**“银行业务管理系统”**

系统设计报告

**学号：PB16020870 PB16051553 PB15000256**

**姓名：王博 雷婷 胡清泳**

**计算机科学与技术学院**

**中国科学技术大学**

2019年6月

**目 录**

1 概述 1

1.1 系统目标 1

1.2 需求说明 1

1.3 本报告的主要贡献 1

2 总体设计 1

2.1 系统模块结构 1

2.2 系统工作流程 1

2.3 数据库设计 1

3 详细设计 2

3.1 \*\*\*\* 模块 2

3.2 \*\*\*\* 模块 2

3.3 \*\*\*\* 模块 2

4 实现与测试 2

4.1 实现结果 2

4.2 测试结果 2

5 总结与讨论 2

附录 团队介绍与工作分工 3

# 

# 1 概述

## 1.1 系统目标

- 实现一个完整的数据库管理系统。计划实现前后端分离的系统，系统组成为数据库业务逻辑，Web服务，前端三部分。

## 1.2 需求说明

- 实现银行业务管理系统，需求与Lab2类似，但是不用维护部门信息。

- 支行，员工，客户，账户要提供增删该查功能

- 贷款要实现增，删，查功能；需提供贷款发放功能，贷款状态查询。

- 实现三类完整性约束。

- 实现业务统计功能，对统计数据进行可视化。

## 1.3 本报告的主要贡献

- 给出了一个前后端分离B/S架构的实现（可支持多种用户自定义客户端，容易扩展实现C/S架构）。本报告阐述了该系统的设计思路，实现细节，以及需求如何被满足。

# 2 总体设计

## 2.1 系统模块结构

本系统为前后端分离的B/S架构（可支持多种用户自定义客户端，容易扩展实现C/S架构）。本系统由数据库，业务逻辑服务器，前端Web服务三部分组成。

模块层级和基本功能如下所示：

- **django 后端**

- RESTful API处理模块

- 业务逻辑模块。用于处理业务逻辑请求（增删改查）。

- 身份认证模块。用于处理登录退出，生成Cookie。

- 数据库连接模块。用于维持数据库连接，发送存储过程调用或者查询语句。

- 静态Web资源模块。用于向客户端传送HTML.CSS.JS。

- **HTML/css/js前端**

- 登录模块。负责发送登录请求和检查登录状态。

- 导航模块。负责引导使用者到达不同的功能页面。

- 增删改查业务模块。包括多个子模块，如支行管理，账户管理等。

- 数据统计模块。负责统计整个银行/某个支行的资产和账户数据。

- 账户控制模块。负责呈现登录/退出界面。

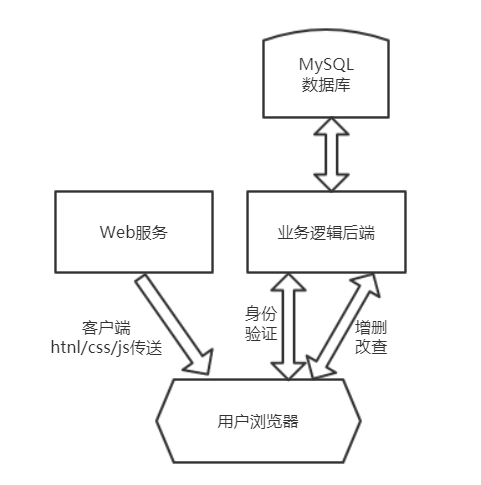
- **MySQL 数据库**

- SQL数据结构创建程序。创建表，定义完整性约束。

- 数据生成器。用于根据数据结构生成初始化测试数据。

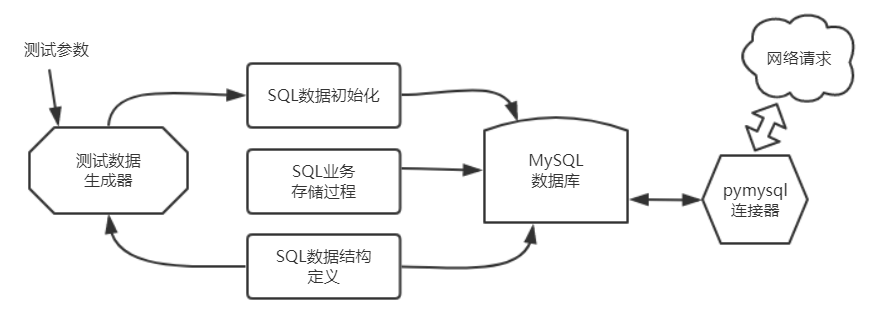
- SQL数据初始化程序。用于将初始化数据存入数据库，并调用存储过程。

- SQL业务逻辑存储过程创建程序。创建复杂业务逻辑所需的存储过程。

顶层模块结构图如下：

顶层模块结构图

本系统的数据库采用MySQL数据库。本系统使用PowerDesigner进行设计，并借助PowerDesigner生成实际代码。数据库的模块结构图如下：

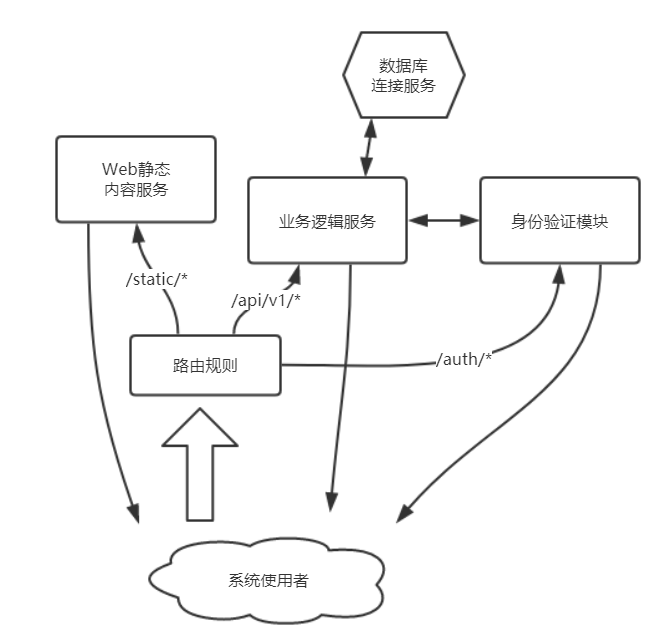


数据库模块结构图

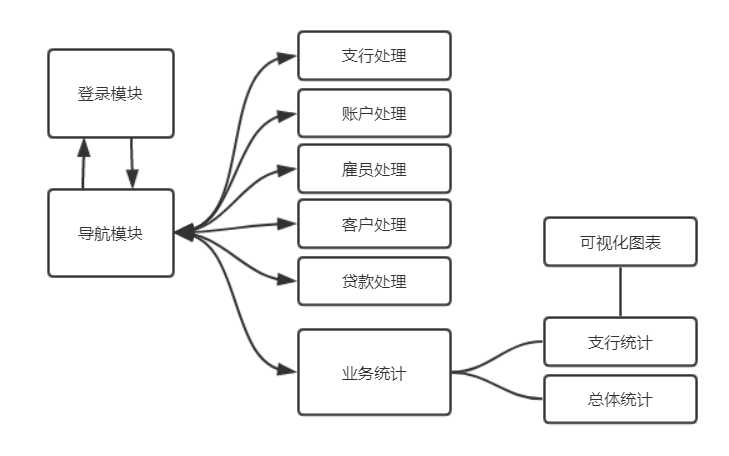
本系统后端有两部分。

- 第一个是业务逻辑服务器，按照RESTful API方式设计，这样可以保证浏览器或者其它客户端都能够使用后端的功能。

- 第二个是逻辑分离的Web静态内容服务，独立于业务逻辑运行，用于向客户端提供前端代码。

后端的模块结构图如下：

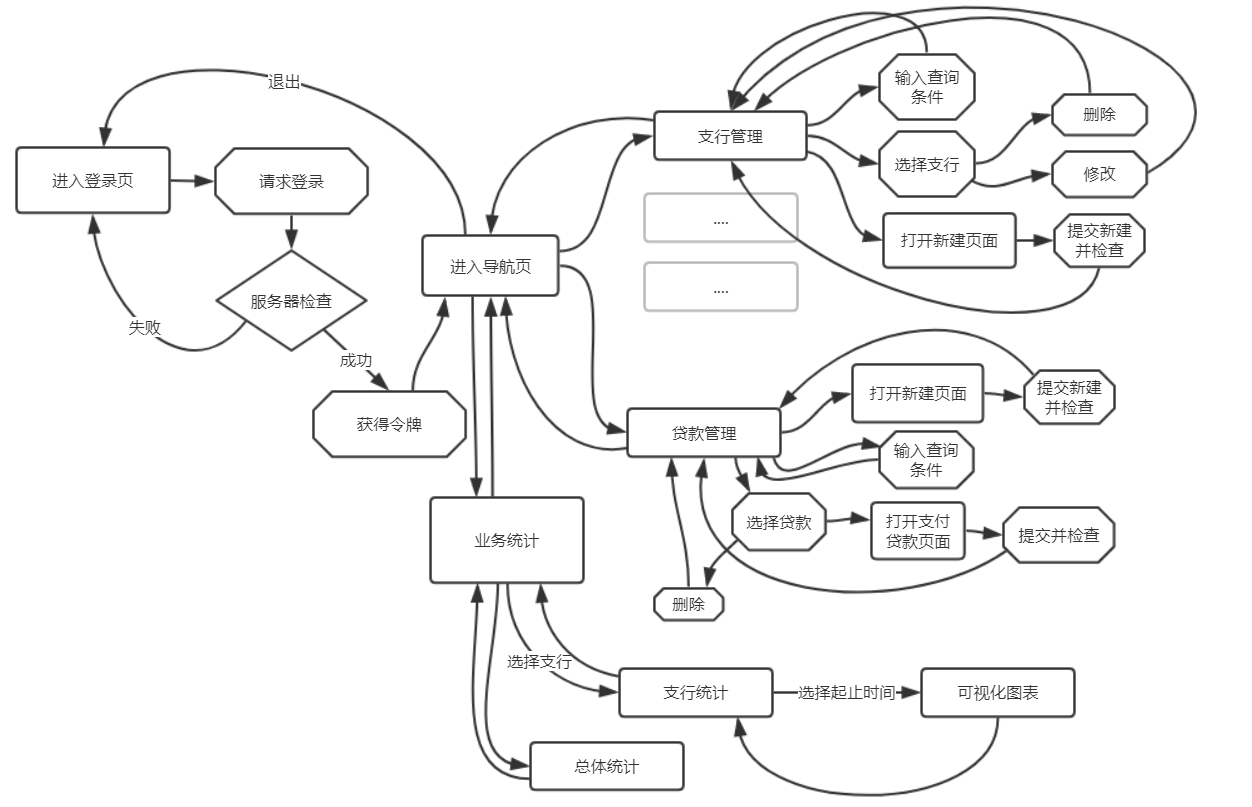
后端模块结构图

本系统前端由可独立运行的HTML, css, js文件组成。使用Bootstrap框架。使用ajax向后端发送命令或请求。使用cookie进行简单的登陆验证。前端的模块结构图如下：

前端模块结构图

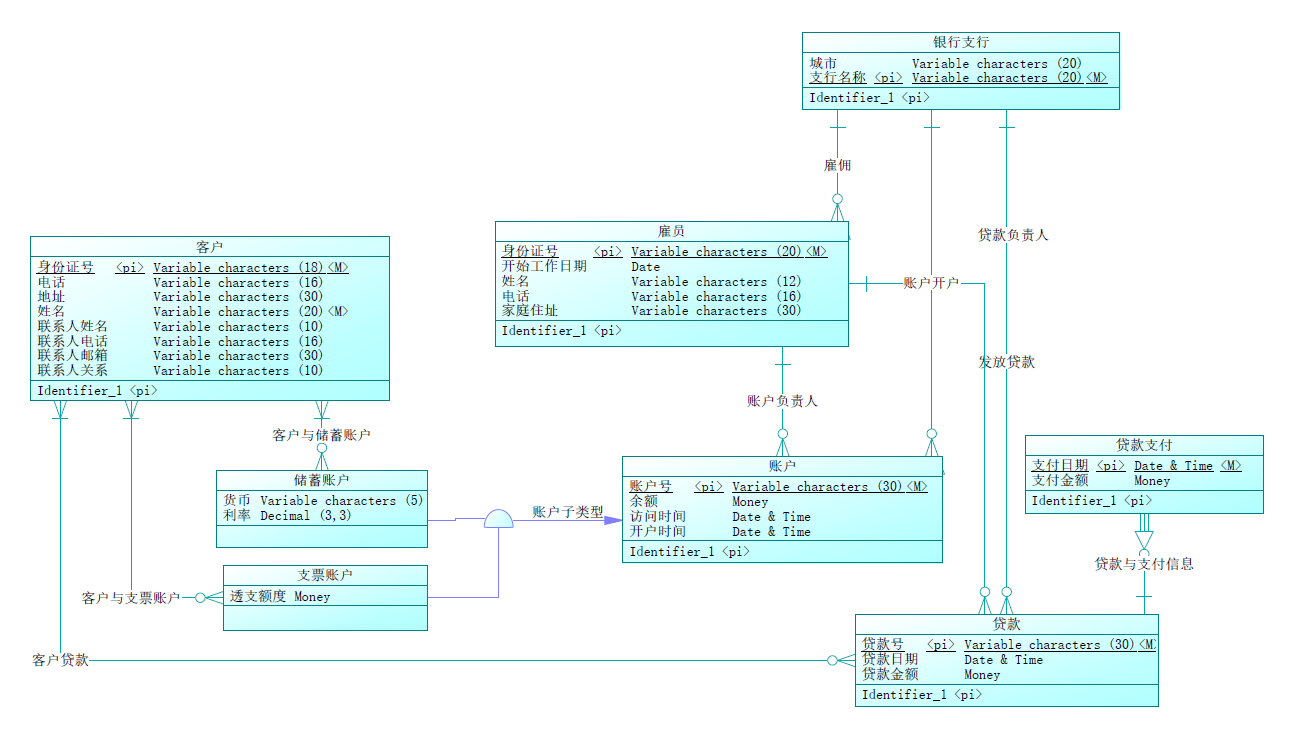
各级模块之间的关系以及基本功能已经在图上和前面的文字中说明。

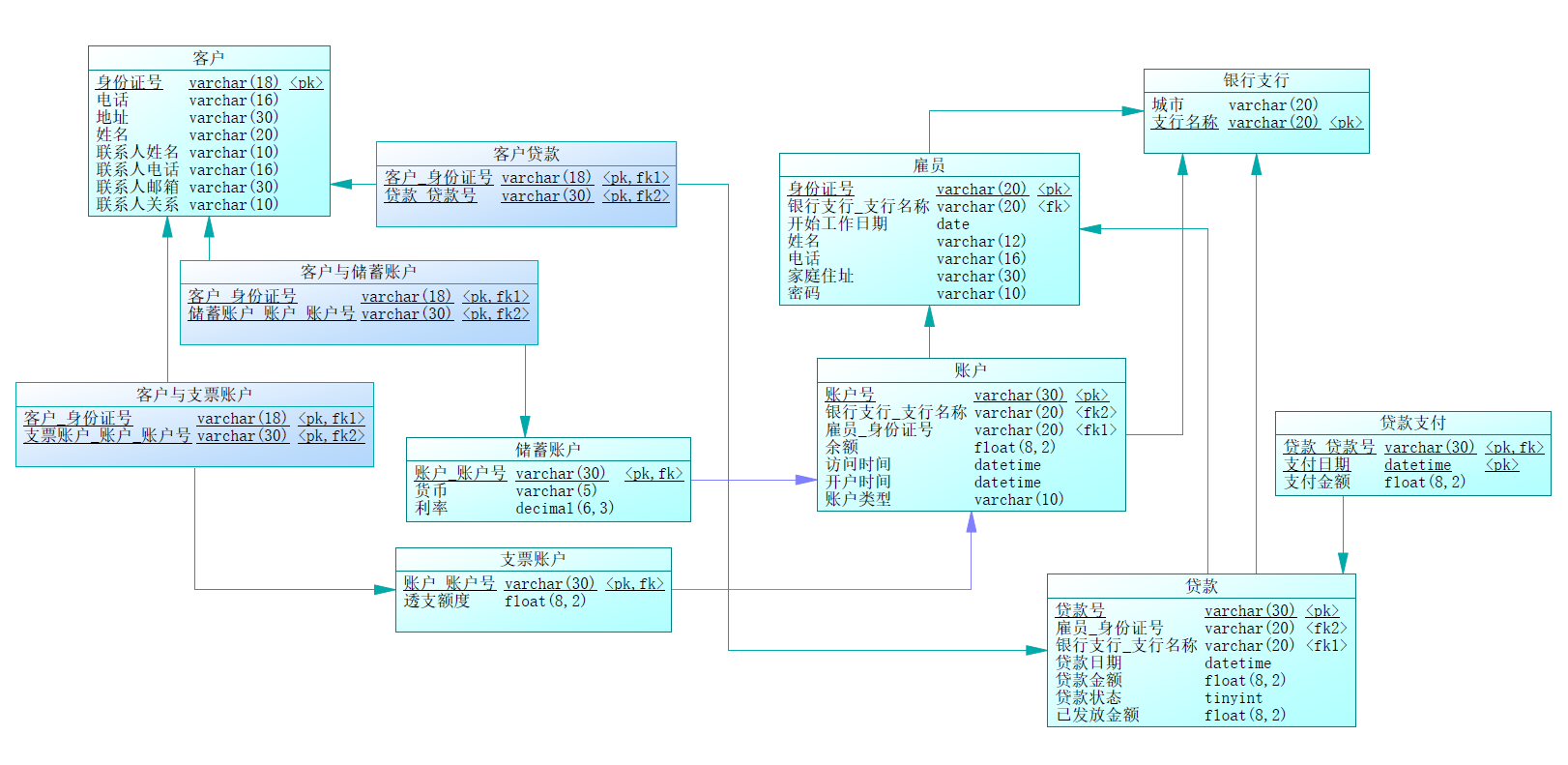
## 2.2 系统工作流程



使用者视角的系统工作流程图

## 2.3 数据库设计

数据库设计ER图



数据库表结构图

数据库表以及约束的代码，详见代码DBLab3/database/create.sql.

# 3 详细设计

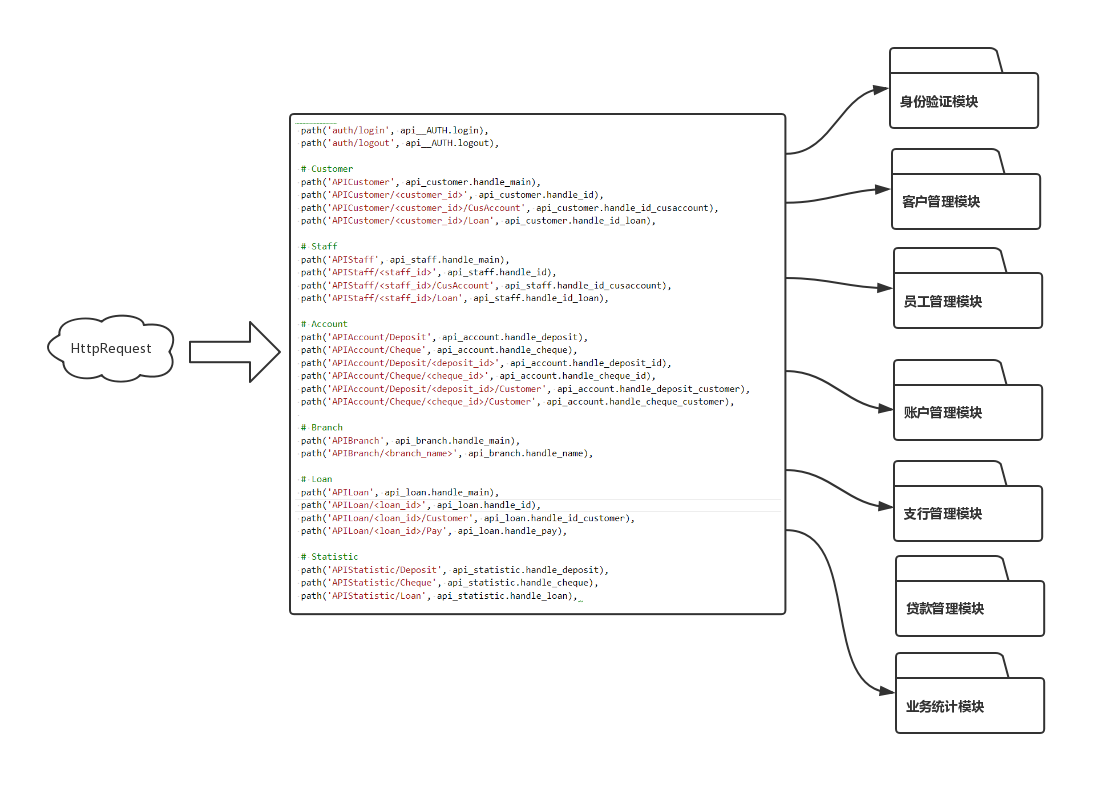
由于前端后端分离，且前端模块是随着前端所用的技术而变化的，再此仅给出后端和数据库模块的详细设计。

## 3.1 后端路由模块

后端路由模块主要负责将请求分发到功能模块去。

输入：任何HttpRequest请求。

输出：对应的处理函数。



## 3.2 后端身份验证模块

身份验证模块接受如下两种输入。

1. 登录请求

格式：GET /auth/login?username=xxx&password=xxx

作用：登录。自动设置域名cookie中的token值。

输出：返回HttpResponse，设定用户浏览器的cookie。

2. 退出请求

格式：GET /auth/logout

作用：使指定cookie失效，并让用户退出登录。

输出：返回HttpResponse，清除用户浏览器的cookie。

## 3.3 后端客户管理模块

客户管理模块支持的输入有如下几类：

1. 查询请求

格式：GET /APICustomer [?customPhone=xxx&customAddress=xxx&customName=xxx…]

查询支持：对于字符串字段，支持字串查询；对于数值，支持范围查询；对于日期，支持起止时间查询。以下所有的查询都支持。

作用：列出所有客户。支持查询。查询参数可选。

返回：HTTP响应, JSON载体，含有所有客户信息。

2. 新建客户请求

格式：POST /APICustomer。请求载荷为JSON格式的客户数据。

作用：新建一个客户

返回：HTTP响应, 新建操作状态。

3. 关于指定客户本身的请求

操作1： GET /APICustomer/ID. 作用：获取某个客户的信息

操作2： PUT /APICustomer/ID. 作用：更新某个客户的信息（提供该客户的部分信息）

操作3： DELETE /APICustomer/ID. 作用：删除某个客户

返回：都是操作状态信息。

4. 获取指定客户账户信息的请求

格式：GET /APICustomer/ID/CusAccount

作用：列出某个指定客户的所有账户

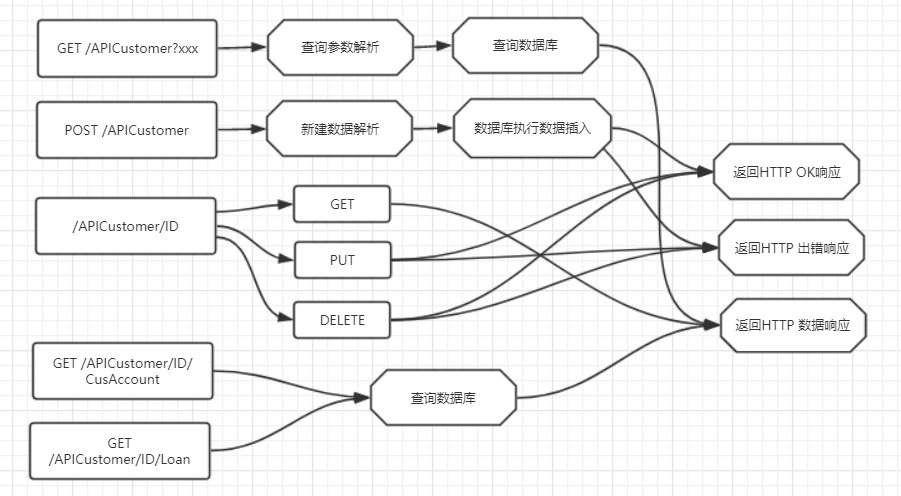
返回：HTTP响应，储蓄账户一个列表，支票账户一个列表。

5. 获取指定客户贷款信息的请求

格式：GET /APICustomer/ID/Loan:

作用：列出某个指定客户的所有贷款

返回：HTTP响应，贷款账户列表。

后端员工管理模块程序流程图

## 3.4 后端账户管理模块

账户管理模块支持的输入有如下几类：

1. 查询储蓄账户

格式：GET /APIAccount/Deposit[?remaingt=xxx&remainlt=xxx&visitTimefrom=xxx...]

输出：HTTP响应，储蓄账户列表。

2. 查询支票账户

格式：GET /APIAccount/Cheque[?remaingt=xxx&remainlt=xxx&visitTimefrom=xxx...]

输出：HTTP响应，支票账户列表。

3. 新建一个储蓄账户

格式：POST /APIAccount/Deposit。载荷为储蓄账户数据。

输出：HTTP响应，操作状态。

4. 新建一个支票账户

格式：POST /APIAccount/Cheque。载荷为储蓄账户数据。

输出：HTTP响应，操作状态。

5. 获取某个储蓄账户的信息

格式：GET /APIAccount/Deposit/<ID>

输出：HTTP响应，单个账户信息。

6. 获取某个支票账户的信息

格式：GET /APIAccount/Cheque/<ID>

输出：HTTP响应，单个账户信息。

7. 给某个储蓄账户添加客户

格式：POST /APIAccount/Deposit/<ID>

输出：HTTP响应，操作状态。

8. 给某个支票账户添加客户

格式：POST /APIAccount/ Cheque/<ID>

输出：HTTP响应，操作状态。

9. 更新某个储蓄账户的信息(只需要提供账户的部分信息)

格式：PUT /APIAccount/Deposit/<ID>, 载荷是部分数据列表。

输出：HTTP响应，操作状态。

10. 更新某个支票账户的信息（只需要提供该账户的全部信息）

格式：PUT /APIAccount/Cheque/<ID>, 载荷是部分数据列表

输出：HTTP响应，操作状态。

11. 删除某个储蓄账户。

格式：DELETE /APIAccount/Deposit/ID

输出：HTTP响应，操作状态。

12. 删除某个支票账户

格式：DELETE /APIAccount/Cheque/ID

输出：HTTP响应，操作状态。

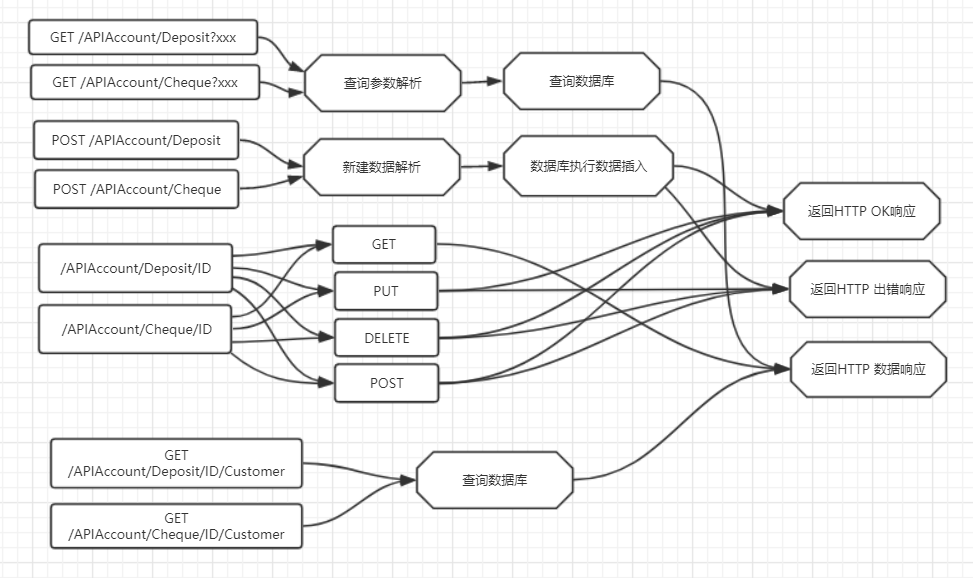
13. 列出某个储蓄账户的所有客户信息

格式：GET /APIAccount/Deposit/ID/Customer

输出：HTTP响应，操作状态。

14. 列出某个支票账户的所有客户信息

格式：GET /APIAccount/Cheque/ID/Customer

输出：HTTP响应，操作状态。

后端账户管理模块流程图

## 3.5 后端员工管理模块

员工管理模块支持的输入有如下几类：

1. 查询所有员工

格式：GET /APIStaff[?bran\_branchName=xxx&staffName=xxx&staffPhone=xxx..]

输出：HTTP响应，员工列表。

2. 新建一个员工

格式：POST /APIStaff

输出：HTTP响应，操作状态。

3, 对于一个员工的操作

操作1：GET /APIStaff/<ID> 作用：获取某个员工的信息

操作2：PUT /APIStaff/<ID> 作用：更新某个员工的信息（提供该员工的部分信息）

操作3：DELETE /APIStaff/<ID> 作用：删除某个员工

输出：GET返回指定员工信息，其余两个返回操作状态。

4. 列出某个指定员工的所有管理账户信息

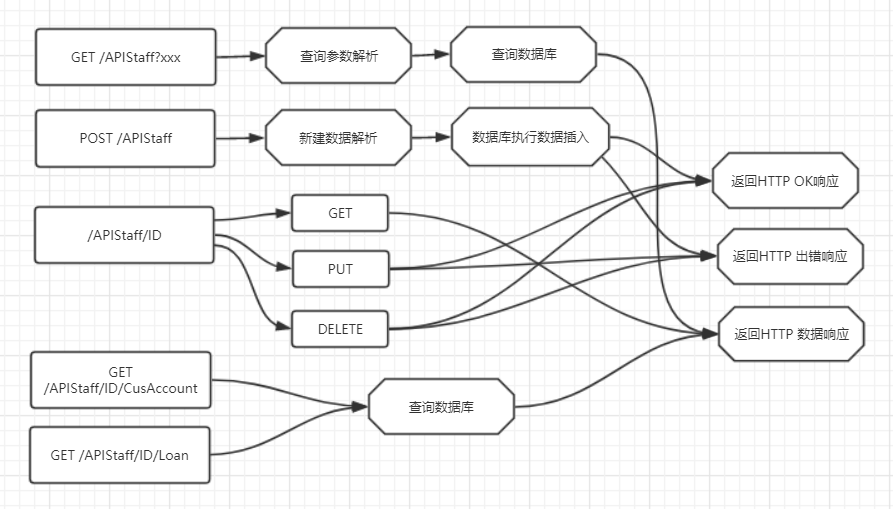
格式：GET /APIStaff/<ID>/CusAccount

输出：HTTP响应，一个储蓄账户列表，一个支票账户列表。

5. 列出某个指定员工的所有负责贷款信息

格式：GET /APIStaff/<ID>/Loan

输出：HTTP响应，贷款列表。



后端员工管理模块流程图

## 3.6 后端贷款管理模块

账户管理模块支持的输入有如下几类：

1. 查询所有贷款。

格式：GET /APILoan[?loanDatefrom=xxx&loanDateto=xxx&loanAmountlt=xxx…]

输出：HTTP响应，贷款列表。

2. 新建一个贷款

格式：POST /APILoan

输出：HTTP响应，操作状态。

3. 对一个贷款的操作

操作1：GET /APILoan/ID 作用：获取某个贷款账户的信息

操作2：DELETE /APILoan/ID 作用：删除某个贷款信息

操作3：POST /APILoan/ID 作用：给某个贷款添加账户. 载荷上传账户号。

输出：GET返回数据项，其它返回操作状态。

4. 获取某个贷款账户的客户

格式：GET /APILoan/ID/Customer

输出：HTTP响应，客户列表。

5. 获取某个贷款账户的发放记录

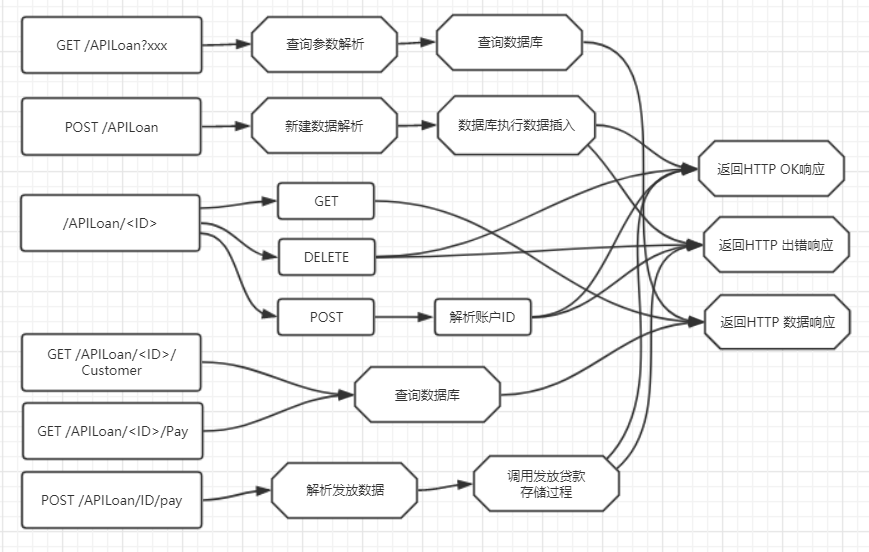
格式：GET /APILoan/ID/Pay

输出：HTTP响应，发放记录列表。

6. 发放贷款

格式：POST /APILoan/ID/pay

输出：HTTP响应，操作状态。



后端贷款管理模块流程图

## 3.7 后端支行管理模块

支行管理模块支持的输入有如下几类：

1. 查询所有支行。

格式：GET /APIBranch[?branchName=xxx&city=xxx]

输出：HTTP响应，支行列表。

2. 新建一个支行

格式：POST /APIBranch, 载荷上传一条支行数据

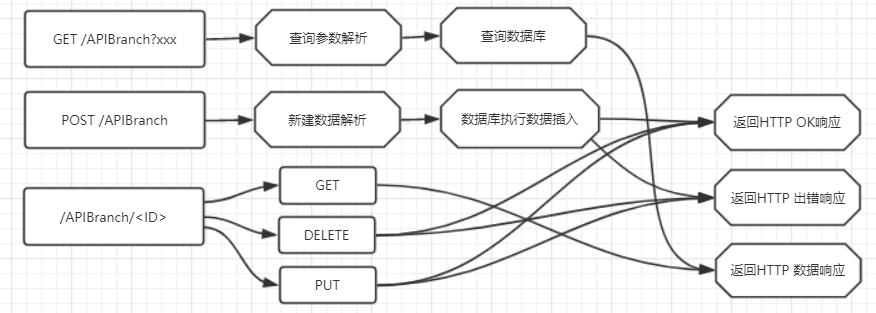
输出：HTTP响应，操作状态。

3. 对一个支行的操作

操作1：GET /APIBranch/BranchName 作用：获取某个支行的信息

操作2：PUT /APIBranch/BranchName 作用：更新某个支行的信息（City）

操作3：DELETE /APIBranch/BranchName 作用：删除某个支行

 输出：GET返回支行数据条目，其它的返回操作状态。

后端支行管理模块流程图

## 3.8 后端业务统计模块

业务统计模块支持的输入有如下几类：

1. 储蓄信息统计

格式：GET /APIStatistic/Deposit[?xxxx]

查询支持：

支行 bran\_branchName=xxx

开户时间起点 openTimefrom=xxx

开户时间终点 openTimeto=xxx

输出：HTTP响应，按照查询条件过滤后的分币种的余额统计信息和账户个数统计信息

2. 支票信息统计

格式：GET /APIStatistic/Cheque[?xxxx]

查询支持：

支行 bran\_branchName=xxx

开户时间起点 openTimefrom=xxx

开户时间终点 openTimeto=xxx

输出：HTTP响应，按照查询条件过滤后的余额统计信息和账户个数统计信息

3. 贷款信息统计

格式：GET /APIStatistic/Loan[?xxxx]

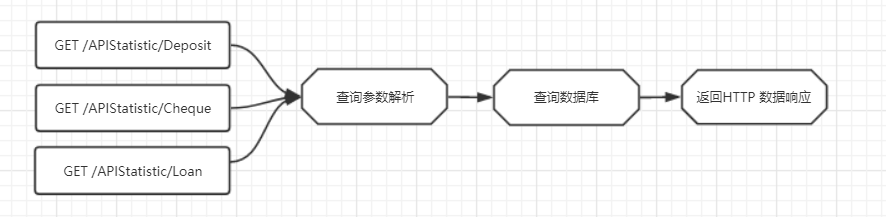
查询支持：

支行 bran\_branchName=xxx

贷款时间起点 loanDatefrom=xxx

贷款时间终点 loanDateto=xxx

输出：HTTP响应，按照查询条件过滤后的贷款金额，贷款发放金额，贷款账户个数信息



后端业务统计模块流程图

# 4 实现与测试

## 4.1 实现结果

// 给出各个功能需求的实现界面和运行结果。

### 4.1.1 支行管理需求

实现的功能界面截图如下：

增加的流程是xxx，运行结果截图如下：（可以多个图在同一行，比如一行3图）

查询的流程是xxx，运行结果截图如下：

### 4.1.2 员工管理需求

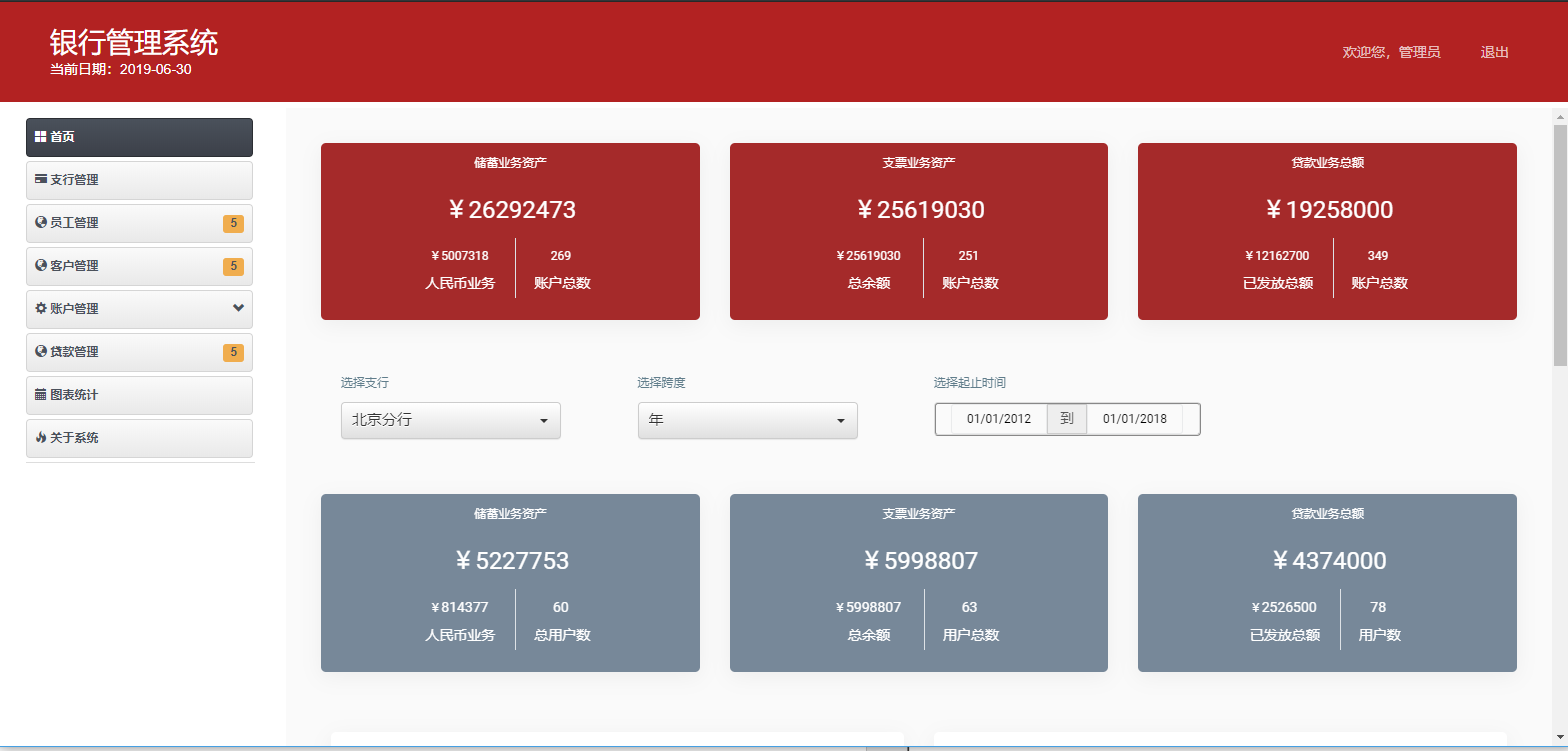
### 4.1.3 客户管理需求

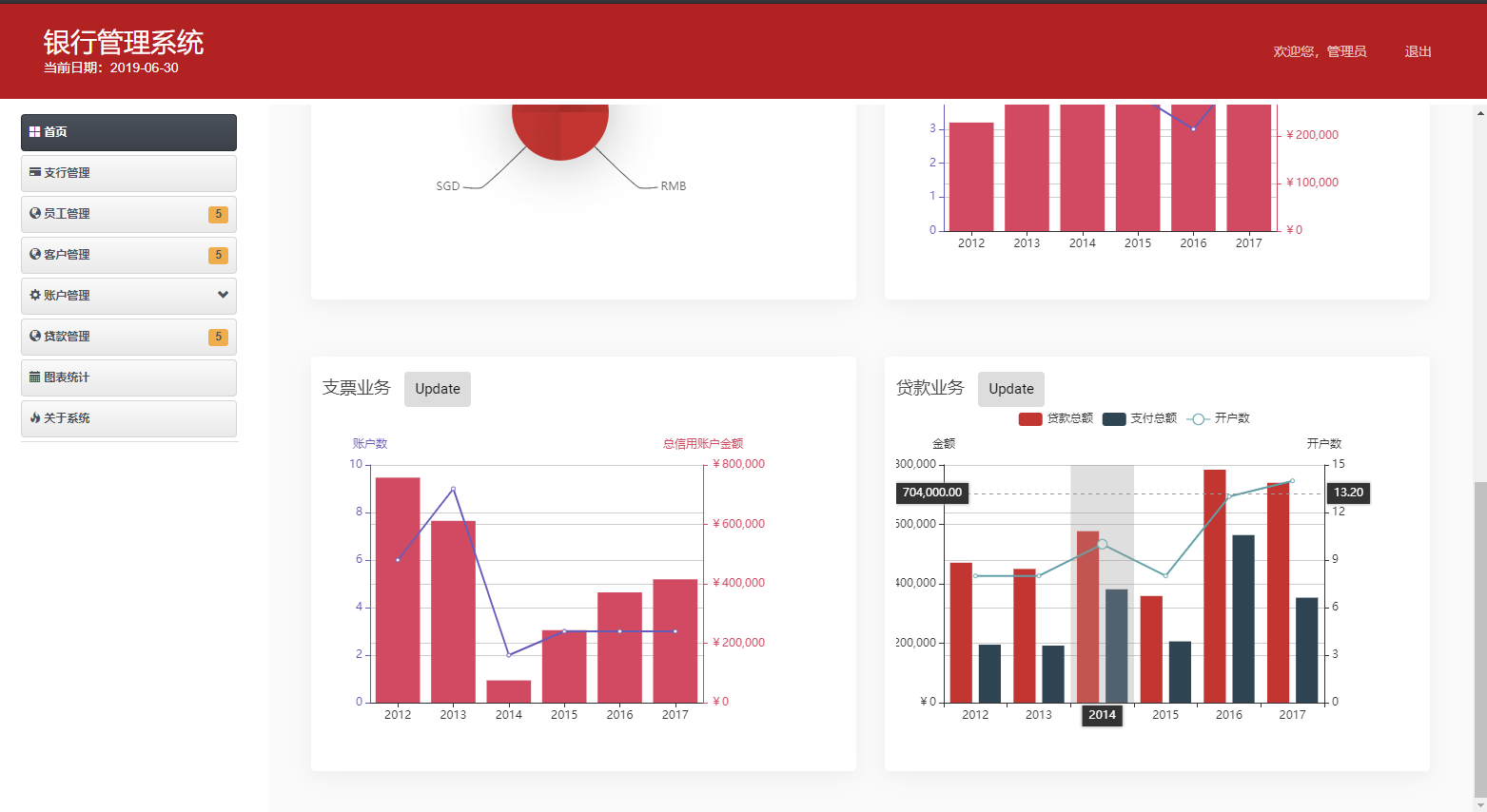
### 4.1.4 账户管理需求

### 4.1.5 贷款管理需求

### 4.1.5 业务统计需求（wb写）

实现的功能界面截图如下：

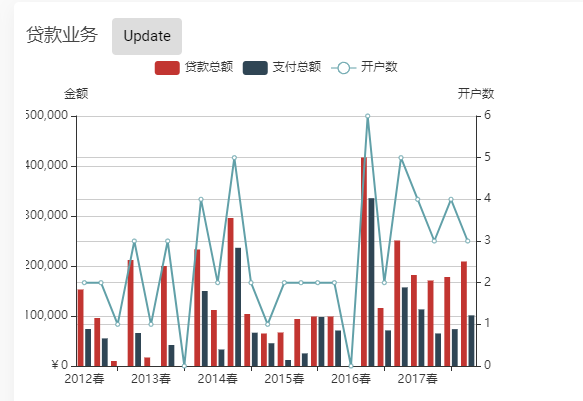
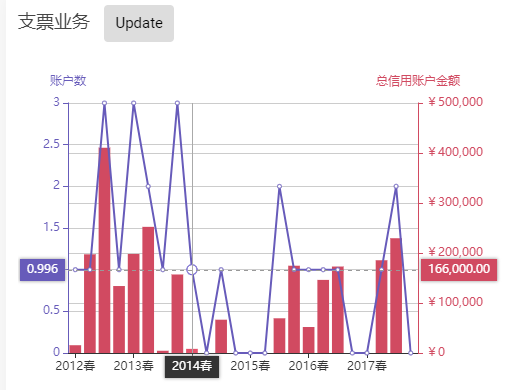




进入图表统计面板，即可在上面三个红色面板中自动看到整个银行三类统计的总数。包括账户总余额，贷款总金额，贷款发放总金额，各类账户总数。

点击选择分行，下拉选择一个分行，则下方三个灰色面板会自动更新该分行的统计结果。

接着，选择时间跨度（如图选择了年），然后在右侧起止时间中输入起止时间，接着在下方的图表上，点击更新即可看到所选择时间范围的柱状图/折线图。

如果切换到季度或者月份会得到更细粒度的图表。如下图所示：

## 4.2 测试结果

为了进行功能测试，我编写了详尽的测试文件，并将后端返回的数据和预期数据进行比较。

测试样例如下（在文件backend/test\_api.py）：

# ------------------ 测试控制 ------------------

test\_ctl = {

'Customer':False,

'Staff':False,

'DepAccount':False,

'CheAccount':False,

'Branch':False,

'Loan':False,

'Statistic':False

}

# # Customer

if test\_ctl['Customer']:

printresp(requests.get("http://localhost:8000/api/v1/APICustomer"))

# 合法查询(参数错误的例子略)

printresp(requests.get("http://localhost:8000/api/v1/APICustomer?customName=%E5%B1%B1"))

# 获取单个（存在）

printresp(requests.get("http://localhost:8000/api/v1/APICustomer/221403200601286584"))

# 获取单个（不存在）

printresp(requests.get("http://localhost:8000/api/v1/APICustomer/123455543211111111"))

# 删除单个（不存在）

printresp(requests.delete("http://localhost:8000/api/v1/APICustomer/221888888888888888"))

# 删除单个（存在）主键约束不可删

printresp(requests.delete("http://localhost:8000/api/v1/APICustomer/221403200601286584"))

# 创建单个（存在）重复主键不能创建

printresp(requests.post("http://localhost:8000/api/v1/APICustomer",

{"customID": "221403200601286584", "customPhone": "1385110025900", "customAddress": "上海", "customName": "宋大山",

"relName": "钱上", "relPhone": "1735054008861", "relEmail": "centbest@hotmail.com", "relRelation": "配偶"}))

# 更新单个

printresp(requests.put("http://localhost:8000/api/v1/APICustomer/221403200601286584", {'field':['customPhone', 'customAddress'], 'field\_value':['1331101010000', '吉林']}))

printresp(requests.get("http://localhost:8000/api/v1/APICustomer/221403200601286584"))

printresp(requests.put("http://localhost:8000/api/v1/APICustomer/221403200601286584", {'field':['customPhone', 'customAddress'], 'field\_value':['1385110025900', '长沙']}))

printresp(requests.get("http://localhost:8000/api/v1/APICustomer/221403200601286584"))

# 测试一个Customer的Account

printresp(requests.get("http://localhost:8000/api/v1/APICustomer/221403200504282880/CusAccount"))

# 测试一个Customer的Loan

printresp(requests.get("http://localhost:8000/api/v1/APICustomer/221403200601286584/Loan"))

';'

# # Staff

if test\_ctl['Staff']:

# 基本

printresp(requests.get("http://localhost:8000/api/v1/APIStaff"))

# 合法查询 参数错误略

printresp(requests.get("http://localhost:8000/api/v1/APIStaff?bran\_branchName=AHB"))

printresp(requests.get("http://localhost:8000/api/v1/APIStaff?staffName=小牛&staffAddress=上海"))

# 获取单个（存在）（不存在略）

printresp(requests.get("http://localhost:8000/api/v1/APIStaff/443403199605063164"))

# 创建单个

printresp(requests.post("http://localhost:8000/api/v1/APIStaff",

{"staffID": "410400200001010000", "bran\_branchName": "AHB", "startWorkDate": "2019-01-01", "staffName": "哇哈哈", "staffPhone": "1887777555555", "staffAddress": "南京"}))

printresp(requests.get("http://localhost:8000/api/v1/APIStaff/410400200001010000"))

# 更新单个

printresp(requests.put("http://localhost:8000/api/v1/APIStaff/410400200001010000", {'field':['staffPhone', 'startWorkDate'], 'field\_value':['1776666444444', '2019-06-15']}))

printresp(requests.get("http://localhost:8000/api/v1/APIStaff/410400200001010000"))

# 删除单个

printresp(requests.delete("http://localhost:8000/api/v1/APIStaff/410400200001010000"))

# 删除单个（约束删不掉）（不存在略）

printresp(requests.delete("http://localhost:8000/api/v1/APIStaff/443403199605063164"))

# 一个Staff的Account

printresp(requests.get("http://localhost:8000/api/v1/APIStaff/443403199605063164/CusAccount"))

# 一个Staff的Loan

printresp(requests.get("http://localhost:8000/api/v1/APIStaff/443403199605063164/Loan"))

if test\_ctl['DepAccount']:

# DepAccount

# 储蓄基本

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Deposit"))

# 储蓄查询

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Deposit?currency=RMB"))

# 单个储蓄账户

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Deposit/00403386247866740608"))

# 单个储蓄账户的客户

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Deposit/00403386247866740608/Customer"))

# 储蓄创建

printresp(requests.post("http://localhost:8000/api/v1/APIAccount/Deposit",

{

"staf\_staffID": "221403198701122751", # AHB银行

"cust\_customID": "221403199203056861",

"accountIDX": "00000088888866666699",

"remain": 10000.0,

"currency": "RMB",

"interest": "3.230"

}))

# 给这个账户添加其它用户

printresp(requests.post("http://localhost:8000/api/v1/APIAccount/Deposit/00000088888866666699",{

"cust\_customID": "410403198512235470"}))

# 这个客户应该多了一个账户 printresp(requests.get("http://localhost:8000/api/v1/APICustomer/410403198512235470/CusAccount"))

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Deposit/00000088888866666699"))

# 储蓄更改（余额变动为-10000(花完)，否则下一步不能删除）

printresp(requests.put("http://localhost:8000/api/v1/APIAccount/Deposit/00000088888866666699", {

'field': ['cust\_customID', 'remain\_change'],

'field\_value':['221403199203056861', -10000]

}))

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Deposit/00000088888866666699"))

# 储蓄删除

printresp(requests.delete("http://localhost:8000/api/v1/APIAccount/Deposit/00000088888866666699"))

# 这个客户应该少一个账户 printresp(requests.get("http://localhost:8000/api/v1/APICustomer/410403198512235470/CusAccount"))

if test\_ctl['CheAccount']:

# CheAccount

# 支票基本

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Cheque"))

# 支票查询

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Cheque?remainlt=20000&neg\_limitgt=3000"))

# 单个支票账户

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Cheque/00278883948040893941"))

# 单个支票账户的客户

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Cheque/00278883948040893941/Customer"))

# 支票创建

printresp(requests.post("http://localhost:8000/api/v1/APIAccount/Cheque",

{

"staf\_staffID": "221403199112040167", # BJB银行

"cust\_customID": "221403200504282880",

"accountIDX": "11111188888866666600",

"remain": 10000.0

}))

# 给这个支票账户添加其它用户(添加后410403199604140379多一个账户)

printresp(requests.post("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600",{

"cust\_customID": "410403199604140379"}))

# 支票修改(改为0)

printresp(requests.put("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600", {

'field': ['cust\_customID', 'remain\_change'],

'field\_value':['221403200504282880', -11000]

}))

printresp(requests.put("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600", {

'field': ['cust\_customID', 'remain\_change'],

'field\_value':['221403200504282880', -1000]

}))

# 这个会花超，失败。

printresp(requests.put("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600", {

'field': ['cust\_customID', 'remain\_change'],

'field\_value':['221403200504282880', -1000]}))

printresp(requests.get("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600"))

printresp(requests.put("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600", {

'field': ['cust\_customID', 'remain\_change'],

'field\_value':['221403200504282880', +2000]

}))

# 支票删除（余额0才能删）（删除后410403199604140379少一个账户）

printresp(requests.delete("http://localhost:8000/api/v1/APIAccount/Cheque/11111188888866666600"))

if test\_ctl['Branch']:

# Branch

# 基本

printresp(requests.get("http://localhost:8000/api/v1/APIBranch"))

# 合法查询

printresp(requests.get("http://localhost:8000/api/v1/APIBranch?branchName=AH"))

# 查询某个支行

printresp(requests.get("http://localhost:8000/api/v1/APIBranch/AHB"))

# 添加某个支行

printresp(requests.post("http://localhost:8000/api/v1/APIBranch", {"city":"NewYork", "branchName":"AAA"}))

# 删除某个支行(约束删不掉, 略) printresp(requests.delete("http://localhost:8000/api/v1/APIBranch/AHB"))

# 添加某个支行(约束加不了, 略) printresp(requests.post("http://localhost:8000/api/v1/APIBranch", {"city":"Hefei", "branchName":"AHB"}))

# 更新某个支行

printresp(requests.put("http://localhost:8000/api/v1/APIBranch/AAA", {"field":['city'], "field\_value": ['Tokyo']}))

printresp(requests.get("http://localhost:8000/api/v1/APIBranch/AAA"))

# 删除某个支行

printresp(requests.delete("http://localhost:8000/api/v1/APIBranch/AAA"))

if test\_ctl['Loan']:

# Loan

# 基本

printresp(requests.get("http://localhost:8000/api/v1/APILoan"))

# 合法查询（非法查询如果是值格式错误有些不报错。注意格式要正确）

printresp(requests.get("http://localhost:8000/api/v1/APILoan?loanAmountgt=2000&loanAmountlt=4000"))

printresp(requests.get("http://localhost:8000/api/v1/APILoan?loanDatefrom=2019-11-01&loanDateto=2020-01-01"))

# 获取某个贷款

printresp(requests.get("http://localhost:8000/api/v1/APILoan/41423895546771431063"))

# 获取某个贷款的用户

printresp(requests.get("http://localhost:8000/api/v1/APILoan/41423895546771431063/Customer"))

# 获取某个贷款的发放记录

printresp(requests.get("http://localhost:8000/api/v1/APILoan/41423895546771431063/Pay"))

# 创建贷款

printresp(requests.post("http://localhost:8000/api/v1/APILoan",

{

"loanIDX": "44449999000022221111",

"staf\_staffID": "443403199605063164",

"bran\_branchName": "SHB",

"loanDate": "2019-06-16 00:00:00",

"loanAmount": 55000.0

}

))

# 获取某个贷款

printresp(requests.get("http://localhost:8000/api/v1/APILoan/44449999000022221111"))

# 给该贷款账户添加客户

printresp(requests.post("http://localhost:8000/api/v1/APILoan/44449999000022221111", {

"cust\_customID": "410403199312140490"}))

# 支付贷款（支付2次，总共达到总额）

# 第一次支付

printresp(requests.post("http://localhost:8000/api/v1/APILoan/44449999000022221111/Pay", {"loanPayDate": datetime.datetime.now(), "loanPayAmount": 22000}))

printresp(requests.get("http://localhost:8000/api/v1/APILoan/44449999000022221111/Pay"))

# 失败的删除

printresp(requests.delete("http://localhost:8000/api/v1/APILoan/44449999000022221111"))

# 第二次支付

printresp(requests.post("http://localhost:8000/api/v1/APILoan/44449999000022221111/Pay", {"loanPayDate": datetime.datetime.now(), "loanPayAmount": 33000}))

printresp(requests.get("http://localhost:8000/api/v1/APILoan/44449999000022221111/Pay"))

# 成功删除贷款

printresp(requests.delete("http://localhost:8000/api/v1/APILoan/44449999000022221111"))

if test\_ctl['Statistic']:

# 按照三类账户统计全部

printresp(requests.get("http://localhost:8000/api/v1/APIStatistic/Loan"))

printresp(requests.get("http://localhost:8000/api/v1/APIStatistic/Deposit"))

printresp(requests.get("http://localhost:8000/api/v1/APIStatistic/Cheque"))

# 按照三类账户,统计起止时间

printresp(requests.get(

"http://localhost:8000/api/v1/APIStatistic/Loan?loanDatefrom=2012-01-01&loanDateto=2020-12-01"))

printresp(requests.get(

"http://localhost:8000/api/v1/APIStatistic/Deposit?openTimefrom=2012-01-01&openTimeto=2020-12-01"))

printresp(requests.get(

"http://localhost:8000/api/v1/APIStatistic/Cheque?openTimefrom=2012-01-01&openTimeto=2020-12-01"))

经过测试，所有测试样例的输出，报错，都和预期一致。由于输出过长，在实验报告中略去。

以下仅举几个例子。

对于成功的数据请求，返回JSON数据:

{

"status": "OK",

"metadata": [

"bran\_branchName",

"sum\_loanAmount",

"bran\_branchName",

"sum\_loanPaid",

"bran\_branchName",

"count\_loan"

],

"data": {

"loan\_sum": [

{

"bran\_branchName": "安徽省分行",

"sum\_loanAmount": 5159000.0

}

],

"paid\_sum": [

{

"bran\_branchName": "安徽省分行",

"sum\_loanPaid": 3175100.0

}

],

"loan\_count": [

{

"bran\_branchName": "安徽省分行",

"count\_loan": 94

}

]

}

}

对于成功的操作请求，返回操作状态:

{"status":"OK", "metadata":null, "data":null}

对于失败的操作请求，返回失败代码和原因：

{

"status": "error",

"describe": "(1451, 'Cannot delete or update a parent row: a foreign key constraint fails (`BankDB`.`loan`, CONSTRAINT `FK\_bank\_and\_loan` FOREIGN KEY (`bran\_branchName`) REFERENCES `branch` (`branchName`))')"

}

# 5 总结与讨论

// 总结本系统开发过程中的主要收获、教训。

# 附录 团队介绍与工作分工

// 团队成员介绍。详细陈述各自的任务分工情况。