# 账户微服务概要设计文档

V1.0版

# 背景

威尔公司现有平台应用较多，分布较分散，平台应用之间账号独立，具有很强的数据壁垒。缺乏一个统一的账户管理出入口。在与第三方合作伙伴进行对接的时候存在账户混乱，一个平台应用需要对接一次第三方合作伙伴。且账户创建过多，存在的很大的数据冗余，没有一个统一的服务输出可以全方位的监控管理统一账户在威尔多平台应用下的行为。账户的分散不利于有效的收集数据，不利于将来大数据下的用户行为分析的数据沉淀。

# 需求分析

## 一、现企业一卡通账户信息分析

1. 目前没有实现的多企业账户主要分为两类：超级管理员组、区域管理员、子企业超级管理员和部门管理员和用户。
2. 单企业账户主要分为三类：超级管理员和部门管理员和用户
3. 权限主要分为区域权限（多企业）、企业权限（多企业）、子应用权限、部门权限、设备权限、菜单权限

## 二、现基教、高教账户信息分析

1. 账户主要分为两类：超级管理员（系统管理人员和高级行政人员）和普通用户（教师）
2. 权限主要分为两类：功能权限（场所、部门等的门禁和考勤控制）和数据权限（数据库的增删改）

# 达成目标

1. 平台应用不再独立建立账户体系，由账户微服务统一管理
2. 每个平台应用具有自己的独立应用id
3. 平台应用可以在账户微服务中查询、维护自己平台应用下的所有账户信息
4. 平台应用可以在账户微服务中查询、维护自己平台应用下的所有角色信息
5. 可查询某个账户在所有威尔平台应用中的使用情况
6. 开发注册接口，平台应用通过调用注册接口完成平台应用的账户注册等功能。
7. 提供找回密码等账户平台的基础接口

# 四、概要设计

## 一、基本功能

1. **应用入驻识别：**一个新的应用进入到微服务系统平台时，如果系统认证通过，应用会获得一个唯一识别码（应用id).账户服务平台通过应用id判断应用是否具有进入账户服务平台的权限。如果具有权限进入成功否则拒绝。
2. **角色信息维护：**每一个应用id在账户服务平台上创建一个角色列表，主要包括应用id，角色名称，权限码等信息；平台应用可通过RESTAPI维护角色列表信息，包含新增，修改，删除，查询功能。
3. **人员信息维护：**每一个应用id在账户服务平台上创建一个人员列表，主要包含应用id，账户的基本信息，角色编码等信息；平台应用可通过RESTAPI维护人员列表信息，包含新增、修改、删除、查询，修改又包含修改密码等功能。
4. **角色和人员信息维护：**不同的应用id会包含不同的角色以及权限码。在添加人员的过程中，为每一个人员分配一个或多个角色，并且这个角色必须存在于对应的应用id的角色列表中，平台应用可通过RESTAPI修改角色和人员的关系。
5. **查询用户信息：**已知某个用户,可以从人员列表中得到这个用户的全部账户信息，包含基本信息、角色信息和权限信息。**注意**：这里提及的权限都是一个字符串的形式的权限码，应用端应该维护一张权限码和具体权限信息的对应表。账户服务平台主要是做一些账户信息的存储和维护。什么样的用户拥有怎样的角色和权限主要由应用端决定。

## 二、基本流程



平台应用入驻流程



账户添加流程

## 三、软件架构设计

数据库：采用postgre，以jsonb的形式存数数据

开发框架：以spring boot作为基础开发框架，spring cloud作为基础微服务开发框架。

非关系型数据库：使用redis作为非关系型数据库，存储，缓存平台应用的角色表。

# 接口详细设计

说明：

1. 所有接口均是基于HTTP协议的RESTFUL风格接口。
2. 加入账户修改通知订阅功能。

## 一、版本说明

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 时间 | 编写人员 | 说明 |
| V1.0.0 | 2017/12/20 | 曹赫洋 | 起稿概要设计 |

## 二、环境说明

|  |  |
| --- | --- |
| 环境 | 地址 |
| 内网测试环境 | 待定 |
| 线上测试环境 | 待定 |
| 线上正式环境 | 待定 |

## 三、接口说明

接口的返回值结构统一如下，每个接口中只在对data部分做说明

|  |
| --- |
| 示例：  {  “code”:0, //返回码 0代表成功 0 以下代表失败 0 以上代表成功  “msg”:”成功”, //状态码说明  “data”:{ }, //接口的具体返回业务体。可能是json 也可能是某一数据类型  } |

## 接口设计

修改备注：

1.增加与基础账户搜索相关的双重搜索条件：账户id/账户编号。

### 新增基础账户接口

1. 暂时所有的应用都可以添加基础账户，后期会加强服务的访问权限，只有开通了该接口使用权限的应用才可以调用该接口；
2. 手机号已存在、邮箱已存在、身份证书号码已存在都认为该基础账户已存在，返回失败；
3. 接口要对添加账户的请求数据做合法性校验；
4. 接口为PUT JSON 方式；
5. 基础账号信息表中有一列是平台来源，标识了该基础数据是哪个平台添加的，通过请求带的令牌在服务网关中获取到对应的appno

**接口说明：**

**接口地址：/basic\_account/add**

**HTTP动词：PUT**

**参数定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 是否为空 | 说明 | |
| id\_card | string | | Y/N | | 身份证号码，身份证号码，手机号码1，邮箱1三者中至少要有一个 |
| name | string | | Y | | 真实姓名 |
| sex | int | | N | | 性别 0男1女 |
| mobilephone | string | | Y/N | | 手机号码1，身份证号码，手机号码1，邮箱1三者中至少要有一个 |
| mobilephone2 | string | | Y | | 手机号码2 |
| telephone | string | | Y | | 固话 |
| email | string | | Y/N | | 邮箱1，身份证号码，手机号码1，邮箱1三者中至少要有一个 |
| email2 | string | | Y | | 邮箱2 |
| inserted\_by | string | | Y | | 添加人 默认值为system |
| 示例：  {  “id\_card”:”xxxxxxxxxxxxxxxxxxxxxxxxx”,  “name”:”zhangsan”,  “sex”:0,  “mobilephone”:”132XXXXXXXX”,  “mobilephone2”:”134XXXXXXXX”,  “telephone”:”0535-802XXXX”,  “email”:”232@163.com”,  “email2”:”2X2@163.com”,  “inserted\_by”:NULL  } | | | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| basic\_acc\_no | int | N | 账户编号，添加失败则为0 |

### 新增应用账户接口

1. 当应用账户信息已经存在的情况下，提示添加应用账户信息失败；
2. 每一个账户只可以有一份基础账户信息，每一个账户在每一个应用下只可以有一个应用账号信息；
3. 应用账户信息中有一个字段是应用id，标识了该应用账户信息属于哪个应用；
4. 接口要对添加账户的请求数据做合法性校验；
5. 每个应用账户都有自己的密码；
6. 需要根据添加应用传递的数据对应用账户状态做初始化，正常应该是0
7. 接口为PUT JSON 方式；
8. 新增人为应用传递来的；
9. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口；

**接口说明：**

**接口地址：/account/add**

**HTTP动词：PUT**

**参数定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 是否为空 | 说明 | |
| basic\_acc\_no | int | | N | | 基础账户编号 |
| app\_account | string | | N | | 应用账户,用一个应用下的应用账户具有唯一性 |
| label | json | | Y | | 应用账户标签,json格式，根据平台入驻时与账户微服务预定义的标签，由平台应用自行存储 |
| password | string | | N | | 应用账户密码 |
| role\_nos | string | | Y | | 角色编码,以”;角色码1;角色码2;”形式存储 |
| inserted\_by | string | | Y | | 添加人 默认值为system |
| column\_1 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_2 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_3 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_4 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_5 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| 示例：  {  “basic\_acc\_no”:122222,  “app\_account”:”0001”, //应用账户,用一个应用下的应用账户具有唯一性  “label”:[“class”,”tearcher”],  “password”:”DDDDCVCCEREWR432DC”,  “role\_nos”:”;0001;0003;”,  “inserted\_by”:”admin”,  “column\_1”:NULL,  “column\_2”:NULL,  “column\_3”:NULL,  “column\_4”:NULL,  “column\_5”:NULL  } | | | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 新增成功失败，true表示成功，false表示失败 |

### 修改基础账户信息接口

1. 以应用id+账户编号定位到一个账户应用信息，某平台想要修改某基础账户信息，首先该平台必须有该基础账户信息对应的账户应用信息存在才可以；
2. 平台应用A默认只可以修改基础账号信息表中平台来源是平台应用A的基础账户信息
3. 在威尔管理后台中，可以给应用配置基础账户管理权限，具有基础账户管理权限的应用可以修改基础账户的数据（未来规划，现在不做
4. 需要对修改的基础账户信息合法性校验；
5. 接口为POST JSON 方式；
6. 更新人为平台传过来的；

**接口说明：**

**接口地址：/basic\_account/modify**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 是否为空 | 说明 | |
| basci\_acc\_no | int | | N | | 账户编号 |
| id\_card | string | | Y/N | | 身份证号码，身份证号码，手机号码1，邮箱1三者中至少要有一个 |
| name | string | | Y | | 真实姓名 |
| sex | integer | | Y | | 性别 0男1女 允许为NULL NULL不修改性别 |
| mobilephone | string | | Y/N | | 手机号码1，身份证号码，手机号码1，邮箱1三者中至少要有一个 |
| mobilephone2 | string | | Y | | 手机号码2 |
| telephone | string | | Y | | 固话 |
| email | string | | Y/N | | 邮箱1，身份证号码，手机号码1，邮箱1三者中至少要有一个 |
| email2 | string | | Y | | 邮箱2 |
| updated\_by | string | | Y | | 修改人 默认值为system |
| 示例：  {  “basic\_acc\_no”:13,  “id\_card”:”xxxxxxxxxxxxxxxxxxxxxxxxx”,  “name”:”zhangsan”,  “sex”:0,  “mobilephone”:”132XXXXXXXX”,  “mobilephone2”:”134XXXXXXXX”,  “telephone”:”0535-802XXXX”,  “email”:”232@163.com”,  “email2”:”2X2@163.com”,  “updated\_by”:NULL  } | | | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 修改成功失败，true表示成功，false表示失败 |

### 修改应用账户信息接口

1. 以应用id+账户编号定位到一个账户应用信息，根据用户传递的数据，对非空字段进行修改；
2. 应用只可以修改应用账户信息部分的数据；
3. 需要对修改账户的请求数据做合法性校验；
4. 接口为POST JSON 方式；
5. 更新人为平台传过来的

**接口说明：**

**接口地址：/account/modify**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 是否为空 | 说明 | |
| basci\_acc\_no | integer | | Y/N | | 账户编号 账户编号与应用账户至少一个不为空,用来锁定修改的哪个应用账户的，这一列的值不可以修改 |
| app\_account | string | | Y/N | | 应用账户 账户编号与应用账户至少一个不为空，用来锁定修改的哪个应用账户的，这一列的值不可以修改 |
| label | json | | Y | | 应用账户标签,json格式，根据平台入驻时与账户微服务预定义的标签，由平台应用自行存储 |
| role\_nos | string | | Y | | 角色编码,以”;角色码1;角色码2;”形式存储 |
| column\_1 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_2 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_3 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_4 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| column\_5 | string | | Y | | 备用标注,应用自定义存储，自行解析 |
| updated\_by | string | | Y | | 修改人 默认值为system |
| 示例：  {  “basic\_acc\_no”:13,  “app\_account”:”000X1”,  “label”:[“class”,”tearcher”],  “role\_nos”:”;0001;0003;”,  “column\_1”:NULL,  “column\_2”:NULL,  “column\_3”:NULL,  “column\_4”:NULL,  “column\_5”:NULL,  “updated\_by”:NULL  } | | | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 修改成功失败，true表示成功，false表示失败 |

### 维护基础账户身份识别介质接口

1. 根据基础账户表的账户INT编号维护账户的JSONB格式的识别介质；
2. 每次修改之后都要把介质版本号+1；
3. 修改完成功之后要通过mq通知通讯服务新的介质版本号；
4. 接口为POST HTTP 格式；

**接口说明：**

**接口地址：/basic\_account/modify\_media**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| basci\_acc\_no | int | N | 账户INT编号 |
| iden\_medianame | string | N | 识别介质名称 |
| iden\_mediavalue | char | N | Y或者N，Y代表开启该识别方式，N代表关闭，目前总用有11中识别介质，分别是hand,finger\_0到finger\_9，默认在添加账户的时候这11中识别介质的默认值都是N |
| 示例：  {  “basci\_acc\_no”:12, //账户INT编号  “iden\_medianame”:”hand”,//识别介质名称  “iden\_mediavalue”:’Y’ //开启该识别介质的识别方式  } | | | |

**接口返回：**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 操作成功失败，true表示成功，false表示失败 |

### 根据账户INT编号查询账户介质接口

1. 根据基础账户表的账户INT编号查询账户的识别介质；
2. 返回的是一个识别介质的JSON；
3. 接口为GET HTTP 格式；

**接口说明：**

**接口地址：/basic\_account/get\_media/{basic\_acc\_no}**

**HTTP动词：GET**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| basci\_acc\_no | int | N | 账户INT编号 |
| 示例：  **/account/get\_media/12** | | | |

**接口返回：**

**Data部分是个json体**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| 示例：  {  “face”:’Y’, //面部识别  “finger\_0”:’Y’, //左手小拇指  “finger\_1”:’Y’, //左手无名指  “finger\_2”:’Y’, //左手中指  “finger\_3”:’N’, //左手食指  “finger\_4”:’Y’, //左手大拇指  “finger\_5”:’N’, //右手大拇指  “finger\_6”:’Y’, //右手食指  “finger\_7”:’Y’, //右手中指  “finger\_8”:’Y’, //右手无名指  “finger\_9”:’Y’, //右手小拇指  } | | | |

### 修改应用账户密码接口

1. 以应用id+账户编号定位到一个账户应用信息；
2. 需要先判断该账户是否存在；
3. 应用自身中应该有密码和确认密码两个列，应用完成两次密码的一致性校验之后，DES加密，将原始密码和新密码传递给账户微服务的修改密码接口；
4. 账户微服务DES解密之后，加上特有的MD5密钥做MD5加密之后存储；
5. 对客户端传递而来的数据做合法性验证，包含是否为空，两次录入的密码是否一致，密码长度是否为6-18位；
6. 接口为POST JSON 方式修改账户识别介质接口；
7. 更新人为平台传过来的；
8. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口；

**接口说明：**

**接口地址：/account/modify\_pass**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| basci\_acc\_no | integer | Y/N | 账户编号 账户编号与应用账户至少一个不为空 |
| app\_account | string | Y/N | 应用账户 账户编号与应用账户至少一个不为空 |
| old\_password | string | N | 原始密码 |
| new\_password | string | N | 新密码 |
| 示例：  {  “basci\_acc\_no”:12,  “app\_account”:”xxx01”,  “old\_password”:”DFGSGGEGEGERGR1234234234SDFS”, //密文  “new\_password”:”DFGSGGEGEGERGR1234234234SDFS” //密文  } | | | |

**接口返回：**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 修改成功失败，true表示成功，false表示失败 |

### 删除应用账户接口

1. 以应用id+账户编号定位到一个账户应用信息；
2. 所谓删除是软删除，更新删除为已删除即可；
3. 需要验证数据的合法性；
4. 需要验证该应用账户是否存在；
5. 执行应用账户的删除操作并不会删除基础账户数据；
6. 接口为DELETE JSON 方式
7. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口

**接口说明：**

**接口地址：/account/delete**

**HTTP动词：DELETE**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| basci\_acc\_no | integer | Y/N | 账户编号 账户编号与应用账户至少一个不为空 |
| app\_account | string | Y/N | 应用账户 账户编号与应用账户至少一个不为空 |
| 示例：  {  “basci\_acc\_no”:12,  “app\_account”:”xxx01”  } | | | |

**接口返回：**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 删除成功失败，true表示成功，false表示失败 |

### 查询某个应用账户接口

1. 首先需要明确，在微服务的服务网关中，通过应用id与令牌进行鉴权成功的请求才会被分发到账户微服务中；
2. 该接口为POST JSON 方式；
3. JOSN体中有包含有一个查询条件的数组；
4. 该接口最终返回的某个应用账户的数据，最多只有一个应用账户的数据会被返回；
5. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的查询条件

**接口说明：**

**接口地址：/account/searchone**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| conditions | 二维数组 | Y | 查询条件，一个二维数组,可选的参数实际开发会提供，不在范围的查询条件会被剔除 |
| 示例：  {  “conditions”:[  [“app\_account”,“com”,true], //代表的意思为app\_account字段完全匹配com的  [“mobilephone”,“312”,false], //代表的意思为mobilephone字段包含001的  [“is\_deleted”,1,true]//代表的意思为is\_deleted等于1的,对于数值类型的字段只能是true  ]  } | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| basic\_acc\_no | int | N | 基础账号编号 |
| app\_account | string | N | 应用账户 |
| id\_card | string | Y | 身份证号码 |
| name | string | Y | 真实姓名 |
| sex | string | N | 性别 男 女 |
| mobilephone | string | Y | 手机号码1 |
| mobilephone2 | string | Y | 手机号码2 |
| telephone | string | Y | 固话 |
| email | string | Y | 邮箱1 |
| email2 | string | Y | 邮箱2 |
| app\_account | string | N | 应用账户 |
| app\_account\_status | int | N | 应用账户状态 |
| label | JSON | Y | 标签 JSON数据体 |
| role\_nos | JSON数组 | N | 角色,信息 JONS 数组 |
| column\_1 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_2 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_3 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_4 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_5 | string | Y | 备用标注,应用自定义存储，自行解析 |
| 示例：  {  “basic\_acc\_no”:13,  “app\_account”:”0000X1”,  “id\_card”:”xxxxxxxxxxxxxxxxxxxxxxxxx”,  “name”:”zhangsan”,  “sex”:’男’,  “mobilephone”:”132XXXXXXXX”,  “mobilephone2”:”134XXXXXXXX”,  “telephone”:”0535-802XXXX”,  “email”:”232@163.com”,  “email2”:”2X2@163.com”,  “app\_account”:“zzhangsan, //  “app\_account\_status”:”正常” //  “label”:[“class”,”tearcher”],  “role\_nos”:[  {  “role\_no”:”000x”, //角色编号  “role\_name”:”admin”, //角色名  “access”:”;0001;0002;”//权限编号，以 ; 分割  },  {  “role\_no”:”000y”, //角色编号  “role\_name”:”com”, , //角色名  “access”:”;0003;0006;” //权限编号，以 ; 分割  }  ]  } | | | |

### 查询应用账户列表接口

1. 该接口为POST JSON 方式；
2. JOSN体中有包含有一个查询条件的数组；
3. 该接口最终一个应用账户的数组，并且带有分页信息；
4. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的查询条件

**接口说明：**

**接口地址：/account/search**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| conditions | 二维数组 | Y | 查询条件，一个二维数组,可选的参数实际开发会提供，不在范围的查询条件会被剔除 |
| pageindex | int | Y | 页码 默认1 |
| pagesize | int | Y | 每页条数 默认20 |
| count | boolean | Y | 是否查询总条数 默认false |
| orderby | string | Y | 排序字段，具体的可排序字段待定，会在接口文档中说明可选的排序字段有哪些，多个排序字段以“，”分割 |
| groupby | string | Y | 分组字段，具体的可分组字段待定，会在接口文档中说明可选的分组字段有哪些，多个排序字段以“，”分割 |
| 示例：  {  “conditions”:[  [“app\_account”,“com”,true], //代表的意思为app\_account字段完全匹配com的  [“mobilephone”,“312”,false], //代表的意思为mobilephone字段包含001的  [“is\_deleted”,1,true]//代表的意思为is\_deleted等于1的,对于数值类型的字段只能是true  ],  “pageindex”:1,  “pagesize”:20,  “count”:true,  “orderby”:NULL,  “groupby”:NULL  } | | | |

**接口返回**

**Data部分的字段说明**

**Records数组单条记录的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| basic\_acc\_no | int | N | 基础账号编号 |
| app\_account | string | N | 应用账户 |
| id\_card | string | Y | 身份证号码 |
| name | string | Y | 真实姓名 |
| sex | string | N | 性别 男 女 |
| mobilephone | string | Y | 手机号码1 |
| mobilephone2 | string | Y | 手机号码2 |
| telephone | string | Y | 固话 |
| email | string | Y | 邮箱1 |
| email2 | string | Y | 邮箱2 |
| app\_account | string | N | 应用账户 |
| app\_account\_status | int | N | 应用账户状态 |
| label | JSON | Y | 标签 JSON数据体 |
| role\_nos | JSON数组 | N | 角色,信息 JONS 数组 |
| column\_1 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_2 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_3 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_4 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_5 | string | Y | 备用标注,应用自定义存储，自行解析 |
| 单条record结构示例：  {  “basic\_acc\_no”:13,  “app\_account”:”0000X1”,  “id\_card”:”xxxxxxxxxxxxxxxxxxxxxxxxx”,  “name”:”zhangsan”,  “sex”:’男’,  “mobilephone”:”132XXXXXXXX”,  “mobilephone2”:”134XXXXXXXX”,  “telephone”:”0535-802XXXX”,  “email”:”232@163.com”,  “email2”:”2X2@163.com”,  “app\_account”:“zzhangsan, //  “app\_account\_status”:”正常” //  “label”:[“class”,”tearcher”],  “role\_nos”:[  {  “role\_no”:”000x”, //角色编号  “role\_name”:”admin”, //角色名  “access”:”;0001;0002;”//权限编号，以 ; 分割  },  {  “role\_no”:”000y”, //角色编号  “role\_name”:”com”, , //角色名  “access”:”;0003;0006;” //权限编号，以 ; 分割  }  ]  } | | | |

**Data的结构为：**

|  |
| --- |
| 示例：  {  “total\_record”:1, //总条数  “total\_page”:20, //总页数  “page\_index”:true, //当前页码  “page\_size”:20, //每页容量  “recordss”:[  {…}, //每条数据  {…},//每条数据  {…},//每条数据  ]  } |

### 平台账户登陆接口

1. 为平台提供账户账户的接口；
2. 根据账户编号 /应用账户+密码+应用id查询应用账户的信息；
3. 如果没有查询到应用账户信息则代表账户不存在或者密码不正确，登陆失败，返回code-1，msg登陆失败即可；
4. 如果查询到了一条应用账户信息，则把应用账户信息返回给应用，并且返回code为0，msg为成功，data为应用账户信息；
5. 理论上不可能查询到多余一条数据，如果该情况发生则代表系统已经被攻击；
6. 接口为POST JSON 方式；
7. 需要做一个计数，五分钟内登陆失败三次以上锁定账户，返回账户异常，防撞库；
8. 应用账户状态为异常的不允许登陆成功，返回账户异常；
9. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口

**接口说明：**

**接口地址：/account/login**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 | |
| basci\_acc\_no | integer | Y/N | | 账户编号 账户编号与应用账户至少一个不为空 |
| app\_account | string | Y/N | | 应用账户 账户编号与应用账户至少一个不为空 |
| password | string | N | | 密码 经过MD5 加密的 算法要处理 |
| 示例：  {  “basci\_acc\_no”:12, //账户编号  “app\_account”:“zzhangsan, //应用账户 账户编号与应用账户至少一个不为空“password”:”AFAAFAF23432TEGAFA” //密码的MD5密文  } | | | | |

**接口返回**

**登陆失败** data=null

**登陆成功Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| basic\_acc\_no | int | N | 基础账号编号 |
| id\_card | string | Y | 身份证号码 |
| name | string | Y | 真实姓名 |
| sex | string | N | 性别 男 女 |
| mobilephone | string | Y | 手机号码1 |
| mobilephone2 | string | Y | 手机号码2 |
| telephone | string | Y | 固话 |
| email | string | Y | 邮箱1 |
| email2 | string | Y | 邮箱2 |
| app\_account | string | N | 应用账户 |
| app\_account\_status | int | N | 应用账户状态 |
| label | JSON | Y | 标签 JSON数据体 |
| role\_nos | JSON数组 | N | 角色,信息 JONS 数组 |
| column\_1 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_2 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_3 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_4 | string | Y | 备用标注,应用自定义存储，自行解析 |
| column\_5 | string | Y | 备用标注,应用自定义存储，自行解析 |
| 示例：  {  basic\_acc\_no”:13,  “id\_card”:”xxxxxxxxxxxxxxxxxxxxxxxxx”,  “name”:”zhangsan”,  “sex”:’男’,  “mobilephone”:”132XXXXXXXX”,  “mobilephone2”:”134XXXXXXXX”,  “telephone”:”0535-802XXXX”,  “email”:”232@163.com”,  “email2”:”2X2@163.com”,  “app\_account”:“zzhangsan, //  “app\_account\_status”:”正常” //  “label”:[“class”,”tearcher”],  “role\_nos”:[  {  “role\_no”:”000x”, //角色编号  “role\_name”:”admin”, //角色名  “access”:”;0001;0002;”//权限编号，以 ; 分割  },  {  “role\_no”:”000y”, //角色编号  “role\_name”:”com”, , //角色名  “access”:”;0003;0006;” //权限编号，以 ; 分割  }  ]  } | | | |

### 新增应用角色接口

1. 必填字段有角色名称，应用id；
2. 接口要对添加角色的请求数据做合法性校验；
3. 角色编号由账户微服务内部生成；
4. 接口为PUT JSON 方式；
5. 新增人为平台传过来的；
6. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口

**接口说明：**

**接口地址：/role/add**

**HTTP动词：PUT**

**参数定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 是否为空 | 说明 | |
| role\_name | String | | N | | 角色名称 |
| access | String | | N | | 权限码 以”;权限码1;权限码2;”形式存储 |
| inserted\_by | String | | Y | | 添加人 默认值为system |
| 示例：  {  “role\_name”:”admin”, //角色名称  “access”:”;xxx01;xxx02;;xxx02;” //权限码  “inserted\_by”:”admin” //添加人 默认值为system  } | | | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 新增成功失败，true表示成功，false表示失败 |

### 修改应用角色接口

1. 以应用应用id+角色编号定位到一个应用角色信息；
2. 角色编号不可修改；
3. 验证接口数据合法性；
4. 维护平台具体角色的权限编号；
5. 每次修改都需要更新最后更新人与时间
6. 更新人为平台传过来的
7. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口

**接口说明：**

**接口地址：/role/modify**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 是否为空 | 说明 | |
| role\_no | String | | N | | 角色编号 |
| role\_name | String | | N | | 角色名称 |
| access | String | | N | | 权限码 以”;权限码1;权限码2;”形式存储 |
| updated\_by | String | | Y | | 更新人 默认值为system |
| 示例：  {  “role\_no”:”xxx01”, //角色编号  “role\_name”:”admin”, //角色名称  “access”:”;xxx01;xxx02;;xxx02;” //权限码  “updated\_by”:”admin”  } | | | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数返回 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 修改成功失败，true表示成功，false表示失败 |

### 删除应用角色接口

1. 以应用应用id+角色编号定位到一个应用角色信息；
2. 所谓删除是软删除，更新删除为已删除即可；
3. 需要验证数据的合法性；
4. 需要验证该应用账户是否存在；
5. 接口为DELETE JSON 方式；
6. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的条件传递给接口

**接口说明：**

**接口地址：/role/delete**

**HTTP动词：DELETE**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| role\_no | string | N | 角色编号 |
| 示例：  {  “role\_\_no”:”xxx01”  } | | | |

**接口返回：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 参数类型 | 是否为空 | 说明 |
| flag | bool | N | 删除成功失败，true表示成功，false表示失败 |

### 查询某个应用角色接口

1. 首先需要明确，在微服务的服务网关中，通过应用id与令牌进行鉴权成功的请求才会被分发到账户微服务中；
2. 该接口为POST JSON 方式；
3. JOSN体中有包含有一个查询条件的数组；
4. 该接口最终返回的某个应用角色的数据，最多只有一个应用角色的数据会被返回；
5. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的查询条件

**接口说明：**

**接口地址：/role/searchone**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| conditions | 二维数组 | Y | 查询条件，一个二维数组 |
| 示例：  {  “conditions”:[  [“role\_name”,“com”,true], //代表的意思为role\_name字段完全匹配com的  [“access”,“001”,false], //代表的意思为access字段包含001的  [“is\_deleted”,1,true], //代表的意思为is\_deleted等于1的,对于数值类型的字段只能是true  ]  } | | | |

**接口返回**

**Data部分的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 参数类型 | 是否为空 | 说明 |
| id | int | N | int |
| app\_no | String | N | 应用编号 |
| role\_no | String | N | 角色编号 |
| role\_name | String | N | 角色名称 |
| access | String | N | 权限码 |
| is\_deleted | bool | N | 删除标志 |
| inserted\_dt | datetime | N | 插入时间 |
| inserted\_by | String | N | 插入人 |
| updated\_dt | datetime | N | 更新时间 |
| updated\_by | String | N | 更新人 |

### 查询应用角色列表接口

1. 该接口为POST JSON 方式；
2. JOSN体中有包含有一个查询条件的数组；
3. 该接口最终一个应用角色的数组，并且带有分页信息；
4. 会根据请求带的令牌在服务网关中获取到对应的appno，作为附加的查询条件

**接口说明：**

**接口地址：/role/search**

**HTTP动词：POST**

**参数定义：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 是否为空 | 说明 |
| conditions | 二维数组 | Y | 查询条件，一个二维数组 |
| pageindex | int | Y | 页码 默认1 |
| pagesize | int | Y | 每页条数 默认20 |
| count | Boolean | Y | 是否查询总条数 默认false |
| orderby | String | Y | 排序字段，具体的可排序字段待定，会在接口文档中说明可选的排序字段有哪些，多个排序字段以“，”分割 |
| groupby | String | Y | 分组字段，具体的可分组字段待定，会在接口文档中说明可选的分组字段有哪些，多个排序字段以“，”分割 |
| 示例：  {  “conditions”:[  [“role\_name”,“com”,true], //代表的意思为role\_name字段完全匹配com的  [“access”,“001”,false], //代表的意思为access字段包含001的  [“is\_deleted”,1,true],//代表的意思为is\_deleted等于1的,对于数值类型的字段只能是true  ],  “pageindex”:1,  “pagesize”:20,  “count”:true,  “orderby”:NULL,  “groupby”:NULL  } | | | |

**接口返回**

**Records数组单条记录的字段说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 参数类型 | 是否为空 | 说明 |
| id | int | N | int |
| app\_no | String | N | 应用编号 |
| role\_no | String | N | 角色编号 |
| role\_name | String | N | 角色名称 |
| access | String | N | 权限码 |
| is\_deleted | bool | N | 删除标志 |
| inserted\_dt | datetime | N | 插入时间 |
| inserted\_by | String | N | 插入人 |
| updated\_dt | datetime | N | 更新时间 |
| updated\_by | String | N | 更新人 |

**Data的结构为：**

|  |
| --- |
| 示例：  {  “total\_record”:1, //总条数  “total\_page”:20, //总页数  “page\_index”:true, //当前页码  “page\_size”:20, //每页容量  “recordss”:[  {…}, //每条数据  {…},//每条数据  {…},//每条数据  ]  } |

# 数据库设计

详见数据库设计文档：《账户微服务数据库表结构v1.0.docx》

## 1.basic\_account\_info基础账户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 注释 | 字段类型 | 主外键/约束属性 |
| id | 自增序号 | serial | 主健 |
| basic\_acc\_no | 账户编号 | INT2 | 主健 |
| id\_card | 身份证号 | VARCHAR(50) | 是否允许为空？ |
| name | 姓名 | VARCHAR(50) | Not null |
| sex | 性别 | INT2 | Not null 0代表男人,1代表女 |
| mobilephone | 手机号 | VARCHAR(50) | 是否允许为空？ |
| mobilephone2 | 手机号2 | VARCHAR(50) |  |
| telephone | 座机号 | VARCHAR(50) |  |
| email | 邮箱 | VARCHAR(100) |  |
| email2 | 邮箱2 | VARCHAR(100) |  |
| iden\_media |  | JSONB | Not null |
| is\_deleted | 删除标志 | INT2 | 0(0,1).删除标志,0代表未删除 1代表已删除，默认0.只有0和1 |
| inserted\_dt | 插入时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| inserted\_by | 插入人 | VARCHAR(50) | ‘’ |
| updated\_dt | 更新时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| updated\_by | 更新人 | VARCHAR(50) | ‘’ |

## 2.app\_account\_info应用账户列表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 注释 | 字段类型 | 主外键/约束属性 |
| id | 自增序号 | serial | 主健 |
| app\_no | 应用编号 | VARCHAR(50) | 主健与acc\_list的acc\_no对应，app\_no和app\_account 组合成为唯一约束 |
| basic\_acc\_no | 基础账号唯一值 | INT2 | Not null与basic\_account\_info的basic\_acc\_no对应 |
| app\_account | 应用账户 | VARCHAR(50) | Not null例淘宝的用户,app\_no和app\_account 组合成为唯一约束 |
| status | 应用状态 | INT2 | Not null,0为正常，1 为异常,默认为0， 只有0和1 |
| label | 标签 | JSONB | Not null |
| password | 密码 | VARCHAR(50) | Not null |
| role\_nos | 角色码 | VARCHAR(1000) | Not null |
| is\_deleted | 删除标志 | INT2 | 0(0,1).删除标志,0代表未删除 1代表已删除，默认0.只有0和1 |
| inserted\_dt | 插入时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| inserted\_by | 插入人 | VARCHAR(50) | ‘’ |
| updated\_dt | 更新时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| updated\_by | 更新人 | VARCHAR(100) | ‘’ |
| column\_1 | 备用1 | VARCHAR(100) |  |
| column\_2 | 备用2 | VARCHAR(100) |  |
| column\_3 | 备用3 | VARCHAR(100) |  |
| column\_4 | 备用4 | VARCHAR(100) |  |
| column\_5 | 备用5 | VARCHAR(100) |  |

## 3.operation\_log 操作表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 注释 | 字段类型 | 主外键/约束属性 |
| id | 自增序号 | serial | 主健 |
| app\_no | 应用编号 | VARCHAR(50) | Not null 与app\_list的app\_no对应 |
| opt\_table | 角色编号 | VARCHAR(100) | Not null |
| opt\_type | 角色名称 | VARCHAR(20) | Not null |
| inserted\_dt | 插入时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| inserted\_by | 插入人 | VARCHAR(50) | ‘’ |
| updated\_dt | 更新时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| updated\_by | 更新人 | VARCHAR(50) | ‘’ |

## 4.role\_list 角色列表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 注释 | 字段类型 | 主外键/约束属性 |
| id | 自增序号 | serial | 主健 |
| app\_no | 应用编号 | VARCHAR(50) | Not null与app\_list的app\_no对应 |
| role\_no | 角色编号 | VARCHAR(50) | 主健 |
| role\_name | 角色名称 | VARCHAR(50) | Not null |
| access | 权限码 | VARCHAR(200) | Not null以;开头以;结尾例;001;002; |
| is\_deleted | 删除标志 | INT2 | 0(0,1).删除标志,0代表未删除 1代表已删除，默认0.只有0和1 |
| inserted\_dt | 插入时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| inserted\_by | 插入人 | VARCHAR(50) | ‘’ |
| updated\_dt | 更新时间 | TIMESTAMP | CURRENT\_TIMESTAMP |
| updated\_by | 更新人 | VARCHAR(50) | ‘’ |

## 数据库备注

### 数值数据类型范围

|  |  |  |  |
| --- | --- | --- | --- |
| 名字 | 存储空间 | 描述 | 范围 |
| smallint | 2 字节 | 小范围整数 | -32768 到 +32767 |
| integer | 4 字节 | 常用的整数 | -2147483648 到 +2147483647 |
| bigint | 8 字节 | 大范围的整数 | -9223372036854775808 到 9223372036854775807 |
| decimal | 变长 | 用户声明精度，精确 | 无限制 |
| numeric | 变长 | 用户声明精度，精确 | 无限制 |
| real | 4 字节 | 变精度，不精确 | 6 位十进制数字精度 |
| double | 8 字节 | 变精度，不精确 | 15 位十进制数字精度 |
| serial | 4 字节 | 自增整数 | 1 到 +2147483647 |
| bigserial | 8 字节 | 大范围的自增整数 | 1 到 9223372036854775807 |