极速开票代码绑定分析设计

1. 功能

（1）提供给后台分配管理端调用

1. 分页查询开票代码分配信息（刘阳等）

查询条件可以为空，为空表示所有（先实现查询所有）

第一期只要实现分页查询即可，后台登录人员都可以看到所有分配信息。

1. 按企业名称模糊查询企业及员工信息（刘阳等）

先调用数据分析接口获取企业税号和名称

后调用用户中心的接口获取企业员工信息

1. 分配代码（刘阳等）

从预开票代码缓存中获取一定数量的开票代码

创建开票代码分配主信息和明细信息

（2）提供给app调用

1. 按税号精确查询企业名片信息（刘定）

~~1.1 调用数据分析提供的接口获取税号和名称（可以用来判定税号是否存在）~~

1.2 根据税号调用企业名片数据中心获取企业名片信息（胡伟中）

1.2.1 按税号从缓存中获取开票代码

1.2.2 按开票代码从缓存中获取开票信息

注：

这次输入的税号在数据层是存在的，不考虑新增的企业。因此直接调用数据层的接口获取企业名片即可。

1. 绑定预开票代码（刘定超）

2.1 更新缓存中该企业对应的6位代码（胡伟中）

2.2 更新缓存中开票代码对应的开票信息（胡伟中）

2.3 更新数据中心的开票信息（胡伟中）

2.4 将企业的原始代码放在需要保留且可恢复的代码池里（胡伟中）

2.5 更新预分配信息（刘定超）

注：

（1）2.1、2.2、2.3、2.4由数据层提供绑定企业名片服务接口

（2）需要考虑并发同时绑定同一个开票代码（根据实际使用场景，可以先暂时不考虑）

1. 校验预开票代码是否可以绑定（刘定超）

~~3.1 用户级别限定，是不是属于二级（包含）以上用户，如果不是，则不可绑定；否则继续校验 (刘定超)~~

3.2从分配信息表中查看一下是否存在预分配代码,如果不存在，则不可绑定；否则继续。（刘定超）

3.3 从分配信息表中查看一下是否已经绑定过了,如果已绑定，则不可绑定；否则继续。（刘定超）

3.4 调用数据层查看是否已经认证过了,如果认证过了，则不可绑定；否则继续。（胡伟中）

3.5. 查看该开票代码是否是保留的代码（需要从保留可恢复和保留不可恢复的代码池中查看），如果是保留的代码，则不可绑定。（胡伟中）

注：

对于3.4和3.5两点数据层提供开票代码状态服务接口

对于3.1这次不用做

1. 数据层接口
2. 根据税号获取企业名片

|  |  |
| --- | --- |
| 功能描述 | 根据税号获取企业名片信息  适用于绑定页面中根据税号精确查询 |
| URL地址 | /v1/helper/taxid2card/{:id} |
| 请求方式 | get |
| 输入参数 | id为企业税号 |
| 输出参数 | **{**  **"code": 200,**  **"data": {**  **"code": "六位代码",**  **"taxid": "税号",**  **"name": "企业名称",**  **"address": "注册地址",**  **"telephone": "注册电话",**  **"bank": "开户银行",**  **"account": "开户账号",**  **"type": "企业类型",**  **"cert": "认证",**  **"source": "企业信息来源",**  **"audit": "审核字段"**  **}**  **}** |
| 备注 |  |

1. 获取开票代码状态（待确定）

|  |  |
| --- | --- |
| 功能描述 | 获取开票代码状态 |
| URL地址 | /v1/cards/helper/status/{:id} |
| 请求方式 | get |
| 输入参数 | id为老6位代码  {  "newCode": "新6位代码"  } |
| 输出参数 | **{**  **"code": "200",（）**  **"data": {**  **"oldCodeStatus": "老代码状态" , "newCodeStatus": "新代码状态"**  **}**  **}** |
| 备注 | **0:未认证 1：已认证 2:保留 3:不存在**  **对于老代码**  **状态：0:未认证 1：已认证**  **对于新代码：**  **状态：2:保留 3:不存在** |

1. 绑定开票代码

|  |  |
| --- | --- |
| 功能描述 | 绑定企业名片（涉及关系表、审核表） |
| URL地址 | /v1/cards/bind/{:id} id为老6位代码 |
| 请求方式 | post |
| 输入参数 | {  "newCode": "新6位代码",  "tax\_id": "企业税号"  } |
| 输出参数 | **{**  **"code": "200",**  **"data": {**  **"code": "六位代码"**  **}**  **}** |
| 备注 | **如果成功，返回新6位代码；如果失败，则返回老6位代码。** |

1. 第三方接口
2. 按企业名称模糊查询

|  |  |
| --- | --- |
| 功能描述 | 按企业名称模糊检索企业信息（包含税号和名称）  适用于后台分配页面 |
| URL地址 | http://ip:port/dataAnalysis-web/company/phrase?text=%s&size=%d  测试环境：ip:192.168.210.200 port:8081 |
| 请求方式 | get |
| 输入参数 | text:企业名称关键字  size:条数 （默认为5条） |
| 输出参数 | **[{**  **"code": "代码",**  **"name": "企业名称",**  **"tax\_id": "企业税号",**  **}]** |
| 备注 | 示例：   |  | | --- | | public List<KpSearchEntity> keySearch(String url) throws IOException {  // Execute request  HttpUriRequest request = new HttpGet(url);    request.addHeader("Content-type", "application/json");  request.addHeader("Connection", "Keep-Alive");  CloseableHttpResponse response= HttpUtils.getClient().execute(request);  try {  if(response.getStatusLine().getStatusCode()!=200){  return null;  }else{  String result=EntityUtils.toString(response.getEntity());  JSONArray jsonObject=JSON.parseArray(result);  if(jsonObject.size()==0){  return null;  }else{  List<KpSearchEntity> kpSearchEntities=new ArrayList<KpSearchEntity>();  KpSearchEntity kpSearchEntity=null;  Iterator it = jsonObject.iterator();  while(it.hasNext()){  kpSearchEntity=new KpSearchEntity();  JSONObject sObj = (JSONObject)it.next();  String code = sObj.getString("code");  String ckpname = sObj.getString("name");  String taxId=sObj.getString("tax\_id");  kpSearchEntity.setCode(code);  kpSearchEntity.setKpName(ckpname);  kpSearchEntity.setKpTaxId(taxId);  kpSearchEntities.add(kpSearchEntity);  }  return kpSearchEntities;  }  }  }catch (Exception e) {  // 打印日志  }finally{  response.close();  }  return null;  } | |

1. 获取企业员工信息

需要调用二个接口，一个接口是获取企业基本信息，从中拿到userId，然后用这个userId调用另外一个获取员工信息的接口。

2.1 获取企业基本信息

|  |  |
| --- | --- |
| 功能描述 | 获取企业基本信息  适用于后台分配员工的选择 |
| URL地址 | http://ip:port/usercenter/syncuser/allow/getuser.do |
| 请求方式 | post |
| 输入参数 | url参数：username（企业税号） |
| 输出参数 | **[{**  **"CUid": "用户Id",**  **"CRegistertype": "用户类型", 0:个人用户；1：企业用户；2:代理商**  **"CUsername": "用户名",**  **"CTexnum": "税号",**  **"CCompanyname": "企业名称"**    **}]** |
| 备注 |  |

2.2 获取企业员工信息

|  |  |
| --- | --- |
| 功能描述 | 获取企业员工信息  适用于后台分配员工的选择 |
| 调用方式 | spring http 远程调用 |
| 配置接口 | 在pom.xml中增加以下依赖jar包  <dependency>   <groupId>nuonuo.usercenter</groupId>   <artifactId>usercenter.api</artifactId>   <version>3.0.1-SNAPSHOT</version> </dependency>  在spring配置文件增加以下配置  <bean name="companyApi"  class="org.springframework.remoting.httpinvoker.HttpInvokerProxyFactoryBean">  <property name="serviceUrl" value="${usercenter.url}/remote/allow/userCenterOrgDataRmtService"></property>  <property name="serviceInterface" value="com.aisino.projects.usercenter.redis.support.service.IUserCenterRedisService"></property>  </bean> |
| 接口方法 | getAllEmployeeList |
| 输入参数 | companyId公司ID |
| 输出参数 | List<Map<String,Object>>对象，Map中的key值为UserCenterEmployee对象属性+头像key为：photo |
| 备注 | 不用取头像，在目前使用场景下员工数不会很多 |

1. 其他接口
2. 分页查询分配信息

直接通过访问数据库进行查询

1. 添加分配信息

直接通过访问数据库进行添加

1. 更新分配信息

直接访问数据库进行更新

1. 获取预分配代码

|  |  |
| --- | --- |
| 功能描述 | 获取一批预分配代码用于分配和绑定  适用于后台分配代码 |
| 调用方式 | 1. 配置redis地址和端口 2. 通过jedis直接获取指定数量的分配代码   Key为code\_prepare\_assign\_pool,value为set列表 |
| 备注 | 通过pop方式把数据取走 |

1. 缓存

根据业务预估分配量最大80万，预先在缓存中分配100万的开票代码池。

开票代码预分配缓存key为code\_prepare\_assign\_pool

1. 数据库

在综合服务平台数据库中创建以下分配表：



1. 开票预分配主信息表
   1. 创建脚本

|  |
| --- |
| drop table if exists jskp\_kpcode\_assign\_main;  create table jskp\_kpcode\_assign\_main  (  ID int not null,  assign\_company\_name varchar(100) comment '分配企业名称',  assign\_company\_taxid varchar(20) default NULL comment '分配企业税号',  assign\_employee\_id varchar(30) default NULL comment '分配员工ID',  assign\_employee\_name varchar(10) default NULL comment '分配员工名称',  assign\_employee\_mobile varchar(10) default NULL comment '分配员工手机',  assign\_amount int comment '分配数量',  create\_person char(10) default NULL comment '创建人',  create\_time datetime default NULL comment '创建时间',  modify\_person char(10) comment '修改人',  modify\_time datetime default NULL comment '修改时间',  primary key (ID)  ) |

* 1. 唯一索引

ID

1. 开票预分配明细信息表
   1. 创建脚本

|  |
| --- |
| drop table if exists jskp\_kpcode\_assign\_detail;  /\*==============================================================\*/  /\* Table: jskp\_kpcode\_assign\_detail \*/  /\*==============================================================\*/  create table jskp\_kpcode\_assign\_detail  (  detail\_id int not null comment '明细ID',  main\_id int comment '主ID',  pre\_assign\_code char(6) default NULL comment '预分配代码',  bind\_company\_name varchar(100) default '' comment '绑定企业名称',  bind\_company\_taxid varchar(20) default NULL comment '绑定企业税号',  bind\_person char(10) default NULL comment '绑定人',  bind\_time datetime default NULL comment '绑定时间',  origin\_code char(6) default NULL comment '原始代码',  status char(1) comment '状态',  primary key (detail\_id)  )  状态：  0：新建 1:已绑定 2:已恢复 |

* 1. 唯一索引

预分配代码

明细ID