

毕业设计(论文)

|  |  |
| --- | --- |
| 课题名称 | 数据可视化分析平台 |
|  | ——API的设计与实现 |
| 二级学院 | 计算机与软件学院 |
| 专 业 | 软件技术 |
| 班 级 | 软件1851 |
| 学 号 | 1802343134 |
| 姓 名 | 尚宇驰 |
| 指导教师 | 许莫淇 |

年 月 日

计算机与软件学院

毕业设计（论文）诚信承诺

我谨在此承诺：本人所写的毕业论文《基于B/S架构的在线数据可视化分析平台——API的设计与实现》，系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释与说明，若有不实，后果由本人承担。

承诺人（签名）：

年 月 日

摘 要

伴随着信息技术与互联网的发展，信息技术对现代社会的渗入可以说是无孔不入。每天各行各业都会产生海量的信息。个人身份信息、交通航线信息、银行各项业务等多种服务都离不开信息技术的支持，其中数据可视化分析平台的建设显得尤为重要，它在整个流程中扮演了幕后英雄的角色。如果缺乏数据可视化平台的幕后支持，所有的硬件和前台界面都不能发挥到应有的功效。随着数据可视化平台的广泛推进，对我国现代各项产业的发展都能起到促进作用。

目前我国正处于数字化转型关键期，面对海量的数据的处理和挖掘这些数据的价值并发挥实际的作用是当今社会的一个重要课题。无论是传统行业还是新型行业，数据越来越被重视；无论是在商业领域还是经济行业，只有基于大量的数据和分析才能产生正确的决策。例如目前的手机应用通过收集用户行为，通过数据分析和算法运算精准为用户推送准确的信息。

本系统以Python flask作为Api端开发的技术框架，以Python作为后端开发语言，借用Python可以全方面处理数据的优势，可以更加得心应手的完成系统的功能，本系统主要完成了用户模块、数据源接入模块、数据预处理模块、数据分析/计算模块和系统维护模块。

关键字：数据可视化、Python、Python flask

Abstract

With the development of information technology and the Internet, the penetration of information technology into modern society can be said to be pervasive. Every day, all walks of life generate massive amounts of information. The development of information technology support, among them, the progress of the construction of the data visualization analysis platform is particularly important, and it plays the role of the hero behind the scenes in the entire process. The effect of. Through the extensive promotion of the data visualization platform, the development of modern modern industries can be promoted.

At present, as a whole, it is in a critical period of digital transformation. Faced with the processing of massive amounts of data, mining the value of these data and playing a practical role is an important issue in today's society. Whether in traditional industries or emerging industries, data is getting more and more attention; for example, existing mobile phone applications collect user behaviors, use data analysis and algorithmic calculations to accurately manipulate accurate information for users.

This system uses Python flask as the technical framework for Api-side development, and Python as the back-end development language. Borrowing the advantages of Python's ability to process data in all aspects, the system can be more handy to complete the functions of the system. This system mainly completes user modules and data sources. Access module, data preprocessing module, data analysis/calculation module and system maintenance module.

Keywords: data visualization, Python, Python flask

目 录

[第1章 绪论 1](#_Toc72667688)

[1.1 课题研究背景与意义 1](#_Toc72667689)

[1.1.1 课题研究背景 1](#_Toc72667690)

[1.1.2 课题研究意义 1](#_Toc72667691)

[1.2 研究目标及内容 1](#_Toc72667692)

[1.2.1 研究目标 1](#_Toc72667693)

[1.2.2 研究内容 2](#_Toc72667694)

[1.3 所做工作 2](#_Toc72667695)

[第2章 开发技术与工具 3](#_Toc72667696)

[2.1 开发技术 3](#_Toc72667697)

[2.1.1 Python Flask 3](#_Toc72667698)

[2.1.2 Python pandas 3](#_Toc72667699)

[2.1.3 PostgreSQL 3](#_Toc72667700)

[2.1.4 MySQL 3](#_Toc72667701)

[2.2 开发工具 4](#_Toc72667702)

[2.2.1 Pycharm 4](#_Toc72667703)

[2.2.2 Navicat 4](#_Toc72667704)

[2.2.3 CentOS 8 4](#_Toc72667705)

[2.2.4 GitHub 4](#_Toc72667706)

[第3章 系统需求分析 5](#_Toc72667707)

[3.1 系统总体需求概述 5](#_Toc72667708)

[3.1.1 系统稳定性分析概述 5](#_Toc72667709)

[3.1.2 产品特点分析概述 5](#_Toc72667710)

[3.2 功能性需求分析 5](#_Toc72667711)

[3.2.1 用户模块 5](#_Toc72667712)

[3.2.2 数据源接入模块 6](#_Toc72667713)

[3.2.3 数据预处理模块 6](#_Toc72667714)

[3.2.4 数据分析/计算模块 7](#_Toc72667715)

[3.2.5 系统维护模块 8](#_Toc72667716)

[第4章 系统概要设计 1](#_Toc72667717)

[4.1 数据库设计 1](#_Toc72667718)

[4.1.1 用户表设计 1](#_Toc72667719)

[4.1.2 用户行为表设计 2](#_Toc72667720)

[4.2 系统日志存储设计 3](#_Toc72667721)

[4.3 系统接口设计 3](#_Toc72667722)

[4.3.1 用户模块接口设计 3](#_Toc72667723)

[4.3.2 数据源接入模块接口设计 4](#_Toc72667724)

[4.3.1 数据预处理模块接口设计 4](#_Toc72667725)

[4.3.2 数据处理/分析模块接口设计 5](#_Toc72667726)

[第5章 系统详细设计 7](#_Toc72667727)

[5.1 用户模块 7](#_Toc72667728)

[5.1.1 GitHub第三方登录功能设计 7](#_Toc72667729)

[5.1.2 记录用户行为功能设计 7](#_Toc72667730)

[5.2 数据源接入模块 8](#_Toc72667731)

[5.2.1 PostgreSQL、MySQL数据库连接功能设计 8](#_Toc72667732)

[5.2.2 Excel、Csv多文件上传功能设计 8](#_Toc72667733)

[5.2.3 查询数据库中所有表功能设计 8](#_Toc72667734)

[5.3 数据预处理模块 8](#_Toc72667735)

[5.3.1 字段分类功能设计 8](#_Toc72667736)

[5.3.2 查询并过滤数据表中所有数据功能设计 8](#_Toc72667737)

[5.3.3 查询数据库中所有表功能设计 8](#_Toc72667738)

[5.3.4 查询指定字段数据功能设计 8](#_Toc72667739)

[5.3.5 自定义构建计算指标功能设计 8](#_Toc72667740)

[5.4 数据处理/分析模块 8](#_Toc72667741)

[5.4.1 数据处理/分析功能设计 8](#_Toc72667742)

[5.4.2 脏数据处理功能设计 8](#_Toc72667743)

[5.5 系统维护模块 8](#_Toc72667744)

[5.5.1 日志记录系统行为功能设计 8](#_Toc72667745)

[5.5.2 统一响应体设计 8](#_Toc72667746)

[第6章 系统功能的实现 1](#_Toc72667747)

[6.1 用户模块 1](#_Toc72667748)

[6.1.1 GitHub第三方登录功能实现 1](#_Toc72667749)

[6.1.2 记录用户行为功能实现 3](#_Toc72667750)

[6.2 数据源接入模块 3](#_Toc72667751)

[6.2.1 PostgreSQL、MySQL数据库连接功能实现 3](#_Toc72667752)

[6.2.2 Excel、Csv多文件上传功能实现 3](#_Toc72667753)

[6.2.3 查询所选连接数据库所有表功能实现 3](#_Toc72667754)

[6.3 数据预处理模块 3](#_Toc72667755)

[6.3.1 字段分类功能实现 3](#_Toc72667756)

[6.3.2 查询并过滤数据表中所有数据功能实现 3](#_Toc72667757)

[6.3.3 查询数据库中所有表功能实现 3](#_Toc72667758)

[6.3.4 自定义构建计算指标功能实现 3](#_Toc72667759)

[6.3.5 指定数据表查询所有字段功能实现 3](#_Toc72667760)

[6.4 数据处理/分析模块 3](#_Toc72667761)

[6.4.1 数据处理/分析功能实现 3](#_Toc72667762)

[6.4.2 脏数据处理功能实现 3](#_Toc72667763)

[6.5 系统维护模块 3](#_Toc72667764)

[6.5.1 日志记录系统行为功能实现 3](#_Toc72667765)

[6.5.2 统一响应体实现 3](#_Toc72667766)

[第7章 系统测试 4](#_Toc72667767)

[7.1 测试环境准备 4](#_Toc72667768)

[7.2 功能性测试 4](#_Toc72667769)

[7.3 系统部署测试 4](#_Toc72667770)

[第8章 总结和展望 5](#_Toc72667771)

[8.1 总结 5](#_Toc72667772)

[8.2 展望 5](#_Toc72667773)

[致谢 6](#_Toc72667774)

# 绪论

近年来，研究人员对数据分析研究的越来越深入，分析的方法也越来越多样化，加之，数据可视化技术的日益成熟，可以借助图形化的方式，直观生动的展现数据的变化趋势，进而分析出数据的发展走向；所以数据可视化平台是当今社会发展必不可少的助力器。本章将先介绍课题研究背景和研究意义，然后明确研究目标叙述研究的内容，最后陈述在准备论文时所准备的工作。

## 课题研究背景与意义

### 课题研究背景

近些年来，随着人类社会的进步，互联网行业快速发展，社会信息量的不断累计，呈现爆炸式增长，可以说大数据时代已经全面到来。大数据时代的背景下，在纷繁的数据中找到规律和结论并创造价值是核心思路，而如何在海量的数据中将有价值的信息分类、汇总及分析，同时将大量的数据和结果以更直观、美观的方式展现出来，并让人快速获取行业内聚焦的重要方向。面对海量数据的处理与分析，简单阅读数据表中的内容已经满足不了人们对于数据的分析而数据可视化分析平台则是与之应对很好的解决方案。

### 课题研究意义

互联网的普及和信息技术的迅速发展使得数据快速增长，海量数据的急剧产生使得人们的分析处理这些数据变得十分困难，急需研究和开发一款新的技术来对数据进行处理和分析。正所谓“一图胜千言”，数据可视化分析平台和分析技术的研究和应用为这一问题提供了解决方式。

理解数据最重要的方式之一就是通过图形的方式来表示数据，这样不仅可以更直观的展示信息，增加数据的灵活性，同时可以清晰地了解一些可能被隐藏的数据模式，从而帮助用户快速获取数据背后隐藏的信息，数据可视化分析平台从复杂的数据中分析出 数据的走向，将用户从一堆杂乱无章的数据中解放出来[1], 快速获取数据提供的信息，实现数据价值。

## 研究目标及内容

### 研究目标

本文研究的主要目标是用户在使用系统时可以完成GitHub的第三方登录，对于指定的数据类型进行数据库的连接，以完成数据的接入，接入成功可以对数据进行分析和处理。具体的研究目标如下：

A. 通过指定类型的数据库完成数据库的连接，查询该连接中的表，和表中的数据，为数据的分析和处理做好铺垫。

B. 根据传入的连接和字段名，查询数据库表中的数据，完成数据的拆分。

C. 根据运算符号的优先级，运算出所传算数式的计算结果，完成自定义构建计算指标的功能。

D. 处理传入数据的脏数据，脏数据的存在会影响数据的分析过程和结果，借助Python中的pandas库对脏数据进行处理。

### 研究内容

本文的研究内容是实现数据可视化平台的API端的设计与实现，用户通过GitHub的第三方登录可以接入数据源，接入数据源的方式有两种，一种是数据库的连接还有一种是csv、excel文件的上传；然后对数据进行清洗、分析、处理将分析结果传给前端进行展示。在数据清洗的功能上，选择了Python提供的pandas库对文件中空值数据进行处理。本课题的研究主要内容包括：

A. 对不同的数据可视化分析平台进行分析，了解竞品中的共性和独有的个性，确定平台应该必有的功能。

B. 为了使用Python语言进行数据的处理，学习Python Flask并选择其为API端的开发技术进行后端框架的搭建。

C. 对于不同的请求方式进行封装，保持代码的整洁性变更将系统开发的测试结果记入日志，提高系统的稳定性。

## 所做工作

为了使系统实现的功能更加完整，系统性能更稳定，在开发中做了如下工作：

A. 体验目前流行的数据可视化分析平台，找出其共同的功能点作为系统的必做功能点。

B. 寻找高效的开发技术和工具，在众多的开发语言中选择Python作为后端的开发语言，因为Python提供丰富的库可以使用，在数据的清洗上具有极强的优势，开发工具则选择Pycharm，它是Python专用的IDE,提供的开发环境对于开发者也十分友好。

C. 选择合适的后端框架和学习使用数据分析处理的需要使用的技术。API端的框架选择使用Python Flask进行搭建，其优势在于轻巧、简洁并且扩展性强；数据处理的库选择Python提供的pandas，其对于数据的脏数据的清理和数据分析有特有的优势，处理数据十分灵活。

# 开发技术与工具

调研近年来数据可视化“Data visualization”方向的资料，我们发现Python、R、Ruby是主要流行语言，从学习成本和快捷开发的角度分析，我们选择了Python来实现此项目，并且在众多框架中选择了轻量级的Web框架——flask，来实现搭建，大量数据处理则选用pandas库来完成。相较与传统的MySQL关系型数据库，PostgreSQL标榜自己是世界上最先的开源数据库，它对于集群的支持更加友好；本章将讲述系统所使用的开发技术和开发工具。

## 开发技术

### Python Flask

Python Flask是一个轻量级的Web框架，它的特色为”micro”即“微型”。“微型”并不意味着整个Web应用程序必须适合单个Python文件，也并非Flask缺少功能。微框架意味着Flask核心保持简单但可扩展。它主要在于可以通过添加你所需要的功能对应的库，来为应用程序提供支持，而不是直接给一个大而全的全家桶框架，不管你是否需要。Flask提供许多扩展库，可以满足你的所有需求。

### Python pandas

Python pandas 是一个数据分析处理能力优秀的开源库。它致力于成为应用Python语言，实际的、真实的解决数据分析问题的基础高级构建块。DataFrame对象是一个快速高效的对象，用于带有集成索引的数据操作，包括对数据进行索引、切割、合并、排序、分类聚合等等操作，是pandas处理数据方便快捷的主要优势。它还提供内存数据结构和不同格式之间读取和写入数据的工具，支持CSV和文本文件，Microsoft Excel，SQL数据库和快速HDF5格式。

### PostgreSQL

PostgreSQL是一个功能强大的、开源的、跨平台的对象-关系型数据库系统，它使用和扩展了SQL语言，用于安全地存储数据，和结合了许多功能，可以扩展最复杂的数据工作负载，并且在众多数据库系统中，拥有更强的稳定性。它具有更多细节功能支持，使用起来舒适度更高，更便捷，比如：text字段可以支持所有大小，支持OVER子句，可以存储array和json，并且在之上建立索引，在复制的特性上做到同步，异步，半同步等等。

### MySQL

MySQL数据库是当下较为流行的对象-关系型数据库之一，是开源的 SQL数据库管理系统。数据库是一种根据数据结构来存储和管理数据的仓库，是数据的结构化集合。[1] 想要更好的应用数据库就需要一个合适的数据库管理系统即MySQL，它的数据库是关系型的，区别在于并非将所有数据随意放在大存储室中，而是把数据存在单独的表中，具有对象结构：数据库、表、视图、行和列。SQL则是“结构化查询语言”，是访问数据库的工具，通过SQL语言对数据库中的数据进行一些列操作。

## 开发工具

### Pycharm

Pycharm 是一个专门用于Python语言的、跨平台的、计算机编程集成开发环境，由捷克JetBrains公司开发。它具备代码智能感应、动态检查和快速修复功能，提供一些主流Web框架（Flask、Django）的强大支持，拥有大量的开箱即用工具集，为编程创造了舒适便捷的环境。

### Navicat

Navicat是一套多连接数据库开发工具。它拥有连接七种数据库的能力，致力于让用户专注在数据库设计上，简化其他一切创造数据模型、数据库的步骤，做到了为专业人士提供适合其特殊需要的复杂功能，同时对于新手上手简单。提供图标编辑器，可以有效的创建和设计数据模型，支持多种图标符号，比如UML、Crow’s Foot和IDEF1x 。

### CentOS 8

CentOS 8是CentOS的第8个发行版本，CentOS是Linux的众多发行版本之一，是从RHEL源代码编译的社区重新发布版。

### GitHub

GitHub是一个代码托管云平台。用户可以上传代码、提交版本来管理自己的代码，更出色的是团队开发时，多人协作管理团队代码，通过提交代码、与旧版本代码合并、更新代码库等操作，实现团队代码管理。

# 系统需求分析

对于大量级数据进行处理，会考虑到处理效率和处理时间的问题，另外，数据分类聚合也是一个挑战点，对于以上问题，本章将从系统总体需求和功能性需求两方面展开分析，首先对数据可视化平台的总体需求进行概述，分析产品的特点和系统对于稳定性的需求；其次，对平台所需要的功能性需求进行模块划分，细化每个模块所实现的具体功能。

## 系统总体需求概述

### 系统稳定性分析概述

稳定性需求反映着系统质量要求，本系统需要处理海量数据，需要考虑到的因素有许多，如响应时间、可维护性等。控制响应时间体现在多个方面，一方面是数据的读取速度，另一方面是数据处理速度，还有最后的数据返回速度。采用psycopg2库以保证连接数据库和读取数据的速度，采用pandas库来保证数据的分析和处理速度。可维护性一方面体现在可操作性，日志的记录可以良好的监控系统内部状态，提供了对运行时行为的可见性，便于运维快速定位问题所在，另一方面体现在简单性，清晰的项目层级、项目架构降低了项目管理的复杂度，模块间松耦合降低了运维操作修改bug、优化性能的复杂度。

### 产品特点分析概述

该系统有如下几个特点：

1. 多数据源连接：并不是传统的项目始终操作一个数据库，配置直接写在一个配置文件即可，而是每次连接都可以是不同的数据库，需要动态的驱动、链接地址、库名。满足不同用户不同数据库的需求。
2. 记录系统行为：将系统的执行过程记录成日志，便于查看系统健康。

## 功能性需求分析

### 用户模块

1. 用户登录。采用GitHub第三方授权的方式进行登录。
2. 记录用户行为。将用户上传文件、添加数据库表生成记录，写入数据库。

用户模块需求分析用例图如下图所示

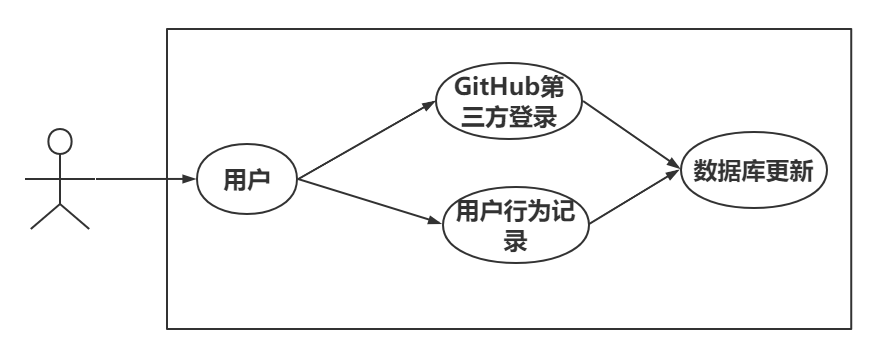


图 3.1用户模块需求分析用例图

### 数据源接入模块

1. 多数据库连接。支持PostgreSQL和MySQL两种数据库的连接。
2. 多文件上传。支持多文件上传。
3. 多类型文件上传。支持Excel和Csv两种类型文件的多文件上传。
4. 查询数据库中所有表。查询指定库中所有的表名。

数据源接入模块需求分析用例图如下图所示

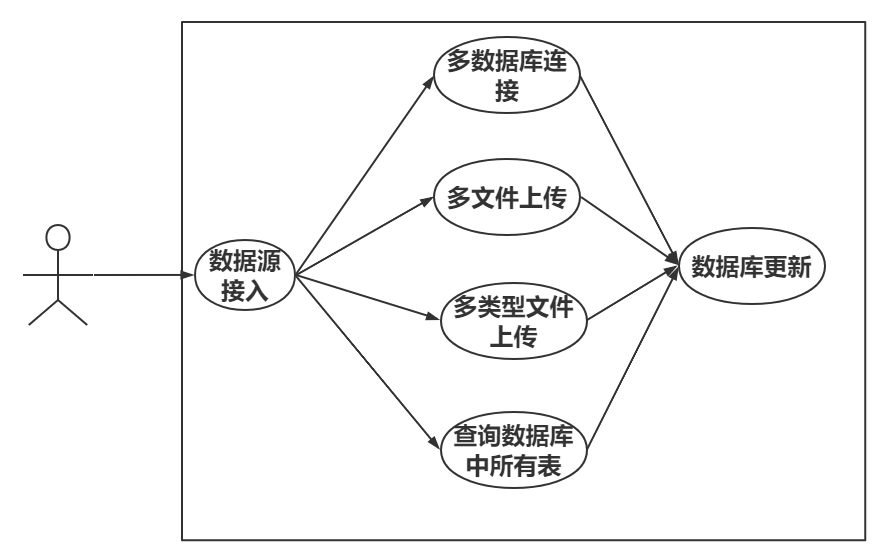


图 3.2数据源接入模块需求分析用例图

### 数据预处理模块

1. 字段分类。将字段分为三类：数值、日期、其他。
2. 查询指定字段所有数据。查询指定字段所有数据。根据传入的连接对象查询所需字段的数据，连接对象中包括数据表和该表中需要查询数据的字段。
3. 自定义构建计算指标。自定义构建计算指标。根据传入需要计算的算式，首先查询算式需要的字段数据，然后根据算式需求和算术符号的优先级，返回算数结果。

数据预处理模块需求分析用例图如下所示

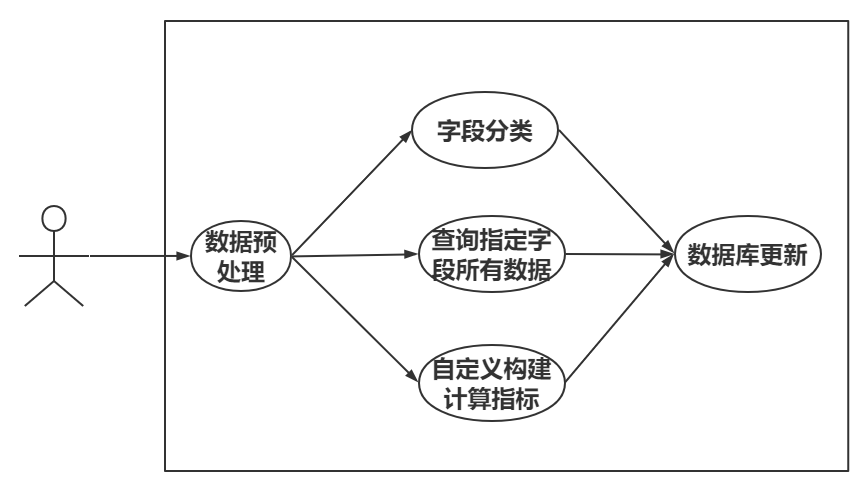


图 3.3数据预处理模块需求分析用例图

### 数据分析/计算模块

1. 数据分类聚合功能。将数据以维度分类，按指标聚合。
2. 脏数据处理。将表中不能解析的空值处理为可解析状态。

数据分析/计算模块需求分析用例图如下图所示

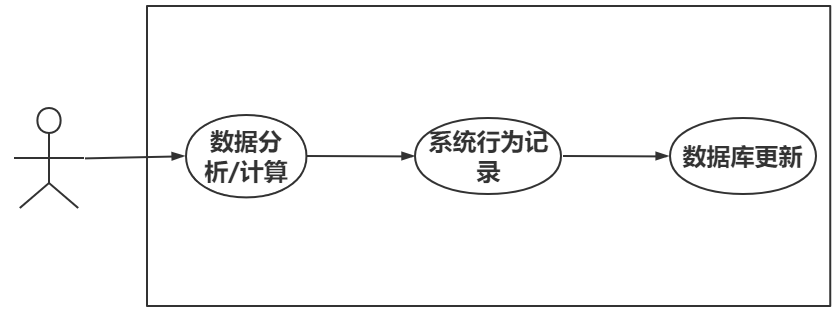


图 3.4数据分析/计算模块需求分析用例图

### 系统维护模块

1. 记录日志。生成日志，记录系统行为。

系统维护模块需求分析用例图如下图所示

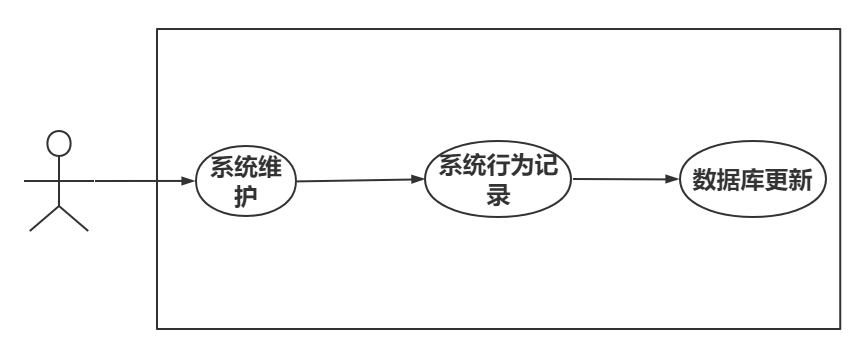


图 3.5系统维护模块需求分析用例图

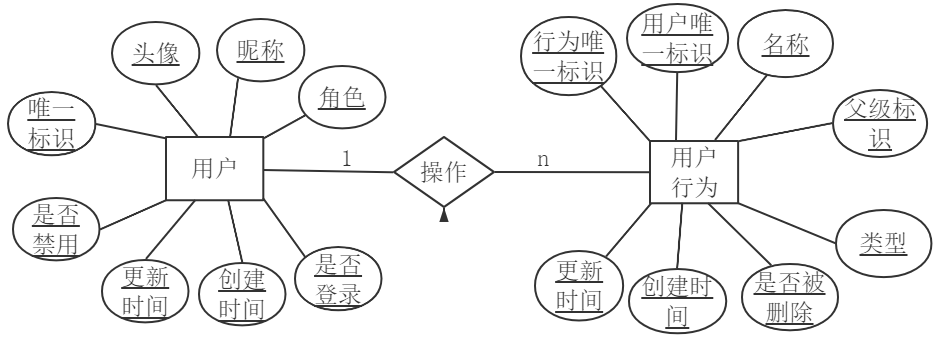
# 系统概要设计

根据上文中对系统的需求分析，本章要对系统有整体的概要设计，首先是数据库的设计其中包括系统实现需要使用的数据库表，其次是系统的日志存储，日志存储可以详细的记录出系统运行打印的信息，是日后系统出错时检查问题的重要线索，最后涉及的是系统各模块的接口设计。

## 数据库设计

整体E-R图如下图所示

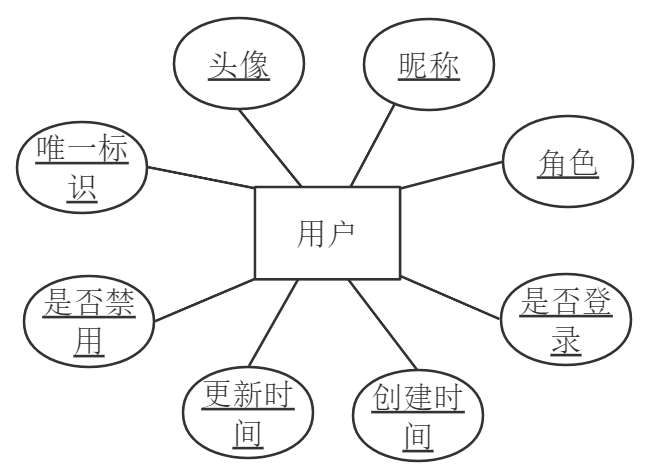
图 .1 整体E-R图



### 用户表设计

1. 用户实体。用户唯一标识、头像、昵称、角色、是否登录等。
2. 实体属性图如图下所示。

图 .2 用户实体属性图

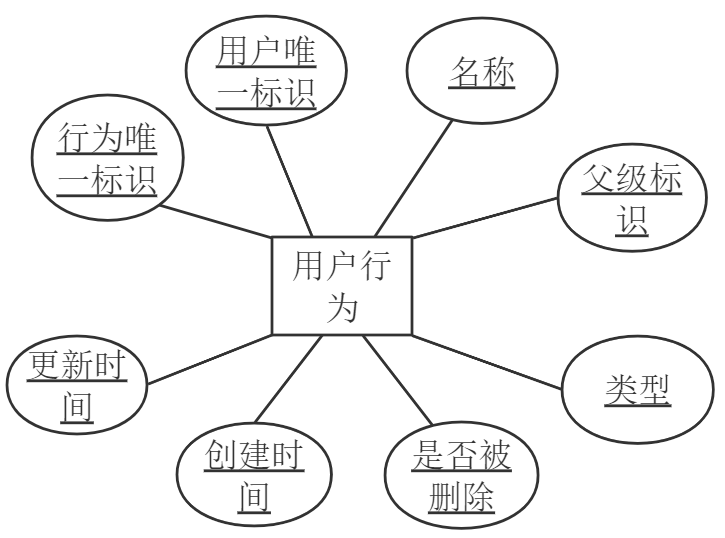


1. 数据表如图xx所示。

### 用户行为表设计

1. 用户行为实体。行为唯一标识、名称、用户唯一标识、类型、父级编码等。
2. 实体属性图如下图所示。

图 .3用户行为实体属性图



1. 数据表如图xx所示。

## 系统日志存储设计

### 设计原因

日志是一种可以追踪某些软件运行时所发生事件的方法。软件开发人员可以向他们的代码中调用日志记录相关的方法来表明发生了某些事情。一个事件可以用一个可包含可选变量数据的消息来描述。此外，事件也有重要性的概念，这个重要性也可以被称为严重性级别（level）。

通过log的分析，可以方便用户了解系统或软件、应用的运行情况；如果项目的log足够丰富，也可以分析以往用户的操作行为、类型喜好、地域分布或其他更多信息，由此可以实现改进业务、提高商业利益；如果一个应用的log同时也分了多个级别，那么可以很轻易地分析得到该应用的健康状况，及时发现问题并快速定位、解决问题，补救损失。

我们通过记录和分析日志可以了解一个系统或软件程序运行情况是否正常，也可以在应用程序出现故障时快速定位问题。日志的作用可以简单总结为1. 程序调试；2. 了解软件程序运行情况，是否正常；3. 软件程序运行故障分析与问题定位。

### 所选技术框架

在此项目中，我们使用标准的 Python logging 作为日志框架。所有与 Flask 相关的消息都用 app.logger 来记录，其名称与 app.name 相同。这个日志记录器可用于日志记录。

logging模块定义的函数和类为应用程序和库的开发实现了一个灵活的事件日志系统。logging模块是Python的一个标准库模块，由标准库模块提供日志记录API的关键好处是所有Python模块都可以使用这个日志记录功能。

### 实现过程

* 1. 配置日志的打印路径和编码
  2. 格式化日志记录的形式和内容
  3. 将日志配置对象挂在到 Flask 中

实现代码如下图所示

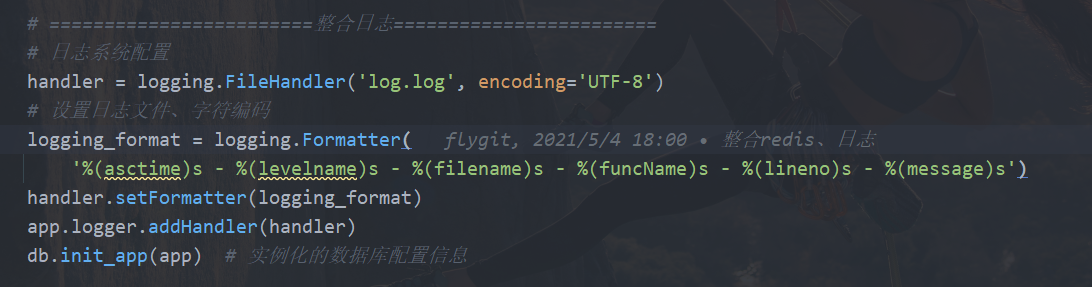


图 4.4 日志整合代码图

### 效果展示

此次项目中，日志记录了所有前端访问接口的行为。详细记录了用户访问的具体接口信息、响应时长及返回值的部分数据。日志文件中的部分数据如下图所示

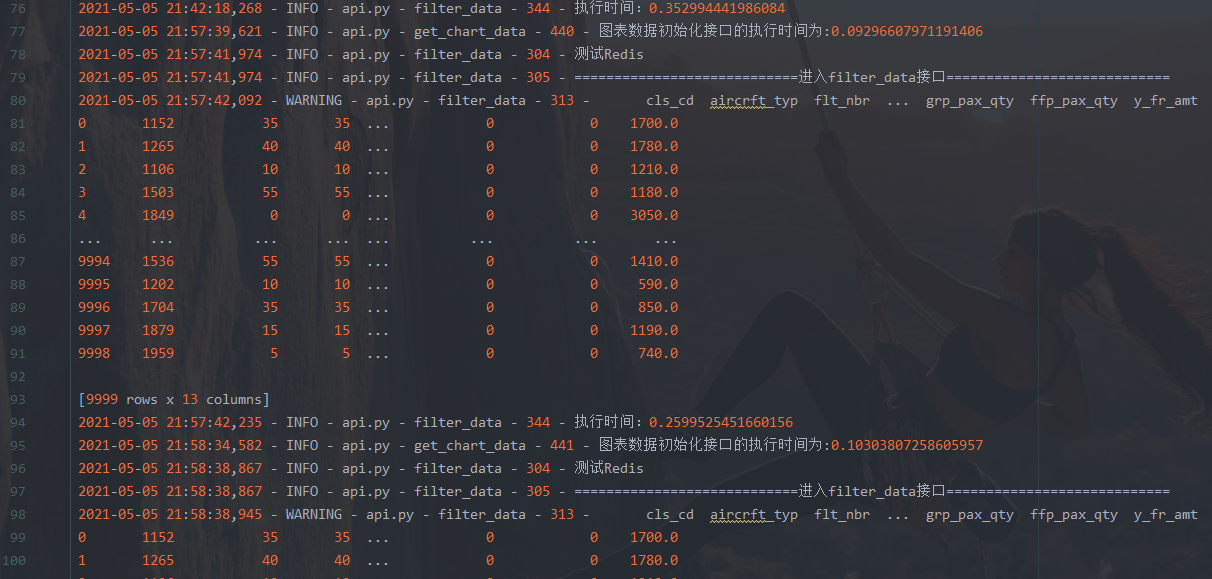


图 4.5 日志文件的部分数据图

## 系统接口设计

### 用户模块接口设计

用户登录接口请求参数如表4.1所示

表 4.1 登录接口请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| code | string | GitHub返回的code |

用户登录接口返回参数如表 4.2 所示

表 4.2 登录接口返回参数

|  |  |  |
| --- | --- | --- |
| 状态码/code | 状态信息/msg | 返回数据/data |
| 200 | 成功 | 用户信息 |

记录用户行为接口请求参数如表4.3所示

表 4.3 用户行为接口请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| user\_id | int | 用户唯一标识 |
| action\_id | int | 行为唯一标识 |
| table\_type | int | 添加表类型  (1 数据包、2 数据库表  3 上传文件表、 4 自定义表) |
| name | string | 名称 |
| parent\_id | int | 父级编码 |

记录用户行为接口返回参数如表 4.4 所示

表 4.4 记录用户行为接口返回参数

|  |  |  |
| --- | --- | --- |
| 状态码 | 状态信息 | 返回参数 |
| 200 | 成功 | 用户行为信息 |

### 数据源接入模块接口设计

数据库连接接口请求参数如下表所示

表 4.5 数据库连接接口请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| sqlType | string | 数据库类型 |
| userName | string | 数据库用户名 |
| password | string | 数据库密码 |
| host | string | 数据库地址 |
| port | int | 数据库端口 |
| database | string | 数据库名 |

数据库连接接口返回参数如下表所示

表 4.6 数据库连接接口返回参数

|  |  |  |
| --- | --- | --- |
| 状态码/code | 状态信息/msg | 返回数据/data |
| 200 | 成功 | 用户信息 |

### 数据预处理模块接口设计

获取指定表的所有字段并划分成数字型和非数字型的形式返回接口请求参数如下表所示

表 4.7 获取指定表的所有字段并划分成数字型和非数字型的形式返回接口请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| tableName | string | 表名 |
| sqlType | string | 数据库类型 |
| userName | string | 数据库用户名 |
| password | string | 数据库密码 |
| host | string | 数据库地址 |
| port | int | 数据库端口 |
| database | string | 数据库名 |

获取指定表的所有字段并划分成数字型和非数字型的形式返回接口返回参数如下表所示

表 4.8 获取指定表的所有字段并划分成数字型和非数字型的形式返回接口返回参数

|  |  |  |
| --- | --- | --- |
| 状态码/code | 状态信息/msg | 返回数据/data |
| 200 | 成功 | 用户信息 |

图表数据初始化接口请求参数如下表所示

图 4.6 图表数据初始化接口请求参数表

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| page | int | 页数 |
| limitCount | int | 每页展示的数据条数 |
| tableName | string | 表名 |
| sqlType | string | 数据库类型 |
| userName | string | 数据库用户名 |
| password | string | 数据库密码 |
| host | string | 数据库地址 |
| port | int | 数据库端口 |
| database | string | 数据库名 |

图表数据初始化接口返回参数如下表所示

图 4.7 图表数据初始化接口返回参数表

|  |  |  |
| --- | --- | --- |
| 状态码/code | 状态信息/msg | 返回数据/data |
| 200 | 成功 | 用户信息 |

### 数据处理/分析模块接口设计

查询数据接口请求参数如下表所示

表 4.9 查询数据接口请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| columnName | array | 列名数组 |
| tableName | string | 表名 |
| sqlType | string | 数据库类型 |
| userName | string | 数据库用户名 |
| password | string | 数据库密码 |
| host | string | 数据库地址 |
| port | int | 数据库端口 |
| database | string | 数据库名 |

查询数据接口返回参数如下表所示

表 4.10 查询数据接口返回参数

|  |  |  |
| --- | --- | --- |
| 状态码 | 状态信息 | 返回数据 |
| 200 | 成功 | 索引 |

数据分类聚合接口请求参数如下表所示

表 4.11数据分类聚合接口请求参数

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 说明 |
| allDataListIndex | int | 查询数据接口返回的索引 |
| allColNameList | string | 表中所有列名的数组 |
| colNameList | string | 指定的部分列名数组 |

数据分类聚合接口返回参数如下表所示

表 4.12 数据分类聚合接口返回参数

|  |  |  |
| --- | --- | --- |
| 状态码 | 状态信息 | 返回数据 |
| 200 | 成功 | 聚合后的数据 |

# 系统详细设计

1完成了概要设计就可以进行系统的详细设计，首先是对用户模块的设计，其中包含了用户使用系统需要进行的操作；其次是数据源接入，数据源是整个系统的灵魂，没有数据源整个系统就是一个空壳，正确的接入数据源整个平台才能正常的运作；接着就是数据预处理模块，包括数据的展示、拆分、聚合、构建计算指标等；然后是数据处/分析理模块对数据进行用户指定的处理和分析；最后是系统维护模块，为整个系统的正常运行保驾护航。

## 用户模块

### GitHub第三方登录功能设计

首先在GitHub上配置Home地址和回调地址（当前接口地址），前端先请求GitHub接口，GitHub会返回给后端接口一个code，再将code拼接到GitHub第三方的请求用户授权的接口尾部进行请求，请求成功后，会拿到一个openId，将openId拼接到请求用户信息的接口进行请求，成功后GitHub会返回用户信息。GitHub第三方登录时序图如下图所示。

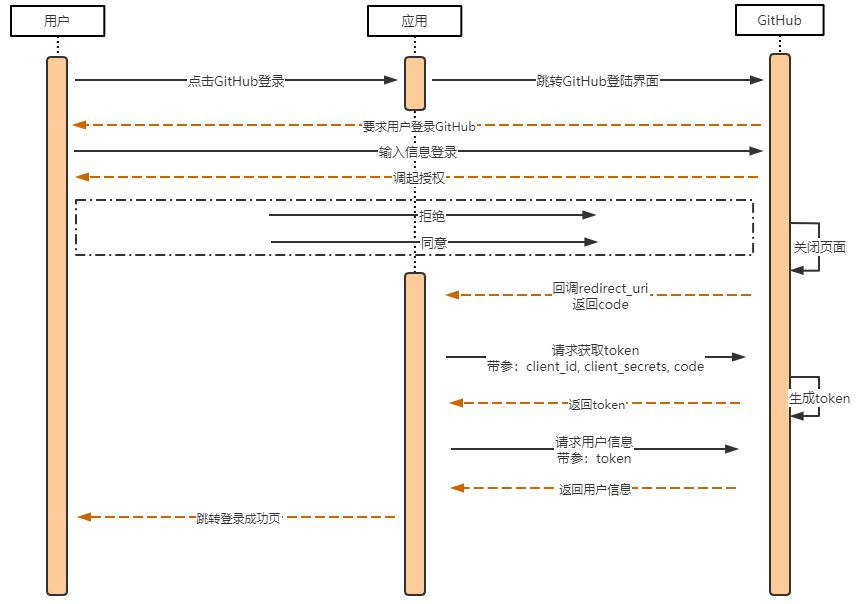


图 .1 GitHub第三方登录时序图

### 记录用户行为功能设计

将用户创建数据包，添加表行为写入数据库，写入参数模型如下表所示。

表 5.1写入参数模型

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用户添加表类型 | 行为唯一标识 | 用户唯一标识 | 名称 | 类型 | 父级编码 |
| 数据包 | 1 | 用户id | 数据包名称 | 1 | 0 |
| 数据库表 | 2 | 用户id | 数据库表名称 | 2 | 对应数据包的行为唯一标识 |
| 上传文件表 | 3 | 用户id | 上传文件表名 | 3 | 对应数据包的行为唯一标识 |
| 自定义表 | 4 | 用户id | 自定义表名 | 4 | 对应数据包的行为唯一标识 |

## 数据源接入模块

### PostgreSQL、MySQL数据库连接功能设计

在此功能上，我们需要对用户所选的数据库进行连接测试功能，无需进行数据访问。目前项目中，我们只实现了对 PostgreSQL、MySQL 两类数据库的连接功能。接下来将详细介绍对这两类数据库的连接功能设计。第一步：用户需要通过客户端将一些必备的参数传递给后端 API 接口，例如：所选择的数据库类型（PostgreSQL 或者 MySQL）、数据库的连接用户名、数据库的连接密码、数据库的主机地址、数据库的连接端口、所连接的数据库名等信息。第二步：接口需要根据对所选择的不同数据库连接类型进行划分，单独连接 PostgreSQL 和单独连接 MySQL 数据库；第三步：接口需要通过判断连接过程中是否抛出异常来判断连接是否成功，如果连接成功，接口返回“连接成功”的数据。反之。PostgreSQL、MySQL数据库连接时序图如下图所示。

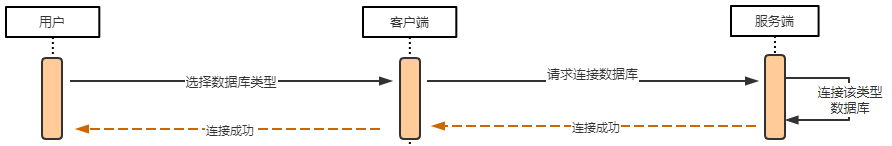


图 5.2 PostgreSQL、MySQL多数据库连接时序图

### Excel、Csv多文件上传功能设计

在前端规范好文件上传的类型，将用户上传的文件信息通过formdata的形式传给后端，在对文件进行文件的读取时，需要对文件类型进行判断，不同的文件类型需要使用不同的文件读取方式。Excel使用的文件读取方法是read\_excel，CSV使用的文件读取方式是read\_csv；在文件读取过程中，需要对文件中的处理空值进行，以防止前端对接口的调用和数据展示出现错误。

### 查询数据库中所有表功能设计

## 数据预处理模块

### 字段分类功能设计

### 查询并过滤数据表中所有数据功能设计

### 查询数据库中所有表功能设计

### 查询指定字段数据功能设计

### 自定义构建计算指标功能设计

## 数据处理/分析模块

### 数据处理/分析功能设计

数据处理/分析时序图如下图所示。

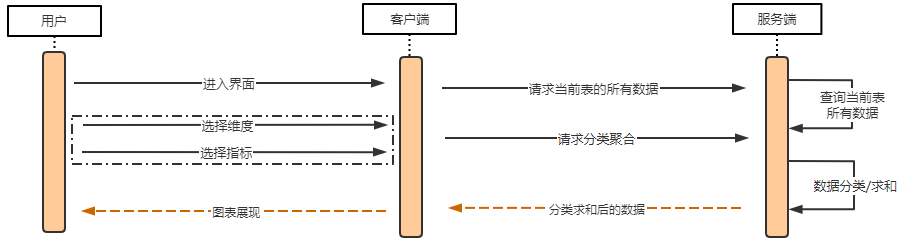


图 5.3数据处理/分析时序图

### 脏数据处理功能设计

表中的空值不能被解析，将空值替换为“null”字符串。

## 系统维护模块

### 日志记录系统行为功能设计

### 统一响应体设计

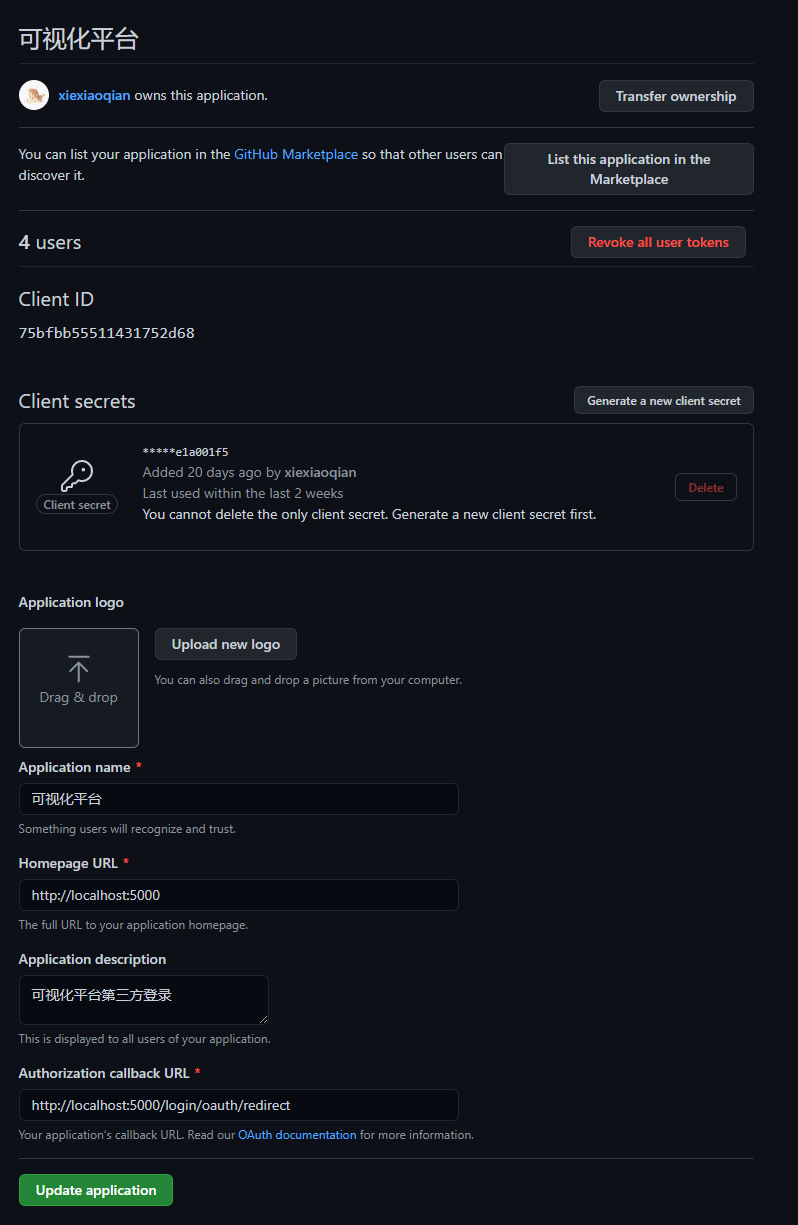
# 系统功能的实现

进行完详细的设计，就要着手实现，本章论述的就是各个模块详细功能的具体实现。

## 用户模块

### GitHub第三方登录功能实现

1. 首先在GitHub上配置平台开放授权，如图xx所示。



1. Home地址：后端Home地址，<http://localhost:5000>。
2. 平台名称：可视化平台。
3. 回调地址：接收openId的接口地址，<http://localhost:5000/login/oauth/redirect>。

配置成功后会生成client\_id和client\_secret。

1. 接口部分
   1. 用request.args.get('code')方法接收到GitHub返回的code
   2. 将client\_id和client\_secret拼接到获取token接口的尾部，作为完整地址即token\_url，设置json请求头，使用requests.post(token\_url, headers=header)方法，请求获取token的接口，方法将会返回请求结果，用res变量接收。
   3. 使用res.json()方法序列化请求结果，以字典的形式拿出access\_token。
   4. 将access\_token作为请求头，使用requests.get(user\_url, headers=headers)方法，请求获取用户信息的接口即user\_url，用res变量接收方法返回的请求结果。
   5. 使用res.json()方法序列化请求结果通过字典拿出用户信息，比如：id、昵称、头像等。

### 记录用户行为功能实现

待实现

## 数据源接入模块

### PostgreSQL、MySQL数据库连接功能实现

### Excel、Csv多文件上传功能实现

### 查询所选连接数据库所有表功能实现

## 数据预处理模块

### 字段分类功能实现

### 查询并过滤数据表中所有数据功能实现

### 查询数据库中所有表功能实现

### 自定义构建计算指标功能实现

### 指定数据表查询所有字段功能实现

## 数据处理/分析模块

### 数据处理/分析功能实现

### 脏数据处理功能实现

## 系统维护模块

### 日志记录系统行为功能实现

### 统一响应体实现

# 系统测试

系统测试是一个系统完成不可或缺的部分，本章从测试环境准备开始论述，然后功能性测试结果的展现，最后是将系统部署到远程服务器上进行测试，只有测试出正常的测试结果，系统的使用的稳定性和安全性才更有保障。

## 测试环境准备

## 功能性测试

## 系统部署测试

# 总结和展望

完成系统测试，整个论文也就接近尾声，本章是对系统的总结和对系统未来发展方向的展望。

## 总结

## 展望

# 致谢

参 考 文 献

[1] 电力公司电力标准化作业管理系统的设计与实现\_马小越[J].