JiCode

JiCode

——敏捷开发管理平台

小组成员：

2151641　王佳垚

2152034　吴　杭

2153282　顾雨杨

2154298　王　颖

目录

一、 项目目的

软件研发过程管理，是一条复杂的管理链条，在当今的商业环境中，敏捷项目管理方法已经成为了高效的项目管理方式之一。敏捷方法特征包括：强调小规模的可交付结果为导向，采用迭代和增量的开放方式，鼓励与客户紧密联系以理解需求、获取反馈，具有高度适应性和灵活性。因此，越来越多的团队和组织采用敏捷方法组织、协调和管理项目。

然而，市场现有的敏捷项目管理工具具有复杂学习曲线和高昂的使用成本，对于新手来说不易上手，且多种复杂功能易分散项目管理者注意力，将时间花费在软件的使用而难以集中在软件开发的过程。我们针对现有敏捷管理工具的痛点，旨在开发一个以Scrum为核心的敏捷项目管理工具，具有账号管理、后台管理、项目管理和产品管理四模块功能，确保工具提供敏捷项目管理的核心功能，如需求获取、迭代管理、团队协作等，帮助团队提高效率、质量和透明性，更好进行敏捷开发。

二、 目标用户

1. 软件开发团队成员：他们使用该系统来协同开发、测试和管理项目。他们可以使用系统来管理项目计划和视图，进行待办需求项的管理、每次迭代管理以及发布管理，帮助适应需求变更，更快生成可交付产品功能。

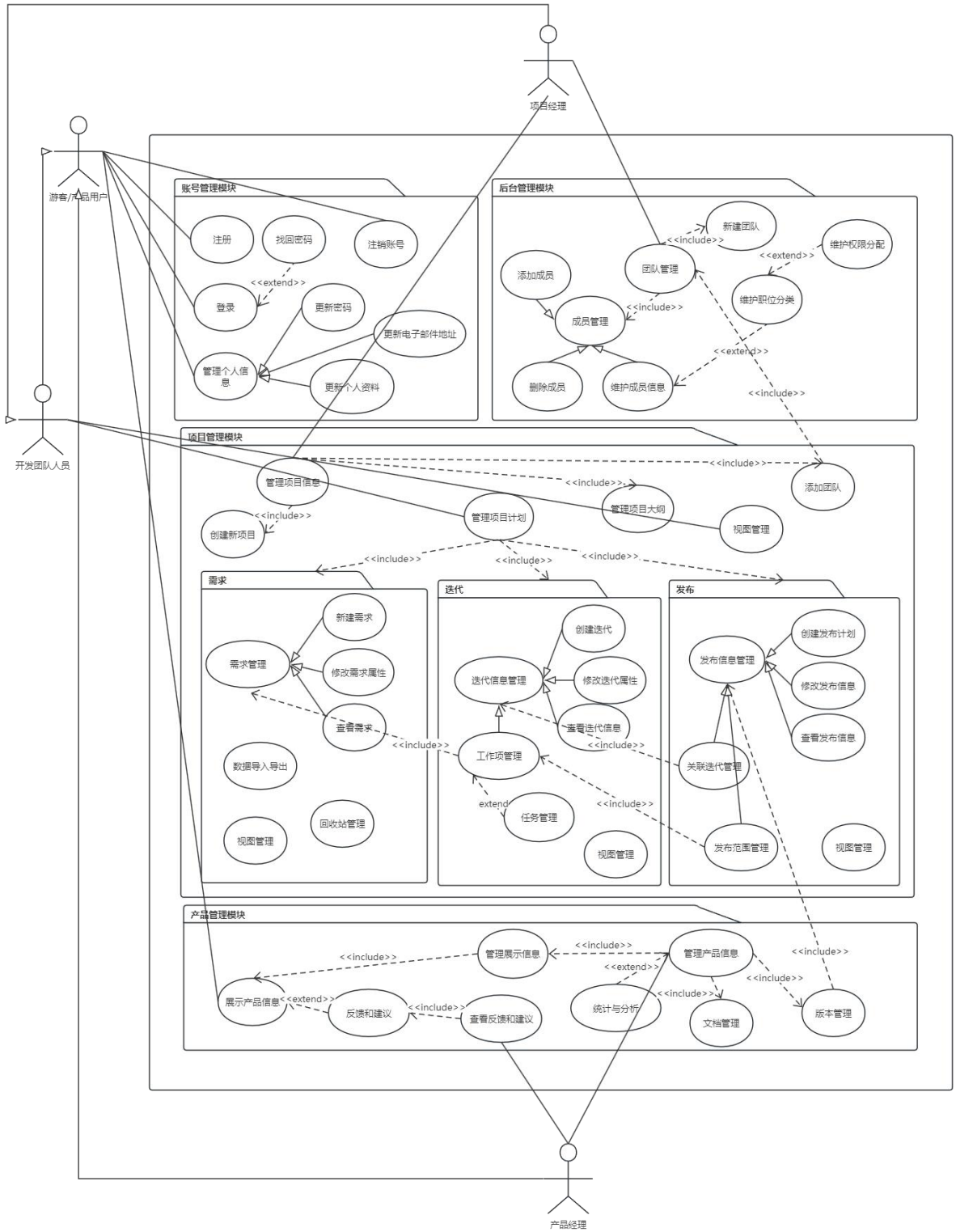
2. 项目经理：他们使用该系统来管理开发团队，包括新建团队、管理团队成员、维护成员职位等；还可以进行项目管理，包括创建项目、管理项目信息、管理项目计划、为项目添加参与团队等，帮助统筹团队开发任务，提高团队管理效率。

3. 产品经理：他们使用该系统来记录和管理产品信息、及时获取产品用户的反馈和建议，以便及时与利益相关者沟通。

4. 游客/产品用户：既包括产品使用者，也包括项目团队人员，可以进行系统的注册、登录以及个人信息的管理等操作。

三、 主要功能

JiCode系统主要提供账号管理、后台管理、项目管理、产品管理四个功能模块，全局用例图如下所示：



3.1 账号管理模块

账号管理模块为有效地管理用户账号提供了相关功能。用户可以创建新账号，通过注册和登录功能进行身份验证，以便安全地访问系统。用户还可以在其账号下管理个人信息，如更改密码、更新电子邮件地址和上传个人相关资料。最后，账号管理模块还允许用户注销账号，或者由系统管理员禁用账号，以维护账号数据的安全和隐私。

3.2 后台管理模块

后台管理模块提供了成员管理、团队管理和职位维护的功能。后台可以轻松管理不同成员角色，不同的角色拥有不同的职位权限。成员角色包括系统角色（管理员、普通成员、只读成员）和自定义角色，自定义角色可以设置为个性化职位。默认角色的设置可以简化项目成员的权限管理，确保一致性。而通过添加组织维护的自定义职位角色，用户可以满足特定需求，使团队管理更加灵活和高效。这些功能协助实现了全面的团队管理和权限控制，确保项目和团队的协作和管理得以顺畅进行。

3.3 项目管理模块

项目管理模块提供了一整套在线项目流程管理功能，支持多项目的敏捷开发。它允许用户轻松创建新项目，定义项目属性，添加成员，并提供了直观的项目访问和筛选功能。通过配置中心，用户可以管理项目的各个方面，包括项目计划、资源分配、任务追踪、风险管理等，从而帮助团队打造高效的一体化研发平台，提升研发协作效率。同时针对项目进展中的需求、迭代和发布提供了相应功能。

3.3.1 需求

需求管理部分提供了全面的工作项管理和追踪功能，为项目管理和团队协作提供了重要支持。用户可以轻松新建各种工作项，包括需求和缺陷，以及进行批量创建操作。此模块还支持工作项的查看和编辑，视图管理，包括自定义视图的创建，方便用户根据需要筛选和查看工作项。回收站功能允许管理已删除工作项，而视图设置使用户可以在树状和平铺展示之间灵活切换。此外，提供了导入和导出需求功能以便数据管理和备份。

3.3.2 迭代

迭代管理部分为项目管理产品提供了全面的迭代管理功能。用户可以通过该模块轻松地查看、创建和编辑项目内的所有迭代信息。支持创建分组和类别，以更便捷地组织和访问迭代。新建迭代操作允许用户指定迭代的名称、所属项目、负责人等信息。此外，用户可以编辑和删除现有迭代。一旦迭代规划工作完成，用户可以开始迭代工作并使用看板视图、任务板、故事板以及工作项列表视图来管理和跟踪工作项。在迭代工作结束后，用户可以标记迭代为已完成，将未完成的工作项移动到待分配工作项或其他未完成的迭代，或者将它们全部标记为已完成。这一功能丰富的迭代管理模块支持项目团队更加高效地进行迭代工作的规划和执行。

3.3.3 发布

发布管理部分提供了一套完整的功能，以便项目团队有效地管理发布流程。用户可以在项目详情页面创建新的发布，定义发布名称、所属项目、负责人、开始发布时间等信息，并可以选择分组和类别。一旦发布创建完成，用户可以进入发布概览页面，查看发布的基本信息、进度、日志、关联迭代和测试计划。用户还可以修改发布阶段和编辑发布日志，以便跟踪发布进展。

发布管理模块还支持关联迭代，以便将特定迭代内容与当前发布关联。用户可以选择要关联的迭代，从而明确发布的范围。发布范围功能允许用户查看发布中要发布的工

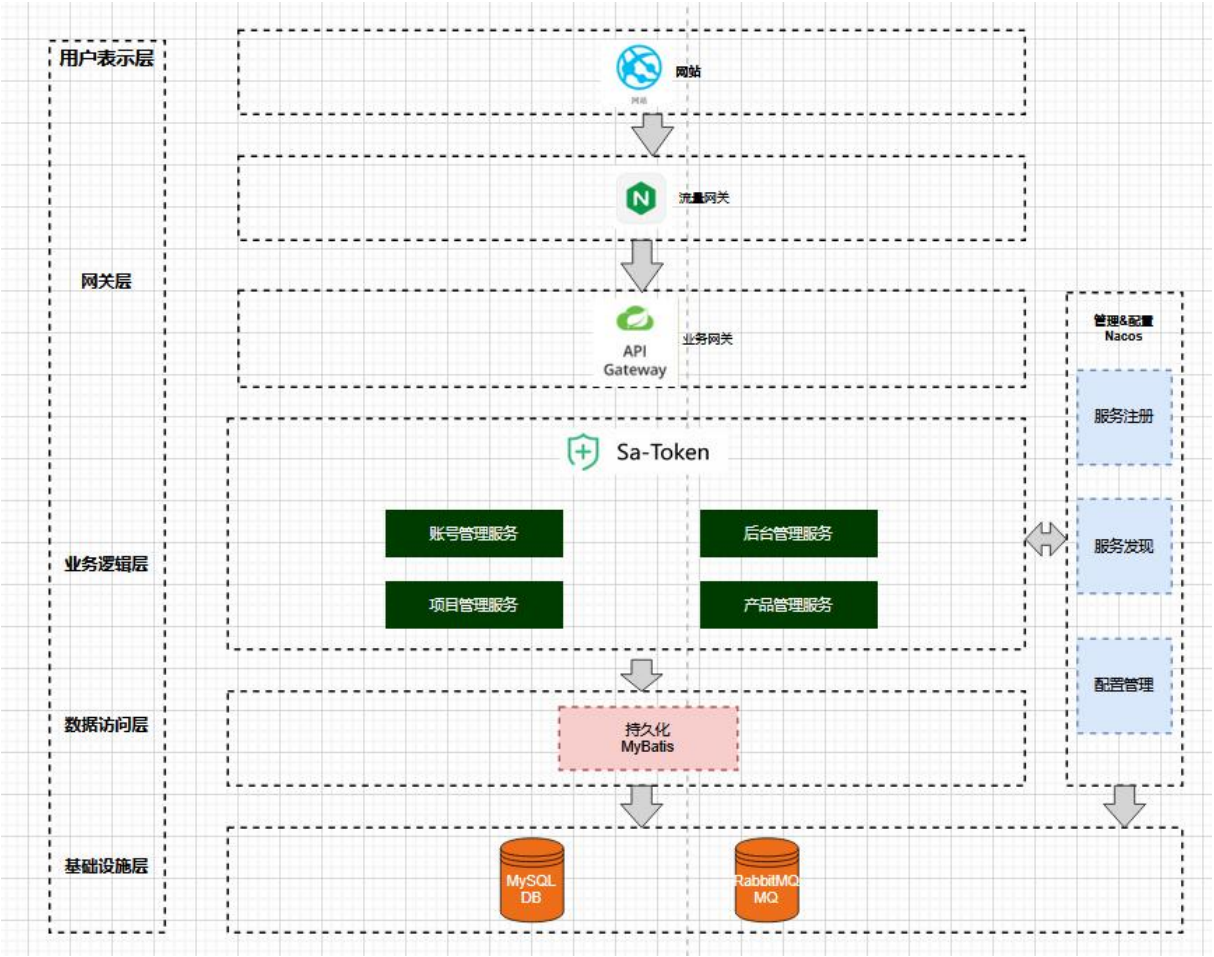
作项内容，并规划更多工作项至当前发布。

用户可以通过发布计划查看项目中的所有发布计划，以及进入发布页面进行编辑和删除操作。鼠标悬停在发布名称后方，可以快速进行编辑和删除发布。

3.4 产品管理模块

产品管理模块汇集并展示了项目完成后的产品信息。产品信息包括产品名称、版本、描述、发布日期等详细信息，方便用户了解产品的特点和演进历程。此外，产品管理模块提供版本管理，以记录发布版本和更新版本的变化，以及文档管理，用于存储用户手册、技术文档等关键材料。用户可以通过此模块提供反馈和建议，包括问题报告、功能请求等，以改进产品。通过统计和分析功能，团队可以评估产品的性能、用户反馈和问题报告等指标，为产品的不断改进提供数据支持。

四、 初步逻辑架构



项目初步逻辑架构采用微服务架构，分为五层，分别是用户表示层、网关层、业务逻辑层、数据访问层和基础设施层。

将领域驱动设计的思想应用于微服务架构体系，确定限界上下文以及划分领域，

本项目中的领域划分如下：

- 核心子域： 项目管理、团队管理
- 支撑子域： 需求管理、项目迭代，产品发布
- 通用子域：个人信息

根据子域的划分确定限界上下文

- 账号管理
- 团队管理
- 项目管理
- 产品管理

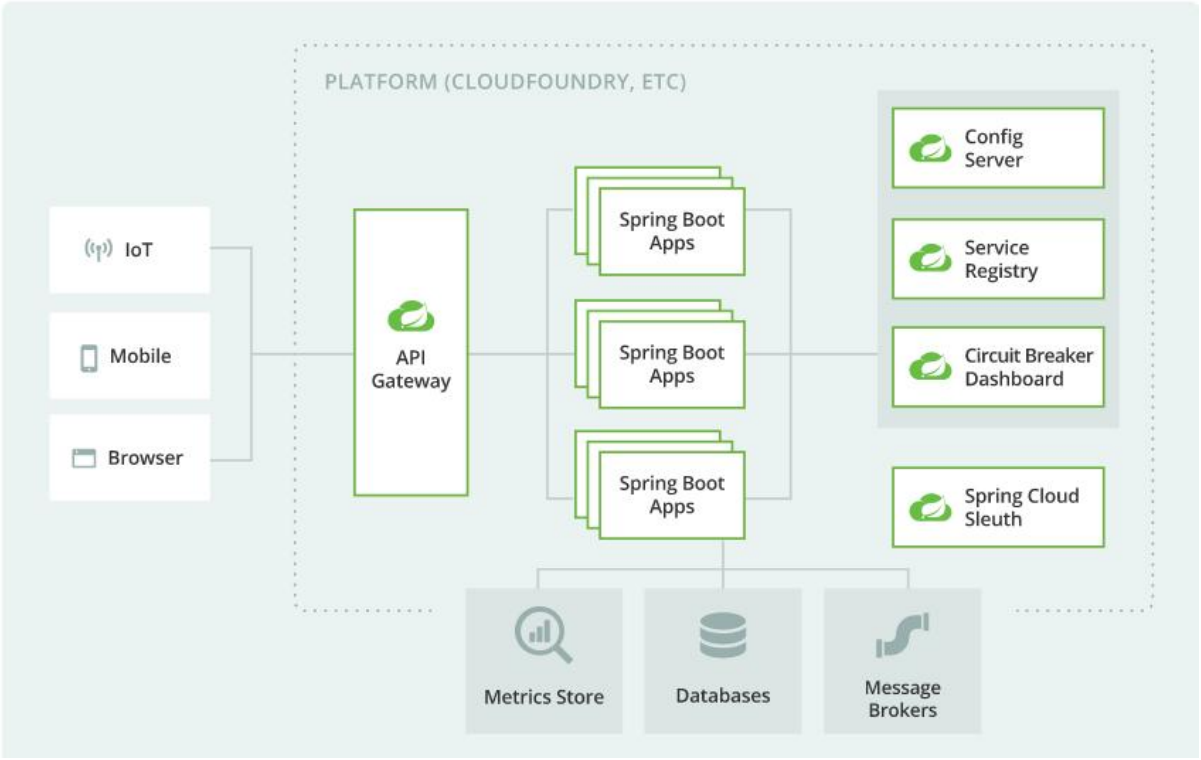
本项目中将每个限界上下文都看作是一个微服务。每个微服务中分为 3 层：

1. 接口层：接口层是限界上下文与外部世界进行交互的入口，它负责接收外部请求，并将请求转发给应用层进行处理。
2. 应用层：应用层是限界上下文的核心层，负责处理业务逻辑和协调领域对象的交互。
3. 领域层：域层是限界上下文的核心领域逻辑层，负责表达和实现业务领域的核心概念和规则。包含领域模型、领域服务、领域事件等。

五、 候选开发技术

5.1 微服务框架

采用 Spring Cloud 作为微服务框架，这是一个开源的微服务解决方案，它基于 Spring Boot 并提供了一系列组件和工具，用于构建分布式系统和微服务架构。Spring Cloud 提供了服务注册与发现、负载均衡、熔断器、配置管理等功能，使得开发人员可以更轻松地构建和管理分布式系统。



5.2 JS 框架

前端使用 Vue.js 作为 JavaScript 框架，Vue.js 是一个轻量级、高性能的前端框架，具有简单易学、灵活、可扩展等特点。它提供了响应式的数据绑定、组件化开发、虚拟 DOM 等功能，使得前端开发更加高效和可维护。

5.3 持久层框架

使用 Mybatis 作为持久层框架，Mybatis 是一个优秀的持久层解决方案，它提供了灵活的 SQL 映射配置和强大的 SQL 执行能力。通过与数据库交互，Mybatis 可以实现高效的数据访问和操作，提供了良好的性能和可维护性。避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集的过程，减少了代码的冗余。

Mybatis 具有以下特点：

1. 简单易用：MyBatis 的配置相对简单，易于上手。它提供了直观的 XML 或注解方式来定义 SQL 映射，使得开发人员能够更加灵活地控制 SQL 语句的执行。
2. 灵活性：MyBatis 允许开发人员使用原生的 SQL 语句，可以编写复杂的查询逻辑，满足各种不同的需求。同时，MyBatis 也支持动态 SQL，可以根据条件动态生成 SQL 语句，提高了开发的灵活性。
3. 易于集成：MyBatis 可以与各种主流的 Java 框架（如 Spring、Spring Boot 等）无缝集成，提供了丰富的整合支持。它与数据库之间的交互是通过 SQL 语句进行的，因此可以与任何数据库兼容。
4. 高性能：MyBatis 采用了预编译的方式执行 SQL 语句，减少了 SQL 解析的开销。同时，MyBatis 还提供了一级缓存和二级缓存机制，可以有效地提高数据库访问性能。

六、 中间件

6.1 负载均衡： Nginx

使用 Nginx 作为负载均衡器，Nginx 是一个高性能的 Web 服务器和反向代理服务器，它可以将请求分发到多个后端服务器，实现负载均衡和高可用性。特点是占有内存少，并发能力强，支持热部署，可以做到 7*24 不间断运行，几个月都不需要重新启动。

功能：

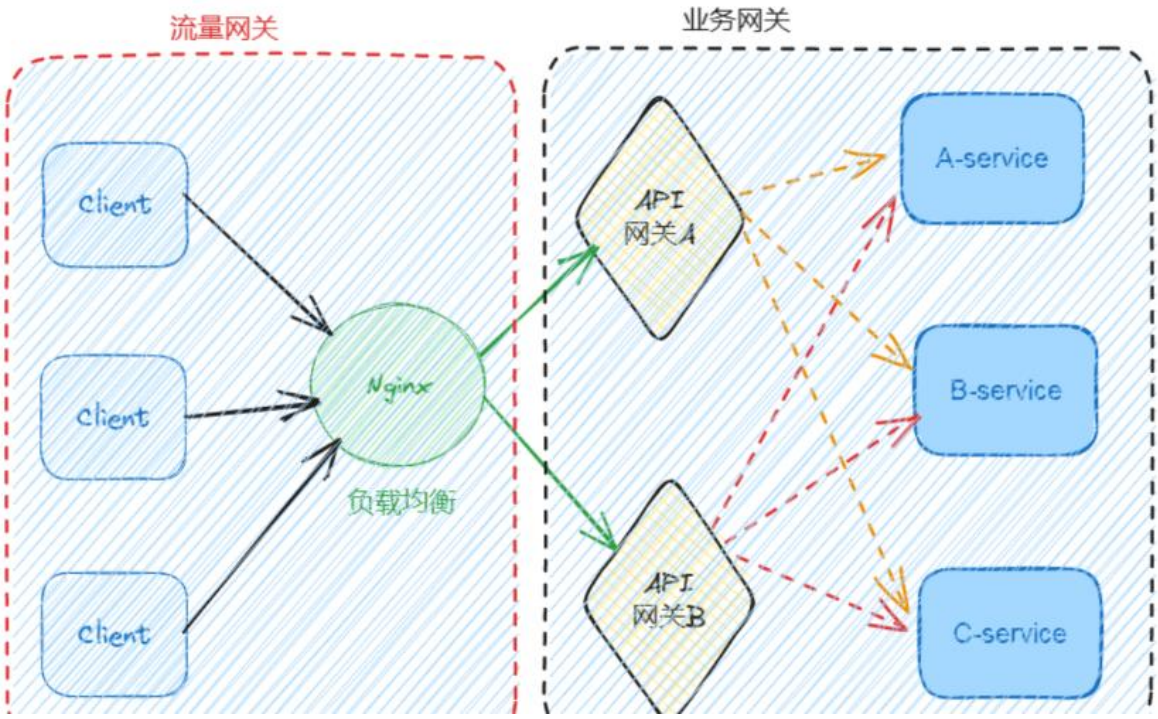
1. 反向代理：客户端对代理是无感知的，因为客户端不需要任何配置就可以访问，我们只需要将请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据后，在返回给客户端，此时反向代理服务器和目标服务器对外就是一个服务器，暴露的是代理服务器地址，隐藏了真实服务器 IP 地址。
2. 负载均衡：分摊到多个操作单元上进行执行，例如 Web 服务器、FTP 服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。Nginx 给出三种关于负载均衡的方式：轮询法、weight 权重模式、ip_hash。
3. 动静分离：为了加快网站的解析速度，可以把动态页面和静态页面交给不同的服务器来解析，来加快解析速度，提高请求的访问效率，降低原来单个服务器的压力。

6.2 网关路由： Gateway

采用 Gateway 作为网关路由，Gateway 是 Spring Cloud 提供的网关服务，它可以接收所有外部请求，并根据预定义的规则将请求转发到相应的微服务。Gateway 提供了路由、过滤、熔断等功能，可以有效管理和保护微服务。

在微服务架构里，服务的粒度被进一步细分，各个业务服务可以被独立的设计、开发、测试、部署和管理。这时，各个独立部署单元可以用不同的开发测试团队维护，可以使用不同的编程语言和技术平台进行设计，这就要求必须使用一种语言和平 台无关的服务协议作为各个单元间的通讯方式。

在与 Nginx 结合使用的时候，Nginx 充当流量网关，专注于全局的 Api 管理策略；Gateway 充当业务网关，针对具体的业务需要提供特定的流控策略、缓存策略、鉴权认证策略等等。



6.3 消息队列： RabbitMQ

使用 RabbitMQ 作为消息队列，RabbitMQ 是一个可靠、可扩展的消息代理，它可以在分布式系统中处理大量的异步消息。通过使用 RabbitMQ，可以实现微服务之间的解耦和异步通信，提高系统的可靠性和性能。

功能：

1. 异步处理：采用发布订阅的模式，发送者可以将消息发送到队列中，而不需要等待接收者的响应。接收者可以在合适的时间从队列中获取并处理消息，实现了解耦和异步处理。

2. 应用解耦：复杂的应用里会存在多个子系统，如果各个子系统之间的耦合性太高，整体系统的可用性就会大幅降低。多个低错误率的子系统强耦合在一起，得到的是一个高错误率的整体系统。通过使用消息队列，可以将不同的组件或服务拆分成独立的模块，并通过消息队列进行通信。这样可以降低系统之间的依赖程度，每个模块可以独立演进和升级，而不会影响其他模块的正常运行。

3. 流量削峰：将请求发送到消息队列中，而不是直接处理请求。这样可以将高峰期的请求暂时存储在队列中，然后按照系统处理能力的速度逐个进行处理，避免了瞬时大量请求对系统造成的压力。

6.4 服务注册： Nacos

选择 Nacos 作为服务注册中心，Nacos 是一个开源的服务发现和配置管理系统，它提供了服务注册、服务发现和健康检查等功能，可以帮助微服务实现自动化的服务注册和发现。

功能：

1. 服务发现与健康检测：使服务更容易注册，并通过 DNS 或 HTTP 接口发现其他服务，还提供服务的实时健康检查，以防止向不健康的主机或服务实例发送请求。

2. 动态配置服务：以中心化、外部化和动态化的方式管理所有环境的应用配置和服务配置，消除了配置变更时重新部署应用和服务的需要，让配置管理变得更加高效和敏捷。

3. 动态 DNS 服务：支持权重路由，更容易地实现中间层负载均衡、更灵活的路由策略、流量控制以及数据中心内网的简单 DNS 解析服务。

4. 服务及其元数据管理：从微服务平台建设的视角管理数据中心的所有服务及元数据，包括管理服务的描述、生命周期、服务的静态依赖分析、服务的健康状态、服务的流量管理、路由及安全策略、服务的 SLA 以及最首要的 metrics 统计数据。

6.5 数据库：MySQL

采用 MySQL 作为数据库，MySQL 是一个开源的关系数据库管理系统，具有高可靠性、稳定性和性能优势。MySQL 广泛用于 Web 应用程序和分布式系统中，提供了可靠的数据存储和处理能力。

在微服务架构中，可以使用 MySQL 实现以下操作：

1. 数据库服务的拆分：根据业务需求，将原本单一的数据库拆分为多个数据库或多个数据表。每个微服务可以有自己独立的数据库或数据表，根据业务职责进行拆分，从而实现微服务之间的隔离和解耦。

2. 数据库访问层的封装：每个微服务可以有自己的数据库访问层，封装了对数据库的访问和操作。通过封装，可以隐藏底层数据库的细节，提供统一的接口给上层服务调用，从而降低对数据库的直接依赖。

3. 事务管理：在微服务之间的操作涉及到多个数据库或数据表时，需要保证事务的一致性。可以使用分布式事务管理框架，如 Seata、XA 等来管理跨数据库的事务，确保微服务之间的操作具有原子性和一致性。

6.6 身份认证：Sa-Token

选择 Sa-Token 作为身份认证框架，Sa-Token 是一个开源的 Java 身份认证框架，它提供了用户认证、权限控制、会话管理等功能。通过使用 Sa-Token，可以实现用户身份的认证和授权，保护系统的安全性。

七、 平台

7.1 后端

选择 Java EE 作为后端开发平台，这是一个成熟且广泛使用的平台，具有丰富的生态系统和强大的开发工具。Java EE 提供了一系列标准化的 API 和规范，使得开发人员能够快速构建可靠、安全和高性能的企业级应用程序。广义的 Java EE 包含各种框架，

其中最重要的就是 Spring 全家桶。Spring 诞生之初是为了改进 Java EE 开发的体验，后来逐渐成为了 Java Web 开发的实际标准。

作为后端开发平台，Java EE 具有以下优势：

1. 广泛的生态系统：Java EE 有一个庞大的生态系统，包括众多的开发工具、框架和第三方库。这些工具和框架可以帮助开发者提高开发效率，加快应用的部署和发布，同时也提供了丰富的解决方案和组件，支持各种企业级应用的开发和部署。

2. 可移植性：Java EE 采用了 Write Once, Run Anywhere（一次编写，到处运行）的原则，即使用 Java EE 开发的应用程序可以在各种操作系统和硬件平台上运行。这种可移植性使得开发者可以更灵活地部署和迁移应用，减少了对特定平台的依赖。

3. 高度可扩展性：Java EE 提供了丰富的 API 和规范，支持各种扩展和定制。开发者可以根据应用的需求选择合适的组件和技术进行开发，从而实现高度可扩展的应用。同时，Java EE 还支持集群和分布式部署，可以实现应用的水平扩展和负载均衡。

4. 事务管理：Java EE 提供了强大的事务管理机制，支持对数据库操作和其他资源的事务控制。开发者可以使用 Java EE 的事务管理机制来确保应用的数据一致性和可靠性，同时也可以减少开发者对事务处理的工作量。

5. 安全性：Java EE 提供了多种安全机制和特性，如身份认证、授权、安全传输等。开发者可以使用这些机制来保护应用程序的安全性，防止未经授权的访问和数据泄露。Java EE 还支持常见的安全标准和协议，如 SSL、TLS 等，提供了安全可靠的通信机制。

在本项目中的应用：

1. web 应用开发： Java EE 提供了一系列的技术和规范，用于开发 Web 应用程序。例如，使用 Java Servlet 和 JavaServer Pages（JSP）进行服务器端的请求处理和动态页面生成；使用 JavaServer Faces（JSF）构建用户界面；使用 Java API for WebSocket 实现实时通信等。

2. 数据库集成： 可以进行对象关系映射（ORM），也可以直接使用 JDBC 访问数据库。本项目中使用 MyBatis 来简化数据库操作。

3. 安全管理：Java EE 提供了强大的安全机制和 API，用于实现用户认证、授权和访问控制等安全管理功能。可以使用 Java Authentication and Authorization Service（JAAS）进行认证和授权；使用 Java Security API 进行数据加密和解密。该项目中选择使用 Sa-Token 进行用户认证和授权等操作。

7.2 部署

选用 Docker 作为部署平台。Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。使用 Docker 进行开发和交付具有以下优点：

1. 快速、一致地交付应用程序：容器非常适合持续集成和持续交付（CI / CD）工作流程，简化了开发的生命周期。

2. 响应式部署和扩展： Docker 是基于容器的平台，允许高度可移植的工作负载。

Docker 容器可以在开发人员的本机上，数据中心的物理或虚拟机上，云服务上或混合环境中运行。

3. 在同一硬件上运行更多工作负载： Docker 轻巧快速。它为基于虚拟机管理程序的虚拟机提供了可行、经济、高效的替代方案，因此可以利用更多的计算能力来实现业务目标。Docker 非常适合于高密度环境以及中小型部署，而可以用更少的资源做更多的事情。

八、 项目进度安排