芋道源码 —— 知识星球

我是一段不羁的公告！
记得给芳芳这 3 个项目加油，添加一个 STAR 噢。
https://github.com/YunaiV/SpringBoot-Labs
https://github.com/YunaiV/onemall
https://github.com/YunaiV/ruoyi-vue-pro
2018-03-10
数据库实体设计

数据库实体设计 —— 交易（2.3）之物流信息（上门自提）

> 芳芳目前正在做一个开源的电商项目，胖友可以 star 下。
> https://gitee.com/zhijiantianya/onemall

1. 概述

本文接 《数据库实体设计 —— 交易（2.2）之物流信息（快递发货）》，主要分享交易模块的物流信息的上门自提的数据库实体设计。

基于如下信息，逆向猜测数据库实体：

> 有赞微商城的上门自提
> 用于核销到店自提订单 API

> 【护脸旁白】
> 笔者非电商行业出身 && 非有赞工程师，所以有错误或不合理的地方，烦请斧正和探讨。
> 有赞是个各方面都很 NICE 的公司，推荐 。

2. 背景了解

整体参见 《上门自提使用和核销操作说明》 文档。

2.1 买家上门自提功能开关

需要打开开关，用户下单才可以自行选择上门自提。

配置界面如下：

2.2 买家上门自提点管理

界面如下：

1. 列表

2. 添加

添加自提点

* 自提点名称：

请填写自提点名称便于买家理解和管理

* 自提点地址：

选择省份

选择城

请填写自提点的具体地址，最短5字，

* 地图定位：

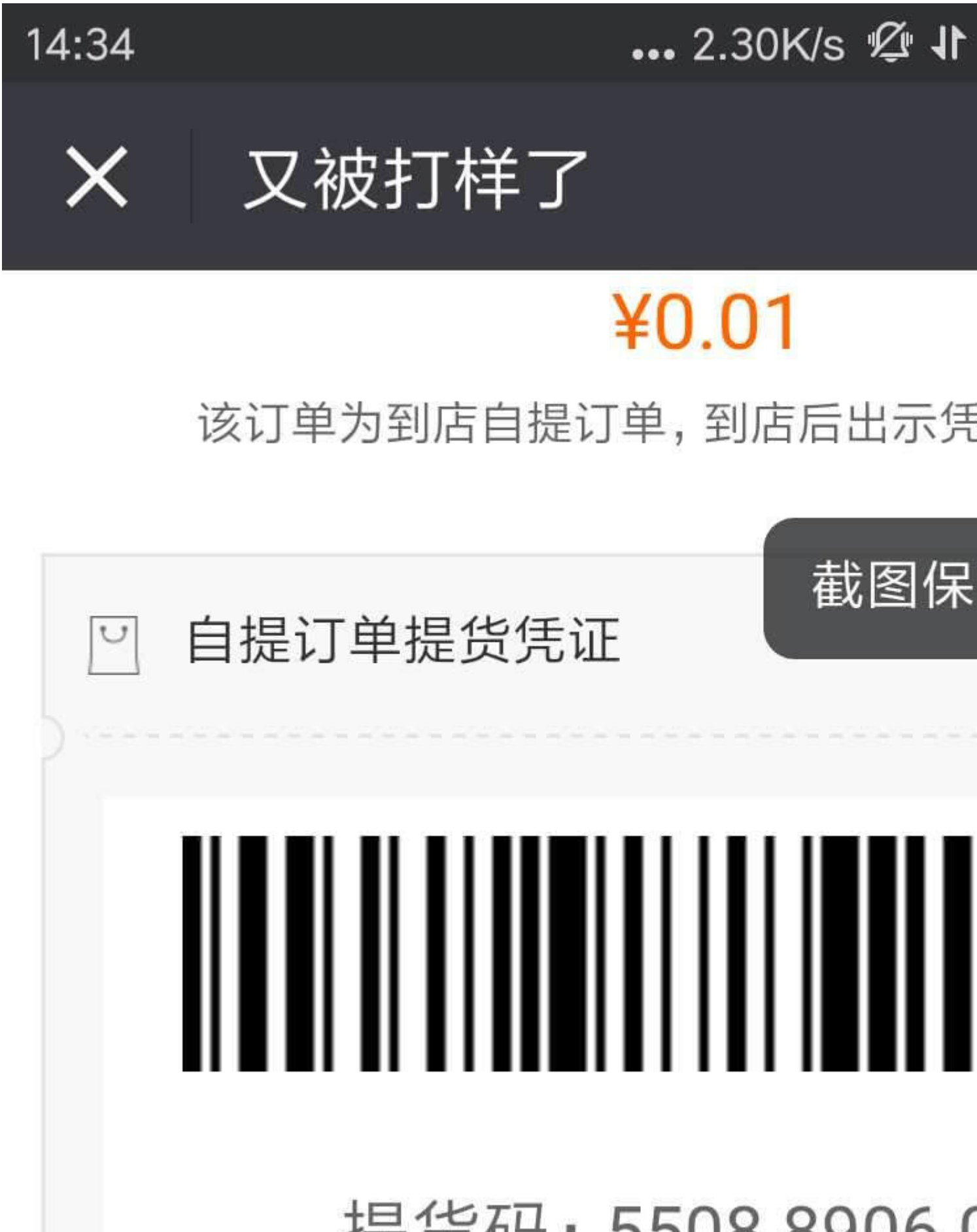
2.3 买家下单选择自提点

界面如下：

2.4 买家交易订单核销二维码

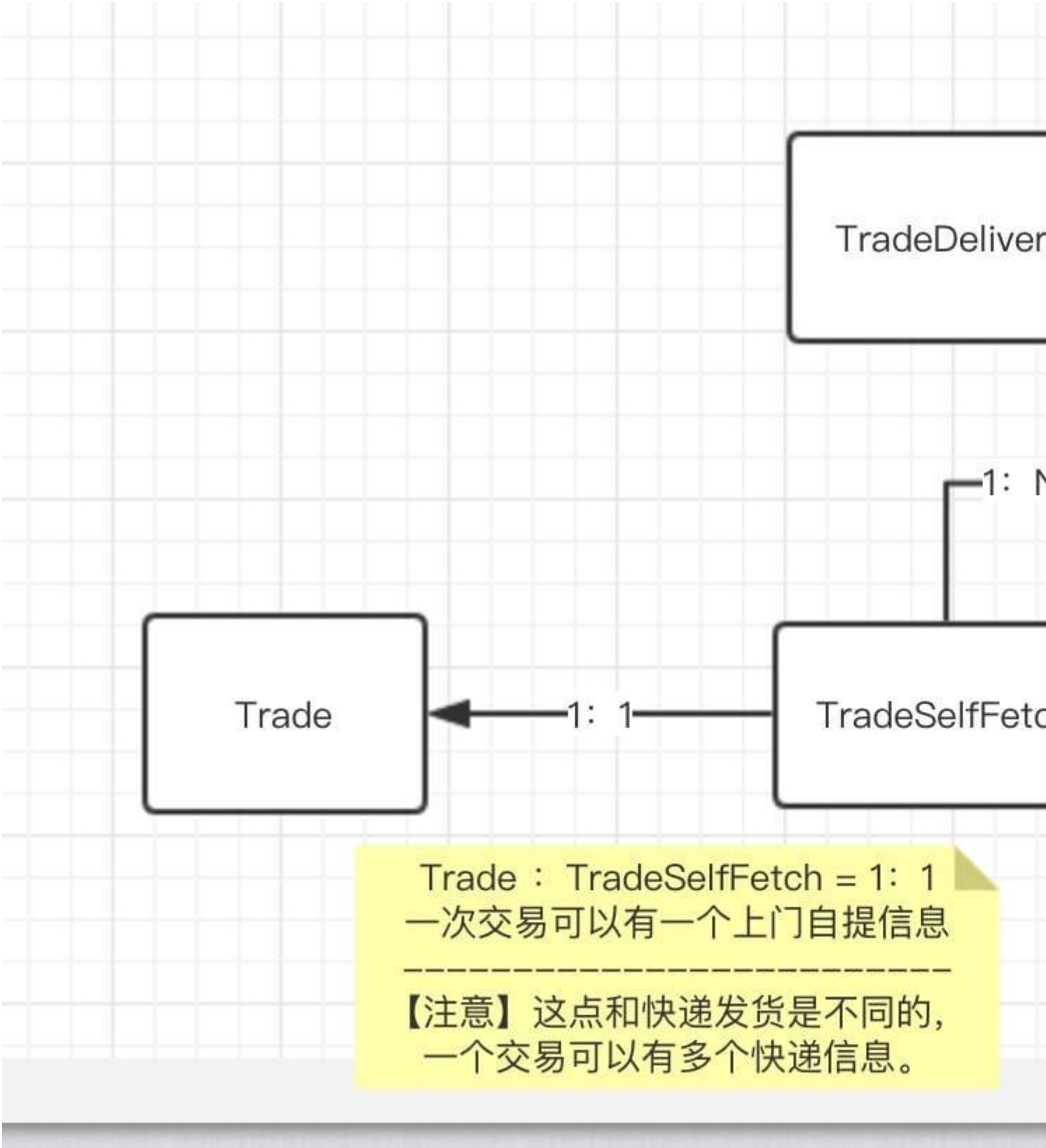
卖家通过该二维码，核销上门自提订单。

界面如下：



3. 数据库实体

整体实体类关系如下图：



3.1 TradeDeliverySetting

在 TradeDeliverySetting 中，isSelf 标记买家上门自提功能开关是否开启，在 《数据库实体设计 —— 交易（2.2）之物流信息（快递发货）》「3.3 TradeDeliverySetting」 有详细解析。

3.2 TradeDeliverySelfFetchAddress

TradeDeliverySelfFetchAddress ，交易买家上门自提点。

```
/**
 * 自提点编号
 */
private Integer id;
/**
 * 店铺编号 {@link cn.iocoder.doraemon.shopgroup.shop.entity.Shop#id}
 */
private Integer shopId;
/**
 * 自提点名
 */
private String name;
/**
 * 商家推荐（描述）
 */
private String description;
/**
 * 自提点照片数组，以逗号分隔。
 */
private String images;
/**
 * 联系电话 - 区号
 */
private String phone1;
/**
 * 联系电话 - 电话或手机
 */
private String phone2;
/**
 * 状态
 *
 * 1-正常
 * 2-删除
 */
private Integer status;
/**
 * 创建时间
 */
private Date createTime;
/**
 * 更新时间
 */
private Date updateTime;
/**
 * 删除时间
 */
private Date deleteTime;

// ========== 地点相关 BEGIN ==========
/**
 * 地点编号 {@link cn.iocoder.doraemon.commongroup.entity.CommonRegion#id}
```

```
     *
     * 最细维度
     */
    private Integer regionId;
    /**
     * 地址
     */
    private String address;
    /**
     * 经度
     */
    private Double lng;
    /**
     * 纬度
     */
    private Double lat;

    // ========== 地点相关 END ==========

    /**
     * 是否作为门店接待
     */
    private Boolean isStore;
    /**
     * 接待时间集合
     *
     * 使用 JSON 将 {@link DateRange}数组 格式化成字符串
     */
    private String businessHoursAdvanceds;
    /**
     * 是否需要买家选择自提时间
     *
     * true 勾选后，买家下单选择上门自提，必须选择自提时间，卖家需要按约定时间备货。
     * false 不勾选，将会提示买家尽快到店自提
     */
    private Boolean isOptionalSelfFetchTime;
    /**
     * 自提时间集合
     *
     * 使用 JSON 将 {@link DateRange}数组 格式化成字符串
     */
    private String offlineBusinessHours;
```

`id` ，自提点编号，自增。
`shopId` ，店铺编号，指向关联的店铺。Shop ： TradeDeliverySelfFetchAddress = 1：N 。一个店铺可以有多个自提点。
`status` ，状态。删除自提点时，标记 `status = 2` 删除。

自提时间相关字段

- `isOptionalSelfFetchTime` ，是否需要买家选择自提时间。
  - `true` ，勾选后，买家下单选择上门自提，必须选择自提时间，卖家需要按约定时间备货。
  - `false` ，不勾选，将会提示买家尽快到店自提，即不用选择上门自提时间。
- `businessHoursAdvanceds` ，自提时间集合。使用 JSON 将 DateRange 数组格式化成字符串存储。DateRange 代码如下：

```
/**
 * 开始时间。
 *
 * 格式为 HHdd ，例如 0730
 */
private Integer openTime;
/**
 * 结束时间。
 *
 * 格式为 HHdd ，例如 2300
 */
private Integer closeTime;
/**
 * 周几集合
 *
 * x 为 周x ，例如，1 = 周一 。
 */
private List<Integer> weekdays;
```

- x

门店接待相关字段，本文暂无相关逻辑。

- isStore ，是否作为门店接待。
- businessHoursAdvanceds ，接待时间集合。使用 JSON 将 DateRange 数组格式化成字符串存储。

3.3 TradeSelfFetch

TradeSelfFetch ，交易上门自提信息。

```
/**
 * 交易编号 {@link Trade#id}
 */
private String id;
/**
 * 店铺编号 {@link cn.iocoder.doraemon.shopgroup.shop.entity.Shop#id}
 */
private Integer shopId;
/**
 * 自提人的手机
 */
private String fetcherMobile;
/**
 * 自提人的名字
 */
private String fetcherName;
/**
 * 自提开始时间
 *
 * 当为空时，任意时间都可自提。
 */
private Date fetchStartTime;
/**
 * 自提结束时间
 *
```

```
     * 当为空时，任意时间都可自提。
     */
    private Date fetchEndTime;
    /**
     * 自提提货码，例如 5508 8906 062
     */
    private String fetchCode;
    /**
     * 上门自提点的编号 {@link cn.iocoder.doraemon.tradegroup.delivery.entity.TradeDeliverySelfFetchAddress#id}
     */
    private Integer fetchAddressId;
    /**
     * 上门自提点的地区编号
     *
     * 冗余，避免删除
     */
    private Integer fetchRegionId;
    /**
     * 上门自提点的地址
     *
     * 冗余，避免修改
     */
    private String fetchAddress;
    /**
     * 上门自提点的联系电话
     *
     * 冗余，避免修改
     */
    private String fetchPhone;
```

id ，交易编号，指向关联的交易。Trade ： TradeSelfFetch = 1：1 。一个交易可以有一个交易上门自提信息。
shopId ，店铺编号，指向关联的店铺。
自提人相关的信息

- fetcherMobile ： 自提人的手机。
- fetcherName ： 自提人的名字。

fetchStartTime ~ fetchEndTime ，自提时间区间范围。若字段为空时，表示选择的自提点 TradeDeliverySelfFetchAddress.isOptionalSelfFetchTime = false 。即，买家无需选择自提时间，直接上门提货。
fetchCode ，自提提货码，例如 5508 8906 062 。买家使用提货码到自提点，商家输入提货码进行核销，进行确认收货。
自提点相关信息，冗余相关字段，避免商家修改自提点的信息，导致和买家下单时选择的自提点信息存在差异。

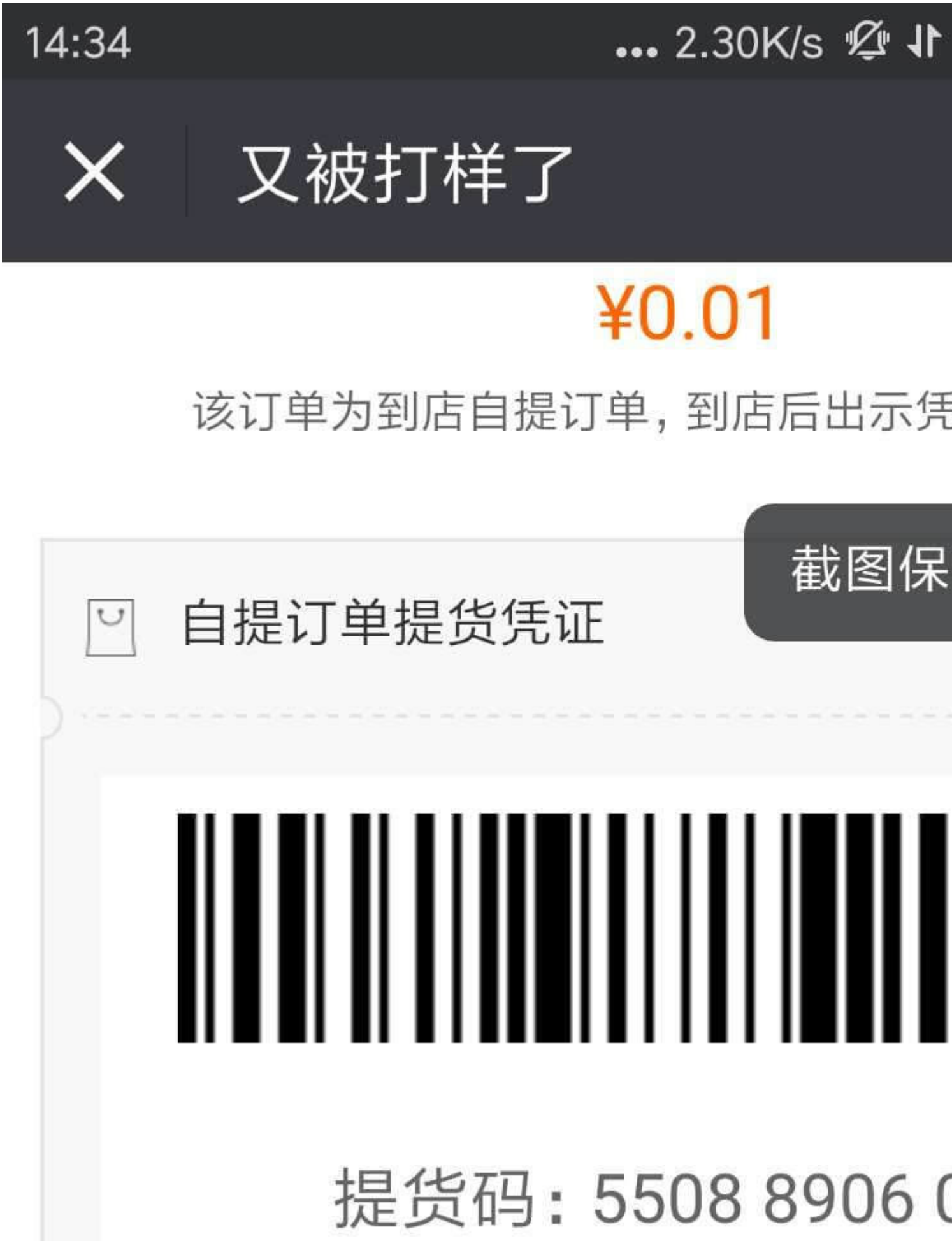
- fetchAddressId ，上门自提点的编号，指向关联的 TradeDeliverySelfFetchAddress 。
- fetchRegionId ，冗余，上门自提点的地区编号。
- fetchAddress ，冗余，上门自提点的地址。
- fetchPhone ，冗余，上门自提点的联系电话。

我们回过头来看 Trade.shippingType 字段，创建交易时的物流方式。这有什么特殊的地方么？注意下，这是创建交易时的物流方式（是不是有点傻，在重复）。实际场景下，买家可以和商家沟通，即使买家下单时选择的是上门自提，实际也可以物流发货。

3.4 交易订单核销二维码

卖家使用统一的扫一扫功能，扫描二维码，根据扫描出的内容进行相应的处理。那么问题就来了，买家在不同的业务模块，会有多种二维码，卖家如何使用统一的扫一扫进行处理的呢？

我们来解析一个上门自提的二维码：



结果是，`selfFetch55088906062` ，即 `selfFetch` + 上门自提提货码。

也就是说，不同的业务模块，通过不同的前缀进行区分和处理。

我们来看看下面的图，就容易理解了

4. API

基于如下整理 API 类。

- 用于核销到店自提订单 API

4.1 TradeCodeAPI

TradeCodeAPI ，交易核销 API 。

666. 彩蛋

没有彩蛋，继续怼下一篇文章。

有点嗨皮。

文章目录

回到首页