用户微店需求概要设计

# 需求概述

主要需求如下：

1. 通过公众号“我的微店”进入分销商的微店页面，需要判读用户是否已绑定。如果没有则显示登录注册页面。
2. 第一次进入微店时需要通过账号密码登录，或者注册新用户。老用户第一次登录成功后，要绑定微信openid。新用户注册成功后就直接绑定微信openid。
3. 微信绑定成功，以后进入微店不需要再输入密码登录。只要是绑定了微信就可通过openid进行登录操作，包括获取用户信息，生成token，更新缓存。达到用户一直是登录状态的效果，除非解除绑定，否则永远是不能退出的。
4. 通过公众号“我的微店”进入微店时，服务端可以获取到openid，可以此识别身份，而在微店内操作时，只能使用token来识别身份。且不能因其它因素导致重新登录（比如修改密码）。
5. 老用户绑定登录成功后，或新用户审核通过后，就可在店铺管理页修改保存微店信息。
6. 审核通过的新用户，会创建默认值的微店信息，同时将审核通过的消息推送到微信消息上。老用户需要SQL统一生成，只分销商类型。



注册来源的类型如下：

1. PC端注册
2. APP注册
3. 微信
4. 移动端浏览器

微店默认信息如下：

* 头像，默认用魔方旅游logo。
* 名称：魔方旅游微店。
* 简介：淳朴地道的本地人，给你最具性价比的本地游。

# 设计方案

需求主要是关于两方面的：一是用户和微店信息维护，二是从微店登录并一直保持登录状态。

关于新字段，微店信息不属于用户基本信息，是新的领域模型，所以需要新建微店表，关联用户id，openid的作用是绑定账号，及认证账号有效性的，注册来源应属于用户基本信息，所以这两个字段增加到用户表上。

修改点如下：

* 用户表、实体上要增加openid和注册来源2个字段，创建和修改用户接口要支持这2个字段。
* 审核用户时发送消息。审核通过时创建默认的微店信息，需要提供单独审核接口。
* 修改、查询微店信息接口。
* 通过openid登录，生成token，重新加载用户信息到缓存，需要新的登录接口。
* 解绑openid，删除登录时用户的缓存。

关于openid登录，由于openid能表示一个合法的微信用户，只要能找到绑定的用户信息，就可执行登录成功,相当于永远都不能退出，除非解绑。

## 绑定用户微店信息接口

为用户绑定微信时，只需将openid记录到用户信息的openid字段上。解绑就是置为空字符。这只是涉及两个用户上的接口：

**创建用户接口**

入参：原接口参数上新增openid、注册来源。

验证：注册来源不能为空。

出参：新用户id。

**更新用户接口**

入参：原接口参数上新增openid、注册来源。

验证：用户id不能为空 。

出参：受影响行数。

## 审核用户通过接口

审核通过时要生成默认的微店信息，同时要发送消息。

入参：用户id。

验证：用户id不能为空。

出参：ture：操作成功，false：操作失败。



## 更新用户微店信息接口

第一次绑定微信并审核通过后需要创建默认的微店信息，以后只能更新微店信息。

入参：用户id，微店信息（名称、简介、头像）。

验证：用户id、微店信息不能为空。

出参：ture：操作成功，false：操作失败。

## 根据userId获取微店信息接口

入参：微信信息所属用户id

验证：用户id不能为空。

出参：精确匹配一个微店信息。

## 根据openid登录接口

只要通过openid能找到一个用户，就能登录成功。由于要求其它地方修改密码后，微店不用重新登录，所以要使用md5加密openid生成token，保证token不受密码修改而失效 。

入参：微信的openid。

验证：openid不能为空。

出参：opneid所属的用户信息。



## 解绑用户微信

处理两个事，一个是将openid置为空，另一个是将登录的缓存删除。

入参：通过openid登录成功会生成的token。

验证：token不能为空。

出参：ture：操作成功，false：操作失败。



# 数据结构

用户表（sys\_user）：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 说明 |
| id | bigint | 20 | 逻辑主键 |
| wx\_openid | varchar | 50 | 微信的openId |
| reg\_source | int | 2 | 注册来源（1、PC端注册；2、APP注册；3、微信；4、移动端浏览器） |

用户微店信息表（user\_microshop）：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 说明 |
| id | bigint | 20 | 逻辑主键 |
| user\_id | bigint | 20 | 用户id |
| name | varchar | 64 | 微店名称 |
| intro | varchar | 500 | 微店简介 |
| avatar | varchar | 20 | 微店头像 |
| create\_time | datetime |  | 创建时间 |
| update\_time | datetime |  | 更新时间 |

# 数据同步

Sql批量插入数据。只对所有分销商生成。头像需事先在图片服务器上传，得到url。

# 消息发送

## 用户审核结果消息

|  |  |  |  |
| --- | --- | --- | --- |
| 主题 | customer | Tag | customer\_check |
| 格式 | **名称** | **类型** | **说明** |
| id | Long | 被审核用户id |
| checkType | Integer | 审核类型，2是用户审核，3是资质审核 |
| checkTime | Date | 审核时间 |
| checkStatus | Integer | 审核状态，1审核通过，2审核拒绝 |
| reason | String | 原因 |
| 示例 | {  id:123,  checkTime:'2016-12-12 18:30:15',  checkStatus: 3,  reason: "信息不全",  chekcType : 2  } | | |

当审核通过时，原因会默认为“符合规范”。任何结果都会发送消息，包括成功和拒绝。

## 微店订单支持完成消息

|  |  |  |  |
| --- | --- | --- | --- |
| 主题 | trade | Tag | orderPaiedEventMsg |
| 格式 | **名称** | **类型** | **说明** |
| saleOrderId | String | 销售订单ID |
| transactionId | String | 交易ID |
| totalAmount | Double | 订单总金额 |
| salePort | Integer | 销售端口 7微店 9 app 3 pc分销 |
| is\_direct | Integer | 是否直签 1-直签，2-非直签（即魔方) |
| is\_dock | Integer | 是否对接商品. 0: 否; 1: 是 |
| 示例 | {  saleOrderId:"",//String 销售订单ID  transactionId:"",//String 交易ID  totalAmount:0.0//double 订单总金额  salePort:9,//int 销售端口 7微店 9 app 3 pc分销  is\_direct:2//int 是否直签 1-直签，2-非直签（即魔方)  is\_dock:0//int 是否对接商品. 0: 否; 1: 是  } | | |