软件体系结构文档(SAD)

小组：这是什么队

说明:

1.《软件(结构)设计说明》(SDD)描述了计算机软件配置项(CSCI 的设计。它描述了 CSCI 级设计决策、CSCI 体系结构设计(概要设计)和实现该软件所需的详细设计。SDD 可用接口设计说明 IDD 和数据库(顶层)设计说明 DBDD 加以补充。

2.SDD 连同相关的 IDD 和 DBDD 是实现该软件的基础。向需方提供了设计的可视性，为软件支持提供了所需要的信息。

3.IDD 和 DBDD 是否单独成册抑或与 SDD 合为一份资料视情况繁简而定。

目录

1 引言

1.1 标识

标识号：XPSL-1234
标题：IT 乐园
缩略词语：IT Playground
版本号：v1.1.1
发行号：20230303

1.2 系统概述

本项目为博客网站项目，网站使用的是前后端分离的构建方式，前端使用 vue 框架来进行搭建，后端则通过 Nodejs 搭建，网站使用 python 实现搜索逻辑，使用 mysql 作为网站的数据库；网站现仍处于开发状态，前后端和数据库均处于相对独立的开发状态，尚未进入连接运行状态。网站的前后端体系结构也在逐步构建，前端利用 VUE 框架及其相关库来实现前端页面的各种功能和交互，后端使用 python，Node.js 和 mysql 等，来实现数据检索等后端信息处理功能,其他相关文档有：可行性研究报告(FAR)，软件规格需求说明(SRS)。

1.3 文档概述

本文档适用于博客网站项目的设计、开发、测试和维护阶段。本文档旨在描述博客网站项目的软件体系结构，为项目的设计、开发、测试和维护提供一个指导和参考。进一步明细项目开发的具体要求，对于要实现的系统体系有更加深入的设计

1.4 基线

本系统设计说明书是基于先前编写的需求规格说明书（SRS）和软件架构设计文档（SAD）编写的。SRS 定义了系统的需求和规范，SAD 定义了系统的整体结构和组件之间的关系。

2 引用文件

本章应列出本文档引用的所有文档的编号、标题、修订版本和日期。本章也应标识不能通过正常的供货渠道获得的所有文档的来源。

- 博客网站项目需求文档

- 技术规范和标准
- 相关的设计文档和架构指南

3 CSCI 级设计决策

1. CSCI 1: 表示层组件

决策：采用响应式 Web 设计，以支持跨平台、自适应和可访问性要求。
理由：响应式 Web 设计可以保证用户界面在不同的设备和屏幕大小下均可正常显示，以满足用户的使用习惯和需求。

2. CSCI 2: 业务逻辑层组件

决策：使用 MVC（模型-视图-控制器）架构模式，以实现业务逻辑的分离和复用。
理由：MVC 架构模式可以将业务逻辑、用户界面和数据存储分开，以便各个组件之间进行独立的开发和测试，从而提高系统的可维护性和可扩展性。

3. CSCI 3: 数据访问层组件

决策：采用 ORM（对象关系映射）框架，以实现对象和数据库之间的映射和持久化。
理由：ORM 框架可以将对象和数据库之间的关系映射为关系型数据表，以方便对数据进行操作和管理，同时减少手动编写 SQL 语句的工作量。

4. CSCI 4: 安全设计

决策：采用 JWT（JSON Web Token）认证和 OAuth 2.0 授权框架，以实现用户身份验证和授权管理。
理由：JWT 认证可以将用户信息和访问令牌进行编码和解码，以保证用户身份的安全性和隐私性。OAuth 2.0 授权框架可以定义不同的用户角色和权限级别，以限制用户对敏感数据的访问和修改。

5. CSCI 5: 部署设计

决策：采用容器化技术，如 Docker 和 Kubernetes，以实现应用程序的可移植性和可伸缩性。
理由：容器化技术可以将应用程序和依赖项打包为可移植的容器，以方便在不同的环境中部署和运行，同时可以利用 Kubernetes 等容器编排工具进行自动化的扩容和负载均衡。

4 CSCI 体系结构设计

4.1 体系结构

4.1.1 程序(模块)划分

模块	名称	标识符	功能	所包含的源标准名
用户界面模块	博客前端界面模块	BlogGUI	向用户展示博客的内容和界面，启动用户与博客系统的交互	HTML、CSS、JavaScript
业务逻辑模块	博客业务逻辑模块	BlogLogic	负责处理业务逻辑，包括用户文章管理、文章推荐管理等	Python、Spring 框架
数据访问模块	博客数据访问模块	BlogDAO	负责与数据库交互，包括用户文章存储和数据库访问接口设计	SQL、JDBC、Hibernate 框架
安全模块	博客系统安全模块	BlogSecurity	负责系统的安全设计，包括用户认证、权限控制和数据加密等	OAuth2、SSL、Spring Security 框架

4.1.1.1 用户界面模块子程序划分

子程序	名称	标识符	功能	所包含的源标准名
文章列表展示	文章列表展示子程序	ArticleList	将博客文章列表展示给用户，用户可以查看文章列表并点击文章访问文章详情	HTML、CSS、JavaScript
文章详情展示	文章详情展示子程序	ArticleDetail	将博客文章详情展示给用户，用户可以查看文章内容、评论和点赞等	HTML、CSS、JavaScript
用户登录	用户登录子程序	UserLogin	实现用户登录功能，用户可以使用用户名和密码登录博客系统	HTML、CSS、JavaScript、Java Servlet

				API
二手交易平台商品展示	二手交易平台商品展示子程序	SecondhandGoodsList	展示二手交易平台的商品列表，用户可以查看不同分类的商品并点击商品访问商品详情	HTML、CSS、JavaScript
二手交易平台商品详情展示	二手交易平台商品详情展示子程序	SecondhandGoodsDetail	展示二手交易平台的商品详情，用户可以查看商品详细信息、评论和购买等	HTML、CSS、JavaScript
人力资源平台需求搜索	人力资源平台需求搜索子程序	JobSearch	展示人力资源平台的需求列表，并支持关键字搜索、职位类型筛选等功能	HTML、CSS、JavaScript
人力资源平台需求详情展示	人力资源平台需求详情展示子程序	JobDetail	展示人力资源平台的需求详情，用户可以查看需求详细信息等	HTML、CSS、JavaScript

4.1.1.2 业务逻辑模块子程序划分

子程序	名称	标识符	功能	所包含的源标准名
用户管理	用户管理子程序	UserManagement	管理博客系统中的用户信息，包括用户注册、用户认证和用户权限管理等	Java、Spring 框架
文章管理	文章管理子程序	ArticleManagement	管理博客系统中的文章信息，包括文章发布、修改和删除等	Java、Spring 框架、Hibernate 框架
二手交易	二手交易平	SecondhandGoods	管理二手交易平台	Java、

平台商品管理	台商品管理子程序	Management	的商品信息，包括商品发布、修改和删除等	Spring 框架、Hibernate 框架
人力资源平台需求管理	人力资源平台需求管理子程序	JobManagement	管理人力资源平台的需求信息，包括需求新增、修改和删除等	Java、Spring 框架、Hibernate 框架

4.1.1.3 数据访问模块子程序划分

子程序	名称	标识符	功能	所包含的源标准名
用户数据访问	用户数据访问子程序	UserDAO	实现用户数据的增删改查等操作	SQL、JDBC、Hibernate 框架
文章数据访问	文章数据访问子程序	ArticleDAO	实现文章数据的增删改查等操作	SQL、JDBC、Hibernate 框架
二手交易平台商品数据访问	二手交易平台商品数据访问子程序	SecondhandGoodsDAO	实现二手交易平台商品数据的增删改查等操作	SQL、JDBC、Hibernate 框架
人力资源平台需求数据访问	人力资源平台需求数据访问子程序	JobDAO	实现人力资源平台职位数据的增删改查等操作	SQL、JDBC、Hibernate 框架

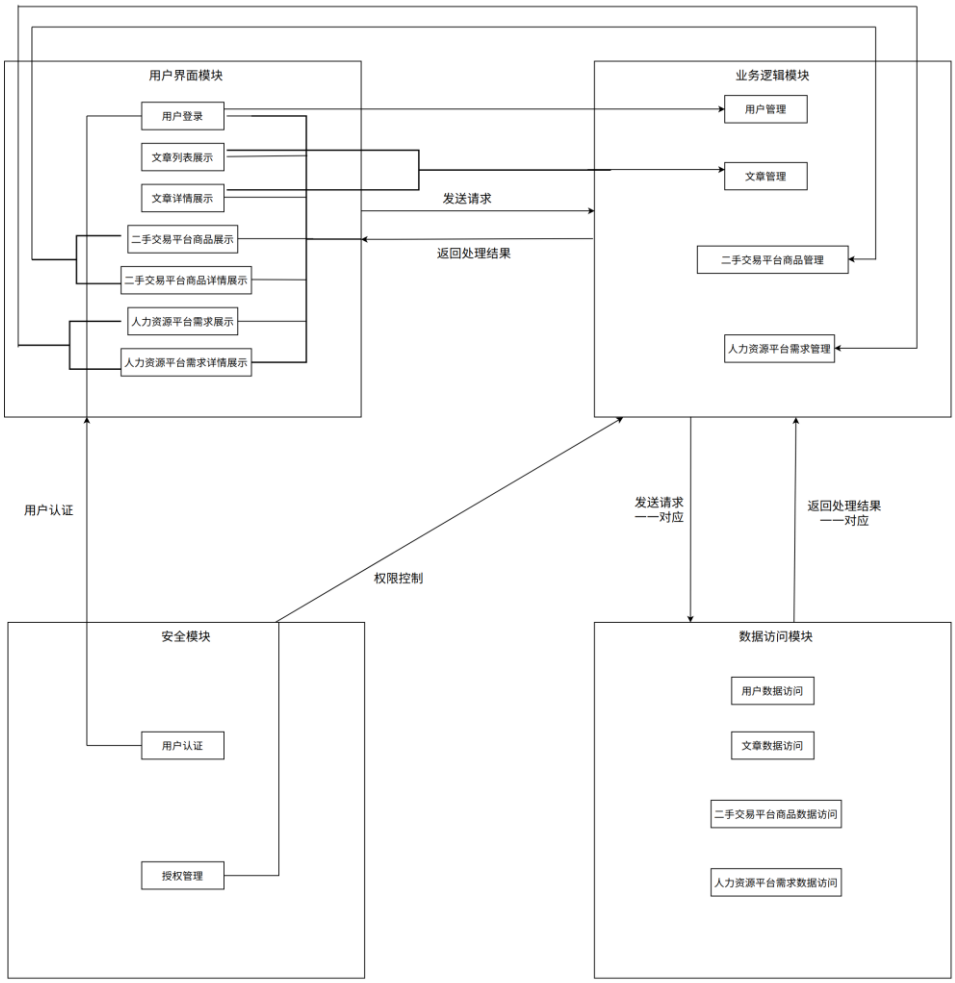
4.1.1.4 安全模块子程序划分

子程序	名称	标识符	功能	所包含的源标准

				名
用户认证	用户认证子程序	UserAuthentication	实现用户认证功能， 确保用户身份合法	OAuth2、Java Servlet API、 Spring Security 框架
授权管理	授权管理子程序	AuthorizationManagement	管理用户对博客系统的 访问权限，确保系统安全	Java Servlet API、 Spring Security 框架

4.1.2 程序(模块)层次结构关系

博客系统内的每个程序(包括每个模块和子程序)之间的层次结构与调用关系如下图:



1. 用户界面模块与业务逻辑模块之间的关系:

用户界面模块向业务逻辑模块发送请求，业务逻辑模块接收请求并处理，返回处理结果给用户界面模块展示。

2. 业务逻辑模块与数据访问模块之间的关系：

业务逻辑模块向数据访问模块发送请求，数据访问模块接收请求并访问数据库，返回查询结果给业务逻辑模块进行

进一步的处理。

3. 安全模块与用户界面模块和业务逻辑模块之间的关系：

安全模块负责用户的认证和权限控制，当用户进行登录或操作时，用户界面模块和业务逻辑模块都需要调用安全模

块进行认证和权限检查。同时，安全模块还负责对敏感数据进行加密和解密。

4. 所有模块之间都需要进行合理的接口设计，以确保模块之间的交互能够顺畅地进行。同时，模块之间还需要进

行适当的错误处理，以避免系统出现异常。

4.2 全局数据结构说明

4.2.1 常量

1. 网站名称：网站的名称，用于在页面标题、Logo 和导航栏等位置展示。
2. 网站描述：网站的简要描述，用于在搜索引擎结果中展示或页面头部显示。
3. 网站主题颜色：网站的主题颜色，用于统一页面的视觉风格和配色方案。
4. 导航菜单：网站的导航菜单项，包括主页、文章列表、分类、标签、关于等链接。
5. 页面尺寸：定义网站页面的默认尺寸，如宽度和高度，用于响应式设计和布局。
6. 默认分页大小：定义列表页面默认的分页大小，用于分页显示文章列表、评论等内容。
7. 日期格式：定义在网站中显示日期的格式，如年-月-日或月/日/年等。
8. 版权声明：网站的版权声明信息，用于显示在网站底部或相关页面。
9. 允许上传的图片类型和大小限制：定义上传图片的类型和大小限制，以确保上传的图片符合要求。
10. 社交媒体链接：定义与博客网站相关的社交媒体账号链接，如 github、微博、微信公众号等。

4.2.2 变量

1. 当前用户：表示当前登录或访问网站的用户，可以在不同页面和功能中使用。
2. 用户权限：表示当前用户的权限级别，用于控制用户在网站中的操作权限。
3. 页面标题：表示当前页面的标题，用于在浏览器标签或页面头部显示。
4. 导航菜单状态：表示当前页面对应的导航菜单项，用于在页面中高亮显示当前活动的菜单项。
5. 当前日期和时间：表示当前的日期和时间，用于在页面中显示或进行时间相关的操作。
6. 错误信息：用于在整个网站中传递和显示错误或异常信息，方便调试和用户提示。
7. 网站配置参数：表示一些全局的配置参数，如数据库连接信息、API 密钥等，供整个网站使用。
8. 全局计数器：用于统计全局的计数信息，如文章浏览量、评论数等。
9. 主题设置：表示当前网站所使用的主题样式，用于控制网站的整体外观和风格。
10. 缓存变量：用于存储和访问一些经常使用的数据，以提高网站性能和减少数据库访问。

4.2.3 数据结构

1. 用户数据：包括用户信息，如用户名、密码、电子邮件、个人资料等。
2. 文章数据：包括文章的标题、内容、作者、发布日期、标签、分类等。
3. 评论数据：包括评论的内容、评论者、评论日期、所属文章等。
4. 标签数据：包括文章所使用的标签，用于分类和组织文章。
5. 分类数据：包括对文章进行分类的信息，用于将文章按照特定的主题进行组织和展示。
6. 用户权限数据：包括用户角色和权限设置，用于控制用户对博客网站的访问和操作权限。
7. 访问日志数据：包括网站的访问日志，记录用户的访问时间、IP 地址、访问页面等信息，用于统计分析和安全审计。
8. 网站配置数据：包括网站的全局配置信息，如网站名称、Logo、主题样式、SEO 设置等。
9. 图片和文件数据：包括上传的图片和文件的存储路径和相关信息，用于展示和下载。
10. 社交媒体数据：包括与博客网站相关的社交媒体账号信息，用于分享和连接到社交媒体平台。

4.3 CSCI 部件

4.3.1 CSCI 1: 用户界面

该 CSCI 负责与用户交互，包括用户信息展示、文章发布、点赞评论提交等。因此，该 CSCI 需要一个响应快速、易于使用和良好的用户体验。基于这些需求，我们决定采用 Web 应用程序作为用户界面。我们将使用 React 框架来实现可重用组件，提高代码的可维护性和可扩展性。此外，我们还将使用 Bootstrap 框架来提供美观的用户界面和响应式设计，以适应各种设备和屏幕尺寸。

1. 标识符：BlogGUICSCI
2. 静态关系：

子程序	静态关系	子程序
ArticleList	使用	ArticleManagement
ArticleDetail	使用	ArticleManagement
UserLogin	使用	UserAuthentication、UserManagement
SecondhandGoodsList	使用	SecondhandGoodsManagement
SecondhandGoodsDetail	使用	SecondhandGoodsManagement
JobSearch	使用	JobManagement
JobDetail	使用	JobManagement

3. 用途

该 CSCI 负责与用户交互，包括用户信息展示、文章发布、点赞评论提交、商品信息展示等

4. 开发状态/类型

HTML、CSS、JavaScript

5. 计划使用的计算机硬件资源

计算机硬件资源	度量级别

CPU	至少 4 核
内存	至少 8GB
硬盘空间	至少 100GB
网络带宽	至少 10Mbps

4.3.2 CSCI 2: 业务逻辑

该 CSCI 负责处理业务逻辑，包括文章管理、推荐计算、评论处理等。基于这些需求，我们将采用微服务架构来实现该 CSCI。每个服务将负责一个特定的业务领域，并使用 REST API 来与其他服务通信。每个服务将使用 Spring Boot 框架和 Java 语言来实现。此外，我们还将使用 Netflix OSS（Open Source Software）套件中的 Eureka 服务发现和 Zuul 网关来实现服务注册和发现、负载均衡和路由功能。

1. 标识符：BlogLogicCSCI
2. 静态关系：

子程序	静态关系	子程序
UserManagement	使用	UserDAO、AuthorizationManagement
ArticleManagement	使用	ArticleDAO、AuthorizationManagement
SecondhandGoodsManagement	使用	SecondhandGoodsDAO、AuthorizationManagement
JobManagement	使用	JobDAO、AuthorizationManagement

3. 用途

该 CSCI 负责处理业务逻辑，包括文章管理、推荐计算、评论处理、交易处理等

4. 开发状态/类型

Python、Spring 框架

5. 计划使用的计算机硬件资源

计算机硬件资源	度量级别

CPU	至少 4 核
内存	至少 8GB
硬盘空间	至少 100GB

4.3.3 CSCI 3: 数据访问

该 CSCI 负责与数据库交互，包括用户和文章信息存储和检索。我们将采用分布式数据库架构，其中每个服务都将有自己的数据存储，并使用事件驱动架构来实现数据同步和一致性。我们将使用 Apache Kafka 作为消息传递中间件来实现事件驱动，并使用 Spring Data 框架来简化数据访问操作。此外，我们还将使用 Spring Cloud Config 来管理配置文件和参数，并使用 Spring Cloud Stream 来实现流式数据处理。

1. 标识符：BlogDAOCSCI
2. 静态关系：

子程序	静态关系	子程序
UserDAO	提供数据	UserManagement
ArticleDAO	提供数据	ArticleManagement
SecondhandGoodsDAO	提供数据	SecondhandGoodsManagement
JobDAO	提供数据	JobManagement

3. 用途

该 CSCI 负责与数据库交互，包括用户、文章及商品信息存储和检索

4. 开发状态/类型

SQL、JDBC、Hibernate 框架

5. 计划使用的计算机硬件资源

计算机硬件资源	度量级别
CPU	至少 2 核

内存	至少 4GB
硬盘空间	至少 500GB
网络带宽	至少 10Mbps

4.3.4 CSCI 4: 安全保障

1. 标识符：BlogSecurityCSCI
2. 静态关系：

子程序	静态关系	CSCI
UserAuthentication	提供数据	BlogGUICSCI
AuthorizationManagement	提供数据	BlogLogicCSCI

3. 用途

保证用户身份的安全性和隐私性，限制用户对敏感数据的访问和修改

4. 开发状态/类型

OAuth2、SSL、Spring Security 框架

5. 计划使用的计算机硬件资源

计算机硬件资源	度量级别
CPU	至少 2 核
内存	至少 4GB
硬盘空间	至少 50GB
网络带宽	至少 10Mbps

4.4 执行概念

1. 执行控制流

在本项目中，执行控制流主要用于描述用户在平台上的操作流程，例如用户注册、发

布博客、发布需求、购买商品等操作。在用户完成一个操作后，系统会根据操作的结果和用户的权限等因素，决定下一步应该执行的操作。执行控制流可以用流程图或控制流图来表示。

2. 数据流

在本项目中，数据流主要用于描述用户信息、博客内容、需求描述、商品信息等数据的流动过程。例如用户注册时，用户输入的信息会被传递到后端服务器进行验证和存储。在数据流图中，可以清晰地描述数据的来源、传输路径和目的地。

3. 动态控制序列

在本项目中，动态控制序列主要用于描述用户在平台上的交互过程，例如用户进入博客详情页后，可以选择发布评论、点赞或分享博客等操作。动态控制序列可以用状态图或活动图来表示，可以清晰地描述用户操作的各个阶段和状态转换的过程。

4. 状态转换图

在本项目中，状态转换图主要用于描述平台中各个状态之间的转换过程。例如，在用户购买商品时，商品会从“待售”状态转换为“已售”状态。状态转换图可以用图形化的方式来表示状态之间的转换关系，便于理解和维护。

5. 时序图

在本项目中，时序图主要用于描述平台中各个操作之间的时序关系。例如，在用户发布博客时，必须先进行身份验证，然后才能进入博客编辑页面。时序图可以用图形化的方式来表示各个操作之间的时序关系，以便于理解和优化。

6. 配置项之间的优先关系

在本项目中，配置项之间可能存在优先关系，例如用户必须先登录才能进行其他操作。在设计系统时，需要考虑这些优先关系，以保证系统的正确性和可靠性。

7. 中断处理

在本项目中，可能会发生各种中断事件，例如用户网络中断、服务器故障等。在设计系统时，需要考虑这些中断事件，并设计相应的中断处理机制，以保证系统的稳定性和可靠性。

8. 时间/序列关系

在本项目中，各个操作之间的时间/序列关系可能会影响系统的性能和用户体验。例如，在用户购买商品时，需要考虑订单生成、支付、发货等过程的时间关系。在设计系统时，需要考虑这些时间/序列关系，并进行优化，以提高系统的性能和用户体验。

9. 异常处理

在本项目中，可能会发生各种异常事件，例如用户输入错误、支付失败等。在设计系统时，需要考虑这些异常事件，并设计相应的异常处理机制，以保证系统的鲁棒性和

可靠性。

10. 并发执行

在本项目中，可能会存在多个用户同时进行操作的情况，例如多个用户同时购买同一件商品。在设计系统时，需要考虑这些并发执行的情况，并设计相应的并发控制机制，以保证系统的正确性和可靠性。

11. 动态分配与去分配

在本项目中，可能会存在动态分配和去分配资源的情况，例如用户发布博客时需要动态生成博客页面。在设计系统时，需要考虑这些动态分配和去分配的情况，并设计相应的资源管理机制，以保证系统的可扩展性和性能。

12. 对象/进程/任务的动态创建与删除

在本项目中，可能会存在动态创建和删除对象、进程或任务的情况，例如用户登录时需要动态创建用户会话。在设计系统时，需要考虑这些动态创建和删除的情况，并设计相应的资源管理机制，以保证系统的可扩展性和性能。

4.5 接口设计

1. 表示层与业务逻辑层的接口：表示层向业务逻辑层发送请求，业务逻辑层返回相应的数据或错误信息。接口特性包括数据格式、数据传输方式和请求响应时间等。

2. 业务逻辑层与数据访问层的接口：业务逻辑层向数据访问层发送数据库操作请求，数据访问层返回相应的数据或错误信息。接口特性包括数据格式、数据传输方式、数据访问协议和数据库连接池大小等。

3. 数据库与数据访问层的接口：数据库提供数据存储和检索服务，数据访问层向数据库发送数据查询和更新请求，数据库返回相应的数据或错误信息。接口特性包括数据格式、数据传输方式、数据访问协议、数据库类型和版本等。

4. 表示层与用户的接口：表示层提供用户界面，用户通过界面进行数据输入和操作。接口特性包括用户界面设计、用户输入验证和错误提示等。

5. 外部实体与数据访问层的接口：外部实体可以通过数据访问层访问系统中的数据。接口特性包括数据访问权限、数据查询和更新协议等。

6. 外部实体与表示层的接口：外部实体可以通过表示层进行数据查询和操作。接口特性包括数据格式、数据传输方式、访问权限和错误提示等。

4.5.1 接口标识与接口图

1.博客平台接口

1. 博客创建接口

接口名称：BlogCreation

功能说明：创建博客，包括博客标题、内容、作者、标签等信息。

接口方法：

•createBlog(title, content, author, tags)

参数：

•title: string，博客标题

•content: string，博客内容

•author: string，博客作者

•tags: list，博客标签列表

返回值：

•成功：blog_id，表示创建的博客的唯一标识符

•失败：返回错误信息，如博客标题不能为空等

2. 博客查询接口（BlogQuery）

queryBlog(keyword, tag, startDate, endDate)

功能说明：允许用户按照关键词、标签、时间等条件查询博客。

接口方法：

•queryBlog(keyword, tag, startDate, endDate): 查询博客，接收关键词、标签、开始时间和结束时间作为参数，并返回符合条件的博客列表。

参数：

•keyword: 查询关键词

•tag: 查询标签

•startDate: 查询开始时间

•endDate: 查询结束时间

返回值：

•成功：符合条件的博客列表

•失败：空列表

3. 博客修改接口（BlogModification）

modifyBlog(blogId, title, content, tags)

功能说明：根据博客 ID 修改博客标题、内容和标签。

接口方法：

• modifyBlog(blogId, title, content, tags): 根据博客 ID 修改博客标题、内容和标签。

参数：

• blogId: 博客 ID

• title: 博客标题

• content: 博客内容
• tags: 博客标签
返回值：
• 成功：true（博客修改成功）
• 失败：false（博客修改失败）

4. 博客删除接口（BlogDeletion）

deleteBlog(blog_id, user_id, password)
功能说明：删除指定的博客，需要验证用户身份并确保用户具有删除权限。
接口方法：
• deleteBlog(blog_id, user_id, password): 删除指定博客，接收博客 ID、用户 ID 和密码作为参数，并返回删除结果。
参数：
• blog_id: 博客 ID
• user_id: 用户 ID
• password: 密码
返回值：
• 成功：true（博客删除成功）
• 失败：false（博客删除失败）

5. 评论创建接口（CommentCreation）

deleteBlog(blog_id, user_id, password)
功能说明：删除指定的博客，需要验证用户身份并确保用户具有删除权限。
接口方法：
• deleteBlog(blog_id, user_id, password): 删除指定博客，接收博客 ID、用户 ID 和密码作为参数，并返回删除结果。
参数：
• blog_id: 博客 ID
• user_id: 用户 ID
• password: 密码
返回值：
• 成功：true（博客删除成功）
• 失败：false（博客删除失败）

6. 评论查询接口（CommentQuery）

queryComments(blogId, startDate, endDate)
功能说明：按照指定的博客 ID、起始日期和结束日期查询博客评论。
接口方法：
• queryComments(blogId, startDate, endDate): 查询博客评论，接收博客 ID、起始日期和结束日期作为参数，并返回评论列表。

参数：
• blogId: 博客 ID
• startDate: 起始日期
• endDate: 结束日期
返回值：
• 成功：评论列表
• 失败：空列表

7. 评论修改接口（CommentEditing）

   editComment(commentId, newContent)

   功能说明：修改指定评论的内容

   接口方法：

   • editComment(commentId, newContent): 修改指定评论的内容，接收评论 ID 和新内容作为参数，并返回修改结果。

   参数：

   • commentId: 评论 ID

   • newContent: 新的评论内容

   返回值：

   • 成功：true（评论修改成功）

   • 失败：false（评论修改失败）

2.交易平台接口

1. 用户身份验证接口（UserAuthentication）：

• authenticateUser(username, password)
功能说明：验证用户身份，确保用户具有合法的登录凭据。
接口方法：
• authenticateUser(username, password): 验证用户身份，接收用户名和密码作为参数，并返回验证结果。
参数：
• username: 用户名
• password: 密码
返回值：
• 成功：true（身份验证通过）
• 失败：false（身份验证失败）

2. 商品列表获取接口（GetItemList）：

• getItemList()

功能说明：获取所有可用的商品列表。
接口方法：
• getItemList(): 获取所有可用的商品列表，并返回商品列表。
参数：无
返回值：
• 成功：商品列表
• 失败：空列表

3. 商品详情获取接口（GetItemDetails）：
• getItemDetails(itemId)
功能说明：获取指定商品的详细信息。
接口方法：
• getItemDetails(itemId): 获取指定商品的详细信息，接收商品 ID 作为参数，并返回商品详情。
参数：
• itemId: 商品 ID
返回值：
• 成功：商品详情
• 失败：空对象

4. 商品购买接口（PurchaseItem）：
• purchaseItem(itemId, userId, paymentMethod)
功能说明：允许用户购买商品并完成支付。
接口方法：
• purchaseItem(itemId, userId, paymentMethod): 购买指定商品，接收商品 ID、用户 ID 和支付方式作为参数，并返回购买结果。
参数：
• itemId: 商品 ID
• userId: 用户 ID
• paymentMethod: 支付方式（例如信用卡、支付宝等）
返回值：
• 成功：true（购买成功）
• 失败：false（购买失败）

5. 用户购买历史记录接口（GetPurchaseHistory）：
• getPurchaseHistory(userId)
功能说明：获取指定用户的购买历史记录。
接口方法：
• getPurchaseHistory(userId): 获取指定用户的购买历史记录，接收用户 ID 作为参数，并返回购买历史记录列表。
参数：

• userId: 用户 ID
返回值：
• 成功：购买历史记录列表
• 失败：空列表

6. 商品搜索接口（SearchItems）：
• searchItems(keyword)
功能说明：在商品列表中搜索指定的关键词。
接口方法：
• searchItems(keyword): 在商品列表中搜索指定的关键词，接收关键词作为参数，并返回匹配的商品列表。
参数：
• keyword: 搜索关键词
返回值：
• 成功：匹配的商品列表
• 失败：空列表

5 CSCI 详细设计

5.1 用户注册登录模块

a、配置项设计决策

对于用户注册登陆模块，在用户进行注册时需要用户提供注册表单，包括用户名、密码、电子邮件等必要信息。对用户输入进行验证，验证用户名的唯一性、密码复杂度等。存储用户注册信息，使用数据库或其他持久化方式。在进行登陆时，对用户输入进行验证，例如验证用户名和密码的正确性。使用安全的方式验证用户凭据，如密码哈希和盐值。在用户成功登录后，可以创建会话或颁发访问令牌用于身份验证和授权。此外还应当具有安全性考虑，使用适当的密码哈希算法和盐值对密码进行加密，确保用户密码的安全性。防止暴力破解和恶意登录尝试，例如实施登录失败限制和验证码功能。使用 HTTPS 协议来保护用户登录信息的传输过程。

b、软件配置项设计中的约束、限制或非常规特征

在用户进行注册时，用户提供的用户名不能与其他用户重复，所输入的手机号，电子邮件的格式，若有则必须正确，对于密码需要进行一定的复杂性限制，保证用户信息安全性；在用户进行登录时，输入的账号密码必须正确。

d、过程式命令组

数据库操作：

- 在用户注册时，将用户提供的信息插入数据库中。

- 在用户登录时，查询数据库以验证用户名和密码的正确性。
- 在密码重置功能中，更新用户的密码信息。

表单验证和处理：

- 对用户在注册和登录表单中输入的数据进行验证，例如检查用户名是否唯一、密码复杂度是否符合要求等。
- 处理表单提交的数据，包括对数据的格式化、清理和存储。

密码哈希和加密：

- 在用户注册和密码重置时，使用适当的密码哈希算法对用户密码进行加密，并将加密后的密码存储到数据库中。
- 在用户登录时，对用户输入的密码进行哈希处理，并与数据库中存储的哈希密码进行比对。

发送电子邮件或短信：

- 在密码重置功能中，生成重置链接或验证码，并通过电子邮件或短信发送给用户。
- 调用相应的库或服务来实现电子邮件或短信的发送功能。

访问控制和权限管理：

- 在用户登录后，根据用户的角色和权限信息，设置相应的访问控制规则，以限制用户对特定功能和数据的访问。
- 根据用户的身份和权限，显示或隐藏网站的特定功能和界面元素。

e、输入、输出和其他数据元素以及数据元素集合体的说明

在用户进行注册时需要用户提供注册表单，包括用户名、密码、电子邮件等必要信息，当成功注册时输出成功注册消息，失败则输出注册失败原因并提示。在进行登录时，提供用户名和密码，当成功登陆时加载相应用户信息，登陆失败则输出登录失败原因。

f、软件配置项使用逻辑

1. 软件配置项执行启动时，其内部起作用的条件：
   - 当用户点击登录或注册按钮时，触发该软件配置项的执行。
2. 把控制交给其他软件配置项的条件：
   - 在用户成功登录或注册后，将控制权交给其他相关模块或功能，例如用户个人资料页面、主页等。

3. 对每个输入的响应及响应时间，包括数据转换、重命名和数据传送操作：

    - 对于用户的登录信息和注册信息，包括用户名、密码等，进行数据验证、转换和传送，确保数据的安全性和完整性。

4. 软件配置项运行期间的操作序列和动态控制序列：

a) 序列控制方法：根据用户的操作流程，设计登录和注册的操作序列，例如验证用户名和密码、创建新用户等。

b) 该方法的逻辑与输入条件，如计时偏差、优先级赋值：定义登录和注册逻辑中的条件判断和操作，例如检查用户名和密码是否匹配、验证用户输入的有效性等。

c) 数据在内存中的进出：将用户登录和注册的数据存储在内存中，进行验证、处理和存储，以便后续使用。

d) 离散输入信号的感知，以及在软件配置项内中断操作之间的时序关系：感知用户的输入动作，例如点击登录按钮、输入用户名和密码等，确保正确捕捉用户的操作，并在合适的时机执行相应的逻辑。

5. 异常与错误处理：

    - 处理登录和注册过程中可能出现的异常和错误，例如用户名或密码错误、用户已存在等情况，提供相应的错误提示和处理机制。

5.2 文章管理模块

a、配置项设计决策

对于文章管理模块，其文章内容存在在数据库中，用户对文章有权限控制，可以对文章进行管理，文章具有多种不同状态，如草稿，已发布，已删除等状态；可以对文章进行评论和点赞；文章浏览具有搜索、过滤和推荐等业务功能；对于文章的阅读数据进行统计，记录文章的数据量、点赞数等指标，与文章共同存储在数据库中，以便于分析文章受欢迎程度和用户偏好。

b、软件配置项设计中的约束、限制或非常规特征

1. 访问权限限制：

    - 只有注册用户可以访问和管理文章，游客只能浏览文章内容。

    - 管理员用户拥有最高权限，可以编辑、发布和删除所有文章，普通用户只能编辑和发布自己的文章。

2. 字数限制：

    - 文章的最大字数限制为 10000 字，以保持阅读体验和页面加载性能。

3. 图片或附件上传限制：

- 仅支持常见的图片格式（如 JPEG、PNG）和常见的文档格式（如 PDF、DOCX）。
- 图片大小限制为 2MB 以内，附件大小限制为 10MB 以内。

4. 文章编辑限制：
   - 已发布的文章只能由作者或具有编辑权限的用户进行编辑。
   - 系统会保留每次编辑的历史记录，以便查看和恢复之前的版本。
5. 文章删除限制：
   - 已发布的文章只能由管理员或具有删除权限的用户进行删除。
   - 删除操作需要进行确认或输入额外的验证信息，以避免误操作。
6. 文章格式限制：
   - 支持富文本编辑器，可以插入图片、链接、表格等丰富的内容。
   - 文章标题必填，正文内容可以为空但不能超过 500 字。

d、过程式命令组

1. 数据库查询命令：使用 SQL 语句执行数据库查询，例如 SELECT、INSERT、UPDATE 和 DELETE 语句，用于获取、插入、更新和删除文章数据。
2. 文件操作命令：使用文件系统命令或 API 进行文件的读取、写入和删除操作，用于上传、存储和管理文章的附件或图片文件。
3. 用户认证命令：包括用户登录验证的命令，检查用户名和密码的正确性，并生成用户会话标识或令牌以进行后续操作的授权。
4. 数据校验命令：对用户输入的文章数据进行校验，包括检查标题是否为空、内容长度是否符合要求、图片大小是否超过限制等。
5. 授权与权限命令：对用户进行权限验证和授权操作，判断用户是否具有编辑、发布、删除等文章管理操作的权限。
6. 审核命令：提交文章审核请求的命令，将待审核的文章发送给审核人员进行审核，并记录审核结果。
7. 发布命令：将通过审核的文章设置为已发布状态的命令，使其在网站上可见。
8. 删除命令：删除已发布或待审核的文章的命令，仅限管理员或具有删除权限的用户执行。

e、输入、输出和其他数据元素以及数据元素集合体的说明

1. 输入数据元素：

- 标题：用户输入的文章标题，例如 "如何提高写作技巧"。
- 内容：用户输入的文章正文内容，例如 "在写作过程中要注重结构和逻辑，避免冗长和重复。"。
- 作者：用户输入的作者信息，例如 "John Doe"。
- 标签：用户选择或输入的关键词标签，例如 "写作技巧"。
- 分类：用户选择的文章分类，例如 "学习与发展"。

2. 输出数据元素：
   - 文章 ID：系统自动生成的唯一标识符，用于识别和检索文章。
   - 发布状态：记录文章的发布状态，例如 "已发布"。
   - 发布时间：记录文章的发布日期和时间，例如 "2023-05-15 10:00:00"。
3. 其他数据元素：
   - 特色图像：用户上传的文章封面图像，用于展示在列表或详情页。
   - 附件：用户上传的附件文件，例如附加的 PDF 文档或其他相关资料。
   - 访问权限：记录文章的访问权限设置，例如 "公开"或 "私密"。
   - 评论数：统计文章收到的评论数量。
   - 点赞数：统计用户给文章点赞的数量。

数据元素集合体是文章的完整数据集，包括上述的输入、输出和其他数据元素的子集。

f、软件配置项使用逻辑

1. 软件配置项执行启动时的内部起作用条件：
   - 用户登录：要求用户进行身份验证，并具有相应的文章管理权限。
2. 控制交给其他软件配置项的条件：
   - 创建新文章：当用户选择创建新文章时，控制权交给文章编辑配置项。
   - 编辑现有文章：当用户选择编辑现有文章时，控制权交给文章编辑配置项。
   - 删除文章：当用户选择删除现有文章时，控制权交给删除文章配置项。
   - 发布文章：当用户选择发布文章时，控制权交给发布文章配置项。
3. 对每个输入的响应及响应时间：
   - 文章标题：用户输入或编辑文章标题时，系统实时响应并保存修改。

- 文章内容：用户输入或编辑文章内容时，系统实时响应并保存修改。

4. 软件配置项运行期间的操作序列和动态控制序列：

- 文章编辑配置项：

a) 序列控制方法：提供文章编辑界面和编辑工具。
b) 逻辑与输入条件：根据用户输入的标题和内容，进行数据验证和格式化处理。
c) 数据在内存中的进出：将用户编辑的文章数据存储在内存中进行临时保存。
d) 时序关系：在编辑过程中，用户可以随时保存或取消编辑。

- 删除文章配置项：

a) 序列控制方法：提供删除确认界面和操作提示。
b) 逻辑与输入条件：验证用户选择的要删除的文章，并执行删除操作。
c) 数据在内存中的进出：无需在内存中存储数据，直接对数据库进行删除操作。
d) 异常与错误处理：处理删除操作时可能出现的异常情况，如权限验证失败或文章不存在。

- 发布文章配置项：

a) 序列控制方法：提供发布确认界面和操作提示。
b) 逻辑与输入条件：验证用户选择的要发布的文章，并更新文章状态为已发布。
c) 数据在内存中的进出：无需在内存中存储数据，直接对数据库进行更新操作。
d) 异常与错误处理：处理发布操作时可能出现的异常情况，如权限验证失败或文章不存在。

5. 异常与错误处理：

- 表单验证：在输入文章标题和内容时，进行必要的验证，确保数据的完整性和合法性。

- 错误处理：处理用户操作过程中可能出现的错误情况，如数据库连接失败或保存失败。

5.3 商品管理模块

a、配置项设计决策

对于商品交易平台，用户可以在网站发布相应的商品并给出自己认为合理的定价；在网站上进行的交易可以使用支付宝或微信等第三方平台进行支付；用户支付后产生的订单通过商家自身配送到相应的地址下；平台对于已经产生的订单应该时刻追踪订单物流情况，并及时给用户发出相应的信息。

b、软件配置项设计中的约束、限制或非常规特征

1. 商品发布限制：

- 约束：限制用户发布商品的频率或数量，以防止滥发或恶意发布。
- 限制：限制用户发布商品的类别或特定商品的发布。
- 非常规特征：设置审核机制，对发布的商品进行审核或人工审核。

2. 图片上传限制：
   - 约束：限制用户上传图片的大小、格式或尺寸，以控制系统资源和页面加载速度。
   - 限制：限制用户上传图片的数量，防止滥用或恶意行为。
   - 非常规特征：图片压缩或缩略图生成，以提高页面加载性能。
3. 商品信息要求：
   - 约束：要求用户填写必要的商品信息，如标题、描述、价格等，以确保信息完整性和准确性。
   - 限制：限制特定字段的字符长度或格式，以防止输入错误或非法字符。
   - 非常规特征：支持富文本编辑器，使用户能够格式化和美化商品描述。
4. 商品状态管理：
   - 约束：限制用户对商品状态的操作，如发布、下架、删除等。
   - 限制：限制用户对已售出或交易中的商品进行编辑或删除。
   - 非常规特征：自动下架或标记已过期的商品，以保持商品信息的实时性。
5. 价格和交易限制：
   - 约束：限制用户设置的商品价格范围，如最低价格、最高价格。
   - 限制：限制用户对商品价格的调整幅度或频率，防止价格操纵或恶意行为。
   - 非常规特征：支持议价或竞价功能，使用户能够自由协商价格。
6. 交易方式限制：
   - 约束：限制用户选择的交易方式，如线上支付、线下交易等。
   - 限制：限制特定商品或用户类型的交易方式，以提供更安全的交易环境。
   - 非常规特征：支持在线支付或第三方支付集成，提供更便捷的交易体验。
7. 交易评价和投诉管理：
   - 约束：要求用户在交易完成后对商品和卖家进行评价，提供真实的交易反馈。

- 限制：限制用户对已评价的交易进行修改或删除，保持评价的可信度。
- 非常规特征：提供投诉管理机制，处理用户之间的纠纷或投诉。

d、过程式命令组

1. 创建商品：
   - 用户填写商品信息并提交表单。
   - 后端接收表单数据，进行数据验证和处理。
   - 将商品信息存储到数据库中，生成唯一的商品标识符。
2. 编辑商品：
   - 用户选择要编辑的商品，并进入编辑页面。
   - 用户修改商品信息并提交表单。
   - 后端接收表单数据，更新数据库中对应的商品信息。
3. 删除商品：
   - 用户选择要删除的商品，并确认删除操作。
   - 后端根据商品标识符删除数据库中对应的商品信息。
4. 查询商品：
   - 用户输入关键字或选择特定条件进行商品查询。
   - 后端接收查询请求，根据条件在数据库中进行联机查询。
   - 返回符合条件的商品列表供用户浏览。
5. 商品展示：
   - 用户点击某个商品，进入商品详情页面。
   - 后端根据商品标识符从数据库中获取商品信息。
   - 将商品信息展示在页面上供用户查看。

e、输入、输出和其他数据元素以及数据元素集合体的说明

1. 输入数据元素：
   - 商品标题：用户输入的商品标题，用于描述商品的名称。
   - 商品描述：用户输入的商品描述信息，包括商品的详细描述、特征等。
   - 商品价格：用户输入的商品价格，表示商品的售价。

- 商品图片：用户上传的商品图片，用于展示商品的外观。
- 商品类别：用户选择的商品类别，用于分类和检索商品。

2. 输出数据元素：
   - 商品列表：根据用户的查询条件和筛选，返回的符合条件的商品列表。
   - 商品详情：展示某个具体商品的详细信息，包括标题、描述、价格等。
   - 商品图片：展示商品的图片，以便用户查看商品的外观。
   - 商品评价：用户对商品的评价和反馈信息。
   - 商品状态：表示商品的当前状态，如上架、下架、已售出等，用于控制商品的展示和可用性。
   - 时间戳：记录商品创建、编辑和删除的时间，用于追踪商品的操作历史和排序。
3. 数据元素集合体：
   - 商品信息集合：包括商品标题、描述、价格等多个数据元素组成的集合，用于表示一个完整的商品信息。
   - 商品列表集合：包含多个商品信息集合体的集合，表示一组符合查询条件的商品列表。

f、软件配置项使用逻辑

1. 软件配置项执行启动时，其内部起作用的条件：
   - 用户登录状态：要求用户登录后才能进行商品管理操作。
   - 用户权限：根据用户的角色和权限级别，限制或授权用户执行特定的商品管理操作。
2. 把控制交给其他软件配置项的条件：
   - 商品上传流程：在商品管理模块中，根据用户的操作，将控制权转交给商品上传、图片上传等子模块来完成相关任务。
3. 对每个输入的响应及响应时间，包括数据转换、重命名和数据传送操作：
   - 用户提交商品信息后，进行数据验证和格式转换，确保数据的准确性和一致性。
   - 图片上传时，进行图片格式转换、重命名和存储操作，以适应系统要求的图片规格和存储方式。

- 数据传送时，确保数据的安全传输和完整性，例如使用加密协议和校验机制。

4. 该软件配置项运行期间的操作序列和动态控制序列：

a) 序列控制方法：根据用户的操作流程和需求，设计合适的序列控制方法，如商品发布流程、商品编辑流程等。

b) 方法的逻辑与输入条件：根据用户的输入和操作条件，执行相应的逻辑判断和处理，例如根据商品状态控制上架或下架操作。

c) 数据在内存中的进出：管理商品信息的读取、写入和更新操作，包括数据库操作和缓存处理。

d) 离散输入信号的感知和时序关系：处理用户的交互操作，对于离散的输入信号（如按钮点击、页面跳转等），相应地执行相应的操作和时序关系。

5. 异常与错误处理：

- 数据验证错误：当用户提交的商品信息不符合规定的格式或要求时，给出相应的错误提示，并要求用户进行修正。
- 数据库操作异常：对于数据库操作的异常情况，如连接中断、查询错误等，进行适当的异常处理和错误日志记录。
- 网络通信异常：当网络连接异常或通信错误发生时，采取相应的容错措施，如重试机制或友好的错误提示。

6 需求的可追踪性

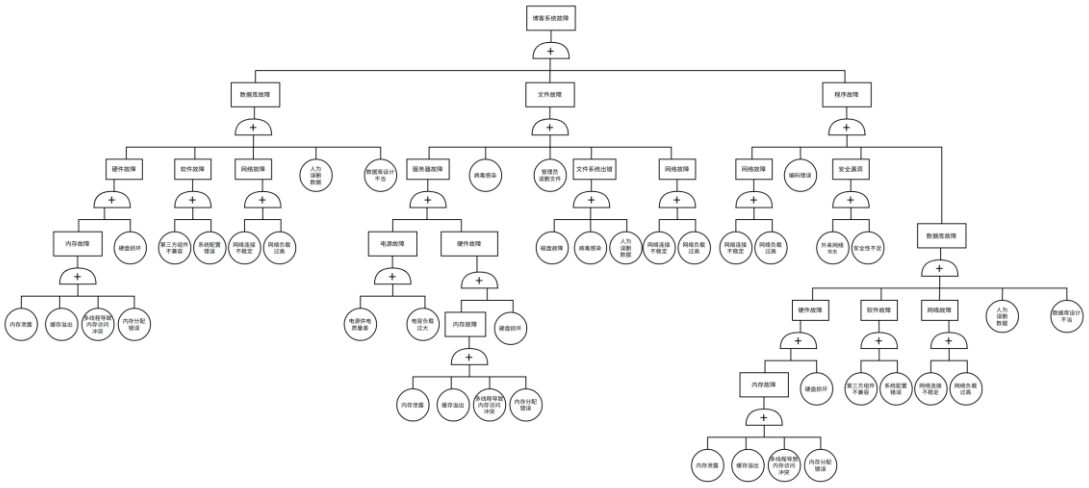
CSCI 需求	标识符	可追踪性
文章列表展示	getArticleList	BlogGUI->AuthorizationManagement-> ArticleDAO->ArticleManagement->ArticleList
文章详情展示	getArticleDetail	BlogGUI->ArticleList- >AuthorizationManagement-> ArticleDAO->ArticleManagement->ArticleDetail
用户登录	getUserLogin	BlogGUI->UserLogin->UserAuthentication ->UserDAO->UserManagement
二手交易平台商品	getSecondhandGoodsList	BlogGUI->AuthorizationManagement- >SecondhandGoodsDAO

展示		->SecondhandGoodsManagement->SecondhandGoodsList
二手交易平台商品详情展示	getSecondhandGoodsDetail	BlogGUI->getSecondhandGoodsList->AuthorizationManagement ->SecondhandGoodsDAO->SecondhandGoodsManagement->SecondhandGoodsDetail
人力资源平台需求搜索	getJobSearch	BlogGUI->AuthorizationManagement->JobDAO ->JobManagement->JobSearch
人力资源平台需求详情展示	getJobDetail	BlogGUI->JobSearch->AuthorizationManagement ->JobDAO->JobManagement->JobDetail

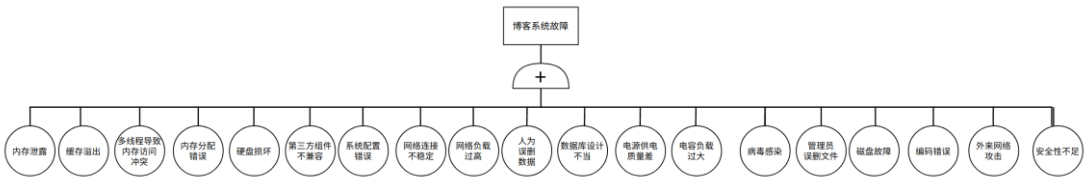
7 注解

8 附录

8.1 博客系统项目故障树



8.1 博客系统项目割集树



9 有关软件体系结构的学习笔记

9.1 基于组件的体系结构（Component-Based Architecture）- 潘云增

基于组件的体系结构（Component-Based Architecture）是一种软件架构设计方法，它将软件系统划分为多个互相

独立的组件，并且这些组件之间具有明确的接口和协议。这种设计方法的目的是提高软件的可重用性、可维护性、

可测试性和可扩展性，以及降低软件开发的成本和周期。

组件是指软件系统中的一个独立的部分，它可以是一个模块、一个模板、一个库或者一个对象。

1、组件具有以下特点：

（1）独立性：组件是一个独立的部分，它可以被单独开发、测试、部署和维护。

（2）可重用性：组件可以被多个系统或者应用程序复用，降低了软件开发的成本和周期。

（3）易于替换：组件之间的关系是松散耦合的，可以很容易地替换一个组件而不影响整个系统。

（4）易于测试：组件具有清晰的接口和协议，可以很容易地测试其功能和性能。

（5）易于扩展：组件可以很容易地扩展和修改，以满足新的需求和功能。

2、组件的分类

组件可以按照不同的角度进行分类，如下：

（1）按照领域分类：如界面组件、数据访问组件、业务逻辑组件等。

（2）按照功能分类：如认证组件、日志组件、缓存组件等。

（3）按照复用粒度分类：如大型组件、中型组件、小型组件等。

（4）按照部署方式分类：如本地组件、远程组件、Web 服务组件等。

3、基于组件的体系结构的设计方法包括以下步骤：

（1）划分系统：将软件系统按照功能和模块划分为多个组件。

（2）定义接口：为每个组件定义清晰的接口和协议，以保证组件之间的互操作性。

（3）实现组件：开发每个组件的实现代码，并且测试其功能和性能。

（4）集成组件：将所有的组件集成到一个整体系统中，并且测试整个系统的功能和性能。

（5）维护和更新：对组件进行维护和更新，以满足新的需求和功能。

4、基于组件的体系结构具有以下优点：

（1）可重用性：组件可以被多个系统或者应用程序复用，降低了软件开发的成本和周期。

（2）易于维护：组件之间的关系是松散耦合的，可以很容易地替换一个组件而不影响整个系统。

（3）易于测试：组件具有清晰的接口和协议，可以很容易地测试其功能和性能。

（4）易于扩展：组件可以很容易地扩展和修改，以满足新的需求和功能。

（5）可靠性：组件之间的关系是松散耦合的，可以很容易地发现和解决问题。

5、基于组件的体系结构的缺点包括：

（1）接口设计：为每个组件定义清晰的接口和协议需要大量的设计和分析工作。

（2）集成测试：将所有的组件集成到一个整体系统中需要进行大量的测试工作。

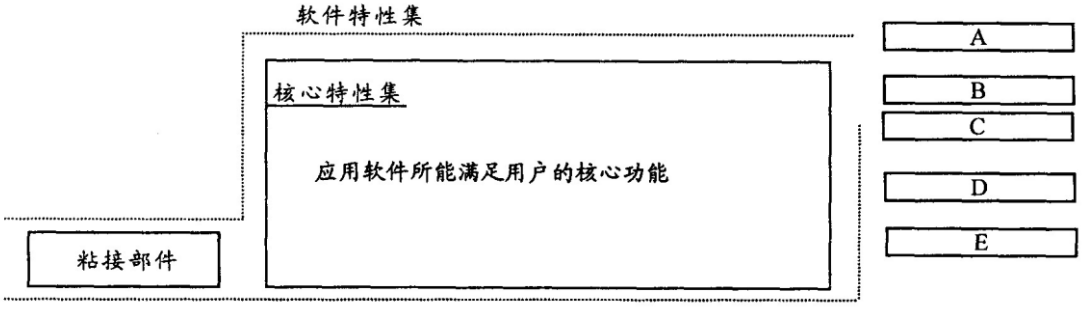
（3）性能问题：如果组件之间的通信过于频繁，可能会影响系统的性能。

（4）组件的选择：选择组件需要考虑组件的质量、可靠性、可重用性和性能等因素。

6、

组件内部是强内聚,组件之间是松耦合,通过粘合部件可以将用户所需的组件找到(本机或者网络上)并临时装配成一

个全新的应用软件从而来满足特殊用户的特殊需求,下图就是一个基于组件技术的软件体系结构示意图：

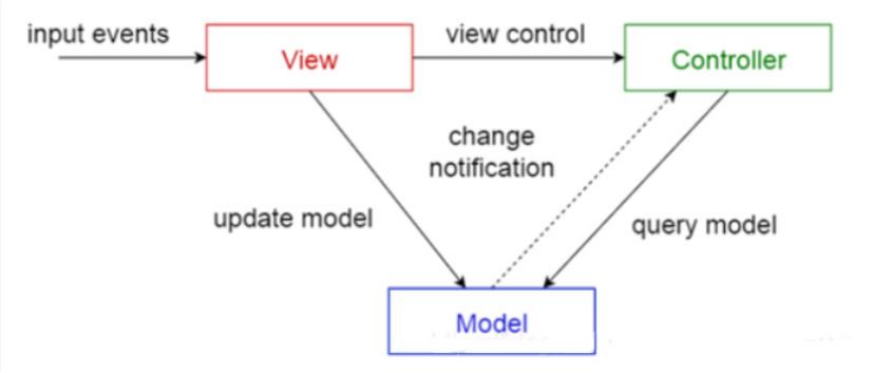


9.2 MVC 体系结构（Model-View-Controller Architecture）- 陆荣周

MVC 体系结构（Model-View-Controller Architecture）：将软件系统划分为模型（Model）、视图（View）和控制器（Controller）三个部分，模型负责处理数据逻辑，视图负责用户界面展示，控制器负责协调模型和视图之间的交互。

1. **模型（Model）**：负责存储系统的中心数据。
2. **视图（View）**：将信息显示给用户（可以定义多个视图）。
3. **控制器（Controller）**：处理用户输入的信息。负责从视图读取数据，控制用户输入，并向模型发送数据，是应用程序中处理用户交互的部分。负责管理与用户交互交互控制。

视图和控制器共同构成了用户接口。而且每个视图都有一个相关的控制器组件。控制器接受输入，通常作为将鼠标移动、鼠标按钮的活动或键盘输入编码的时间。时间被翻译成模型或试图的服务器请求。用户仅仅通过控制器与系统交互。



实现一种动态的程序设计，是后序对程序的修改和扩展简化，并且使程序某一部分的重复利用称为可能。通过对复杂度的简化，使程序结构更加直观。将信息的内部表示与信息的呈现方式分离开来，并接受用户的请求。它分离了组件，并允许有效的代码重用。即，将模型和视图的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据你可以分别用柱状图、饼图来表示。C 存在的目的则是确保模型和视图的同步，一旦模型改变，视图应该同步更新

同时 MVC 体系结构实现了两种分离：

1、视图和数据模型的分离：使用不同的视图对相同的数据进行展示；分离可视和不可视的组件，能够对模型进行独立测试。因为分离了可视组件减少了外部依赖利于测试。（数据库也是一种外部组件）

2、视图和表现逻辑(Controller)的分离：Controller 是一个表现逻辑的组件，并非一个业务逻辑组件。MVC 可以作为表现模式也可以作为建构模式，意味这 Controller 也可以是业务逻辑。分离逻辑和具体展示，能够对逻辑进行独立测试。

优点：耦合性低；重用性高；生命周期成本低；部署快；可维护性高；有利软件工程化管理。
缺点：没有明确的定义；不适合小型，中等规模的应用程序；增加系统结构和实现的复用性；视图与控制器间的过于紧密的连接；视图对模型数据的低效率访问；一般高级的界面工具或构造器不支持模式。

9.3 基于微服务的体系结构 (Microservices Architecture) - 王晓明

基于微服务的体系结构是一种软件架构模式，它将一个大型的应用程序拆分成多个较小的、可独立部署的服务。每个服务都拥有自己的数据存储、业务逻辑和用户界面等组件，可以独立地进行开发、测试、部署和扩展。这种体系结构具有高度的灵活性、可伸缩性和可维护性，适用于大型、复杂的应用程序。

以下是基于微服务的体系结构的主要特点：

1. 拆分为多个服务：将一个大型的应用程序拆分为多个较小的、可独立部署的服务。每个服务都有自己的数据存储、业务逻辑和用户界面等组件，可以独立地进行开发、测试、部署和扩展。

2. 松散耦合：各个服务之间采用松散耦合的方式进行通信，例如使用 RESTful API、消息队列等。这样可以使得每个服务都能够独立地进行开发和部署，而不会受到其他服务的影响。

3. 独立部署：每个服务都可以独立地进行部署，这意味着可以更容易地进行版本更新和升级。

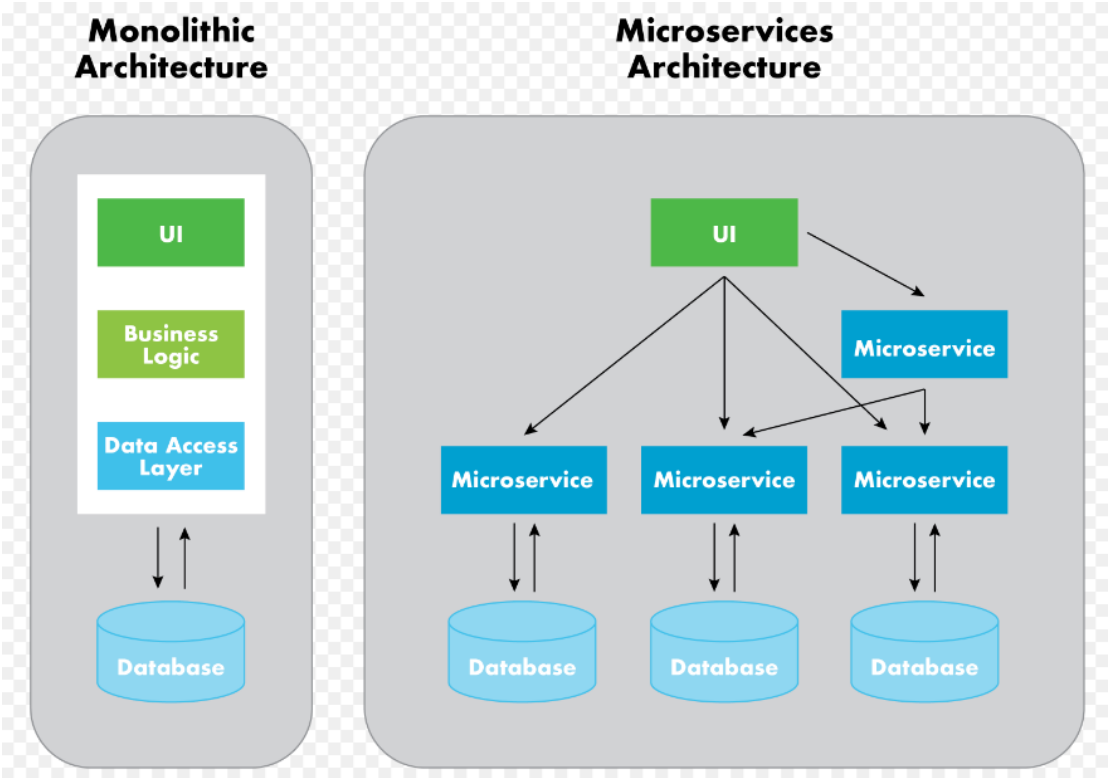
4. 自动化部署：基于微服务的体系结构通常使用自动化部署工具，例如容器化技术（如 Docker）和容器编排工具（如 Kubernetes），以便更快地进行部署和扩展。

5. 多语言支持：各个服务可以使用不同的编程语言和技术栈，这样可以更好地满足开发人员的需求。

6. 可伸缩性：由于每个服务都可以独立地进行部署和扩展，因此可以更容易地实现应用程序的伸缩性。

基于微服务的体系结构具有许多优点，例如更好的可维护性、灵活性和可扩展性，但也存在一些挑战，例如服务间的通信、服务的管理和监控等。因此，在实现基于微服务的体系结构时，需要仔细考虑这些挑战，并选择适当的工具和技术来解决它们。

下图是单服务与微服务的对比图：



9.4 基于分层体系结构（Layered Architecture）- 孙鑫

分层体系结构是一种常用的软件系统设计模式，将系统划分为多个层次结构，每个层次结构有特定的责任和功能。分层体系结构的目标是实现松耦合设计，使各层之间的依赖性最小化，从而使系统更易于维护和扩展。每个层次结构只与相邻的层次结构交互，并通过明确定义的接口进行通信，这些接口通常由标准协议和数据格式定义。分层体系结构的优点包括:

1. 易于维护和扩展：分层体系结构的松耦合设计使得每个层次结构可以独立地开发、测试和维护，从而提高了软件系统的可维护性和可扩展性。

2. 提高可重用性：由于每个层次结构只与相邻的层次结构交互，因此可以更容易地将一个层次结构从一个系统中提取出来并用于另一个系统中。

3. 提高可靠性：分层体系结构可以提供额外的安全层，以确保系统的稳定性和安全性。

4. 提高效率：分层体系结构的每个层次结构都专注于自己的任务，这可以提高整个系统的效率和性能。

比较常见的分层体系结构包括 OSI（开放系统互连）模型、TCP/IP 协议栈、MVC（模型-视图-控制器）模式等。