

班 级 1703019
学 号 17030199018

西安电子科技大学

本科毕业设计论文



题 目 基于微服务架构的敏捷

开发平台的设计与实现

学 院 计算机科学与技术学院

专 业 计算机科学与技术专业

学生姓名 朱健行

导师姓名 鱼滨

毕业设计（论文）诚信声明书

本人声明：本人所提交的毕业论文《基于微服务架构的敏捷开发平台的设计与实现》是本人在指导教师指导下独立研究、写作成果，论文中所引用他人的无论以何种方式发布的文字、研究成果，均在论文中加以说明；有关教师、同学和其他人员对本文本的写作、修订提出过并为我在论文中加以采纳的意见、建议，均已在我致谢辞中加以说明并深致谢意。

本文和资料若有不实之处，本人承担一切相关责任。

论文作者：_____ (签字) 时间： 年 月 日

指导教师已阅：_____ (签字) 时间： 年 月 日

摘 要

伴随着云计算、Docker 容器化技术的发展，微服务架构在全球范围内被广泛应用，其理念是细粒度模块划分、服务化接口封装、轻量级通信交互，具有模块自治性强和扩展性好等显著优点。

对于个人开发者，往往缺少成熟、集成的敏捷开发工具。考虑到数据库相关功能是项目数据存储、交互的关键，同时也是最基础的环节，故决定围绕数据库建立一个功能集成的、基于微服务架构的敏捷开发平台。

本文采用软件工程学方法，按照系统需求分析、系统详细设计、编码、测试等软件开发流程，对软件的设计实现过程进行了详细的论述。

通过验证，本平台基于 OAuth2.0 协议提供了完整的认证和鉴权服务，实现了数据库增删改查代码的自动生成、数据库设计文档的自动生成、历史文档的版本管理、数据表数据批量导入/导出等功能。简化了软件设计、开发、测试等流程，有效提高了用户的开发效率。

关键词：微服务 敏捷开发 数据库 OAuth2.0 自动生成规则

摘要

Abstract

With the development of cloud computing and Docker containerization technology, microservice architecture is widely used worldwide. Its concept is fine-grained module division, service-oriented interface encapsulation, lightweight communication interaction, and It has obvious advantages such as strong module autonomy and good scalability.

For individual developers, there is often a lack of mature and integrated agile development tools. Considering that database-related functions are the key to the data storage and interaction of the project, and also the most basic link, it was decided to build a function-integrated agile development platform based on the microservice architecture around the database.

This article adopts software engineering method, according to the software development process of system requirement analysis, detailed system design, coding, testing, etc., the design and realization process of software is discussed in detail.

Through verification, the platform provides a complete authentication service based on the OAuth2.0 protocol, and realizes the automatic generation of database addition, deletion, modification and query code, automatic generation of database design documents, version management of historical documents, and batch import/export of data tables. Simplifies the software design, development, testing and other processes, effectively improving the user's development efficiency.

Key words: Microservice Agile Development Database OAuth2.0 Automatically Generate Rules

Abstract

目 录

第一章 绪论	1
1.1 研究背景及意义	1
1.2 国内外发展现状	2
1.2.1 微服务研究现状	2
1.2.2 敏捷开发平台研究现状	2
1.3 本文的主要工作	3
1.4 本文的组织结构	3
1.5 本章小结	4
第二章 项目关键技术分析	5
2.1 微服务框架 Spring Cloud	5
2.2 API 网关 Spring Gateway	6
2.3 OAuth2.0 和 Spring Security	8
2.4 NoSQL 键值数据库 Redis	9
2.5 阿里云 OSS	11
2.6 模板引擎 Velocity 和 Poi-tl	11
2.7 B/S 架构	12
2.8 前端构建框架 Vue 和组件库 Element UI	13
2.9 本章小结	14
第三章 项目需求分析	15
3.1 项目概述	15
3.2 项目功能性需求	17
3.3 项目非功能性需求	19
3.4 本章小结	20
第四章 项目设计与实现	21
4.1 项目总体设计	21

4.2 数据存储层详细设计	22
4.2.1 关系型数据库 MySQL	22
4.2.2 非关系型数据库 Redis	24
4.3 服务层通用设计	25
4.3.1 服务注册与发现	25
4.3.2 分布式全局配置	26
4.3.3 API 网关服务	26
4.3.4 服务调用与声明式接口	26
4.4 服务层通用设计	26
4.4.1 类结构层次设计	26
4.4.2 响应消息体设计	27
4.4.3 提取公共模块	27
4.4.4 统一认证鉴权	27
4.4.5 数据库相关对象主体类设计	27
4.5 项目模块核心功能详细设计	29
4.5.1 注册登录模块详细设计	29
4.5.2 数据库代码自动生成模块详细设计	31
4.5.3 数据库描述文档自动生成模块详细设计	35
4.5.4 文件版本管理模块详细设计	36
4.5.5 围绕数据库的服务组件库模块详细设计	36
4.6 本章小结	37
第五章 项目测试与结果展示	39
5.1 项目测试环境	39
5.2 项目核心功能测试过程及结果	39
5.2.1 注册登录模块测试	39
5.2.2 数据库代码自动生成模块测试	39
5.2.3 数据库描述文档自动生成模块测试	40
5.2.4 文件版本管理模块详细设计测试	40
5.2.5 围绕数据库的服务组件库模块测试	40
5.3 本章小结	41

第六章 总结与展望	43
6.1 工作总结	43
6.2 工作展望	43
6.2.1 数字转中文	44
6.3 表格	44
6.3.1 普通表格	44
6.3.2 复杂点的表格	45
6.3.3 长表格	45
6.4 图片	47
6.4.1 普通图片的插入	47
6.4.2 图片并排插入	47
6.5 公式	48
6.5.1 普通公式	48
6.5.2 复杂公式	48
6.6 休息一下	49
6.6.1 山水之间	49
6.6.2 念奴娇·赤壁怀古	50
第七章 两个重点	51
7.1 环境	51
7.1.1 定理类环境	51
7.1.2 算法环境	52
7.1.3 代码环境	52
7.2 参考文献的引用	53
致谢	55
参考文献	57
附录 A 数据	59
A.1 放松一下	59
A.1.1 惊鸿一面	59
A.1.2 无题	60
A.2 代码	60

第一章 绪论

1.1 研究背景及意义

互联网时代，信息爆炸，网络用户数量快速增长，系统负载承受考验，同时，用户需求不断增多，系统支持的业务需要不断地进行需求迭代。当系统性能达到瓶颈，传统的单体架构往往采用增加计算机硬件方式过渡，但是长远来看，系统会逐渐变得臃肿，模块之间耦合度过高、开发低效、后期维护困难，无法根本性地解决问题。显然，传统单体应用架构已经不能很好的帮助企业应对市场和需求灵活的变化。于是，一种新的软件设计思想和架构应运而生——微服务^[1]。伴随着云计算、Docker 容器化技术的发展，微服务架构在全球范围内被广泛应用^[2,3]，其理念是细粒度模块划分、服务化接口封装、轻量级通信交互，具有模块自治性强和扩展性好的显著优点。

敏捷开发^[4,5]是一种软件开发方式，以用户的需求为核心，主张简单，拥抱变化，采用循序渐进的方法进行软件快速迭代、交付。在软件项目的构建初期，把一个项目划分为多个可独立运行但相互联系的小项目，并分别开发，在此过程中软件一直处于可用状态。

对于个人开发者和小型企业开发团队，往往缺少成熟、集成的服务端快速开发的工具资源，开源项目也很少。根据课题任务书，考虑到数据库相关功能是项目数据存储、交互的关键，同时也是最基础的环节，故决定围绕数据库建立一个功能集成的敏捷开发平台，完成代码、文档的自动生成，历史文档版本的管理，以及其他服务组件库。简化开发者的设计、开发、测试等流程，使开发者快速进入开发，专注于自身业务逻辑的实现，提高开发效率。

1.2 国内外发展现状

1.2.1 微服务研究现状

微服务架构中，单个应用程序被开发为一组小型服务，即微服务，每个微服务都是自治的，运行在自己的独立进程中，并使用轻量级的通信机制进行服务间通信，具有易于维护、高容错、可扩展性强、技术异构、微服务模块间松耦合、独立部署等优点^[6]。

目前，大部分传统企业仍然处在单体应用向微服务应用过渡的阶段。但是国内外的互联网企业都已经早早转向微服务架构：

在国外，Uber 公司为应对整合新业务和迭代旧业务中遇到的困难，改用微服务架构，从模块功能维度将应用拆分为基于云的微服务。转向微服务架构后，服务访问的速度和质量得到提高，团队只需关注需要扩展的业务，服务独立部署实现了更可靠的容错。作为微服务架构践行的先行者，亚马逊开发了几种支持微服务架构的解决方案，比如亚马逊 AWS 云服务。

在国内，微服务架构同样发展迅猛：阿里巴巴公司的阿里云是全球领先的云计算产品，它很好的解决了阿里巴巴大量用户和流量带来的负载问题，承载了每年的“双十一”、春运售票等活动的高并发访问，提供解决方案，以实践证明了阿里云的强大实力。

1.2.2 敏捷开发平台研究现状

关于敏捷开发平台，目前没有一个统一的定义，从不同的需求出发，各个平台实现的功能也不尽相同。但一般认为敏捷开发平台旨在抽象出项目开发中的横切模块，简化开发、测试、运维等流程，使开发者专注于自身业务逻辑的实现，快速进入开发，提高开发效率。

在国内外，许多中大型互联网公司都有自己的企业级的敏捷开发平台。例如字节跳动公司内部研发了基于微服务架构和轻量级容器技术的字节云开发平台，将项目的开发、部署、运维、测试等有机整合为一体，并提供了独立部署、高容错、可扩展等特性。

此外，还有 JeeWeb 敏捷开发平台，主要定位于快速开发平台建设，内置很多优秀的基础功能，包括系统权限组件、数据权限组件、数据字典组件、代码生成、

UI 模版标签库等。不管是中小型企业团队还是个人开发者都能利用功能组件，快速进入开发。

1.3 本文的主要工作

本文介绍了微服务架构和敏捷开发平台的发展现状，列举了一些国内外同类软件的研究成果。详细介绍了 Spring 系列开发框架、OAuth2.0 安全协议、B/S 架构模型、Redis 存储技术、模板引擎等关键技术和理论，以及不同技术选型的对比和抉择标准。

基于课题任务书以及结合本人在字节跳动实习期间了解的项目开发规范和需求，对项目进行需求分析，决定围绕数据库展开功能，完成代码、文档的自动生成，历史文档版本的管理，以及其他服务组件库。通过对分析目前成熟的项目架构和可行的技术，提出项目总体架构、核心功能的解决方案：

本项目采用 B/S(Browser/Server) 结构实现，在服务层部署 Tomcat 服务器，基于 Spring Cloud 框架实现微服务架构，SpringBoot 框架构建微服务模块，Gateway 组件构建全局统一网关，以功能维度划分微服务模块，以 Nacos 组件作为服务注册中心，以 OpenFeign 组件进行服务发现。在展示层，基于 Vue 框架构建前端 Web 前端应用。在数据层，选择 MySQL 作为底层数据库，完成结构化数据的持久化存储，选择非关系型数据库 Redis 作为缓存。选择使用阿里云 OSS 作为对象存储服务器，存储用户生成的文件。基于 Velocity 和 Poi-tl 模板引擎实现代码、文档自动生成核心功能。

1.4 本文的组织结构

本文介绍了基于微服务架构的敏捷开发平台的设计与实现。首先对理论内容和应用技术进行介绍，分析系统需要的功能需求，再对软件的总体设计实现进行介绍。同时对软件的各模块进行了细致的分析和设计，并给出相应的程序流程图和状态图。最后对开发的软件进行了测试，并给出测试结果。

本课题分为以下六个章节，简要阐述如下：

第一章是绪论：本章主要对本课题的背景进行介绍，以及论文所涉及的工作和章节结构。

第二章是项目关键技术分析：本章主要介绍本课题涉及的关键技术和理论，包括前后端开发框架、网关技术、OAuth2.0 安全协议、Redis 存储技术、Velocity 和 Poi-tl 两种模板引擎以及不同技术选型的对比和本项目选型标准等。

第三章是项目需求分析：本章对敏捷开发系统进行了具体的需求分析。以软件工程知识为理论基础，结合本人在北京字节跳动科技有限公司的实习工作经验和了解到的开发者工作的痛点，对系统的功能性需求和非功能性需求进行分析，并给出了功能结构图和用例图，为以后的工作奠定基础。

第四章是项目设计与实现：在需求分析完成后，本章进行了具体的软件系统设计。提出项目的总体框架，以及系统存储层 MySQL 数据表结构设计和 Redis 键值对设计。针对各个模块的核心功能进行说明与详细设计实现，并提出相应的流程图。

第五章是项目测试：本章介绍了软件测试环境、测试过程和测试结果，对核心功能的实现给出关键性截图并说明。

第六章是总结与展望：本章总结全文工作内容，分析课题工作中的问题以及对未来工作方向的展望。

1.5 本章小结

本章主要介绍了项目的研究背景及意义、分析了微服务架构和敏捷可开发平台的发展现状，概述了各章节的研究内容，总结了本文的结构。

第二章 项目关键技术分析

2.1 微服务框架 Spring Cloud

微服务是面向服务的体系结构（SOA）样式的一种变体，将应用程序构造为一组松散耦合的服务。微服务架构^[7] 如图2.1所示，从实际业务出发，服务是细粒度的，单个应用程序由许多松散耦合且可独立部署的较小组件或服务组成；协议是轻量级的，服务与服务之间的调用通常采用轻量级的通信机制，如基于 RPC 或是 HTTP 的 RESTful API^[8]。

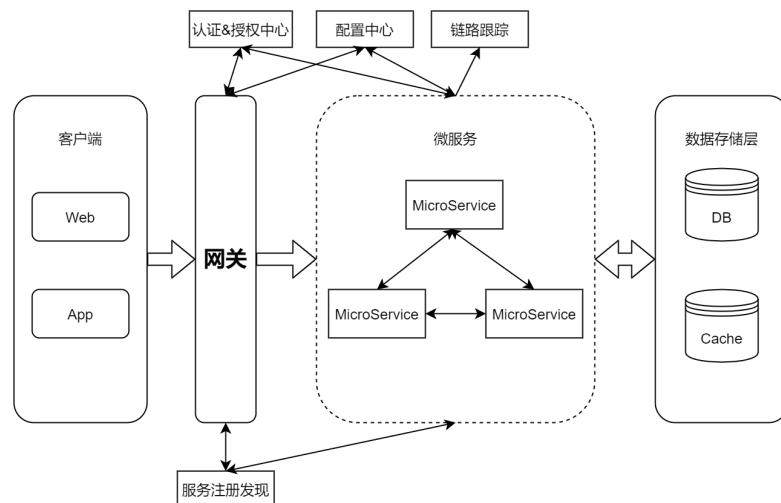


图 2.1 微服务架构

与传统的面向服务架构模式相比，微服务架构具有以下特征和优势：

- 1) 单一职责：每个微服务都需要满足单一职责原则，微服务本身是内聚的，因此微服务通常比较小，一个小规模团队就可以专注于开发一个微服务，可以有效控制项目的复杂度，提高团队的开发效率和应用程序的可维护性。
- 2) 独立部署：微服务之间具有明显的边界，每个微服务都是一个独立自治的实体，有独立的进程和运行环境。当需求迭代、修改漏洞导致代码变更，微服务模块可以独立完成编译、部署和运维进行项目升级，大大降低对整个系统可用性影响。

3) 高容错性：微服务间独立部署，一个微服务的异常不会导致其它微服务同时异常。通过隔离、融断等技术可以极大的提升微服务的可靠性。

4) 技术异构：在一个大型系统中，不同的功能具有不同的特点，并且不同的团队可能具备不同的技术能力。因为微服务间松耦合，不同的微服务可以根据项目实际需求选择不同的技术栈进行开发。

5) 可扩展性强：基于微服务自治和独立部署的特性，项目具有较强的水平扩展性，且技术选型灵活

近年来，随着微服务架构发展和盛行，许多微服务框架应运而生，在国内较为知名的是 Spring Cloud 系列框架^[9,10]。Spring Cloud 包含于 Spring 官方提供的 Spring 系列框架，是一系列框架组件的有序集合，如服务发现注册、配置中心、消息总线、负载均衡、断路器、链路跟踪等^[11]。Spring Cloud 中一些组件闭源了，不再维护，如注册中心 Eureka 在 2.0 版本之后宣布闭源了，容错组件 Hystrix、网关组件 Zuul 也已经宣布停止维护了。如果在生产环境出现了问题，官方不再维护，可能会造成生产事故。

Spring Cloud Alibaba 是在 Spring Cloud 基础上开发并开源的一套微服务架构体系，相比之下，Spring Cloud Alibaba 技术栈中的各个组件的开源社区活跃度高，且很多技术经历过实际生产考验，在性能上更具优势^[12]。根据以上分析，采用 Spring Cloud 框架，结合 Spring Cloud Alibaba 框架进行开发。

2.2 API 网关 Spring Gateway

在项目引入微服务框架前，我们需要思考，客户端如何和微服务系统进行交互^[13]。理论上来说，每个微服务都可以作为自治的个体直接给客户端提供服务，即传统面向服务架构的单体模式，但是这种直接通信的方式会存在一些问题：

1) 公共横切功能冗余：一些公用横切功能服务，如认证和鉴权，如果没有 API 网关，每个微服务模块都需要实现认证和鉴权功能，容易造成代码冗余和项目维护困难。

2) 迁移和重构困难：由于每个微服务接口直接暴露于外部，当项目存在迁移或重构，客户端很难发现服务端结构改变之后的访问地址。

3) 混合通信协议问题：虽然面向外部的 API 通常提供基于 HTTP 的 RESTful API，但是内部微服务可能使用不同的通信协议，如 AMQP、SOAP 和 RPC 协议

等。这种协议之间的差异性以及一些并不是 web 友好的通信协议可能会造成客户端的通信问题。基于以上问题分析，在引入微服务时，构建 API 网关是很有必要的。作为全局流量的入口，所有请求都需要通过网关进行路由转发，API 网关根据请求的 URL 地址，根据路由的映射规则，将请求转发到对应的微服务实例^[14-17]。本项目采用 API 网关构建微服务统一流量入口，具有以下显著优势：

- 1) 隐藏微服务内部细节：API 网关将外部公开 API 与内部微服务 API 分开，允许添加微服务和更改边界。通过为所有微服务提供单一入口，对客户端隐藏了服务发现和版本控制详细信息，可以防止内部信息暴露给外部。其结果是能够在不对外产生负面影响的情况下迁移和重构微服务。
- 2) 支持混合通信协议：API 网关可以在这些不同的协议之上提供给外部统一的 HTTP API，允许团队选择最适合内部架构的 API。
- 3) 降低微服务复杂性：如果若干微服务具有公共模块功能，如认证和授权，可以将公共模块的代码逻辑写在网关的过滤器中，如果想修改公共模块的逻辑，后端服务不需要关注公共功能模块的代码，即不需要升级所有已存在的微服务，同时可以允许团队开发者专注于微服务模块差异化的业务功能。

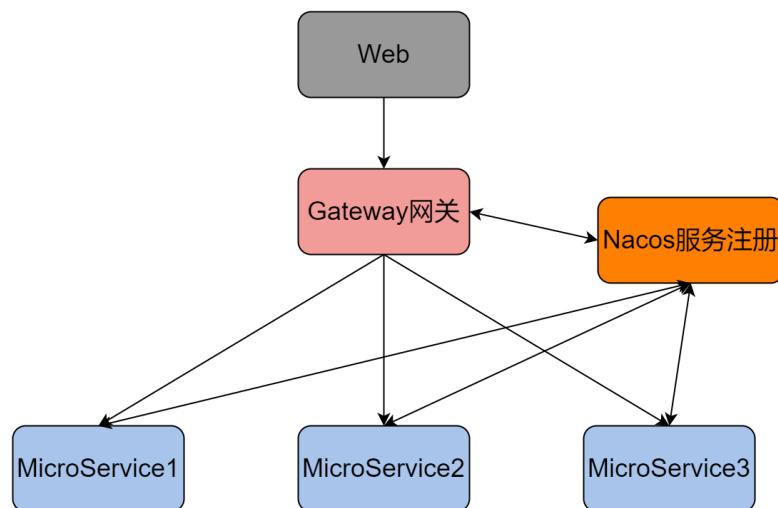


图 2.2 Spring Gateway 网关

Spring Gateway 是 Spring 官方推出的网关框架^[18]，图2.2展示了由 Spring Gateway 实现的 API 网关，跟 MicroService1、MicroService2 等其它微服务一样，Gateway 作为一个微服务，注册在 Nacos 注册中心上。注册了的微服务可以相互发现和调用。Gateway 能够通过过滤器（GatewayFilter）处理身份验证、Token 验证等公共预处理。服务端新增微服务时，API 网关层只需要添加路由规则即可。作为全局流

量的入口，Gateway 可以根据定义的路由转发规则，将客户端请求转发到指定的微服务上。

与 Gateway 功能相似的还有 Kong^[19]、Spring Cloud Zuul^[20] 等框架，考虑到项目架构的兼容性和扩展性，以及 Spring Zuul 停止维护的现状，最终选择和 Spring 系列框架兼容性更强、持续维护的 Spring Gateway 作为项目的 API 网关。

2.3 OAuth2.0 和 Spring Security

OAuth 是一个关于授权的网络标准。OAuth 在第三方应用（客户端）和资源服务器之间增设了一个授权服务器。第三方应用只能登录授权服务器获得授权，而不能直接登录资源服务器。授权服务在资源服务端的服务器上或是专门的认证服务器上。OAuth 协议使第三方应用在无法获取用户敏感信息（如用户密码）的条件下获得用户某些资源的权限。认证服务提供商会给第三方应用发放一个临时令牌，令牌具有权限范围和有效期，第三方应用可以携带令牌来获取用户资源^[21]。

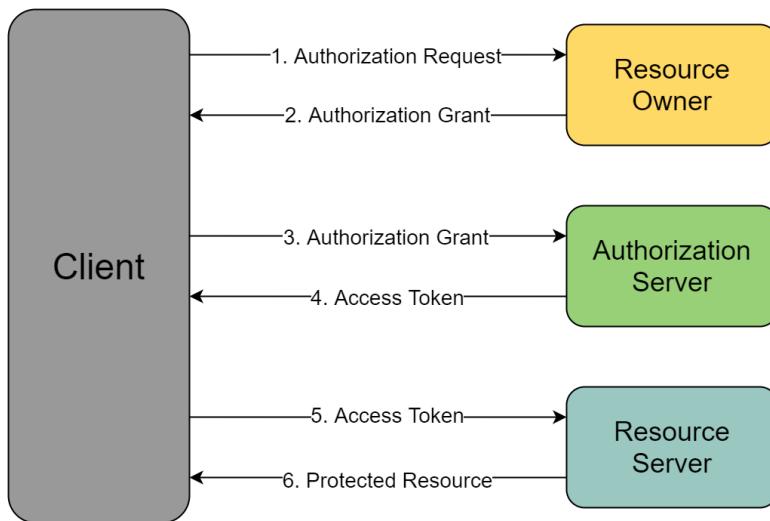


图 2.3 OAuth2.0 标准授权流程

OAuth 2.0 是目前最常见的安全访问协议，只要授权方和被授权方双方遵守这套协议规范，就实现了 OAuth 模式。如图2.3所示，OAuth2.0 的授权流程主要分为以下六个步骤：

- 1) 用户打开客户端，客户端向用户请求认证访问授权；
- 2) 用户同意授权给客户端；
- 3) 获得用户授权后，客户端向认证中心申请 Token 令牌；

- 4) 认证中心认证通过后，同意发放 Token 令牌给客户端；
- 5) 客户端携带 Token，向资源服务器申请获取资源；
- 6) 资源服务器验证 Token 的合法性，若合法，则给客户端开放资源访问权限。

Spring Security 是一个功能强大、高度可定制的身份验证和访问控制安全框架，由 Spring 官方开源。它的核心功能包含认证和授权等。认证即验证“你是谁”的问题，授权即确认“你能干什么”的问题。此外，它还提供加密、csrf 保护等功能等。

与 Spring Security 功能相似的主流框架还有 Shiro。考虑到 Spring Security 与项目现有 Spring 系列框架技术选型良好的兼容性，广泛的使用范围，以及活跃的开源社区，良好的可扩展性和可移植性，决定采用 Spring Security 框架。

2.4 NoSQL 键值数据库 Redis

NoSQL 最通用的解释是“non-relational”，此外，“Not Only SQL”也被很多人接受。NoSQL 仅仅是一个概念，区别于关系数据库，泛指非关系型的数据库，它们不保证关系数据的 ACID 特性。得益于 NoSQL 的无关系性，数据库的结构简单，NoSQL 数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀^[22,23]。

NoSQL 数据库可以分类为键值 (Key-Value) 存储数据库、列存储数据库、文档型数据库、图形数据库。考虑到本项目键值存储和缓存的实际需求，选用键值存储数据库。

在国内比较主流的非关系型键值存储数据库是 Redis 数据库和 MemCached 数据库，两者都具有很高的性能，在功能上相似。MemCached 为纯内存操作，数据都存储在内存中，断电或重启后内存中的数据消失。相比之下，Redis 具有良好的数据持久化机制，可以保证数据的可靠性和可用性。支持内存快照 (RDB) 模式，可以按照一定的周期把内存数据以快照形式保存为硬盘二进制文件。支持日志文件 (AOF) 模式，Redis 的每个写命令都追加到日志文件 appendonly.aof 上，当 Redis 服务宕机重启，会通过执行文件中保存的写命令来重建内存数据。

考虑到本项目对 NoSQL 的读写比较均衡，且需要数据的持久化存储，选用 Redis 作为 NoSQL 数据库。

Redis 具有很快的存取速度，分析原因如下：

1) Redis 是基于内存的存取，绝大部分请求都是纯内存操作，相比于对硬盘的存取要快很多。

2) Redis 采用单线程处理网络请求，可以减少上下文切换和竞争条件，因此不会因为多线程切换而消耗 CPU 等资源，不用考虑各种锁的问题，不存在加锁、释放锁操作，不会因为可能出现死锁而导致性能消耗。

3) Redis 使用非阻塞 I/O 多路复用机制，非阻塞指每个网络请求无需阻塞式持续等待，多路指多个网络请求，复用即复用同一个线程。通过引入非阻塞 I/O 多路复用机制，减少了网络 I/O 的时间损耗，提高了网络通信模型的性能，同时保证了 Redis 服务实现的简单和轻量，具体工作原理见下图。

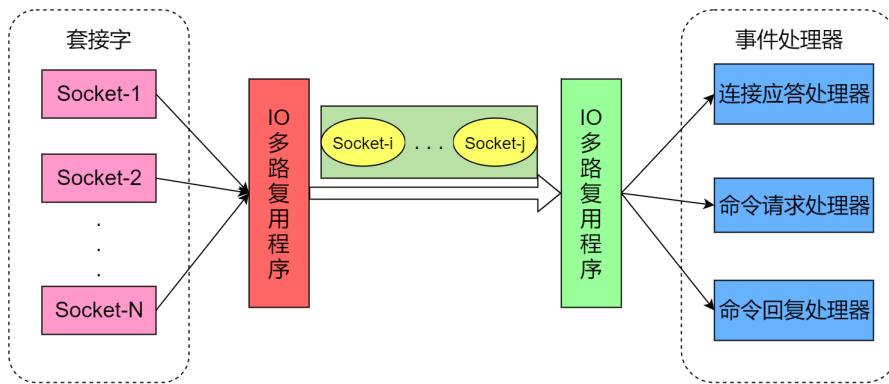


图 2.4 Redis 非阻塞 I/O 多路复用机制

由图2.4可见，IO 多路复用程序会把所有产生事件的 socket 压入一个队列中，然后有序地压出一个 socket 传送给文件事件分派器，文件事件分派器接收到 socket 之后会根据 socket 产生的事件类型调用对应的事件处理器进行处理。

在已有底层关系型数据库的基础上，引入 Redis 存储具有以下优势^[24]：

1) Redis 存取速度快，可以满足部分子功能即时性的需求。
2) Redis 作为缓存，可以减少客户端对底层关系型数据库的访问次数，缓解底层数据库的负载。

3) Redis 可以以 Key-Value 键值对的形式存储一些在关系型数据库中并不方便存储的碎片化数据。

4) 操作简单，没有复杂的存取逻辑，对数据的操作像使用变量一样简单。相比于关系型数据库需要编写事件来删除数据，Redis

可以在创建 Key 的时候便捷地设置键值的过期时间，可以很好的契合验证码、登陆凭证等有效期的功能需求。综上所述，在已有的关系型数据库 MySQL 的基础

上，引入非关系型数据库 Redis 存储部分数据，用来优化项目结构和性能^[25]。

2.5 阿里云 OSS

对象存储服务（Object Storage Service，OSS）是一种网上在线存储服务，即把数据存放在通常由第三方托管的多台虚拟服务器，而非专属自建的服务器上，是在大数据环境下产生的一种存储方式，主要解决因资源增多而产生的存储问题，适合存放任意类型的文件，全面优化存储成本。

OSS 具备丰富 API 接口和 SDK 包，可以像操作本地资源一样的操作云端的资源；支持多种类型的资源存储，如图片、文件、音频等资源。因此，使用 OSS 作为存储介质具有可行性，可以解决项目中文件资源的存储问题。

同时，相比于自建服务器存储，OSS 具有如下显著优势^[26]：

- 1) 数据可靠性高：OSS 采用多重冗余备份，消除因硬件持久性产生的单点故障，不易丢失数据。
- 2) 弹性存储：不受限于物理硬件的容量限制，存储容量大且可以自动扩容、弹性伸缩。
- 3) 维护、存储成本低：无需专人维护，带宽资源充足，并具有可视化、点击式的容器管理工具。

综合以上分析，选择第三方 OSS 方案。同时，考虑到阿里云在国内云服务行业绝对的龙头地位，选择阿里云 OSS 作为项目对象存储服务器，负责存储文档资源。

2.6 模板引擎 Velocity 和 Poi-tl

简单来说，模板引擎技术的基本原理是用数据对象替换模板中设定的标签。它实现了表现层和逻辑层的分离。相比于用 IO 流生成文件，利用模板引擎生成文件具有以下好处：

- 1) 减少了字符串的拼接操作，生成文档的内容和样式更加丰富、灵活
- 2) 代码更易阅读和维护

Velocity 是一个基于 Java 的模板引擎。它允许使用简单但强大的模板语言来引用由 Java 代码定义的对象，包含数据对象和工具对象^[27]。它与 Jsp、FreeMarker

并称为三大视图展现技术。相比于其它两者，Velocity 具有简单和轻量级的特性。同时，Velocity 具有庞大的用户社区，在代码模板生成方面比较成熟。

Poi-tl (poi template language) 是 Word 类型文档的模板引擎，免费开源的 Java 类库，基于 docx 格式的 Word 模板和数据对象生成新的 Word 文档^[28]。在 Word 文档生成的功能上，相比于 Velocity，Poi-tl 具有以下优势：

1) 兼容性好、样式丰富：Velocity 由于长时间没有官方维护，对 docx 格式的 Word 文档支持度比较低，容易出现未知错误。Poi-tl 专注于生成 Word 文档，且支持丰富的样式；

2) 简单易用：Velocity 要在 xml 格式文档中定义 Word 文档的样式和内容，模板繁杂且不具可读性。Poi-tl 的模板是 docx 格式文档，这意味着我们可以点击式的预先在 docx 格式的模板中设计文档样式（如字体大小、颜色、表格属性等），简单高效且通俗易懂。

基于以上分析，考虑到模板引擎生成文件的方式的灵活性、模板的可读性和兼容性，以及代码的可维护性，选择使用 Velocity 模板引擎作为 Java 代码生成方式，选择 Poi-tl 模板引擎作为 Word 文档生成方式。

2.7 B/S 架构

B/S 架构即浏览器/服务器架构模式，是随着 Internet 技术的兴起，对 C/S 架构的一种改进的架构。在这种架构下，用户界面是通过浏览器来实现，主要事务逻辑在服务器端 (Server) 实现，极少部分事务逻辑在客户端 (Browser) 实现^[29]。B/S 架构如图2.5所示。

B/S 架构具有以下优点：

1) 用户友好：与 C/S 架构相比，大大简化了客户端，用户只需要有网络和浏览器，就可以随时随地进行查询、浏览等业务处理，不需要下载 APP 或是桌面应用等客户端。

2) 升级和维护简单：基于 B/S 架构天然的跨平台属性，项目的需求迭代简单方便，只需要针对服务器端进行升级操作。维护简单方便，只需要改变网页，即可实现所有用户的同步更新，无需客户端更新升级。

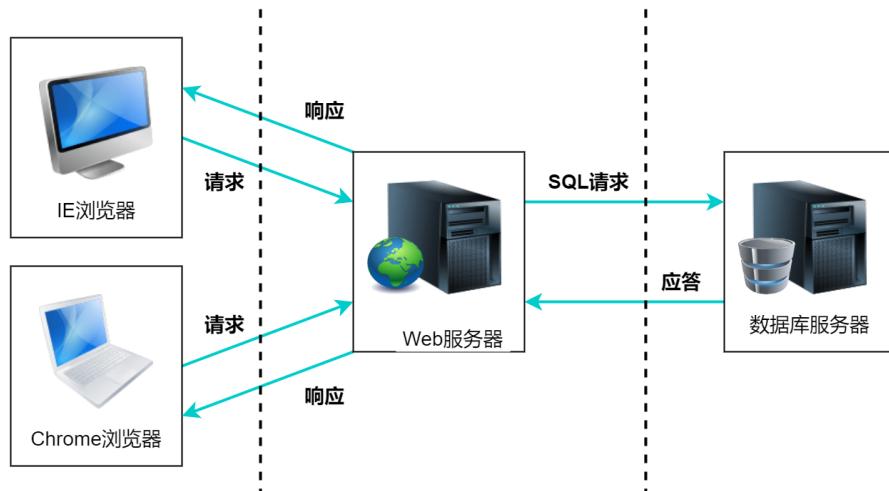


图 2.5 B/S 架构

2.8 前端构建框架 Vue 和组件库 Element UI

Vue 是一套用于构建展示界面的渐进式框架。Vue 的设计理念为自底向上逐层应用。它采用 MVVM 的设计思想，实现数据的双向绑定，核心库只需要关注视图层，不仅易于上手，还便于与第三方库或其它项目整合。渐进式框架指使用 Vue 框架的时候，并不需要把整个框架的所有部分都应用，可以根据项目开发的实际情況选择用户个人需要的部分，提供足够的选择，不需要强制性整合在一起，如 Vue 提供了路由管理、状态管理等部件的专业解决方案，但用户可以在核心的基础上任意选用其它整合方案。自底向上逐层应用，是指由底层开始做起，先把基础的部分写好，再逐层往上添加整合复杂功能。MVVM（Model-View-ViewModel）是一种设计思想，Model 层代表数据模型，也可以在 Model 中定义数据修改和操作的业务逻辑；View 代表 UI 组件，它负责将数据模型转化成 UI 展现出来；ViewModel 是一个同步 View 和 Model 的对象，MVVC 架构图如图2.6所示：

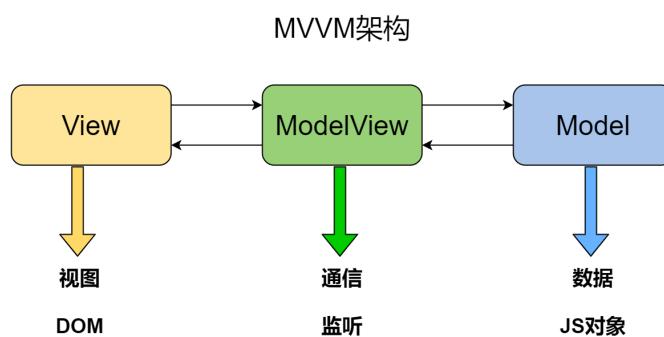


图 2.6 MVVM 架构

在 MVVM 架构下，View 和 Model 之间并没有直接的联系，而是通过 View-Model 进行交互，保证视图和数据的一致性。Model 和 ViewModel 之间的交互是双向的，View 数据的变化会同步到 Model 中，而 Model 数据的变化也会立即反应到 View 上，从而实现双向数据绑定的功能。开发人员可以专注于业务逻辑和数据本身，无需考虑页面渲染具体细节。

Element 是一套基于 Vue 框架的 UI 组件库，由饿了么公司前端团队开发并开源。使用成熟的第三方 UI 组件库，能快速搭建项目，便于后期维护，帮助网站快速成型，使开发者不必将时间花费在具体组件样式的调整上，这种积木式的基于 UI 组件的开发可以节省很多精力和时间。

考虑到本项目的研究重点并不在于视图层，为了快速开发和迭代，选择 Vue 框架构建客户端页面应用，使用 Element UI 组件库设计和构建 UI 界面^[30]。

2.9 本章小结

本章节主要分析和介绍了本项目软件所使用的部分关键技术，包括 Spring 系列开发框架、API 网关技术、OAuth2.0 安全协议、B/S 架构模型、Redis 存储技术、模板引擎等关键技术和理论，以及不同技术选型的对比和本项目的选型抉择原因。

第三章 项目需求分析

3.1 项目概述

本项目要求实现基于微服务架构的敏捷开发平台，设计基于 Spring Boot 和 Mybatis 框架风格的数据库增删改查代码的生成规则，实现代码自动生成；设计数据库描述文档的生成规则，实现文档自动生成；建立文档版本管理，使历史版本文件可追溯；围绕数据库功能建立服务组件库，辅助软件项目开发。

本项目旨在提供集成的、便捷的数据库功能组件库，协助开发者专注于自身业务功能，快速进入软件设计、代码实现和功能测试，提高开发效率。提供对数据库可定制的增删改查代码自动生成功能，使开发者不必将精力花费在重复且枯燥的代码编写上；提供数据库描述文档自动生成功能，协助开发者快速完成数据库设计文档的编写，以简明扼要的可视化表格连接起产品、前端、后端等各方沟通的桥梁，满足多方需求；提供围绕数据库的服务组件库，进行数据的批量导入和全部导出，使开发者可以轻松完成不同数据源的数据迁移工作，同时方便进行功能测试；建立历史版本文档管理，使导出记录可追溯，文档版本可管理。

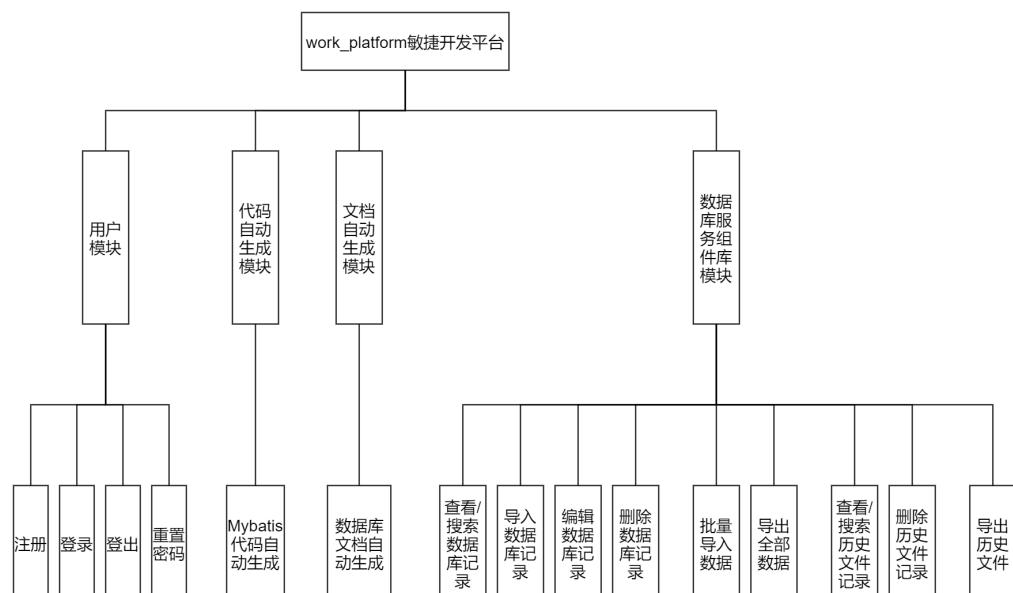


图 3.1 项目功能模块图

根据上述项目概述，提出图3.1所示系统功能模块图。

由图3.1可见，该系统具有四个功能模块。各个模块分别为用户模块、代码自动生成模块、文档自动生成模块、数据库服务组件模块。具体包括以下功能：

- 1) 用户的注册、登录、登出、重置密码；
- 2) Mybatis 代码自动生成；
- 3) 数据库描述文档自动生成；
- 4) 数据表的批量导入和全部导出；
- 5) 数据库记录的导入、编辑、删除、查看和搜索；
- 6) 历史文件记录的下载、删除、查看和搜索；

根据上述总体功能分析，以用户角度展示用户与系统交互的最简表示形式，提出系统的用例图如图3.2所示：

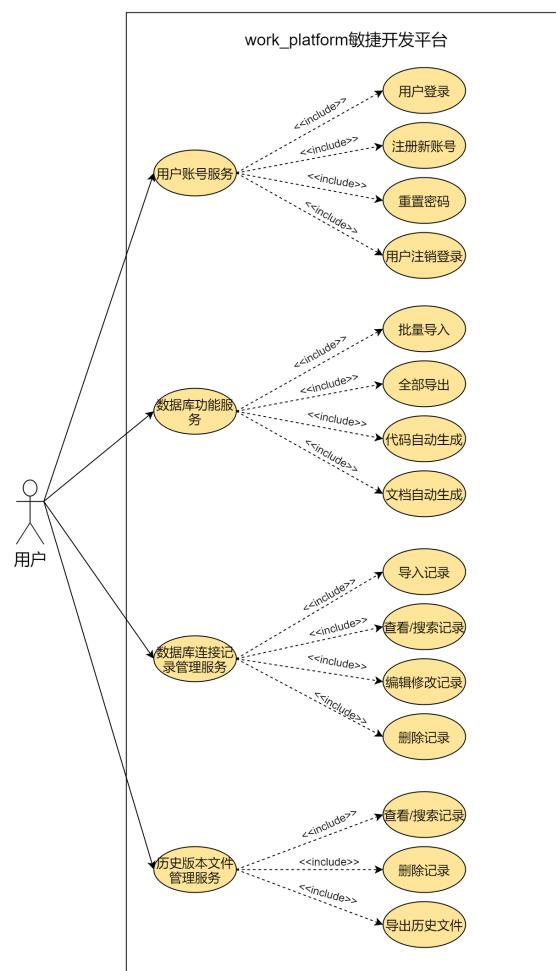


图 3.2 项目用例图

3.2 项目功能性需求

系统的功能性需求具体描述如下：

- 1) 用户注册：提供用户注册的功能，用户填写用户名、密码、确认密码、邮箱、验证码进行注册，系统持久化注册记录。若邮箱已被注册，则会提示该用户该邮箱已被注册。此时，用户为未激活状态，不能登录。用户输入的注册字段检查合法后，系统邮箱会给注册邮箱发送一封激活邮件，邮件内含激活连接，用户在规定时间内点击方可激活账号进行登录。
- 2) 用户登录：用户输入用户名、密码、验证码进行登录，系统会根据验证码、账号激活情况和账号密码匹配情况，判断是否登录成功。若登录成功，跳转到平台主页；若登录失败，则提示失败原因。
- 3) 用户注销登录：用户点击登出按钮后，即可注销当前用户，跳转至登录界面。
- 4) 用户重置密码：允许用户在未登录状态重置密码，用户填写邮箱、新密码、确认新密码、验证码进行密码重置。系统邮箱会给该账号的注册邮箱发送一封重置密码的确认邮件，邮件内含确认重置密码的链接，用户在规定时间内点击后方可完成密码重置。
- 5) 导入数据库记录：用户填写自己在公网上的数据库名称、账号、密码、主机名、端口、表名以及该数据库记录的描述进行导入。
- 6) 查看/搜索数据库记录列表：用户可以分页查询导入的数据库连接记录列表，同时，支持多字段（包括时间范围查询、字段模糊查询等）查询记录。每条数据库记录最右边一列为操作按钮，对应的是对该条记录的操作，包含编辑、删除、全部导出、批量导入、生成代码、生成文档按钮。
- 7) 编辑数据库记录：点击数据库记录操作栏的编辑按钮后，编辑按钮提示文字变为“ok”。用户可以点击式的修改数据表的 Tag，可以新增数据表或是删除原有数据表。用户编辑完成后，点击 ok 按钮即可完成编辑操作。
- 8) 删除数据库记录：点击数据库记录操作栏的删除按钮后，用户可以删除该条数据库记录。
- 9) Mybatis 代码自动生成：点击数据库记录操作栏的生成代码按钮后，系统会自动生成该数据库连接下所有指定数据表的基于 Spring boot 和 Mybatis 框架^[31,32]

风格的增删改查代码，并遵循分层规范，依次生成数据库交互层（mapper 层）、数据获取层（dao 层）、服务层（service 层）、服务实现层（serviceImpl 层）、控制层（controller 层）代码文件。并持久化导出记录。功能包括：根据主键查询单条记录、分页查询、全字段条件查询、插入单条记录、批量插入记录、根据主键更新记录、根据主键删除记录。同时，考虑到实际开发的需求，一般创建二级索引是为了根据索引查询记录，创建唯一索引是为了根据索引更新记录，因此本功能要求有针对性的生成上述方法。此外，支持代码定制，即用户可以在界面选择想要生成的层次代码以及需要实现的增删改查功能。

10) 数据库文档自动生成：点击数据库记录操作栏的生成文件按钮后，系统会自动生成该数据库连接下所有数据表的描述信息，生成 docx 格式的文件并持久化导出记录，文档以表格的形式描述数据表，表格内容包含数据表的描述、列名、列类型、列描述、列默认值、列键值类型、是否允许为空等。

11) 批量导入数据表：点击数据库记录操作栏的批量导入按钮后，会显示一个悬浮窗口，以下拉菜单的方式让用户选择该记录下的一个数据表，并让用户上传该表的 csv 格式的数据文件。系统会读取 csv 的表头，并和数据表的列名进行识别对比，支持识别下划线命名和驼峰命名的转换。识别成功后，批量导入数据至指定数据表中，支持在数据表合法规则下空缺列、空缺字段导入。

12) 导出数据表全部数据：点击数据库记录操作栏的全部导出按钮后，系统会以追加写的方式，对该数据库下所有记录的数据表的数据全部导出，并生成 zip 格式的文件，持久化导出记录。

13) 查看/搜索文件历史版本导出记录列表：用户可以在同一页面的不同 Tab 页中分页查询某种类型的文件导出记录（类型包括代码生成文件、文档生成文件、数据表数据文件），同时，支持多字段（包括时间范围查询、字段模糊查询等）查询记录。每条历史导出记录最右边一列为操作按钮，对应的是对该条记录的操作，包含删除记录、下载按钮。

14) 删除历史导出记录：点击文件历史导出记录操作栏的删除记录按钮后，用户可以删除该条历史导出记录。

15) 导出历史记录文件：点击文件历史导出记录操作栏的下载按钮后，用户可以下载该历史记录的 zip 格式的文件。

对于上述功能描述，补充说明总体需求要求如下：

- 1) 所有表单条目都需要在前端页面进行简单的合法性检查，如字段长度限制、检查确认密码和密码一致、正则匹配有效邮箱等，减少服务端的工作量，但同时服务器端也需要实现完整的检查工作，避免非法构造请求访问带来的风险。
- 2) 本项目为对外应用，需要实现良好的鉴权机制，考虑到服务端不能信任前端传递的请求字段，在用户登录后，需要进行统一鉴权，完成认证，避免用户非法获取到不属于自己的资源。
- 3) 在多字段搜索查询功能中，涉及到字符串查询的尽量使用模糊查询；涉及到时间范围查询的需要设计快捷查询方式，如“最近一周”、“最近一个月”等；涉及到类别查询的选用选择器方式。为用户带来良好的产品体验。
- 4) 激活码的设计，是为了确认用户的注册邮箱是本人的邮箱，并作为后续未登录状态下重置密码时的认证方式。

3.3 项目非功能性需求

为提高软件的质量并符合软件设计标准和规范，系统在满足主要的功能性需求的同时，还需要满足一些非功能性需求。

本系统的非功能性需求主要包括以下几点：

1. 易用性：系统的功能需要做到用户友好。用户界面简洁美观，对于各种功能的操作，具备简明扼要的提示性文字，易于学习和操作，使用户具有良好的产品体验。
2. 平台适应性：作为 B/S 架构的客户端，一般来说，不同浏览器对于页面渲染具有较好兼容性，但是由于不同浏览器的渲染内核不同，对文件的解释存在细微差异，所以系统需要有较好的平台适应性，能够在不同类型的浏览器上成功运行，需要兼顾市场上的主流浏览器和不同类型操作系统。
3. 可扩展性：系统可以轻松快捷的完成新功能的需求迭代，具有良好的扩展能力。
4. 可维护性：在展示层，提取出模块化组件进行页面复用。在服务端，合理拆分微服务模块，提取公共常量、工具类模块以及全局配置，遵循三层开发结构，逐级复用，使软件结构层次更加清晰，提高代码的可维护性和灵活性^[33]。
5. 可靠性：在系统出现内部错误时，需要有良好的服务降级策略。在热点请求中，需要根据实际 QPS（每秒请求数）、RT（响应时间）等设置流量控制策略，

进行限流。同时，系统的故障率需维持在一定水平以下。

3.4 本章小结

本章对敏捷开发系统进行了具体的需求分析，包含项目概述以及功能性需求和非功能性需求，并提出了相应的功能结构图、用例图。同时，结合了实际的开发生产需求，对软件的具体功能进行分析，简要阐述了此次项目的意义和目标。

第四章 项目设计与实现

4.1 项目总体设计

本项目整体基于 B/S 架构，前后端分离，提出项目架构图如4.1所示：

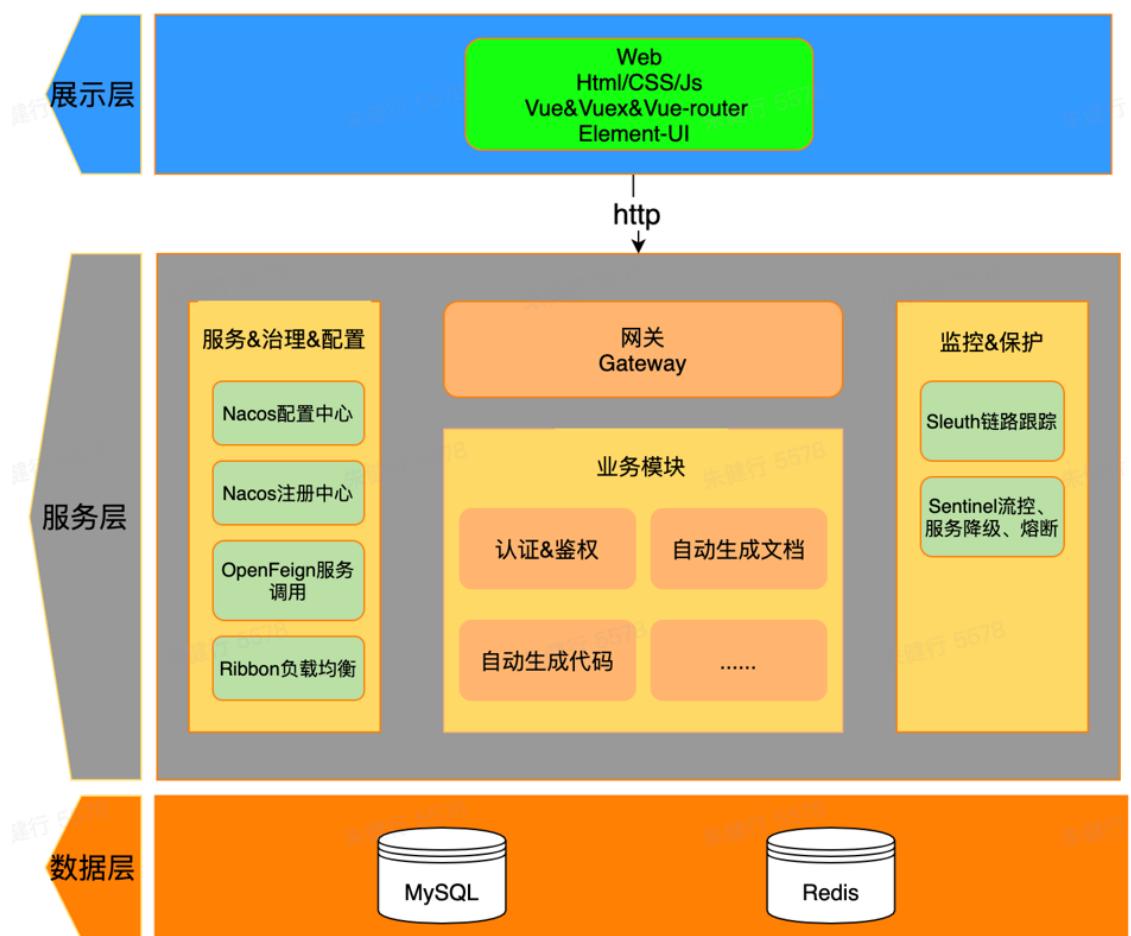


图 4.1 项目架构图

在展示层，基于 Vue 框架构建前端 Web 前端应用，使用 Element UI 组件库用来网格化布局和页面的 UI 设计，使用 Vuex 进行变量状态管理，Vue-Router 作为路由管理器。

在服务层，项目整体基于 Spring Cloud 框架实现微服务架构，SpringBoot 和 Mybatis 框架构建微服务模块，Gateway 组件构建全局统一网关，作为所有请求的

入口进行路由转发。以功能维度划分模块；以 Nacos 组件作为服务注册中心和配置中心；以 Ribbon 和 OpenFeign 组件用于服务负载和服务发现；基于 Sentinel 组件配置服务流量控制、服务降级、服务熔断策略；基于 Sleuth 进行微服务模块的分布式链路跟踪和监控。

在数据层，MySQL 作为底层关系型数据库，完成结构化数据的持久化存储。非关系型数据库 Redis 作为缓存，并存储碎片化 Key-Value 键值对数据，实现快速访问、减轻底层数据库压力。

4.2 数据存储层详细设计

4.2.1 关系型数据库 MySQL

在数据的存储层，使用 MySQL 数据库作为数据的持久化存储引擎。

根据项目需求分析，设计了系统数据库的 ER 图。在本项目中，数据持久层需要维护 3 个实体，分别为用户实体、数据库信息实体、导出文件历史版本记录实体。具体 ER 图设计如图4.2所示^①：

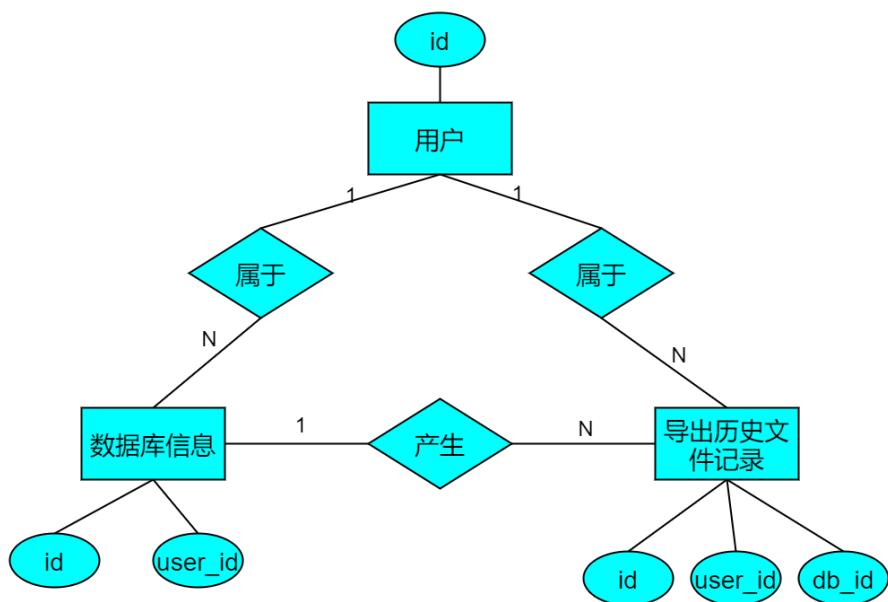


图 4.2 系统数据库 ER 图

用户、数据库连接信息、导出文件历史版本信息这三个实体在数据库中以数据表的形式结构化、持久化保存，通过外键相关联。由于用户与数据库连接信息、

^①此处绘制 ER 图仅为描述实体间的关系，部分实体属性省略。

数据库连接信息与导出文件历史版本信息等实体关系均为一对多关系，故无需多对多关系表。本项目一共建有 3 个数据库表，分别为用户信息表、数据库连接信息表、导出文件历史版本信息表。下面对数据表进行详细说明：

(1) 用户信息表

表 4.1 用户信息表 user

列名	类型	描述	key 类型
id	int(11)	递增主键 id	PRI
username	varchar(64)	用户名	MUL
password	varchar(64)	密码	
email	varchar(64)	邮箱	MUL
status	int(11)	激活状态	
activation_code	varchar(128)	激活码	
reset_code	varchar(128)	重置码	
header_url	varchar(256)	头像 url 地址	
create_time	timestamp	创建时间	
update_time	timestamp	更新时间	

user 表记录所有注册的用户信息 (status=1 表示已激活, status=0 表示未激活)。其中, id 为递增主键, 唯一标识一条用户信息记录。在建立的数据库信息和文件导出历史信息中, id 作为外键存储, 用于用户鉴权。

(2) 数据库连接信息表

表 4.2 数据库连接信息表 db_info

列名	类型	描述	key 类型
id	int(11) unsigned	主键	PRI
user_id	int(11)	所属 userId	
db_name	varchar(64)	数据库名称	
db_comment	text	数据库描述	
tables	text	json 格式表名列列表	
username	varchar(64)	用户名	
password	varchar(64)	密码	
host	text	主机名	
port	varchar(8)	端口	
create_time	timestamp	创建时间	
update_time	timestamp	更新时间	

数据库连接信息表, 存储用户导入系统的所有数据库连接信息记录。围绕数据库的代码和文档自动生成等服务都基于此表。其中, id 为递增主键, 唯一标识一条数据库连接信息记录。user_id 作为外键, 和 user 表的 id 相关联。

(3) 导出文件历史版本信息表

表 4.3 导出文件历史版本信息表 download_info

列名	类型	描述	key 类型
id	int(11) unsigned	主键	PRI
user_id	int(11)	用户 id	
db_id	int(11)	数据库 id	
oss_details	text	json 格式表名列表	
file_type	varchar(64)	文件类型	
password	varchar(32)	数据库连接密码	
username	varchar(32)	数据库用户名	
db_name	varchar(64)	数据库名称	
tables	text	生成的 table 数组	
host	text	数据库 host	
create_time	timestamp	创建时间	
update_time	timestamp	更新时间	

导出文件历史版本信息表，存储用户导出的所有文件记录。文件历史版本管理服务基于此表。其中，id 为递增主键，唯一标识一条导出文件历史版本信息记录。user_id 作为外键，和 user 表的 id 相关联。db_id 作为外键，和 db_info 表的 id 相关联。type 为生成文件的类别，code 为生成 mybatis 代码文件，doc 为生成数据库信息文档，data 为数据表批量导出数据文件。oss_details 为阿里云的 bucket 存储文件路径。

4.2.2 非关系型数据库 Redis

根据项目需求，设置了两处 Redis 的存储场景，分别是验证码的存储和用户登录 Token 的存储。两者的具体键值设计如表4.4所示：

表 4.4 Redis 数据库键值对设计

名称	Key	Value	过期时间
验证码	kaptcha:{uuid}	验证码值	2 分钟
登录 Token	token:{token}	user 对象	根据登录时间选项

其中，uuid 为每次生成验证码时生成的随机字符串，存储于浏览器 Cookie 中，在检验验证码时，读取携带的 Cookie 值即可拼接为当前验证码的 key 值，并从 Redis 中读取出验证码值。

4.3 服务层通用设计

4.3.1 服务注册与发现

在微服务架构中，不同的微服务往往具有不同的 ip 和端口，当一个微服务访问其它微服务时，如果采取人工硬编码的方式将服务提供者的 ip 信息写到程序中，后期的需求迭代会有很大的问题，无法满足微服务功能需求。

基于上述分析，在微服务架构中实现了服务发现机制，即每个微服务将自身的 ip、端口号等信息统一注册到注册中心中，由注册中心统一维护一个服务注册列表。微服务之间基于此表进行服务调用。

本项目基于 Nacos 实现服务注册与发现，主要按照以下步骤实现：

1. 在阿里云服务器上，运行 startup.sh 启动脚本，启动 Nacos 服务。
2. pom 文件中为项目添加 Nacos discovery 的依赖 spring-cloud-starter-alibaba-nacos-discovery。
3. application.yml 配置文件中添加注册信息：

```
server:  
  port: 8443 # 微服务端口号  
  
spring:  
  application:  
    name: gateway-service # 微服务名称  
  profiles:  
    active: dev # 表示开发环境  
  cloud:  
    # 注册进 nacos Server  
  nacos:  
    discovery:  
      server-addr: 47.99.131.144:8848
```

4. 微服务主启动类添加 @EnableDiscoveryClient 注解，为服务开启服务注册与发现功能。

4.3.2 分布式全局配置

微服务架构中，每个微服务都是一个独立的进程，如果每个微服务都拥有自身互不关联的配置，会对项目后期维护带来困难。因此项目引入全局配置管理，提取项目的数据库连接信息等项目公共配置，并持久化存储到系统数据库中。

目前主流的实现方案有配置中心 Spring Cloud Config 整合消息总线 Bus 以及 Nacos Config 两种，但是前者需要自建 Git 仓库和消息队列，且更新配置后需要人工向 Config Server 发送 Post 请求实现更新，非常不方便。因此，本项目基于 Nacos Config 实现分布式全局配置功能，主要按照以下步骤实现：

1. 在阿里云服务器上，运行 startup.sh 启动脚本，启动 Nacos 服务。
2. pom 文件中为项目添加 Nacos config 的依赖 spring-cloud-starter-alibaba-nacos-config。
3. bootstrap.yml 配置文件中添加配置中心信息
4. 在需要获取配置信息的类中添加 @RefreshScope 注解，实现全局配置修改后自动刷新。
5. 全局配置数据持久化：执行 Nacos/conf 下的 nacos-mysql.sql 文件中的 SQL 命令，创建 nacos_config 数据库，并于 application.properties 中配置 Nacos 数据持久化的数据库连接信息：

4.3.3 API 网关服务

4.3.4 服务调用与声明式接口

4.4 服务层通用设计

4.4.1 类结构层次设计

服务端采用 Dao、Service、Controller 三层结构开发：

- 1) Dao 主要是做数据持久层的工作，负责与数据库进行原子性交互。
- 2) Service 层主要负责业务模块的逻辑应用设计，定义接口和接口实现类，根据业务需求对 D 层接口实现复用。
- 3) Controller 层：Controller 层负责具体的业务模块流程的控制，在此层里面主要调用 Service 层的接口来控制业务流程。这样不仅使程序结构变得清晰，也大大

减少了代码量。

4.4.2 响应消息体设计

考虑到本项目为前后端分离项目，故服务端返回 Json 格式数据，不包含视图。采用统一数据格式的响应结构体，如表4.5下：

表 4.5 统一响应消息体 CommonResult 设计

字段名	类型	描述	demo
code	Integer	状态码	STATUS_OK=200
message	String	提示消息	
data	Object	json 格式数据	

4.4.3 提取公共模块

考虑到本项目的微服务架构，提取出公共模块 api-common，封装公用常量、工具类、实体类，打包作为其它微服务模块的依赖。基于 Nacos 配置中心作全局配置，提取项目的数据库连接信息等项目公共配置，并持久化存储到系统数据库中。

4.4.4 统一认证鉴权

本项目为对外应用项目，应该做到良好的鉴权，不能让用户获取到不属于自身的资源。添加横切功能，统一拦截请求头中的 Token 并认证。用 Token 拼接成 key 值，并将用户 user 对象信息存入 Redis 中。根据 Redis 存储的 user 对象的 id 鉴权，如某个历史文件导出记录归属于该认证用户。

4.4.5 数据库相关对象主体类设计

由需求分析，除了持久层的 3 个数据表实体外，围绕数据库相关功能建立 3 个主体类，分别为数据库对象 DBStruct，数据表对象 TableInfo，数据列对象 ColumnInfo，如图4.3所示。

下面对这 3 个对象实体类进行具体说明：

- (1) DBStruct 实体类
- (2) TableInfo 实体类

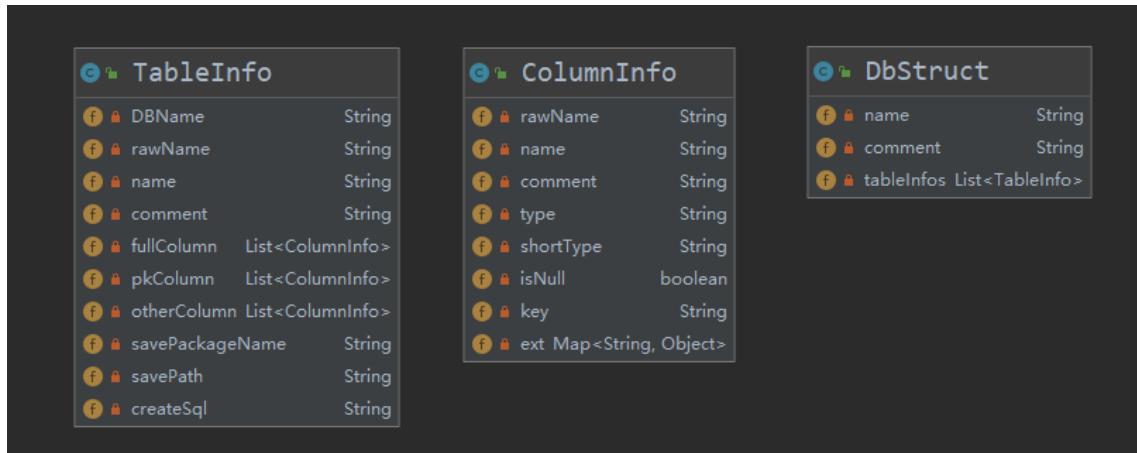


图 4.3 实体类设计概述

表 4.6 DBStruct 实体类

字段名	类型	描述	demo
name	String	Java 对象名称(一般为 驼峰式)	createTime
comment	String	数据表描述	
tableInfos	List<TableInfo>	数据库下表对象信息	

表 4.7 TableInfo 实体类

字段名	类型	描述	demo
DbName	String	所属数据库名称	
rawName	String	数据库原始名称 (一 般为下划线式)	
name	String	Java 对象名称(一般为 首字母大写驼峰式)	DbInfo
comment	String	数据表描述	
fullColumn	text	全部列信息	
pkColumn	List<ColumnInfo>	主键信息	
otherColumn	List<ColumnInfo>	除了主键的列信息	
savePackageName	List<ColumnInfo>	保存的包名称	com.godx.cloud
savePath	String	保存路径	/service/impl
createSql	String	建表 sql 语句	

(3) ColumnInfo 实体类

表 4.8 ColumnInfo 实体类

字段名	类型	描述	demo
-----	----	----	------

4.5 项目模块核心功能详细设计

4.5.1 注册登录模块详细设计

1. 验证码功能详细设计

生成验证码：验证码图片的生成基于 Google Kaptcha 框架，定制生成数字和英文字母的 4 位随机组合。每次生成验证码时，会随机生成一个 uuid 字符串，并存于浏览器 Cookie 上，同时用 uuid 拼接 key 将验证码值存于 Redis 中。最后写入图片流到响应输出流中。

校验验证码：服务端接收到客户端验证码参数和 Cookie 后，拼接成 Redis Key，从而校验传参和 Redis 中存储验证码的匹配性。

具体程序流程图如图4.4所示。

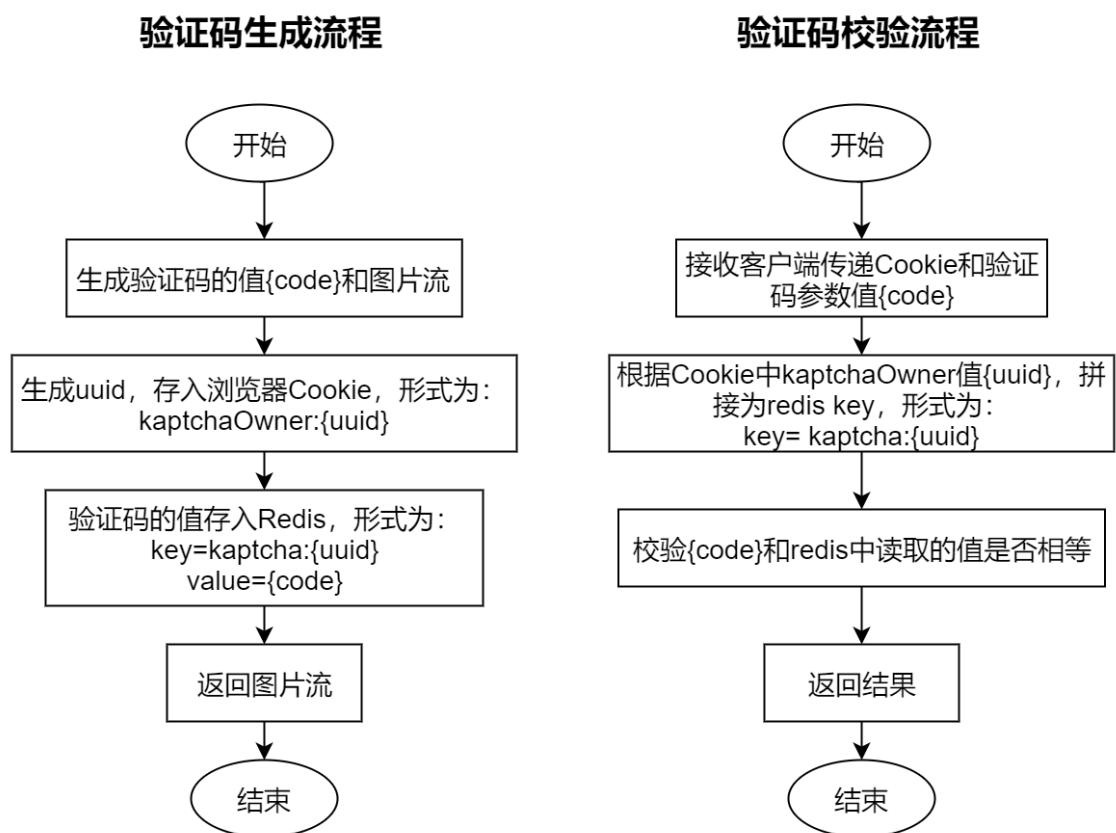


图 4.4 验证码功能程序流程图

在注册登录等表单中设置验证码功能的目的是为了防止恶意访问，通过验证码校验减少恶意访问次数。并且优先从 Redis 中检查验证码，再访问数据库查询记录，减轻服务器和数据库访问压力。用户填写验证码时，可以点击验证码图片或

“看不清，换一张”链接更换验证码图片。

2. 注册功能详细设计

进入注册页面，用户输入账号、密码、确认密码、邮箱、验证码的有效信息，前端表单检查机制在字段失去焦点时检查输入字段是否合法，用户填写过程中，可以点击重置按钮重置表单。点击注册按钮，不合法则进行针对性的提示，要求重新输入信息；若检查合法，则发起用户注册请求，服务端收到请求后，先检查验证码正确性，若不正确则直接返回错误信息；若正确，则会完整检查表单项的合法性，会从数据库中查询该邮箱是否已经被注册过。若注册过，会提示该邮箱已被注册提示；若没有注册过，则会存储该用户信息到数据库，并置当前用户状态为“未激活”。此时，系统会发送一封激活邮件给注册邮箱，邮件内含有 Restful 格式的激活链接，用户在规定时间内点击激活链接后方可激活账号，进行正常登陆，反之则继续处于未激活状态，不可登陆。客户端接收到响应数据，若注册成功，则跳转到登陆页面进行登陆；若注册失败，则根据错误提示信息重新注册。用户/客户端、服务端、系统邮箱的 SMTP 发件服务器的交互具体如图4.5所示。

其中，激活链接的 API 设计形如/activation/userId/code。其中，userId 为 user 表的主键，唯一标识一条 user 记录，code 包含随机生成的 uuid 字符串和发送时间的时间戳信息。用户点击链接后，根据 userId 找到数据库中的 user 记录的激活码并和参数 code 对比，则根据当前时间戳判断用户点击操作是否在有效期 10 分钟内。若均检查通过，则账号激活成功；反之则失败。

3. 重置密码功能详细设计

重置密码功能流程，和上述注册/激活流程大致相同，可以概述为信息提交/邮件确认，在此不多做阐述。

4. 登录功能详细设计

登录认证功能采用 OAuth2.0 方案，图4.6所示的活动图描述客户端、认证中心和各微服务模块的交互关系，主要步骤如下：

- 1) 客户端根据用户名和密码请求登录
- 2) 认证中心校验用户名和密码后，生成 Token 凭证并返回给客户端
- 3) 客户端以后的每次请求，都携带 Token 进行资源请求
- 4) 服务模块接收到客户端携带的 Token 后，于认证中心验证 Token 的合法性
- 5) 服务模块验证完合法性后，如果 Token 合法，则响应请求；若 Token 不合

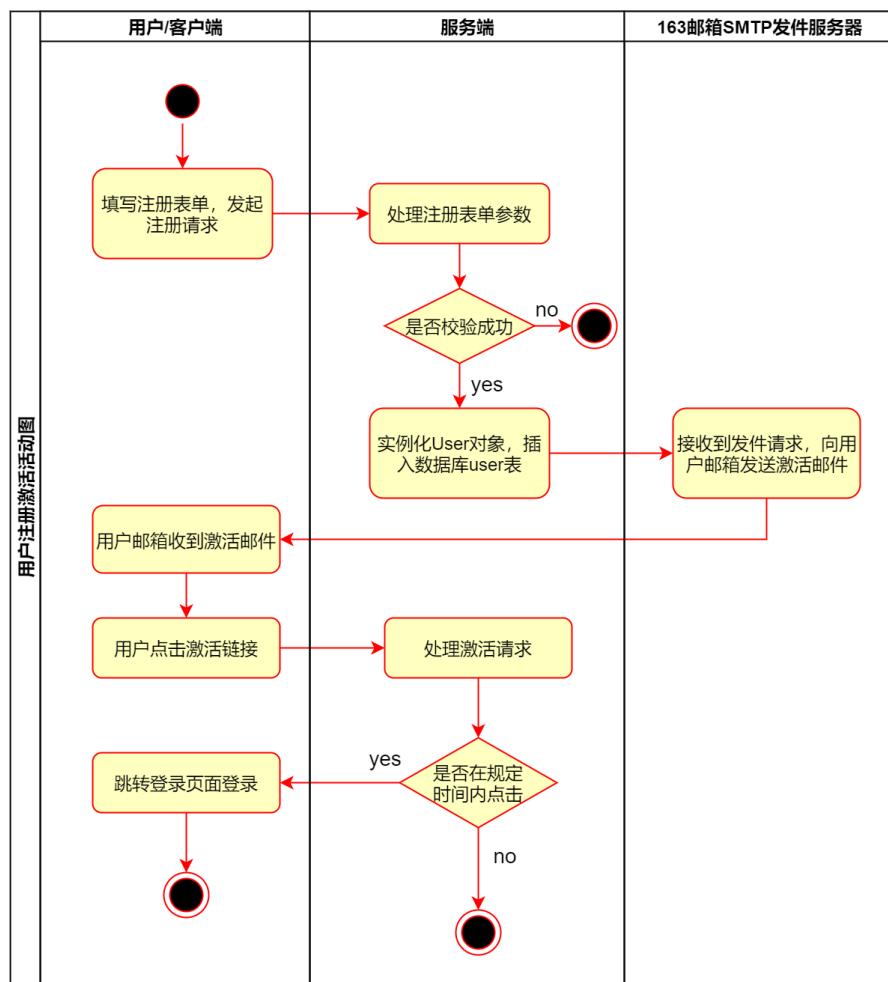


图 4.5 注册激活功能活动图

法，拒绝响应请求资源，返回未认证信息

登录功能具体校验流程：进入登录页面，输入用户名、密码和验证码，前端表单检查机制在字段失去焦点时检查输入字段是否合法，用户填写过程中，可以点击重置按钮重置表单。表单检查通过后，点击登录按钮，向服务端发起登录请求。服务端接收后，先检查验证码正确性，若不正确则直接返回错误信息；若正确，则会完整检查表单项的合法性，查询数据库检查是否存在该条用户信息。若校验通过，服务端返回登录凭证 Token 令牌以及用户的个人信息（头像、用户名等），客户端跳转到主页。若校验不通过，则弹出对应的错误消息。

4.5.2 数据库代码自动生成模块详细设计

代码生成功能程序流程图和生成步骤细节流程图如图4.7所示。服务端收到请求后，根据 id 从系统数据库 db_info 表中找到该条数据库连接记录，鉴权通过后，

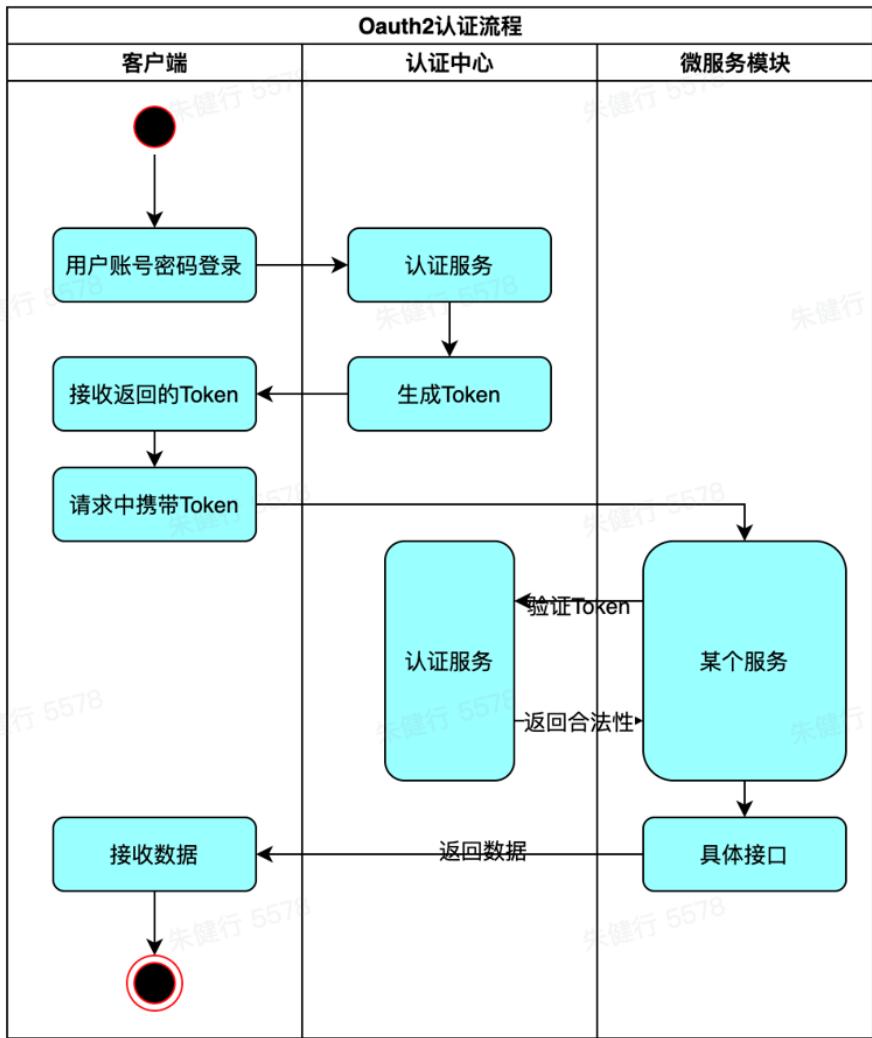


图 4.6 登录功能活动图

通过该记录的连接参数建立数据库连接。连接成功后，以数据表为维度，依次生成 mapper、service 等层的代码文件。数据表全部生成成功后，将文件夹后压缩为 zip 格式文件，上传至阿里云 OSS 统一路径下，删除本地文件，并将该条文件导出记录插入数据库 download_info 表，标记文件类别为 code。

Mybatis 代码生成功能基于 Velocity 模板引擎，生成代码文件主要分为以下几步：

- 1) 加载 Velocity 配置参数文件；
- 2) 创建 VelocityContext 上下文对象，将数据对象添加到此上下文中；
- 3) 加载事先写好的 Velocity 模板文件；
- 4) 合并 Velocity 模板和上下文对象；
- 5) 将文件的写入流写入创建的文件中。

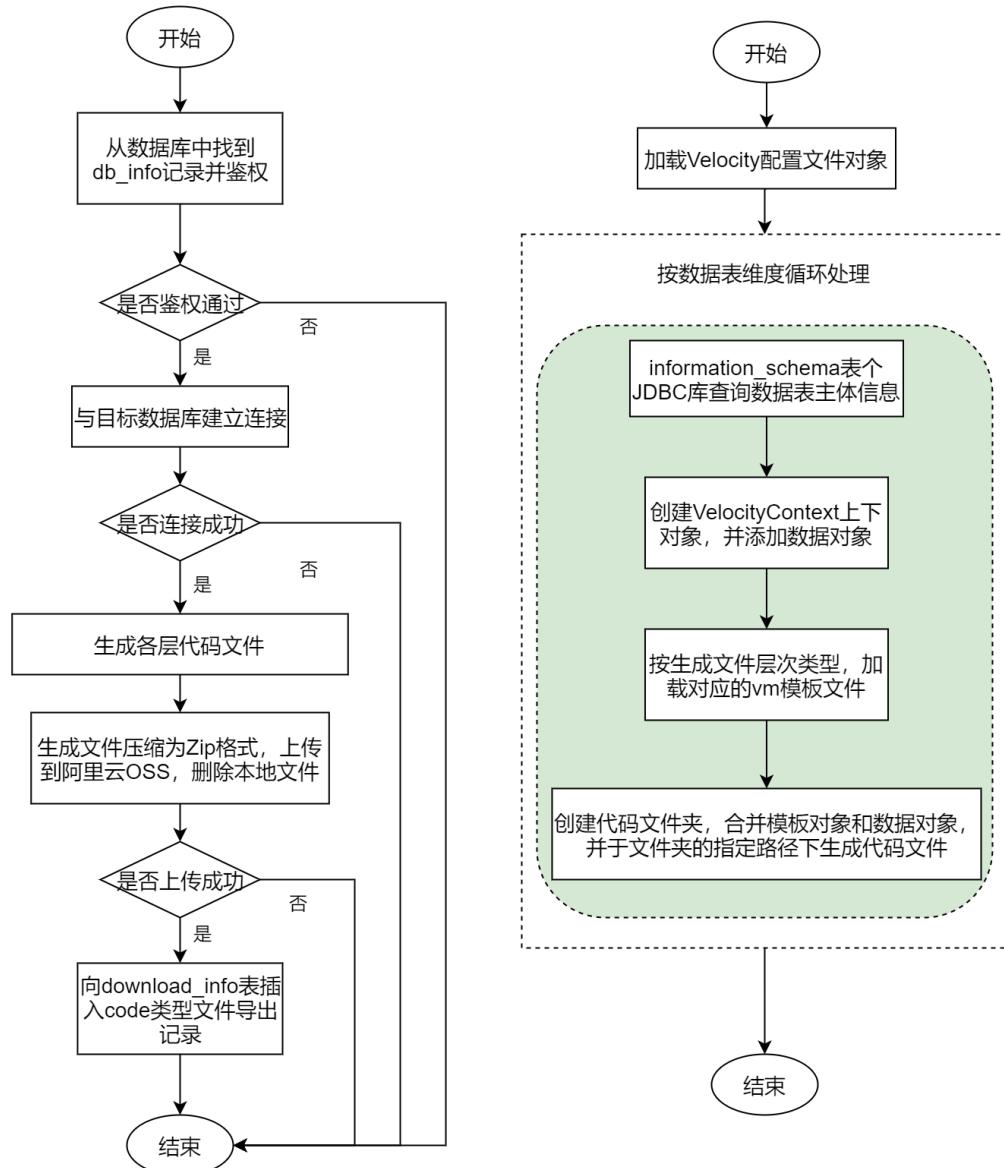


图 4.7 代码生成功能程序流程图

上述代码生成步骤的核心在于 Velocity 代码模板的编写和数据的导入。

在 Velocity 模板层面，以 model 层模板为例：定义 macro 宏定义，实现按需自动导入 API 包。根据 Java 代码规范，java.lang 下的包无需手动导入，故定义导入包集合 set，根据字段类型，过滤长类型为 java.lang 开头的包，导入其它需要导入的包，如 Date 类型需要导入 java.util.Date 包。使用 #foreach 命令循环解析 set，生成代码行，具体宏定义如下：

代码 4.1 自动导入 API 包宏代码

```

1  #set($temp = $tool.newHashSet())
2  #foreach($column in $tableInfo.fullColumn)

```

```

3  ## 带有反回值的方法调用时使用$tool.call来消除返回值
4  #if(($column.type.startsWith("java.util.")))
5      $tool.call($temp.add($column.type))
6  #end
7 #end
8
9 ##自动导入包（仅导入实体属性需要的包）
10 #foreach($import in $temp)
11 import $!import;
12 #end

```

```

#set($temp = $tool.newHashSet())
#foreach($column in $tableInfo.fullColumn)
## 带有反回值的方法调用时使用$tool.call来消除返回值
#if(($column.type.startsWith("java.util.")))
    $tool.call($temp.add($column.type))
#end
#end

```

```

##自动导入包（仅导入实体属性需要的包）
#foreach($import in $temp)
import $!import;
#end

```

同时，为简化模板开发，定义公共宏模板，如代码注释、get/set 方法，用来生成不同类别、相同样式的注释和代码。

mapper 层、dao 层，service 层、serviceImpl 层、controller 层模板编写和上述 model 层 Velocity 模板原理方法相似，在此不做过多叙述。

在数据层面，主要通过 JDBC API 库和数据库的原生库 information_schema 获取数据库、数据表、数据列的数据信息。同时，编写单例模式工具库，实例化工具类后传入 VelocityContext 上下文，供 Velocity 模板引擎调用。

4.5.3 数据库描述文档自动生成模块详细设计

文档生成功能程序流程图和生成步骤细节流程图如图4.7所示。服务端收到请求后，根据 id 从系统数据库 db_info 表中找到该条数据库连接记录，鉴权通过后，通过该记录的连接参数建立数据库连接。连接成功后，查询该数据库下所有数据表的信息，以数据表为维度，绘制成表格，并生成 docx 格式的文件。生成成功，将文件夹后压缩为 Zip 格式文件，上传至阿里云 OSS 统一路径下，删除本地文件，并将该条文件导出记录插入数据库 download_info 表，标记文件类别为 doc。

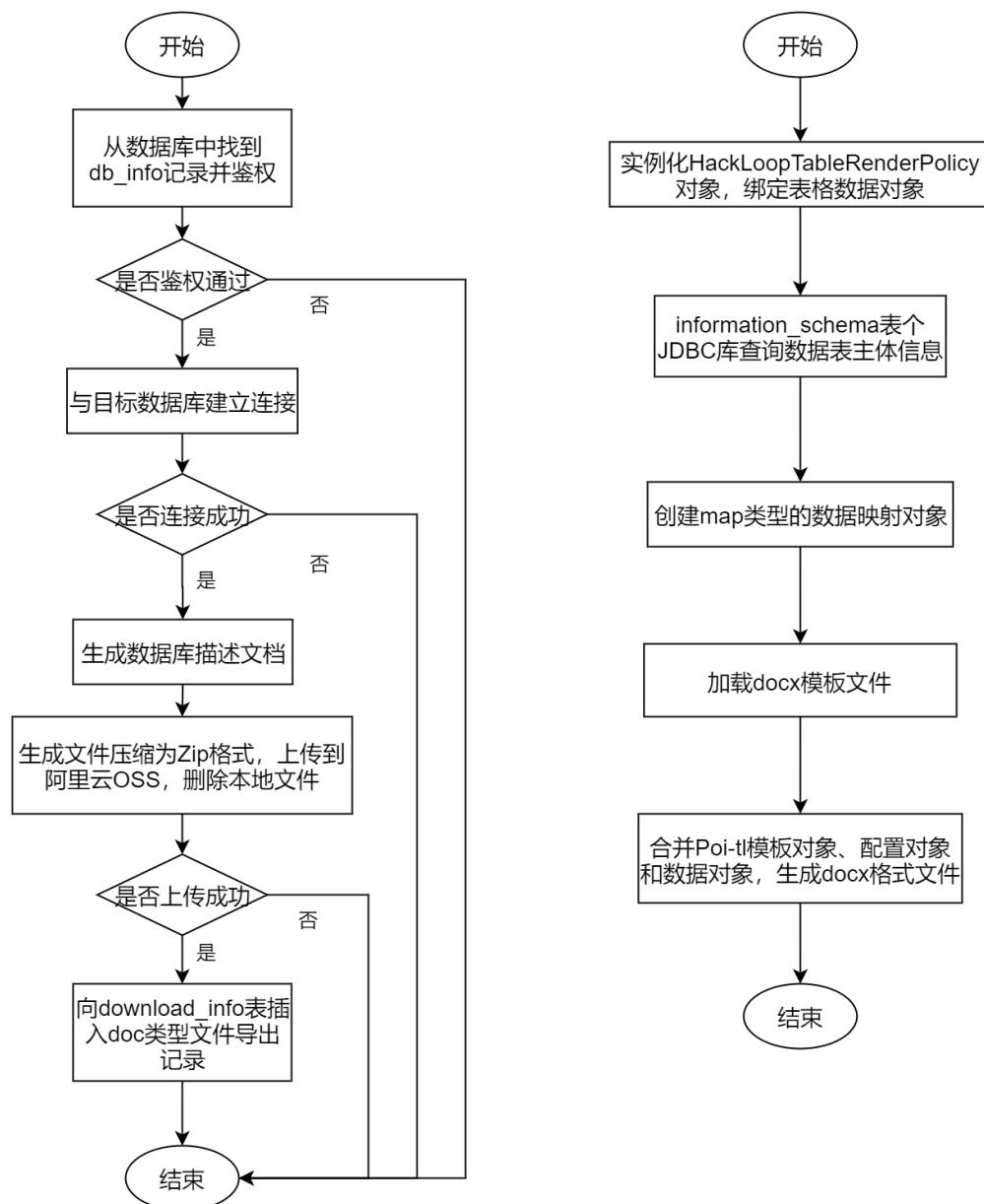


图 4.8 文档生成功能程序流程图

生成 docx 文件主要分为以下几步：

- 1) 加载 Poi-tl 模板文件；
- 2) 创建模板配置对象，添加配置；
- 3) 创建 map 格式的数据对象，添加模板需要的数据对象；
- 4) 合并 Poi-tl 模板，渲染数据对象、编译配置对象；
- 5) 将文件流写入创建的文件中。

上述步骤的核心在于 Poi-tl 文档模板的编写、配置对象以及数据的导入。

在 Poi-tl 模板文件编写上，利用区块对进行数据循环解析，表格行循环进行循环表格行。

在模板配置上，新建 Poi-tl Configure 配置对象，根据功能需求，需要自动生成表格，故配置表格行循环规则，新建 HackLoopTableRenderPolicy 对象，绑定表格对象。

在模板渲染数据上，主要通过 JDBC API 库和数据库的原生库 information_schema 获取数据库、数据表、数据列的数据信息。

4.5.4 文件版本管理模块详细设计

1. 文档历史版本导出记录的查询/搜索、删除功能

文档历史版本导出记录对应数据库 download_info 表，在认证和鉴权的前提下完成对 download_info 表的删、查基础功能。

2. 文档历史版本的下载功能

阿里云 OSS 提供了 URL 下载方式，用户可以直接访问 OSS 下载文件。但是考虑到文件的鉴权问题，所有的下载请求都需要通过服务器端鉴权后方能下载。

根据 OSS 的终端地址、秘钥与 OSS 建立连接，通过桶名称以及数据库 download_info 表中存储的 OSS 文件路径即可定位文件位置进行下载，将文件流写入到响应的输出流中。

4.5.5 围绕数据库的服务组件库模块详细设计

1. 数据库连接记录的导入、查询/搜索、编辑、删除

文档历史版本导出记录对应数据库 db_info 表，在认证和鉴权的前提下完成对 db_info 表的增、删、改、查功能。

导入时，用户输入字段检查合法后，系统会判断用户提供的字段是否可以连接成功。若连接成功，则向系统数据库添加该条数据库连接信息记录；若连接失败，则提示失败原因。

2. 批量导入

点击数据库记录操作栏的批量导入按钮后，显示一个悬浮窗口，以下拉菜单的方式让用户选择该记录下的一个数据表，并让用户上传该表的 csv 格式的数据文件。服务器端读取 csv 文件的表头，并和数据表的列名进行逐一识别对比，基于正则表达式，支持识别下划线命名和驼峰命名的转换。识别成功后，编写 SQL 语句，批量导入数据至指定数据表中，支持在数据表合法规则下文件空缺列、空缺字段导入。

3. 全部导出

点击数据库记录操作栏的全部导出按钮后，以数据表为维度，对该数据库下所有记录的数据表的数据全部导出。考虑到大批量数据查询对底层数据库的访问压力，采用流式导出的方式，生成 csv 格式的数据文件，单次查询 10000 条数据并追加写到文件尾部。然后将所有数据文件打包压缩成 zip 格式的文件，将文件上传到阿里云 OSS 指定路径下，删除本地文件，并将该条文件导出记录插入数据库 download_info 表，标记文件类别为 data。

4.6 本章小结

本章主要在项目需求分析的基础上，构建了 B/S 结构的软件体系整体架构，分析了关系型数据库 MySQL 数据表结构设计和非关系型数据库 Redis 的键值对设计。在上述基础上，抽象出服务端整体设计的规范和主体类设计。以用例图的角度，将项目划分为几个模块，并分别介绍每个模块核心功能的详细实现过程，逐一对各个模块设计，提出程序流程图描述各功能实现的详细流程，提出活动图，对客户端与服务器端的交互流程进行详细分析。

第五章 项目测试与结果展示

5.1 项目测试环境

在本地用 Tomcat 部署微服务功能模块，Nacos 服务注册中心、MySQL、Redis 运行、部署在阿里云服务器上。客户端浏览器使用 Google Chrome。

5.2 项目核心功能测试过程及结果

5.2.1 注册登录模块测试

1. 用户注册

进入注册页面，用户输入账号、密码、确认密码、邮箱、验证码的有效信息，点击注册按钮后，收到系统发送的激活邮件，在规定时间内点击激活链接后完成账号激活，进行正常登陆。

2. 用户登录

进入登录页面，用户输入用户名、密码和验证码的有效信息，点击登录按钮后登陆成功，页面跳转到主页，展示用户头像、昵称等。

3. 重置密码

进入重置密码页面，用户输入邮箱、密码、确认密码的有效信息，点击确认按钮后，收到系统发送的重置确认邮件，在规定时间内点击确认链接后完成密码重置，进行正常登陆。

5.2.2 数据库代码自动生成模块测试

点击一条数据库连接信息操作栏的生成代码按钮，系统会生成一条 Mybatis 代码文件生成记录，下载解压后可以直接运行，完成对数据库的增删查改操作。

5.2.3 数据库描述文档自动生成模块测试

点击一条数据库连接信息操作栏的生成文档按钮，系统会生成一条数据库描述文档生成记录，下载解压后为 docx 格式的 Word 文档，文档以表格的形式逐个展示数据库下所有数据表的信息。

5.2.4 文件版本管理模块详细设计测试

1. 文档历史版本导出记录的查询/搜索

进入文档历史版本管理页面，可以按照类别分页查询 Mybatis 代码生成文件记录、数据库描述文档生成记录、数据表数据生成记录。选择文件导出时间范围、填入数据库名称等筛选条件，点击搜索按钮，可以准确查询数据。

2. 文档历史版本导出记录的删除功能

点击一条文件导出记录操作栏的删除按钮，该记录被删除。

3. 文档历史版本的下载功能

点击一条文件导出记录的下载按钮，可以下载历史文件。

5.2.5 围绕数据库的服务组件库模块测试

1. 数据库连接记录的查询/搜索

进入数据库连接管理页面，可以分页查询已导入的数据库连接信息记录。选择记录创建时间范围、填入数据库名称等筛选条件，点击搜索按钮，可以准确查询数据。

2. 数据库连接记录的导入

点击表头导入数据库按钮，弹出一个悬浮窗，用户填写自己公网上的数据库名称、账号、密码、主机名、端口、表名并添加该数据库记录的描述，点击确认后。该条数据库连接信息记录被添加。

3. 数据库连接记录的编辑功能

点击一条数据库连接记录操作栏的编辑按钮，该记录的表格属性变得可编辑。编辑完成后，点击 ok 按钮后，完成记录修改。

4. 数据库连接记录的删除功能

点击一条数据库连接记录操作栏的删除按钮，该记录被删除。

5. 批量导入

点击一条数据库连接信息操作栏的批量导入按钮，平台弹出一个悬浮框，选择数据表和 csv 格式的数据文件，点击确认按钮，发现数据表成功插入了 csv 文件中的数据。

6. 全部导出点击一条数据库连接信息操作栏的全部导出按钮，系统会生成一条数据库数据文件生成记录，下载解压后，每个数据表都有一个 csv 文件，内容为该表下所有数据记录。

5.3 本章小结

本章主要介绍在客户端上进行的功能性测试，对每个功能模块进行测试，分别描述了各个模块测试的内容、过程与结果，保证了软件的正确性、完整性和安全性，提高了软件的质量。测试结果达到了预期的目标。

第六章 总结与展望

6.1 工作总结

总体来说，本文主要完成以下工作：

- 1) 研究了目前国内外的微服务架构和敏捷开发平台的现状，分析了项目的背景和应用，对比了同类软件成果，确定了项目的需求方向；
- 2) 详细介绍了 Spring 系列开发框架、API 网关技术、OAuth2.0 安全协议、B/S 架构模型、Redis 存储技术、模板引擎等关键技术和理论，并对不同技术选型进行对比，说明选型原因；
- 3) 对项目的功能性需求和非功能性需求进行合理的分析，给出各个功能模块的具体需求；
- 4) 基于微服务架构，提出系统的总体框架图，对各个功能模块进行设计和实现，设计出相应流程图和活动图；
- 5) 对各个模块的核心功能进行测试。

通过验证，本平台提供了完整的认证和鉴权服务，实现了数据库增删改查代码的自动生成、数据库设计文档的自动生成、历史文档的版本管理与下载、数据表数据批量导入/导出等功能。简化了软件设计、开发、测试等流程，有效提高了用户的开发效率。

6.2 工作展望

此次开发的敏捷开发平台是一款围绕数据库功能的工具平台，能够协助开发者快速开发，但项目仍然有一些可以完善的地方，主要有以下几点：

- 1) 功能具有一定的局限性：本项目的数据库功能服务对象是服务端开发主流的 MySQL 数据库的使用者，没有考虑对 SQL Server、Oracle 等数据库的兼容性。在代码生成功能中，服务对象也是主流的 Java 语言、Spring 系列框架的使用者，没有对其它服务端开发语言如 GoLang 等提供支持。

2) 功能增强：在服务组件库中，以 csv 格式的数据文件作为导入/导出的载体，成为数据迁移的中间一环。然而，在不同数据源的数据迁移的背景下，用户还需要编写对其他数据源的增、查操作。基于此分析，考虑在后续增强数据迁移功能，对用户隐去中间环节，提供 Hive、ElasticSearch、MySQL 等多数据源之间的数据迁移功能。

3) 项目部署：由于购买的服务器资源不足以支持部署整个项目，故本次只将 Nacos 注册中心、存储层的 MySQL 和 Redis 放在服务器，提供公网访问。在后续计算机资源允许的情况下，会考虑将整个项目基于 Docker 技术容器化^[34,35] 部署到公网，以实践检验项目成果。

在后面的学习工作中，我会提高自己专业能力，对比和总结自身的不足，不断完善项目开发，对软件进行优化提升。

测试文字测试文字^①

6.2.1 数字转中文

测试数字：1234.233. 转为中文数字：一千二百三十四点二三三；转为中文字符串：一二三四点二三三。

6.3 表格

6.3.1 普通表格

先来看一个无标题的普通表格：

标题	标题	标题
1	2	3

如果想要居中可以使用 center 环境。

带标题表格。

表 6.1 普通表格 1

标题	标题	标题
1	2	3

来一个表格并排，每个表格一个标题：

再来一个表格并排

^①注脚 2。

表 6.2 并排表格 1

标题	标题	标题
1	2	3

表 6.3 并排表格 2

标题	标题	标题
1	2	3

表 6.4 并排表格

(a) 并排表格 1

标题	标题	标题
1	2	3

(b) 并排表格 2

标题	标题	标题
1	2	3

6.3.2 复杂点的表格

表 6.5 主要用到的是列合并单元格、跨行合并单元格、混合合并单元格、表格横线的自定义粗细以及控制表格横线的自定义连接等。^②

表 6.5 复杂表格

1	3	4
5	6	7
9	10	11
13	15	16
	19	20

表格填充颜色，如表 6.6 所示。此次，主要用到的出上述介绍外，有单个单元格填充颜色、整列填充颜色、整行填充颜色；此外，还添加了单元格划分的功能。

以上，就是普通表格的常用例子了，其他的应用技巧以及功能大家自学吧，毕竟这只是个模板的使用样例，不是 L^AT_EX 教案。

6.3.3 长表格

呐，在开始长表格之前，我们应该怎么样呢？对，先说点废话。为什么呢？你想啊，既然是长表格，肯定是能够跨页存在的，不说点废话把它顶下去，顶到换页，咋能对得起它的 NB 功能呢，是吧。

咳咳，严肃点，主角登场了。表 6.7 就是这一小节的主角——长表格了。首先说明一下，这种表格如果放的数据太多的话，就不要在正文里面用了，放到附录里就可以了。

^②表格虽然难看，主要是为了让大家看效果，里面技巧选用。

表 6.6 填色表格

No.	Title	L-Title	R-Title
1		One	First
2		Two	Second
3		Three	Third

表 6.7 这是一个长表格

行号	标题 1	标题 2	标题 3	标题 4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16
5	17	18	19	20
6	21	22	23	24
7	25	26	27	28
8	29	30	31	32
9	33	34	35	36
10	37	38	39	40
11	41	42	43	44
12	45	46	47	48
13	49	50	51	52
14	52	54	55	56
15	57	58	59	60
16	61	62	63	64
17	65	66	67	68
18	69	70	71	72
19	73	74	75	76
20	77	78	79	80
21	81	82	83	84

接下页续表……

续表 6.7 这是一个长表格

行号	标题 1	标题 2	标题 3	标题 4
22	85	86	87	88
23	89	90	91	92
24	93	94	95	96
25	97	98	99	100

6.4 图片

6.4.1 普通图片的插入

如同表格一样，插图一般都用浮动体来控制，这样排出来的文章美观。^①比如，图 6.1 所示，是一个图片的样例。

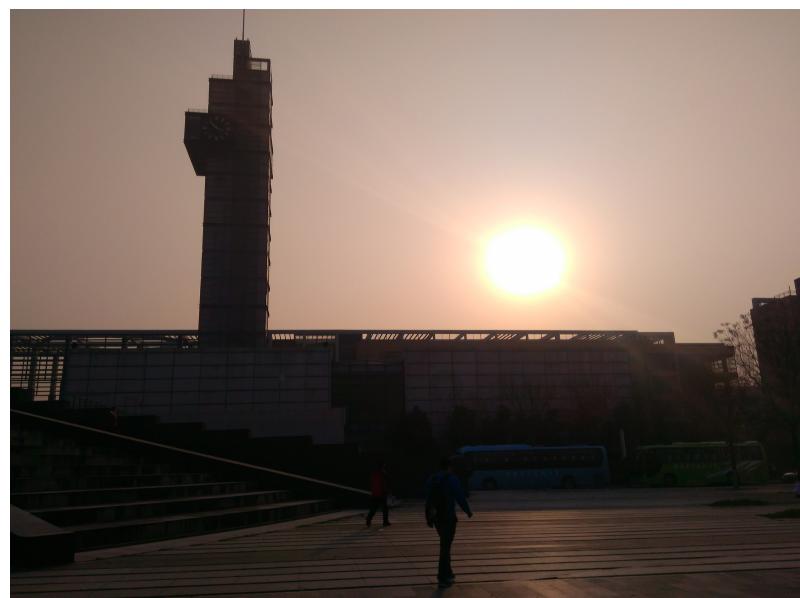


图 6.1 西电 1

6.4.2 图片并排插入

先来看第一个，两个图片分别一个标题，如图 6.2 和图 6.3.

再来看一个统一大标题带子标题的，如图 6.4 所示。

插图就介绍到这，其他知识自学。

^①本文涉及到的所有图片，除我校校徽等标记外，均为个人拍摄。



图 6.2 西电 2



图 6.3 西电 3



(a) 西电 4



(b) 西电 5

图 6.4 并排插图

6.5 公式

6.5.1 普通公式

先来看一个行内公式: $y = x + 1$.

下面是一个居中的公式:

$$f(X) = \sum_{i=1}^n \sin \frac{\pi}{2} x_i$$

看一个编号的公式, 如式(6-1):

$$f(X) = \sum_{i=1}^n \sin \frac{\pi}{2} x_i \quad (6-1)$$

6.5.2 复杂公式

- 多行公式

$$\begin{aligned} f(x) &= \sin(a+b) \\ &= \sin(a)\cos(b) + \cos(a)\sin(b). \end{aligned} \tag{6-2}$$

$$f(x) = \begin{cases} -x+1, & \text{if } x < 0; \\ x+1, & \text{if } x \geq 0; \end{cases} \tag{6-3}$$

• 矩阵

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}, \quad C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ \vdots & \ddots & \vdots \\ c_{31} & c_{32} & c_{33} \end{bmatrix}.$$

行列式也类似

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}.$$

• 其他公式

$$\mathbb{L} = \int_{x_1}^{x_2} \sqrt{1 + (y')^2} dx = \frac{1}{\alpha} \int_{x_1}^{x_2} y''(x) dx$$

6.6 休息一下

6.6.1 山水之间

山水之间

许嵩

昨夜同门云集 推杯又换盏
 今朝茶凉酒寒 豪言成笑谈
 半生累 尽徒然
 碑文完美有谁看

隐居山水之间 誓与浮名散

湖畔青石板上 一把油纸伞

旅人停步折花 淋湿了绸缎

满树玉瓣多傲然

江南烟雨却痴缠

花飞雨追 一如尘缘理还乱

落花雨 你飘摇的美丽

花香氤 把往日情勾起

我愿意 化浮萍躺湖心

只陪你 泛岁月的涟漪

6.6.2 念奴娇·赤壁怀古

念奴娇·赤壁怀古

苏轼

大江东去，浪淘尽，千古风流人物。

故垒西边，人道是：三国周郎赤壁。

乱石穿空，惊涛拍岸，卷起千堆雪。

江山如画，一时多少豪杰。

遥想公瑾当年，小乔初嫁了，雄姿英发。

羽扇纶巾，谈笑间樯橹灰飞烟灭。

故国神游，多情应笑我，早生华发。

人生如梦，一尊还酹江月。

这一章就写到这里吧，下一章重点介绍两个内容

第七章 两个重点

这一章，会涉及到环境以及参考文献两个重点东西。

7.1 环境

7.1.1 定理类环境

命题 7.1： 这是一个命题。

它还可以这么写，所有定理类环境，都可这么写。

命题 7.2 (命题名)： 由命题 7.1，这也是一个命题。

假设 7.1： 距离水平面 $100 m$ 以内，重力加速度 g 是不变的。

引理 7.1 (法图引理)： 设 (S, Σ, μ) 为一个测度空间， $(f_n)_{n \geq 0}$ 是一个实值的可正值函数列。那么：

$$\int_S \liminf_{n \rightarrow \infty} f_n d\mu \leq \liminf_{n \rightarrow \infty} \int_S f_n d\mu.$$

其中函数极限是在逐点收敛的意义上的极限，函数取值和积分可以是无限大的。

定理 7.1： 若 $\lim_{n \rightarrow \infty} x_n = a$ ，则 $\{x_n\}$ 的任何子列 $\{x_{n_k}\}$ 都收敛于 a .

证明 因 $\lim_{n \rightarrow \infty} x_n = a$ ，对任意给定的 $\varepsilon > 0$ ，必定存在正整数 N ，当 $n > N$ 时，有

$$|x_n - a| < \varepsilon.$$

今取 $K = N$ ，则对一切 $k > K$ ，有 $n_k > n_K = n_N \geq N$ ，这时就有

$$|x_{n_k} - a| < \varepsilon.$$

□

7.1.2 算法环境

如算法 7.1 所示，详细使用方法见文档 algorithmicx（此条参考自中国科学院大学学位论文 LaTeX 模板）

算法 7.1 Euclid's algorithm

```

1: procedure EUCLID(a, b)                                ▷ The g.c.d. of a and b
2:   r  $\leftarrow a \bmod b$ 
3:   while r  $\neq 0$  do                                     ▷ We have the answer if r is 0
4:     a  $\leftarrow b$ 
5:     b  $\leftarrow r$ 
6:     r  $\leftarrow a \bmod b$ 
7:   end while
8:   return b                                         ▷ The gcd is b
9: end procedure

```

7.1.3 代码环境

```

1 #include<stdio.h>
2
3 int main()
4 {
5     printf("Hello World!");
6     return 0;
7 }

```

代码 7.1 Java 代码

```

1 package com.stick.test;
2
3 /**
4 * 公主类;
5 * 类名必须与文件名相同
6 */
7 public class Test{
8     public static void main(String[] args) {

```

```
9         System.out.println("Hello World!");  
10    }  
11 }
```

```
x<-c(1,2,3,4,5,6)  
y<-sin(x)  
lines (x,y)
```

7.2 参考文献的引用

以下为常用的参考文献。

期刊^[? ?], 专著(无页码)^[? ? ?], 专著(含页码)^[? ? ?], 论文集^[? ?]。

硕士^[?]/博士论文^[?], 科技报告^[?]。

以下参考文献, 原有 plain 风格中并无, 为 Stick 本人自己写的。

网络内容引用^[?], 译著^[?]

此外, 多说一句, 本文模板加载的 `natbib` 宏包使得 `\cite` 命令还支持排序等功能。

比如下列文献^[? ? ? ?]

致 谢

毕业论文的完成意味着在西安电子科技大学四年的本科学习生涯即将结束，我将无比珍惜这段求学经历。

首先，我十分感谢我的指导老师鱼滨教授。在论文开题、中期、答辩等各个环节悉心指导，提出了宝贵的意见。在他的指导下，我的论文越来越趋于完善，撰写能力也得到了明显的提升！感谢大学期间，家人对我学业上的支持和精神上的鼓励。同时，感谢所有关心帮助过我的同学和无私奉献的老师们！在今后的研究生学习和工作中，我将加倍努力，以期取得更大的成绩回报他们。

参考文献

- [1] 代飞, 刘国志, 李章, 等. 微服务技术: 体系结构、通信和挑战[J]. 应用科学学报, 2020, 38(05):761-778.
- [2] 邢贞明, 李登辉, 潘博. 微服务架构与容器技术探析[J]. 金融科技时代, 2021(02):66-69.
- [3] 姚刚, 王从镔, 吴海莉. 基于 Docker 技术的微服务架构探析[J]. 信息系统工程, 2020(04):127-128.
- [4] 孙浩. 产品管理之敏捷说[J]. 中国信息化, 2020:44+47.
- [5] 李闯. 中台系统敏捷开发管理的研究[D]. 北京邮电大学, 2020.
- [6] 冯志勇, 徐砚伟, 薛霄, 等. 微服务技术发展的现状与展望[J]. 计算机研究与发展, 2020, 57(05):1103-1122.
- [7] 黄显琛. 基于微服务架构的系统设计与实现[J]. 信息技术与信息化, 2020(11):16-17.
- [8] 于洋. RESTful 架构风格及其演变与发展[J]. 计算机时代, 2020(04):10-13.
- [9] 吴旭君. 基于 Spring Cloud 微服务架构的数据服务系统的实践[J]. 计算机产品与流通, 2020(03):93.
- [10] 郭致远, 魏银珍. 基于 Spring Cloud 服务调用的设计与应用[J]. 信息技术与网络安全, 2019, 38(02):87-91.
- [11] 郭致远, 魏银珍. 基于 Spring Cloud 服务调用的设计与应用[J]. 信息技术与网络安全, 2019, 38:87-91.
- [12] 方永敢. 微服务架构的研究及小区生活服务平台的实现[D]. 电子科技大学, 2020.
- [13] 庄夏. API 网关架构设计实例[J]. 信息系统工程, 2018(05):99-100.
- [14] 姚刚, 吴海莉, 王从镔. 浅析微服务架构 API 网关的作用[J]. 信息系统工程, 2020(12):16-18.
- [15] 宋欣. 基于微服务架构的工业互联网网关的设计与实现[D]. 北京交通大学, 2020.
- [16] 廖俊杰, 陶智勇. 微服务 API 网关的设计及应用[J]. 自动化技术与应用, 2019, 38:85-88.
- [17] 吴润. 基于 API 网关的微服务组合策略研究[J]. 数码世界, 2019(03):84-86.
- [18] 杨秦. 基于微服务架构的云平台服务端的设计与实现[D]. 电子科技大学, 2020.
- [19] 张虎, 张秋萍. 基于 KONG 的 API 集成系统的设计与实现[J]. 计算机技术与发展, 2019, 29:102-106.
- [20] 李浪. 基于微服务网关 Zuul 的 TCP 功能扩展和限流研究[D]. 武汉理工大学, 2019.
- [21] 吴龙波. 基于 Oauth2.0 构建空管云数据中心的资源认证授权机制[J]. 科技传播, 2020, 12:

168-169.

- [22] 全野. 基于 NoSQL 数据库的系统设计与开发[D]. 南京邮电大学, 2018.
- [23] 梁力源. 基于 NoSQL 存储系统的研究与应用[D]. 重庆交通大学, 2016.
- [24] 唐文慧. 内存 NoSQL 系统 Redis 性能优化研究与实现[D]. 国防科技大学, 2017.
- [25] 杜丽娟. 关系型数据库与 NoSQL 数据库的性能对比[J]. 智能计算机与应用, 2017, 7(03): 132-134.
- [26] 陆地, 储蓄蓄, 陶静联, 等. 基于阿里云 OSS 的传统文件中心改造[J]. 有线电视技术, 2019 (06):99-101.
- [27] 丁岚, 周起如, 王英明, 等. 基于 Velocity 的代码生成器设计与实现[J]. 电脑编程技巧与维护, 2019(08):33-34+44.
- [28] 齐敬佩. 检测报告和企业报表自动生成系统的设计与实现[D]. 北京邮电大学, 2020.
- [29] 程文清, 赵建立, 孔英会. 基于 B/S 架构的开放实验室管理系统设计与实现[J]. 中国现代教育装备, 2020:31-33.
- [30] 王志文. Vue+Elementui+Echarts 在项目管理平台中的应用[J]. 山西科技, 2020, 35(06):45-47.
- [31] 夏睿. 综合运维平台中运维流程管理子系统的设计与实现[D]. 南京大学, 2019.
- [32] 赵一品. 基于 Spring Boot 和 MyBatis 的银行知识库管理系统的应用与实践[D]. 山东大学, 2020.
- [33] 杨德宇. 面向微服务架构的软件可维护性质量模型研究[D]. 南京大学, 2020.
- [34] 梁光瑞, 魏国, 杨光. 微服务架构与容器技术的应用集成实践[J]. 科技创新与应用, 2020: 166-167.
- [35] 丁超. 一种分布式系统自动化集成部署的实现方法[D]. 山东大学, 2020.

附录 A 数据

这里是附录的数据部分，其实是瞎写；来看个公式编号对不对，如式(A-1)所示。

$$h_i = \frac{\max_{j=1}^N \left\{ \frac{c_j}{f_j} \right\} - \frac{c_i}{f_i}}{\max_{j=1}^N \left\{ \frac{c_j}{f_j} \right\} - \min_{j=1}^N \left\{ \frac{c_j}{f_j} \right\}} \quad (\text{A-1})$$

A.1 放松一下

A.1.1 惊鸿一面

惊鸿一面^[A1]

许嵩

翻手为云 覆手为雨

金盆洗手止风雨

不恋红尘 却难舍回忆

每一段都有你

年少初遇 常在我心

多年不减你深情

江山如画 又怎能比拟

你送我的风景

柳下闻瑶琴 起舞和一曲

仿佛映当年 翩若惊鸿影

谁三言两语 撩拨了情意

谁一颦一笑 摆曳了风景

纸扇藏伏笔 玄机诗文里

紫烟燃心语 留香候人寻

史书列豪杰 功过有几许

我今生何求 唯你

远山传来清晨悠然的曲笛

晓风掠走光影

残月沉霜鬓里

有了你

恩怨都似飞鸿踏雪泥

A.1.2 无题

无题

李商隐^[?]

相见时难别亦难，东风无力百花残。

春蚕到死丝方尽，蜡炬成灰泪始干。

晓镜但愁云鬓改，夜吟应觉月光寒。

蓬山此去无多路，青鸟殷勤为探看。

A.2 代码

代码 A.1 C++ code

```

1  /*
2   Program: Hello world;
3   Author: Stick Cui;
4   Time: 2015/12/04.
5   */
6   #include<stdio.h>
7
8   int main()
9   {
10      printf("Hello world!");

```

```
11     return 0;  
12 }
```

Java code

```
1 /**
2 * Program: Hello world;<br>
3 * Time: 2015/12/04.
4 * @author Stick Cui
5 */
6 public class JavaTest{  
7     /**
8      * The main method.  
9      * @param args The parameter when the method is called.  
10     */  
11    public static void main(String[] args){  
12        System.out.println("Hello world!");  
13    }  
14 }
```

%floyd 算法通用程序，输入 a 为赋权邻接矩阵

%输出为距离矩阵 D, 和最短路径矩阵 path

```
function [D,path]=floyd(a)
```

```
n=size(a,1);
```

```
D=a;
```

```
path=zeros(n,n);
```

```
for i=1:n
```

```
    for j=1:n
```

```
        if D(i,j)~=inf
```

```
            path(i,j)=j;
```

```
    end
```

```

end

end

for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
                D(i,j)=D(i,k)+D(k,j);
                path(i,j)=path(i,k);
            end
        end
    end
end

% 配合 floyd 算法的后续程序, s 为源点, t 为宿点
% L 为长度, R 为路由

function [L,R]=router(D,path,s,t)
L=zeros(0,0);
R=s;
while 1
    if s==t
        L=fliplr(L);
        L=[0,L];
        L=L(end);
        return
    end
    L=[L,D(s,t)];
    R=[R,path(s,t)];
    s=path(s,t);
    if s==0
        return
    end

```

end

参考文献

[A1] 许嵩. 惊鸿一面 [A]. 海蝶音乐. 不如吃茶去 [C]. 北京: 北京海蝶音乐有限公司, 2014, 8.