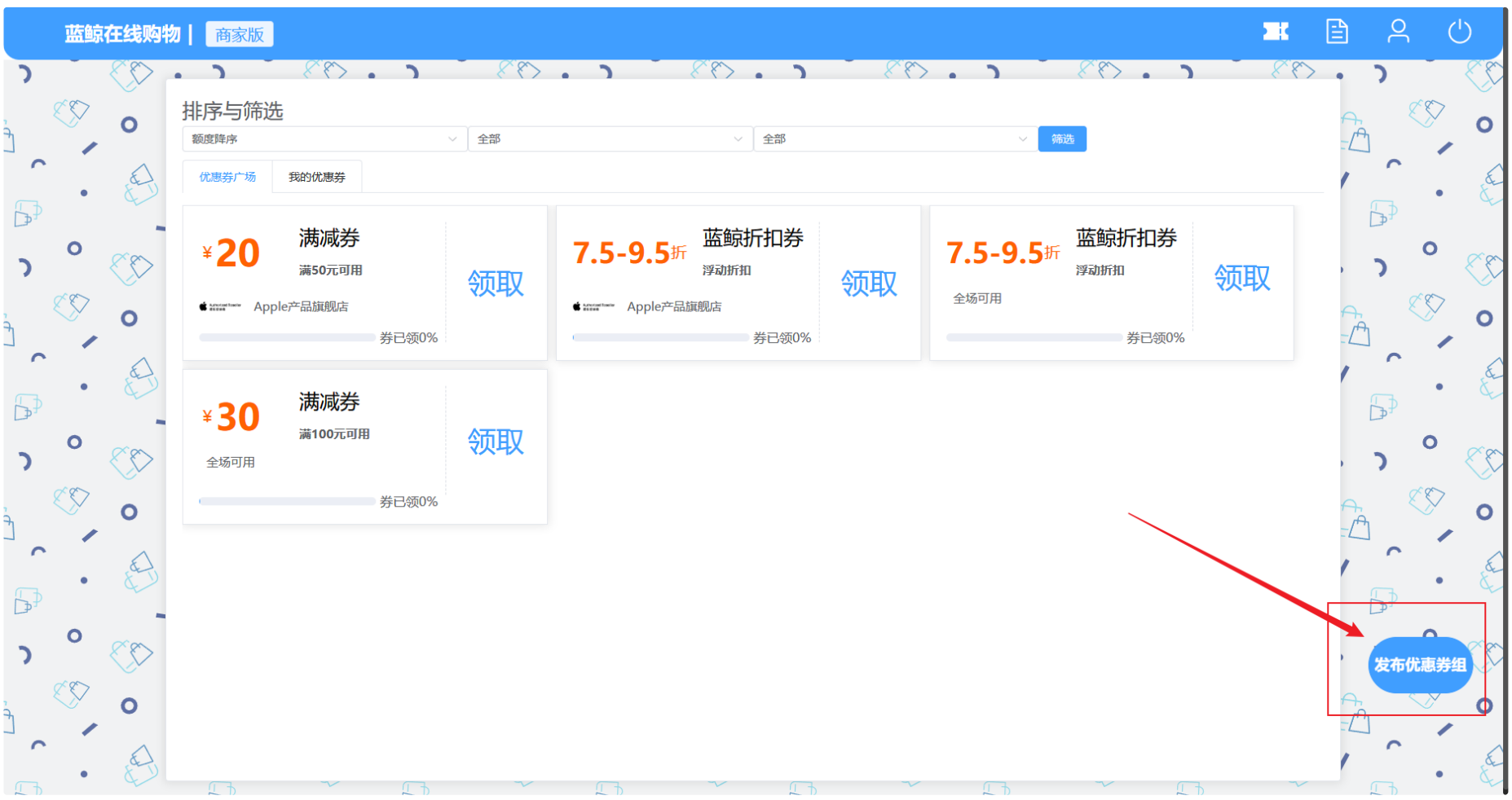
前端具体实现

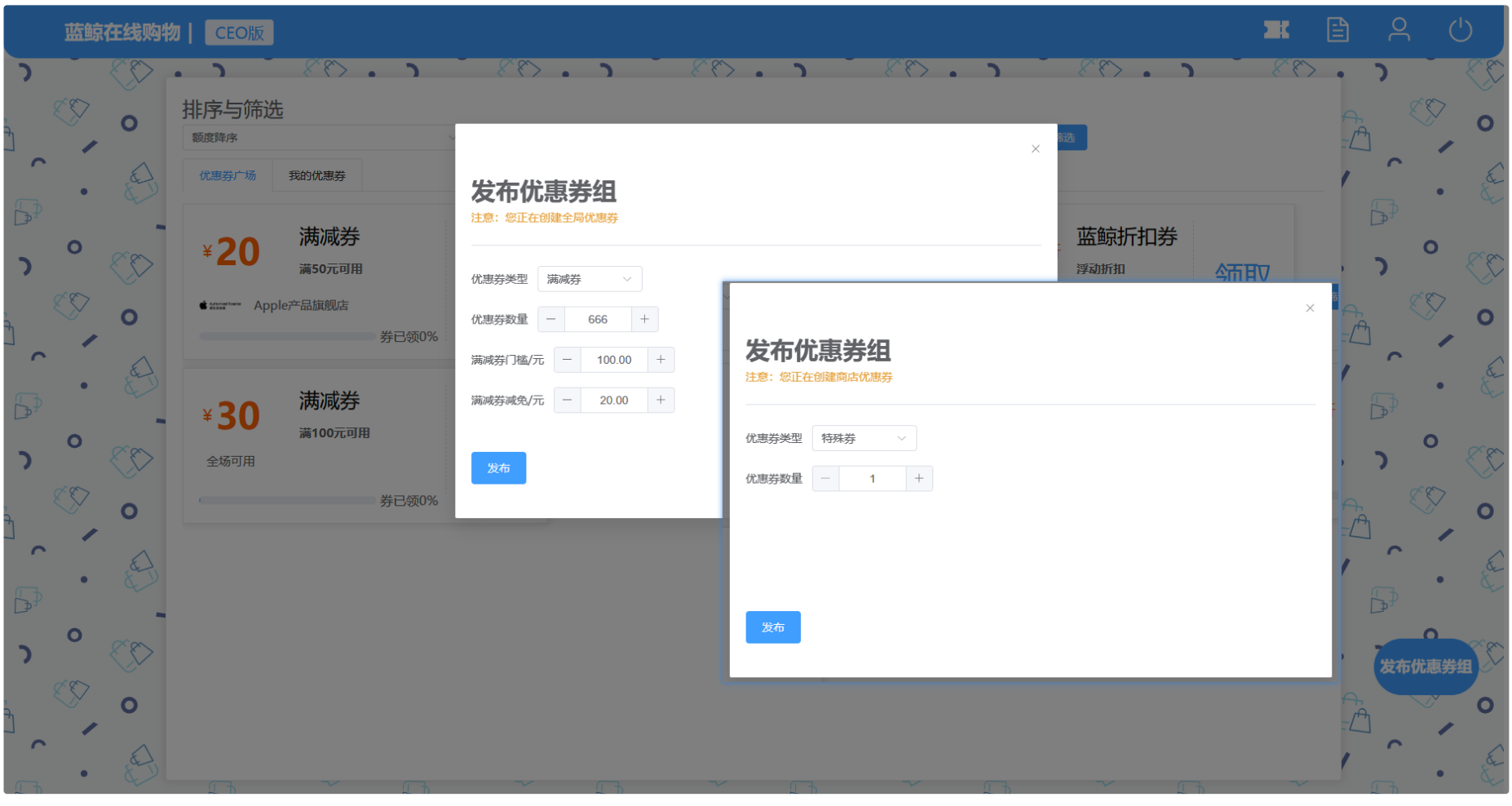
优惠券模块基本功能

发布优惠券组

> 1. 经理（CEO）可以发布全局的优惠券组，门店工作人员可以发布本门店内的优惠券组

- 门店工作人员和经理可以通过优惠券广场页面右下角的**发布优惠券组**按钮打开发布弹窗
- **提示**：发布界面会提示经理正在创建全局的优惠券组；提示门店工作人员正在创建商店的优惠券组
- 发布弹窗的**表单**可以填写优惠券的相关信息
- 点击发布后向后端发送创建优惠券组的**请求**





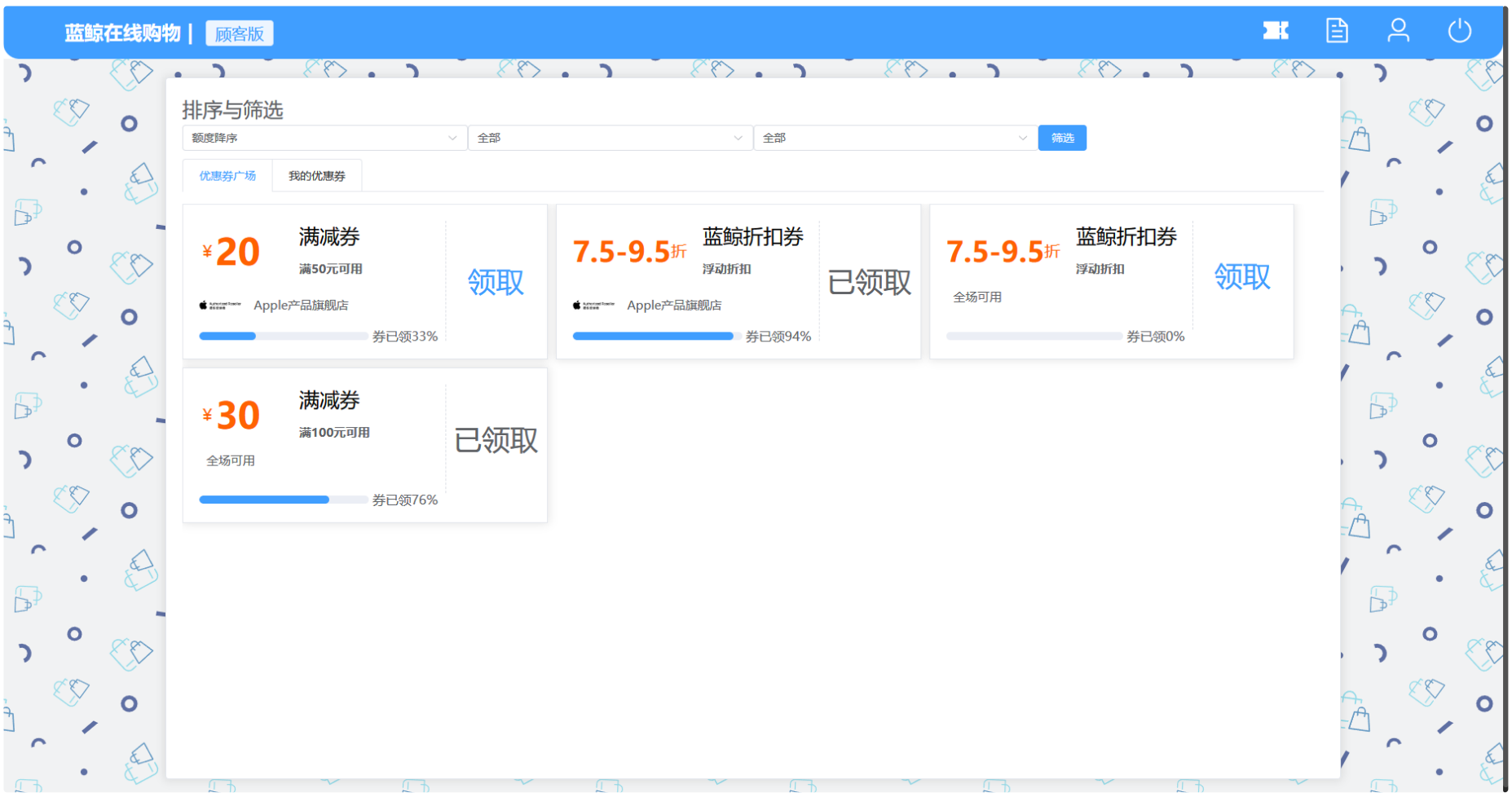
- 网络请求包含发布成功或失败的提示。并且为了及时更新优惠券列表，在发布之后页面会刷新。

```
function handleCreate() {
  reqCouponGroupCreate({
    type: type.value,
    count: count.value,
    threshold: threshold.value,
    reduction: reduction.value
  }).then(res => {
    if(res.data.code === '000'){
      ElMessage({
        message: '发布成功',
        type: 'success',
        center: true
      })
    } else {
      ElMessage({
        message: '发布失败',
        type: 'error',
        center: true
      })
    }
  })
  // 刷新页面
  router.go(0)
}
```

优惠券广场&我的优惠券

2. 经理可以查看所有优惠券组，门店工作人员可以查看本门店内的优惠券组。包括这些优惠券组的领取情况。
3. 顾客可以查看所有优惠券组，并领取优惠券。同一个优惠券组不能重复领取。

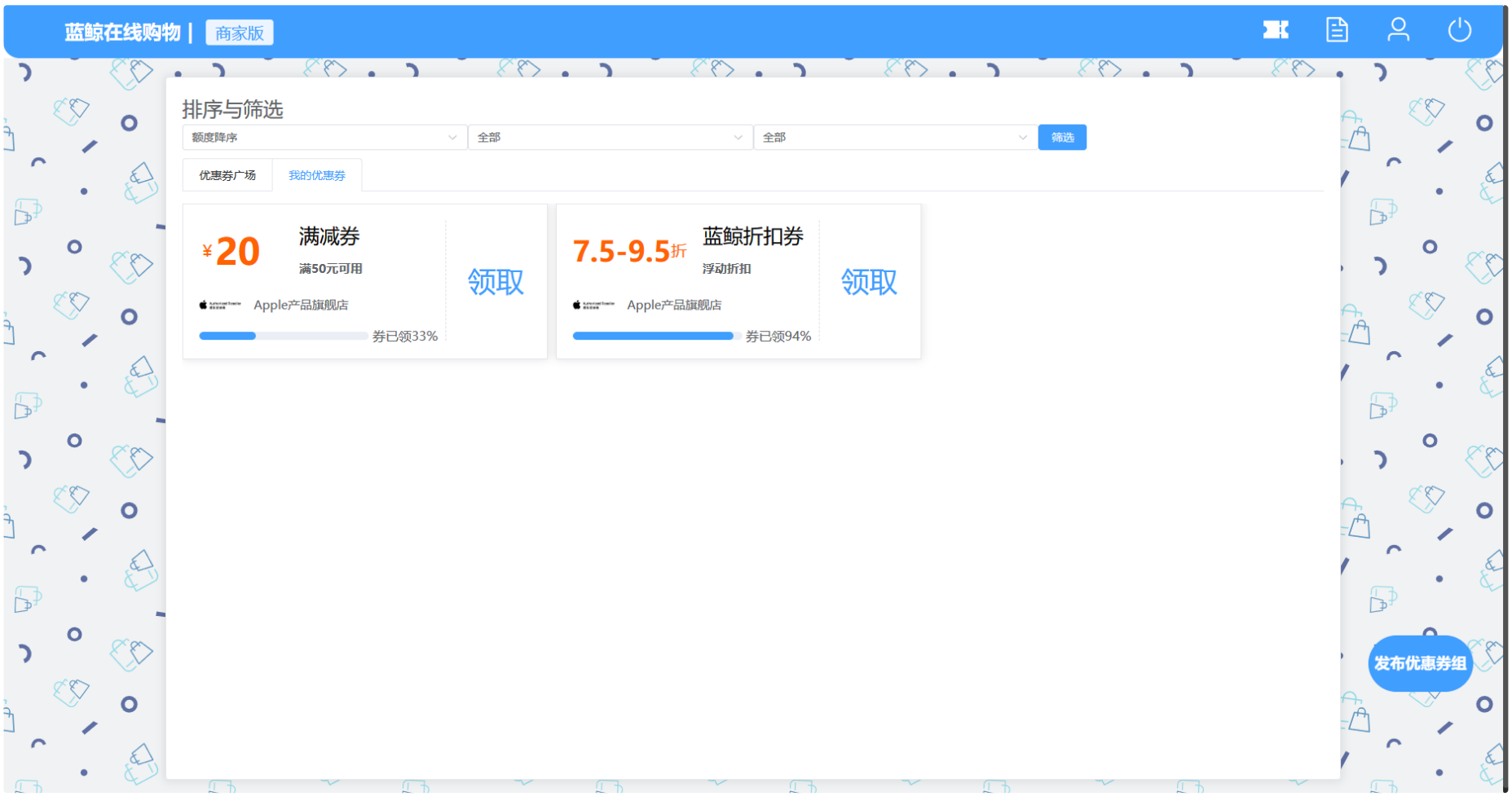
- 我们合并了一下以上需求，做了统一的**优惠券广场**标签页。



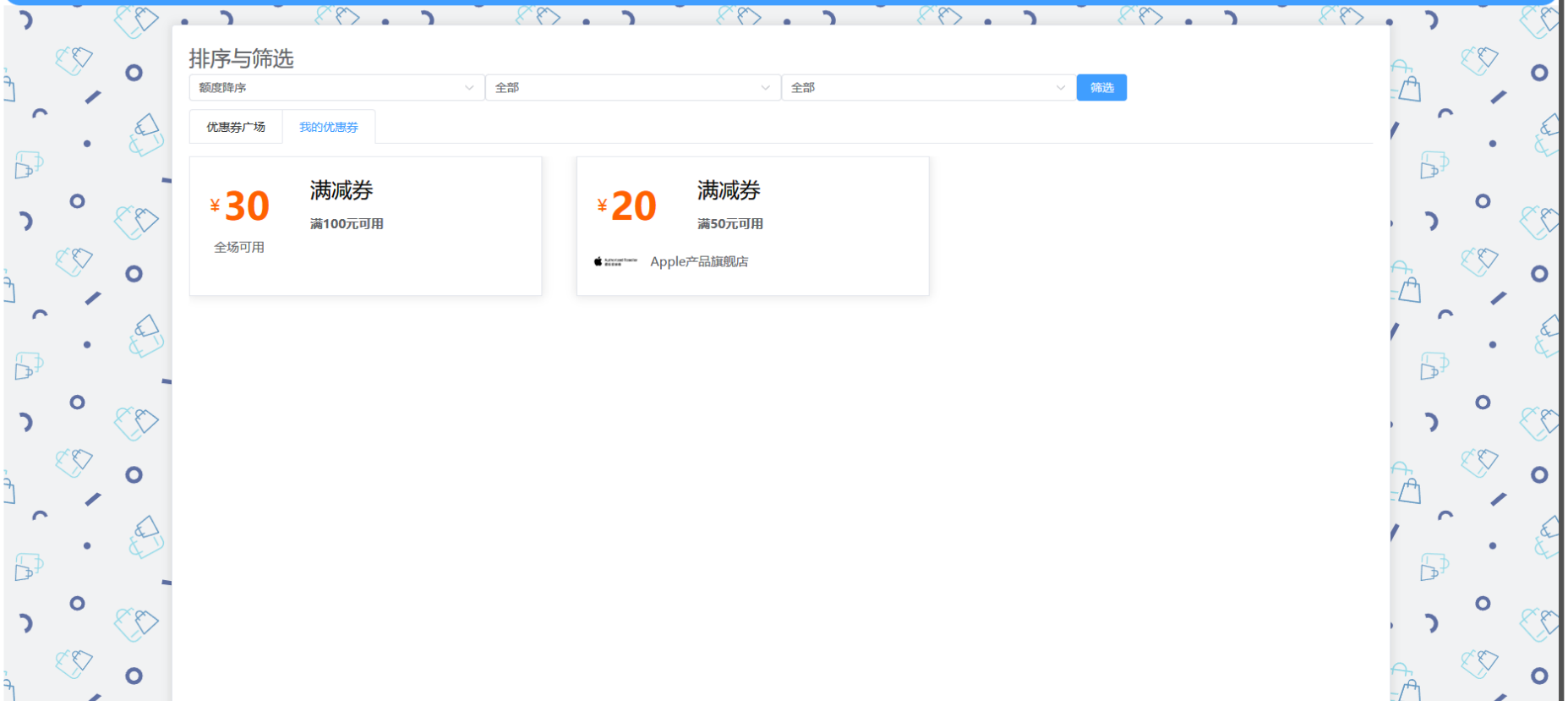

```
<!— 领取进度的实现 —>
<el-progress style="width: 72%" :percentage="(1 - CGInventory / CGCount) *
100.0" :stroke-width="10">
  <span>券已领{{Math.floor(((1 - CGInventory / CGCount) * 100.0))}}%</span>
</el-progress>
```

- 此页面可通过页头的**优惠券图标**跳转

- 优惠券广场中，所有身份的用户都可以看到所有的优惠券以及领取情况。
- **我的优惠券**标签页中，不同身份的用户会看到不同的内容。以下附图为门店工作人员的和顾客的。
  - 顾客：自己领取到的所有**优惠券**
  - 门店工作人员：本商店的**优惠券组**(包含领取情况)
  - CEO：全局**优惠券组**(包含领取情况)



- 顾客可以**领取优惠券**，领取过的优惠券会显示已领取并且不可以再次领取



- 其他身份虽然也能看到领取的按钮，但是点击领取会提示无法领取

- 优惠券和优惠券组可以筛选和排序
  - 可以按额度升序或者降序
  - 可以筛选全局/店铺优惠券、蓝鲸券/满减优惠券
  - 以上规则可以共同作用


```
// 过滤店铺优惠券的部分举例代码
if(listFieldFilter.value === "STORE"){
  for(let i = 0; i < currentList.value.length; i++){
    if(currentList.value[i].storeId == 0){
      currentList.value.splice(i, 1);
      i = i - 1; // 注意删除元素后原列表下标的变化
    }
  }
}
// 排序
if(listOrder.value === "ASC"){
  currentList.value.sort((a, b) => a.reduction - b.reduction)
}else if(listOrder.value === "DSC"){
  currentList.value.sort((a, b) => b.reduction - a.reduction)
}
```

使用优惠券

> 4. 顾客在**支付**订单时，可以查看当前订单可用的优惠券，并选择是否使用优惠券、使用哪张优惠券，且可以实时看到选择该优惠券后的优惠价格。

- 在我们的业务逻辑中，顾客会在创建订单的时候查看当前可用的优惠券
  - 优惠券的展示使用**横向滚动条**实现，顾客有多个优惠券的时候可以拉动滚动条来进行选择


```
<el-scrollbar always>
  <div class="coupon-list">
    <el-button
        v-for="item in couponList"
        :key="item.id"
        :coupon-info="item"
        class="coupon-item"
        :type="couponSelectStatus(item)"
        @click="handleSelectCoupon(item)"
        >
      <coupon-item :coupon-info="item"></coupon-item>
    </el-button>
  </div>
</el-scrollbar>
```

- 点击优惠券可以进行选择，被**选中**的优惠券的边框会由灰色变为蓝色
  - 为了复用优惠券的组件，这里我们直接给优惠券套了一个通过点击切换状态的按钮组件。按钮组件就是优惠券外面那层边框
  - 选中优惠券之后可以查看优惠金额以及总价格。优惠的价格是后端传来的，这样能让前端可以不需要再写一遍计算优惠的业务逻辑


```
// 选择优惠券之后计算价格
function handleSelectCoupon(coupon: CouponInfo) {
  if (selectCouponList.value.includes(coupon)) {
    selectCouponList.value = selectCouponList.value.filter(item => item !==
coupon)
    discountPrice.value = totalPrice.value
  } else {
    selectCouponList.value.push(coupon)
    reqCouponCalculate({
      couponId: coupon.id,
      price: totalPrice.value
    }).then(res => {
      discountPrice.value = res.data.result
    })
  }
}
```

体验优化

Header图标悬浮文字提示

- 虽然header使用了icon来表示跳转信息，但我们觉得单纯的图标还是难以理解图标的跳转目标(如订单列表就很难看出来)。所以给图标添加了文字提示，在光标置于icon上方时文字提示会弹出。
- 使用**el-tooltip**组件实现

```
<el-col :span="1" class="header-icon">
  <el-tooltip content="优惠券" effect="light">
    <router-link to="/coupons" v-slot="{navigate}">
        <el-icon @click="navigate" :size="35" color="white" ><Ticket /></el-
icon>
      </router-link>
    </el-tooltip>
</el-col>
```

评论区图片放大预览

- 可以点击评论区图片进行预览。预览窗口内可以对图片进行放大、缩小、旋转等操作。
- **具体实现**
  - 将原图片容器从原生img换成了**el-image**，并使用element-plus自带的图片预览功能实现。

```
<p v-for="imageUrl in imageUrlList" :key="imageUrl.id">
    <el-image
      :src="imageUrl.url"
      style="width: 100px; height: 100px; margin: 10px"
      :preview-src-list="imageUrlList.map((item) => item.url)"></el-image>
</p>
```

后端具体实现

- 订单(order)-新增字段

字段	类型	备注
coupon_id	List	使用的优惠券id

- 优惠券(coupon)

字段	类型	备注
id	unique identifier & not null & primary key	
userId	int & not null	优惠券所属用户id
groupId	int & not null	优惠券所属优惠券组id
orderId	int	优惠券目前被哪个订单占用了
used	bool	优惠券是否已经被使用了

- 优惠券组(CouponGroup)

字段	类型	备注
id	unique identifier & not null & primary key	
type	enum & not null	enum{FULL_REDUCTION, SPECIAL}
count	int & not null	优惠券上限
storeId	int & not null	优惠券所属商店（-1为全局 ）
threshold	float	优惠券使用门槛
redcution	float	优惠券优惠额度
inventory	int & not null	当前库存

优惠券模块

获取用户优惠券

- 根据用户id返回用户所持有的优惠券列表
- **实现细节**
  - 直接在后端获取当前用户的id并返回其优惠券列表
  - 之前的与用户id相关的方法是前端传用户id，后端再根据此id进行业务逻辑的处理。这样处理不安全，遂改成现在的写法。

```
@Override
    public List<CouponVO> getByUserId() {
        int userId = securityUtil.getCurrentUser().getId();
        List<Coupon> poList = couponRepository.findByUserId(userId);
        List<CouponVO> voList = new ArrayList<>();
        for(Coupon po: poList){
            voList.add(po.toVO());
        }
        return voList;
    }
```

领取优惠券

- 用户领取优惠券时，后端要要处理将优惠券绑定到用户下，并返回该优惠券对应的id。
- **实现细节**
  - 先检验领取的优惠券所在的优惠券组是否存在。
  - 检查用户是否领取过该优惠券组
  - 检查优惠券组库存是否充足
  - 将用户id写入优惠券中，视为领取
  - 更新优惠券组库存

```
public boolean receiveCoupon(int groupId) {
        CouponGroup couponGroup = couponGroupRepository.findById(groupId);
        if (couponGroup == null){
            // 优惠券组不存在
            throw CouponException.couponGroupNotExist();
        }
        // 检查是否已经领取过
        int userId = securityUtil.getCurrentUser().getId();
        if (couponRepository.findFirstByUserIdAndGroupId(userId, groupId)
≠ null){
            // 已经领取过
            throw CouponException.couponAlreadyReceived();
        }
        if (couponGroup.getInventory() ⩽ 0){
            // 库存不足
            throw CouponException.couponGroupInventoryNotEnough();
        }
        // 领取优惠券
        Coupon coupon = couponGroup.toCoupon(userId);
        couponRepository.save(coupon);
        couponGroup.setInventory(couponGroup.getInventory() - 1);
        couponGroupRepository.save(couponGroup);
        return true;
    }
```

计算价格

- 用户选择优惠券后，需要根据订单原价以及优惠券来计算优惠后的价格。
- 价格的计算采用了策略模式和表驱动

```
@Override
public float clacPrice(int couponId, float price) {
    Coupon coupon = couponRepository.findById(couponId).orElse(null);
    if (coupon == null){
        // 优惠券不存在
        throw CouponException.couponNotExist();
    }
    return couponContext.calculate(coupon, price);
}
```

- 策略模式环境类
  - 使用HashMap建立从优惠券组类型到具体策略类的映射，以减少if-else的数量，且便于后续维护。

```
@Service
public class CouponContext {
    Map<CouponTypeEnum, CouponStrategy> strategyMap = new HashMap<>();

    @Autowired
    private ClaculateFullReduction claculateFullReduction;
    @Autowired
    private ClaculateSpecial claculateSpecial;

    // 初始化策略的hashmap
    @PostConstruct
    public void setStrategyMap(){
        strategyMap.put(CouponTypeEnum.FULL_REDUCTION,
claculateFullReduction);
        strategyMap.put(CouponTypeEnum.SPECIAL, claculateSpecial);
    }

    public float calculate(Coupon coupon, float price){
        CouponStrategy couponStrategy = strategyMap.get(coupon.getType());
        return couponStrategy.calculate(coupon, price);
    }
}
```

- 抽象策略以及具体策略

```
// CouponStrategy.java
public interface CouponStrategy {
    float calculate(Coupon coupon, float price);
}

// CalculateFullReduction.java
@Service
public class ClaculateFullReduction implements CouponStrategy{
    @Override
    public float calculate(Coupon coupon, float price) {
        float threshold = coupon.getThreshold();
```

```
        float reduction = coupon.getReduction();
        if (price ≥ threshold)
            return price - reduction;
        else
            return price;
    }
}

// CalculateSpecial.java
@Service
public class ClaculateSpecial implements CouponStrategy{
    // 表驱动
    float[][] driver = {
            {100, 0.95f},
            {200, 0.9f},
            {300, 0.85f},
            {400, 0.8f},
            {500, 0.75f},
            {Float.POSITIVE_INFINITY, 0.7f}
    };
    @Override
    public float calculate(Coupon coupon, float price) {
        for (float[] floats : driver) {
            if (price < floats[0]) {
                return price * floats[1];
            }
        }
        return price;
    }
}
```

优惠券组模块

获取商店优惠券组

- 根据商店id返回该商店的优惠券组List
- 为了简化数据结构，增强统一性，全局的优惠券组的商店id视为0
- **实现细节**

```
 public List<CouponGroupVO> getCouponGroupByStoreId(int storeId) {
        List<CouponGroupVO> couponGroupVOs =
couponGroupRepository.findAllByStoreId(storeId);
        return couponGroupVOs;
    }
```

发布优惠券组

- 店铺工作人员和经理可以发布优惠券组
- **实现细节**
  - 获取用户role及其所属商店id
  - 店铺工作人员可以发布所属商店的优惠券组
  - 经理可以发布全局优惠券组

```
public int postCouponGroup(CouponGroupVO couponGroupVO) {
        User user = securityUtil.getCurrentUser();
        CouponGroup couponGroup = couponGroupVO.toPO();
        //验证身份
        if(user.getRole() == RoleEnum.STAFF){
            //STAFF可发布所属商店优惠券组
            couponGroup.setStoreId(user.getStoreId());
        } else if (user.getRole() == RoleEnum.CEO) {
            //CEO可发布全局优惠券组
            couponGroup.setStoreId(0);
        }
        else{
            throw AuthorityException.notAllowToOperate();
        }
        couponGroup.setInventory(couponGroupVO.getCount());
        CouponGroup couponGroupSaved =
couponGroupRepository.save(couponGroup);
        return couponGroupSaved.getId();
    }
```

测试模块

订单预算价格

- 测试模板

```
@Autowired
CouponService couponService;
@Autowired
CouponContext context;
@MockBean
CouponRepository couponRepository;

@Test
void clacPriceTest() {
    Coupon coupon = new Coupon();
    coupon.setId(1);
    coupon.setUserId(1);
    coupon.setGroupId(1);
    coupon.setType(CouponTypeEnum.SPECIAL/CouponTypeEnum.FULL_REDUCTION);
```

```
 Mockito.when(couponRepository.findById(1)).thenReturn(Optional.of(coupon));
    assert couponService.clacPrice(couponId,price) == targetPrice;
}
```

- 通过调整测试中优惠券的类型和价格，可以测试不同类型优惠券对不同价格的优惠价格是否与预期相符
- 使用Mockito模拟了CouponRepository的接口，可以使测试的业务逻辑不依赖真实的接口和类，简化逻辑
- **蓝鲸券**

  共7个测试用例，分别对对每个价格区间以及整百的边界进行了测试
- **满减券**

  共2个测试用例，分别计算价格到达优惠券使用门槛以及达不到使用门槛的情况