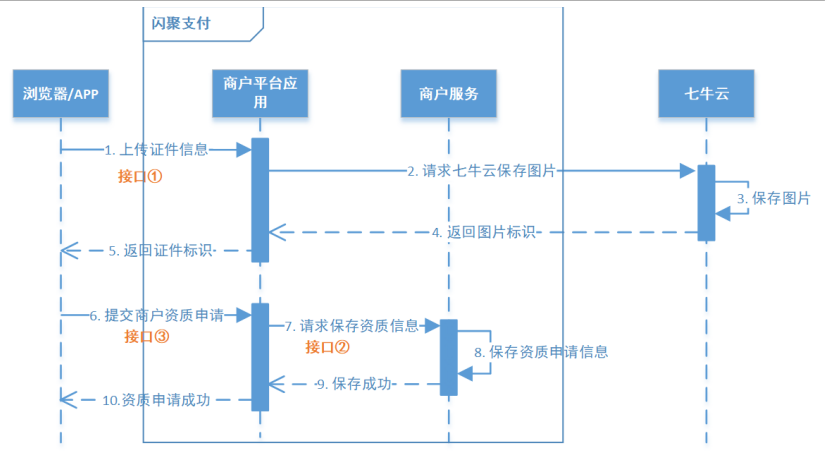
## 1.需求分析

### 1.1.系统交互流程

商户资质申请交互流程如下：



七牛云：

七牛云是国内一家云服务提供商，本系统与七牛云对接将商户资质照片存储在七牛云。 交互流程如下：

1. 前端上传证件照片，请求商户平台应用。

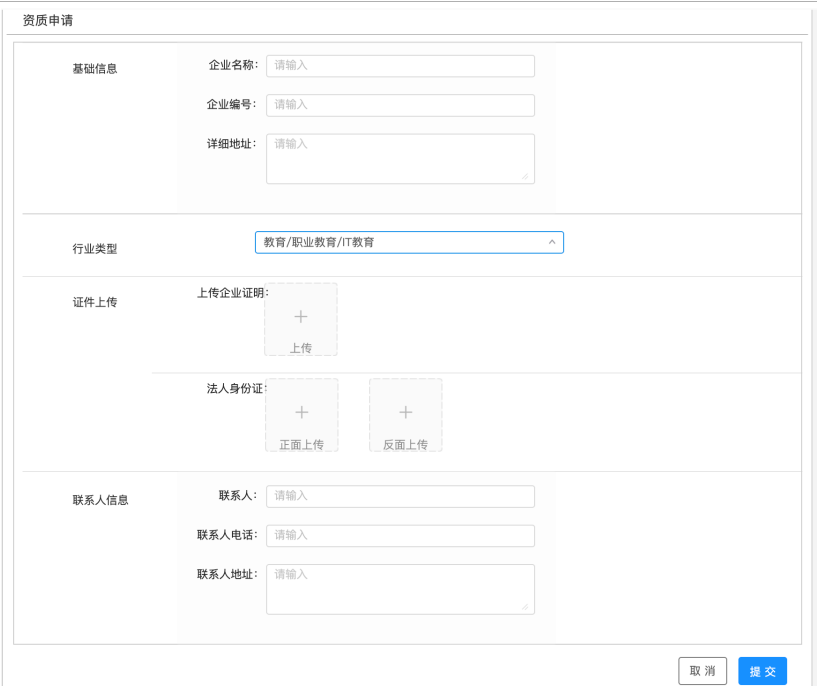
2. 商户平台应用请求七牛云上传图片。

3. 上传成功返回图片标识给前端。

4. 前端携带证件图片标识和资质申请信息提交到商户平台应用。

5. 请求商户服务保存资质申请。

6. 保存成功返回给前端。 商户资质申请界面如下：



### 1.2.资质信息存储

商户资质 信息存储在商户表，上传的资质证件照片存储Url绝对路径。

关于资质 申请状态 说明如下 ：

1、提交资质申请，审核状态 为1（已申请待审核）

2、资质审核后，审核状态 为2（审核通过 ）或3（审核不通过 ）。



## 2.七牛云

商户上传证件照片要与七牛云对接。

参照：[3.七牛云对象存储使用指南.docx](../00234002%20宜亩网（支付项目）2-各种手册/3.七牛云对象存储使用指南.docx)

## 3.上传证件

商户平台应用证件上传(接口①):

### 3.1.接口定义

1、接口描述

1）前端携带证件信息请求商户平台应用

2）商户平台应用请求七牛云服务上传证件图片

3）七牛云返回图片地址给前端

2、接口定义如下:

定义FileService负责文件上传至七牛云：

|  |
| --- |
| **package** com.me.yimu.pay.merchant.service;  **import** java.sql.BatchUpdateException;  /\*\*  \* 文件服务  \* **@author** PC  \*  \*/  **public** **interface** FileService {  /\*\*  \* 上传文件  \* **@param** bytes 文件字节  \* **@param** fileName 文件名称  \* **@return** 文件下载路径  \* **@throws** BatchUpdateException  \*/  **public** String upload(**byte**[] bytes, String fileName) **throws** BatchUpdateException;    } |

3）在MerchantController定义upload负责接收前端上传证件的请求。

|  |
| --- |
| **@ApiOperation("证件上传")**  **@PostMapping("/upload")**  **public String upload(@ApiParam(value = "上传的文件", required = true)**  **MultipartFile file) {**  // 上传文件名称  String originalFilename = file.getOriginalFilename();    // 文件后缀  String suffix = originalFilename.substring(originalFilename.lastIndexOf(".") - 1);    // 文件名称  String fileName = UUID.*randomUUID*().toString() + suffix;    // 上传文件，返回文件名  String upedfileName = **null**;  **try** {  upedfileName = fileService.upload(file.getBytes(), fileName);  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** BusinessException(CommonErrorCode.***E\_100106***);  }    **return** upedfileName;  **}** |

### 3.2.接口实现

使用提供的七牛云工具类完成图片上传到七牛云：

1.在yimupay-common工程引入

|  |
| --- |
| <!-- 七牛云上传文件利用 -->  <dependency>  <groupId>com.qiniu</groupId>  <artifactId>qiniu-java-sdk</artifactId>  </dependency> |

2.编写工具类

在common工程中编写七牛云上传工具类，此工具类被FileService调用。

|  |
| --- |
| /\*\*七牛云测试工具类  \* **@author** Administrator  \* **@version** 1.0  \*\*/  **public** **class** QiniuUtils {  **private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(QiniuUtils.**class**);  /\*\*  \* 文件上传的工具方法  \* **@param** accessKey  \* **@param** secretKey  \* **@param** bucket  \* **@param** bytes  \* **@param** fileName 外部传进来，七牛云上的文件名称和此保持一致  \*/  **public** **static** **void** upload2qiniu(String accessKey,String secretKey,String bucket, **byte**[] bytes,String fileName) **throws** RuntimeException{  //构造一个带指定 Region 对象的配置类，指定存储区域，和存储空间选择的区域一致  Configuration cfg = **new** Configuration(Region.*huanan*());  //...其他参数参考类注释  UploadManager uploadManager = **new** UploadManager(cfg);  //默认不指定key的情况下，以文件内容的hash值作为文件名  String key = fileName;  **try** {  //认证  Auth auth = Auth.*create*(accessKey, secretKey);  //认证通过后得到token（令牌）  String upToken = auth.uploadToken(bucket);  **try** {  //上传文件,参数：字节数组，key，token令牌  //key: 建议我们自已生成一个不重复的名称  Response response = uploadManager.put(bytes, key, upToken);  //解析上传成功的结果  DefaultPutRet putRet = **new** Gson().fromJson(response.bodyString(), DefaultPutRet.**class**);  System.***out***.println(putRet.key);  System.***out***.println(putRet.hash);  } **catch** (QiniuException ex) {  Response r = ex.response;  System.***err***.println(r.toString());  ***LOGGER***.error("上传文件到七牛：{}",ex.getMessage());  **try** {  ***LOGGER***.error(r.bodyString());  } **catch** (QiniuException ex2) {  //ignore  }  **throw** **new** RuntimeException(r.bodyString());  }  } **catch** (Exception ex) {  ***LOGGER***.error("上传文件到七牛：{}",ex.getMessage());  **throw** **new** RuntimeException(ex.getMessage());  }  } |

3. FileServiceImpl

merchant-application.yaml

在nacos配置七牛云上传参数：

|  |
| --- |
| # 覆盖访问路径  server:  servlet:  context‐path: /merchant  #启用Swagger  swagger:  enable: true  spring:  profiles:  active: dev  sms:  url: "http://localhost:56085/sailing"  effectiveTime: 600  ## 七牛云配置  oss:  qiniu:  url: ""  accessKey: lDZ4GHa3wN73xBboENGdCOkVLtWnVy2c7Qx7O7gY  secretKey: Bs7XOewJFNc923bdSZ4EY4WkS8j073X3dZKwHAHx  bucket: yimu-pay |

FileServiceImpl上传证件实现方法如下：

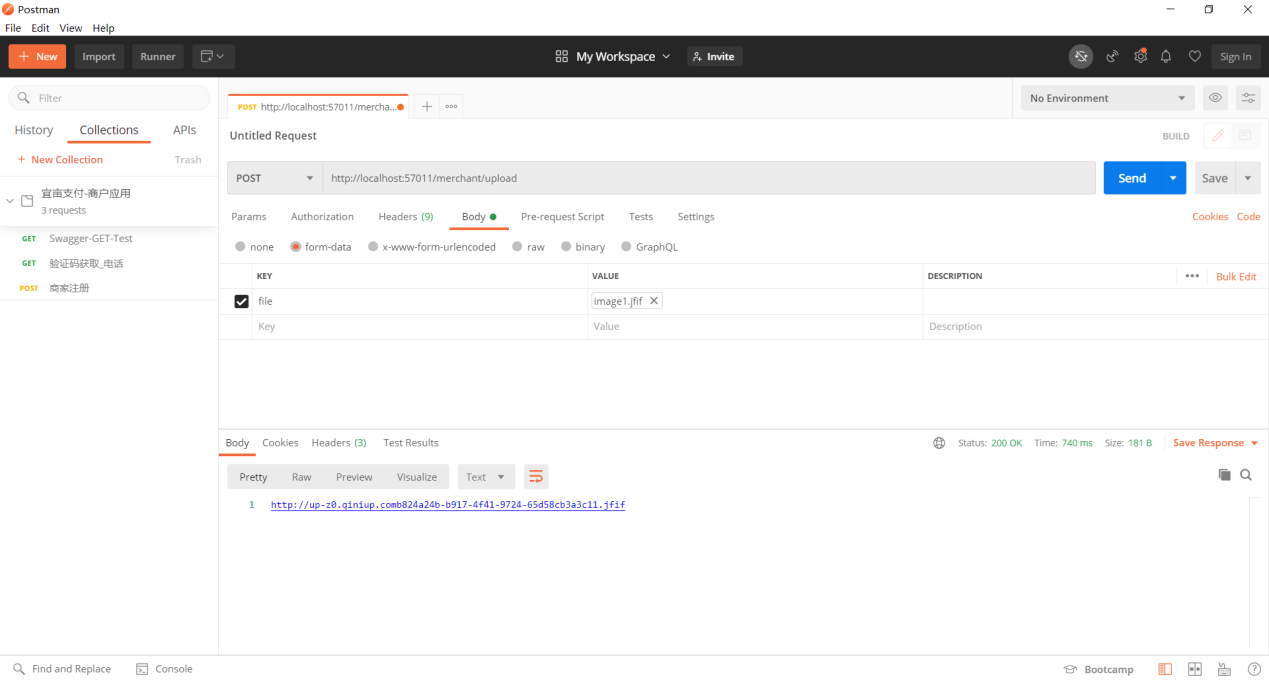
|  |
| --- |
| **package** com.me.yimu.pay.merchant.service.impl;  **import** java.sql.BatchUpdateException;  **import** org.springframework.beans.factory.annotation.Value;  **import** org.springframework.stereotype.Service;  **import** com.me.yimu.pay.domain.BusinessException;  **import** com.me.yimu.pay.domain.CommonErrorCode;  **import** com.me.yimu.pay.merchant.service.FileService;  **import** com.me.yimu.pay.utils.QiniuUtils;  **import** lombok.extern.slf4j.Slf4j;  /\*\*  \* 文件处理服务类  \* **@author** PC  \*  \*/  @Slf4j  @Service  **public** **class** FileServiceImpl **implements** FileService {  // 配置参数  @Value("${oss.qiniu.url}")  **private** String qiniuUrl;  @Value("${oss.qiniu.accessKey}")  **private** String accessKey;  @Value("${oss.qiniu.secretKey}")  **private** String secretKey;  @Value("${oss.qiniu.bucket}")  **private** String bucket;    @Override  **public** String upload(**byte**[] bytes, String fileName) **throws** BatchUpdateException {  ***log***.info("FileServiceImpl.upload start");    **try** {  QiniuUtils.*upload2qiniu*(accessKey, secretKey, bucket, bytes, fileName);  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** BusinessException(CommonErrorCode.***E\_100106***);  }    String fileUrl = qiniuUrl + fileName;  ***log***.info("FileServiceImpl.upload end. resultParam: fileUrl={0}", fileUrl);  // 返回文件路径  **return** fileUrl;  }  } |

4. MerchantController层处理

|  |
| --- |
| @ApiOperation("证件上传")  @PostMapping("/upload")  **public** String upload(@ApiParam(value = "上传的文件", required = **true**)  MultipartFile file) {  // 上传文件名称  String originalFilename = file.getOriginalFilename();    // 文件后缀  String suffix = originalFilename.substring(originalFilename.lastIndexOf(".") - 1);    // 文件名称  String fileName = UUID.*randomUUID*().toString() + suffix;    // 上传文件，返回文件下载url  String fileUrl = **null**;  **try** {  fileUrl = fileService.upload(file.getBytes(), fileName);  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** BusinessException(CommonErrorCode.***E\_100106***);  }    **return** fileUrl;  } |

### 3.3.接口测试

请求地址：<http://localhost:57011/merchant/upload>



上传成功，登录七牛云查询图片是否上传成功。

## 4.资质申请

### 4.1.商户服务资质申请(接口②)

#### 4.1.1.接口定义

1、接口描述

1）接收资质申请信息，更新商户信息及审核状态（待审核）

2）返回结果

2、接口定义如下：

在 MerchantService中定义applyMerchant接口

|  |
| --- |
| /\*\*  \* 资质申请  \* **@param** merchantId 商户id  \* **@param** merchantDTO 资质申请信息  \* **@throws** BusinessException  \*/  **void** applyMerchant(Long merchantId, MerchantDTO merchantDTO) **throws** BusinessException; |

#### 4.1.2.接口实现

实现MerchantServiceImpl 中的applyMerchant方法：

|  |
| --- |
| /\*\*  \* 商户资质申请  \*/  @Override  **public** **void** applyMerchant(Long merchantId, MerchantDTO merchantDTO) **throws** BusinessException {  // 接收资质申请信息，更新到商户表  **if** (merchantDTO == **null** || merchantId == **null**) {  **throw** **new** BusinessException(CommonErrorCode.***E\_100108***);  }    // 根据id查询商户  Merchant merchant = merchantMapper.selectById(merchantId);  **if** (merchant == **null**) {  **throw** **new** BusinessException(CommonErrorCode.***E\_200002***);  }    Merchant merchantUp = MerchantConvert.***INSTANCE***.dto2entity(merchantDTO);  // 已申请待审核  merchantUp.setAuditStatus("1");  merchantUp.setTenantId(merchant.getTenantId());    // 更新  merchantMapper.updateById(merchantUp);  } |

### 4.2.商户平台应用资质申请(接口③)

#### 4.2.1.接口定义

1.接口描述

1）商户登录宜亩支付平台

2）商户上传证件，填写资质信息



3）请求商户平台应用进行资质申请

4）商户平台应用请求商户服务完成资质申请

5）返回结果

2.接口定义

根据原型编写商户资质申请VO：MerchantDetailVO

|  |
| --- |
| **package** com.me.yimu.pay.merchant.vo;  **import** java.io.Serializable;  **import** io.swagger.annotations.ApiModel;  **import** io.swagger.annotations.ApiModelProperty;  **import** lombok.Data;  /\*\*  \* 商户资质申请VO  \* **@author** PC  \*  \*/  @ApiModel(value = "MerchantDetailVO", description = "商户资质申请信息")  @Data  **public** **class** MerchantDetailVO **implements** Serializable {    **private** **static** **final** **long** ***serialVersionUID*** = 1L;  @ApiModelProperty("企业名称")  **private** String merchantName;    @ApiModelProperty("企业编号")  **private** String merchantNo;    @ApiModelProperty("企业地址")  **private** String merchantAddress;    @ApiModelProperty("行业类型")  **private** String merchantType;    @ApiModelProperty("营业执照")  **private** String businessLicensesImg;    @ApiModelProperty("法人身份证正面")  **private** String idCardFrontImg;    @ApiModelProperty("法人身份证反面")  **private** String idCardAfterImg;    @ApiModelProperty("联系人")  **private** String username;    @ApiModelProperty("联系人地址")  **private** String contactsAddress;  } |

在MerchantController中定义saveMerchant

|  |
| --- |
| @ApiOperation("商户资质申请")  @ApiImplicitParams({  @ApiImplicitParam(name = "merchantInfo", value = "商户认证资料", required = **true**,  dataType = "MerchantDetailVO", paramType = "body")  })  @PostMapping("/my/merchants/save")  **public** **void** saveMerchant(@RequestBody MerchantDetailVO merchantInfo) {  } |

#### 4.2.2.获取商户身份

1.商户登录临时方案

因前期未实现登录功能，故目前手动指定的商户ID生成Token（用户登录后的身份令牌），将Token配置在前端，前端拥有了token则说明该商户Id对应的商户登录成功。

商户登录及身份解析流程如下：

1）前端携带token访问商户平台应用。

2）商户平台应用解析token取出商户id

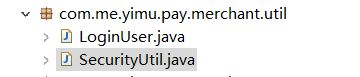
2.生成token

代码如下：

|  |
| --- |
| package com.me.yimu.pay.merchant;  import com.alibaba.fastjson.JSON;  import com.alibaba.fastjson.JSONObject;  import com.me.yimu.pay.merchant.api.MerchantService;  import com.me.yimu.pay.merchant.api.dto.MerchantDTO;  import com.me.yimu.pay.utils.EncryptUtil;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.boot.test.context.SpringBootTest;  import org.springframework.test.context.junit4.SpringRunner;  /\*\*  \* @author Administrator  \* @version 1.0  \*\*/  @RunWith(SpringRunner.class)  @SpringBootTest  public class TokenTemp {  @Autowired  MerchantService merchantService;  @Test  public void createTestToken() {  Long merchantId = 1209826678635798530L;//填写用于测试的商户id  MerchantDTO merchantDTO = merchantService.queryMerchantById(merchantId);  JSONObject token = new JSONObject();  token.put("mobile", merchantDTO.getMobile());  token.put("user\_name", merchantDTO.getUsername());  token.put("merchantId", merchantId);  String jwt\_token = "Bearer " + EncryptUtil.encodeBase64(JSON.toJSONString(token).getBytes());  System.out.println(jwt\_token);  }  } |

向merchantId中设置商户id，运行此测试方法。（运行测试之前停止商户平台应用服务）

3.暂时使用工具类从请求中获取Token并解析



#### 4.2.3.资质申请实现

1.编写对象转换类

|  |
| --- |
| package com.me.yimu.pay.merchant.convert;  import org.mapstruct.Mapper;  import org.mapstruct.factory.Mappers;  import com.me.yimu.pay.merchant.api.dto.MerchantDTO;  import com.me.yimu.pay.merchant.vo.MerchantDetailVO;  /\*\*  \* 将商户资质申请vo和dto进行转换  \* Created by Administrator.  \*/  @Mapper  public interface MerchantDetailConvert {  MerchantDetailConvert INSTANCE = Mappers.getMapper(MerchantDetailConvert.class);  //将dto转成vo  MerchantDetailVO dto2vo(MerchantDTO merchantDTO);  //将vo转成dto  MerchantDTO vo2dto(MerchantDetailVO merchantDetailVO);  } |

2.编写MerchantController中的saveMerchant方法

前端携带Token请求此方法，在此方法中需要解析token获取当前商户的Id。

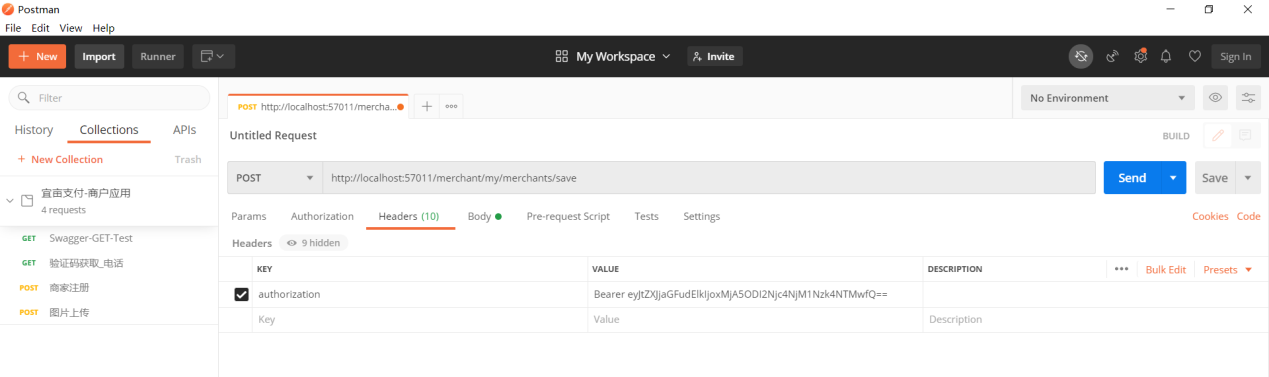
|  |
| --- |
| @ApiOperation("商户资质申请")  @ApiImplicitParams({  @ApiImplicitParam(name = "merchantInfo", value = "商户认证资料", required = **true**,  dataType = "MerchantDetailVO", paramType = "body")  })  @PostMapping("/my/merchants/save")  **public** **void** saveMerchant(@RequestBody MerchantDetailVO merchantInfo) {  // 解析token得到商户id  Long merchantId = SecurityUtil.*getMerchantId*();  MerchantDTO merchantDTO = MerchantDetailConvert.***INSTANCE***.vo2dto(merchantInfo);  //资质申请  merchantService.applyMerchant(merchantId,merchantDTO);  } |

#### 4.2.4.接口测试

1.生成token

运行createTestToken测试方法生成临时token。

在Header中添加 ：key:authorization value：token 例子如下 ：



**注意：token内容前边固定添加 “Bearer ”(后边一个空格 )**

2.上传证件，获取证件标识

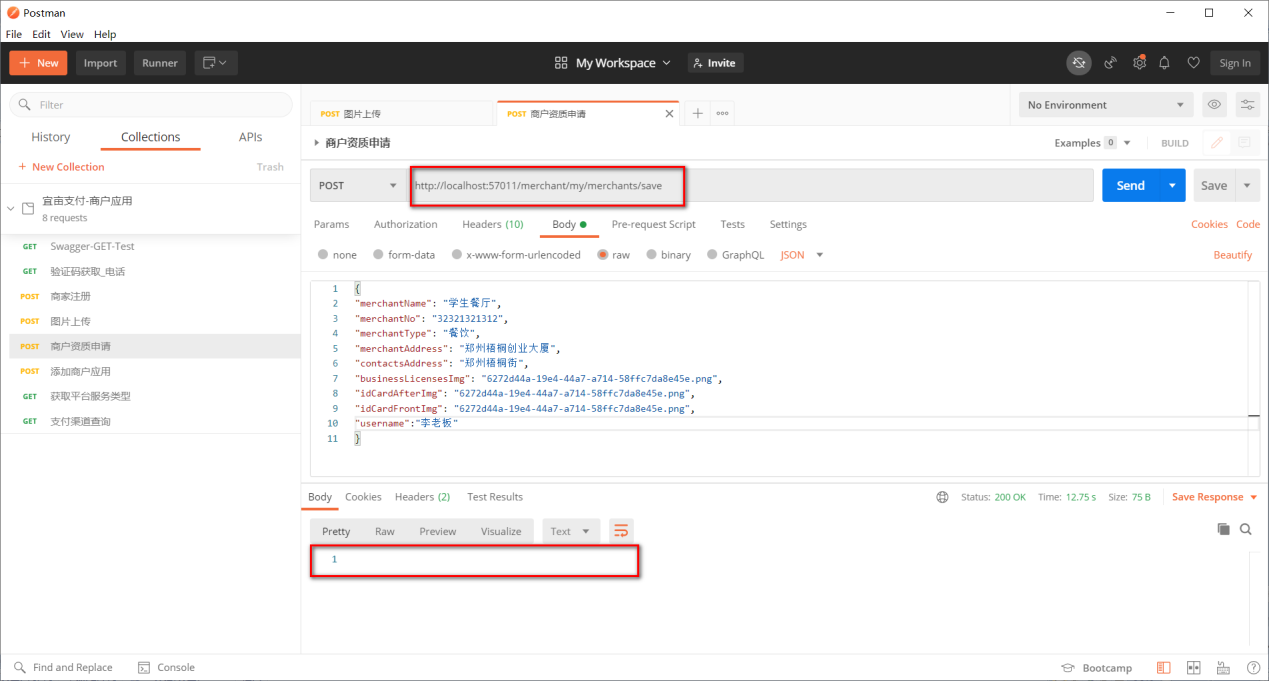
参考证件上传测试。

3.资质申请

请求数据：

|  |
| --- |
| {  "merchantName": "学生餐厅",  "merchantNo": "32321321312",  "merchantType": "餐饮",  "merchantAddress": "郑州梧桐创业大厦",  "contactsAddress": "郑州梧桐街",  "businessLicensesImg": "6272d44a‐19e4‐44a7‐a714‐58ffc7da8e45e.png",  "idCardAfterImg": "6272d44a‐19e4‐44a7‐a714‐58ffc7da8e45e.png",  "idCardFrontImg": "6272d44a‐19e4‐44a7‐a714‐58ffc7da8e45e.png",  "username":"李老板"  } |

4.测试截图



## 5.参照

https://blog.csdn.net/qq\_45952745/article/details/122497054