基于 Django 开发的权限管理系统
--所有应用系统的奠基石

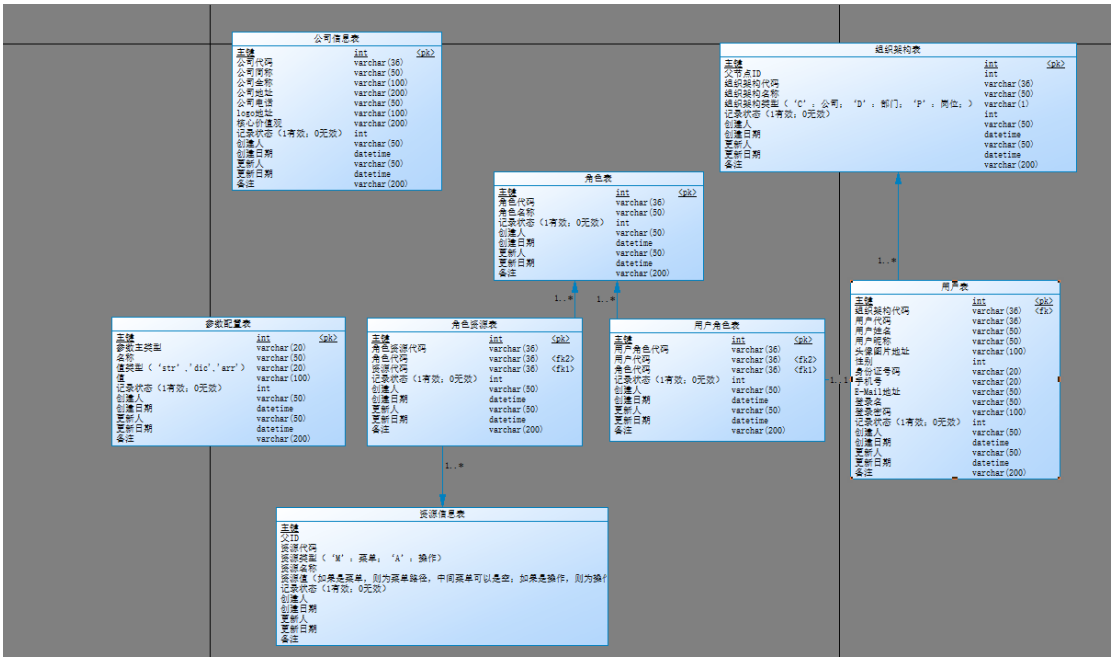
一、前言

由于几个项目中应用到了 Django 框架开发，所以一顿猛操作下来，对其算是有些了解。世间万物，没有十全十美，然 Django 也一样，有其好的一面，也有其不完美的一面，如界面布局、权限控制等，有时真满足不了自己的需求。某天，突然想到一个念头，能不能在 Django 便捷开发的基础上，来实现自定义界面，扩展其功能，如单表操作，直接通过数据表映射，复杂些的操作，通过自定义界面和功能实现，这样既能达到效果，又能提高效率；同时，对于权限控制，也可以通过自己编写中间件来实现控制。

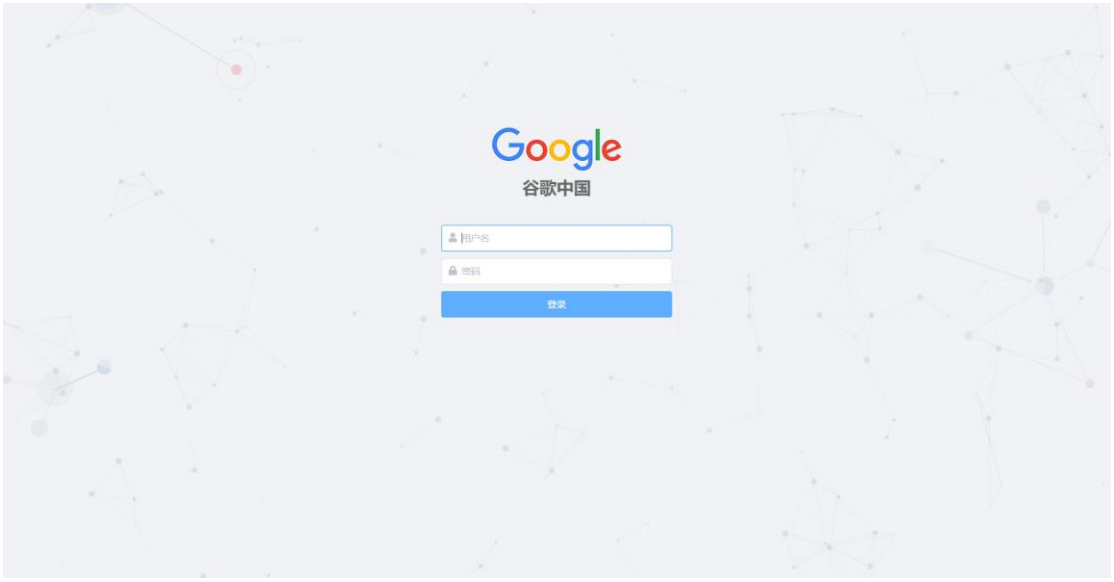
另外，任何一个应用系统，都需要一个系统管理模块，需要一台“脚手架”。基于此，开发出了这个适合大部分系统的基础框架，系统设置—权限管理模块（RABC）。

二、系统介绍

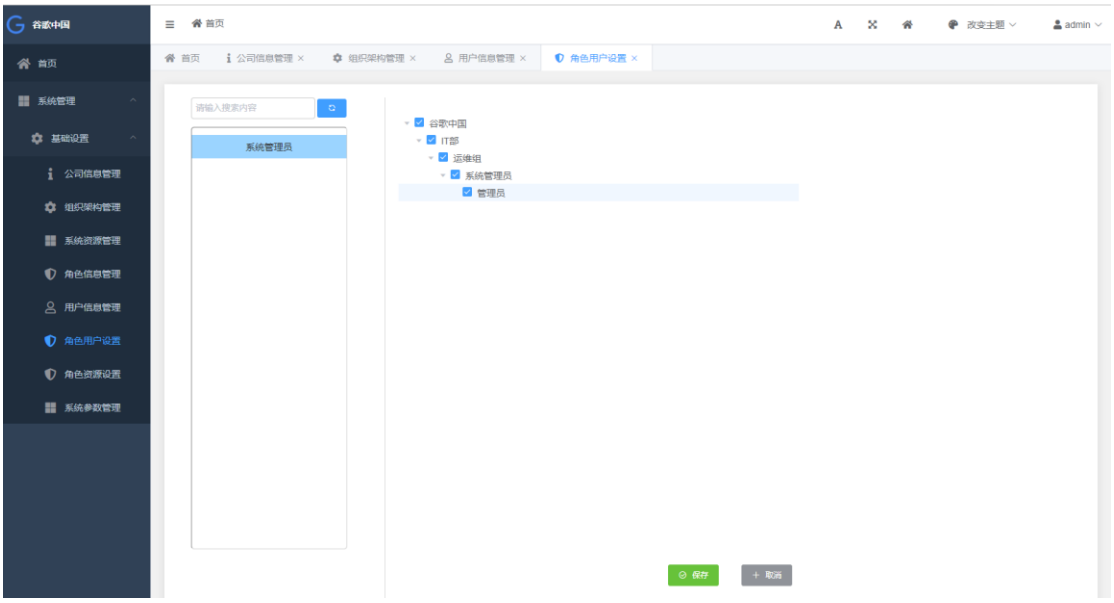
系统采用 Django 框架开发，前端 Vue，同时用到了开源的 SimpleUI，数据库 MySql 等。先上张效果图（Logo、名称等都是可以通过动态配置，此系统暂时取了个 Google 的名，应该不侵权吧）：



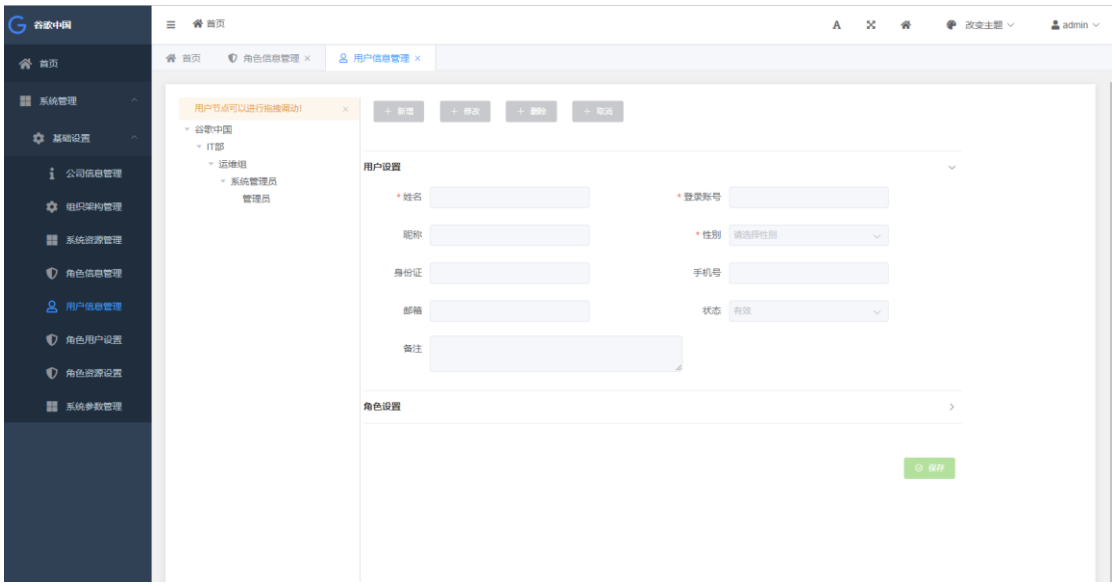
数据表设计图



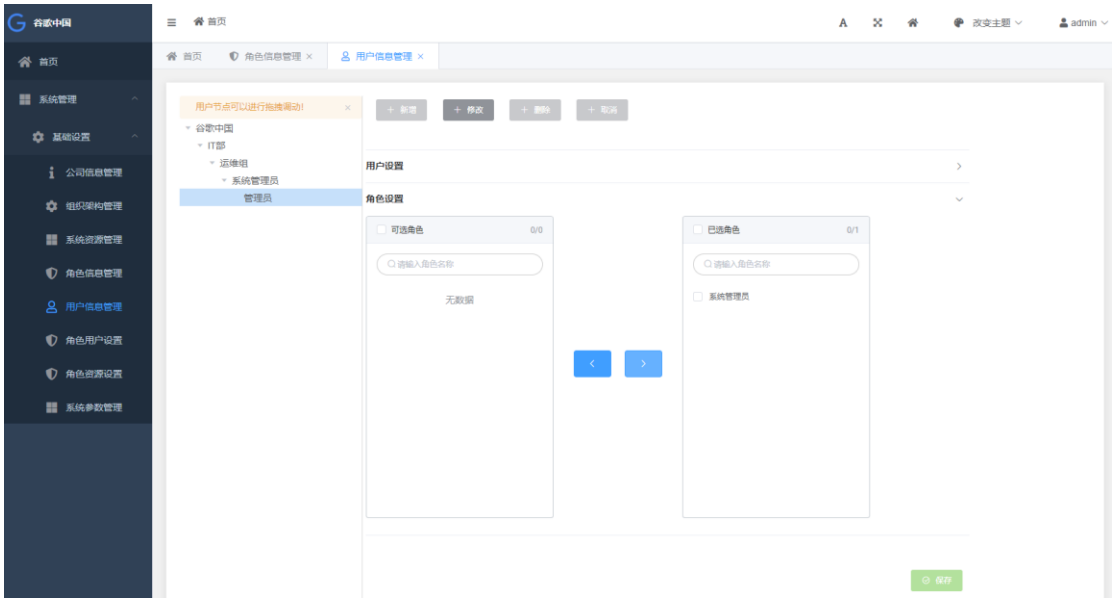
登录界面



主界面（角色用户设置）



主界面（用户信息设置-用户设置）



主界面（用户信息设置-角色设置）

三、运行

本项目开发 Python 用的是 3.7.2，其他版本也可以，最好新建一个虚拟环境，然后运行：

pip install –r requirements.txt

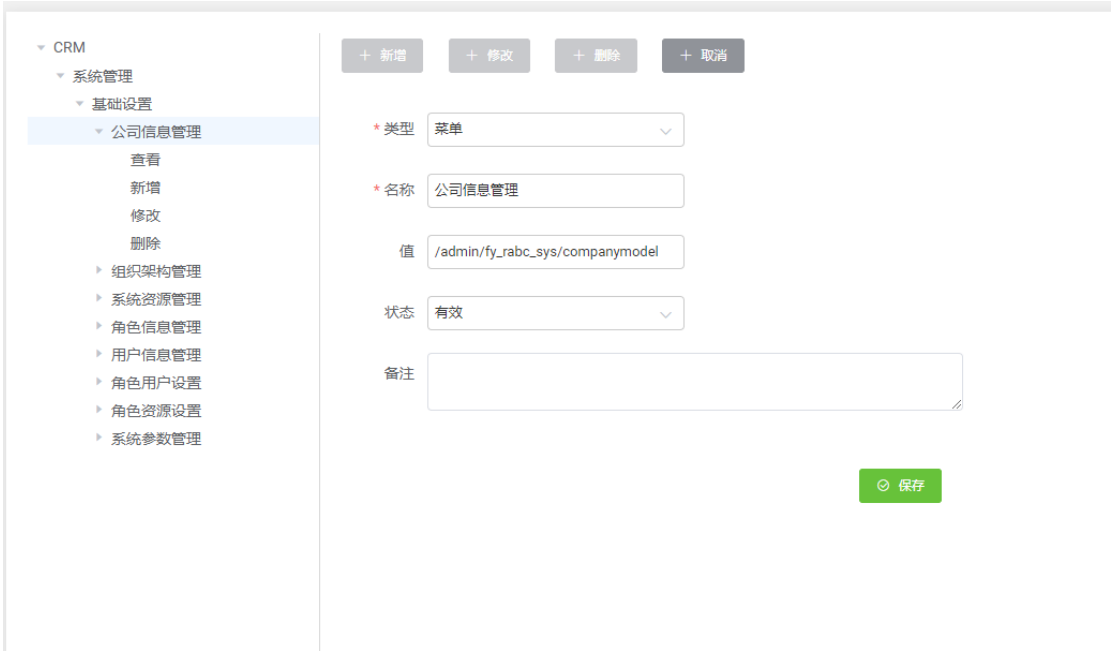
数据库直接执行脚本文件内容，createDB.sql。然后修改 settings.py 里数据库的配置：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'fy_rabc',
        'HOST': '192.168.0.111',
        'PORT': 3306,
        'USER': 'root',
        'PASSWORD': 'root',
        'OPTIONS': {
            'init_command': 'SET sql_mode="STRICT_TRANS_TABLES"',
            'charset': 'utf8mb4',
            # "init_command": "SET foreign_key_checks=0",
        }
    },
```

直接运行试试吧，其他细节就不敷说了。

四、开发应用

新增其他 APP，跟正常一样，在系统资源里可以配置它（名称，路径等）：



系统资源管理界面

然后再赋予不同的角色，在“角色资源管理”里，再把用户添加到对应的角色里，这样就可以控制对应的权限了。

如果想让某个实体（Model）能够直接应用到前端，需继承“CustAdminModel”类：

```python
@admin.register(ConfigModel)
class ConfigAdmin(CustAdminModel):
    list_display = ['pk', 'main_type', 'name', 'value_type', 'val']
    search_fields = ['main_type', 'name', 'value']
    list_filter = ('main_type', 'status')
    fields = ['main_type', 'name', 'value_type', 'value', 'status', 'creator', 'createdate', 'updator', 'updatedate',
              'remark']  # 详细页面的字段
    list_display_links = ('pk', 'name')  # 点击可跳转
    # list_editable = ["val"]  # 列表界面可编辑
    readonly_fields = ('creator', 'createdate', 'updator', 'updatedate')  # 只读字段

    def formfield_for_dbfield(self, db_field, **kwargs):
        formfield = super(ConfigAdmin, self).formfield_for_dbfield(db_field, **kwargs)
        if db_field.name in ['value', 'remark']:
            # formfield.widget = forms.Textarea(attrs=formfield.widget.attrs)
            formfield.widget = forms.Textarea(attrs={'cols': '80', 'rows': '5'})
        return formfield

    # 对相关列值进行处理
    def pk(self, obj):
        return str(obj.pk)
```

前端开发：

权限控制，直接可以用不同权限判断：“has_add_permission”、“has_delete_permission“、“has_change_permission”等来判断，如：

```
{% if has_add_permission %}
    <el-button size="small" type="info" icon="el-icon-plus" @click="addControl" :disabled="disAdd">
        {{ 'Add'|customtrans }}
    </el-button>
{% endif %}
```

路径说明：

所有的页面路由前缀，必须是不一样，同一个页面的其他功能路径，必须是相同开头，因为权限控制是根据子串查找来确定的，如：

其他：由于时间关系，没耐心写东东，就随便搞了下，有其他疑问，可以来个邮件：37412449@qq.com