第四次作业-基于 BKVision 图表平台实现用户画像与行为分析

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 BKVision 图表平台的产品功能与使用方法

(3) 掌握基本 Django 中间件的开发技能与数据采集

(4) 掌握蓝鲸图表平台的嵌入方式与 SDK 使用

(5) 提升 SaaS 开发技能，巩固基础数据分析能力与数据采集技能

(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

在此前两期 SaaS 开发作业的基础上，借助蓝鲸 BKVision 图表平台实现用户行为可视化分析与前端嵌入，通过设计并开发 Django 中间件，实现用户行为数据埋点采集并存储至数据库，通过 BKVision 实现仪表盘嵌入。

四、 实验评分标准

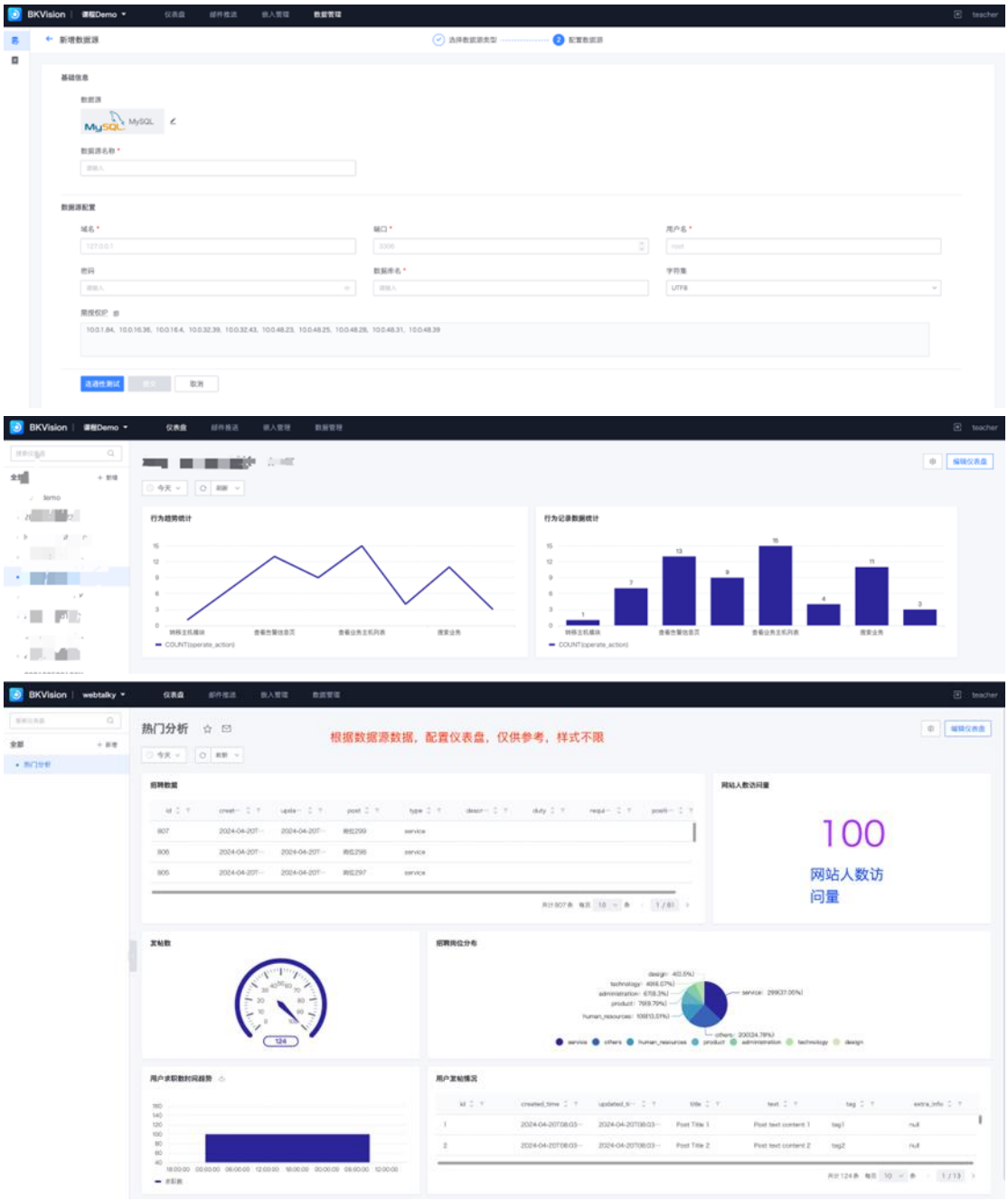
整体要求：请同学们采用迭代方式进行需求分析、面向对象设计和编程实现，实训课报告中需包含相应的需求规约、设计规约、接口文档，项目开发说明

考点一： 在此前两期课程 SaaS 作业的基础上，通过 Django 中间件实现用户行为采集并存储到 SaaS 数据库，比如：登录行为、查询业务列表行为、执行作业行为等

相关资料：

1. Django 中间件开发：Django 中间件开发
2. Django 中间件原理及示例：Django 中间件原理及示例

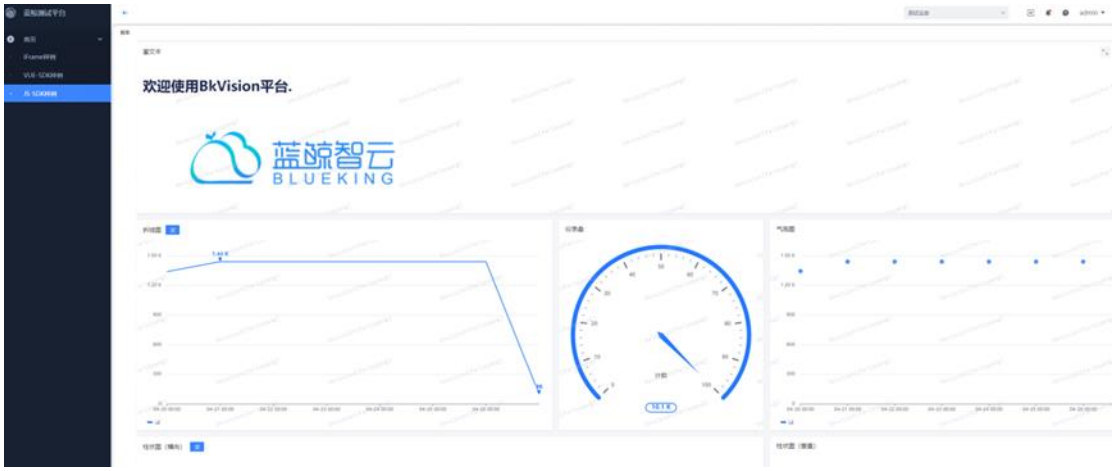
考点二：在 BKVision 图表平台创建空间，接入对应的 SaaS 数据库，并对采集的数据进行仪表盘配置（仪表盘样式不限，鼓励大家自由发挥），并发布仪表盘，参考图如下：



相关资料：

1、蓝鲸图表平台：[蓝鲸 BKVision 图表平台]

考点三：设计通过 iFrame 或 BKVision-SDK 方式，实现仪表盘发布并嵌入到对应的 SaaS 前端界面中，参考图如下：



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.Django 中间件实现出色，采集覆盖大部分接口场景，可酌情加分

4.Django 中间件在实现数据存储时，能够通过 Celery 异步任务实现，可酌情加分

5.前端界面优美，用户交互体验良好，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. 设计实现用户行为数据存储 Model

（1） 在 home_application/models.py 中实现 API 请求次数记录表数据 Model ApiRequestCount 如下：

```python
class ApiRequestCount(models.Model):
    """
    API 请求次数记录模型，用于运营分析
    """

    api_category = models.CharField(verbose_name="API 类别", max_length=255)
    api_name = models.CharField(verbose_name="API 名称", max_length=255)
    request_count = models.IntegerField(verbose_name="请求次数", default=0)

    class Meta:
        unique_together = ("api_category", "api_name")  # 联合唯一索引
        verbose_name = "API 请求次数"
        verbose_name_plural = "API 请求次数"

    def __str__(self):
        return f"{self.api_category}-{self.api_name}"
```

（2） 执行数据库迁移

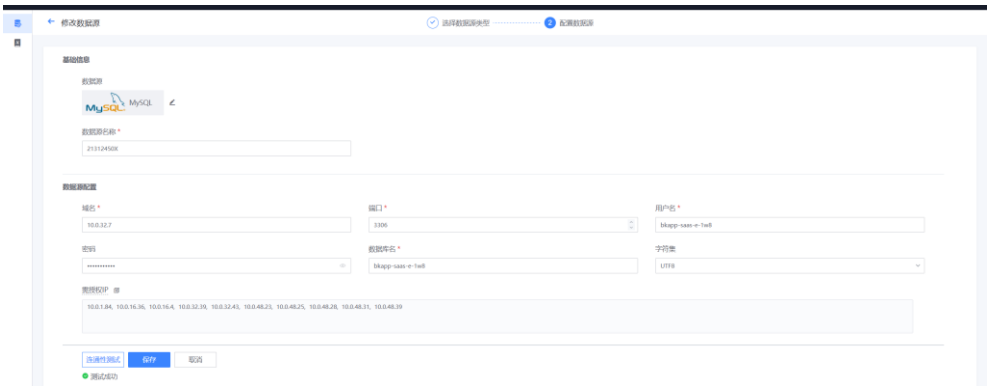
2. 设计自定义中间件，实现用户行为埋点记录

（1） 编写 core/middleware.py 如下：

```python
# -*- coding: utf-8 -*-

from django.utils.deprecation import MiddlewareMixin
import logging
from django.db.models import F

from home_application.models import ApiRequestCount

logger = logging.getLogger(__name__)

# 这里的CMDB和JOB对应的名称应该同你的URL定义的前缀，详见 /home_application/urls.py

CMDB_BEHAVIORS = [
    'biz-list',
    'set-list',
    'module-list',
    'host-list',
    'host-detail'
]

JOB_BEHAVIORS = [
    'search-file',
    'backup-file',
    'backup-record'
]


class RecordUserBehaviorMiddleware(MiddlewareMixin):
    """
    自定义中间件-记录用户行为，进行埋点
    """

    def process_request(self, request):
        try:
            # 获取需要埋点存储的信息：用户名、请求的API名称、请求的API所属的类别（CMDB/JOB）
            username = request.user.username
            # 可以观察一下这里的request.path 数据格式为 xxxx/xxxx/实际API名称，因此我们使用split方法，以/进行分割，只取最后的API名称部分
            api_name = request.path.split('/')[-1]
            # 判断接口所属类别
            api_category = 'CMDB' if api_name in CMDB_BEHAVIORS else 'JOB' if api_name in JOB_BEHAVIORS else 'Unknown'

            # TODO：这里的埋点记录行为，涉及DB操作，会影响接口响应时间，能否改为异步记录？ 参考Celery异步任务

            # 根据 api_category 和 api_name 记录请求次数
            api_request_count, _ = ApiRequestCount.objects.get_or_create(api_category=api_category, api_name=api_name)
            api_request_count.request_count = F("request_count") + 1
            api_request_count.save()
        except Exception as e:  # pylint: disable=broad-except
            # 这里即使产生了异常，也应该继续往后执行，因为埋点记录不应该影响用户请求接口，应该是静默的，所以建议学有余力的同学尝试进行异步优化
            logger.exception(f"Unexpected Exception when record user behavior:{e}")
            pass
        return None
```

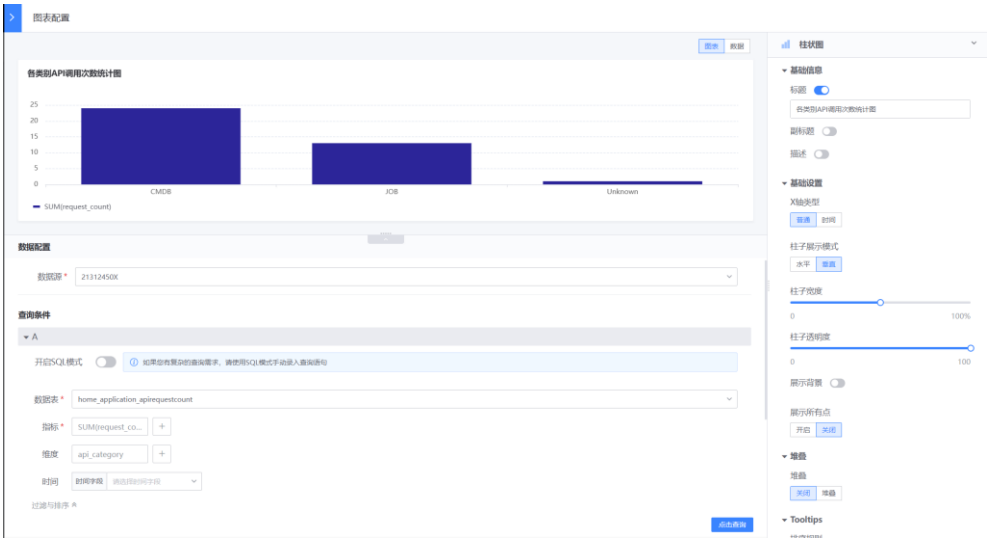
（2） 添加自定义中间件到 config/default.py 中：

3. 在 BKVision 平台添加并配置数据源

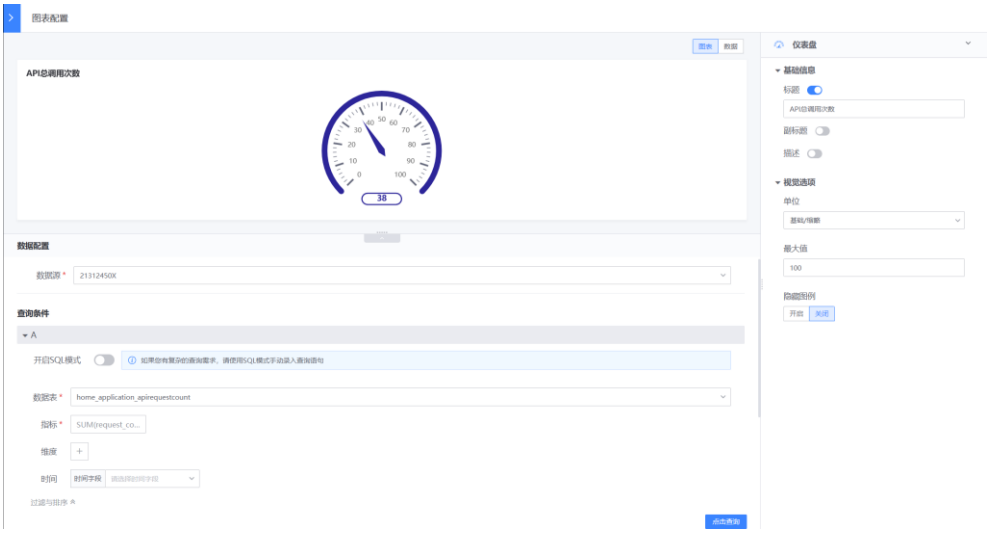


4. 创建并配置仪表盘

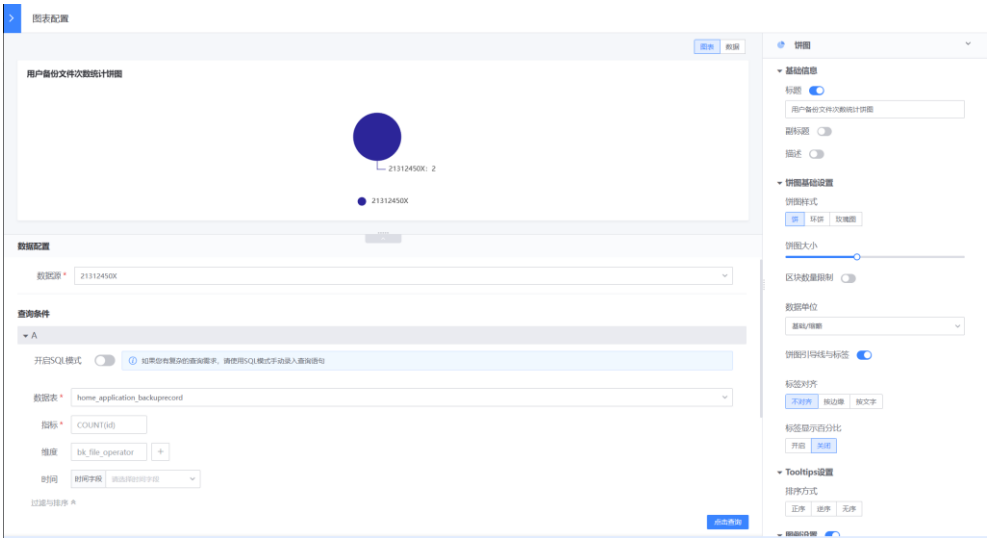
（1） 配置实现 API 调用类别柱状统计图



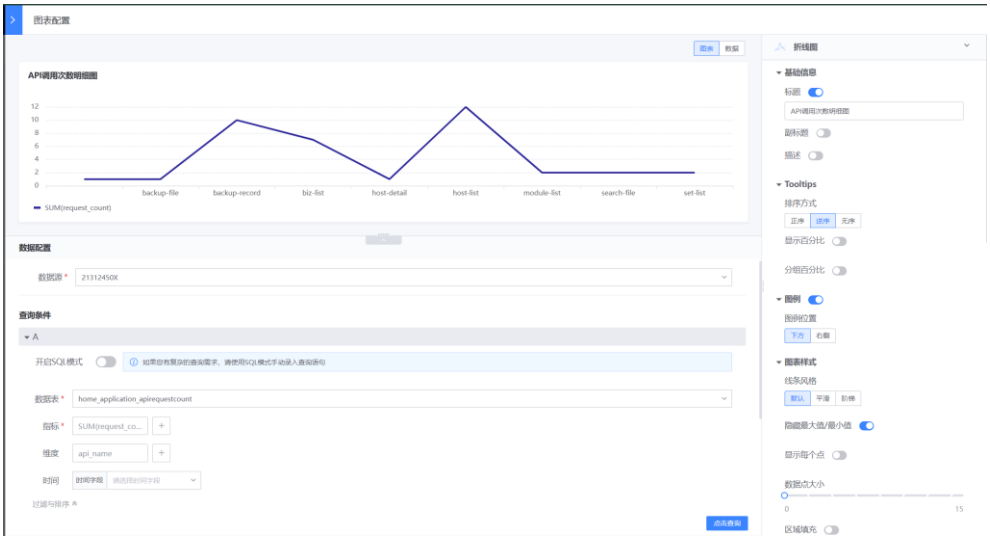
（2） 配置实现 API 总调用次数仪表盘



（3） 配置实现用户备份文件次数统计饼图

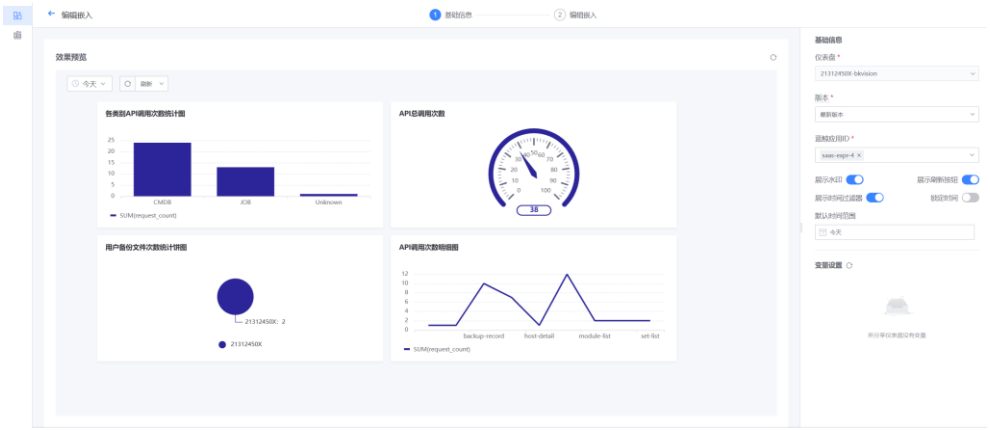


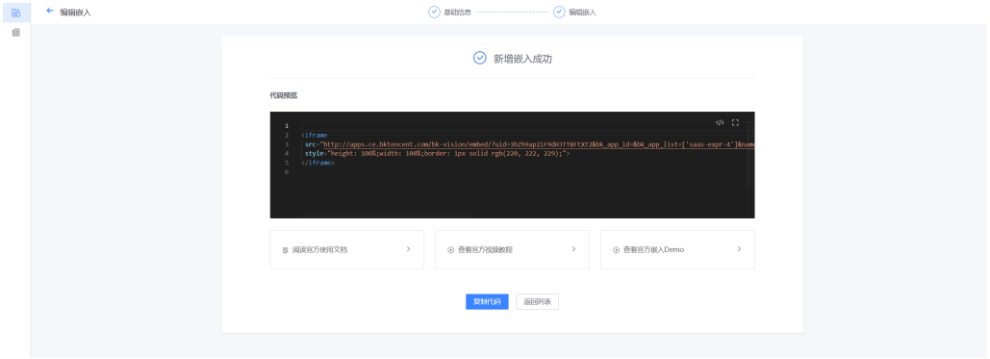
（4） 配置实现 API 调用次数明细图



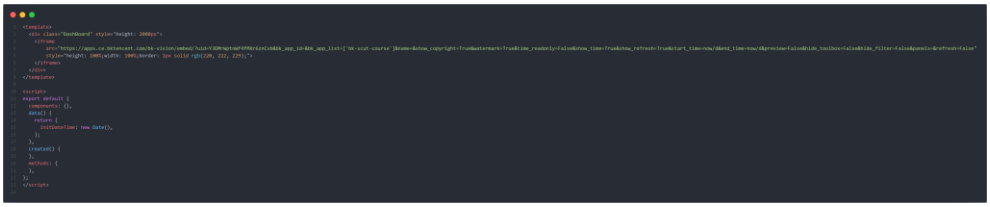
5. 将仪表盘嵌入到自己的前端应用

（1） 新增 iFrame 嵌入



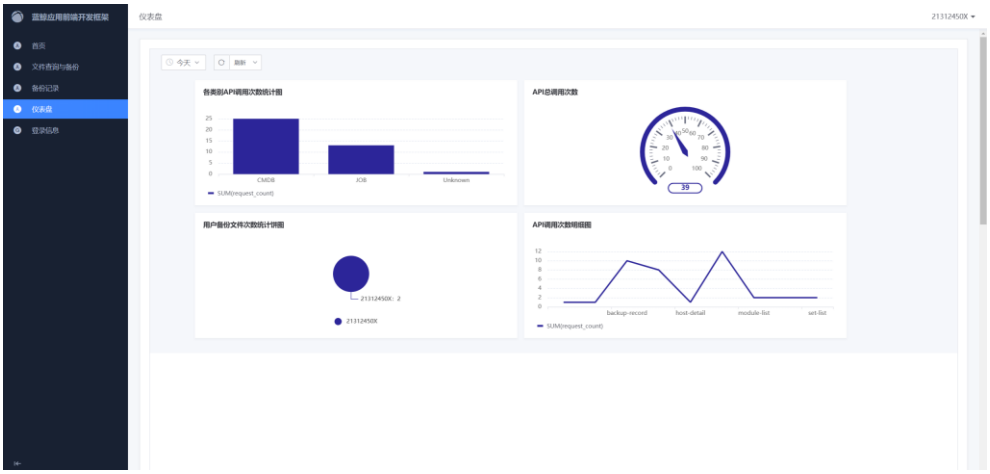


（2） 将 iframe 代码块添加到前端的 src/views/DashBoard/index.vue 中，如下：



6. 部署上线

最终效果如下：



六、 实验心得与体会

通过这次基于 BKVision 图表平台实现用户画像与行为分析的实验，我深入了解了数据可视化的重要性和实现方法。开发 Django 中间件进行用户行为数据采集，不仅提升了我的后端开发技能，也让我理解了数据埋点的原理和价值。在 BKVision 平台上创建和配置仪表盘的过程，让我掌握了数据可视化的基本技巧。将仪表盘嵌入 SaaS 应用的实践，使我对前后端集成有了更深入的认识，同时也体会到了数据驱动决策的重要性。