实现可审计的安全API设计

> 21301102 钱波

1. 本地数据库创建

- 创建数据库api3
- 使用如下语句创建student表

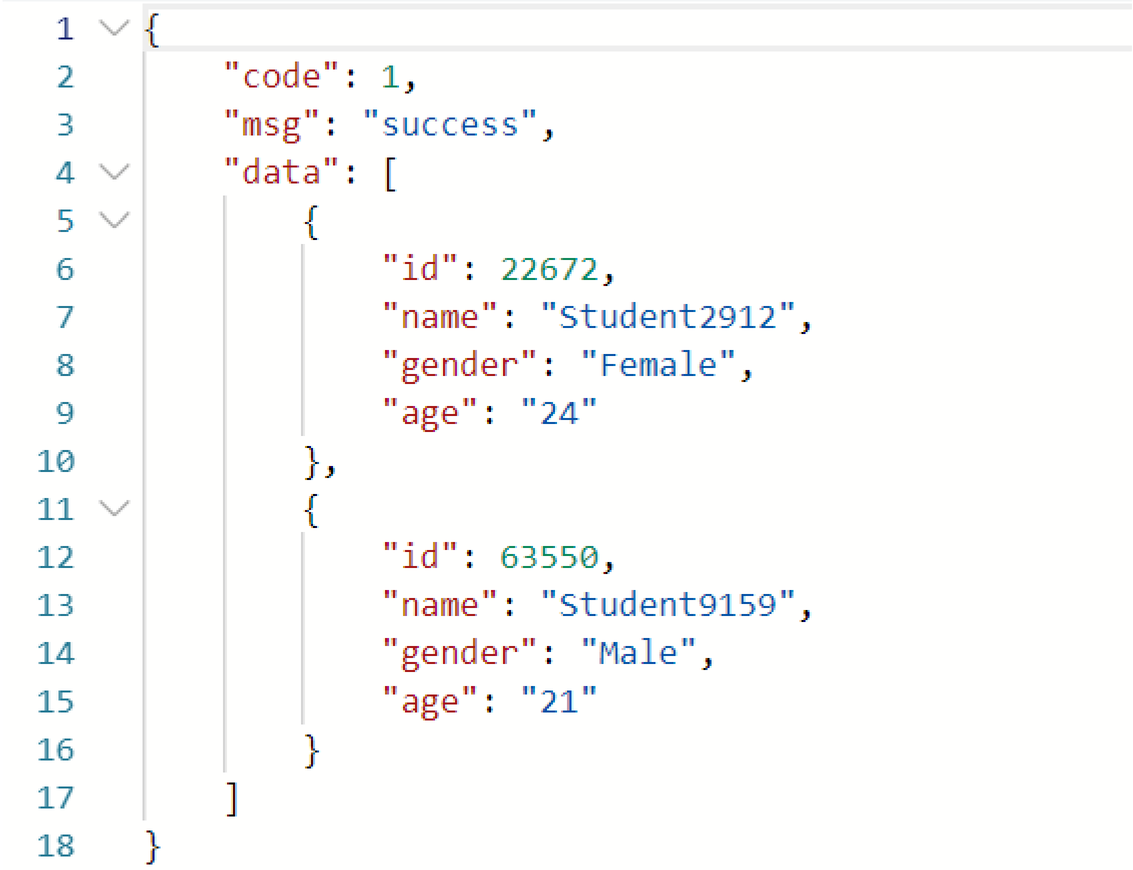
```
create table if not exists student
(
    id     int auto_increment comment '学生id'
        primary key,
    name   varchar(100) null comment '姓名',
    gender varchar(8)   null comment '性别',
    age    int          null comment '年龄'
);
```

- 使用存储过程向数据库中插入10w条数据

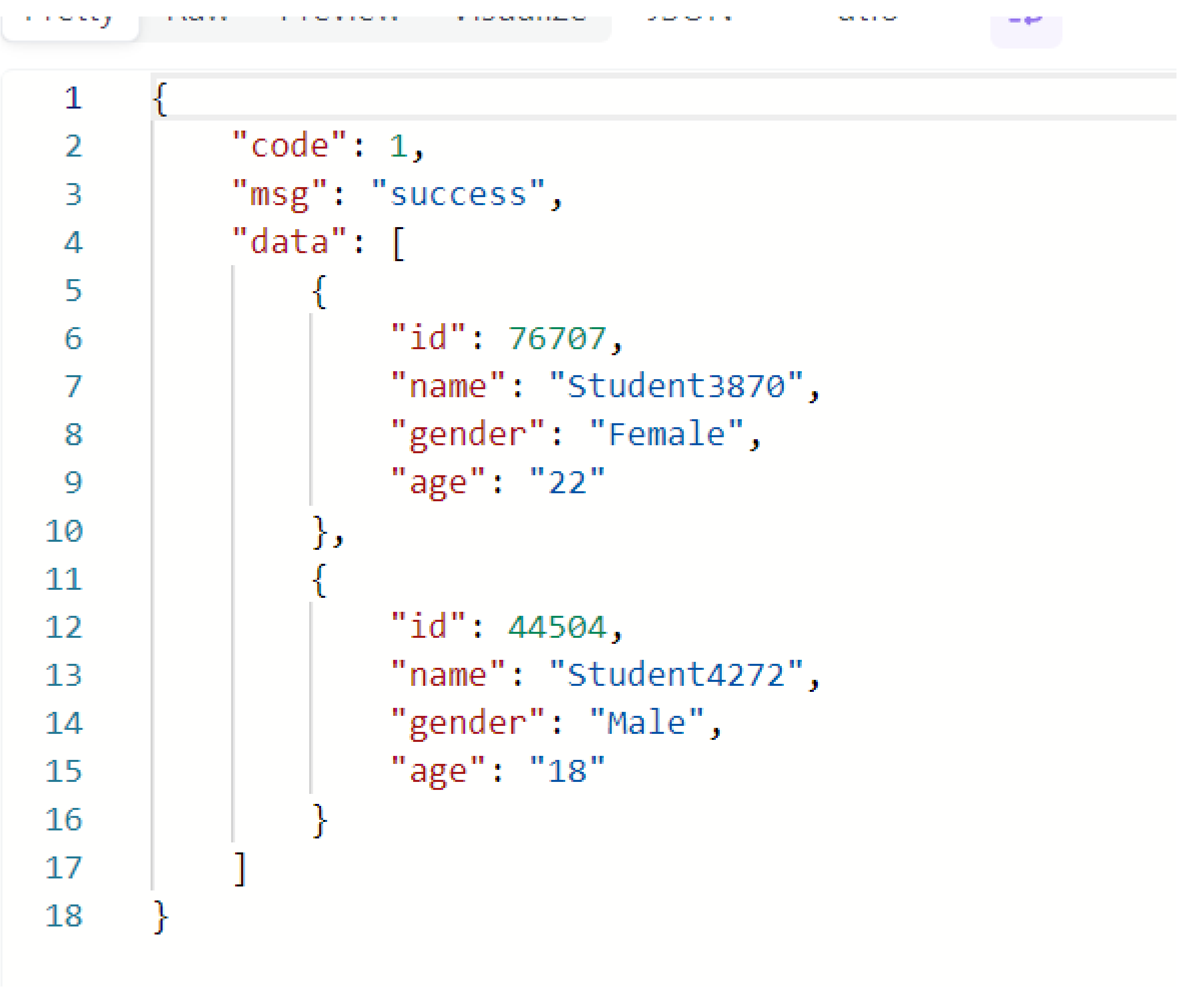
```
CREATE PROCEDURE insert_random_students(IN num_rows INT)
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE name VARCHAR(100);
    DECLARE gender VARCHAR(8);
    DECLARE age INT;

    WHILE i < num_rows DO
            SET name = CONCAT('Student', FLOOR(1 + RAND() * 10000)); -- 随机生成名字
            SET gender = CASE WHEN RAND() > 0.5 THEN 'Male' ELSE 'Female' END; -- 随机生成性别
            SET age = FLOOR(18 + RAND() * 10); -- 随机生成年龄（18到27岁）

INSERT INTO student (name, gender, age) VALUES (name, gender, age);

SET i = i + 1;
END WHILE;
END;
```

2. 实现学生信息查询接口

- 使用实现学生信息查询接口(http接口)，入参为1~1000的int值，返回相应数量的随机学生信息
  - 实现方式
    - 数据库中随机排序
    - 选择前num条数据，num为传入的参数
  - 结果验证

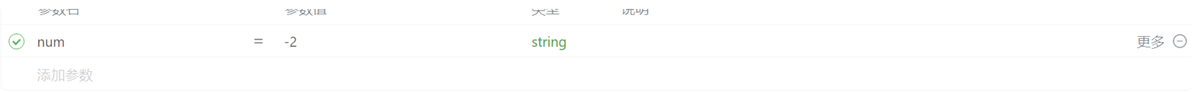


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 22672,
            "name": "Student2912",
            "gender": "Female",
            "age": "24"
        },
        {
            "id": 63550,
            "name": "Student9159",
            "gender": "Male",
            "age": "21"
        }
    ]
}
```

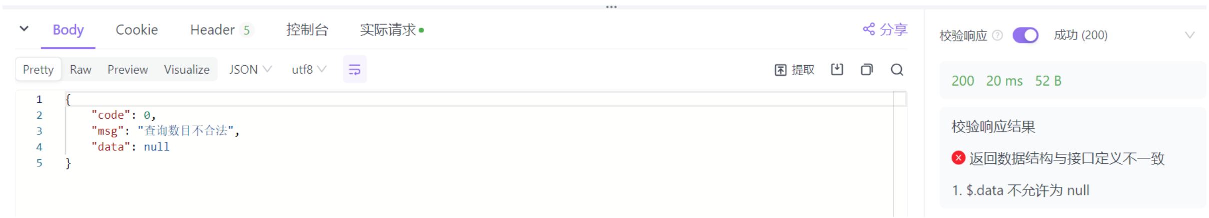


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 76707,
            "name": "Student3870",
            "gender": "Female",
            "age": "22"
        },
        {
            "id": 44504,
            "name": "Student4272",
            "gender": "Male",
            "age": "18"
        }
    ]
}
```

- 入参异常值的接口自我保护
  - 实现方式
    - 对入参num进行判断，参数小于1或大于1000则返回error
  - 结果验证



参数名		参数值	类型	说明	
✓ num	=	-2	string		更多 ⊖
添加参数					



```
{
    "code": 0,
    "msg": "查询数目不合法",
    "data": null
}
```

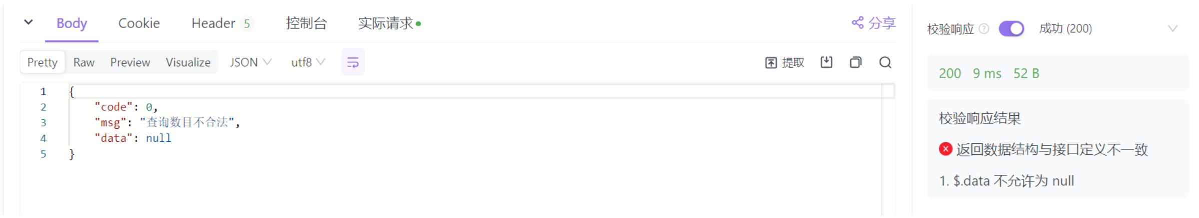
校验响应 成功 (200)

200 20 ms 52 B

校验响应结果

☒ 返回数据结构与接口定义不一致

1. $.data 不允许为 null



```
{
    "code": 0,
    "msg": "查询数目不合法",
    "data": null
}
```

校验响应 成功 (200)

200 9 ms 52 B

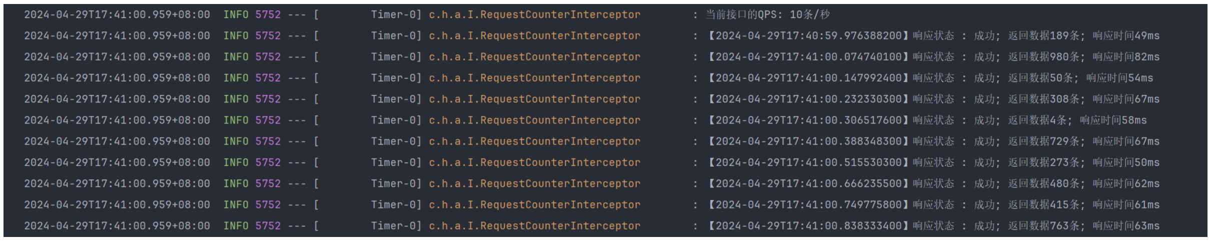
校验响应结果

☒ 返回数据结构与接口定义不一致

1. $.data 不允许为 null

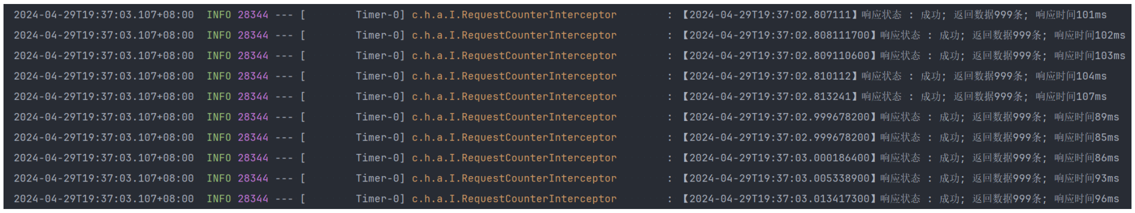
- 当前接口承接流量大小的监控（每秒多少条请求）

    - 实现方式
        - 使用springboot的拦截器，在preHandle中记录每秒的请求数量QPS，类型AtomicInteger，保证原子操作，防止并发操作影响。同时使用计时器来实现每隔一秒对QPS清零
    - 结果验证（这里使用apifox进行自动化测试，循环次数20，线程数1）



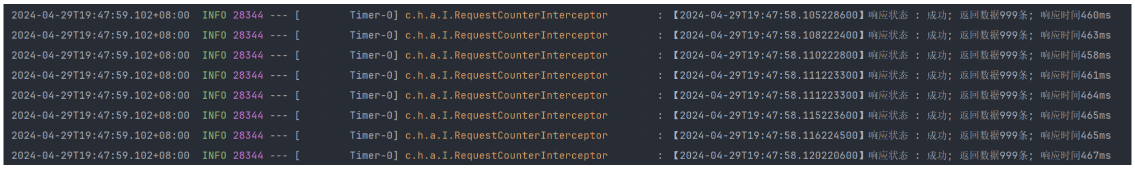
```
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 当前接口的QPS: 10条/秒
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:40:59.976388200】响应状态 : 成功; 返回数据189条; 响应时间49ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.074740100】响应状态 : 成功; 返回数据980条; 响应时间82ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.147992400】响应状态 : 成功; 返回数据50条; 响应时间54ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.232330300】响应状态 : 成功; 返回数据308条; 响应时间67ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.306517600】响应状态 : 成功; 返回数据4条; 响应时间58ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.388348300】响应状态 : 成功; 返回数据729条; 响应时间67ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.515530300】响应状态 : 成功; 返回数据273条; 响应时间50ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.666235500】响应状态 : 成功; 返回数据480条; 响应时间62ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.749775800】响应状态 : 成功; 返回数据415条; 响应时间61ms
2024-04-29T17:41:00.959+08:00  INFO 5752 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T17:41:00.838333400】响应状态 : 成功; 返回数据763条; 响应时间63ms
```

- 当前接口每条请求响应时间的监控（接口耗时），并分析不同qps下的性能
    - 实现方式
        - 在preHandle中记录请求的到达时间startTime，存入request中。在afterCompletion中记录处理完的时间，计算时间差即为响应时间。同时对于每条log信息，采用CopyOnWriteArrayList来储存，保证同步操作。
    - 结果验证（这里采用自定义脚本文件进行测试，仅截取部分数据）
        - qps为10时，平均耗时约为90ms



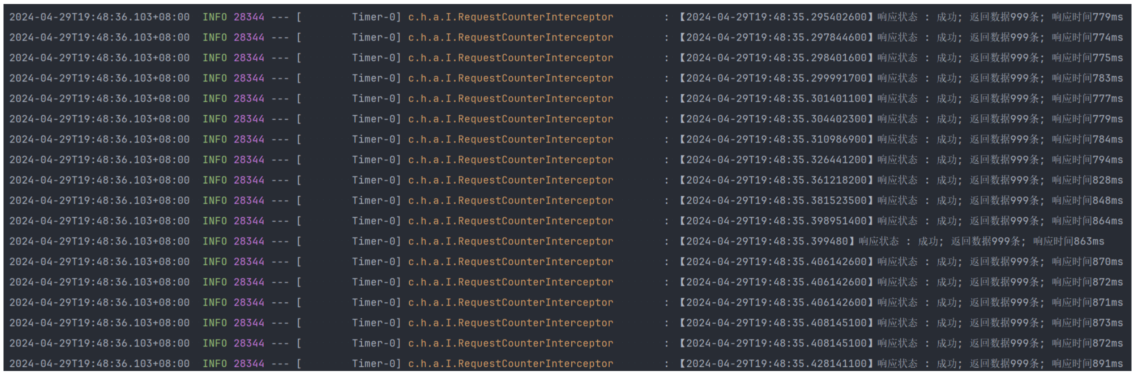
```
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.807111】响应状态 : 成功; 返回数据999条; 响应时间101ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.808111700】响应状态 : 成功; 返回数据999条; 响应时间102ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.809110600】响应状态 : 成功; 返回数据999条; 响应时间103ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.810112】响应状态 : 成功; 返回数据999条; 响应时间104ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.813241】响应状态 : 成功; 返回数据999条; 响应时间107ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.999678200】响应状态 : 成功; 返回数据999条; 响应时间89ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:02.999678200】响应状态 : 成功; 返回数据999条; 响应时间85ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:03.000186400】响应状态 : 成功; 返回数据999条; 响应时间86ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:03.005338900】响应状态 : 成功; 返回数据999条; 响应时间93ms
2024-04-29T19:37:03.107+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:37:03.013417300】响应状态 : 成功; 返回数据999条; 响应时间96ms
```

        - qps为50时，平均耗时约为465ms



```
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.105228600】响应状态 : 成功; 返回数据999条; 响应时间460ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.108222400】响应状态 : 成功; 返回数据999条; 响应时间463ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.110222800】响应状态 : 成功; 返回数据999条; 响应时间458ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.111223300】响应状态 : 成功; 返回数据999条; 响应时间461ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.111223300】响应状态 : 成功; 返回数据999条; 响应时间464ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.115223600】响应状态 : 成功; 返回数据999条; 响应时间465ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.116224500】响应状态 : 成功; 返回数据999条; 响应时间465ms
2024-04-29T19:47:59.102+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:47:58.120220600】响应状态 : 成功; 返回数据999条; 响应时间467ms
```

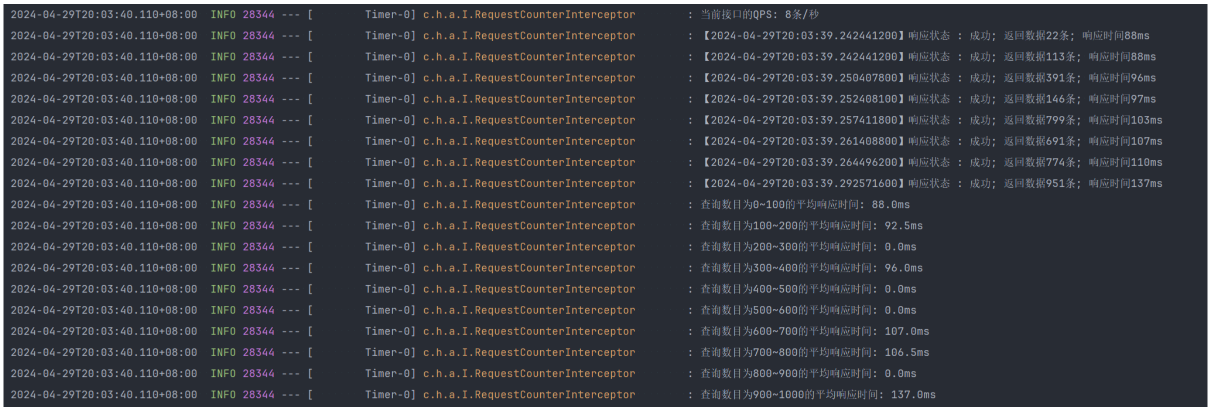
        - qps为100时，平均耗时约为700ms



```
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.295402600】响应状态 : 成功; 返回数据999条; 响应时间779ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.297844600】响应状态 : 成功; 返回数据999条; 响应时间774ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.298401600】响应状态 : 成功; 返回数据999条; 响应时间775ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.299991700】响应状态 : 成功; 返回数据999条; 响应时间783ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.301401100】响应状态 : 成功; 返回数据999条; 响应时间777ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.304402300】响应状态 : 成功; 返回数据999条; 响应时间779ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.310986900】响应状态 : 成功; 返回数据999条; 响应时间784ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.326441200】响应状态 : 成功; 返回数据999条; 响应时间794ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.361218200】响应状态 : 成功; 返回数据999条; 响应时间828ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.381523500】响应状态 : 成功; 返回数据999条; 响应时间848ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.398951400】响应状态 : 成功; 返回数据999条; 响应时间864ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.399480】响应状态 : 成功; 返回数据999条; 响应时间863ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.406142600】响应状态 : 成功; 返回数据999条; 响应时间870ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.406142600】响应状态 : 成功; 返回数据999条; 响应时间872ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.406142600】响应状态 : 成功; 返回数据999条; 响应时间871ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.408145100】响应状态 : 成功; 返回数据999条; 响应时间873ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.408145100】响应状态 : 成功; 返回数据999条; 响应时间872ms
2024-04-29T19:48:36.103+08:00  INFO 28344 --- [           Timer-0] c.h.a.I.RequestCounterInterceptor        : 【2024-04-29T19:48:35.428141100】响应状态 : 成功; 返回数据999条; 响应时间891ms
```

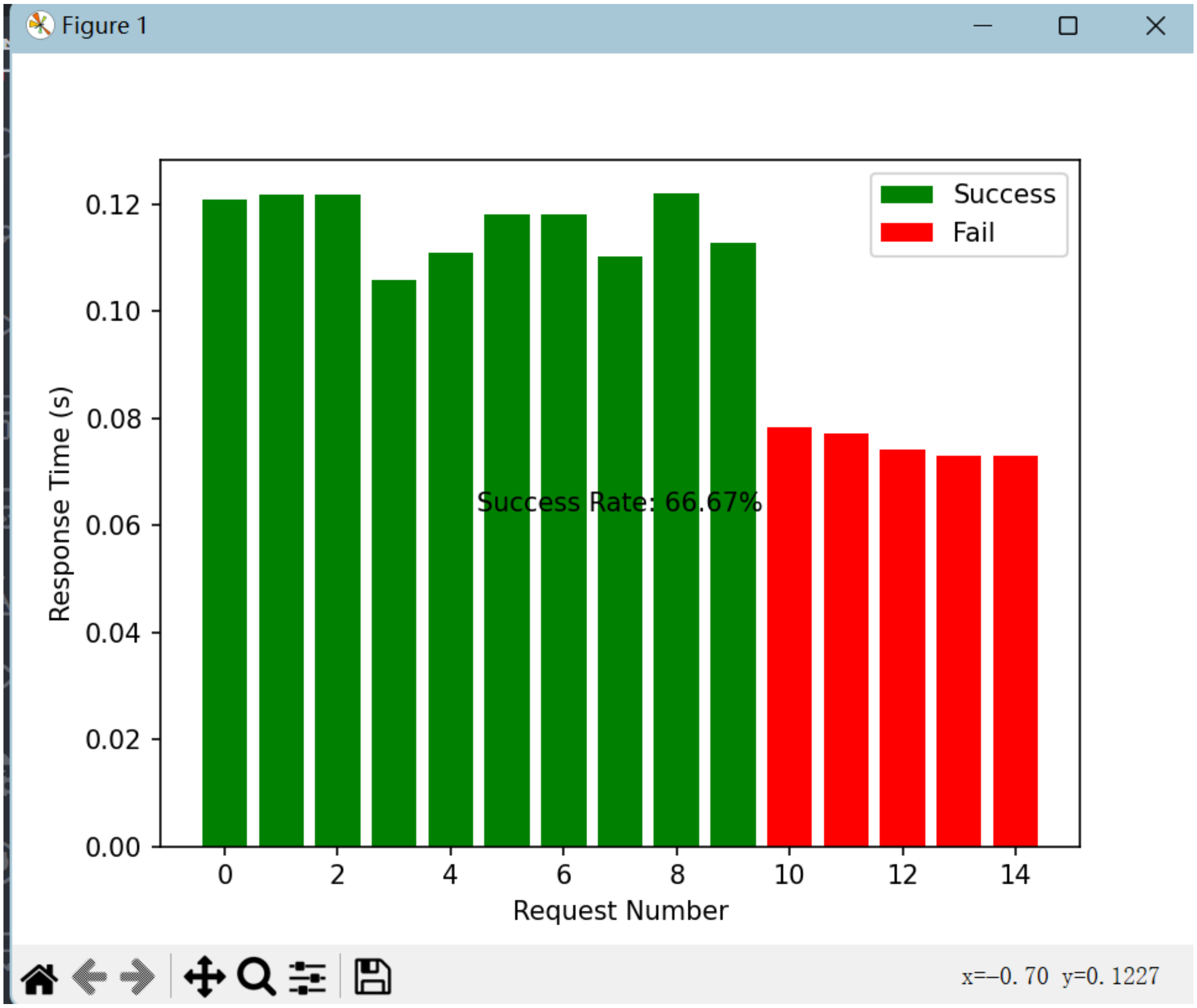
    - 结果分析
        - 不难看出，当qps逐渐增大的时候，接口响应耗时在逐渐增加。

- 当前接口信息运行信息日志（包括系统 业务信息），统计各档位（1~1000入参分为10个档位）的平均耗时
    - 实现方式
        - 实现方式与上一题类似，为了统计不同档位的入参，使用一个大小为10的数组来保存。
    - 结果验证（这里采用apifox进行自动化测试，循环为20，线程数为100）



```
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 当前接口的QPS: 8条/秒
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.242441200】响应状态 : 成功; 返回数据22条; 响应时间88ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.242441200】响应状态 : 成功; 返回数据113条; 响应时间88ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.250407800】响应状态 : 成功; 返回数据391条; 响应时间96ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.252408100】响应状态 : 成功; 返回数据146条; 响应时间97ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.257411800】响应状态 : 成功; 返回数据799条; 响应时间103ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.261408800】响应状态 : 成功; 返回数据691条; 响应时间107ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.264496200】响应状态 : 成功; 返回数据774条; 响应时间110ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 【2024-04-29T20:03:39.292571600】响应状态 : 成功; 返回数据951条; 响应时间137ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为0~100的平均响应时间: 88.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为100~200的平均响应时间: 92.5ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为200~300的平均响应时间: 0.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为300~400的平均响应时间: 96.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为400~500的平均响应时间: 0.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为500~600的平均响应时间: 0.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为600~700的平均响应时间: 107.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为700~800的平均响应时间: 106.5ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为800~900的平均响应时间: 0.0ms
2024-04-29T20:03:40.110+08:00  INFO 28344 --- [       Timer-0] c.h.a.I.RequestCounterInterceptor       : 查询数目为900~1000的平均响应时间: 137.0ms
```

- qps保护（即qps>10时，进行限流处理）
    - 实现方式，在handle方法中对qps进行判断，当qps>=10时，response的status设置为SC_SERVICE_UNAVAILABLE。
    - 结果验证（采用自定义脚本进行验证，当qps超过10的时候请求失败）



3. 实现批量curl请求的脚本

- 代码

```
from concurrent.futures import ThreadPoolExecutor
import random
import time
import requests
import matplotlib.pyplot as plt
def send_request(url, delay):
    try:
        start_time = time.time()
        response = requests.get(url)
        end_time = time.time()
        time.sleep(delay)  # 暂停一段时间
        if response.status_code != 200:
            raise requests.exceptions.HTTPError(f'HTTP
{response.status_code}: {response.reason}')
        return response.status_code, end_time - start_time
    except requests.exceptions.RequestException as e:
        end_time = time.time()
        return e, end_time - start_time

def batch_curl(urls, rate):
    delay = 1.0 / rate  # 计算每个请求之间的延迟
    success_times = []
    fail_times = []
    with ThreadPoolExecutor(max_workers=100) as executor:
        # 将参数打包成元组
        results = executor.map(send_request, urls, [delay]*len(urls))
    for i, result in enumerate(results):
        if isinstance(result[0], requests.exceptions.RequestException):
            print(f"Request failed with error: {result[0]}")
            fail_times.append((i, result[1]))
        else:
            print(f"Request succeeded with status code: {result[0]}")
            success_times.append((i, result[1]))
    # 计算成功率
    success_rate = len(success_times) / (len(success_times) +
len(fail_times))
    # 绘制响应时间的条形图
    if success_times:
        plt.bar(*zip(*success_times), color='green', label='Success')
    if fail_times:
        plt.bar(*zip(*fail_times), color='red', label='Fail')
    plt.legend(loc='upper right')
    plt.xlabel('Request Number')
    plt.ylabel('Response Time (s)')
    # 在图中添加成功率
    plt.text(0.5, 0.5, f'Success Rate: {success_rate*100:.2f}%',
horizontalalignment='center', verticalalignment='center',
transform=plt.gca().transAxes)
    plt.show()



# 每秒的请求数量
qps = 15
url = "http://localhost:8888/students"
urls = []
# 使用示例
for i in range(qps):
    num = 999
    urls.append(url + "?num=" + str(num))
batch_curl(urls, qps)
```

- 实验验证：（qps大于10时限流）

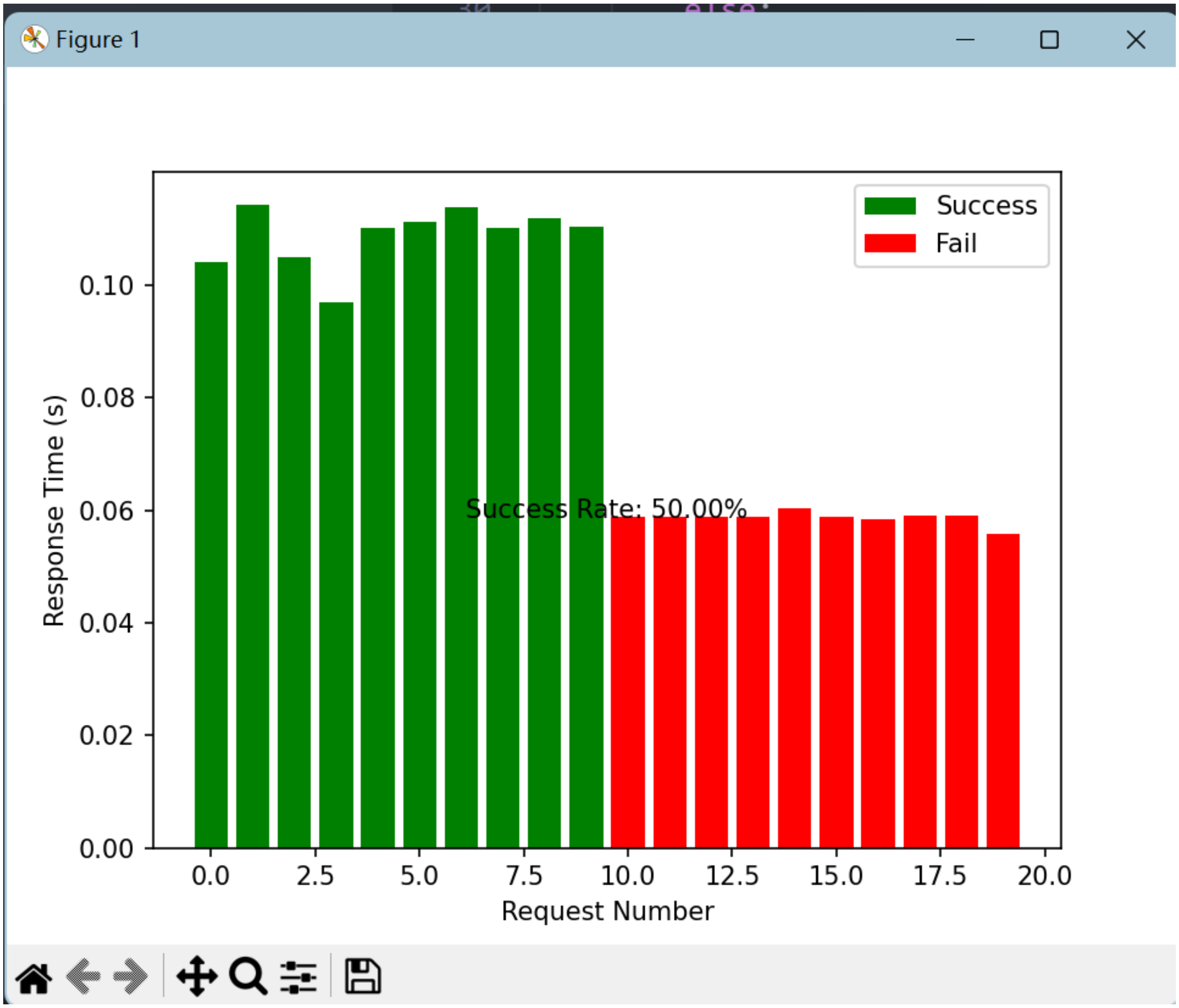


Figure 1
Success
Fail
Success Rate: 50.00%
Response Time (s)
0.00
0.02
0.04
0.06
0.08
0.10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Request Number