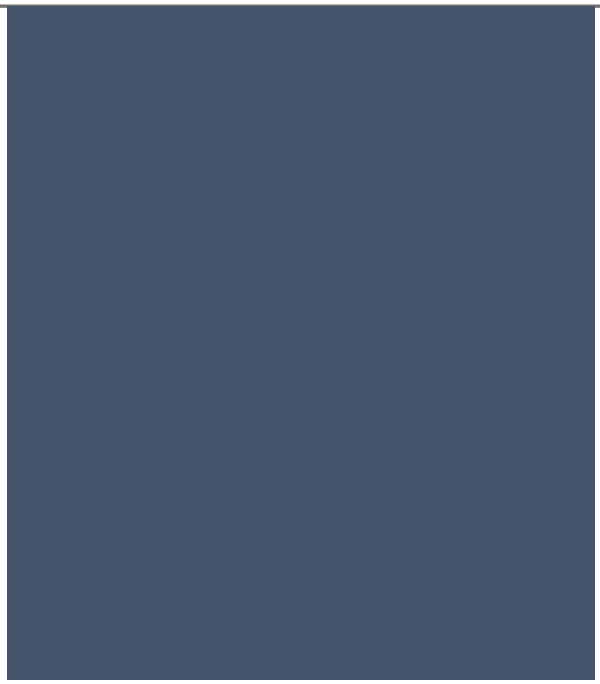


Design Report

Assignment 2

https://github.com/xiehao123/javaee-second-work.git

得分点	实现状态	实现报告
Basic requirement	完成	1-2 章节
Rate Limiting	完成	4 章节
API pagination	完成	5 章节
Online API document	完成	5 章节
Oauth2	完成	3 章节
Cache	完成	5.3 章节
Hateos	完成	5.4 章节

1 项目数据库设计

1.1 实体关系图

项目现阶段，主要包含以下三个数据库表，实体之间的关系如下图所示:

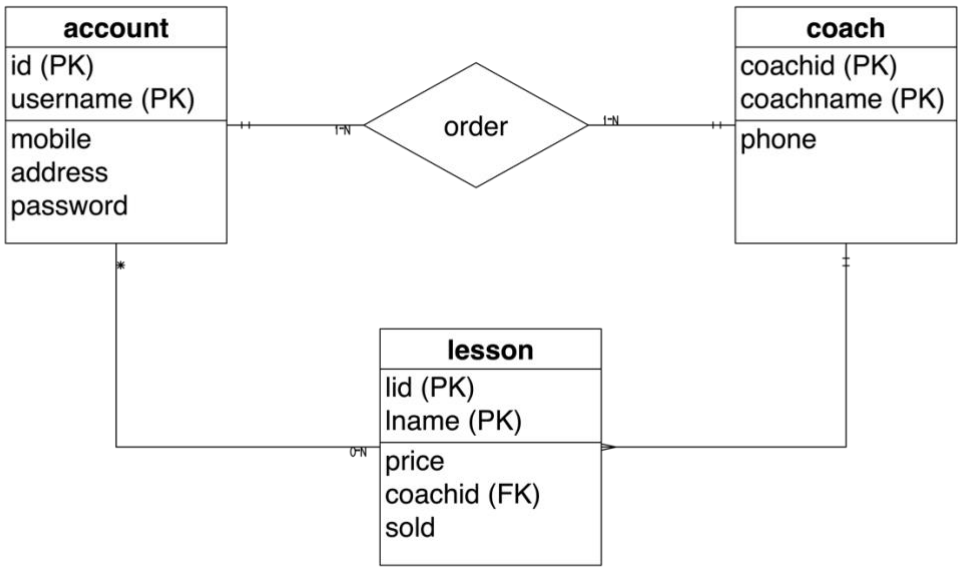


图 1 实体关系图

一个用户可以上多个课程，每个教练也可以同时教授多个课程。而一个用户也选择多个教练学习，而两者之间是通过用户购买课程生成订单进行的。

1.2 数据库表格设计

1. account 表

Id	username	mobile	address	Password
integer	string	integer	string	sting

2. coach 表

id	coachname	phone
integer	string	integer

3. lesson 表

Lid	lname	price	coachid	sold
integer	string	double	intger	integer

2 部分 RESTFUL API

JAVA EE RESTFUL ONLINE DOCUMENT

JAVA EE 课堂作业在线文档

account-controller : Account Controller — Show/Hide | List Operations | Expand Operations

DELETE	/v1/account	删除用户（通过用户名）
PUT	/v1/account	添加一个用户
GET	/v1/collection/accounts	获取用户列表
POST	/v1/login	登录

coach-controller : Coach Controller — Show/Hide | List Operations | Expand Operations

DELETE	/v1/coach	删除一个教练
PUT	/v1/coach	添加一个教练
GET	/v1/collection/coaches	获取教练列表

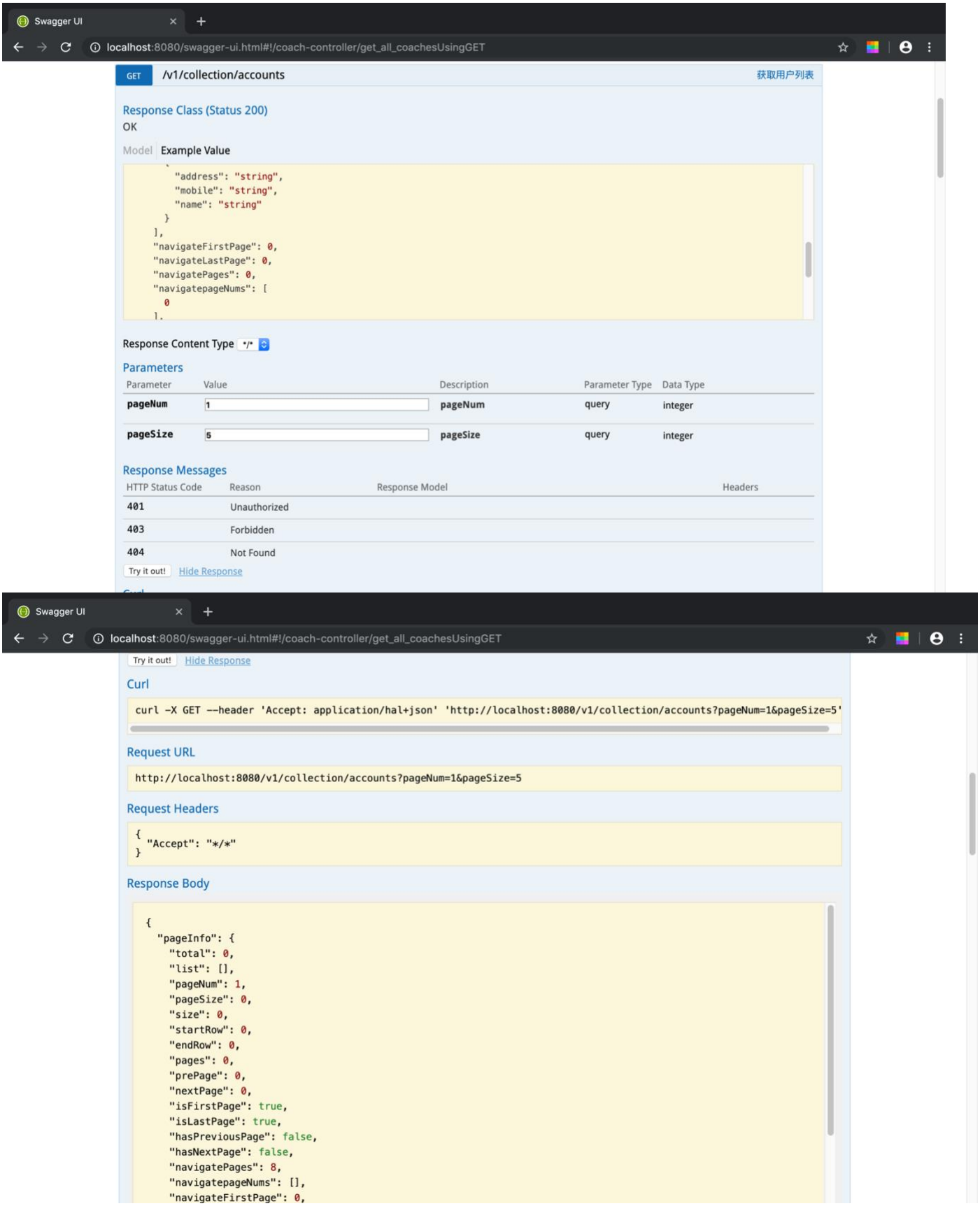
[BASE URL: / , API VERSION: 1.0]

2.1 Restful API 获取所有用户列表

Method: GET

2.1.1 Request URL: /v1/collection/accounts

Demo:



2.2 Restful API 添加一个用户

Method: POST

Request URL:

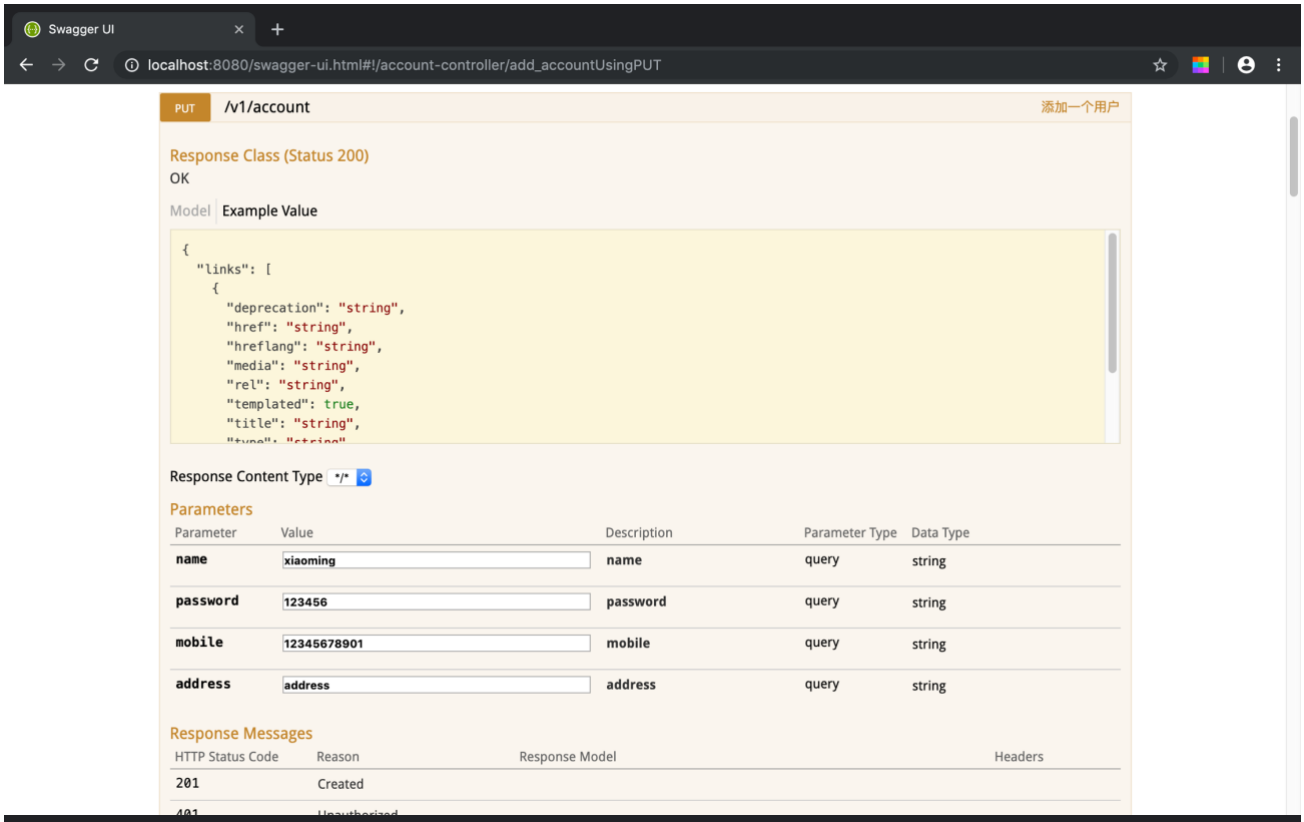
http://localhost:8080/v1/account?name=xiaoming&password=123456&mobile=12345678901&address=address

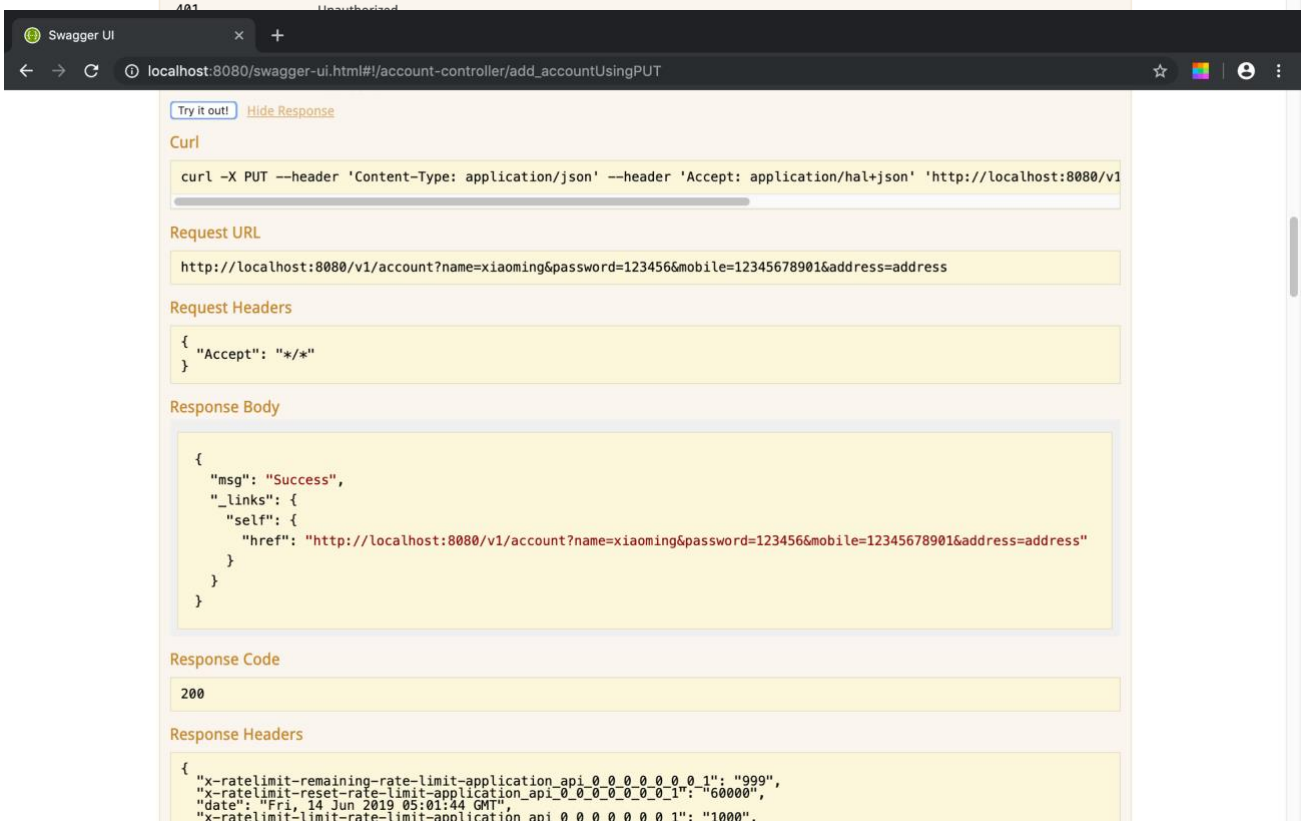
Params:

name

password

Demo:





2.3 Restful API 通过用户名删除一个用户

Method: POST

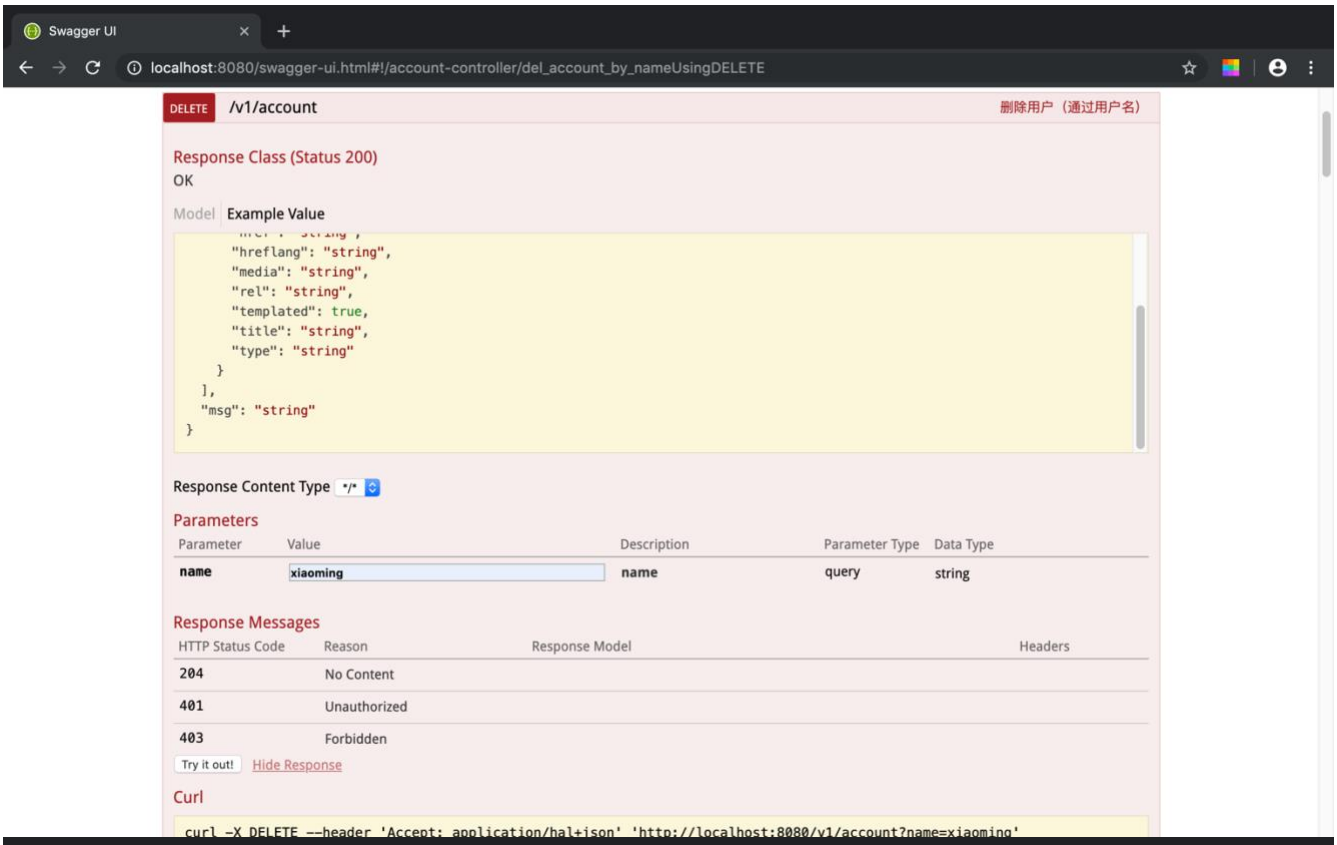
Request URL:

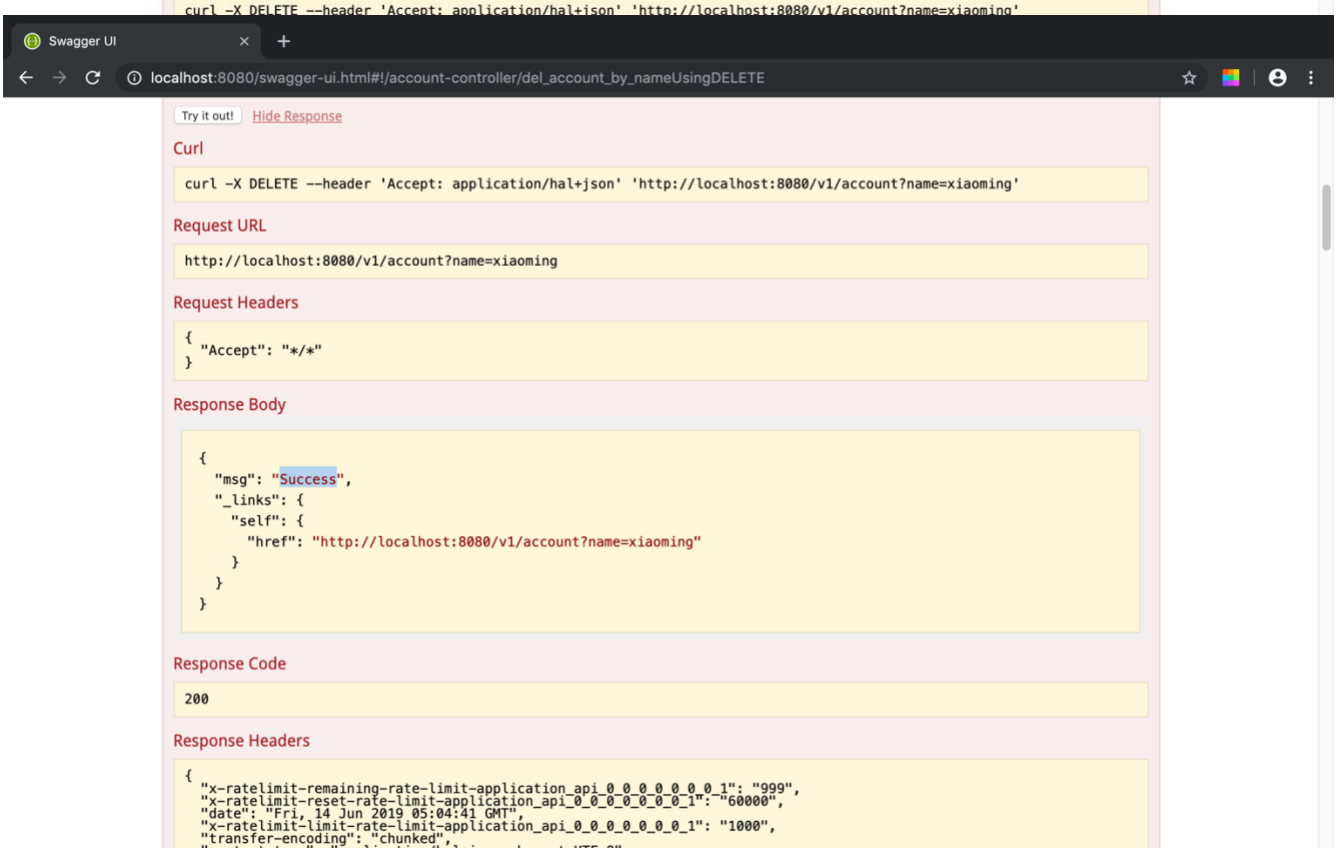
http://localhost:8080/v1/account?name=xiaoming

Params:

name

Demo:





2.4 Restful API 添加一个教练

Method: POST

Request URL:

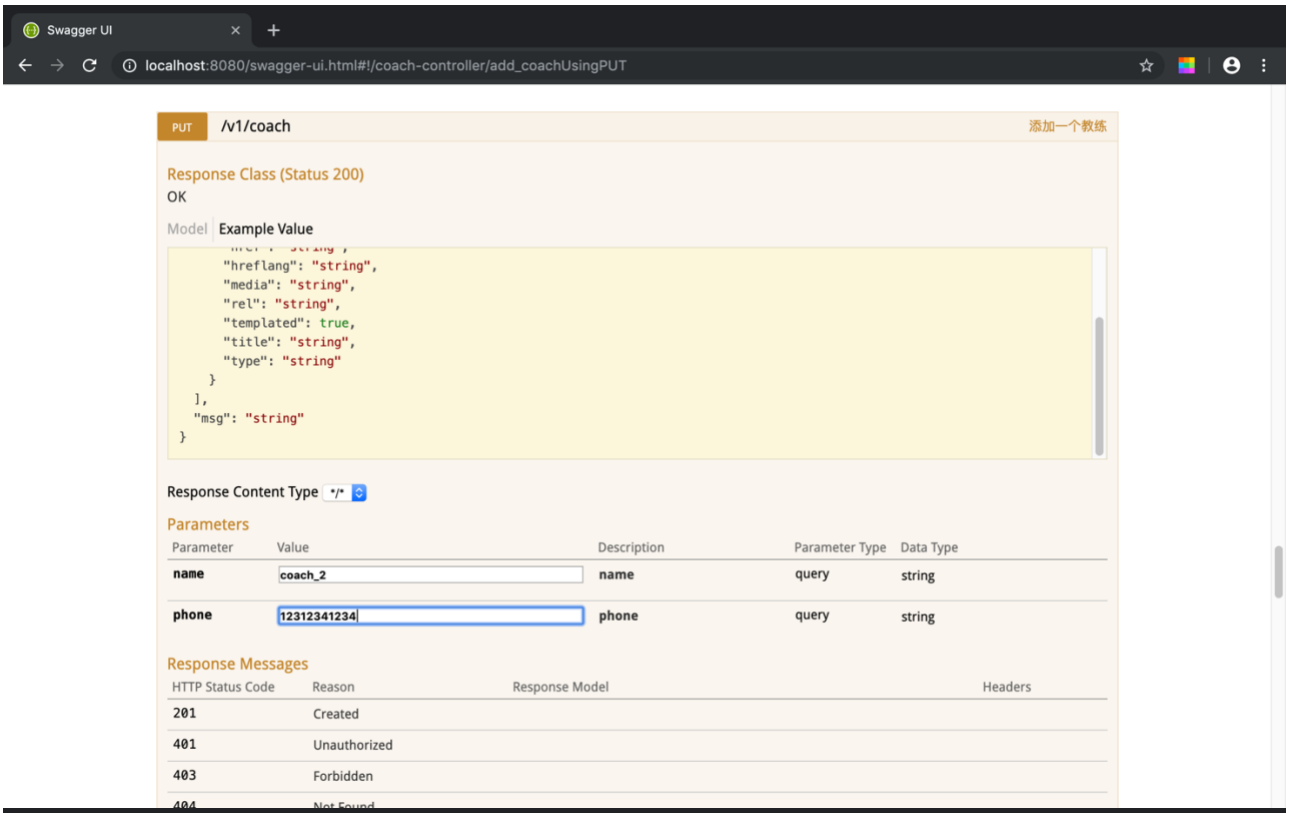
http://localhost:8080/v1/coach?name=coach_2&phone=12312341234

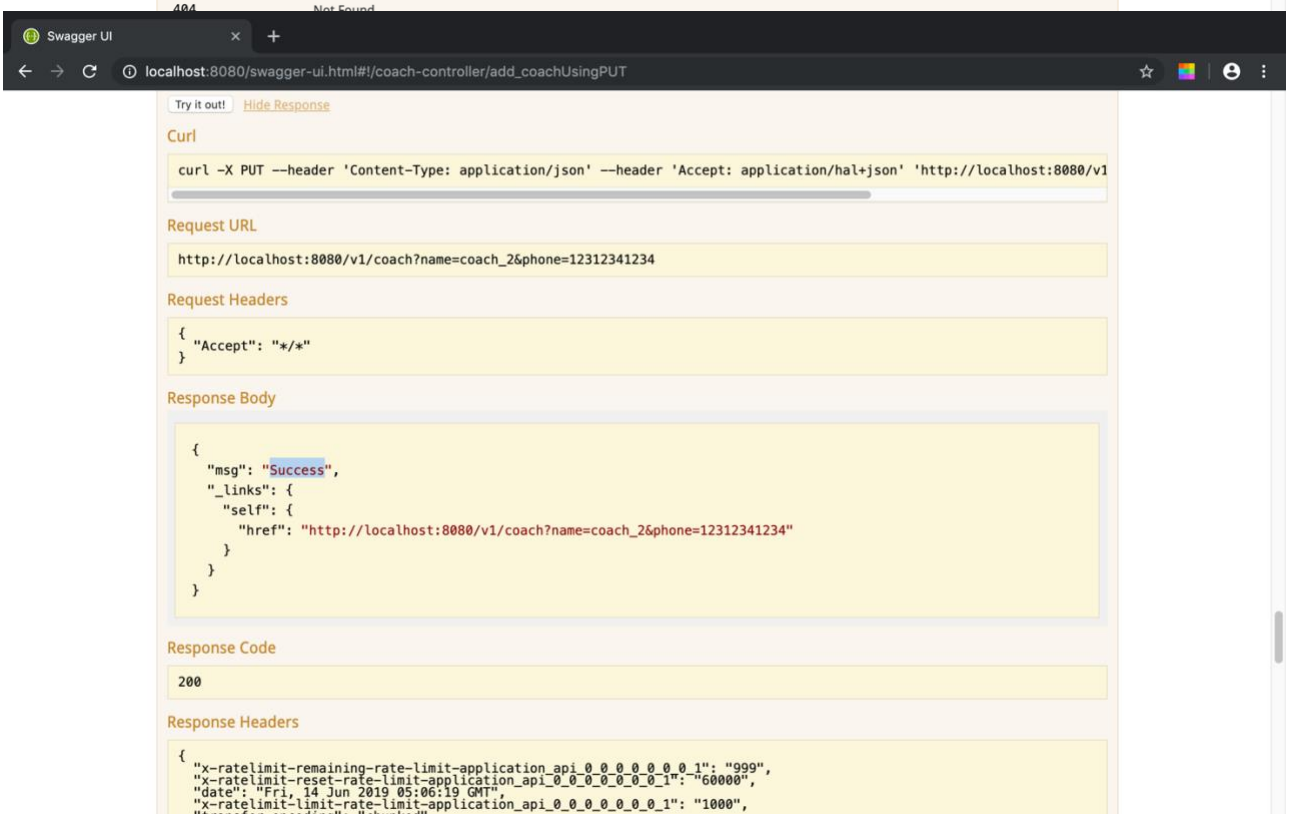
Params:

name

phone

Demo:





2.5 Restful API 获取所有教练列表

Method: GET

Request URL:

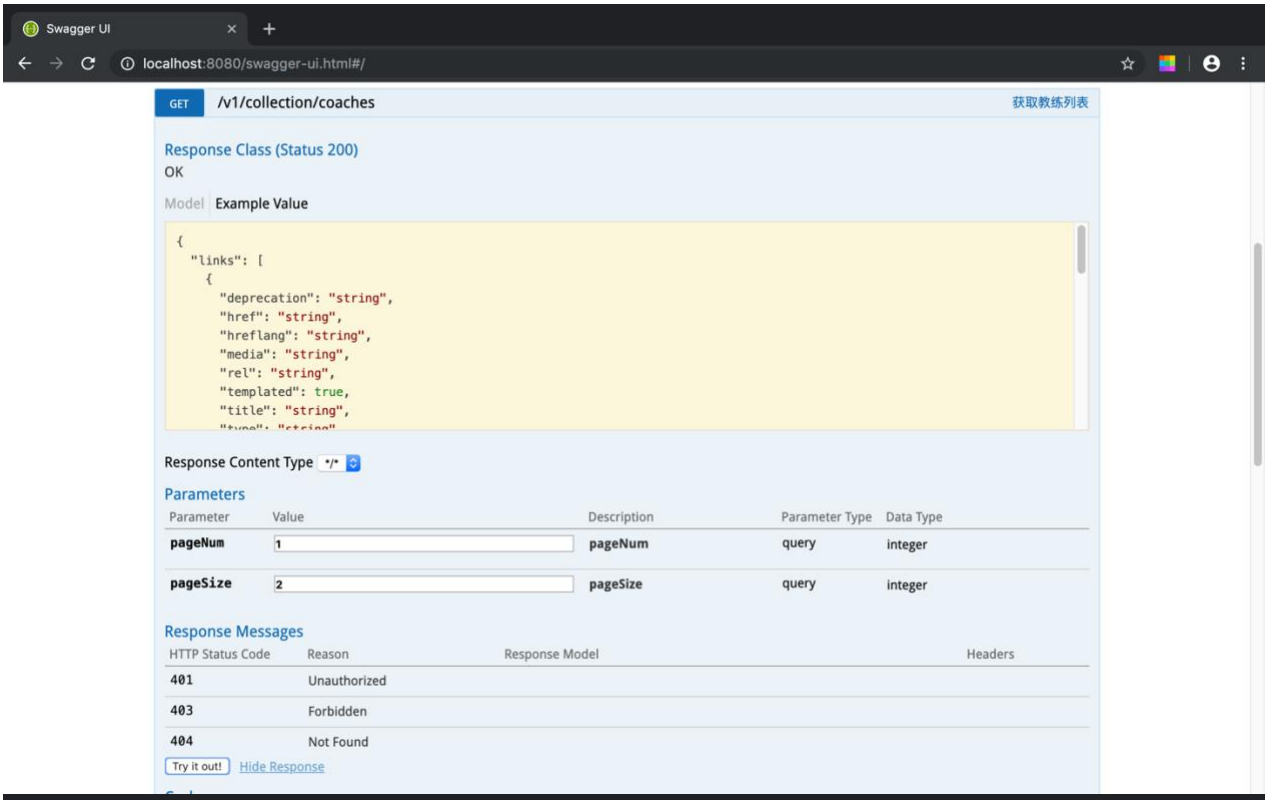
http://localhost:8080/v1/collection/coaches?pageNum=1&pageSize=2

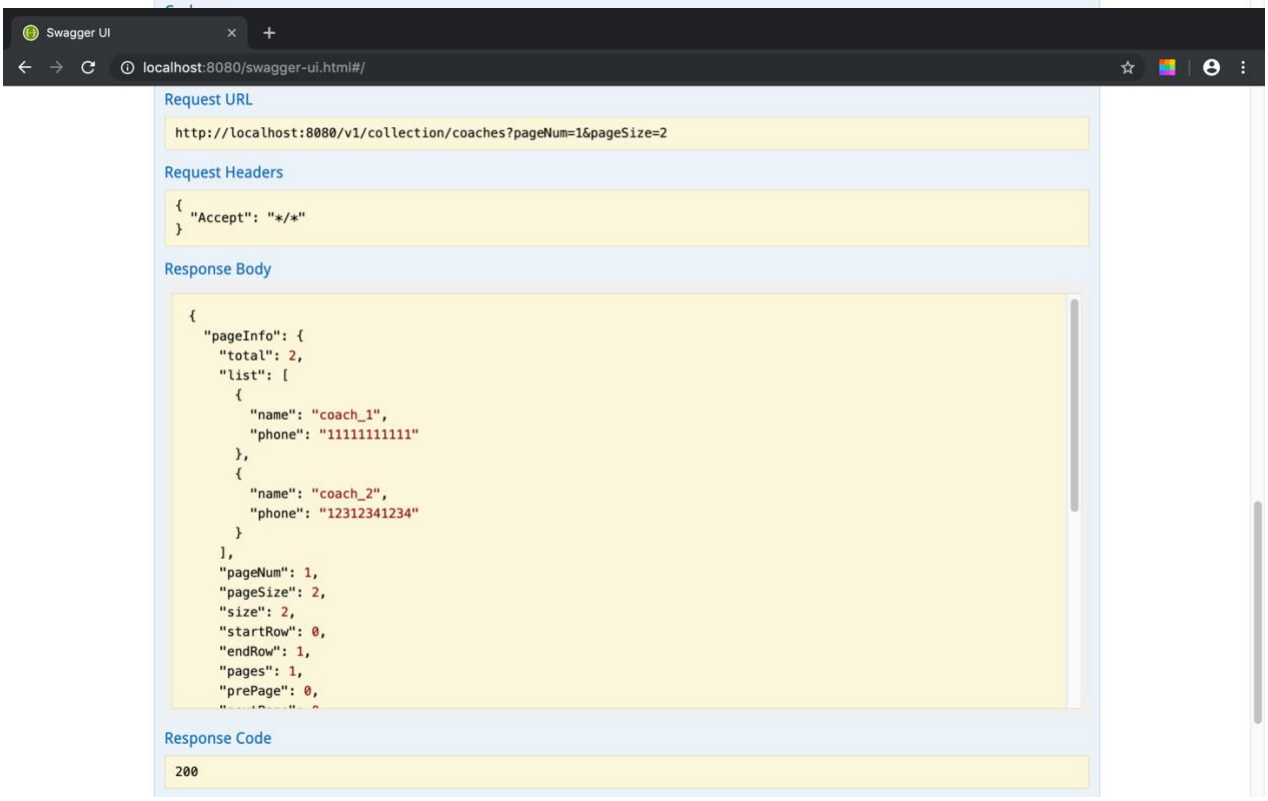
Params:

name

phone

Demo:





2.6 Restful API 通过电话删除一个教练

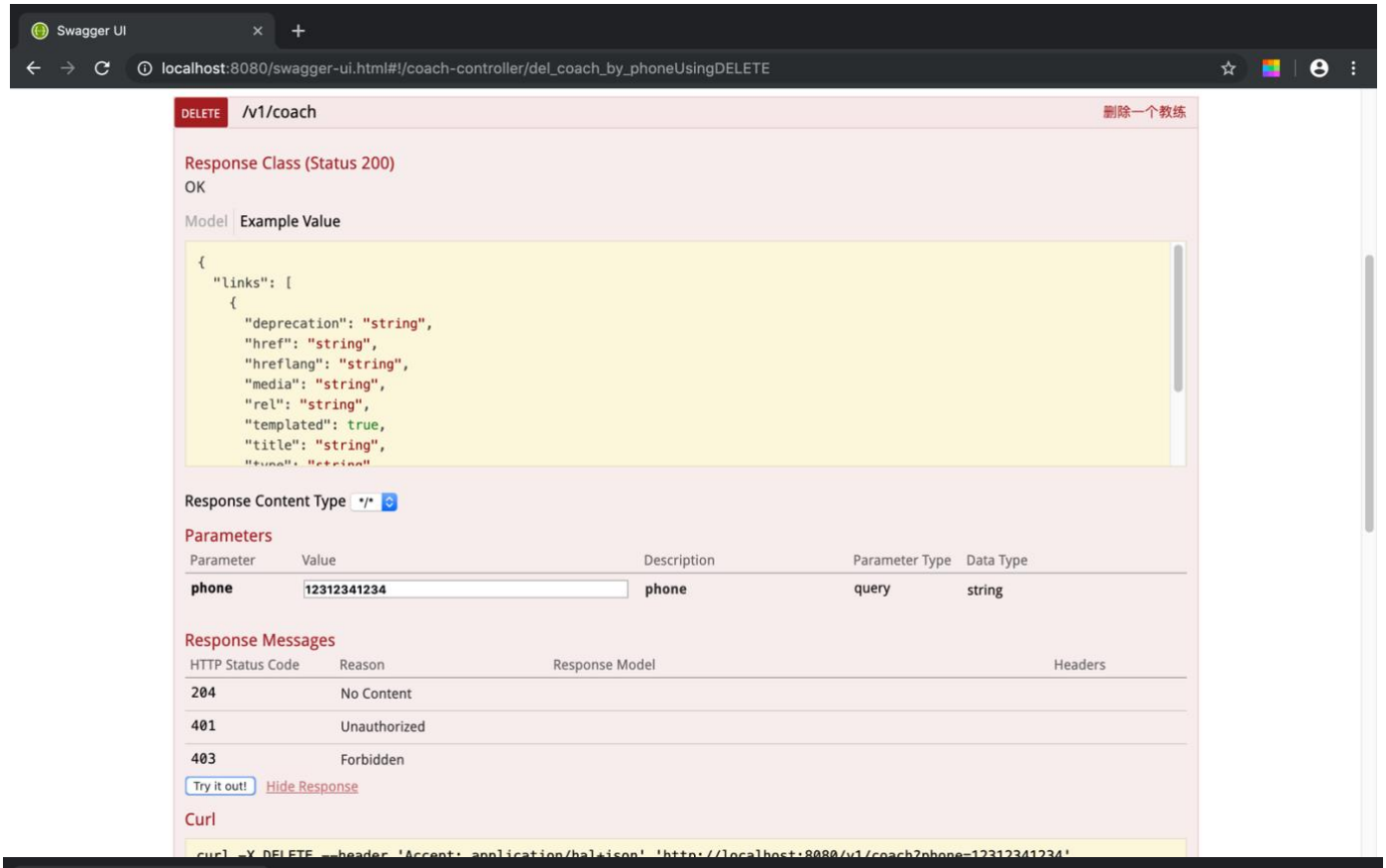
Method: POST

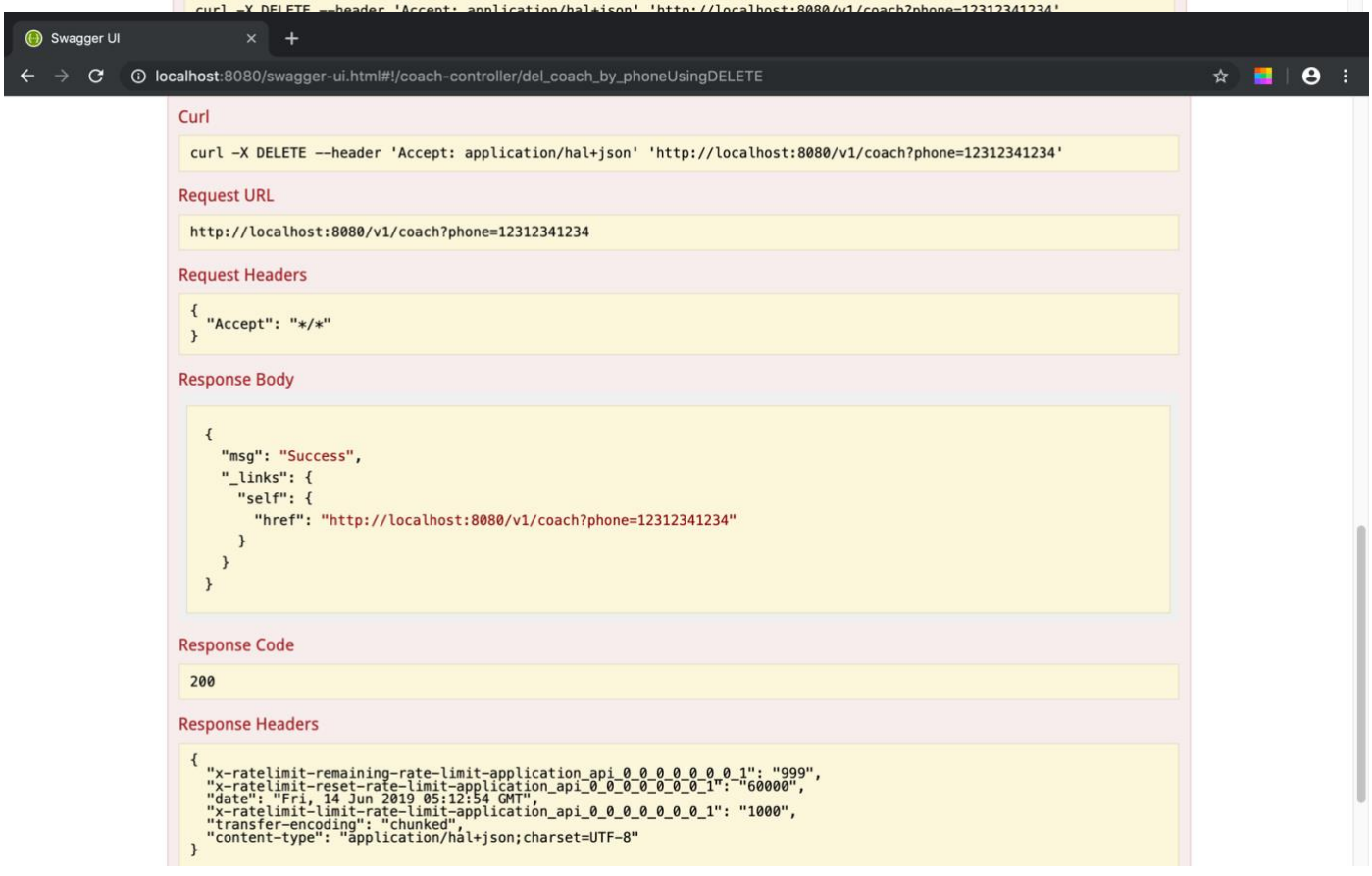
Request URL: http://localhost:8080/v1/del_trainer_by_phone

Params:

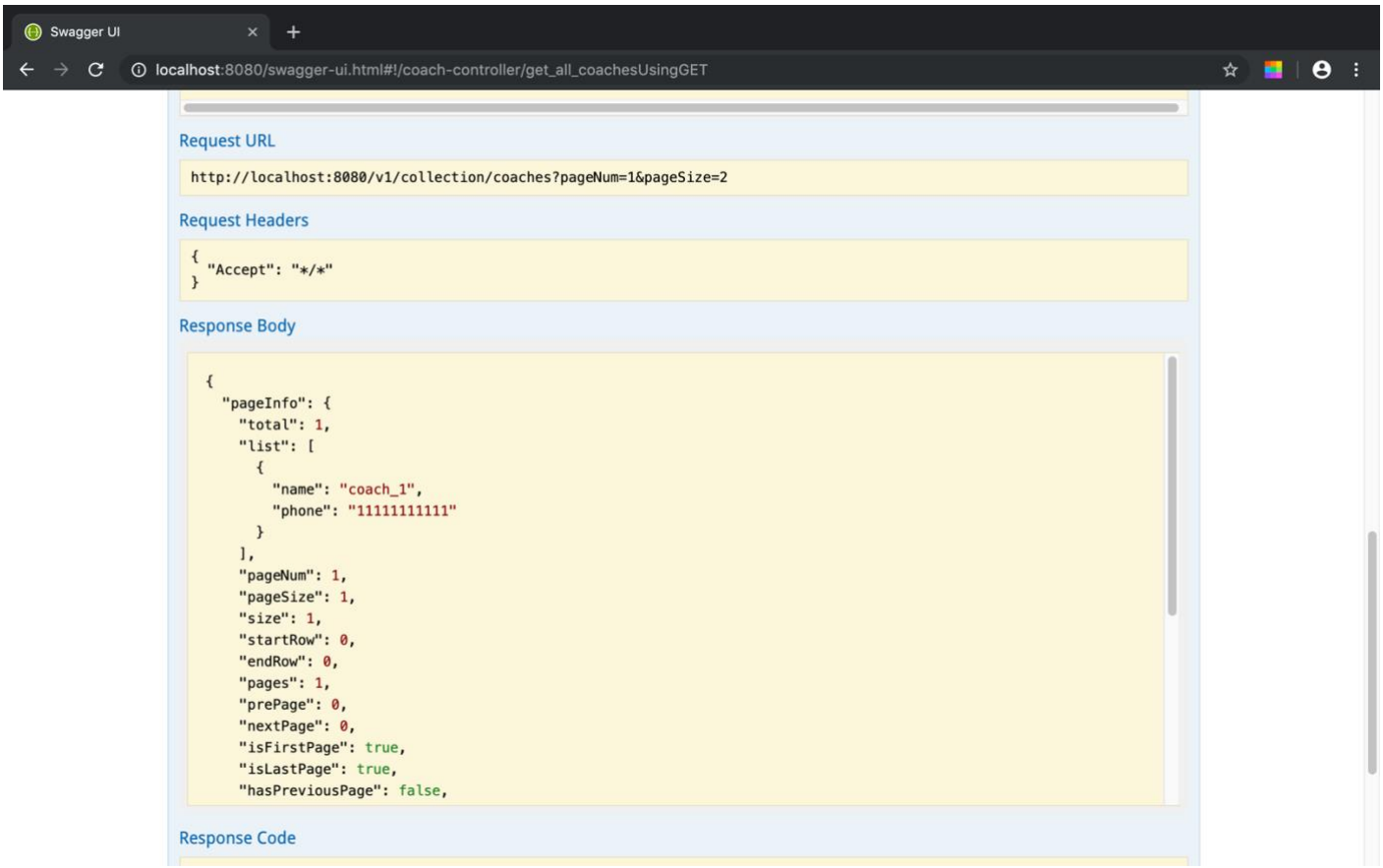
phone

Demo:

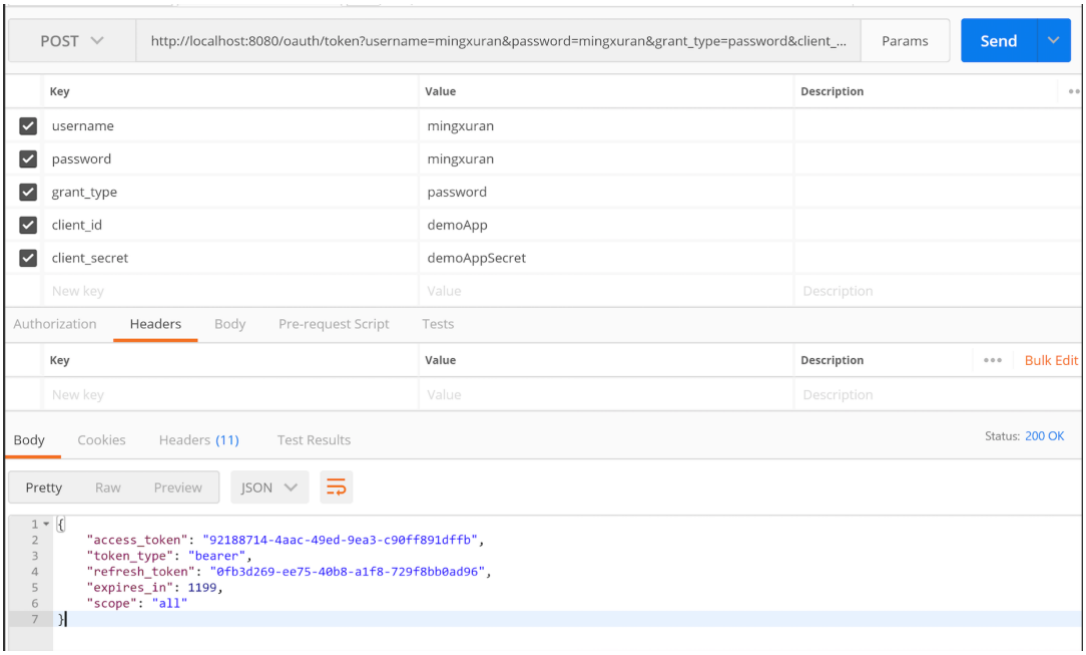




此时重新获取全部的教练列表，可以看到删除成功“12312341234”已经不见了。



3 Oauth2


POST
http://localhost:8080/oauth/token?username=mingxuran&password=mingxuran&grant_type=password&client_...
Params
Send
Key
Value
Description
username
mingxuran
password
mingxuran
grant_type
password
client_id
demoApp
client_secret
demoAppSecret
New key
Value
Description
Authorization
Headers
Body
Pre-request Script
Tests
Key
Value
Description
Bulk Edit
New key
Value
Description
Body
Cookies
Headers (11)
Test Results
Status: 200 OK
Pretty
Raw
Preview
JSON
{
"access_token": "92188714-4aac-49ed-9ea3-c90ff891dffb",
"token_type": "bearer",
"refresh_token": "0fb3d269-ee75-40b8-a1f8-729f8bb0ad96",
"expires_in": 1199,
"scope": "all"
}

4 RateLimit

Framework:

spring-cloud-zuul-ratelimit

Enable Zuul Proxy

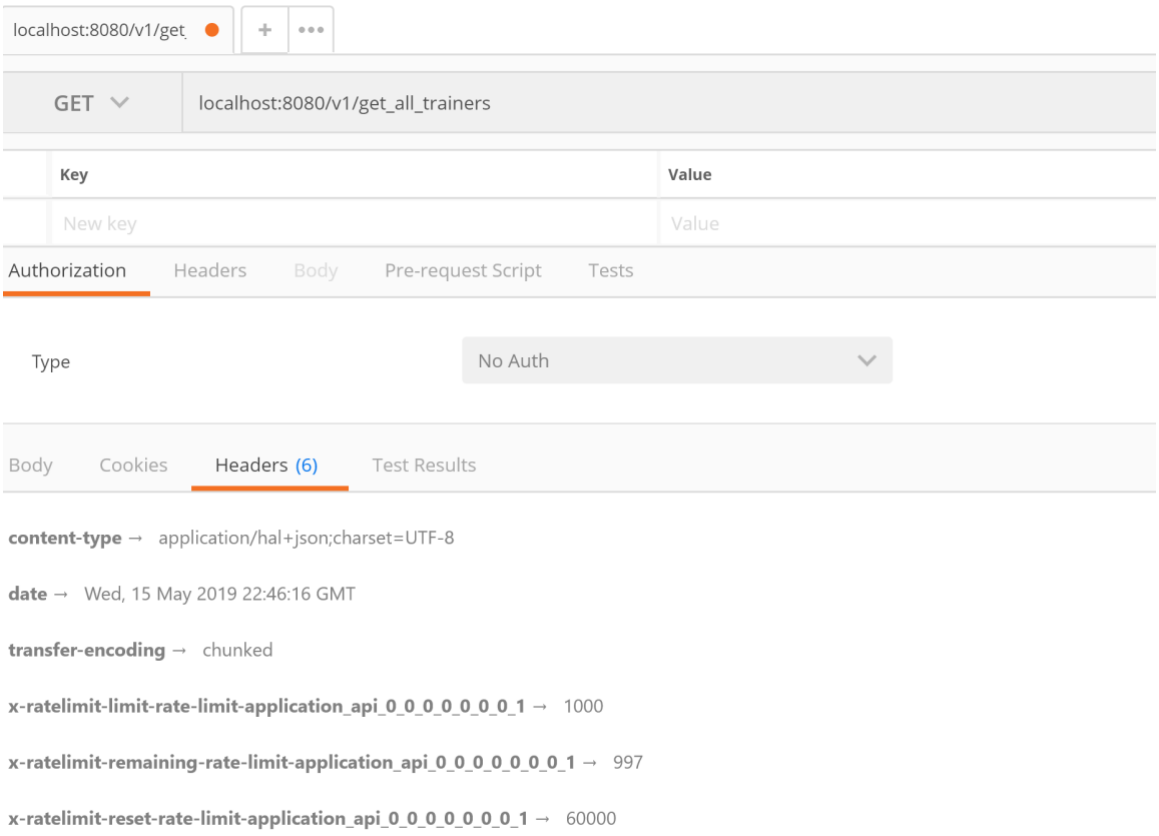
```java
package cn.edu.bjtu.ming;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cache.annotation.EnableCaching;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
@EnableCaching
public class Application {

    public static void main(String[] args) { SpringApplication.run(Application.class, args); }
}
```

Config Zuul in application.yml

Application.java | application.yml | gs-accessing-data-rest | Trainer.java | TrainerRepository.java | UserControll

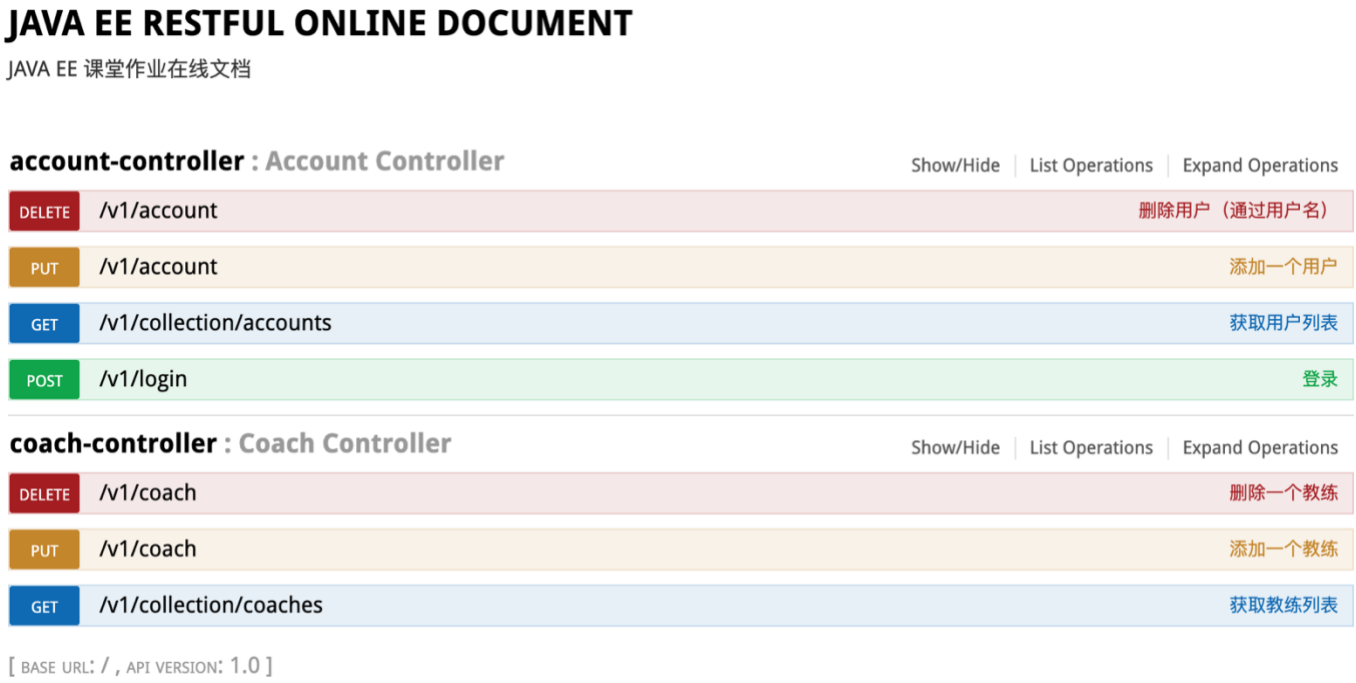
```yaml
zuul:
  ratelimit:
    enabled: true
    repository: redis
    default-policy-list: #optional - will apply unless specific policy exists
      - limit: 1 #optional - request number limit per refresh interval window
        refresh-interval: 60 #default value (in seconds)
        type: #optional
          - origin
    policy-list:
      api:
        - limit: 1000
          refresh-interval: 60
          type:
            - origin
  strip-prefix: true
  routes:
    api:
      path: /**
      url: forward:/
```

From the headers we can say rate was limited


localhost:8080/v1/get
GET
localhost:8080/v1/get_all_trainers
Key
Value
New key
Value
Authorization
Headers
Body
Pre-request Script
Tests
Type
No Auth
Body
Cookies
Headers (6)
Test Results
content-type → application/hal+json;charset=UTF-8
date → Wed, 15 May 2019 22:46:16 GMT
transfer-encoding → chunked
x-ratelimit-limit-rate-limit-application_api_0_0_0_0_0_0_1 → 1000
x-ratelimit-remaining-rate-limit-application_api_0_0_0_0_0_0_1 → 997
x-ratelimit-reset-rate-limit-application_api_0_0_0_0_0_0_1 → 60000

5 Others

5.1 Swagger

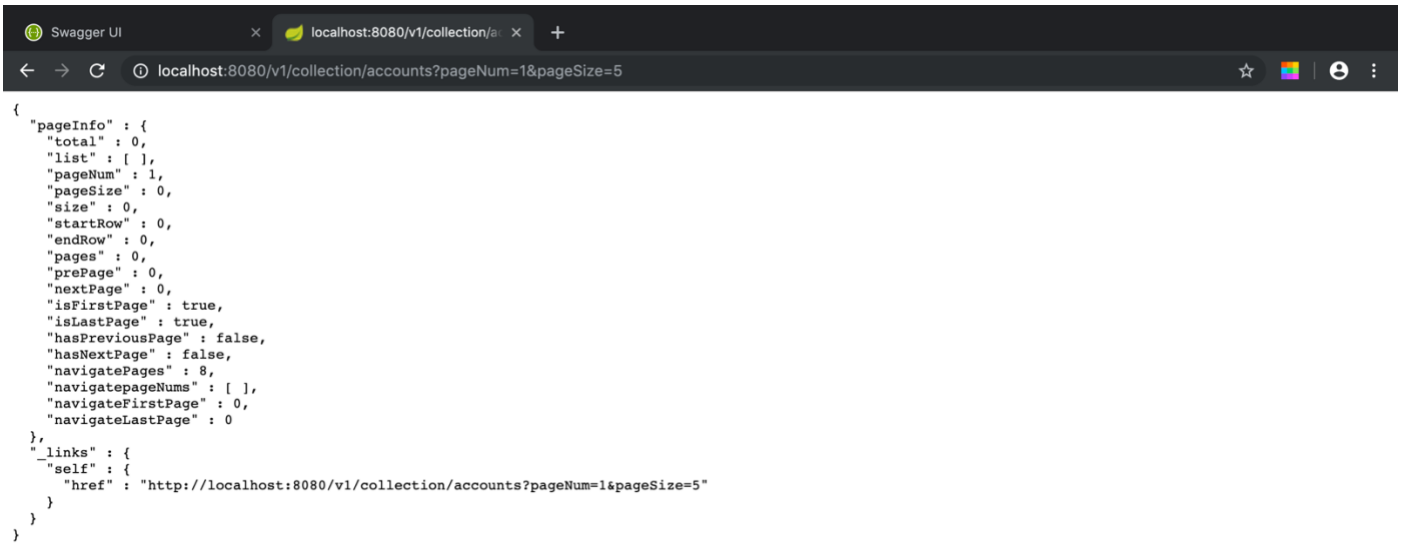


5.2 API pagination

Framework:

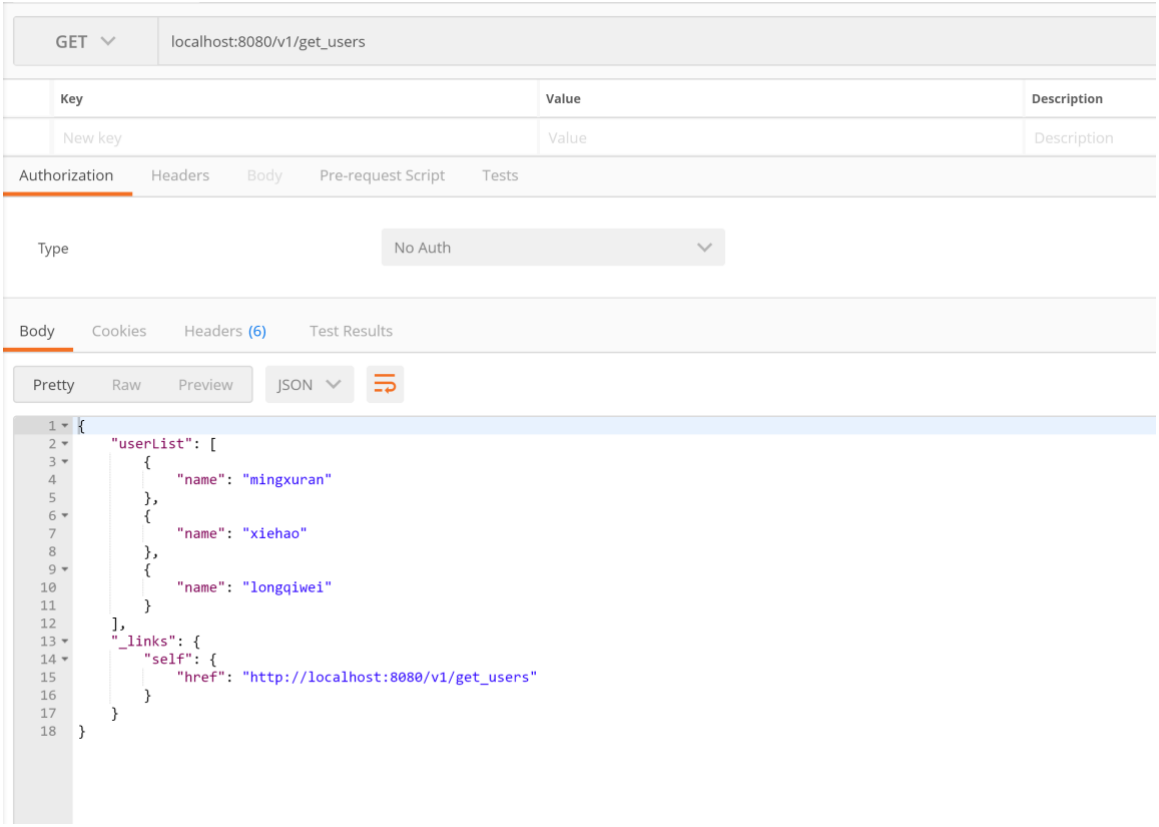
PageHelper

Demo:



5.3 Cache

We enabled Cache and cache GET methods in Dao

```java
package cn.edu.bjtu.ming.dao;

import cn.edu.bjtu.ming.domain.User;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.data.repository.CrudRepository;

import javax.transaction.Transactional;
import java.util.List;

public interface UserRepository extends CrudRepository<User, Integer> {

    @Transactional
    @Cacheable("test")
    List<User> findByNameAndPassword(String name, String password);
```

5.4 Hateoas