第二次作业-基于配置平台进行游戏主机管理

一、 实验目的

（1）巩固 SaaS 应用的软件开发&设计能力

（2）了解蓝鲸 CMDB 配置平台的功能、数据结构与使用方法

（3）掌握蓝鲸网关/ESB 组件 API 的调用方法与鉴权模式

（4）能够通过蓝鲸 API 联通 CMDB 平台获取业务主机与架构信息

（5）提升 SaaS 开发技能，能够进行前后端联调并设计接口

（6）提升 SaaS 开发技能，进一步熟悉开发框架与后台建模

二、 实验环境

(1) 硬件环境需求： PC 或笔记本， 支持外网访问

(2) 软件环境需求

系统： Windows， MacOS， Linux

安装 Python 3.6.12

安装 MySQL 8.3

安装 Git (最新版本即可)

安装 pre-commit 代码检查工具（可选）

安装 VSCode，PyCharm 或其它 IDE

三、 实验内容

基于蓝鲸 SaaS 开发框架开发一个独立 SaaS 应用，借助蓝鲸 CMDB 配置平台实现游戏业务主机资源拉取与查询，通过蓝鲸网关/ESB 组件 API 联通 CMDB 平台实现数据获取，并根据 CMDB 主机数据结构，设计查询条件与对应接口。

四、 实验评分标准

整体要求：请同学们采用迭代方式进行需求分析、面向对象设计和编程实现，实训课报告中需包含相应的需求规约、设计规约、接口文档，项目开发说明

考点一： 创建 SaaS 应用，通过蓝鲸 ESB 组件 API 联通 CMDB 配置平台，实现业务、集群、模块级联拉取接口，并在前端进行下拉框组件展示与数据渲染

相关资料：

1.蓝鲸 CMDB 配置平台（CE 环境）：蓝鲸 CMDB 配置平台

2.蓝鲸组件 API 文档：蓝鲸组件 API 文档中心

3.蓝鲸 MagicBox 组件-下拉选框：蓝鲸 MagicBox 组件-下拉选框 Select

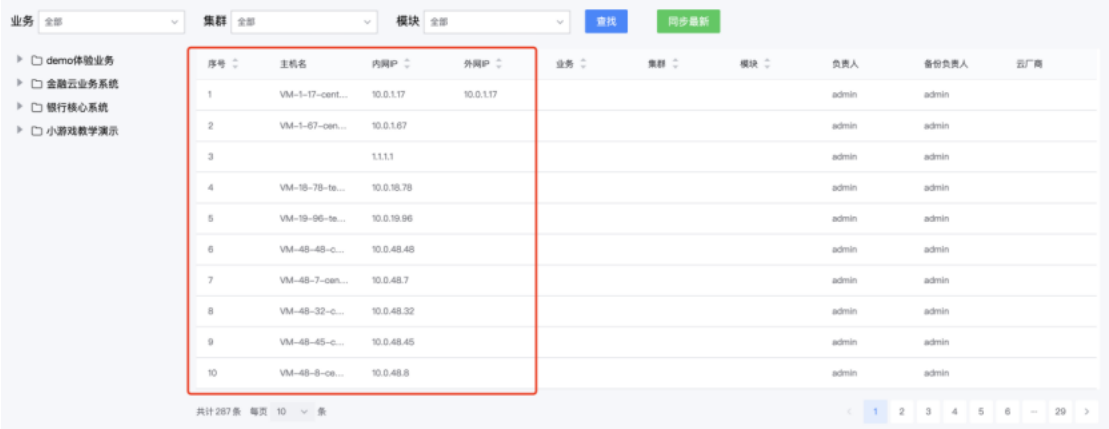
考点二：添加 根据蓝鲸 CMDB 配置平台的主机数据结构设计查询条件（包括但不限于主机名称、主机维护人、主机备份人等字段），实现主机查询接口（模糊查询可加分）

相关资料：

1、输入框组件：蓝鲸 MagicBox 组件-输入框 Input

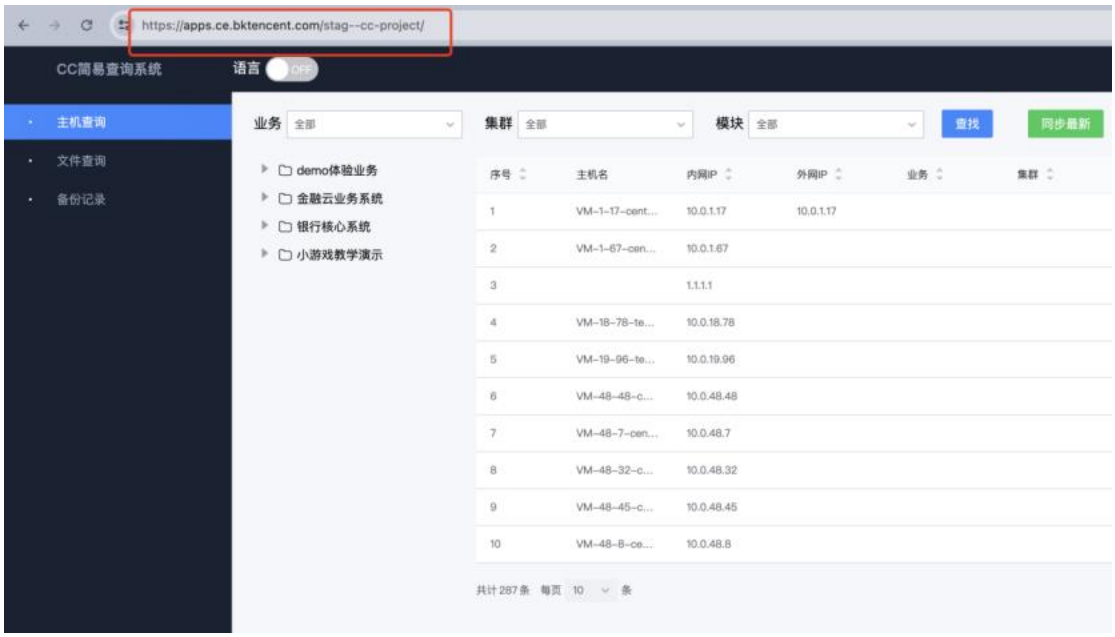
2、主机查询接口文档：主机查询接口文档

考点三：设计前端界面（可参考课程前端样例代码与 MagicBox 组件库），进行前后端联调，实现前端主机列表数据渲染，如下图：



考点四：实现主机详情展示接口&界面，要求点击主机后能够查看主机详情信息并通过前端界面进行数据展示，可参考下图：

考点五：将实现的后端&前端代码上传至 Git 代码托管平台，并部署到 PaaS 平台，数据交互展示无误，不存在 CORS、CSRF 等问题



相关资料：

1、环境搭建&应用部署文档：手把手搭建蓝鲸开发框架环境-Windows 系统

其他评分项：

1.Python 代码符合 PEP8 规范，可酌情加分

2.系统边界考虑完善，系统性能优良，可酌情加分

3.设计对应数据 Model，将 CMDB 配置信息存储到 SaaS 应用数据库中，可酌情加分

4.前端界面优美，用户交互体验良好，可酌情加分

5.实现数据同步功能，可酌情加分

6.后端代码能够实现单元测试以及日志、异常处理等，可酌情加分

五、 实验过程与结果

1. 后端前置准备

与实验 1 中的步骤大致相同，可直接参考实验 1。（后端模块的模板使用实验材料中的模板）

2. 前端前置准备

（1） 安装 nodejs

```
(saas) [K] 3540.22ms saas-expr-1 $ scoop list | findstr nodejs
Installed apps:
nodejs          23.2.0       main              2024-11-14 18:34:37
(saas) [K] 311.92ms saas-expr-1 $ npm -v
10.9.0
```

（2） 在 gitee 上新建前端用的 gitee 仓库

（3） 在蓝鲸开发者中心中新建前端模块（前端模块的模板使用实验材料中的模板）

在蓝鲸开发者中心中->进入应用->模块配置->新增一个模块作为前端部署模块：

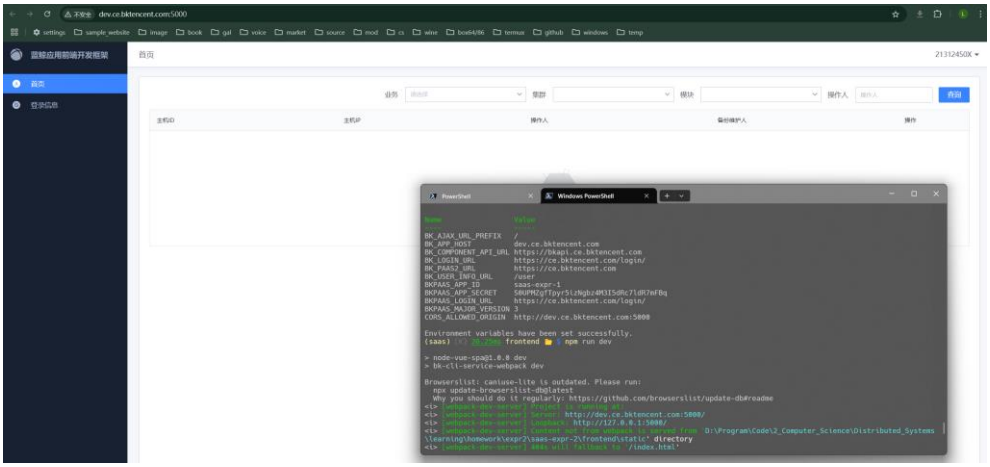
（4） 到前端项目根目录中执行 **npm install** 进行依赖包的下载

```
(saas) [K] 36.46ms frontend $ npm install
(node:32632) ExperimentalWarning: CommonJS module D:\ScoopApp\Scoop\apps\nodejs
debug\src\node.js is loading ES Module D:\ScoopApp\Scoop\apps\nodejs\23.2.0\noc
lor\index.js using require().
Support for loading ES Module in require() is an experimental feature and might
(Use `node --trace-warnings ...` to show where the warning was created)
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: 'node-vue-spa@1.0.0',
npm warn EBADENGINE   required: { node: '>= 14.16.1', npm: '>= 6.4.1 <7' },
npm warn EBADENGINE   current: { node: 'v23.2.0', npm: '10.9.0' }
npm warn EBADENGINE }
```

（5） 编写用于初始化前端环境变量的脚本 **set-env.ps1**

```
$env:BK_LOGIN_URL = 'https://ce.bktencent.com/login/'
$env:BK_APP_HOST = 'dev.ce.bktencent.com'
$env:BK_AJAX_URL_PREFIX = '/'
$env:BK_USER_INFO_URL = '/user'

Get-ChildItem Env: | Where-Object { $_.Name -like "BK*" -or $_.Name -like "CORS*" } | Format-Table -AutoSize

Write-Host "Environment variables have been set successfully."
```

（6） 执行 **set-env.ps1** 脚本并执行 **npm run dev** 启动 **vue** 框架



3. 编写后端

（1） 创建 local_settings.py，将本地数据库信息编写到其中

```python
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "lesson11",  # noqa
        "USER": "root",
        "PASSWORD": "",
        "HOST": "localhost",
        "PORT": "3306",
    },
}
```

（2） 编写 home_application/views.py 如下：

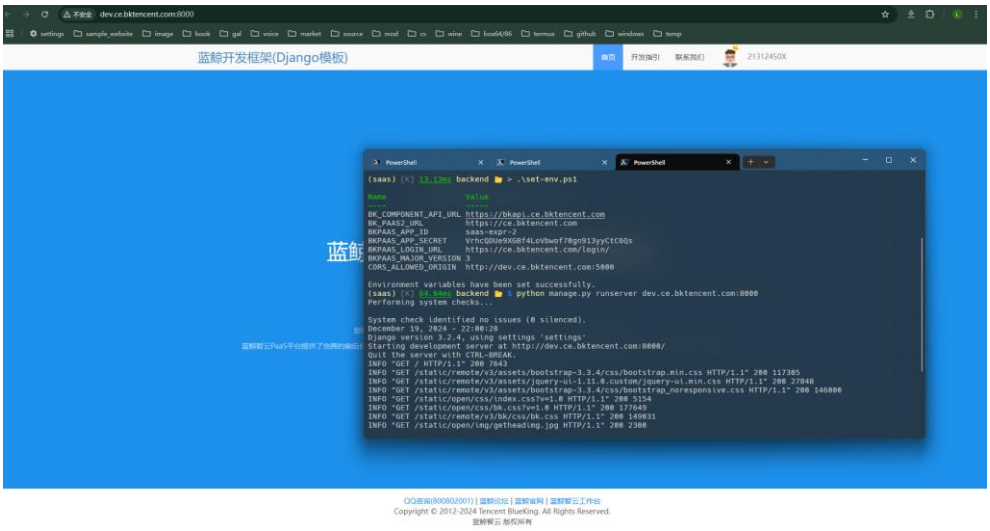
```python
from django.shortcuts import render

from blueking.component.shortcuts import get_client_by_request


# 开发框架中通过中间件默认是需要登录态的，如有不需要登录的，可添加装饰器login_exempt
# 装饰器引入 from blueapps.account.decorators import login_exempt
def home(request):
    """
    首页
    """
    return render(request, "home_application/index_home.html")


def dev_guide(request):
    """
    开发指引
    """
    return render(request, "home_application/dev_guide.html")


def contact(request):
    """
    联系页
    """
    return render(request, "home_application/contact.html")
```

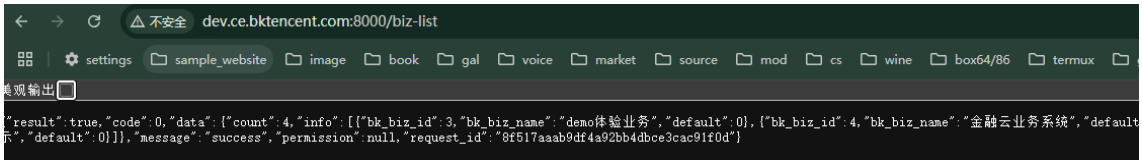
```python
# 开发样例，在这里实现了【查询业务列表接口】
def get_bizs_list(request):
    """
    获取业务列表
    """
    # 从环境配置获取APP信息，从request获取当前用户信息
    client = get_client_by_request(request)
    # 请求参数
    kwargs = {
        "fields": [
            "bk_biz_id",
            "bk_biz_name"
        ],
        # 社区版环境中业务数量有限，故不考虑分页情况
        "page": {
            "start": 0,
            "limit": 10,
            "sort": ""
        }
    }
    # 这里需要填写对应的组件API的入口地址
    result = client.cc.search_business(kwargs)
    return JsonResponse(result)


def get_sets_list(request):
    """
    根据业务 ID，查询业务下的集群列表
    """
    client = get_client_by_request(request)
    # 请求参数
    kwargs = {
        "bk_biz_id": request.GET.get('bk_biz_id'),  # 从 request.GET 中获取传递的查询参数
        "fields": ["bk_set_id", "bk_set_name", "bk_biz_id",
                   "bk_created_at", "bk_supplier_account"],
    }
    result = client.cc.search_set(kwargs)
    return JsonResponse(result)


def get_modules_list(request):
    """
    根据业务 ID 和集群 ID，查询对应下的模块列表
    """
    client = get_client_by_request(request)
    # 构造请求参数
    kwargs = {
        "bk_biz_id": request.GET.get('bk_biz_id'),
        "bk_set_id": request.GET.get('bk_set_id'),
        "fields": ["bk_module_id", "bk_module_name", "bk_set_id",
                   "bk_biz_id", "bk_created_at", "bk_supplier_account"],
    }
    result = client.cc.search_module(kwargs)
    return JsonResponse(result)


def get_hosts_list(request):
    """
    根据传递的查询条件，包括但不限于（业务 ID、集群 ID、模块 ID、主机 ID、主机维护人）
    查询主机列表
    """
    client = get_client_by_request(request)
    # 构造请求参数
    kwargs = {
        "bk_biz_id": request.GET.get("bk_biz_id"),
        # TODO 待优化项：学有余力的同学可尝试实现分页
        "page": {
            "start": 0,
            "limit": 100,
        },
        "fields": [
            "bk_host_id",      # 主机ID
            "bk_cloud_id",     # 云区域ID
            "bk_host_innerip", # 主机内网IP
            "bk_os_type",      # 操作系统类型
            "bk_mac",          # 主机MAC地址
            "operator",        # 操作人
            "bk_bak_operator"  # 备份维护人
        ]
    }

    # 添加可选参数，包括但不限于主机ID、集群ID、模块ID...
    if request.GET.get("bk_set_id"):
        # kwargs["bk_set_id"] = request.GET.get("bk_set_id")
        # 错误写法，注意数据结构和数据结构约束已文档为准
        kwargs["bk_set_ids"] = [int(request.GET.get("bk_set_id"))]  # 注意这里的数据结构，仔细阅读接口文档

    if request.GET.get("bk_module_id"):
        # kwargs["bk_set_id"] = request.GET.get("bk_set_id")
        # 错误写法，注意数据结构和数据结构约束已文档为准
        kwargs["bk_module_ids"] = [int(request.GET.get("bk_module_id"))]  # 注意这里的数据结构，仔细阅读接口文档

    result = client.cc.list_biz_hosts(kwargs)
    return JsonResponse(result)

def get_host_detail(request):
    """
    根据主机ID，查询主机详情信息
    """
    client = get_client_by_request(request)

    kwargs = {
        "bk_host_id": request.GET.get("bk_host_id")
    }

    result = client.cc.get_host_base_info(kwargs)
    return JsonResponse(result)
```

（3） 编写 urls.py 如下：

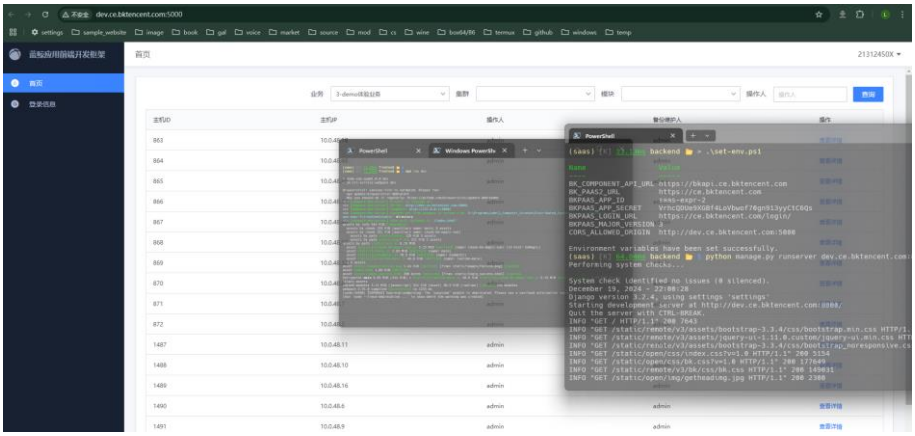
```python
from django.conf.urls import url

from . import views

urlpatterns = (
    url(r"^$", views.home),
    url(r"^dev-guide/$", views.dev_guide),
    url(r"^contact/$", views.contact),
    url(r"biz-list",views.get_bizs_list),
    url(r"set-list",views.get_sets_list),
    url(r"module-list",views.get_modules_list),
    url(r"host-list",views.get_hosts_list),
    url(r"host-detail",views.get_host_detail)
)

```

（4） 运行测试如下：





4. 本地运行测试

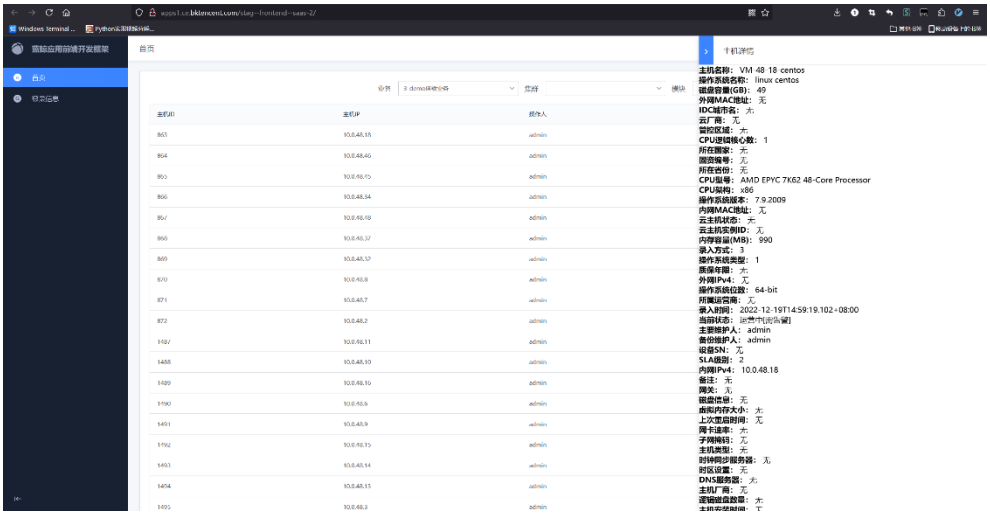


5. 蓝鲸部署运行测试

（1） 设置后端模块的环境变量如下：

（2） 设置前端模块的环境变量如下

（3） 部署后的效果如下：



六、 实验心得与体会

通过这次基于配置平台进行游戏主机管理的实验，我深入理解了蓝鲸 CMDB 配置平台的功能和数据结构。学习使用蓝鲸网关/ESB 组件 API 不仅让我掌握了 API 调用和鉴权的方法，也让我体会到了不同系统间数据交互的重要性。

在开发过程中，实现业务、集群、模块级联拉取接口，以及设计主机查询和详情展示功能，极大地提升了我的 SaaS 开发技能。前后端联调的经验让我更好地理解了接口设计的重要性，也提高了我处理 CORS、CSRF 等问题的能力。

将项目部署到 PaaS 平台的过程，让我对整个开发流程有了更全面的认识。这次实验不仅巩固了我的软件开发和设计能力，也让我深刻认识到了在实际项目中处理复杂数据结构和系统集成的重要性。